Int J Adv Manuf Technol (1998) 14:85–92

The International Journal of
Advanced Manufacturing Technology

Rough Cut Tool Path Planning for B-spline Surfaces Using Convex Hull Boxes

Sheng H. Chuang and C. C. Pan

Department of Mechanical Engineering, National Chung-Hsing University, Taichung, Taiwan

The objective of this paper is to present a non-uniform layered rough cut plan for B-spline surfaces using convex hull boxes. The tool path plan generated by this method will rapidly remove most redundant material from stock material without overcutting.

First, a B-spline surface is decomposed into piecewise Bezier surfaces, of which the convex hull boxes form an approximate model for rough cutting. Then, according to the top planes of those convex hull boxes, the stock material is divided into layers of different thickness. The cavity contour for each layer is obtained using a simplified union Boolean operation on convex hull boxes. Finally, from the top down, each layer is processed like a 2D pocket die cavity.

The algorithm is implemented on a personal computer. It is shown that the rough cut plan is very efficient since no computation for solving nonlinear equations is needed, and no overcutting occurs since B-spline surfaces are protected by the convex hull property of Bezier surfaces.

Keywords: Approximate model; Boolean union; B-spline surface; Convex hull box; Pocket machining; Rough cut planning

1. Introduction

A plan for surface rough cutting is expected to remove most unwanted material from stock without overcutting. Flat-ended milling cutters are the most commonly used tools for rough cutting. Traditionally, the rough cut for a B-spline surface uses equally spaced slicing planes, and the surface is cut into layers of equal height [1]. Each layer is machined using the 2D pocketing method. The intersecting contour of a slicing plane and the surface is obtained by solving high-order nonlinear equations using numerical methods. The numerical instability and time-consuming problems involved remain to be solved satisfactorily.

Correspondence and offprint requests to: Professor S. H. Chuang, Department of Mechanical Engineering, Chung-Hsing University, 250 Kuo-kuang Road, Taichung, Taiwan.
Fax: 886 4 287 7170

Others use a simpler model to approximate the surface and use the approximation model to simplify rough cut planning [2–4]. The Z-map method and the octree method are used for the construction of approximation models. The Z-map method computes the z-coordinates for vertical rays from grid points of an X-Y net [2,3]. The computation for obtaining the z-mapped values is complex. The octree method requires containment tests between boxes and the solid formed by the surface; such computation is not easy [4].

In this paper, an efficient method for automatic rough cut path planning of a B-spline surface, with a geometric certainty of no overcutting, is proposed. At the beginning, a B-spline surface is transformed into piecewise Bezier surfaces using the knot insertion method [5,6]. Each Bezier surface can be contained in a box of minimal size formed by its control points, called a convex hull box, and all boxes containing those Bezier surfaces make up the rough cut approximation model for the B-spline surface. After the approximation model has been constructed, the minimum box containing the approximation model is used as the stock material for the surface cut.

Subsequently, the stock is sliced into layers of different heights according to the top plane of the containing convex hull box of each Bezier surface. When a layer height is too large, the layer is recursively subdivided to match the required accuracy, and when a layer height is too small, it is merged with its neighbours to improve the cutting efficiency. After completing the layer-dividing procedures, every layer is considered as a 2D pocket, and the tool paths can be generated using a pocketing tool path plan. The sum of tool paths for all layers is the total number of paths for the rough cut of a B-spline surface.

2. Transforming a B-spline Surface into Bezier Surfaces

In order to build the approximation model for a B-spline surface using the convex hull property of Bezier surfaces, a B-spline surface is transformed into piecewise Bezier surfaces. The knot insertion method is applied to repeat the knots of a B-spline surface and the corresponding piecewise Bezier control points will be generated without changing the shape of the

original surface [5–7]. If $S(u, v)$ represents the position vector of an arbitrarily selected point on a B-spline surface, for a bicubic example with degree 3 by 3, a B-spline surface is defined as follows [8–10]:

$$S(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} N_{i,3}(u) N_{j,3}(v) P_{i,j}$$

where, $P_{i,j}$ is a control point on the B-spline control net; the number of control points is $(m + 1) \times (n + 1)$; $N_{i,3}(u)$ and $N_{j,3}(v)$ are respectively the B-spline basis functions defined in U and V knot vector directions. The knot vectors may be open, uniform and non-periodic which are frequently used in free-form surface design. Such knot vectors are defined as follows [8]:

$$U = \{0,0,0,0,u_4, \ldots, u_m,1,1,1,1\}$$
$$V = \{0,0,0,0,v_4, \ldots, v_n,1,1,1,1\}$$

