ARTICLE IN PRESS

Available online at www.sciencedirect.com

ScienceDirect

Computer-Aided Design ▮ (▮▮▮▮) ▮▮▮–▮▮▮

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

Self-intersection detection and elimination in freeform curves and surfaces

Diana Pekerman[a,*,1], Gershon Elber[b], Myung-Soo Kim[c]

[a] *Department of Mathematics, Technion – IIT, Haifa 32000, Israel*
[b] *Department of Computer Science, Technion – IIT, Haifa 32000, Israel*
[c] *School of Comp. Science and Eng., Seoul National University, Seoul, Republic of Korea*

Received 3 May 2007; accepted 4 October 2007

Abstract

We present several algorithms for self-intersection detection, and possible elimination, in freeform planar curves and surfaces. Both local and global self-intersections are eliminated using a binormal-line criterion and a simple direct algebraic elimination procedure that enables the direct solution of the algebraic (self-)intersection constraints.

All algorithms have been fully implemented and tested. Examples are presented for applications in self-intersection detection, self-intersection-free metamorphosis of curves, and proper trimming of self-intersections in offset curves.

Keywords: Global/local self-intersection; Surface binormal line; Antipodal points; Multivariate polynomial constraints; Offset trimming; Self-intersection-free metamorphosis

1. Introduction

The need for computing intersections of freeform parametric curves and surfaces can be found in many operations in computer graphics, geometric and solid modeling, and computer-aided design. Hoffmann [10] listed the intersection problem as one of the most fundamental problems in the integration of geometric and solid modeling systems. An extensive body of research can be found on intersection problem, with numerous articles presenting different approaches for the intersection of freeform curves (and surfaces).

Nevertheless, only a few results address the problem of *self-intersections* of freeform curves, much less in the case of surfaces. Curve self-intersection detection and elimination is a crucial problem, for example, in the computation of offsets for NC machining and robot navigations [11]. Once the shape's offset goes beyond the local radius-of-curvature, a self-intersection is bound to occur in the offset. Other applications could benefit from proper self-intersection detection and elimination as well. Another example, demonstrated herein, is the problem of creating a self-intersection-free metamorphosis between freeform curves.

The intersection of two surfaces can be topologically complex in general, comprising several univariate components that we will refer to as closed loops and open branches. In the intersection of two freeform curves, no such components are present and it is clear that the intersection of two freeform curves is a considerably simpler task. In order to detect the endpoints of an open branch in the surface–surface intersection case, one can simply intersect the boundary curves of one surface with the other surface. On the other hand, the detection of closed loops has been a major challenge in the surface intersection algorithms.

Sinha et al. [23] presented a parallel normal criterion for guaranteeing the detection of closed loops in the intersection of two surfaces. The criterion states that if two surfaces intersect in a closed loop, at least one line that is normal to one surface must be parallel to a line that is normal to the other surface. The drawback of the parallel normal criterion is that there are, in general, an infinite number of parallel normal lines within a closed loop. The same criterion can be applied to the intersection of two planar curves. Sederberg and Meyers [20] presented a similar criterion based on cones. The criterion states that if a cone that tightly bounds all the normal vectors on one

* Corresponding author. Tel.: +972 54 5741700.
E-mail address: d.pekerman@samsung.com (D. Pekerman).
[1] Currently working at Samsung Telecom. Research Israel.

0010-4485/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2007.10.004

surface lies outside a cone that tightly bounds all the normal vectors on the other surface, no closed loops exist.

Sederberg and Meyers [20] noted that their loop detection test has been observed to require a considerable amount of computation time for some straightforward cases. In addition, they reasoned that if two space curves intersect at all, while their bounding cones do not overlap, they intersect exactly once.

Finally, a decisive collinear normal criterion for loop detection was presented by Sederberg et al. [21]: if two surfaces intersect in a closed loop and their normal vectors do not deviate more than 90°, then there exists a line that is perpendicular to both surfaces. In contrast to the parallel normal criterion of Sinha et al. [23], there are only a finite number of lines that are normal to the two surfaces and each loop in the intersection must encircle at least one collinear normal line. While Sederberg et al.'s [21] theory for a loop detection method was correct, an efficient implementation has thus far eluded the authors.

Ho and Cohen [2] defined self-intersection as a global intrinsic property of the geometry and introduced a necessary condition for surface self-intersection that can be computed from the normal and tangent bounding cones of the surface. Using this condition, the authors developed a divide-and-conquer algorithm to find the self-intersection curves of the surface.

Galligo and Pavone [9] presented two different contributions to the determination of a self-intersection locus for a Bézier bicubic surface. The first one uses a specific sparse resultant and produces an implicit equation of a plane projection of this locus. The second one accurately computes the coordinates of critical points on this locus, by solving a system of four polynomial equations in four variables, derived from a previously computed plane projection of the self-intersection locus.

In Maekawa and Patrikalakis [17], the observation is made that the term $(t-r)$ always exists in the self-intersection (offset) expression $C(t) - C(r) = 0$. A numeric elimination procedure to remove the term $(t-r)$ is presented. The procedure poses the problem as a set of linear constraints in the coefficients of the lower order's reduced expression. Given that this observation is crucial to being able to efficiently find self-intersection locations, in this work, we take a similar approach but show a direct closed-form elimination procedure for the same task, and we extend the result to B-spline curves.

It is a nontrivial task to detect and trim all the local and global self-intersections in offset curves [5,13]. Lee et al. [13] applied a plane sweep algorithm to detect all self-intersections of planar offset curves. Elber and Cohen [5] detected local self-intersections of offset curves by checking whether the tangent field of the curve, $C(t)$, and its offset, $O_d(t)$, have opposite directions.

The plane sweep approach is difficult to implement, and the tangent field approach of [5] is limited to detecting local self-intersections only. In a more recent work, Elber [7] and Seong et al. [22] presented a scheme to trim both local and global self-intersections of offset curves and surfaces. The scheme is based on the derivation of an analytic distance map between the original curve/surface and its offset. By solving one/three bivariate/fourvariate polynomial equations for an offset curve/surface, respectively, all the local and global self-intersection regions in the offset curves/surfaces can be identified. The trimming of these regions is completed by projecting the zero set of polynomial equation(s) into the desired parametric domain. The proposed scheme is reasonably efficient and robust.

The problem of trimming self-intersections in offset surfaces is considerably more difficult to solve [1,2,16,18,24,25]. Ho and Cohen [2] introduced a trimming algorithm that is based on a necessary condition for a freeform surface to have self-intersections. This approach works for general surfaces. However, it is not an optimal solution for offset surfaces since no special consideration of the relationship between the original surface and its offset is taken into account.

Wang [25] proposed an algorithm to compute the intersection curve between two offset surfaces. The method is based on the concept of normal projection. The intersection is represented in the parameter spaces of the base surfaces and no offset surface approximation is needed. This algorithm can deal with global self-intersections only. Aomura and Uehara [1] presented a similar approach based on numerical integration starting from random initial points. Nevertheless, this method does not guarantee the detection of all the components of self-intersections.

