A 3D Rough-cut Model Generation Algorithm Based on Multi-resolution Mesh for Sculptured Surface Machining

Jiang Zhu**, Tomohisa Tanaka** and Yoshio Saito**

**Department of Mechanical and Control Engineering, Tokyo Institute of Technology,
I1-33, 2-12-1, O-okayama, Meguro, Tokyo, 152-8552, Japan
E-mail: zhujiang@mep.titech.ac.jp

Abstract

In the rough-cut stage of sculptured surface machining, using rough-cut models with simple geometric complexity to generate tool paths could effectively reduce the machining time and improve the machining efficiency. In this paper, a 3D rough-cut model generation algorithm for sculptured surface machining is presented. The proposed algorithm is based on offset and multi-resolution mesh. Offsetting guarantees that the rough-cut models always involve the input model, while multi-resolution mesh allows the simplification of a complex model into coarser approximations. This proposed algorithm could generate 3D rough-cut models with different resolutions from the input model. These rough-cut models have simple geometric complexities and incremental volume changes. The experimental results indicate that the generated rough-cut models are suitable for sculptured surface machining.

Key words: Rough-cut, Sculptured Surface Machining, Multi-resolution Mesh, Self-intersection Removal, Mesh Simplification

1. Introduction

In the material removal process, such as CNC machining and robot integrated hotwire cutting(1-3), the sculptured surface is machined by executing both rough-cut and finish-cut. In the rough-cut stage, the main goal is to remove the material in the most efficient manner. While producing the desired surface finish and accuracy is the primary driving factor in finish-cut. In such process tool path planning is the fundamental task. Generally the tool paths are approximated by straight lines or circular arc segments, since most CNC machines and robots provide only line or arc interpolators. In such process the data size becomes quite large and the cutting tool must accelerate and decelerate between each segment. In order to overcome those drawbacks, recently some CNC manufacturers and researchers(4) proposed NURBS (Non-uniform Rational B-spline) interpolators for NURBS-based machining. However, the machining time is increased because of dwell between consecutive CNC control commands, and a large percentage of the machining time is spent on waiting for the next instruction. For a given machining tolerance, a complex surface with more peaks and valleys needs more NC commands compared to a simple surface with less peaks and valleys for machining. Therefore, machining a complex shape requires more time than machining a simple shape of the same surface area because of more dwell. However, in the rough-cut stage there is no necessity for the tool path to have all the geometric details of the final workpiece. The machining efficiency may be improved by using simple shapes away from the desired surface.
To handle surface rough machining, most researchers are likely to employ spatially ordered approaches. In order to reduce the machining time, Stifter \(^5\) presents a dexel model for analysing and simulating machining accuracy. Tangelder and Vergeest \(^6\) propose an algorithm, which uses a voxel model to represent an object. Also Chen and Hu \(^7\) adopt a grid–set model for rough milling. It uses uniform grid height array to approximate the tool, toolholder and component. All of the above methods have shown that the spatially ordered approaches have their own specific applicability in rough machining. Differing from spatially ordered method, there are other offset-based approaches for rough-cut tool path planning. In practice, zigzag and contour parallel methods are most widely used for pocketing and contouring, but they may not generate optimal tool paths when the shape becomes complex. Narayanaswami and Pang \(^8\) use the theory of B-spline wavelets to solve that problem. It decomposes a complex curve into various resolution curves using wavelet theory, and uses curves in different resolution as tool paths under different conditions.

Recently, multi-resolution mesh and mesh simplification methods have been intensively researched in the field of computer graphics. They are widely used for data reduction, rendering, remodelling, progressive transmission, and mesh compression. Mesh simplification reduces the number of geometric elements needed to represent a model in retaining a good approximation to the original model and appearance. Multi-resolution mesh provides a series of approximations of the original model with different number of geometric elements, which can be used in different applications.

In the previous research \(^1\), a multi-resolution mesh for sculptured surface machining has been developed, which is capable of generating faithful approximation of the original model with high-quality mesh. In this paper, a rough-cut model generation algorithm based on multi-resolution mesh is presented. From an input model, the proposed algorithm could generate a series of rough-cut models in different resolutions with simple geometric complexities and incremental volume changes. Rough-cut models with higher-resolution are used to generate tool paths close to the object boundary, and models with lower-resolution are used to generate tool paths away from the object boundary. Also, the produced rough-cut model is composed of large flat faces, which make it suitable for material removal process.

