19th Annual Fall Workshop on Computational Geometry

and

CRA-W/CDC Workshop on Computational Geometry

Dept. of Computer Science • November 13-15, 2009 • Medford, MA

Forward

The 19th Fall Workshop on Computational Geometry was held at Tufts University, Medford, MA, on November 13-14, 2009. It was followed by a one-day tutorial day on Computational Geometry, held November 15, 2009. The first event was sponsored by the National Science Foundation, while the second event was supported by CRA-W/CDC. Supplementary funding was provided by Tufts University. These generous grants made the workshops possible. The Fall Workshop followed the tradition of being an annual forum for the presentation, discussion, and dissemination of new results, as well as free exchange of questions and research challenges. The tutorial day was designed to encourage the success of women and members of underrepresented groups, aiming at helping these people become interested in and knowledgeable about the paradigms and research of Computational Geometry. These proceedings contain the technical papers accepted for presentation at the Fall Workshop, as well as abstracts of the lectures given on the tutorial day.

The 35 contributed talks given at the Fall Workshop covered a broad range of topics, from theoretical to practical aspects of Computational Geometry, and related fields, e.g., Discrete Geometry, Computer Graphics, GPU usage, Sensor Networks, Graph Drawing, etc. In addition, the workshop included five invited talks delivered by researchers in academia and industry, and the traditional open-problems session. The eight lectures of the tutorial day were given by experts in Discrete and Computational Geometry, and covered a broad spectrum of the field.

A lot of work went into forming this program and producing these proceedings. The program committee of the Fall Workshop reviewed the submissions on a tight schedule. The committee needed to handle an unexpected flood of submissions, and had to make some difficult choices, being unable to accept all of the exciting submissions. We would like to thank all the authors who submitted papers to the workshop. They provided the material that made it possible to assemble such a distinguished technical program. Thanks also go to the speakers in the tutorial day, which made the creation of an attractive program possible.

Gill Barequet
Technion and Tufts Univerity

Program Committee of FWCG'09

- Esther M. Arkin (SUNY Stony Brook)
- Gill Barequet (Technion and Tufts University, chair)
- Lenore J. Cowen (Tufts University)
- R.L. Scot Drysdale (Dartmouth College)
- Audrey Lee-St. John (Mt. Holyoke College)
- Anna Lubiw (Univ. of Waterloo and MIT)
- Joseph S.B. Mitchell (SUNY Stony Brook)
- Diane L. Souvaine (Tufts University)
- Csaba D. Tóth (Univ. of Calgary and Tufts University)
- Godfried T. Toussaint (McGill University and Harvard)

Local Organization

- Gill Barequet (Technion and Tufts University)
- Mashhood Ishaque (Tufts University)
- Diane L. Souvaine (Tufts University)
- Jeannine L. Vangelist (Tufts University)

Technical Support

- John Sotherland (Tufts University)

Fall Workshop on Computational Geometry (11/13-14)

Friday 11/13

9:00 Opening

1st session: Approximation (chair: Anna Lubiw):

9:05 *Catalin Constantin, Shawn Brown, and Jack Snoeyink*
Streaming Simplification of Labeled Meshes Using Quadric Error Metrics

9:20 *David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu*
A Practical Approximation Algorithm for the LTS Estimator

9:35 *Leonidas Guibas, Nikola Milosavljević, and Arik Motskin*
Connected Dominating Sets on Geometric Graphs with Mobile Nodes

9:50 *Gary L. Miller, Todd Phillips, and Donald R. Sheehy*
Approximating Voronoi Diagrams with Voronoi Diagrams

10:05 *Dengpan Zhou and Jie Gao*
Maintaining Approximate Minimum Steiner Tree and *k*-center for Mobile Agents in a Sensor Network

10:20 Break

2nd session: Invited Talk (chair: Jack Snoeyink):

10:35 *Hanspeter Pfister* (Harvard University)
The Connectome - Discovering the Wiring Diagram of the Brain

11:20 Break

3rd session: Representations, Data Structures, and Operators (chair: Godfried T. Toussaint):

11:35 *Benoît Hudson*
Succinct Representation of Well-Spaced Point Clouds

11:50 *Sorelle A. Friedler and David M. Mount*
Spatio-temporal Range Searching over Compressed Kinetic Sensor Data

12:05 *Danny Z. Chen and Haitao Wang*
Dynamic Data Structures for Simplicial Thickness Queries
12:20 *Timothy M. Chan, David L. Millman, and Jack Snoeyink*
Discrete Voronoi Diagrams and Post Office Query Structures without the InCircle Predicate

12:35 Short break

12:45 Lunch and Open Problems (chair: Joseph S.B. Mitchell)

13:35 Short break

4th session: Discrete Geometry (chair: Diane L. Souvaine):

13:45 *Ronnie Barequet, Gill Barequet, and Günter Rote*
Formulae and Growth Rates of High-Dimensional Polycubes
14:00 *William Steiger, Mario Szegedy, and Jihui Zhao*
Six-way Equipartitioning by Three Lines in the Plane
14:15 *Mohammad Irfan, Justin Iwerks, Joondong Kim, and Joseph S.B. Mitchell*
Guarding Polyominoes
14:30 *Karin Arikushi and Csaba D. Tóth*
Drawing Graphs with 90° Crossings and at Most 1 or 2 Bends per Edge
14:45 *Javier Cano, Csaba D. Tóth, and Jorge Urrutia*
A Tight Bound For Point Guards in Piece-Wise Convex Art Galleries

15:00 Break

5th session: Invited Talk (chair: Gill Barequet):

15:15 *Dan A. Klain* (University of Massachusetts Lowell)
If You Can Hide Behind It, Can You Hide Inside It?

16:00 Break

6th session: Mesh Generation and Sensor Networks (chair: Csaba D. Tóth):

16:15 *Benoît Hudson, Gary L. Miller, Steve Y. Oudot, and Donald R. Sheehy*
Mesh-Enhanced Persistent Homology
16:30 *Gary L. Miller and Todd Phillips*
Fast Meshing for Acute PLCs
16:45 *Rik Sarkar, Wei Zeng, Xiaotian Yin, Jie Gao, Feng Luo, and Xianfeng Gu*
Greedy Routing with Guaranteed Delivery Using Ricci Flows

17:00 *Wei Zeng, Rik Sarkar, Feng Luo, Xianfeng Gu, and Jie Gao*
Resilient Routing for Sensor Networks Using Hyperbolic Embedding of Universal Covering Space

17:15 Break

18:15 Reception at Hyatt Place

Saturday 11/14

7th session: Invited Talk (chair: Nancy M. Amato):

9:00 *Chris Bishop* (State Univ. of NY at Stony Brook)
Conformal Mapping in Linear Time

9:45 Break

8th session: Optimization (chair: Lenore J. Cowen):

10:00 *Petter Brass*
Computing the Largest Bounded Intersection of Halfspaces

10:15 *Joseph S.B. Mitchell and Eli Packer*
On Non-crossing (Projected) Spanning Trees of 3D Point Sets

10:30 *Steven Bitner, Yam K. Cheung, Atlas F. Cook IV, Ovidiu Daescu, Anastasia Kurdia, and Carola Wenk*
Visiting Points with a Bevel-Tip Needle

10:45 *Esther M. Arkin, Irina Kostitsyna, Joseph S.B. Mitchell, Valentin Polishchuk, and Girishkumar R. Sabhnani*
The Districting Problem

11:00 *Steven Bitner, Yam-Ki Cheung, and Ovidiu Daescu*
On the Minimum Color Separation Circle

11:15 *Esther M. Arkin, Sándor P. Fekete, Joondong Kim, Joseph S.B. Mitchell, Girishkumar R. Sabhnani, and Jingyu Zou*
The Pencil Packing Problem

11:30 *Umut A. Acar, Benoît Hudson, and Duru Türkoglu*
A Dynamic Algorithm for Well-Spaced Point Sets

11:45 Break

9th session: Invited Talk (chair: Joseph O'Rourke):

12:00 *Brigitte Servatius* (Worcester Polytechnic Institute)
_k_-Plane Matroids and Whiteley's Flatenning Conjectures

12:45 Lunch break

10th session: Metrics, Spaces, and Embeddings (chair: Esther M. Arkin):

14:00 *Afra Zomorodian*
Fast Construction of the Vietoris-Rips Complex
14:15 *Atlas F. Cook IV, Jessica Sherette, and Carola Wenk*
Computing the Fréchet Distance Between Polyhedral Surfaces with Acyclic Dual Graphs
14:30 *Joondong Kim, Joseph S.B. Mitchell, and Jingyu Zou*
Routing Parallel and Merging Lanes
14:45 *Anand Kulkarni and Sandeep Koranne*
The Gemcutting Approach to the Hirsch Conjecture (and Other Open Problems on Polytopal Graphs)
15:00 *P. Thomas Fletcher, John Moeller, Jeff M. Phillips, and Suresh Venkatasubarmanian*
Computing Hulls In Positive Definite Space

15:15 Break

11th session: Invited Talk (chair: Audrey Lee-St. John):

15:30 *Kirk Haller* (Dassault Systemes SolidWorks Corporation)
Computational Geometry Problems in CAD

16:15 Break

12th session: Graphics and Applications (chair: Joseph S.B. Mitchell):

16:30 *Marta Fort, J. Antoni Sellarès, and Nacho Valladares*
Reporting Flock Patterns via GPU
16:45 *Shu Ye and Karen Daniels*
Triangle-based Prism Mesh Generation on Interconnect Models for Electromagnetic Simulations
17:00 *Sarang Joshi, Raj Varma Kommaraju, Jeff M. Phillips, and Suresh Venkatasubramanian*
Matching Shapes Using the Current Distance

17:15 *Christopher S. Stuetzle, Zhongxian Chen, Katrina Perez, Jared Gross, Barbara Cutler, W. Randolph Franklin, and Thomas Zimmie*
Segmented Height Field and Smoothed Particle Hydrodynamics in Erosion Simulation

17:30 *Sergey Bereg*
Orthogonal Morphing of Orthogonal Drawings

17:45 Adjourn

Abstracts of the

Fall Workshop on Computational Geometry

Streaming Simplification of Labeled Meshes Using Quadric Error Metrics

Catalin Constantin Shawn Brown Jack Snoeyink
Department of Computer Science, University of North Carolina at Chapel Hill

Abstract We present a *streaming simplification* algorithm for labeled meshes based on edge contraction under the Quadric Error Metric of Garland and Heckbert. Given as input a *spatially-finalized* streaming mesh, whose vertices are labeled (with object ID, ground vs. building, or some discrimination between parts of the mesh that is to be preserved during simplification), our algorithm achieves fast, high-quality simplification of gigabyte datasets using a small memory footprint.

1 Introduction

LIDAR collects high density point clouds from vast geographic regions. In triangulated form [1], this data enables important computations such as shortest paths, visibility line-of-sight, and object identification. Because many of these algorithms have different data resolution requirements, mesh simplification is often a standard preprocessing step. We address the need for fast and accurate simplification of gigabyte size meshes by iterative *edge contractions* [2] performed on an active processing buffer, stored as a streaming the Quad Edge [5] data structure.

Streaming processing [3] is effective on large data sets that exhibit high spatial coherence. Simplification while streaming imposes additional limitations. Because it is performed in a small active buffer whose border contains locked edges (connecting to written or unread vertices), some simplifications must be set aside in a waiting buffer. When the allocated memory fills up, we must write some of these locked mesh elements to the output stream.

Our algorithm scales to arbitrarily large meshes; the buffer allocation precisely determines the runtime memory footprint. It successfully runs on commodity laptops as well as on powerful processing workstations. With good engineering, we strike a balance between simplification quality and time, given the memory constraints.

2 Simplification Policies

A simplification policy determines to what point an edge may contract to, and a priority for *eligible* contractions—those that leave the mesh data structure in a consistent state. Most policies used in *global simplification* carry over naturally to the *streaming simplification.* We have adopted *subset placement* [2] for the contraction point and aggregate vertex quadrics by addition. Also, we use the *constraint quadrics* described by Garland et al. to enforce inter-object boundaries as well as true mesh boundaries.

Whereas *Global simplification* [2, 4] begins only after the whole mesh has been initialized, *Streaming simplification* runs continuously as we pass through the file. The active buffer contains output and input border elements which cannot be changed—either they have been written to the output stream, or their neighbors are still to come in the input stream—so we use the following stricter conditions to decide if an edge (u, v) is eligible:

1. Labels of u and v must be the same. We have used ground vs. non-ground labeling in our tests.
2. The *left* and *right* faces of the edge (u, v) contain no written vertices and no unfinalized vertices (implying that u and v are also both finalized).
3. Static restrictions on the edges (i.e. length) may affect eligibility.

Eligible edges are kept in a priority queue.

In global simplification, the main condition for eligibility is whether an edge contraction would cause undesirable topology changes such as degenerate faces and mesh inversions. We actually allow such edges in the priority queue, since forbidding them interacted with the above conditions and significantly worsened simplification quality. We use a *suspend* operation to prevent contraction of such edges, and an efficient, trigger-based check of whether suspended edges can resume their place in the priority queue.

Model	Buffer	Footprint	seconds @ simplified rate			
			25%	10%	5%	1%
platform 200K verts 7MB	GLOB	335MB	2.62	2.93	3.03	3.08
	10K	16MB	1.68	1.70	1.74	1.74
	50K	47MB	1.98	2.29	2.35	2.36
	100K	68MB	2.03	2.51	2.76	2.76
	250K	200MB	2.20	2.62	3.01	3.04
citymesh 995K verts 36MB	GLOB	1.45GB	15.40	16.90	17.40	17.80
	50K	47MB	2.31	2.39	2.40	2.41
	100K	68MB	2.58	2.75	2.76	2.77
	250K	200MB	2.67	2.95	3.01	3.03
	1M	600MB	2.70	3.00	3.06	3.11
ottawasq 14.25M verts 494MB	GLOB	—	–	–	–	–
	50K	47MB	149	151	153	155
	100K	68MB	177	179	181	183
	250K	200MB	202	209	211	216
	1M	600MB	264	265	267	271
ottawa.smb 65M verts 2.3GMB	GLOB	—	–	–	–	–
	50K	47MB	723	730	731	732
	100K	68MB	821	848	849	850
	250K	200MB	930	993	993	995
	1M	600MB	1139	1211	1247	1248

We find that a simple retry queue mechanism keeps the running time low.

The key aspect of streaming is the ability to pipe output from one application as input to the next, ideally by using a single pass through the data. This requires maintaining finalization and spatial locality of the mesh; refinalization after each processing stage is slow for large and highly disordered meshes.

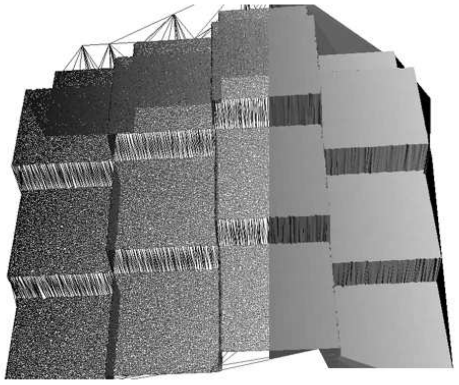
We transfer the same spatial locality and finalization properties from the input to the output with no additional costs. Each triangle in the active Quad Edge mesh is linked in a FIFO (age based) queue. As simplification progresses, edge contractions occur at various unrelated places in the Quad Edge mesh. Each edge contraction causes up to two triangles to disappear from the mesh; they also get spliced out from the queue. Decisions to commit some data to the output stream will write out the triangles that lasted longest in the queue, along with proper finalization tags. This model allows simplification decisions be independent of the streaming constraints.

3 Results

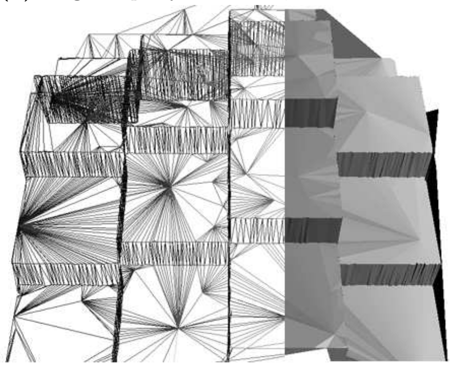
We obtain a throughput of approximately 1.8MB/s (50K vertices/s) for a buffer size of 1M edges. The simplification rate does not have a major impact in timing performance. A higher simplification rate requires less computation but more I/O writing and vice versa. As the rate approaches 100% (no simplification), we do have a gain in performance, which means pure I/O is slightly faster.

Buffer size impact on performance is as expected, logarithmic, similar to a heap increase. The factor $\log b_1/\log b_2$ in performance change of (b_1, b_2 two given buffer sizes) is reflected in the results (i.e. for the last two rows of **Ottawa** we have a factor of $1.24 \simeq \log 1M/\log 250K$).

Larger buffers improve the quality, but this depends on the system availability. Also, the modularity design of the streaming pipeline implies that some other streaming modules may be running at the same time and their memory allocation should be somewhat balanced.



(a) original *platform* mesh, 200K vertices



(b) simplified *platform* mesh, 2K vertices

References

[1] M. Isenburg, Y. Liu, J. Shewchuk, and J. Snoeyink, “Streaming computation of Delaunay triangulations,” in *SIGGRAPH*, pp. 1049–1056, 2006.

[2] M. Garland and P. S. Heckbert, “Surface simplification using quadric error metrics,” in *SIGGRAPH*, pp. 209–216, 1997.

[3] M. Isenburg and P. Lindstrom, “Streaming meshes,” *Proceedings of IEEE Visualization 2005*, pp. 231–238, 2005.

[4] H. Hoppe, “New quadric metric for simplifiying meshes with appearance attributes,” in *IEEE*, pp. 59–66, 1999.

[5] L. J. Guibas and J. Stolfi, “Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams,” in *ACM TOG*, pp. 74–123, ACM, 1985.

A Practical Approximation Algorithm for the LTS Estimator

David M. Mount* Nathan S. Netanyahu† Christine D. Piatko‡ Ruth Silverman§
Angela Y. Wu¶

In the 19th Annual Fall Workshop on Computational Geometry, November 2009

Fitting a model to a collection of points is a fundamental problem in computational geometry and computational statistics. In standard *linear regression* we are given a set P of n *observations*, where each observation consists of a sequence of independent variables and one dependent variable. The objective is to express the dependent variable as a linear function of the independent variables. The standard least squares linear estimator is defined to be the hyperplane that minimizes the sum of the squared residuals. It is well known that the least squares estimator is very sensitive to *outliers*, that is, points that fail to follow the linear pattern of the majority of the points. This has motivated interest in the study of *robust estimators*. The basic measure of the robustness of an estimator is its *breakdown point*, that is, the fraction (up to 50%) of outlying data points that can corrupt the estimator arbitrarily. The study of efficient algorithms for robust statistical estimators has been an active area of research in computational geometry. The most widely studied robust linear estimator is Rousseeuw's *least median of squares estimator* (LMS) [9], which is defined to be the hyperplane that minimizes the median squared residual. (In general, given a trimming parameter h, it minimizes the hth smallest squared residual.) A number of papers, both practical and theoretical, have been devoted to solving this problem in the plane and in higher dimensions [1–3, 5, 7, 8, 12].

The vast majority of work in the field of computational geometry on robust linear estimators has been devoted to the study of the LMS estimator. As observed by Rousseeuw and Leroy [10], however, this is not the robust linear estimator of choice because it has relatively low (statistical) asymptotic efficiency. Intuitively, the reason stems from the fact that, in dimension d, the LMS estimator is determined by just the $d + 1$ inliers having the largest squared residuals—the remaining inliers play essentially no role in the estimator's value. The preferred estimator is the *least trimmed squares* (or LTS) linear estimator [9]. Given an n-element point set P and a positive integer *trimming parameter* $h \leq n$, this estimator is defined to be the nonvertical hyperplane that minimizes the *sum* (as opposed to the maximum) of the h smallest squared residuals. Note that when $h = n$ this is equivalent to the well-known least squares estimator. Typically, h is roughly $n/2$, so the LTS fit can be thought of as the best least-squares fit determined by any majority of the points.

The computational complexity of LTS is less well understood than LMS. Hössjer [4] presented an exact $O(n^2 \log n)$ algorithm for LTS in the plane based on plane sweep. In an earlier paper, we showed that LTS could be computed in time $O(n^2)$ in the plane and in time $O(n^{d+1})$ in any fixed dimension $d \geq 3$ [6]. We also established an $\Omega(n^{d-1})$ lower bound on the time to compute any constant factor approximation for a closely related problem (where residuals are not squared), under the assumption of the hardness of affine degeneracy. The most practical approach is the Fast-LTS heuristic of Rousseeuw and Van Driessen [11]. In practice this approach works well, but it provides no assurances of the quality of the resulting fit.

*Department of Computer Science, University of Maryland, College Park, Maryland. Partially supported by the NSF under grant CCR-0635099 and the ONR under grant N00014-08-1-1015. Email: mount@cs.umd.edu.

†Department of Computer Science, Bar-Ilan University, Ramat-Gan 52900, Israel and Center for Automation Research, University of Maryland, College Park, Maryland. Email: nathan@{macs.biu.ac.il,cfar.umd.edu}.

‡Johns Hopkins University Applied Physics Laboratory and Johns Hopkins University Human Language Technology Center of Excellence, Baltimore, Maryland. Email: christine.piatko@jhuapl.edu.

§Center for Automation Research, University of Maryland, College Park, Maryland. Email: ruthsilverman@verizon.net.

¶Department of Computer Science, American University, Washington, DC. Email: awu@american.edu.

This raises the question of whether it is possible to bridge the gap between widely-used heuristics like Fast-LTS, which provide no theoretical guarantees, and provably correct algorithms that have high worst-case asymptotic complexity and/or are too complicated to merit practical implementation. In this work we present an algorithm that is *provably correct* (to within a user-supplied approximation bound), *simple* enough to implement, and is of *comparable efficiency* to the aforementioned heuristics on well-conditioned data sets.

We present a number of results in the full version of this paper. First, we present an analysis of a simple randomized $O(n)$-time heuristic, which is based on fitting a hyperplane to a random sample of d points. We introduce a parameterized model of *well-conditioned point sets* for multidimensional linear regression. We show that with high probability this heuristic provides a constant factor approximation to LTS, under the assumption that the point set is sufficiently well conditioned. Second, we propose an *data-sensitive* approximation algorithm, which is based on a geometric branch-and-bound search of the space of hyperplane coefficients. Unlike existing heuristics, this algorithm produces an approximation of guaranteed quality, no matter how badly the data are distributed. Finally, we have implemented this algorithm and present empirical evidence that it runs efficiently when presented with well-conditioned point sets.

References

[1] T. Bernholt. Computing the least median of squares estimator in time $o(n^d)$. In *Proc. Intl. Conf. on Computational Science and its Applications*, volume 3480 of *Springer LNCS*, pages 697–706, Berlin, Germany, 2005. Springer-Verlag.

[2] H. Edelsbrunner and D. L. Souvaine. Computing median-of-squares regression lines and guided topological sweep. *J. Amer. Statist. Assoc.*, 85:115–119, 1990.

[3] J. Erickson, S. Har-Peled, and D. M. Mount. On the least median square problem. *Discrete Comput. Geom.*, 36:593–607, 2006.

[4] O. Hössjer. Exact computation of the least trimmed squares estimate in simple linear regression. *Comput. Statist. Data Anal.*, 19:265–282, 1995.

[5] D. M. Mount, N. S. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu. Quantile approximation for robust statistical estimation and k-enclosing problems. *Internat. J. Comput. Geom. Appl.*, 10:593–608, 2000.

[6] D. M. Mount, N. S. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu. On the least trimmed squares estimator. Submitted for publication. Available at `http://www.cs.umd.edu/~mount/pubs.html`, 2007.

[7] D. M. Mount, N. S. Netanyahu, K. R. Romanik, R. Silverman, and A. Y. Yu. A practical approximation algorithm for the LMS line estimator. *Computational Statistics & Data Analysis*, 51:2461–2486, 2007.

[8] D. M. Mount, N. S. Netanyahu, and E. Zuck. Analyzing the number of samples required for an approximate monte-carlo LMS line estimator. In M. Hubert, G. Pison, A. Struyf, and S. Van Aelst, editors, *Theory and Applications of Recent Robust Methods*, Statistics for Industry and Technology, pages 207–219. Birkhauser, Basel, 2004.

[9] P. J. Rousseeuw. Least median-of-squares regression. *J. Amer. Statist. Assoc.*, 79:871–880, 1984.

[10] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection.* Wiley, New York, 1987.

[11] P. J. Rousseeuw and K. van Driessen. Computing LTS regression for large data sets. *Data Min. Knowl. Discov.*, 12:29–45, 2006.

[12] D. L. Souvaine and J. M. Steele. Time- and space- efficient algorithms for least median of squares regression. *J. Amer. Statist. Assoc.*, 82:794–801, 1987.

Connected Dominating Sets on Geometric Graphs with Mobile Nodes

Leonidas Guibas
Stanford University
guibas@cs.stanford.edu

Nikola Milosavljević
Stanford University
nikolam@cs.stanford.edu

Arik Motskin
Stanford University
amotskin@stanford.edu

Abstract

We propose algorithms for efficiently maintaining a constant-approximate connected dominating set (CDS) for geometric graphs with mobile nodes. Assuming that nodes follow bounded degree algebraic trajectories, and that two nodes are adjacent in the graph iff they are within a fixed geometric distance, we show that an $O(1)$-approximate CDS can be maintained with $\widetilde{O}(n^2)$ work for graphs in $\mathbb{R}^1$ and $\mathbb{R}^2$, over the entire course of motion, as compared with $\Omega(n^3)$ work to maintain the optimal CDS. In our approach, we ensure that a topology change caused by node mobility only affects the CDS solution in a local neighborhood, and show that a small set of range queries is then sufficient to efficiently repair the CDS.

