Tool Path Generation for Freeform Surface Models

Gershon Elber and Elaine Cohen
Department of Computer Science
University of Utah
Salt Lake City, UT 84112 USA

Abstract

Generating optimal NC code to drive milling machines for models defined by freeform trimmed surfaces is a difficult problem. In practice, two main approaches are used to generate toolpaths for surfaces, neither of which is optimal, in general. The first exploits the parametric representation and generates isocurves that are uniformly distributed across the parametric domain. This approach is not optimal if the surface mapping into Euclidean space is not isometric. The second approach contours the models by intersecting the surfaces with planes equally spaced in Euclidean space, resulting in a piecewise linear toolpath approximation which is nonadaptive to the local surface geometry. Furthermore, the toolpath generated by contouring is suitable for 3 axis milling but is inappropriate for 5 axis milling.

In this paper, an algorithm developed to adaptively extract isocurves for rendering [9] is adapted and enhanced to generate milling toolpaths for models consisting of trimmed surfaces, and can be used in both 3 and 5 axis milling. The resulting toolpaths do not gouge locally and combine the advantages of both prior approaches. The output toolpath is appealing since it is composed of isoparametric curves and is therefore compact, exact, and easy to process. Furthermore, it is more optimal than the previous methods in that the resulting toolpath is shorter and it provides a direct quantitative bound on the resulting scallop height. This algorithm has been used to compute gouge avoiding toolpaths for automatically milling freeform surfaces without requiring the introduction of auxiliary check and drive surfaces.

Categories and Subject Descriptors: 1.3.5 [Computer Graphics]: Computational Geometry and Object Modeling; Splines; Curve, surface, solid, and object representations.

Additional Key Words and Phrases: NURBs, adaptive isocurves, machining toolpath.

1 Introduction

In order to evaluate the quality of toolpaths, two criteria are introduced. One deals with the validity of a set of toolpaths and the other with its optimality.

Definition 1 A set of curves \( C \) in a given surface \( S \) is called a valid coverage for \( S \) with respect to some constant \( \delta \) if for any point \( p \) on \( S \) there is a point \( q \) on one of the curves in \( C \), such that \( \| p - q \|_2 < \delta \), where \( \| \cdot \|_2 \) denotes the Euclidean distance.

Definition 1 provides a validation criterion on a given toolpath and a tolerance \( \delta \) such that any point on the surface is at most \( \delta \) from the nearest...
toolpath curve. Definition 1 takes into consideration only the distance between an arbitrary point \( p \) on the surface and the closest point on the toolpath. One can easily derive a condition on the scallop height \( S_h \) (see Figure 1) from the distance between two adjacent toolpaths \( D_a \) and the tool radius, \( r \). If two adjacent isocurves of a valid coverage are closer than \( 2r \), where \( r \) us the tool radius, an upper bound of \( 2r \) is immediately established on the scallop height, since the scallop may span at most 180 degrees of the ball end circumference. If a curvature of zero is assumed, a tighter bound of \( r - \sqrt{r^2 - \left( \frac{D_a}{2} \right)^2} \) is established. Using other criteria, such as bounds on the curvature, a tighter bound on the scallop height can be achieved without the zero curvature assumption and without affecting how the rest of the algorithm works.

We also would like to consider the optimality of a valid toolpath.

**Definition 2** A toolpath for a given surface is considered optimal if it is valid and its path length minimal.

Definition 2 considers optimality based only on the cutting motion part of the toolpath. Tool retraction and traversals are not used as optimality conditions in this paper. One might decide to traverse the iso-curves in incremental cross iso-direction so that the portion of the surface of the machining tool that performs the actual milling is approximately the same throughout the milled surface. Any other type of traversal might, in some stage of the milling, require the tool to cut using its entire milling surface perimeter, an undesired tool machining motion. Unfortunately, the time to find an optimal traversal of the piecewise cutting motion toolpath is exponential in nature; more on this problem can be found in [13].

