일반적인 내벽을 가진 자유바닥 곡면 파켓의 NC 가공을 위한 단일화된 황삭과 정삭 알고리즘 - Part 1. Simulation

(A unified rough and finish cut algorithm for NC machining of free form pockets with general polygon - Part 1. Simulation)

최용훈, 조지운, 김상진
(Yong-hoon Choi, Chi-woon Cho, Sang-jin Kim)

요 약 효율적인 최종 NC코드를 만들어내기 위해서 공구경로는 효과적인 방법으로 결정되어 져야한다. 이는 3축 CNC 공작기계상에서 자유형상의 벽을 가진 자유곡면을 가공할 때 특히 중요하다. 많은 CAD/CAM 시스템들은 자유곡면가공을 위한 NC코드 생성시 직선보간을 이용한다. 그러나 이 방법은 가공된 바닥 표면의 매끄러움과 가공시간, 그리고 CL파일 사이즈를 줄이기 위해서 보정되어야 한다. 곡선 가공이 이러한 문제들을 줄이기 위한 해결책이 될 수 있다. 단일화된 황삭과 정삭 알고리즘과 공구경로가 그래픽으로 시뮬레이션 되었다. 본 연구에서는 자유곡면 바닥을 가진 파켓을 3축 CNC 공작기계를 이용하여 일반적인 파켓 밀링을 위한 직선보간과 직선+아크를 혼합한 보간을 위한 NC 공구경로 데이터를 만들기 위한 3D 격자 항행 알고리즘을 개발하였다.

핵심주제어 : NC 코드생성, 직선보간, 직선+아크보간, 3D 격자 항행 알고리즘

Abstract The tool path needs to be determined in an efficient manner to generate the final NC (numerical control) code for efficient machining. This is particularly important in machining free form pockets with an arbitrary wall geometry on a three-axis CNC machine. Many CAD/CAM systems use linear interpolation to generate NC tool paths for curved surfaces. However, this needs to be modified to improve the smoothness of the machined bottom surface, reduce machining time and CL (cutter location) file size. Curved machining can be a solution to reduce these problems. The unified rough and finish cut algorithm and the tool motion is graphically simulated. In this paper, a grid based 3D navigation algorithm for generating NC tool path data for both linear interpolation and a combination of linear and circular interpolation for three-axis CNC milling of general pockets with sculptured bottom surfaces is developed.

Key Words : NC tool path, linear interpolation, linear and circular interpolation, 3D navigation algorithm

1. INTRODUCTION

With increased demands for complicated form and functional surface shapes in many industrial products, the need for an efficient method of CNC machining of compound-curved and intricate surfaces. One of the most important features in determining CNC machining efficiency and productivity is the cutter path motion planning. This cutter motion on compound-curvature surfaces determines the machining time and surface roughness (or cusp height). It is to be

noted that surface roughness always exists because of the lack of geometry matching between the cutter and work-piece surface. The major problem in automatically generating ball end milling cutter path is the difficulty of determining the cutter path of general polygons with compound-curvature surfaces. Generally, a *staircase* or *window-frame* procedure is used.

Most CAD/CAM systems connect the data points with linear segments to generate an NC tool path. This method creates many short linear move commands with sudden changes in direction. Tool paths created in this way can cause problems for the CNC machine during the machining process, and this can affect the finished quality of the part.

Moreover, machining with linear segments leads to rough surfaces caused by interpolation errors and significantly longer machining times due to repeated *stop and go* operations. Curvilinear machining algorithms can reduce these problems. Moreover, with curvilinear machining, the cutter location file size can be reduced for machining the same volume. This is possible since curved data only needs starting and ending points, and the arc center or intermediate control points while linear movements need all data points to specify the tool path.

In view of the short comings mentioned above, a window frame approach with a grid based 3D navigation algorithm for generating NC tool path data for both linear interpolation and a combination of linear and circular interpolation that minimizes the length of the cutter paths for milling general polygons with sculptured surfaces is developed in this paper. Zigzag motions along the X and Y direction are added to make the smoother surface. Deep pockets are handled by a unified rough and finish cut algorithm that uses the z height of the surface grid data. The grid based unified rough and finish cut algorithm is graphically simulated using OpenGL. NC codes for the simulations are generated automatically depending on line or arc + line movement. Figure 1 shows the schematics of the algorithm.

