Trimming Local and Global Self-intersections in Offset Curves using Distance Maps [⋆]

Gershon Elber

Computer Science Department
Technion
Haifa 32000, Israel
gershon@cs.technion.ac.il

Abstract. The problem of detecting and eliminating self-intersections in offset curves is a fundamental question that has attracted numerous researchers over the years. The interest has resulted in copious publications on the subject.
Unfortunately, the detection of self-intersections in offset curves, and more so, the elimination of these self-intersections are difficult problems with less than satisfactory answers.
This paper offers a simple, and equally important robust, scheme to detect and eliminate local as well as global self-intersections in offsets of freeform curves. The presented approach is based on the derivation of an analytic distance map between the original curve and its offset.

1 Introduction and Background

Offsets of curves and surfaces are crucial in many applications from robotic navigation, through CNC machining, to geometric design. This basic operation could be found in virtually any contemporary geometric modeling system or environment. The offset curve $C_d^o(t)$ by amount d to a given rational curve $C(t)$ equals,

$$C_d^o(t) = C(t) + N(t)d, \tag{1}$$

where $N(t)$ is the unit normal of $C(t)$. The offset of rational curve $C(t)$ is not rational, in general, due to the necessary normalization of $N(t)$.

The fact that $C_d^o(t)$ is not rational introduces an enormous difficulty into the computation of $C_d^o(t)$, as virtually all geometric design systems support only rationals. Hence, $C_d^o(t)$ must be approximated. Numerous publications can be found on this topic of approximating offsets of rational curves as rationals [5, 7, 10–12].

The problem of proper offset computation is reflected in a more gloomy light if one considers why offsets are so useful. The offset to a border line can offer the

⋆ The research was supported in part by the Fund for Promotion of Research at the Technion, IIT, Haifa, Israel.

path to follow at a fixed distance from that border, for robot navigation. Successive offsets from a boundary of a pocket could similarly serve as the toolpath along which to drive a CNC cutter. In geometric modeling, the offset of some shape is a simple way of building a constant thickness wall out of the shape.

Given a general curve as is presented in Figure 1, in all these applications, be it for robotic navigation, CNC machining, or geometric modeling, the offset result shown in Figure 1 is not the one typically desired. Nonetheless and while it is the accurate offset, the self-intersections in the resulting computed offset are expected to be trimmed away.

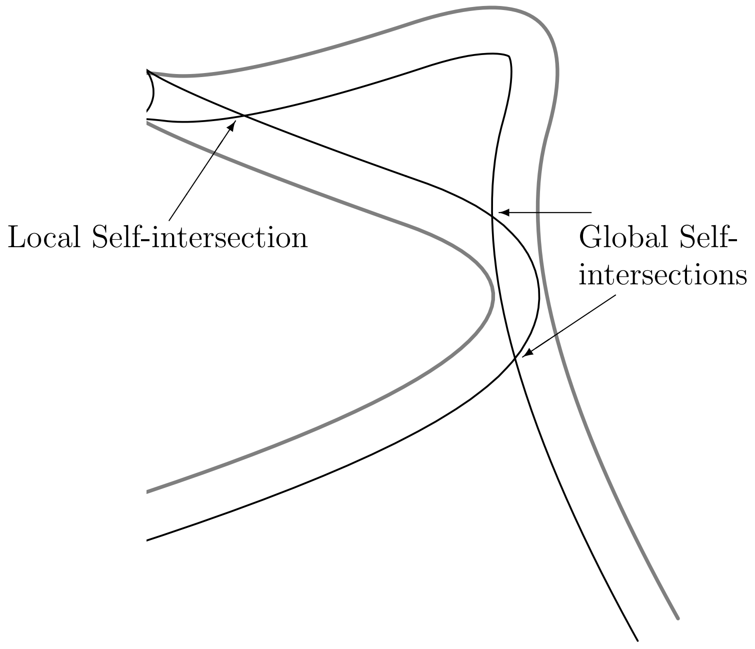

Fig. 1. A general curve (in gray) and its accurate offset. Both local and global self-intersecting could occur in the offset curve, and should be trimmed away.

