Non-isoparametric Three Axis NC Tool Path Generation for Finish Machining of Sculptured Surfaces

Susan X. Li  Robert B. Jerard
Department of Mechanical Engineering, University of New Hampshire, Durham, NH 03824

Abstract
Sculptured surfaces are usually machined to their final shapes by a series of rough and finish passes. After each pass, excess material may be left at some concave or C1 discontinuous areas. Traditionally these undercut areas are “cleaned up” by repeatedly machining the whole surface with progressively smaller cutting tools. We present an algorithm for automatically identifying areas with excess material, and generating tool paths for machining them to within the desired tolerance.

Keywords:NC machining, sculptured surfaces, computer aided manufacturing

1. Introduction
Sculptured surfaces are used in many industries to represent free form shapes like automotive and airplane bodies. Mathematical representations of these surfaces include the well known Bezier, B-spline, Coons and NURBS forms [FP79, M85]. A numerically controlled (NC) machining program is used to machine models, molds and dies required for manufacture. The cutting tools remove material from the workpiece until the final part is within an acceptable tolerance of the mathematical surfaces. Tolerances are of vital importance in the process, and the tool path generation algorithms must guarantee that deviations between the ideal surface and the actual surface are bounded.

Three axis machining of sculptured surfaces usually starts with rough cutting using a flat-end cutter. Algorithms for roughing are discussed in [CR87, LC91], and many commercial systems exist which can efficiently perform rough cutting. One common approach uses intersection of planes of constant Z value with the surfaces to create a contour map of the surfaces. The flat-end cutter then cuts each contour level to create a terraced workpiece in which the bulk of the unwanted material has been removed [CR87].

In finish machining, the part is repeatedly machined with ball-end cutters of different sizes. The simplest approach used in finish machining is isoparametric tool path generation. Surface points and normal vectors are calculated as a function of U, V parameter space. The tool is then indexed along the surface by incrementing U and V, and by using the surface normal information to find
the position of the center of the spherically shaped cutter. Tool path planning is simply accomplished by holding the V parameter constant and indexing the U parameter, hence the term isoparametric machining. Step-forward increments in U must be carefully chosen since tool movements are linearly interpolated and the chordal deviation between the straight lines and the actual surface must be less than the desired tolerance. Step-over increments in V must be small enough to keep the cusps between the spherically shaped cutter paths to less than the desired tolerance [LO87, HO92].

To whom correspondence should be addressed
A skilled NC programmer relies on experience to select a sequence of cutter diameters for multiple passes of finish machining. A larger tool removes material more efficiently, but also creates the potential for either gouging (overcut) or excess material (undercut). The programmer will try to select the largest possible cutter since a large cutter will produce an acceptable part more rapidly. However, problems can occur if a tool radius larger than the concave radius of curvature of the surface is selected. Gouging may occur (Fig. 1a) or excess material may be left. (Fig. 1b) Since the large cutter may not fit into the concave areas of the surface it is often necessary to remachine these areas with a smaller cutter.

![Figure 1](image)

Figure 1 If the cutter diameter is too large it will either gouge (a) or leave excess material (b).

We have developed non-isoparametric tool path generation algorithms [A90, JADS90, JAD91] which eliminate gouging. However, only by remachining with a smaller cutter can excess material be eliminated. If the whole surface is remachined then a great deal of time will be wasted since the cutter will also remachine the convex areas which have already been correctly cut. It would be very desirable to automatically identify those areas of the surface where there is excess material and only generate tool paths which cut those areas.

The system described here is called an “automatic clean-up” system since it will automatically identify those areas which need to be machined and “clean them up” so that the surfaces are
within tolerance. The following sections briefly summarize our approach for non-isoparametric tool path generation, and then describes the details of the automatic clean-up method.

2. Non-isoparametric Tool Path Generation

There are a variety of methods which can be used to do non-isoparametric machining. The various approaches include:

1. Exact solution for the tool-surface tangent point via iterative methods.
2. Calculation of offset surfaces.
3. Polygonalization of surfaces and solution for tool positions which touch the highest polygon.
4. Discretization of the surfaces into a Surface Point Set (SPS) and solution for tool positions which touch the highest point.