Maekawa et al. [18] presented a method for tracing self-intersection loops in the parameter domain. In their method, starting points are computed by solving a system of nonlinear polynomial equations; nonetheless, they are solving five equations in five variables and their algorithm requires special treatment for trivial solutions. Wallner et al. [24] considered the problem of computing the maximum offset distance that guarantees no local or global self-intersections. While all these schemes strove for precise self-intersection trimming, in this work, we present an alternative approach that eliminates the self-intersection by altering the shape itself.

Samoilov and Elber [19] introduced two new methods for eliminating self-intersections in freeform curve metamorphosis. Both their algorithms exploit the matching algorithm of Cohen et al. [3]. The first algorithm investigates building and employing a homotopy between the two original curves, which is a composition of a ruled surface with an appropriate subjective continuous function that causes the curve of homotopy to be self-intersection-free. The second algorithm constructs the best correspondence of the relative parameterizations of original curves. Then, it eliminates the remaining self-intersections using the flipping algorithm. Nonetheless, the presented algorithms in [19] can be applied only on one particular kind of freeform curve metamorphosis: a ruled surface, which is the result of a simple linear blend of the two curves.

To the best of our knowledge, no methods are known in the literature that can efficiently detect self-intersections in arbitrary metamorphosis of freeform curves and then use a flipping scheme to eliminate the detected self-intersections. In addition, no algebraic method was presented to eliminate $(u - v)$ terms, in self-intersections. With these contributions,

Please cite this article in press as: Pekerman D, et al. Self-intersection detection and elimination in freeform curves and surfaces. Computer-Aided Design (2007), doi:10.1016/j.cad.2007.10.004

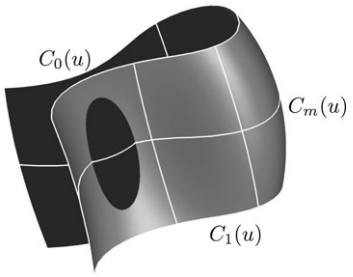

Fig. 1. A globally self-intersecting surface with a closed-loop intersection component.

the rest of this paper is organized as follows. Section 2 presents the detection algorithm, which uses a binormal-line criterion for global self-intersection detection in freeform surfaces. In Section 3, we present the global self-intersection elimination algorithm for surfaces, that is based on the shape alteration. In Section 4, we present an alternative, algebraic, approach to the self-intersection detection of planar curves. Section 5 provides some examples of offsets and metamorphosis of freeform curves, which employ the proposed algorithms for both detecting and/or eliminating the self-intersections. Finally, this paper is concluded in Section 6.

2. Global self-intersection detection using binormal lines

One way of representing the metamorphosis between two univariate shapes, $C_0(u)$ and $C_1(u)$, $u \in [u_0, u_1]$, is by introducing, a second parameterization for time as $S(u, t)$. Let $S(u, t)$, $(u, t) \in [u_0, u_1] \times [0, 1]$, denote a regular surface that represents a locally self-intersection-free metamorphosis between two simple, regular, planar freeform curves, $C_0(u)$ and $C_1(u)$. (A surface is locally self-intersection free if its Jacobian is never singular.)

In contrast, $S(u, t)$ can globally self-intersect. Global self-intersection of $S(u, t)$ can be composed of several components in the parametric domain of the surface: open branches, closed loops (see Fig. 1) and singular cases (i.e., tangency conditions with closed loops of zero size). If the curves, $C_0(u)$ and $C_1(u)$, are open, the end points of an open branch self-intersection are easy to detect by intersecting the surface $S(u, t)$ with its boundary curves $S(u_0, t)$ and $S(u_1, t)$. In contrast, the detection of a closed-loop intersection is a more complex task. Fig. 1 illustrates an example of a surface that represents a metamorphosis between two open curves, $C_0(u)$ and $C_1(u)$, with a closed-loop self-intersection component.

While we pose the problem using the metamorphosis application, the presented algorithms are all general. We start with an algorithm for loop detection in the self-intersection of freeform curves and surfaces. We first define some terms to be used in this section:

Definition 2.1. A **surface normal-line** at the parameter (u_1, t_1) is a line through $S(u_1, t_1)$ that is parallel to the surface normal, $N_S(u_1, t_1) = \frac{\partial S(u_1,t_1)}{\partial u} \times \frac{\partial S(u_1,t_1)}{\partial t}$.

Definition 2.2. The line through $S(u_1, t_1)$ and $S(u_2, t_2)$ is denoted as a **surface binormal line** at parameters (u_1, t_1) and (u_2, t_2), where $(u_1, t_1) \neq (u_2, t_2)$, if it is a surface normal line at both $S(u_1, t_1)$ and $S(u_2, t_2)$.

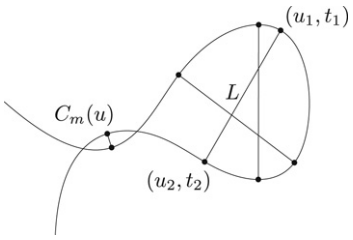

Fig. 2. The iso-parametric curve of the surface seen in Fig. 1, shown with its binormal lines.

Definition 2.3. Let L be a surface binormal line of a surface $S(u, t)$ at two different parameters (u_1, t_1) and (u_2, t_2), such that $\langle N_S(u_1, t_1), N_S(u_2, t_2)\rangle < 0$. Then, $S(u_1, t_1)$ and $S(u_2, t_2)$ are denoted **antipodal points** of the surface $S(u, t)$.

Fig. 2 presents the middle iso-parametric curve, $C_m(u)$, of the surface from Fig. 1 that contains the binormal lines with antipodal points.

We make several assumptions regarding the given surface $S(u, t)$. First, we assume that $S(u, t)$ is locally self-intersection-free. Second, we assume that if the surface $S(u, t)$ has a global self-intersection at some location, then there are exactly two different points in the parametric domain $(u_1, t_1) \neq (u_2, t_2)$ that are mapped to the each Euclidean location. Finally, we assume that if $S(u, t)$ has global self-intersections, then the normal directions of the surface inside each loop component do not vary by more than 90°, and the normal field of the whole surface $S(u, v)$ has a span of less than 360°. The requirement for less than 360° is to prevent from considering surfaces that spiral on themselves (like a surface of revolution that revolves more than 360°).

Denote the loop intersection components as $\mathcal{C}_i, i = 1, 2$, and the two regions in the parametric domain of $S(u, t)$ that is interior to $\mathcal{C}_i$ as $Int(\mathcal{C}_i)$, $i = 1, 2$.

Then, under the last assumption, any normal-line at $(u, t) \in Int(\mathcal{C}_i), i = 1, 2$, intersects $S(u, t)$ exactly once in its domain $Int(\mathcal{C}_i), i = 1, 2$. In addition, any normal-line at $(u, t) \in Int(\mathcal{C}_i), i = 1, 2$, intersects $S(u, t)$ exactly once in the *antipodal domain*, $Int(\mathcal{C}_{3-i})$ (see [21] for a proof).