The self-intersections in offsets are typically classified into two types. Let κ be the curvature of curve $C(t)$. *Local self-intersections* in offset curves are intersections that are due to regions in $C(t)$ that present a radius of curvature, $1/\kappa$, that is smaller than d, the offset distance.

Additionally, two unrelated points in $C(t)$, $C(t_0)$ and $C(t_1)$, that are close, could intersect in the offset curve. If

$$C_d^o(t_0) = C(t_0) + N(t_0)d = C(t_1) + N(t_1)d = C_d^o(t_1),$$

$C_d^o(t_0)$ and $C_d^o(t_1)$ are at the same location and a *global self-intersection* will result.

The detection and trimming of local, and more so, global self-intersections in offset curves are considered to be difficult problems. A monotone curve cannot self-intersect, which leads to a conceptually simple, yet computationally complex approach. We can subdivide $C_d^o(t)$ into monotone regions, and then intersect each of the regions against all other regions, in order to detect all the self-intersection locations. A somewhat simplified approach, taken in [10], converts the curves to a piecewise linear approximation and processes all the (monotone) linear segments using a plane sweep [1] scheme.

Solutions to eliminate the self intersecting regions in closed curves that are piecewise-lines and -arcs were also proposed in the past, by computing the Voronoi map of the curves, [13, 2, 8]. The bisectors forming the Voronoi diagram directly hint on the distance from the boundary (the closed curve). As one moves along the bisectors starting from the boundary, the distance to the boundary increases monotonically. Together, all the bisectors' arrangment that form the Voronoi diagram, could be viewed as a distance map (proximity map in [8]). Then, the proper offset could be extracted as the iso-surface of the Voronoi diagram seen as height field with the distance mapped to height. The extension of this approach for freeform shapes is difficult. The bisectors between points, lines and arc are all conics. Unfortunately, the bisectors of rational planar curves are not rational, in general [6], and forming the complete Voronoi diagram of freeform planar shape is still an open question.

Let $T(t)$ and $T_d^o(t)$ be the tangent fields of $C(t)$ and $C_d^o(t)$, respectively. For a self-intersection-free offset curve, $T(t)$ and $T_d^o(t)$ are parallel, for all t. In [3], the mutual characteristic of the tangent fields of $C(t)$ and $C_d^o(t)$ is used to detect local self-intersections. Consider,

$$\delta(t) = \langle T(t), T_d^o(t) \rangle \, . \tag{2}$$

$\delta(t)$ is negative only at the neighborhood of a local self-intersection as the tangent field of $C_d^o(t)$ flips its direction, for $1/\kappa$ that is smaller than d.

Global self-intersections are more difficult to detect. Nevertheless, the need to robustly detect and eliminate self-intersections stems from the simple fact that a failure in the detection and/or the elimination process would take the robot into a collision path or will gouge into the part on the CNC machine.

In this paper, we present a simple yet robust scheme to detect and eliminate self-intersections in offsets of freeform planar curves. While we do employ the concept of a distance maps, we do attempt to build the complete Voronoi diagram for the given curve. Instead, the distance function is computed only for a small neighborhood of the computed offsets. The rest of this paper is organized as follows. In Section 2, the proposed offset trimming approach is discussed, an approach that is based on an analytic distance function computation. In Section 3, some examples are presented and finally, we conclude in Section 4.

