ORAL EXAM: DEGREE-DRIVEN GEOMETRIC ALGORITHM DESIGN

DAVID L. MILLMAN

Recall that my thesis investigates adding precision as an optimization criterion for the design and analysis of geometric algorithms. Classically, we analyze an algorithm's precision by bounding the extent to which a small perturbation of the input changes the output. This definition is natural for algorithms that model real valued functions, but what about geometric algorithms that produce combinatorial structures?

Liotta, Preparata, and Tamassia, suggested that we analyze the precision of a geometric algorithm in terms of the algebraic degree of its predicates, calling it *degree-driven algorithm design*. I seek to investigate and implement degree-driven algorithms for Voronoi diagrams, and I would like to organize the implementations into a kernel for a degree-driven algorithm library.

Precision analysis helps us classify the conditions under which an algorithm's implementation will be "correct". As demonstrated below, the term "correct" has different meanings in different contexts and to different communities. I provide a sample of papers from GPU computing, computational geometry, and CSG that highlight notions of correctness, common sophisticated techniques for ensuring it and their implementations.

1. Examples of the Hazards of Blindly Apply Floating Point Arithmetic

I begin with a paper on simple examples that warn us of the potential hazards of blindly using floating point arithmetic.

[KMP08] L. Kettner, K. Mehlhorn, S. Pion, S. Schirra and C. Yap Classroom examples of robustness problems in geometric computations *Computational Geometry*, 40(1): 61–78, 2008

2. Correctness Frameworks

Three complimentary correctness frameworks are discussed in the CSG and computational geometry literature.

- Yap, observed that geometric algorithms are a combination of combinatorial tests and numerical predicates. He proposed a framework in which numerical predicates are evaluated exactly, thus, making numerical error a non-issue. I achieve Yap's framework with static analysis carried out during an algorithm's design.
- Suighara *et al.* proposed a framework for topological correctness, in which some set of topological invariants are maintained through a construction. Should ambiguity arise, the branch most likely to be correct, is chosen. Sugihara *et al.*'s discussion of data representations will play an important role in the thesis.
- Liotta *et al.* proposed a framework for analyzing the precision of a geometric algorithm. I use this framework though out the thesis.

[Y97] C.-K. Yap. Towards exact geometric computation. *Comput. Geom. Theory Appl.*, 7(1-2):3–23, 1997.

[SII00] K. Sugihara, M. Iri, H. Inagaki, and T. Imai. Topology-Oriented Implementation—An Approach to Robust Geometric Algorithms. *Algorithmica*, 27(1):5–20, 2000.

Date: April 18, 2011.

Department of Computer Science, University of North Carolina at Chapel Hill
{dave}@cs.unc.edu

[LPT99] G. Liotta, F. P. Preparata, and R. Tamassia. Robust Proximity Queries: An Illustration of Degree-Driven Algorithm Design. *SIAM J. Comput.*, 28(3):864–889, 1999.

2.1. **Examples of Exact Geometric Computation.** Many have implemented the exact geometric computation framework. This section's papers highlight two implementations.

[ABD97] F. Avnaim, J. Boissonnat, O. Devillers, F. Preparata, and M. Yvinec. Evaluating signs of determinants using single-precision arithmetic. *Algorithmica*, 17(2):111–132, 1997.

[BBP01] H. Brönnimann, C. Burnikel, and S. Pion. Interval arithmetic yields efficient dynamic filters for computational geometry. *Discrete Appl. Math.*, 109(1-2):25–47, Apr. 2001.

2.2. **Example of Snap Rounding.** When we use EGC, we may need to round an internal representation of a geometric construction for output. Snap rounding is a popular choice for outputting a construction. Degree-driven analysis provides us with a static bounds on the precision of a construction and uses ideas from snap rounding in the algorithm design process. For example, a Voronoi Polygon is a snap-rounded representation of a Voronoi cell.

[dBHO07] M. de Berg, D. Halperin, and M. Overmars An intersection-sensitive algorithm for snap rounding. *Computational Geometry*, 36(3):159–165, Apr. 2007.

