Reliable Computing **00:** 1–16, 2002.

Robust Approximation of Offsets, Bisectors, and Medial Axes of Plane Curves *

João Batista Oliveira (oliveira@inf.pucrs.br)
Faculdade de Informática, Pontifícia Universidade Católica do Rio Grande do Sul
Av. Ipiranga 6681, 90619-900 Porto Alegre, RS, Brazil

Luiz Henrique de Figueiredo (lhf@impa.br)
IMPA–Instituto de Matemática Pura e Aplicada
Estrada Dona Castorina 110, 22461-320 Rio de Janeiro, RJ, Brazil

Received: ......; Accepted: ......

Abstract. Most methods for computing offsets, bisectors, and medial axes of parametric curves in the plane are based on a local formulation of the distance to a curve. As a consequence, the computed objects may contain spurious parts and components, and have to be trimmed. We approach these problems as global optimization problems, and solve them using interval arithmetic, thus generating robust approximations that need not be trimmed.

Keywords: parametric curves; offset curves; global optimization; interval arithmetic; range analysis.

1. Introduction

Industrial applications often need to perform complex geometric processing on existing forms. For instance, the offsets of a curve play a crucial role in tool path generation for numerical-control machining, robot path planning, and tolerance analysis. Offsets are rather complex geometrical entities, as we shall see below.

Given a curve Γ in the plane $\mathbf{R}^2$ and a positive number r, the *r-offset* of Γ is the set O of all points in the plane whose distance to Γ is r:

$$O = \{p \in \mathbf{R}^2 : d(p,\Gamma) = r\}.$$

The distance $d(p,\Gamma)$ of a point p to Γ is the smallest distance of p to a point of Γ:

$$d(p,\Gamma) = \min\{d(p,q) : q \in \Gamma\},$$

where $d(p,q)$ is the ordinary Euclidean distance between two points p and q in the plane.

From the formulation above, it is clear that finding offset curves is a *global* problem, because a *global minimum* of the distance is sought. In practice, the

* Revised and expanded version of "Robust Approximation of Offsets and Bisectors of Plane Curves", *Proceedings of SIBGRAPI 2000 (XIV Brazilian Symposium on Computer Graphics and Image Processing)*, pp. 139–145 © 2000 IEEE.

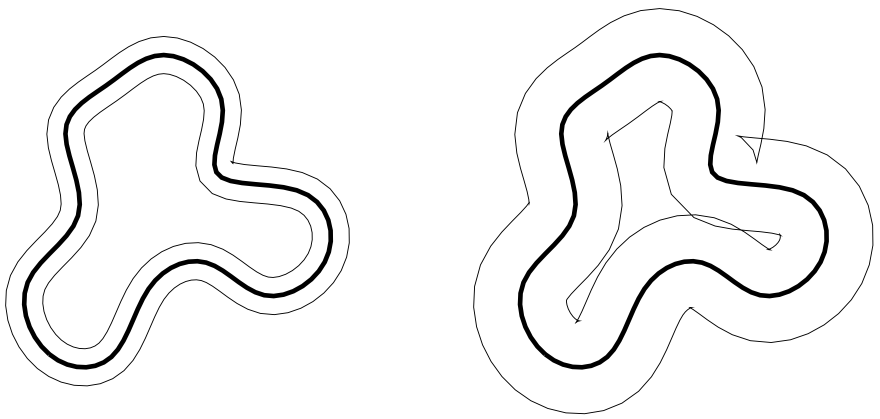

Figure 1. Parametric offsets are correct near the curve (left), but not far away (right) (extracted from Snyder [35]).

curve Γ is given in parametric form, that is, as the *trace* $\gamma(I)$ of a parametric curve $\gamma\colon I \subseteq \mathbf{R} \to \mathbf{R}^2$. Computing the distance of a point p to Γ is then equivalent to solving a global minimization problem on the interval I:

$$d(p,\Gamma) = \min\{d(p,\gamma(t)) : t \in I\}.$$

In general, global minimization problems are seen as hard to solve, and local formulations are used instead.

The natural local formulation for the offset of a parametric curve γ is to measure the distance to γ along its normal, thus obtaining the offset in parametric form over I as

$$O(t) = \gamma(t) \pm rN(t),$$

where $N(t)$ is the unit normal vector at $\gamma(t)$. If $\gamma(t) = (x(t), y(t))$, then $N(t) = \frac{1}{\|\gamma'(t)\|}(-y'(t), x'(t))$, where $\|\gamma'(t)\| = \sqrt{x'(t)^2 + y'(t)^2}$.

This local formulation works well when r is small, but when r is large the parametric offset has self-intersections and does not correspond to the true, global offset defined above (see Figure 1). (The maximum value of r for which the parametric offset is correct is the radius of the largest *tubular neighborhood* around Γ, an elusive global property of Γ related to the smallest radius of curvature of a point in Γ.) Thus, local formulations are easy—as long as you know how to compute the normal vector $N(t)$—but require a complicated post-processing *trimming* step that identifies and removes extraneous pieces [12, 11].