2 The Offset Trimming Approach

Consider the rational offset approximation of rational curve $C(t)$ by amount d, $C_{d_\epsilon}^o(t)$, where $\epsilon \in \mathbb{R}^+$ denotes the accuracy of the approximation. That is, the offset distance is bound to be between $d \pm \epsilon$. The presented trimming process of self-intersections is independent of the offset approximation scheme. Henceafter and unless otherwise stated, we assume $C^o_{d_\epsilon}(t)$ is parameterized independently of $C(t)$ as $C^o_{d_\epsilon}(r)$. Consider the new distance square function of,

$$\Delta^2_d(r,t) = \left\langle C(t) - C^o_{d_\epsilon}(r), C(t) - C^o_{d_\epsilon}(r) \right\rangle . \tag{3}$$

If no self-intersection occurs in $C^o_{d_\epsilon}(r)$, then $\Delta^2_d(r,t) \geq (d-\epsilon)^2$. In contrast, if $C^o_{d_\epsilon}(r)$ is self-intersecting, there exist points in $C^o_{d_\epsilon}(r)$ that are closer than $d-\epsilon$ to $C(t)$. Therefore, any pair of points $C^o_{d_\epsilon}(r)$ and $C(t)$ such that $\Delta^2_d(r,t) < (d-\epsilon)^2$ hints at a self-intersection. Moreover, any point $C^o_{d_\epsilon}(r)$ for which there exists a point $C(t)$ such that $C(t) - C^o_{d_\epsilon}(r) < d - \epsilon$, for some t, must be trimmed away.

Let $\rho \in I\!R^+$ be another small positive real value and let $\mathcal{D}(r,t) = \Delta^2_{d_\epsilon}(r,t) - (d-\epsilon-\rho)^2$. Any point in the zero set of $\mathcal{D}(r_0,t_0)$ represents two points, $C(t_0)$ and $C^o_{d_\epsilon}(r_0)$, that are $(d-\epsilon-\rho)$ apart. Every such point $C^o_{d_\epsilon}(r_0)$ must be purged away as a self-intersecting point. We denote this trimming process of ρ below the offset approximation, a *ρ-accurate trimming or ρ-trimming*, for short. Hence, we now offer the following algorithm to detect and eliminate the self-intersection regions:

```
Algorithm 1
Input:
    $C(t)$, A rational curve;
    $C^o_{d_\epsilon}(r)$, A rational approximation offset of $C(t)$ by distance $d$
                                                  and tolerance $\epsilon$;
    $\rho$, a trimming tolerance for the self-intersections.

Output:
    $\overline{C}^o_{d_\epsilon}(r)$, A rational approximation offset of $C(t)$ by distance $d$,
                                          tolerance $\epsilon$, and $\rho$-trimming;

Begin
    $\Delta^2_d(r,t) \Leftarrow \langle C(t) - C^o_{d_\epsilon}(r), C(t) - C^o_{d_\epsilon}(r) \rangle$;
    $\mathcal{D}(r,t) \Leftarrow \Delta^2_d(r,t) - (d-\epsilon-\rho)^2$;
    $\mathcal{Z} \Leftarrow$ the zero set of $\mathcal{D}(r,t)$;
    $\mathcal{Z}_r \Leftarrow$ the projection of $\mathcal{Z}$ onto the $r$ axis;
    $\overline{C}^o_{d_\epsilon}(r) \Leftarrow$ the $r$ domain(s) of $C^o_{d_\epsilon}(t)$ not included in $\mathcal{Z}_r$;
End.
```

$\Delta^2_d(r,t)$ and $\mathcal{D}(r,t)$ are clearly piecewise rational, provided $C(t)$ and $C^o_{d_\epsilon}(t)$ are, and that they are the result of products and differences of piecewise rational functions. See, for example, [4].

With $\mathcal{D}(r,t)$ as a piecewise rational, the zero set, $\mathcal{Z}$, could be derived by exploiting the convex hull and subdivision properties, yielding a highly robust divide and concur zero set computation that is reasonably efficient. Nevertheless, we are not really interested in the zero set of $\mathcal{D}(r,t)$, but merely in all r such that

$\overline{C}^o_{d_\epsilon}(r)$ is closer to $C(t)$ more than $(d-\epsilon-\rho)$, for some t. Hence, $\mathcal{Z}$ is projected onto the r axis, as $\mathcal{Z}_r$. The domain of r covered by this projection prescribes the regions of $C^o_{d_\epsilon}(r)$ that must be purged away.