1 Introduction

For an undirected graph $G = (V, E)$, we say that $S \subseteq V$ is a *dominating set* of G if every node in $V \setminus S$ is adjacent to some node in S. If S also induces a connected subgraph of G, then S is a *connected dominating set (CDS)* of G. The problem of finding a CDS with the fewest vertices, or *minimum CDS (MCDS)*, in a given graph has been of interest for a long time because of its application in networking. If G represents a communication network, a small CDS S of G can be used to form a routing "backbone" [5]. The advantage of a CDS-based approach is that all nodes are close to some "backbone" node, and long distance communication is achieved within the sparser subgraph induced by the CDS; potentially expensive routing infrastructure is required only within the small set S.

The problem has also been explored in the special case of *ad-hoc wireless networks*, where *geometric graphs* are of particular interest [5, 9]. In a geometric graph, nodes are identified with points in a metric space (typically two-dimensional Euclidean space), and two nodes are connected if they are sufficiently close to each other. More recently, there has been a growing interest in ad-hoc networks whose nodes are *mobile*, causing the induced geometric graph to change over time. The topic of this paper is the problem of maintaining a small CDS in such dynamic graphs. The CDS may require repair when an edge is added or deleted, so the challenge is to maintain a CDS that is sufficiently sparse, while also minimizing the necessary repair work.

1.1 New Results

We show that maintaining the MCDS can be prohibitively expensive: for bounded degree algebraic motion, it can require $\Omega(n^3)$ work to maintain the optimal CDS over the entire course of motion (where n is the number of nodes).

Our main contribution is a pair of algorithms that efficiently maintain a constant-approximate CDS for geometric graphs in $\mathbb{R}^1$ and $\mathbb{R}^2$ with mobile nodes. In particular, we demonstrate that in the same bounded degree algebraic model of motion, we can maintain an $O(1)$-approximate CDS with $\widetilde{O}(n^2)$ work over the entire course of motion[1]. In addition, we analyze our algorithms (and the underlying data structures that they maintain) in the *kinetic data structure* framework [2], and prove that they are *responsive*, *local* and *compact*.

For geometric graphs in $\mathbb{R}^1$, the CDS that we maintain is an *alternating independent set*. In $\mathbb{R}^2$, the CDS that we maintain comprises a maximal independent set (MIS) of nodes – which is already a dominating set –

[1]For any function g, the notation $\widetilde{O}(g(n))$ is equivalent to $O(g(n) \log^c n)$, for a constant c.

along with a small set of paths chosen to make the MIS connected while retaining the constant approximation factor. While the idea of augmenting a MIS with a small set of connectors to form a CDS is not new [1], our paper is the first to address the challenge of efficiently selecting such a sparse structure from an arbitrarily complex underlying set of mobile nodes. Our construction ensures that graph topology changes affect the CDS solution only locally, while a small set of range queries is sufficient to repair the CDS.

All of the algorithms in this paper are *centralized*. As in the literature on maintaining approximate (but not necessarily connected) minimum dominating sets for mobile nodes [6, 8], the quantity of interest is the computational complexity of simulating the network of mobile nodes on a single processor.

1.2 Related Work

The MCDS problem is very well-studied, long known to be NP-complete [7] for general graphs, but also for unit-disk graphs [4], the context that we are considering. While difficult to approximate in general graphs [3], a polynomial-time approximation scheme for MCDS in static unit-disk graphs has been developed [3].

However, there has been relatively little work on maintaining a CDS in the presence of mobile nodes. For the related minimum dominating set (MDS) problem, Gao *et al.* [6] give a randomized hierarchical algorithm for maintaining an $O(1)$-approximation to the minimum dominating set, requiring $\widetilde{O}(n^2)$ cumulative work if nodes follow bounded degree algebraic motion. Similarly, Hershberger [8] offers a constant-approximate deterministic algorithm for maintaining a covering of nodes by axis-aligned unit boxes. Observe that we achieve similar approximation and cumulative work bounds as these solutions[2], but with the additional challenge that our structure remain *connected* over the course of motion, a key requirement for networking applications. Unlike Gao *et al.* [6], our solution is deterministic and non-hierarchical, but as in both previous MDS solutions, we work in the ℓ_∞ norm, so our solution comprises a covering of the mobile nodes by a connected set of axis-aligned unit boxes, each centered at a node.

Alzoubi *et al.* [1] maintain a constant approximate MCDS solution with mobile nodes, but a change in topology can require $\Omega(\Delta)$ work in repairs (where Δ is the maximum node degree), aggregating to potentially $\Omega(n^3)$ total work over the course of the entire pseudo-algebraic motion.

References

[1] Khaled M. Alzoubi, Peng-Jun Wan, and Ophir Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc)*, pages 157–164, New York, NY, USA, 2002. ACM.

[2] Julien Basch and Leonidas J. Guibas. Data structures for mobile data. *J. Algorithms*, 31(1):1–28, 1999.

[3] X. Cheng, X. Huang, D. Li, W. Wu, and D. Du. A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks. *Networks*, 42(4):202–208, 2003.

[4] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Math.*, 86(1-3):165–177, 1990.

[5] B. Das and V. Bharghavan. Routing in ad-hoc networks using minimum connected dominating sets. In *IEEE International Conference on Communications*, volume 1, pages 376–380, 1997.

[6] Jie Gao, Leonidas J. Guibas, John Hershberger, Li Zhang, and An Zhu. Discrete mobile centers. In *Proc. of the 17th ACM Symposium on Computational Geometry (SoCG)*, pages 188–196, 2001.

[7] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1979.

[8] John Hershberger. Smooth kinetic maintenance of clusters. In *Proc. of the 19th ACM Symposium on Computational Geometry (SoCG)*, pages 48–57, 2003.

[9] Jie Wu and Hailan Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM)*, pages 7–14, New York, NY, USA, 1999. ACM.

[2]We prove the approximation bounds for our CDS by showing its size is within a constant factor of the MDS size.

Approximating Voronoi Diagrams with Voronoi Diagrams

Gary L. Miller
glmiller@cs.cmu.edu

Todd Phillips
tp517@cs.cmu.edu

Donald R. Sheehy
dsheehy@cs.cmu.edu

1 Introduction

The tremendous usefulness of Voronoi diagrams is tempered by their worst-case $O(n^{\lceil d/2 \rceil})$ size blowup. This makes them an obvious target for approximation, and indeed, several methods have been proposed that produce linear size approximations to the Voronoi diagram supporting logarithmic-time approximate nearest neighbor queries. All such methods use quadtrees to approximate the Voronoi cells. But what if the input does not have a "bad" Voronoi diagram? There is a huge gap between the best-case and the worst case complexity. Sometimes, the exact solution is both simpler and more precise than an approximation (Figure 1).

We present a new method for constructing approximate Voronoi diagrams that uses the Voronoi diagram of a superset of the input as an approximation to the true Voronoi diagram. The approximate Voronoi cells are unions of Voronoi cells of the superset. If the input has a Voronoi diagram with good aspect ratio (and thus has linear size) then the approximate Voronoi diagram we produce will simply be the Voronoi diagram of the input. The size of the diagram is $O(n \log \Delta)$ where Δ is the *spread* of the input (the ratio of largest to smallest interpoint distances). Moreover, it supports approximate nearest neighbor queries in time $O(\log \Delta)$. We also discuss methods for eliminating the dependence on the spread in the size and reducing it to $O(\log n)$ for queries. The construction will be based on sparse meshing technology [4].

Formally, a $(1+\varepsilon)$-*approximate Voronoi diagram* of an input set $N \in \mathbb{R}^d$ is a spatial decomposition of $\mathbb{R}^d$ into $n = |N|$ pieces with the property that exactly one input point p_i lies in each piece, V_i, and for each $x \in V_i$, $|x - p_i| \leq (1+\varepsilon)|x - p_j|$ for all $p_j \in N$.

In related work, Har-Peled introduced a method based on a quadtree construction in which the approximate Voronoi cells are unions and differences of quadtree cells[3]. The nearest neighbor search reduces to searching the quadtree and assigning a near input point to each quadtree cell. This approach was later refined by Sabharwal et al. [7]. More quadtree based methods were presented by Arya and Malamatos [1] and again by Arya et al. [2].

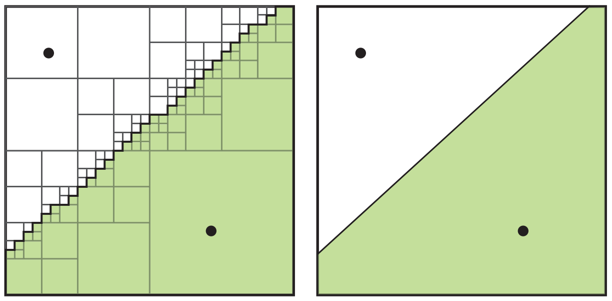


Figure 1: Sometimes, an approximation is not necessary. Here, the diagram on the right achieves both greater simplicity and greater precision.

2 History DAGs

The history DAG is a point location structure that models the incremental changes to a cell complex. We focus on history DAGs over the sequence of Voronoi Diagrams induced by prefixes $P_i \subset N$, where $P_i = \{p_1, \ldots, p_i\}$ for some ordering $(p_1, \ldots, p_n)$ of N (Figure 2, left).

The history DAG has a node (i, j) for each $1 \leq i \leq j \leq n$ corresponding to $\mathrm{Vor}_j(p_i)$, the Voronoi cell of p_i in the Voronoi diagram of P_j. It has a directed edge from (i, j) to $(k, j+1)$ if and only if $\mathrm{Vor}_j(p_i) \cap \mathrm{Vor}_{j+1}(p_k) \neq \emptyset$.

Nearest neighbor queries are answered by searching the DAG (Figure 2, right). The longest possible path in the history DAG is known as the *depth* of the DAG and bounds the worst-case time for a query.

If δ is the maximum degree of any Voronoi cell at any time during the incremental construction, then the history DAG has $O(\delta m)$ vertices and edges.

3 Construction

Let $\mathrm{AVD}(N)$ denote the approximate Voronoi diagram constructed for input $N \subset \mathbb{R}^d$. We use the Sparse Voronoi Refinement meshing algorithm (SVR) to produce a superset $M \supset N$. SVR can be instrumented to keep track of the nearest neighbor among the inputs of each new vertex.

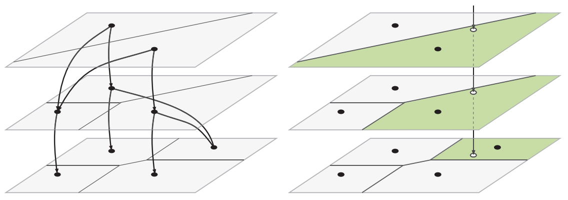


Figure 2: **Left:** The link structure of a simple history DAG. **Right:** A search through the history DAG.

SVR guarantees that the Voronoi cells all have some bound on their aspect ratio. Let $f_S(x)$ be the distance to the second-nearest neighbor of x from the set S. (x is it's own nearest neighbor if $x \in S$. Define the $R_v := \max_{x \in Vor_M(v)} |xv|$[1]. By over-refining the mesh we can guarantee that for all $v \in M$,

$$R_v \leq \frac{\varepsilon}{\varepsilon + 2} f_N(v). \tag{1}$$

The history DAG of the mesh is constructed as part of the SVR algorithm. After the algorithm terminates, the history DAG is kept and used to answer nearest neighbor queries. The depth of the history DAG is guaranteed to be $O(\log \Delta)$ and the degree δ is bounded by a constant τ independent of n.

An approximate nearest neighbor query for N is answered by finding the nearest neighbor among the mesh vertices, M, and returning the nearest input point to that mesh vertex. Indeed the approximate Voronoi cell of a vertex $p \in N$, AVD(p) is the union of Voronoi cells in the mesh of all mesh vertices in $Vor_N(p)$.

4 Analysis

Theorem 4.1. *Let $M \supset N$ be the superset of the input constructed for* AVD(N). *For all x,* $\mathrm{NN}_N(\mathrm{NN}_M(x)) \leq (1+\varepsilon)\mathrm{NN}_N(x)$.

Proof. Let u be the nearest neighbor of x among N. Let v be the nearest neighbor of x from M and u' be the nearest neighbor of v from N, then we seek to show $|xu'| \leq (1+\varepsilon)|xu|$.

If $u = u'$ or $v \in N$, we are done, so suppose $v \notin N$ and $u \neq u'$, thus $f_N(v) \leq |v - u|$. This bound on f and Equation 1 together imply that

$$|vx| \leq R_v \leq \frac{\varepsilon}{\varepsilon+2} f_N(v) \leq \frac{\varepsilon}{\varepsilon+2}|vu| \tag{2}$$

$$\leq \frac{\varepsilon}{\varepsilon+2}(|vx| + |xu|) \leq \varepsilon|xu|. \tag{3}$$

[1]Slight modification is necessary at boundaries.

Collecting terms from this equation then yields $|vx| \leq \varepsilon|xu|$. This fact, the triangle inequality, and the observation $|vu'| \leq |vu|$ imply

$$|xu'| \leq |xv| + |vu'| \leq |xv| + |vu| \tag{4}$$

$$\leq 2|xv| + |xu| \leq (1+\varepsilon)|xu|. \tag{5}$$

□

The following Theorem follows in a straightforward way from the properties of the SVR algorithm [4].

Theorem 4.2. *The size of $AVD(N)$ is $O(n \log \frac{\Delta}{\varepsilon})$. The depth of the corresponding history DAG is $O(\log \frac{\Delta}{\varepsilon})$. Both terms suppress constants that may be exponential in the dimension.*

5 Extensions

We believe it is possible to extend these results to achieve fully linear size approximate Voronoi diagrams by applying ideas from linear-size meshing [5]. We also believe it is possible to replace the $\log \frac{\Delta}{\varepsilon}$ with $\log \frac{n}{\varepsilon}$ by applying methods from geometric separator theory [6]. This is an area of ongoing research.

References

[1] S. Arya and T. Malamatos. Linear-size approximate Voronoi diagrams. In *SODA*, 2002.

[2] S. Arya, T. Malamatos, and D. M. Mount. Space-efficient approximate Voronoi diagrams. In *STOC*, 2002.

[3] S. Har-Peled. A replacement for Voronoi diagrams of near linear size. In *FOCS*, 2001.

[4] B. Hudson, G. Miller, and T. Phillips. Sparse Voronoi Refinement. In *Proceedings of the 15th International Meshing Roundtable*, pages 339–356, Birmingham, Alabama, 2006. Long version available as Carnegie Mellon University Technical Report CMU-CS-06-132.

[5] G. L. Miller, T. Phillips, and D. R. Sheehy. Linear-size meshes. In *CCCG: Canadian Conference in Computational Geometry*, 2008.

[6] G. L. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Finite-element meshes and geometric separators. *SIAM Journal on Scientific Computing*, 19(2):364–386, March 1998.

[7] Y. Sabharwal, N. Sharma, and S. Sen. Nearest neighbor search using point location in balls with applications to approximate Voronoi decompositions. *Journal of Computer and Systems Sciences*, 2006.

Maintaining Approximate Minimum Steiner Tree and k-center for Mobile Agents in a Sensor Network

Dengpan Zhou* Jie Gao*

* Department of Computer Science, Stony Brook University. {dpzhou, jgao}@cs.sunysb.edu

I. Introduction

In this paper we examine the challenges arising from a system of embedded static wireless nodes and a set of mobile agents acting in the same physical space. The agents could be either mobile robots or human users collaborating on a task. The embedded sensor nodes provide real-time monitoring of the environment, possibly in-network processing to aid situation understanding, and interact with both coordination and communication of the agents. This model captures many real-world scenarios.

We focus two specific problems in this framework. The first is to maintain group communication of the mobile agents, in particular, a spanning tree for the agents to exchange information. The second problem is the mobile k-center problem, that asks for k sensor nodes as *centers* such that the maximum distance from each agent to its closest center is minimized.

Challenges. There are two fundamental challenges to allow such coordination and communication between mobile agents. The first problem is *location management*, that is, the tracking and dissemination of the current location of the agents. In our setting, we identify the location of an agent with its proxy sensor node. Although the proxy node is aware of the agent in its proximity, it is difficult to inform other nodes/agents of the current location of an agent. We need to decide how frequently to disseminate the current agent location. The second challenge is efficient maintenance of approximate minimum Steiner tree or approximate k-center of mobile agents, as efficient algorithms even in the centralized setting are lacking.

Our approach. In short, we first preprocess the sensor network with $O(n \lg n)$ total messages. With the preprocessing no location management is needed for MST and k-center maintenance; the agent are not aware of the current location of other agents. thus we save the communication cost necessary to update and query for the current agent location. In particular, a $O(\lg n)$ approximate MST and k-center are maintained with an expected update cost of $O(\lg n)$ messages each time the proxy node of an agent hops to a neighboring node.

In our preprocessing phase, we extract a tree metric on the sensor nodes called a *hierarchical well-separated tree*. This hierarchical well-separated tree is going to approximate the original graph metric G by a logarithmic distortion factor, in terms of the shortest path distance between any two nodes. The advantage of having the HST is that, the minimal Steiner tree on the HST metric is trivial– it is simply the connection of edges from all the agents/proxy nodes to their common ancestor on HST.

II. HST review

Definition 2.1 (α-HST). *A rooted weighted tree H is an α-HST if the weights of all edges between an internal node and its children are the same, all root-to-leaf paths have the same hop-distance, and the edge weights along any such path decrease by a factor of α as we go down the tree.*

Fakcharoenphol *et al.* [1] gives a centralized algorithm to compute a 2-HST metric d_H for an n-point metric (P, d_G). Figure 1 shows an HST example.

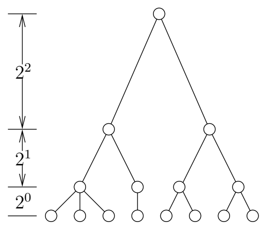


Fig. 1. An example HST constructed from the algorithm.

Since this algorithm is centralized, in [2] we provide a distributed implementation of the HST construction. The value of β, chosen uniformly at random from $[\frac{1}{2}, 1)$, is distributed to all nodes in the network by a single global flood. Initiatively, every node is a candidate, that is, $P_0 = P$. In round $i+1$, the nodes remaining in P_i flood the network up to distances $2^{i+1}\beta$. The flooding packets are cached at each node receiving these packets until the end of each round. Then each node u in the network will choose the node v_{min} with the lowest rank among all the nodes it receives in this round, and nominate it as its $(i+1)$-th level element for its signature vector, i.e. $S(u)_{i+1} = v_{min}$. As the algorithm proceeds, fewer and fewer nodes remain active though the flooding range increases. Therefore the total message cost is still near linear. We have proved in [2] the following lemma.

Lemma 2.2. *For a sensor network with n nodes, the total communication cost for constructing the 2-HST is $O(n \lg n)$ in expection, and each node uses expected storage space for $O(\lg n)$ node IDs.*

Lemma 2.3 (α-HST properties). *Suppose the HST metric (H, d_H) corresponding to the original metric (P, d_G)*

1) *For any $u, v \in P$, $d_H(u,v) \geq d_G(u,v)$ and $E[d_H(u,v)] \leq O(\lg n) \cdot d_G(u,v)$.*

2) *For any $u \in P$, suppose its i-th level ancestor (from leaf to root) in H is u_i. We have $d_H(u_{i+1}, u_i) = \alpha \cdot d_H(u_{i-1}, u_i)$.*
3) *For $\forall i, j (1 \leq i, j \leq \ell, \ell$ is the HST height), the distance between all the nodes in level i and j are of the same value.*

III. Maintaining approximate minimal Steiner tree

A. Maintenance Algorithm

First we compute a 2-HST for the original metric network in a distributed way as stated in our previous section. Each node keeps a signature vector of the ancestors in each level. This process is implemented at the beginning of our application, and it will be used for all the following applications.

1) Compute approximate minimum Steiner tree:

- Each agent is assigned to the nearest sensor node, denoted as the proxy node. A sensor node assigned by some agent sends a counter with the number of agents residing in that node to its parent.
- Each internal node generates a new counter by adding up the counters receiving from its children, and continues to send the counter to its parent.
- Finally, when the counter value equals m, this node will be the root of the minimal Steiner tree in the HST metric.
- After we get the minimal Steiner tree $H(S)$ on the HST H, then replace each edge uv of $H(S)$ with the shortest path between u, v. This gives us a Steiner tree $T(S)$ in the original metric[1].

If the sensor nodes are not aware of the number of agents m in the network, we can solve this by sending notification to upper level until the HST root from the proxy nodes, then tracing back to the child of the lowest internal node u that gets only one notification from its children. And this node u is the root of the minimal Steiner tree. Figure 2 shows this process.

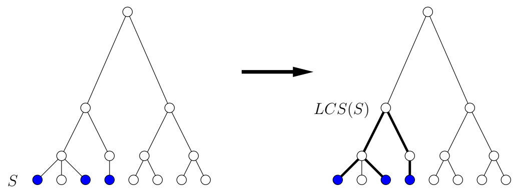


Fig. 2. How to get minimal spanning tree on HST for a subset agents.

2) Maintenance under agent motion: When an agent moves from a proxy node p to a proxy node q, we have to update the minimal Steiner tree on the HST. The idea is that we only need to update the paths from p, q to their lowest common ancestor. In our next section, we show that the expected update cost is small.

- Both p and q send a notification upward until their lowest common ancestor u in the constructed HST.
- All the nodes along the path from p to u will decrease their counter by 1. And any node v with counter to be 0 will delete the edge to its parent in the tree.
- For those nodes on the path from q to u will increase their counter by 1. And add the path to the Steiner tree $H(S)$ if necessary (as well as the Steiner tree $T(S)$ in the original metric).

Figure 3 shows this process. It is easy to see that the approximate minimal spanning tree property is kept in this way.

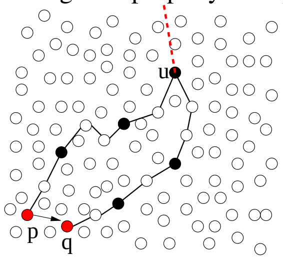


Fig. 3. Repair the paths from p, q to their lowest common ancestor.

B. Analysis and performance

Theorem 3.1. *For a set of agents S, the minimum Steiner tree $T(S)$ is a $O(\lg n)$ approximation for the optimal minimum Steiner tree $MStT(S)$ in the original graph G.*

Theorem 3.2. *For a metric (P, d_G) with bounded growth rate and a HST H with metric (P, d_H), the expected update cost of the minimum Steiner tree on H for each hop an agent moves is bounded by $O(\lg n)$.*

IV. Maintaining approximate k-center

Definition 4.1 (k-center). *Given a sensor network with metric (P, d_G), where d_G is taken on the shortest hop distance for any two node $p, q \in P$ in network G, for a set of agents (actually their proxy nodes) $S \subseteq P$ and an integer k, compute a node set $K = \{a_1, a_2, \ldots, a_k\} \subseteq P$ such that the maximum distance from any agent to its closest center,*

$$\max_{p \in S} \min_{a \in K} d(p, a),$$

is minimized.

For a set of agents S, the lowest common ancestor u of S will be a good candidate for the 1-center with a $O(\lg n)$ approximation. To find a good k-center solution, we simply take the lowest level on the HST such that the number of nodes with non-zero counter is no greater than k. These nodes are the k-centers. We show below that the approximation ratio is $O(\lg n)$ compared with the optimal k-center of G.

Theorem 4.2. *The k-center solution in the HST metric gives a $O(\lg n)$ approximation to the optimal k-center in G.*

References

[1] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 448–455, New York, NY, USA, 2003. ACM.
[2] J. Gao, L. J. Guibas, N. Milosavljevic, and D. Zhou. Distributed resource management and matching in sensor networks. In *Proc. of the 8th International Symposium on Information Processing in Sensor Networks (IPSN'09)*, pages 97–108, April 2009.

[1]$T(S)$ is a logical tree on G. By replacing each edge of $H(S)$ with the shortest path in G we may end up with duplicate edges and possible cycles. One can remove duplicate edges or cut open the cycles to obtain a real Steiner tree on S in the graph G. This operation can only reduce the weight of the tree.

The Connectome – Discovering the Wiring Diagram of the Brain

Hanspeter Pfister, Harvard University

Abstract

The Harvard Center for Brain Science and the School of Engineering and Applied Sciences have been working together since 2007 on the Connectome Project. This ambitious effort aims to apply biology, computer science and software engineering to the grand challenge of determining the detailed neural circuitry of the brain. In this talk I will give an overview of the computational challenges and describe some of the tools that we are developing to address them.