In this paper, we approach geometric processing problems based on distance as the global optimization problems they are, and obtain robust approximations for offsets and other related constructions by using interval arithmetic [27].

The paper is organized as follows: We start by discussing the role of range analysis in global geometric problems in Section 2. We then explain briefly how range analysis can be implemented with interval arithmetic in Section 3. In Section 4 we describe in detail how these ideas can be used in a robust algorithm for computing offsets of parametric curves. This algorithm is then extended to handle point/curve bisectors in Section 5 and medial axes in Section 6. The paper ends with a discussion of previous, related work in Section 7 in the light of our approach and with some directions for future work in Section 8.

2. Range analysis and global geometric processing

Range analysis is the study of the global behavior of real functions based on estimates for their set of values. Given a function $f\colon \Omega \subseteq \mathbf{R}^n \to \mathbf{R}$, range analysis methods provide an *inclusion function* for f, that is, a function F defined on the subsets X of Ω such that

$$F(X) \supseteq f(X) = \{f(x) : x \in X\}.$$

Thus, $F(X)$ is an estimate for the *complete* set of values taken by f on X. The estimate is usually an interval, which is not required to be tight; so, $F(X)$ may be strictly larger than $f(X)$. Nevertheless, $F(X) \supseteq f(X)$ implies that $\min F(X) \leq \min f(X)$, and so if $r \leq \min F(X)$, then $r \leq f(x)$, for *all* points $x \in X$. Similarly, if $r \geq \max F(X)$, then $r \geq f(x)$, for all $x \in X$.

We shall see in Section 3 that interval arithmetic [27] is the natural computational tool for range analysis. For the moment, we show how inclusion functions can be used for solving global geometric processing problems.

Suppose that we need to find the r-offset O of a parametric plane curve $\gamma\colon I \subseteq \mathbf{R} \to \mathbf{R}^2$ inside a region $\Omega \subseteq \mathbf{R}^2$ that contains the trace Γ of γ. To compute an approximation for O in Ω, we perform a recursive exploration of Ω, discarding subregions X of Ω when we can *prove* that X does not contain any part of the offset O. This proof relies on an inclusion function for the distance function $d(X,\Gamma)$: if r is outside the interval estimate provided by this inclusion function, then X *cannot* contain a part of O.

The actual algorithm for approximating the offset O is more complicated than that because we cannot easily compute an inclusion function for the distance function $d(X,\Gamma)$. Instead, we perform a global optimization sub-algorithm to minimize $d(X,\gamma(t))$ for $t \in I$ that incorporates testing r in its rejection criteria. The details are given in Section 4.

Here is a skeleton of the algorithm described above. This skeleton is typical of interval methods [28] and works for other global geometric processing problems.

```
explore(X):
    if X does not contain a part of O then
        discard X
    elseif X is small enough then
        output X
    else
        divide X into smaller pieces X_i
        for each i, explore(X_i)
```

We start the exploration with a call to explore(Ω). We may assume that Ω is a rectangle and that X is divided into rectangles too. A typical choice is to divide X into four equal rectangles, thus generating a quadtree [33, 32]. The exploration stops when X is small enough, which we interpret as meaning that $\mathrm{diam}(X) < \varepsilon$, where ε is a user-specified tolerance. The result of the algorithm is a list of boxes whose union is *guaranteed* to contain the offset O within the tolerance ε.

The crucial step in the algorithm is the first one: testing whether X does not contain a part of O. This step requires the solution of global optimization problems on the curve γ using X as input data. We shall see in Section 4.2 that these optimization problems need not be solved from scratch for each X. This is important for the overall efficiency of the approximation algorithm.

3. Interval arithmetic

Interval arithmetic (IA) [27] is probably the most important technique for range analysis currently in use, even though it was originally introduced to improve the reliability of numerical computations through automatic error control. IA is a simple and reliable technique of probing a function over an entire domain, obtaining information used in a variety of algorithms and avoiding risky point sampling. Interval arithmetic has been used successfully in several computer graphics problems [28, 38, 24, 37, 36, 35, 8, 1].

IA provides robust bounds for the range of functions by representing ranges of values as intervals and providing basic arithmetic operations as well as elementary functions to operate on them. Here are some examples of interval operations:

$$
\begin{aligned}
[a,b]+[c,d] &= [a+c, b+d] \\
[a,b]\times[c,d] &= [\min(ac,ad,bc,bd), \max(ac,ad,bc,bd)] \\
[a,b]^2 &= \begin{cases} [0, \max(a^2,b^2)], & a \le 0 \le b \\ [\min(a^2,b^2), \max(a^2,b^2)], & \text{otherwise} \end{cases} \\
\exp([a,b]) &= [\exp(a), \exp(b)].
\end{aligned}
$$

Similar formulas can be given for all other elementary operations and functions. Using this observation, Moore [27] proved that every computable function f can be extended to an interval function F, called its *natural interval extension*, simply by replacing the operations used to evaluate f with their equivalent interval operations. (This can be done automatically using operator overloading.) Moore also proved that the natural interval extension F is an inclusion function for f, which shows that interval arithmetic can be used for range analysis. There are several packages for interval arithmetic in the Internet [21].

