Piecewise Developable Surface Approximation of General NURBS Surfaces, with Global Error Bounds

Jacob Subag and Gershon Elber

Technion - Israel Institute of Technology, Haifa 32000, Israel
jsubag@cs.technion.ac.il

Abstract. Developable surfaces possess qualities that are desirable in the manufacturing processes of CAD/CAM models. Specifically, models formed out of developable surfaces can be manufactured from planar sheets of material without distortion. This quality proves most useful when dealing with materials such as paper, leather or sheet metal, which cannot be easily stretched or deformed during production.

In this work, we present a semi-automatic algorithm to form a piecewise developable surface approximation of a general NURBS surface. These developable surfaces are constructed as envelopes of the tangent planes along a set of curves on the input surface. Furthermore, the Hausdorff distance between the given surface and the approximating set of developables is globally bounded by a user-provided threshold.

1 Introduction and Related Work

Developable surfaces are surfaces that can be unfolded to the plane (flattened) with no distortion and are divided into three families of surfaces: cylinders, cones[1] and envelopes of the tangent planes along curves on surfaces [3]. In many manufacturing processes, specifically when dealing with sheet materials such as paper, leather or sheet metal, developable surfaces are used to determine the actual regions to be cut from the material in order to construct the final product. However, most freeform surfaces created by CAD/CAM applications are not developable by design nor can they be divided in such a way to produce parts that are all developable.

This modeling/manufacturing problem has been addressed by several approaches in the past. One approach has been to model with developable surfaces to begin with. Aumann [1] provided necessary and sufficient conditions for Bézier surfaces, interpolating two curves, to be developable and free of singular points. Pottmann and Farin [19] presented ways to represent and model developable Bézier and B-spline surfaces and Park et al. [16] described an interpolative optimal control problem, generating a developable surface from two points and a curve of tangent directions connecting them.

Another approach was to assume the existence of a developable surface and reconstruct or approximate it. Peternell [17] [18] presented algorithms for reconstructing

[1] By cylinders we refer to surfaces which can be represented as a one parameter family of parallel lines and by cones we refer to surfaces which can be represented as a one parameter family of lines that share an intersection point.

developable surfaces from point clouds. Hoschek and Pottmann [13] used samples of tangent planes, on a developable surface, to find a developable B-spline surface which interpolates/approximates them. Leopoldseder and Pottmann [14] approximated pre-existing developable surfaces with (parts of) cones. Lastly, Pottmann and Wallner [20] approximated a set of tangent planes with a developable surface, measuring the quality of the approximation with a distance metric for tangent planes in a small region of interest. While these two approaches circumvent some design problems, there are objects, in everyday products, which cannot be modeled as a union of developable surfaces, as they are inherently non-developable, i.e. they are doubly curved.

The third approach was the approximation of a freeform surface, or samples taken from one, by a set of developable surfaces. Hoschek [12] presented algorithms to approximate surfaces of revolution with developable polygonal strips or cones while bounding the Hausdorff distance with a predetermined threshold, by reducing the problem to bounding the distance between a curve and a polyline, both in the plane. However, surfaces of revolution are not expressive enough to be used exclusively when modeling, in real life applications. Elber [8] approximated a general freeform surface by generating developable surfaces that connect isolines on the input surface and bound the Hausdorff distance with a predetermined threshold, by bounding the magnitude of the surface-developables difference. However, this restriction of the developable surfaces to be between isolines of the input surface, results in a limited set of possible approximating developables for each approximated surface. Over-segmentation of the original surface with the approximating set of developables may result, in many cases. Finally, Chen et al. [4] approximated a set of sampled points and accompanying normals with a set of smoothly joined cones and cylinders. This algorithm performs well for samples on a developable (or nearly developable) surface, but would require heavy segmentation of the samples to handle complex non-developable surfaces. Furthermore, Chen et al. bound the approximation error only for the sampled points.

Our proposed method subscribes to the latter approach. We approximate general freeform surfaces unlike [12], which approximates only surfaces of revolution, and use envelopes of the tangent planes along a set of curves on the input surface, which generalize cones and cylinders, rather than limit the output surfaces to cones and cylinders such as in [14,4] or ruled surfaces between two isocurves on the input surface such as in[8]. Furthermore, the approximation error is globally bounded with a user supplied threshold, unlike in [17,18,13,20].

The approximating developable surfaces are constructed by sampling specific lines, on the envelopes, which are restricted to the minimal length needed for approximating a calculated region on the input surface. By interpolating in-between these sampled lines, a finite developable surface is formed, which is contained in the infinite envelope of tangents. The computed approximation error is used to refine our developable surfaces until the user threshold is achieved.

The rest of this article is divided as follows: Section 2 details our main algorithm. Section 3 presents experimental results and in Section 4, we summarize and discuss future work.

2 Approximation Technique

In this section, the piecewise developable approximation algorithm is presented. Our approximation errors are measured using the Hausdorff distance metric, which is defined for two objects, O_1 and O_2, as:

$$D_H(O_1, O_2) = \max\{\max_{p_1 \in O_1}\{\min_{p_2 \in O_2}\{\|p_1 - p_2\|\}\}, \max_{p_2 \in O_2}\{\min_{p_1 \in O_1}\{\|p_1 - p_2\|\}\}\},$$

where p_i is a point on O_i.

The piecewise developable approximation algorithm comprises of several stages. The specifics of each stage are given below. We now present a short overview of the algorithm. Given a parametric curve, $c = (u(t), v(t))$, supplied by the user, generate an infinite developable surface, E, as the envelope of tangent planes along $S(c)$. Then, calculate the region of S, which is approximated up to an ε by an individual line along E. Using a sampled set of such lines, a new finite developable surface, $\widetilde{E} \subset E$, is interpolated and the Hausdorff distance between $\widetilde{E}$ and a relevant region in S, denoted $\widetilde{S}$, is bounded. Then, the developable surface is subsequently refined, until the user supplied threshold of approximation is met and its boundaries are smoothed. The user is interactively prompted to continue and add curves over S, for which the above process is repeated, until the union of the resulting developable surfaces forms a complete approximation of S. Possible ways of automating the process of adding the curves are discussed in Section 4.