In contrast to isoparametric machining where the tool movement is determined by the u and v parameters, these methods decouple the motion of the tool from the surface description. The user can specify tool movements with regular increments in x and y cartesian coordinates, spiral shaped cuts or radial cuts.

The decoupling of the tool motion from the surface description has the unfortunate effect of making tool positioning much more difficult. The problem is related to the inverse point problem. That is, given the x,y,z coordinates of a point, find the u,v coordinates of the point on the surface. For three axis machining, the problem is likely to be posed this way: given the x,y coordinate of a cutting tool, find the z value of the center of the cutting tool such that the tool is tangent to a point on the surface and does not gouge elsewhere.

The APT drive surface - check surface paradigm [FP79] is an example of an exact solution method which can be quite time consuming, and which is also susceptible to numerical problems [HA88]. Since the algorithm relies on Newton's method for iterative solution, it only converges to the correct solution if it has a starting position which is close to the correct one. In the multiple surface case, the next tangent point might not even be on the same surface as the current tangent point. In this case the APT algorithm will converge to a tool position with a tangent point close to the previous one, but which will probably gouge the adjacent surface. An improved exact method is presented by [HO92] in which gouges are eliminated in the concave areas of a single surface.

An alternative approach is to calculate offset surfaces [CR87, SKA87] and then index the center of the tool along the offset surface. However, the offset surface may contain self intersections and/or gaps that must be detected and eliminated.

Another approach used in some commercial systems represents the surfaces as a set of polygons. This technique was first suggested by Duncan [DM83] in a method that he called polyhedral machining. The algorithms for tessellating a surface into a set of polygons within a certain
tolerance have been investigated by many researchers, and fairly robust routines for dropping the
cutting tool onto the highest polygon can be written [HA88, L93].

In the point based approach the surfaces are discretized into a set of points and tool positions are
calculated by a relatively simple "highest point" calculation. Vogel [V85] describes a point based
approach for a biomedical application where the points are derived from a computed tomography
image of human anatomy. Zhang and Bowyer [ZB86] also create a grid of points on the surface
by casting rays parallel to the axis of the tool into a CSG model. A point based method using a
G-buffer (essentially a modified Z buffer) is discussed by [ST91]. The CHIPS system developed
by Ford Motor company for stamping die manufacturing is based on a set of evenly spaced points
to approximate the surface [J91]. Our approach differs from these primarily in our surface
discretization method which takes into account the required tolerances when choosing points.
The evenly spaced points used in other approaches do not guarantee that the resulting surface will
be within tolerance since they do not bound the errors caused by the surface discretization. This
is a subtle but essential requirement since this error must be bounded to a level smaller than the
required machining tolerances. A complete discussion of our error bounds methodology can be
found in [DJSH89, JD91], but it is briefly summarized here for the convenience of the reader.

2.1 Surface Subdivision

The SPS is generated in a systematic manner which guarantees that the tolerance deviation will
not exceed a specified amount. We start by generating a triangulated polyhedron, all of whose
vertices are on the surface. We then subdivide triangles into smaller triangles (again with all
vertices on the surface) until the triangulation is good enough to satisfy the user defined error
bounds. The vertices of the triangulation become our SPS. The triangles themselves play no part
in the simulation (although they are used to graphically display the surface). In order to decide if
a triangulation is "good enough" we must examine the sources of simulation error.

The discrete approximation of the surface introduces errors in two ways. First, the only
representation of the true surface is an interpolation based on the sample points. Effectively we
have replaced the surface by a triangulated polyhedral approximation. This approximation is not
exact, so the true surface differs from the polyhedral approximation. Errors of this type (e_s) are
reduced by choosing more sample points in areas of high curvature. The test for deciding if more
points must be chosen in a given area is, "is the distance between the interpolated surface and the
true surface small in this area?"

The second source of error comes from the fact that we know the cut surface only at the sample
points. It may seem that this gives little information about other points on the surface, but this is
not the case as long as the sample point spacing is small compared to the radius of the tool. The
amount that the tool can protrude into the polyhedral surface (e_p) is a function of the tool
geometry and the distance between points. Since the time to generate tool paths is directly
proportional to the number of points in the SPS we want this triangulation of the surface to have close to the minimum number of points possible.