2. Requirements for Rough-cut Models in Sculptured Surface Machining

In some material removal processes, such as CNC machining, the sculptured surface is machined layer-by-layer. The input mesh is required to possess a valid topological structure. In such process, slicing or contouring techniques are widely used to generate the tool path on each layer. Any topological problems, such as self-intersection, degenerate facets, undesired holes or flipped normal vectors, would lead to invalid cross-sections, which cannot be manufactured as layers. In the rough-cut process of the sculptured surface machining, the main purpose is to remove the redundant material efficiently. In order to reduce dwell and save the machining time, the rough-cut models are required to have a geometric complexity as simple as possible. Meanwhile, since any miss-cut of the final workpiece will lead to the failure of the whole machining process, it requires the rough-cut model should always completely involve the final workpiece to prevent the over-cut.

Since the purpose of this approach is the application of sculptured surface machining, the algorithm is developed based on the above requirements. In the following part of this paper, the algorithm will be described and the solutions to meet these requirements will be addressed respectively.

3. Algorithm of Generating Multi-resolution Rough-cut Models

While NURBS is the standard for exact curve and surface, triangular mesh is probably the most popular representation for approximate shape in engineering applications, as well as in computer graphics. In this developed algorithm only the triangular mesh is considered,
since arbitrary mesh can be converted into triangle mesh through a triangulation process. The input mesh is assumed to be a closed manifold surface without internal void and self-intersection.

The rough-cut model generation algorithm is based on offset and multi-resolution mesh\(^1\). It first offsets the input model for a given distance to generate a triangular mesh, which is apart from the original model. Then, it regularizes that offset mesh, and removes the self-intersections of the offset mesh. After that, the regularized surface is simplified based on edge collapse transformation. In the simplification process, a local error metric and a global error bound are used to guide the simplification sequence. They ensure that the simplified approximation will never intersect with the original model. By repeatedly applying the above steps, a series of such approximations with different resolutions will be generated. It continuously transforms the input model into rough-cut models with simple geometric complexities and incremental volume changes. Following the reverse sequence of the generating multi-resolution rough-cut models, the sculptured surface can be machined step by step from the raw material block.

3.1. Offsetting the Input Mesh

To offset a triangular mesh, the most direct method would be offsetting each triangular facet with the given distance in its corresponding normal direction. Such method is widely used in CNC machining for generating the cutter location path. However, this will result in intersections or gaps between two neighbouring triangles in the offset surface. In order to make a closed 3D offset surface, the developed algorithm offsets each vertex instead of offsetting each triangular facet. Fig.1(a) illustrates the process of offsetting a triangular facet. For a user specified offset value \( \epsilon \), the position of each vertex is extended along its normal vector by \( \epsilon \) to create an offset surface. The normal vector of vertex can be calculated using the area-weighted averaged normal vectors of the triangles that are connected to this vertex.

![Figure 1: Illustration of Offset, and Self-intersection](image)

While offsetting is able to obtain a triangular mesh apart from the original mesh by a given distance, in some local areas with concave features, the resulting mesh may have self-intersections, which is illustrated in Fig.1(b). Fig.2 shows the original mesh of cow model, and its triangular mesh after offsetting. The offset value \( \epsilon \) is assigned as the...
percentage of its diagonal of model’s bounding box. In Fig.2(b), the intersected triangles are highlighted in the red color. Since the mesh with self-intersection cannot be used to generate valid tool path for surface machining, the offset mesh should be regularized. In the next section, a self-intersection removal process will be introduced.

3.2. Removing the Self-intersection

In order to preserve the topology of the input mesh, and provide a manifold triangular mesh, a mesh regularization algorithm is developed to remove the self-intersections. In the triangular mesh with self-intersections, the intersection lines separate the mesh into two parts, one part is on the outer surface of the mesh, which can be seen from the outside of the mesh, and the other part is inside the mesh surface, which cannot be seen from outside(9).

The objective of mesh regularization is to find a set of triangles, which define the outer surface, from the triangular mesh after offsetting, and to reject the triangles, which are inside the intersection areas. The overall procedure of the mesh regularization algorithm can be outlined as three steps.

