Medial Axis Computation for Planar Free–Form Shapes

O. Aichholzer[a], W. Aigner[a], F. Aurenhammer[a], T. Hackl[a], B. Jüttler[b], M. Rabl[b]

[a] University of Technology Graz, Austria
[b] Johannes Kepler University Linz, Austria

Abstract. We present a simple, efficient, and stable method for computing—with any desired precision—the medial axis of simply connected planar domains. The domain boundaries are assumed to be given as polynomial spline curves. Our approach combines known results from the field of geometric approximation theory with a new algorithm from the field of computational geometry. Challenging steps are (1) the approximation of the boundary spline such that the medial axis is geometrically stable, and (2) the efficient decomposition of the domain into base cases where the medial axis can be computed directly and exactly. We solve these problems via spiral biarc approximation and a randomized divide & conquer algorithm.

Keywords. Planar shape, biarc approximation, medial axis, stability, divide & conquer, randomized algorithm, numerical robustness

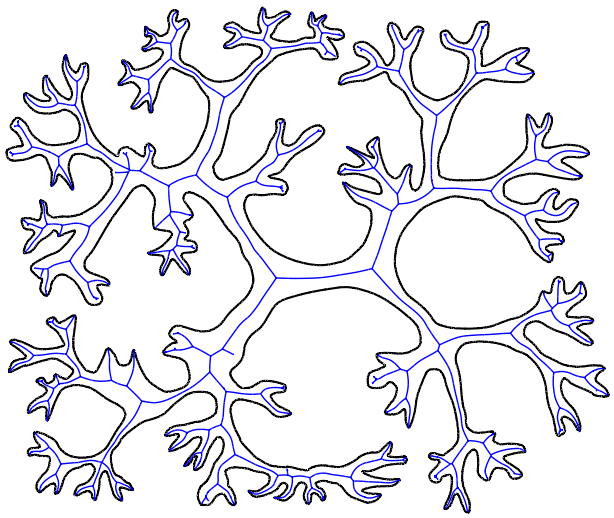


Figure 1: Medial axis of a planar free-form shape. The shape was approximated by 9440 arcs within 0.65 seconds, and the computation of the medial axis took less than 1 second. All computations were done on standard PCs.

1 Introduction

The medial axis has been introduced by H. Blum [5] as a concept for efficient shape description. Meanwhile it has proven useful in many scientific areas, and its fast and stable computation is of vital interest. However, even in $\mathbb{R}^2$, the task of computing the correct medial axis of a given free-form shape is a highly nontrivial one. See Fig. 1 for a first example.

The efficiency and quality of the axis' computation critically depend on the available boundary representation of the input shape. Algorithms for polygonal boundaries [10, 22, 26, 34] work at satisfactory runtimes, but do not produce stable medial axis approximations for the original shape without expensive pruning. The same is true for point sample representations [4, 6], which also (and even more) tend to increase the data volume.

On the other hand, implementations that work directly on curved boundaries suffer from high numeric complexity and the arising robustness problems. Also, they usually are inherently slow, as many existing efficient algorithms do not apply to complicated curved objects; see e.g. [13] for a short overview of relevant previous work until 2002. As interest in computing the medial axis has found renewal in recent years, let us briefly further comment on this challenging problem.

There exist two principal problems—apart from stability issues—that need to be addressed when computing a medial axis. One of them is determining the combinatorial structure (i.e., the topology) of the medial axis. This problem has been well solved, from both a theoretical and practical point of view, only for point sample and polygonal inputs. For curved boundary objects, most theoretically fast algorithms compute the entire Voronoi diagram, leaving the need of pruning away unwanted and incorrect features. Complicated merging, or insertion, steps have to be performed, depending on whether the algorithm was based on divide & conquer [25, 34], or on incremental insertion [3, 31]. As such steps process previously computed parts of the medial axis, they are numerically involved and subject to errors if not implemented with care [19].

Algorithms based on domain decomposition [11] avoid these drawbacks. They lead to a divide & conquer construction [12, 18] as well, but their merging steps are trivial, as effort is shifted to the process of splitting into independent subproblems. In other words, they allow for separating combinatorial calculations from geometric calculations in the medial axis computation. The algorithm we are going to describe in this paper is of this type.

Even when the topology of the medial axis is assumed to be known, the (usually hard) problem of computing its bisectors remains. Quite a lot of work has been devoted to this geometric aspect of the medial axis. See, for example, [17] who focus on

rational boundary curves, and [15] where curvature properties are utilized for treating cubic boundary splines. A popular approach is local tracing [14, 32], where the medial axis is calculated by tracing either the shape boundary or the axis bisectors. In particular, so-called predictor/corrector methods [8, 15] have been proposed for approximating the medial axis in a piecewise manner.

All these approaches described above are rather theoretical work—a practical one is given in [16]. They compute the medial axis by first approximating the boundary spline curve by circular biarcs and then applying the VRONI-package developed by M. Held [22]. VRONI can compute the medial axis of a collection of N points and line segments in (practically) $\mathcal{O}(N \log N)$ time; circular arcs are accepted, too, and are converted into a polygonal description. The implemented algorithm is basically incremental insertion, and is capable of constructing the entire Voronoi diagram. Although the computation is done very fast in terms of the input size, N, the resulting two-step approximation [16] blows up the data volume significantly.[1] Also, no guarantee for the stability of the medial axis approximation can be given.

In the present paper, we describe a simple and fast method that is less data consuming (and thus is efficient also in this sense), and that comes with a stability guarantee. We use an approximation of the shape boundary by biarcs as well, though in a tailored manner. Our algorithm then works directly (and exactly) on shapes bounded by circular arcs. This bears two major advantages: (1) For a fixed accuracy of the approximation, the data volume drops from N to $n = \mathcal{O}(N^{2/3})$ compared to using a polygonal description. (2) The biarc approximation scheme can be tuned to preserve monotonicity of curvature of the original shape, which makes the computed medial axis converge to the exact one. Note that the medial axis of a shape with piecewise circular boundary is composed of conic arcs, and thus has the same analytic complexity as for polygonal domains.

We adopt the shape decomposition approach [11] to achieve simplicity and numerical robustness of the algorithm. As decomposition is by inscribed maximal disks, it is naturally suited to shapes with piecewise circular boundaries. The resulting randomized divide & conquer algorithm runs in expected time $\mathcal{O}(n \log n)$ if mild assumptions on the graph diameter of the medial axis are met. A high-level description, including a formal runtime and data volume analysis, and a proof of convergence (medial axis stability) are given in [1]. The theoretical foundations being laid, the paper at hands concentrates on practical and experimental aspects of the algorithm.

Section 2 details the method we use for approximating a given polynomial spline curve by spiral biarcs. A careful description of our medial axis algorithm follows in Section 3. Continuing preliminary work in [2], a variant of the algorithm is worked out that performs the best concerning speed while ensuring robustness in the presence of geometric degeneracies. This includes (but is not restricted to) the proper classification and treatment of base cases, in order to establish correctness and to gain running speed, for both smooth and non-smooth circular boundary spline inputs. Implementation details, experimental data, and selected examples are presented in Section 4. Finally, Section 5 offers some concluding remarks.

[1] We recently learned that an advanced version of VRONI is under implementation, which will be able to process circular arc inputs directly. A sweepline algorithm for computing the Voronoi diagram of a set of circles has been presented in [24].

2 Approximating the shape

Biarc approximation of free-form curves has been studied by many authors, see e.g. [23, 27, 33] and the references cited therein. In order to make this paper self-contained, we present the algorithms which we use for approximating general free-form domains with domains bounded by arc splines.

2.1 Biarcs

A biarc (a_0, a_1) is obtained by joining two circular arcs a_0 and a_1 in a way such that they possess a common unit tangent vector at their joint J. For any given set of G^1 Hermite data, which consists of two endpoints P_0, P_1 and associated unit tangent vectors v_0, v_1, there exists a one–parameter family of interpolating biarcs.

The possible joints J form a circle, which is called the joint circle, cf. Fig. 2. This circle passes through the endpoints P_0 and P_1 and it spans the same oriented angles with the tangent vectors v_0 and v_1, respectively (see Fig. 2 and e.g. [33]); its center is found by intersecting the perpendicular bisectors of the two point pairs (P_0, P_1) and $(P_0 + v_0, P_1 + v_1)$.

