Journal of Computational and Applied Mathematics 288 (2015) 215–222

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Weak visibility counting in simple polygons

Mojtaba Nouri Bygi [a,*], Shervin Daneshpajouh [a], Sharareh Alipour [a], Mohammad Ghodsi [a,b]

[a] Computer Engineering Department, Sharif University of Technology, Iran

[b] School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Iran

HIGHLIGHTS

- Studying weak visibility counting problem for line segments in simple polygons.
- Presenting several exact algorithms, and showing a trade-off between preprocessing costs and query time costs.
- Presenting an approximation algorithm for the problem, and reducing preprocessing costs.

ARTICLE INFO

Article history:
Received 7 July 2014
Received in revised form 8 December 2014

Keywords:
Computational geometry
Weak visibility
Visibility counting

ABSTRACT

For a simple polygon $\mathcal{P}$ of size n, we define weak visibility counting problem (WVCP) as finding the number of visible segments of $\mathcal{P}$ from a query line segment pq. We present different algorithms to compute WVCP in sub-linear time. In our first algorithm, we spend $O(n^7)$ time to preprocess the polygon and build a data structure of size $O(n^6)$, so that we can optimally answer WVCP in $O(\log n)$ time. Then, we reduce the preprocessing costs to $O(n^{4+\epsilon})$ time and space at the expense of more query time of $O(\log^5 n)$. We also obtain a trade-off between preprocessing and query time costs. Finally, we propose an approximation method to reduce the preprocessing costs to $O(n^2)$ time and space and $O(n^{1/2+\epsilon})$ query time.

1. Introduction

Two points inside a polygon are *visible* to each other if their connecting segment remains completely inside the polygon. *Visibility polygon* of a point p in a simple polygon $\mathcal{P}$ is the set of points in $\mathcal{P}$ that are visible from p.

In many problems, it is good to have a sense of the size of visible area before computing it. The visibility counting problem (VCP) in a polygonal domain is to find the number of objects (segments, edges, etc.) that are visible from a point p. An inefficient solution would be to compute the visibility polygon of p and then report its size.

For a simple polygon, Bose et al. showed that by preprocessing the polygon in $O(n^3 \log n)$ time and building a data structure of size $O(n^3)$, one can answer VCP for a query point in $O(\log n)$ time [1]. For a set of n disjoint line segments, Suri and O'Rourke introduced the first 3-approximation algorithm for VCP, by representing the visibility polygon of the segments as a set of convex (triangular) regions [2]. Gudmundsson and Morin improved this result to a 2-approximation algorithm by using an improved covering scheme [3]. The same result was achieved by Alipour and Zarei [4] and Nouri and Ghodsi [5]. Alipour and Zarei proved that the number of visible end-points of the segments is a 2-approximation of VCP [4].

* Corresponding author.
E-mail addresses: nouribygi@ce.sharif.edu (M. Nouri Bygi), daneshpajouh@ce.sharif.edu (S. Daneshpajouh), alipour@ce.sharif.edu (S. Alipour), ghodsi@sharif.edu (M. Ghodsi).

http://dx.doi.org/10.1016/j.cam.2015.04.018
0377-0427/

Table 1
A summary of our results on WVCP. The first part shows exact algorithms, and the second part shows the approximation result.

	Prep. time	Size	Query time
Naive	–	$O(n)$	$O(n)$
Exact	$O(n^7)$	$O(n^6)$	$O(\log n)$
Exact	$O(n^{4+\epsilon})$	$O(n^{4+\epsilon})$	$O(\log^5 n)$
Exact	$O(n^{3+\epsilon})$	$O(n^3)$	$O(n^{1/2+\epsilon})$
Approx	$O(n^2)$	$O(n^2)$	$O(n^{1/2+\epsilon})$

There are two other approximation algorithms for VCP by Fischer et al. [6,7]. The first algorithm uses a data structure of size $O((m/r)^2)$ by which the queries are answered in $O(\log n)$ time, with an absolute error of r ($1 \leq r \leq n$). Here, m is the size of Visibility Graph, which is $O(n^2)$ [8]. The second algorithm uses a random sampling method to build a data structure of size $O((m^2 \log^{O(1)} n)/l)$ to answer any query in $O(l \log^{O(1)} n)$ time, where $1 \leq l \leq n$.

The visibility problem has also been considered for line segments. A point v is said to be *weakly visible* from a line segment pq if there exists a point $w \in pq$, such that w and v are visible to each other. The problem of computing the *weak visibility polygon* (or WVP) of a line segment pq inside a polygon $\mathcal{P}$ is to compute all points of $\mathcal{P}$ that are weakly visible from pq. If $\mathcal{P}$ is a simple polygon, WVP(pq) can be computed in linear time [9]. Also, Nouri Bygi and Ghodsi showed that WVP(pq) can be computed in output sensitive query time of $O(\log n + |\text{WVP}(pq)|)$, by building a data structure of size $O(n^3)$ in $O(n^3 \log n)$ time [10,11].

1.1. Our contributions

In this paper we consider the weak visibility counting problem (WVCP) for line segments in simple polygons. We define the weak visibility count of a line segment inside a simple polygon as the size of the weak visibility polygon of the line segment.

Our approaches are divided into two categories: exact counting and approximate counting. In the first part of the paper, we present two algorithms that compute the exact size of the weak visibility polygon in sub-linear time, after preprocessing the polygon. In the second part, we show how to reduce the preprocessing costs of the exact algorithms, by approximating the counting problem. A summary of our results is given in Table 1.

The rest of this paper is organized as follows. In Section 2, we review some definitions and structures that will be used throughout the paper. In Section 3, we present three exact algorithms for computing WVCP. We also give an approximation algorithm based on random sampling in Section 4. Section 5 concludes the paper.

1.2. Motivations and applications

Over the years, visibility has become one of the most studied problems in computational geometry, and has many applications in computer graphics and computational geometry. As visibility is a natural phenomenon in everyday life [12], such problems arise in areas where computational geometry tools and algorithms find applications. In addition, we can use their solutions as building blocks to solve other problems, such as motion planning and art gallery problems. The reader is referred to [13,12].

Sometimes, we do not want to compute all the visible parts of the scene, but we need to have a sense of the size of the visible area. This problem is a special case of *hidden surface removal* which has many applications in computer graphics and computer games [14]. For example, consider a 3-D scene and a rectangular observer (e.g. a ship in a computer game), and assume that we want to find the primitives that can be seen from the observer. The 3D scene and rectangular shape can be simplified as a 2-D polygon and a segment observer. Z-buffer algorithm is available in current computer graphic cards and is widely used for solving these problems. The z-buffer can render scenes of n triangles in time linear in n. But, for a scene consisting of several millions of triangles, even this algorithm may be too slow.

By calculating the size of visibility polygon, we can efficiently deduce some visibility properties of the scene and remove unnecessary objects from our consideration, without computing all the visible parts of the scene.

1.2.1. Computing weak visibility line simplification inside a simple polygon

As another practical example, we consider line simplification problem inside a simple polygon, under visibility measure.

Consider a building and its plan in 2D, and a guard robot with visibility sensors. Also, assume that there are some moving adversary objects in the building. A path is given to the robot and it should move along the path and report any adversary object it sees. Note that the path may be designed in such a way that certain parts of the building can be seen several times. We would like to find a simplified path such that the visibility of the robot does not change, or remains within a given factor ϵ. In other words, while moving along the simplified path, we would like not to lose a large portion of visible parts of the polygon with respect to the original path.