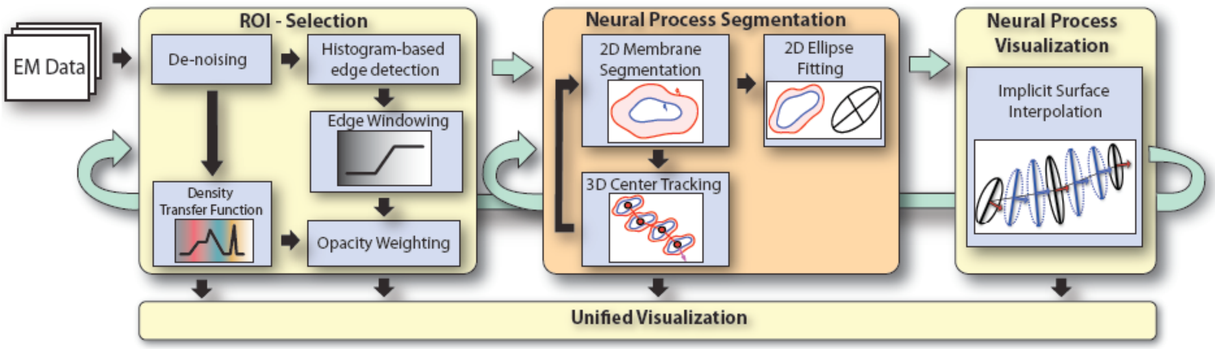
Introduction

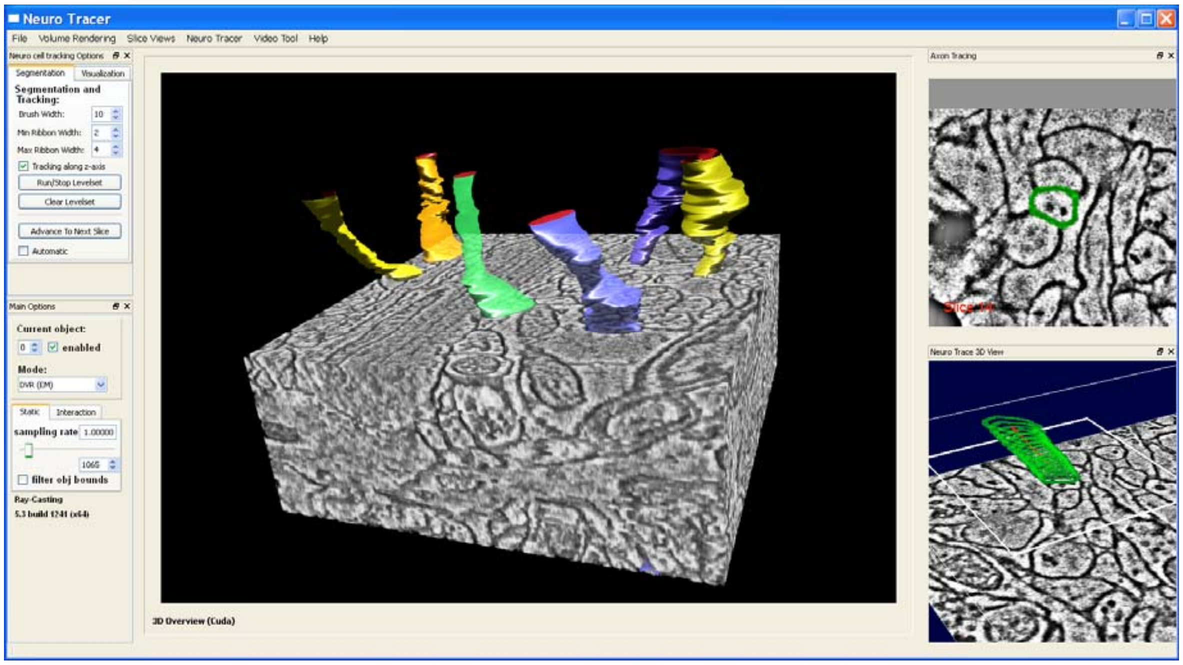
The nervous system is unique among the organ systems in animals because of vast number of interconnections between its individual cells (synapses) and the diversity of its cell types (neurons). One cubic millimeter of cerebral cortex contains roughly 50,000 neurons, each of which establishes approximately 6,000 synapses with neighboring cells (Beaulieu and Colonnier, 1983). These 300 *million* interconnections are highly specific: Neurons innervate some target cells but avoid others. The complexity is further amplified by the fact that neurons come in many kinds. For example, some neurons make excitatory connections, while others establish inhibitory ones. It is thought that there are well over 100 types of neurons, differing in shape, neurochemistry, and function. In action, each neuron integrates the signals from hundreds or thousands of synaptic signals, and this history determines whether or not it will send an electrical signal to its target cells. A cubic millimeter is but a miniscule part of the full circuitry, which is estimated to contain $60x10^{12}$ synaptic connections.

Given these large numbers, it is not surprising that the circuits underlying even the simplest of behaviors are not understood. Until recently, attempts to describe fully such circuits were never even seriously entertained, as it was considered too numerically complex. Now, however, new forms of laser-scanning optical microscopy and semi-automated electron microscopy allow high resolution imaging of “connectomes”—that is, the full sets of neural wires that connect neurons and their targets. Determining the detailed connections in brain circuits is a fundamental unsolved problem in neuroscience. Understanding this circuitry will enable brain scientists to confirm or refute existing models, develop new ones, and come closer to an understanding of how the brain works.

Project Overview

Recent advances in scanning technology provide high resolution EM (Electron Microscopy) datasets that allow neuroscientists to reconstruct these complex neural connections. However, due to the enormous size and complexity of the resulting data, segmentation and visualization of neural processes in EM data is usually a difficult and very time-consuming task. In collaboration with VRVis, Vienna, and Tufts University we have developed NeuroTrace, a novel EM volume segmentation and visualization system that consists of two parts: a semi-automatic multiphase level set segmentation with 3D tracking for reconstruction of neural processes, and a specialized volume rendering approach for visualization of EM volumes.





Segmentation and Tracking:
Brush Width: 10
Min Ribbon Width: 2
Max Ribbon Width: 4
☑ Tracking along z-axis
☐ Automatic

Current object: 0 ☑ enabled
Mode: DVR (EM)
sampling rate 1.00000
1065
☐ filter obj bounds
Ray-Casting
5.3 build 1241 (x64)

The integrated workflow of NeuroTrace provides a unified user interface for easily exploring large EM volumes and extracting neural processes at interactive rates (top). At bottom is a screenshot of NeuroTrace.

NeuroTrace employs view-dependent on-demand filtering and evaluation of a local histogram edge metric, as well as on-the-fly interpolation and ray-casting of implicit surfaces for segmented neural structures. Both methods are implemented on the GPU for interactive performance. NeuroTrace is designed to be scalable to large datasets and data-parallel hardware architectures. A comparison of NeuroTrace with a commonly used manual EM segmentation tool shows that our interactive workflow is faster and easier to use for the reconstruction of complex neural processes.

References

A. Vázquez-Reina, E. Miller and H. Pfister. 2009. Multiphase geometric couplings for the segmentation of neural processes. Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition.

W.-K. Jeong, J. Beyer, M. Hadwiger, A.Vázquez-Reina, H. Pfister, and R. T. Whitaker. Scalable and interactive segmentation and visualization of neural processes in EM data sets. IEEE Transactions on Visualization and Computer Graphics, Oct. 2009.

Succinct Representation of Well-Spaced Point Clouds[*]

Benoît Hudson
Toyota Technological Institute at Chicago

Storing n points naively costs $\Theta(nw)$ bits of space on a machine with w-bit words. I show how to reduce the space usage to $O(n)$ bits, independent of the word length. This is an asymptotic reduction: since the n points are all distinct, it must be that $w \in \Omega(\log n)$. The proof is contingent on some sampling assumptions: the points must be well-spaced with respect to some (not necessarily known) subspace. This assumption applies in the case of the input for surface reconstruction, or the output for mesh refinement. In practice, the encoding uses 14 bits per vertex (bpv) for the Stanford bunny, or 18 bpv for the Lucy dataset. This compares to 96 bpv for the uncompressed point set represented with 32-bit float coordinates.

In addition, the data structure is a succinct form of a balanced 2^d-tree [BEG94] (for legibility I call this a quadtree). That is, it supports quadtree operations, such as looking up what leaf quadtree square contains a given point, or what points are contained within a quadtree square. Funke and Milosavljevic [FM07] show how to use these operations to reconstruct a 2-manifold given a set of points in three or higher dimensions. Hudson and Türkoğlu [HT08] show how to use a quadtree to augment an arbitrary set of points in d dimensions and generate a well-spaced point set, that is, one whose Delaunay triangulation will produce a good mesh for scientific applications. Whereas in an uncompressed quadtree, the operations take constant time, in this succinct format, they take $O(\log n)$ time assuming $w \in \Theta(\log n)$: the structure trades a logarithmic slowdown to achieve a logarithmic reduction in memory usage.

I give a lower bound that shows that the encoding is necessarily lossy: a point cloud that matches the sampling assumption can be randomly perturbed to be almost incompressible. The compression technique ensures that the quadtree built over the rounded points is isomorphic to the one built over the original input. I show how to compute the rounding without loading the entire file, uncompressed, into memory; this takes $O(w)$ scans of the input file. While the encoding is lossy, one can easily store additional bits of accuracy. Storing the bunny in 32 bpv allows storing an additional six bits, provably bounding the error interpoint distances to within 6%.

Prior work: It has been repeatedly shown how to compress a point cloud with known connectivity down to 10 to 20 bpv [PKK05, survey]. This matches or beats my result, but has a significant weakness: these formats require that we already know the connectivity. However, the connectivity is precisely what we want to compute in surface reconstruction, and is too expensive to maintain during mesh refinement. Blandford *et al* propose a succinct data structure that computes the Delaunay triangulation while keeping the connectivity information compressed, but the geometry uncompressed [BBCK05]. The results here complement theirs.

Input Assumption: Given a point set P and a compact space S, the *restricted Voronoi diagram* assigns to each $p \in P$ a cell V_p consisting of the set of points $x \in S$ that are closer to p than to any other point $q \in P$. In other words, it is the intersection of the Voronoi diagram of P and the space S. We can now define the input assumption:

Definition 1 *The point cloud P is ρ-**well-spaced** if for all $p \in P$, $\frac{\max_{x \in V_p} |px|}{\min_{q \in P} |pq|} \leq \rho$.*

Well-spacedness arises naturally from my targeted applications. In the mesh refinement problem, the goal is precisely to generate a well-spaced set of points. In surface reconstruction, a common assumption is that the input is an (ϵ, δ)-sample of S. Simple algebra shows that an (ϵ, δ)-sample is $\epsilon/(\delta(1-\epsilon))$ well-spaced.

[*]Long version available as arXiv:0909.3137

Rounding: It it necessary to round because an adversary can take a well-spaced set described using $w/2$-bit coordinates and add $w/2$ random low-order bits, thwarting any asymptotic space savings. I describe a rounding routine that approximately maintains inter-point distances. First we compute the balanced quadtree of Bern *et al* [BEG94], then we round points to the minimum corner of their corresponding leaf quadtree box. The minimum corner of a box is given by the leading bits of the coordinates of the points inside: rounding is masking off a principled number of low-order bits. Given original points p and q rounded to p' and q', the ratio $|p'q'|/|pq|$ is in the range $[(1+2\sqrt{d})^{-1}, 1+2\sqrt{d}]$. Storing γ additional bits improves the approximation ratio to $(1 + 2^{1-\gamma}\sqrt{d})$. Using six additional bits—32 bpv on the Stanford bunny—bounds distortion within $[0.949, 1.055]$.

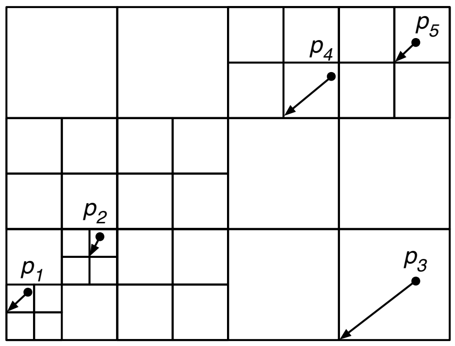


Figure 1: Balanced quadtree of five points in Morton order pointing to their rounded form.

Morton order: The Morton number of a point coordinates is constructed by interleaving the bits of the integer representations of the coordinates; this gives a new, very wide integer. Ordering points by their Morton number gives the Morton order, also known as the Z-order. Comparing two points by their place in the order takes constant time assuming standard bitwise operations, even for floating-point inputs [CK08]. Given an array of points in Morton order, the points contained in a given quadtree box are consecutive. Thus looking up what points a box contains corresponds to performing two binary searches, one each for the minimum and maximum corners of the box. The points in the Figure are numbered according to Morton order.

Compression: Essentially the compressed format stores not the points, but their corresponding quadtree boxes: the depth of the box, and the coordinates of the minimum corner of the box. Since we know the depth of the box, all lower-order bits are zero and need not be represented. Furthermore, the depth and the leading bits do not change much from point to point because they are stored in Morton order. Therefore, it pays to store the data using a difference-encoding: keeping track of the previous point in the order, store only the difference in depth from the prior point, and store only the XOR for each coordinate after rounding. Using a variable-length encoding scheme for these integers allows exploiting the fact that on average they are small.

Space usage: The space usage is proved using a correspondence between the bits of the encoding of the coordinates, and a depth-first traversal of the quadtree. The leading bit of an XOR corresponds to the depth of the least common ancestor of the current box and its predecessor: storage is proportional to the path length. Since the points are stored in Morton order, each quadtree box is entered from above by only one such path: the total number of bits is proportional to the number of quadtree nodes. Hudson *et al* [HMPS09] prove that a minimal-size volume mesh built over a well-spaced surface mesh has only a constant factor more elements. The balanced quadtree forms such a volume mesh, so it has $O(n)$ leaves for n well-spaced points. Ergo, the storage cost is $O(n)$ total bits.

References

[BBCK05] Daniel K. Blandford, Guy E. Blelloch, David E. Cardoze, and Clemens Kadow. Compact Representations of Simplicial Meshes in Two and Three Dimensions. *Int. J. Comp. Geom. & App.*, 15(1):3–24, 2005.

[BEG94] Marshall Bern, David Eppstein, and John R. Gilbert. Provably Good Mesh Generation. *Journal of Computer and System Sciences*, 48(3):384–409, 1994.

[CK08] M. Connor and Piyush Kumar. Parallel construction of k-nearest neighbour graphs for point clouds. In *Eurographics Symposium on Point-Based Graphics*, 2008.

[FM07] Stefan Funke and Nikola Milosavljevic. Network Sketching or: "How Much Geometry Hides in Connectivity?–Part II". In *SODA*, pages 958–967, 2007.

[HMPS09] Benoît Hudson, Gary L. Miller, Todd Phillips, and Don Sheehy. Size complexity of volume meshes *vs.* surface meshes. In *SODA*, 2009.

[HT08] Benoît Hudson and Duru Türkoğlu. An efficient query structure for mesh refinement. In *Canadian Conference on Computational Geometry*, 2008.

[PKK05] Jingliang Peng, Chang-Su Kim, and C.-C. Jay Kuo. Technologies for 3D mesh compression: A survey. *Journal of Visual Communication and Image Representation*, 16:688–733, 2005.

Spatio-temporal Range Searching Over Compressed Kinetic Sensor Data

[Abstract]

Sorelle A. Friedler[*] sorelle@cs.umd.edu http://www.cs.umd.edu/~sorelle
David M. Mount[†] mount@cs.umd.edu http://www.cs.umd.edu/~mount
Dept. of Computer Science, University of Maryland, College Park, MD 20742

Sensor networks and the data they collect have become increasingly prevalent and large. Sensor networks are frequently employed to observe objects in motion and are used to record traffic data, observe wildlife migration patterns, and observe motion from many other settings. In order to perform accurate statistical analyses of this data over arbitrary periods of time, the data must be faithfully recorded and stored. Large sensor networks, for example observing a city's traffic patterns, may generate gigabytes of data each day. The vast quantities of such data necessitate compression of the sensor observations, yet analysis of these observations is desirable. Ideally, such analysis should operate over the compressed data without decompressing it. Before more sophisticated statistical analyses of the data may be performed, retrieval queries must be supported. In this work, we present the first range searching queries operating over compressed kinetic sensor data.

In an earlier paper, we presented an algorithm for losslessly compressing kinetic sensor data and a framework for analyzing its performance. We assume that we are given a set of sensors, which are at fixed locations in a space of constant dimension. (Our results apply generally to metric spaces of constant doubling dimension.) These sensors monitor the movement of a number of kinetic objects. Each sensor monitors an associated region of space, and at regular time steps it records an occupancy count of the number of objects passing through its region. Over time, each sensor produces a string of occupancy counts, and the problem considered previously is how to compress all these strings.

Previous compression of sensor data in the literature has focused on stream algorithms and lossy compression of the data. We consider lossless compression. This is often more appropriate in scientific contexts, where analysis is performed after the data has been collected and accurate results are required. Lossless compression algorithms have been studied in the single-string setting, but remain mostly unstudied in a sensor-based setting.

In order to query observed sensor data, which ranges over time and space, we need to consider both temporal and spatial queries. *Temporal range queries* are given a time interval and return an aggregation of the observations over that interval. *Spatial range queries* are given some region of space (e.g., a rectangle, sphere, or halfplane) and return an aggregation of the observations within that region. *Spatio-temporal range queries* generalize these by returning an aggregation restricted by both a temporal and a spatial range. We assume that occupancy counts are taken from a commutative semigroup, and the result is a semigroup sum over the range. There are many different

[*]The work of Sorelle Friedler has been supported in part by the AT&T Labs Fellowship Program.

[†]The work of David Mount has been supported in part by the National Science Foundation under grant CCR-0635099 and the Office of Naval Research under grant N00014-08-1-1015

Bounds for Range Searching

	Temporal	Spatio-temporal
Preprocessing time	$O(\mathrm{Enc}(X))$	$O(\mathrm{Enc}(\mathbf{X}))$
Query time	$O(\log T)$	$O(((1/\varepsilon^{d-1}) + \log S) \log T)$
Space	$O(\mathrm{Enc}(X))$	$O(\mathrm{Enc}(\mathbf{X}) \log S)$

Table 1: Asymptotic time and space results achieved for temporal range searching and ε-approximate spherical spatio-temporal range searching in $\mathbb{R}^d$ space, where S is the number of sensors in the network, T is the length of the observation period, and $\mathrm{Enc}(X)$ and $\mathrm{Enc}(\mathbf{X})$ denote the size of the compressed representations of a single sensor stream (for temporal range searching) and sensor system (for spatio-temporal range searching).

---

data structures for range searching (on uncompressed data), depending on the properties of the underlying space, the nature of the ranges, properties of the semigroup, and whether approximation is allowed.

We present data structures for storing compressed sensor data and algorithms for performing spatio-temporal range queries over this data. We analyze the quality of these range searching algorithms in both time and space by considering the information content of the set of sensor outputs. There are two well-known ways in which to define the information content of a string, classical Shannon entropy and empirical entropy. Shannon entropy is defined in a statistical context, under the assumption that X is a stationary, ergodic random process. The normalized Shannon entropy, denoted $H(X)$, provides a lower bound on the number of bits needed to encode a character of X. In contrast, the empirical entropy, denoted $H_k(X)$, is similar in spirit to the Shannon entropy, but assumes no underlying random process and relies only on the observed string and the context of the most recent k characters.

Previous retrieval over compressed text (without relying on decompression) has been studied in the context of strings and XML files. For example, Ferragina and Manzini show that it is possible to retrieve all occurrences of a given pattern in the compressed text with query time equal to the number of occurrences plus the length of the pattern. Their space requirement is $5T \cdot H_k(X) + o(T)$ bits for a string X of length T. However, their data structure allows substring queries, which are very different from semigroup range searching queries, which we consider here.

Results

In this work we present the first range query results over compressed data; our results apply specifically to compressed kinetic sensor data. In addition we generalize the analysis of a previously presented compression algorithm to hold in an empirical context. We analyze the range query results in both a statistical and an empirical context. The preprocessing bounds show that we only make one pass over the compressed data. The query bounds are logarithmic in the input size. The space bounds, given in bits, show that we achieve these results without decompressing the data. For specific bounds see Table 1. Our temporal bounds rely on an extension of the Sleator and Tarjan data structure for dynamic trees to query time periods and aggregate underlying data. Our spatio-temporal bounds combine these temporal aggregations with a variant of approximate range searching that relies on the separation properties of the compressed data.

Dynamic Data Structures for Simplicial Thickness Queries [*]

Danny Z. Chen[†] Haitao Wang[†]

1 Introduction

Designing data structures for processing simplex related operations is a fundamental computational geometry topic. In this abstract, we study dynamic data structures for the *simplicial thickness query* problem. For a set S of simplices in the d-D space E^d ($d \geq 2$ is fixed), the *simplicial thickness* of a point p is defined as the number of simplices in S that contain p, which we denote by $\beta(p, S)$. Given a set of n points in E^d, $P = \{p_1, p_2, \ldots, p_n\}$, for a simplex σ, we define the *simplicial thickness* of σ as the minimum simplicial thickness among all points in $\sigma \cap P$, and denote it by $\beta(\sigma, S)$. We seek dynamic data structures to support the following operations (initially, $S = \emptyset$). (1) Simplex insertion: Insert a simplex into S. (2) Simplex deletion: Delete a simplex from S. (3) Simplicial thickness query: Given a query simplex σ, report $\beta(\sigma, S)$. We call this problem the *simplicial thickness query* problem, and denote it by STQ.

While we are not aware of any previous work on STQ, the simplex range searching problem has been studied extensively [6, 2, 3, 4]. By using standard approaches [1, 5], the author in [3] also gave results on the dynamic version of the simplex range searching which allows point insertions and deletions (rather than simplex operations). Further, the standard dynamic data structure design techniques such as those in [1, 5] cannot be applied to solving our problems.

2 Our Solutions

Based on the (static) simplex range searching data structure in [3], we propose a dynamic STQ data structure whose performances match those in [3], i.e., it occupies $O(n)$ space and can be built in $O(n \log n)$ time, and can support the simplex insertion, simplex deletion, and simplicial thickness query each in $O(n^{1-1/d}(\log n)^{O(1)})$ time. Below, we first sketch the simplex range searching data structure in [3].

A *simplicial partition* of a point set P is a collection $\Pi = \{(P_1, \Delta_1), \ldots, (P_m, \Delta_m)\}$, where the P_i's are pairwise disjoint subsets (called the *classes* of Π) forming a partition of P, and each Δ_i is a possibly open simplex containing the set P_i (Δ_i may also contain other points in P than those in P_i). A partition is called *special* if $\max_{1 \leq i \leq m}\{|P_i|\} < 2\min_{1 \leq i \leq m}\{|P_i|\}$.

The data structure in [3] is a *partition tree*, T, based on constructing special simplicial partitions on P recursively (see Fig. 1). The leaves of T form a partition of P into constant-sized subsets. Each internal node v corresponds to a subset P_v (and its corresponding simplex Δ_v) of P and to a special simplicial partition Π_v of size $|P_v|^{1-1/d}$ of P_v. The *cardinality* of P_v (i.e., $|P_v|$) is stored at v and v has $|P_v|^{1-1/d}$ children that correspond to the classes of Π_v. We can construct T in $O(n \log n)$ time and $O(n)$ space [3] .

For each simplex range query σ, we start from the root of T. For each internal node v, we check Π_v one by one, and handle directly those contained in σ or disjoint from σ; we proceed with the corresponding child nodes for the other simplices. Each of the latter ones must be intersected by at least one of the hyperplanes bounding σ. If v is a leaf node, for each point p in P_v, we determine directly whether $p \in \sigma$. Each query takes $O(n^{1-1/d}(\log n)^{O(1)})$ time [3].

Given P and S, our STQ data structure T' is built using the same partitioning scheme as T. For each internal node $v \in T'$, instead of storing the cardinality of P_v, we store two key values $k_1(v)$ and $k_2(v)$. To define $k_1(v)$ and $k_2(v)$, we need some new concepts. We say that a simplex s' *pokes* another simplex s'' if $s' \cap s'' \neq \emptyset$ and s' does not contain s'' completely. Note that s' poking s'' does not necessarily imply s'' also poking s'. Let $parent(v)$ denote the parent node of v in T'. For each node $v \in T'$, if v is the root, then let $IS(v)$ be the set S; otherwise, let $IS(v)$ be the set of simplices in $IS(parent(v))$ that poke Δ_v ($IS(\cdot)$ is only a conceptual notion, and we do not explicitly compute and maintain it). We define $k_1(v)$ as the number of simplices in $IS(parent(v))$ that contain Δ_v completely (if v is the root, let $k_1(v) = 0$); we define $k_2(v)$ to be the minimum simplicial thickness among

[*]This research was supported in part by NSF under Grants CCF-0515203 and CCF-0916606.

[†]Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA. E-mail: {dchen, hwang6}@nd.edu.

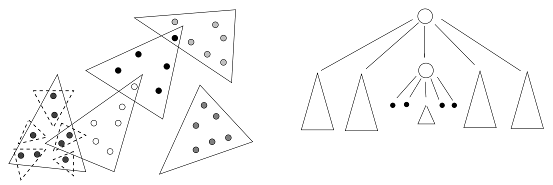


Figure 1: Illustrating a simplicial partition and the corresponding partition tree. The dotted triangles form the partition computed recursively for the class corresponding to the middle child of the root; the resulting five "subclasses" are stored in five subtrees below the middle child.

all points in P_v with respect to the simplices in $IS(v)$, i.e., $k_2(v) = \min_{p_i \in P_v}\{\beta(p_i, IS(v))\}$. For each point p_i in a leaf node v (i.e., $p_i \in P_v$), we store $k_1(p_i)$ that is the number of simplices in $IS(v)$ containing p_i, i.e., $k_1(p_i) = \beta(p_i, IS(v))$. Suppose the k_1 values of all nodes and points in T' have been set correctly; then the k_2 values of all nodes in T' can be obtained by the lemma below (with proof omitted). For each internal node v, denote by $C(v)$ the set of all children of v.

Lemma 1 *For each internal node $v \in T'$, $k_2(v) = \min_{v' \in C(v)}\{k_1(v') + k_2(v')\}$; for each leaf node $v \in T'$, $k_2(v) = \min_{p_i \in P_v}\{k_1(p_i)\}$.*

It is easy to see that T' uses the same space as T, which is $O(n)$. Since $S = \emptyset$ initially, all key values in T' are initially zero, and thus T' can be constructed in $O(n \log n)$ time as T.