The range estimates obtained with interval arithmetic can be much wider than the exact ranges because interval formulas, such as the ones above, assume that all operands are independent. IA gives pessimistic estimates when some dependency between operands exists. Therefore, we cannot use the IA estimates to guarantee that f *takes* a certain value in an interval, but we can use these estimates to assure that f does *not* take other values. So, even rather pessimistic estimates still assure that values outside that range cannot be achieved by f. This safety is the cornerstone of interval algorithms.

4. Robust approximation of offsets

Now that we have seen that range analysis can be easily implemented, we return to the computation of offsets using the skeleton algorithm described in Section 2 for approximating the r-offset of a parametric curve γ inside a region Ω.

Recall that the main step is to test whether a subregion X of Ω contains a part of the offset O. As discussed in Section 2, we discard subregions that we can prove to contain no part of O. Such subregions are called *empty*. Range analysis is essential for this proof, as we discuss below.

To test whether a subregion X is empty, we recursively explore the curve domain I looking for pieces of Γ whose distance to X is close to r. If no such pieces are found, then we can discard X. The main step in this exploration of I is an interval estimate of the distance between X and $\gamma(T)$. More precisely, we use the natural interval extension $D(X,T)$ of the distance function

$$d(p,\gamma(t)) = \sqrt{(x-x(t))^2 + (y-y(t))^2},$$

for $p=(x,y) \in X \subseteq \Omega$ and $t \in T \subseteq I$. The value of $D(X,T)$, computed on a rectangle X and an interval T, is an interval that contains *all* distances from points in X to points on $\gamma(T)$. We compare $D(X,T)$ with r as follows:

– If the distance from X to $\gamma(T)$ is strictly less than r, then we can stop the exploration of I, because we have proved that the distance from X to Γ is strictly less than r. Thus, we can discard X because it cannot contain a part of the offset O.

– If the distance from X to $\gamma(T)$ is strictly greater than r, then we can discard T in the exploration of I, because we have proved that $\gamma(T)$ cannot induce a piece of the offset O inside X. We cannot discard X at this point because there may still be a piece of O in X, induced by a different subinterval of I. However, if *all* sub-intervals T visited during the exploration of I are discarded for this reason, then we *can* discard X, because we have proved that the distance from X to Γ is strictly greater than r.

– If neither X nor T can be discarded, then T is bisected and the two pieces are explored recursively. The recursion stops when $\gamma(T)$ is small; at this moment, we have found a point of Γ at distance (approximately) r from X. To test whether $\gamma(T)$ is small, we use an interval estimate $G(T)$ for $\gamma(T)$. This estimate is a box $X(T) \times Y(T)$, where X and Y are the inclusion functions for the components of γ that were used to compute $D(X,T)$. Then, $\gamma(T)$ is small when $\operatorname{diam} G(T) < \varepsilon$.

The core of the algorithm is thus the function below:

```
test(X,T,r):
    if max D(X,T) < r then
        return true
    if min D(X,T) > r then
        return false
    if diam G(T) < ε then
        all ← false
        return false
    else
        bisect T into T1 and T2
        return test(X,T1,r) ∨ test(X,T2,r)
```

Note how we compare the distance of X to $\gamma(T)$ using the interval estimate $D(X,T)$, as described in Section 2. Note also that the exploration of I stops as soon as we find T such that $\max D(X,T) < r$, assuming that the "or" operator $\vee$ is short-circuited. The flag "*all*" controls whether all pieces of Γ visited during the exploration of I have distance to X greater than r, as discussed above.

This algorithm is a kind of interval global optimization algorithm [15], except that it does not always need to *find* the global minimum of the distance, but instead is guided only by the need to find a minimum distance close to r. For regions X that are far away from Γ, the algorithm never actually computes the distance of X to Γ, because it stops as soon as it establishes that this distance is greater than r.

The complete algorithm for computing a robust approximation of the offset O within tolerance ε using a quadtree exploration of the region Ω is then:

```
explore(X):
    if empty(X,r) then
        discard X
    elseif diam(X) < ε then
        output X
    else
        divide X into four equal pieces X_i
        for each i, explore(X_i)

empty(X,r):
    all ← true
    return test(X,I,r) ∨ all
```

Starting with a call to explore(Ω), this algorithm performs an adaptive exploration of Ω, quickly discarding subregions that cannot contain the offset O, and working harder near O, where it needs to, so that it can approximate O well. Figure 2 shows a series of offsets with decreasing radius for the same base curve (a "butterfly" shape). Figure 3 shows a typical quadtree decomposition of Ω.

4.1. Reconstructing offset curves

As mentioned in Section 2, the output of algorithm explore is a list of boxes whose union is *guaranteed* to contain the offset O within the tolerance ε. For some applications, simply drawing these boxes will be enough. The base curve in Figure 3 and all curves in Figure 2 are drawn this way.

