Computational and Structural Advantages of Circular Boundary Representation*

O. Aichholzer[†], F. Aurenhammer[‡], T. Hackl[§]
B. Jüttler[¶], M. Oberneder[||], Z. Šír[**]

Abstract

Boundary approximation of planar shapes by circular arcs has quantitive and qualitative advantages compared to using straight-line segments. We demonstrate this by way of three basic and frequent computations on shapes – convex hull, decomposition, and medial axis. In particular, we propose a novel medial axis algorithm that beats existing methods in simplicity and practicality, and at the same time guarantees convergence to the medial axis of the original shape.

1 INTRODUCTION

The plain majority of algorithms in computational geometry has been designed for processing *linear* objects, like lines, planes, or polygons. On the one hand, this is certainly due to the fact that many interesting and deep computational and combinatorial questions do arise already for inputs of this simple form. Again, the pragmatic reason is that algorithms for linear objects are usually both easier to develop and simpler to implement. To make things work for nonlinear objects, which arise frequently in practical settings, such objects then have to be approximated in a piecewise-linear manner and up to a tolerable error.

In its simplest form, the input is a single planar shape, $\mathcal{A}$, with curved and connected boundary $\partial\mathcal{A}$. Frequent tasks to be performed on $\mathcal{A}$ – each being prior to a variety of more involved computations – are constructing the convex hull of $\mathcal{A}$, decomposing $\mathcal{A}$ into primitives, and calculating the medial axis of $\mathcal{A}$. These tasks are well investigated in the case of polygonal shapes. In certain situations, however, the number of line segments required for approximating $\partial\mathcal{A}$ with high accuracy may be prohibitively large. Even more seriously, making a piecewise-linear approximation of $\partial\mathcal{A}$ and invoking a polygonal-shape algorithm may generate results that are topologically incorrect; the medial axis is a well-known example.

The intention of the present paper is to highlight the use of circular arcs for boundary representation. It is well known that for nonlinear curve segments the approximation order increases in comparison to using

*Research supported by the FWF Joint Research Project Industrial Geometry S9205-N12
[†]Institute for Software Technology, University of Technology, Graz, Austria, oaich@ist.tugraz.at
[‡]Institute for Theoretical Computer Science, University of Technology, Graz, Austria, auren@igi.tugraz.at
[§]Institute for Software Technology, University of Technology, Graz, Austria, thackl@ist.tugraz.at
[¶]Institute for Applied Geometry, Johannes Kepler University of Linz, Austria, bert.juettler@jku.at
[||]Institute for Applied Geometry, Johannes Kepler University of Linz, Austria, margot.oberneder@jku.at
[**]Institute for Applied Geometry, Johannes Kepler University of Linz, Austria, zbynek.sir@jku.at

straight-line segments. In particular, if a given accuracy ε is achieved by using N line segments, then as few as $n = \Theta(N^{2/3})$ circular arcs can accomplish the same. This has been an issue in approximation theory, but in computational geometry this gain seems to have been less valued than eliminating small factors in the complexity of the subsequently applied algorithm. Boundary approximation by circular arcs may be of advantage also in a qualitative respect. For instance, it avoids the mentioned topological inconsistencies in medial axis computations, and it supports the computation of shape offsets, as the class of shapes bounded by circular arcs is closed under offset operations.

We will show that for the three basic problems mentioned above – convex hull, triangulation, and medial axis – simple and practical, though still efficient algorithms exist that work for circular arc inputs. The first two problems are less demanding; we treat them mainly to point out the respective favorable (in our opinion) approach, whose practicality shall encourage the use of circular arc boundary representation. Nevertheless, substantial differences to the polygonal case occur; see below. For computing the medial axis, we propose a novel and extremely simple algorithm that is based on a known (though less recognized) decomposition lemma. After having computed a purely combinatorial description of the medial axis using tailored shape splitting, its individual parts (conics and line segments, like in the polygonal case) are re-assembled *without* the need of merging.

Suitable circular arc approximations of shapes can be found in linear time. In summary, the obtained shape processing algorithms are superior in runtime to their line segment based counterparts, retain much (if not all) of their simplicity, and are even more natural in some cases.

1.1 Outline and background

We briefly describe the contributions of this paper and relate them to existing literature.

Section 2 deals with approximating general curves by suitable primitives. This is a topic of importance in geometric modeling and in CAD and NC applications, and many quite recent results are available [23, 24, 26, 32, 16, 30]. Our aim is to approximate a parametric curve $\boldsymbol{c}(t)$ by circular arcs. We assume that $\boldsymbol{c}(t)$ is piecewise-polynomial of constant degree, and we use biarcs [29, 24, 30] as primitives. A straight-forward bisection algorithm for biarc generation already fits our purposes. It uniquely assigns biarcs to parameter intervals, which facilitates the error evaluation. An approximating spline curve $\boldsymbol{b}$ of size n is computed in $O(n)$ time. It fits the input curve $\boldsymbol{c}(t)$ in slope at biarc endpoints, and can be tuned to match $\boldsymbol{c}(t)$ in curvature at certain points (a fact being important in subsequent medial axis computations). Though not being optimal in the number of arcs, the approximation order of $\boldsymbol{b}$ is still three [23, 30]. In contrast, with line segments one cannot exceed order two, and a polyline of size $N = \Theta(n^{3/2})$ is needed to arrive at the same precision.

The remaining sections propose algorithms for *circular arc shapes* $\mathcal{A}$, where the boundary $\partial\mathcal{A}$ of $\mathcal{A}$ is given as a connected curve composed of n circular arcs. Choice is guided by efficiency as well as by reducibility to basic operations that have robust implementations [10].

Section 3 outlines an algorithm for computing the convex hull of $\mathcal{A}$. This task is one of the most basic to be performed for a given shape, and has a variety of applications including shape fitting, motion planning, shape separation, and many others. At least four linear-time algorithms have been developed for polygonal shapes [4, 15, 22, 25]. The incremental method by Melkman [25] stands out by its simplicity, and it is this candidate we generalize for circular arc shapes. Compared to the original setting, two difficulties arise. Deciding inclusion for a currently inserted arc in the convex hull constructed so far is no trivial test, and the convex hull cannot be described by a sequence of input vertices of the shape. We show that a runtime of $O(n)$ is still possible. The basic subroutine of the algorithm computes the convex hull of only two circular arcs.

Section 4 deals with shape triangulation, a fundamental building block in algorithms for decomposition, shortest path finding, and visibility – to name a few. Most existing algorithms are meant for polygonal shapes. They partition a given (simple) n-vertex polygon into triangles without introducing Steiner points. Efficient candidates are [13, 21, 3, 17, 7] which all show an $O(n \log n)$ runtime. Theoretically more efficient methods do exist, but when aiming at simplicity, choice should be made from the list above.

When trying to generalize to shapes $\mathcal{A}$ bounded by circular arcs, we face two problems. First of all, if the use of Steiner points is disallowed, then a partition of $\mathcal{A}$ into primitives bounded by constantly many circular arcs need not exist. Also, not all triangulation methods are suited to generalization. This applies, for instance, to the extremely simple ear cutting method in [19] which runs in time $O(r \cdot n)$, where r is the number of reflex vertices of $\mathcal{A}$. The triangulation algorithm we propose is closest to Chazelle's [7]. It manages with an (almost) worst-case minimal number of Steiner points on $\partial\mathcal{A}$, runs in $O(n \log n)$ time, and uses a dictionary as its only nontrivial data structure. The produced primitives are arc triangles with at least one straight edge. The most complex geometric operation is intersecting a circle with a line.