2.1 Envelope Surfaces

Given a C^2 continuous surface, $S(u, v)$, and a C^2 continuous curve, $c(t) = (u(t), v(t))$ in the parametric domain of S. An envelope surface along $C(t) = S(c(t))$ is defined as:

$$E(t, r) = C(t) + r\hat{W}(t), \quad r \in I\!\!R, \tag{1}$$

where $W(t) = N(t) \wedge N'(t)$, $N(t)$ is defined as the normal of S at point $S(c(t))$, $\wedge$ denotes the cross product, and $\hat{A} = \frac{A}{\|A\|}$.

$E(t, r)$ is developable [3]. The definition presented by Equation (1) necessitates the normalization factor of W, which is non-rational. We use normalized vectors for some of the pointwise evaluations (denoted by $\hat{W}$) and employ the unnormalized W, whenever possible. When $C(t)$ passes through planar regions of S, $N'(t)$ vanishes and $\hat{W}$ is undefined. This problem can be solved by trimming planar regions from S, being developable, before proceeding with our algorithm[2]. In the ensuing discussion, we assume W is always defined.

Another problem that stems from Equation (1) is the order of curve $W(t)$. Recalling that $N = \frac{\partial S}{\partial u} \wedge \frac{\partial S}{\partial v}$, composed over $c(t)$, we know that $W(t)$ can frequently attain extremely high orders. High order curves may produce numerical errors when evaluated. In our implementation, we fit low order spline approximation [6] to $W(t)$, while bounding the error, a lower order representation used hereafter. See [5] for more details on reducing degrees of symbolically computed curves.

[2] This segmentation could be performed by trimming away from S all regions for which $|K| < \varepsilon$, K being the Gaussian curvature, is computed symbolically [9].

2.2 Approximation by Ruling

A *single ruling* at parameter value t_0 is an infinite line, $L_{t_0}(r) = E(t_0, r)$, parallel to $W(t_0)$ and incident on $C(t_0)$ (see Figure 1 (b)). The region of S that is approximated by L_{t_0} is, therefore, all the points on S that are within some prescribed distance, ε, (in the Euclidean sense) from L_{t_0}. In other words, all the points on S for which the following equation holds:

$$\|S(u,v) - C(t_0) - \langle S(u,v) - C(t_0), \hat{W}(t_0)\rangle \hat{W}(t_0)\| \leq \varepsilon. \qquad (2)$$

Equation (2) can be reformulated as:

$$F_{t_0}(u,v) = \|S(u,v) - C(t_0) - \langle S(u,v) - C(t_0), \hat{W}(t_0)\rangle \hat{W}(t_0)\|^2 - \varepsilon^2 \leq 0, \quad (3)$$

in order to deal with a rational function. Furthermore, $F_{t_0}(u,v) = 0$ defines an implicit curve in the parametric domain of S, which bounds the region(s) of S that are within ε from L_{t_0}. Geometrically speaking, $F_{t_0} = 0$ defines the intersection curve of an ε-radius cylinder centered around L_{t_0} and the input surface S.

The topology of implicit Equation (3) is analyzed by employing a modified version of the algorithm presented by Hass et al. in [11], which provides the accurate topology of an implicit curve. The result of Hass' algorithm is a set of polygons, $\{P^i_{t_0}\}$, whose vertices lie on $F_{t_0} = 0$ and are topologically equivalent to the implicit curve defined by $F_{t_0} = 0$ (see Figure1 (c)). This result is insufficient for our purpose, as the edges of $\{P^i_{t_0}\}$ can be arbitrarily far from $F_{t_0} = 0$. More importantly, $\{P^i_{t_0}\}$ may contain points not approximated by the ruling to within ε, i.e., points for which $F_{t_0} > 0$. In order to guarantee that $\forall i, F_{t_0}(P^i_{t_0}) \leq 0$, we enhanced Hass' algorithm by sampling $F_{t_0} = 0$ up to a threshold, adding the inflection points of $F_{t_0} = 0$ (see [2] for an analysis of inflection points of implicit curves) as vertices of the resulting polygons, and adding a post processing refinement step, ensuring the result, $\{P^i_{t_0}\} \subset (F_{t_0} \leq 0)$. Due to space limitations, the complete account of these modifications is omitted.

Only one polygon in $\{P^i_{t_0}\}$ contains $c(t_0)$. Denote it as $P^0_{t_0}$ (see Figure 1 (a)) and denote its individual edges as $\{e_j\}$. Since $F_{t_0}(c(t_0)) = -\,\varepsilon^2$ and S (and therefore, F) is continuous, there is an environment surrounding $c(t_0)$, for which Equation (3) holds and which will be contained in $\{P^i_{t_0}\}$. This last step eliminates disjoint approximated regions as they may result in undesirable disjoint developable surfaces.

Only $P^0_{t_0}$ will be employed in the coming steps. The region $S(P^0_{t_0})$ is approximated to within ε by the infinite ruling L_{t_0}. However, there is a unique and minimal interval along L_{t_0} that approximates $P^0_{t_0}$ to within ε. We seek to derive this minimal interval by calculating the minimal and maximal coordinate values of this interval, with respect to Equation (1), parameterized by r.

