Visualizing Quickest Visibility Maps

Topi Talvitie

Department of Computer Science, University of Helsinki, Finland

Abstract

Consider the following modification to the shortest path query problem in polygonal domains: instead of finding shortest path to a query point q, we find the shortest path to any point that sees q. We present an interactive visualization applet visualizing these *quickest visibility paths*. The applet also visualizes *quickest visibility maps*, that is the subdivision of the domain into cells by the quickest visibility path structure.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases path planning, visibility

Digital Object Identifier 10.4230/LIPIcs.SOCG.2015.26

1 Introduction

Finding shortest paths within a polygonal domain is a classical problem in computational geometry. The query version of the shortest path problem for a fixed point s is

Shortest Path Query: Given a point q, how should we move from s to **reach** q?

The problem of shortest path queries is solved by building a *shortest path map* for s, defined as the decomposition of the domain into cells such that the shortest paths to all points q within that cell is the same (the only changing vertex is the endpoint q). Consider a modification of this problem, where we only need to see the query point:

Quickest Visibility Query: Given a point q, how should we move from s to **see** q?

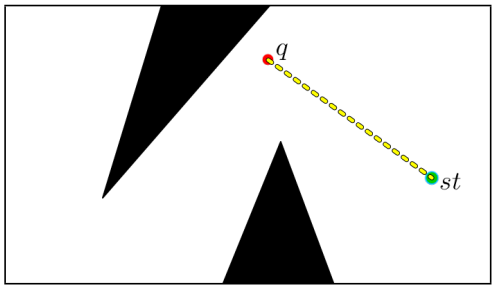
This kind of query would be natural in applications where it is important to only see or become seen by the target point, for example for inspecting the query point or establishing communication with it. Quickest visibility queries were first studied in the case of simple polygons in [2]. This visualization accompanies the paper [1] which presents algorithms for the general case of polygons with holes and improves the results in the case of simple polygons. The visualization applet demonstrates the core concepts of quickest visibility queries: quickest visibility paths, quickest visibility maps and visibility wave propagation. These are briefly outlined below. For more formal definitions and proofs please refer to [1].

2 Quickest visibility paths

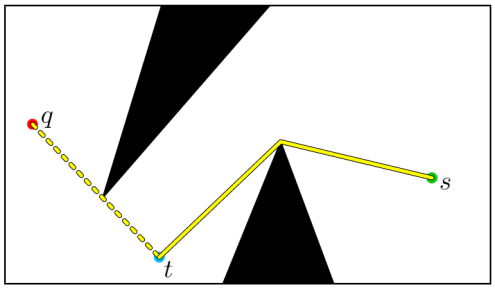
A *quickest visibility path* (QVP) from s to q is the shortest path from s to some endpoint t such that q is visible from t. It is possible that q is directly visible from s. In that case, the quickest visibility path is the path of length zero from s to s (Fig. 1a). If t lies in the interior of the domain, the path always enters t orthogonally to line tq, because otherwise we could adjust the location of t to shorten the path (Fig. 1b). Otherwise t is either in a vertex of a polygon (Fig. 1c) or on an edge of a polygon such that tq contains a polygon vertex (Fig. 1d).

31st International Symposium on Computational Geometry (SoCG'15).
Editors: Lars Arge and János Pach; pp. 26–28

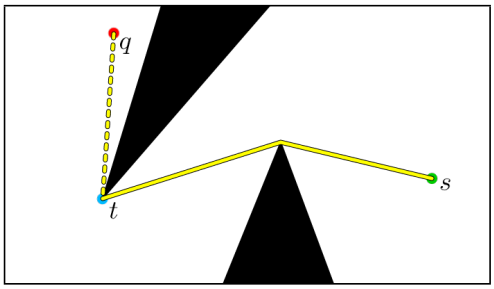
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



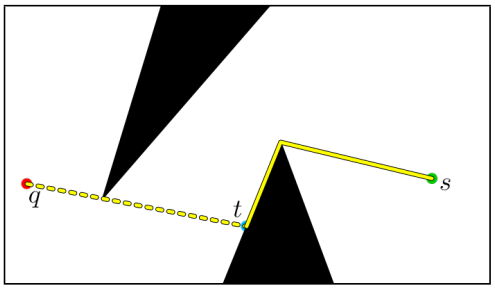
(a) q is directly visible from s. Therefore $s = t$ and the QVP has length 0.



(b) t lies in free space. The QVP and tq meet orthogonally in t.



(c) t lies on a polygon vertex.



(d) t lies on a polygon edge. Segment tq touches a polygon vertex.

Figure 1 The four different types of quickest visibility paths from s to q. The quickest visibility path is drawn as a solid yellow line.