2.3. **Examples of Degree-driven Geometric Algorithm Design.** Aside from the paper that introduces LPT's framework, I highlight Bossonnat and Preparata's paper, which considers segment intersection problems in the plane.

[BP00] J.-D. Boissonnat and F. P. Preparata. Robust Plane Sweep for Intersecting Segments. *SIAM J. Comput.*, 29(5):1401–1421, 2000.

3. Predicates

The two chapters from Schneider and Eberly describe representations for geometric primitives, and predicates and constructions on these representations. In the algebraic decision tree model of computation, commonly used in computational geometry, we evaluate ternary predicates that evaluate to zero, positive, or negative. In practice, the zero evaluations are difficult to handle, and perturbation methods, such as the one described by Seidel, are commonly employed.

[PE03a] P. Schneider and D. Eberly. Chapter 5: Geometric Primitives in 2D, pages 171–188. *Geometric Tools for Computer Graphics*, 2003.

[PE03b] P. Schneider and D. Eberly. *Chapter 7: Intersection in 2D*, pages 241–284. *Geometric Tools for Computer Graphics*, 2003.

[R98] R. Seidel, The Nature and Meaning of Perturbations in Geometric Computing. *Discrete and Computational Geometry*, 19(1), 1-17. Springer, 1998.

4. CSG

CSG has a wide range of definitions for a correct construction. Below, we see three of the strongest definitions, listed in increasing order of exactness. Varadhan *et al.* consider homotopy, Plantinga and Vegter consider isotopy and Bernstein and Fussell consider exact constructions.

[VKS04] G. Varadhan, S. Krishnan, T. Sriram, and D. Manocha, Topology preserving surface extraction using adaptive subdivision *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 235–244, ACM, 2004.

[PV06] S. Plantinga and G. Vegter. Isotopic meshing of implicit surfaces. *The Visual Computer*, 23(1):45–58, Oct. 2006.

[BF09] G. Bernstein and D. Fussell. Fast, Exact, Linear Booleans. *Computer Graphics Forum*, 28(5):1269–1278, July 2009.

5. GPU

Many GPU algorithms rely on floating point computations and define an implementation as correct if it terminates and produces an output close to the desired result. The early work of Hoff *et al.* observed that their output's correctness was determined by the resolution of the depth buffer and discuss the errors that can occur in in their final result. Krishnamurthy and McManis consider their problem from the stand point of numerical analysis by investigating 1-, 2- and 3- point Gaussian quadrature schemes and analyze the errors introduced by their method. Cao *et al.* do not discuss numerical error. In personal discussions, they reported that they have yet to detect any errors in their implementation, however, we have constructed degenerate examples that break their perturbation method.

[HKL99] K. E. Hoff III, J. Keyser, M. Lin, D. Manocha, and T. Culver. Fast computation of generalized Voronoi diagrams using graphics hardware. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 277–286, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[KM10] A. Krishnamurthy and S. McMains. *Accurate moment computation using the GPU*. ACM Press, New York, New York, USA, Sept. 2010.

[CTM10] T.-T. Cao, K. Tang, A. Mohamed, and T.-S. Tan. Parallel Banding Algorithm to Compute Exact Distance Transform with the GPU. In *I3D '10: Proceedings of the 2010 symposium on Interactive 3D graphics and games*, New York, NY, USA, 2010. ACM.

6. Kernel Design

One of my goals is to implement the low precision algorithms described in my thesis and to organize the implementations into a kernel for a degree-driven algorithm library. Below are papers relevant to the design of some successful kernels.

[SL95] A. Stepanov, and M. Lee. The Standard Template Library. Technical Report Hewlett-Packard, 1995.

[FGL96] A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schönherr. The CGAL kernel: A basis for geometric computation. 1148:191–202–202, 1996.

[KPY99] V. Karamcheti, C. Li, I. Pechtchanski, and C. Yap. A core library for robust numeric and geometric computation. In *SCG '99: Proceedings of the fifteenth annual symposium on Computational geometry*, pages 351–359, New York, NY, USA, 1999. ACM.