Section 5 is devoted to the medial axis, a frequently used structure associated with a given input shape. Its main applications include shape recognition, solid modeling, pocket machining, and others. Interest in mathematical properties of the medial axis for general shapes found renewal in recent years [9, 27, 5, 6, 2]. In our case, where the shape $\mathcal{A}$ is simply connected and $\partial\mathcal{A}$ consists of n circular arcs, its medial axis $M(\mathcal{A})$ is known to be a tree composed of $O(n)$ conic edges. Algorithmic work on the medial axis either concentrated on the case where $\mathcal{A}$ is a polygon [20, 7, 8], or on general sets of curved arcs [32, 18, 27, 1] (and their Voronoi diagram) without, however, exploiting the fact that the input arcs define a simple curve. (The various existing methods for computing digital versions of the medial axis are not considered here.) Though theoretically efficient as $O(n \log n)$ or better, these algorithms suffer from involved merge or insertion steps which, even for straight arcs as input, are difficult to implement. In addition, numerical stability issues arise heavily; intersections of conics have to be determined repeatedly which, when not calculated exactly, are bound to cumulate the error.

We present a simple randomized divide-and-conquer algorithm for computing $M(\mathcal{A})$ that overcomes these drawbacks. In contrast to comparable algorithms, the costly part is delegated to the divide step. The basic subroutine there is an inclusion test for an arc in a circle. The merge step is trivial: it concatenates two medial axes. The expected runtime is bounded by $O(n^{3/2})$, but is provably better for most types of shape. For example, $O(n \log n)$ expected time suffices if the diameter of $M(\mathcal{A})$ is $\Theta(n)$. No nontrivial data structures are used.

To guarantee applicabiliy of our methods to approximating the medial axes of general shapes $\mathcal{B}$, a convergence result is needed. We prove in Section 6 that, for a suitable approximation of $\partial\mathcal{B}$ by biarcs, $M(\mathcal{B})$ is the limit of $M(\mathcal{A})$ when the approximating arc shape $\mathcal{A}$ converges to $\mathcal{B}$. Related results exist, but either presuppose C^2 conditions on $\partial\mathcal{A}$ not attainable by circular arcs [6], or concern subsets of the medial axis [5] that survive after pruning the Voronoi diagram of point samples from $\partial\mathcal{B}$. (As a negative side effect, the medial axis approximation obtained from a point sample is not C^1.) It is well known [2] that medial axis convergence is *not* given for polygonal approximations of $\mathcal{B}$. In conclusion, circular arcs are the simplest possible tool for boundary conversion that guarantees a stable medial axis approximation.

2 CIRCULAR ARCS

In order to represent a general shape $\mathcal{A}$ in a form suitable for geometric computations, we discuss methods for approximating $\partial\mathcal{A}$ by circular arcs. We assume that $\partial\mathcal{A}$ is given as a polynomial spline curve of constant degree. Attention is restricted to degree 3, as every free-form curve can efficiently and with any desired precision be converted into cubics [28], and in many applications the input will already be available in this common form [11].

Several approaches to generating circular arc splines exist; see [23] for a review. We consider a simple bisection algorithm consisting of two steps, approximation and error measurement. A geometric primitive b (an arc or a biarc) is fitted to a segment s of the given cubic curve $\boldsymbol{c}(t)$, and the distance from b to s is computed. The algorithm is relatively easy to implement and still adapts the degrees of freedom to the input data. As a slight disadvantage, the number of primitives (the resulting data volume) is minimal only in the asymptotic sense.

Define the one-sided Hausdorff distance from a primitive b to a segment $s \subseteq \boldsymbol{c}(t)$ as

$$\delta(b, s) = \max_{p \in b} \min_{q \in s} ||p - q||.$$

(We consider b and s as closed sets.) Let ε denote the error tolerance to be met by the algorithm.

Algorithm BISECT(t_0, t_1)

Construct b
Compute $\delta = \delta(b, \boldsymbol{c}[t_0, t_1])$
If $\delta \leq \varepsilon$ then return $\{b\}$
Else return BISECT$(t_0, \frac{t_0+t_1}{2}) \cup$ BISECT$(\frac{t_0+t_1}{2}, t_1)$

Depending on the primitive b used, Algorithm BISECT produces splines of different quality: merely continuous (C^0) circular arc splines, or tangent continuous (C^1) arc splines. When being content with the former type, we can simply choose for b the unique circular arc passing through the three points $\boldsymbol{c}(t_0)$, $\boldsymbol{c}(\frac{(t_0+t_1)}{2})$, and $\boldsymbol{c}(t_1)$. To obtain C^1 arc splines, so-called biarcs [29] are utilized.

A *biarc* b consists of two circular arcs with common unit tangent vector at their joint. Usually, b is described by its source x with associated unit tangent vector v_x, and its target y with unit tangent vector v_y. Given these data, there exists a one-parameter family of interpolating biarcs. All possible joints are located on the circle σ passing through x and y and having the same oriented angles with v_x and v_y. Several ways for choosing the joint m have been proposed; see e.g. [24, 30]. For many applications, taking $m = \sigma \cap \boldsymbol{c}[t_0, t_1]$ is appropriate. To calculate m, a polynomial of degree 4 has to be solved (where a closed-form solution is still available). The output is a C^1 arc spline with all arc endpoints sitting on $\boldsymbol{c}(t)$.

In view of subsequent stable medial axis computations, the choice of m has to be made more carefully. Define an *apex* of $\boldsymbol{c}(t)$ as a local curvature maximum. The apices split the curve $\boldsymbol{c}(t)$ into pieces of monotonic signed curvature, so-called *spirals*. Following [24], we aim at approximating spirals of $\boldsymbol{c}(t)$ by circular arc spirals. To this end, we split $\boldsymbol{c}(t)$ at its apices. These points can be found by solving polynomials of degree 5. Now, we exploit that spiral biarcs can be constructed that connect two given points x and y, match unit tangents there, and assume a predefined curvature in one of them. Let k_x and k_y be the curvature of $\boldsymbol{c}(t)$ at x and y, respectively, and suppose $k_x < k_y$. To match curvature at x, we choose the radius of the first arc, b_1, equal to $r_x = 1/k_x$. The joint m is obtained by intersecting the circle supporting b_1 with the joint circle σ. According to [24], the radii and curvatures satisfy $r_x > r_y > 1/k_y$. When starting the next biarc from y with $r_y = 1/k_y$ (unless y is an apex), monotonicity of signed curvature will be preserved.

Each arc is found in $O(1)$ time, where the constant depends on the degree of the polynomial to be solved. Figure 1 shows an example of a biarc conversion. The scaled curve normals visualize the (magnified) error distribution.

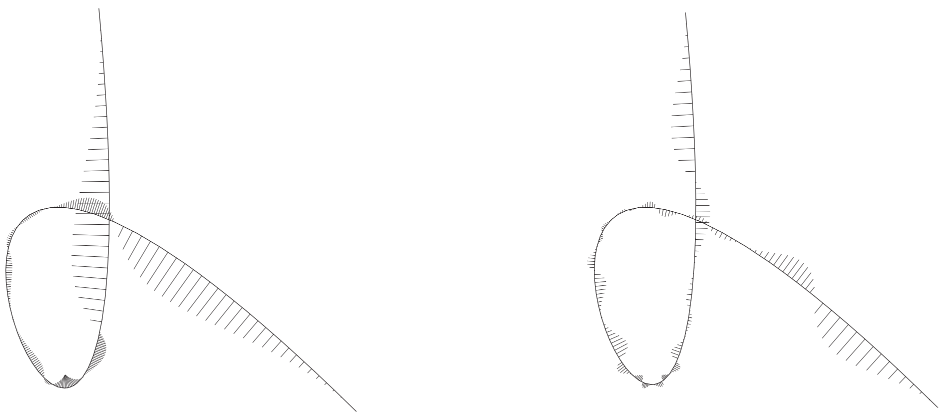

Figure 1: z-magnified error for n biarcs

Concerning the error measurement, each produced circular arc b_i has to be matched to its corresponding segment $s = \boldsymbol{c}[t'_0, t'_1]$. This is, of course, trivial when the biarc joint m has been chosen to lie on $\boldsymbol{c}(t)$. In the case of biarc spirals, we intersect $\boldsymbol{c}(t)$ with the normal of b_i at m. This leads to a cubic equation. If multiple solutions within the total biarc interval $[t_0, t_1]$ exist, then the error is set to ∞. Otherwise, we compute the one-sided Hausdorff distance $\delta(b_i, s)$ by substituting the parametric representation of s into the implicit equation K (with leading coefficients 1) of the circle supporting b_i. If r is the radius of K, and d and D are the minimum and maximum values of $(K \circ \boldsymbol{c})(t)$ for $t \in [t'_0, t'_1]$, we get