3 Quickest visibility maps

A *quickest visibility map* (QVM) for a polygonal domain and a source point s is the subdivision of the domain into cells such that for all points q within a cell, the QVP from s to q has the same structure. We say that two QVPs have the same structure if the paths differ only in the last point q, and in the case of paths of type shown in Fig. 1b, the second-to-last point t that moves when point q moves. See Fig. 2 for an example of a QVM.

Once a QVM has been built for source point s, one can query quickest visibility paths from s to any q by using point location query data structures to find the QVM cell q lies in. Therefore QVM is the analogue of shortest paths maps in the case of quickest visibility paths.

4 Implementation

The visualization applet consists of two parts: the backend library, implemented in C++, and the user interface, implemented in JavaScript.

The backend implements an algorithm for finding the quickest visibility path to a given query point. It does not use quickest visibility maps, as exact QVM construction would very complicated to implement. Instead, it handles quickest visibility paths of types shown in figures 1a, 1c and 1d by querying shortest paths using the visibility graph. That leaves only visibility paths of type where the path endpoint lies in free space (Fig. 1b), which can be found by iterating all polygon vertices v visible to q, and all points from which the extension of vq is orthogonally visible. This can be implemented simultaneously for all v as two ray

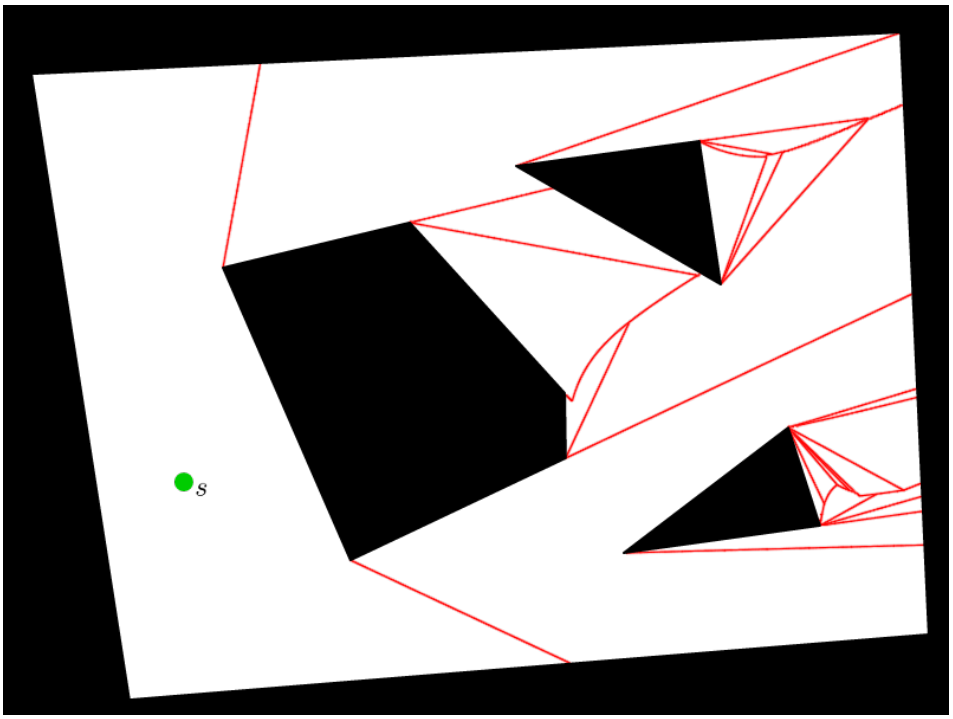


Figure 2 A screenshot from the visualization applet, showing the subdivision of the polygonal domain by the red lines into the quickest visibility map for source point s.

sweeps around q to both directions in $O(n \log n)$ time with the help of precomputed shortest paths to all polygon vertices.

The user interface contains a polygon editor, in which the user can edit the polygon and set the source point s. It visualizes quickest visibility paths using the backend library. It draws the QVM by querying the quickest visibility paths to a dense grid of points in the domain, drawing points of the QVM edges in locations where adjacent grid query points have different quickest visibility paths. A local optimization algorithm is used to improve the precision of the QVM edges.

The visualization is a client-side HTML5/JavaScript applet. The backend library is compiled from C++ to JavaScript using the Emscripten compiler. The precomputation used for drawing the QVM edges is parallelized using Web Workers, which makes the loading phase faster in multi-core systems. The applet should work on all modern browsers, including the newest versions of all major web browsers.

Acknowledgments. The research was supported by the University of Helsinki Research Funds.

References

1 E. M. Arkin, A. Efrat, C. Knauer, J. S. B. Mitchell, V. Polishchuk, G. Rote, L. Schlipf, and T. Talvitie. Shortest path to a segment and quickest visibility queries. In *SoCG*, 2015.

2 R. Khosravi and M. Ghodsi. The fastest way to view a query point in simple polygons. In *European Workshop on Computational Geometry*, pages 187–190. Technische Universiteit Eindhoven, 2005.