Error Bounded Variable Distance Offset Operator for Free Form Curves and Surfaces *

Gershon Elber† and Elaine Cohen
UUCS-91-001
Department of Computer Science
University of Utah
Salt Lake City, UT 84112 USA
Feb 1991

*This work was supported in part by DARPA (N00014-88-K-0689). All opinions, findings, conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the views of the sponsoring agencies.
†Appreciation is expressed to IBM for partial fellowship support of the first author.

Error Bounded Variable Distance
Offset Operator
for
Free Form Curves and Surfaces *

Gershon Elber† and Elaine Cohen
UUCS-91-001
Department of Computer Science
University of Utah
Salt Lake City, UT 84112 USA
Feb 1991

Abstract

Most offset approximation algorithms for freeform curves and surfaces may be classified into two main groups. The first approximates the curve using simple primitives such as piecewise arcs and lines and then calculates the (exact) offset operator to this approximation. The second offsets the control polygon/mesh and then attempts to estimate the error of the approximated offset over a region. Most of the current offset algorithms estimate the error using a finite set of samples taken from the region and therefore can not guarantee the offset approximation is within a given tolerance over the whole curve or surface.

This paper presents new methods to globally bound the error of the approximated offset of freeform curves and surfaces and then automatically derive new approximations with improved accuracy. These tools can also be used to develop a global error bound for a variable distance offset operation and to detect and trim out loops in the offset.

1 Introduction

Offset surfaces are very important in manufacturing, and their computation and approximation have undergone extensive research. The curve offset is an intuitive operation and has been mathematically known for more than a hundred years [2, 18, 21]. The offset operator is closed for arcs and lines, i.e. the offsets of an arc and a line are an arc and a line, respectively. This is not so, in general, for Bezier and NURB curves, so approximations are usually derived.

*This work was supported in part by DARPA (N00014-88-K-0689). All opinions, findings, conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

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

Two methods for finding approximations to offsets are commonly used for freeform curves. The first approximates the curve using piecewise lines and arcs and then finds the exact offset to the approximation. Such a technique was introduced [15] and has been used successfully [11]. The second method attempts to approximate the offset by directly transforming the control polygon [16, 3, 17, 9, 10, 5]. To improve the accuracy of the approximated offset in the second method, the original curve is usually subdivided or refined when the error is above, prespecified tolerance level and the same technique is applied to each of the subdivided pieces. The original curves are usually subdivided in the middle of their parametric domain [16, 9, 10], although that is not the optimal location, in general. Curve inflection points have also been considered as splitting points for offsets [10].

Both method are unable to bound the offset error globally. In order to bound the error introduced by the piecewise arcs and lines approximation, a curve-line and a curve-arc maximum global distance computation is required, which is traditionally performed using a finite set of samples. A bound on the maximum error over the entire curve region can not be guaranteed using such a technique. In the second method, a finite number of samples are examined to estimate the error for the entire curve region (typically one, in the middle of the parametric domain), which again can not insure global error bound. Both methods usually result in a piecewise representation of the approximation to the offset, a more difficult representation to use in further applications if the offset is to be used as a modeling tool. Only the use of B-spline refinement [3], results in a single curve. Approximations to offsets of freeform surfaces are more difficult to determine because the subdivided components are subsurfaces. Bicubic patches have been used to approximate the offset surface of a given surface [7]. This method loses continuity across patches, unlike the refinement technique [3], which can be adapted for surfaces and which maintains the original continuity.

Because of the advantages of the curve/surface B-spline refinement technique, we have used this method as the basis of this implementation for bounding the global error. However, the presented method for bounding error is *not* limited to this type of representation.

Trimming the loops formed by the self-intersection curves of the offset is considered a difficult problem [11]. An attempt has been made to attack this problem using numerical techniques by using a direct search for cusps to detect and identify self-intersections [8]. However, an approximation to the offset may have no cusps simply because it is exactly that, an approximation. Unidimensional successive searches have been used to isolate self-intersection points by minimizing the ratio of the Euclidean space distance (which goes to zero at a self-intersection point) over the parametric space distance (which should be nonzero at such point) [1]. Since this method converges to a local minimum, the initial guess location is crucial and is picked at *random*. Thus robustness is not guaranteed. The curve of self intersection has been traced using surface "walking" technique [1] and that method can be combined with the detection methods developed here.

Section 2 develops the error bounding method and then shows how to use the information extracted from the curve in that method to isolate the maximum error regions, so local improvement steps may be applied iteratively and in more a optimal way than using current methods. Section 3 extends this method to support a variable offset operator that can be used as a modeling tool. Section 4 shows how to use the tools developed in section 2 to robustly detect and trim loops formed by self-intersections of the offset. The Alpha_1 solid modeler, using NURBs as its only representation, has been used to create all the examples in this paper.

2 Global Bound for the offset operator

Let $C(t)$ be a regular planar parametrized curve. Without loss of generality, assume $C(t)$ is in the $x-y$ plane. An offset curve for $C(t)$ by an amount d is defined mathematically as

$$\hat{\mathcal{C}}_d(t) = C(t) + N(t)d \tag{1}$$

where $N(t)$ is the unit normal to the curve at t. Since $N(t)$ flips its direction by 180^o at inflection points, a different definition for $N(t)$ should be used to define a manufacturing or design offset. Define the *offset binormal*, $B_o(t)$, to always point in $+z$ direction, and then define the *offset normal*, $N_o(t)$ as $N_o(t) = B_o(t) \times T(t)$, where $T(t)$ is the unit tangent to the curve. Throughout this paper, and unless otherwise specified, only the offset normal, $N_o(t)$, will be used:

$$\mathcal{C}_d(t) = C(t) + N_o(t)d \tag{2}$$

Similarly for surfaces, an offset surface for surface $S(u,v)$ by an amount d is mathematically defines as

$$\mathcal{S}_d(u,v) = S(u,v) + n(u,v)d \tag{3}$$

where $n(u,v)$ is the surface unit normal to the surface at parameter values (u,v).