For other applications, such as modeling, the boxes must be processed and transformed into polygonal lines or curves. Gleicher and Kass [14] faced this very same problem when computing the intersection curve of parametric surfaces using interval methods, and solved it by using a "stringing" algorithm. However, it is probably simpler to select the midpoint of each box and then reconstruct the offset curve using one of the good algorithms that have appeared recently [7, 22]. If needed, the reconstructed polygonal curve can then be approximated with B-splines [34].

4.2. Cache trees

A large fraction of the cost of algorithm explore is the cost of computing the interval estimates $D(X,T)$ and $G(T)$. The interval estimate $G(T)$ is used twice in test: once in the computation of $D(X,T)$ and once for testing whether $\gamma(T)$ is small. Since $G(T)$ does not depend on X, it is natural to *cache* the values of $G(T)$ and reuse them across calls to test. We now describe briefly how to organize this cache.

Since the domain I is always explored in the same way in test, that is, by starting with $T = I$ and bisecting each T that cannot be discarded, we

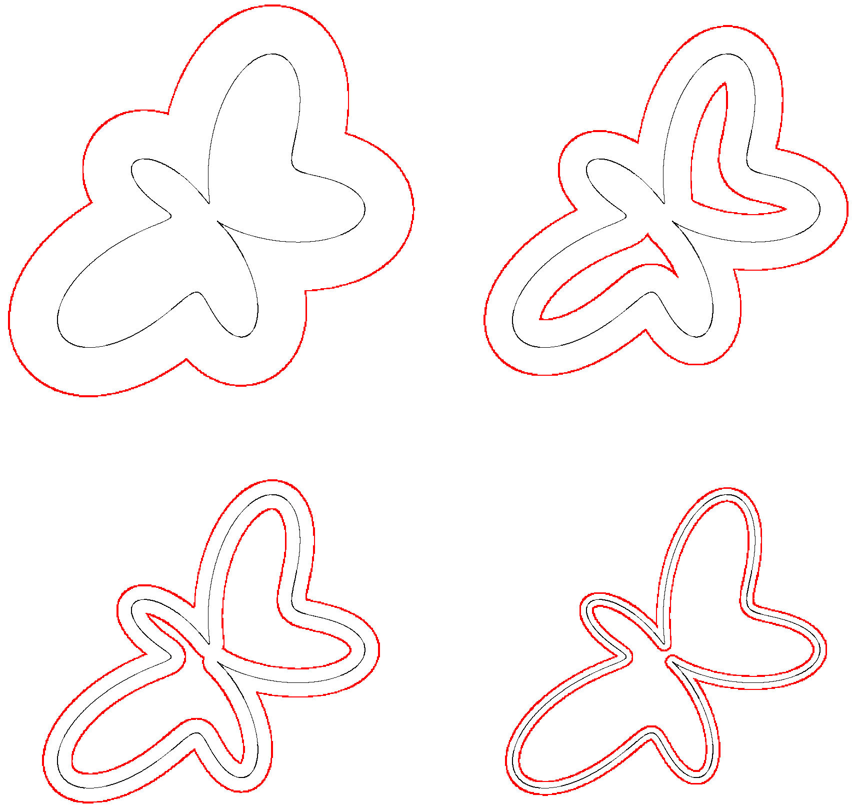

Figure 2. Offsets of decreasing radius.

use a simple binary tree to cache the values of G. The nodes in this *cache tree* correspond to the sub-intervals T of I visited during the repeated explorations of I, for varying X. The root node corresponds to $T = I$. In each node, we store the box $G(T)$, which is guaranteed to contain $\gamma(T)$, and possibly pointers to children nodes corresponding to T_1 and T_2, the two halves of T created for recursion. To reduce overestimation, the estimates in a node are updated (from the bottom up) to be the smallest box that contains the estimates of its children nodes (this is a simple min/max computation because only rectangles are involved).

Because I does not always need to be fully explored, the cache tree is in effect a dynamic adaptive representation of the function γ on I: it summarizes the behavior of γ at various resolution scales, and gets locally refined as needed when X varies.

To compute the approximation shown in Figure 3, 220089 evaluations of G were required, but 218618 of these were read from the cache; only 1471 fresh

