Triangulating and Guarding Realistic Polygons

G. Aloupis [*]   P. Bose [†]   V. Dujmovic [‡]   C. Gray [§]   S. Langerman [¶]   B. Speckmann [§]

Abstract

We propose a new model of realistic input: *k-guardable* objects. An object is *k*-guardable if its boundary can be seen by *k* guards in the interior of the object. In this abstract, we describe a simple algorithm for triangulating *k*-guardable polygons. Our algorithm, which is easily implementable, takes linear time assuming that *k* is constant.

1 Introduction

Algorithms and data structures in computational geometry often display their worst-case performance on intricate input configurations that seem artificial or unrealistic when considered in the context of the original problem. In "practical" situations, many algorithms and data structures—binary space partitions are a notable example—tend to perform much better than predicted by the theoretical bounds. An attempt to understand this disparity and to quantify "practical" or "normal" with respect to input are *realistic input models* [5]. Here one places certain restrictions on the shape and/or distribution of the input objects so that most, if not all, hypothetical worst-case examples are excluded. Analyzing the algorithm or data structure in question under these input assumptions tends to lead to performance bounds that are much closer to actually observed behavior.

Many realistic input models have been proposed. These include *low-density* scenes [5], where it is assumed that the number of "large" objects intersecting a "small" volume is bounded, and *local* polyhedra [7], where it is assumed that the ratio of lengths between edges coming from a single vertex is limited by a constant. One of the most widely studied realistic input models assumes that input objects are *fat*, that is, they are not arbitrarily long and skinny. There are several ways to characterize fat objects—see the full paper for

formal definitions.

We propose a new model defining realistic input: the number of guards that are required to see the boundary of an input object. We use the term *k-guardable* to denote any object whose boundary can be seen by *k* guards. A rigorous definition of what it means for a guard to *see* can be found in the next section.

In the full paper, we discuss the connection between *k*-guardable polygons and other measures of realistic input. In particular, we show that (α, β)-*covered* polygons are *k*-guardable. An (α, β)-covered polygon is a type of fat polygon designed to model the intuitive notion of fatness for non-convex input. This is a type of polygon P that has the property that every point $p \in \partial P$ admits a triangle with minimum angle at least α and minimum edge length at least $\beta \cdot \mathrm{diam}(P)$ for given constants α and β. In the full paper we prove:

Theorem 1 *The boundary of any (α, β)-covered polygon can be guarded with $\lceil 32\pi/(\alpha\beta^2) \rceil$ guards.*

In this abstract we describe an algorithm for triangulating *k*-guardable polygons. Our algorithm, which was designed with simplicity in mind, takes $O(kn)$ time, that is, linear time assuming that *k* is constant. We also show that, if the *link diameter*—see the next section for a formal definition—of the input polygon is d, this algorithm takes $O(dn)$ time—a slightly stronger result. In the full paper we describe a second algorithm which also triangulates *k*-guardable polygons in $O(kn)$ time. That algorithm uses even easier subroutines than the one given in this abstract, but it requires the actual guards as input, which might be undesirable in certain situations.

In 1991 Chazelle [2] presented a linear time algorithm to triangulate any simple polygon. However, after all these years it is still a major open problem in computational geometry to design an implementable linear-time algorithm for triangulation. There are several implementable algorithms which triangulate polygons in near-linear time. For example, Kirkpatrick *et al.* [11] describe an $O(n \log \log n)$ algorithm and Seidel [15] presents a randomized algorithm which runs in $O(n \log^* n)$ expected time. We contend that our algorithm is conceptually simpler than the $O(n \log \log n)$ algorithm and that it has a slight advantage over the Seidel algorithm because it is deterministic. It is also interesting to note that the Seidel algorithm requires $\Omega(n \log n)$ random bits, which makes it theoretically undesirable in some models.

---
[*]Carleton University & Université Libre de Bruxelles, greg@cg.scs.carleton.ca.

[†]Carleton University, jit@cg.scs.carleton.ca.

[‡]McGill University, vida@cs.mcgill.ca.

[§]TU Eindhoven, {cgray,speckman}@win.tue.nl. C.G. is supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.023.301. B.S. is supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.022.707.

[¶]Chercheur Qualifié du FNRS, Université Libre de Bruxelles, Belgique, stefan.langerman@ulb.ac.be.

Relationships between shape complexity and the number of steps necessary to triangulate polygons have been investigated before. For example, it has been shown that monotone polygons [17], star-shaped polygons [14], and edge-visible polygons [16] can all be triangulated in linear time by fairly simple algorithms. Other measures of shape complexity that have been studied include the number of reflex vertices [9] or the sinuosity [3] of the polygon.

