Spiral serpentine polygonization of a planar point set

Justin Iwerks[1], **Joseph S. B. Mitchell**[1]

[1] Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY 11794-3600, USA
jiwerks@ams.sunysb.edu, jsbm@ams.sunysb.edu

Abstract. We introduce a simple algorithm for constructing a spiral serpentine polygonization of a set S of $n \geq 3$ points in the plane. Our algorithm simultaneously gives a triangulation of the constructed polygon at no extra cost, runs in $O(n \log n)$ time and uses $O(n)$ space.

Introduction

A polygonization of a planar point set S is a simple polygon having S as the set of its vertices. Different types of polygonizations have been investigated in settings where objects are being constructed from limited data, such as pattern recognition and image reconstruction **[3, 5, 6]**. The number of polygonizations for a given point set can be exponential in n, even when restricted to monotone or star-shaped polygonizations **[7]**.

Agarwal et al. have discussed the attractiveness of the subset of polygonizations that admit *thin* triangulations, which minimize the number of nodes of degree three in the dual, and in particular, *serpentine* triangulations, whose dual graph is a path. In **[2]**, the authors gave an $O(n \log n)$ algorithm for computing a serpentine polygonization of a point set S.

We show that any point set S has a *spiral* serpentine polygonization. A spiral serpentine polygon is a simple polygon possessing at most one chain of reflex vertices and exactly one chain of convex vertices (see Figure 1) and admitting a serpentine triangulation. We present a simple algorithm in Section 1 for constructing such a polygonization in $O(n \log n)$ time, requiring $O(n)$ space and explicitly giving a serpentine triangulation at no extra cost. In Section 2, a series of claims are presented (many without proof due to space constraints) establishing the correctness of the algorithm.

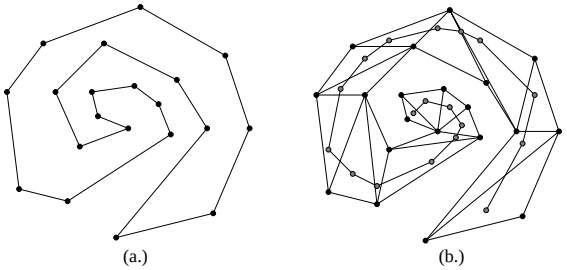

FIGURE 1. A spiral polygon is shown in (a.) and a serpentine triangulation of this polygon is shown in (b.), where the dual of the triangulation is the path depicted.

[1]This work is partially supported by the National Science Foundation (CCF-1018388).

1 The algorithm

Here we introduce the algorithm `SpiralSerpentinePolygonize`, which produces a spiral serpentine polygonization of a planar set S of $n \geq 3$ points. During the first step of the algorithm's execution, we compute the *convex layers* of all points in S. The points of a convex layer L have *depth* d_L, the number of times one would have to iteratively remove the convex hull of S until the points of L are removed [**1, 4**]. After identifying the rightmost point with minimum y-coordinate, the algorithm processes one unvisited point of S at a time. Each point becomes a vertex of a new triangle that is attached to a visible edge of the triangulation constructed so far. The spiral and serpentine invariants are maintained throughout as we discuss in Section 2.

Algorithm: `SpiralSerpentinePolygonize`(S)

1.) Compute the convex layers of S.
2.) Determine $p_1 \in S$, the point with smallest y-coordinate (breaking ties by maximizing the x-coordinate). Mark p_1 as visited.
3.) Set $p^* = p_1$ as the *pivot* point. Let p_2 be the first point encountered by rotating counterclockwise the rightwards ray emanating from p_1 (breaking ties by picking the point closest to p^*); mark p_2 as visited. Set $p^{**} = p_2$; we call p^{**} the *pass through* point. Set $\widehat{p^*}$ = null; $\widehat{p^*}$ is the *previous pivot point*. Draw the segment $\overline{p^*p^{**}}$.
4.) **while** all points have not been visited

Find the next unvisited point q encountered by rotating ccw the ray $\overrightarrow{p^*p^{**}}$ about the pivot point p^*. (q is found in time $O(\log n)$ using binary search on points of depth within 1 of the depth of p^*. Ties are broken by picking the point closest to p^*.)
if $p^{**}p^*q$ forms a right turn
Mark q as visited
Draw segments $\overline{p^*q}$ and $\overline{p^{**}q}$
Set $\widehat{p^*} = p^*$
Set $p^* = q$
else
Set $p^{**} = p^*$
Set $p^* = \widehat{p^*}$
Set $\widehat{p^*}$ = null
end while

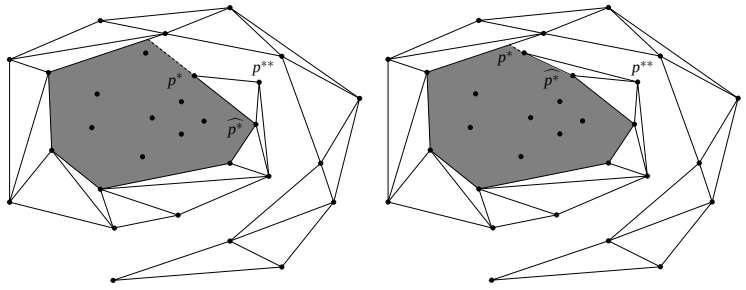

FIGURE 2. The states of the algorithm at iterations i and $i+1$, respectively, during the execution of `SpiralSerpentinePolygonize` on an example point set. The gray region indicates where unvisited points may lie. No point will be marked as visited during the $(i+1)^{th}$ iteration in this example.

In Figure 2 we illustrate a typical state of the algorithm at iteration i. The region of points yet to be visited is depicted in gray. In this example, the point with the maximum y-coordinate among the unvisited points in the gray region will be marked as visited during the i^{th} iteration. After updating the labels of the points according to the **if** clause, we obtain the state shown on the right. Clearly, the next point q encountered by rotating $\overrightarrow{p^*p^{**}}$ will not be such that $p^{**}p^*q$ forms a right turn. Thus, the **else** clause is executed, causing a "back-up" relabeling of the pivot point and reassignment of the pass through point.

2 Correctness

Lemma 2.1. *The while loop of* `SpiralSerpentinePolygonize` *terminates within* $2n$ *iterations.*

The reasoning behind this claim is that execution of the **else** clause of the **while** loop, during which no point is marked as visited, can only happen at most on every other iteration of the loop. We now show that the algorithm's output is a spiral polygon:

Lemma 2.2. `SpiralSerpentinePolygonize` *constructs a spiral polygon* P.

Proof. The proof is by induction on the iteration count. After the first iteration, we have just one triangle, which is trivially spiral. Assume the claim holds after $k < n$ iterations and consider the state after the $(k+1)^{th}$ iteration. We remove the point q_{k+1} that was most recently appended along with the two edges incident to this point (if no point was appended on the $(k+1)^{th}$ iteration, then the triangulation is unchanged and we are done). Use labels $\widehat{p^*}$, p^*, p^{**} as assigned at the end of the k^{th} iteration. The resulting polygon is spiral by the induction hypothesis. We need to show that, when q_{k+1} is processed, (a) p^{**} remains a convex vertex, and (b) p^* becomes a reflex vertex (unless $p^* = p_1$).

We consider p^{**} first. When this vertex was originally marked as visited during iteration $j < k$, it took on the label p^* during iteration j. On the subsequent iteration, no unvisited point q was discovered such that $p^{**}p^*q$ formed a right turn; otherwise, p^* would have been given the label $\widehat{p^*}$ and could never become a pass through point during some future iteration. It follows that any unvisited point q (including q_{k+1}) encountered in a subsequent iteration must be to the left of the oriented line $p^{**}p^*$. Hence, p^{**} remains a convex vertex.

We now look at p^*. If $p^* = p_1$, then it is on the convex hull and will necessarily be a convex vertex.

Now, let i be the first iteration for which $p^* \neq p_1$ and $p^{**}p^*q$ forms a right turn. If no such i exists, we argue as before. Since p^* necessarily has a depth of 2 at iteration i, there must be a point to the left of the oriented line p_1p^* with an edge to p^* in the polygonization, ensuring that p^* is reflex. This establishes the base case.

Suppose the claim holds after $k > i$ iterations. Then we wish to show that the current pivot point p^* becomes a reflex vertex upon the appendage of q_{k+1}. If no point is marked as visited on the $(k+1)^{th}$ iteration, then we simply relabel the pivot and pass through points, and the polygonization remains the same. Otherwise, we observe that q_{k+1} cannot be to the right of the oriented lined vp^*, where v is the reflex vertex previous to p^* along the reflex polygonal chain ($v = \widehat{p^*}$ if $\widehat{p^*} \neq$ null); otherwise, q_{k+1} would have been discovered previously according to the behavior of the algorithm. In other words, as in the left image of Figure 2 ($v = \widehat{p^*}$ in this example), the remaining unvisited points

are to the left of the oriented line vp^*. It follows that vp^*q_{k+1} forms a left turn, making p^* a reflex vertex. We conclude that Q is spiral. □

The subsequent two lemmas establish that, at the completion of the algorithm, a serpentine triangulation of the polygon is constructed. Although the proofs are omitted here, they are based on the observation that the algorithm constructs the polygon by simply attaching triangles iteratively. The serpentine property follows from each triangle being attached to a visible edge of the most recently formed triangle.

Lemma 2.3. `SpiralSerpentinePolygonize` *constructs a triangulation T of P.*

Lemma 2.4. *The triangulation T constructed by* `SpiralSerpentinePolygonize` *is serpentine.*

The previous four lemmas yield the desired result, stated in the following theorem:

Theorem 2.5. `SpiralSerpentinePolygonize` *constructs a spiral serpentine polygon.*

We see below that we need only consider a subset of all unvisited points during an iteration of the algorithm's **while** loop. Finally, we examine the algorithm's runtime and space usage:

Lemma 2.6. *When searching for the next unvisited point during the execution of* `SpiralSerpentinePolygonize`*, we need only consider points having depth within one of the depth of the current pivot point.*

Theorem 2.7. `SpiralSerpentinePolygonize` *runs in $O(n \log n)$ time and requires $O(n)$ space.*

Proof. The convex layers of S can be computed in $O(n \log n)$ time using $O(n)$ space [**4**]. Each subsequent point marked as visited during the execution of the while loop has depth within one of the depth of the current pivot point by Lemma 2.6. Hence, we may perform binary type searches on unvisited points of candidate convex layers in $O(\log n)$ time per iteration. There are at most $2n$ iterations by Lemma 2.1 arriving us at the desired $O(n \log n)$ run time. Since the convex layers, input points, and outputted segments can be stored in arrays of linear size, we require just $O(n)$ space. □

References

[1] M. Abellanas, J. García, G. Hernández-Peñalver, F. Hurtado, O. Serra, and J. Urrutia, Onion polygonizations, *Information Processing Letters* **57** (1996), 165–173.

[2] P. K. Agarwal, F. Hurtado, G. T. Toussaint, and J. Trias, On polyhedra induced by point sets in space, *Discrete Applied Mathematics* **156** (2008), 42–54.

[3] E. Arkin, S. P. Fekete, F. Hurtado, J. S. B. Mitchell, M. Noy, V. Sacristán, and S. Sethia, On the reflexivity of point sets, *Discrete and Computational Geometry: The Goodman–Pollack Festschrift, Algorithms and Combinatorics* **25** (2003), 139–156.

[4] B. Chazelle, On the convex layers of a planar set, *IEEE Transactions on Information Theory* **31** (1985), 509–517.

[5] L. Deneen and G. Shute, Polygonizations of point sets in the plane, *Discrete and Computational Geometry* **3** (1988), 77–87.

[6] S. P. Fekete, On simple polygonizations with optimal area, *Discrete and Computational Geometry* **23** (2000), 73–110.

[7] M. Newborn and W. O. J. Moser, Optimal crossing-free hamiltonian circuit drawings of K_n, *Journal of Combinatorial Theory, Series B* **29** (1980), 13–26.