At this point, it is crucial to emphasize the total separation between the two stages: the offset approximation step and the self-intersection trimming stage. The result of the offset approximation step, $C^o_{d_\epsilon}(r)$, can be an arbitrary offset curve approximation of $C(t)$ that is accurate to within ϵ. Furthermore, $C^o_{d_\epsilon}(r)$ can assume any regular parameterization. Algorithm 1 assumes nothing of the parameterization of the curve and its offset, in contrast, for example, to the local self-intersection detection and elimination method proposed in [3].

The outcome of Algorithm 1 is a subset of $C^o_{d_\epsilon}(r)$. The latter comprises of curve segments that have no point closer than $(d-\epsilon-\rho)$ to $C(t)$. Clearly, the offset path should be computed conservatively, to be a bit more than d. ϵ and ρ could be added to d, computing an offset approximation to a distance of $(d+\epsilon+\rho)$, only to ensure a minimal distance constraint of d.

A point on $C^o_{d_\epsilon}(r_0)$ is said to be *a match to point* $C(t_0)$ if it matches location $C(t_0)+N(t_0)d$. Denote by $\varDelta^2_d(t)$ the distance square between the matched point on $C^o_{d_\epsilon}(r)$ to $C(t)$ and $C(t)$. The need to exploit a positive ρ value stems from the fact that $d+\epsilon \geq \varDelta^2_d(t) \geq d-\epsilon$. For numerical stability, ρ should be as large as possible, reducing the chance of detecting matched points as self-intersections. In contrast, the larger ρ is, the bigger the likelihood that we will miss small self-intersections. In practice, ρ was selected to be between 95% and 99% of d.

By selecting $\rho > 0$, the curve segments that result from Algorithm 1 are not exactly connected. Instead, a sequence of curve segments is output with end points that are very close to each other. Numeric Newton Raphson marching steps at each such close pair of end points could very quickly converge to the exact self-intersection location. Very few steps are required to converge to the highly precise self-intersection location. In Section 3, we present several examples that demonstrate this entire procedure, including the aforementioned numerical marching stage.

3 Examples and Extensions

We now present several examples of trimming of both local and global self-intersections in offset curves, via the (square of the) distance map, $\varDelta^2_d(r,t)$. In Figure 2, the example from Figure 1 is presented again. In Figure 2 (a), the original curve and its offset are presented. In Figure 2 (b), the result of trimming the curve using the $\varDelta^2_d(r,t)$ functions is shown while Figure 2 (c) presents the same trimmed offset curve after the numerical marching stage.

In Figure 3, the log of the distance (square) function, $\varDelta^2_d(r,t)$, is presented for the curve in Figure 2. The minimal distance is in the order of the offset distance itself whereas the maximal distance is in the order of the diameter of the curve and hence, the figure is shown in a logarithmic scale. Also presented in Figure 3 are the zero set, $\mathcal{Z}$, of $\varDelta^2_d(r,t)-(d-\epsilon-\rho)^2$ and its projection, $\mathcal{Z}_r$, on the $t=0$ axis. In this case, the r axis is divided into four valid domains by

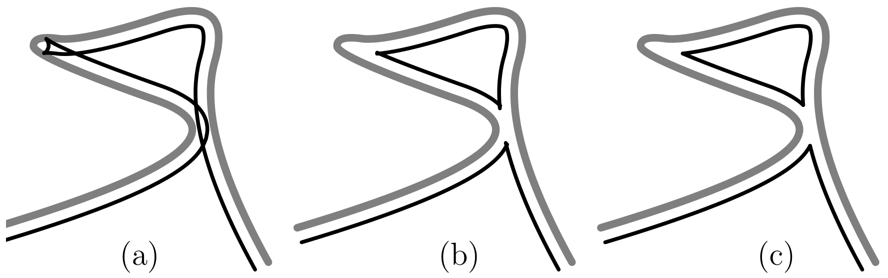

Fig. 2. A simple curve and its offset, from Figure 1. The curve and its offset are presented in (a). (b) is the result of ρ-trimming the curve using $\Delta_d^2(r,t)$ whereas, in (c), the result is improved using numerical marching.

three sub-regions that are self-intersecting. The first and third black sub-regions along the r axis are due to the global self-intersection of the curve in Figure 2, whereas the middle large black sub-region is due to the local self-intersection in the curve. Extracting the four valid sub-regions, we see the resulting curve segments in Figure 2 (b). A numerical marching step completes the computation in Figure 2 (c).