Consider point $(u_0, v_0) \in P^0_{t_0}$. The r-coordinate value of (u,v) prescribes the point on $L_{t_0}(r)$ closest to $S(u,v)$ and equals,

$$r_{t_0}(u,v) = \frac{\langle S(u,v) - C(t_0), W(t_0)\rangle}{W(t_0)^2}, \qquad (4)$$

where the unnormalized $W(t_0)$ is used, resulting in r-coordinate values that counter its magnitude's effect in the upcoming construction of the developable surface.

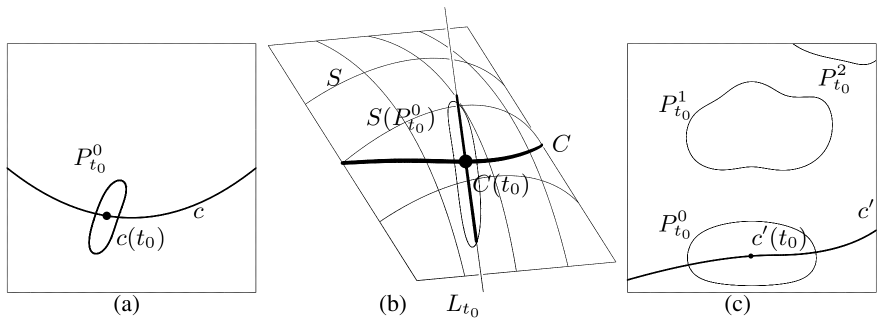

Fig. 1. (a) The parametric domain of S with the parametric curve $c(t)$ and the parametric region approximated by the ruling L_{t_0}, i.e. $P^0_{t_0}$. (b) A single ruling along $C(t) = S(c(t))$. The thick section of the ruling is the minimal interval needed to approximate $P^0_{t_0}$ to within ε, i.e. $L_{t_0}(r)$. Also shown, in (b), is the composition of S over $P^0_{t_0}$. (c) An example of several approximated regions, $\{P^i_{t_0}\}$, for different surface and parametric curve.

Equation (4) can be evaluated for all $(u, v) \in P^0_{t_0}$. Since r_{t_0} is continuous and $P^0_{t_0}$ is a one connected region, in the parametric domain of S, the locus of r-coordinate values over $P^0_{t_0}$ and its interior, is a continuous interval. As such, we need only find the minimal and maximal values of r. We do so by first solving for:

$$\nabla r_{t_0}(u, v) = 0, \quad (u, v) \in P^0_{t_0}, \tag{5}$$

finding interior local extrema. Then, the solutions of

$$\frac{\partial r_{t_0}(e_j(s))}{\partial s} = 0, \quad \forall e_j \in P^0_{t_0}, \tag{6}$$

detect extrema along edges of $\in P^0_{t_0}$. Finally, endpoints of every edge e_j (the vertices of $P^0_{t_0}$), are examined. This analysis is possible when S is piecewise C^3 which results in $r_{t_0}(u, v)$ being differentiable. When this is not the case, we need to also examine Equation (4) along the isolines corresponding to each knot. Denote these computed minimal and maximal values of r by $r^{t_0}_{min}$ and $r^{t_0}_{max}$. Since the r-coordinate value for $c(t_0)$ is zero and $P^0_{t_0}$ was chosen to include $c(t_0)$, $r^{t_0}_{min}$ is negative and $r^{t_0}_{max}$ is positive.

We now consider the line segment:

$$L_{t_0}(r) = C(t_0) + rW(t_0), \quad r \in [r^{t_0}_{min}, r^{t_0}_{max}], \tag{7}$$

as the ruling approximating $P^0_{t_0}$, see the thick segment along L_{t_0} in Figure 1 (b).

2.3 Inter-ruling Interpolation

Consider a set of rulings $\{L_{t_i}\}$, sampled along $C(t)$ at $\{t_i\}^n_{i=0}, n \geq 1$. Without loss of generality, assume $t_i < t_{i+1}$. Analyzing each ruling's approximated region, $P^0_{t_i}$ (see Figure 2 (a)), we seek to interpolate a complete developable surface along $C(t)$. As a first step, interpolate two scalar curves, $r_{min}(t)$ and $r_{max}(t)$, such that: $r_{min}(t_i) = r^{t_i}_{min}$ and $r_{max}(t_i) = r^{t_i}_{max}, \forall i$. Then, we proceed by creating two new curves:

$$C_1(t) = C(t) + r_{min}(t)W(t), \quad C_2(t) = C(t) + r_{max}(t)W(t). \tag{8}$$

Finally, construct the local developable surface, as a ruled surface between $C_1(t)$ and $C_2(t)$:

$$\widetilde{E}(t,r) = (1-r)C_1(t) + rC_2(t), \quad r \in [0,1], t \in [t_0, t_n], \tag{9}$$

(see Figure 2 (b)). $\widetilde{E} \subset E$ (recall Equation (1)) and is hence developable.

So far, this construction guarantees nothing with regard to the approximating qualities and quantities of $\widetilde{E}$ at $t \notin \{t_i\}_{i=0}^n$. The next section provides such bounds.

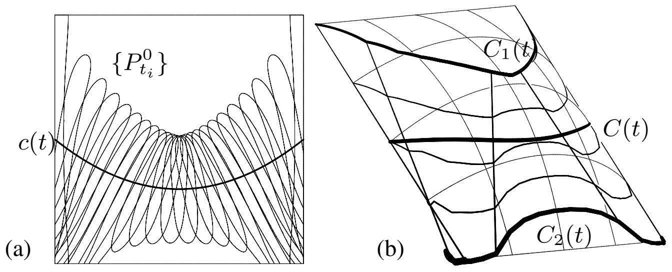

Fig. 2. (a) The individual regions approximated by a set of sampled rulings in the paramteric domain of S, $\{P^0_{t_i}\}_{i=0}^n$. (b) The developable ruled surface $\widetilde{E}$ and curves $C_1(t)$ and $C_2(t)$, which are used to generate it.