1. It first computes the intersected triangles in the mesh, and finds out the intersection lines among the intersected triangles.
2. Since in the intersected triangles, some regions are on the outer surface of the mesh and some regions are inside the surface, in this step the intersected triangles are sub-triangulated according to the intersection lines.
3. After the sub-triangulation, the triangles left in the mesh can be classified into two groups. The triangles on the outer surface of the mesh are called as outer surface triangles, and the triangles inside the mesh surface are called as inside triangles. In this step, it finds out all the outer surface triangles. It starts with finding a seed triangle, which lies on the outer surface of the mesh. Then the outer surface triangle group grows from the seed triangle to its neighbouring triangles until all the triangles on the outer surface are found. After that the remaining triangles, which are inside the surface, are removed. And a regularized offset mesh can be calculated.

3.2.1. Checking the Self-intersections And Computing the Intersection Lines

Before removing the self-intersections of the mesh, the algorithm needs to find out which triangles intersect with each other. Intuitively, the triangular mesh self-intersection problem can be considered as a set of triangle-to-triangle intersection (TTI) problems and can be solved by the known TTI techniques(10). However, this approach seems very time consuming because every pairs of triangles in the mesh have to be compared. Since an efficient structure to reduce the number of TTI test is necessary, a tracing neighbour of intersected triangles (TNOIT)(11) technique is used in this approach to calculate the intersection lines. Generally, the intersection parts of triangular mesh are represented in terms of straight-line segments, which can be determined by considering a pair of triangular facets at a time. The intersection lines can be formed progressively by connecting each line segment end to end. The following are the strategies for constructing the intersection lines by neighbour tracing technique.

1. Find out and record the neighbours of each triangle in the triangular mesh. If two triangles have a common edge then they are neighbours to each other.
2. Remove the non-intersection conditions by AABB(12) (axis-aligned bounding box) check. As intersections only happen in a few parts of the mesh, in order to increase the computation efficiency, it is necessary to eliminate the triangles that do not possibly have any intersection. If two triangles, whose AABBs do not overlap with each other, they obviously have no intersection.
3. Check the intersections of the triangle pairs whose AABBs overlap with each
other. If there is one pair, whose triangles intersect with each other, it is selected as the seed triangle pair for intersection line growing. Here the TTI method suggested by Tomas\textsuperscript{(10)} is used for testing the intersection in each triangle pair.

4. Determine the associated intersection line by tracing neighbouring elements. Along each intersection line, the intersected triangles in the mesh are neighbours to one another. Making use of this neighbouring relationship, the intersection line can be constructed by tracing neighbouring triangles one after the other.

Figure 3 gives the illustration of intersection line growing process. For example, triangle $t_a$ and $t_b$ intersect with each other, as shown in Fig.3(a). The intersection line segment is expressed as $mn$. As shown in Fig.3(b), since the intersection line segment of $t_a$ and $t_b$ ends on point $m$, which is on the common edge of triangle $t_a$ and its neighbouring triangle $t_c$, it can be concluded that $t_c$ also intersects with $t_b$. The intersection line grows from triangle $t_a$ to triangle $t_c$. Similarly, as shown in Fig.3(c), the intersection line continues growing from triangle $t_b$ to triangle $t_d$ by neighbour tracing.

In the neighbour tracing process, if the intersection line goes back to the starting point, a closed loop will be formed; otherwise, an open chain can be defined with two end points. Generally, in the offset mesh, there are more than one intersection loops or chains. In this case, the intersection lines can be traced out one by one. To improve the efficiency, pairs of triangles already considered for intersections are flagged. When all the pairs of candidate triangles are flagged, it indicates that there are no more intersections, the self-intersection checking process terminates. Fig.4 shows the intersection lines in the offset mesh of cow model, which is shown in Fig.2(b). The intersection lines are highlighted by the red lines.

![Figure 4: Intersection Lines in the Offset Mesh of Cow Model](image)

3.2.2. Sub-triangulating the Intersected Triangles

After computing the intersections, relevant intersection line segments are stored for each intersected triangle. These segments divide the each intersected triangle into several regions, which are either inside or outside the mesh surface. In the self-intersection removal process, the regions outside the mesh surface should be preserved, and the regions inside the surface should be eliminated. In order to produce a consistent mesh, each intersected triangle needs to be sub-triangulated. The sub-triangulation is constrained along the

![Figure 3: Illustration of Intersection Line Growing Based on Neighbour Tracing](image)
intersection line segments to separate the regions inside and outside the mesh surface. For example, triangle \( t_k \) intersects with other triangles along two intersection line segments, as shown in Fig.5. The sub-triangulation process can be outlined as three steps.