Several algorithms and data structures exist for collections of realistic objects. For example, the problem of ray-shooting in an environment consisting of fat objects has been studied extensively [1, 4, 10]. But there are few results concerning individual realistic objects. We hope that our results on triangulating realistic polygons will encourage further research in this direction.

The following section introduces the definitions used throughout this abstract and presents several useful tools. Section 3 describes the triangulation algorithm. We conclude in Section 4 with some open problems.

2 Tools and definitions

Throughout this paper let P be a simple polygon with n vertices. We denote the interior of P by $\text{int}(P)$, the boundary of P by ∂P, and the *diameter* of P by $\text{diam}(P)$. The boundary is considered part of the polygon, that is, $P = \text{int}(P) \cup \partial P$. We say that a point p is *in P* if $p \in \text{int}(P) \cup \partial P$.

The segment or edge between two points p and q is denoted by $\overline{pq}$. The same notation implies the direction from p to q if necessary. Two points p and q in P *see* each other if $\overline{pq} \cap P = \overline{pq}$. If p and q see each other, then we also say that p is *visible* from q and vice versa. We call a polygon P *k-guardable* if there exists a set G of k points in P called *guards* such that every point $p \in \partial P$ can see at least one point in G.
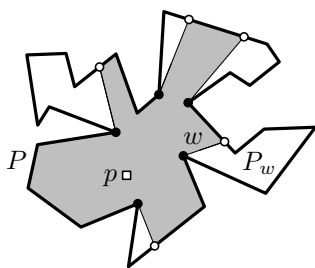
A *star-shaped* polygon is defined as a polygon that contains a set of points—the *kernel*—each of which can see the entire polygon. If there exists an edge $\overline{pq} \subset \partial P$ such that each point in P sees some point on $\overline{pq}$, then P is *weakly edge-visible*. The *visibility polygon* of a point

$p \in P$ with respect to P, denoted by $VP(p, P)$ is the set of points in P that are visible from p. Visibility polygons are star-shaped and have complexity $O(n)$.

$VP(p, P)$ can be computed in $O(n)$ time [6] with an algorithm that is non-trivial but fairly simple. It involves a single scan of the polygon and a stack. See O'Rourke's book [13] for a good summary.

A concept related to visibility in a polygon P is the *link distance*, which we denote by $ld(p, q)$ for two points p and q in P. Consider a polygonal path π that connects p and q while staying in $\text{int}(P)$. We say that π is a minimum link path if it has the fewest number of segments (links) among all such paths. The link distance of p and q is the number of links of a minimum link path between p and q. We define the *link diameter* d of P to be $\max_{p,q \in P} ld(p, q)$. The link diameter of a polygon may be much less than the number of guards required to see its boundary, and is upper bounded by the number of guards required to see the boundary. This can be seen in the so-called "comb" polygons that generally have a low link diameter but need a linear number of guards.

Let Q be a subpolygon of P, where all vertices of Q are on ∂P. If all vertices of Q coincide with vertices of P, then we call Q a *pure subpolygon*. If ∂P intersects an edge w of ∂Q only at w's endpoints, then w is called a *window* of Q. Any window w separates P into two subpolygons. The one not containing Q is the *pocket* of w with respect to Q (see Fig. 1).

The *edge-visibility polygon*, $EVP(e, P)$ of an edge e with respect to polygon P consists of all points in P that are visible from at least one point on e. We define an *extended edge-visibility polygon* of e with respect to P, denoted by $EEVP(e, P)$, to be the smallest pure subpolygon of P that contains $EVP(e, P)$. These concepts are illustrated in Figure 2.

Figure 1: The visibility polygon $VP(p, P)$ is shaded. P_w is the pocket of w with respect to $VP(p, P)$.
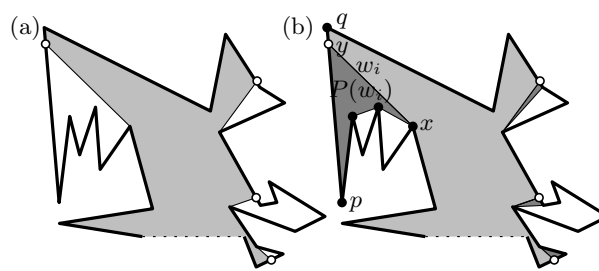
Figure 2: (a) The shaded area is the edge-visible polygon of the dashed edge; (b) the associated extended edge visible polygon.

The *geodesic* between two points in P is the shortest polygonal path connecting them that is contained in P. The vertices of a geodesic (except possibly the first and last) belong to ∂P.