Lemma 2.4. *If a surface $S(u, t)$ intersects itself in the pair of closed loops, $\mathcal{C}_1$ and $\mathcal{C}_2$, then there must exist a binormal line at two antipodal points (u_1, t_1), (u_2, t_2) in the parameter domain of the surface, where $(u_1, t_1) \in Int(\mathcal{C}_1)$ and $(u_2, t_2) \in Int(\mathcal{C}_2)$.*

Proof. In the proof of this lemma, we use the "Brouwer fixed point theorem" (see [26]): every continuous mapping f of a closed n-ball of radius one to itself has a fixed point. The "Brouwer fixed point theorem" also holds for any closed region that allows a parameterization by a closed n-ball of radius one.

We apply the "Brouwer fixed point theorem" to the parameter domain of the surface $S(u, t)$, where the closed region is $\overline{Int(\mathcal{C}_1)}$. Consequently, every continuous mapping $f : \overline{Int(\mathcal{C}_1)} \to \overline{Int(\mathcal{C}_1)}$ has a fixed point.

Please cite this article in press as: Pekerman D, et al. Self-intersection detection and elimination in freeform curves and surfaces. Computer-Aided Design (2007), doi:10.1016/j.cad.2007.10.004

From our last assumption above, we know that a normal-line at $(u_1, t_1) \in Int(\mathcal{C}_1)$ intersects the surface at a single parameter $(u_2, t_2) \in Int(\mathcal{C}_2)$. The same holds in the opposite direction—namely, a normal-line at $(u_2, t_2) \in Int(\mathcal{C}_2)$ intersects the surface at a single parameter $(u_3, t_3) \in Int(\mathcal{C}_1)$. Thus, we match (u_1, t_1) to (u_3, t_3), and denote such a continuous mapping from $Int(\mathcal{C}_1)$ to itself as f.

The "Brouwer fixed point theorem" guarantees the existence of a fixed point $(u_1, t_1) \in Int(\mathcal{C}_1)$. The normal-line at this fixed point, which intersects the surface $(u_2, t_2) \in Int(\mathcal{C}_2)$, is a binormal line with two antipodal points, (u_1, t_1) and (u_2, t_2). ■

Clearly, Lemma 2.4 is a fundamental and important part in any global self-intersection detection algorithm. The self-intersection detection problem is now reduced to determining all the antipodal points in the parametric domain of $S(u, t)$. In the next section, Section 2.1, we explain how to detect the binormal lines with antipodal points and show a tractable way of computing them for a given surface $S(u, t)$.

2.1. Computing the binormal lines of a surface

The problem of detecting the surface's binormal lines could be reduced to solving the following system of five constraints:

$$\begin{aligned}
&\text{(a)} \quad \left\langle S(u,t) - S(v,s), \frac{\partial S(u,t)}{\partial u} \right\rangle = 0,\\
&\text{(b)} \quad \left\langle S(u,t) - S(v,s), \frac{\partial S(u,t)}{\partial t} \right\rangle = 0,\\
&\text{(c)} \quad \left\langle S(u,t) - S(v,s), \frac{\partial S(v,s)}{\partial v} \right\rangle = 0,\\
&\text{(d)} \quad \left\langle S(u,t) - S(v,s), \frac{\partial S(v,s)}{\partial s} \right\rangle = 0,\\
&\text{(e)} \quad \langle N_S(u,t), N_S(v,s) \rangle < 0, \qquad (1)
\end{aligned}$$

with four unknowns, u, t, v, s, and recalling that $N_S(u, v) = \frac{\partial S(u,v)}{\partial u} \times \frac{\partial S(u,v)}{\partial v}$.

Notice that constraints (a) and (b) in Eq. (1) seek surface normal-line at the parameter location (u, t), while (c) and (d) seek surface normal-line at the parameter location (v, s). Any attempt to employ just the first four constraints, (a)–(d), in Eq. (1) is futile, as the entire parameter domain of S satisfies these equations for $(u, t) = (v, s)$. To remedy this situation, we add (e), which constrains the normal of $S(u, t)$ to be in the opposite direction to the normal of $S(v, s)$, seeking the antipodal points, with an opposite orientation. This fifth constraint ensures that our solutions set will contain no points of the form $(u, t) = (v, s)$, since at such points the normals are obviously identical.

Remark. Constraint (e) could be replaced by

$$\text{(ê)} \quad \langle N_S(u,t), N_S(v,s) \rangle = -1.$$

However, this replacement would produce five (redundant) equations in four variables. Thus, we employ the constraint (e) instead of an equivalent constraint (ê), for the sake of numerical stability in solving the system of equations.

The system of constraints given in Eq. (1) is still numerically unstable, since the value of $S(u, t) - S(v, s)$ vanishes in the first four constraints, at any global self-intersection point (for which $(u, t) \neq (v, s)$). Thus, we replace two of the constraints by an equivalent, numerically more stable, conditions as follows:

$$\begin{aligned}
&\text{(ā)} \quad \left\langle S(u,t) - S(v,s), \frac{\partial S(u,t)}{\partial u} \right\rangle = 0,\\
&\text{(b̄)} \quad \left\langle S(u,t) - S(v,s), \frac{\partial S(u,t)}{\partial t} \right\rangle = 0,\\
&\text{(c̄)} \quad \left\langle N_S(v,s), \frac{\partial S(u,t)}{\partial u} \right\rangle = 0,\\
&\text{(d̄)} \quad \left\langle N_S(v,s), \frac{\partial S(u,t)}{\partial t} \right\rangle = 0,\\
&\text{(ē)} \quad \langle N_S(u,t), N_S(v,s) \rangle < 0. \qquad (2)
\end{aligned}$$

That is, now only Equations (ā) and (b̄) vanish at the global self-intersection points. In constraints (c̄) and (d̄), the value $S(u, t) - S(v, s)$ is replaced by the normal to the surface that never vanishes in the case of a regular surface. In addition, at a global self-intersection that is transversal and $(u, t) \neq (v, s)$, $N_S(v, s)$ is not orthogonal to $\frac{\partial S(u,t)}{\partial u}$. In other words, (c̄) is not vanishing at transversal self-intersections. This is clearly more efficient and more robust, as we seek antipodal points and no time/effort will be spent on finding global self-intersections that we do not seek.

The presented set of Eq. (2) forms the multivariate rational spline function constraints. To find the solution, we apply the geometric constraint solver [6], and select a subset of antipodal points inside the loop components, by examining the orientations of the surface at the two antipodal locations. Let $S(u_0, t_0)$ and $S(v_0, s_0)$ be two antipodal points. The vector $S(u_0, t_0) - S(v_0, s_0)$ flips its direction when these two locations penetrate each other. By comparing this direction to the surface normals at $S(u_0, t_0)$ and $S(v_0, s_0)$, arbitrarily small intersections can be isolated.

Armed with the ability to detect the existence of self-intersections, we will present, in the next section, a heuristic approach that eliminates self-intersections.