There are two main approaches used to generate toolpaths for freeform surfaces. In one, isoparametric curves are extracted from the surface, usually in equally spaced parametric steps [3, 6, 13, 15]. These isocurves usually span the entire parametric domain of the surface (see Figure 2a) and will be referred to as complete-isocurves. Isocurves that span only a portion of the surface parametric domain (see Figure 2c) will be referred to as sub-isocurves. Although simple to determine, toolpaths created using complete isocurves equally spaced in parametric space are clearly not optimal according to definition 2 and are redundant, as can be seen in the example of Figure 2a, where the toolpath is redundant in the middle region of the surface. In order to guarantee the validity of the toolpath, a certain parametric stepsize is selected for the complete isocurves (for example, derived by the top and bottom regions of the surface in Figure 2a) and which undoubtly leads to a much smaller distances between adjacent complete isocurves in other surface regions than required causing redundancy (in the middle of the surface in Figure 2a). Further, it might be difficult for the user to determine the parameter stepping tolerance that will create valid toolpaths to within a given \( \delta \), even if the top and bottom regions of the surface in Figure 2a are treated separately. The user is interested mainly in the shape of the represented geometry and the associated milling, so the parametric representation of the surface should not require his attention, but be internal.

An alternative method for generating toolpaths is based on contouring planes, in which the surface is intersected by (usually geometrically equally spaced) parallel planes. The intersection curves are used to drive the milling tools [2, 3]. The resulting toolpath is, in general, only a piecewise linear approximation to the real intersection, and the size of the piecewise linear approximations of the intersection curves is usually several magni-
tudes larger than isocurve data. For relatively flat surfaces the contouring algorithm seems to yield acceptable results (see Figure 2b). However, as is the case for the complete isocurves algorithm, some frequently occurring surfaces can be pathological to this contouring algorithm. If the surface has regions almost coplanar to the contouring plane, adjacent contours would be distant from each other, as can be seen from Figure 3b, invalidating the toolpath. How to set the parallel plane spacing and the parallel plane direction to create a valid toolpath is not obvious. Even if an algorithm could be created to adaptively space the contours based on the coplanarity of one surface region, this spacing would be fixed for the entire contoured model. Local coplanarity in one region of the surface would set the spacing for the entire model.

Attempts to improve those techniques have been geared mainly toward local adaptation of the algorithm to specific regions which require a different number of samples to gain the required tolerances [12, 15]. Others used adaptation of scanline fashion rendering [18, 19] to get a piecewise linear approximation for the toolpath.

An adaptive sub-isocurve extraction approach is introduced for rendering in [9]. That scheme provides a more optimal and valid coverage of the surface by adaptively introducing partial sub-isocurves in regions yet uncovered by already created sub-isocurves (definition 1). Furthermore, the algorithm frees the user from the need to determine either the surface parameter spacing or contouring plane spacing, depending on prior method used, and the direction to use to insure adjacent isocurve distances produce a valid coverage. Instead, a bound on the required distance between adjacent sub isocurves can be directly specified, and guaranteed automatically.

It is clear that a valid coverage generated using complete isocurves can be very inefficient (see Figure 2a), which can increase machining time and affect part finish. If the redundant portion of each complete isocurve could be \textit{a priori} detected and not be generated as part of the valid coverage, one would be able to generate a more optimal toolpath with the appeal of the isocurves. The adaptive isocurve extraction algorithm does exactly that for rendering (see Figures 2c and 3c). Since the adaptive isocurve extraction algorithm is developed for rendering in [9] it will be briefly discussed here. The interested reader can also refer to [9].

It is appealing to use isocurves since their representations are compact, exact, and they are straightforward to use as milling toolpaths. Isocurves can be approximated more compactly and accurately using piecewise arcs (and lines), if circular motion is supported by the milling machines than by using piecewise linear approximation alone. Furthermore, isocurves could be sent directly to a milling machine that supports NURBs or Bézier curve toolpaths. Isocurves are also invariant under affine transformations and therefore are view direction independent, unlike the results of the contouring technique. Scallop resulting from isocurve based toolpaths are usually more attractive than those resulting from contoured based toolpaths since they follow the model's basic streamlines (see Plates 2 and 4). Finally, when computing toolpaths for models having trimmed surfaces, it is easier to trim isocurves to the appropriate domains than to trim contours whose parametric domain representations can be arbitrary.