To insert a simplex σ into S, beginning from the root of T', for each internal node v, we check the its children one by one. Precisely, for each $v' \in C(v)$, if $\Delta_{v'}$ is completely contained in σ, we increase $k_1(v')$ by 1; if $\Delta_{v'}$ is disjoint from σ, we do nothing; if σ pokes $\Delta_{v'}$, we process the children of v' recursively. For a leaf node v, we check the points in P_v one by one: For each $p_i \in P_v$, if p_i is contained in σ, we increase $k_1(p_i)$ by 1. After the k_1 values of all involved nodes in T' have been updated, we update the k_2 values by Lemma 1 accordingly in a bottom-up manner. Note that the nodes visited by this procedure are exactly the nodes visited by the simplex range query answering procedure if we applied σ on T'. Thus, the running time for handling the simplex insertion is $O(n^{1-1/d}(\log n)^{O(1)})$. The simplex deletion can be done analogously with the same running time.

For each simplicial thickness query σ, if we apply σ to T' as if a simplex insertion is done, then it will induce multiple search paths each of which ends up at a node v (or a single point p_i) such that σ contains Δ_v (or the point p_i) and there is no proper ancestor v' of v such that σ contains $\Delta_{v'}$ (for a leaf node v and a point $p_i \in P_v$, we treat v as the parent of p_i). We call such nodes or points the *ending nodes*. Note that the ending nodes are those nodes or points in T' whose k_1 values would be increased if σ were inserted. Let $V(\sigma)$ (resp., $P(\sigma)$) denote the set of nodes (resp., points) in T' that are ending nodes for σ. We have the following lemma (with proof omitted).

Lemma 2 *For each $v \in V(\sigma)$, let $A(v)$ denote the set of all ancestors of v in T' including v. For each $p_i \in P(\sigma)$, let $v(p_i)$ be the leaf node such that $p_i \in P_{v(p_i)}$. Then, the answer to the simplicial thickness query, i.e., $\beta(\sigma, S)$, is $\min\{\beta_1, \beta_2\}$ where $\beta_1 = \min_{v \in V(\sigma)}\{k_2(v) + \sum_{v' \in A(v)} k_1(v')\}$ and $\beta_2 = \min_{p_i \in P(\sigma)}\{k_1(p_i) + \sum_{v' \in A(v(p_i))} k_1(v')\}$.*

To handle a simplicial thickness query σ, we use a similar procedure to the simplex insertion, in which we accumulate the k_1 values from the root of T' down to the nodes on each search path. As we encounter an ending node, if it is a node v (resp. a point p_i contained in P_v of the leaf node v), we compute the value $k_2(v) + \sum_{v' \in A(v)} k_1(v')$ (resp. $k_1(p_i) + \sum_{v' \in A(v)} k_1(v')$), and call it the *thickness-query value*. Since we have the value $\sum_{v' \in A(v)} k_1(v')$ available, the above thickness-query value can be obtained in $O(1)$ time. Further, we also maintain the minimum thickness-query value among those that have already been computed. When the procedure stops, we report the minimum thickness-query value as the answer to the simplicial thickness query. It takes $O(1)$ time at each node to either accumulate the k_1 values or compute the thickness-query value and maintain the minimum one. Thus, the running time is the same as a simplex insertion, which is $O(n^{1-1/d}(\log n)^{O(1)})$.

References

[1] J.L. Bentley. Decomposable searching problems. *Information Processing Letters*, 8:244–251, 1979.

[2] B. Chazelle, M. Sharir, and E. Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. In *Proc. of the 6th ACM Symposium on Computational Geometry*, pages 23–33, 1990.

[3] J. Matoušek. Efficient partition trees. *Discrete and Computational Geometry*, 8(3):315–334, 1992.

[4] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete and Computational Geometry*, 10(1):157–182, 1993.

[5] M.H. Overmars. *The design of dynamic data structures.* Springer-Verlag, Berlin, 1983.

[6] E. Welzl. Partition trees for triangle counting and other range searching problems. In *Proc. of the 4th ACM Symposium on Computational Geometry*, pages 23–33, 1988.

Discrete Voronoi Diagrams and Post Office Query Structures without the InCircle Predicate

Timothy M. Chan [*] David L. Millman [†] Jack Snoeyink [‡]

Abstract. We develop an algorithm to compute the Voronoi diagram (or distance transform) of n sites (feature pixels) in a $U \times U$ grid using double precision, instead of the usually 4-fold precision InCircle test. Our algorithm runs in $O(U^2 + n \log U)$ time.

1 Introduction

Geometric algorithms use numerical computations to perform geometric tests, so correct algorithms may produce erroneous results if they require more arithmetic precision than is available. Liotta *et al.* [5] suggested analyzing the precision of an algorithm by the maximum degree of its predicates. They demonstrated that a Voronoi diagram of sites on a grid could be reduced from a degree 6 to a degree 2 point location structure that correctly answers "Post Office" queries: find the nearest site to a query grid point. They still used degree 4 InCircle tests to compute the Voronoi before reducing it; Millman and Snoeyink [6] recently were able to start with a restricted Voronoi diagram computed with degree 3 predicates.

The distance transform of a $U \times U$ pixel image $\mathbb{I}$ can use the Voronoi diagram to label each pixel of $\mathbb{I}$ with its closest feature pixel; Breu *et al.* [1] showed that this could be done in $O(U^2)$ time. Their algorithm used InCircle (degree 4). Chan [2] generalized to compute the d-dimensional *discrete Voronoi diagram* in $O(U^d)$ time; his algorithm can be implemented using predicates of degree 3.

The distance transform can be computed using degree 2 computations by simply comparing squared distances to all feature pixels; in fact, Hoff and others [4] implement this idea in graphics hardware to compute 2D and 3D discrete Voronoi diagrams very quickly, albeit in $\Theta(nU^2)$ total work, and with the fixed resolution of the screen and precision of the Z-buffer.

We show how to compute the 2D discrete Voronoi diagram with double precision in $O(U^2)$ expected time with a scan line algorithm. The 2D discrete Voronoi diagram requires $O(U^2)$ space and answers post office queries in constant time. In some situations this memory requirement is too large and only $O(\log n)$ time post office queries are required. For these situations we describe an intermediate degree 2 data structure called the *Voronoi Polygon Set* that requires only $O(n \log U)$ space and adds no time complexity. Finally, we create a post office query structure from the Voronoi Polygon Set in $O(n \log n)$ expected time using $O(n)$ expected space that answers post office queries in $O(\log n)$ expected time with degree 2 predicates.

2 Geometric Preliminaries

Let $\mathbb{U} = \{1, \dots, U\}$, and given a set of n feature pixels, called *sites*, $S = \{s_1, \dots, s_n\}$ in $\mathbb{U}^2$ the *Voronoi Polygon Set* partitions $\mathbb{U}^2$ into n convex polygons $\{C(s_1), \dots, C(s_n)\}$ where $C(s_i)$ is the convex hull of the grid points in closure of the Voronoi cell of s_i. We call each $C(s_i)$ a *Voronoi Polygon*, and each polygon stores a leftmost and rightmost grid point. Figure 1 shows that *gap* regions may remain between Voronoi Polygons; without higher degree predicates, it seems difficult to determine neighboring Voronoi cells across gaps.

For the remainder let h_i denote the horizontal line $y = i$ and ℓ_j denote the vertical line $x = j$.

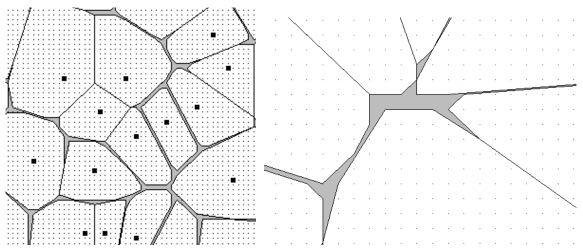


Figure 1: Left: The *Voronoi Polygon Set* of sites randomly chosen on a grid. To recover the Voronoi diagram's precise structure in the gray gaps seems to require higher degree predicates. Right: A close-up of a complex *gap*.

[*]David R. Cheriton School of Computer Science, University of Waterloo, tmchan@uwaterloo.ca

[†]Department of Computer Science, University of North Carolina at Chapel Hill, dave@cs.unc.edu

[‡]Department of Computer Science, University of North Carolina at Chapel Hill, snoeyink@cs.unc.edu

3 Discrete Voronoi Diagram

Given grid size U and a set of n sites S we specialize Chan's algorithm for computing the 2D discrete Voronoi diagram [2] in $O(U^2)$ expected time but using degree 2 predicates only.

Bin: First sort the sites of S by x breaking ties with y. Then partition S into U lists where $S_i = \{s \in S \mid s_x = i\}$, a list for each vertical grid line ℓ_i.

Preprocess: For each S_i solve the $1D$ problem of marking each grid point on ℓ_i with its closest neighbor in S_i. Let $\mathcal{R} = \{R_1, \ldots, R_U\}$ where list R_j contains $s \in S$ such that there exists a grid point q on h_j with $\text{mark}(q) = s$. Preprocessing constructs $\mathcal{R}$ where each R_i has at most U sites.

Scan line: For each vertical line h_j corresponding to R_j, label each grid point p on h_j with its closest site in S. The labels are stored in a list I_j of tuples (s, a, b) where grid points in the interval $[a, b]$ on h_j have site s as a label. This creates U lists $\{I_1, \ldots, I_U\}$ each with at most U intervals. The interval lists encode the discrete Voronoi diagram.

Space and time analysis omitted for abstract.

Bin compares degree 1 coordinate values, Preprocess and Scan line compare degree 2 squared distances. Therefore, we compute the Discrete Voronoi diagram in degree 2.

4 Post Office Query Structure

We describe an $O(U^2)$ space algorithm for clarity and point out that the Voronoi Polygon Set can be built one scan line at a time to reduce memory.

Given the interval set from the previous section, an alternative output is an expected $O(n)$ size data structure for solving post office queries in $O(\log n)$ expected time with degree 2 predicates. Computing this uses $O(U + n \log U)$ space and expected $O(U^2 + n \log n)$ time.

Compress: For each interval set I_j, merge the intervals of I_j into the Voronoi Polygon Set of rows $\{1, \ldots, j-1\}$. Update the leftmost and rightmost grid points of modified Voronoi Polygons.

Build: Represent each Voronoi Polygon $C(s_i)$ with a segment from a leftmost to rightmost grid point and a link to associated site s_i. Build the trapezoid graph [3] of the representative segments.

Space and time analysis omitted for abstract.

Compress and Build evaluate degree 2 orientation predicates on grid points and compare degree 1 x-coordinates. Thus, building the polygon set and trapezoid graph are degree 2.

Given a query point q, we solve post office queries on the trapezoid graph in $O(\log n)$ expected time by locating the trapezoid containing q [3]. The trapezoid is defined by at most two non-vertical segments each with an associated site. The post office query is answered by the site closer to q.

Point location uses the same predicates as building the trapezoid graph with the addition for comparing degree 2 squared distances. Therefore, point location is degree 2.

5 Conclusion

Our algorithm provides an exact method for computing the discrete Voronoi diagram and a post office query structure on a point set with single precision coordinates in double precision. The algorithm is simple to implement and degeneracies are easily handled.

We believe that our algorithm is a good candidate for parallelization. Each scan line step is independent; horizontal slabs can be processed concurrently and merged with a parallel reduce. We would like to investigate parallelizing our algorithm on graphical processing units since they show impressive results for Hoff *et al.* [4] and our algorithm does not require any libraries for robust computation.

We believe that our algorithm may generalize in dimension and apply to the discrete Power and furthest point Voronoi diagrams while maintaining degree 2 algebraic complexity. However, it is still unknown if a post office query structure is computable in subquadratic time with degree 2 predicates.

References

[1] H. Breu, J. Gil, D. Kirkpatrick, and M. Werman. Linear time Euclidean distance transform algorithms. *IEEE PAMI*, 17:529–533, 1995.

[2] T. M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom. Theory Appl.*, 35(1):20–35, 2006.

[3] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag New York, Inc., 3rd edition, 2008.

[4] K. E. Hoff III, J. Keyser, M. Lin, D. Manocha, and T. Culver. Fast computation of generalized Voronoi diagrams using graphics hardware. In *ACM SIGGRAPH*, pages 277–286, 1999.

[5] 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.

[6] D. L. Millman and J. Snoeyink. Computing the implicit Voronoi diagram in triple precision. In *WADS*, volume 5664 of *LNCS*, pages 495–506. Springer, 2009.

Formulae and Growth Rates of High-Dimensional Polycubes

Ronnie Barequet[*] Gill Barequet[†] Günter Rote[‡]

Abstract

A d-D polycube is a facet-connected set of cubes in d dimensions. Fixed polycubes are considered distinct if they differ in shape or orientation. A *proper* d-D polycube spans all d dimensions. We prove some formulae for fixed (proper and improper) polycubes, show that the growth-rate limit of the number of polycubes in d dimensions is $2ed - o(d)$, and estimate it at $(2d-3)e + O(1/d)$.

1 Introduction

A d-D *polycube* of size n is a connected set of n cubical cells on $\mathbb{Z}^d$, where connectivity is through $(d-1)$-faces. Two *fixed* polycubes are equivalent if one is a translation of the other. A fixed polycube is *improper* in d-D if the centers of all its cubes lie in a common hyperplane of dimension less than d. Counting polyominoes (2-D polycubes) is a long-standing problem, originating in statistical physics [3]. The number $A(n)$ of fixed polyominoes is currently known up to $n = 56$ [7]. Tabulations of counts of polycubes in higher dimensions appear in the mathematics literature [1, 10], as well as in the statistical-physics literature [4, 6]. The growth-rate limit of polyominoes has also attracted much attention. Klarner [8] showed that $\lambda_2 = \lim_{n\to\infty} \sqrt[n]{A(n)}$ exists. The convergence of $A(n+1)/A(n)$ to λ_2 was proved much later by Madras [11]. The best-known lower [2] and upper [9] bounds on λ_2 are 3.9801 and 4.6496, respectively. It is widely assumed [4, 6, 7] that $\lambda_2 \approx 4.06$. In $d > 2$ dimensions, the growth-rate limit λ_d of the number of polycubes is also guaranteed to exist [11]. The most accurate estimate of λ_d is given [5] as

$$\ln \lambda_d = \ln \sigma + 1 - \tfrac{2}{\sigma} - \tfrac{79}{24\sigma^2} - \tfrac{317}{24\sigma^3} - \tfrac{18321}{320\sigma^4} - \tfrac{123307}{240\sigma^5} + O(\tfrac{1}{\sigma^6}), \qquad (1)$$

where $\sigma = 2d-1$. We follow Lunnon [10], denoting by $\mathrm{DX}(n,d)$ (resp., $\mathrm{CX}(n,d)$) the number of proper (resp., all) fixed polycubes of size n in d-D. For a fixed value of n, we denote these functions as $\mathrm{CX}_n(d)$ and $\mathrm{DX}_n(d)$, respectively. Lunnon [10] observed the following relation: $\mathrm{CX}(n,d) = \sum_{i=0}^{\min(n-1,d)} \binom{d}{i} \mathrm{DX}(n,i)$.

2 Diagonals, or: Minimal Proper Polycubes

Obviously, $\mathrm{DX}(n,d) = 0$ for $n < d+1$. The number of these minimal proper polycubes is given by $\mathrm{DX}(n,n-1) = 2^{n-1}n^{n-3}$ [6]. The limit ratio between the growth rate of the number of minimal proper polycubes and the size of the polycubes is $2e$: $\frac{\mathrm{DX}(n+1,n)/\mathrm{DX}(n,n-1)}{n+1} \xrightarrow[n\to\infty]{} 2e$. Formulae for $\mathrm{DX}(n,n-i)$, $i = 2,3$, can be extrapolated from the known values of $\mathrm{DX}(\cdot,\cdot)$. They are widely used in the statistical-physics literature without a rigorous proof. We prove that $\mathrm{DX}(n,n-2) = 2^{n-3}n^{n-5}(n-2)(2n^2-6n+9)$ and $\mathrm{DX}(n,n-3) = 2^{n-6}n^{n-7}(n-3)(12n^5 - 104n^4 + 360n^3 - 679n^2 + 1122n - 1560)/3$. As above, $\lim_{n\to\infty} \frac{\mathrm{DX}(n+2,n)/\mathrm{DX}(n+1,n-1)}{n+2} = \lim_{n\to\infty} \frac{\mathrm{DX}(n+3,n)/\mathrm{DX}(n+2,n-1)}{n+3} = 2e$.

3 Rows, or: Polycubes of Fixed Size

We show that for any fixed n, $\mathrm{CX}_n(d)$ is a polynomial in d, and find the formulae for the first three sequences of the leading coefficients of these polynomials.

[*]Depts. of Math and of Computer Science, Tel Aviv Univ., Tel Aviv 69978, Israel. E-mail: `ronnieba@post.tau.ac.il`
[†]Dept. of Computer Science, The Technion, Haifa 32000, Israel. E-mail: `barequet@cs.technion.ac.il`
[‡]Inst. für Informatik, Freie Universität Berlin, Takustraße 9, D-14195 Berlin, Germany. E-mail: `rote@inf.fu-berlin.de`

Theorem 1 $\mathrm{CX}_n(d)$ *is a polynomial in* d *of degree* $n-1$.

Write $\mathrm{CX}_n(d) = \sum_{i=0}^{n-2} a_{n,i} d^{n-i-1}$. There are formulae for the sequences of coefficients of these polynomials. We show that the nth element of $(a_{n,0}) = (1, 1, 2, 16/3, 50/3, \ldots)$ is $\mathrm{DX}(n, n-1)/(n-1)! = 2^{n-1}n^{n-3}/(n-1)!$. From (1) we infer the nth element of the sequence of second coefficients of $\mathrm{CX}_n(d)$, $(0, 0, -1, 15/2, -42, \ldots)$: $a_{n,1} = \mathrm{DX}(n, n-1)\frac{\sum_{i=0}^{n-2}(-i)}{(n-1)!} + \mathrm{DX}(n, n-2)\frac{1}{(n-2)!} = -3 \cdot 2^{n-3}n^{n-5}(2n-3)/(n-3)!$. Similarly, $a_{n,2} = \mathrm{DX}(n, n-1)\frac{\sum_{0 \le i < j \le n-2} ij}{(n-1)!} + \mathrm{DX}(n, n-2)\frac{\sum_{i=0}^{n-3}(-i)}{(n-2)!} + \mathrm{DX}(n, n-3)\frac{1}{(n-3)!} = \frac{2^{n-6}n^{n-7}(108n^3-463n^2+1122n-1560)}{3(n-4)!}$, which is the nth element in $(0, 0, 0, 19/6, 239/6, 986/3, \ldots)$. Again, $\lim_{n\to\infty}\frac{a_{n+1,0}}{a_{n,0}} = \lim_{n\to\infty}\frac{a_{n+1,1}}{a_{n,1}} = \lim_{n\to\infty}\frac{a_{n+1,2}}{a_{n,2}} = 2e$.

4 Columns, or: Growth Rate

We prove that the asymptotic growth rate of d-D polycubes is $2ed - o(d)$, and estimate it at $(2d-3)e$.

Theorem 2 *For any fixed value of* d, $\lim_{n\to\infty} \sqrt[n]{\mathrm{CX}(n, d)} \le (2d-1)e$.

Theorem 3 $\lim_{d\to\infty} \lambda_d/d \ge 2e$.

The main difficulty is to justify the exchange of limits in $\lim_{d\to\infty}\lim_{n\to\infty}\sqrt[n]{\mathrm{CX}(n, d)}/d$. The proof depends only on the formula for $a_{n,0}$. Combining Theorems 2 and 3 gives our main result.

Theorem 4 $\lambda_d = 2ed - o(d)$ *as* $d \to \infty$.

This was already observed, but not proven rigorously, in the statistical-physics literature. We know the first three coefficients of $\mathrm{CX}_n(d) = a_{n,0}d^{n-1} + a_{n,1}d^{n-2} + a_{n,2}d^{n-3} + \cdots$. Considering n fixed, we write $\mathrm{CX}_{n+1}(d)/\mathrm{CX}_n(d) = f_1(n)d + f_2(n) + \frac{f_3(n)}{d} + O\left(\frac{1}{d^2}\right)$, where $f_1(n) = 2((n+1)/n)^{n-2}$, $f_2(n) = -3(n-1)(n+1)^{n-4}(2n^3+6n^2-5n-6)/(2n^n)$, and $f_3(n) = -(n-2)(n-1)(n+1)^{n-6}(31n^6+217n^5-4888n^4+743n^3+13209n^2+12660n+3708)/(48n^{n+2})$. It is now easy to verify that $\lim_{n\to\infty} f_1(n) = 2e$, $\lim_{n\to\infty} f_2(n) = -3e$, and $\lim_{n\to\infty} f_3(n) = -31e/48$. Hence, we conjecture that $\lambda_d = (2d-3)e + O(1/d)$, and thus λ_d tends to $(2d-3)e$ as $d \to \infty$. We are able to show how this agrees with (1).

References

[1] G. Aleksandrowicz and G. Barequet, Counting polycubes without the dimensionality curse, *Discrete Math.*, to appear.

[2] G. Barequet, M. Moffie, A. Ribó, and G. Rote, Counting polyominoes on twisted cylinders, *Integers* **6** (2006) #A22.

[3] S.R. Broadbent and J.M. Hammersley, Percolation processes: I. Crystals and mazes, *Proc. Cambridge Philosophical Society* **53** (1957) 629–641.

[4] D.S. Gaunt, The critical dimension for lattice animals, *J. Phys. A: Math. Gen.* **13** (1980) L97–L101.

[5] D.S. Gaunt and P.J. Peard, $1/d$-expansions for the free energy of weakly embedded site animal models of branched polymers, *J. Phys. A: Math. Gen.* **33** (2000) 7515–7539.

[6] D.S. Gaunt, M.F. Sykes, and H. Ruskin, Percolation processes in d-dimensions, *J. Phys. A: Math. Gen.* **9** (1976) 1899–1911.

[7] I. Jensen, Counting polyominoes: A parallel implementation for cluster computing, *Proc. Int. Conf. on Computational Science*, III (Melbourne, Australia and St. Petersburg, Russia, 2003), *LNCS*, 2659, Springer, 203–212.

[8] D.A. Klarner, Cell growth problems, *Canadian J. of Mathematics* **19** (1967) 851–863.

[9] D.A. Klarner and R.L. Rivest, A procedure for improving the upper bound for the number of n-ominoes, *Canadian J. of Mathematics* **25** (1973) 585–602.

[10] W.F. Lunnon, Counting multidimensional polyominoes, *The Computer Journal* **18** (1975) 366–367.

[11] N. Madras, A pattern theorem for lattice clusters, *Annals of Combinatorics* **3** (1999) 357–384.

Six-way Equipartitioning by Three Lines in the Plane

William Steiger* Mario Szegedy Jihui Zhao

October 17, 2009

In 1949 Buck and Buck [7] showed that for any convex body K in the plane, there always exists three concurrent lines that equipartition K; that is, each of the six sectors they define cuts off the same area in K. Courant and Robbins [8] later gave a simple proof by continuity, and there followed a succession of interesting results [1], [2], [3], [4], [5], [6], [12], [13], [15], [16], [17], [19], [20] about the possibilities/impossibilities of partitioning measures in various ways, facts that are consequences of analysis, topology, and algebra (see especially Matoušek's beautiful book on the Borsuk-Ulam theorem [14]).

Here we reopen an apparently unexplored aspect of the six-way equipartitioning. First, we will assume area$(K) = 1$. In general, three lines will divide the plane into seven regions. The central, bounded one is a triangle τ that degenerates into a point if the lines are concurrent. In [7] Buck and Buck also showed that no convex set can be partitioned by three lines into seven regions, each with area $1/7$. They then asked whether there are partitions where six of the seven regions each has area $(1-t)/6$, and the seventh has area t; the previous statement shows this is impossible for every K when $t = 1/7$, and their original result shows it is always possible for every K when $t = 0$. Finally they showed that if such a six-way partitioning of K did exist for some $t > 0$, then it must be the central triangle τ that has area t.

They stated the conjecture that in every six-way equipartition with three lines, area$(\tau) = t$ is at most $t_0 = 1/49$. This is the value that does occur when K is, itself, a triangle; it is easily seen that this K may be six-way equipartitioned by lines parallel to its own sides. The fact that t_0 is the maximum central triangle area over any possible six-way equipartition of any convex body K was later proved by Sholander [18] (see also [9], [10], [11]).

The present paper deals with the existence of such partitions. To make things concrete we give

Definition 1: *Given a convex body K with area$(K) = 1$, lines ℓ_1, ℓ_2, ℓ_3 form a six-way partition of K if (i) the points $P_{ij} = \ell_i \cap \ell_j, i < j$ are in K, (ii) the triangle $\tau = \Delta P_{12}P_{13}P_{23}$ has area t, and (iii) each of the other six regions of $K \setminus \tau$ has area $(1-t)/6$.*

The main fact is

Theorem 1 *Given a convex body $K \subset R^2$ with area$(K) = 1$ and a unit vector $v \in R^2$, there exists a unique trio of lines that form a six-way partition of K, with one of them having normal vector v.*

According to Theorem 1, for each convex body K and $\theta \in [0, 2\pi)$ there is a unique six-way partition of K where one of the lines has normal vector $v = (\cos(\theta), \sin(\theta))$ and we write $f_K(\theta)$ as the area t of the central triangle in the partition. We can use this function to characterize certain convex sets K. In addition there is an interesting algorithmic problem when discussing six-way partitions of n points in general position in R^2.

*Department of Computer Science, Rutgers University, 110 Frelinghuysen Road, Piscataway, New Jersey 08854-8004; ({steiger,szegedy,zhaojih}@cs.rutgers.edu.)

References

[1] D. Avis. On the Partitionability of Point Sets in Space. *Proc. First ACM Symposium on Computational Geometry*, 116-120, 1985.

[2] I. Bárány, A. Hubard, and J. Jerónimo. " Slicing Convex Sets and Measures by a Hyperplane". *Discrete and Computational Geometry 39 (1-3)*, 67-75 (2008).