Given two freeform NURB curves $C_1(t)$ and $C_2(t)$, one can compute and represent as B-spline curve their sum, difference and product [14, 6]. Derivatives of NURB curves are also representable as NURB curves, as are constant functions (i.e. d in Eqs. 2,3).

Therefore, if $N_o(t)$ $(n(u,v))$ could be computed and represented as a NURB, so could $\mathcal{C}_d(t)$ $(\mathcal{S}_d(u,v))$. Unfortunately, however, the representational form of a normal involves a square root which is usually not representable in either $\mathcal{P}$ (polynomials) or in $\mathcal{PP}$ (piecewise polynomials). Thus, offsets of freeform surfaces will, in general, be approximations.

Let $\mathcal{C}_d^a(t)$ be an approximation to the offset curve of $C(t)$ by an amount d (Eq. 2), and let $\delta(t) = \mathcal{C}_d^a(t) - C(t)$ be the difference curve. Ideally, if $\mathcal{C}_d^a(t) \equiv \mathcal{C}_d(t)$, $\delta(t) = dN_o(t)$.

Two tests can be applied to $\delta(t)$ to determine the accuracy of the offset approximation. First, the deviation of $\delta(t)$ from the direction of $N_o(t)$ can be measured. If $\hat{T}(t) = C'(t)$, the $T(t) = \frac{\hat{T}(t)}{\|\hat{T}(t)\|}$ is the unit tangent of $C(t)$. The deviation from the offset normal direction can be tested by finding the deviation of the magnitude of $\frac{\hat{T}(t)}{\|\hat{T}(t)\|} \cdot \frac{\delta(t)}{\|\delta(t)\|}$, which is equal to the cosine of the angle between the two vectors, and for the exact offset curve is equal to 0. However finding $T(t)$ and $\|\delta(t)\|$ require representing square roots, and hence are quite impractical using a piecewise rational representation. However one can represent the square of this inner product:

$$\frac{(\hat{T}(t) \cdot \delta(t))(\hat{T}(t) \cdot \delta(t))}{\|\hat{T}(t)\|^2 \|\delta(t)\|^2}. \tag{4}$$

Although a representation for Eq.4 is computable as a piecewise rational, it is a complex process.

Instead, a second test can be applied to determine the accuracy of $\mathcal{C}_d^a(t)$ by measuring the magnitude of $\delta(t)$. Computationally it is much more attractive. Current offset techniques usually evaluate this test on a set of sampled points. Direct representation of $\|\delta(t)\|$ still requires the representation of a square root, so $\psi(t) = \|\delta(t)\|^2$ is used instead and compared with d^2.

$$\psi(t) = \|\delta(t)\|^2 = \delta_x(t)^2 + \delta_y(t)^2 + \delta_z(t)^2 \tag{5}$$

where $\delta_x(t)$, $\delta_y(t)$ and $\delta_z(t)$ are the components of $\delta(t)$.

Eq.5 can be directly represented using multiplication and addition which are computable for rationals and piecewise rationals. Hereafter, assume $\psi(t)$ can be computed and represented as a scalar NURB curve. For exact offsets, ψ is a constant curve equal to d^2 and by subtracting d^2 from ψ one can find the difference curve for a particular approximation:

$$\epsilon(t) = \psi(t) - d^2. \tag{6}$$

The extremal values of the coefficients of ϵ provide a global error measure. It is important to examine the consequences for computing $\epsilon(t)$ instead of $\varepsilon(t) = \|\delta(t)\| - d$, the real error between the exact offset curve and its approximation:

$$\epsilon(t) = \psi(t) - d^2 = \|\delta(t)\|^2 - d^2 = (\varepsilon(t) + d)^2 - d^2 = \varepsilon(t)^2 + 2d\varepsilon(t) \approx 2d\varepsilon(t) \tag{7}$$

In other words, by computing the differences of the squared magnitude, the resulting error bound is scaled by the magnitude of twice the offset distance, $2d$, which is a constant and therefore easy to control. $\varepsilon(t)^2$ has been ignored since it is much smaller than $2d\varepsilon(t)$.

The problem of finding the global offset error has been reduced to a problem of finding the extrema of a freeform explicit curve. Since the values of a scalar B-spline curve over an interval lie between the maximum and minimum values of the coefficients of the non-zero B-spline functions, a simple and computationally efficient way of locally bounding the curve is immediately available.

The error between a C^2 continuous function and its Schoenberg variation diminishing spline approximation over a knot vector $\{t_i\}$ is $O(|\{t_i\}|^2)$, where $|\{t_i\}| = max_i\{t_{i+1} - t_i\}$. By using a sequence of Schoenberg variation diminishing spline approximations to $N_o(t)$, each one based on a knot vector that is a refinement of the previous one, and a sequence, $\{C_i(t)\}$, of refined representations to C, based on the same sequence of knot vectors, we form a convergent sequence of approximations to C_d. If the approximation is close over one interval, it is unnecessary to refine over that interval just to make the mesh norm smaller, since the approximation error is based on maximum error bounds over local regions. Hence, we need only refine over intervals where the error is large, as determined by the extrema of ϵ.

We derive an iterative algorithm in which each step uses the direct polygon transformation method [3] to compute offset approximations. The criteria for proceeding to the next step uses the magnitude of the extrema of $\epsilon(t)$. Then, the locations of the extrema are used to refine $C(t)$ (going from $C_i(t)$ to $C_{i+1}(t)$ and to create a new approximation to the offset. The process terminates when the magnitudes of the extrema of ϵ are within the tolerance.

Algorithm 2.1

Input:
    Tolerance, required offset curve accuracy.
    $C(t)$, input curve.
    d, offset distance.

Output:
    $C_d^a(t)$, offset curve approximation within Tolerance accuracy.

