Computational Geometry 54 (2016) 17–31

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

Distance-sensitive planar point location ☆

Boris Aronov [a], Mark de Berg [b], David Eppstein [c], Marcel Roeloffzen [d,e], Bettina Speckmann [b]

[a] Dept. of Computer Science and Engineering, Tandon School of Engineering, New York University, USA
[b] Dept. of Computer Science, TU Eindhoven, The Netherlands
[c] Dept. of Computer Science, Donald Bren School of Information and Computer Sciences, University of California, Irvine, USA
[d] National Institute of Informatics (NII), Tokyo, Japan
[e] JST Kawarabayashi ERATO Large Graph Project, Japan

ARTICLE INFO

Article history:
Received 28 April 2014
Accepted 4 February 2016
Available online 9 February 2016

Keywords:
Point location
Quadtree
Mesh generation

ABSTRACT

Let $\mathcal{S}$ be a connected planar polygonal subdivision with n edges that we want to preprocess for point-location queries, and where we are given the probability γ_i that the query point lies in a polygon P_i of $\mathcal{S}$. We show how to preprocess $\mathcal{S}$ such that the query time for a point $p \in P_i$ depends on γ_i and, in addition, on the distance from p to the boundary of P_i—the further away from the boundary, the faster the query. More precisely, we show that a point-location query can be answered in time $O\left(\min\left(\log n, 1 + \log \frac{\text{area}(P_i)}{\gamma_i \Delta_p^2}\right)\right)$, where Δ_p is the shortest Euclidean distance of the query point p to the boundary of P_i. Our structure uses $O(n)$ space and $O(n \log n)$ preprocessing time. It is based on a decomposition of the regions of $\mathcal{S}$ into convex quadrilaterals and triangles with the following property: for any point $p \in P_i$, the quadrilateral or triangle containing p has area $\Omega(\Delta_p^2)$. For the special case where $\mathcal{S}$ is a subdivision of the unit square and $\gamma_i = \text{area}(P_i)$, we present a simpler solution that achieves a query time of $O\left(\min\left(\log n, \log \frac{1}{\Delta_p^2}\right)\right)$. The latter solution can be extended to convex subdivisions in three dimensions.

1. Introduction

Point location is one of the most fundamental problems in computational geometry. Given a subdivision $\mathcal{S}$ the goal is to preprocess it so that we can determine efficiently which region of $\mathcal{S}$ contains a query point p. Many different variants of the point-location problem exist; in our work we first focus on planar point location in polygonal subdivisions and later extend one of our results to convex polyhedral subdivisions in three dimensions. In the following, unless otherwise specified, our subdivision $\mathcal{S}$ is subdivision of a polygonal domain in the plane into polygons. The subdivision need not be conforming—we

☆ B. Aronov has been supported by United States–Israel Binational Science Foundation grant 2006/194, by NSF Grants CCF-08-30691, CCF-11-17336, and CCF-12-18791, and by NSA MSP Grant H98230-10-1-0210. D. Eppstein has been supported by NSF grant 1217322 and ONR grant N00014-08-1-1015. M. Roeloffzen and B. Speckmann were supported by the Netherlands' Organisation for Scientific Research (NWO) under project nos. 600.065.120 and 639.023.208, respectively.

E-mail addresses: boris.aronov@nyu.edu (B. Aronov), mdberg@win.tue.nl (M. de Berg), eppstein@ics.uci.edu (D. Eppstein), marcel@nii.ac.jp (M. Roeloffzen), speckman@win.tue.nl (B. Speckmann).

http://dx.doi.org/10.1016/j.comgeo.2016.02.001
0925-7721/

may have T-junctions, for instance—but when considering a polygon P_i of the subdivision we ignore the subdivision vertices whose angle inside P_i is exactly π. A triangulation is a subdivision consisting of triangles.

There are several different solutions for planar point location that are worst-case optimal. In particular, there are structures that require $O(n \log n)$ preprocessing, use $O(n)$ space, and can answer a point-location query in $O(\log n)$ time; see the surveys by Preparata [18] and Snoeyink [22] for an overview. In three dimensions no point location structure is known for general subdivisions that uses linear space and has logarithmic query time. For convex subdivisions Preparata and Tamassia [19] showed that combining dynamic planar point location and persistency techniques yield an $O(n \log^2 n)$ space structure that answers queries in $O(\log^2 n)$ time. This method was later extended and improved so that it works for general subdivisions and requires only $O(n \log n)$ space and preprocessing time for $O(\log^2 n)$ query time [12,22].