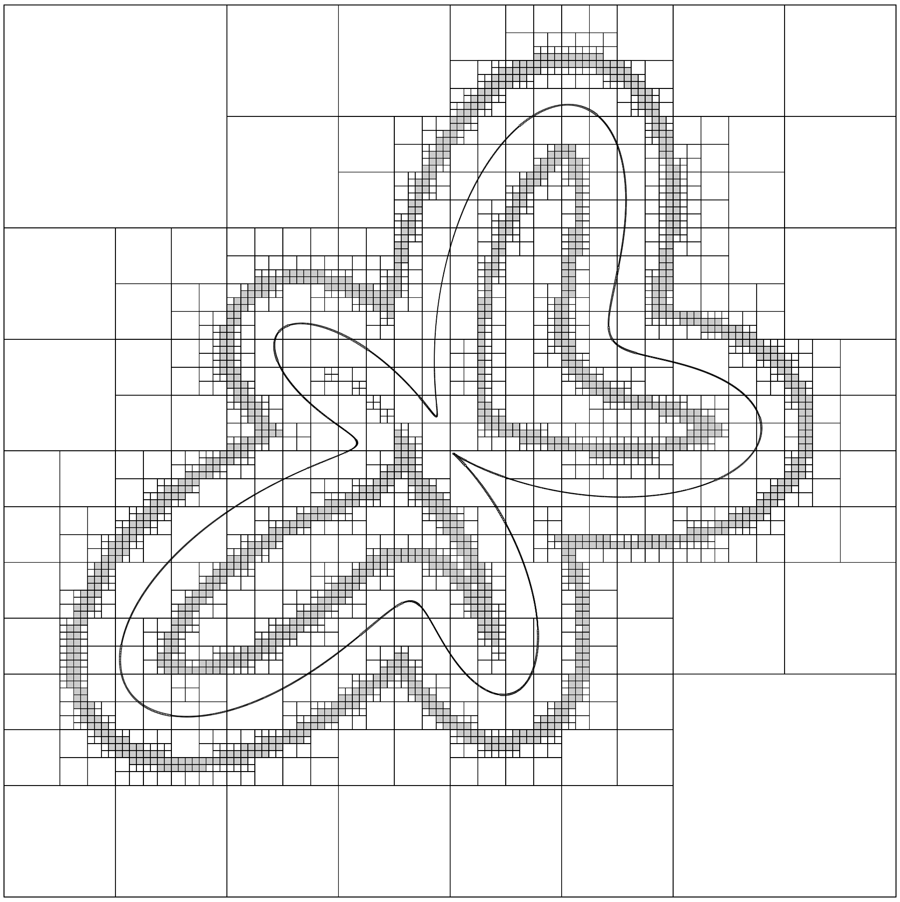

Figure 3. Adaptive approximation of an offset curve.

evaluations were required. So, in this case, the cache contained the answer to more than 99% of all evaluations. The use of a cache in our implementation allowed interactive response times as we varied the offset radius r. The use of cache trees in interval methods has many other applications.

5. Point/curve bisectors

Bisectors are another example of geometric objects that are defined globally. Given a curve Γ and a point p_0 in the plane, the *bisector* of Γ and p_0 is the set $\mathcal{B}$ of all points in the plane whose distance to Γ and to p_0 are the same:

$$\mathcal{B} = \{p \in \mathbf{R}^2 : d(p,\Gamma) = d(p,p_0)\}.$$

When Γ is given as the trace of a parametric curve γ, a local formulation for the bisector is

$$\mathcal{B}(t) = \gamma(t) \pm r(t)N(t),$$

where $N(t)$ is the unit normal vector at $\gamma(t)$ and

$$r(t) = \frac{\|p_0 - \gamma(t)\|^2}{2\langle p_0 - \gamma(t), N(t)\rangle}.$$

Thus, bisectors can be seen as "variable-distance" offsets [10]. As in the case of offsets, this local formulation results in curves that need to be trimmed into the true, global bisector $\mathcal{B}$.

Given the similarity between offsets and bisectors, it is not surprising that the method described in Section 4 can be adapted to compute bisectors. What is somewhat surprising is that it is almost trivial to do so: All we need to do is to compare $D(X,T)$ with $D(X,p_0)$, the set of all distances from points in X to p_0. As in the case of offsets, we use the natural interval extension of the distance function $d(p,p_0) = \sqrt{(x-x_0)^2 + (y-y_0)^2}$, for $p = (x,y) \in X$ and $p_0 = (x_0,y_0)$. In this case, the interval evaluation gives the *exact* value, with no overestimation, because the variables x and y occur only once in the expression above [27]. (In the expression for $D(X,T)$, the parameter t occurs many times, and so overestimation is expected.)

Using the lines below instead of the corresponding lines in the test function of Section 4 is all that is needed to computed bisectors instead of offsets:

```
if max D(X,T) < min D(X,p0) then
    return true
if min D(X,T) > max D(X,p0) then
    return false
```

Some bisectors computed with this modified algorithm are shown in Figure 4. Again, the use of a cache was essential for good performance; the cache contained the answer to more than 98% of all evaluations.

6. Medial axes

Medial axes are still another example of geometric objects that are defined globally. Intuitively, the medial axis of a curve Γ in the plane is a kind of "self-bisector" of Γ. More precisely, the *medial axis* of Γ is the set $\mathcal{M}$ of all points in the plane that are at the same (minimum) distance to two different points p_1 and p_2 in Γ:

$$\mathcal{M} = \{p \in \mathbf{R}^2 : d(p,p_1) = d(p,\Gamma) = d(p,p_2),\ p_1,p_2 \in \Gamma,\ p_1 \neq p_2\}.$$

Among the many equivalent definitions of medial axis [25], this one is best suited to algorithmic framework we have been discussing.