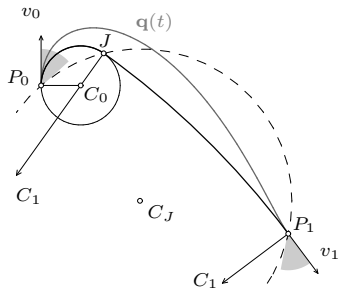


Figure 2: A planar curve $\mathbf{q}(t)$ (grey), G^1 Hermite data (P_0, v_0) and (P_1, v_1), joint circle (dashed) with oriented angles (light grey), and the spiral biarc.

The biarc is uniquely determined once a joint J on the joint circle is selected. Various possible choices have been proposed in the literature [27, 33]. In view of the medial axis computation we need a representation of the given shape boundary that preserves the curvature extrema. We, therefore, focus on so-called spiral biarcs.

2.2 Spiral biarcs

Meek and Walton [28] propose a biarc construction scheme that guarantees that the arc spline approximation of a smooth spiral

(i.e., of a curve with monotonic curvature) is again a spiral. Assume that the G^1 Hermite data are sampled from a spiral curve, and let k_0 and k_1 denote its curvatures at P_0 and P_1, respectively, where we assume that $k_0 > k_1 > 0$. We choose the arc a_0 as a segment of the osculating circle of the spiral at P_0, hence the joint J is obtained by intersecting the joint circle with the osculating circle. The second arc a_1 passes through J and P_1 and matches the tangent v_1. According to [28], radii and curvatures satisfy $r_0 < r_1 < 1/k_1$.

Let P_2 and v_2 be a further given set of Hermite data, sampled from the same spiral, with curvatures $k_1 > k_2 > 0$. The first arc of the following biarc is chosen as a segment of the osculating circle at P_1, hence its radius satisfies $1/k_1 > r_1$. It follows that, when using spiral biarcs, one obtains an approximation by a curve with piecewise constant, but monotonic, curvature. The approximation order of spiral biarcs is three.

In order to apply this method to a polynomial spline curve, it is necessary to split the curve at points with stationary curvature, which we will refer to as *apices* throughout this paper (since the notion of vertices will be used with a different meaning), and at points with curvature discontinuities. In the cubic case, the apices can be found by numerically solving polynomials of degree 5, and the curvature discontinuities are located at knots with multiplicity ≥ 2.

The method for computing a spiral biarc is summarized in Algorithm 1.

Algorithm 1 `spiralbiarc`(P_0,P_1,v_0,v_1,k_0)
{Construct a spiral biarc}

$b_1 \leftarrow$ bisector of P_0 and P_1
$b_2 \leftarrow$ bisector of $P_0 + v_0$ and $P_1 + v_1$
$C_J \leftarrow b_1 \cap b_2$ {center of joint circle}
$r_J \leftarrow \|C_J - P_0\|$ {radius of joint circle}
$r_0 \leftarrow 1/k_0$ {radius of a_0}
$C_0 \leftarrow P_0 + r_0 \cdot v_0^{\perp}$ {center of a_0}
$J \leftarrow$ (circle $(C_J, r_J) \cap$ circle $(C_0, r_0)) \setminus \{P_0\}$ {joint}
$a_0 \leftarrow (C_0, r_0, P_0, J)$ {first arc}
$C_1 \leftarrow$ line $(J, C_0) \cap$ line $(P_1, P_1 + v_1^{\perp})$ {center of a_1}
$r_1 \leftarrow \|P_1 - C_1\|$ {radius of a_1}
$a_1 \leftarrow (C_1, r_1, J, P_1)$ {second arc}
return (a_0, a_1)

2.3 Adaptive bisection

Assume we have a spline curve segment $\mathbf{q}(t)$, $t \in [t_0, t_1]$ without apices. In order to produce a spiral biarc approximation, where the maximum error is bounded by a given threshold ε, we use adaptive bisection:

1. Create the biarc (a_0, a_1) for the given segment.
2. Evaluate the approximation error using Algorithm 2.
3. If the error is too large, then split the segment into halves and apply the algorithm to the two subsegments, else stop.

Alternatively, other techniques—such as the method proposed in [23]—can be used. We choose the simple bisection algorithm because of its simplicity and the runtime complexity of $\mathcal{O}(n)$ with respect to the number of output elements, the n arcs.

In order to evaluate the approximation error between the spiral biarc (a_0, a_1) and the given curve, we measure the normal distances with respect to the circular arcs in sampled points on $\mathbf{q}(t)$. Since the joint J is not located on the curve, we first match each circular arc to its corresponding segment of $\mathbf{q}(t)$, $t \in [t_0, t_1]$. This is done by projecting J to the curve $\mathbf{q}(t)$, where C_0 is used as the center of projection. The parameter value t_J of the projected joint J_q is found by solving a polynomial equation of degree d, where d is the degree of the spline curve. If there exist multiple solutions within the given interval, then the error is set to ∞, otherwise we estimate the one-sided Hausdorff distance.

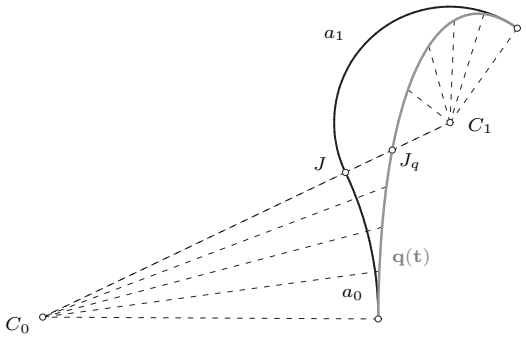


Figure 3: Estimating the normal distances between the curve and the approximating spiral biarc.

The method for estimating the approximation error is summarized in Algorithm 2.

Algorithm 2 `errorbiarc`(C_0,r_0,C_1,r_1,$\mathbf{q}(t)$,$[t_0, t_1]$)
{Distance of biarc and curve}

$D_0 \leftarrow 0$, $D_1 \leftarrow 0$ {initialization}
$t_J \leftarrow$ line $(J, C_0) \cap \mathbf{q}(t), t \in [t_0, t_1]$ {project J onto curve}
if t_J is unique **then**
  for $i = 0$ to s **do** {$s \ldots$ number of sampled points}
    $D_0 \leftarrow \max(D_0, |\,\|\mathbf{q}\,(t_0 + i(t_J - t_0)/s) - C_0\| - r_0|)$
    $D_1 \leftarrow \max(D_1, |\,\|\mathbf{q}\,(t_J + i(t_1 - t_J)/s) - C_1\| - r_1|)$
  end for
  return $\max(D_0, D_1)$
end if
return ∞

The sampling-based approach leads to a slight underestimation of the error. In practice it performs quite well and it is very fast.[2]

[2]The following alternative for bounding the error could be used. One can directly compute the points on the curve which have extremal distances to the given curve, by solving the piecewise polynomial equations $\dot{\mathbf{q}}(t) \cdot (\mathbf{q}(t) - C_i) = 0$, $i = 0, 1$, where t varies within $[t_0, t_J]$ and $[t_J, t_1]$ for the first and the second arc, respectively. In the case of cubic splines, this leads to quintic equations. The maximum distance then can be computed as the maximum of the distances at these finitely many points. Instead of this exact approach, it is also possible to

2.4 Approximation properties

Circular arcs segments approximate a given curve segment with approximation order three. Similarly, an approximating spiral biarc spline with n circular arcs possesses the same approximation order, and therefore the error ε improves as $\Theta(n^{-3})$; cf. [27, 28].

Given a sequence of approximating curves that converge to the (exact) boundary of a given planar domain, the medial axes of the approximate domains do not necessarily converge to the medial axis of the given domain. For instance, this is obvious in the case of approximation by polygons, where each vertex creates its own branch of the medial axis.

The case of approximation by spiral biarcs, however, is different, and has been analyzed in [1]. Since the curvature maxima are preserved by the spiral biarc approximation, the number of leaves of the approximate medial axis is equal to the number of leaves of the exact medial axis. Consequently, the approximation does not create any additional branches of the medial axis. Moreover, we have geometric convergence as follows.

