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: I.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.

*This work was supported in part by DARPA (N00014-92-J-4113) and the NSF and DARPA Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-89-20219). All opinions, findings, conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

†Appreciation is expressed to IBM for partial fellowship support of the first author.

‡Current address: Computer Science Department, Technion, Haifa 32000, Israel.

Definition 1 *A set of curves $\mathcal{C}$ in a given surface S is called a* valid coverage for S with respect to some constant δ *if, for any point p on S, there is a point q on one of the curves in $\mathcal{C}$, 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 δ such that any point on the surface is at most δ 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 q 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 is 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 $r - \sqrt{r^2 - (\frac{Da}{2})^2}$ 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.

Now consider the *optimality* of a valid toolpath.

Definition 2 *A toolpath for a given surface is considered* optimal *if it provides a valid coverage for S and its path length is 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 [14]. We will not address this issue in this paper.

There are two main approaches used to generate tool paths for freeform surfaces. In one, isoparametric curves are extracted from the surface, usually in equally spaced parametric steps [3, 6, 14, 16]. 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). This 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 system that invokes the toolpath generator to determine the parameter stepping tolerance that will create valid toolpaths to within a given δ, even if the top and bottom regions of the surface in Figure 2a are treated separately. Further, it is the shape of the represented geometry and the associated milling that is of interest so the parametric representation of the surface should be internal and not be shown in the effects of milling.

An alternative method for generating toolpaths is based on contouring planes, in which the surface is intersected by (frequently geometrically equally spaced) parallel planes. The resulting numerically derived intersection curves are used to drive the milling tools [2, 3]. Sometimes this curve is modified as it is evaluated to remove sections that might gouge along this plane [12] 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 orders of magnitude larger than isocurve data. For relatively flat surfaces, the contouring algorithm seems to yield acceptable results (see Figure 2b). However, as 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 invalidating the toolpath, as can be seen from Figure 3b.

Selection of the parallel plane spacing and the parallel plane direction to create a valid toolpath is not obvious, although work has been done in estimating plane spacing based on estimating a planar curvature in the direction orthogonal to that of the cutting plane at discrete points [12]. Even if an algorithm could be created to adaptively space a next cutting plane based on the coplanarity of one region of the surface with the the cutting plane direction, this spacing would be fixed over the entire next tool path, being much closer than might be necessary in another noncoplanar region. Local coplanarity in one region of the surface would dictate the spacing for the entire model.

Attempts to improve the techniques described above have been geared mainly toward local adaptation of the algorithm to specific regions which require a different number of samples to achieve the required tolerances [13, 16]. Adaptation of scanline rendering has been used by other researchers [19, 20] to obtain a piecewise linear approximation for the toolpath.

An adaptive sub-isocurve extraction approach is introduced for rendering in [9]. This scheme provides a more optimal and valid coverage of the surface by adaptively introducing partial sub-isocurves in regions void of existing sub-isocurves (definition 1). Furthermore, the algorithm frees the user from the need to determine either the surface parameter spacing or contour 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 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 *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 exploits this approach for rendering (see Figures 2c and 3c). Although the adaptive isocurve extraction algorithm is developed for rendering in [9] it will be briefly discussed here.

It is appealing to use isocurves since the representation is compact, exact, and 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 piecewise linear approximation allows. Furthermore, isocurves can 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. Scallops 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, 13] 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, 13]. A different approach [13] 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 Alpha_1 solid modeler.

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 $S(u,v)$, $C_1(u) = S(u, v_1)$, $u \in [u_1^s, u_1^e]$ and $C_2(u) = S(u, v_2)$, $u \in [u_2^s, u_2^e]$, $v_1 \le v_2$, from a given set $\mathcal{C}$ of isocurves forming a valid coverage for S are considered* adjacent *if, along their common domain $\mathcal{U} = [u_1^s, u_1^e] \cap [u_2^s, u_2^e]$, there is no other isocurve from $\mathcal{C}$ between them. That is, there does not exist $C_3(u) = S(u, v_3) \in \mathcal{C}$, $u \in [u_3^s, u_3^e]$ such that $v_1 \le v_3 \le v_2$ and $[u_3^s, u_3^e] \cap \mathcal{U} \ne \emptyset$.*

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

$$\begin{aligned} \Delta_{12}(u) &= \|C_1(u) - C_2(u)\|_2 \\ &= \sqrt{(c_1^x(u) - c_2^x(u))^2 + (c_1^y(u) - c_2^y(u))^2 + (c_1^z(u) - c_2^z(u))^2}. \end{aligned} \quad (1)$$

