Computational Geometry 47 (2014) 918–926

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

Computing a visibility polygon using few variables ☆

Luis Barba [a,b], Matias Korman [c,d,*,1], Stefan Langerman [a,2], Rodrigo I. Silveira [e,f,3]

[a] Université Libre de Bruxelles (ULB), Brussels, Belgium
[b] Carleton University, Ottawa, Canada
[c] National Institute of Informatics, Tokyo, Japan
[d] JST, ERATO, Kawarabayashi Large Graph Project, Tokyo, Japan
[e] Dept. de Matemática & CIDMA, Universidade de Aveiro, Portugal
[f] Universitat Politècnica de Catalunya (UPC), Barcelona, Spain

ARTICLE INFO

Article history:
Received 24 September 2012
Accepted 7 November 2013
Available online 23 April 2014

Keywords:
Computational geometry
Memory-constrained algorithms
Time-space-trade-off visibility
Simple polygon

ABSTRACT

We present several algorithms for computing the visibility polygon of a simple polygon $\mathcal{P}$ of n vertices (out of which r are reflex) from a viewpoint inside $\mathcal{P}$, when $\mathcal{P}$ resides in read-only memory and only few working variables can be used. The first algorithm uses a constant number of variables, and outputs the vertices of the visibility polygon in $O(n\bar{r})$ time, where $\bar{r}$ denotes the number of reflex vertices of $\mathcal{P}$ that are part of the output. Whenever we are allowed to use $O(s)$ variables, the running time decreases to $O(\frac{nr}{2^s}+n\log^2 r)$ (or $O(\frac{nr}{2^s}+n\log r)$ randomized expected time), where $s\in O(\log r)$. This is the first algorithm in which an exponential space-time trade-off for a geometric problem is obtained.

1. Introduction

The *visibility polygon* of a simple polygon $\mathcal{P}$ from a viewpoint q is the set of all points of $\mathcal{P}$ that can be seen from q, where two points p and q can see each other whenever the segment pq is contained in $\mathcal{P}$. The visibility polygon is a fundamental concept in computational geometry and one of the first problems studied in planar visibility. The first correct and optimal algorithm for computing the visibility polygon from a point was found by Joe and Simpson [20]. It computes the visibility polygon from a point in linear time and space. We refer the reader to the survey of O'Rourke [23] and the book of Gosh [17] for an extensive discussion of such problems.

☆ A preliminary version of this paper appeared in the Proceedings of the 22nd International Symposium on Algorithms and Computation (ISAAC 2011) [10].

* Corresponding author at: National Institute of Informatics, Tokyo, Japan.
E-mail addresses: lbarbafl@ulb.ac.be (L. Barba), korman@nii.ac.jp (M. Korman), stefan.langerman@ulb.ac.be (S. Langerman), rodrigo.silveira@upc.edu, rodrigo.silveira@ua.pt (R.I. Silveira).

[1] Partially supported by projects MINECO MTM2012-30951, Generalitat de Catalunya DGR2009SGR1040, and ESF EUROCORES programme EuroGIGA, CRP ComPoSe: MICINN Project EUI-EURC-2011-4306. M.K. was also partially supported by the Secretary for Universities and Research of the Ministry of Economy and Knowledge of the Government of Catalonia and the European Union.

[2] Directeur de Recherches du FRS–FNRS.

[3] Funded by Portuguese funds through CIDMA and FCT, within project PEst-OE/MAT/UI4106/2014, and by FCT grant SFRH/BPD/88455/2012, and partially supported by projects MINECO MTM2012-30951, ESF EUROCORES programme EuroGIGA, CRP ComPoSe: MICINN Project EUI-EURC-2011-4306, and Gen. Cat. DGR2009SGR104.

http://dx.doi.org/10.1016/j.comgeo.2014.04.001
0925-7721/

In this paper we look for an algorithm that computes the visibility polygon of a given point and uses few variables. This kind of algorithm not only provides an interesting trade-off between running time and memory needed, but is also useful in portable devices where important hardware constraints are present (such as the ones found in digital cameras or mobile phones). In addition, this model has direct applications in concurrent environments where several devices with limited memory resources perform some computation on a large centralized input. Since several devices may access the input at the same time, allowing writing to the input memory can result in compromising its integrity.

A significant amount of research has focused on the design of algorithms that use few variables, some of them even dating from the 80s [21]. Although many models exist, most of the research considers that the input is in some kind of read-only data structure. In addition to the input values, we are allowed to use few additional variables (typically a variable holds a logarithmic number of bits).