[3] I. Bárány and J. Matoušek. Equipartition of Two Measures by a 4-Fan. *Discrete and Computational Geometry 27*, 293-301 (2002).

[4] I. Bárány and J. Matoušek. "Simultaneous Partitions of Measures by $k-$fans". *Discrete and Computational Geometry 25 (3)*, 317-334 (2001).

[5] S. Bereg. "Equipartitions of Measures by 2-Fans". *Discrete and Computational Geometry 34 (1)*, 87-96 (2005).

[6] S. Bespamyatnikh, D. Kirkpatrick, and J. Snoeyink. Generalizing Ham-Sandwich Cuts to Equitable Subdivisions. *Discrete and Computational Geometry 24*, 605-622 (2000).

[7] R. Buck and E. Buck. "Equipartitioning of Convex Sets". *Math. Mag. 22 (4)*, 195-198 (1949).

[8] R. Courant and H. Robbins. What is Mathematics? *Oxford University Press*, (1941).

[9] H. Eggleston. "Some Properties of Triangles as Extreme Convex Curves". *J. London Math. Soc. 28*, 32-36 (1953).

[10] H. Eggleston. *Problems in Euclidean Space: Applications of Convexity.* Pergamon Press, (1957).

[11] H. Eggleston. *Convexity, Cambridge Tracts in Math. and Math. Physics 47*, Cambridge Univ. Press (1958).

[12] A. Kaneko and M. Kano. Balanced Partitions of Two Sets of Points in the Plane. *Comp. Geo.. Theory and Application 13*, 253-261 (1999).

[13] C.-Y. Lo, J. Matoušek, and W. Steiger. "Algorithms for Ham-Sandwich Cuts". *Discrete and Computational Geometry 11*, 433-452 (1994).

[14] J. Matoušek. *Using the Borsuk-Ulam Theorem.* Springer-Verlag (2003).

[15] S. Roy and W. Steiger. "Some Combinatorial and Algorithmic Aspects of the Borsuk-Ulam Theorem". *Graphs and Combinatorics 23*, 331-341, (2007).

[16] T. Sakai. Balanced Convex Partitions of Measures in R^2. *Graphs and Combinatorics 18*, 169-192 (2002).

[17] L. Schulman. An Equipartition of Planar Sets *Discrete & Computational Geometry 9*, 257-266, (1992).

[18] M. Sholander. "Proof of a Conjecture of R.C. Buck and E.F. Buck". *Math. Mag. 24 (1)*, 7-19 (1950).

[19] W. Steiger and J. Zhao. "Generalized Ham-Sandwich Cuts for Well-Separated point Sets". *Proceedings of the* 20^{th} *Canadian Conference on Computational Geometry, Montreal*, August 2008.

[20] W. Steiger and J. Zhao. "Generalized Ham-Sandwich Cuts". *Discrete and Computational Geometry* (to appear).

Guarding Polyominoes

Mohammad Irfan* Justin Iwerks† Joondong Kim† Joesph S. B. Mitchell†

Abstract

We explore the art gallery problem for the special case that the domain (gallery) P is an *m-polyomino*, consisting of a connected union of m 1×1 unit squares ("pixels"). We study the combinatorics of guarding polyominoes in terms of the parameter m, in contrast with the traditional parameter n, the number of vertices of P; in particular, we show that $\lceil \frac{m-1}{3} \rceil$ point guards are always sufficient and sometimes necessary to guard an m-polyomino. We consider both *point guards* and *pixel guards*. We show that determining the pixel guard number for a given m-polyomino is NP-hard. We give a simple greedy algorithm for finding the optimal point guard number for the special case of *thin polyomino paths*, and give approximation algorithms for guarding general *thin polyominoes*.

Keywords: art gallery problem, computational geometry, visibility problem, polyomino

1 Introduction

The problem of determining the minimum number of guards sufficient to cover the interior of an art gallery modeled as a simple polygon with n vertices was originally posed in 1973 by Victor Klee. The solution to this problem, first proven by Vasek Chvátal, was shown to be that $\lfloor \frac{n}{3} \rfloor$ guards are sometimes necessary and always suffice to cover a polygon possessing n vertices[2]. This original problem has since grown into a significant area of study in computational geometry and computer science. Art gallery problems are of theoretical interest but also play a central role in visibility problems arising in applications in robotics, digital model capture, sensor networks, motion planning, vision, and computer-aided design [3, 4].

Here, we analyze a variation of Klee's art gallery problem in which we are given a connected union of m (unit square) pixels, called an *m-polyomino P* (see Figure 1). We state the problem formally as follows:

Problem Statement: Let P be an m-polyomino. A point $a \in P$ *covers* a point $b \in P$ iff the line segment $ab \subseteq P$. (Possibly ab contains points on the boundary, ∂P.) Define $G(P)$ to be the minimum number of guards required to cover all of P: $G(P) = \min_k(\forall b \in P, \exists a_i \in P, 1 \leq i \leq k : a_i b \subseteq P)$. Then we define $g(P)$ to be the maximum value of $G(P)$ over all m-polyominoes: $g(P) = \max_P(G(P))$. For a given positive integer m, what is $g(P)$?

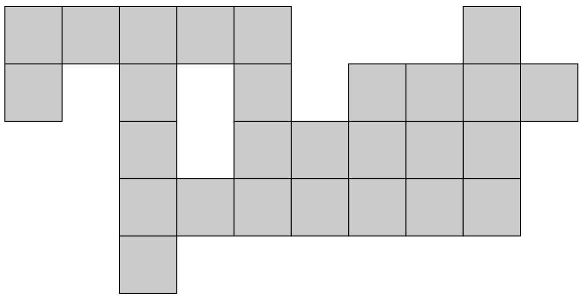


Figure 1: A polyomino with 27 pixels

We first give sufficient and sometimes necessary guard numbers as a function of m for covering P when using *point guards* or *pixel guards*. As its name suggests, a point guard occupies a point location in P and can see in every direction without limit on distance. A pixel guard occupies an entire pixel; any point $b \in P$ that sees any point of the pixel is covered by the pixel guard.

Secondly, we show that determining an optimal pixel guard placement in an m-polyomino is NP-hard. Lastly, we consider special cases of *thin m-polyominoes*, for which the dual graph associated with the pixels is a tree. In particular, we show that a simple greedy algorithm yields an optimal set of point guards in a *thin m-polyomino path* (see Figure 2), and we give a constant-factor approximation algorithm for optimizing point guards in general thin m-polyominoes.

*Dept. Computer Science, Stony Brook University, mtirfan@cs.sunysb.edu

†Dept. Applied Mathematics and Statistics, Stony Brook University, {jiwerks,jdkim,jsbm}@ams.sunysb.edu

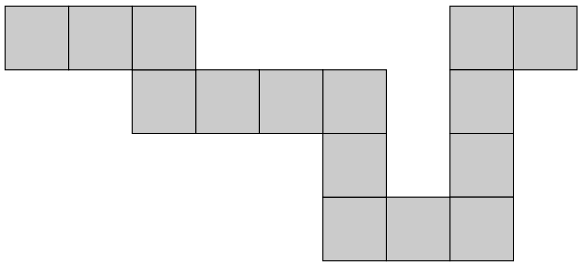


Figure 2: A thin polyomino path

2 Results

For an m-polyomino P where $m \geq 2$ we demonstrate that $\lceil \frac{m-1}{3} \rceil$ point guards is always sufficient and sometimes necessary to cover P. The inductive proof of this sufficiency condition is obtained by arguing that one can take any m-polyomino (holes allowed) and iteratively remove certain types of *subpolyominoes.* In order for this induction to work, we must be able to find and remove a pixel to use the inductive hypothesis. This property is found in the following lemma:

Lemma 1 *Let P be an m-polyomino with $m \geq 2$. Then $\exists$ a pixel p that can be removed from P yielding a connected $(m-1)$-polyomino P'.*

With the help of this lemma and three others that address certain details (not discussed in this abstract) on decomposing and guarding certain varieties of polyominoes, we state the crux of the argument in the next theorem:

Theorem 2 *For any m-polyomino P with $m \geq 3$, subpolyominoes having 3, 4, 5 (with two exceptions), 6, 7, and a special case of 8 and a special case of 9 pixels can be removed from P yielding connected polyominoes $P_1, P_2, ..., P_f$ where P_i is the connected polyomino remaining after removing the i^{th} subpolyomino from P. P_f may contain 0, 1, or 2 pixels.*

The subpolyomino varieties listed above obey a certain property: the ratio between guards required to cover the subpolyomino and the number of pixels the subpolyomino possesses never exceeds 1:3. Details involving the boundary conditions give us the -1 term in the numerator and the ceiling function. We summarize this and an immediate result about pixel guards in the following corollary:

Corollary 3 *Given an m-polyomino P with $m \geq 2$, $\lceil \frac{m-1}{3} \rceil$ point guards is sufficient and sometimes necessary to cover P. Also, $\lceil \frac{m-1}{3} \rceil$ pixel guards is sufficient to cover P.*

However, we give a construction that establishes $\lceil \frac{m+1}{11} \rceil + \lceil \frac{m+5}{11} \rceil + \lceil \frac{m+9}{11} \rceil$ pixel guards as sometimes necessary for $m \geq 2$, leaving a gap between our necessary and sufficient conditions. In the event that $m = 1, 2$, one guard of any type is trivially both necessary and sufficient to guard P.

By a reduction from the minimum line cover problem (find a smallest set of points that hit a given set of lines in the plane), which is known to be APX-hard, we establish that finding the optimal number of pixel guards for an m-polyomino P is NP-hard and hard to approximate.

Theorem 4 *Given an m-polyomino P and an integer k, it is NP-hard to determine whether k pixel guards are enough to cover P. Further, there is an $\epsilon > 0$ such that it is NP-hard to determine $k^*(1+\epsilon)$ guards covering P where k^* is the minimum number of pixel guards for P. The same statements hold for point guards.*

Finally, we examine special cases of m-polyominoes. We show that a greedy algorithm yields the minimum point guard number for a thin m-polyomino path. The strategy employed by the algorithm is to iteratively remove largest possible *star-shaped* (one guard coverable) subpolyominoes beginning at one end of the path and working toward the other. For general thin polyominoes, we show that a related strategy yields a constant-factor approximation of the optimal guard number.

Acknowledgements. This work was partially supported by NSF (CCF-0528209, CCF-0729019), NASA Ames, and Metron Aviation.

References

[1] M, J. Katz and G. S. Roisman. On Guarding the Vertices of Rectilinear Domains. *Computational Geometry* 39(3):219–228, 2008.

[2] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford: Oxford University Press, 1987.

[3] T. C. Shermer. Recent Results in Art Galleries. *Proceedings of the IEEE*, 80(9):1384-1399, Sept. 1992.

[4] J. Urrutia. Art gallery and illumination problems. In J. Sack and J. Urrutia, editors, *Handbook of Computational Geometry,* pages 973-1027. Elsevier Science publishers, Amsterdam, 2000.

Drawing Graphs with 90° Crossings and at Most 1 or 2 Bends per Edge[*]

Karin Arikushi[†] Csaba D. Tóth[‡]

Abstract

Let G be a graph with n vertices and m edges. Didimo et al. showed that if G admits a right angle crossing (RAC) drawing where the edges have at most 1 bend or 2 bends, then $m < 3.1n^{4/3}$ or $m < 36n^{7/4}$, respectively. We improve these bounds on the number of edges and show that $m \leq 21n$ and $m \leq 150n$.

1 Introduction

The problem of drawing simple graphs in the plane where the edges are poly-lines and crossings are orthogonal has been studied by Didimo et al. [2]. In particular, they consider graph drawings where the edges can have at most 0, 1, 2 or 3 bends. It has been shown that any simple graph can be drawn when 3 bends are allowed. Moreover, when no bends are allowed the number of edges is at most $4n - 10$, where n is the number of vertices. This bound is tight. In the case when 1 or 2 bends are allowed the number of edges m satisfies $m < 3.1n^{4/3}$ and $m < 36n^{7/4}$, respectively.

We improve the upper bound on the number of edges when 1 or 2 bends are permitted. We show that $m \leq 21n$ when edges have at most 1 bend, and that $m \leq 150n$ when edges have at most 2 bends.

2 Preliminaries

Let G be a graph with n vertices and m edges. The *crossing number* of any graph G, denoted cr(G), is the minimum number of edge crossings in any drawing of G in the plane. The Crossing Lemma, a lower bound for $\mathrm{cr}(G)$, was established independently by Ajtai et al. [1] and Leighton [3]. The current strongest version is due to Pach et al. [4].

Lemma 1 *[4] Let G be a graph with n vertices and m edges. If $m \geq \frac{103}{6}n$, then $\mathrm{cr}(G) \geq 0.032\frac{m^3}{n^2}$.*

Let D be a drawing of G. We say that D is a *right angle crossing* (RAC) drawing of G if D is a poly-line drawing of G where any two crossing edge segments of G are orthogonal. In this paper, we assume that G admits an RAC drawing where edges have at most 1 bend or 2 bends. Thus, each edge has at most 2 or 3 segments, respectively.

Given a drawing D of G, we define a *block graph* of G, $B(G)$. A *block* of D is a connected component in the union of pairwise parallel or orthogonal segments in D. Formally, we define a symmetric binary relation on the segments in D where two segments are related if and only if they cross. The transitive closure of this relation is an equivalence relation. A block of D is defined as the union of all segments in an equivalence class.

Note that if $m \geq \frac{103}{6}n$, then the average number of crossings per segment is at least $\frac{.064m^2}{(b+1)n^2}$, where b is the maximum number of bends per edge. We say a segment is *heavy* if it crosses at least $\frac{.064c_1m^2}{n^2}$ other segments, for a constant c_1 specified later. A block is *heavy* if it contains a heavy segment. We define a bipartite graph $B(G)$. The vertices of $B(G)$ are the vertices of G and the heavy blocks of D. A vertex v of G and a heavy block are adjacent if a segment from the block is incident to v in G.

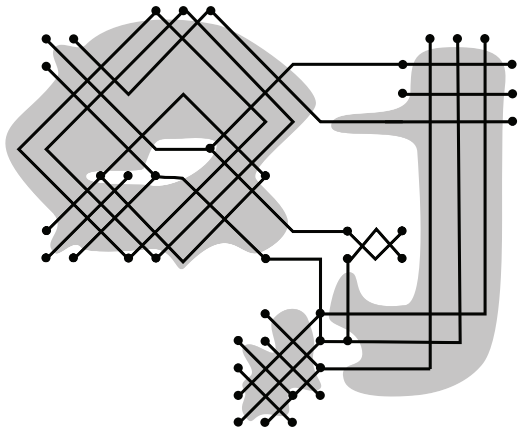


Figure 1: RAC drawing with 1 bend and heavy blocks

Lemma 2 *If G admits an RAC drawing D where each edge has at most 1 or 2 bends, then a block graph of G is planar.*

3 Main Results

Theorem 3 *Let G be a graph with n vertices and m edges that admits an RAC drawing with at most 1 bend per edge. Then, $m \leq 21n$.*

[*]Supported in part by NSERC grant RGPIN 35586.

[†]Department of Mathematics and Statistics, University of Calgary, karikush@math.ucalgary.ca

[‡]Department of Mathematics and Statistics, University of Calgary, cdtoth@ucalgary.ca

Theorem 4 *Let G be a graph with n vertices and m edges that admits an RAC drawing with at most 2 bends per edge. Then, $m \leq 150n$.*

We introduce the following notation. Let n' and m' be the number of vertices and edges of $B(G)$, respectively. Let p be the number of segments that participate in some heavy block and let H be the number of heavy blocks. To prove Theorem 3 we require some bounds on the parameters H and p.

Let G be a graph with an RAC drawing D with at most one bend per edge. We say a segment is heavy if it crosses at least $\frac{.064m^2}{4n^2}$ others. Thus, the number of heavy blocks, H, can be bounded above by

$$\begin{aligned} H &\leq \frac{\text{number of segments in all heavy blocks}}{\text{number of segments in a smallest heavy block}} \\ &\leq \frac{p}{.016m^2/n^2} = \frac{pn^2}{.016m^2}. \end{aligned}$$

Lemma 5 *Let G be a graph with n vertices and $m > 21n$ edges that admits an RAC drawing with at most one bend per edge. Then, the number of edges participating in some heavy block is at least $\frac{m}{6}$. In particular, $\frac{m}{6} \leq p$.*

Proof. Suppose that the number of edges e participating in some heavy block is less than $\frac{m}{6}$. Let G' be the graph obtained from G by deleting the e edges. The number of edges in G' is greater than $\frac{5m}{6} > \frac{5\cdot 21n}{6} = \frac{105n}{6}$. By the Crossing Lemma, the average number of crossings per segment in G' is greater than $.032\left(\frac{5m}{6n}\right)^2$.

There exists a segment in the new graph with more than $.032\left(\frac{5m}{6n}\right)^2$ crossings. This segment is heavy, which cannot be since we deleted all edges in heavy blocks. Thus, there are at least $\frac{m}{6}$ edges participating in some heavy block. Since each edge contains at least one segment, $\frac{m}{6} \leq p$. □

We now prove Theorem 3.

Proof. Suppose $m > 21n$ and consider a block graph $B(G)$ with $n' = H + n$ vertices and m' edges. Since $B(G)$ is planar and bipartite, we have

$$m' \leq 2n' - 4 < 2n' \leq 2(H+n).$$

Since $p \leq m'$ and $\frac{m}{6} \leq p$ we obtain

$$\begin{aligned} p &< 2\left(\frac{pn^2}{.016m^2} + n\right) < 2\left(\frac{pn^2}{.016m^2} + \frac{m}{21}\right) \\ &\leq 2\left(\frac{pn^2}{.016m^2} + \frac{6p}{21}\right). \end{aligned}$$

Rearranging this inequality, we see that

$$m^2 < \frac{1}{.016}\left(\frac{1}{2} - \frac{6}{21}\right)^{-1} n^2.$$

Therefore, $m < 17.07n$. This contradicts our assumption that $m > 21n$ so in fact, $m \leq 21n$. □

The proof of Theorem 4 is similar to that of Theorem 3 so we only give a sketch. When G admits an RAC drawing with at most 2 bends, we say a segment is heavy if it crosses at least $\frac{.064m^2}{30n^2}$ other segments. Note that every edge has two *end segments* and one *middle segment*. We have two cases: 1) at least $\frac{p}{20}$ segments in heavy blocks are end segments, or 2) less than $\frac{p}{20}$ segments in heavy blocks are end segments. We also have the following lemma.

Lemma 6 *Let G be a graph with n vertices and $m > 150n$ edges that admits an RAC drawing with at most 2 bends per edge. Then, the number of edges participating in some heavy block is at least $\frac{m}{2}$.*

In Case 1, we can show that $m \leq 150n$ by applying the lemma and following the argument in the proof of Theorem 3.

We show that Case 2 cannot occur. In a heavy block, the edge segments can be partitioned into 2 sets of pairwise parallel segments. Let G' be the graph obtained from G by deleting all segments in the smaller partition (and the edges they are a part of) of all heavy blocks. Then, the number of deleted edges is at most

$$\begin{aligned} &\frac{1}{2}(\text{number of middle segments in heavy blocks}) \\ &+ \frac{1}{2}(\text{number of end segments in heavy blocks}) \\ \leq &\frac{1}{2}\left(m + \frac{m/20}{1-1/20}\right) = \frac{20m}{38}. \end{aligned}$$

Thus, G' has at least $\frac{18m}{38}$ edges. By the Crossing Lemma, the average number of crossings per segment is at least

$$\frac{.064(1-3/20)^2m^2}{3(2(1-1/20))^2n^2} \geq \frac{.064\cdot 17^2m^2}{3\cdot 38^2n^2},$$

which implies that G' has a heavy segment. However, we deleted edges so that there are no more heavy blocks and hence no heavy segments. Thus, Case 2 cannot occur.

References

[1] M. Ajtai, V. Chvátal, M. Newborn, E. Szemerédi, Crossing-free subgraphs, *Ann. Discrete Mathematics* 12 (1982), 9–12.

[2] W. Didimo, P. Eades, G. Liotta, Drawing graphs with right angle crossings (Extended Abstract), In: WADS 2009. *LNCS,* vol. 5664, 206–217.

[3] T. Leighton, *Complexity Issues in VLSI, Foundations of Computing Series,* MIT Press, Cambridge, MA, 1983.

[4] J. Pach, R. Radoicic, G. Tóth, Improving the crossing lemma by finding more crossings in sparse graphs. *Discrete Comput. Geom.* 36(4) (2006), 527–552.

A Tight Bound For Point Guards in Piece-Wise Convex Art Galleries

Javier Cano* Csaba D. Tóth† Jorge Urrutia*

1 Introduction

In the context of Art Gallery problems, Karavelas, Tóth and Tsigaridas [3] recently considered *Curvilinear Art Galleries*, that is, polygons in which the edges are locally *convex* or *concave* curves with respect to the interior of the gallery. They proved that when all the edges of a curvilinear art gallery are convex, $\lfloor \frac{2n}{3} \rfloor$ vertex guards are always sufficient and sometimes necessary to guard a curviliner art gallery. They also proved that $\lfloor \frac{2n}{3} \rfloor$ point guards are always sufficient and $\lceil \frac{n}{2} \rceil$ are sometimes necessary. Cano, Espinosa and Urrutia [2] improved the upper bound for point guards to $\lfloor \frac{5n}{8} \rfloor$.

In this paper we show that every curvilinear polygon P with n vertices, where every edge is a convex arc, can be monitored by a set of $\lceil \frac{n}{2} \rceil$ point guards. This is achieved by constructing a convex partition of P similar to that proposed by Al-Jubeh *et al.* [1].

2 Definitions

Let a be an arc of a curve. We say that a is a *convex arc* if the region bounded by a and the line segment joining its endpoints is convex. Let $V = \{v_0, v_1, \ldots, v_{n-1}\}$ be a set of n points on the plane together with a set of convex arcs $A = \{a_0, a_1, \ldots, a_{n-1}\}$ such that the endpoints of a_i are v_i and v_{i+1} (addition taken in modulo n), and no two arcs intersect with the possible exception of common endpoints. The region bounded by A is called a *piece-wise convex art gallery*. In what follows, we shall assume that all the edges of a curvilinear art gallery are convex with respect to the interior of the gallery.

A *convex partition* of a piece-wise convex art gallery P is a family of open convex sets C contained in the interior of P such that they are pairwise disjoint and the union of their closures is P. All convex partitions in this paper are obtained by partitioning the interior of P along directed edges starting at vertices of P. Those edges form a directed planar straight-line graph δ whose vertices are vertices of P or endpoints of directed segments (Steiner vertices). δ is restricted to be such that exactly one directed edge starts at every vertex of P and every Steiner vertex in the interior of P; no directed edge starts at Steiner points in the interior of the arcs of P.

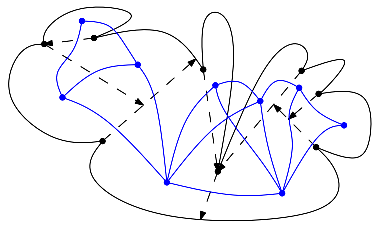


Figure 1: A convex partition of the interior of a Piece-wise Convex Polygon and its dual graph.

We define the *dual graph* $D(C)$ of a convex partition C as the graph whose vertices are the elements of C, two of which are adjacent if they have at least a common point on their boundaries. In particular, cells incident to a vertex or a Steiner point form a clique in $D(C)$.

For a cell $c \in C$ we will refer to its boundary as $B(c)$. A cell c such that $B(c)$ has at least two vertices of P will be called a *good cell*, else it is called a *bad cell*. Our proof is based on the following result, given without proof:

Lemma 1 *Any piece-wise convex polygon P with n vertices can be partitioned into $n+1$ good convex cells.*

It is possible that the boundary of a cell in this partition intersects the boundary of P at vertices only. Such a cell is called a *degenerate cell*.

3 Finding a Guarding Set

Let C be a convex partition of a piece-wise convex art gallery P as in Lemma 1. For a component t of δ, let $D(t)$ be the subgraph of $D(C)$ induced by the cells bounded by t. The next result, given without proof, will be useful.

Lemma 2 *$D(t)$ has a spanning subgraph $D'(t)$ isomorphic to a triangulated polygon.*

We will refer to the boundary of such triangulated polygon $D'(t)$ by $H(t)$. Next we prove:

Theorem 3 *Let P be piece-wise convex Art Gallery with n vertices. Then $\lceil \frac{n}{2} \rceil$ guards are always sufficient to guard P.*

*Instituto de Matemáticas, Universidad Nacional Autónoma de México, D.F. México, j_cano@uxmcc2.iimas.unam.mx, urrutia@matem.unam.mx

†Department of Mathematics, University of Calgary, cdtoth@math.ucalgary.ca

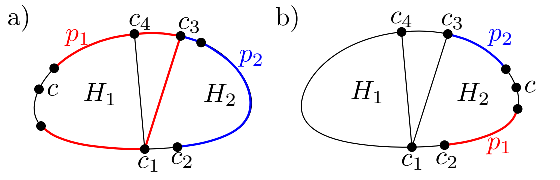


Figure 2: Illustration for case 2.

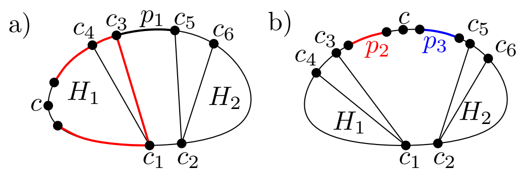


Figure 3: Illustration for case 3.

Proof. Let Γ be a graph whose vertices are the connected components of δ, two of which are joined by an edge if and only if at least an edge of each of them bound an element of C. Notice that Γ is a connected graph in which each 2-connected component is a clique induced by a convex cell c, whose vertices are the connected components of δ bounding c. Let Γ' be a spanning tree of Γ obtained during a BFS traversal of Γ.

The main idea is to traverse Γ' in a top-down fashion, placing at each step, guards for the cells bounded by each component of δ. Notice that when processing a component $t_i \in \Gamma'$ which has a child t_j, we can leave a cell adjacent to t_j unguarded, it will be guarded later when we process t_j. We distinguish two types of steps:

Unconstrained. There is no cell bounded by t_i guarded in a previous step. Let $n(t_i)$ be the number of cells bounded by t_i. If $n(t_i)$ is even one can find a perfect matching for $H(t_i)$. The cells bounded by t_i can be guarded by placing a guard in the common boundary of two matched cells. If $n(t_i)$ is odd, $D'(t_i)$ has an ear, and in this case, we can place a guard at the vertex of t_i incident to the three cells forming such an ear. The rest of $H(t_i)$ is a path with even length and those cells could be guarded as in above.

Constrained. There is a cell c bounded by t_i already guarded. Let $n(t_i)$ be the number of unguarded cells bounded by t_i. If $n(t_i)$ is even a perfect matching in $H(t_i) \setminus \{c\}$ allows us to guard the remaining cells with $n(t_i)/2$ guards. Suppose then that $n(t_i)$ is odd. If t_i does not bound a degenerate cell then there has to be an edge of t_i that ends at a point x on the interior of an arc of P. Let c_1 and c_2 be the cells incident to x. The following cases arise:

Case 1. c_1 and c_2 are bounded by some other component of δ. Then one of them, say c_j, together with c splits $H(t_i)$ into two paths of even length. Leave c_j unguarded and guard those paths.

Case 2. One of c_1 or c_2, say c_2, is bounded by other component of δ, but the other not. Let v be the first vertex in a backward path along the boundary of c_1 starting from x. Let c_3 and c_4 the two cells incident in v, adjacent in $H(t_i)$. W.l.o.g. suppose that c_1c_3 splits $H(t_i)$ into two subpolygons as in Figure 2. We have:

Case 2.1. $c \in H_1$. Consider the paths as shown in Figure 2a. If p_1 has even length then $p_2 \setminus \{c_2, c_3\}$ also has even length, match its vertices, and leave c_2 unguarded. If p_1 has odd length, then remove c_3 from p_1, and join c_1 with c_4, match p_1 and $p_2 \setminus \{c_2\}$ and leave c_2 unguarded.

Case 2.2. $c \in H_2$. This case is depicted in Figure 2b. If p_1 has odd lenght, match the vertices in p_1, leaving c_2 unguarded. If p_2 has even length match its vertices and remove c_3 from H_1. If p_2 has odd length remove c_3 from it and match the rest. H_1 is a simple polygon that could be guarded as in the unconstrained case.

Case 3. c_1 and c_2 are not bounded by other vertices of Γ. Then c_2 has to be adjacent to two cells, say c_5, c_6, such that the edge c_5c_6 is an edge of $H(t_i)$. Subcases arise, but only one is considered in this abstract:

Case 3.1. $c_3c_4 \neq c_5c_6$. Let $D(t_i)$ be as in Figure 3. The cases when c is in H_1 or H_2 are symmetric, so suppose $c \in H_1$, Figure 3a. Removing c from H_1 we obtain a path p_2. If p_2 has even length, then make $p_1 = p_1 \setminus c_3$, otherwise, remove c_3 from p_2 and join c_1 with c_4 in p_2. If p_1 has even length remove c_5 from H_2 and join c_2 with c_6, otherwise remove c_5 from p_1. Now p_1 and p_2 had even length and the vertices of H_2 induce a triangulated polygon, then can be guarded as in previous cases.

There are three remaining cases, when c is in p_1 (Figure 3b), when $c_3c_4 = c_5c_6$ and when t_i bounds a degenerate cell. Those cases can be solved using similar arguments as in the previous cases. Those cases are not presented for space reasons.

For each t_i, $\lfloor (n(t_i))/2 \rfloor$ guards have been placed. Clearly $\sum_{t_i \in \Gamma'} n(t_i) = n+1$. Thus at most $\lfloor (n+1)/2 \rfloor = \lceil n/2 \rceil$ guards have been placed. □

References

[1] M. Al-Jubeh, M. Hoffmann, M. Ishaque, D. L. Souvaine and Cs. D. Tóth. Convex partitions with 2-edge connected dual graphs. In *Proc. COCOON*, 2009, pp. 192–204.

[2] J. Cano, J. Espinosa and J. Urrutia. Vigilancia en Galerías de Arte Curvilíneas. In *Actas de los XIII Encuentros de Geometria Computacional*, 2009, pp. 59–66.

[3] M. I. Karavelas, Cs. D. Tóth and E. P. Tsigaridas. Guarding curvilinear art galleries with vertex or point guards. *Comput. Geom.* **42** (2009), 522–535.

IF YOU CAN HIDE BEHIND IT, CAN YOU HIDE INSIDE IT?

DANIEL A. KLAIN

Abstract. Suppose that K and L are compact convex subsets of Euclidean space, and suppose that, for every direction u, the orthogonal projection (that is, the shadow) of L onto the subspace $u^{\perp}$ normal to u contains a translate of the corresponding projection of the body K. Does this imply that the original body L contains a translate of K? Can we even conclude that K has smaller volume than L?

In other words, suppose K can "hide behind" L from any point of view (and without rotating). Does this imply that K can "hide inside" the body L? Or, if not, do we know, at least, that K has smaller volume?

Although these questions have easily demonstrated negative answers in dimension 2 (since the projections are 1-dimensional, and convex 1-dimensional sets have very little structure), the (possibly surprising) answer to these questions continues to be *No* in Euclidean space of any finite dimension.

In this talk I will give concrete constructions for convex sets K and L in n-dimensional Euclidean space such that each $(n-1)$-dimensional shadow of L contains a translate of the corresponding shadow of K, while at the same time K has strictly greater volume than L. This construction turns out to be sufficiently straightforward that a talented person could conceivably mold 3-dimensional examples out of modeling clay.

The talk will then address a number of related questions, such as: under what additional conditions on K or L does shadow covering imply actual covering? What bounds can be placed on the volume ratio of K and L if the shadows of L cover those K?

Department of Mathematical Sciences, University of Massachusetts Lowell, Lowell, MA 01854 USA

E-mail address: `Daniel_Klain@uml.edu`

Mesh-Enhanced Persistent Homology[*]

Benoît Hudson
bhudson@tti-c.edu

Gary L. Miller
glmiller@cs.cmu.edu

Steve Y. Oudot
steve.oudot@inria.fr

Donald R. Sheehy
dsheehy@cs.cmu.edu

Abstract

Existing algorithms for computing the persistent homology of a set of n points in $\mathbb{R}^d$ can take up to $n^{\Omega(d)}$ time and space in the worst case. We apply ideas from mesh generation to reduce the time to $2^{O(d)}n + O(n \log \Delta)$ and the space to $2^{O(d)}n^2$, where Δ denotes the spread of the point cloud. This makes the problem of computing the full persistence barcode tractable for data in (moderate) dimensions higher than 3. Our technique is based on a new filtration, the α-mesh filtration, that can be intertwined with the offset filtration and avoids the $n^{O(d)}$ blowup of the α-complex.

1 Introduction

Persistent homology is a powerful tool for understanding the intrinsic structure of a set of data points at different scales. Unfortunately, existing methods for computing the persistent homology of points in $\mathbb{R}^d$ can take up to $n^{\Omega(d)}$ time and space, thus limiting our ability to use it in even moderate dimensions. Here we show how to reduce the dependence of both time and space on the ambient dimension to $2^{O(d)}$. We achieve this by preprocessing the points using Delaunay mesh refinement, then building a filtration on the mesh such that the persistent homology of the mesh closely approximates the persistent homology of the offsets of the input points. This makes it practical to compute persistent homology at all scales in higher dimensions than was previously possible.

The homology groups give an algebraic description of the shape of a topological space. For sets in $\mathbb{R}^3$, the 0th, 1st, and 2nd homology groups describe the connected components, tunnels, and voids respectively. Among topological invariants, homology stands out for its ease of computation: indeed, computing the homology of a simplicial complex reduces to a sequence of simple operations in linear algebra [8].

Persistent homology is useful when the input is only a point sample rather than a full complex. A sequence of complexes are constructed on the points at different scales. The persistent homology captures those topological features that remain over a significant range of scales (see [13, 6] for surveys).

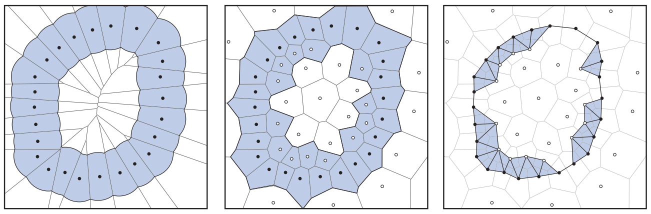


Figure 1: From left to right: The offset filtration, P^α; the α-Voronoi filtration, V^α; and the α-mesh, D^α.

A *filtration* is a nested sequence of topological spaces. The *offset filtration* of a point set P is the filtration whose elements, P^α, are the unions of α-radius balls centered at the points of P (See Figure 1, left). The α-complex filtration is a filtration of the Delaunay triangulation of P that is known to have the same persistent homology as the offset filtration [5]. Unfortunately, the size of the Delaunay triangulation can be up to $\Omega(n^{\lceil d/2 \rceil})$, even when the points of P are distributed along some submanifold of $\mathbb{R}^d$ [1].

Sparse meshing technology can bypass the worst-case blow-up in the size of the Delaunay triangulation while maintaining many of the same properties of the original Delaunay triangulation [9]. It does this by adding extra points to the input in such a way that the Delaunay triangulation complexity is less for the superset than for the input. Under very mild sampling assumptions, the size of the superset is at most $2^{O(d)}n$ [10].

2 The α-mesh filtration

Constructing the α-mesh filtration starts with a quality mesh of the input points. Let $P \subset \mathbb{R}^d$ be the input points. Let $M \supset P$ be a superset of the input with the property that for each v in M, the ratio of the radii of the smallest enclosing ball and largest enclosed ball, both centered at v, of the Voronoi cell of v is bounded by a constant τ. The set M and its Voronoi diagram constitute a τ-quality mesh. The sizes of P and M are

[*]This work was partially supported by the National Science Foundation under grant number CCF-0635257.

denoted by n and m respectively. The points of $M \setminus P$ are known as *Steiner points*.

The mesh M can be filtered to closely approximate the offset filtration P^α of P. The result is called the α-mesh filtration and is denoted D^α. For each Steiner point s, we assign a value $t(s)$ to be the distance from s to P. For each input point p, we let $t(p)$ be half the distance from p to $M \setminus \{p\}$. The filtration D^α is the set of input vertices as well as all simplices σ of $Del(M)$ such that each $v \in \sigma$ has $t(v) \leq \alpha$ (See Figure 1, right).

Why it works. The α-mesh has a natural dual formed by the union of Voronoi cells of the vertices $v \in M$ with $t(v) \leq \alpha$ (See Fig. 1, middle). For technical reasons, we add to this union the open balls of radius $\min\{\alpha,\ t(p)\}$ centered at the input points $p \in P$, which does not affect the combinatorial structure of the dual, and we call *α-Voronoi filtration V^α* the obtained filtration. Ignoring some technicalities involving the boundary of the mesh, V^α and P^α are *multiplicatively τ-interleaved*:

$$\forall \alpha \geq 0,\ P^{\alpha/\tau} \subseteq V^\alpha \subseteq P^{\alpha\tau}. \tag{1}$$

Consider now $P^\alpha_{\log}$ and $V^\alpha_{\log}$, the reparametrizations of the filtrations P^α and V^α over a logarithmic scale:

$$\forall \alpha \geq 0,\ P^\alpha_{\log} = P^{2^\alpha} \text{ and } V^\alpha_{\log} = V^{2^\alpha}.$$

Multiplicative τ-interleaving of P^α and V^α in the sense of Eq. 1 implies *additive* $\log(\tau)$*-interleaving* of their reparametrizations $P^\alpha_{\log}$ and $V^\alpha_{\log}$:

$$\forall \alpha \geq 0,\ P^{\alpha - \log\tau}_{\log} \subseteq V^\alpha_{\log} \subseteq P^{\alpha + \log\tau}_{\log}.$$

Additively $\log(\tau)$-interleaved filtrations are known to have $\log(\tau)$-close persistence diagrams in the bottleneck distance [2, 4]. In addition, the Persistent Nerve Lemma of Chazal and Oudot [3, Lem. 3.4] implies that $V^\alpha_{\log}$ and $D^\alpha_{\log}$ (the log-reparametrization of D^α) have identical persistence diagrams. Therefore, the persistence diagrams of $D^\alpha_{\log}$ and $P^\alpha_{\log}$ are $\log(\tau)$-close in the bottleneck distance.

3 Time and Space complexities

Computing the persistence barcode involves a variant of Gaussian elimination of a matrix with dimensions proportional to the number of mesh simplices. Since M is well-spaced, the number of simplices is $2^{O(d)}m$ [12]. Assuming the input point set P is an (ϵ, δ)-sample as is often assumed (or, more generally, a *well-paced point set* [11]), then $m \in 2^{O(d)}n$ [10]. Thus, the persistence computation involves a matrix in $2^{O(d)}n$ dimensions. In theory, this could take as much as $2^{O(d)}n^2$ space and $2^{O(d)}n^3$ time using the persistence algorithm [7, 14]. In practice, near-linear running time and memory usage have been observed.

We know how to compute a small but well-spaced superset and its Delaunay complex in $2^{O(d)}n$ space and $2^{O(d)}n\log(\Delta)$ time [9], where Δ is the spread of the input—the ratio of the largest to smallest interpoint distances. The spread is usually small: on a regular (ϵ, δ)-sample, it is at most $O(n)$ so that meshing takes $2^{O(d)}n\log(n)$ time. These space and time bounds imply our technique is practical in moderate dimensions, certainly up to six.

4 Extensions

The methods presented here have several natural extensions. A tighter approximation to the offset filtration is possible by refining the mesh. The α-mesh and the offset filtration can achieve a multiplicative $(1+\varepsilon)$-interleaving at a cost of $O(\varepsilon^{-d})$ blowup in the complex size. Also, it is possible to compute the persistent homology of the complement of the offset filtration by filtering the Voronoi diagram of V *backwards*. A third extension is the complete elimination of any assumptions on the sampling by applying linear-size Delaunay meshing technology [11].

References

[1] N. Amenta, , D. Attali, and O. Devillers. Complexity of delaunay triangulation for points on lower-dimensional polyhedra. In *SODA*, 2007.

[2] F. Chazal, D. Cohen-Steiner, M. Glisse, L. J. Guibas, and S. Y. Oudot. Proximity of persistence modules and their diagrams. In *SOCG*, 2009.

[3] F. Chazal and S. Y. Oudot. Towards persistence-based reconstruction in Euclidean spaces. In *SOCG*, 2008.

[4] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Discrete Comput. Geom.*, 37(1):103–120, 2007.

[5] H. Edelsbrunner. The union of balls and its dual shape. In *SOCG*, 1993.

[6] H. Edelsbrunner and J. Harer. Persistent homology – a survey. In J. E. Goodman, J. Pach, and R. Pollack, editors, *Twenty Years After*. AMS Press, 2007.

[7] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete Comput. Geom.*, 28:511–533, 2002.

[8] J. Friedman. Computing Betti numbers via combinatorial Laplacians. In *STOC*, 1996.

[9] B. Hudson, G. Miller, and T. Phillips. Sparse Voronoi Refinement. In *Proceeding of the International Meshing Roundtable*, 2006. Long version available as Carnegie Mellon University Technical Report CMU-CS-06-132.

[10] B. Hudson, G. L. Miller, T. Phillips, and D. R. Sheehy. Size complexity of volume meshes vs. surface meshes. In *SODA*, 2009.

[11] G. L. Miller, T. Phillips, and D. R. Sheehy. Linear-size meshes. In *CCCG*, 2008.

[12] G. L. Miller, D. Talmor, S.-H. Teng, and N. Walkington. A Delaunay based numerical method for three dimensions: generation, formulation, and partition. In *STOC*, 1995.

[13] A. Zomorodian. *Topology for Computing*. Cambridge Univ. Press, 2005.

[14] A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete Comp. Geom.*, 33(2):249–274, 2005.

Fast Meshing for Acute PLCs

Gary L. Miller Todd Phillips
Carnegie Mellon University

October 20, 2009

Abstract

We present a new algorithm to mesh an arbitrary piecewise linear complex in three dimensions. The algorithm achieves an $O(n \log \Delta + m)$ runtime where n, m, and Δ are the input size, the output size, and spread respectively. This represents the first non-trivial runtime guarantee for this class of input. The new algorithm extends prior work on runtime-efficient meshing by allowing the input to have acute input angles (called creases). Features meeting at creases are handled with protective collars. A new procedure is given for creating these collars in an unstructured fashion, without the need for expensive sizing precomputation as in prior work. The collar surface dividing these two regions is represented implicitly using surface reconstruction techniques. This new approach allows the collar to be dynamically generated , allowing the whole algorithm to run in a single pass. For inputs with Δ bounded by a polynomial in n, this runtime is optimal.

1 Motivation

Conventional wisdom has often regarded the meshing problem as a pre-process to a series of heavy computations on the output mesh. However, state of the art problems now rely on evolving geometries, particularly for simulation problems involving fluid-solid interactions. In such simulations, a new mesh must be generated at every simulation timestep, either dynamically or from scratch. Additionally, the linear system solves which traditionally dominate compute time are increasingly fast, with modern solvers running at least sub-quadratic in the mesh size.

This creates a need for advanced algorithms that are efficient enough to avoid dominating the overall runtime. The Sparse Voronoi Refinement (SVR) algorithm [3] represented the first algorithm for meshing non-acute PLCs in more than two dimensions with a non-trivial runtime bound. We have extended this algorithm to handle acute PLCs in three dimensions with the same overall runtime guarantees.

2 Algorithm Overview

We follow a standard iterative Delaunay (Voronoi) refinement paradigm, wherein we continually add new vertices to a mesh with the two goals of increasing element quality, and conforming to an input PLC. The algorithmic difficulties for handling acute PLCs arise due to a conflict between these two goals. It is impossible to fit quality tetrahedra inside creases of the PLC. This is circumvented by giving two different guarantees on element quality.

Adjacent to the creases of the input, we generate tetrahedra with no large dihedral angles. In regions disjoint from the creases, we generate tetrahedra with a good ratio of circumradius to shortest edge. These are almost all good aspect ratio tetrahedra, with the exception of slivers. (The sliver removal post-process of [4] could be easily modified to improve the aspect ratio without additional runtime cost.)

Unfortunately, a proper choice for these two regions is not known in advance, and the problem of distinguishing these regions is fundamentally the same as generating the entire mesh. We observe that the regions that need to be specially treated are precisely those where a traditional meshing algorithm will not terminate, refining forever based on the conflicting goals of quality and conforming.

Ruppert [6] was the first to address the issue of small input angles in a refinement based algorithms. His idea for 2D meshing was to add “small” protective balls around vertices with small input angles thus removing the small input angles from the mesh. In a series of papers, this idea has been extended to 3D by adding a set of balls (collars) around input points and edges that are common to a small input angles [5, 1]. Their constructions are nontrivial. Unfortunately, none of these algorithms have sub-quadratic run times even for inputs with polynomially bounded spread.

As with all refinement algorithms, SVR does not terminate when the input features contain small angles. Even though such algorithms do not terminate, they do converge to a fixed mesh, though possibly infinite.

That is, if we pick any point p in the input domain other than the vertex of a small input angle, the algorithm

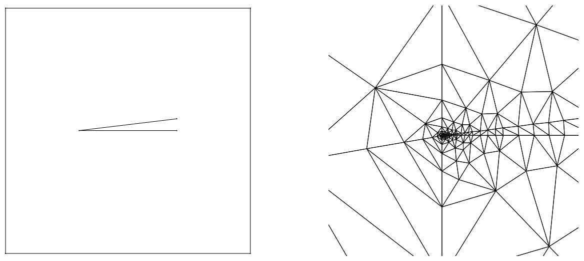


Figure 1: **Left:** An simple input consisting of two lines meeting at a crease. **Right:** An infinite quality mesh for this input.

will eventually include the point in a fixed, finite triangle/simplex or Voronoi cell of the output. Our idea is to simulate the refinement algorithm generating an infinite mesh and use this mesh to specify our final finite mesh. Of coarse we will not let the mesh become of unbounded size.

In keeping with traditional approaches, we too create a *collar region* around the creases, and the mesh refinement within is handled as a special case to ensure termination. We achieve better runtime by creating this special region dynamically. Our on the fly calculations allow the region to be computed in a single pass with the rest of the algorithm, avoiding any expensive pre-processes.

When the mesh elements in the neighborhood of a crease are approximately small enough, a proper size is computed and the collar region is augmented. Previous approaches to collar regions either required expensive $O(n^2)$ pre-computations or assumed a lower-bound on size and create constant-sized collar regions. We use careful predicates and show that the collar region is always augmented before refinement progresses too far, and that the collar-sizing calculations can always be performed efficiently. In three dimensions, the creases may be corners or whole segments. The collar region will consist of a union of balls centered on these creases that will eventually cover all of the creases.

3 Collar Surface Construction

The infinite mesh will be generated fine enough so that it has a good approximation to the boundary of the collar region. It is necessary to have some approximation to this boundary surface so that the two meshes (exterior and interior the collar region) can be properly stitched together.

To extract from the infinite mesh an approximation to this surface, we use the notion of the *restricted Delaunay triangulation* [2]. Most prior work has proposed meshing so that all the vertices in the restricted Delaunay of the surface are explicitly on the surface. Forcing all these points

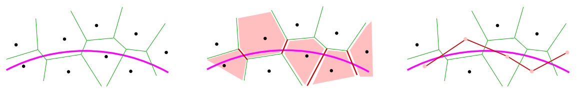


Figure 2: *Left:* A Voronoi diagram with a collar surface cutting through. *Center:* The restricted Voronoi cells. *Right:* The restricted Delaunay triangulation approximates the surface.

to be on the boundary of the ball is not always necessary and may generate more vertices than needed. When the collar region is dynamically augmented and the surface is updated, we allow the existing mesh vertices to appear in the restricted Delaunay, and only require that any new vertices be placed on the surface. This relaxed approach makes it easier to analyze the runtime, and we believe will generate fewer vertices in practice.

A proper approximation to the surface of the collar region will need the appropriate topology to ensure a proper mesh when the interior and exterior are stitched together. Since the collar region is a union of balls, the interior will be meshed with stars of tetrahedra emanating from the crease to the surface of the collar region. To ensure that these tetrahedra have no large dihedral angles, the facets of the restricted Delaunay must approximate the surface normals.

To meet these two goals of topology and normal approximation, additional low-priority refinements are added to the meshing process whenever there are violations. Violations can be detected efficiently, and ε-sampling arguments bound the number of additional vertices to be inserted.

References

[1] S.-W. Cheng and S.-H. Poon. Three-dimensional delaunay mesh generation. *Discrete Comput. Geom*, 36:419–456, 2006.

[2] T. K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge University Press, Oct. 2006.

[3] B. Hudson, G. Miller, and T. Phillips. Sparse Voronoi Refinement. In *Proceedings of the 15th International Meshing Roundtable*, pages 339–356, Birmingham, Alabama, 2006. Long version available as Carnegie Mellon University Technical Report CMU-CS-06-132.

[4] X.-Y. Li and S.-H. Teng. Generating well-shaped Delaunay meshed in 3D. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 28–37. ACM Press, 2001.

[5] S. E. Pav and N. J. Walkington. Robust Three Dimensional Delaunay Refinement. In *Thirteenth International Meshing Roundtable*, pages 145–156, Williamsburg, Virginia, Sept. 2004. Sandia National Laboratories.

[6] J. Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995. Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) (Austin, TX, 1993).

Greedy Routing with Guaranteed Delivery Using Ricci Flows

Rik Sarkar* Wei Zeng* Xiaotian Yin* Jie Gao* Feng Luo† Xianfeng Gu*

* Department of Computer Science, Stony Brook University. {rik, zengwei, xyin, gu, jgao}@cs.sunysb.edu
† Mathematics Department, Rutgers University. fluo@math.rutgers.edu

***Abstract*—Greedy forwarding with geographical locations in a wireless sensor network may fail at a local minimum. In this paper we propose to use *conformal mapping* to compute a new embedding of the sensor nodes in the plane such that greedy forwarding with the virtual coordinates guarantees delivery. In particular, we extract a planar triangulation of the sensor network with non-triangular faces as holes, by either using the nodes' location or using a landmark-based scheme without node location. The conformal map is computed with *Ricci flow* such that all the non-triangular faces are mapped to perfect circles. Thus greedy forwarding will never get stuck at an intermediate node. The computation of the conformal map and the virtual coordinates is performed at a preprocessing phase and can be implemented by local gossip-style computation. The method applies to both unit disk graph models and quasi-unit disk graph models. Simulation results are presented for these scenarios.**

I. Introduction

This paper studies greedy routing in sensor networks using a form of virtual coordinates. Greedy routing has received a lot of attention since it was proposed for routing in ad hoc wireless networks [1], [6], due to its simplicity and efficiency. A node forwards the packet to its neighbor whose distance to the destination is the smallest. Thus routing decision is made with only local knowledge. On a dense network without holes greedy routing typically works very well and gives close to optimal routing paths.

A well-known problem with geographical forwarding is that packets may get stuck at nodes with no neighbor closer to the destination. There have been many ways to handle this problem, the most well-known one is face routing [1], [6]. In this paper we take the approach of constructing a map of the network and finding *virtual coordinates* for the sensor nodes such that simple greedy routing with virtual coordinates always succeeds.