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 δ, it is possible to compute the parameter values where the iso-distance between $C_1(u)$ and $C_2(u)$ is exactly δ by computing the zero set of $(\Delta_{12}^2(u) - \delta^2)$ [15]. By subdividing the two curves at these parameters, new sub-isocurve pairs, $\{C_1^i(u), C_2^i(u)\}$, are formed with the characteristic that each pair is always iso-distance smaller or always larger than δ, in their open interval domains. If the two curves in the pair $\{C_1^i(u), C_2^i(u)\}$ are closer than δ 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 δ, a new sub-isocurve, $C_{12}^i(u)$, is introduced between $C_1^i(u)$ and $C_2^i(u)$ along their common domain $\mathcal{U}$ and the same iso-distance computation is recursively invoked for the two new pairs $\{C_1^i(u), C_{12}^i(u)\}$ and $\{C_{12}^i(u), C_2^i(u)\}$.

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 δ by verifying that their iso-distance is not larger than δ. Because δ is small, there will be only a small variation in the speed in the v direction of the surface S, $\frac{\partial S}{\partial v}$. Since a middle isocurve is introduced iff the iso-distance is larger than δ and δ is small, resulting iso-distances between adjacent isocurves, as computed, are rarely significantly less than $\frac{\delta}{2}$.

Algorithm 1: Adaptive isocurve extraction. Iso-v curves are assumed.

Input:
    Surface $S(u, v)$.
    Iso-distance tolerance δ.
Output:
    Adaptive isocurve toolpath for $S(u, v)$.
Algorithm:
    AdapIsoCurve($S(u, v)$, δ)
    begin
        $C_1(u)$, $C_2(u)$ $\Leftarrow$ $S(u, v)$ two u boundary curves.
        return AdapIsoCurveAux($S(u, v)$, δ, $\{C_1(u), C_2(u)\}$).
    end
    AdapIsoCurveAux($S(u, v)$, δ, $\{C_1(u), C_2(u)\}$)
    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
                $C_{12}(u)$ $\Leftarrow$ Middle isocurve between $C_1(u)$ and $C_2(u)$.
                return AdapIsoCurveAux($S(u, v)$, δ, $\{C_1(u), C_{12}(u)\}$) $\bigcup$
                       AdapIsoCurveAux($S(u, v)$, δ, $\{C_{12}(u), C_2(u)\}$).
            end
        else
            begin
                $\{C_1^i(u), C_2^i(u)\}$ $\Leftarrow$ subdivided $\{C_1(u)$, $C_2(u)\}$ at all u
                        such that $\Delta_{12}^2(u) = \delta^2$.
                return $\bigcup_i$ AdapIsoCurveAux($S(u, v)$, δ, $\{C_1^i(u), C_2^i(u)\}$).
            end
    end

Furthermore, since the resulting set of isocurves covers the entire surface S, it can serve as a valid toolpath for S with maximum tolerance distance δ.

Algorithm 1, the adaptive isocurve extraction algorithm, generates a valid and more optimal coverage by minimizing the cutting speed motion required. This is accomplished 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. 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 the ball end tool on the offset surface, keeps the tool tangent to the original surface so it cannot gouge.

The adaptive isocurve toolpath is computed for each surface of the original model. The toolpath is not trimmed yet even if the surfaces are trimmed surfaces. Instead, an offset model is computed by offsetting all the surfaces of the model an amount that is equal to the tool radius and retrimming the surfaces in their new offset position. The adaptive isocurve toolpath is then mapped, using the same parametric domain values, to the offset surfaces and only then is the toolpath properly trimmed, creating the final toolpath. The result is a gouge free toolpath, even in concave corners of the model.

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 $O(\mathcal{A})$ denote the exact offset of $\mathcal{A}$. It is unfortunate, but $O(\mathcal{A} \cap \mathcal{B})$ is not always the same as $O(\mathcal{A}) \cap O(\mathcal{B})$. For example, $\mathcal{A} \cap \mathcal{B}$ and hence $O(\mathcal{A} \cap \mathcal{B})$ could be empty but $O(\mathcal{A}) \cap O(\mathcal{B})$ might be nonempty.

