International Journal of Computational Geometry & Applications

Mitered Offsets and Skeletons for Circular Arc Polygons

Bastian Weiß, Bert Jüttler

Institute of Applied Geometry, Johannes Kepler University, Altenberger Str. 69
Linz, 4040, Austria
bert.juettler@jku.at

Franz Aurenhammer

Institute of Theoretical Computer Science, Graz University of Technology
Inffeldgasse 16b, Graz, 8010, Austria
auren@igi.tugraz.at

Received (received date)
Revised (revised date)
Communicated by (Name)

ABSTRACT

The offsetting process that defines straight skeletons of polygons is generalized to arc polygons, i.e., to planar shapes with piecewise circular boundaries. The offsets are obtained by shrinking or expanding the circular arcs on the boundary in a co-circular manner, and tracing the paths of their endpoints. These paths define the associated shape-preserving skeleton, which decomposes the input object into patches. While the skeleton forms a forest of trees, the patches of the decomposition have a radial monotonicity property. Analyzing the events that occur during the offsetting process is non-trivial; the boundary of the offsetting object may get into self-contact and may even splice. This leads us to an event-driven algorithm for offset and skeleton computation. Several examples (both manually created ones and approximations of planar free-form shapes by arc spline curves) are analyzed to study the practical performance of our algorithm.

Keywords: Mitered offset; skeletal structure; medial axis; biarc

1. Introduction

Offsets and skeletons of planar shapes are widely used in shape analysis, shape design, motion planning, image processing, and tool path generation. Besides classical offsets, various generalizations have been considered.

The computation and representation of untrimmed offsets of planar shapes is well understood, see Refs. 1, 2 and the references cited therein. Trimming, i.e., the removal of the offset curves' self-intersections, has attracted substantial attention. A simple method for detecting and removing invalid loops of planar offsets is described by Lai et al.[3] Trimming of local and global self-intersections of offsets based on distance maps is discussed by Seong et al.[4] Circular arc-based auxiliary structures are employed in Refs. 5, 6. The extension of trimming techniques to offset curves on surfaces (represented as meshes) is described by Xu et al.[7]

Singularities and self-intersections of offset curves are closely related to the *medial axis*[8] which is a particular skeleton, i.e., a structural shape descriptor. Algorithms for computing the medial axes of planar shapes are studied in several publications. Lee et al.[9] develop a divide-and-conquer algorithm for computing the medial axis of a simple polygon in $\mathcal{O}(n \log n)$ time by exploiting its Voronoi diagram. The medial axis can be approximated with the help of the Voronoi diagram of point samples.[10] A spiral-preserving circular spline approximation[11] is used by Aichholzer et al.[12,13] to derive algorithms for the computation of medial axes and trimmed offsets of planar shapes. Spline approximations of the medial axis are described in Ref. 14. Exact medial axis computation is greatly simplified when considering piecewise linear metrics. Such metrics can be exploited to compute the exact medial axis of triangulated solids.[15] For a general account of properties and algorithms concerning the medial axis, we refer the reader to Refs. 16, 17.

In the case of piecewise linear shapes, *mitered offsets* provide an alternative to classical offsets with enhanced shape-preserving properties around reflex vertices.[18] They are defined procedurally, by specifying the evolution of the boundary of a shape as the offset distance increases. Special attention has to be paid to topological changes of the offsets, which can be classified into so-called events.

Aichholzer et al.[19,20] use this evolution to define the *straight skeleton* of simple polygons and planar straight line graphs. Mitered offsets and straight skeletons for non-uniform speed offsetting are analyzed in Ref. 21. The approximate medial axis can be computed by using straight skeletons.[22] The recent extension of straight skeletons to three-dimensional polytopes is highly nontrivial.[23]

In this paper, we propose a generalization of mitered offsets and straight skeletons to *arc polygons*, i.e., to simply connected planar shapes with a boundary defined by circular arcs. The mitered offsets are again defined by specifying the evolution of the boundary, where the use of arcs leads to a wider variety of events.

Figure 1 compares classical offsets (top) and mitered offsets (bottom) of a heart-shaped curve composed of circular arcs (left) and line segments (right), respectively. The offsets differ around the reflex vertex, where the mitered offsets exhibit better

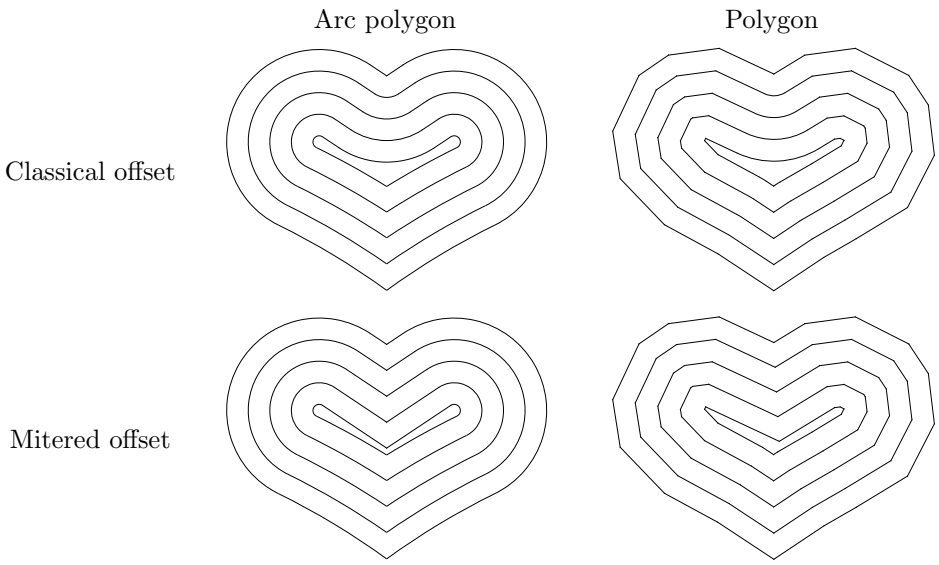


Fig. 1: Comparison of classical and mitered offsets

shape-preserving properties. Clearly, the use of a piecewise linear representation (i.e. a polygon, see right column) requires a much larger number of geometric primitives to achieve a similar accuracy.

The remainder of the paper consists of six sections. Sections 2 and 3 define and analyze mitered offsets, the resulting skeleton, and associated patch decomposition of the input shape. Section 4 is dedicated to the problem of computing the patch decomposition. We introduce necessary data structures and show how to detect and handle resulting events. Experimental results and examples are presented in Section 5. Finally we conclude the paper in Section 6.

2. Mitered offsets

For our input objects, we consider simply connected planar shapes represented as *arc polygons*, i.e., simple closed curves formed by circular arcs (or straight line segments) joined together. Objects of more general shape can be converted into arc polygons easily and efficiently.[24]

We define mitered offsets, and subsequently the associated skeleton, as the result of an *evolution* of a circular arc polygon P that represents the boundary of a given shape. Offsetting P means that the edges (which can be circular arcs or line segments) shrink or expand in radial direction (parallel for straight line segments) with constant speed towards the interior of the polygon. Simultaneously, the edges' endpoints travel on certain paths (more precisely, on arcs of conic sections), which

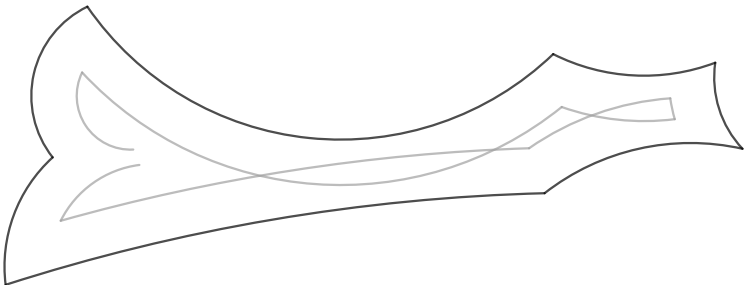


Fig. 2: Untrimmed offset of an arc polygon.

are determined by the evolution of the adjacent edges[a]. This process is well-defined until we encounter (self-)intersections or the boundary becomes disconnected, see Fig. 2. The latter situation appears when the evolution reaches a point where two adjacent arcs possess anti-parallel tangents at the common endpoint. We introduce *events* in order to obtain a globally consistent definition of trimmed offsets. More precisely, we distinguish between four classes of events.