Our Approach: Conformal Maps The map we will use is a *conformal map*. A *conformal map* between two surfaces preserves angles. For any two arbitrary curves γ_1, γ_2 on the surface S, a conformal map ϕ maps them to $\phi(\gamma_1)$, $\phi(\gamma_2)$ with the same intersection angle as that of γ_1, γ_2. According to conformal geometry theory, a genus zero surface with multiple boundaries (a topological multi-holed annulus) can be conformally mapped to the unit disk with circular holes, as shown in Figure 1. Recent advances in differential geometry, in particular, on *Ricci flow* [2], [3] lead to computationally efficient algorithms [5] to construct such kind of conformal mapping.

Our Contributions: We investigate in this paper algorithms for computing a conformal map of a sensor field and the application in enabling greedy routing. We first extract from the communication network a planar graph H such that all non-triangular faces map to network holes that will be later mapped to *circular holes* in the embedded domain D. We present a method to construct a suitable triangulation from any unit disk graph and certain quasi-unit disk graphs. The conformal map is computed on this triangulation. There are two major advantages of using Ricci flow method for computing the virtual coordinates: *distributed nature* and *conformality*.

On a last note, our method considers triangulation of a domain with possible holes and converges to an embedding of the network with holes mapped to circles as long as such embedding exists [4]. The existence of embedding can be verified by the combinatorics of the network.

II. Theory

Riemannian Metric and Curvature Riemannian metric determines the Gaussian curvature K and the geodesic curvature k. Suppose $u : S \to \mathbb{R}$ is a scalar function defined on surface S, then it can be verified that $e^{2u}\mathbf{g}$ is another Riemannian metric on S, denoted by $\bar{\mathbf{g}}$. $\bar{\mathbf{g}}$ is *conformal* to $\mathbf{g}$ and now e^{2u} is called the conformal factor. The Gaussian curvatures are related by

$$\bar{K} = e^{-2u}(-\Delta u + K), \tag{1}$$

where Δ is the Laplace-Beltrami operator under the original metric $\mathbf{g}$. According to Gauss-Bonnet theorem, the total curvature doesn't change.

Surface Ricci Flow Suppose S is a smooth surface with Riemannian metric $\mathbf{g}$. The Ricci flow is the process to deform the metric $\mathbf{g}(t)$ according to its induced Gaussian curvature $K(t)$, where t is the time parameter

$$\frac{dg_{ij}(t)}{dt} = -2K(t)g_{ij}(t). \tag{2}$$

Discrete Ricci Flow can be formulated in the variational setting, namely, it is a negative gradient flow of some special energy form.

$$f(\mathbf{u}) = \int_{\mathbf{u_0}}^{\mathbf{u}} \sum_{i=1}^{n} (\bar{K}_i - K_i) du_i, \tag{3}$$

where $\mathbf{u_0}$ is an arbitrary initial metric. The integration above is well defined, and called the *Ricci energy*. The discrete Ricci flow is the negative gradient flow of the discrete Ricci energy. The discrete metric which induces $\bar{k}$ is the minimizer of the energy. The discrete Ricci energy is strictly convex (namely, its Hessian is positive definite). The global minimum uniquely exists, corresponding to the metric $\bar{\mathbf{u}}$, which induces $\bar{\mathbf{k}}$. The discrete Ricci flow converges to this global minimum [2].

III. Algorithms

1. Obtain a triangulation We describe methods for obtaining a triangulation in cases where the nodes are aware of their locations as well as in cases where locations are not available: (1) *location-based triangulation* , and (2) *landmark-based triangulation*. The methods apply to quasi-unit disk graphs with suitable parameter.

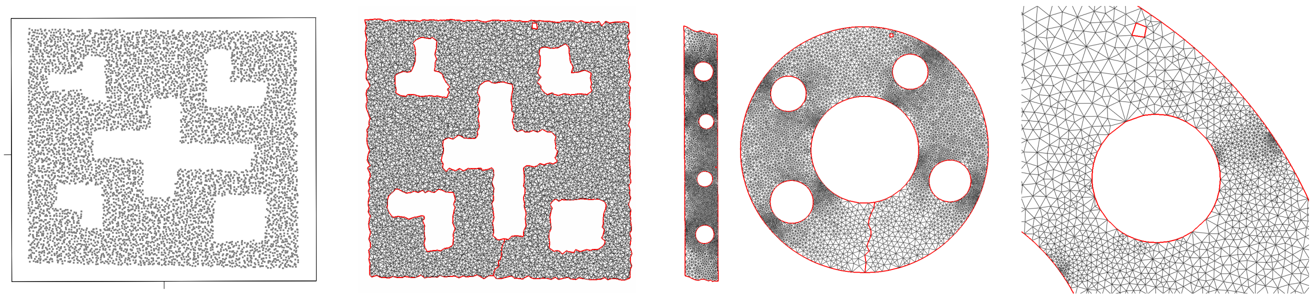

(a) Nodes (b) Triangulation (c) Virtual Coordinates (d) Zoomed in

Fig. 1. **Algorithm pipeline**. Given a network in (a), a triangulation is constructed in (b). The boundaries are traced as γ_k's in (b). A cut between γ_1 and γ_2 is located as η. The triangular mesh is sliced open along η. By using Ricci flow the sliced mesh is flattened to a parallelogram in (c), then mapped to the unit disk with circular holes in (c). Local region is zoomed in as shown in (d), which shows all the triangles are with acute angles.

2. Compute conformal map Figure 1 shows the algorithm pipeline for computing the conformal mapping. The accuracy of the computation is mainly controlled by the curvature error bound ϵ in the Ricci flow algorithm [5]. In theory, and in practice, the method also works for very fragmented networks with many holes.

3. Routing Figure 2 shows the effect of greedy routing on the virtual coordinates. The path under the virtual coordinates, however, easily gets past the hole to the other side.

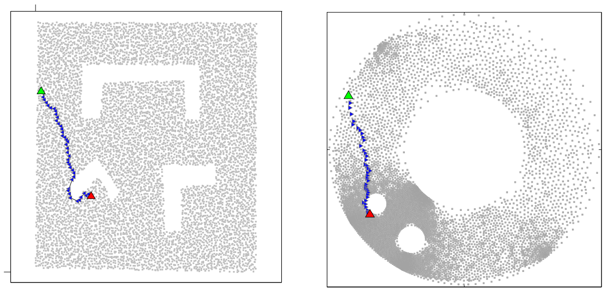

(a) (b)

Fig. 2. Routing. Network of about 8700 nodes, average degree of about 20 in quasi-UDG setting, and nodes in a perturbed grid distribution. (a) Routing based on virtual coordinates successfully goes around the hole, while Geedy routing gets stuck at a hole boundary. (b) The routing path in the virtual coordinate space.

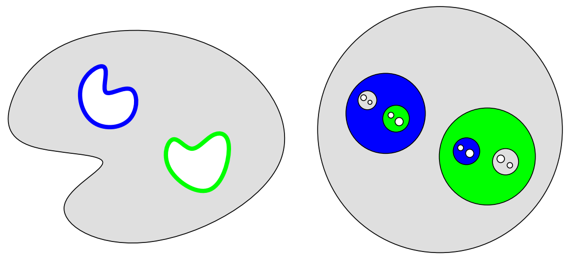


Fig. 3. Circle reflection for a domain with 2 holes.

Application to Information Brokerage by Double Ruling. Searching for a piece of data is a common problem in sensor networks. Suppose sensor A detects an event. Sensor B may need to know the details of the event. Since B does not know which sensor holds the data, it is not easy to find A efficiently. *Double ruling* [8] provides a clean flexible solution to this problem in simple networks. In this method, A will store the data (or a pointer) along a curve through A, and B will search along a different curve through B. These curves are designed such that an intersection between the two curves is guaranteed. This makes it easy and efficient to answer queries of type described above.

The embedding we compute with circular boundaries can be used to adapt double ruling to multiply connected domains. The circular boundaries make it easier to *fill up* the holes. The following is the essential idea. We reflect the entire network about the boundary of a hole h, using the usual definition of reflection in a circle (see Fig. 3). This creates images of other boundaries and corresponding holes inside h, but these are necessarily smaller than h. Now we repeat the operation on these interior holes. With each level of reflection, the holes become smaller exponentially fast. We can carry out the same process at each boundary to obtain a network with arbitrarily small holes. Observe that the process can be executed on demand. When a double ruling curve hits a boundary, the corresponding reflection can be computed and attached to the message. We are investigating the details of this method.

IV. Experimental Results

We conducted extensive experimentation on UDG or quasi-UDG based network of Figure 2, and on networks of similar topology but different number of nodes. From a routing performance point of view, the following are important observations from the experiments and simulations: (1) **100% routing guarantee**, and (2) **small routing stretch**. Table I shows that our algorithm incurs a larger stretch (ratio of routing path length to shortest path length) than NoGeo, but guarantees delivery. More experimental results are illustrated in [7].

TABLE I
Stretch and Delivery Comparison

Method	Delivery rate	Avg Stretch	Max Stretch
Our Method	100%	1.59	3.21
NoGeo	83.66%	1.17	1.54

References

[1] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[2] B. Chow and F. Luo. Combinatorial ricci flows on surfaces. *Journal Differential Geometry*, 63(1):97–129, 2003.

[3] R. S. Hamilton. Three manifolds with positive ricci curvature. *Journal of Differential Geometry.*, 17:255–306, 1982.

[4] Z.-X. He and O. Schramm. Fixed points, Koebe uniformization and circle packings. *Ann. of Math. (2)*, 137(2):369–406, 1993.

[5] M. Jin, J. Kim, F. Luo, and X. Gu. Discrete surface ricci flow. *IEEE Transaction on Visualization and computer Graphics*, 14(5):1030–1043, 2008.

[6] B. Karp and H. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 243–254, 2000.

[7] R. Sarkar, X. Yin, J. Gao, F. Luo, and X. D. Gu. Greedy routing with guaranteed delivery using ricci flows. In *IPSN'09*.

[8] R. Sarkar, X. Zhu, and J. Gao. Double rulings for information brokerage in sensor networks. In *MobiCom'06*.

Resilient Routing for Sensor Networks Using Hyperbolic Embedding of Universal Covering Space

Wei Zeng* Rik Sarkar* Feng Luo† Xianfeng Gu* Jie Gao*

* Department of Computer Science, Stony Brook University. {zengwei, rik, gu, jgao}@cs.sunysb.edu
† Mathematics Department, Rutgers University. fluo@math.rutgers.edu

I. Introduction

This paper is motivated by routing algorithm designs that are resilient to dynamics in a sensor network. In a typical large scale sensor network, there are network changes of different scales. At a large scale, communication links in a region can be temporarily disabled by jamming attacks, either imposed by malicious parties [7], or as a result of co-located multiple benign wireless networks interfering with each other. We need resilient routing schemes that can tolerate such sudden changes of link quality and have flexibility to reactively choose one from many possible paths to the destination.

For a source s and a destination t there are many different paths from s to t. Explicitly storing all of them, for all possible source destination pairs, is clearly not feasible in a resource constrained sensor network. Previous work on routing resilience has mainly focused on heuristic algorithms for multi-path routing [3]. Randomization may also be used in the routing metric to introduce some path diversity. However, it is still unclear how to evaluate the resilience of a set of paths obtained this way. In this paper we study the characteristics of the 'space of paths': the classification of paths and design of light-weight routing schemes that can easily navigate in this space of paths, leading to improved resilience to failures at different scales. We focus on large networks with non-uniform sensor distribution. The network can have a complex shape with multiple holes.

Essentially, two paths *homotopy equivalent* if one can continuously deform one to the other. Paths that are pairwise homotopy equivalent are said to have the same homotopy type. The number of homotopy types is infinite, as one can tour around a hole k times, with any integer k. But for most routing scenarios we only care for a small number of homotopy types.

Understanding the homotopy types of the paths from s to t is important for load balancing and resilience. For example, sporadic link dynamics (unless they create a hole) can be possibly avoided by taking a slightly different path with the same homotopy type. But to get around a large jamming attack that destroys a 'bridge' of the network we will have to take a path of different homotopy type.

Now the question is, how to compute the homotopy type of a given path? How to tell two paths are homotopy equivalent or not? How to choose a path that has a different homotopy type from the previous one? How to find the shortest path of a given homotopy type? For all these questions we need a compact way to encode the homotopy type of all the paths, and a distributed, local algorithm to find a path of a given homotopy type.

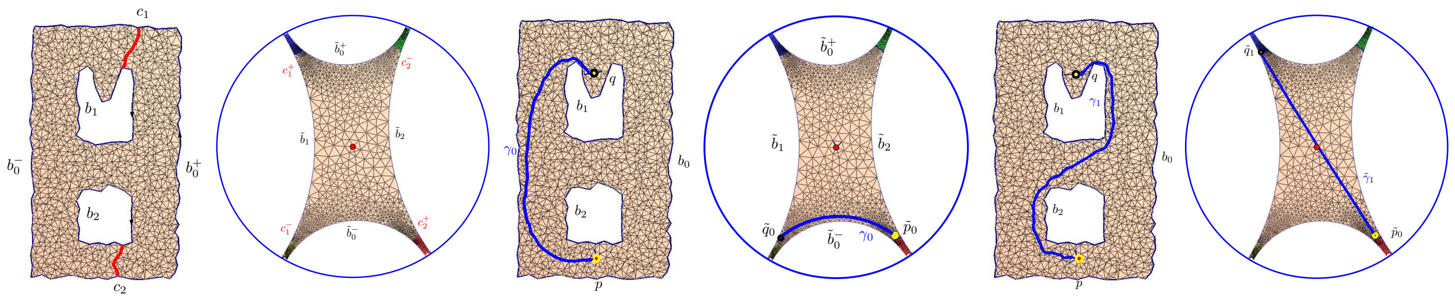
II. Our Method

Our solution is to use a particular embedding of the given network in *hyperbolic space*. We first compute a triangulation of the network from the connectivity graph by a distributed and local algorithm developed in [6]. Then we cut the holes open along paths connecting each hole to the outer boundary. This results in a triangulation T that is simply connected. Using a *Ricci flow* algorithm we conformally modify the metric to have uniform negative Gaussian curvature everywhere in the interior and zero geodesic curvature at boundary points. Discrete Ricci flow is a powerful technique that can be executed distributedly in the network [6]. Now T can be embedded isometrically into a convex region S in hyperbolic space. Each node is given a hyperbolic coordinate. This way, greedy routing with the hyperbolic metric, i.e., send the message to the neighbor closer to the destination measured by hyperbolic distance, has *guaranteed delivery*.

In fact, we can get the embedding of the triangulation T to infinitely many convex patches, each of which is a copy of the complete network. The patches are congruent (isometric) to each-other and they tile an unbounded portion of the hyperbolic plane. Since the patches are congruent, two patches can be transformed to one another by a suitable isometry-preserving transformation in the hyperbolic plane. Different patches are glued naturally along their shared boundaries. This is called the *universal covering space* of the topology of the network.

We fix the source $\hat{s}_1$ in one patch $\hat{T}_1$, and take the image of the destination $\hat{t}_i$ in patch $\hat{T}_i$. All the paths that connect s to t with the same homotopy type will map to the paths that all connect $\hat{s}_1$ to the same image $\hat{t}_i$. Thus, if we would like to get a path of different homotopy type from the previous one, we can simply use greedy routing to find one between $\hat{s}_1$ and a different image $\hat{t}_j$. The set of transformations that map the different patches to each-other form a group called the Fuchsian group. The generators of this group are all the information necessary for generating paths of different homotopy types. There are $2h$ generators for a network with h holes, and these generators are computed easily and locally at the cuts once the patch has been embedded.

The hyperbolic embedding is convex for the entire covering space. That is, all the hole boundaries are on the outer boundary of the embedding, and the shortest path of a particular homotopy type between any two points will *not* pass through a boundary unless either the source or the destination is on the boundary. This is different from many greedy routing schemes



(a) π-group (b) 1-ringed UCS (c) shortest path 1 (d) geodesic of (c) (e) shortest path 2 (f) geodesic of (e)

Fig. 1. Computing the shortest paths using the hyperbolic embedding of a 3-connected domain with 1286 nodes as shown above. (a) Network with cuts. (b) The basic embedded patch with its neighboring patches (c). The base path γ_0 from p to q within the basic patch. (d) The geodesic homotopic to γ_0 is shown. (e) Another path γ_1 from p to q is in (e). The homotopy type of γ_1 ($[\gamma_1\gamma_0^{-1}]$) is given by generator a_1^{-1}.

that route 'around holes' by following the hole boundaries [1], [2], [4], [6], with which boundary nodes necessarily carry more traffic. We evaluated the load distribution of our greedy routing algorithm and compared with other greedy routing schemes.

The universal covering space can be used to find loops of different homotopy types when $t = s$. For any point s, we can find its images in different patches $\hat{s}_1$, $\hat{s}_i$, $i \neq 1$. The path connecting $\hat{s}_1$, $\hat{s}_i$ is a loop in the original triangulation. The paths connecting $\hat{s}_1$ with different other images $\hat{s}_i$ have different homotopy types (i.e., surrounding different set of holes). This can be useful for the applications when we want to find a loop surrounding a target hole or multiple target holes. Or, given any target hole, test whether a group of sensors successfully surround it (mathematically speaking, cycle contractibility). For example, if we want to count the number of people entering/leaving a building, we only need to activate a loop of sensors surrounding the building and aggregate their detections. As there can be many different loops, by choosing different s and different paths connecting two images of s, we can have different loops taking turns to accomplish the sensing task.

III. Simulations

We carried out simulations on a large network. In the routing process the destination was chosen from a restricted set of patches. We considered the image of the destination in the same patch as the source, as well as those in the neighboring patches, and then selected the image that was nearest to the source in the hyperbolic metric. This does not necessarily select the a path homotopic to the shortest path, neither does it consider all possible homotopy types. However, as the results below demonstrate, the method works very well in practice. It produces paths almost as small at the shortest path and distributes load more evenly in the network.

We tested the properties of the routing method described in the previous section. The following are the major conclusions in that respect.

Load balancing and stretch. We ran routing queries on $10,000$ randomly selected source-destination pairs, and compared load balancing properties with Kleinberg's method [5] of embedding a spanning tree of the network in hyperbolic space and shortest path routing. In Table I, *load* represents the total number of messages a node has to handle.

TABLE I
Load comparison

Method	Average Load	Max Load
Covering space embedding (ours)	23.37	368
Spanning tree embedding [5]	32.92	1918
Shortest path routing	19.44	538

Other greedy routing methods such as [6] tend to hug the boundary and thus produce uneven load similar to shortest path routes.

We measured the stretch on the paths, and found that the stretch was remarkably small, only about 1.15, whereas the stretch in the spanning tree embedding method is much higher - about 1.78, while the method in [6] has a stretch of 1.59.

More simulation results and the details of the theory and algorithms are available online [8].

References

[1] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[2] Q. Fang, J. Gao, and L. Guibas. Locating and bypassing routing holes in sensor networks. In *Mobile Networks and Applications*, volume 11, pages 187–200, 2006.

[3] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(4):11–25, 2001.

[4] B. Karp and H. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 243–254, 2000.

[5] R. Kleinberg. Geographic routing using hyperbolic space. In *Proceedings of the 26th Conference of the IEEE Communications Society (INFOCOM'07)*, pages 1902–1909, 2007.

[6] R. Sarkar, X. Yin, J. Gao, F. Luo, and X. D. Gu. Greedy routing with guaranteed delivery using ricci flows. In *Proc. of the 8th International Symposium on Information Processing in Sensor Networks (IPSN'09)*, April 2009.

[7] W. Xu, K. Ma, W. Trappe, and Y. Zhang. Jamming sensor networks: attack and defense strategies. *Network, IEEE*, 20(3):41–47, May-June 2006.

[8] W. Zeng, R. Sarkar, F. Luo, X. Gu, and J. Gao. Resilient routing for sensor networks using hyperbolic embedding of universal covering space. http://www.cs.sunysb.edu/˜rik/papers/hyperbolic.pdf, 2009.

Conformal Mapping in Linear time

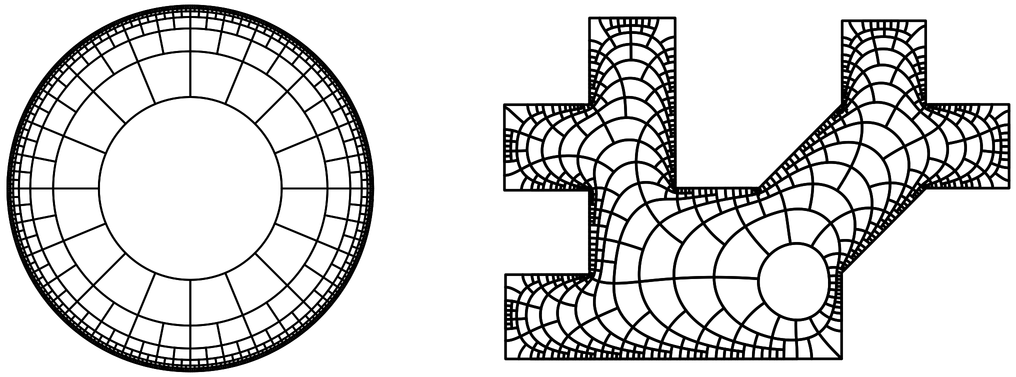
Christopher Bishop
Stony Brook

Abstract

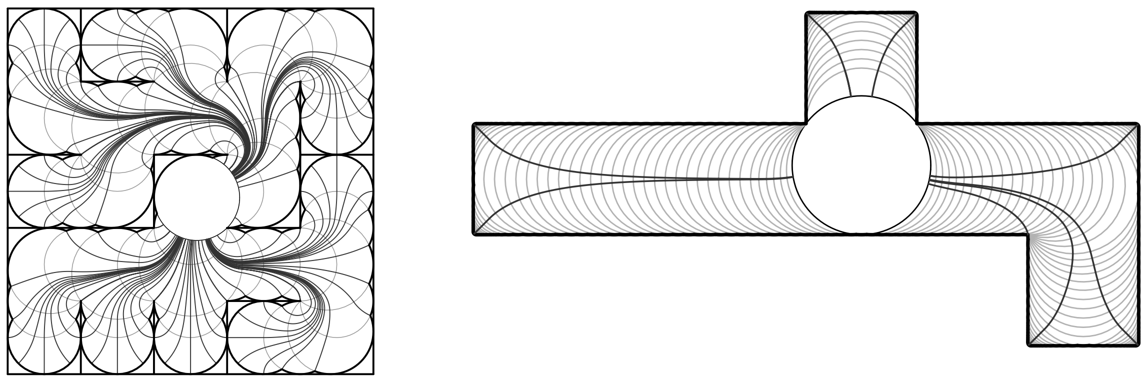
A conformal map between planar domains is one which preserves angles. This is equivalent to saying the tangent maps are Euclidean similarities. The Riemann mapping theorem says that any simply connected plane domain (except the plane itself) is the conformal image of a disk, but there is no simple formula for this map in general. The Schwarz-Christoffel formula gives a formula when the image domain is a polygon, but contains unknown parameters. There are various numerical methods to approximate these parameters, but many are slow and few are proven to always converge. In this talk I will describe an algorithm that finds the conformal map onto an n-gon in time $O(n)$ (with constant depending on the desired accuracy, but not on the geometry of the domain).

The connection to computational geometry comes in the proof. It turns out that given a polygon, there is a relatively simple way to define a map of the interior to a disk which is "close to" a conformal map in a precise sense. This map was first discovered in the context of hyperbolic 3-manifolds, but can more easily be described in terms of the medial axis of the polygon (the centers of all interior disks hitting the boundary in at least two places). Thus there is a formula for a "not quite, but almost" conformal map to the disk. Approximating the real Riemann map starts from this initial guess and uses an iteration that improves the degree of conformality at each step. The linear time bound comes from the known linear time construction of the medial axis.

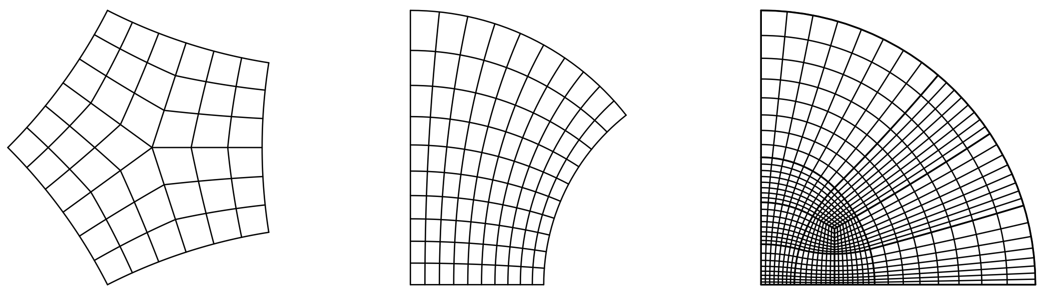
I will review the relevant definitions and give a bit of the background about this connection between conformal maps, hyperbolic geometry and computational geometry. As time permits, I will give a more precise statement of the theorem, a sketch of the proof and an application to meshing (every simple polygon has a quadrilateral mesh with all new angles in $[60^\circ, 120^\circ]$ and which can be computed in linear time).



A decomposition of the disk into Whitney squares and the images of these squares under a conformal map to a polygon. Note that angles are preserved but some boxes undergo extreme expansion.



The curved lines illustrate a flow from the boundary of a domain to an interior circle. This flow has a simple geometric definition in terms of the medial axis, but was inspired by a result of Dennis Sullivan on hyperbolic 3-manifolds. On the left we use a sparse collection of disks, on the right, a denser collection. The map defined by the flow approximates the conformal map in a precise and uniform sense.