$$\delta(b_i, s) \leq \max\{|\sqrt{r^2 - d} - r|, |\sqrt{r^2 + D} - r|\}$$

and this bound is sharp. Consequently, $\delta(b_i, s)$ can be evaluated by solving a quintic polynomial equation on the interval $[t'_0, t'_1]$. Alternatively, a simpler upper bound can be calculated (without polynomial solving) by replacing d and D with the minimum and maximum coefficient of the Bernstein-Bézier representation [12] of $(K \circ \boldsymbol{c})$ with respect to $[t'_0, t'_1]$. As the length of s decreases, this bound converges to $\delta(b_i, s)$. As another simple but important observation, the *two-sided* Hausdorff distance between b_i and s, $\max\{\delta(b_i, s), \delta(s, b_i)\}$, vanishes with $\delta(b_i, s)$ because b_i and s are of constant degree. Thus controlling the latter distance already ensures that b_i and s are ε-close.

In summary, when algorithm BISECT spans a binary recursion tree with n leaves (the returned n primitives), any of the described types of arc splines can be constructed in $O(n)$ time.

Let us discuss the asymptotic behaviour of the number n for decreasing tolerance ε. For a given curve $\boldsymbol{c}(t)$ with domain $[t_0, t_1]$, which is assumed to contain neither inflections nor apices, we consider primitives having approximation order k. Adapting the analysis in [23, 30] (as done in the Appendix), we get $\delta = \Theta(h^k)$ for the one-sided Hausdorff distance δ, provided that $\boldsymbol{c}(t)$ is approximated with (small) parameter step size h, and that k is considered a constant.

This relation implies a general lower bound. For *any* approximation of $\boldsymbol{c}(t)$ by n primitives of order k, the largest step size satisfies $\Delta t \geq \frac{t_1 - t_0}{n}$. Thus from $\varepsilon \geq \delta$, which is to be achieved by the approximation, and from $\delta = \Theta((\Delta t)^k)$, we get $n = \Omega(1/\varepsilon^{1/k})$. On the other hand, the smallest step size Δt taken by algorithm BISECT satisfies $\Delta t \leq \frac{t_1 - t_0}{n}$. When doubling the step size we have $\delta = \Theta((2\Delta t)^k)$ but $\varepsilon < \delta$, as the tolerance is not yet achieved. Thus $n = O(1/\varepsilon^{1/k})$. We obtain:

Lemma 1 *For sufficiently small tolerance ε, the number n of primitives constructed by algorithm BISECT is asymptotically optimal.*

Lemma 1 also holds in the general case where $\boldsymbol{c}(t)$ contains inflections and apices, because the resulting number of spirals of $\boldsymbol{c}(t)$ is independent of n. In conclusion, to arrive at tolerance ε, Algorithm BISECT needs $n = \Theta(1/\sqrt[3]{\varepsilon})$ cicular arcs (order 3), whereas $N = \Theta(1/\sqrt{\varepsilon})$ line segments (order 2) have to be invested by any polygonal approximation method.

Corollary 1 *Compared to approximating $\boldsymbol{c}(t)$ with a polyline, the data volume drops from N to $\Theta(N^{2/3})$ when circular arc splines are used.*

It should be observed that, the other way round, when approximating $\boldsymbol{c}(t)$ with a *point sample*, the data volume increases to $\Theta(n^3)$ compared to n circular arcs.

3 CONVEX HULL

Let $\mathcal{A}$ be some shape given in boundary representation. More specifically, $\partial\mathcal{A}$ is approximated by a simple (i.e., not self-crossing) and connected curve $\boldsymbol{b}$ composed of n circular arcs. Clearly, if $\boldsymbol{b}$ converges to $\partial\mathcal{A}$ then the convex hull of $\boldsymbol{b}$ converges to the convex hull of $\mathcal{A}$. We show that the convex hull algorithm for polylines in Melkman [25] can be generalized to simple circular arc curves $\boldsymbol{b}$ while retaining its $O(n)$ runtime.

In a nutshell, this algorithm processes each of the vertices of the given polyline in order and maintains their convex hull. If the currently processed vertex v_i falls into the convex hull, CH_{i-1}, constructed so far then v_i is deleted and we put $CH_i = CH_{i-1}$. Otherwise, tangents are placed from v_i to CH_{i-1}, and the sequence of vertices (if any) between the corresponding two vertices of tangency is deleted from CH_{i-1} in order to construct CH_i.

The linear runtime of this strategy hinges on two propositions: (1) A constant-time inclusion test $v_i \in CH_{i-1}$, and (2) deletion of vertices of CH_{i-1} which are non-extreme in CH_i in time proportional to their number. While (2) is achieved by a standard Graham scan [14], proposition (1) is met by exploiting simplicity of the given polyline: $v_i \in CH_{i-1}$ is equivalent to $v_i \in \alpha(v)$, where $\alpha(v)$ is the interior angle at the last vertex, v, added to CH_{i-1}.

Staying with vertices works correctly with polygonal curves because the convex hull of two points equals the convex hull of their connecting line segment. This is, of course, not true for a connecting circular arc. As a consequence, the set of vertices of the convex hull to be constructed is, in general, no subset of the input vertices. Also, the inclusion test for a circular arc to be inserted is a more complicated operation, The following variant of Melkman's algorithm is able to cope with circular (and more general) arcs and still runs in $O(n)$ time. Its main subroutine computes the convex hull of only two arcs.

Let $b_1 \dots b_n$ be the given simple circular arc curve. Some of the arcs may be line segments, and the curve may be cyclic. Assume first that the curve is C^1. Let CH denote the convex hull operator, and abbreviate $CH(b_1 \dots b_i)$ as CH_i. Consult Figure 2.

Algorithm HULL

Construct $CH_2 = CH(b_1 b_2)$. Let v be the last point along
the chain $b_1 b_2$ that lies on CH_2.