Several types of manufacturing offsets can be defined for piecewise C^1 models [17, 18] that are generated via 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 the center of the ball end tool would 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,

$$\|c - S_i(u^i, v^i)\|_2 \geq r, \qquad \forall i, \tag{2}$$

where c and r are the center and the radius of the ball end tool, respectively, $S_i(u, v)$ is the ith 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, $S_i \bigcup S_j$, to be the union of the offset surfaces, that is $\widehat{O}(S_i \bigcup S_j) \equiv O(S_i) \bigcup O(S_j)$, 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, $S_i \bigcap S_j$ is defined as the intersection of the offset surfaces, that is $\widehat{O}(S_i \bigcap S_j) \equiv O(S_i) \bigcap O(S_j)$. 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. Thus the $\widehat{O}$ 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.

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 [14] 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, similar 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 an offset [8] of the knight surface in 3 axis milling mode. The knight model consists of a single highly complex NURBs surface.

The adaptive isocurve toolpath algorithm produces only isoparametric curves that are simple to clip against the surface trimming curves defining a trimmed surface.

A "house on a hill" model, consisting of several trimmed surfaces was used for this example. This model was milled using a ball end tool in 3 axis mode. Plate 3 shows a raytraced version of the model, while Figure 5 shows the adaptive isocurve toolpath used in the finish stage of the model in Plate 4. The offset of the model was automatically computed using the the $\widehat{O}$ offset method described in section 3. Furthermore, it was unnecessary to introduce any auxiliary check or driver surfaces [6] as part of this automated toolpath generation process.

To gain some insight regarding this algorithm, Table 1 provides some timing results for computing the adaptive isocurve toolpaths for the tests displayed. Tests were run on a SGI4D 240 GTX (R3000 25MHz Risc machine). The surface in Figure 2 is a B-spline ruled surface with 3 Bézier patches (patches of a NURBs surface are enumerated as the number of Bézier patches that would result from subdividing the NURBs surface at each original interior knot). The knight is a far more complex NURBs surface. Its 56 Bézier patches account for its long processing time. Although the "house on the hill" model is defined by seven NURBs surfaces, none of them is as complex as the single surface defining the knight.

6 Conclusion

The adaptive isocurve extraction method for freeform surfaces introduced in [9] was used as the foundation for the nearly optimal toolpath generation method described. This algorithm eliminates most of the redundancy that occurs when equally spaced complete isocurves are used as a toolpath, while retaining all the advantageous properties of isocurve toolpaths. The toolpath is exact, easy to trim, and now almost optimal.

The use of this algorithm in three-axis milling has been demonstrated. The adaptive isocurve extraction algorithm can be used to generate toolpaths for 4 or 5 axis milling, virtually unmodified. Multi-axis milling operations require additional information to orient the tool. Normal information was propagated with the adaptive isocurve toolpath to orient the tool in 5-axis flat end milling mode, in a similar way to the normal computation approach taken in the rendering application of the adaptive isocurve algorithm [9]

The surface normal information can be represented as a normal isocurve accompanying the position isocurve [10] and should be included when 4 or 5 axis milling is needed. Unfortunately, using 4 or 5 axis milling operations introduces difficulties in detecting gouging and accessibility of locations. These areas are under current research.

References

[1] **S. Aomura and T. Uehara.** Self-Intersection of an Offset Surface. *Computer Aided Design*, Vol. 22, No. 7, pp 417-422, September 1990

[2] **J. E. Bobrow.** NC Machine Tool Path Generation From CSG Part Representations. *Computer Aided Design*, Vol. 17, No. 2, pp 69-96, March 1985.

[3] **B. K. Choi and C. S. Jun.** Ball-End Cutter Interference Avoidance In NC Machining of Sculptured Surfaces. *Computer Aided Design*, Vol. 21, No. 6, pp 371-378, July/August 1989.

[4] **B. Cobb.** Design of Sculptured Surfaces Using The B-spline representation. *Ph.D. thesis*, University of Utah, Computer Science Department, June 1984.

[5] **R. T. Farouki.** The Approximation of Non-Degenerate Offset Surfaces *Computer Aided Geometric Design*, Vol. 3, No. 1, pp 15-43, May 1986.

[6] **I. D. Faux and M. J. Pratt.** *Computational Geometry for Design and Manufacturing.* John Wiley & Sons, 1979.

[7] **G. Elber and E. Cohen.** Second Order Surface Analysis Using Hybrid Symbolic and Numeric Operators. *ACM Transactions on Graphics*, Vol. 12, No. 2, pp 160-178, April 1993.