If there are no internal knots, a B-spline basis function is simplified as a Bernstein function and the corresponding B-spline surface is equivalent to a Bezier surface. For an arbitrary point on the Bezier surface, $Q(u, v)$ is defined as follows [10,11]:

$$Q(u,v) = \sum_{i=0}^{3} \sum_{j=0}^{3} B_{i,3}(u) B_{j,3}(v) P_{i,j}$$

where, $P_{i,j}$ is a control point on the Bezier control net; the number of control points is 4 by 4. $B_{i,3}(u)$ and $B_{j,3}(v)$ are, respectively, Bernstein functions defined in U and V knot vector directions.

For a bicubic B-spline surface, executing knot insertion three times for each internal knot of knot vector U, then, three times for each internal knot of knot vector V, the control points for the B-spline surface become the control points for piecewise Bezier surfaces [6,12]. For a bicubic B-spline surface, as in Fig. 1, after the transformation through knot insertion procedures, it becomes four piecewise bicubic Bezier surfaces.

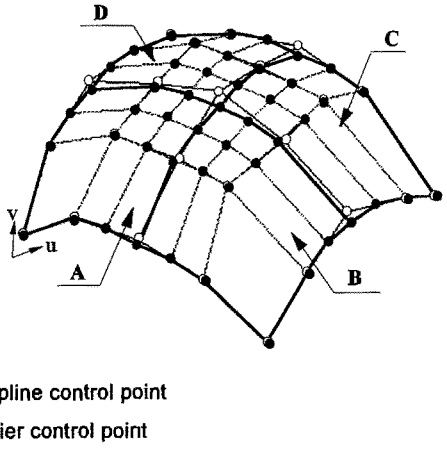

Fig. 1. B-spline surface to Bezier surface.

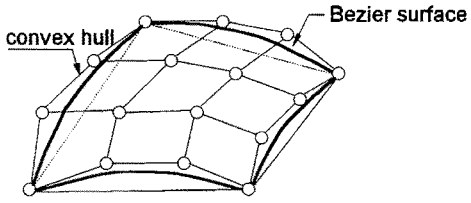

Fig. 2. Convex hull of Bezier surface.

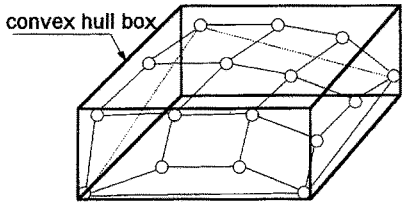

Fig. 3. Convex hull box.

3. Constructing an Approximation Model

In planning a surface rough cut, a B-spline surface is replaced with a surface approximation model, which is composed of a number of boxes. According to this approximation model, the stock is sliced into layers and the machining boundary is obtained for each layer. There are two reasons for constructing this approximation model:

1. To derive layered pocketing boundaries by a simple union operation.
2. To avoid overcutting according to this approximation model.

In this research, the approximation model is constructed by applying the convex hull property of Bezier surfaces. The convex hull property of Bezier surfaces is that a Bezier surface is contained in the minimal convex polyhedron wrapping up its control points. As shown in Fig. 2, a Bezier surface is wrapped in the convex polyhedron formed by its 16 control points. Since operations on polyhedrons, such as slicing and Boolean operations, are complex, a minimal box to contain the convex hull is used to simplify the wrapping space. This greatly simplifies the operations in rough cut planning (see Fig. 3).

After transforming a B-spline surface into many piecewise Bezier surfaces, a minimal convex hull box can be found for every Bezier surface, as in Fig. 4. The minimum box enclosing all convex hull boxes can be used as the stock material for

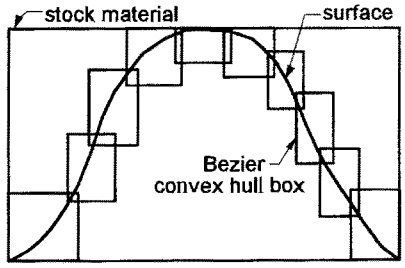

Fig. 4. Convex hull boxes and stock material.