Algorithm:
    $C_0(t) = C(t)$
    $i = 0$
    Do
        Compute offset approximation $C_d^a(t)$ for $C_i(t)$
        Compute offset error $\epsilon(t)$ for $C_i(t)$ and $C_d^a(t)$ (Eqs. 5,6)
        $C_{i+1}(t) \Leftarrow C_i(t)$ refined at $\epsilon(t)$ highest error region(s).
        i = i + 1
    While ($\epsilon(t)$ highest error $>$ Tolerance).

Alg. 2.1 retains its curve refinement history in the $C_i(t)$ sequence. The last curve in the sequence can be offset to within a provided tolerance by an amount d. Since the algorithm "knows" more about the curve, improvements can be applied in a more optimal way than simply subdividing the curve at its midpoint as has been done in the past. Even for polynomial representations such as Bezier curves, it is common to split the curve at the middle of the parametric domain if the accuracy of the offset is not good enough. Using the global error measure, one can now split the curve near the parameter value with the highest error. This will usually result in requiring fewer subdivisions to achieve a given tolerance.

One can compute and refine the curve at the maxima of $\epsilon(t)$ only in each iteration. However, simultaneous refinement of all regions whose respective errors were bigger than allowable was found to be much faster. The computation of $\epsilon(t)$ is much demanding than single knot insertion and by using simultaneous refinement this computation is fully exploited.

Fig. 1 shows 4 stages of Alg. 2.1, using global refinement, operating on a chess pawn crosssection. Single knots have been inserted in all parametric regions whose error was above

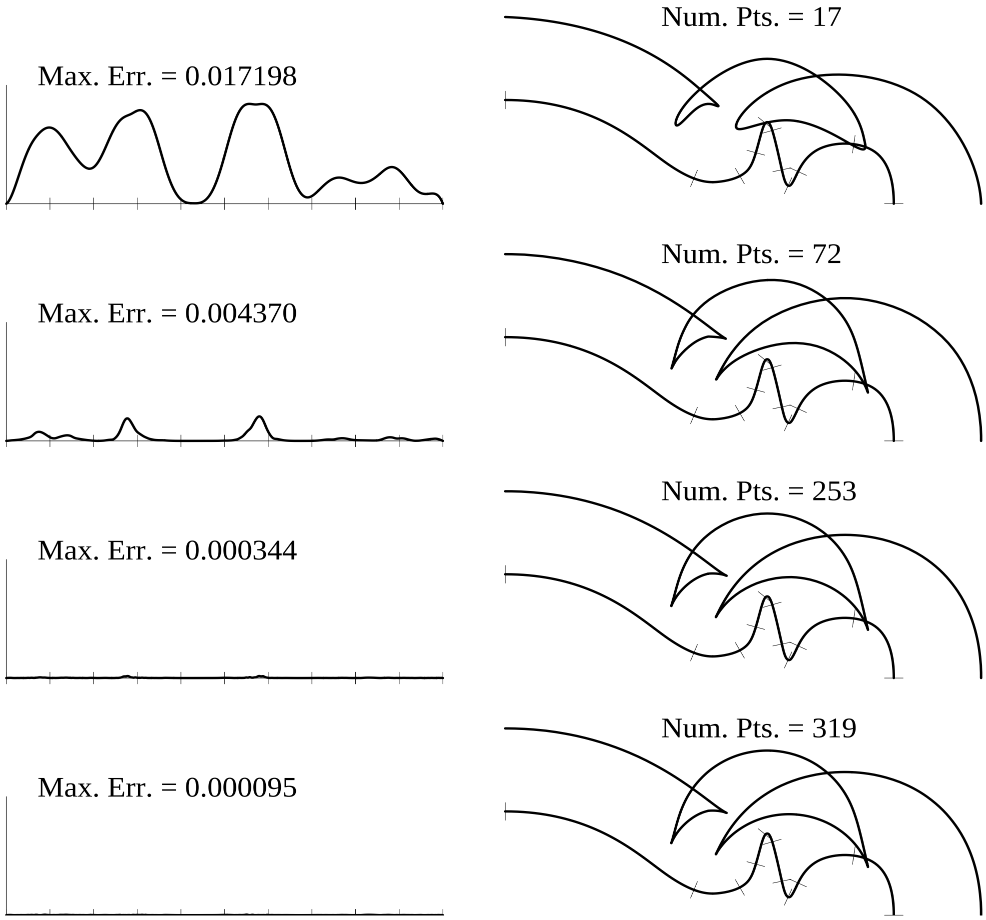

Figure 1: 4 stages in global error bounding $\epsilon(t)$ and simultaneous auto refinement.

the tolerance level. Also provided in Fig. 1 are the number of control points and the respective error function $\epsilon(t)$ for each iteration. The error is improved by almost an order of magnitude on each iteration up to the required tolerance of 0.0001.

Finding approximations to offsets of surfaces are usually more difficult, but the above method can be applied to finding errors of offset surfaces as well. δ, ψ and ϵ would be simply explicit surfaces instead of explicit curves, i.e. $\delta(u,v)$, $\psi(u,v)$ and $\epsilon(u,v)$. In Fig. 2, this error bounding extension surface is used to automatically iterate, refine, and improve an offset B-spline surface to a specified tolerance. It is interesting to compare the two offset surfaces in Fig. 2. They both have the same tolerance but the offset distance is different. The offset error increases as d becomes larger and therefore more refinements are required to achieve the same accuracy.