For $i = 3, \dots, n$, process the arc b_i as follows:

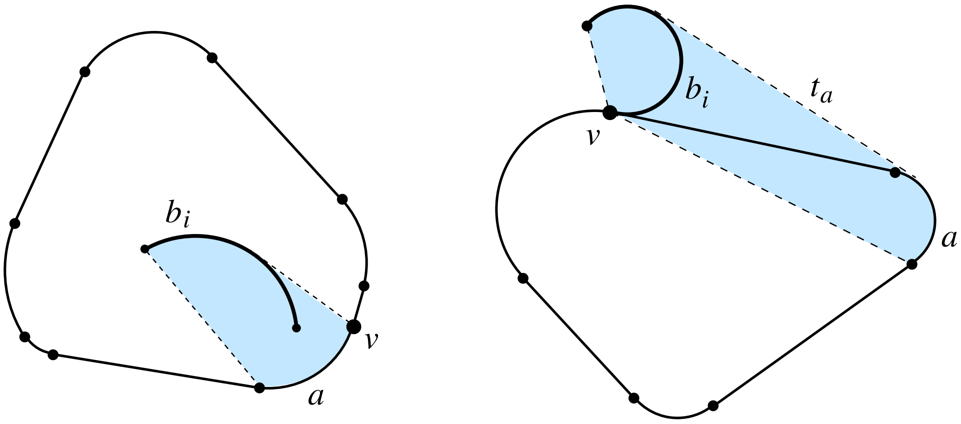

Figure 2: Cases 2.1 (left) and 2.4 (right)

Search for the first arc, a, of CH_{i-1} clockwise from v that contributes with non-zero length to $CH(a, b_i)$ and has this hull and CH_{i-1} on the same side. Similarly, search for the first arc, c, counter-clockwise from v with analogous properties. ($a = c$ is possible.) Arcs a and c already provide the information needed to construct CH_i correctly.

Case 1 Arc a (and equivalently, arc c) does not exist. This means $CH_{i-1} \subset CH(b_i)$. Put $CH_i = CH(b_i)$, and assign to v the target of b_i.

Case 2 Arcs a and c do exist. Check for some tangent, t_a, which appears on $CH(a, b_i)$ and is clockwise tangent to CH_{i-1}. Also, check for some tangent, t_c, which appears on $CH(c, b_i)$ and is counter-clockwise tangent to CH_{i-1}.

Case 2.1 Tangents t_a and t_c both do not exist. This means $b_i \in CH_{i-1}$. Put $CH_i = CH_{i-1}$.

Case 2.2 t_a exists (uniquely) but t_c does not. Let $t_a = x_a y_a$, where x_a is its point of tangency on CH_{i-1}. To obtain CH_i, delete from CH_{i-1} the clockwise part between v and x_a, and add t_a and the piece of the arc b_i between y_a and v. Update v as the last point along b_i on CH_i (either y_a or b_i's target).

Case 2.3 t_c exists (uniquely) but t_a does not. Let $t_c = x_c y_c$, with x_c being its point of tangency on CH_{i-1}. To get CH_i, delete from CH_{i-1} the counter-clockwise part between v and x_c, and add t_c and the piece of the arc b_i between y_c and v. Update v as in Case 2.2 (either y_c or b_i's target).

Case 2.4 t_a and t_c both do exist. Here we get CH_i by deleting arcs from CH_{i-1} as in Cases 2.2 and 2.3, and then adding t_a, t_c, and the piece of b_i between y_a and y_c. We update v as the point among y_a and y_c that is closer to the target of b_i.

Correctness of algorithm HULL is verified by observing that t_a and t_c are indeed tangents from the currently inserted arc b_i to the convex hull CH_{i-1} constructed so far. Thereby, as the algorithm stands now, it is of importance that the input curve is C^1. This guarantees that the boundary of CH_{i-1} is C^1 as well (except possibly at the target of b_{i-1}), such that the arcs a and c are found correctly. Minor modifications in the selection criteria for these arcs will make the algorithm work without this restriction.

The runtime is dominated by the search for a and c, where the necessary number of calls of the two-arc hull subroutine is proportional to the total number of arcs constructed or deleted. This number is $O(n)$ because only $O(1)$ arcs are constructed per i-loop. The rest can be accomplished in $O(1)$ time per arc b_i if CH_i is stored as a doubly linked list, or in $O(n)$ total time if CH_i is represented in a (more space-saving) dequeue.

4 TRIANGULATION

We next propose a triangulation algorithm for circular arc shapes. Define an *arc triangle* as a (simply connected) face bounded by at most three circular arcs or line segments.

As a first observation, a partition of a circular arc shape $\mathcal{A}$ into arc triangles does not always exist, unless the use of Steiner points is allowed. This is even true when the n arcs describing ∂A are given as x-monotone pieces (and hence span semi-circles at most), which we will assume below. In fact, there are examples where at least $2n - 7$ Steiner points are necessary. See Figure 3. For no pair of vertices of the depicted shape $\mathcal{A}$ does there exist a connecting circular arc inside $\mathcal{A}$. Thus no part of $\mathcal{A}$ can be split off using a circular arc between two vertices. The interested reader may convince himself that placing $n - 4$ Steiner points as shown is no waste. The asserted lower bound then follows, because each of the resulting faces needs additional Steiner points. Note that a single point per face suffices only if circular arcs rather than line segments are used to split the face.

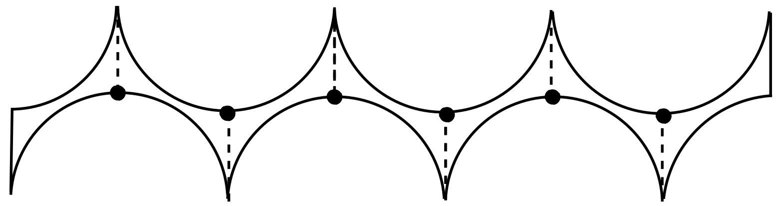

Figure 3: Many Steiner points

The triangulation algorithm we are going to describe introduces at most $2n - 5$ Steiner points (on the boundary of $\mathcal{A}$, rather than in its interior), runs in $O(n \log n)$ time, and uses a dictionary as its most involved data structure. The produced primitives are arc triangles where at least one edge is a line segment. Standard plane sweep is used to compute the vertical visibilities inside $\mathcal{A}$ for each pair (vertex, arc) of ∂A. Each such pair defines a vertical line segment that splits $\mathcal{A}$ and ends at a Steiner point on ∂A. A decomposition of $\mathcal{A}$ into arc triangles and arc trapezoids results. No priority queue is needed, as all events guiding the plane sweep (namely, the vertices of ∂A) are known in advance and thus can be x-sorted beforehand. For simplicity, suppose that their x-coordinates are pairwise different.

Lemma 2 *The decomposition above contains exactly $n - 2$ Steiner points.*

Proof. Let us call a vertex *type k* if it vertically sees k arcs, i.e., defines k Steiner points. We have vertices of types $0, 1$, and 2. At each type-2 vertex v, the shape $\mathcal{A}$ is vertically split into three parts, each part having a type-0 vertex as an x-extremum. Two such parts lie on the same side of the splitting segment, and among their extreme type-0 vertices, we map v to the one which is x-closer to v. This mapping is injective, and does not address the two x-extrema of ∂A. The lemma follows. □

The obtained faces are exactly $n - 1$ in number, at least two being arc triangles. Each face F that is an arc trapezoid can be easily split into arc triangles. If F is convex then a line segment will do. Also, if at least one of the two arcs on ∂F is avoided by the central line g of their supporting circles, then a single splitting arc or line segment for F exists (because there is a normal to g that touches that arc at an endpoint). Otherwise, we use an intersection of g with a reflex arc on ∂F as a Steiner point and split F with two arcs. Figure 4 illustrates two typical cases. In total, at most $2n - 5$ Steiner points are used for an arc triangulation.

We stress the fact that generalizing the classical plane sweep for polygon triangulation [17] – though well possible in $O(n \log n)$ time – results in a more complicated algorithm. Large parts already swept across have to be remembered for later processing, and the produced primitives are more complex than arc trapezoids. Also, line segments being simultaneously tangent to two given circles have to be calculated, whereas in our algorithm the most complex operation is intersecting a circle with a straight line.