Lemma 2 *Let x be a vertex of polygon P and let y be a point on edge $\overline{vw} \in P$. If y sees x, then the geodesic between x and v: (a) is a convex chain and entirely visible from y, and (b) can be computed in $O(n)$ time.*

Proof. Property (a) holds trivially if x sees v. Consider the case where x does not see v. Then, the triangle (x, y, v), denoted by T, must contain at least one vertex of P in its interior. Let I be all the vertices of P inside T and let $CH(I)$ be the convex hull of I. The path $S = CH(I) \setminus \overline{xv}$ is the geodesic from x to v. Any other path from x to v inside T can be shortened. Thus, Property (a) holds.

To prove property (b), note that since the geodesic we seek is entirely visible from y by part (a) it is fully contained in $VP(y, P)$. We compute $VP(y, P)$ in linear time. Let z be the first intersection of ∂P and the ray emanating from x in the direction $\overline{yx}$. Consider the polygonal chain from x to v along $\partial VP(y, P)$ that avoids y. By construction of $VP(y, P)$, the shortest path from x to v is part of the convex hull of this path. By Melkman's algorithm [12], the convex hull of a simple polygonal chain can be computed in linear time. $\square$

The computation of a geodesic and of an edge-visibility polygon are the two subroutines that we use to compute the *EEVP*. Hence, we can compute the *EEVP* in linear time. Also, the *EEVP* is a very structured type of polygon—the union of an edge-visible polygon and "fan" polygons—and as such can be triangulated in linear time.

Lemma 3 *EEVP(e, P) can be computed and triangulated in $O(n)$ time.*

3 Triangulating k-guardable polygons

In this section we show how to triangulate a k-guardable polygon in $O(kn)$ time. The most complicated procedure used in our algorithm is computing the visibility polygon from an edge in linear time [8]. Our algorithm relies on computing and triangulating extended edge-visibility polygons.

We begin with an arbitrary edge e of a polygon P and compute $EEVP(e, P)$. When $EEVP(e, P)$ is triangulated, the diagonals of P that are on the boundary
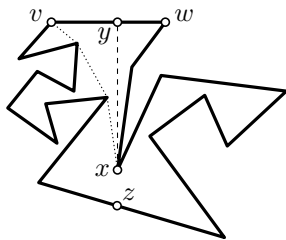

Figure 3: The geodesic from x to v.

of $EEVP(e, P)$ become windows of new pockets. Each such window serves as the edge from which a new visibility polygon will be computed and triangulated, within its respective pocket. In this recursive manner we break pockets into smaller components until all of P is triangulated. The procedure, although straightforward, is outlined below in more detail. This is followed by the analysis of the time complexity, where we show that the recursion depth is of the order of the number of guards that suffice to guard ∂P.

We will maintain a queue $\mathcal{S}$ of non-overlapping polygons (pockets) such that each $P_i \in \mathcal{S}$ has one edge w_i labelled as a window. Thus elements of $\mathcal{S}$ are pairs (w_i, P_i). We start with $\mathcal{S} := (w, P)$, where w is an arbitrary boundary edge of P. We process the elements of $\mathcal{S}$ in the order in which they were inserted. The main loop of our algorithm is as follows:

TRIANGULATEWITHOUTGUARDS(P)

1   $\mathcal{S} := (w, P)$ where w is an arbitrary edge of P
2   **while** $\mathcal{S} \neq \emptyset$
3   **do for** each $(w_i, P_i) \in \mathcal{S}$
4   **do** remove (w_i, P_i) from $\mathcal{S}$.
5   Compute and triangulate $EEVP(w_i, P_i)$.
6   Add the edges of the triangulation to P.
7   **for** each boundary edge w_j of $EEVP(w_i, P_i)$ that is a diagonal of P.
8   **do** identify Q_j as the untriangulated portion of P whose boundary is enclosed by w_j and ∂P.
9       $\mathcal{S} := \mathcal{S} \cup \{\bigcup_j (w_j, Q_j)\}$
10 **return** P.

Theorem 4 *The algorithm* TRIANGULATEWITHOUT-GUARDS *triangulates an n-vertex k-guardable polygon in $O(kn)$ time.*

Proof. We first note that a tree T is created by our algorithm. At the root of T is $EEVP(w, P)$. For every window w_j of $EEVP(w_i, P_i)$, $EEVP(w_j, P_j)$ is a child of $EEVP(w_i, P_i)$. The construction of the child nodes from their parents ensures that no $EEVP$ overlaps with any other and that the triangulation covers the entire polygon P.

We now show that T has at most $3k$ levels (a *level* is a set of nodes, each of which has the same distance from the root), which implies that the main loop of the algorithm performs at most $3k$ iterations. Let ℓ_i, ℓ_{i+1}, and ℓ_{i+2} be three successive levels of T, such that all nodes in ℓ_{i+1} are descendants of nodes in ℓ_i, and all nodes in ℓ_{i+2} are descendants of nodes in ℓ_{i+1}. Suppose that a set of points G is a *guarding* of P: every point $p \in \partial P$ sees at least one guard of G. Assume, for the purpose of obtaining a contradiction, that there are no guards from G in ℓ_i, ℓ_{i+1}, and ℓ_{i+2}.