Assume that the Hausdorff distance between the exact and the approximate domain boundary is at most ε. For any point p on the exact medial axis which is sufficiently far away from the leaves (where the required distance tends to zero as $\varepsilon \to 0$), it is possible to derive a bound on the distance d_p to the nearest point on the medial axis of the approximate domain, namely,

$$d_p \leq \frac{4}{1-\cos(\xi_p/2)} \cdot \varepsilon$$

Here, $\xi_p \in [0, \pi]$ is the maximum angle between any two rays that connect the center of the maximal inscribed circle which is centered at p with any two of its tangency points. Consequently, except for the vicinity of the leaves, the medial axis inherits the approximation order 3 of the boundary approximation by spiral biarcs. The global error—including the leaves—can be shown to behave as $\Theta(n^{-1})$. See [1] for more information.

If only nodes with valency three are present, then the spiral biarc approximation preserves the topology of the medial axis, provided that the error of the boundary approximation is sufficiently small. In the case of nodes of higher valency, these nodes may split in various ways into neighbouring nodes of lower valency.

3 Computing the medial axis

In this section, we develop a variant (and provide a detailed implementation) of the randomized divide & conquer algorithm in [1] which performs at high speed and is relatively robust against degenerate inputs. The algorithm computes the exact medial axis of a shape given in (any) piecewise circular boundary representation. Together with the advantages from the spiral biarcs approximation, this means that the computed axis converges towards the axis of the original shape with increasing approximation quality. The expected runtime is $\mathcal{O}(n \log n)$ under the assumption that the graph diameter of the medial axis is $\Theta(n)$. This condition does not mean a real restriction in practice. The number of branching points of the medial axis is independent from the input size n (the number of circular arcs) which, in turn, grows arbitrarily with the user-defined accuracy of the output.

derive an upper bound on the distance by analyzing the B-spline coefficients of $\|q(t) - C_i\|^2$.

3.1 Overall algorithm

The algorithm is based on the fact that decomposing a given shape with an inscribed disk leaves two (or more) subdomains whose medial axes can be computed independently. This observation has been extensively made use of in [11]. It holds for simply connected planar shapes of any form, and is particularly suited for our purposes because we deal with piecewise circular boundaries already.

In a nutshell, the algorithm proceeds as follows. Its *divide step* calculates a random dividing disk and checks whether the induced decomposition is progressive, i.e., whether the resulting subdomains are combinatorially smaller (containing less arcs) than the domain itself. In the negative case, the disk is recomputed deterministically to fulfill this requirement. Each subdomain is then treated recursively, until one of the base cases is reached and the medial axis is calculated directly. The *conquer step* only concatenates the already computed medial axes for the subdomains, as they fit together at the centers of the dividing disks.

Thus, the expensive and critical computations are delegated to the divide step. In the conquer step, the subsolutions are simply glued together without the need of any merging or adjustment operations. This reduces the effect of error accumulation, and keeps numerical imprecision, if it occurs at all, locally restricted.

In the remainder of this section, let $\mathcal{A}$ denote the piecewise circular approximation of the original shape, and let $\partial\mathcal{A}$ stand for its boundary. The algorithm will accept any circular arc spline for $\partial\mathcal{A}$, and thus will also work if $\partial\mathcal{A}$ is polygonal.

Before proceeding to a detailed descriptions of the algorithm's steps, let us recall the formal definition of a medial axis. Let MAT be the set of all maximal disks that can be inscribed into the shape $\mathcal{A}$. A disk D is called *maximal* if there exists no other disk $D' \subset \mathcal{A}$ such that $D' \supseteq D$. The medial axis of $\mathcal{A}$ is defined as

$$M(\mathcal{A}) := \{\mathcal{P} \mid \exists D \in MAT : \mathcal{P} \text{ is center of } D\}\,.$$

$M(\mathcal{A})$ defines a tree (in the graph-theoretical sense) because the underlying shape $\mathcal{A}$ is simply connected.

3.2 Divide step

The divide step carefully chooses a maximal disk D and splits the shape boundary $\partial\mathcal{A}$ into two or more chains, depending on the number of tangency points of D. The resulting subshapes are completed with circular arcs which have D as their supporting circle. We call such arcs *artificial* arcs. Every maximal disk is, via its tangency points, uniquely assigned to two or more arcs

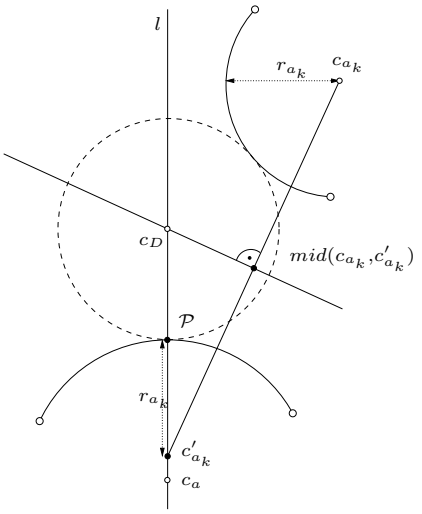


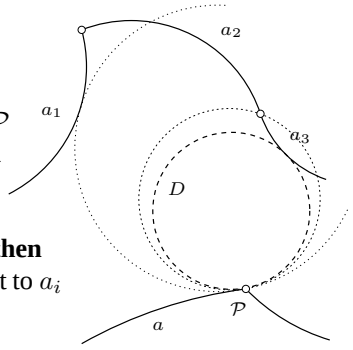
Figure 4: Constructing a disk that is tangent to two arcs.

on $\partial\mathcal{A}$. A possible way to pick a random maximal disk [1] is to choose a random arc a on $\partial\mathcal{A}$ and to construct the disk that is tangent to a at a fixed point $\mathcal{P}$, e.g., its midpoint or one of its endpoints.

Given the set of arcs $a_i, i = 1 \dots n$, that represent $\partial\mathcal{A}$ in clockwise order, this is accomplished by iteratively constructing disks that are tangent to a at the point $\mathcal{P}$ and to some other arc a_k. If the resulting disk still intersects or overlaps an arc a_l with $1 \le k \le l \le n$, then a new disk (which is smaller than the preceding one) tangent to a_l is computed, until we obtain a valid maximal disk D. (For step by step details of the `maximaldisk` procedure see Algorithm 3). As we have to check all n arcs of the boundary, we obtain an $O(n)$ time complexity for the computation of a single maximal disk.

Algorithm 3 `maximaldisk`(a,$\partial\mathcal{A}$)
{Compute a maximal disk on a in $\mathcal{A}$}

1: **if** a has a reflex endpoint **then**
2: $\mathcal{P} \leftarrow$ reflex endpoint
3: **else**
4: $\mathcal{P} \leftarrow$ midpoint of a
5: **end if**
6: $D \leftarrow$ halfplane tangent at $\mathcal{P}$
7: $k \leftarrow$ number of arcs on $\partial\mathcal{A}$
8: **for** $i = 1 \dots k$ **do**
9: $a_i \leftarrow$ ith arc of $\partial\mathcal{A}$
10: **if** $a \neq a_i \wedge D \cap a_i \neq \emptyset$ **then**
11: $D \leftarrow$ disk at $\mathcal{P}$ tangent to a_i
12: **end if**
13: **end for**
14: **return** D



The central part of this calculation is the geometric construction of a disk which is tangent to an arc a at a fixed point $\mathcal{P}$, and which is arbitrarily tangent to another arc a_k. See Fig. 4 for an illustration. The point c_D, which is the center of the desired maximal disk, is the matter of interest. This point must lie on the line l through c_a, the center of a, and $\mathcal{P}$. If we move from the point $\mathcal{P}$ a distance of length r_{a_k} (the radius of a_k) towards c_a, we arrive at the point c'_{a_k}. Together with c_{a_k} and c_D this point forms an isosceles triangle. This fact can be exploited to construct c_D. We compute the perpendicular bisector between c_{a_k} and c'_{a_k} and intersect it with l, which gives the point c_D. This construction can, with slight modificiations, be applied to pairs of arcs in arbitrary position. If we replace the circular arc a_k by a line segment, the problem can be reduced to the intersection of the line l with an angle bisector of the line perpendicular to l through $\mathcal{P}$ and the supporting line of the segment.