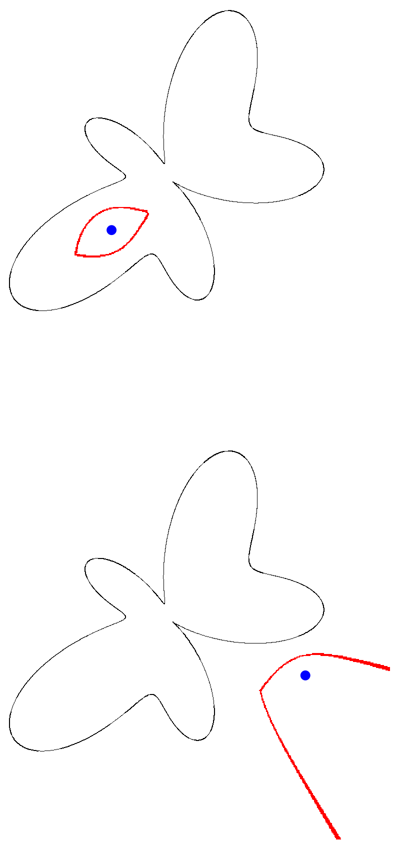

Figure 4. Point/curve bisectors for various points.

The geometric similarity between bisectors and medial axes suggests that the method described in Section 5 for computing bisectors can be adapted to compute the medial axis of a curve. The main structure of the algorithm remains the same: perform a quadtree subdivision of Ω, testing each box X to determine whether there are no points of $\mathcal{M}$ in X, discarding X if so, and otherwise either outputting or subdividing X depending on its size. Two important details differ, however.

First, the distance from X to Γ is still estimated using a global optimization algorithm on I, but now we need to perform a *complete* global optimization, because the search cannot be guided by a priori bounds, as it was for offsets and bisectors. Second, the important information to be obtained from the optimization process is not actual distance from X to Γ but rather whether this distance can be attained at more than one point in Γ. If there is only one minimum point, then by definition there are no points of $\mathcal{M}$ in X, and X can be discarded.

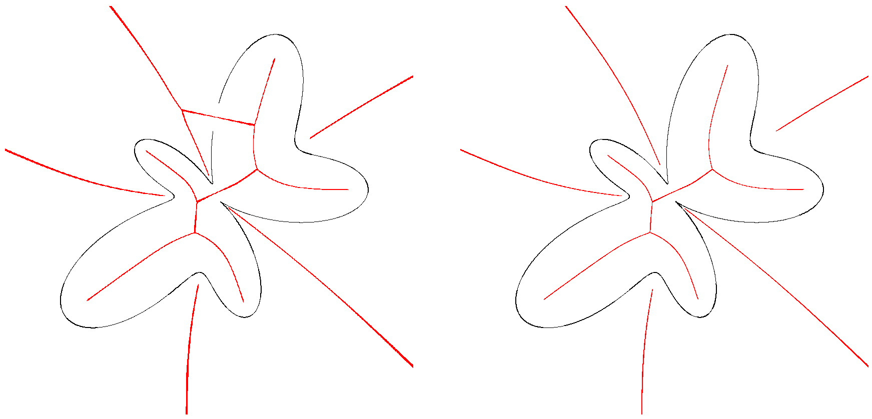

Figure 5. Medial axes for an open curve (left) and for a closed curve (right).

To decide whether the distance from X to Γ is attained at more than one point in Γ, we perform *two* global optimizations on I, one in each direction. More precisely, when we have decided to subdivide $T \subseteq I$ to refine the search for the minimum point, in the first run we give priority to the left half of T and in the second run we give priority to the right half of T. In both cases, we return the first minimum point found. Thus, if both runs give the same point, then X contains no point of $\mathcal{M}$. Except for this important detail, the algorithm on I is a typical interval algorithm for global optimization [15].

Figure 5 (left) shows a medial axis computed with the algorithm described above. Unlike the previous examples, we now show an *open* curve: Our medial axis algorithm will have trouble with closed curves, in the parts of $\mathcal{M}$ that are related to the closing point of Γ, because both optimization processes will report the same point. Closed curves can be handled correctly by subdividing I into a few subintervals: this will find local minima in each subinterval, which are then filtered to select the global minima. Figure 5 (right) shows a medial axis for a closed curve computed with this modified algorithm. In both cases, the use of a cache was essential for good performance, as it contained the answer to more than 95% of all evaluations.

7. Related work

Several methods for approximating offset curves and surfaces were reviewed in a survey by Pham [29], which was recently updated by Maekawa [23]. Most methods are of a local nature and work hard at trimming procedures.

A prime example of the local approach is the work of Farouki and Neff [12, 11], who did an in-depth study of analytic and algebraic properties of offsets of algebraic parametric curves in the plane. They gave algebraic implicit equations for the offset and procedures for identifying self-intersections, which together allow the computation of trimmed offsets. Farouki and Johnstone [10] did a similar study for point/curve bisectors.