[8] **G. Elber and E. Cohen.** Error Bounded Variable Distance Offset Operator for Free Form Curves and Surfaces. *International Journal of Computational Geometry and Applications*, Vol. 1., No. 1, pp 67-78, March 1991.

[9] **G. Elber and E. Cohen.** Adaptive Isocurves Based Rendering for Freeform Surfaces. Submitted for publication.

[10] **G. Elber.** Free Form Surface Analysis using a Hybrid of Symbolic and Numeric Computation. *Ph.D. thesis*, University of Utah, Computer Science Department, 1992.

[11] **C. M. Hoffmann.** *Geometric & Solid Modeling, An Introduction.* Morgan Kaufmann Publisher, Inc..

[12] **Y. Huang and J. H. Oliver.** Non-constant Parameter NC Tool Path Generation on Sculptured Surfaces. *ASME Computers in Engineering*, v.1, August 1992, pp. 411-419.

[13] **R. B. Jerard, J. M. Angleton and R. L. Drysdale.** Sculptured Surface Tool Path Generation with Global Interference Checking. *Design Productivity Conference*, Feb. 6-8, 1991, Honolulu, Hawaii.

[14] **J .J. Chou.** Numerical Control Milling Machine Toolpath Generation for Regions Bounded by Free Form Curves and Surfaces. *Ph.D. thesis*, University of Utah, Computer Science Department, June 1989.

[15] **J. M. Lane and R. F. Riesenfeld.** Bounds on a Polynomial *BIT* 21 (1981), 112-117.

[16] **G. Loney and T. Ozsoy.** NC Machining of Free Form Surfaces. *Computer Aided Design*, Vol. 19, No. 2, pp 85-90, March 1987.

[17] **A. A. G. Requicha.** Toward a Theory of Geometric Tolerancing. *International Journal of Robotics Research*, Vol. 2, No. 4, pp. 45-49, Winter 1983.

[18] **J. R. Rossignac and A. A. G. Requicha.** Constant Radius blending In Solid Modeling. *Computers in Mechanical Engineering*, pp 65-73, July 1984.

[19] **S. W. Thomas.** Scanline Rendering for 3-Axis NC Toolpath Generation, Simulation, and Verification. Dept. of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2122, *Technical Report CSE-TR-43-90*, January 1990.

[20] **D. Zhang and A. Bowyer.** CSG Set-Theoretical Solid Modelling and NC Machining of Blend Surfaces. *The Second Computation Geometry Conference*, ACM 1986.

Table 1: CPU times for adaptive iso-curves extraction.

model	cpu time	# curves.
Figure 2c - surface	5.6 sec.	61
Plate 2 - knight	54.3 sec.	122
Plate 4 - house	132.0 sec.	1013