One of the most studied problems in this setting is that of selection. For any constant $\epsilon \in (0,1)$, Munro and Raman [22] gave an algorithm that runs in $O(n^{1+\epsilon})$ time and uses $O(1/\epsilon)$ variables. Frederickson [16] extended this result to the case in which s working variables are available (and $s \in \Omega(\log n) \cap O(2^{\log n/\log^* n})$). Raman and Ramnath [24] gave several exact and approximation algorithms for the case in which fewer variables are available. Among other results, they provide a 2/3-approximation of the median that runs in $O(sn^{1+1/s})$ time using $O(s)$ variables (for $s \in o(\log n)$), or $O(n \log n)$ time, using $O(\log n)$ variables. More recently Chan [14] provided several lower bounds for performing selection with few variables.

In recent years there has been a growing interest in finding algorithms for geometric problems that use a constant number of variables. An early example is the well-known gift-wrapping algorithm (also known as Jarvis march [19]), which can be used to report the points on the convex hull of a set of n points in $O(n\bar{h})$ time using a constant number of variables, where $\bar{h}$ is the number of vertices on the convex hull. Recently, Asano and Rote [8] and afterwards Asano et al. [5,7] gave efficient methods for computing well-known geometric structures, such as the Delaunay triangulation, the Voronoi diagram, a polygon triangulation, and a minimum spanning tree (MST) using a constant number of variables. These algorithms run in $O(n^2)$ time (except computing the MST, which needs $O(n^3)$ time). Observe that, since these structures have linear size, they are not stored but reported.

1.1. Results

In this paper we present a novel approach for computing the visibility polygon of a given point inside a simple polygon. It is easy to see that reflex vertices have a much larger influence on the visibility polygon than convex vertices. Therefore, whenever possible we express the running time of our algorithms not only in terms of n, the complexity of $\mathcal{P}$, but also in terms of r and $\bar{r}$ (the number of reflex vertices of $\mathcal{P}$ that are present in the input and in the output, respectively). This approach continues a line of research relating the combinatorial and computational properties of polygons to the number of their reflex vertices. We refer the reader to [3,11,12] and references found therein for a deep review of existing similar results.

In Section 2 we begin the paper with some preliminaries, followed by some observations and basic algorithms in Section 3. In Section 4 we give an output-sensitive algorithm that reports the vertices of the visibility polygon in $O(n\bar{r})$ time using $O(1)$ variables. Using this algorithm as a stepping stone, in Section 5 we present a divide-and-conquer algorithm. This algorithm runs in $O(\frac{nr}{2^s} + n\log^2 r)$ time (or $O(\frac{nr}{2^s} + n\log r)$ randomized expected time) using $O(s)$ variables (for any $s \in O(\log r)$), giving an exponential trade-off between running time and space.

Remark. Prior to this work, there was no algorithm for computing the visibility polygon in memory-constrained models. Indeed, this problem was explicitly posed as an open problem by Asano et al. [6] for the case in which only a constant number of variables is allowed. Following the conference version of this paper [10], De et al. [15] presented a linear-time algorithm that uses $o(n)$-variables. Unfortunately, the result turned out to be incorrect [1,2]. In a companion paper [9] we give a general method for transforming stack-based algorithms into memory constrained workspaces. Since Joe and Simpson's algorithm for computing the visibility polygon [20] is stack-based, that approach can be used for this problem as well.

2. Preliminaries

2.1. Model definition and considerations on input/output precision

We use a slight variation of the constant workspace model, introduced by Asano and Rote [8]. In this model the input of the problem resides in a read-only data structure and we are allowed to perform random access to any of the values of the input in constant time. An algorithm can use a constant number of variables and we assume that each variable or pointer contains a data word of $O(\log n)$ bits. Implicit storage consumption required by recursive calls is also considered part of the workspace. This model is also referred as *logspace* [4] in the complexity literature.

Many other similar models have been studied. We note that in some of them (like the *streaming* [18] or the *multi-pass* model [13]) the values of the input can only be read once or a fixed number of times. As in the constant workspace model of Asano and Rote [8], our model allows scanning the input as many times as necessary. However, our model differs from theirs in two aspects: we are allowed to use a workspace of $O(s)$ variables (instead of $O(1)$), and we do not require random access to the vertices of the input.

The input to our problem is a simple polygon $\mathcal{P}$ in a read-only data structure and a point q in the plane, from where the visibility polygon needs to be computed. We do not make any assumptions on whether the input coordinates are rational