The maximum possible depth of a gouge can be analyzed in terms of the surface error $e_s$, and the protrusion error $e_p$. In the worst case these two errors combine, so our triangulation must insure that $e_s + e_p < e$, where $e$ is the user-defined tolerance deviation (see figure 2a). However, if the triangle lies entirely outside of the surface the errors are in the same direction and cancel, so it is enough to insure that $\max(e_s, e_p) < e$ (see figure 2b and 2c).

![Figure 2](image-url)

Figure 2  Tolerance error is caused by deviation between the actual surface and the polyhedral approximation ($e_s$) and protrusion of the tool between the surface points ($e_p$). When the polyhedral approximation lies inside the surface, as in (a), the error is the sum of $e_s$ and $e_p$. When the polyhedral approximation lies outside the surface, as in (b) and (c), the error is the maximum of either $e_p$ or $e_s$.

In our previous work [DJS/89] we showed that when the maximum distance between the vertices of any triangle is $d$, the maximum tool protrusion, $e_p$, is given by:

$$e_p = r - \sqrt{r^2 - d^2}/3$$  \hspace{1cm} (1)

where $r$ is the radius of the ball end cutter. Thus the maximum tool protrusion depends on the size of the triangle and the radius of the ball end mill. The tool protrusion error can be made as small as desired by decreasing the size of the triangle. This formula can be used to determine $d$ given $r$ and $e$. Solving for $d$, we get:
\[ d = _6r e_p - 3e_p^2 \]  

(2)

When \( e_p \) is small with respect to \( r \) (which is usually the case), \( d \) can be conservatively approximated by:

\[ d \sim _6r e_p \]  

(3)

The spacing between the points is proportional to the square root of both the cutting tool radius and the tool protrusion error. Since the number of points is inversely proportional to \( d^2 \) we conclude that in areas of low curvature the number of points (and therefore the CPU time) grows linearly with the inverse of the tool radius and the inverse of the desired tool protrusion accuracy.

Our goal is to triangulate each patch with nearly equilateral triangles; a method used to keep the number of points close to the minimum. We start by dividing each patch into strips. Each strip is initially triangulated so that the triangles are as close to equilateral as possible with sides less than 95% of \( d \). In areas of low curvature \( e \), is close to zero and the triangulation is complete, because \( e \_s + e_p < e \).

The subdivision described in the preceding paragraph is performed more efficiently in parameter space than in cartesian space. We measure the width and length of the patch in cartesian space and use this information to decide how many strips are needed and then how many triangles can fit within each strip. We have found it reasonable to assume that equal sized triangles in parameter space will map into equal sized triangles in cartesian space and therefore perform the subdivision in parameter space. If the surfaces were not as well behaved an iterative method could be used to obtain the nearly equilateral triangles.

One advantage of either the point or polygon based methods is that it is applicable to any surface representation (e.g. Bezier, B-Spline, NURBS, Coons patch, CSG surfaces, or even surfaces derived from tomography data). The only modification required is in the routines which calculate the points or polygons.

### 2.2 Tool Positioning

Each tool position is calculated by dropping the tool onto the highest point or onto the tangent plane associated with the highest point [A90] (Fig. 3). Increments between tool positions are in the Cartesian coordinate system and tool paths are parallel across the surface. Algorithms for step-forward and step-over incrementation are discussed in [A90, JAD91].
For efficiency in tool positioning, the points (SPS) are localized with a “bucket strategy”. For each tool position, only the points in the buckets that overlap with the cutter shadow are taken into account. The cutter shadow is the projection of the cutter onto the xy plane (assuming that the cutter axis is parallel to the z axis).

3. Identification of Undercut Areas

The point representation gives us an ideal method for finding points which have been undercut. We simulate the machining process and each discrete point is assigned an error value which is equal to the closest distance from the point to the nearest tool path [JDSHM89]. Points with excess material (undercut) are stored in a file. Note that undercut points are seldom distributed over the whole object surface. To realize automatic clean up, these discrete undercut points need to be sorted into a series of continuous areas. Each area will be called an undercut pocket.

