Degree-driven Geometric Algorithm Design

The Challenge

We usually think of computers as good at mathematical calculations but, in fact, they often allow a small amount of error so that they can go fast: 1/3 might be stored as just the first 16 digits of 0.3333…, , so that 3*(1/3) < 1.

This can cause problems when numerical calculations are used to figure out geometric relationships: rounding the coordinates of the point q actually takes it off the lines ac and bd that define it!
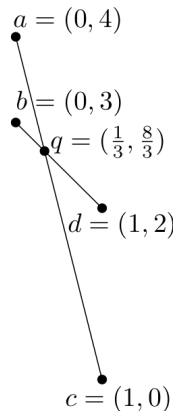
You can sometimes see this in video games – your character might be able to put a hand inside a wall because the computer is allowing a small error. And while you may enjoy the possibility to "cheat" in a video game, you would not want the software in an airplane or a surgical robot to violate the laws of nature in this way.

$a = (0, 4)$

$b = (0, 3)$

$q = (\frac{1}{3}, \frac{8}{3})$

$d = (1, 2)$

$c = (1, 0)$

Yet most techniques to avoid these errors require special programming and slow down both the development and execution of computer programs.

The Approach

Algorithm designers try to minimize use of resources of time and memory space. Our work considers arithmetic precision as another resource, and minimizes the degree of polynomials used in the geometric tests or *predicates* that are applied.

For example, consider the problem of overlaying maps of roads and county boundaries. If the input is 2D points with b bit coordinates, then testing if two line segments intersect is degree 2 (double precision) but actually computing the intersection is a rational polynomial with degree 3 over degree 2. Some algorithms that compute intersections also sort them by x coordinate, which takes degree 5, or five-fold precision. (Since computer hardware usually provides fast double precision only, it is not surprising that occasional errors occur when you need quintuple precision!)

Developing fast algorithms with limited precision requires creativity. If you restrict yourself to degree two, you can determine whether two lines intersect, but you learn very little about where the intersection actually occurs, as in Figure 1.

Treating precision as a limited resource brings to our attention the high cost of sophisticated geometric operations. Moreover, it allows us to better guarantee that implementations of our algorithms are not only efficient, but also correct.

Highlights

- **Design and analysis of algorithms to optimize not only the traditional metrics of running time and memory space, but also arithmetic precision required.**
- **Near-optimal time, optimal precision algorithms for computing a Voronoi diagram.**
- **Optimal time and precision algorithm for computing a distance transform of an image.**
- **Optimal algorithms for polygon overlay and Boolean operations.**
- **The limitation of restricting precision forces creative new solutions of classic problems that have practical applications in CAD, image processing, GIS, molecular biology, and more.**

The results

Red and Blue Line Segment Intersection. The map overlay problem (and the related problems of polygon clipping in graphics and 2D Boolean operations in computer-aided design) can be abstracted to the problem of building an arrangement from a set of red and blue line segments that have no red/red or blue/blue intersections.
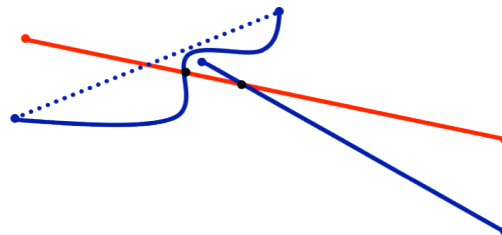

Figure 1: With double precision segments defined by endpoints, we cannot test the precise location of an intersection, much less sort intersection points by *x*-coordinate. The restricted precision means that we may as well think of segments as curvy "spaghetti."