Figure 4 presents another example of a curve with several offsets to both directions. In Figure 4 (a), the original curve is shown (in gray) with the offsets. With the aid of the distance function square, $\Delta_d^2(r,t)$, the self-intersections are ρ-trimmed in (b), whereas the result of applying the numerical marching stage is presented in Figure 4 (c).

Figures 5 and 6 present two more complex examples. Here (a) is the original curve (in gray) and its offsets, and (b) is the result of ρ-trimming, $\rho = 95\%$, of the self-intersections with the aid of the $\Delta_d^2(r,t)$ function. In all the examples presented in this work, the trimming distance ρ was from 95% to 99% of the offset distance, with an offset tolerance about ten times better (i.e. offset accuracy of 99% to 99.9% of the offset distance). (c) and (d) in Figures 5 and 6 present the result of trimming at $\rho = 95\%$ and $\rho = 99\%$ of the offset distance, respectively. Small local self-intersections escape the ρ-trimming step at $\rho = 95\%$ but are properly trimmed at $\rho = 99\%$.

These small local self-intersections could clearly appear at any percentage of the ρ-trimming distance, below 100%. In many applications, such as robotics and CNC machining, local and arbitrary small self-intersections will enforce large accelerations along the derived path, and hence are highly undesired. One could employ the local self-intersection test presented in Equation (2) as another filtering step that should completely resolve such small local events.

The computation of the offset curves as well as the trimming of the curves in Figures 5 and 6 took about one minute on a modern PC workstation.