3. Global self-intersection elimination by shape alteration

Let $S(u, t)$ be a surface with some closed-loop intersection components, and let $\{(u_i, t_i, v_i, s_i)\}_{i=1}^{N}$ be the set of N antipodal points of $S(u, t)$ that lie inside the closed-loop components, as described in Section 2. While many of the existing published results, as presented in Section 1, strive to detect the exact self-intersection curve and eliminate the spurious components, in this section, we present an algorithm that attempts to eliminate all loop intersection components in $S(u, t)$ by shape alteration, seeking a surface $\widehat{S}(u, t)$ that is close to $S(u, t)$, but globally self-intersection-free.

We begin by introducing a flipping constraint over $\widehat{S}$ between a pair of antipodal points (u, t) and (v, s) as follows:

$$\widehat{S}(u,t) = S(v,s), \quad \widehat{S}(v,s) = S(u,t). \qquad (3)$$

Please cite this article in press as: Pekerman D, et al. Self-intersection detection and elimination in freeform curves and surfaces. Computer-Aided Design (2007), doi:10.1016/j.cad.2007.10.004

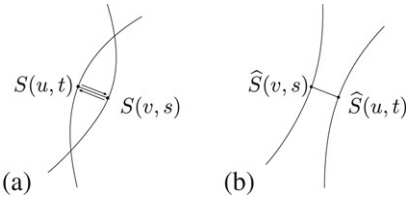

Fig. 3. (a) Two curves containing the antipodal points; (b) the resulting curves after applying the flipping constraints.

In other words, we introduce a pair of constraints that coerce the new surface $\widehat{S}$ to flip the two positions of the antipodal locations. Fig. 3 illustrates the result of the flipping operation between the two antipodal points defined in Eq. (3), which leads to the elimination of the intersection loop.

To solve the flipping constraint, we consider the minimization of the following functional, over $\widehat{S}$:

$$\sum_{i=1}^{N} \alpha_i \left\| \widehat{S}(u_i, t_i) - S(v_i, s_i) \right\|^2 + \sum_{i=1}^{N} \beta_i \left\| \widehat{S}(v_i, s_i) - S(u_i, t_i) \right\|^2. \quad (4)$$

In the two summations in Eq. (4), we seek a new surface $\widehat{S}(u,t)$ that satisfies the flipping constraints (Eq. (3)) for all the N antipodal points. Note the weights, α_i and β_i, $1 \le i \le N$, are introduced to enable asymmetrical flipping. The case of $\alpha_i = \beta_i = \frac{1}{2}$ provides for the symmetric application of the pair of constraints.

In order to find the target surface $\widehat{S}(u,t)$ that is close to $S(u,t)$, in the L^2-norm sense, one should also minimize a similarity functional, such as:

$$\int_{t_0}^{t_1} \int_{u_0}^{u_1} \left\| \widehat{S}(u,t) - S(u,t) \right\|^2 \mathrm{d}u\mathrm{d}t. \quad (5)$$

For the application of metamorphosis between two curves, one can combine the minimization of the functionals defined in Eqs. (4) and (5) with the additional constraint that surface $\widehat{S}(u,t)$ must equate with $C_0(u)$ and $C_1(u)$ for $\widehat{S}(u,0)$ and $\widehat{S}(u,1)$, respectively.

In sum, the optimization problem that should be solved is:

minimize

$$\sum_{i=1}^{N} \alpha_i \left\| \widehat{S}(u_i, t_i) - S(v_i, s_i) \right\|^2 + \sum_{i=1}^{N} \beta_i \left\| \widehat{S}(v_i, s_i) - S(u_i, t_i) \right\|^2 + \int_{t_0}^{t_1} \int_{u_0}^{u_1} \left\| \widehat{S}(u,t) - S(u,t) \right\|^2 \mathrm{d}u\mathrm{d}t$$

subject to

$$\widehat{S}(u, t_0) = S(u, t_0) = C_0(u),$$
$$\widehat{S}(u, t_1) = S(u, t_1) = C_1(u). \quad (6)$$

Employing the B-spline representation of the surface $\widehat{S}(u,t)$, one can verify that the optimization problem, defined in Eq. (6) is actually a linearly constrained quadratic optimization (see [14]). Thus, it can be solved to find a local optimal solution by one of the known solvers, e.g. the "optimization toolbox" in [15].

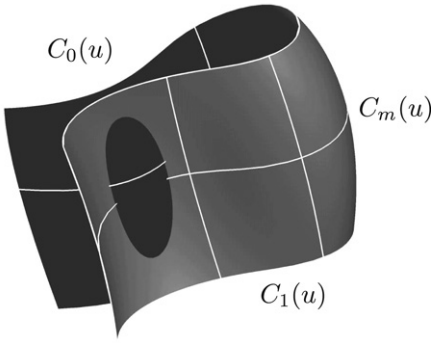

Fig. 4. Globally self-intersection-free intermediate curves of a globally self-intersecting metamorphosis surface with a closed-loop intersection components.

In general, the solution surface $\widehat{S}(u,t)$ can still have loop intersection components because it is only locally optimal. In this case, one can again apply the test of the binormal-line criterion from Section 2 to find the antipodal points of the obtained surface $\widehat{S}(u,t)$ and solve the optimization problem in Eq. (6) for the new surface $\widehat{S}(u,t)$. To achieve a globally self-intersection-free surface, one can continue by applying the above algorithm gradually. Unfortunately, there is no guarantee that this optimization procedure will terminate.

Clearly, if surface $S(u,t)$ is globally self-intersection-free, its iso-parametric curves' $S(u,t_0)$, t_0 constant, will all be globally self-intersection-free as well. However, a globally self-intersecting metamorphosis surface $S(u,t)$ does not guarantee the existence of a self-intersecting iso-parametric curve $S(u,t_0)$, as illustrated in Fig. 4. (See the globally self-intersection-free intermediate curve, denoted as $C_m(u)$.)

The following lemma presents a sufficient condition for a globally self-intersecting surface to possess self-intersecting iso-curves:

Lemma 3.1. *Let $S(u,t)$ be a globally self-intersecting surface. If the iso-parametric curves $S(u,t_0)$, t_0 constant, are all planar, $S(u,t)$ holds self-intersecting iso-parametric curves.*

Proof. Let $\mathcal{C}_i, i = 1, 2$, be loop intersection components of a surface $S(u,t)$. Let $t_{\min}$ and $t_{\max}$ denote the minimal and maximal times on the loop $\mathcal{C}_1$. Then, $\forall t_0 \in (t_{\min}, t_{\max})$, the planarity of $S(u,t_0)$ implies the existence of (at least) two parameters u_1 and u_2 (without loss of generality, assume $u_1 \le u_2$), such that (u_1, t_0) and $(u_2, t_0) \in \mathcal{C}_1$, and two corresponding parameters (u_3, t_0) and $(u_4, t_0) \in \mathcal{C}_2$ (without loss of generality, assume $u_3 \le u_4$), such that $S(u_1, t_0) = S(u_4, t_0)$ and $S(u_2, t_0) = S(u_3, t_0)$. Thus, there are some iso-parametric curves of $S(u,t)$ that have self-intersections. ■

While $S(u,t)$ can contain only planar iso-parametric curves, nothing on the optimization function of $\widehat{S}(u,t)$, Eq. (6), ensures that $\widehat{S}(u,t)$ will contain only planar iso-parametric curves as well. Here, one can either make sure all flips are performed

Please cite this article in press as: Pekerman D, et al. Self-intersection detection and elimination in freeform curves and surfaces. Computer-Aided Design (2007), doi:10.1016/j.cad.2007.10.004

in the same plane, and hence preserve the planarity of all iso curves, or, a one-to-one projection of all iso-parametric curves to some plane must be established, if one exists, following Elber [8] for example.