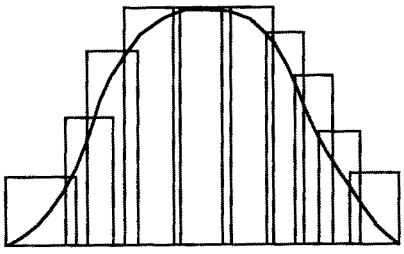

Fig. 5. Surface approximation model.

producing the B-spline surface. When the stock box has been found, every convex hull box is extended to the bottom of the stock, as shown in Fig. 5. The union of all extended convex hull boxes is the approximation model of the B-spline surface. The approximation model will wrap the B-spline surface completely. If the rough cut plan can avoid cutting the interior of the approximation model, the approximation model can protect the B-spline surface from overcutting.

Although B-spline surfaces themselves have the local convex hull property, the local convex hulls overlap considerably so that the approximation model constructed according to this characteristic cannot approximate a B-spline surface well. Since the overlapping between convex hull boxes of Bezier surfaces transformed from a B-spline surface are comparatively smaller, the corresponding approximation model is much better than that when directly using a B-spline model. The stock volume will be smaller and the removed volume is larger, which means a better cut plan is found.

4. Slicing in Non-uniform Spacing

When the approximation model is established, the top planes of convex hull boxes are used as slicing planes to slice the stock material, so that the pocket boundary for each layer can be constructed. As in Fig. 6, the stock material is sliced into layers with different thicknesses according to the top planes of convex hull boxes. Some of the sliced layers may exceed the maximum depth of cut or fall short of the minimum depth of cut. For the layers exceeding the maximum depth of cut, the cutting tool experiences too much cutting force. This causes excessive tool wear and poor cutting results. Such layers are subdivided to decrease layer thicknesses. When layers are below the minimum depth of cut, the cutting efficiency is lowered. Such layers will be merged with neighbouring layers to increase layer thicknesses and thus to increase rough cut

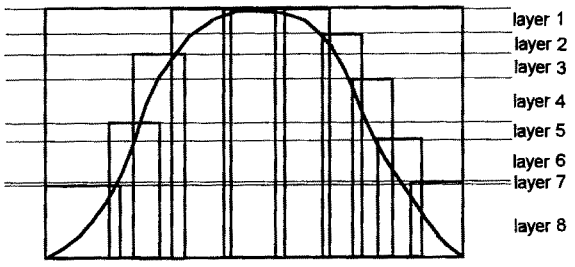

Fig. 6. Non-uniform layer slicing.

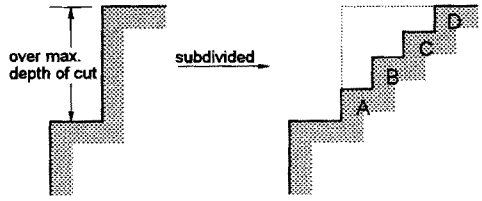

Fig. 7. Subdivision for over depth cut.

efficiency, but the merged layers should not exceed the maximum depth of cut.

For a layer with a thickness greater than the maximum depth of cut, the corresponding Bezier surface is subdivided by the de Casteljau algorithm [9]. The subdivision is completed in the following two steps:

1. Subdivide for the control points $\{V_{ij}\}$ in the v direction (or u direction).
2. Subdivide for the newly generated control points, $\{V_{ij}\}$, in the u direction (or v direction).

A bicubic Bezier surface is thus subdivided into four in one iteration. The convex hull box for an original Bezier surface is replaced with four new convex hull boxes. The layer over the maximum depth of cut is thus subdivided into several layers, of which the depths are decreased. A result of such subdivision is illustrated in Fig. 7.

For a layer with a thickness of less than the minimum depth of cut, the top faces of all convex hull boxes in this layer are elevated to the upper layer and the boxes are merged with the boxes of the upper layer. Before the merging process proceeds, make sure that the thickness of the resulting merged layer does not exceed the maximum depth of cut. If it does, the merge is cancelled. An example of the merging result is shown in Fig. 8. If the merged layer thickness is still less than the minimum depth of cut, the merging procedure is applied again, until the layer thickness is between the maximum and minimum depths of cut.