We use a bucketing strategy to identify undercut pockets. Undercut points are sorted into the buckets according to their x and y components (Fig. 4).
To save memory and simplify the sorting process, the data structure of each bucket also contains the x and y extents of the point data for that particular bucket. It is possible that some buckets may only contain a single point. In this case, the rectangular domain of that bucket reduces to a point.

Recursive bucket subdivision is carried out after the bucketing. It starts with the overall rectangular domain which contains all the buckets. The rectangular domain is subdivided by the empty rows and columns. There are several possible subdivision results, as shown in Fig. 5.

1. divided into 4 parts by an empty row and an empty column (Fig. 5a),
2. into 2 parts by an empty row or column (Fig. 5b and 5c),
3. unchanged (Fig. 5d),
4. reduced to a smaller one (Fig. 5e).

These subdivided areas are called undercut subdomains. The row and column indices are stored for each subdomain. The recursive subdivision is continued until the subdomains do not contain any empty bucket rows or columns. These subdomains are stored in a linked list. Each undercut pocket contains the row and column indices which will be used for path planning. The accuracy of approximating the undercut areas depends on the bucket size. A smaller bucket size results in a more accurate approximation but more CPU time is required. Currently, the bucket size is defined to be 1.5 times the cutter diameter.

Figure 5 Five cases in bucket subdivision

4. Tool Path Planning
Once the pockets have been identified, an efficient plan for machining them must be formulated. The goal of the algorithm is to remove the excess material as completely as possible with maximum efficiency. In the following sections we compare three different approaches.

4.1. Block Cutting  
In block cutting, tool paths are generated over the whole rectangular domain of an undercut pocket, as shown in Fig. 6. The domain is obtained by searching all the rectangles stored in the non-empty buckets. In our simulation system, a point represents a small local area. It is possible that points may be located at the edges of the rectangle domain. In order to machine these undercut areas completely, the rectangle domain is enlarged in x direction by 75% of the cutter diameter, and in y direction by the default stepsize (DS) (assuming that tool paths are parallel to the x axis). The enlargement in the y direction ensures that no excess material will be left at the sides of the pocket parallel to the x axis. The enlargement in the x direction is designed to properly handle the entry and exit of the cutter, which will be discussed later.

![Figure 6 Block cutting](image)

4.2. Outer-Pruning Cutting  
The block cutting style works well for the pockets that have a high ratio of undercut area to finished area. However, if the undercut area is distributed as shown in Fig. 7, efficiency can be gained by deleting the unproductive portion of the tool path at the ends of each row. We called this “outer-pruning”.

This algorithm differs from the block cutting in that the outer empty buckets are pruned from the undercut domain. The buckets in a pocket with the same row index are considered as a unit. The rectangular domain that covers the points in the bucket row is calculated individually, as shown in Fig. 7. The rectangular domain that covers the undercut points in a bucket row is stored in a linked list. As in block cutting, each rectangle in the linked list of a pocket is enlarged in x direction by 75% of the cutter diameter and in y direction by the default stepover. To completely machine the undercut areas the tool paths in the vicinity of the interior edges are defined to be the union of the paths over the consecutive rectangles (as shown in Fig. 8).
4.3. Interior-Pruning Cutting

The outer-pruning method provides a minimal improvement in efficiency when the undercut areas form closed loops, as shown in Fig. 9. Perhaps the easiest way to increase efficiency is to lift the cutter when it passes over the interior empty bucket areas. The cutter may need to be lifted several times along each tool path and even though rapid feedrates can be used, these paths are not desirable. We propose an alternative method to finish this type of area called interior-pruning cutting.

In this cutting style, tool paths which have been outer-pruned are further subdivided into several segments. These segments are then connected based on the distances between adjacent path segments in both the x and y directions.
The rectangular domains of the non-empty bucket row segments are stored in a linked list. These segments are linked in order of increasing or decreasing x value. Two consecutive bucket row segments are merged if the space between them is less than twice the cutter diameter. This avoids excessive lifting of the cutter.