Vanish event One edge e of P vanishes. This happens if the two endpoint vertices of e become coincident and the length of e shrinks to zero, see Fig. 3.

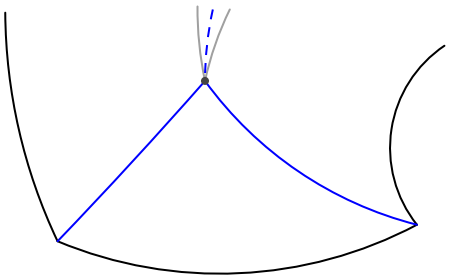


Fig. 3: Vanish event. The gray curve is the offset and the blue lines form the skeleton.

Terminal event Such an event occurs if a connected component of P collapses. One, two, or three edges of P then disappear simultaneously; see Fig. 4.

Contact event The boundary of P gets into self-contact. There are three events of this type: A *split event* (Fig. 5a) occurs when a reflex vertex hits an edge of P. Two edges that touch in their interiors create a *squeeze event*

[a]We choose the normal if two adjacent edges are segments of the same circle or line.

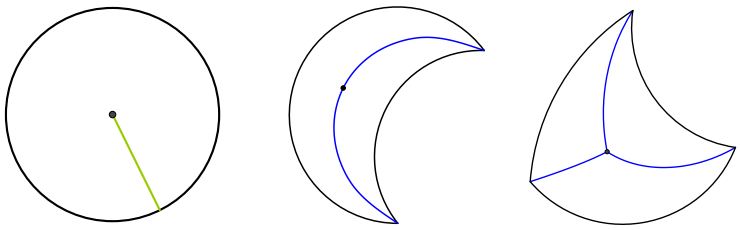
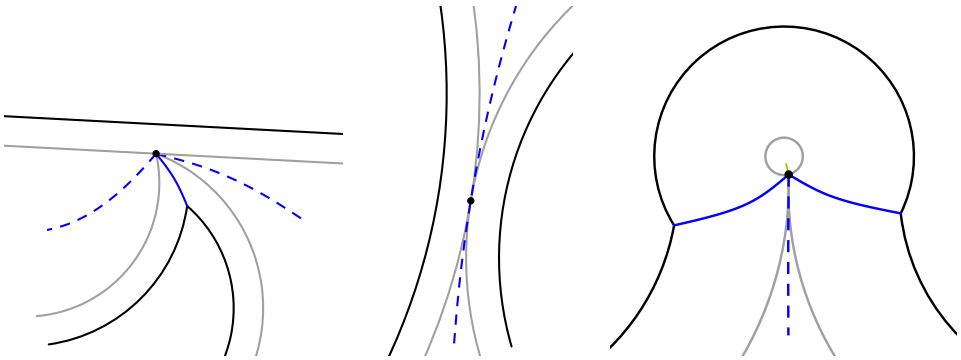


Fig. 4: There are three possible terminal events.

(Fig. 5b). Finally, a *bubble event* (Fig. 5c) takes place when the endpoints of a shrinking edge meet but that edge does not vanish (but rather bends to a full circle).



(a) Split event (b) Squeeze event (c) Bubble event

Fig. 5: The three types of contact events.

Vanish events and contact events, respectively, generalize the *edge event* and the *split event* that have been introduced when defining straight skeletons of simple polygons.[20]

Two adjacent edges of P can lose contact during the evolution process, see Fig. 2. Since this situation was not encountered for classical straight skeletons, we need to introduce a new type of event.

Splice event A reflex vertex v of P splices at the moment when its two incident edges become tangential (Fig. 6). The continuation of the offsetting process is not unique. We propose to close the gap in the boundary of P by inserting a semicircle that starts to expand at v.

Depending on practical requirements, we may generalize the definition of a splice

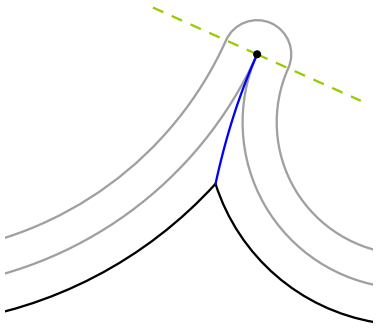


Fig. 6: Splice event. The gray curve is the offset, the solid blue line forms the skeleton, and the green lines are paths of smooth vertices.

event in the following way: We introduce σ as the exterior angle at which a reflex vertex v is assumed to splice. The resulting *σ-splice event* is identical with the just introduced splice event for $\sigma = 2\pi$. An angle of $\pi \leq \sigma < 2\pi$ forces v to splice earlier. In particular, setting $\sigma = \pi$ causes an immediate splice of v, which results in the classical offset at v defined by the medial axis of P.

At this point, let us mention that—in contrast to classical offsets—mitered offsets possess a *local invertibility property*. Namely, for any circular arc polygon P, there exists a threshold $\varepsilon > 0$ such that the following holds: If we offset P by a distance $\delta \leq \varepsilon$, and then offset the resulting arc polygon by a distance $-\delta$, then P itself is obtained again; see Fig. 7. The value of ε is determined by the first vanish event. This property does not hold for classical offsets, since the convex vertices of P will not be preserved.

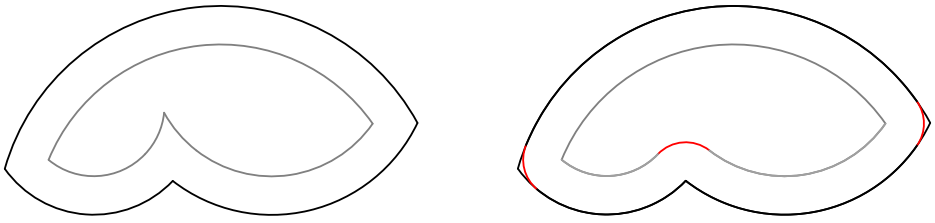


Fig. 7: Mitered offsets (left) possess the local invertibility property, while classical offsets (right) do not.

Most of the events caused by arc polygons are of local nature, since they involve only a small number of adjacent edges. Exceptions are the split event and the squeeze event, which entail global modifications of the polygon's topology. Consequently, we will distinguish between *local* events and *global* events later on.

3. The skeleton

In this section, we will analyse the basic geometric and combinatorial properties of skeletons defined by mitered offsets of arc polygons.

Vertex paths

Let our circular arc polygon P consist of N *vertices* v_i (with indices modulo N) connected by *edges* $e_i = v_i v_{i+1}$ in counterclockwise order. Each edge is a portion of a circle or a straight line, and has a *center* m_i (possibly at infinity) and a *radius* r_i (non-zero, possibly infinite). Positive and negative values of the radius correspond to circular arcs with counterclockwise and clockwise orientation, respectively. In the offsetting process for P, centers and signs of radii stay fixed, whereas radii values are changing.

The path that a vertex v_i traces during the evolution of P is a conic section and is determined by its incident edges. Vertices with interior angle π are called *smooth*. Note that all edges can be straight line segments, hence we include straight skeletons[20] as a special case. More precisely we have the following result:

Lemma 1. *The path of a vertex v_i is a linear, elliptic, hyperbolic, or parabolic arc, according to the classification in Table 1.*

Table 1: Different paths of vertices.