While a scheme to eliminate the self-intersection by altering the shape was presented in this section, an alternative use to the antipodal points would be to subdivide the shape at all antipodal points and apply regular curve–curve or surface–surface intersection algorithms between all the remaining pieces, that would now contain interior self-intersections no longer. A curve will simply be divided at the antipodal locations. A surface will be divided along an iso-parametric curve through the antipodal point. For the additional details about the presented schemes see [4].

Before we present some results and examples of using the binormal line to detect self-intersections and the flipping scheme to eliminate them in Section 5, we introduce, in Section 4, another scheme to derive the self-intersections of a planar curve.

4. Algebraic self-intersection detection in planar curves

In Section 2, we introduced a method to detect the existence of global self-intersections in curves and surfaces via the existence of binormal lines. Recall now the direct approach to finding the intersection locations of two different curves:

$$\begin{aligned} x_0(u) &= x_1(v), \\ y_0(u) &= y_1(v), \end{aligned} \tag{7}$$

where (x_i, y_i) are the x and y components of C_i, $i = 0, 1$. Having two constraints in two unknowns, Eq. (7) can be solved to find all the intersection locations of $C_0(u)$ and $C_1(v)$. However, consider now these constraints in the context of a self-intersection query. Parameterizing C as both $C(u)$ and $C(v)$, where u and v are two independent parameters, yields the following

$$\begin{aligned} x(u) &= x(v), \\ y(u) &= y(v). \end{aligned} \tag{8}$$

Clearly, all locations for which $(u = v)$ satisfy Eq. (8) and hence we need to handle all redundant solutions along the diagonal $u = v$. In fact, these pairs of constraints contain the redundant algebraic component of $(u - v)$. Assume C is a polynomial curve, $C(t) = \sum_i a_i t^i$. Then, we prove the observation presented in [17],

Lemma 4.1. *Let u and v be two independent parameters of the same polynomial curve C. Then, $C(u) - C(v)$ contains the term $(u - v)$.*

Proof.

$$\begin{aligned} C(u) - C(v) &= \sum_{i=0}^{n} a_i u^i - \sum_{i=0}^{n} a_i v^i \\ &= \sum_{i=0}^{n} a_i (u^i - v^i) \end{aligned} \tag{9}$$

and $(u^i - v^i)$ is known to hold $(u - v)$ for all $i > 0$ and to vanish for $i = 0$. ■

While [17] computes the reduced expression by numerically solving a set of linear constraints, and finding the coefficients of the degree-reduced bivariate expression, we now present a direct closed-form scheme to do the same task. Let $\mathcal{I}(u, v) = C(u) - C(v)$. We know that $\mathcal{I}$ holds a redundant term of the form $u - v$. In the interest of removing all solutions along the diagonal, we seek to eliminate $(u - v)$ from $\mathcal{I}$. Given a bivariate Bézier function $\mathcal{I}(u, v)$ that is assumed to hold the factor of $(u - v)$, we seek to find the Bézier bivariate function $\widehat{\mathcal{I}}(u, v)$ (that exists under our assumption) such that

$$(u - v)\widehat{\mathcal{I}}(u, v) = \mathcal{I}(u, v).$$

In the Bézier basis function terms, this reduces to

$$\begin{aligned} &(u - v) \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} p_{ij} \theta_i^{m-1}(u) \theta_j^{n-1}(v) \\ &= \sum_{k=0}^{m} \sum_{l=0}^{n} q_{kl} \theta_k^m(u) \theta_l^n(v), \end{aligned} \tag{10}$$

where θ_k^m is the kth Bézier basis function of order m.

Lemma 4.2. *Let $\mathcal{I}(u, v) = \sum_{k=0}^{m} \sum_{l=0}^{n} q_{kl} \theta_k^m(u) \theta_l^n$ be a bivariate polynomial in the Bézier basis functions known to contain a factor of $(u - v)$. Then, $\mathcal{I}$ could be represented as $(u - v)\widehat{\mathcal{I}}(u, v)$ where $\widehat{\mathcal{I}}(u, v) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} p_{ij} \theta_i^{m-1}(u) \theta_j^{n-1}(v)$ and*

$$\begin{aligned} p_{0,j-1} &= -\frac{n}{j} q_{0j}, \\ p_{i,j-1} &= \frac{i}{j} \frac{n-j}{m-i} p_{i-1,j} - \frac{n}{j} \frac{m}{m-i} q_{ij}, \quad i > 0. \end{aligned}$$

Proof.

$$\begin{aligned} &(u - v) \sum_{i=0}^{m-1} \sum_{j}^{n-1} p_{ij} \theta_i^{m-1}(u) \theta_j^{n-1}(v) \\ &= \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} p_{ij} u \theta_i^{m-1}(u) \theta_j^{n-1}(v) \\ &\quad - \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} p_{ij} \theta_i^{m-1}(u) v \theta_j^{n-1}(v) \\ &= \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} p_{ij} \binom{m-1}{i} u^{i+1} (1-u)^{m-1-i} \theta_j^{n-1}(v) \\ &\quad - \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} p_{ij} \theta_i^{m-1}(u) \binom{n-1}{j} v^{j+1} (1-v)^{n-1-j} \\ &= \sum_{i=1}^{m} \sum_{j=0}^{n-1} p_{i-1,j} \binom{m-1}{i-1} u^i (1-u)^{m-i} \theta_j^{n-1}(v) \\ &\quad - \sum_{i=0}^{m-1} \sum_{j=1}^{n} p_{i,j-1} \theta_i^{m-1}(u) \binom{n-1}{j-1} v^j (1-v)^{n-j} \end{aligned}$$