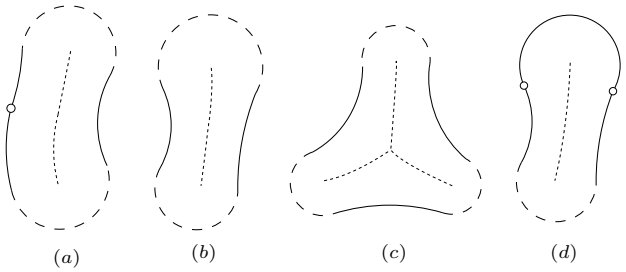


Figure 5: Base cases for smooth boundaries.

This disk construction is, together with intersection and overlap checks, the most frequent and numerically most complex step in the entire medial axis algorithm. Thus the main atomic operations are computing intersections of circles and lines.

Let us proceed to the classification and analysis of appropriate termination conditions for the divide step. If we consider a G^1 boundary as precondition, then we can decompose any shape bounded by circular arcs and line segments into only four base cases; see Fig. 5. This is simply accomplished by dividing iteratively until the number of non-artifical arcs drops below four. Let us argue that the cases in Fig. 5 cover all possibilities. Observe first that no consecutive artificial arcs may occur, because for smooth boundaries we construct every maximal disk at the midpoint of an arc a.

- All possible constellations with 3 non-artificial arcs are covered in the cases (a), (c), and (d), provided no consecutive artificial arcs are allowed.
- The combination shown in case (b) is the only one which may occur with 2 non-artificial arcs.[3]

If we do allow reflex and convex vertices on the boundary, then we have to pay more attention to the choice of the point $\mathcal{P}$ to reduce the number of arising base cases. If a randomly chosen arc a has some reflex endpoint, we do not choose its midpoint but rather the reflex endpoint itself as $\mathcal{P}$.

[3] A base case with two *consecutive* non-artificial arcs, connected by an artificial arc while guaranteeing smoothness at all vertices, is only possible in a degenerate case: All arcs would have to be on the same supporting circle. The same applies to the hypothetical case of one artificial and one non-artificial arc.

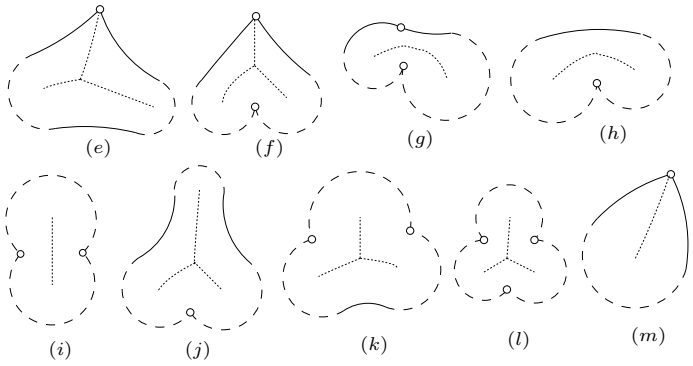
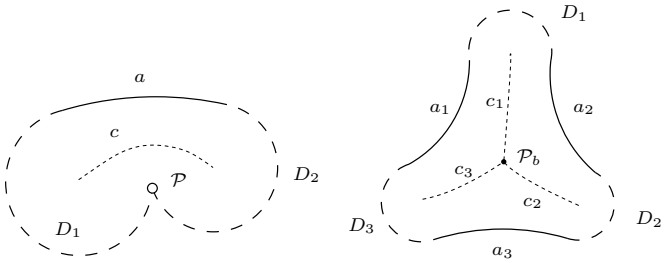


Figure 6: Base cases for G^0 boundaries.



(a) Base case (h) (b) Base case (c)

Figure 7: Two base cases in detail.

Furthermore the termination conditions have to be slightly extended. We keep on splitting until all of the following criteria are satisfied:

1. The number of non-artificial arcs is ≤ 3.
2. There exists no non-artificial arc with a reflex vertex.
3. If three non-artificial arcs are consecutive then no convex vertex occurs. (Note that this last criterion might lead to redundant cuts, which are dealt with in section 3.4.)

This results in nine additional possible base cases as shown in Fig. 6. These new cases cover all possible non-smooth varations of the cases (a), (b), (c), and the degenerate case from footnote[3]. Variations include the turning of a smooth vertex into a convex one and the replacement of an isolated non-artificial arc with a reflex vertex. The base case (d) has no non-smooth derivatives because of splitting rule 3. Additionally, if we consider a reflex vertex as an arc with length zero, we can maintain the observation that no consecutive artificial arcs do occur. Together with the following analytic enumeration of the new base cases it is obvious that the arguments concerning completeness of the smooth cases apply to the situation of a G^0 boundary as well.

- For smooth case (a) the joint vertex can become convex, the isolated non-artificial arc can be exchanged by a reflex vertex, or both. These variations are covered by the cases (e), (g), and (f).
- The two variations of smooth case (b) are obtained by replacing either one or both non-artificial arcs by reflex vertices. See case (h) and case (i) for a realization of this.
- The new base cases (j), (k), and (l) represent all possible combinatorial variations of smooth case (c) caused by turning isolated non-artificial arcs into reflex vertices.
- Finally the degenerate case mentioned in footnote[3] allows one variation by creating a convex vertex from a smooth one. This constellation is covered by base case (m).

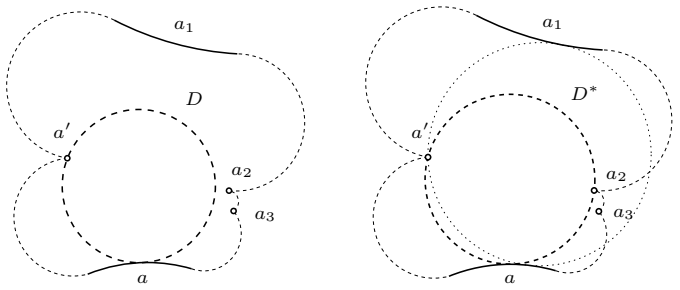
3.3 Conquer step

In the conquer step, the medial axes of the base cases are computed directly, and then are concatenated at centers of maximal disks which support the respective artifical arcs. At this point, we know exactly which parts of the (global) medial axis correspond to which parts of the boundary of the shape. As the shape boundary is piecewise circular, the medial axis consists of conic arcs. Each such arc is assigned to two primitives on the boundary where it is equidistant from. Possible primitives are circular arcs, line segments, and points (boundary vertices). Different pairs of primitives result in different types of conics:

- Two circular arcs may define an elliptic or a hyperbolic arc, depending on the position of the two supporting disks, and the orientation of the arcs on the boundary.
- A circular arc and a line always define a parabolic arc.
- A circular arc and a point define an elliptic arc if the point lies inside the arc's supporting disk, and a hyperbolic arc, otherwise.
- Two line segments define a straight line.
- A line segment and a point define a parabolic arc.
- Two points again define a straight line.

For illustratory reasons, let us give two examples. Consider the base case (h) with a labeling as in Fig. 7a. The only two non-artificial primitives on the boundary, arc a and point $\mathcal{P}$, define the conic arc c. As $\mathcal{P}$ lies inside the supporting disk of a, the curve c, leading from the center of D_1 to the center of D_2, is an elliptic arc.

Next, consider base case (c) where a branching of the medial axis occurs (Fig. 7b). Curve c_1 is a hyperbolic arc defined by a_1 and a_2. The same holds for the curves c_2 and c_3 which stem from the pairs a_2, a_3 and a_3, a_1, respectively. The special feature of this base case is the branching point $\mathcal{P}$. A branching point is a point on the medial axis which is equidistant from at least three primitives on the boundary. Its assigned maximal disk touches the boundary at more than two points. Branching points are required as endpoints for our conic arcs, and thus have to be computed directly. In our example, we have to compute a point which has the same distance to three arcs. If we replace the arcs by line



(a) Non-reducing disk D (b) Branching disk D^*

Figure 8: The disk D^*, centered at a branching point, is constructed by `maximaldisk`* after detection of a non-reducing dividing disk D.