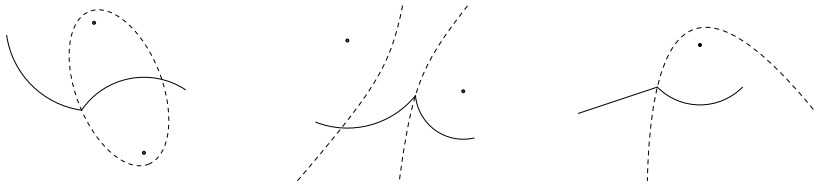
condition	conic	foci
$r_i r_{i-1} < 0$	elliptic	m_i, m_{i-1}
$r_i r_{i-1} > 0$ and $r_i \neq r_{i-1}$	hyperbolic	m_i, m_{i-1}
$r_i = r_{i-1}$ or v_i is smooth	straight line	-
$r_i = \infty$, $r_{i-1} \neq \infty$	parabola	m_{i-1}
$r_i \neq \infty$, $r_{i-1} = \infty$	parabola	m_i

Proof. The type of the path of v_i is determined by its incident edges e_{i-1} and e_i. First we assume that their radii r_{i-1} and r_i are finite.

On the one hand, the vertex $v_i(t+\delta)$ at time $t+\delta$ satisfies the equation

$$\|v_i(t+\delta) - m_i\| + \|v_i(t+\delta) - m_{i-1}\| = |r_i - \delta| + |r_{i-1} - \delta|.$$

The right-hand side is constant if r_i and r_{i-1} have opposite signs, hence the vertex travels on an ellipse with foci m_i and m_{i-1} in this case, see Fig. 8a. The ellipse degenerates into a line segment if $v_i(t)$ is located on the line segment that connects m_i with m_{i-1} which implies that $v_i(t)$ is smooth.



(a) elliptic (b) hyperbolic (c) parabolic

Fig. 8: Possible paths of a vertex.

On the other hand, the vertex $v_i(t+\delta)$ also satisfies the equation

$$\|v_i(t+\delta) - m_i\| - \|v_i(t+\delta) - m_{i-1}\| = |r_i - \delta| - |r_{i-1} - \delta|.$$

The right-hand side is constant if r_i and r_{i-1} have identical signs, hence the vertex then travels on a hyperbola with foci m_i and m_{i-1}, see Fig. 8b. There are two cases in which this hyperbola degenerates into a straight line: First, if $r_i = r_{i-1}$ holds. Second, if $v_i(t)$ is located on the line defined by m_i and m_{i-1} but not on the corresponding segment. Again, the second case implies that $v_i(t)$ is a smooth vertex.

Extending this analysis to the cases of one or two arcs with infinite radii (i.e., straight lines) confirms that the path becomes a parabola or a straight line in this situation. In the first case, the center of the curved edge serves as focus, while the directrix is parallel to the edge which is straight, see Fig. 8c. Note that the parabola degenerates into a straight line when $v_i(t)$, m_{i-1}, and m_i are collinear (one of the neighbors centers is a far point). In this case $v_i(t)$ is a smooth vertex. □

Definition 1. The *skeleton*, $\mathcal{S}(P)$, of a circular arc polygon P is formed by the paths of all non-smooth vertices under the mitered offsetting process for P.

Figure 9 shows a complex circular arc polygon P, its various mitered offsets, and its skeleton $\mathcal{S}(P)$. Note that the skeleton consists of three connected components, two of them being just single hyperbolic arcs emerging from reflex vertices of P.

Properties of the skeleton

We will show that the skeleton $\mathcal{S}(P)$ is a *forest* (Lemma 4) with at most $r+1$ connected components, where r is the number of reflex vertices of P. Thus this structure is slightly more complex than the straight skeletons of polygons,[19] which is known to be a tree.

In addition, the size of the skeleton will be shown to be linear in the size of P. While the latter property is rather obvious for straight skeletons of polygons, the

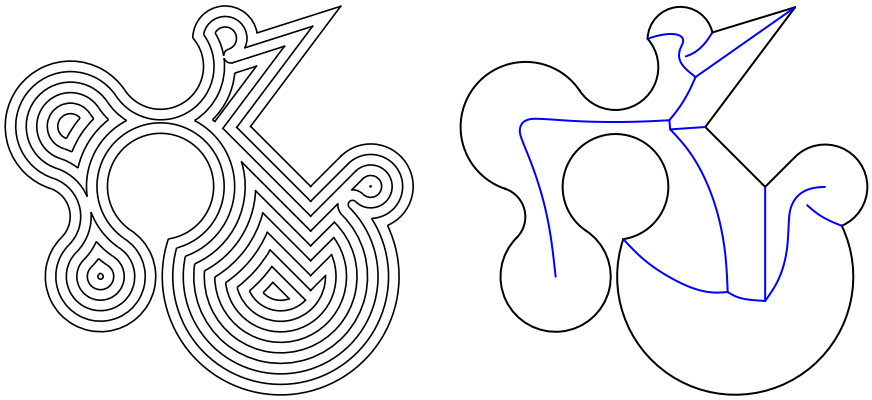


Fig. 9: Mitered offsets (left) and skeleton (right).

analysis is more involved for circular arc polygons, due to the presence of additional events. We start with stating a technical lemma.

Lemma 2. *Neither the evolution process between events, nor the individual events themselves, increase the number of reflex vertices of the offsetting arc polygon.*

Proof. First, we consider the evolution process without the occurrence of any events, which does not create any new vertices. Since angles at vertices change continuously, the transformation from a convex vertex to a reflex vertex would need to pass through a smooth one. However, smooth vertices are preserved by the evolution, as is reflected in Fig. 10.

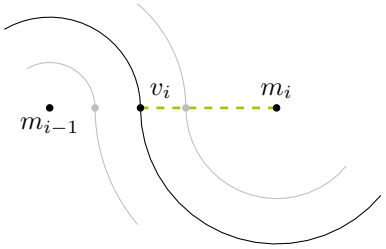


Fig. 10: Smooth vertices move on the line between centers.

Second, we analyze the various events and show that they do not increase the number of reflex vertices. Splice events create two smooth vertices, squeeze events create two zero angle vertices, thus no reflex ones. Any terminal event destroys the respective component of the polygon.

Let us argue that bubble events also do not create reflex vertices. Indeed, any

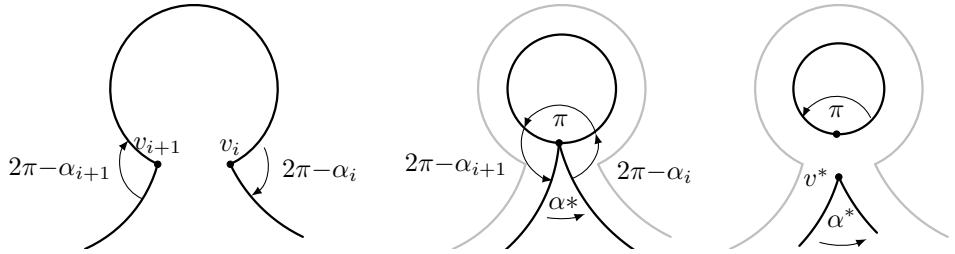
two adjacent interior vertex angles satisfy $\alpha_i + \alpha_{i+1} < 4\pi$. We consider the edge e_i that turns into a full circle, thereby creating a smooth vertex. At the time of the event, the other newly created vertex angle α^* satisfies

$$2\pi = \alpha^* + (2\pi - \alpha_i) + \pi + (2\pi - \alpha_{i+1}),$$

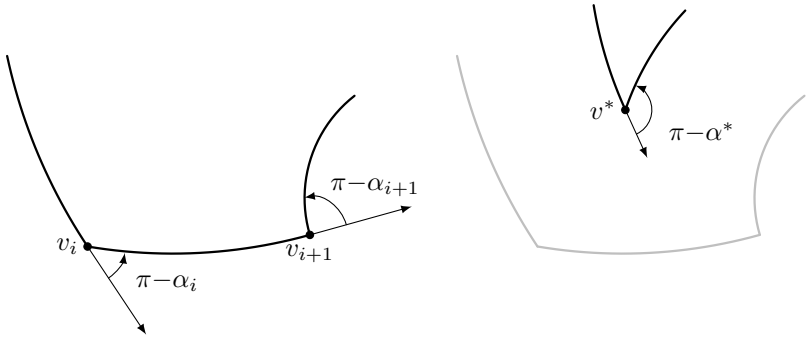
see Fig. 11a, thus

$$\alpha^* = \alpha_i + \alpha_{i+1} - 3\pi < \pi.$$

The analysis of split events is analogous. Finally we consider vanish events. These