3 The Offset Operator as a Modeling Tool

The offset operator can be used as a modeling tool. In fact, one can extend the global error finding method developed in section 2 and allow variable distance offsets as well. Given a

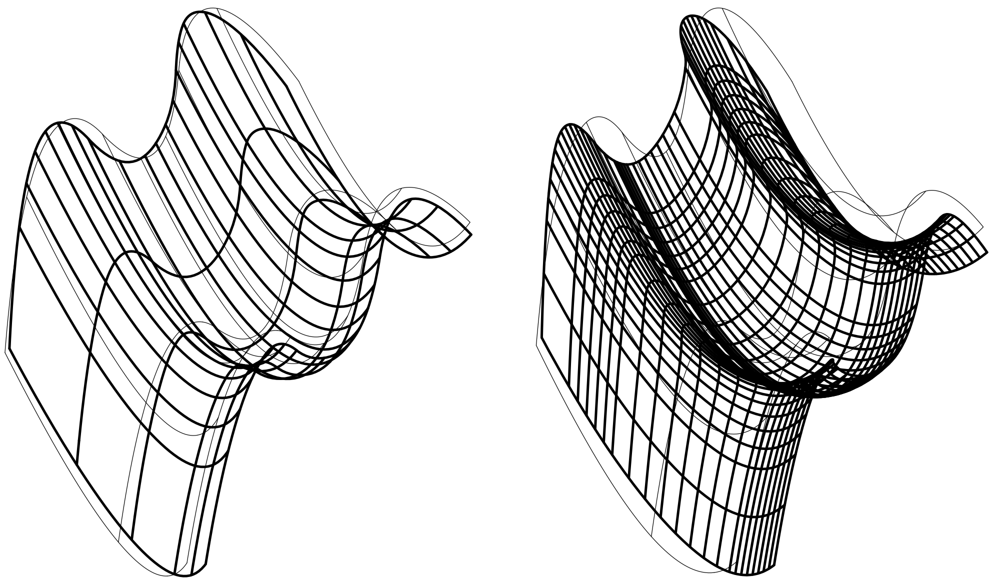

Figure 2: Error bounded offset surface example, using simultaneous auto refinement.

parameter value, t, one needs to specify the offset distance required at that location. A scalar explicit distance function $d(t)$ (or $d(u, v)$ for surfaces) having the same domain as $C(t)$ ($S(u, v)$) can be used. The only change that must be made to the method developed in section 2 is that Eq.6 should now read:

$$\epsilon(t) = \psi(t) - d^2(t), \tag{8}$$

where d, which used to be constant, is now a distance function. In Eq.7 it was shown that the global error bound depends on d, so now the extrema of $d(t)$ are used to bound the error. Alg. 2.1 described in section 2 is identical to the one that should be used here. Figs. 3 and 4 show some simple examples of the operator's power, for both curves and surfaces.

4 Trimming Self Intersection Loops

Two types of loops are sometimes created in $C_d^a(t)$ when $C(t)$ is a C^1 continuous curve. If $\kappa(t)$, the curvature of $C(t)$, is bigger than $\frac{1}{d}$, where d is the offset distance, a loop will be formed (see Fig. 5). Since this loop is local to a region in which the curvature is too high, this type of loops will be referred to as a *local loop*. However, not all loops resulting from offset operations are of this kind. Some of the loops formed, as can be seen in Fig. 7, are the result of two separate regions in $C(t)$ so close that the offset curve in those regions intersects itself. This type of loop is referred to as a *global loop*.

Detection of these loops is a difficult. A search for cusps was suggested as a method to detect local loops [8]. However, since $\mathcal{C}_d^a(t)$ is only an approximation, it is possible that no cusps will be formed (see first (top) stage of Fig. 1). Moreover the cusps, when detected, must

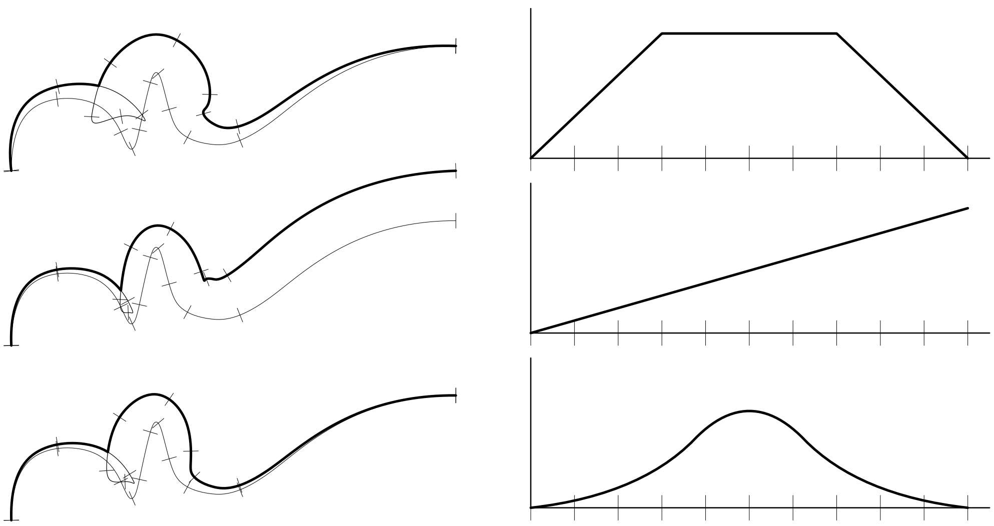

Figure 3: Variable distance curve offset (left) using a scalar distance function (right).

be grouped in pairs, which is not a natural process to this technique. We use a more robust method to correctly detect all loops.

Luckily, local loops have a very distinct characteristic. If $\mathcal{T}(t)$ be the tangent vector to $\mathcal{C}_d(t)$ and if $\kappa(t)$, the curvature of $C(t)$, is equal to $\frac{1}{d}$ at $t = t_0$ then $\|\mathcal{T}(t_0)\| = 0$ or $\mathcal{C}_d(t)$ has a cusp at t_0 (see [5] and appendix A). So, if $C(t)$ is curvature continuous, each time $\kappa(t) = \frac{1}{d}$ and $N(t) = N_o(t)$, $\|\mathcal{T}(t)\| = 0$. If $\kappa(t) > \frac{1}{d}$ and the normals coincide, $\mathcal{T}(t)$ flips its direction 180^o. When $\kappa(t)$ continuously changes from $< \frac{1}{d}$ to $> \frac{1}{d}$ and then back to $< \frac{1}{d}$ and the normals coincide, two cusps will be formed in $\mathcal{C}_d(t)$ at the places where $\kappa(t) = \frac{1}{d}$.