Of special interest for CAD applications are B-spline approximations of offsets, because even when the base curve is given in B-spline form, its offsets cannot in general be given in B-spline form too. Elber et al. [9] compared several B-splines offset approximation methods.

Chiang et al. [2] approached the computation of offsets of a curve Γ as a global problem and extracted the offset as a level set of the distance function to Γ. They discretized the curve Γ and the region Ω uniformly and solved the eikonal equation on these grids. (The eikonal equation models wavefront propagation in fluid dynamics.) A similar approach was used by Kimmel and Bruckstein [19].

The techniques for obtaining medial axes from shapes can be roughly classified into three areas [20]: skeleton extraction from images, skeleton extraction from polygons, and skeleton extraction from general, smooth shapes. Algorithms for extracting skeletons from images were the first to be developed, because images are simpler than analytical shape descriptions and because such skeletons have immediate use in pattern recognition. Next came algorithms for computing medial axes for polygons, aiming at exact solutions (for a recent algorithm that uses Voronoi diagrams, see [18]). Medial axes of general, smooth shapes are much harder. For a recent paper on the subject, which also contains a good review of previous work in the area, see [31]. However, the geometrical details and subtleties of medial axes produced by shapes described analytically still pose rather difficult questions [13]. These problems include handling open or self-intersecting curves, curves with multiple components, and trimming curves. In an effort to handle these cases more conveniently, authors have turned to non-conventional geometries to represent shapes more conveniently [26, 30].

In contrast to all these approaches, our approach adaptively samples both Γ and Ω, avoiding error-prone choices of grid sizes and simultaneously computing a guaranteed approximation that does not require trimming. Moreover, our method works for curves that are not necessarily smooth, because we do not need a normal vector defined at each point of the curve. Actually, the method works even for discontinuous curves: all we need are inclusion functions for each continuous piece.

Closest to our approach is the work of Snyder [35], who described a framework for geometric modeling based on interval methods for the reliable solution of systems of constraints and global optimization problems. However, Snyder first finds a tight interval estimate $D(X,T)$ using global optimization, and then compares it with r to decide whether to discard X, whereas we test r as we explore the parameter domain I, thus avoiding a complete global optimization at each step. Moreover, Snyder does not use a cache for accelerating the search and does not develop the computation of bisectors or medial axes.

As mentioned in Section 4.1, our method needs post-processing for applications that need a geometric model of the result. This characteristic is shared with other interval methods for geometric modeling. We feel that the reconstruction methods mentioned in Section 4.1 should give good results, but testing these methods was beyond the scope of this paper.

8. Future work

The dependency problem mentioned in Section 3 directly affects the efficiency of interval algorithms that use IA: the larger the overestimation, the longer it will take to discard subregions. It is natural then to consider alternatives to IA that suffer less from the dependency problem and can provide tighter estimates. Affine arithmetic [3] is one of these tools, and its use in interval methods has resulted in faster algorithms for several problems in computer graphics [6, 5, 17, 16, 4]. A natural next step is to use affine arithmetic instead of interval arithmetic in the global processing algorithms we have described here. We expect that performance will be improved, specially when computing estimates $G(T)$ for pieces of the curve and the corresponding distance estimates $D(X,T)$.

Acknowledgements

This research was done while J. B. Oliveira was visiting the Visgraf laboratory at IMPA during IMPA's summer post-doctoral program in 2000. Visgraf is sponsored by CNPq, FAPERJ, FINEP, and IBM Brasil. L. H. de Figueiredo is a member of Visgraf and is partially supported by a CNPq research grant.

References

1. Barth, W., R. Lieger, and M. Schindler: 1994, 'Ray Tracing General Parametric Surfaces Using Interval Arithmetic'. *The Visual Computer* **10**(7), 363–371.
2. Chiang, C.-S., C. M. Hoffmann, and R. E. Lynch: 1991, 'How to compute offsets without self-intersection'. Technical Report CSD-TR-91-072, Department of Computer Sciences, Purdue University.
3. Comba, J. L. D. and J. Stolfi: 1993, 'Affine arithmetic and its applications to computer graphics'. In: *Proceedings of SIBGRAPI'93*. pp. 9–18.
4. de Cusatis Jr., A., L. H. de Figueiredo, and M. Gattass: 1999, 'Interval methods for ray casting implicit surfaces with affine arithmetic'. In: *Proceedings of SIBGRAPI'99*. pp. 65–71.