(a) A bubble event (center) and the arc polygon shortly before (left) and after (right) the event.



(b) A vanish event (right) and the arc polygon shortly before the event (left).

Fig. 11: Bubble events (a) and vanish events (b) do not create reflex vertices.

events can only occur if $\alpha_i + \alpha_{i+1} < 2\pi$. At the time of the event, the newly created exterior angle $\pi - \alpha^*$ of the arc polygon satisfies

$$\pi - \alpha^* = (\pi - \alpha_i) + (\pi - \alpha_{i+1}),$$

see Fig. 11b. Consequently, α^* does not achieve π. □

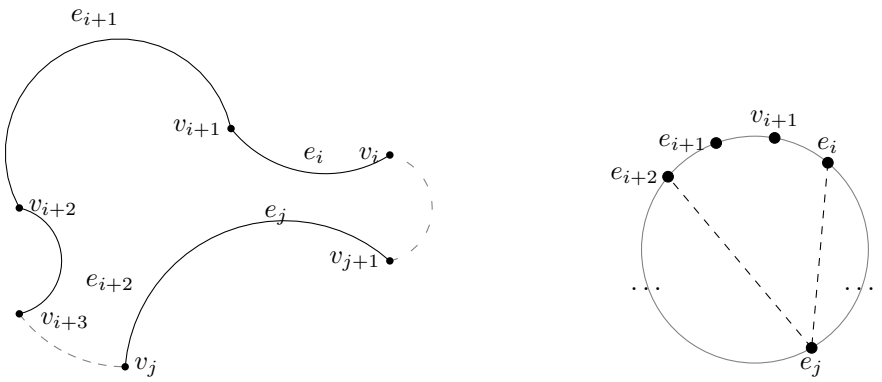


Fig. 12: Schematic representation of two squeeze events.

Remark 1. The proof above also shows that reflex vertices disappear whenever they are involved in an event. For instance, even if one of the vertices in a vanish event is reflex, the newly created vertex is always convex.

Lemma 3. *The skeleton $\mathcal{S}(P)$ of a circular arc polygon P has linear complexity.*

Proof. We observe first that an event can create at most two new edges and one new vertex of $\mathcal{S}(P)$. Consequently, it suffices to show that the number of events is linear.

Let P have N edges and r reflex vertices. Splice events, split events, and bubble events emerge from reflex vertices of P. The associated reflex vertex gets eliminated in such an event, and no new reflex vertices are ever created, by Lemma 2. This implies

$$|\text{splice events}| + |\text{split events}| + |\text{bubble events}| \leq r.$$

Next we analyze the number of squeeze events. A squeeze event involves two edges, which either are offsets of edges of P, or have been introduced when some reflex vertex of P got spliced. Let us associate the N edges of P with the corners of a regular N-gon R, in the same cyclic order. Each squeeze event then represents a diagonal of R, and these diagonals do not cross because each squeeze event splits the arc polygon into two components; see Fig. 12. Clearly, the number of such diagonals is smaller than the number of vertices of R, and we obtain

$$|\text{squeeze events}| < N.$$

Finally, let us consider the vanish and terminal events. Either type destroys at least one edge of P. On the other hand, splice and split events create one new edge, while squeeze events increase the number of edges by two. Therefore

$$|\text{vanish \& terminal events}| \leq N + |\text{splice events}| + |\text{split events}| + 2|\text{squeeze events}|.$$

Combining these results confirms that the number of events is less than $4N+2r$. In conclusion, the complexity of $\mathcal{S}(P)$ is $O(N+r) = O(N)$. □

Lemma 4. *The skeleton $\mathcal{S}(P)$ of P is a forest consisting of at most $r+1$ trees.*

Proof. We define the 'evolving skeleton', $E(t)$, at time t as the part of the skeleton generated by the paths of P's vertices for time at most t, plus the planar domains that are enclosed by the connected components of the evolving arc polygon at time t.

We argue that $E(t)$ is always simply connected. This is clearly true for $t = 0$, since P is simply connected. Moreover, none of the events creates connected components of $E(t)$ that are not simply connected: The topological effect of three of the events is shown schematically in Fig. 13, and the remaining events (squeeze, bubble, and terminal events) also do not create cycles in $E(t)$. Concerning the number of components of $E(t)$, each of the (at most r) splice events increases this number by 1, whereas all the other events keep this number constant. We complete the proof by observing that $E(t)$ starts with one component and ends up with $\mathcal{S}(P)$, which therefore is a collection of at most $r+1$ trees. □

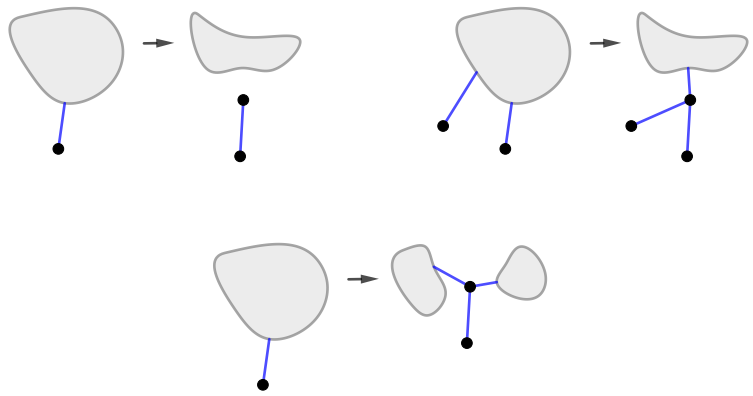


Fig. 13: How a splice event (left), a vanish event (right), and a split event (bottom) modify the topology of the evolving skeleton.

Patch decomposition and roof

The proof of Lemma 4 reflects that the connected components of $\mathcal{S}(P)$ fall into two categories: The first one is created by reflex vertices that are subject to splice events. Each such component is just a single edge of $\mathcal{S}(P)$ traced out by a reflex vertex before it gets spliced. The second category consists of the remaining parts of

$\mathcal{S}(P)$ and contains only one component, which we will call the *principal component* of $\mathcal{S}(P)$.

To make the skeleton a connected structure, we may add all the paths of the *smooth* vertices to it. A smooth vertex can either be part of the input arc polygon P already, or it can be created later in a splice event (Fig. 6). The paths of the latter vertices will connect the edges in the first component category above to the principal component of $\mathcal{S}(P)$. The resulting structure, which can contain cycles now, is termed the *extended skeleton* of P. It defines a segmentation of the arc polygon's interior into disjoint patches, which we denote as the *patch decomposition* of P.

The extended skeleton shown in Figure 14 (left) contains two paths of smooth vertices (emanating from the splicing reflex vertex) and defines a patch decomposition consisting of four patches.