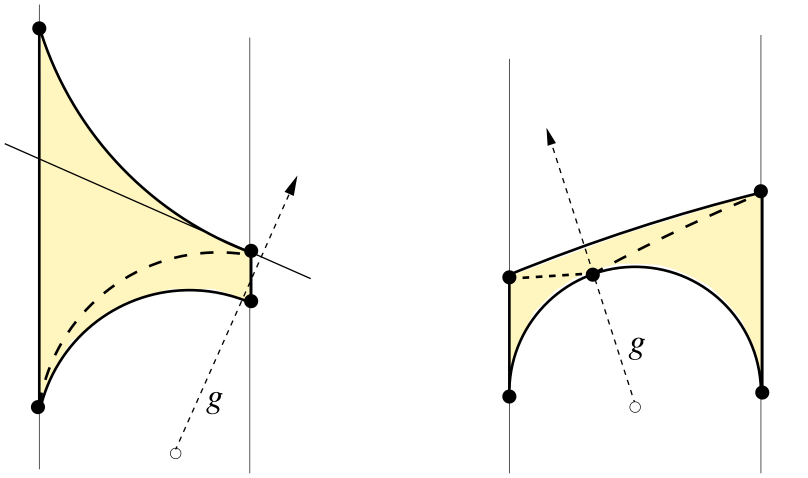

Figure 4: Splitting arc trapezoids

5 MEDIAL AXIS

Let $\mathcal{A}$ be the circular arc shape under consideration. (All objects are considered to be closed sets in the sequel). Call a disk $D \subseteq \mathcal{A}$ *maximal* if there exists no disk D' different from D such that $D' \supset D$ and $D' \subseteq \mathcal{A}$. The medial axis, $M(\mathcal{A})$, of $\mathcal{A}$ is defined as the set of all centers of maximal disks.

As the boundary of $\mathcal{A}$ is a connected and simple curve with n circular arcs, $M(\mathcal{A})$ is finite, connected and cycle-free [9] and thus forms a tree. $M(\mathcal{A})$ can be decomposed into $O(n)$ *edges*, which are maximal pieces of straight lines and (possibly all four types of) conics. Endpoints of edges will be called *vertices* of $M(\mathcal{A})$.

The contribution of this section is a simple and practical randomized algorithm for computing $M(\mathcal{A})$. It works by divide-and-conquer and accepts as input any description of $\partial\mathcal{A}$ by circular arcs and/or line segments. The costly part is delegated to the divide step, which basically consists of inclusion tests for arcs in circles. The merge step is trivial; it just concatenates two partial medial axes. The expected runtime is bounded by $O(n^{3/2})$, and will be proved to be $O(n \text{ polylog } n)$ for several types of shape. A qualitative difference to existing medial axis algorithms is that a *combinatorial* description of $M(\mathcal{A})$ is extracted first, which can then be directly (and robustly) converted into a geometric representation. We base our algorithm on the following simple though elegant decomposition lemma [9].

Lemma 3 *Consider any maximal disk D for $\mathcal{A}$. Let $A_1, \ldots, A_t$ be the connected components of $\mathcal{A} \setminus D$, and denote with p the center of D.*

$$\text{(1)} \quad M(\mathcal{A}) = \bigcup_{i=1}^{t} M(A_i \cup D)$$

$$\text{(2)} \quad \{p\} = \bigcap_{i=1}^{t} M(A_i \cup D)$$

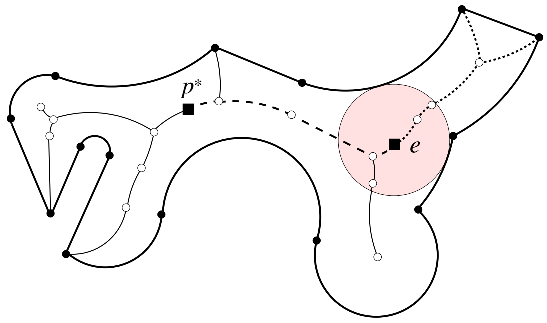

Figure 5: Walk (dashed) and cut (dotted)

In plain words, having at hands some maximal disk one can compute the medial axes for the resulting components recursively, and then glue them together at a single point. However, the desired efficiency of this strategy calls for a balanced decomposition. Its existence is given below.

Lemma 4 *There exists a maximal disk D for $\mathcal{A}$ such that at most $\frac{n}{2}$ arcs from $\partial\mathcal{A}$ are (completely) contained in each component of $\mathcal{A} \setminus D$.*

Proof. Each point $p \in M(\mathcal{A})$ corresponds to a unique maximal disk D_p for $\mathcal{A}$. Let $f(D_p)$ be the number of arcs from $\partial\mathcal{A}$ in the largest component induced by D_p. As long as $f(D_p) > \frac{n}{2}$, the component that realizes $f(D_p)$ is unique, and we can decrease $f(D_p)$ by continuously moving p on $M(\mathcal{A})$ such that D_p enters into this component. This process terminates at some point p^* where $f(D_{p^*}) \leq \frac{n}{2}$. We never move back the way we came, as the component we move out never exceeds a size of $\frac{n}{2}$. $\square$

We are left with the algorithmic problem of finding some maximal disk that yields a well-balanced partition. Observe that the optimal point p^* above may be not unique, because the number $f(D_p)$ is invariant under motion of p within the relative interior of any fixed edge $e \subset M(\mathcal{A})$. Let us define $Walk(e)$ as the path length in $M(\mathcal{A})$ from e to p^*. Further, define $Cut(e)$ as the size of the smaller one among the two subtrees which constitute $M(\mathcal{A}) \setminus \{e\}$. See Figure 5. Any tree with small 'cuts' tends to have short 'walks', in the following respect.

Lemma 5 *Let e be an edge of $M(\mathcal{A})$, chosen uniformly at random. Then $E[Walk(e)] = \Theta(E[Cut(e)])$.*

Proof. Orient all the paths in $M(\mathcal{A})$ away from the point p^*. This defines a partial order $\prec$ on the edges of $M(\mathcal{A})$. We have the set equality

$$\bigcup_{e \in M(\mathcal{A})} \{(a, e) \mid a \prec e\} = \bigcup_{e \in M(\mathcal{A})} \{(e, b) \mid b \succ e\}$$

because either set contains each pair of the relation exactly once. The (disjoint) subsets united in the left set, L, represent all the paths in $M(\mathcal{A})$ between its edges e and p^*. Thus we have $E[Walk(e)] = \frac{1}{m} \cdot |L|$, where m is the number of edges of $M(\mathcal{A})$. The (disjoint) subsets united in the right set, R, represent those subtrees defined by the edges e of $M(\mathcal{A})$ which avoid p^*. If we neglect subtrees of sizes larger than $\frac{m}{2}$, then the cardinality of the set drops by a constant factor (of at most 4, if $\prec$ would be a total order, hence less). This implies $\frac{1}{m} \cdot |R| > E[Cut(e)] > \frac{1}{m} \cdot \frac{|R|}{4}$. The lemma now follows from $|R| = |L|$. $\square$

Lemma 5 motivates the following disk finding algorithm which combines random cutting with local walking. Its main subroutine, MAX(b), selects for an arc $b \subset \partial\mathcal{A}$ its midpoint x and returns the unique maximal disk for $\mathcal{A}$ with x on its boundary. For the ease of description, we assume that this disk splits $\mathcal{A}$ into exactly two components. Let $c \geq 3$ be a (small) integer constant.