5. Finding a Pocketing Boundary for Single Layer

After the stock has been sliced into layers, the next step is to compute the pocket boundary for each layer. In general, a pocket boundary is composed of one periphery with several islands. For each layer, the stock boundary is used as the pocket periphery and convex hull boxes are considered as islands. The pocket boundary for a layer is composed of the line segments obtained by using the bottom plane of the layer intersecting the stock boundary and those convex hull boxes.

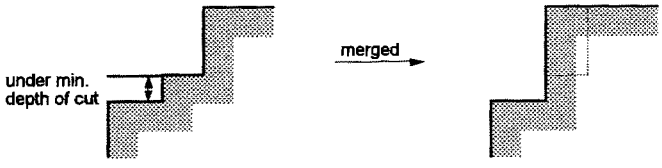

Fig. 8. Merging for under depth of cut.

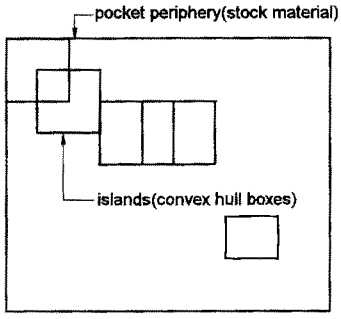

Fig. 9. Pocket contour before restructuring.

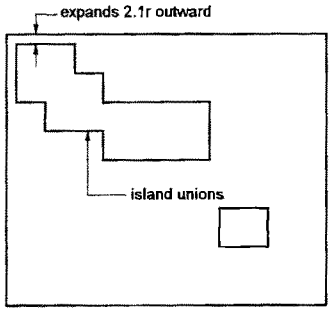

Fig. 10. Pocket contour after restructuring.

The pocket boundary of a layer, as shown in Fig. 9, is found to overlap on some line segments of pocket periphery and islands, and there are intersections and overlappings among line segments of islands. A pocket with such structures cannot be directly used for tool path generation, and the boundary segments must be restructured.

To resolve the overlapping between the pocket periphery and islands, the pocket periphery is expanded outward by a distance, say $2.1r$, where r is the tool radius. A simplified Boolean union for islands is used to deal with the intersections and overlappings between islands. A correct pocket structure is shown in Fig. 10.

Boolean operations, such as union, intersection and difference, are very useful for geometric processing in solid modelling [13]. Fig. 11 shows how the union operation is used to combine two 2D objects A and B. The boundary edges of A and B are split at intersection points. After the splitting, the boundary edges of A and B are classified into 8 categories.

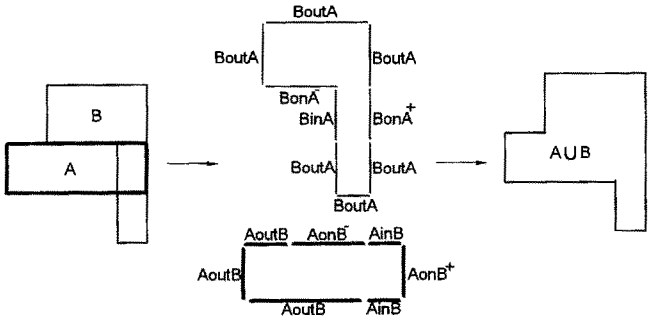

Fig. 11. Union operating on objects A and B.

A out B boundary edges of A which are outside B

B out A boundary edges of B which are outside A

A in B boundary edges of A which are inside B

B in A boundary edges of B which are inside A

A on B^+ boundary edges of A which overlap the boundary of B, and the edges in A and B which have the same direction

B on A^+ boundary edges of B which overlap the boundary of A, and the edges in A and B which have the same direction

A on B^- boundary edges of A which overlap the boundary of B, and the edges in A and B which have opposite directions

B on A^- boundary edges of B which overlap the boundary of A, and the edges in A and B which have opposite directions

The result of the union of objects A and B is composed of 3 categories of boundary edges. The expression is shown as follows.