Recent literature [2, 3, 12] has suggested that the contact point numerical improvement approach, such as used by APT [6], is unstable and slow. Computations of a toolpath for a single surface are usually measured in minutes [3, 12]. A different known approach [12] was selected in this work. The model was offset by the tool ball end radius and toolpaths for the tool center were generated using the offset surface.

Section 2 briefly discusses the adaptive sub-isocurve algorithm. Section 3 describes the offset computation required for ball end tool milling, and section 4 deals with the method used for the rough cutting process. Finally, section 5 presents some results obtained from an implementation of the new algorithm for NURBs based models using the Alpha1 solid modeler.
Figure 2: Isocurves are obviously not an optimal solution as a toolpath for this surface (a). Contouring with equally spaced parallel planes is more optimal but is piecewise linear (b). Adaptive isocurves are, in general, more optimal, exact, and compact (c).

Figure 3: Toolpath using isocurves will be not optimal in this complex surface (a). Contouring with equally spaced parallel planes is too sparse in coplanar regions (b). Adaptive isocurves are more optimal, exact, and still correctly spans the entire surface (c).
2 Adaptive Isocurves Extraction Algorithm

Using isocurves as the coverage for a surface, we define adjacency and iso-distance between isocurves.

Definition 3 Two (sub) isocurves of surface \( S(u, v) \). \( C_1(u) = S(u, v_1), u \in [u_1^1, u_1^2] \) and \( C_2(u) = S(u, v_2), u \in [u_2^1, u_2^2] \), \( v_1 \leq v_2 \), from a given set \( C \) of isocurves forming a valid coverage for \( S \) are considered adjacent if, along their common domain \( U = [u_1^1, u_1^2] \cap [u_2^1, u_2^2] \), there is no other isocurve from \( C \) between them. That is, there does not exist \( C_3(u) = S(u, v_3) \in C, u \in [u_3^1, u_3^2] \) such that \( v_1 \leq v_3 \leq v_2 \) and \( [u_3^1, u_3^2] \cap U \neq \emptyset \).

Definition 4 The iso-distance function \( \Delta_{12}(u) \) between two adjacent (sub) isocurves along their common domain \( U \) is equal to

\[
\Delta_{12}(u) = \|C_1(u) - C_2(u)\|_2 = \sqrt{(c_1(u) - c_2(u))^2 + (c_1(u) - c_2(u))^2 + (c_1(u) - c_2(u))^2. \quad (1)}
\]

Given two isocurves, \( C_1(u) \) and \( C_2(u) \), on a surface \( S(u, v) \), one can compute and represent the square of the iso-distance, \( \Delta_{12}^2(u) \), between them symbolically as a NURBs or as a Bézier curve [10]. Computing the coefficients for the representation of \( \Delta_{12}^2(u) \) requires the difference, sum, and product of curves, all computable and representable in the polynomial and piecewise polynomial domains [10]. Furthermore, given some tolerance \( \delta \), it is possible to compute the parameter values where the iso-distance between \( C_1(u) \) and \( C_2(u) \) is exactly \( \delta \) by computing the zero set of \( (\Delta_{12}^2(u) - \delta^2) \) [14]. By subdividing the two curves at these parameters, new sub-isocurve pairs, \( \{C_1(u), C_2(u)\} \), are formed with the characteristic that each pair is always iso-distance smaller or always larger than \( \delta \), in their open interval domains. If the two curves in the pair \( \{C_1(u), C_2(u)\} \) are closer than \( \delta \) in the iso-distance metric then the Euclidean distance tolerance condition is met for that pair. If, however, the two curves’ iso-distance is larger than \( \delta \), a new sub-isocurve, \( C_{12}(u) \), is introduced between \( C_1(u) \) and \( C_2(u) \) along their common domain \( U \) and the same iso-distance computation is recursively invoked for the two new pairs \( \{C_1(u), C_{12}(u)\} \) and \( \{C_{12}(u), C_2(u)\} \).

Starting with the two \( U \) boundaries or two \( V \) boundaries of the surface, the algorithm can invoke this iso-distance computation recursively and ensure two adjacent isocurves will always be closer than some specified distance \( \delta \) by verifying that their iso-distance is not larger than \( \delta \). Since a middle isocurve is introduced if the iso-distance is larger than \( \delta \) and \( \delta \) is small, resulting iso-distances between adjacent isocurves, as computed, are rarely less than \( \frac{\delta}{2} \). Furthermore, since the resulting set of isocurves covers the entire surface \( S \), it can serve as a valid toolpath for \( S \) with distance \( \delta \).