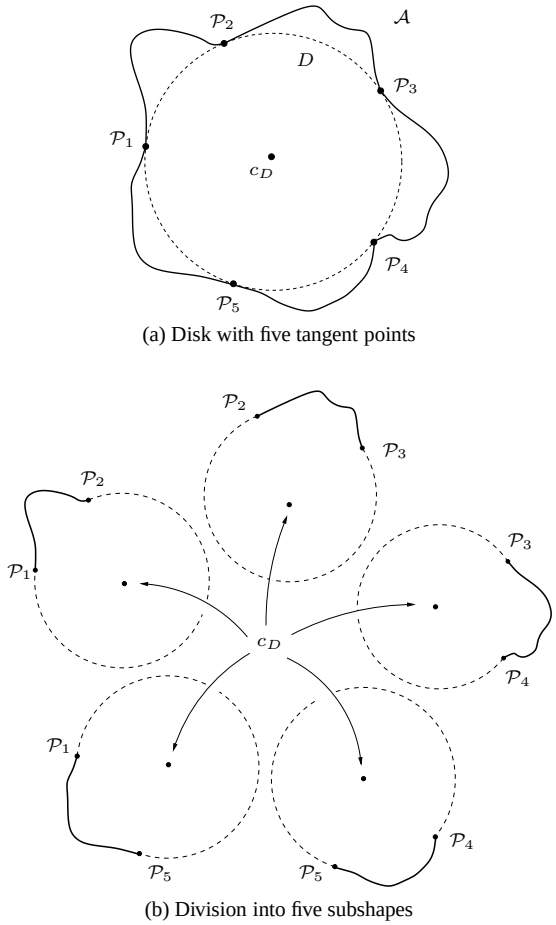
segments or (reflex) points, as in the base cases (j), (k), and (l), we get ten possible combinations of three primitives. What we are looking for is the disk that is simultaneously tangent to all three of them.

This problem is known as the Apollonius problem, named after the ancient Greek geometer who posed this problem about 200 B.C. (discussed among others by [20]). As up to eight circles may satisfy the tangency conditions to the circles (lines) supporting the primitives, we have the problem of singling out the unique valid disk that touches them at the right portion. We have implemented this task for all triples of primitives, as this is needed in the computation of all the branching points ocurring in the base cases (c), (e), (f), (j), (k), and (l).

3.4 Preventing redundant cuts

During the division process—especially when dealing with reflex boundary vertices—situations may occur where a disk obtained by the `maximaldisk` algorithm fails to decompose the shape into (combinatorially) smaller subdomains. As the property for shapes to shrink is needed to assure a termination of the algorithm, such a situation may lead to an infinite loop. To see an example, an arc a for the construction of the maximal disk may be chosen that causes base cases of the form (h) to be cut away from a over and over again. The remaining subdomains have the same number of non-artificial arcs as the preceding ones, and so undergo no combinatorial reduction. See Fig. 8a for an illustration. This unwanted phenomenon can be detected, and subsequently be avoided, by a more sophisticated choice of the dividing disk. As a pleasing side effect, this choice will also handle the intriguing case of multi-branching of the medial axis.

As soon as a non-reducing disk D, as shown in Fig. 8a, is detected, we invoke algorithm `maximaldisk`*, which computes a disk D^* that is tangent to the shape boundary at three (or more) points instead of only two. Similar to the original algorithm `maximaldisk`, the procedure traverses all boundary arcs. It checks, however, which of them yields the third point of tangency of the needed branching point disk. The first two contact points are known to be on the footarc a and on the arc a' chosen by the `maximaldisk` algorithm for D. The main feature of the



(a) Disk with five tangent points

(b) Division into five subshapes

Figure 9: A degenerate case where five branches of the medial axis meet at a single point c_D. The shape is decomposed into five subshapes. This situation is handled by `maximaldisk`*.

new construction is the lack of a fixed point $\mathcal{P}$ on any of the arcs. As is revealed in Fig. 8b, a disk tangent to the three arcs a, a', and a_1 gets constructed first.[4] As long as this disk overlaps another arc (here e.g. a_2), a new disk on a, a', and this very arc is constructed. This process terminates with the desired disk centered at a branching point of the medial axis. Each of the three resulting subshapes is lacking at least one non-artificial arc, namely, one of the tangent primitives. Thus a reduction is guaranteed.

The `maximaldisk`* algorithm also recognizes and handles multi-branchings, i.e., nodes of the medial axis with valency four or more. If a valid branching point disk D is tangent (or, for the implementation, ε-tangent for a predefined small ε) to $m \geq 4$ primitives, then such a multiple branching point occurs. Every tangent arc defines a point of tangency for D on the shape boundary, and the shape is divided into m subshapes which are all joined together at D. Fig. 9 gives an illustration.

When several reflex vertices agglomerate in a relatively small

[4] Unlike in this example, the first defining arc a_1 after a' does not necessarily result in an applicable starting disk. Note that an arc a_1 in unfavorable geometric position might lead to a disk which has its center on the wrong side of the line defined by the points of tangency on a and a'.

area of the shape (perhaps with no separating boundary parts) then another non-reducable case may occur: a subshape consisting of an arbitrary number of artificial arcs, separated by arcs of zero length (as they result from reflex boundary vertices). The medial axis of such a case is a subset of the standard Voronoi diagram, with the zero length arcs as the defining points. With a construction very similar to the `maximaldisk`* algorithm, these cases can be reduced to base cases of the form (l) from Fig. 6. Two zero length arcs (points) neighboured on the subshape's boundary are fixed. A third zero length arc is then determined in the iterative process, such that the disk defined by these three points does not contain any other point. This disk is a valid maximal disk, which is tangent to the boundary at three points.

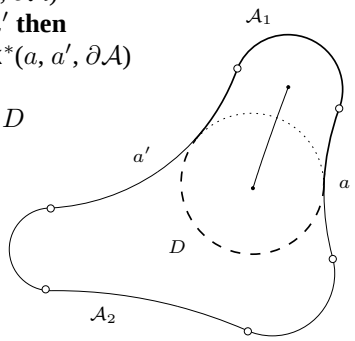
3.5 Putting things together

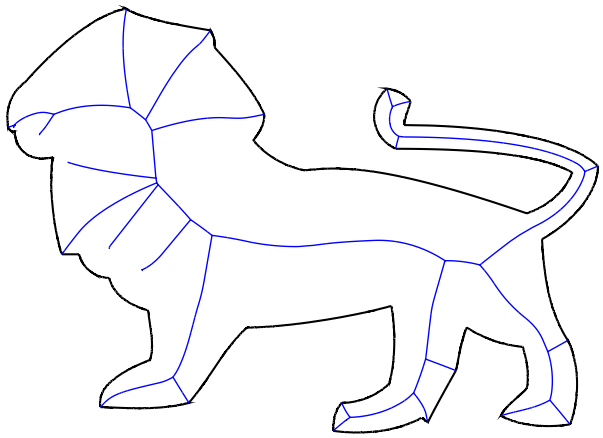
By combining the procedures introduced above we obtain the main algorithm for the medial axis computation, as lined out in Algorithm 4. Its input is the shape approximation, $\mathcal{A}$, represented by its piecewise circular boundary $\partial\mathcal{A}$. The algorithm divides $\mathcal{A}$ recursively into partial shapes, until they match any of the base cases introduced before. The choice of the disk (constructed by `maximaldisk`) which is used for the decomposition is random[5] at first. If a non-reducing disk occurs, then a disk centered at a branching point is computed by the extended algorithm `maximaldisk`*. If the state of a base case is reached, we may proceed in two possible ways:

- The medial axis of the base case is computed directly. It exclusively consists of conic arcs. This is one of the benefits from the circular boundary representation.
- For certain applications, the curve equations of the axis segments may be of small or no interest at all, as rather the topological or combinatorial structure is needed. Through the use of base cases, which reveal various special features of the shape and its medial axis (branching points, local curvature maxima, etc.), it is easy to derive useful information on the axis without calculating the conic arcs right away. By storing the combinatorics of all base cases, the exact medial axis can be computed at a later point, and for any required part of the shape.