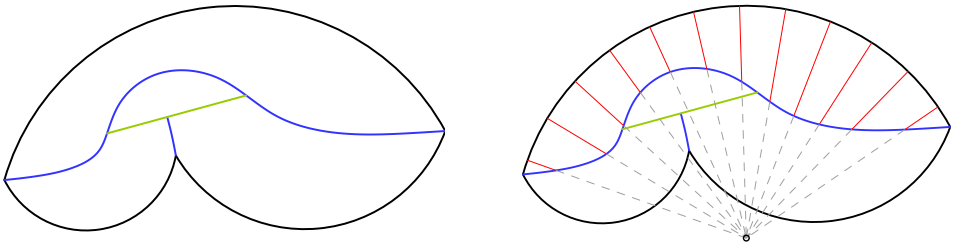


Fig. 14: Extended skeleton (left) and radial decomposition of one patch (right).

Similar to the case of planar straight skeletons[20] we consider a so-called *roof model* and discuss its link to the patch decomposition. The roof $\mathcal{R}(P)$ of P is constructed as follows: Each point (x, y) of the mitered offset of P is lifted into xyt-space by adding the time t as a third coordinate. This results into a composite surface in 3D consisting of planar facets (generated by straight edges of P) and facets of right circular cones (generated by circular edge of P), both with slope 1. Clearly, $\mathcal{R}(P)$ forms a globally continuous surface.

The roof provides the alternative definition of the skeleton, namely

$$\mathcal{S}(P) = \{(x, y) \mid (x, y, t) \in \mathcal{R}(P) \text{ is singular and not isolated}\}.$$

$\mathcal{R}(P)$ simultaneously encodes the patch decomposition of P, since its pieces when separated by the projected paths of (smooth or non-smooth) vertices, are in one-to-one correspondence to the resulting patches.

It is known that the patch decomposition induced by the classical straight skeleton possesses a monotonicity property:[20] Each patch is monotonic with respect to the direction of the defining edge. We generalize this fact to arc polygons.

Lemma 5. *Each patch g of the patch decomposition of P possesses the property of* radial monotonicity *with respect to the center c of its defining edge.*

Proof. We have to show that the intersection of g with any radial line emanating from c is either empty or a single line segment. This can be done using the roof $\mathcal{R}(P)$. Let f be the facet on $\mathcal{R}(P)$ corresponding to the patch g. Assuming that the lemma is not true, there have to exist two points p and q where some projected radial line through c leaves and re-enters f. Between p and q, the slope of the roof must be different from 1, and in order to achieve the roof height at q, this slope must be larger than 1 at some places. This is a contradiction, since the maximal slope on $\mathcal{R}(P)$ is 1. □

By the radial monotonicity property, the patches can be decomposed in a radial manner, as is depicted for the uppermost patch in Figure 14 (right).

4. Computing mitered offsets and skeletons

The skeleton $\mathcal{S}(P)$ of an arc polygon P generalizes the classical straight skeleton. It has been noted that, for the later structure, well-known algorithmic principles (such as sweep-line and divide & conquer) cannot be employed directly, due to the absence of an underlying Voronoi structure.[19] Additional effort is needed to bound the influence of reflex vertices to the straight skeleton, since their speeds (which are determined by the interior vertex angles of P) can take any positive values. In fact, the majority of known algorithms compute the straight skeleton by simulating the event-driven evolution that defines mitered offsets; see e.g.[25]

We present such an algorithm that generates the patch decomposition determined by the mitered offsets of a given arc polygon P. From this decomposition, the individual offsets can be restored in two steps: First, we compute the roof $\mathcal{R}(P)$ by projecting the patch decomposition onto the cones in space-time. Second, the intersection of $\mathcal{R}(P)$ with a horizontal plane at height δ gives the mitered offset of P at distance δ.

Algorithm

Our algorithm operates on three main data structures. The evolving circular arc polygon is represented as a doubly linked list E which maintains the current status of the evolution. The output is recorded in a DCEL (doubly-connected edge list[26]) $\mathcal{D}$ that represents the patch decomposition. Finally, the events are managed by a priority queue Q. Algorithm 1 gives an outline of our method.

The procedure *initEvolvingPolygon*(P) initializes E with alternating edges and vertices, sorted counterclockwise. Additionally, the center m_i of each edge is computed, as well as a flag for each vertex that determines whether it is smooth,

Algorithm 1: Patch Decomposition

Input: Circular arc polygon P
Output: Patch decomposition $\mathcal{D}$

```
E ← initEvolvingPolygon(P)
D ← initPatchDecomposition(P)
Q ← initLocalEventList(E)
while E ≠ ∅ do
    nextEvent ← chooseNextEvent(Top(Q), E)
    handleEvent(nextEvent, E, D, Q)
end
return D
```

convex, or reflex. We initialize $\mathcal{D}$ by calling *initPatchDecomposition*(E). This procedure copies all edges and vertices from E into the DCEL $\mathcal{D}$. Additionally, it creates an edge of the extended skeleton for each vertex in $\mathcal{D}$. The procedure *initLocalEventList*(E) initializes the local event list Q. We loop over all edges in E to compute vanish and bubble events – see the next section for details. Similarly, we loop over all reflex vertices to find splice events. In case a connected component of E has three or fewer edges, terminal events have to be considered as well. All these events, in ascending order, are stored in the local events list Q.

Now we explain the two steps of the main loop. First, we need to determine the next event. This can be either a local event, which is then already stored in Q, or a global (yet unknown) event. The computation of possible global events requires to test all edge-edge and edge-vertex pairs (for reflex vertices only). Second, we perform event handling: For all events q (local or global), we use pointers to the associated edges or vertices in E to perform the required adaptations of E, $\mathcal{D}$, and Q. Details are given below.

Prediction and handling of events

For computational purposes, the edges $e_i = v_iv_{i+1}$ in E are represented as right circular cones in space-time, determined by their centers m_i and signed radii r_i; consult Fig. 15. Events are computed by intersecting such cones with certain planes. Only intersection points in the future, i.e., with time coordinates exceeding the current time, are considered.

Local events The intersection of the three cones c_{i-1}, c_i, and c_{i+1} for three adjacent edges e_{i-1}, e_i, and e_{i+1} is used to detect vanish and bubble events. The latter events are obtained for edges e_i with strictly positive radius r_i and with both endpoints being nonconvex (but not both being smooth).

The intersection of the two cones c_{i-1}, c_i for two adjacent edges e_{i-1}, e_i with the (vertical) bisector plane of the centers m_{i-1}, m_i is used to detect a splice event,

if v_i is a reflex vertex.

Finally, terminal events occur if the number of edges of a connected component of E drops below four. They can be handled like vanish and bubble events.

Global events A squeeze event corresponds to the intersection of two non-adjacent cones c_i, c_j (that is, $|i-j| > 1$) with the plane that contains their intersection curve.

A split event can be associated with the intersection of two adjacent cones c_{i-1}, c_i with some non-adjacent cone c_j, where v_i is a reflex vertex.

Note that these intersections do not necessarily determine valid events. In order to single out those, we compute events only if they occur not later than the next local event, and we check their validity by performing a local evolution of the arc polygon (which is safe in the time frame determined by the next local event). Figure 16 gives an illustration.

Event handling Once having been detected, the generic handling of an event proceeds as follows. We evolve the offset until we reach the time of the event, and we identify the involved edges and vertices in E. According to the type of the event, we perform the necessary adaptations of the patch decomposition $\mathcal{D}$ (creation of skeleton vertices, finalization of patches, or creation of new ones, etc.) and of the evolving offset E (deleting, adding, or splitting of edges). Subsequently, we remove the events from the local event queue Q that are no longer relevant (e.g., vanish events where one of the three involved edges has been deleted), and we add the new local events created by the modified edge structure (e.g., vanish events involving three consecutive edges that became adjacent by deleting an edge).