Algorithm 1, the adaptive isocurve extraction algorithm, generates a valid and more optimal coverage by minimizing the cutting speed motion required by minimizing redundancies while providing a bound on the scallop height via the bound on the distance between two adjacent isocurves.

It is important to realize that bounding the distance between adjacent isocurves is a necessary condition to bound the scallop height. It is also shown that if adjacent isocurves are closer than \( 2r \), where \( r \) is the tool radius, an upper bound of \( 2r \) is immediately established on the scallop height. The surface curvature bound (See [7]) could be added to the definition of validity to decide whether to introduce a middle isocurve in algorithm 1 and obtain a tighter bound on the scallop height.

3 The Offset Computation

Since the toolpath generated by the adaptive isocurve algorithm provides a valid coverage of the surface, it can serve as a toolpath for both 3 axis and 5 axis milling. In this discussion, we will concentrate on 3 axis milling using ball end tools. Such a method requires the computation of an offset surface to the model at a distance equal to the radius of the ball end tool. This simplifies the toolpath generation since keeping the center of
Algorithm 1: Adaptive isocurve extraction. Iso-\(v\) curves are assumed.

**Input:**
- Surface \(S(u,v)\).
- Iso-distance tolerance \(\delta\).

**Output:**
- Adaptive isocurve toolpath for \(S(u,v)\).

**Algorithm:**

AdapIsoCurve( \(S(u,v), \delta\) )

begin
\(C_1(u), C_2(u) \leftarrow S(u,v)\) two \(u\) boundary curves.
return AdapIsoCurveAux( \(S(u,v), \delta, \{C_1(u),C_2(u)\}\) )
end

AdapIsoCurveAux( \(S(u,v), \delta, \{C_1(u),C_2(u)\}\) )

begin
\(\Delta_{12}^2(u) \leftarrow ||C_1(u) - C_2(u)||^2\), iso-distance between \(C_1(u)\) and \(C_2(u)\).
if \((\Delta_{12}^2(u) < \delta^2, \forall u)\) then
return \(\emptyset\).
else if \((\Delta_{12}^2(u) > \delta^2, \forall u)\) then
begin
\(C_{12}(u) \leftarrow \text{Middle isocurve between } C_1(u) \text{ and } C_2(u)\).
return AdapIsoCurveAux( \(S(u,v), \delta, \{C_1(u),C_{12}(u)\}\) ) \(\cup\) AdapIsoCurveAux( \(S(u,v), \delta, \{C_{12}(u),C_2(u)\}\) )
end
else
begin
\(\{C_1^i(u),C_2^i(u)\} \leftarrow \text{subdivided } \{C_1(u),C_2(u)\} \text{ at all } u \text{ such that } \Delta_{12}^2(u) = \delta^2\).
return \(\bigcup_i\) AdapIsoCurveAux( \(S(u,v), \delta, \{C_1^i(u),C_2^i(u)\}\) )
end
end

the ball end tool on the offset surface, keeps the tool tangent to the original surface so it can not gouge.

Unfortunately, the exact offset of a freeform piecewise polynomial or rational surface is not representable, in general, as a piecewise polynomial or rational surface [8]. Quite a few methods have been developed in recent years to provide approximations to surface offsets [1, 4, 5, 8]. In [8], a technique to approximate offsets of freeform Bézier and NURBs surfaces by Bézier and NURBs surfaces was developed with the property that error in the approximation surface is *globally* bounded. That global bound can be used directly to determine a global bound on the accuracy of the milling and the amount of gouging that may occur.

Extending the generation of surface toolpaths to models defined using constructive solid geometry [11] and consisting of several, possibly trimmed, surfaces is not obvious. Let \(O(A)\) denote the exact offset of \(A\). It is unfortunate but \(O(A \cap B)\) is not always the same as \(O(A) \cap O(B)\). For example, \(A \cap B\) and hence \(O(A \cap B)\) could be empty but \(O(A) \cap O(B)\) might be a nonempty.