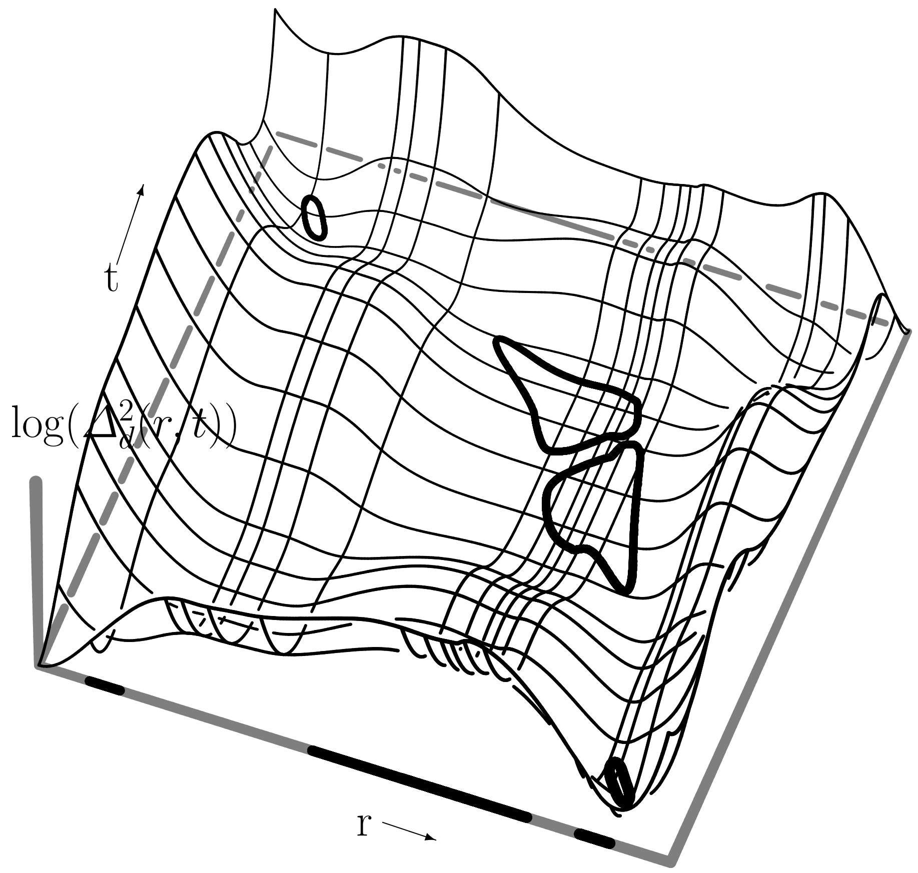

Fig. 3. The distance function, $\Delta_d^2(r,t)$, of the curve in Figure 2, on a logarithmic scale. Also shown, in thick lines, are the zeros, $\mathcal{Z}$, of $\Delta_d^2(r,t) - (d - \epsilon - \rho)^2$ as well as the projection of $\mathcal{Z}$ on the $t = 0$ axis.

4 Conclusions and Future Work

We have presented a robust and reasonably efficient scheme to trim both local and global self-intersections in offsets of freeform planar curves. No complete distance map was defined for the plane, which is highly complex computationally. Instead, the distance function was examined only for the neighborhood of the computed offset curve.

The presented scheme is robust in the sense that the trimmed offset curve that result is at least trimming distance apart, $(d - \epsilon - \rho)$, from the original curve. Nevertheless, since the trimming distance must be some finite distance smaller than the offset approximation, small loops such as those from local self-intersections might still exist and a combined scheme that employs an approach

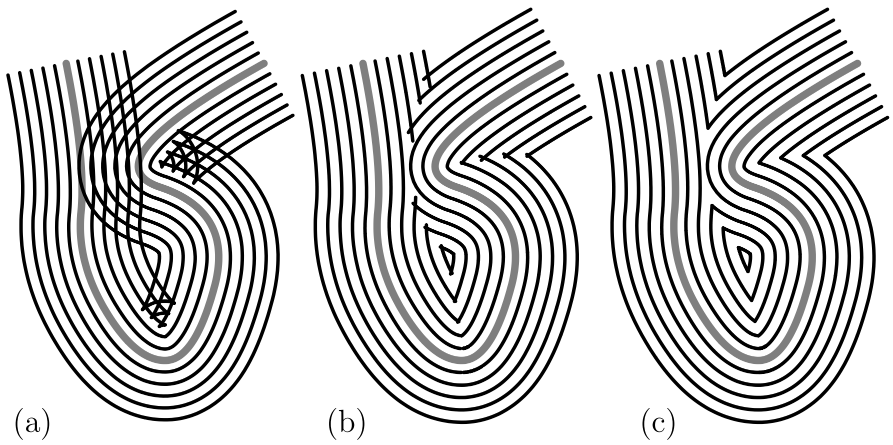

Fig. 4. Another example of a curve (in gray) and its offsets is shown in (a). (b) is the result of ρ-trimming the curve using $\Delta_d^2(r,t)$ and (c) is the result of applying the numerical marching stage.