2.4 Bounding the Hausdorff Distance Between the Developable and Input Surface

In this section, we bound the Hausdorff distance between $\widetilde{E}$ (recall Equation (9)) and the relevant region of S, which is approximated by $\widetilde{E}$ and denoted as $\widetilde{S}$. Construct a "matching" function $T(t,r) \rightarrow (u,v)$ that assigns to each point in the parametric domain of $\widetilde{E}$, a point in the parametric domain of S. Then, globally bound $\|S(T(t,r)) - \widetilde{E}(t,r)\|$. If this bound is found to be smaller than or equal to ε, clearly $D_H(\widetilde{S}, \widetilde{E}) = D_H(S(T(t,r)), \widetilde{E}(t,r))$ will also be bounded by ε.

Thus, we seek a matching function $T(t,r) = (t_u(t,r), t_v(t,r))$, which reparameterizes S so as to diminish:

$$\max_{t,r} \|S(T(t,r)) - \widetilde{E}(t,r)\|. \tag{10}$$

Therefore, an ideal T would assign to each point on $\widetilde{E}$, the parametric coordinates of the closest point on S. We compromise by constructing T as a piecewise bilinear mapping, which interpolates solutions of Equation (10) for a set of (initially) three points for each ruling, corresponding to the ruling's start, incidence and end points. Specifically,

$$T(t,r) = \sum_{i=0}^{n}\sum_{j=0}^{2} Q_{i,j} B_{i,1}(t) B_{j,1}(r), \tag{11}$$

where $B_{i,1}$ is the linear i'th B-spline basis function and where

$$\{Q_{i,j}\}_{i=0,j=0}^{n,2} = \begin{cases} \underset{u,v}{argmin} \| S(u,v) - \widetilde{E}(t_i, 0) \|, j = 0, \\ \qquad c(t_i), \qquad\qquad j = 1, \\ \underset{u,v}{argmin} \| S(u,v) - \widetilde{E}(t_i, 1) \|, j = 2. \end{cases} \tag{12}$$

Having constructed T, we are ready to bound $\| S(T(t,r)) - \widetilde{E}(t,r) \|$ with the user supplied ε. As before and in order to deal with rational functions, we bound the following against ε^2:

$$\| S(T(t,r)) - \widetilde{E}(t,r) \|^2 \triangleq \| \widetilde{S}(t,r) - \widetilde{E}(t,r) \|^2. \tag{13}$$

Consider the original surface $S(u,v) = \sum_{i=0}^{N} \sum_{j=0}^{M} P_{i,j} B_{i,n}(u) B_{j,m}(v)$, where $P_{i,j}$ are the control points of S and $B_{i,n}$ is the i'th B-spline basis function of degree n. In order to calculate $S(t_u(t,r), t_v(t,r))$, divide S at all its internal knots into Bézier patches, $\{S_k\}$, $S_k(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} P_{i,j}^k \theta_{i,n}(u) \theta_{j,m}(v)$, where $P_{i,j}^k$ are the control points of S_k and $\theta_{i,n}$ is the i'th Bézier basis function of degree n. Now composing S_k over T, one gets,

$$S_k(t_u^k(t,r), t_v^k(t,r)) = \sum_{i=0}^{n}\sum_{j=0}^{m} P_{i,j}^k \theta_{i,n}(t_u^k(t,r)) \theta_{j,m}(t_v^k(t,r)),$$

where $t_u^k(t,r) = \frac{t_u(t,r) - u_{min}^k}{u_{max}^k - u_{min}^k}$ and $t_v^k(t,r) = \frac{t_v(t,r) - v_{min}^k}{v_{max}^k - v_{min}^k}$ and where u_{min}^k, u_{max}^k, v_{min}^k, v_{max}^k are the knot values that bound the region in the parametric domain of S, from which S_k was extracted.

Recalling that, $\theta_{i,n}(t) = \binom{n}{i}(1-t)^{n-i} t^i$, we end up with

$$S_k(t_u^k(t,r), t_v^k(t,r)) = \sum_{i=0}^{n}\sum_{j=0}^{m} P_{i,j}^k \binom{n}{i}\binom{m}{j} (1 - t_u^k)^{n-i} {t_u^k}^i (1 - t_v^k)^{m-j} {t_v^k}^j. \tag{14}$$

In the last expression, t_u^k and t_v^k may exceed the domain of S_k. While this limitation prevents us from computing the complete composition of S over T, it is sufficient for the error bound analysis we require. Interested in the maximal value of Equation (13), instead of trimming t_u^k and t_v^k to fit each of the S_k's domains, we leave them "as is" and evaluate each S_k composition formula (14). The resulting set of surfaces $S_k \circ T$, denoted as $\widetilde{S}_k$, are each identical to $S \circ T$ for their shared domain. Therefore, only extremums inside the domain of $\{S_k\}$, are considered in:

$$(k_{max}, t_{max}, r_{max}) = \underset{k,t,r}{argmax} \| \widetilde{S}_k(t,r) - \widetilde{E}(t,r) \|^2.$$

Now, the evaluation of $\| \widetilde{S}_{k_{max}}(t_{max}, r_{max}) - \widetilde{E}(t_{max}, r_{max}) \|^2$ provides a bound on Expression (13) and, hence, a bound for the squared Hausdorff distance between $\widetilde{E}$ and $\widetilde{S}$.

2.5 Refinement of the Approximated Region

If the bound calculated in the previous subsection, is smaller than or equal to ε^2, then our approximation is sufficiently accurate. Otherwise, we must refine T and $\widetilde{E}$ until the bound is smaller than or equal to ε^2. Refinement can be achieved by three complementary methods, along the t and r parametric directions of $\widetilde{E}$ (see Figure 3).

The first method of refinement adds more rulings. This method improves T and $\widetilde{E}$, when the extremum (t_{max}, r_{max}) lies in-between two consecutive rulings corresponding to t_i and t_{i+1}, which means that either T performs poorly when interpolating the bilinear region between said rulings or $\widetilde{E}$ is too far from S at that point. In such cases, we add a new ruling at $\frac{t_i+t_{i+1}}{2}$, update $\widetilde{E}$ and T and re-analyze expression (13), see Figure 3 (b) for an example of the first method of refinement.

The second method of refinement adds control points to T, for each ruling. This method is meant to handle cases when T interpolates poorly on or near a specific ruling. In such cases, $t_{max} \approx t_i$ and the former refinement method would require many iterations to reduce the bound or even fail. Thus, when $t_{max} \approx t_i$, we apply both the first and the second refinement methods. Recall Equation (12), in which we constructed T with three control per ruling. In this second refinement method, we generalize Equation (12) with:

$$\{Q_{i,j}\}_{i=0,j=0}^{n,2^m} = \begin{cases} \underset{u,v}{argmin}\|S(u,v) - (1-\alpha_j)\widetilde{E}(t_i,0) - \alpha_j C(t_i)\|, & j < 2^{m-1}, \\ c(t_i), & j = 2^{m-1}, \\ \underset{u,v}{argmin}\|S(u,v) - (\alpha_j - 1)\widetilde{E}(t_i,1) - (2-\alpha_j)C(t_i)\|, & j > 2^{m-1}, \end{cases} \tag{15}$$

where $\alpha_j = \frac{j}{2^{m-1}}$.

Then, the meaning of the second refinement method is to increment m in Equation (15) by one, (almost) doubling the amount of control points corresponding to each ruling. After applying this method of refinement we, again, need to update T, re-analyze expression (13) and apply further refinements as needed. See Figure 3 (c) for an example of the second method of refinement.

The third method of refinement shrinks $\widetilde{E}$ towards $C(t)$ and T towards $c(t)$ and is applied when a pre-determined number of refinements, $max\_refinements$, fails to reduce the bound for the Hausdorff distance. This shrinkage is attained by halving the values of $r_{min}^{t_i}$ and $r_{max}^{t_i}$, for every sampled ruling and reconstructing $\widetilde{E}$ and T. This step is taken to ensure convergence of the refinement algorithm, when

$$\forall(t,r), \exists(u,v) \in Image(T) \text{ such that } \|S(u,v) - \widetilde{E}(t,r)\| \leq \varepsilon, \tag{16}$$

while

$$\exists(u,v) \in Image(T) \text{ such that } \forall(t,r), \|S(u,v) - \widetilde{E}(t,r)\| > \varepsilon. \tag{17}$$

Meaning that $\widetilde{E}$ is approximated well by $\widetilde{S}$ (Equation (16)) while some points on $\widetilde{S}$ are too far from $\widetilde{E}$ (Equation (17)). See Figure 3 (d) for an example of the third method of refinement.

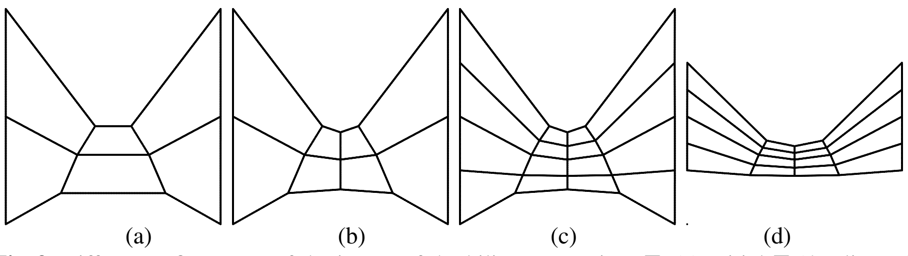

Fig. 3. Different refinements of the image of the bilinear mapping, T. (a) Initial T (4 rulings, 3 samples per ruling). (b) After applying the first refinement method to (a) (having now, 5 rulings, 3 samples per ruling). (c) After applying the second refinement method to (b) (having now, 5 rulings, 5 samples per ruling). (d) After applying the third refinement method to (c).

In summary, the algorithm used to refine T and $\widetilde{E}$ is:

Refine($S, \widetilde{E}, \{t_i\}$)

```
1: $m \leftarrow 1$;
2: $num\_refinements \leftarrow 0$;
3: Construct $T$, using the current value of $m$;
4: Find the point of maximal difference, $(t_{max}, r_{max})$;
5: while $\|\widetilde{S}(t_{max}, r_{max}) - \widetilde{E}(t_{max}, r_{max})\|^2 > \varepsilon^2$ do
6:    $num\_refinements \leftarrow num\_refinements + 1$;
7:    Find $i$ such that $t_i \le t_{max} \le t_{i+1}$;
8:    Add ruling at $\frac{t_i + t_{i+1}}{2}$; {refinement of the first kind}
9:    if $t_{max} \simeq t_i$ or $t_{max} \simeq t_{i+1}$ then
10:      $m \leftarrow m + 1$; {refinement of the second kind}
11:   if $num\_refinements = max\_refinements$ then
12:      $num\_refinements \leftarrow 0$;
13:      for all ruling $t_i$ do
14:         $r^{t_i}_{min} \leftarrow \frac{r^{t_i}_{min}}{2}$,  $r^{t_i}_{max} \leftarrow \frac{r^{t_i}_{max}}{2}$; {refinement of the third kind}
15:   Reconstruct $\widetilde{E}$;
16:   Reconstruct $T$, using the current value of $m$;
17:   Find the point of maximal difference, $(t_{max}, r_{max})$;
18: End
```

Using the combined application of the first and second methods of refinement, T converges to a mapping function, which solves Equation (10) for every point (t, r) in the domain of $\widetilde{E}$. This process is usually sufficient. However, in cases identified by Equations (16) and (17), the third method of refinement ensures convergence, as it shrinks $\widetilde{E}$ to $C(t)$ and T to $c(t)$. In summary, the Hausdorff distance between the limit developable surface and the composition of S over the limit matching function is bounded by ε and the refinement algorithm stops.