$$A \cup B = A \text{ out } B + B \text{ out } A + A \text{ on } B^+$$

To eliminate the intersections and overlappings of the rectangular island boundaries obtained from slicing, rectangular islands are considered as 2D solid objects. Using the 2D Boolean operation, such rectangles can be unionised to obtain the correct representation structure of a pocket boundary. In order to apply the union operation, the representation of a pocket must be defined correctly and precisely. In this paper, the periphery of a pocket is composed of a list of edges $\{L_1, L_2, \ldots, L_n\}$. These edges connect with one another to form a closed loop, and the rotating direction is arbitrarily defined as counterclockwise. On the other hand, an island boundary is also composed of a closed edge loop, but the rotating direction is opposite to the periphery, i.e. clockwise. The union operation of islands is as follows:

1. Find the intersection points of edges and split the edges at the intersection points.
2. Compute the containment values for all edges.
3. Restructure the sequences of edges and form new island loops.

Using Fig. 9 as an example, the details of the above steps are further described as follows.

5.1 Find Intersection Points

In the first step of this simplified union operation, find the edges in different loops which intersect properly and find the intersection points. Two edges that intersect at intermediate points, but not at end points, are called properly intersecting. Every island loop will be selected to find intersection points with respect to all the others. In order to reduce the computing time, each pair of box loops is tested to determine whether they intersect or not. If they do, find the intersecting edges in different loops, and find the intersection points.

Box Intersecting Test

Let points P_1 and P_2 represent the lefthand bottom corner and righthand top corner of box A and points P_3 and P_4 represent the lefthand bottom corner and righthand top corner of box B. If the coordinates x and y of points P_1 and P_2 of A are both inside or both outside the coordinates of P_3 and P_4 of B, boxes A and B are non-intersecting. Otherwise, boxes A and B intersect.

Line Intersecting Test

If two boxes intersect, take an edge in one box to test intersections with all edges in the other box. When all edges have been tested, all intersecting lines can be found and the intersection points can be obtained. Since the directions of edges of boxes are either horizontal or vertical, there are three intersecting conditions.

1. One edge is a horizontal edge and the other is vertical. Two edges intersect if and only if the x-coordinate of the vertical edge is between the x-coordinates of the two ends of the horizontal edge, and the y-coordinate of the horizontal edge is between the y-coordinates of the two ends of the vertical edge.
2. Both edges are horizontal. They intersect if and only if both edges have the same y-coordinates, and the x-coordinate of one end or the x coordinates of both ends of an edge are between the x-coordinates of the two ends of the other edge.
3. Both edges are vertical. They intersect if and only if both edges have the same x-coordinates, and the y-coordinate of one end or y-coordinates of both ends of an edge are between the y-coordinates of the two ends of the other edge.

When two edges are found to be intersecting, the coordinates of the intersecting point are the intersecting combinations of the coordinates of the four endpoints. Fig. 12 shows the results after obtaining the intersection points for Fig. 9. After the intersecting points have been found, the corresponding edges are split at the intersection points, as in Fig. 13, where the splitting edges have the same directions as the original edges.

5.2 Compute Containment Values

After intersection points have been found and edges split, the containment relationships of edges with respect to island loops are assigned containment values, which are defined as follows.

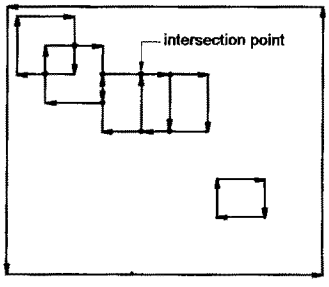

Fig. 12. Find intersection points.

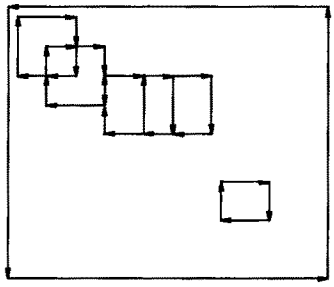

Fig. 13. Split line segments.

OUT indicates that the given edge is outside an island loop, and the containment value is assigned the value "0".

ON^+ indicates that the edge overlaps a part of an island loop, that both overlapping edges have the same direction, and the edge is assigned the value "1".

ON^- indicates that the edge overlaps a part of an island loop, that two overlapping edges have opposite directions, and the edge is assigned the value "2".

IN indicates that the given edge is inside an island loop, and the edge is assigned the value "3".

In this research, the computation of containment values can be simplified since all island loops are rectangular boxes, and edges are either horizontal or vertical. The containment value of an edge can be obtained from the relationship of the midpoint position of the edge with respect to a box loop.