1. Split each edges of triangle \( t_k \) by all the intersection points, and split the intersection segments if they intersect with each other. The splitting process is illustrated in Fig.5(a).
2. Decompose triangle \( t_k \) into several sub-regions, which are enclose by its edge segments and intersection line segments. As shown in Fig.5(b), triangle \( t_k \) is decomposed into four sub-regions, which are distinguished in different colors.
3. Triangulate each sub-region. Here the triangulation method proposed by Kalvin\(^{(13)}\), is used in this approach. As shown in Fig.5(c) and Fig.5(d), it first decomposes each sub-region into star polygons if it is not a star polygon. Then, each star polygon is triangulated.

![Figure 5: Illustration of Sub-triangulation](image)

**3.2.3. Growing the Outer Surface Triangle Group**

A closed manifold surface requires that every edge in the surface connects exactly two faces\(^{(14)}\). However, after splitting the intersected triangles into sub-triangles, along the intersection lines, four triangles share one common edge. Here this kind of edge, which connects four triangles, is called as non-manifold edge. Oppositely, the ordinary edge, which connects two triangles, is called as manifold edge. In order to compute a valid manifold surface, the non-manifold edges should be trimmed into manifold edges, and the triangles, which are inside the outer surface, should be eliminated. Here, an outer surface triangle group growing process based on frontier edge is developed to find all the outer surface triangles. In the process, the outer surface triangle group \( \mathcal{S}_T \) is defined as the triangles, found to be on the outer surface\(^{(15)}\), and the frontier edge group \( \mathcal{E}_f \) is defined as the border edges between the outer surface triangle group \( \mathcal{S}_T \) and the unvisited triangles. The outer surface triangle group growing process works following the next steps, which are illustrated in Fig.6.

1. All the triangles are marked as unvisited. The growing process starts with finding a seed triangle, which is expected to be on the outer surface. As shown in Fig.6(a), since the triangles touching the AABB of the input mesh are obviously on the outer surface\(^{(15)}\), one triangle among them is selected as the seed triangle. The seed triangle is put into the outer surface triangle group \( \mathcal{S}_T \), and its three edges are put into the frontier edge group \( \mathcal{E}_f \). The edges in \( \mathcal{E}_f \) form a closed loop.
2. As shown in Fig.6(b), edge \( e \) in \( \mathcal{E}_f \) is a manifold edge, which connects two triangles, \( t_j \) and \( t_k \). Triangle \( t_j \) is in group \( \mathcal{S}_T \). The outer surface triangle
group grows from \( t_j \) to \( t_k \). Triangle \( t_k \) is put into \( T_S \), and edge \( e_p \) is removed from \( E_f \). For the other two edges of triangle \( t_k \), if it is not in \( E_f \), put it into \( E_f \); else remove it from \( E_f \). This operation repeats until all the edges left in \( E_f \) are non-manifold edges, or \( E_f \) is empty. If all the edges in \( E_f \) are non-manifold edges, go to Step 3 to find the next seed triangle. If \( E_f \) is empty, go to Step 4.

Figure 6: Illustration of Outer Surface Triangle Group Growing Process

3. As shown in Fig.6(c) and Fig.6(d), edge \( e_q \) in \( E_f \) is a non-manifold edge, which connects four triangles, \( t_a, t_b, t_c, \) and \( t_d \). Triangle \( t_a \) is in group \( T_S \). The algorithm selects the valid triangle to continue the outer surface triangle group growing according to the normal compatibility. As shown in Fig.6(d), triangle \( t_a \) lies in plane \( P_a \), and \( \vec{n}_a \) is the normal vector of \( t_a \). Plane \( P_a \) separates the space into two half-spaces. Since the valid triangle lies on the outside of the intersection parts, the valid triangle should be in the half-space, which \( \vec{n}_a \) points to. Therefore, triangle \( t_d \) is selected as the valid triangle, which connects \( t_a \) on the outer surface, and it is put into \( T_S \) as the next seed triangle for outer surface triangle group growing. For the other two edges of
4. As shown in Fig. 6(e), the unvisited triangles, which are inside the intersection parts, are removed from the mesh. The outer surface triangle group growing process completes.

