An Implementation for Maintaining Arrangements of Polygons

Michael Goldwasser *
Stanford University

Abstract

Constructing arrangements of geometric objects is a basic problem in computational geometry. Applications relying on arrangements arise in such fields as robotics, assembly planning, computer vision, graphics, and computer-assisted surgery. Arrangements are also used as a building block for other theoretical results in computational geometry. Many papers and textbooks have presented algorithms for maintaining arrangements under various conditions.

This paper is a discussion of the practical issues that arose during the development of a software package which constructs an arrangement of polygons and segments using a basic randomized incremental approach. The need to handle polygons in addition to segments, and to deal with arrangements on a sphere as well as a plane, guided many design decisions. Also the need to cope with degeneracies and numerical inaccuracy in an efficient and consistent manner, brought up issues that are often glossed over in theoretical presentations of algorithms.

This software is written in C and is available via ftp at `flamingo.stanford.edu` in the directory `/pub/wass/arrangement/`.

1 Project Background

This software was originally developed for use in the Stanford Assembly Analysis Tool (STAAT), an assembly planner for polyhedral parts [4]. For each pair of parts, analysis is performed to calculate the set of directions in which one part will collide with the other. This region is represented as a polygon on the sphere of directions where vertices are points on the sphere and edges are great circle arcs. The overall arrangement of these polygons is then traversed and analyzed to develop possible assembly plans.

2 Features and Limitations

There are several features of this software which allow extra flexibility that was necessary for the original project.

- The basic objects are polygons rather than isolated segments. The important distinction between the representation of a polygon as an object versus as a set of individual edges is that the resulting arrangement must *guarantee* that common endpoints in the polygon are truly unified in the arrangement.
- A polygon may be degenerate and represent a simple segment, or even an isolated vertex. Therefore this software has the generality to handle arrangements of segments.
- Many algorithms are designed for arrangements of non-intersecting objects; however in this application, polygon segments may intersect segments from other polygons.
- The input polygons may lie on the sphere rather than on the plane. Although this brings up some key programming issues, it turns out to be a minor conceptual issue.

The notable limitation of this software is the fact that it is only semi-dynamic. For this application, polygons could be incrementally inserted however there was no need to delete polygons from the arrangement.

3 Algorithm

We represent the arrangement using a vertical decomposition into trapezoids[1]. To avoid the maintanance of faces with arbitrarily many edges, we decompose each face into a number of trapezoids. This is done by extending an imaginary vertical *thread* from each vertex of the face upwards (downwards) into the face until an edge of the face is reached. See Figure 1.

*Department of Computer Science, Stanford University, Stanford, CA, 94305. E-mail: `wass@cs.stanford.edu`. Research supported by a grant from the Stanford Integrated Manufacturing Association (SIMA), by NSF/ARPA Grant IRI-9306544, and by NSF Grant CCR-9215219.

The algorithm used for construction is a basic randomized incremental construction, similar to that of Mulmuley[3] or Seidel[5]. We give a brief explanation here, but we refer the reader to [3], [5] for details.

We maintain two data structures. The primary structure stores the features of the actual arrangement, as well as all the topological information connecting neighboring features. The secondary structure, a *layered dag*[2], allows us to take a new query point, and efficiently determine which cell of the arrangement contains it. Our algorithm for incrementally inserting a new polygon uses the point location query to locate one of the endpoints, and then uses a "travel and split" approach to insert an edge, while updating the data structures. Once we reach the other endpoint, we continue with the next edge of the polygon.

While inserting a new edge, if we intersect an existing vertex, we break the new edge into two pieces at this vertex. If we intersect an existing edge, we insert a new vertex at this intersection point, and we break both edges into two pieces at this new vertex. The vertical threads incident to each edge do not create vertices or break the edge. Instead, each edge maintains an ordered list of incident threads. These lists are used to cross from one side of an edge to the other.

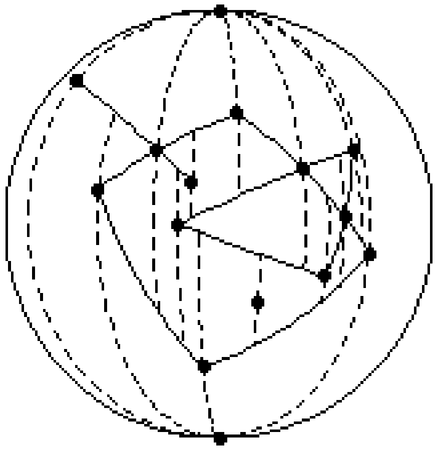

Figure 1: A Spherical Arrangement

4 Spherical Arrangements

Maintaining arrangements on a sphere versus on a plane seems like a very different task at the implementation level. However, they can both be handled by the identical package with a slight bit of care. The key is that there is an exact correspondence between the spherical coordinate system and the double-sided projective plane by viewing the coordinates (x, y, z) as homogeneous with weight z. A good explanation is given by Stolfi in [6].

To handle the sphere, we break it into two hemispheres using a "great meridian," M. Then we maintain the arrangement on both hemispheres separately, yet in a way so that the features are *glued* properly at M. On each hemisphere, concepts such as "vertical", "left", and "above" can be defined. To define *vertical*, we choose a pair of antipodes lying on the meridian and consider these to be the top and bottom poles. The set of vertical threads corresponds to the pencil of half circles passing through these poles.

5 Experimental Results

This software was developed in C for both a Decstation 3100 as well as a Dec Alpha workstation. It is currently used successfully in the STAAT assembly planning project. The package has been used there to construct an arrangement with over 1600 vertices, 3800 edges, and 5300 trapezoids. Such an arrangement was constructed in just under 2 seconds on a Dec Alpha using floating point calculations.

6 Source Code Available

This software is written in C and is available via ftp at `flamingo.stanford.edu` in the directory `/pub/wass/arrangement/`.

Acknowledgements: The author wishes to thank Dan Halperin, Leo Guibas, Cyprien Godard, and G. D. Ramkumar for their helpful discussions. Also, extra thanks to Bruce Romney for reviewing thousands of lines of source code.

References

[1] B. Chazelle and J. Incerpi. Triangulation and shape-complexity. *ACM Trans. Graph.*, 3(2):135–152, 1984.

[2] H. Edelsbrunner, L. J. Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM J. Comput.*, 15:317–340, 1986.

[3] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms.* Prentice Hall, New York, 1993.

[4] B. Romney, C. Godard, M. Goldwasser, and G. Ramkumar. An efficient system for analyzing assembly complexity. Manuscript, Dept. Comput. Sci., Stanford Univ., Stanford, CA, 1994.

[5] R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom. Theory Appl.*, 1:51–64, 1991.

[6] J. Stolfi. *Oriented Projective Geometry: A Framework for Geometric Computations.* Academic Press, 1991.