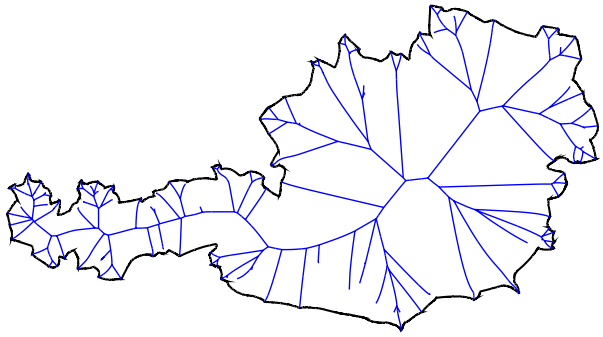
Algorithm 4 `medialaxis`($\mathcal{A}$)
{Compute the medial axis of $\mathcal{A}$}

```
1:  if A is base case then
2:     compute medial axis of A
3:  else
4:     a ← random arc in ∂A
5:     D ← maximaldisk(a, ∂A)
6:     if D is non-reducing at a' then
7:        D ← maximaldisk*(a, a', ∂A)
8:     end if
9:     k ← # tangent points on D
10:    split A into A_1, ..., A_k
11:    for i = 1 ... k do
12:       medialaxis(A_i)
13:    end for
14: end if
```





(a) Non-smooth lion shape



(b) Non-smooth Austria shape

Figure 10: Two shapes whose boundaries are not entirely smooth, but have some convex and reflex corners. The medial axis reaches the boundary at the convex vertices.

4 Details and examples

4.1 Implementation with CGAL

The algorithm presented in the previous section has been implemented in C++ for matters of performance and availability of supporting libraries. As many geometrical constructions and checks are necessary during the course of the algorithm, the Computational Geometry Algorithms Library (CGAL) [9] proved to be the most appropriate choice. CGAL is a C++ pack-

[5] As discussed in [1] it is possible to achieve a more balanced decomposition of the shape (and thus a stronger theoretical bound on the runtime) by using the so-called cut and walk principles for the determination of a dividing disk. For multi-processor architectures and parallel processing this might prove useful, on single CPU architectures, however, using solely random choices has turned out to be more efficient [2].

age for combinatorial, algorithmic, and geometrical solutions with an emphasis on flexibility, stability, exactness, and performance. It provides simple geometric calculations as intersection, position, and distance checks and also supports the visual output with simple GUIs and visualization libraries as Qt [30].

The main benefit of CGAL is, however, the possibility to choose between various number types which satisfy the demanded requirements, and which may be varied with minimal effort due to CGAL's template architecture. The implementation of the medial axis algorithm has been realized in two different versions:

- To achieve an implementation as reliable as possible, the exact rational number type $Gmpq$ from the GNU Multiple Precision Arithmetic Library [21] has been chosen in one version. The main reason for this decision is the representation of a circle as a quadratic equation in CGAL. An arbitrary point on a circle is a solution of this equation, and thus has irrational coordinates, in general. As float numbers then are necessarily imprecise, we seek rational points which exactly lie on a circle defined by three rational points. It is known that such a circle has the following properties:

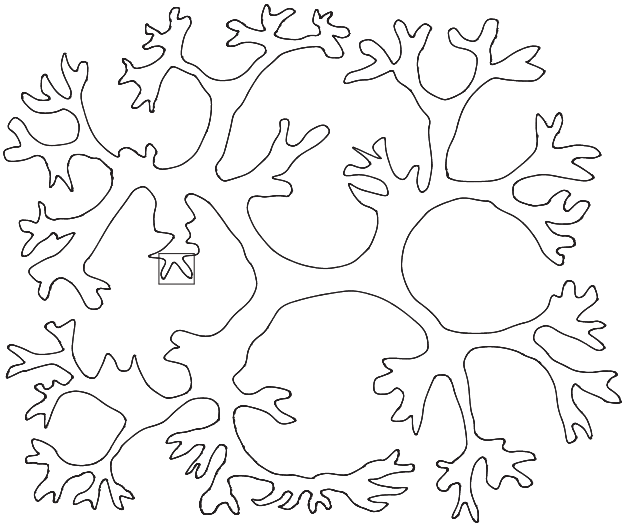
  - The center of the circle has rational coordinates.
  - Points with rational coordinates lie dense on the circle.

  So it is possible to find a rational point as near to any point on a circle as desired. This has been implemented in our program following the instructions from [7]. Due to the very large integers needed in these calculations, the choice of an elaborate rational type as $Gmpq$ is inevitable for a reliable implementation.

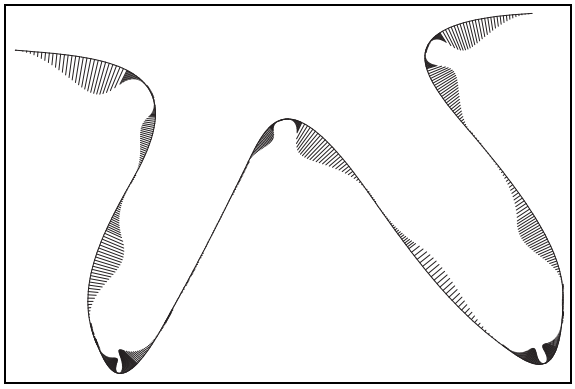
- If exactness is not the main issue (and, as observed in practical tests, the results do often not decisively differ) then the use of a float number type results in faster runtimes. A version of the program which uses explicitely $double$ numbers has been implemented for this purpose. As the statistical evaluation below will show, the gain in runtime is considerable, but computational inaccuracy may possibly result in incorrect (though locally restricted) partial solutions.

Problems with the $double$ implementation arise especially when dealing with very large circles, which result from three almost collinear points defining an arc. Lines which are nearly parallel also raise a problem, as the resulting intersection point can often not be properly represented by a float type. If such situations occur, the $double$ implementation reaches its limitations, and locally incorrect sets of maximal disks are the outcome.

The ideal solution for this problem would be an algorithm which computes the medial axis with one of both number types, dependent on which one is needed. In fact, almost all calculations can be handled by $double$ without difficulties. Only in case of an error, an exact number type, as $Gmpq$, should recompute the relevant (and by use of our approach, locally restricted) part of the shape, and provide the correct result. The main problems



(a) Initial Bézier curve (snow flake shape)



(b) Approximation details with error magnified by 10

Figure 11: Bézier curve and its biarc approximation.

in this context are, on one hand, the parallel handling of two separate kernels (what is hopefully only a matter of clever implementation), and on the other hand, the recognition of a potential error as soon as it occurs (what may be the more challenging issue). Efforts in this direction are among the motivations for future work.

The described algorithms offer several features for the manipulation of both the input and the output. Using some of them is occasionally necessary to generate appropriate data, others can be seen as a possibility to experiment with the problem:

- The algorithm for the computation of spiral biarc approximations offers a convenient possibility to vary the parameter ε that bounds the allowed Hausdorff distance between the original shape and its circular boundary representation.

- As the approximating boundary is a collection of arcs and line segments, and does not consist of one single differentiable function, it does make sense to take a closer look at the connecting vertices between two arcs. The spiral biarcs approximation generally assures a smooth boundary, but as

the representation is not totally exact, it makes sense to introduce a small error constant. Via this constant it is decided whether a vertex defines a (convex or reflex) corner of the shape, or if the shape is considered smooth in the neighborhood of this vertex. The constant can be varied to fit the quality of the used input data.

- The output of the computed circular boundary representation and its medial axis is realized in two different ways. On one hand, the popular Qt library from Trolltech [30] is used for the visualization on screen, supporting various functions as translation and zoom. On the other hand, it is possible to write the obtained medial axis directly to PostScript, where the conic arcs are represented either simply by line segments or by cubic Bézier curves; cf. footnote [6].

- There exist various other possible modifications to tune the input or the ouput, as for example the possibility to convert the input arcs into x-monotone arcs before processing, or the use of a bound flag for the arc's radii which causes arcs defined by almost collinear points to be recognized as line semgents (which makes sense to avoid numerical errors especially when working with the $double$ kernel).

4.2 Examples

In this section we report on the experimental behavior of our algorithms, and display and interpret the produced output for selected examples. We start with commenting on the biarcs approximation algorithm.

Depending on the number of spiral biarcs used to represent a shape boundary, the error between the original shape and its approximation varies. This deviation from the original shape is not uniformly distributed along the boundary, as can be seen in Fig. 11b. For simple and smooth shapes these errors are expected to be rather small, even for a small number of approximating arcs.