Procedure CUT$(\mathcal{A})$

Put $A' = \mathcal{A}$
Repeat
  Choose a random arc b of $\partial A'$
  Compute D=MAX(b) and let $\mathcal{A}_0$ be the larger
    component of $\mathcal{A}$ induced by D
  Assign $A' = A' \cap \mathcal{A}_0$
Until $\mathcal{A}_0$ contains less than $n - \frac{n}{c}$ arcs
Report D

Procedure WALK$(\mathcal{A})$

Choose a random arc b of $\partial\mathcal{A}$
Compute D=MAX(b)
Let $\mathcal{A}_0$ be the larger component induced by D
While $\mathcal{A}_0$ contains more than $n - \frac{n}{c}$ arcs do
  Let b_1 (b_2) be the first (last) complete arc of $\partial\mathcal{A}$ in $\mathcal{A}_0$
  Compute D_1=MAX(b_1) and D_2=MAX(b_2)
  Assign to $\mathcal{A}_0$ the smaller one of the respective larger
    components of $\mathcal{A}$ for D_1 and D_2
  Memorize the corresponding disk $D \in \{D_1, D_2\}$
Report D

The disk finding algorithm now runs CUT$(\mathcal{A})$ and WALK$(\mathcal{A})$ in parallel and terminates as soon as the first disk is reported. To analyze its runtime, let us first consider the assignment of arcs on $\partial\mathcal{A}$ to edges of $M(\mathcal{A})$, as done in subroutine MAX. Namely, if MAX(b)=D then arc b is mapped to the edge e that contains the center of D. Observe that either $0, 1$, or 2 arcs are mapped to a fixed edge. Moreover, no two unaddressed edges and no two doubly addressed edges are neighbored. This assignment is sufficiently uniform to convey randomness from arcs to edges. Let now $E[Cut(e)] = \Theta(k)$. The expected number of loop executions in procedure CUT$(\mathcal{A})$ is $O(\frac{n}{k})$, because each time the expected reduction in size of A', and thus of $\mathcal{A}_0$, is $\Theta(k)$. On the other hand, by Lemma 5, the expected number of loop executions in procedure WALK$(\mathcal{A})$ is $O(k)$, because each time a progress of one edge is made on the respective path in $M(\mathcal{A})$. In the worst case of walk length being balanced with cut number, a bound of $O(\sqrt{n})$ on the expected number of loop executions in at least one of CUT$(\mathcal{A})$ and WALK$(\mathcal{A})$ holds.

The costly part in both procedures is their subroutine MAX, whose expected number of calls obeys the same bound. D=MAX(b) has a simple implementation which runs in $O(n)$ time: We initialize the disk D as the (appropriately oriented) halfplane that supports b at its midpoint x and, for all remaining arcs $b_i \subset \partial\mathcal{A}$ that intersect D, we shrink D so as to touch b_i while still being tangent to b at x. The most complex operation for shrinking D is an inclusion test for a point in a circle. In particular, and unlike previous medial axis algorithms, no conics take part in geometric operations.

In summary, the randomized complexity for computing the medial axis is given by $T(n) = T(\frac{1}{c}n) + T((1-\frac{1}{c})n) + O(n^{3/2}) = O(n^{3/2})$. In many cases, however, will the algorithm perform substantially better. Let d be the graph diameter of $M(\mathcal{A})$. Then the loop in WALK$(\mathcal{A})$ is executed less than d times.

So, for example, if $d = \Theta(\log n)$ then an overall runtime of $O(n \log^2 n)$ is met. For the other extreme case, $d = \Theta(n)$, our strategy is even faster. With constant probability, an edge on the diameter is chosen, and $\Theta(n)$ such edges e have $Cut(e) = \Theta(n)$. The expected number of loop executions in CUT($\mathcal{A}$) now is only $O(1)$, and an $O(n \log n)$ algorithm results. We conjecture that the latter situation is quite relevant in practice. For most shapes, their medial axes will not branch extensively, and even if so, the branching will be independent of n, because each branch will be approximated by a large number of circular arcs in order to achieve the predefined precision.

The output of the algorithm is a list of $O(n)$ points on $M(\mathcal{A})$, namely, the centers of the splitting disks, plus a list of $O(n)$ edges connecting them. Each edge is given implicitly by its defining two arcs on $\partial\mathcal{A}$. To make sure that the reported point list includes all the vertices of $M(\mathcal{A})$, base cases that involve constantly many (pieces of) original arcs from $\partial\mathcal{A}$ have to be solved directly. (The constant equals 2 or 3 if $\partial\mathcal{A}$ is C^1.) Note that the algorithm works exclusively on $\partial\mathcal{A}$ except for a final step, where the conic edges of $M(\mathcal{A})$ are explicitly calculated and reassembled. This gives rise to increased numeric stability in comparison to existing approaches.

Opposed to approximating $\partial\mathcal{A}$ with the same accurracy by a polyline of size N, our circular arc algorithm takes $O(n^{3/2}) = O(N)$ time; see Corollary 1 in Section 2. Thus, even for (probably rare) worst-case inputs, our simple algorithm competes asymptotically well with previous methods. Another advantage over polygonal approximation is described in Section 6.

6 CONVERGENCE

A well-known unpleasant phenomenon of the medial axis is its instability under perturbations of the shape boundary. Several papers discussing this issue have been published recently. A result in [6] shows that stability is, in general, not given unless perturbations are C^2. To deal with general shapes, the so-called λ-medial axis has been introduced as a tool in [5]. After drawing a point sample from the shape boundary, the Voronoi diagram of these points is constructed and pruned appropriately. The λ-medial axis converges to the original for vanishing sample distance. Drawbacks are the large sample size for a close (and homotopy-equivalent) approximation, the lack of its C^1 behavior, and the need of computing a general planar Voronoi diagram. Figure 6 gives an illustrative example.

We prove in this section that medial axis convergence under the Hausdorff distance comes as a byproduct of the careful (though, of course, still C^1) biarc boundary conversion described in Section 2. We start with two technical lemmas.

For some shape $\mathcal{A}$ and a point $p \in M(\mathcal{A})$, let D_p denote the unique maximal disk with center p. Recall that $M(\mathcal{A})$ is defined as the union of the centers of all maximal disks. Define $\xi_p \leq \pi$ as the largest angle at p spanned by two points in the set $D_p \cap \partial\mathcal{A}$. The lemma below does not assume any regularity condition for the shape boundaries.

Lemma 6 *Let $\mathcal{A}$ and $\mathcal{B}$ be two shapes whose (two-sided) Hausdorff distance satisfies $H(\partial\mathcal{A}, \partial\mathcal{B}) = \varepsilon$. Define*

$$k = \frac{4}{1 - \cos(\xi_p/2)}$$

and let D_p denote any maximal disk for $\mathcal{A}$ whose radius r fulfills $r > k \cdot \varepsilon > 0$. Then there exists a maximal disk D_q for $\mathcal{B}$ such that $\|p - q\| < k \cdot \varepsilon$.

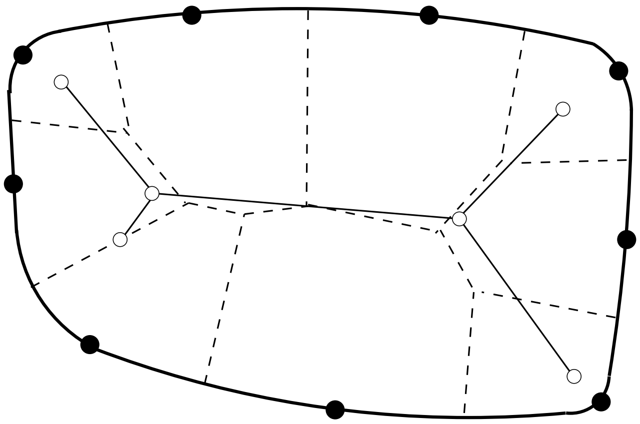

Figure 6: Few arcs versus small point sample

Proof. Let $x, y \in D_p \cap \partial\mathcal{A}$ be two points realizing the angle ξ_p, that is, $\angle xpy = \xi_p$. Let D'_p be the largest disk centered at p and contained in $\mathcal{B}$, and let $z \in D'_p \cap \partial\mathcal{B}$. Define D_q as the maximal disk with respect to $\mathcal{B}$ and containing D'_p; then clearly $z \in \partial D_q$. The claimed inequality follows from geometric considerations. It can be proved, in a straightforward way, that the disk D_s containing D'_p and satisfying $\|p - s\| = k \cdot \varepsilon$ will contain a small neighborhood of at least one of x, y and therefore, in addition to z, the disk D_s will contain at least one point of $\partial\mathcal{B}$. Therefore $D_q \subset D_s$ and

$$\|p - q\| \leq \|p - s\| = k \cdot \varepsilon.$$

□

Define a *leaf* of the medial axis as a vertex with a single incident edge. The following lemma describes the behavior of $M(\mathcal{A})$ in the vicinity of its leaves. Recall that an apex of $\partial\mathcal{A}$ is a point of maximal curvature.