For planar point location a query time of $O(\log n)$ is optimal in the worst case, but it may be possible to do better for certain types of query points. For example, if the query points are not distributed uniformly among the regions of $\mathcal{S}$, then it may be desirable to reduce the query time for points in frequently queried regions. Iacono [13] showed that this is indeed possible: given a triangulation $\mathcal{S}$ where each triangular region R_i has a probability γ_i associated with it—the probability that the query point p falls in R_i—then one can answer a point-location query in expected time $O(H(\mathcal{S}))$, where

$$H(\mathcal{S}) := \sum_{R_i \in \mathcal{S}} \gamma_i \log \frac{1}{\gamma_i},$$

is the *entropy* of $\mathcal{S}$. This result is optimal, because the entropy is a lower bound on the expected query time [16,21]. Several other point-location structures have been proposed that answer queries in $O(H(\mathcal{S}))$ expected time [1,2]. The structure presented by Arya, Malamatos, and Mount [1] is relatively simple and efficient in practice. It works for subdivisions with constant-complexity regions and, for any region R_i the worst-case query time for points inside R_i is $O(1+\min(\log \frac{1}{\gamma_i}, \log n))$. The results mentioned so far assume that the distribution is known in advance. Recently Iacono [14] proposed an algorithm that eventually achieves $O(H(\mathcal{S}))$ query time, but does not need any knowledge of the query distribution. Instead, the algorithm changes the structure according to the queries received. The results mentioned above require the regions of the input subdivision $\mathcal{S}$ to have constant complexity. This requirement is necessary. Indeed, if a subdivision with n edges has only two regions, each with associated probability $1/2$, then we cannot hope to achieve $O(1)$ query time. One could of course subdivide the regions into constant complexity regions, say triangles, and distribute the query probability evenly among these regions. However, in many cases one would expect that queries are not evenly distributed within each polygon. For example if queries come from users selecting polygons by clicking on them, one would expect most queries to occur far from the boundary as users are inclined to click in the 'middle' of a region. This raises the question if it is possible to improve query times depending on where the query point is within the polygon that contains it. In our work we investigate the possibility of relating the query time to the distance of a query point to the nearest point on the boundary of the region that contains it. We call this *distance-sensitive point location*.

Differentiating between query points within higher complexity polygons is not new. Collette et al. [7] showed how to compute, for any simple polygon P and any probability distribution over P, a Steiner triangulation with near-optimal entropy, and they proved that the minimum entropy of any triangulation is a lower bound on the expected query time for point-location in the linear decision-tree model. By applying their Steiner triangulation to every region in the given subdivision, and using the resulting triangles as input for an entropy-based point-location structure, near-optimal expected query time is achieved. In the case of distance-sensitive point location we could define a probability distribution based on the distance of points to the region boundary and construct such a Steiner triangulation. Unfortunately, a near-optimal entropy does not imply any bounds on specific query points. Indeed the construction by Collette et al. can generate very small triangles, even in high probability areas. A point p that is far from the boundary can end up in such a very small triangle, which has a small total probability. As a result a query for p has a long query time. We will focus on creating a point location structure that guarantees fast query time for any point far from the region boundary.

Problem definition Let p be a query point inside polygon $P_i \in \mathcal{S}$ with area $\text{area}(P_i)$ and probability γ_i that a query point is inside P_i. We want the time of a query for p to be

$$O\left(\min\left(\log n, 1 + \log \frac{\text{area}(P_i)}{\gamma_i \Delta_p^2}\right)\right).$$

Here, Δ_p denotes the minimum Euclidean distance from p to the boundary of P_i. When the polygons in the subdivision have constant complexity, then this can be achieved using, for instance, the entropy-based point-location structure of Arya, Malamatos, and Mount [1]. Since for any point $p \in P_i$ the distance to the boundary of P_i is $O(\sqrt{\text{area}(P_i)})$, this gives the desired query bound. When polygons have higher complexity we can also use an entropy-based structure, but first have to decompose each polygon into constant complexity regions and assign probabilities appropriately. Specifically we show that it is sufficient to compute for each polygon $P \in \mathcal{S}$ with n_P vertices a *distance-sensitive decomposition* of P into $O(n_P)$ regions with the following properties: