CHAPTER 5

MACHINING APPLICATIONS

Computing and numerical control (NC) has made great progress at that time, and it was certain that only numbers, transmitted from drawing office to tool drawing office, manufacture, patternshop, and inspection, could provide an answer; of course, drawings would remain necessary, but they would only be explanatory, their accuracy having no importance. Numbers would be the only and final definition.

P. Bézier (on NC capabilities in the 1960s)

5.1 Introduction

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. This Chapter addresses some of the relevant issues in this field of realizing computer models.

In section 5.2, an algorithm developed to adaptively extract isocurves for rendering [26] 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. Section 5.2.1

develops and defines this new algorithm. Sections 5.2.2 and section 5.2.3 deal with some practical problems while section 5.2.4 provides some examples and results. Finally, in section 5.3, a whole new approach to realizing computer models is derived using piecewise ruled and developable surface approximations.

5.2 Adaptive Isocurves Toolpath

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 5.1** *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 5.1 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 5.1 takes into consideration only the distance between an arbitrary point p on the surface and the closest point on the toolpath. Other criteria, such as bounding the curvature, could be added to the definition of *validity* of a toolpath to provide a tighter bound on the resulting scallop height without affecting any of the rest of the algorithm.

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

> **Definition 5.2** *A toolpath for a given surface is considered* optimal *if it is valid and if its path length is minimal.*

Definition 5.2 considers optimality based only on the cutting motion part of the toolpath. Tool retraction and traversals are not considered as optimality conditions in this Chapter. 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 [12].

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 [10, 12, 32, 46]. These isocurves usually span the entire parametric domain of the surface (see Figures 5.1a and 5.2a) and will be referred to as *complete-isocurves*. Isocurves that span only a portion of the surface parametric domain (see Figures 5.1b and 5.2b) will be referred to as *subisocurves*. Although simple to determine, toolpaths created using complete isocurves equally spaced in parametric space, are clearly not optimal according to definition 5.2 and are redundant, as can be seen in the example of Figure 5.1a, 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 5.1a) 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 5.1a). Further, it might be difficult for the user 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 5.1a 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.

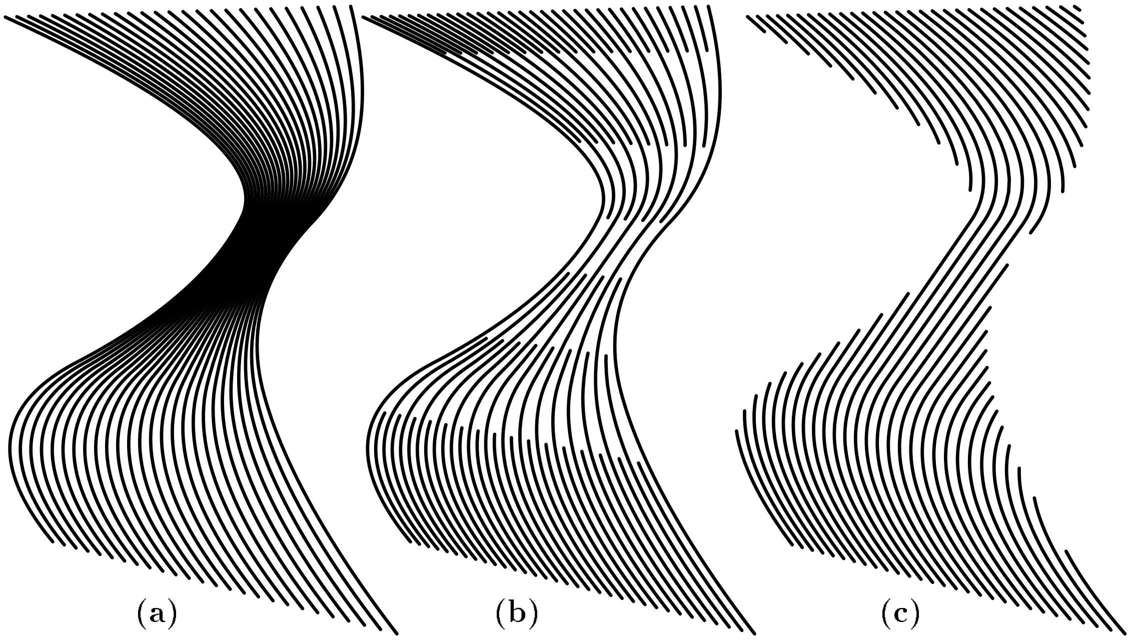

Figure 5.1. Isocurves are obviously not an optimal solution as a toolpath for this surface (a). Adaptive isocurves are, in general, more optimal, exact, and compact (b). Contouring with equally spaced parallel planes might be optimal but is piecewise linear (c).

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 [8, 10]. 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 magnitudes larger than isocurve data. For relatively flat surfaces the contouring algorithm seems to yield acceptable results (see Figure 5.1c). 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 5.2c, 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

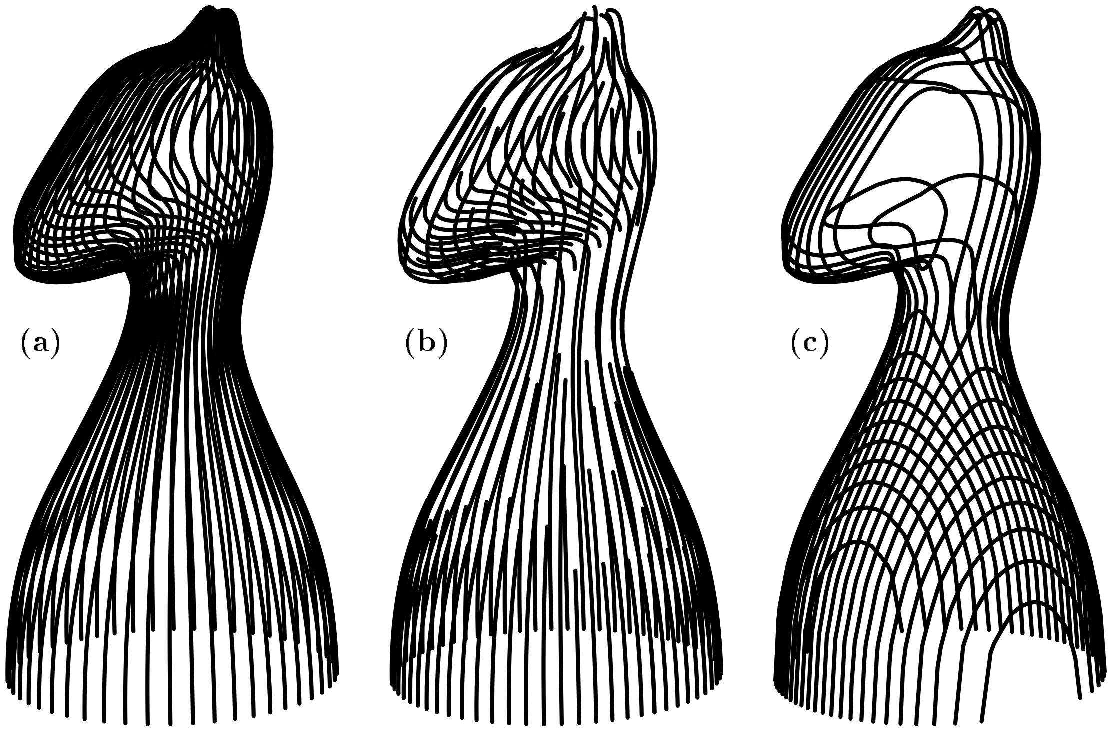

Figure 5.2. Toolpath using isocurves will be not optimal in this complex surface (a). Adaptive isocurves are more optimal, exact, and still correctly spans the entire surface (b). Contouring with equally spaced parallel planes is too sparse in coplanar regions (c).

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 [40, 46]. Others used adaptation of scanline fashion rendering [64, 67] to get a piecewise linear approximation for the toolpath.

An adaptive subisocurve extraction approach is introduced for rendering in [26]. That scheme provides a more optimal and valid coverage of the surface by adaptively introducing partial subisocurves in regions yet uncovered by already created subisocurves (definition 5.1). Furthermore, the algorithm frees the user from the need to determine both the surface parameter spacing or contouring plane spacing, and the direction to use to insure adjacent isocurve distances to produce a valid coverage. Instead, a bound on the required distance between adjacent subisocurves can be directly specified, and guaranteed automatically.

It is clear that a valid coverage generated using complete isocurves can be very inefficient (see Figure 5.1a), 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 does exactly that for rendering (see Figures 5.1b and 5.2b). because the adaptive isocurve extraction algorithm is developed for rendering in [26] it will be briefly discussed here. The interested reader can also refer to [26].

It is appealing to use isocurves because their representations are compact, exact, and they are straightforward to use as milling toolpath. 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. Scallops resulting from isocurve based toolpaths are usually more attractive than those resulting from contoured based toolpaths because they follow the model's basic streamlines. 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 representation can be arbitrary.

Recent literature [8, 10, 40] has suggested that the contact point numerical improvement approach, such as used by APT [32], is unstable and slow. Computations of a toolpath for a single surface are usually measured in minutes [10, 40]. A different known approach [40] 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 5.2.1 briefly discusses the adaptive subisocurve algorithm. Section 5.2.2 describes the offset computation required for ball end tool milling, and section 5.2.3 deals with the method used for the rough cutting process. Finally, section 5.2.4 presents some results obtained from an implementation of the new algorithm for NURBs based models using the Alpha_1 solid modeler.

5.2.1 Adaptive Isocurves Algorithm

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

Definition 5.3 *Two (sub)isocurves of surface* $S(u,v)$, $C_1(u) = S(u,v_1)$, $u \in [u_1^s, u_1^e]$ *and* $C_2(u) = S(u,v_2)$, $u \in [u_2^s, u_2^e]$, $v_1 \leq v_2$, *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* $v_1 \leq v_3 \leq v_2$ *and* $[u_3^s, u_3^e] \cap \mathcal{U} \neq \emptyset$.

Definition 5.4 *The* iso-distance *function* $\Delta_{12}(u)$ *between two adjacent (sub) isocurves along their common domain* $\mathcal{U}$ *is equal to*

$$\begin{aligned} \Delta_{12}(u) &= \|C_1(u) - C_2(u)\|_2 \\ &= \sqrt{(c_1^x(u) - c_2^x(u))^2 + (c_1^y(u) - c_2^y(u))^2 + (c_1^z(u) - c_2^z(u))^2}. \end{aligned} \tag{5.1}$$

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. 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. 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 $(\Delta_{12}^2(u) - \delta^2)$ [44]. By subdividing the two curves at these parameters, new subisocurve 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 subisocurve, $C_{12}^i(u)$, is introduced between $C_1^i(t)$ and $C_{12}^i(t)$ 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 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 less than $\frac{\delta}{2}$. Furthermore, because the resulting set of isocurves covers the entire surface S, it can serve as a valid toolpath for S with distance δ.

Algorithm 5.1, 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

Algorithm 5.1

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
```

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 [24]) could be added to the definition of validity to decide whether to introduce a middle isocurve in algorithm 5.1 and obtain a tighter bound on the scallop height.

5.2.2 The Offset Computation

Because 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 because keeping the center of the ball end tool on the offset surface, keeps the tool tangent to the original surface so it can not gauge.

Unfortunately, the exact offset of a freeform piecewise polynomial or rational surface is not representable, in general, as a piecewise polynomial or rational

Algorithm 5.1 continued

```
AdapIsoCurveAux( S(u,v), δ, {C_1(u), C_2(u)} )
begin
  Δ²_12(u) ⇐ ||C_1(u) − C_2(u)||_2, iso-distance between C_1(u) and C_2(u).
  if (Δ²_12(u) < δ², ∀u) then
    return ∅.
  else if (Δ²_12(u) > δ², ∀u) then
    begin
      C_12(u) ⇐ Middle isocurve between C_1(u) and C_2(u).
      return  AdapIsoCurveAux( S(u,v), δ, {C_1(u), C_12(u)} ) ∪
              AdapIsoCurveAux( S(u,v), δ, {C_12(u), C_2(u)} ).
    end
  else
    begin
      {C^i_1(u), C^i_2(u)} ⇐ subdivided {C_1(u), C_2(u)} at all u
                             such that Δ²_12(u) = δ².
      return ∪_i AdapIsoCurveAux( S(u,v), δ, {C^i_1(u), C^i_2(u)} ).
    end
end
```

surface [25]. Quite a few methods have been developed in recent years to provide approximations to surface offsets [2, 13, 25, 29]. In [25], 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 [34] 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 a nonempty.

Several types of manufacturing offsets can be defined for piecewise C^1 models [54, 57] 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,

$$\|T_c - S_i(u^i, v^i)\|_2 \geq T_r, \qquad \forall i, \tag{5.2}$$

where T_c and T_r are the center and the radius of the ball end tool, respectively, where $S_i(u, v)$ 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 T_r, it is clear that the ball end tool could not gouge S_i if T_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 $\widehat{O}(S_i \bigcap S_j) \equiv O(S_i) \bigcap O(S_j)$. Because 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 $\widehat{O}$ definition of an offset of a Boolean intersection operation supports the milling of the entire region along the intersection curves.

Because an offset of a single surface is another single surface [25], 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.

5.2.3 Rough Cutting Stage

The toolpath derived in section 5.2.1 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 excessive material is removed crudely. Then, in the final stage, when the toolpath derived in section 5.2.1 is applied, it is necessary to remove only a limited amount of material.

One way to discard the excessive material, in 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 [12] can be used to remove the excessive material at each contoured layer. Figure 5.3 shows those contours of a “house on the hill” model. The rough cutting stage can be automated, similarly to the adaptive isocurve extraction algorithm.

5.2.4 Results

Several results are presented in this section, as are some timing considerations. The adaptive isocurve toolpaths for the knight in Figure 5.2b 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. Figure 5.4 shows a raytraced version of the model while Figure 5.5 shows the milled piece. A ball end tool was driven along an

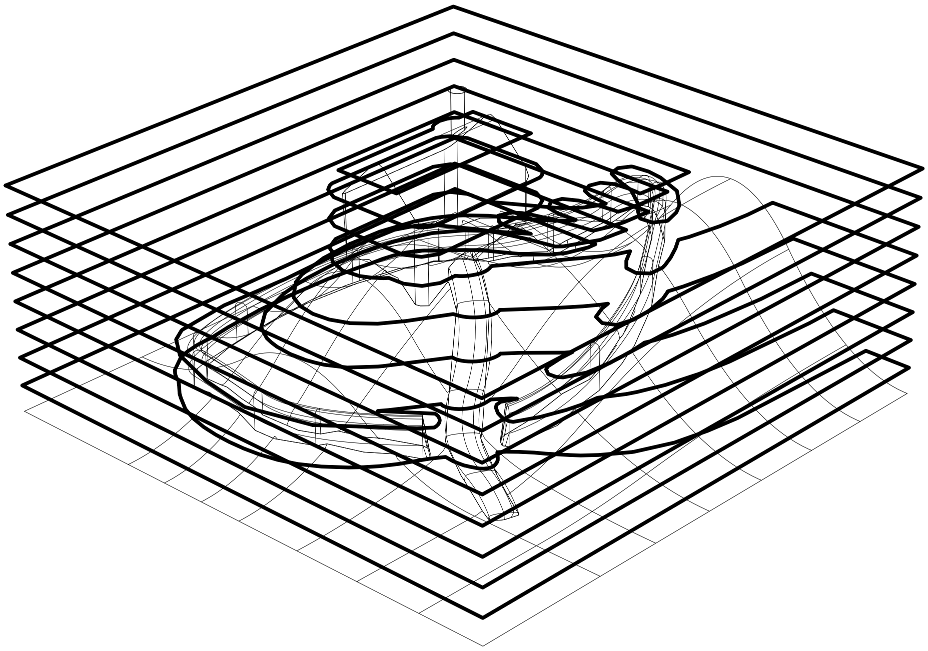

Figure 5.3. Parallel plane contouring is used to generate pockets for rough cutting.

offset [25] of the knight surface in 3 axis milling mode. The knight model consists of a single highly complex NURBs surface.

This 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. Figure 5.6 shows a raytraced version of the model, while Figure 5.7 shows the adaptive isocurve toolpath used in the finish stage of the model in Figure 5.8. The offset of the model was automatically computed using the the $\widehat{O}$ offset method described in section 5.2.2. Furthermore, it was unnecessary to introduce any auxiliary check or driver surfaces [32] as part of this automated toolpath generation process.

To gain some insight regarding this algorithm, Table 5.1 provides some timing results for computing the adaptive isocurve toolpaths for the tests displayed. Tests were running on a SGI4D 240 GTX (R3000 25MHz Risc machine). The surface in Figure 5.1 is a B-spline ruled surface with 3 Bézier patches (patches of a NURBs

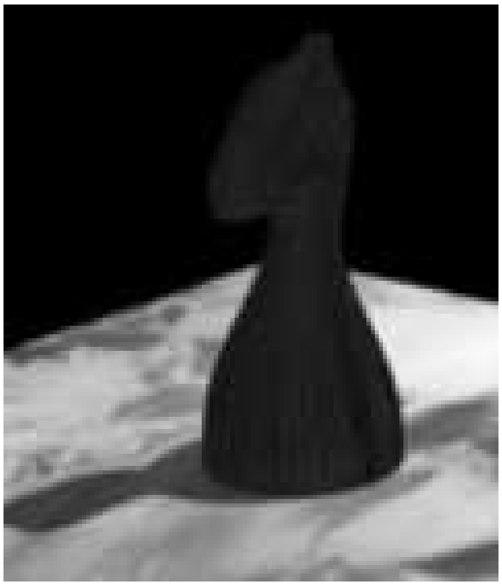

Figure 5.4. Raytraced image of the knight model.

Table 5.1. CPU times for adaptive iso-curves extraction.

model	cpu time	# isocurves.
Figure 5.1b - surface	5.6 sec.	61
Figure 5.5 - knight	54.3 sec.	122
Figure 5.7 - "house on a hill"	132.0 sec.	1013

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 accounts for its long processing time. Although the "house on the hill" model has 7 NURBs surfaces in it, none of them is as complex as the single surface defining a knight.

5.3 Fabrication Using Layout Projection

It is common to find freeform surfaces manually approximated and assembled as sets of piecewise developable surfaces. "Developable surfaces are of considerable importance to sheet-metal- or plate-metal-based industries and to a less extent to fabric-based industries" [56]. Parts of aircrafts and ships are assembled from

Figure 5.5. Aluminum milled version of the knight model.

piecewise planar sheets unidirectionally bent into their model positions. Certain fabric and leather objects are made using patterns made from planar sheets.

Because developable surfaces can be unrolled onto a plane without distortion, they can be cut from planar sheets, bent back into their final position, and stitched together.

In [5], a flattening approximation is computed for freeform surfaces to eliminate the distortion in texture mapping. Surfaces are split into patches along feature (geodesic) lines and approximated as flats. However, we are mainly interested in isometric projections that preserves intrinsic distances and angles [21]. Physically, such maps only bend the surface with no stretching, tearing, or distortion. One of the most interesting properties of developable surfaces is their ability to be laid flat on a plane without distortion by simply unrolling them [21, 32]. Therefore, we would like to generate a surface approximation using piecewise developable surfaces [21, 32], for which an isometric map to a plane exists.

Currently, the process that determines how and where to decompose the model requires human ingenuity and does not provide a bound on the accuracy of the

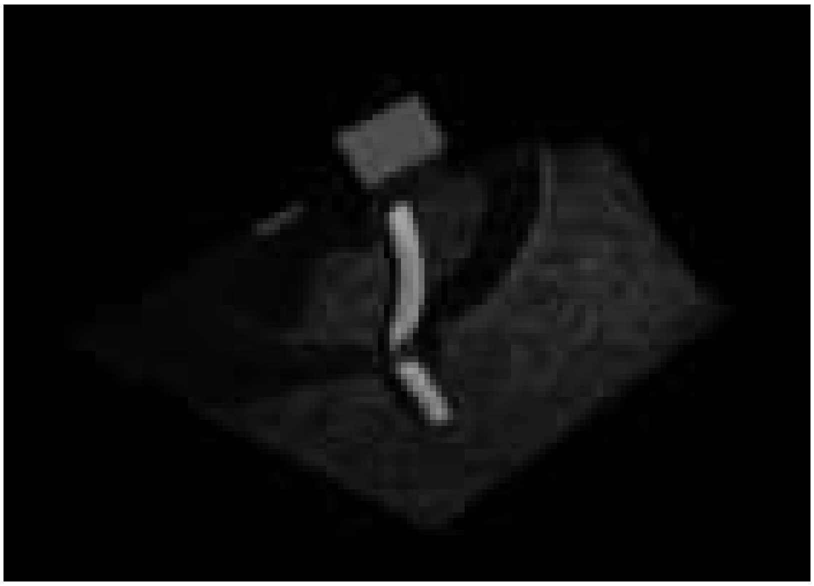

Figure 5.6. Raytraced image of the "house on the hill" model.

approximation. We explores a technique for automatically decomposing the sculptured model, using a C^0 approximation with error bound control, into sets of developable surfaces.

The Gaussian curvature of a developable surface $S(u, v)$, K, is zero everywhere [21, 32], i.e., $K(u, v) \equiv 0$. The class of developable surfaces is difficult to deal with, so we will first concentrate on a superset of it, namely the class of ruled surfaces. In order to be able to use ruled surfaces instead, we need to derive the conditions in which a ruled surface is also developable. Let $|G|$ and $|L|$ be the determinants of the first and second fundamental form [21], respectively.

Lemma 5.1 *Let R be a regular ruled surface, $R(u, v) = C_1(u) * v + C_2(u) * (1 - v)$, $v \in (0, 1)$. R is developable if and only if $\left\langle n_r, \frac{\partial^2 R}{\partial u \partial v} \right\rangle \equiv 0$,*

Proof: Given a regular surface S, its Gaussian curvature, K, is zero everywhere (therefore, it is developable) if $|L| \equiv 0$ because $K = \frac{|L|}{|G|}$, and $|G| \neq 0$ for regular surfaces.

$$|L| = \left\langle n, \frac{\partial^2 S}{\partial u^2} \right\rangle \left\langle n, \frac{\partial^2 S}{\partial v^2} \right\rangle - \left\langle n, \frac{\partial^2 S}{\partial u \partial v} \right\rangle^2 . \tag{5.3}$$

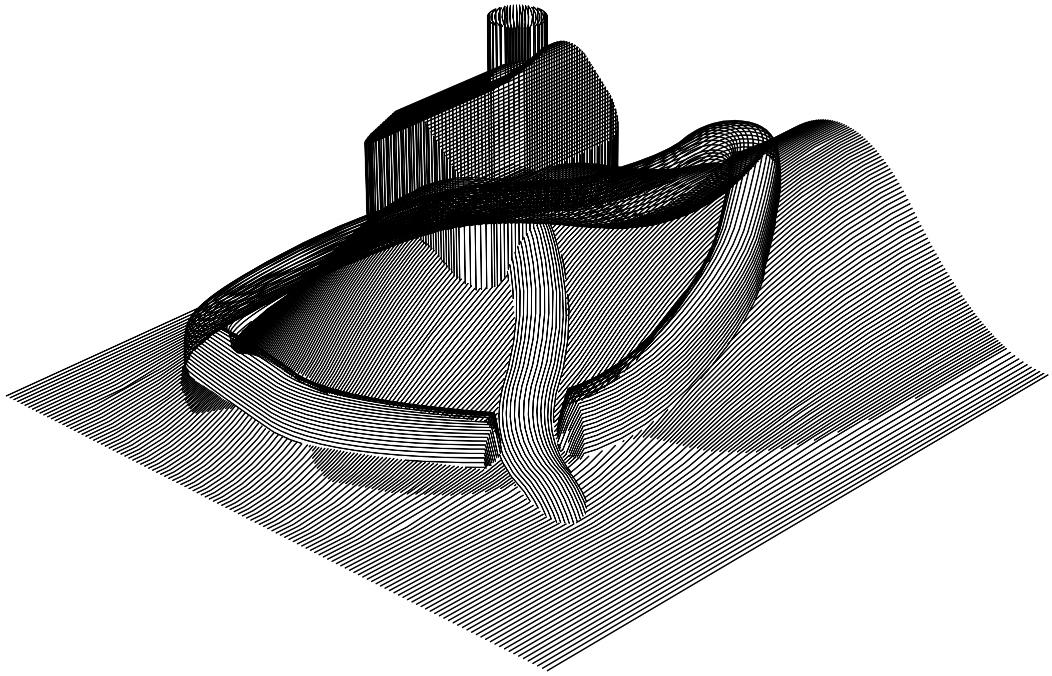

Figure 5.7. Adaptive isocurves toolpath for $\hat{O}$ offset of "house on the hill" model.

By differentiating R twice in v, it is clear that $\frac{\partial^2 R}{\partial v^2} \equiv 0$. We can immediately rewrite $|L|$ as

$$|L_R| = -\left\langle n_r, \frac{\partial^2 R}{\partial u \partial v} \right\rangle^2, \tag{5.4}$$

and the result follows. ■

Therefore, to determine if a ruled surface is developable, one can symbolically compute $\sigma(u, v) = \left\langle n_r, \frac{\partial^2 R}{\partial u \partial v} \right\rangle$ (That is, represent the scalar surface $\sigma(u, v)$ as a Bézier or NURBs scalar surface) and make sure it is zero everywhere within a prescribed tolerance. In other words, using the convex hull property of the Bézier and NURBs representations, all the coefficients of the scalar surface $\sigma(u, v)$ must be zero within a prescribed tolerance.

The mixed partials, also called the twist of the surface [3], are a measure of the "crosstalk" in the parameterization. Equation 5.4 measures this "crosstalk" projected in the direction of the surface normal.

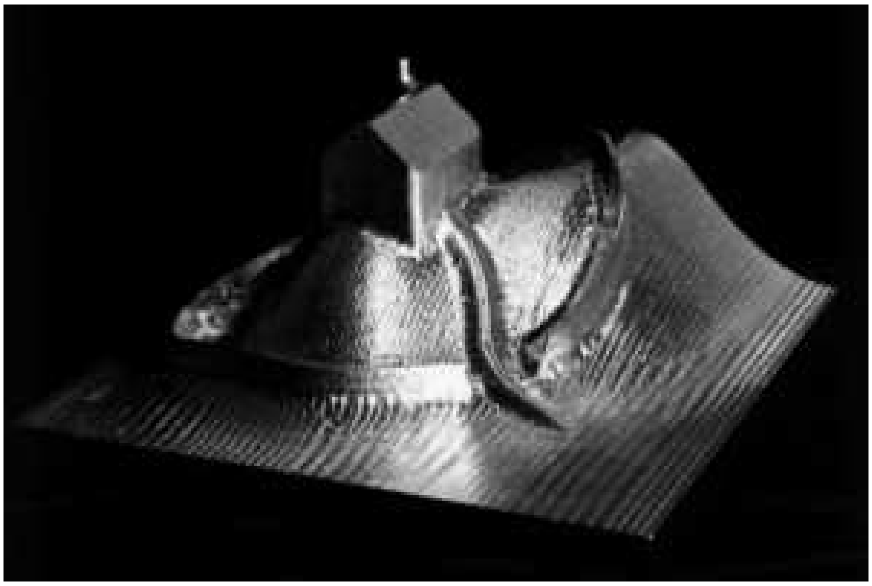

Figure 5.8. Aluminum milled version of the “house on the hill” model.

Assuming one can approximate a given surface by a set of disjoint (except along boundaries) piecewise ruled surfaces within a prescribed tolerance, lemma 5.1 can be used to test that each member of the set of ruled surfaces is also developable. Each developable surface can then be unfolded, laid flat and cut from a planar sheet such as paper or metal. By folding each back to its Euclidean orientation and stitching them all together, a C^0 approximation of the computer model is constructed.

Section 5.3.1 develops the background required for this method, and presents the basic algorithm. In section 5.3.2 we investigate several possible extensions including optimization, stub generation, and handling of trimmed surfaces. Section 5.3.3 lays out several examples including some models assembled from paper.

We will concentrate in our discussion on the NURBs representation although the developed technique may very well fit into any other piecewise polynomial or rational representation.

5.3.1 Algorithm

Let $S(u,v)$ be a nonuniform polynomial B-spline surface. Let the curves $C_1(u) = S(u, Vmin)$ and $C_2(u) = S(u, Vmax)$ be the $Vmin$ and $Vmax$ boundary curves of $S(u,v)$ respectively, $C_1(u) \neq C_2(u)$. Let $R(u,v)$ be the ruled surface constructed between $C_1(u)$ and $C_2(u)$. Let $\hat{R}(u,v)$ be the representation for $R(u,v)$ in the same B-spline basis as that of $S(u,v)$. $\hat{R}(u,v)$ can be obtained from $R(u,v)$ via appropriate degree raising [16, 17] and refinement [15] in the linear (ruled) direction, v. Then

$$\begin{aligned} \|S(u,v) - \hat{R}(u,v)\| &= \|\sum_{i=0}^{m}\sum_{j=0}^{n} P_{ij} B_{i,\tau}^{m}(u) B_{j,\xi}^{n}(v) - \sum_{i=0}^{m}\sum_{j=0}^{n} Q_{ij} B_{i,\tau}^{m}(u) B_{j,\xi}^{n}(v)\| \\ &= \|\sum_{i=0}^{m}\sum_{j=0}^{n} (P_{ij} - Q_{ij}) B_{i,\tau}^{m}(u) B_{j,\xi}^{n}(v)\| \\ &\leq max(\|P_{ij} - Q_{ij}\|, \ \forall i, j), \end{aligned} \tag{5.5}$$

because the B-spline basis functions are nonnegative and sum to one.

The difference of two rational surfaces can be computed in a similar way although it is more complex and must deal with products of scalar surfaces when the two are brought to a common denominator.

From the way $\hat{R}$ is constructed it is clear that the first row of S control mesh is the same as the first row of $\hat{R}$ control mesh, that is $P_{0j} = Q_{0j}$ $\forall j$. Similarly, the last row of S control mesh is the same as the last row of $\hat{R}$ control mesh, that is $P_{mj} = Q_{mj}$ $\forall j$. The jth column of S control mesh will be referred to as $P_{\bullet j}$. Equation (5.5) provides a simple mechanism to bound the maximum distance between S and the ruled surface R.

Isocurves of R (and $\hat{R}$) in the ruled parametric direction have constant speed because R is linear in this parameter. The bound in equation 5.5 provides a good bound of the distance when the v isocurves of S also have constant speed, that is when $\|\frac{\partial S(u_0,v)}{\partial v}\| = c$, for all u_0.

Unfortunately, when $\frac{\partial S}{\partial v}$ is not constant, the number of ruled surfaces in the resulting approximation can be unnecessarily large in order to meet the required tolerance. A method to correct for this problem uses the fact that control points can be associated with spline node values to obtain a surface-mesh parametric relation [55]. By degree raising R into $\hat{R}$, equally spaced in Euclidean space rows are introduced into the mesh that preserves the constant speed in the ruled direction, v. Because S does not have, in general, constant speed v isocurves, one can consider unequal spacing of the introduced mesh rows. Such strategy can project a single column in the v direction of the control mesh of S, $P_{\bullet j}$, onto the linear segment connecting P_{0j} and P_{mj} which are also control points of $C_1(u)$ and $C_2(u)$ respectively (see Figure 5.9). The spacing of these projected points can then be used to place the interior control points of $\hat{R}$. Figure 5.9 demonstrates this process. Figure 5.9a has the original surface S. The control mesh of S is used in Figure 5.9b to define the mesh of $\hat{R}$, by projecting a single column of the mesh of S, $P_{\bullet j}$, onto the line connecting P_{0j} and P_{mj}. The new ruled surface, $\hat{R}$, constructed with this new spacing is shown in Figure 5.9c.

The added degree of freedom of a nonuniform v speed ruled surface approximation includes the uniform v speed ruled surface as a special case and so can always be as good approximation as the uniform speed approximation. Let $C_1(v)$ and $C_2(v)$ be two isocurves of S in the v direction. Because we consider only one column of S mesh, this strategy will be able to emulate S v speed well only if $\left|\frac{dC_1(v)}{dv}\right| \Big/ \left|\frac{dC_2(v)}{dv}\right|$ is almost constant for all v. This condition holds fairly well for large classes of surfaces, but will not necessarily hold for surfaces constructed via highly nonisometric operations such as warp [13]. However, it does eliminate the need for degree raising or refinement in the construction of $\hat{R}$, because the continuity (knot vector) of S in the v direction is inherited.

A distance bounded algorithm approximating an arbitrary tensor product sur-