The classical approach to solving the problem is: first, identify all red/blue segment intersections; second, sort the intersection points; third build the arrangement. However, comparing intersection points of segments specified by their end points requires 5-fold precision. Our group solved the red and blue segment intersection problem optimally in time, space and precision with an algorithm that takes only double precision.
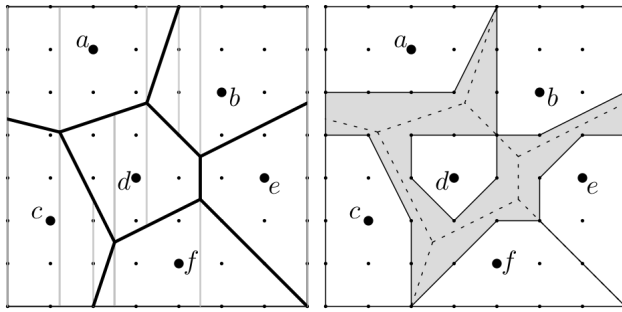
Figure 2: *Left*: The trapezoidation, in grey, of a Voronoi diagram, in black, of six sites. *Right*: The Voronoi Polygon Set, shown as white polygons, is a double precision representation of the grid points in each Voronoi cell. To recover the Voronoi diagram's precise structure in the grey gaps requires more than double precision.

Figure 3: *Left*: Each white polygon of the Voronoi Polygon Set can be represented by an orange *proxy segment*. **Right**: Our proximity query data structure is computable with only double precision. The trapezoids shaded in light blue contain all the grid points for which *f* is the closest site.

Proximity Queries and Voronoi diagrams. Proximity query structures efficiently answer the question, "given a set of point *sites* in the plane and a query point, what is the closest site?" Generalizations of these queries arise in fields ranging from geographic information systems to economics to physics.

The classic proximity query structure is built from the Voronoi diagram, which partitions the plane into maximally connected regions with the same set of closest sites. As Figure 2 shows, this diagram is further broken down into trapezoids, which are stored in a data structure that can be searched efficiently.

Unfortunately, a straightforward construction requires 6-fold precision. It has been known since 1999 that double precision query structures answering all single precision queries was possible, but it was not known how to build such a structure with less than 4-fold precision. Our group provided a new diagram, shown in Figure 3, that can be constructed with only double precision.

Members
Jack Snoeyink, Professor
David Millman, Graduate Research Assistant

Research Sponsors
Bettis Atomic Power Laboratory

Keywords
Computational Geometry; Algorithms; Robust geometric computation; Low-degree primitives; Segment intersection; Voronoi diagram, Post-office problem; Distance transform
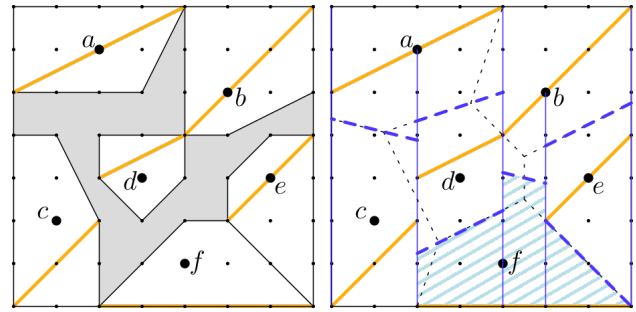
Recent Publications
D. L. Millman and J. Snoeyink. Computing planar Voronoi diagrams in double precision: A further example of degree-driven algorithm design. To appear in SCG '10: Proceedings of the Twenty-sixth annual Symposium on Computational Geometry 2010.

T. M. Chan, D. L. Millman, and J. Snoeyink. Discrete Voronoi diagrams and post office query structures without the incircle predicate. In FWCG '09: Proceedings of the Nineteenth Annual Fall Workshop on Computational Geometry, pages 33–34, 2009. electronic proceedings.

D. L. Millman and J. Snoeyink. Computing the implicit Voronoi diagram in triple precision. In, WADS 2009: Proceedings of the Eleventh International Symposium on Algorithms and Data Structures, Volume 5664 of Lecture Notes in Computer Science, pages 495–506. Springer, 2009.

For More Information
Dr. Jack Snoeyink
Phone (919) 962-1969
E-mail: snoeyink@cs.unc.edu