Using this characteristic, one can identify the cusp pairs by finding the zero set of $\tau(t) = \mathcal{T}(t) \cdot T(t)$. The regions where $\tau(t)$ is negative are the regions where $\mathcal{T}(t)$ flips its direction (i.e. normals coincide and $\kappa(t) > \frac{1}{d}$). Fig. 6 demonstrates this process. The tangent curves $T(t)$ ((a) in Fig. 6) and $\mathcal{T}(t)$ ((b) in Fig. 6) have been derived. Their dot product ((c) in Fig. 6), $\tau(t) = T(t) \cdot \mathcal{T}(t)$, is computed and used to identify the two local loops in the resulting offset approximation in its two negative regions. ((d) in Fig. 6). Once the two loop have been identified, they can be trimmed away ((e) in Fig. 6).

The usage of $\tau(t)$ to identify local loops make this process more robust, even if no cusps are formed in the offset approximation. The tangent vector, $\mathcal{T}(t)$, still flips its direction and still makes $\tau(t)$ negative (Fig. 6 (c)). Furthermore, by detecting the negative regions of $\tau(t)$ the cusps are virtually paired since each cusp pair is the negative $\tau(t)$ region boundary.

Once a local loop has been identified using $\tau(t)$, the curve should be split into three parts, the region before the first cusp, the region after the second cusp, and the region between the two cusps. The third part, between the cusps, must be deleted. The first two should then be intersected against each other to find the self intersection point using standard curve-curve intersection algorithms [4, 12, 19], trimmed properly to the intersection point, and then merged

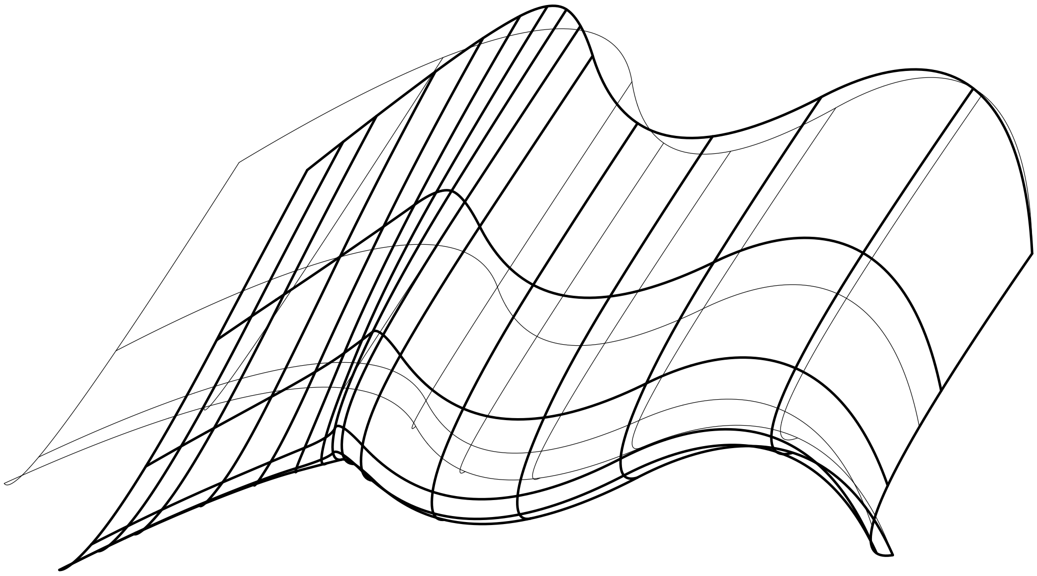

Figure 4: Variable distance surface offset (u direction linear, v constant).

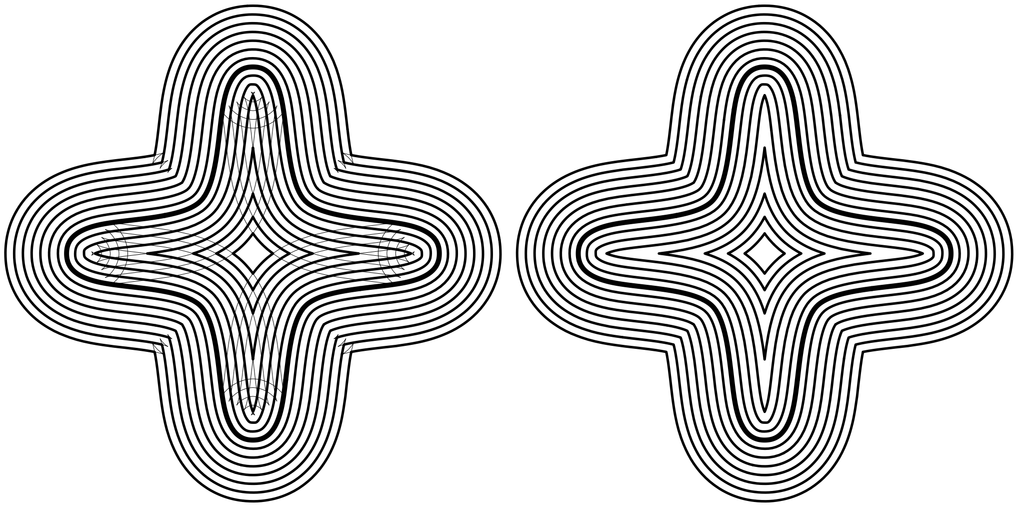

Figure 5: Offset operation local loops are trimmed using a distinct characteristic.

back. See Figs. 5 and 7 for some examples.