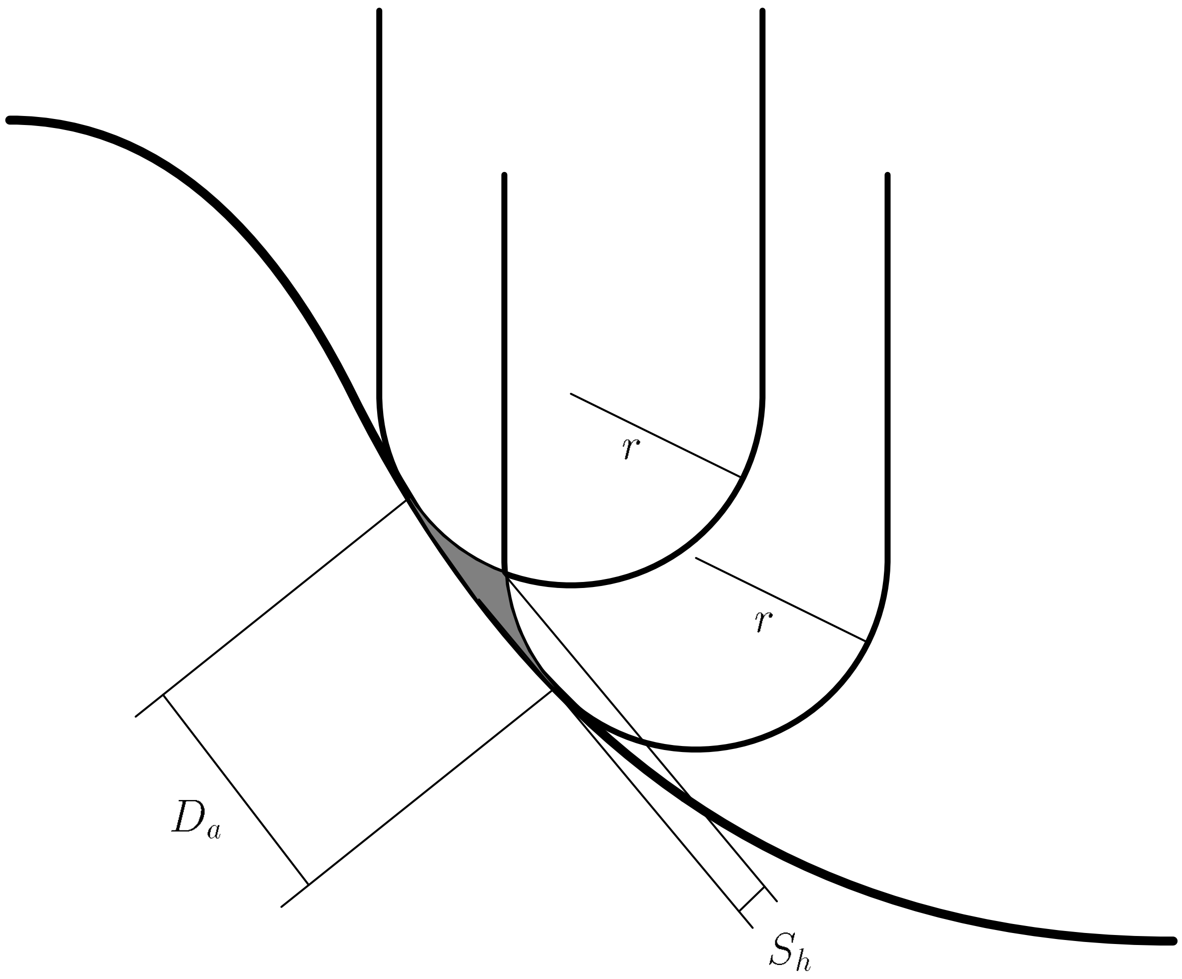


Figure 1: Given the tool radius, S_h can easily be derived from D_a.

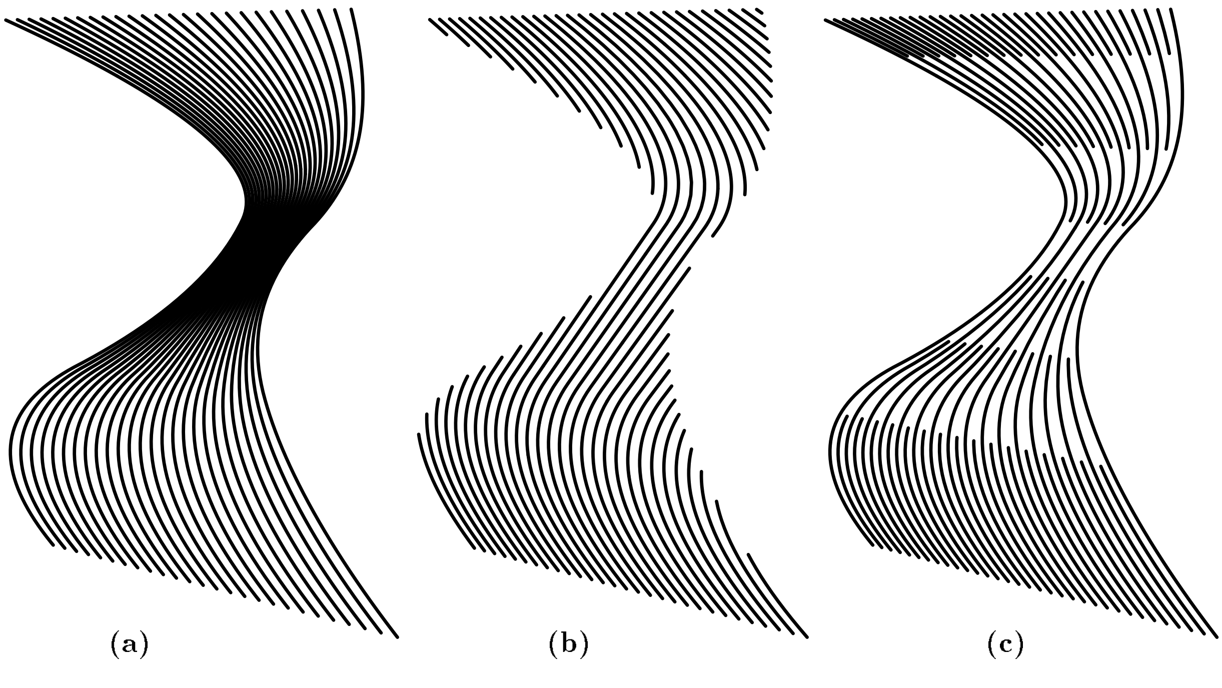


Figure 2: Isocurves are obviously not an optimal solution as a toolpath for this surface (a). Contouring with equally spaced parallel planes is more optimal but is piecewise linear (b). Adaptive isocurves are, in general, more optimal, exact, and compact (c).

