Computational Geometry 47 (2014) 296–306

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

Triangulating and guarding realistic polygons

Greg Aloupis [a,b,*], Prosenjit Bose [a], Vida Dujmović [a], Chris Gray [c,1], Stefan Langerman [d], Bettina Speckmann [c,2]

[a] Carleton University, Canada
[b] Université Libre de Bruxelles, Belgium
[c] TU Eindhoven, Netherlands
[d] Chercheur Qualifié du FNRS, Université Libre de Bruxelles, Belgium

ARTICLE INFO

Article history:
Received 13 September 2008
Received in revised form 19 March 2013
Accepted 20 March 2013
Available online 2 April 2013
Communicated by Pat Morin

Keywords:
Triangulation
Art gallery
Guarding
Polygon

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. We show that *k*-guardable polygons generalize two previously identified classes of realistic input. Following this, we give two simple algorithms for triangulating *k*-guardable polygons. One algorithm requires the guards as input while the other does not. Both take linear time assuming that *k* is constant and both are easily implementable.

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. Indeed, 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 the so-called *realistic input models* [7]. Here one places certain restrictions on the shape and/or distribution of the input objects so that most unusual 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 [7], where it is assumed that the number of "large" objects intersecting a "small" volume is bounded, and *local* polyhedra [11], 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 Section 3 for formal definitions.

In this paper, we propose a new measure to define 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.

* Corresponding author at: Carleton University, Canada.
E-mail addresses: greg@cg.scs.carleton.ca (G. Aloupis), jit@cg.scs.carleton.ca (P. Bose), vida@cg.scs.carleton.ca (V. Dujmović), cgray@win.tue.nl (C. Gray), stefan.langerman@ulb.ac.be (S. Langerman), speckman@win.tue.nl (B. Speckmann).

[1] Chris Gray is supported by the Netherlands' Organisation for Scientific Research (NWO) under project No. 639.023.301.
[2] Bettina Speckmann is supported by the Netherlands' Organisation for Scientific Research (NWO) under project No. 639.022.707.

0925-7721/$ – see front matter © 2013 Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.comgeo.2013.03.005

A rigorous definition of what it means for a guard to *see* can be found in the next section. In Section 3, we discuss the connection between k-guardable polygons and other measures of realistic input. In particular, we show that (α, β)-*covered* polygons are $O(1)$-guardable. (α, β)-covered polygons model the intuitive notion of fatness for non-convex input: an (α, β)-covered polygon P has the property that every point $p \in \partial P$ admits a triangle inside P with minimum angle at least α and minimum edge length at least $\beta \cdot \mathrm{diam}(P)$ for given constants α and β.

In Section 4, we describe two algorithms for triangulating k-guardable polygons. Our algorithms, which were designed with simplicity in mind, take $O(kn)$ time, that is, linear time assuming that k is constant. If the *link diameter*—see the next section for a formal definition—of the input polygon is d, then the algorithm described in Section 4.1 takes $O(dn)$ time—a slightly stronger result. This algorithm uses the linear-time computation of an edge-visibility polygon as a subroutine. In Section 4.2, we describe an algorithm that uses only scans of the input polygon and stacks, but requires the actual guards as input.

Related work. In 1991 Chazelle [4] 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. [16] describe an $O(n \log\log n)$ algorithm and Seidel [22] presents a randomized algorithm which runs in $O(n \log^* n)$ expected time, and Amato et al. [2] present another randomized algorithm that runs in $O(n)$ expected time that is based in large part on Chazelle's algorithm. 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 and the algorithm of Amato et al. 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 of computation.

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 [23], star-shaped polygons [21], and edge-visible polygons [24] can all be triangulated in linear time by fairly simple algorithms. Other linear algorithms have been given, under the assumption that the number of reflex vertices [13] or the sinuosity [5] is bounded by a constant.

The subject of guarding is also well studied. The book by O'Rourke [19] gives a good overview of the early results. Perhaps most relevant are the hardness results. Determining whether the entire interior of a polygon P can be seen by at most k guards is NP-complete [20] and NP-hard to approximate within a $(1+\varepsilon)$ factor [9]. The same results hold when only the boundary of P needs to be guarded: it is NP-complete to determine whether the boundary of P can be seen by at most k guards [17] and this problem is also APX-hard [9]. Finally, it has recently been shown [1] that $O(k)$ guards suffice to see the interior of a polygon if k guards suffice to see the boundary.

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 [3,6,14]. However, there are few results concerning individual realistic objects. We hope that our results on triangulating realistic polygons will encourage further research in this direction.

2. Tools and definitions

Throughout this paper let P be a simple polygon with n vertices. We assume that P has no vertical edges. If P has vertical edges, it is easy to rotate it by a small amount until the vertical edges are eliminated.

We denote the interior of P by $\mathrm{int}(P)$, the boundary of P by ∂P, and the *diameter* of P by $\mathrm{diam}(P)$. The boundary is considered part of the polygon, that is, $P = \mathrm{int}(P) \cup \partial P$. We say that a point p is *in* P if $p \in \mathrm{int}(P) \cup \partial P$.

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—see Fig. 1. Visibility polygons are star-shaped and have complexity $O(n)$. We do not describe the procedure for computing the visibility polygon in this paper. The following lemma summarizes the time complexity of computing a visibility polygon.

Lemma 1. *(See El Gindy and Avis [10].) $VP(p, P)$ can be computed in $O(n)$ time.*

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). Any vertex added to the polygon (such as the endpoint of a window) is called a *Steiner point*.

The algorithm by El Gindy and Avis [10], while not trivial, is fairly simple. It involves a single scan of the polygon and a stack. See O'Rourke's book [19] for a good summary.