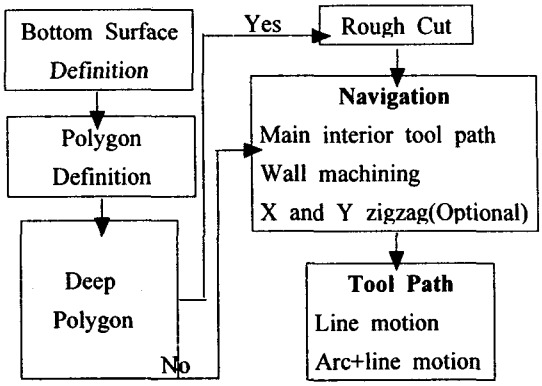


Fig. 1. Schematic representation of the unified rough and finish cut algorithm for machining free form pockets with arbitrary wall machining.

2. RELEVANT LITERATURE

A number of machining strategies and tool-path planning algorithms for free form surfaces have been developed by researchers. These include iso-parametric machining, adaptive iso-curve machining, and pocketing with window frame or staircase style patterns.

By using iso-parametric curves as tool paths, costly surface-surface intersection algorithms can be avoided. However, the spacing between iso-parametric lines, in general, will be non-uniform causing over machining or under machining [6]. Another problem with iso-parametric curves is that tool paths are not optimal for complex surfaces. Some sections could be densely machined while others may be sparsely machined [3]. To counter this problem they developed an adaptive sub-isocurve extraction approach that provided more optimal and valid coverage of the surface by adaptively introducing partial sub-isocurves. Iso-curvature machining was proposed due to the advantages of local placement and orientation of the tool [7]. However, there are difficulties associated with the computation of lines of the curvature.

Pocket machining is further complicated than open freeform surfaces because of the surrounding walls. Contour-parallel machining and directional-parallel machining are the commonly used pocketing strategies. Contour- parallel machining uses offset

segments of the boundary elements of the pocket as tool path segments (depending on the shapes this can be an instance of a window frame approach). Thus the pocket is machined spirally by the tool being driven along the boundary offset that are at constant distances form the pocket's boundary. In directional-parallel machining (staircase style), the tool is moved along line segments, which are parallel to a reference line selected initially. Planning tool motion when the walls of the pocket are non-convex is difficult. Hansen and Arbab [4] developed an algorithm for generating NC tool paths for arbitrarily shaped pockets with islands. Proximity maps [5] has also been recently used for efficient pocket machining with complex wall geometry. Proximity maps are based on the pocket boundary and lead to contour-parallel machining. Suh and Shin [8] developed a neural network model to solve the tool path planning problem as an optimization of the traveling salesman problem. It is based on a grid approach. However, this method is applicable only for rough cuts and for flat bottom surfaces. So it did not need to consider about geometrical mismatch between the tool and workpiece. A tool path navigation strategy [2] with legal moves on a grid has also been suggested to machine non-convex polygonal pockets with flat bottoms. However, to date we note that there is very little research reported for machining free form pockets with arbitrary wall geometry. Because zigzag machining is not required for flat bottoms with flat end mills, but ball and mills are recommended for free surfaces with due to geometrical mismatch.

Beard [1] notes that the linear interpolation approach has become a limiting factor on the contour machining process since it is not accurate, and forces the machinist to work from an approximation of the true work-piece surface. Moreover, the machined surface is not smooth and file sizes are also enormous.

Linear interpolation based curved surface machining can result in thousands of small linear motions and is very inefficient for complex surfaces. Curvilinear interpolation schemes can be used to circumvent these problems. Circular, spline and NURBS based interpolation schemes have recently been proposed by researchers. [9][10]

3. MAIN ALGORITHM

3.1 Surface Definition

The bottom surface in this algorithm can be defined by any analytical or parametric method or even as scattered 3D data. In this paper, the B-spline surface is used as an example to illustrate the algorithm. The B-spline surface formulation uses a control net of 3D points and the B-spline polynomial basis functions for blending. Points on the B-spline surface are specified by the tensor product

$$P(u,w)=\sum_{i=0}^{m}\sum_{j=0}^{n}P_{i,j}N_{i,k}(u)N_{j,l}(w)$$

where, $P_{i,j}$ are the vertices of the control net and $N_{i,k}$and $N_{j,l}$ are the basis functions. B-spline surfaces do not necessary pass though the corner points of the characteristic polygon, and have greater local control.

3.2 Grid Definition

The region to be machined is divided into a grid with size equal to or less than $D/\sqrt{2}$, where, D is the diameter of the tool, as shown in Figure 2. The following assumption can be made.

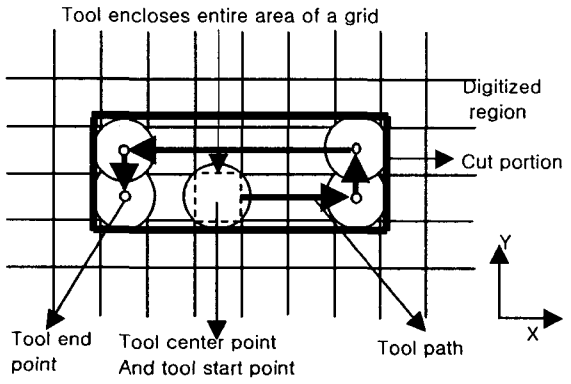


Fig. 2. Digitization and tool path motion.

"Whenever a cutter passes through the center point of each grid, this grid is completely machined because cutter will cover entire area of each grid."

This assumption is true only when the bottom surface of the pocket is flat. When the pocket has an arbitrarily shaped bottom surface, uncut material may still remain due to the geometric mismatch between the tool and workpiece. For 3-axis machining this situation will occur whenever there are any regions that the tool center does not pass through in X or Y directions. To solve this problem, additional X and Y zigzag motions are needed for later execution depending on how smoother surface is required. The selection of grid size on the specified tool size is important since it determines machining time and surface roughness. Larger grid size requires less machining time and larger surface roughness and *visa versa*.

Any general polygon can be defined for pocket milling. General polygon means that it includes both convex and concave shapes.

3.3 Navigation

The navigation strategy plans a sequence of legal moves from one grid point to the next. A set of priorities governs movement selection. The priority depends upon the previous or precedent move. Generally, the direction to go is always the first permissible direction in a counterclockwise direction. Before the details of the proposed methodology are given the following terms are defined.

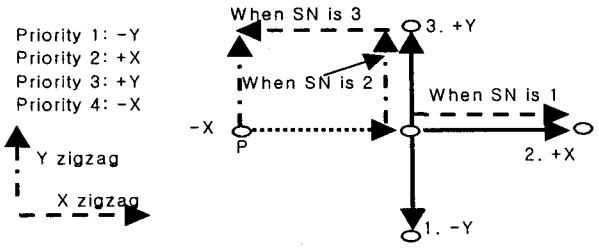
Definitions

Precedent Navigation (PN) : Direction from previous point P to current point (each center point of the figures)

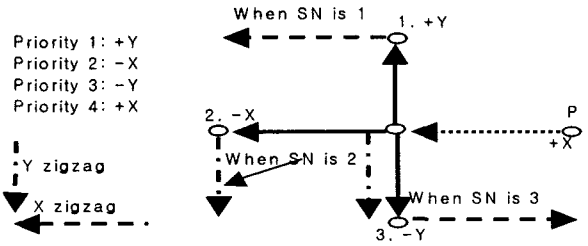
Successive Navigation (SN) : Direction from current point to next point

Priority Number : Direction (E,W,N,S) that has a priority on precedent navigation

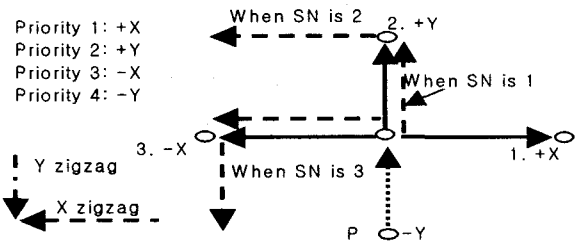
Each end point is tool center.



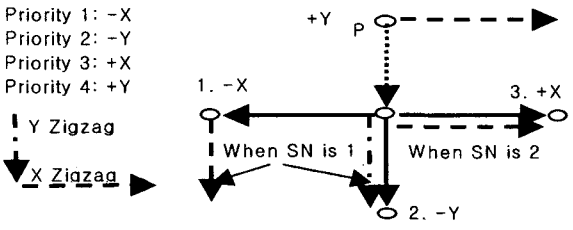
a) When precedent navigation was in +X direction.



b) When precedent navigation was in -X direction



c) When precedent navigation was in +Y direction.



d) When precedent navigation was in -Y direction.

Fig. 3. Navigation priority and associated X and Y zigzag motions.

The entire set of legal moves is summarized in Figure 3. In Figure 3, the movement from the precedent point to the current point is denoted by a dotted line. At each current position (at each center point of figures), the next move can be planned based on the 4 priorities. If there is

no obstruction or this is the first cut, the tool moves to the direction of priority 1. If the direction of priority 1 is has an obstruction then directions of priority 2, 3, 4 in sequential order are checked for possible moves. The X and Y zigzag motions are shown as dashed lines. These motions are made depending on the successive navigation number.

As can be seen from Figure 3, regardless of the precedent navigation direction, the searching process will start with right hand side direction first. If that side is blocked by a pocket wall or already searched previously, then it turns left 90 degree, straight direction from the precedent navigation. By this method the final priority is the direction to the precedent point. That is why the searching process starts from the left most minimum grid in the pocket polygon to prevent redundancy and finishes somewhere at the center part of the pocket polygon.

3.4 Machining by linear interpolation

Linear interpolation proceeds by line movement from one grid point to the next. Linear moves can be made by one or any combination of all the active axes. An example is shown in Figure 4

Fig. 4. Linear movements to generate curve.

As it can be seen in Figure 4, there are consecutive linear motions to generate mountain + valley shape curve. In this case, the advantage is that there is no need of changing active axis plane for machining. Just the next destination coordinate (X, Y, and Z) needs to be specified after one movement. However, the disadvantages are that there are numerous 'stop & go' motions which cause machining delay and large CL file sizes.

3.5 Machining by Curvilinear Interpolation

The navigation grid points can be post-processed for curvilinear machining. In this paper a recursive approach to approximate a curve by a series of arcs and lines is used. This approach is similar to Vickers and Bradley [10] and is illustrated in Figure 5. The segmentation algorithm in case of XZ plane proceeds as follows.

i. Start with a set of cross-sectional points (X_i, Z_i), i =1,...,N.(IF N<3, linear interpolation is formed)

ii. Calculate the data set mid-point.

iii. Calculate the arc center through the first, mid, and last points.

iv. Calculate and test the error between the arc and the intermediate surface points. The error is given as

$$Error = | r - d |$$

$$\left| r - \left[(X_i - X_c)^2 + (Z_i - Z_c)^2 \right]^{1/2} \right|$$

where, d is the distance to the data point (X_i, Z_i) from the arc center (X_c, Z_c) and r is the arc radius.

v. If the test fails (exceeding the specified maximum error tolerance), the span of the arc is decreased from {i =1,...,N} to {i =1,...,N-1}

vi. If the test passes, the radius of the arc is checked to see if it is larger than a threshold value. If so, a linear command is formed from start to end point.

vii. If the radius of the arc is not larger than the threshold value, the appropriate circular interpolation command is generated, otherwise a linear interpolation command is generated between starting and ending points on a given series of points.

viii. The process is stopped if the last point is reached, otherwise a new arc center for the current location and the old end point is calculated.

A circle of radius r and center (h, k) in XZ plane is expressed in implicit form to determine the center and radius of a circular arc passing through three data points. The form in YZ plane is the

same as XZ plane.

$$X^2 + Z^2 + aX + bZ + c = 0$$

where $a = -2h$, $b = -2k$ $c = h^2 + k^2 - r^2$

A system of three equations, in the variables a, b, and c, are created for the data points $\{(X_1, Z_1), (X_2, Z_2), (X_3, Z_3)\}$, all of with lie on the arc. The matrix form of three equations is like follows.

$$\begin{bmatrix} X_1 & Z_1 & 1 \\ X_2 & Z_2 & 1 \\ X_3 & Z_3 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} -(X_1^2 + Z_1^2) \\ -(X_2^2 + Z_2^2) \\ -(X_3^2 + Z_3^2) \end{bmatrix}$$

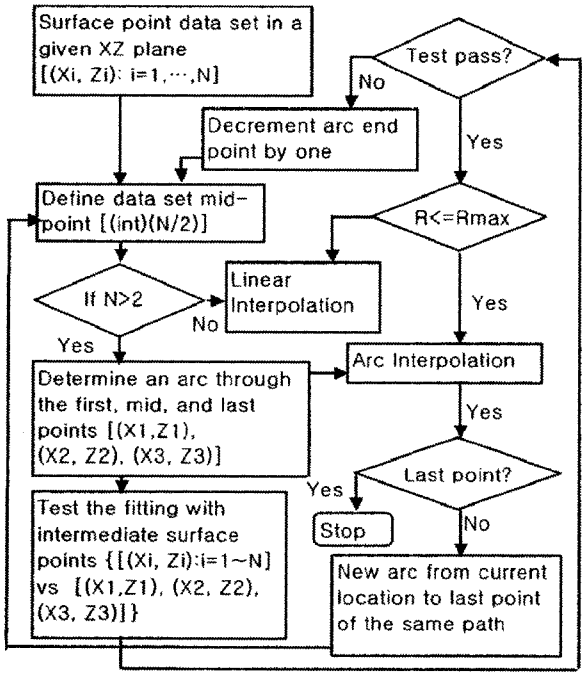


Fig. 5. Arc and line segmentation algorithm.

A curve can thus be represented by a series of arcs and lines as shown in Figure 6.

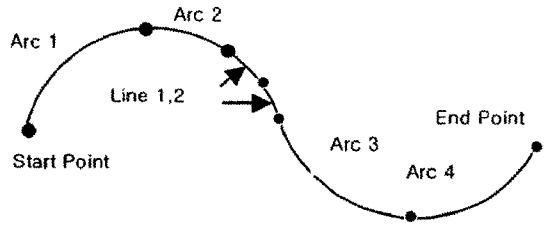


Fig. 6. Illustration of arc + line movements.

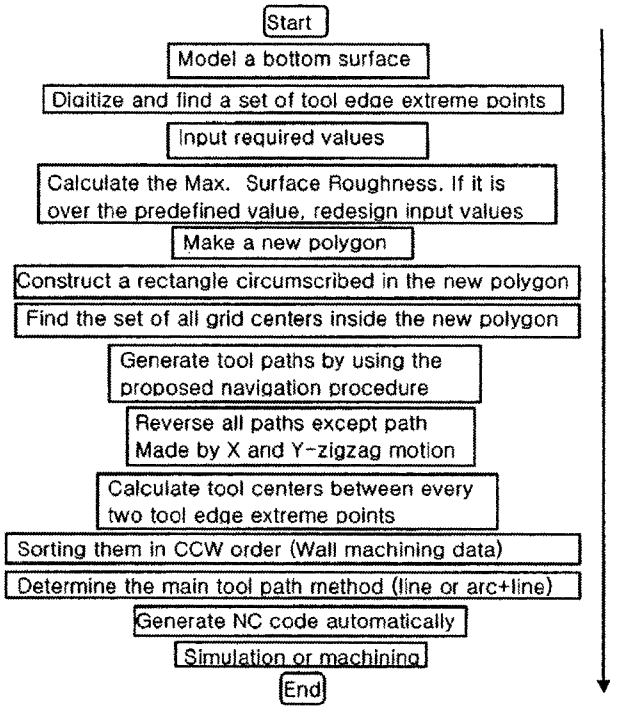


Fig 7. Flowchart for NC machining of freeform pocket.

3.6 Overall Algorithm Procedure

Input values are as follows:

a. Vertex points of a general polygon

b. Tool diameter

c. Grid size in X and Y direction

d. Predefined tolerance of surface roughness.

Interior Navigation for Main Tool Path

1. Model the sculptured bottom B-spline surface with Z direction control points.

2. Digitize the 2D shape in XY as a function of tool diameter and grid size in X and Y directions.

3. Depending on the number of digitized grids in XY plane, determine the unit parameter value (u and v) of X and Y directions from B-spline surface in step 1.

4. Match each X, Y values from step 2 with Z values from step 3 based on an ordered specification. Call this K.

5. Calculate the maximum surface roughness. If it is over the predefined value, redesign input values.

Finally all surface coordinates are determined in relation to tool diameter, X and Y grid size.

6. Find a set of edge cutter extreme points (tool center point) at every vertex (polygon point) defined by user. Call this set C.

7. Make a new polygon based on C.

8. Determine the set of all grid centers inside the new polygon found in step 5. Call this new set S.

9. Find the navigation path from the point that has the left-most minimum Y value by using proposed navigation procedure. Call this set U. And keep the path made by X zigzag and Y zigzag algorithm separately and find Z value by using set K. Call this set N (zigzag motion).

10. Reverse the navigation path (U) and find Z value by using set K. Call this set P (Main tool path).

Finding paths for wall machining and combine with main, X zigzag, and Y zigzag tool paths.

1. Find the nearest neighborhood point within set C to begin wall machining.

2. Calculate tool centers between every two vertices in set C. Call this set F.

3. Sort the counterclockwise order of set C and set F and find Z value by using set K. Call this new set W (Pocket wall path).

4. Final navigation path such that T={P+W+N}.

5. Determine line movements or arc + line movements for machining and simulation path.

6. Generate final NC codes automatically

This procedure can be summarized as shown in Figure 7 and made step by step in each different code. If there is something to be modified, it can be done easily at the specific code. So computational intensity and time are not issue in this algorithm.

4. ROUGH AND FINISH CUT

When deep pockets are to be machined a rough-cut algorithm is needed. The rough-cut algorithm is based on a layer based machining strategy. Instead of machining material in one pass, the material is removed in a series of layers.

Also, in most real machining situations, it is required to separate machining depth into some layers to protect tools from deflection and wear and to overcome the limitation of the tool length.

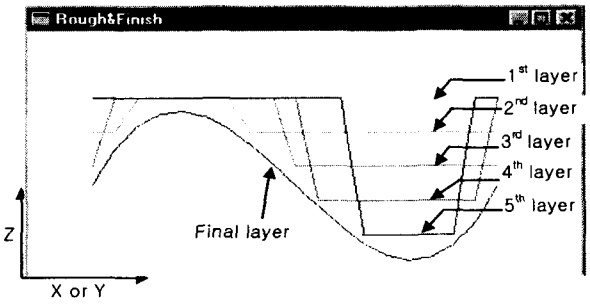


Fig. 8. An example of rough cutting steps.

The proposed method is to control the Z coordinate on bottom surface points, which means that Z coordinates of surface points need to be specified to fit at each layer's specification range. An example is shown in Figure 8.

From the top view, XY grid navigation tool path is the same for each layer (only the depth varies). At each layer, there is a range of Z values, such as $-0.1in < Z < 0$ (top surface), $-0.2in < Z < -0.1in$, and $-0.3in < Z < -0.2in$ etc. If Z values of the bottom surface belong to at each layer's Z value zone at each layer machining time, those Z values would be changed into the highest layer's Z value bigger than the maximum Z value of the bottom surface to prevent unwanted cut which include over-cut.

As the same principal for material removal is being used in this paper, the algorithm is termed as a unified rough and finish cut method. The sequence of machining steps for a deep pocket are 1) Machine layers 1 through N, 2) Finish cut, 3) Wall machining, and 4) X zigzag and Y zigzag machining (optional)

5. CASE STUDY

The proposed procedure is implemented in Visual C++, and OpenGL.

5.1 General Polygon

1. Input

a. Vertex points of a general polygon

b. Tool diameter: 0.25 in

c. Grid size in X direction: 0.125 in

d. Grid size in Y direction: 0.0625in

e. Predefined tolerance: 0.04in

2. Output

a. Output is a set of cutter path to remove inside area of a general polygon with an arbitrarily shaped bottom surface.

b. Drawing of sculptured surface of the polygon.

The top view of the polygon used in the case study and the grid defined is shown in Figure 9. The cutter edge extreme points in this Figure are the points used for wall machining. The grid points interior to the polygon formed by the cutter edge extreme points are used for the main navigation. There is a mountain on Y direction and mountain and valley on X direction on the surface.

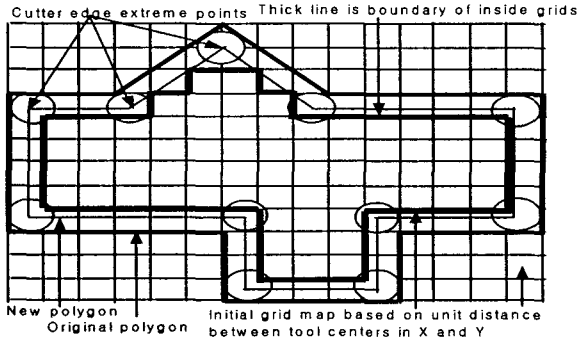


Fig 9. Polygon definition (Grid and cutter edge extreme points are shown)

Line movement

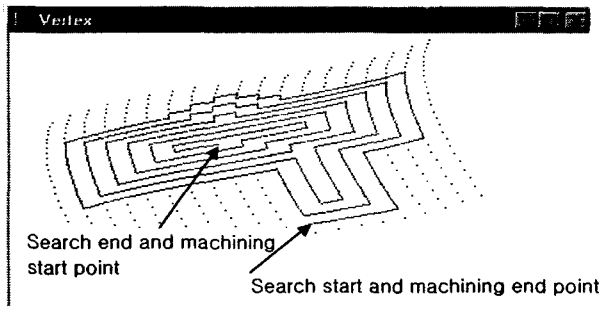


Fig. 10. Isometric view of main tool path (in red)

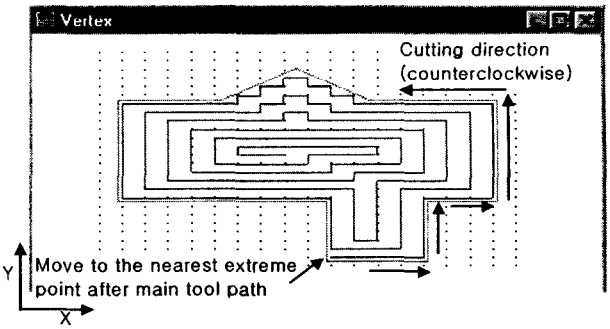


Fig. 11. The pocket wall tool path (in green)

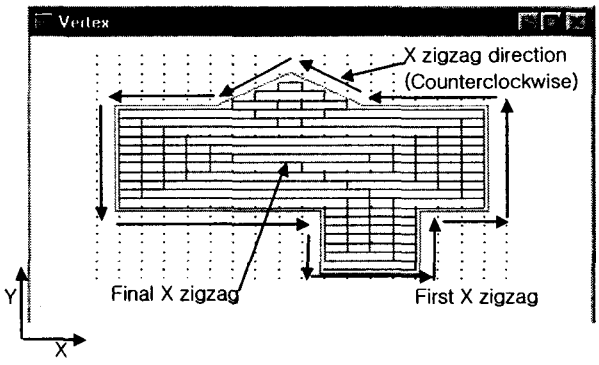


Fig. 12. X zigzag tool path

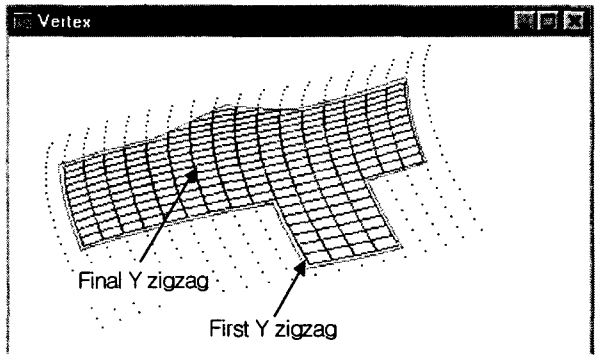


Fig. 13. Isometric view of final result with Y zigzag

Figure 10 shows main tool path and wall machining is displayed in Figure 11. Optional X and Y zigzag tool path to make smoother surface is shown in Figure 12 and 13.

Arc + line movement

Substituting linear main tool path to arc + line tool path had been resulted on the following Figure 14.

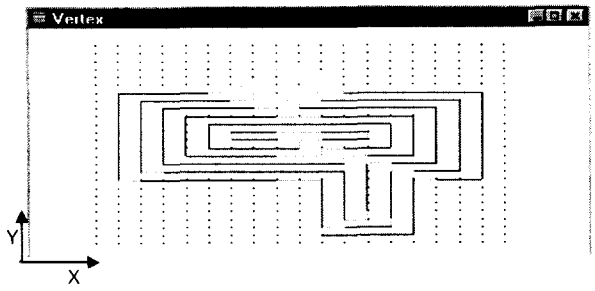


Fig. 14. Arc + line movements.

Yellow(light) lines show linear movement and blue(dark) lines show arc movement. A lot of line segments are saved by arc movement.

5.2 Rough and Finish Cut with Rectangle Pocket

The rough and finish cut is shown in Figure 15. The final layer is the main tool path as explained in the navigation algorithm section. After final layer is machined, it is followed by pocket wall machining, and X and Y zigzag machining. The algorithms for those are the same as the previous case study.

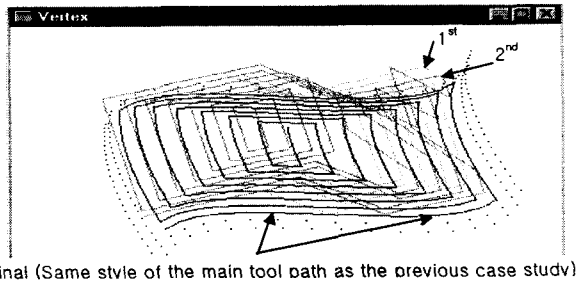


Fig. 15. Rough & finish cut with layers.

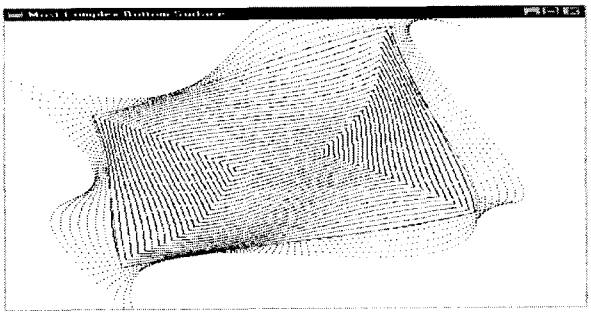
5.3 Comparison of line and arc+line movement

1. Input

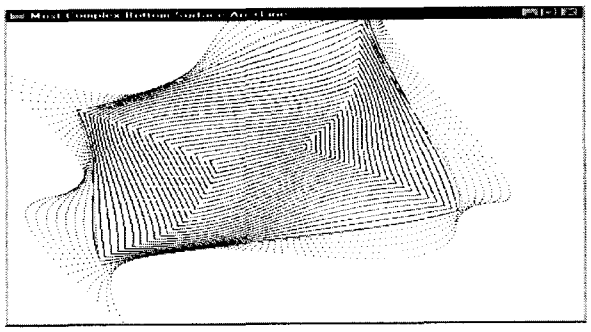
a. Vertex points of a rectangular
b. Tool diameter: 0.25 in
c. Grid size in X and Y direction: 0.05 in each
e. Predefined tolerance: 0.01in

2. Output

Output is a set of cutter path to remove inside area of a rectangular and drawing of free form bottom surface as shown in Figure 16.



(a) All line movements (in red and dark)



(b) Arc (in blue and dark) + line (in yellow and light) movements

Fig. 16. Simulation of complex bottom surface (Max. surface roughness: 0.0089in)

6. CONCLUSION

An algorithm to generate the final NC codes for machining a general polygon with a sculptured bottom surface was presented and simulated with linear interpolation and a combination of circular and linear interpolation. The algorithm improves on the classical window frame approach by using the strategy of navigation based on a set of grid points. Both convex and non-convex polygons can be handled by this algorithm. The bottom surface of the polygon can also be any freeform surface. The algorithm presented here also lays a framework for the implementation of advanced curvilinear interpolation algorithms. The curvilinear algorithms improve the accuracy of machining and reduce the machining time and CL file sizes by reducing the number of

NC commands. With new NURBS based controllers beginning to emerge on the horizon this strategy will be invaluable to fully exploit the new controllers. The algorithm presented here uses a unified approach to machine deep pockets by a series of rough-cuts.

REFERENCE

[1] Beard, T. (1997), "Interpolating Curves", *Modern Machine Shop*, October, pp. 60-67.

[2] Bao, H. P. and Yim, H. (1992), "Tool Path Determination for End Milling of Non-Convex Shaped Polygons", *NAMRI Transactions*, pp. 151-158.

[3] Elber, G. and Cohen, E. (1994), "Toolpath Generation for Freeform Surface Models", *Computer Aided Design*, Vol. 26, No. 6, pp. 490-496.

[4] Hansen, A. and Arbab, F. (1992), "Algorithm for generating NC tool paths for arbitrarily shaped pockets with islands", *ACM Transactions on Graphics*. Vol. 11, pp. 152-82.

[5] Held, M., Lukacs, G., and Andor, L., (1994), "Pocket Machining Based on Contour-Parallel Tool Paths Generated by means of Proximity Map", *Computer Aided Design*, Vol. 26, No 3, pp.189-202.

[6] Lee, Y. S. (1998), "Non-isoparametric tool path planning by machining strip evaluation for 5-axis sculptured surface machining", Computer-Aided Design, Vol. 30, Num. 7, pp. 559-570.

[7] Sarma, R. and Dutta, D., (1997), "The Geometry and Generation of NC Tool Paths", *Journal of Mechanical Design*, Vol. 119, pp. 253-258.

[8] Suh, S. and Shin, Y. (1996), "Neural Network Modeling for Tool Path Planning of the Rough Cut in Complex Pocket Milling", *Journal of Manufacturing Systems*, Vol. 15, No. 5, pp. 295-304.

[9] Vickers, G.W. and Bradley, C., (1992), "Curved Surface Machining through Circular Arc Interpolation", *Computers in Industry*, Vol.19, pp. 329-337.

[10] Zhang, Q. and Greenway, R. (1998), "Development and implementation of a NURBS curve motion interpolator", Robotics and Computer Integrated Manufacturing, Vol. 14, pp. 27-36.

최 용 훈 (Yong-hoon Choi)

울산대 기계공학과학사(97년)
Iowa State University
산업공학 석사(99년)
Iowa State University
산업공학 박사(02년)
UC-Davis기계공학과 Intelligent Manufacturing Systems Mechatronics Lab 연구원 (03년)
현재 현대자동차 현대-기아생산개발총괄본부 생산기술개발팀 과장
(관심분야 : 생산공정설계/모니터링, 정밀가공, CAD/CAM, 신호처리)

조 지 운 (Chi-woon Cho)

울산대 산업공학과 학사(89년)
University of Missouri - Columbia
산업공학과 석사(92년)
Iowa State University
산업공학 박사(01년)
현대중공업 산업기술연구소, 선임연구원(92-98년)
LG CNS 컨설팅 부문, 책임컨설턴트(02년)
현재 삼성 SDS 컨설팅 부문, 수석컨설턴트
(관심분야 : ERP, SCM 등 기업정보화 및 CAD/CAM)

김 상 진 (Sang-jin Kim)

한양대 전기공학과 학사(73년)
한양대 전기공학과 석사(75년)
Kazakh National Academy of Sciences 전기공학과 박사(97년)
현재 시립인천전문대학 제어계측과 교수(81년~)
(관심분야 : 산업제어공학, 센서공학)