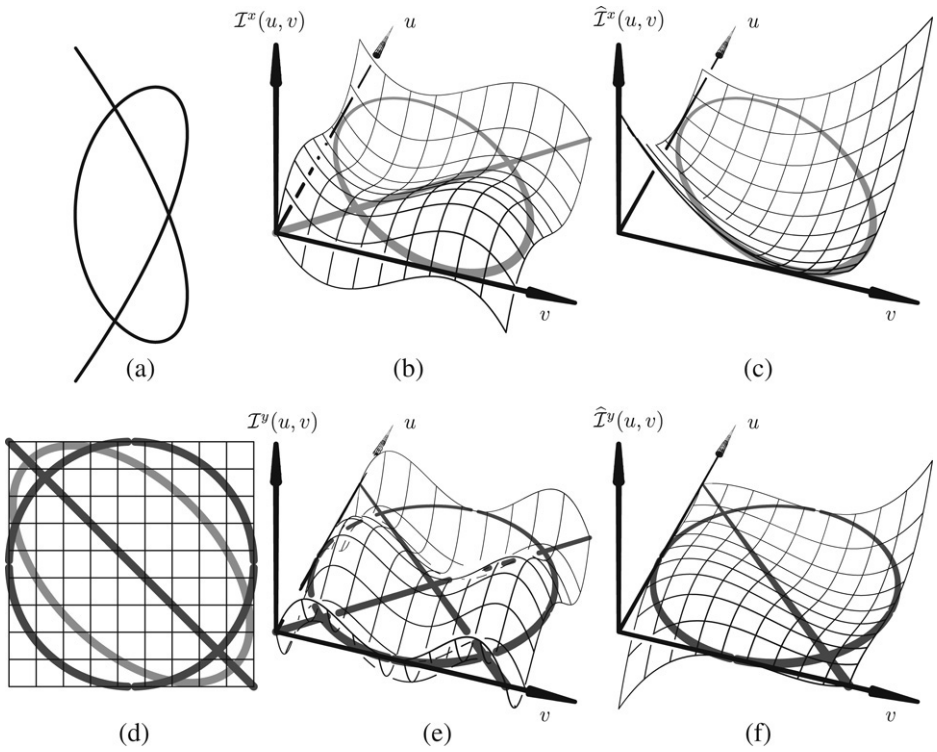

Fig. 5. The quartic Bézier curve $C(t) = (x(t), y(t))$ in (a) yields the constraints $\mathcal{I}^x(u, v) = x(u) - x(v)$ in (b) and the constraints $\mathcal{I}^y(u, v) = y(u) - y(v)$ in (e), when attempting to solve for its self-intersection locations. The entire diagonal of $(u = v)$ shows up in both $\mathcal{I}^x$ and $\mathcal{I}^y$ and hence renders the solution difficult. By eliminating the term (u, v) from both $\mathcal{I}^x$ and $\mathcal{I}^y$, $\widehat{\mathcal{I}}^x$ (in (c)) and $\widehat{\mathcal{I}}^y$ (in (f)) are easily solved for the three self-intersection locations of $C(t)$. Finally, (d) presents the superposition of $\widehat{\mathcal{I}}^x$ and $\widehat{\mathcal{I}}^y$. The zero sets of both functions are shown in thick gray color.

$$
\begin{aligned}
&= \sum_{i=1}^{m}\sum_{j=0}^{n-1} p_{i-1,j}\frac{i}{m}\theta_i^m(u)\theta_j^{n-1}(v)\\
&\quad - \sum_{i=0}^{m-1}\sum_{j=1}^{n} p_{i,j-1}\theta_i^{m-1}(u)\frac{j}{n}\theta_j^n(v)\\
&= \sum_{i=0}^{m}\sum_{j=0}^{n-1} \frac{i}{m}p_{i-1,j}\theta_i^m(u)\theta_j^{n-1}(v)\\
&\quad - \sum_{i=0}^{m-1}\sum_{j=0}^{n} \frac{j}{n}p_{i,j-1}\theta_i^{m-1}(u)\theta_j^n(v)\\
&=^{\dagger} \sum_{i=0}^{m}\sum_{j=0}^{n} \frac{i}{m}\left(\frac{j}{n}p_{i-1,j-1} + \frac{n-j}{n}p_{i-1,j}\right)\theta_i^m(u)\theta_j^n(v)\\
&\quad - \sum_{i=0}^{m}\sum_{j=0}^{n} \frac{j}{n}\left(\frac{i}{m}p_{i-1,j-1} + \frac{m-i}{m}p_{i,j-1}\right)\theta_i^m(u)\theta_j^n(v)\\
&= \sum_{i=0}^{m}\sum_{j=0}^{n}\left(\frac{i}{m}\frac{n-j}{n}p_{i-1,j} - \frac{j}{n}\frac{m-i}{m}p_{i,j-1}\right)\theta_i^m(u)\theta_j^n(v),
\end{aligned}
$$

where the step † employs Bézier degree raising.

Hence,

$$q_{ij} = \frac{i}{m}\frac{n-j}{n}p_{i-1,j} - \frac{j}{n}\frac{m-i}{m}p_{i,j-1}.$$

Deduce $p_{ij}, \forall j$ such that $i = 0$ as

$$p_{0,j-1} = -\frac{nm}{j(m-0)}q_{0j} = -\frac{n}{j}q_{0j},$$

and then deduce the rest of p_{ij}, one row at a time. ■

Eliminating the factor $(u - v)$ tremendously improves the performance of finding the self-intersection. By directly solving the equation $\widehat{\mathcal{I}}(u, v) = 0$, no computation time or effort is spent on the diagonal, simply because $\widehat{\mathcal{I}}(u, v)$ no longer contains zeros along the $(u = v)$ diagonal. Fig. 5 presents one complete example.

Being able to efficiently solve the self-intersection problem for the polynomial (Bézier) case by eliminating $(u - v)$, one should now consider the piecewise-polynomial extension and examine whether a similar approach could be used for B-spline curves as well. Interestingly enough, the problem in the piecewise-polynomial domain is not that much more difficult. Consider $C(t)$, a planar B-spline curve, and consider the solutions of the vector equation $\mathcal{J}(u, v) = (C(u) - C(v)) = 0$. $\mathcal{J}(u, v)$ now holds a pair of scalar bivariate B-spline functions (for x and y), each of which is symmetric along the diagonal. Subdivide $\mathcal{J}(u, v)$ at all its interior domains into polynomial patches $\mathcal{I}_{kl}$. Let the domain of $\mathcal{I}_{kl}$ be $(u, v) \in [u_k, u_{k+1}] \times [v_l, v_{l+1}]$. The following cases can then occur for a polynomial patch $\mathcal{I}_{kl}$ of $\mathcal{J}$:

- $k = l$, or the patch is on the diagonal. Then, solve for the polynomial case, using Lemma 4.2.
- $|k - l| = 1$, $\mathcal{I}_{kl}$ now represent two adjacent regions of the curve. Solve using regular curve–curve intersection techniques between $C(u), u \in [u_k, u_{k+1}]$ and $C(u), u \in [u_l, u_{l+1}]$, while ignoring/purging the solution point at the shared boundary location.

Please cite this article in press as: Pekerman D, et al. Self-intersection detection and elimination in freeform curves and surfaces. Computer-Aided Design (2007), doi:10.1016/j.cad.2007.10.004

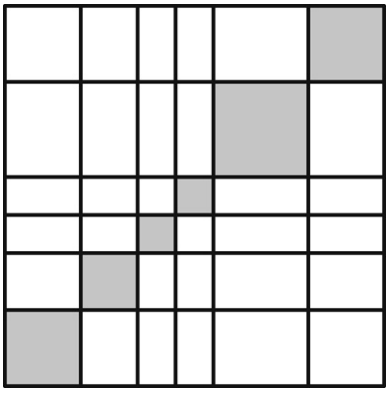

Fig. 6. The polynomial domains of $\mathcal{J}(u, v)$ for some B-spline curve $C(t)$, after dividing at all internal knots of $C(t)$. Only the diagonal domains, $\mathcal{I}_{kk}$, require the elimination of the $(u - v)$ terms.