The relation between the achieved accuracy and the data volume is visualized in Fig. 12 for several example shapes: CAD, Austria, tree, and snow flake shape. The slopes of the shown graphs confirm an approximation order of three, as is theoretically provable for circular splines.

The size of the error, however, does not affect the number of leaves of the resulting medial axis. This is not true for polygonal representations, no matter how small is the deviation of the polygon from the approximated shape. As can be seen in Fig. 13a, many additional branches show up in the medial axis, which do not appear in the original shape's axis, nor in the axis of its spiral biarcs approximation (Fig. 13b).

With growing approximation quality of the boundary, the computed medial axis converges to the exact axis of the original shape. To rate the influence of the approximation accuracy on the speed of the implementation, several different boundary representations of a particular shape (the tree shape in Fig. 14) have been generated. The resulting runtimes are shown in detail in Table 1. (The calculation of the coefficients for the equations of the conics building the medial axis is included.[6]) The amount of elapsed seconds grows in an almost linear fashion with regard

[6]These coefficients can be stored and assigned to the respective base cases.

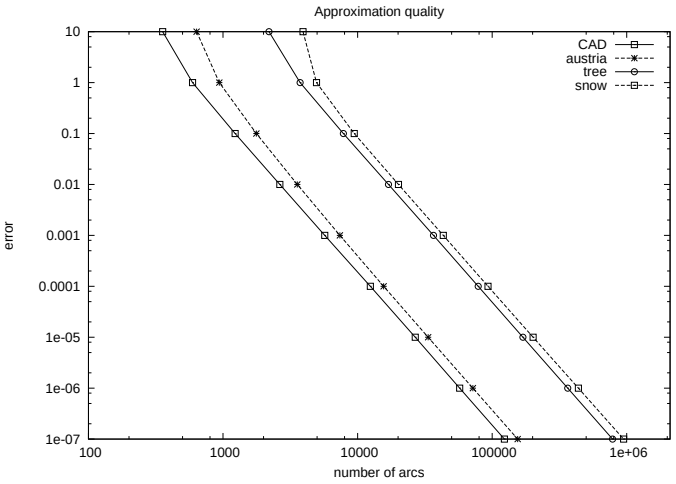
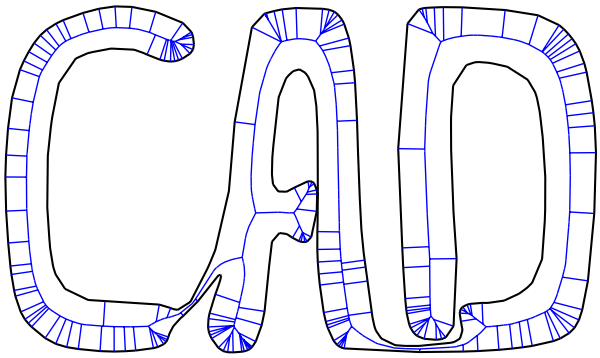
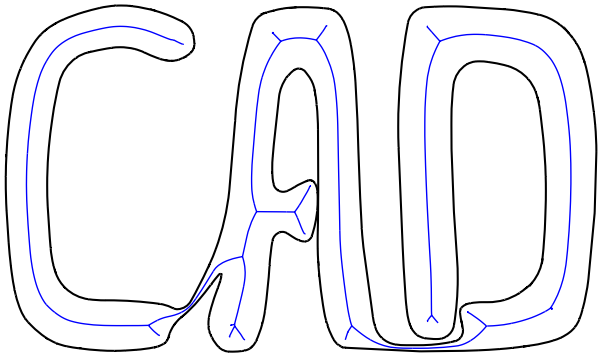


Figure 12: Relation between accuracy and data volume.



(a) Polygon approximation



(b) Spiral biarcs approximation

Figure 13: The calculation of the CAD shape was done with 292 primitives in both cases. The medial axis of the polygonal approximation has additional branches at the convex corners, while the medial axis of the circular boundary representation is topologically correct and geometrically more accurate.

For the output to PostScript it has proven more useful to choose cubic Bézier curves that approximate the conic arcs.

error	arcs	∂ fl	MA fl	b fl	MA Gmpq
$k \cdot 10^{-1}$	2096	0.06	0.18	0.03	12.15
$k \cdot 10^{-2}$	3736	0.14	0.36	0.04	19.31
$k \cdot 10^{-3}$	7840	0.32	0.67	0.08	42.54
$k \cdot 10^{-4}$	16970	0.75	1.53	0.12	90.55
$k \cdot 10^{-5}$	36674	1.65	3.45	0.24	194.43
$k \cdot 10^{-6}$	78736	3.59	7.21	0.58	427.36
$k \cdot 10^{-7}$	169418	7.76	16.25	1.23	918.35
$k \cdot 10^{-8}$	364528	16.91	36.49	2.81	2169.5
$k \cdot 10^{-9}$	784972	36.76	83.04	5.89	4607.18

Table 1: Runtimes in seconds for different approximations of the shape in Fig. 14. The column ∂ fl shows the time needed for the boundary conversion with an error relative to a bounding box parameter k. The two MA columns give the seconds elapsed for the medial axis construction using $double$ and $Gmpq$, respectively. (We have averaged over 5 runs). The time needed for the base cases (column b fl) is already included.

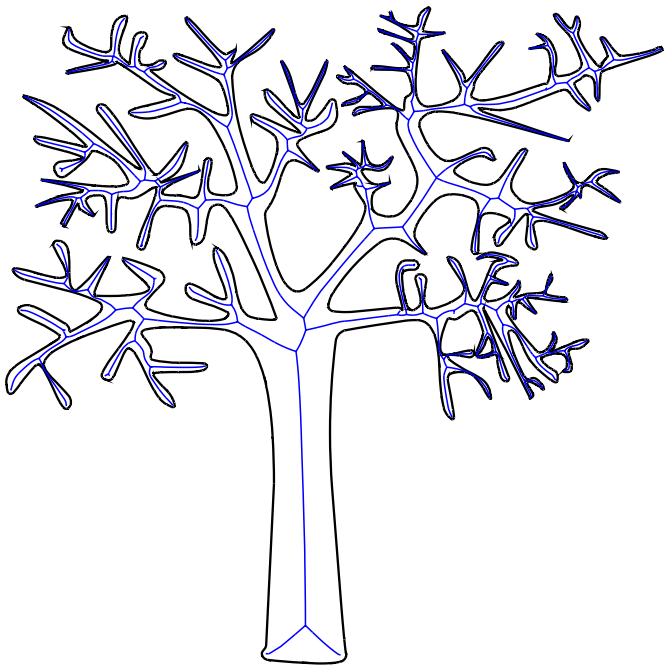


Figure 14: Tree shape and its medial axis.

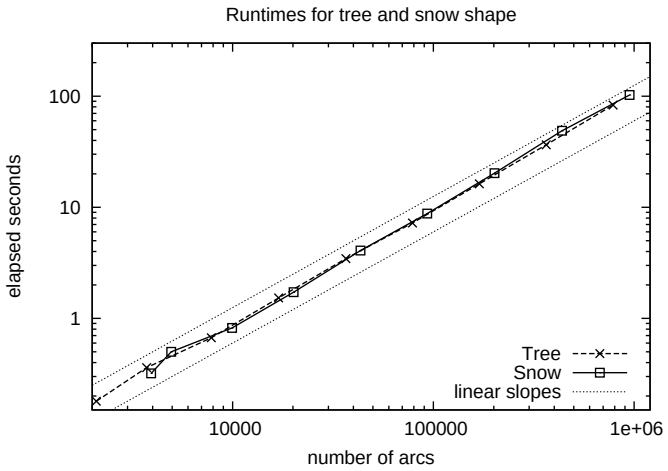


Figure 15: Runtimes for snowflake (Fig. 11a) and tree (Fig. 14) shape. The two dotted lines show linear reference functions for hypothetical runtimes of 125 μs per arc (upper) and 60 μs per arc (lower).