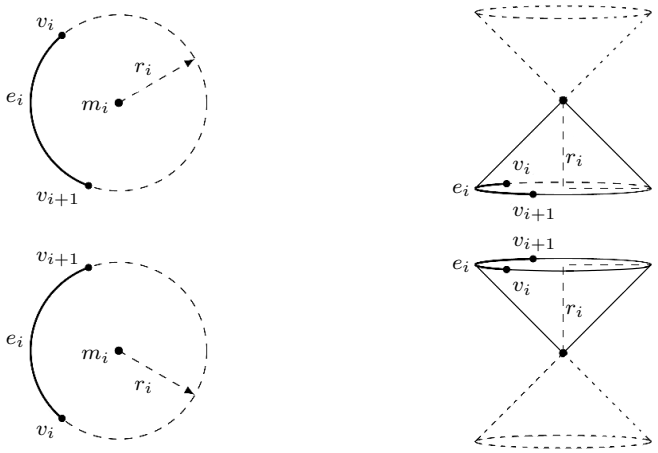


Fig. 15: Representing edges with positive (top) and negative (bottom) radius by (double-) cones.

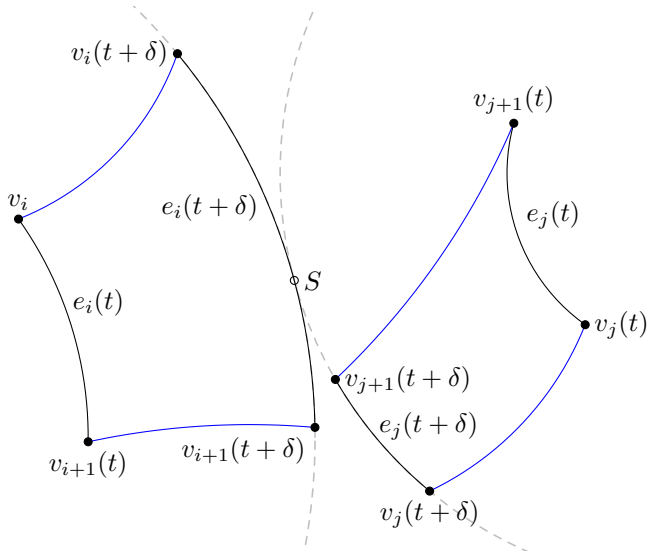


Fig. 16: Detection of an invalid candidate squeeze event point S.

Computational complexity

The runtime of Algorithm 1 is clearly dominated by the loop in Line 4. This loop deals with one event per iteration, and is therefore executed $\mathcal{O}(N)$ times, as each event either contributes to the skeleton or reduces the size of E, both of which have linear size. The detection of the next event may take $\mathcal{O}(N^2)$ time, since for global events there is a quadratic number of edge-edge or edge-vertex pairs to be tested. Consequently, the total time complexity is $\mathcal{O}(N^3)$. The space complexity is $\mathcal{O}(N)$.

To speed up the global event prediction, we have used a sweep line approach: For a certain time step δ, an axis-aligned bounding box (AABB) for each edge and each reflex vertex is established, which covers the regions that are traced by these entities in the time frame from t to $t+\delta$; see Fig. 17. A sweep line algorithm is invoked to identify all pairs of overlapping AABBs. Among those we select the candidates for global events (squeeze or split events), which are then analyzed as before.

The time step δ is chosen as an upper bound for the time to the next (local or global) event. Clearly, δ should be chosen as small as possible, since the size of the AABBs grows as δ increases. In most situations, δ is taken as the time distance to the next local event. However, there are cases when no such event exists. We then compute the 'life spans' of all vertices as the time distance until their two incident edges are in tangential contact, see Fig. 18, and choose δ as the minimum of all those. (The situation shown in the figure occurs at terminal events of the second kind, see Fig. 4, center, and at splice events, see Fig. 6.)

The overlapping AABBs can be found in $\mathcal{O}((N+k)\log N)$ time, where k is the number of intersections.[26] For a small k (that is, $k = \mathcal{O}(N)$), this is an improve-

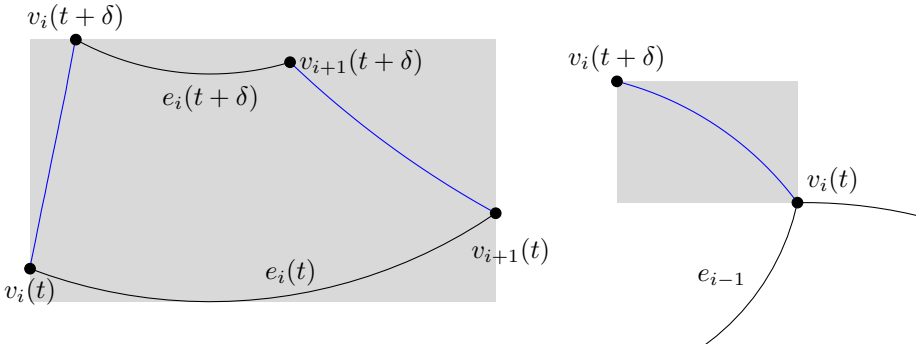


Fig. 17: The bounding boxes of an edge e_i (left) and a reflex vertex v_i (right) for the time frame δ.

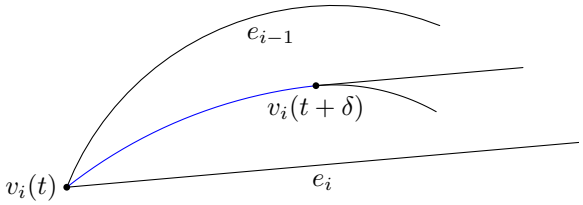


Fig. 18: Tangential contact of the edges incident to vertex v_i.

ment over the quadratic complexity of global event prediction. Note that the space complexity remains linear. In Section 5 we demonstrate that an $\mathcal{O}(N^2 \log N)$ time complexity can be achieved for realistic data.

5. Experimental results and examples

We have performed experiments with manually designed arc polygons as well as with arc polygons created by approximating planar free form shapes with spiral biarcs (arc splines).[13] These arc splines have also been used to evaluate our algorithms' performance. The algorithm is implemented in Python 3.6 on an Intel i7-6700 CPU machine with 8 GB Memory.

Some results for mitered offset computation are depicted in Fig. 19. All three examples are arc spline approximations of free-form curves. While the topmost arc polygon is smooth everywhere, the other two arc polygons possess vertices of all types (convex, reflex, and smooth).