The parametric region of S we approximated is $Image(T)$. Notice that as T is piecewise bilinear, $Image(T)$ is a polygon. Specifically, it is the polygon $\{Q_{0,j}\}_{j=0}^{2^m}$, $\{Q_{i,2^m}\}_{i=0}^{n}$, $\{Q_{n,j}\}_{j=2^m}^{0}$, $\{Q_{i,0}\}_{i=n}^{0}$, where $\{Q_{i,j}\}_{i=0,j=0}^{n,2^m}$ is the set of control points of T.

2.6 Smoothing the Approximated Region

As the boundary of $Image(T)$ is piecewise linear, depending on the amount of rulings sampled, these boundaries may be jagged. In order to avoid these jagged boundaries which, in turn, lead to jagged developables, we smooth the boundaries of $Image(T)$.

$Image(T)$ is comprised of the control points $\{Q_{i,j}\}_{i=0,j=0}^{n,2^m}$, of T. As $n+1$, the number of rulings, is typically significantly larger than 2^m+1, the number of sampled control points along a ruling, we opt to smooth the boundary polylines: $\{Q_{i,0}\}_{i=0}^{n}$ and $\{Q_{i,2^m}\}_{i=0}^{n}$, which are, therefore, more prone to jagged features.

This smoothing problem is defined as follows: Let $c(t)$ be a curve in the parametric domain of S, as described in Section 2.1 and $\{Q_{i,j}\}_{i=0,j=0}^{n,2^m}$ be the set of control points of the piecewise bilinear surface defined in Section 2.4 as T. Then, we wish to find two new sets of control points $\{\widetilde{Q}_{i,0}\}_{i=0}^{n}$ and $\{\widetilde{Q}_{i,2^m}\}_{i=0}^{n}$, which satisfy the following expressions:

$$\widetilde{Q}_{i,j} = (1-\beta_{i,j})c(t_i) + \beta_{i,j}Q_{i,j}, \quad 0 \le \beta_{i,j} \le 1, \tag{18}$$

for $0 \le i \le n$, $j \in \{0, 2^m\}$, such that the following expression is minimized:

$$\gamma \sum_{i=1}^{n-1} \sum_{j\in\{0,2^m\}} \|\widetilde{Q}_{i-1,j} - 2\widetilde{Q}_{i,j} + \widetilde{Q}_{i+1,j}\|^2 + \sum_{i=0}^{n} \sum_{j\in\{0,2^m\}} \|\widetilde{Q}_{i,j} - Q_{i,j}\|^2, \tag{19}$$

where γ is a user selected smoothness weight. The left-hand side of Expression (19) penalizes (curved) jagged edges and the right-hand side penalizes large changes.

This problem is linear with regards to $\beta_{i,j}$ and we solve it as an over-determined, linearly constrained, least squares problem. See Figures 4 and 5 for an example of the smoothing algorithm.

After smoothing the edges of $Image(T)$, we recalculate for every ruling, $r_{min}^{t_i}$ and $r_{max}^{t_i}$ over the smoothed approximated region, and then reconstruct $\widetilde{E}$ and T (with the same values of n and m as in the last version) and re-analyze the Hausdorff distance. If the Hausdorff distance is bounded by ε without additional refinement, we stop. Otherwise, we refine T and $\widetilde{E}$, as needed, and repeat the smoothing algorithm and so on.

Note that as each smoothing only diminishes the magnitudes of $r_{min}^{t_i}$ and $r_{max}^{t_i}$, i.e. shrinks $\widetilde{E}$ towards $C(t)$, bounding the Hausdorff distance would require a less refined T. In all our experiments, we never needed to apply further refinement after applying the smoothing algorithm.

2.7 Adding Developables

So far, we handled one user supplied curve, c over S, which we used to create the first developable surface. In order to generate a complete piecewise developable approximation of S, we need to add more developable surfaces along additional curves on S. Our algorithm provides the user with the boundary of the already approximated region as a candidate curve, which he can accept or modify using the GUI. Alternatively, the user can provide an entirely new curve by drawing it on S, which we project to the parametric domain of S for our algorithm to use. Alternatively, a greedy scheme that uses the boundary of an approximated region to construct one developable surface after another can be used, until all the domain of S is covered.

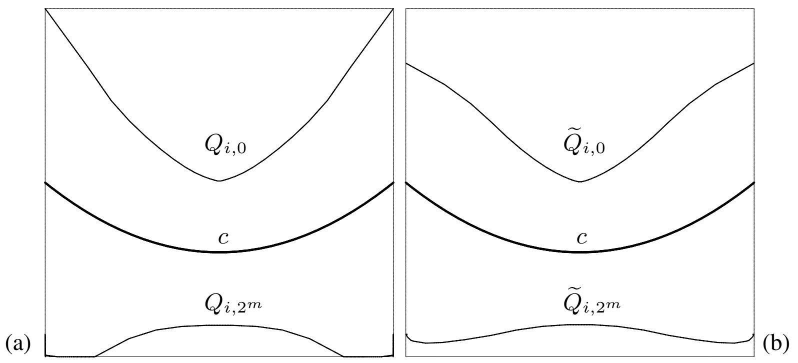

Fig. 4. (a) The polylines $\{Q_{i,0}\}_{i=0}^{n}$, and $\{Q_{i,2^m}\}_{i=0}^{n}$ in the parametric domain of S prior to and (b) after smoothing, see also Figure 5