Figure 7 shows the regularized offset mesh of cow model. From the enlarged image compared with Fig. 2(b), it can be found that along the intersection lines, the self-intersections are efficiently removed. Noticing that the triangle number in the regularized offset mesh is larger than the original model, it is because in the sub-triangulation process, it introduces new triangles to remove the self-intersection. However, in the later simplification process, the model will be highly simplified and there is no need to consider the complexity increasing in this regularization process.

Figure 7: Regularized Offset Mesh of Cow Model, Self-intersection removed (7,505 Triangles)

3.3. Simplifying the Offset Surface

After offsetting and regularization, a surface outside the original model without self-intersection can be generated. However, it still contains a lot of detailed geometric features, which will be time consuming for rough-cut machining. In order to generate the rough-cut models with simple geometric complexities, multi-resolution mesh, which was developed in the previous work, is implemented to simplify the surface. In this section, the algorithm of multi-resolution mesh is outlined, and the method of using multi-resolution mesh to generate rough-cut model is introduced.

3.3.1. Edge Collapse According to Vertex Importance Value (VIV)

The developed multi-resolution mesh algorithm simplifies the model based on edge collapse transformation. In each edge collapse transformation, it unifies two adjacent vertices into a single vertex. One vertex and two adjacent triangles vanish in the process. Therefore, the initial mesh can be simplified into a coarser approximation by applying a sequence of successive edge collapse transformations.

One among the main tasks of simplification process is the selection of candidate edge to collapse. In the algorithm, the edge selection sequence is guided by an error metric, which is called vertex importance value (VIV). This error metric, VIV, reflects the local geometric feature of the model, and also helps to eliminate the generation of sliver triangles. Furthermore, it is mathematically simple to compute. In practice, according to the geometric feature, the VIV of a vertex is given as the product of its local curvature and the length of its shortest connecting edge. The detailed discussion on calculating VIV can be found in Ref.(1). In the simplification process, the vertex with the minimal VIV in the mesh is collapsed first, and it is collapsed with its closest adjacent vertex. Thus, there are two loops to search the candidate edge to collapse. The outer loop is searching all the vertices in the mesh to locate the one with the minimal VIV. The inner loop is searching all the edges connecting that vertex to find out the shortest edge.

Another task is the placement of the new vertex after the collapse transformation. Considering that complex mesh models usually have tremendous data quantity, the original
vertices subset is used as new vertex position to increase the simplification efficiency. In practice, the vertex of the candidate edge, which has a higher VIV, is selected as the position of the new vertex.

3.3.2. Boundary Envelope

Boundary envelopes (17) are originally designed to guarantee that the distance between the original model and the approximations is within an absolute threshold. It generates two surfaces for an input mesh, one surface on the outside, and the other on the inside. These two surfaces are defined as boundary envelopes, and used to guarantee that the simplification results would never deviate over these boundaries.

Here in this application, since no miss-cut of the final workpiece is permitted in the rough-cut machining process, it requires that the rough-cut model should be always larger than the original model to be machined, or should always involve the original model. Therefore, using the original model as the boundary envelope will guarantee that the simplified surface will never deviate over the original model. In the simplification process, intersection test is performed to each edge collapse transformation. If the simplified approximation does not intersect with the original model after the edge collapse transformation, the transformation is acceptable. If there occurs any intersection, the transformation is skipped. The simplification process stops till there is no edge to collapse or the simplified approximation reaches a user specified resolution, which is indicated by the number of triangles left in the approximation. Thus, by keeping the simplified surface outside the original model, the algorithm guarantees that the simplified surface never intersects with the original model and it does not self-intersect.

3.4. Rough-cut Model Generation Algorithm Summary

The specific details of the algorithm have been described, and the rough-cut model generation algorithm can be summarized as the following outlines:

1. Read the original model, which is the final workpiece to be machined.
2. Offset the input mesh.
3. Remove the self-intersection and regularize the offset mesh.
4. Calculate VIV for each vertex, and make an order list for all the vertices by VIV.
5. Repeat the following steps until it reaches the desired approximation, which is the rough-cut model to be machined.
   a) Select the top vertex from the order list, and take the edge with its closest adjacent vertex as the candidate edge to collapse. Calculate the position of the new generated vertex.
   b) Check the intersection of the collapse transformation with the original model. If failed, skip this collapse transformation and go back to step a.
   c) Perform the edge collapse transformation, and calculate the VIV of the vertices affected by this transformation.
   d) Update the order list.