Figure 20 shows mitered offsets that have been computed for a lion shape, for various values of the splicing parameter σ (see Section 2). In particular, Fig. 20a

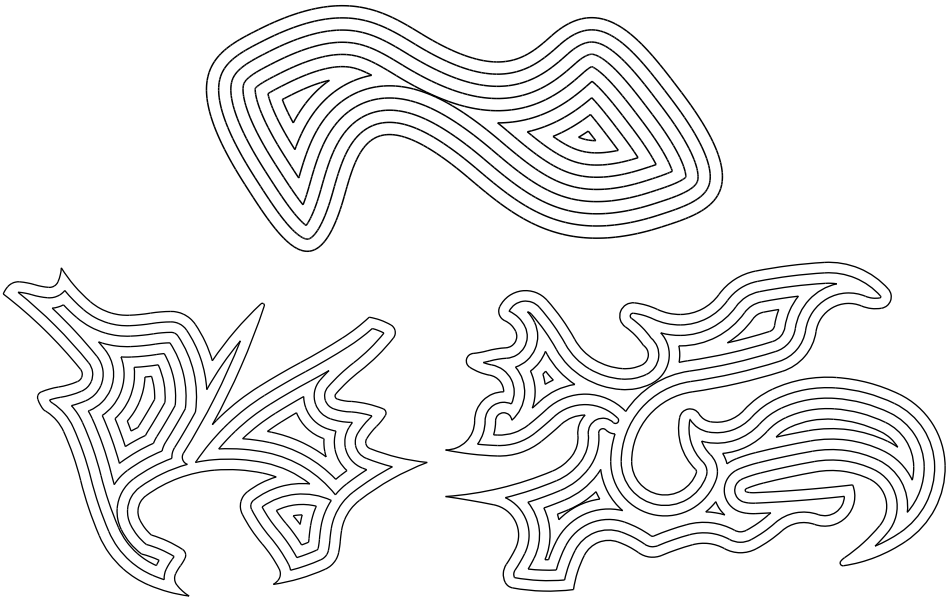


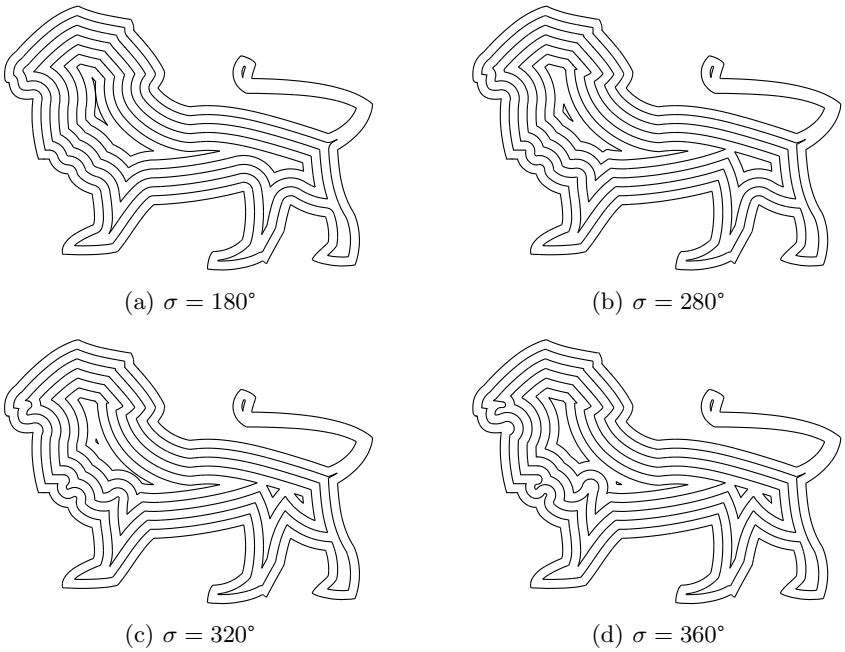
Fig. 19: Mitered offsets of the arc polygons `smooth`, `reflex`, and `snake`.

visualizes the classical offset defined by the medial axis of the shape. Increasing σ (Figs. 20b to 20d) delays the occurring splice events, thereby extending the life span of the reflex vertices.

Fig. 21 illustrates how the patch decomposition varies for different values of the splicing parameter. When σ is increased, the skeleton edges of the reflex vertices extend, and the σ-splice events move towards the interior of the polygon.

When only the (unextended) skeleton is considered, rather than the patch decomposition, the way how we represent the boundary of an arc polygon can influence the skeleton greatly. Figure 22 visualizes this effect. The different skeletons of the `snake` example in Figure 19 are shown, for a C^0 smooth and a C^1 smooth representation of the boundary, respectively. Concerning the combinatorial size of these skeletons, the size of both structures is clearly linear in the number of edges in the input, but the C^0 representation creates twice as many entities; see Fig. 23. This is due to the fact that smooth vertices do not contribute to the skeleton.

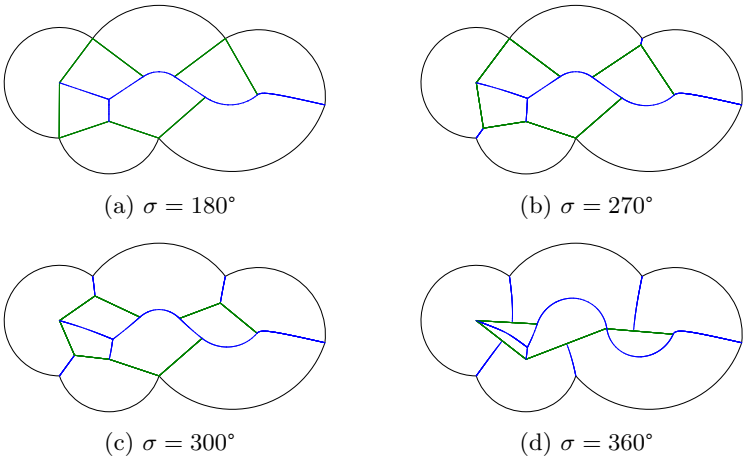
Finally, let us demonstrate experimentally that the time complexity of our algorithm is $\mathcal{O}(N^2 \log N)$. Figure 24 gives the relation between computation time and number of input edges, for the examples `smooth`, `reflex`, and `snake`. The reference lines (black) show $cN^2 \log N$.



(a) $\sigma = 180°$ (b) $\sigma = 280°$

(c) $\sigma = 320°$ (d) $\sigma = 360°$

Fig. 20: The effect of increasing the splicing parameter σ.



(a) $\sigma = 180°$ (b) $\sigma = 270°$

(c) $\sigma = 300°$ (d) $\sigma = 360°$

Fig. 21: Patch decompositions of an arc polygon, for various values of σ.

6. Conclusion

We have presented an extension of straight skeletons to shapes bounded by circular arcs and, in particular, arc splines. Experimental results indicate that the proposed

algorithm computes mitered offsets and their associated skeleton in $\mathcal{O}(N^2 \log N)$ time, where N is the number of edges of the input shape. The size of the resulting skeleton and patch decomposition is linear, the patches possess a radial monotonicity property, and the mitered offsets are locally invertible.

The introduction of the parameter σ allows us to control the exterior angle at which reflex vertices on the offsetting boundary will splice, thereby influencing the structure of the skeleton. Consequently, our algorithm can be adapted to compute either the medial axis or the skeleton (which is the straight skeleton for the special case of polygons with only straight edges). The algorithm can be directly extended to general circular arc figures in the plane that consist of various arc polygons.

Practical efficiency could be gained by using a different algorithmic approach, via the maintaining of a partitioning structure for the interior of the offsetting shape.[20] However, for arc polygons we cannot use a triangulation for partitioning

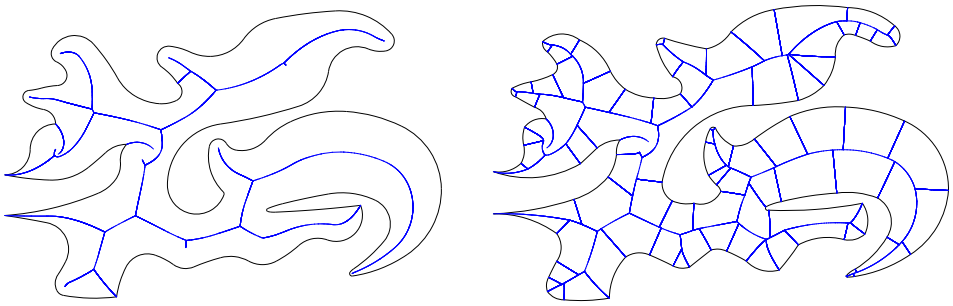


Fig. 22: Different boundary representations lead to different skeletons.