1. If the midpoint of an edge is outside a box loop, the edge is outside the box loop and the containment value of the edge is OUT(0) (see Fig. 14(*a*)).
2. If the midpoint of an edge is on an edge of a box loop, the edge overlaps parts of the box loop. If the edge has the same direction as the overlapped edge, the containment value for this edge is $ON^+(1)$; if the edge has an opposite direction to the overlapped edge, the containment value for this edge is $ON^-(2)$ (see Fig. 14(*b*)).
3. If the midpoint of an edge is inside a box loop, the edge is inside the box loop and the containment value of the edge is IN(3) (see Fig. 14(*c*)).

After the procedures for the classification and computation of containment values are completed, the containment values for a number of box loops, as in Fig. 15, are computed by the following procedure.

Initially, all edges of all boxes are given containment values of zero. All edges of a box loop are taken to compute containment values with respect to each of the other boxes.

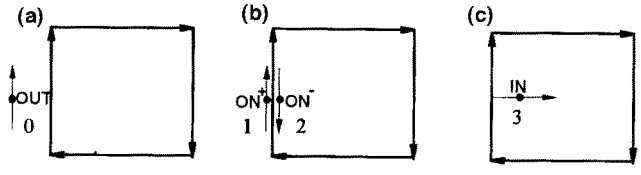

Fig. 14. Find containment values.

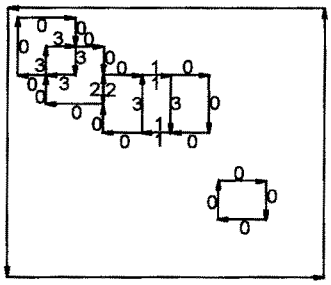

Fig. 15. Computing containment values.

An old containment value is replaced with a new containment value if and only if a newer one is greater than an older one. The resulting value is the correct containment value for such an edge in the locating layer.

Figure 15 shows the results of Fig. 13 after the containment value for every edge has been found. It is found that the edges with containment values 2 and 3 can be deleted from an island structure. Where more than one edge with a value of 1 exist, only one of them is retained, and all the other overlapping edges are deleted. With this deletion procedure, the left edges with 0s and 1s are put into an edge stack for finding new island loops.

5.3 Forming New Island Loops

In forming a new loop, start by obtaining a starting edge from the edge stack and sequentially find each next connecting edge. A new loop is formed when the end edge meets the starting edge.

As shown in Fig. 16, there appears more than one next connecting edge. If such a condition occurs, based on the direction of the current edge L_1, the edge with maximum relative rotation angle is selected as the next edge from among the connecting edges. The relative angle is defined as positive for counterclockwise and negative for clockwise directions.

There are some island areas having one point contact (see Fig. 17). The reason for selecting the maximum angle is to make those islands with one point contact form a single loop, not to break into several loops.

After a new loop is found, the direction for the loop has to be determined. Newell's method is used to derive the loop direction for a sequence of loop vertices [14]. After completing the above restructuring, the case shown in Fig. 15 results in the correct pocket structure as in Fig. 10.

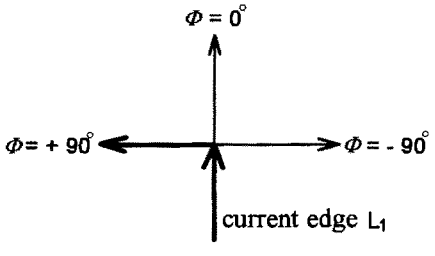

Fig. 16. Edge with more than one next connecting edge.

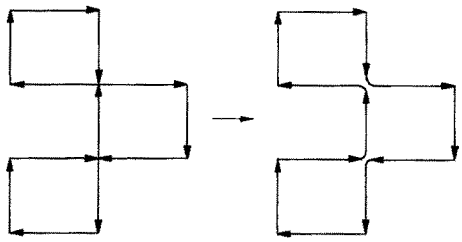

Fig. 17. Islands with one point contacts.