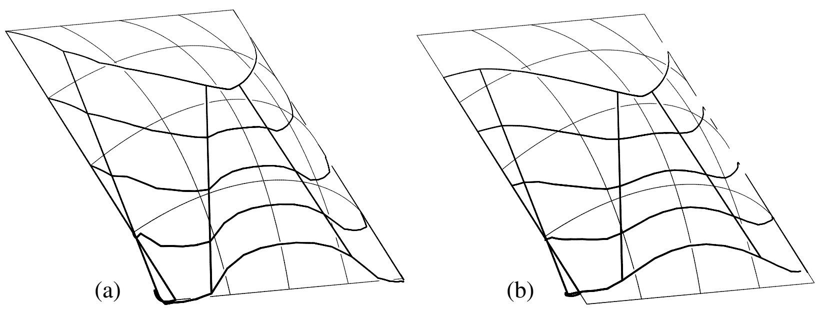

Fig. 5. (a) The developable ruled surface $\widetilde{E}$ before smoothing and (b) after smoothing the parametric edges, see also Figure 4

An obvious difficulty to overcome, when more than one developable is present, is the problem of overlaps between adjacent envelope surfaces, i.e. having the coverage of S being mutually exclusive. We've used three approaches to solving this problem. Due to lack of space they are only briefly described. The first approach limits the r-values generated in the analysis of Expression (4). This is achieved by subtracting the already approximated regions from the region approximated by each sampled ruling, $P^0_{t_i}$. The second approach constructs each developable surface independently, then projects boundaries of the already approximated regions as trimming curves of the new developable surface. The third approach uses a triangulation of the developables, created by either the first or the second approach, followed by a merge process of vertices on adjacent developables' boundaries.

3 Results

Figures 6 and 7 show experimental results of our algorithm. In Figure 6 we show an approximation of a bi-cubic B-spline surface (not a surface of revolution), bounded by the unit cube and modeling half of a fruit bowl shown in Figure 6 (a). The approximation

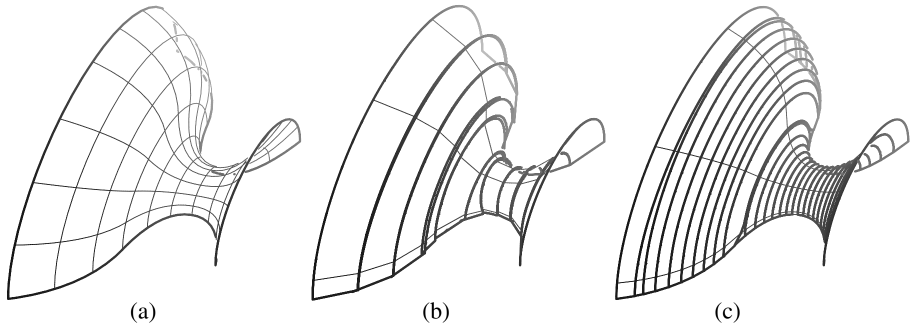

Fig. 6. Approximation of a bi-cubic surface bounded by the unit cube. (a) The input surface. (b) Approximation with tolerance of $\varepsilon = 10^{-2}$ (11 developable surfaces needed). (c) Approximation with tolerance of $\varepsilon = 10^{-3}$ (27 developable surfaces needed).

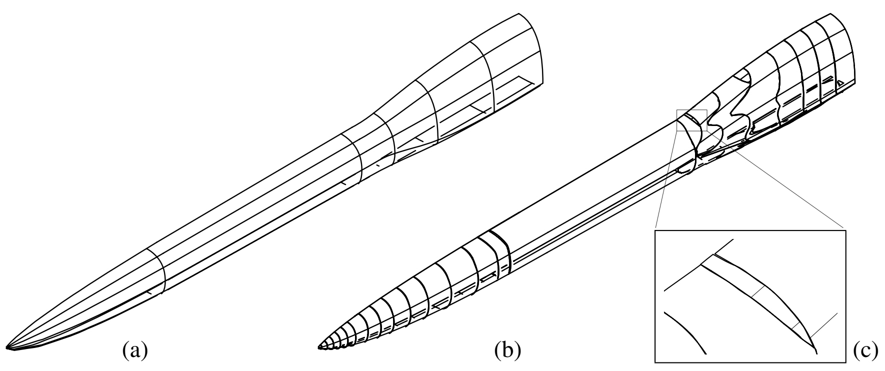

Fig. 7. Approximation of a bi-quadratic surface bounded by a $10 \times 1 \times 1$ box. (a) The input surface. (b) Approximation with tolerance of $\varepsilon = 10^{-3}$ (33 developable surfaces needed).

shown in Figure 6(b) is up to $\varepsilon = 10^{-2}$ and required 11 developables. The approximation shown in Figure 6 (c) is up to $\varepsilon = 10^{-3}$ and required 27 developables. In Figure 7 we show an approximation of a bi-quadratic B-spline surface, bounded by a $(10 \times 1 \times 1)$ box and modeling part of a jet fighter's fuselage shown in Figure 7 (a). The approximation shown in Figure 7 (b) is up to $\varepsilon = 10^{-3}$ and required 33 developables, some of which are too small to notice in the figure (see Figure 7 (c)). All of these examples were executed on a P-IV 2.8Ghz with 512mb of RAM and required, on average, 20 seconds were needed to generate each developable surface (aside from the time it took the user to add each curve).

In our implementation, we used MATLAB [15] to solve the constrained linear least squares problem defined in Section 2.5 and the IRIT solid modeler [7] as the GUI and as our symbolic computational environment (for more details on symbolic multivariate computations see [10]).