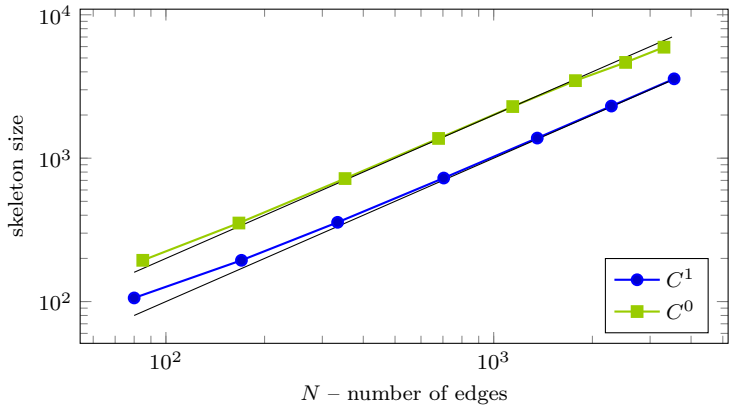


Fig. 23: Skeleton size of the `snake` example, given as a C^0 continuous arc spline (squares) and a C^1 continuous arc spline (dots), respectively.

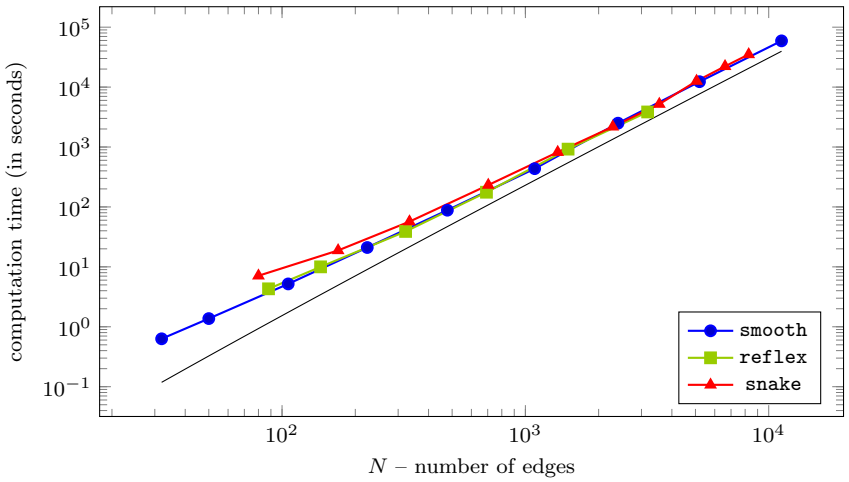


Fig. 24: Runtime behavior when the size of the arc spline representation is increased.

any more, because such a structure need not exists for an arc polygon.[24] A trapezoidal decomposition[26] will work, but will lead to an involved algorithm which is much harder to implement than ours.

Acknowledgement

Supported by the Austrian Science Fund (FWF), Project I-1836 (VORONOI++).

References

1. T. Maekawa, An overview of offset curves and surfaces, *Computer-Aided Design* **31** (1999) 165.
2. R. T. Farouki, *Pythagorean-hodograph curves: algebra and geometry inseparable*, volume 1 (Springer Science & Business Media, 2008).
3. Y.-L. Lai, J.-S. Wu, J.-P. Hung and J.-H. Chen, A simple method for invalid loops removal of planar offset curves, *International Journal of Advanced Manufacturing Technology* **27** (2006) 1153.
4. J.-K. Seong, G. Elber and M.-S. Kim, Trimming local and global self-intersections in offset curves/surfaces using distance maps, *Computer-Aided Design* **38** (2006) 183.
5. Y.-J. Kim, J. Lee, M.-S. Kim and G. Elber, Efficient offset trimming for planar rational curves using biarc trees, *Computer Aided Geometric Design* **29** (2012) 555.
6. J. Lee, Y.-J. Kim, M.-S. Kim and G. Elber, Efficient offset trimming for deformable planar curves using a dynamic hierarchy of bounding circular arcs, *Computer-Aided Design* **58** (2015) 248.
7. J. Xu, Y. Sun and L. Zhang, A mapping-based approach to eliminating self-intersection of offset paths on mesh surfaces for CNC machining, *Computer Aided Design* **62** (2015) 131.
8. H. Blum, A transformation for extracting new descriptors of shape, in *Models for the Perception of Speech and Visual Form*, ed. W. Wathen-Dunn (MIT Press, Cambridge, 1967), pp. 362–380.
9. D.-T. Lee, Medial axis transformation of a planar shape, *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1982) 363.
10. T. K. Dey and W. Zhao, Approximate medial axis as a Voronoi subcomplex, *Computer-Aided Design* **36** (2004) 195.
11. D. Meek and D. Walton, Spiral arc spline approximation to a planar spiral, *Journal of Computational and Applied Mathematics* **107** (1999) 21.
12. O. Aichholzer, W. Aigner, F. Aurenhammer, T. Hackl, B. Jüttler, E. Pilgerstorfer and M. Rabl, Divide-and-conquer for Voronoi diagrams revisited, *Computational Geometry: Theory and Applications* **8** (2010) 688.
13. O. Aichholzer, W. Aigner, F. Aurenhammer, T. Hackl, B. Jüttler and M. Rabl, Medial axis computation for planar free–form shapes, *Computer-Aided Design* **41** (2009) 339.
14. Y. Zhu, F. Sun, Y.-K. Choi, B. Jüttler and W. Wang, Computing a compact spline representation of the medial axis transform of a 2d shape, *Graphical Models* **76** (2014) 252.
15. O. Aichholzer, W. Aigner, F. Aurenhammer and B. Jüttler, Exact medial axis computation for triangulated solids with respect to piecewise linear metrics., *Curves and Surfaces* (2010) 1.
16. D. Attali, J.-D. Boissonnat and H. Edelsbrunner, Stability and computation of medial axes—a state-of-the-art report, in *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, ed. B. R. T. Müller, B. Hamann (Springer Series on Mathematics and Visualization, 2008), pp. 109–125.
17. F. Aurenhammer, R. Klein and D. T. Lee, *Voronoi Diagrams and Delaunay Triangulations* (World Scientific, 2013).
18. S. C. Park and Y. C. Chung, Mitered offset for profile machining, *Computer-Aided Design* **35** (2003) 501.
19. O. Aichholzer, F. Aurenhammer, D. Alberts and B. Gärtner, A novel type of skeleton for polygons, *Journal of Universal Computer Science* (1996) 752.
20. O. Aichholzer and F. Aurenhammer, Straight skeletons for general polygonal figures in the plane, in *International Computing and Combinatorics Conference* (Springer,

1996), pp. 117–126.
21. T. Biedl, M. Held, S. Huber, D. Kaaser and P. Palfrader, Weighted straight skeletons in the plane, *Computational Geometry* **48** (2015) 120.
22. M. Tanase and R. C. Veltkamp, A straight skeleton approximating the medial axis, *Springer Lecture Notes in Computer Science* (2004) 809.
23. F. Aurenhammer and G. Walzl, Straight skeletons and mitered offsets of nonconvex polytopes, *Discrete & Computational Geometry* **56** (2016) 743.
24. O. Aichholzer, F. Aurenhammer, T. Hackl, B. Jüttler, M. Rabl and Z. Šír, Computational and structural advantages of circular boundary representation, *International Journal of Computational Geometry & Applications* **21** (2011) 47.
25. M. Steinkogler, F. Aurenhammer and R. Klein, On merging straight skeletons, in *EuroCG* (2018), pp. 263–268.
26. M. De Berg, O. Cheong, M. Van Kreveld and M. Overmars, *Computational Geometry: Introduction* (Springer, 2008), 3rd edition.