Several types of manufacturing offsets can be defined for piecewise \(C^1\) models [16, 17] that are constructed by constructive solid geometry. In general, one should attempt to prevent gouging even at the expense of not being able to mill the entire model. A \(C^1\) discontinuous concave corner
created by a union of two surfaces cannot be milled using a ball end tool of any size. One could define the offset operator for a piecewise $C^1$ model so that at no time would the center of the ball end tool be closer than its radius to any of the surfaces of the model. Using such definition it can be guaranteed that during the entire milling process,

$$\|c - S_i(u^i, v^i)\|_2 \geq r, \quad \forall i,$$  \hspace{1cm} (2)

where $c$ and $r$ are the center and the radius of the ball end tool, respectively, where $S_i(u, v)$ is the $i$th surface in the model, and $(u^i, v^i)$ is a parametric location in the untrimmed domain of surface $S_i$.

If $O(S_i)$ designates the exact offset surface to surface $S_i$ at distance $r$, it is clear that the ball end tool could not gouge $S_i$ if $c$ were kept on $O(S_i)$. We define the manufacturing offset of a Boolean union operation of two surfaces, $S_i \cup S_j$, to be the union of the offset surfaces, that is $\tilde{O}(S_i \cup S_j) \equiv O(S_i) \cup O(S_j)$, even though the model might not be completely milled along the intersection curve of $S_i$ and $S_j$ in concave regions. Such a definition guarantees that the tool will gouge neither $S_i$ nor $S_j$. Similarly, the manufacturing offset of a Boolean intersection operation of two surfaces, $S_i \cap S_j$ is defined as the intersection of the offset surfaces, that is $\tilde{O}(S_i \cap S_j) \equiv O(S_i) \cap O(S_j)$. Since the Boolean intersection operation only "removes material", it is not possible for it to form concave corners from an intersection of two $C^1$ continuous surfaces, so the $\tilde{O}$ definition of an offset of a Boolean intersection operation supports the milling of the entire region along the intersection curves.

Since an offset of a single surface is another single surface [8], Boolean operations can be performed on the offset surfaces in much the same way they were computed for the original models. Consider surfaces $S_i$ and $S_j$ that intersect. If the intersection occurs near the boundary of either surface, it can happen that $O(S_i)$ does not intersect $O(S_j)$. For open surfaces, one solution that forms correct intersection curves is to extend them in the cross boundary tangent directions.

Figure 4: Parallel plane contouring is used to generate 3 axes pockets for rough cutting.

4 Rough Cutting Stage

The toolpath derived in section 2 cannot, in general, be directly applied to the stock from which the model is to be machined. In some cases, the depth of milling required is simply too large. A rough cutting stage is usually applied in which the excess material is removed crudely. Then, in the final stage, when the toolpath derived in section 2 is applied, it is necessary to remove only a limited amount of material.

One way to discard the excessive material using 3 axis milling is to slice the offset approximation of the model with several parallel planes and remove the material external to the part at each contour level. Two dimensional pocketing operations [13] can be used to remove the excessive material at each contoured layer. Figure 4 shows those contours of a "house on the hill" model. The rough cutting stage can be automated, similarly to the adaptive isocurve extraction algorithm.

5 Results

Several results are presented in this section, as are some timing considerations. The adaptive isocurve toolpaths for the knight in Figure 3c have been used to mill the complete knight. Two fixtures, one for the right side and one for the left side of the knight have been used. Plate 1 shows a raytraced version of the model while Plate 2 shows the milled piece. A ball end tool was driven along
Table 1: CPU times for adaptive iso-curves extraction.

<table>
<thead>
<tr>
<th>model</th>
<th>cpu time</th>
<th># curves</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2c - surface</td>
<td>5.6 sec.</td>
<td>61</td>
</tr>
<tr>
<td>Plate 2 - knight</td>
<td>54.3 sec.</td>
<td>122</td>
</tr>
<tr>
<td>Plate 4 - house</td>
<td>132.0 sec.</td>
<td>1013</td>
</tr>
</tbody>
</table>


Figure 5: Adaptive isocurves toolpath for the $\hat{O}$ offset of the "house on the hill" model.