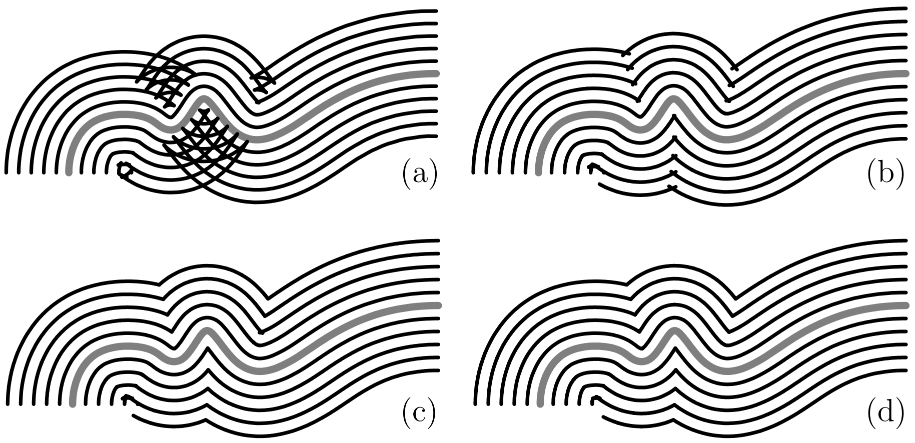

Fig. 5. (a) presents the original curve (in gray) and its offsets, while (b) is the result of ρ-trimming, $\rho = 95\%$, of the offset with the aid of $\Delta_d^2(r,t)$. (c) and (d) present two different ρ-trimming percentages of 95% and 99%, respectively, after a numerical marching stage.

similar to that of [3] to detect and eliminate local self-intersection, using Equation (2), could provide the complete solution.

The potential of extending this trimming offset approach to surfaces is an immediate issue that needs to be considered, in this context. While trimming of self-intersection of curves is considered a difficult problem, the question of trimming self-intersections in offsets surfaces is far more complex. An even more challenging question is the issue of trimming self-intersections in related applications such as the computation of bisector sheets. The topology of the self-

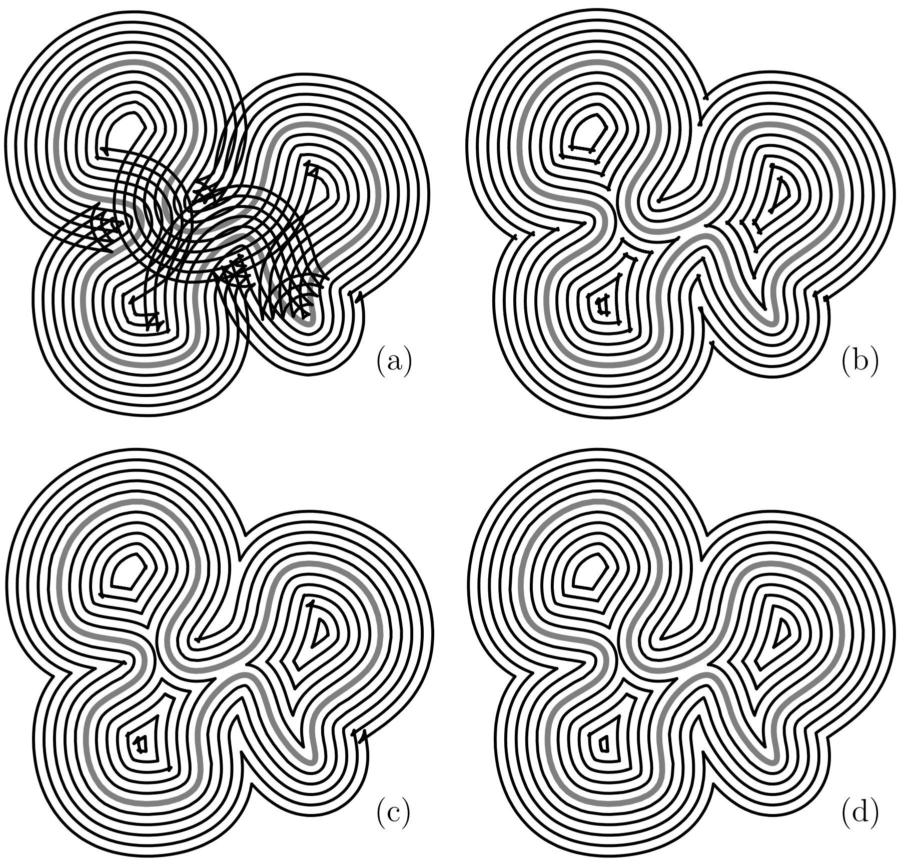

Fig. 6. (a) presents the original curve (in gray) and its offsets, while (b) is the result of ρ-trimming, $\rho = 95\%$, of the offset with the aid of $\varDelta_d^2(r,t)$. (c) and (d) present two different trimming percentages of 95% and 99%, respectively, with a numerical marching stage.