4 Conclusions and Future Work

In this paper, we presented an algorithm capable of approximating a general NURBS surface with a set of developable surfaces, constructed as envelopes of tangents of curves along the input surface and whose Hausdorff distance to the input surface is bounded by a user defined threshold. Currently, the algorithm suggests boundary curves to the user, who can accept, modify or ignore these suggestions and draw completely new curves. The presented procedure is time consuming and requires a certain expertise from the user. However, we feel this investment is justified, as this semi-automatic development process will be performed only once, and with typically better results than a completely automatic process, which can then be manufactured, in any quantity. The selection of curves greatly affects the number of developables needed to approximate the input surface as well as the size of developables. Clearly, the question of which parametric curves would yield the best results needs to be further explored.

Furthermore, we experimented with a completely automatic method of curve generation. This method uses one of the input surface's (trimmed) boundaries as the initial curve and creates an initial developable. Then, the boundary of the region approximated by the generated developable is used as the next curve and so on. This greedy "advancing front" process, when repeated, can generate a complete approximation with no user intervention. However, the resulting developable surfaces are not visually pleasing and are certainly not optimal, in the number of developables. We feel that a completely automatic process is the next logical step and intend to improve or extend this method to generate better results.

We also intend to further investigate the stitching problem of adjacent developables. Intersecting adjacent developables can be stitched, and sometimes trimmed, along the intersection curve. However, adjacent developables do not always intersect and simply connecting boundary sections of adjacent developables with new surfaces results in many small additional surfaces. Regardless, in real life, the developable surfaces, cut from the manufacturing material, have non-negligible thickness and therefore by selecting an error threshold smaller than the manufacturing tolerance, the user can ensure proper contact between adjacent developables.

Finally, some materials, such as cloth, latex etc., can be deformed (stretched) during the manufacturing process. These qualities result in relaxed demands on the developability of the approximating surfaces. An interesting subset of these materials can even handle anisotropic deformation, such as fabrics which can be stretched to different degrees in different directions. We hope to enhance our algorithm in order to exploit these qualities, thereby, generating a better approximation.

Acknowledgments

This work was partially supported by European FP6 NoE grant 506766 (AIM@SHAPE) and in part by the Minerva Schlesinger Laboratory for Life Cycle Engineering.

References

1. G. Aumann. Interpolation with developable Bezier patches. *Computer Aided Geometric Design*, 8(5):409–420, 1991.
2. J. Bloomenthal and B. Wyvill, editors. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997. Section 2.6.4.
3. M. P. D. Carmo. *Differential Geometry of Curves and Surfaces*, pages 195–197. Prentice-Hall, 1976.
4. H.-Y. Chen, I.-K. Lee, S. Leopoldseder, H. Pottmann, T. Randrup, and J. Wallner. On surface approximation using developable surfaces. *Graphical Models and Image Processing*, 61(2):110–124, 1999.
5. X. Chen, R. F. Riesenfeld, and E. Cohen. Degree reduction for NURBS symbolic computation on curves. In preparation.
6. E. Cohen, R. F. Riesenfeld, and G. Elber. *Geometric Modeling with Splines*, chapter 9. A K Peters, 2001.
7. G. Elber. Irit solid modeler. `http://www.cs.technion.ac.il/~irit`
8. G. Elber. Model fabrication using surface layout projection. *Computer-aided Design*, 27(4):283–291, April 1995.
9. G. Elber and E. Cohen. Second-order Surface Analysis Using Hybrid Symbolic and Numeric Operator. *ACM Trans. Graph*, 12(2):160–178, 1993.
10. G. Elber and M.-S. Kim. Geometric constraint solver using multivariate rational spline functions. In *The Sixth ACM/IEEE Symposium on Solid Modeling and Applications, Ann Arbor, Michigan*, pages 1–10, June 2001.
11. J. Hass, R. T. Farouki, C. Y. Han, X. Song, and T. W. Sederberg. Guaranteed consistency of surface intersections and trimmed surfaces using a coupled topology resolution and domain decomposition scheme. To appear in *Advances in Computational Mathematics*, 2005. `http://mae.ucdavis.edu/~farouki/index.html`
12. J. Hoschek. Approximation of surfaces of revolution by developable surfaces. *Computer-Aided Design*, 30(10):757–763, 1998.
13. J. Hoschek and H. Pottmann. Interpolation and approximation with developable B-spline surfaces. In M. Dæhlen, T. Lyche, and L. L. Schumaker, editors, *Proceedings of the first Conference on Mathematical Methods for Curves and Surfaces (MMCS-94)*, pages 255–264, Nashville, USA, June 16–21 1995. Vanderbilt University Press.
14. S. Leopoldseder and H. Pottmann. Approximation of developable surfaces with cone spline surfaces. *Computer-Aided Design*, 30(7):571–582, 1998.
15. Matlab ©, copyright 1984-2002, The Mathworks, Inc.
    See also http://www.mathworks.com/.
16. F. Park, J. Yu, C. Chun, and B. Ravani. Design of developable surfaces using optimal control. *Journal of Mechanical Design*, 124(4):602–608, December 2002.
17. M. Peternell. Developable surface fitting to point clouds. In *Computer Aided Geometric Design*, pages 785–803, 2004.
18. M. Peternell. Recognition and reconstruction of developable surfaces from point clouds. In *GMP*, pages 301–310, 2004.
19. H. Pottmann and G. E. Farin. Developable rational Bézier and B-spline surfaces. *Computer Aided Geometric Design*, 12(5):513–531, 1995.
20. H. Pottmann and J. Wallner. Approximation algorithms for developable surfaces. *Computer Aided Geometric Design*, 16(6):539–556, 1999.