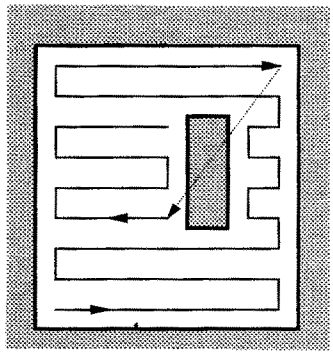

Fig. 18. Continuous zigzag cutting path.

6. Path Planning for Layer Pocketing

Since the representation of the cutting boundary for each pocketing layer is established, current pocketing methods can be used for tool path planning of each layer. Among the current available pocketing methods, continuous zigzag, as in Fig. 18, is selected for rough cutting in this research, since it has the advantages of easy path generation and very few undercutting problems [15].

For a B-spline surface, pocketing paths are generated for each layer from the top down to the bottom. The combination of paths for all layers is then the paths for the rough cutting of the B-spline surfaces.

7. Implementation and Results

The proposed method for B-spline rough cutting has been implemented in Visual C++ on Windows on a PC. The system infrastructure is shown in Fig. 19, where the surface editing and rough cut planning are two major subsystems.

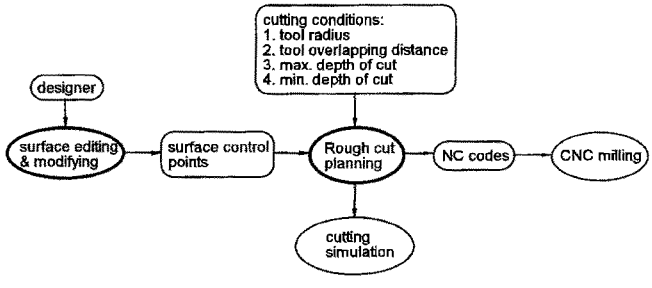

Fig. 19. Infrastructure of rough-cut system.

There are 3 major functions in the surface editing and modifying subsystem:

1. Read in the data of a control net and construct a B-spline surface model.
2. Transform the B-spline surface into piecewise Bezier surfaces and display the surface on a computer screen.
3. Move control points and modify the surface shape until the shape is satisfactory.

In this editing subsystem, a plane model for a control net has been used to help select control points in order that the points on the screen can be clearly distinguished and the topological relationships between control points are explicitly displayed [16].

The rough cut planning subsystem has two major functions:

1. Construct the surface approximation model according to those piecewise Bezier surfaces transformed from the B-spline surface.
2. According to the user-provided cutting conditions, the tool paths are generated through layer slicing and structuring of pocketing boundaries.

The tool paths are further simulated for overcut checking on a computer. The generated CL data can be transformed into NC code and transferred to a CNC machine for actual cutting.

Parts of the simulation for the rough cut of a water tap is shown in Fig. 20. The cutting conditions are: tool radius = 6 mm, overlapping = 0 mm, maximum depth of cut = 2 mm, minimum depth of cut = 1 mm. The tool paths for layers 1, 2 and 3 of 15 layers are shown in Fig. 20. The cutting result using the stock material of acrylic is shown in Fig. 21.

In rough cut planning, cutting efficiency can be increased by changing the following settings:

1. Use a larger radius of cutter.
2. Increase the maximum depth of cut.
3. Increase the minimum depth of cut.

The total distance of tool paths for each layer will be reduced by using a larger cutter radius, but some narrow regions may restrict the cutter from entering, resulting in larger undercuts. By increasing the maximum depth of cut, the number of layer subdivisions will be reduced. By increasing the minimum depth of cut, the number of merges will be increased. Both increases will reduce the number of cutting layers, but at the expense of enlarging the undercutting volume.

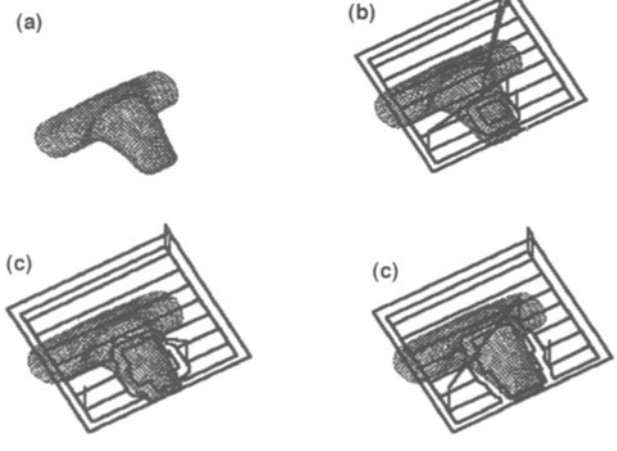

Fig. 20. Rough cut for a water tap. (*a*) Surface shape. (*b*) Tool path of first layer. (*c*) Tool path of second layer. (*d*) Tool path of third layer.