Next, the bucket row segments are sorted into a list based upon their x and y ranges (guidance list). A sorting example is shown in Fig. 10. The active element is the ith segment in row j. This segment is copied to the guidance list. The active element is compared with the segments in row j+1 on the x range and distance in the y direction. If the x range of the active element overlaps with that of segment k in row j+1 and the distance in the y direction is less than a pre-defined distance, segment k becomes the active element and is moved into the guidance list. The pre-defined distance is set to twice the distance between tool paths in the y direction. When an element becomes the active element, this element is moved from the bucket row to the guidance list; therefore, the segment list in a bucket row is updated each time. The process is continued to the next row (j+2) and so on. If no segments in the next row are eligible to be added, then a flag is set in the last segment in the guidance list. This flag will be used during tool path generation to indicate that the cutter must be moved to the clearance plane before moving to the next set of segments. The sorting is then restarted with segment i+1 in row j. If there are no segments left in row j, start with the first segment left in row j+1. The procedure is repeated until no segments are left.

Tool path planning now proceeds in the same manner as in outer-pruning cutting. Segments in the guidance list are enlarged and tool paths are arranged on them. Thus each segment is treated like a bucket row in outer-pruning cutting. However, some tool positions must be inserted after the tool paths on the segments with lifting flags. These tool positions guarantee that the cutter does not gouge the part surface when it moves to the next tool path. The tool paths obtained for the example are shown in Fig. 11, where the dashed line indicates that the cutter is lifted to a clearance plane.
5. Dwell Mark Handling
Generally, an undercut pocket is only a portion of an object surface. If a cutter comes straight down onto the surface to machine the pocket, dwell marks will be left on the surface at the start and end positions. As described in the above section, rectangles used to define the lengths of tool paths are enlarged by 75% of the cutter diameter in the $-x$ and $+x$ directions. For smooth entry and exit, an expanded cutter radius is used to position the cutter when the cutter is in the vicinity of the start and end positions along each tool path.

The cutter radius is forced to follow the pattern as shown in Fig. 12, where the expanded cutter radius is defined by half of the parabolic curves at start and end positions along each tool path. The actual cutter size is used when the cutter is between point e and f (Fig. 12). During machining, the actual distance from the center of the cutter to the surface contact point also follows the curve in Fig. 12. The contact points are calculated with an expanded cutter radius.

6. Experimental Results
Our test cases were provided by the Body Design Group of Ford Motor Company. The test surfaces are shown in Figures 13-16.

Cutting conditions for the four cases are listed as follows:
1. Cutter diameter for first pass cutting are as shown in Figures 13-16. The “clean-up” pass used a 12.7mm diameter for all four cases.
2. Cusp height used in step-over calculation for rough cutting: 0.2 (mm)
3. Undercut points are the points where excess material is greater than 0.26 (mm)
4. Feedrate for machining time estimation: 40 (inches / min.)

Results for the test cases are listed in Table 1.