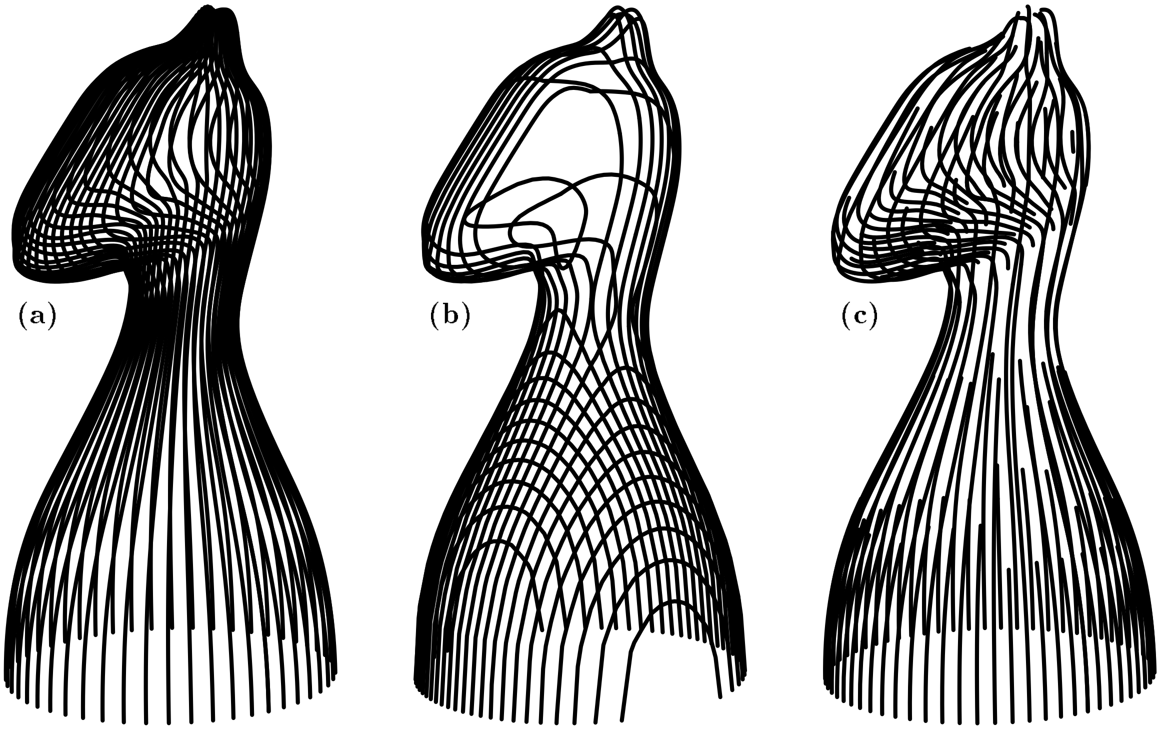


Figure 3: Toolpath using isocurves will be not optimal in this complex surface (a). Contouring with equally spaced parallel planes is too sparse in coplanar regions (b). Adaptive isocurves are more optimal, exact, and still correctly spans the entire surface (c).