5. de Figueiredo, L. H.: 1996, 'Surface intersection using affine arithmetic'. In: *Proceedings of Graphics Interface'96*. pp. 168–175.
6. de Figueiredo, L. H. and J. Stolfi: 1996, 'Adaptive enumeration of implicit surfaces with affine arithmetic'. *Computer Graphics Forum* **15**(5), 287–296.
7. Dey, T. K., K. Mehlhorn, and E. A. Ramos: 2000, 'Curve reconstruction: connecting dots with good reason'. *Computational Geometry: Theory and Applications* **15**(4), 229–244.
8. Duff, T.: 1992, 'Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry'. *Computer Graphics* **26**(2), 131–138 (SIGGRAPH'92 Proceedings).
9. Elber, G., I.-K. Lee, and M.-S. Kim: 1997, 'Comparing Offset Curve Approximation Methods'. *IEEE Computer Graphics & Applications* **17**(3), 62–71.
10. Farouki, R. T. and J. K. Johnstone: 1994, 'The bisector of a point and a plane parametric curve'. *Computer Aided Geometric Design* **11**(2), 117–151.
11. Farouki, R. T. and C. A. Neff: 1990a, 'Algebraic properties of plane offset curves'. *Computer Aided Geometric Design* **7**(1-4), 101–127.
12. Farouki, R. T. and C. A. Neff: 1990b, 'Analytic properties of plane offset curves'. *Computer Aided Geometric Design* **7**(1-4), 83–99.
13. Farouki, R. T. and R. Ramamurthy: 1998, 'Degenerate point/curve and curve/curve bisectors arising in medial axis computations for planar domains with curved boundaries'. *Computer Aided Geometric Design* **15**(6), 615–635.
14. Gleicher, M. and M. Kass: 1992, 'An interval refinement technique for surface intersection'. In: *Proceedings of Graphics Interface'92*. pp. 242–249.
15. Hansen, E.: 1992, *Global Optimization using Interval Analysis*. Marcel Dekker.
16. Heidrich, W. and H.-P. Seidel: 1998, 'Ray-Tracing Procedural Displacement Shaders'. In: *Proceedings of Graphics Interface'98*. pp. 8–16.
17. Heidrich, W., P. Slusallik, and H.-P. Seidel: 1998, 'Sampling procedural shaders using affine arithmetic'. *ACM Transactions on Graphics* **17**(3), 158–176.
18. Kim, D.-S.: 1998, 'Polygon offsetting using a Voronoi diagram and two stacks'. *Computer-Aided Design* **30**(14), 1069–1076.
19. Kimmel, R. and A. M. Bruckstein: 1993, 'Shape offsets via level sets'. *Computer-Aided Design* **25**(5), 154–162.
20. Kimmel, R., D. Shaked, and N. Kiryati: 1995, 'Skeletonization via distance maps and level sets'. *Computer Vision and Image Understanding* **62**(3), 382–391.
21. Kreinovich, V., 'Interval Software'. `http://cs.utep.edu/interval-comp/intsoft.html`.
22. Lee, I.-K.: 2000, 'Curve reconstruction from unorganized points'. *Computer Aided Geometric Design* **17**(2), 161–177.
23. Maekawa, T.: 1999, 'An overview of offset curves and surfaces'. *Computer-Aided Design* **31**(3), 165–173.
24. Mitchell, D. P.: 1990, 'Robust Ray Intersection with Interval Arithmetic'. In: *Proceedings of Graphics Interface'90*. pp. 68–74.
25. Montanari, U.: 1968, 'A Method for Obtaining Skeletons Using a Quasi-Euclidean Distance'. *Journal of the ACM* **15**(4), 600–624.
26. Moon, H. P.: 1999, 'Minkowski Pythagorean hodographs'. *Computer Aided Geometric Design* **16**(8), 739–753.
27. Moore, R. E.: 1966, *Interval Analysis*. Prentice-Hall.
28. Mudur, S. P. and P. A. Koparkar: 1984, 'Interval Methods for Processing Geometric Objects'. *IEEE Computer Graphics & Applications* **4**(2), 7–17.
29. Pham, B.: 1992, 'Offset curves and surfaces: a brief survey'. *Computer-Aided Design* **24**(4), 223–229.

30. Pottmann, H. and M. Peternell: 1998, 'Applications of Laguerre geometry in CAGD'. *Computer Aided Geometric Design* **15**(2), 165–186.
31. Ramanathan, M. and B. Gurumoorthy: 2002, 'Constructing medial axis transform of planar domains with curved boundaries'. To appear in *Computer-Aided Design*.
32. Samet, H.: 1990a, *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*. Addison-Wesley.
33. Samet, H.: 1990b, *The Design and Analysis of Spatial Data Structures*. Addison-Wesley.
34. Saux, E. and M. Daniel: 1999, 'Data reduction of polygonal curves using B-splines'. *Computer-Aided Design* **31**(8), 507–515.
35. Snyder, J. M.: 1992a, *Generative Modeling for Computer Graphics and CAD*. Academic Press.
36. Snyder, J. M.: 1992b, 'Interval analysis for computer graphics'. *Computer Graphics* **26**(2), 121–130 (SIGGRAPH'92 Proceedings).
37. Suffern, K. G. and E. D. Fackerell: 1991, 'Interval methods in computer graphics'. *Computers and Graphics* **15**(3), 331–340.
38. Toth, D. L.: 1985, 'On Ray Tracing Parametric Surfaces'. *Computer Graphics* **19**(3), 171–179 (SIGGRAPH'85 Proceedings).