Global loops have no such characteristic and are therefore more difficult to isolate. It is necessary to find all the self-intersections of a curve. However, a curve which is monotone in one dimension can never intersect itself. Therefore, one way to approach this problem is to split the curve into monotone subcurves, intersect all the subcurves against each other using curve-curve intersection algorithms, and isolate all the self-intersection points if any. Loops can now be formed by tracing the self-intersection points along the parameter space. Given an intersection point P_i, when $C(t_i^1) = C(t_i^2)$, the sign of the dot product $T(t_i^1) \cdot N_o(t_i^2)$ may be used to determine if a loop is to be purged or not. Given P_i, the normal $N_o(t_i^2)$ defines the relative position of the original and offset curve. If the dot product is negative, it means

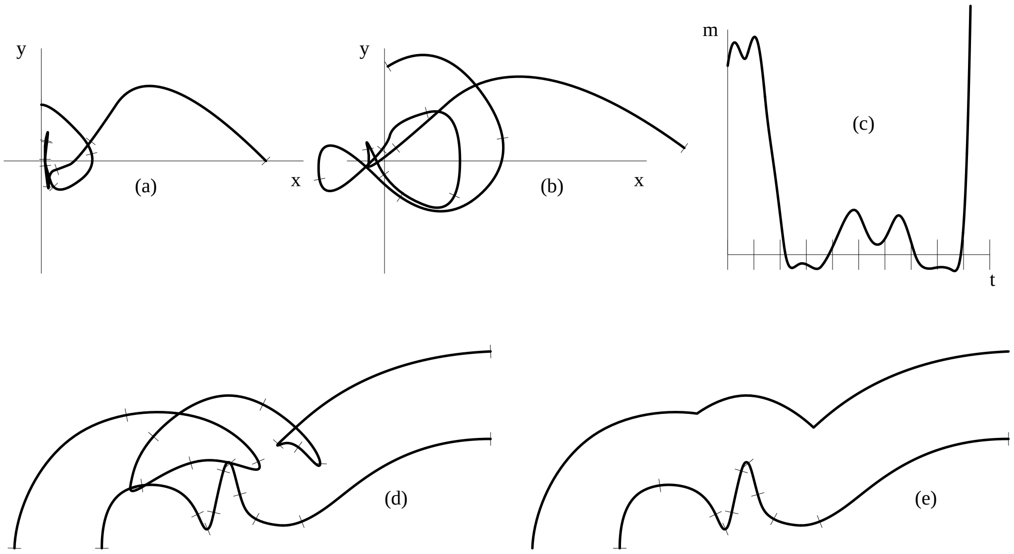

Figure 6: Product of a curve and its offset tangents is used to identify local loops.

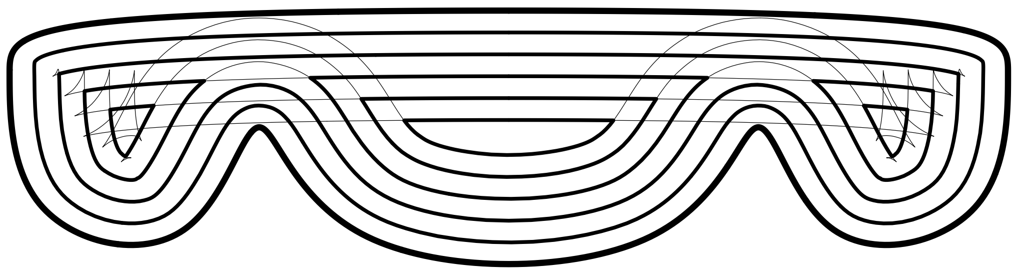

Figure 7: Global loop are being trimmed using numerical techniques.

the intersecting curve (with tangent $T(t_i^1)$) in P_i is closer locally (P_4 in Fig. 8) to the original curve than the offset amount. Since curves are continuous, it implies the whole loop is closer than the offset amount and therefore should be removed (loop 4 in Fig. 8). Similarly, the dot product is found to be positive in P_5 (Fig. 8) so in the neighborhood of P_5, loop 5 distance to the offset curve in the N_5 direction is larger than the offset amount and therefore loop 5 is locally (and globally) valid. The loops are tested while following the parameter values of the curve beginning to its end. For each intersection of an untested loop i, the tangent T_i of the current curve parameter is computed along with the offset normal N_i for the *other* curve parameter of the intersection point i. Using the example in Fig. 7, loop 1 is tested first. $T_1 \cdot N_1$ is found negative and therefore loop 1 should be purged. Since $T_2 \cdot N_2$ is positive loop 2 should not be purged etc.. This approach has been used to trim out the global loops of Fig. 7.

The curve offset local loop detection method may be extended to surfaces as well. If the surface radius is smaller than the offset distance, the normal of the offset surface flips its direction. Therefore the dot product of the original and offset surface normals may be used

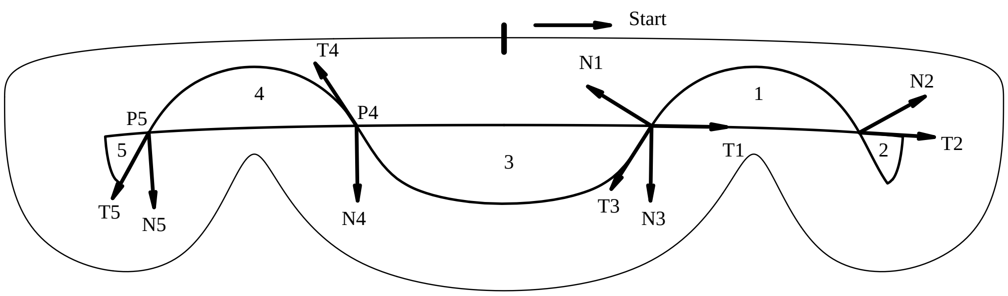

Figure 8: Global loop classification is based on $N_i(t_i^1) \cdot T_i(t_i^2)$ sign.

in such detection. Trimming surface loops is much more difficult since they are, in general, not isoparametric. Furthermore surface self-intersections may not be complete, that is the intersection curve may not subdivide the surface parameter space into two separate regions. These topics are current research areas. The approach taken [1] for the self-intersection curve tracing using surface "walking" may be used.

References

[1] S. Aomura and T. Uehara. Self-intersection of an offset surface. Computer Aided Design, vol. 22, no. 7, pp 417-422, september 1990

[2] J. Bertrand. La theorie des courbes a double courbure. Journal de Mathematique Pures et Appliquees, 15 (1850), 332-350.