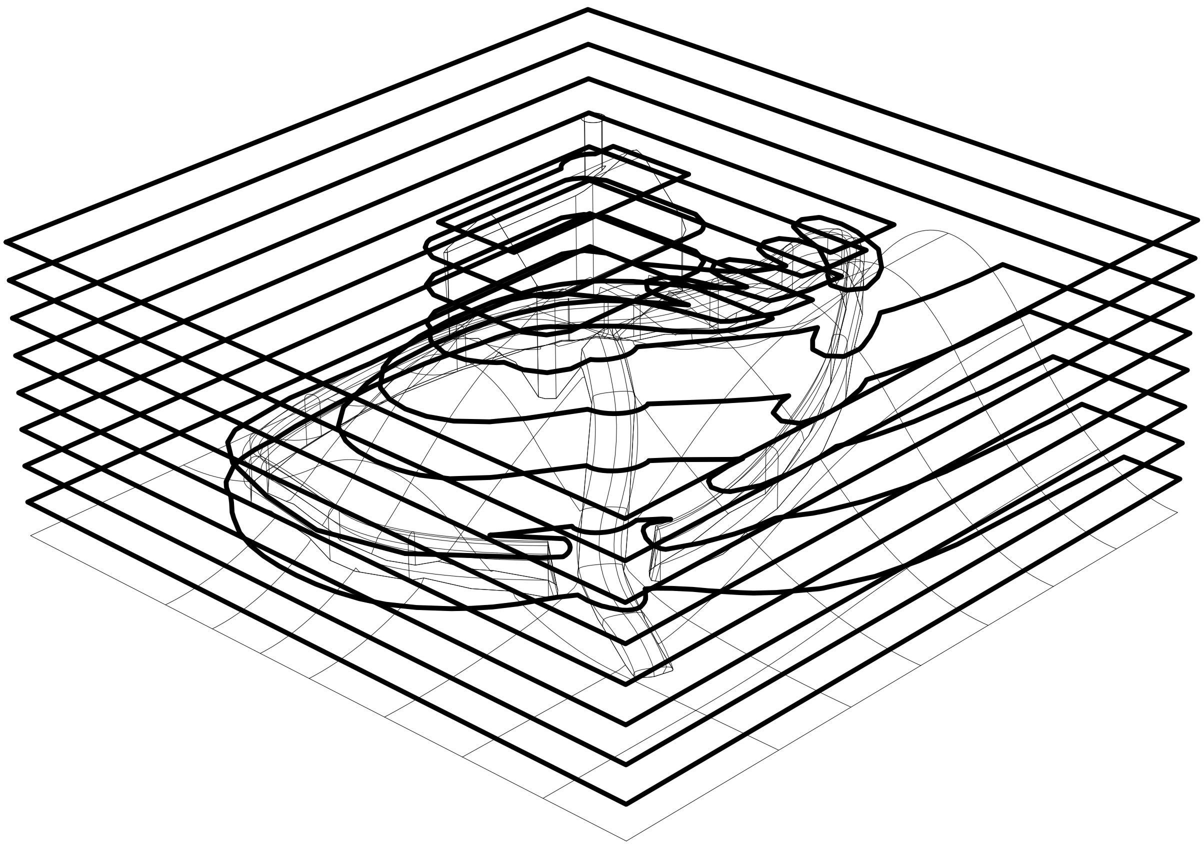


Figure 4: Parallel plane contouring is used to generate 3 axes pockets for rough cutting.

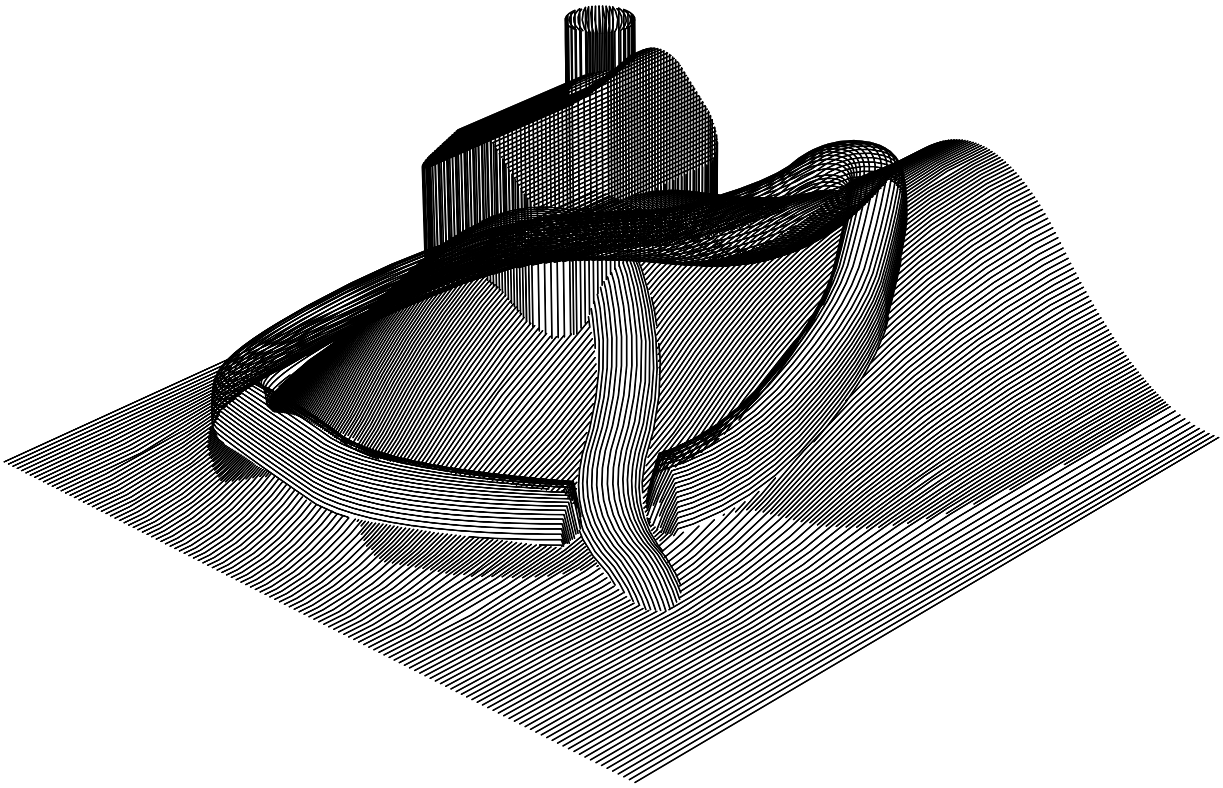


Figure 5: Adaptive isocurves toolpath for the $\hat{O}$ offset of the "house on the hill" model.

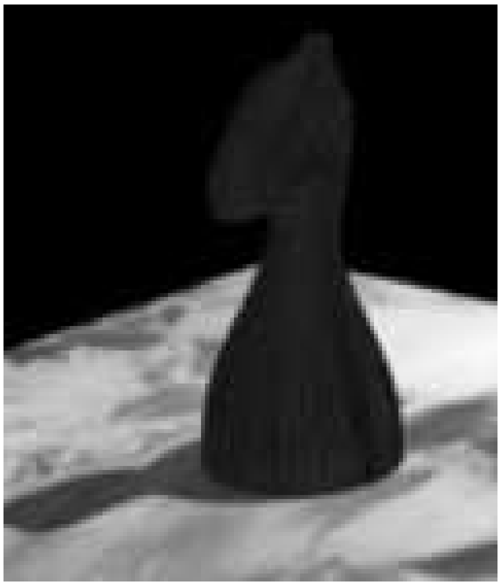
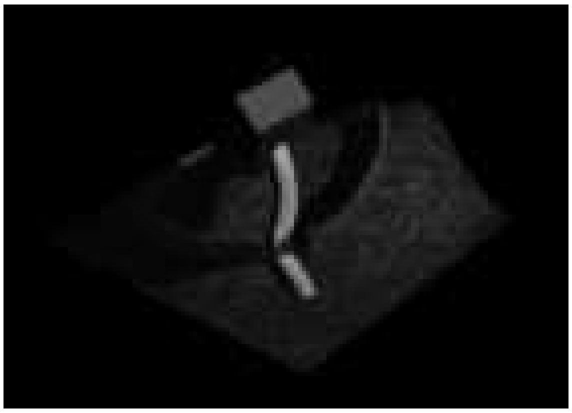


Plate 1

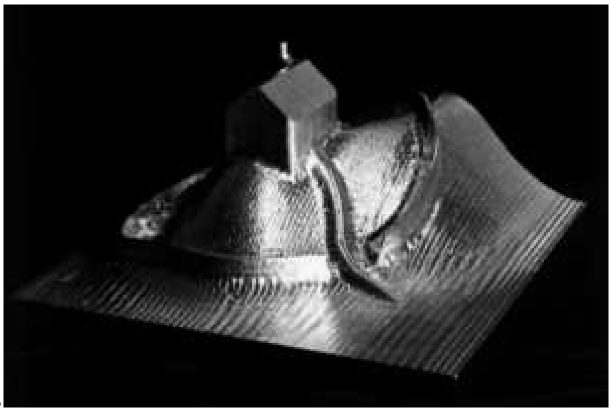
Raytraced image of the knight model.

Plate 2

Aluminum milled version of the knight model.

Plate 

Raytraced image of the "house on the hill" model.

Plat 

Aluminum milled version of the "house on the hill" model.