4. Experiment of Rough-cut Model Generation

The algorithm is implemented on a P4 2.8GHz PC with 512M memories, using JAVA language. Many objects have been tested using this developed program. Fig.8 gives the rough-cut models of the cow model, which originally have 5,807 triangles and is shown in Fig.2(a). Fig.8(a) gives the rough-cut model using a value \( \varepsilon = 2\% \) to offset the input mesh. The mesh after offsetting and its regularized mesh are shown in Fig.2(b) and Fig.7 respectively. And Fig.8(b) gives a further simplified rough-cut model using a value \( \varepsilon = 4\% \) to offset the input mesh. The algorithm outputs these models until there is no valid edge to collapse. In Fig.9, the rough-cut models of human head model are given. The original model...
is composed of 1,390 triangles, which is shown in Fig. 9(a). The rough-cut models, which are generated by applying different values to offset the original head model, are shown in Fig. 9(b) and Fig. 9(c). And Fig. 9(d) gives the hybrid view of the original model and the rough-cut model, which are shown in Fig. 9(a) and Fig. 9(c) respectively.

From the results it can be seen that the input model is highly simplified, most of the local details have vanished. The rough-cut models are mainly composed of large, flat faces, which is easy for rough machining. Additionally, by keeping the simplified approximation on the outside of the original model, the topological features of the original model are strictly preserved. Meanwhile, since the algorithm uses VIV to select the edge to collapse, it always leads to generate regular triangles to represent the flat surface in the approximation, and eliminates the generation of sliver triangles. These virtues make the approximation easy to be machined.

In some cases, the original models are generated using the 3D laser scanning devices. Those meshes are usually composed of thousands of triangles. It will be time consuming for the algorithm to check the self-intersections and to regularize the offset mesh. In order to generate the rough-cut models of those complex models, which have a large amount of data, the algorithm firstly simplifies the original mesh into a coarser approximation. The multi-resolution mesh \(^{(1)}\), which was developed in the previous work, could simplify the
input mesh within a user specified error tolerance. After simplification, the simplified approximation is used as the input mesh to generate the rough-cut models. Fig.10 gives the example of rough-cut model generation for model with large data amount. Fig.10(a) shows the original mesh of armadillo model, which has 345,944 triangles. Fig.10(b) shows the simplified approximations of those models within the error tolerance of 0.2%. And the multi-resolution rough-cut models are shown in Fig.10(c) and Fig.10(d). From Fig.10(d), it can be found that the triangle number in the rough-cut model is only about 0.05% of the original model. Although the rough-cut model is highly simplified, the basic shape features of the original model are still intact. Furthermore, this rough-cut model lies completely outside the original model. These virtues make the rough-cut model suitable for efficiently removing the redundant material in the rough-cut machining process.

(a) Original Mesh of Armadillo Model
(345,944 Triangles)

(b) Simplified Approximation of Armadillo Model with Error Tolerance within 0.2%
(7,346 Triangles).

(c) Armadillo Rough-cut Model with Offset Value $e = 1\%$ (752 Triangles)

(d) Armadillo Rough-cut Model with Offset Value $e = 4\%$ (186 Triangles)

Figure 10: Rough-cut Models of Armadillo Model

5. Conclusion

In this paper, a rough-cut model generation algorithm for sculptured surface machining has been presented. The proposed algorithm is based on offsetting and multi-resolution mesh to generate the rough-cut models. It first offsets and regularizes the input model to generate a non-self-intersection surface apart from the original model. Then it simplifies that surface using edge collapse transformation. In the simplification process, error metric VIV and boundary envelope works together to guide the simplification sequence. VIV is used to control the quality of the approximation and eliminate the generation of sliver triangle, and boundary envelope is used to guarantee that the simplified approximation never intersects with the original input model. By applying different values to offset the original model, the rough-cut models with different resolutions can be generated, and they can be used to generate tool path in different rough-cut machining steps to approach the final workpiece. The experimental results indicate that the rough-cut models generated by this algorithm are qualified for sculptured surface machining.
6. Acknowledgement

The author would like to thank the Stanford University Computer Graphics Laboratory, for offering the armadillo model used in the experiments. Prof. Qian in Illinois Institute of Technology also gave the author many instructive suggestions.

7. References