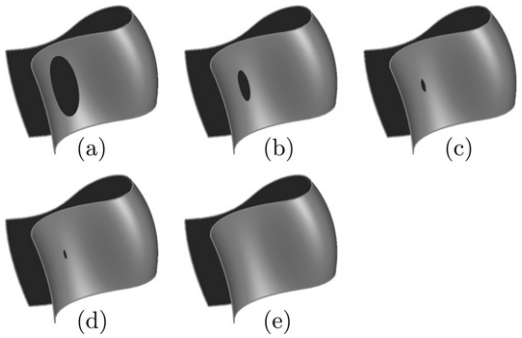

Fig. 7. A sequence of surfaces, acquired from the global self-intersection elimination algorithm, resulting in a self-intersection-free surface. Note that the top and bottom boundary curves are unmodified.

- $|k - l| > 1$, $\mathcal{I}_{kl}$ now represent two independent regions of the curve. Solve using regular curve–curve intersection techniques between $C(u), u \in [u_k, u_{k+1}]$ and $C(u), u \in [u_l, u_{l+1}]$.

In other words, all polynomial patches on the diagonal, $\mathcal{I}_{kk}$, will be solved in reduced form, following Lemma 4.2, whereas all other, off-diagonal patches will be treated regularly. Fig. 6 shows one example of the polynomial domains of the $\mathcal{J}(u, v)$ bivariate derived from a B-spline curve, with the diagonal domains highlighted in light gray.

In the next section, we present examples for all the algorithms introduced in this work, namely, detection of global self-intersections using the binormal line introduced in Section 2, self-intersection elimination using flipping in Section 3, and finding self-intersection locations via the $(u - v)$ elimination.

5. Examples

The first example of global self-intersection elimination is illustrated in Fig. 7. Fig. 7(a) shows the same example of a globally self-intersecting surface with a closed-loop intersection as in Fig. 1. Fig. 4(b)–(e) show several surfaces, gradually acquired from the presented algorithm and producing a self-intersection-free metamorphosis surface, after four iterations.

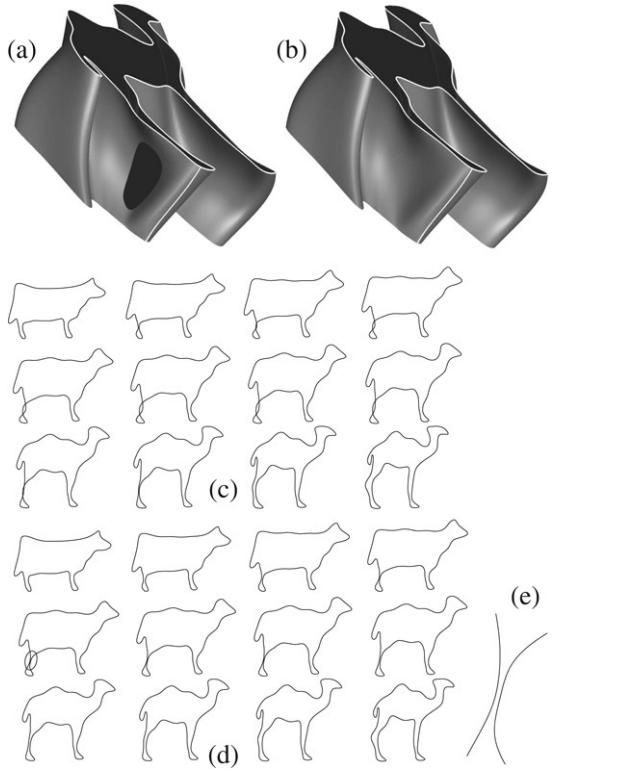

Fig. 8. (a) A globally self-intersecting metamorphosis surface with loop intersection components; (b) a globally self-intersection-free metamorphosis surface, the result of the global self-intersection elimination algorithm; (c) a morphing sequence between a cow and a camel, derived from the surface in (a); (d) a morphing sequence, derived from the surface in (b); (e) zoomed-in area of the intermediate curve from (d) (middle row, left shape), which illustrates the result of global self-intersection elimination.

Another example is shown in Fig. 8. Fig. 8(a) shows a globally self-intersecting (metamorphosis) surface with a loop intersection component between a cow and a camel. (Fig. 8(c) shows the globally self-intersecting metamorphosis sequence.) Fig. 8(b) presents the self-intersection-free (metamorphosis) surface produced by applying the proposed global elimination algorithm to the surface, whereas Fig. 8(d) shows the globally self-intersection-free metamorphosis sequence. In Fig. 8(e) a zoom on the corrected region is shown, a region that is also circled in (d) (middle row, left shape). The loop intersection component was eliminated in a single iteration of the optimization problem with large values of $\alpha = \beta = 10$ for fast convergence. Note that the use of large α and β values can improve the convergence at the cost of a less reliable process because it might produce new self-intersections during this optimization process.

Looking at the $(u - v)$ elimination approach, Fig. 9 presents one example of a periodic cubic B-spline curve with one hundred control points and several hundred self-intersections. All these self-intersections are identified in less than a second on a modern PC workstation (2GHz T7200 Intel/Windows XP machine)

Our last example computes the properly trimmed offsets from a given curve. Fig. 10 shows several properly trimmed offsets of a given cubic B-spline curve (in thick gray) with

Please cite this article in press as: Pekerman D, et al. Self-intersection detection and elimination in freeform curves and surfaces. Computer-Aided Design (2007), doi:10.1016/j.cad.2007.10.004

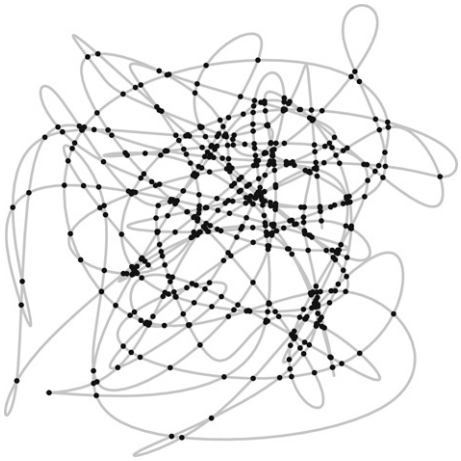

Fig. 9. Several hundred self-intersections of this periodic cubic B-spline curve with one hundred control points are identified in less than a second on a modern PC workstation, using the $(u - v)$ elimination approach.