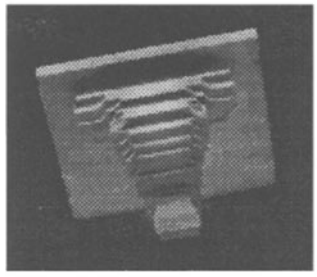

Fig. 21. Result of rough cut for a water tap.

8. Conclusions and Future Work

In this research, an efficient and robust rough cut planning method is proposed. The method uses non-uniform layered subdivisions with convex hull boxes and a simple union operation to compute the pocketing boundaries for cutting layers. Geometrically, the generated paths can avoid overcutting completely and remove most unwanted material volume efficiently. Computationally, the search for surface and slicing plane intersections required in most current methods has been avoided, and thus the method reduces the problems of numerical instability and the errors in obtaining pocketing boundaries. As usual, path planning by this method, which is based on 3-axis NC machines, cannot reach side hollows.

There are three directions for future work:

1. Generate tool paths for multi-patched B-spline surfaces based on this method.
2. Determine the cutter radii for each layer automatically and optimise tool changes such that rough cut efficiency can be optimised.
3. Investigate the tool path planning for accurate and efficient finish cut.

With this work, it is expected that automatic and efficient production of free-form surfaces can be achieved in the near future.

Acknowledgement

The authors are indebted to the Metal Industrial Research and Development Center, Taiwan for its support under Project 84EC2A15094.

References

1. W. Chungwatana, A. C. Lin and W. F. Lu, "Automated process planning and numerical control code generation for mechanical parts with sculptured surfaces", The Second International

Conference on Automation Technology, Vol. 2, pp. 171–178, July 1992.
2. R. B. Jerard and R. L. Drysdake, "Geometric simulation of numerical control machining", Proceedings ASME International Computers in Engineering Conference, pp. 129–136, 1988.
3. C. F. You and C. H. Chu, "NC rough cut machining for solid models", The Second International Conference on Automation Technology, vol. 2, pp. 75–82, July 1992.
4. K. Lee, T. J. Kim and S. E. Hong, "Generation of tool path with selection of proper tools for rough cutting process", Computer Aided Design, 26 (11), pp. 822–831, November 1994.
5. W. Boehm, "Inserting new knots into B-Spline curves", Computer Aided Design, 12 (4), pp. 199–201, July 1980.
6. W. Boehm, "Generating the Bezier points of B-Spline curves and surfaces", Computer Aided Design, 13 (6), pp. 365–366, November 1981.
7. W. Boehm and H. Prautzseh, "The insertion algorithm", Computer Aided Design, 17 (2), pp. 58–59, March 1985.
8. L. Piegl and W. Tiller, "Curve and surface construction using rational B-Splines", Computer Aided Design, 19 (9), pp. 485–498, November 1987.
9. J. Farin, Curves and Surfaces for Computer Aided Geometric Design, Academic Press, 1990.
10. D. F. Rogers and J. A. Adams, Mathematical Elements for Computer Graphics, McGraw-Hill Publishing Company, 1990.
11. B. K. Choi, Surface Modeling for CAD/CAM, Elsevier, 1991.
12. N. Anantakrishnan and L. Piegl, "Integer subdivision algorithm for rendering NURBS curves", The Visual Computer, 8 (3), pp. 149–161, 1992.
13. M. Mäntylä, An Introduction to Solid Modeling, Computer Science Press, 1987.
14. I. E. Sutherland, R. F. Sproull and R. A. Schumacker, "A characterization of ten hidden-surface algorithms", ACM Computing Surveys, 6, pp. 1–55, March 1974,
15. C. S. Yang, "Tool path generation of pockets with islands", MS Thesis, National Chung-Hsing University, Taiwan, June 1993.
16. C. H. Lin, "Computer aided design using a plane model", MS Thesis, National Chung-Hsing University, Taiwan, July 1994.