Let g be a guard which sees into a node n_i at level ℓ_i through window w_i. There are two cases: either g is in a level higher than ℓ_i or it is in a lower level. If g is in a higher level and is visible from a window of n_i, then g can be in only one level: ℓ_{i+1} (because ℓ_{i+1} contains the union of all the edge-visibility polygons of the windows of the nodes in ℓ_i). We have assumed that this can not happen. Otherwise, if g is in a lower level, it cannot see into any higher level than ℓ_i, because w_i must be the window which created n_i.

The combination of these two facts implies that no guard from G can be able to see into ℓ_{i+1}. This is a contradiction to G being a guarding set. Therefore, G must have at least one guard in ℓ_i, ℓ_{i+1}, or ℓ_{i+2}. This implies that there is at least one guard for every three levels, or at most three levels per guard.

Each level of the tree can be processed in $O(n)$ time by Lemma 3, since all nodes of a level are disjoint. Thus the algorithm terminates successfully in $O(kn)$ time. $\qquad\square$

As is apparent from the proof of Theorem 4, our algorithm runs in $O(tn)$ time, where t is the number of iterations of the while-loop. The above argument also implies a stronger result. The number of iterations, t, of the while loop is proportional to the link diameter, d, of the polygon, since any minimum link path between two points must have at least one bend for every three levels. This leads to the following corollary:

Corollary 5 *The algorithm* TRIANGULATEWITHOUT-GUARDS *triangulates an n-vertex polygon with link diameter d in $O(dn)$ time.*

4   Open Problems

Our work raises some open problems. First, can these techniques be used to design a triangulation algorithm which does not depend on the number of guards? Second, are there other problems that can be solved efficiently for k-guardable polygons? Finally, can we find similar results for other measures of realistic input?

Acknowledgments

References

[1] B. Aronov, M. de Berg, and C. Gray. Ray shooting and intersection searching amidst fat convex polyhedra in 3-space. In *Proc. 22nd Symposium on Computational Geometry*, pages 88–94, 2006.

[2] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry*, 6(5):485–524, 1991.

[3] B. Chazelle and J. Incerpi. Triangulation and shape-complexity. *ACM Transactions on Graphics*, 3(2):135–152, 1984.

[4] M. de Berg. Vertical ray shooting for fat objects. In *Proc. 21st Symposium on Computational Geometry*, pages 288–295, 2005.

[5] M. de Berg, A. F. van der Stappen, J. Vleugels, and M. J. Katz. Realistic input models for geometric algorithms. *Algorithmica*, 34(1):81–97, 2002.

[6] H. A. El Gindy and D. Avis. A linear algorithm for computing the visibility polygon from a point. *Journal of Algorithms*, 2:186–197, 1981.

[7] J. Erickson. Local polyhedra and geometric graphs. *Computational Geometry: Theory and Applications*, 31:101–125, 2005.

[8] P. J. Heffernan and J. S. B. Mitchell. Structured visibility profiles with applications to problems in simple polygons. In *Proc. 6th Symposium on Computational Geometry*, pages 53–62, 1990.

[9] S. Hertel and K. Mehlhorn. Fast triangulation of simple polygons. In *Proc. FCT-Conference on Fundamentals of Computation Theory*, pages 207–218, LNCS 158, Springer Verlag, 1983.

[10] M. J. Katz. 3-d vertical ray shooting and 2-d point enclosure, range searching, and arc shooting amidst convex fat objects. *Computational Geometry: Theory and Applications*, 8:299–316, 1997.

[11] David G. Kirkpatrick, Maria M. Klawe, and Robert Endre Tarjan. Polygon triangulation in $O(nloglogn)$ time with simple data structures. *Discrete & Computational Geometry*, 7:329–346, 1992.

[12] A. A. Melkman. On-line construction of the convex hull of a simple polyline. *Information Processing Letters*, 25(1):11–12, Apr. 1987.

[13] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, New York, NY, 1987.

[14] A. Schoone and J. van Leeuwen. Triangulating a star-shaped polygon. Technical Report RUU-CS-80-03, Institute of Information and Computing Sciences, Utrecht University, 1980.

[15] R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Computational Geometry: Theory and Applications*, 1:51–64, 1991.

[16] G. T. Toussaint and D. Avis. On a convex hull algorithm for polygons and its application to triangulation problems. *Pattern Recognition*, 15(1):23–29, 1982.

[17] G. T. Toussaint. A new linear algorithm for triangulating monotone polygons. *Pattern Recognition Letters*, 2:155–158, 1984.