Table 1
Simulation results vs cutting styles

<table>
<thead>
<tr>
<th>Case</th>
<th>Number of Points</th>
<th>Style</th>
<th>Generation time (sec)</th>
<th>Total tool movements</th>
<th>Machining time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zip1</td>
<td>390 (see Fig. 13)</td>
<td>block</td>
<td>3</td>
<td>533</td>
<td>1.28</td>
</tr>
<tr>
<td></td>
<td></td>
<td>outer-pruning</td>
<td>3</td>
<td>533</td>
<td>1.28</td>
</tr>
<tr>
<td></td>
<td></td>
<td>inner-pruning</td>
<td>4</td>
<td>533</td>
<td>1.28</td>
</tr>
<tr>
<td></td>
<td></td>
<td>traditional</td>
<td>20</td>
<td>3,029</td>
<td>8.16</td>
</tr>
<tr>
<td>bumper</td>
<td>7,842 (see Fig. 14)</td>
<td>block</td>
<td>390</td>
<td>4,804</td>
<td>16.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>outer-pruning</td>
<td>330</td>
<td>4,279</td>
<td>14.34</td>
</tr>
<tr>
<td></td>
<td></td>
<td>inner-pruning</td>
<td>303</td>
<td>4,101</td>
<td>13.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>traditional</td>
<td>520</td>
<td>15,365</td>
<td>57.70</td>
</tr>
<tr>
<td>z6324r</td>
<td>3,704 (see Fig. 15)</td>
<td>block</td>
<td>40</td>
<td>1,163</td>
<td>4.71</td>
</tr>
<tr>
<td></td>
<td></td>
<td>outer-pruning</td>
<td>40</td>
<td>1,103</td>
<td>4.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td>inner-pruning</td>
<td>40</td>
<td>980</td>
<td>3.61</td>
</tr>
<tr>
<td></td>
<td></td>
<td>traditional</td>
<td>200</td>
<td>6,310</td>
<td>48.67</td>
</tr>
<tr>
<td>corner</td>
<td>17,966 (see Fig. 16)</td>
<td>block</td>
<td>852</td>
<td>10,748</td>
<td>37.19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>outer-pruning</td>
<td>684</td>
<td>9,068</td>
<td>33.56</td>
</tr>
<tr>
<td></td>
<td></td>
<td>inner-pruning</td>
<td>672</td>
<td>9,133</td>
<td>32.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td>traditional</td>
<td>962</td>
<td>13,597</td>
<td>40.81</td>
</tr>
</tbody>
</table>

In general, the method generates fewer tool movements and requires less machining time than the traditional method (whole surface machining). Among the three algorithms (block, outer-
pruning, and interior-pruning), the interior-pruning cutting style generates the fewest tool movements. Block cutting generates the largest number of tool movements.

Corner is a very complex surface with many concave areas. Large portions of the object surfaces need to be cleaned up. Therefore, the decrease in machining time compared with the traditional method is not as evident.
Figure 13 (zip1) A small test surface consisting of two convex patches which share an edge. The cutter used for roughing is a 76mm ball-end tool. Excess material will be left along the shared edge where there is a concave C1 discontinuity. The cutting is simulated using our method described in [JDSHM89]. A color fringe bar on the left side of (13b) indicates deviations between the mathematical surface and the machined surface. In the black & white version shown here areas undercut by more than 0.3mm are dark. The vertical strip in the center of the plan view of the surface (13b) clearly indicates undercutting at the boundary of the two patches.
Figure 14 (bumper) A portion of a car bumper. It consists of two hundred and sixty three patches, including patches that are nearly vertical, flat, overlapping, concave and convex. It is rough cut with a 38mm ball-end cutter as shown in (14a), and the plan view of the simulation shows undercut areas (14b).
Figure 15 (z6324r) An automotive quarter panel. It is first cut with a 76mm ball-end cutter. Undercut areas are confined to one small area with a small concave radius of curvature (on the left in Fig. 15b).
Figure 16 (corner) An inner panel of a car hood. It is a very complex surface, consisting of one hundred and sixty seven patches with many concave edges. It is initially cut with a 25.4mm ball-end cutter 16(a). The plan view of the simulation (16b) shows widely scattered undercut areas.
7. Future Work

Currently the cutting style is chosen by the user and is fixed for all undercut pockets. It would be possible to automate this selection process. For example, the cutting style can be chosen according to the distribution of the undercut points and ratio of empty buckets to non-empty ones in a pocket.

Currently, tool paths are confined to straight lines. In many cases, curved paths would be appreciably better, especially for the cases like the one shown in Fig. 16. It is a fairly common for the undercut areas to fall along the fillet area where two concave edges meet. The zig-zag straight line cuts used in this paper are not desirable for cutting this area unless they just happen to line up with the edge. A more desirable approach would be to use algorithms developed for contour machining of pockets as described in [H88]. Ideally, the cutter paths will cut parallel to the concave edges.

A fully automatic system should calculate feedrates based on material removal rates, cutting forces, tool deflections and required tolerances. Our initial work in this area is described in [FEJ92].

Acknowledgements

The support of the National Science Foundation under grant DDM-9015851 and of the Ford Motor company are gratefully acknowledged.

References :


[H88] M Held, Computational Geometry for Pocket Machining, RISC-LINZ series no. 88-4.0, Johannes Kepler University, Linz, Austria