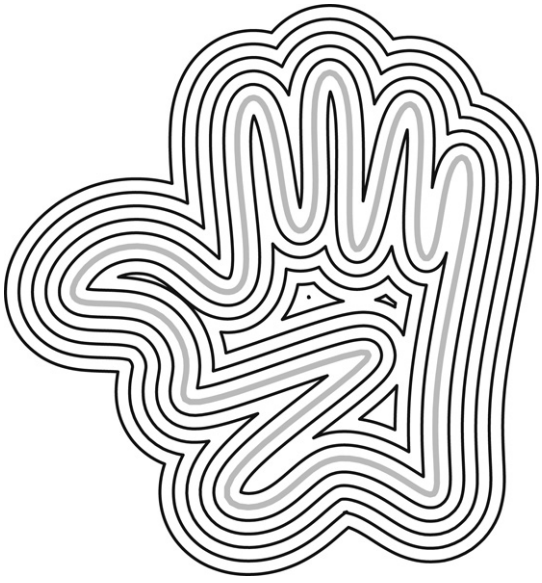

Fig. 10. The proper trimming of these offsets was conducted by subdividing the offset curves at all their self-intersection locations and purging any offset segment whose minimal distance to the original curve (thick gray) is below the offset distance, up to the offset approximation tolerance.

thirty control points. The offsets of this B-spline curve are first approximated using piecewise polynomials. The self-intersections in the offsets are computed using the $(u - v)$ elimination scheme and the offsets are divided at these intersections. All offset segments are then examined for their minimal distance to the original curve and if found below the offset amount (up to the offset approximation tolerance), the segments are purged. The returned result of this trimming process is a set of offset segments, appearing as B-spline curves.

6. Conclusions

In this paper, we have presented several algorithms for self-intersection detection and elimination for freeform curves and surfaces. We use a surface binormal-line criterion to detect antipodal points on intersection loops, which is reduced to solving a system of five multivariate polynomial constraints, one of which is an inequality constraint. Then, we offered a scheme to gradually flip the locations of the antipodal points by presenting the flips as constraints, until the resulting surface is globally self-intersection-free. The presented algorithm provides a solution to the global self-intersection problem that can occur in the arbitrary metamorphosis between freeform curves. Finally, we showed that the self-intersection constraints for a curve must include the term $(u - v)$; and by eliminating it, we have presented a robust and efficient scheme for self-intersection computation.

In the future, the presented binormal-line algorithm for global self-intersection detection could be extended to the case of metamorphosis volume $V(u, v, t)$, between freeform surfaces, in $\mathbb{R}^4$, by applying the "Brouwer fixed point theorem" to a closed sphere, thus detecting the antipodal points.

Extending the elimination of a redundant term $(u - v)$ to the freeform surface case is more challenging. In fact, the obvious attempt to eliminate $(u - r)(v - t)$ for a surface S, which is independently parameterized as $S(u, v)$ and $S(r, t)$, will fail simply because the expression $S(u, v) - S(r, t)$ does not contain a term of $(u - r)(v - t)$. We are currently working on this problem.

Acknowledgements

This research was partly supported by the New York metropolitan research fund, Technion, in part by the Software Technology Center of Excellence, Technion, and in part by the Israel Science Foundation (grants No. 857/04 and 346/07).

All the algorithms and figures presented in this paper were implemented and generated using the IRIT solid modeling system [12] developed at the Technion, Israel.

The Authors would also like to thank the reviewers of this paper for their valuable comments that made this paper more readable.

References

[1] Aomura S, Uehara T. Self-intersection of an offset surface. Computer-Aided Design 1990;22(7):417–21.
[2] Ho C-C, Cohen E. Surface self-intersection. Mathematical methods for curves and surfaces: Oslo 2000. 2001. p. 183–94.
[3] Cohen S, Elber G, Bar-Yehuda R. Matching of freeform curves. Computer-Aided Design 1997;29(5):369–78.
[4] Pekerman D. Optimization of the metamorphosis process between freeform curves. MSc thesis. Technion. 2006.
[5] Elber G, Cohen E. Error bounded variable distance offset operator for freeform curves and surfaces. International Journal of Computational Geometry & Applications 1991;1(1):67–78.
[6] Elber G, Kim M-S. Geometric constraint solver using multivariate rational spline functions. In: Proc. of the ACM symposium on solid modeling and applications. 2001. p. 1–10.
[7] Elber G. Trimming local and global self-intersections in offset curves using distance maps. In: Proc. of the 10th IMA conference on the mathematics of surfaces. 2003. p. 213–22.
[8] Elber G. Global curve analysis via a dimensionality lifting scheme. In: Proc. of the IMA conference on the mathematics of surfaces. 2005. p. 184–200.

Please cite this article in press as: Pekerman D, et al. Self-intersection detection and elimination in freeform curves and surfaces. Computer-Aided Design (2007), doi:10.1016/j.cad.2007.10.004

[9] Galligo A, Pavone JP. Self intersections of a Bézier bicubic surface. In: Proc. of the international symposium on symbolic and algebraic computation. 2005.
[10] Hoffmann CM. Geometric and solid modeling. San Mateo (CA): Morgan Kaufmann; 1989.
[11] Hoschek J, Lasser D. Fundamentals of computer aided geometric design. Wellesley (MA): AK Peters; 1993.
[12] IRIT 9.5 User's Manual. Technion. 2005. http://www.cs.technion.ac.il/~irit.
[13] Lee IK, Kim MS, Elber G. Planar curve offset based on circle approximation. Computer-Aided Design 1996;28(8):617–30.
[14] Luenberger DG. Linear and nonlinear programming. Reading (MA): Addison-Wesley; 1984.
[15] MATLAB, Version 7.0.4. January 2005.
[16] Ravi Kumar CVV, Shastry KG, Prakash BG. Computing non-self-intersecting offsets of NURBS surfaces. Computer-Aided Design 2002; 34(3):209–28.
[17] Maekawa T, Cho W, Patrikalakis NM. Computation of singularities and intersections of offsets of planar curves. Computer Aided Geometric Design 1993;10(5):407–29.
[18] Maekawa T, Patrikalakis NM. Computation of self-intersections of offsets of bezier surface patches. Journal of Mechanical Design: ASME Transactions 1997;119(2):275–83.
[19] Samoilov T, Elber G. Self-intersection elimination in metamorphosis of two-dimensional curves. The Visual Computer 1998;14: 415–428.
[20] Sederberg T, Meyers R. Loop detection in surface patch intersections. Computer Aided Geometric Design 1988;5(2):161–71.
[21] Sederberg T, Christiansen H, Katz S. Improved test for closed loops in surface intersections. Computer-Aided Design 1989;21(8):505–8.
[22] Seong J-K, Elber G, Kim M-S. Trimming local and global self-intersections in offset curves/surfaces using distance maps. Computer-Aided Design 2006;38(3):183–93.
[23] Sinha P, Klassen E, Wang KK. Exploiting topological and geometric properties for selective subdivision. In: Proc. of the annual symposium on computational geometry. 1985. p. 39–45.
[24] Wallner J, Sakkalis T, Maekawa T, Pottmann H, Yu G. Self-intersections of offset curves and surfaces. International Journal of Shape Modeling 2001;7(1):1–21.
[25] Wang Y. Intersection of offsets of parametric surfaces. Computer Aided Geometric Design 1996;13(5):453–65.
[26] Munkres JR. Topology, a first course. Englewood Cliffs (NJ): Prentice-Hall; 1975.

Please cite this article in press as: Pekerman D, et al. Self-intersection detection and elimination in freeform curves and surfaces. Computer-Aided Design (2007), doi:10.1016/j.cad.2007.10.004