Lemma 7 *For an apex x of $\partial\mathcal{A}$, consider the unique maximal disk D_p for $\mathcal{A}$ that avoids a segment of $\partial\mathcal{A}$ through x of fixed (small) length ℓ. Further, consider the maximal disk D_q osculating at x. If $\partial\mathcal{A}$ is piecewise analytic C^2 in the neighborhood of x then*

$$\|q - p\| \to 0 \textit{ as } O(\ell^2).$$

Proof. For a point $y \in \partial\mathcal{A}$, let n_y denote the normal line to $\partial\mathcal{A}$ through y. Given an apex x of $\partial\mathcal{A}$, define

$$I_x = \{n_y \cap n_z \mid d(y, x) \leq \ell, d(z, x) \leq \ell\},$$

with distance d being measured along $\partial\mathcal{A}$. Clearly, $q \in I_x$. Using the formal Taylor expansion of $\partial\mathcal{A}$ at x it can be proved directly that $H(p, I_x) \to 0$ as $\mathcal{O}(\ell^2)$. □

We are now prepared to prove the claimed convergence result. Slightly more general than in Section 2, we assume that $\partial\mathcal{A}$ for the original shape $\mathcal{A}$ is C^2 and piecewise analytic. (These requirements are fulfilled if $\partial\mathcal{A}$ is a cubic spline.) The proof generalizes easily to the case where $\partial\mathcal{A}$ is an arbitrary concatenation of analytic pieces, and thus, in particular, is allowed to contain 'sharp' vertices.

Let $\mathcal{B}_n$ denote some circular arc shape that comes from approximating $\partial\mathcal{A}$ by a suitable biarc spline; see Section 2. For sufficiently large n, each leaf of $M(\mathcal{A})$ is also a leaf of $M(\mathcal{B}_n)$, and all leaves of $M(\mathcal{B}_n)$ are contained in $M(\mathcal{A})$. This is because the spline preserves not only spirals, but also position, normal vector, and curvature at each apex x of $\partial\mathcal{A}$. All leaves are centers of osculating disks at some apex x.

Let us remove from $\partial\mathcal{B}_n$ the containing circular arc b_x for each apex x whose osculating disk is included in $\mathcal{B}_n$ (and hence is maximal for $\mathcal{B}_n$). This decomposes $\partial\mathcal{B}_n$ into components. The lengths of the arcs b_x shrink to zero as $\Omega(n^{-1})$ by construction of $\mathcal{B}_n$, as does their minimum d_n. Apart from disks for leaves, each maximal disk D_p for $\mathcal{B}_n$ has contact to at least two different components. (Otherwise, there would be a supplementary leaf of $M(\mathcal{B}_n)$.) For such a disk D_p, we have the angle inequality $\xi_p \geq \xi_n$, for

$$\xi_n = 2\arcsin(d_n/2L) \tag{1}$$

and L denoting the geometric diameter of $\mathcal{B}_n$. Because $d_n \to 0$ as $\Omega(n^{-1})$ and since L is a constant, we have $1-\cos(\xi_n/2) = \Omega(n^{-2})$. Moreover, $H(\partial\mathcal{A}, \partial\mathcal{B}_n) \to 0$ as $O(n^{-3})$ by construction. That is, the condition in Lemma 6 holds for almost all maximal disks D_p for $\mathcal{B}_n$ when n is sufficiently large. Consequently, for each point $p \in M(\mathcal{B}_n)$ there exists a point $q \in M(\mathcal{A})$ such that $\|p-q\| \to 0$ as $O(n^{-1})$. That is, the one-sided Hausdorff distance $\delta(M(\mathcal{B}_n), M(\mathcal{A}))$ converges at this speed.

The other direction can be proved similarly. For each apex x of $\mathcal{A}$, we define a neighborhood c_x on $\partial\mathcal{A}$ of length $n^{-3/4}$. Removal of the segments c_x leads us to two types of maximal disks D_q for $\mathcal{A}$, depending on whether D_q touches a single segment c_x (q is then close to x), or not. For the latter type, the analysis is the same as above, and shows that q approaches the center of some maximal disk for $\mathcal{B}_n$ at speed $O(n^{-3/2})$. For the former type, due to Lemma 7, the distance $\|q-p\|$ between q and the leaf $p \in M(\mathcal{B}_n)$ associated with c_x behaves as $\Theta(n^{-3/4})^2$, i.e., the same. The one-sided Hausdorff distance $\delta(M(\mathcal{A}), M(\mathcal{B}_n))$ thus converges at that speed.

Note that the global convergence speed of the medial axis with respect to the Hausdorff distance is $\Theta(n^{-1})$, whereas the error of the boundary approximation improves as $\Theta(n^{-3})$. This is due to the behavior of the medial axis close to its leaves. When we restrict ourselves to the λ-medial axis [5] for *any* $\lambda > 0$, then d_n in formula (1) becomes a constant, and the approximation speed is $\Theta(n^{-3})$ by Lemma 6. This well compares to using a size-n point sample on $\partial\mathcal{A}$ and pruning its Voronoi diagram, as the approximation speed then is only $\Theta(n^{-1})$.

7 CONCLUSIONS

We have given several examples for the efficient handling of shapes with nonlinear boundaries. In particular, the use of circular arcs for boundary conversion has been propagated. Our results profit from the confluence of geometric approximation theory and computational geometry. To our knowledge, this is the first systematic approach in this direction. Compared to conversion into polylines, the gain in efficiency increases with the complexity of the subsequent algorithm. This makes affordable suboptimal (hence sometimes less complicated) algorithms.

Other approximating primitives could be considered (e.g., cubics), but circular arc splines seem to yield the best trade-off. The presented algorithms, in principle, work for arbitrary primitives. In particular, in our medial axis algorithm, the added numerical complexity is not raised further by the algorithm itself. This is a nontrivial property of this algorithm, which is the first to combine practicality, efficiency, and stability. Its generalization to shapes with holes is possible, as Lemma 3 has a counterpart for this case.

Finally, we raise the question of whether results of this paper can be extended to three-space.

Acknowledgements Thanks go to Raimund Seidel for discussions on Section 5.

References

[1] H. Alt, O. Cheong, A. Vigneron. *The Voronoi diagram of curved objects.* Discrete & Computational Geometry 34 (2005), 439-453.

[2] D. Attali, J.-D. Boissonnat, H. Edelsbrunner. *Stability and computation of medial axes – a state-of-the-art report.* Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration, T. Mller, B. Hamann, B. Russell (eds.), Springer Series on Mathematics and Visualization, to appear.

[3] D. Avis, G.T. Toussaint. *An efficient algorithm for decomposing a polygon into star-shaped polygons.* Pattern Recognition 13 (1981), 395-398.

[4] B.K. Bhattacharya, H. El Gindy. *A new linear convex hull algorithm for simple polygons.* IEEE Trans. Information Theory IT-30 (1984), 85-88.