intersections in offsets of surfaces is exceptionally complex, which makes this problem extremely difficult to handle. Yet, we are hopeful that the distance map could aid simplifying this problem by globally examining the distance function. Let $S(u,v)$ and $S^o_{d_\epsilon}(u,v)$ be a freeform rational surface and its rational offset approximation with tolerance ϵ, and let

$$\varDelta_d^2(u,v,r,t) = \left\langle S(u,v) - S^o_{d_\epsilon}(r,t), S(u,v) - S^o_{d_\epsilon}(r,t) \right\rangle . \tag{4}$$

The zero set of $\varDelta_d^2(u,v,r,t) - (d - \epsilon - \rho)^2$, projected onto the (u,v) domain, will prescribe the domains of $S(u,v)$, which need to be purged. Trimming curves could then be constructed in the parametric domain of $S(u,v)$, representing the resulting ρ-trimmed offset as a trimmed tensor product surface. The question of numerically improving the ρ-trimming for offset surfaces is more difficult, and while probably feasible, will have to be dealt with more cautiously.

This same approach could also be employed to handle the trimming of self-intersections in *variable distance* offsets of curves and surfaces where we now seek the zero set of $\Delta_d^2(r,t) - (d(t) - \epsilon - \rho)^2$ and $\Delta_d^2(u,v,r,t) - (d(u,v) - \epsilon - \rho)^2$, with the distance being a function of the parametric location.

5 Acknowledgment

The implementation of the presented algorithm was conducted with the aid of the IRIT [9] modeling environment.

References

1. M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf. “Computational Geometry, Algorithms, and Applications (2nd ed.)”, Springer-Verlag, Berlin, 2000.
2. J. J. Chou. “NC Milling Machine Toolpath Generation for Regions Bounded by Free Form Curves and Surfaces.” PhD thesis, Department of Computer Science, University of Utah, April 1989.
3. G. Elber and E. Cohen. “Error Bounded Variable Distance Offset Operator for Free Form Curves and Surfaces.” International Journal of Computational Geometry & Applications, Vol 1, No 1, pp 67-78, March 1991.
4. G. Elber and E. Cohen. “Second Order Surface Analysis Using Hybrid Symbolic and Numeric Operators.” Transactions on Graphics, Vol 12, No 2, pp 160-178, April 1993.
5. G. Elber, I. K. Lee, and M. S. Kim. “Comparing Offset Curve Approximation Methods.” CG&A, Vol 17, No 3, pp 62-71, May-June 1997.
6. G. Elber and M. S. Kim. “Bisector Curves of Planar Rational Curves.” Computer Aided Design, Vol 30, No 14, pp 1089-1096, December 1998.
7. R. T. Farouki and Y. F. Tsai and G. F. Yuan. “Contour machining of free-form surfaces with real-time PH curve CNC interpolators”, Computer Aided Geometric Design, No 1, Vol 16, pp 61-76, 1999.
8. “Pocket Machining Based on Countour-Parallel Tool Paths Generated by Means of Proximity Maps.” Computer Aided Design. Vol 26, No 3, pp 189-203. 1994.
9. IRIT 8.0 User’s Manual. The Technion—IIT, Haifa, Israel, 2000. Available at `http://www.cs.technion.ac.il/~irit`.
10. I. K. Lee, and M. S. Kim, and G. Elber. “Planar Curve Offset Based on Circle Approximation.” Computer Aided Design, Vol 28, No 8, pp 617-630, August 1996.
11. Y. M. Li and V. Y. Hsu. “Curve offsetting based on Legendre series.” Computer Aided Geometric Design, Vol 15, No 7, pp 711-720, 1998.
12. M. Peternell and H. Pottmann. “A Laguerre geometric approach to rational offsets.” Computer Aided Geometric Design, Vol 15, No 3, pp 223-249, 1998.
13. H. Persson. “NC machining of arbitrary shaped pockets”. Computer Aided Design. Vol 10, No 3, pp 169-174. 1978.