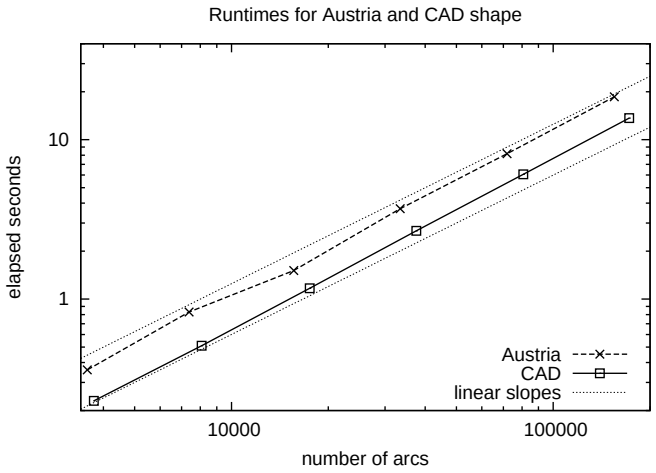


Figure 16: Runtimes for Austria shape (Fig. 10b) and CAD lettering (Fig. 13b). The two dotted lines show linear reference functions for hypothetical runtimes of 125 μs per arc (upper) and 60 μs per arc (lower).

to the number of arcs. Note that, by construction, the number of branching points of the medial axis stays the same for all approximations, because the number of leaves is the same as for the original free-form shape.

In Fig. 15 and Fig. 16, for several shapes the ratio between the computation time and the number of arcs is displayed graphically. Note that the coordinate axes of the graphs are logarithmically scaled.

The graphs in both figures show that in practice runtimes grow (almost) linear with the number of arcs used for the approximation. The snowflake shape (Fig. 11a) and the tree shape (Fig. 14) evaluated in Fig. 15 branch similarly, so the resulting runtimes are almost the same.

Out of the two shapes interpreted in Fig. 16, the medial axis of the Austria shape (Fig. 10b) has more branching points than the CAD lettering (Fig. 13b). This results in a better relative runtime for the latter shape, shown as offset between the two graphs in log-log scale.

The configuration used for all tests is a 64 bit installation of Linux Debian on an Intel Core 2 Duo 6700 architecture with 8 GB RAM. As no parallel processing is implemented yet, only one core is used so far.

5 Conclusion

We have provided an efficient and stable implementation of a medial axis algorithm for planar free-form shapes. To our knowledge, this is the first algorithm that runs fast in practice and at the same time guarantees convergence to the exact medial axis of the input shape. The program can compete with current state-of-the-art implementations on this field with regard to correctness, speed, and reliability. The basic idea was using a piecewise circular boundary conversion, which allows for appropriate feature preservation of the shape, as well as for a simple and fast medial axis algorithm. An implementation of the algorithm which combines the $double$ and the $Gmpq$ kernels to achieve speed and stability in one single program is possibly an issue for future work. In addition, we plan to the extend the method to multiply connected domains and we will apply the results to obtain efficient methods for the computation of offset curves.

Acknowledgements

The authors were support by the Austrian Science Fund (FWF) through the National Research Network S92 "Industrial Geometry", subprojects 2 and 5. We also thank the anonymous referees for their helpful comments, which improved the presentation of the paper.

References

[1] Aichholzer O, Aurenhammer F, Hackl T, Jüttler B, Oberneder M and Šír Z. Computational and structural advantages of circular boundary representation. In: Dehne F, Sack JR, Zeh N, editors. Algorithms and Data Structures. Springer LNCS; 2007. p. 374–85.

[2] Aigner W. The medial axis of planar shapes. Master Thesis, Institute for Theoretical Computer Science, University of Technology, Graz, Austria 2007.

[3] Alt H, Cheong O, Vigneron A. The Voronoi diagram of curved objects. Discrete & Computational Geometry 2005; 34: 439–53.

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

[5] Blum H. A transformation for extracting new descriptors of shape. In: Wathen-Dunn W, editor. Models for the Perception of Speech and Visual form. MIT Press; 1967. p. 362–80.

[6] Brandt JW, Algazi VR. Continuous skeleton computation by Voronoi diagram. CVGIP: Image Understanding 1992; 55: 329–38.

[7] Burnikel C. Rational points on circles. Research Report MPI-I-98-1-023, Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany. 1998

[8] Cao L and Liu J. Computation of medial axis and offset curves of curved boundaries in planar domain. Computer-Aided Design 2008; in press.

[9] CGAL. Computational Geometry Algorithms Library. http://www.cgal.org/.

[10] Chin F, Snoeyink J, Wang CA. Finding the medial axis of a simple polygon in linear time. Discrete & Computational Geometry 1999; 21: 405–20.

[11] Choi HI, Choi SW and Moon HP. Mathematical theory of medial axis transform. Pacific Journal of Mathematics 1997; 181: 57–88.

[12] Choi HI, Choi SW and Moon HP and Wee NS. New algorithm for medial axis transform of plane domain. Graphical Models and Image Processing 1997; 59: 463–83.

[13] Choi HI and Han CY. The Medial Axis Transform. In: Farin G., Hoschek J, Kim MS, editors. The Handbook of Computer Aided Geometric Design. Amsterdam: North-Holland; 2002. p. 451–71.

[14] Chou JJ. Voronoi diagrams for planar shapes. IEEE Computer Graphics and Applications 1995; 15; 52–9.

[15] Degen WLF. Exploiting curvatures to compute the medial axis for domains with smooth boundary. Computer Aided Geometric Design 2004; 21: 641–60.

[16] Elber G, Cohen E and Drake S. MATHSM: medial axis transform toward high speed machining of pockets. Computer-Aided Design 2005; 37: 241–50.

[17] Elber G and Kim MS. Bisector curves of planar rational curves. Computer-Aided Design 1998; 30: 1089–96.

[18] Evans G, Middleditch AE, Miles N. Stable computation of the 2D medial axis transform. Int. J. Computational Geometry & Applications 1998; 8: 577-98.

[19] Farouki RT and Ramamurthy R. Degenerate point/curve and curve/curve bisectors arising in medial axis computations for planar domains with curved boundaries. Computer Aided Geometric Design 1998; 15: 615–35.

[20] Gisch D and Ribando JM. Apollonius' problem: A study of solutions and their connections. 2004

[21] GMP. GNU Multiple Precision Arithmetic Library. http://gmplib.org/.

[22] Held M. VRONI: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments. Computational Geometry 2001; 18: 95–123.

[23] Held M and Eibl J. Biarc approximation of polygons within asymmetric tolerance bands. Computer-Aided Design 2005; 37: 357–71.

[24] Jin L, Kim D, Mu L, Kim D-S and Hu S-M. A sweepline algorithm for Euclidean Voronoi diagram of circles. Computer-Aided Design 2006; 38: 260–72.

[25] Kim D-S, Hwang I-K and Park B-J. Representing the Voronoi diagram of a simple polygon using rational quadratic Bézier curves. Computer-Aided Design 1995; 27: 605–14.

[26] Lee DT. Medial axis transformation of a planar shape. IEEE Pattern Analysis and Machine Intelligence 1982; 4: 363–9.

[27] Meek DS and Walton DJ. Approximating smooth planar curves by arc splines. Journal of Computational and Applied Mathematics 1995; 59: 221–31.

[28] Meek DS and Walton DJ. Spiral arc spline approximation to a planar spiral. Journal of Computational and Applied Mathematics 1999; 107: 21–30.

[29] Pottmann H and Peternell M. Applications of Laguerre geometry in CAGD. Computer Aided Geometric Design 1998; 15: 165–86.

[30] Qt. Qt Cross-Platform Application Framework. http://trolltech.com/products/qt/.

[31] Ramamurthy R and Farouki RT. Voronoi diagram and medial axis algorithm for planar domains with curved boundaries II: detailed algorithm description. Journal of Computational and Applied Mathematics 1999; 102: 253–77.

[32] Ramanathan M and Gurumoorthy B. Constructing medial axis transform of planar domains with curved boundaries. Computer-Aided Design 2003; 35: 619–32.

[33] Šír Z, Feichtinger R and Jüttler B. Approximating curves and their offsets using biarcs and Pythagorean hodograph quintics. Computer-Aided Design 2006; 38: 608–18.

[34] Yap CK. An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments. Discrete & Computational Geometry 1987; 2: 365–93.