[5] F. Chazal, A. Lieutier. *Stability and homotopy of a subset of the medial axis.* Proc. 9^{th} ACM Symp. Solid Modeling and Applications, 2004, 243-248.

[6] F. Chazal, R. Soufflet. *Stability and finiteness properties of medial axis and skeleton.* J. Dynamical and Control Systems 10 (2004), 149-170.

[7] B. Chazelle. *A theorem on polygon cutting with applications.* Proc. 23^{rd} IEEE Symp. FOCS, 1982, 339-349.

[8] F. Chin, J. Snoeyink, C.A. Wang. *Finding the medial axis of a simple polygon in linear time.* Proc. 6^{th} Int. Symp. Algorithms and Computation, Springer LNCS 1004 (1995), 382-391.

[9] H.I. Choi, S.W. Choi, H.P. Moon. *Mathematical theory of medial axis transform.* Pacific J. Mathematics 181 (1997), 57-88.

[10] I.Z. Emiris, A. Kakargias, S. Pion, M. Teillaud, E.P. Tsigaridas. *Towards an open curved kernel.* Proc. 20^{th} Ann. ACM Symp. Computational Geometry, 2004, 438-446.

[11] G. Farin. Curves and Surfaces for Computer Aided Geometric Design. Academic Press, San Diego, 1997.

[12] G. Farin, J. Hoschek, M.-S. Kim. Handbook of Computer Aided Geometric Design, Elsevier, 2002.

[13] M.R. Garey, D.S. Johnson, F.P. Preparata, R.E. Tarjan. *Triangulating a simple polygon.* Information Processing Letters 7 (1978), 175-179.

[14] R.L. Graham. *An efficient algorithm for determining the convex hull of a finite planar set.* Information Processing Letters 1 (1972), 132-133.

[15] R.L. Graham, F.F. Yao. *Finding the convex hull of a simple polygon.* J. Algorithms 4 (1984), 324-331.

[16] M. Held, J. Eibl. *Biarc approximation of polygons with asymmetric tolerance bands.* Computer-Aided Design 37 (2005), 357-371.

[17] S. Hertel, K. Mehlhorn. *Fast triangulation of the plane with respect to simple polygons.* Information & Control 64 (1985), 52-76.

[18] R. Klein, K. Mehlhorn, S. Meiser. *Randomized incremental construction of abstract Voronoi diagrams.* Computational Geometry: Theory and Applications 3 (1993), 157-184.

[19] X. Kong, H. Everett, G.T. Toussaint. *The Graham scan triangulates simple polygons.* Pattern Recognition Letters 11 (1990), 713-716.

[20] D.T. Lee. *Medial axis transformation of a planar shape.* IEEE Trans. Pattern Analysis and Machine Intelligence PAMI-4 (1982), 363–369.

[21] D.T. Lee, F.P. Preparata. *Location of a point in a planar subdivision and its applications.* SIAM J. Computing 6 (1977), 594-606.

[22] D. McCallum, D. Avis. *A linear algorithm for finding the convex hull of a simple polygon.* Information Processing Letters 9 (1979), 201-206.

[23] D.S. Meek, D.J. Walton. *Approximation of a planar cubic Bézier spiral by circular arcs.* J. Computational and Applied Mathematics 75 (1996), 47-56.

[24] D.S. Meek, D.J. Walton. *Spiral arc spline approximation to a planar spiral.* J. Computational and Applied Mathematics 107 (1999), 21-30.

[25] A. Melkman. *On-line construction of the convex hull of a simple polygon.* Information Processing Letters 25 (1987), 11-12.

[26] C.J. Ong, Y.S. Wong, H.T. Loh, X.G. Hong. *An optimization approach for biarc curve fitting of B-spline curves.* Computer-Aided Design 28 (1996), 951-959.

[27] R. Ramamurthy, R.T. Farouki. *Voronoi diagram and medial axis algorithm for planar domains with curved boundaries I. Theoretical foundations.* J. Computational and Applied Mathematics 102 (1999), 119-141.

[28] U. Reif. *Uniform B-spline approximation in Sobolev spaces.* Numerical Algorithms 15 (1997), 1-14.

[29] M.A. Sabin. *The use of circular arcs to form curves interpolated through empirical data points.* Rep. VTO/MS/164, British Aircraft Corporation, 1976.

[30] Z. Šír, R. Feichtinger, B. Jüttler. *Approximating curves and their offsets using biarcs and Pythagorean hodograph quintics.* Computer-Aided Design 38 (2006), 608-618.

[31] X. Yang. *Efficient circular arc interpolation based on active tolerance control.* Computer-Aided Design 34 (2002), 1037-1046.

[32] C.K. Yap. *An $O(nlogn)$ algorithm for the Voronoi diagram of a set of simple curve segments.* Discrete & Computational Geometry 2 (1987), 365-393.

APPENDIX

Let $\boldsymbol{c}(t)$ be a given analytic curve on the domain $[t_0, t_1]$ and suppose that $\boldsymbol{c}(t)$ contains neither inflections nor apices in $[t_0, t_1]$. For given step size h, consider geometric primitives $b(t, h)$ that approximate the curve segments $\boldsymbol{c}[t, t+h]$. Assume that the domain of $\boldsymbol{c}(t)$ can slightly be enlarged to $[t_0, t_1 + h_{\max}]$, where $h_{\max}$ is a suitable constant which specifies the largest stepsize.

In order to evaluate the one-sided Hausdorff distance from $b(t, h)$ to $\boldsymbol{c}[t, t+h]$, we analyze the stationary points $\tau = \tau_i$ of the function

$$d(\tau, t, h) = \min_{q \in b(t,h)} \|q - \boldsymbol{c}(\tau)\|, \quad \tau \in [t, t+h],$$

which are characterized by

$$\frac{\partial}{\partial \tau} d(\tau, t, h)\bigg|_{\tau=\tau_i} = 0, \quad i = 1, \ldots, s(t) .$$

Provided that h is sufficiently small, the number $s(t)$ of stationary points is independent of t. For instance, this number is 1 for line segments and 2 for arcs interpolating three points. For each stationary point τ_i we consider the associated distance

$$d_i(t, h) = d(\tau_i(t, h), t, h)$$

where we define $d_i(t, 0) = 0$. The one-sided Hausdorff distance $\delta(b(t, h), \boldsymbol{c}[t, t+h])$ is the maximum of all these distances.

For each value of t, consider the Taylor expansion at $t = 0$. The first non-vanishing derivative is used to define the remainder term,

$$d_i(t, h) = \frac{1}{k!}\, d_i^{[k]}(t, h_i^*(h)) \cdot h^k,$$

where $^{[k]}$ indicates the k^{th} derivative with respect to the step size h and $h_i^*(h) \in [0, h]$. The order k of this term is called the *approximation order* of the geometric primitive; it equals 2 for line segments and 3 for circular arcs.

Since the curve $\boldsymbol{c}(t)$ contains neither inflections nor apices, and due to the compactness of its domain $[t_0, t_1]$, there exist positive constants C, D such that the functions d_i satisfy

$$0 < C < d_i^{[k]}(t, 0) < D.$$

Moreover, since the k^{th} derivative is continuous, there exists a step size $g > 0$ such that

$$\forall (t, h^*) \in [t_0, t_1] \times [0, g]: \quad \frac{C}{2} \le d_i^{[k]}(t, h^*) \le 2D.$$

Consequently, if h is sufficiently small, then the one-sided Hausdorff distance δ satisfies

$$\frac{1}{k!}\frac{C}{2} h^k \le \delta \le \frac{1}{k!} 2D h^k .$$

This proves $\delta = \Theta(h^k)$ for the case where k is a constant.