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

[4] E. Cohen, T. Lyche, and R. Riesenfeld. Discrete B-splines and subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics. Computer Graphics and Image Processing, 14, 87-111 (1980).

[5] R. T. Farouki and C. A. Neff. Some Analytic and Algebraic Properties of Plane Offset Curves. Research Report 14364 (#64329) 1/25/89, IBM Research Division.

[6] R. T. Farouki and V. T. Rajan. Algorithms For Polynomials In Bernstein Form. Computer Aided Geometric Design 5, pp 1-26, 1988.

[7] R. T. Farouki. The approximation of non-degenerate offset surfaces Computer Aided Geometric Design, Vol. 3, No. 1, pp 15-43, May 1986.

[8] J. Hoschek. Offset curves in the plane Computer Aided Design, vol. 17, no. 2, pp 77-82, march 1985

[9] J. Hoschek. Spline approximation of offset curves. Computer Aided Geometric Design 5 (1988) p. 33-40

[10] J. Hoschek and N .Wissel. Optimal approximate conversion of spline curves and spline approximation of offset curves. Computer Aided Design, vol. 20, no. 8, pp 475-483, october 1988.

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

[12] J. Lane and R. Riesenfeld. A Theoretical Development for the Computer Generation and Display of Piecewise Polynomial Surfaces. IEEE Transaction on pattern analysis and machine intelligence, vol. PAMI-2, No. 1, January 1980.

[13] Millman and Parker. Elements of Differential Geometry Prentice Hill Inc., 1977.

[14] K. Morken. Some Identities for Products and Degree Raising of Splines. To appear in the journal of Constructive Approximation.

[15] H. Persson. NC machining of arbitrarily shaped pockets. Computer Aided Design, vol. 10, no. 3, pp 170-174, may 1978.

[16] B. Pham. Offset Approximation of Uniform B-splines. Computer Aided Design, vol. 20, no. 8, pp 471-474, october 1988.

[17] S. Coquillart. Computing offset of Bspline curves. Computer Aided Design, vol. 19, no. 6, pp 305-309, july/august 1987.

[18] F. Salkowski. Zur Transformation von Raumkurven. Mathematische Annalen, 66 (1909), 517-557.

[19] T. Sederberg and S. Parry. Comparison of Three Curve Intersection Algorithms. Computer Aided Design, Volume 18, Number 1, January/February 1986.

[20] J. J. Stoker. Differential Geometry Wiley-Interscience 1969.

[21] A. Voss. Uber Kurvenpaare in Raume Sitzungsberichte, Akadamie der Wissenschaften zu Munchen. 39 (1909), 106.

Appendix

A Cusp existence proof

This appendix shows that a cusp is formed in the offset curve $\mathcal{C}_d(t)$ any time the curve, $C(t)$, has curvature $\kappa(t)$ equal to $\frac{1}{d}$ where d is the offset distance and the mathematical curve normal $N(t)$, coincides with offset normal $N_o(t)$. Conditions for detecting curvature higher than $\frac{1}{d}$ are also derived.

Let $C(t)$ be a regular planar parametric curve. $C(t)$ is not necessarily arc length parameterized. Without lose of generality assume $C(t)$ is in the $x - y$ plane. Let $\mathcal{C}_d(t)$ be the offset curve of $C(t)$ by amount d. Let T, N and $\mathcal{T}$, $\mathcal{N}$ be their unit tangents and normals respectively. A non unit length vector will be tagged with a hat. I.e $\hat{T}$.

The tangent T of the planar curve C is equal to

$$T(t) = \frac{\hat{T}(t)}{\|\hat{T}(t)\|} = \frac{(x'(t), y'(t))}{\sqrt{x'(t)^2 + y'(t)^2}}. \tag{9}$$

From differential geometry theory [13, 20]

$$\kappa(t)B(t) = \frac{C'(t) \times C''(t)}{\|C'(t)\|^3} = \frac{(x'(t), y'(t), 0) \times (x''(t), y''(t), 0)}{\sqrt{x'(t)^2 + y'(t)^2}^3} = \frac{(0, 0, x'(t)y''(t) - y'(t)x''(t))}{\|\hat{T}\|^3}$$
$$= \frac{(0, 0, \Psi)}{\|\hat{T}\|^3}. \tag{10}$$

Since $B_o(t)$ as been selected to be in $+z$ direction (see equations 1 and 2), $N_o(t)$ is equal to

$$N_o(t) = B_o(t) \times T(t) = \frac{(-y'(t), x'(t))}{\sqrt{x'(t)^2 + y'(t)^2}} = \frac{(-y'(t), x'(t))}{\|\hat{T}\|}. \tag{11}$$

The offset curve $\mathcal{C}_d(t)$ of the planar curve $C(t)$ by amount d is defined as (equation 2)

$$\mathcal{C}_d(t) = C(t) + N_o(t)d = (x(t), y(t)) + \frac{(-y'(t), x'(t))}{\|\hat{T}\|}d$$
$$= \frac{(x(t)\|\hat{T}\| - y'(t)d, y(t)\|\hat{T}\| + x'(t)d)}{\|\hat{T}\|}. \tag{12}$$

The first derivative $\hat{\mathcal{T}}(t)$ of the offset curve $\mathcal{C}_d(t)$ is:

$$\hat{\mathcal{T}}(t) = \mathcal{C}_d'(t)$$

$$= \left(\frac{(x'(t)\|\hat{T}\| + x(t)\|\hat{T}\|' - y''(t)d)\|\hat{T}\| - (x(t)\|\hat{T}\| - y'(t)d)\|\hat{T}\|'}{\|\hat{T}\|^2}, \right.$$

$$\left. \frac{(y'(t)\|\hat{T}\| + y(t)\|\hat{T}\|' + x''(t)d)\|\hat{T}\| - (y(t)\|\hat{T}\| + x'(t)d)\|\hat{T}\|'}{\|\hat{T}\|^2} \right)$$

$$= \left(\frac{x'(t)\|\hat{T}\|^2 - y''(t)\|\hat{T}\|d + y'(t)\|\hat{T}\|'d,\, y'(t)\|\hat{T}\|^2 + x''(t)\|\hat{T}\|d - x'(t)\|\hat{T}\|'d}{\|\hat{T}\|^2} \right) \tag{13}$$

We now ready to inspect the value of $\hat{\mathcal{T}}(t)$ in a case where d is equal to $\frac{1}{\kappa(t)}$. Using equation 10:

$$d = \frac{1}{\kappa(t)} = \frac{\|\hat{T}\|^3}{\mid \Psi \mid}.$$

Substituting d in the x component of $\hat{\mathcal{T}}(t)$ we have:

$$\hat{\mathcal{T}}_x(t) = \frac{x'(t)\|\hat{T}\|^2 - y''(t)\|\hat{T}\|d + y'(t)\|\hat{T}\|'d}{\|\hat{T}\|^2}$$

$$= x'(t) - \frac{y''(t)\|\hat{T}\|^2}{\mid \Psi \mid} + \frac{y'(t)\|\hat{T}\|\|\hat{T}\|'}{\mid \Psi \mid}$$

$$= \frac{x'(t) \mid \Psi \mid - y''(t)x'(t)^2 - y''(t)y'(t)^2 + y'(t)x'(t)x''(t) + y'(t)^2 y''(t)}{\mid \Psi \mid}$$

$$= \frac{x'(t) \mid x'(t)y''(t) - x''(t)y'(t) \mid - y''(t)x'(t)^2 - y''(t)y'(t)^2 + y'(t)x'(t)x''(t) + y'(t)^2 y''(t)}{\mid x'(t)y''(t) - y'(t)x''(t) \mid}$$

$$= \begin{cases} \equiv 0, & \Psi > 0 \\ = 2x'(t) & \Psi < 0 \end{cases} \tag{14}$$

since

$$\|\hat{T}\|\|\hat{T}\|' = \sqrt{x'(t)^2 + y'(t)^2} \frac{1}{2\sqrt{x'(t)^2 + y'(t)^2}} (2x'(t)x''(t) + 2y'(t)y''(t))$$

$$= x'(t)x''(t) + y'(t)y''(t)$$

and

$$\|\hat{T}\|^2 = x'(t)^2 + x'(t)^2.$$

d may be substituted into the y component of $\hat{\mathcal{T}}(t)$, $\hat{\mathcal{T}}_y(t)$, in a similar way for the same result. Therefore $\hat{\mathcal{T}}(t) \equiv 0$ in this situation or $\mathcal{C}(t)$ has a cusp if $\Psi = x'(t)y''(t) - x''(t)y'(t) > 0$ or the binormal $B(t)$ is positive and coincides with $B_o(t)$ definition and so are $N(t)$ and $N_o(t)$.

Moreover, if $d > \frac{1}{\kappa(t)}$ then the tangent vector $\hat{\mathcal{T}}$ flips direction as can be shown by its dot product with $\hat{T}$. Rewriting equation 13 as

$$\hat{\mathcal{T}}(t) = (x'(t), y'(t)) + \frac{(-y''(t), x''(t))d}{\|\hat{T}\|} + \frac{(y'(t), -x'(t))\|\hat{T}\|'d}{\|\hat{T}\|^2}$$

and substituting it into

$$\hat{\mathcal{T}}(t) \cdot \hat{T}(t) = \left[(x'(t), y'(t)) + \frac{(-y''(t), x''(t))d}{\|\hat{T}\|} + \frac{(y'(t), -x'(t))\|\hat{T}\|'d}{\|\hat{T}\|^2}\right] \cdot (x'(t), y'(t))$$

$$= (x'(t)^2 + y'(t)^2) + \frac{(-y''(t)x'(t) + x''(t)y'(t))d}{\|\hat{T}\|} = (x'(t)^2 + y'(t)^2) - \frac{\Psi d}{\|\hat{T}\|}$$

since the last term of $\hat{\mathcal{T}}(t)$ is perpendicular to $\hat{T}(t)$. Using equation 10:

$$\hat{\mathcal{T}}(t) \cdot \hat{T}(t) = (x'(t)^2 + y'(t)^2) - \frac{\Psi d}{\|\hat{T}\|}$$

$$= \begin{cases} (x'(t)^2 + y'(t)^2) - \frac{\kappa(t)(x'(t)^2+y'(t)^2)^{\frac{3}{2}} d}{\sqrt{x'(t)^2+y'(t)^2}} = (x'(t)^2 + y'(t)^2)(1 - \kappa(t)d), & \Psi > 0 \\ (x'(t)^2 + y'(t)^2) + \frac{\kappa(t)(x'(t)^2+y'(t)^2)^{\frac{3}{2}} d}{\sqrt{x'(t)^2+y'(t)^2}} = (x'(t)^2 + y'(t)^2)(1 + \kappa(t)d), & \Psi < 0. \end{cases}$$

Since $C(t)$ is a regular curve, $T(t)$ is never zero and so is $(x'(t)^2+y'(t)^2)$ which is positive everywhere. Therefore for cases were the mathematical normal, $N(t)$, coincides with the offset normal, $N_o(t)$ or $\Psi > 0$ we get

$$sign(\hat{\mathcal{T}}(t) \cdot \hat{T}(t)) = sign((x'(t)^2 + y'(t)^2)(1 - \kappa(t)d)) = sign(1 - \kappa(t)d). \tag{15}$$

Now for small $\kappa(t)$ or relatively straight curve $(1 - \kappa(t)d)$ is positive. When $\kappa(t)$ reaches $\frac{1}{d}$ the expression becomes zero or $\hat{\mathcal{T}}(t) = 0$ since $\hat{T}(t)$ is never zero. if $\kappa(t)$ is bigger than $\frac{1}{d}$ the expression becomes negative or $\hat{\mathcal{T}}(t)$ flipped its direction.

If $\Psi < 0$ the expression is never zero since both d and $\kappa(t)$ are positive scalar. This is not surprising result since such offset only *increases* the curve osculating circle radius and hence can never make it vanish.