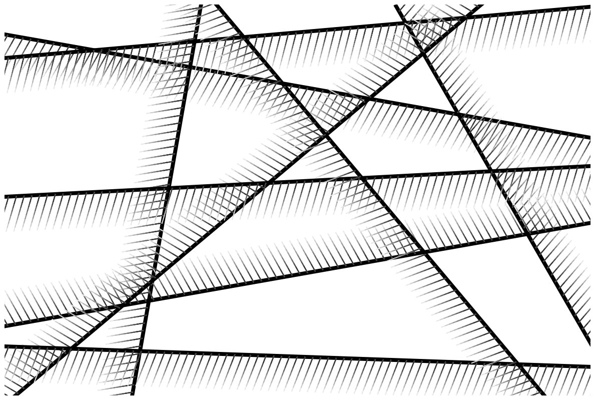
Meshes of polygons in hyperbolic geometry. These are the building blocks used to mesh any polygon with all new angles between 60° and 120°.

Computing the largest bounded intersection of halfspaces

Peter Brass
City College of New York, CUNY

FWCG 2009, November 13–14, 2009

Given n halfplanes in the plane, or half-spaces in space, we wish to compute the largest bounded intersection of k of them. This problem looks like some dual of the largest k-gon spanned by n points [1–3], but this is misleading; the existence of unbounded intersections causes a fundamental difference. Every family of n halfplanes contains a subfamily of $\frac{1}{2}n$ halfplanes whose intersection is unbounded; we look for a subfamily of k halfplanes whose intersection is as large as possible while being still bounded. Such a family need not exist; if, e.g., the halfplanes are all exterior tangents to a circle, then any intersection is either unbounded or empty.

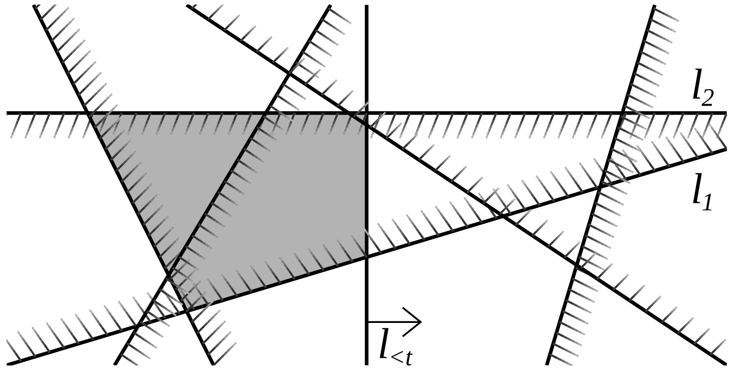


An Arrangement of Halfplanes

This is not an LP-type problem. Neither does it have the exchange property: there are two triples of halfplanes that have the same intersection area, but any mixed triple selected from these six halfplanes has an unbounded intersection.

We present an $O(n^3k)$-algorithm for the planar problem, which works for largest area, largest perimeter, or indeed any size measure with a suitable piecewise algebraic parametrization. The higher-dimensional problem is still open.

The algorithm is based on a line-sweep. Let $h_1, \ldots, h_n$ be the n halfspaces, with boundary lines $l_1, \ldots, l_n$, and $l_{\leq t}$ be the sweepline, with $h_{\leq t}$ be the halfspace to the left of it. We maintain for each pair $\mu, \nu \in \{1, \ldots, n\}$ and each $\kappa \in \{1, \ldots, k\}$ the function max_area(μ, ν, κ, t), which is the largest area of a polygon in $h_{\leq t}$ which is the intersection of $h_{\leq t}, h_\mu, h_\nu$, and $\kappa - 2$ further h_ι. This function is a quadratic function of t, so we need to maintain only its coefficients; these can change whenever the sweepline $l_{\leq t}$ passes an intersection point of some line l_ι with l_μ or l_ν. There are $\binom{n}{2}$ intersection points, and at each of them, we need to update $O(nk)$ coefficients, which gives the claimed bound.



The Region selected for l_1, l_2, and $\kappa = 3$

This is joint work with Hyeon-Suk Na of Soongsil University, Korea [4].

References

[1] James E. Boyce, David P. Dobkin, Robert L. Drysdale III., Leo J. Guibas: Finding Extremal Polygons, *SIAM Journal on Computing* **14** (1985) 134–147.

[2] Alok Aggarwal, Baruch Schieber, Takeshi Tokuyama: Finding a minimum-weight k-link path in graphs with Monge property and applications, *Discrete & Computational Geometry* **12** (1994) 263–280.

[3] Baruch Schieber: Finding a minimum-weight k-link path in graphs with the concave Monge property, *Journal of Algorithms* **29** (1998) 204–222.

[4] Peter Brass, Hyeon-Suk Na: Computing the largest bounded intersection of k out of n halfplanes, to appear in *Information Processing Letters*

On Non-crossing (Projected) Spanning Trees of 3D Point Sets

Joseph S. B. Mitchell[*] Eli Packer[†]

1 Introduction

We study the problem of optimizing the spanning tree of a set of points in $\mathbb{R}^3$, whose projection onto a given plane contains no crossing edges. The *non-crossing minimum spanning tree* (NCMST) problem is defined as follows. Given a set of points P in $\mathbb{R}^3$ and a plane F, find a spanning tree of P whose projection onto F contains no crossing edges and its length is minimum among all such spanning trees. The NCMST is motivated by areas such as shape modeling and surface reconstruction. We prove that the problem is NP-hard and show that greedy algorithms (analogous to Prim and Kruskal) may perform arbitrarily badly. Nevertheless, we report that experimentally these algorithms perform well in practice.

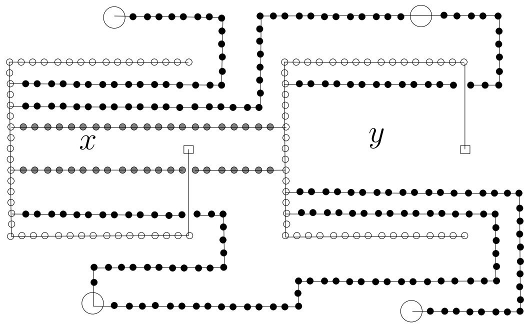


Figure 1: Construction for $\sigma = x(x+y)(\bar{x}+\bar{y})\bar{y}$. Negated literals are directed to the clauses below. Here, σ is satisfied by assigning *true* and *false* to x and y, respectively.

2 Proof of Hardness

In order to prove hardness of NCMST, we use a reduction from *positive-negative planar 3-SAT* (PN-planar 3-SAT for short). It is a special restriction of planar 3-SAT that remains NP-complete [1] and whose restriction is defined as follows. Let σ be a formula in CNF form and let $G_\sigma = (V, E)$ be the corresponding graph of σ. $V = X \cup C$ includes the variables ,X, and the clauses, C, of σ and $E = E_1 \cup E_2$ where $E_1 = \{x_i, x_{i+1} | 1 \leq i < n\} \bigcup \{(x_n, x_1)\}$ (n is the number of variables in σ) and $E_2 = \{(c_i, x_j) |$ clause c_i contains variable $x_j\}$. In this embedding, the edges of E_1 partition the plane into two faces, F_1 and F_2. An instance of PN-planar 3-SAT satisfies the following restriction. If two clauses c_1 and c_2 contain opposite occurrences of some variable x, then one lies inside F_1 and the other inside F_2.

Theorem 2.1 *NCMST is NP-hard.*

Proof: We give a polynomial reduction from PN-planar 3-SAT. Refer to Figure 1 for an example (for simplicity having just 1 or 2 literals per clause), with different points drawn with different shapes and textures for clarity. The figure illustrates the projections of the points onto the xy-plane ($z = 0$). The orientation of the xy-plane is the standard (the x axis spans from left to right and the y axis from bottom to top). Each variable v_i of σ is assigned a gadget, which consists of points along three edges of a rectangle $R(v_i)$ (left, bottom and top, denoted by $l(v_i)$, $b(v_i)$ and $t(v_i)$ respectively) and another special point denoted by $s(v_i)$. In the figure, which corresponds to a formula of two variables, x and y, the white small discs are the points of $R(x)$ and $R(y)$ and the two squares represent $s(x)$ and $s(y)$. For any variable v_i, $s(v_i)$ is located on the middle of $r(v_i)$, the imaginary right edge of $R(v_i)$. All of these gadgets are identical, aligned horizontally and long enough to contain all of the wires that we describe next. The n variables gadgets are connected by $n-1$ pairs of wires. One of the wires in any pair passes slightly above (in y-coordinate in the projection) $s(v_i)$ and the other passes slightly below (in y). In the example, those are the two shaded wires. These wires also partition the xy-plane into two parts, which are analogous to the partition in the PN-planar 3-SAT problem. The points represented by big white discs are the gadgets for the clauses. Wires connecting variable gadgets and clause gadgets correspond to inclusions of literals in clauses of σ. These wires are connected in both sides to $\{l(v_i) | 1 \leq i \leq n\}$ from inside the variable gadgets and to the clause points. Since σ satisfies embedding of PN-planar 3-SAT, opposite literals will go to different directions (up and down) in our reduction. We assign the z values of the points as follows. All of the points in our reduction except from the points $\bigcup_{1 \leq i \leq n} \{b(v_i) \cup t(v_i) \cup s(v_i)\}$ have z value M where M is much bigger than the diameter length any variable gadget. $b(v_i)$ and $t(v_i)$ slide down linearly from left, where the z value is M, all the way to the right, where the z value is 0 (both have the same structure). The z value of $s(v_i)$ is 0 as well. It is known that drawing planar graphs with grid points requires a polynomial number of grid points. Thus, our reduction generates a set of $\mathcal{P}(\sigma)$ points where $\mathcal{P}$ is some polynomial function of σ. In our reduction, the distance between any two adjacent points is a constant denoted by ε.[1] We make sure that any two wires connecting two variable gadgets or

[*]Dept. Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY 11794-3600. jsbm@ams.sunysb.edu.
[†]IBM Research Center, Haifa, Israel. packer_eli@yahoo.com.

[1]The adjacency relation here is implied from our description as points that are consecutive in wire chains, variable gadgets, etc.

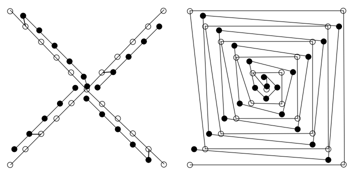


Figure 2: **Left**: The spanning tree obtained with the constrained Prim algorithm. Thick edges connect white to black points. **Right**: Connecting the points using two spiral sequences of links requires only one edge that connects white and black points. (In this case it connects the center points, so is invisible here.)

variable gadgets to clause gadgets are sufficiently separated, meaning that their separation is much bigger than ε. Let the distance between $b(v_i)$ and $s(v_i)$ be denoted by δ (this is also the distance between $t(v_i)$ and $s(v_i)$). It is fixed in all variable gadgets. We make sure that $\delta >> \varepsilon$. For some formula σ, let $K = \mathcal{P}(\sigma)$. Note that n points in the reduction are denoted by $S(\sigma) = \{s(v_i) | 1 \leq i \leq n\}$. Under this construction, any spanning tree will have $K - 1$ links.

We next prove that the size of the MST in our reduction (denoted by $L(\sigma)$) is at most $(K - n - 1)\varepsilon + n\delta$ if and only if σ is satisfiable. We first note (based on simple observations about our construction) that the length of any spanning tree is never smaller than $(K - n - 1)\varepsilon + n\delta$. Thus, we exclude such cases. Suppose σ is satisfiable. We first connect all of the points of $S(\sigma)$ to either $b(v_i)$ or $t(v_i)$, depending on the value of v_i in σ. This connection is such that the wires that correspond to true values of v_i will not be cut by this connection. We refer to these connections asd *gates*. Now, the wires that correspond to truth values can be connected to their corresponding clauses without crossing any edge on the projected xy-plane. Since σ is satisfiable, all clauses will be connected in this way. All wires that are cut by any of the gates, can be connected in both sides from the variable gadgets' left walls and from one of the clause gadgets. Regarding the variable connecting wires (the shaded wires in the figure), one of them will be able to connect its two variable gadgets and the other will be connected by both gadgets from left and right. Based on this construction, all points are connected by a spanning tree with a length $L(\sigma) = (K - n - 1)\varepsilon + n\delta$. To show the converse, suppose that an MST has length at most $(K - n - 1)\varepsilon + n\delta$. Based on our construction, it must be the case that all of the points except the points of $S(\sigma)$ are connected, with links of length ε, and the points of $S(V)$ are connected to exactly one point above or below (in the xy projected plane) that belongs to variable gadgets, with links of length δ. This connection determines the truth value of the variables as explained above and these truth values satisfy all of the clauses. $\square$

3 Performance of Greedy Heuristics

It is natural to aply a greedy heuristic to NCMST, analogous to Kruskal or Prim, with the added constraint that the projection of the next edge e does not intersect the projection of the current tree. We refer to the corresponding MST algorithms as either *constrained Prim* or *constrained Kruskal*. We next show that such greedy heuristics can produce arbitrarily bad results. Consider the point set in the left side of Figure 2. It consists of two "stars" (one has white points and the other black points), each with t arms (in the figure each star has four arms). Note that this figure shows the projection of the points onto the xy plane. The z value of the white points is 0 and the z value of the black points is M, a positive huge value such that $M >> K$, where K is the diameter of any of the stars. Consider the constrained Prim algorithm. For the moment assume that the number of arms in each star is 6 or smaller. Without loss of generality, it will first connect all of the white points, and then the black points will be connected. Note that the links connecting the white points will block any arm of the black star from being connected directly to another black arm. Thus, each black arm will be connected from a white point (see Figure 2(a)). It follows that the length of the spanning tree will be roughly tM (recall that $M >> K$). It is easy to infer that the constrained Kruskal algorithm will construct roughly the same output. On the other hand, we can position the white and black points such that they can be connected by two spiral chains, as illustrated in the right side of Figure 2. In this case, one link connecting some white and black points is sufficient. Thus, the length of the corresponding spanning tree is roughly M. This value constitutes an upper bound on the length of the MST. Note that in this construction, the number of arms is limited to 6, since the maximum degree of the center of the stars cannot be greater than 6 in a spanning tree that is constructed by a greedy algorithm like Prim. In order to support more than 6 arms, we can easily place the arms in slightly different levels (each level with different z value) and connect each level through the center points that will have the same xy coordinates. Using this construction, t can be arbitrarily large, and the ratio R between the MST and the spanning trees obtained with the constrained Prim and Kruskal can be arbitrarily bad ($R \geq \frac{tM}{M} = t$).

We conducted experiments to compare the heuristics with the optimal trees (obtained using a brute-force algorithm). We tried several kinds of input: (a) Uniformly-sampled points in a cube; (b) Points from real data scanned models; and (c) Points in 3D artificial structures. In all experiments the results of the greedy heuristics were within 10% of the optimal solution.

Acknowledgements. J. Mitchell is partially supported by NSF (CCF-0528209, CCF-0729019), NASA Ames, and Metron Aviation.

References

[1] D. Lichtenstein. Planar formulae and their uses. In *SIAM J. on Computing 11*, 1981.

Visiting Points with a Bevel-Tip Needle*

Steven Bitner†, Yam K. Cheung†, Atlas F. Cook IV§,1, Ovidiu Daescu†, Anastasia Kurdia†, Carola Wenk§

I. Introduction

Many surgical procedures could benefit from guiding a bevel-tip needle through a sequence of treatment points in a patient. For example, brachytherapy procedures implant radioactive seeds to treat cancer, and biopsy procedures extract tissue samples to test for cancer.

A bevel-tip needle has an asymmetric tip that cuts through tissue along a circular arc. By rotating the needle during its insertion into soft tissue, the needle can be steered along a sequence of circular arcs so that it reaches a treatment point while avoiding bones and vital organs. Although current procedures typically create a new puncture for each treatment point, our work permits multiple treatment points to be visited with a single puncture. Since each reorientation of the needle inherently complicates the path for the physician, we minimize the number of circular arcs that are required for the needle to visit a sequence of treatment points.

Needle paths have similarities to touring problems [5], curvature-constrained shortest path problems [3], and link distance problems [2], [4]. Alterovitz et al. [1] use a Markov model to heuristically steer a needle through a plane with polygonal obstacles to a *single* treatment point. Our main result is an exact algorithm to steer a needle through a *sequence* of treatment points in a plane without obstacles.

A *needle arc* is a directed circular arc that lies on a *needle circle* with radius r. A *needle vector* is a vector that is tangent to the needle's current position on a needle arc. Figure 1a illustrates that a *needle path* $\pi_N(s,t)$ is a connected sequence of needle arcs that connects two points s and t while satisfying the following properties. First, consecutive needle arcs of $\pi_N(s,t)$ lie on tangent needle circles. Second, consecutive needle arcs always have opposite orientations (e.g., if the current arc travels clockwise, then the next arc must travel counter-clockwise). Third, the *length* $d_N(s,t)$ of a needle path is optimal in that it equals the smallest possible number of needle arcs that can be used to connect s and t.

II. Results

A needle path $\pi_N(p_1,...,p_n)$ *visits* a sequence of points $p_1,...,p_n$ if the path passes through the sequence of points in order. Let $d_N(p_1,...,p_n)$ be the number of needle arcs on $\pi_N(p_1,...,p_n)$. Assume for now that the needle vector is allowed to be arbitrary when the needle visits each point p_i. A sequence of points $p_1,\dots,p_n$ is *moderately-separated* whenever $||p_i - p_{i-1}|| \geq 2r$ for all $2 \leq i \leq n$ and *well-separated* whenever $||p_i - p_{i-1}|| \geq 4r$ for all $2 \leq i \leq n$.

Figure 1b illustrates that optimal substructure can fail for a needle path that visits an arbitrary sequence of points. All optimal paths to p_2 use only one arc but require a total of three arcs to reach p_3. By contrast, a globally optimal path to p_3 is possible with just two arcs if we use a locally suboptimal path with two arcs to reach p_2. To see that optimal substructure holds for moderately-separated points, consider two paths to p_i. Let c_o be a needle circle that intersects p_i and lies on an (optimal) needle path $\pi_N(p_1,...,p_i)$ with length $d_N(p_1,...,p_i) = j$. Let c_s be a needle circle that intersects p_i and lies on a suboptimal path with length at least $j+1$. Figure 1c shows that the shaded set of all points reachable from c_o with one additional needle arc will always contain all points on the suboptimal circle c_s that lie outside the disk d_o that is bounded by c_o. This implies that for any point $p_{i+1} \notin d_o$, a path $\pi_N(p_1,...,p_{i+1})$ containing c_o can always be at least as short as any path $\pi'_N(p_1,...,p_{i+1})$ containing c_s. Since moderately-separated points guarantee that $p_{i+1} \notin d_o$, optimal substructure holds for moderately-separated points.

For each point p_i, we can now define a partition of the plane such that each point $t \in \mathbb{R}^2$ has an associated distance $d_N(p_1,...,p_i,t)$. Such a partition can be described by a sequence of pairwise disjoint layers $\mathcal{L}_i^1,...,\mathcal{L}_i^M$ such that all points $t \in \mathcal{L}_i^j$ have associated distance $d_N(p_1,...,p_i) + j - 1$ (see Figure 1d).

Optimal substructure of moderately-separated points ensures that it is always possible to construct any layer $\mathcal{L}_i^{j\geq 2}$ from $\mathcal{L}_i^1$ by additively *enlarging* the radius of each needle arc on the boundary of $\mathcal{L}_i^1$. We can also use $\mathcal{L}_i^1$ to

*This work has been supported by the National Science Foundation grants NSF CAREER CCF-0643597, CCF-0635013, and by the 2009 University of Texas at San Antonio Presidential Dissertation Fellowship.

†University of Texas at Dallas: {sbitner, ykcheung, daescu, akurdia}@utdallas.edu

§University of Texas at San Antonio: {acook,carola}@cs.utsa.edu

1Corresponding author

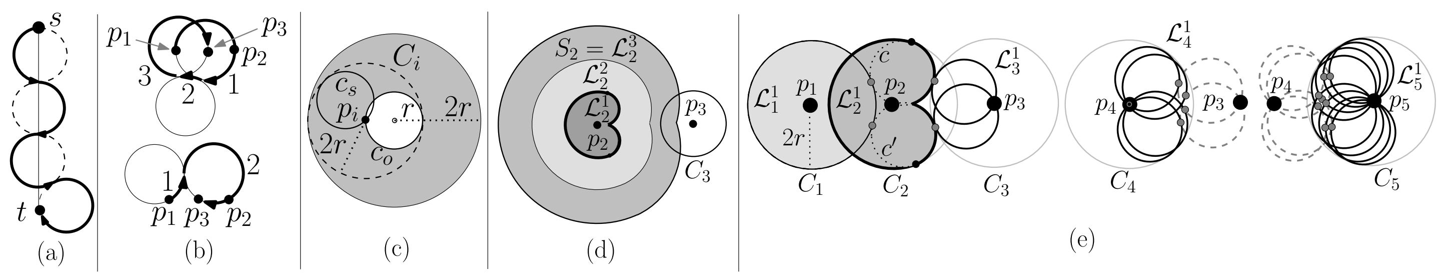


Fig. 1. (a) A needle path $\pi_N(s,t)$ with length four (b,c) Optimal substructure (d) Layers for p_2 (e) Exponential explosion

compute $\mathcal{L}_{i+1}^1$. Let C_{i+1} be the circle with radius $2r$ that is centered at the point p_{i+1}. Let S_i be the first layer in the sequence $\mathcal{L}_i^1, ..., \mathcal{L}_i^M$ that intersects C_{i+1} (see Figure 1d). $\mathcal{L}_{i+1}^1$ is the union of all needle circles that intersect p_{i+1} and are tangent to a circle in S_i.

A sequence of non-well-separated points $p_1, ..., p_n$ can define a layer $\mathcal{L}_n^1$ that is composed of $\Theta(2^n)$ circles. Place $p_1, ..., p_n$ on a line such that $2r < ||p_i - p_{i-1}|| < 4r$ for all $2 \leq i \leq n$. By appropriately positioning p_{2i} to the left of p_{2i-1} and p_{2i+1} to the right of p_{2i} for every $i \geq 2$, $\mathcal{L}_n^1$ is composed of 2^{n-2} circles (see Figure 1e).

By contrast, a sequence of well-separated points $p_1, ..., p_n$ defines a layer $\mathcal{L}_n^1$ that is composed of $O(n)$ critical circles. A *critical circle* is a needle circle that touches p_{i+1} and is tangent to an arc on the *boundary* of S_i. The tremendous reduction in complexity follows because $||p_{i+1} - p_i|| \geq 4r$ ensures that C_{i+1} intersects $\mathcal{L}_i^1$ in at most one point and intersects the boundary of any S_i at most twice for the additively enlarged portion of C_i defining S_i and at most twice for each pair of critical circles defining S_i. Hence, if S_i has x critical circles, then $\mathcal{L}_{i+1}^1$ has at most $x+2$ critical circles. These circles can be computed for $\mathcal{L}_{i+1}^1$ by traversing the boundary of S_i in $O(x)$ time. Thus, the critical circles for all layers $S_1, ..., S_{n-1}$ and $\mathcal{L}_n^1$ can be built in $O(n^2)$ total time and space. By tracing an optimal path through these layers, we obtain the below result.

Theorem 1. *Given a sequence $p_1, ..., p_n$ of well-separated points, $\pi_N(p_1, ..., p_n)$ can be computed in $O(n^2 + K)$ time and space, where K is the complexity of the returned path.*

Note that if we require the needle to have a specific needle vector when it visits each p_i, then each layer $\mathcal{L}_i^1$ consists of the at most two needle circles that are tangent to this vector. For this scenario, our layers technique can return a needle path in only $O(n + K)$ time and space, where K is the complexity of the returned path.

Using a fixed-parameter tractable algorithm, we can also compute a needle path for a sequence of points $p_1, ..., p_{m+n}$, where any m pairs of consecutive points are positioned arbitrarily and the remaining pairs of consecutive points are well-separated. Since optimal substructure can fail for this scenario, we propagate not only locally optimal paths but also locally suboptimal paths that end in a needle circle that visits multiple consecutive points in the sequence.

Theorem 2. *A needle path $\pi_N(p_1, ..., p_{m+n})$ can be computed in $O(2^m n + n^2 + K)$ time and space, where K is the complexity of the returned path.*

Proof: Given a sequence of m arbitrary points, layer $\mathcal{L}_m^1$ can be composed of $\Theta(2^m)$ circles. Each of the well-separated pairs of consecutive points adds only a constant number of additional critical circles to the current layer. Consequently, the worst-case time to compute the critical circles for $\mathcal{L}_1^1, ..., \mathcal{L}_{m+n}^1$ is on the order of $\Sigma_{i=1}^n (2^m + i) \in O(2^m n + n^2)$. Once these critical circles have been computed, a traceback procedure can construct $\pi_N(p_1, ..., p_{m+n})$. ■

III. Conclusion

We developed two algorithms to guide a bevel-tip needle through a sequence of treatment points in the plane. Such paths are potentially useful for biopsy and brachytherapy procedures because they reduce the number of times that a physician is required to insert and withdraw a needle during a medical procedure.

References

[1] R. Alterovitz, M. Branicky, and K. Goldberg. Motion planning under uncertainty for image-guided medical needle steering. *International Journal of Robotics Research*, 27(1361), 2008.

[2] E. M. Arkin, J. S. B. Mitchell, and S. Suri. Optimal link path queries in a simple polygon. *3rd Symposium on Discrete Algorithms (SODA)*, pages 269–279, 1992.

[3] S. Bereg and D. Kirkpatrick. Curvature-bounded traversals of narrow corridors. *21st Symposium on Computational Geometry (SoCG)*, pages 278–287, 2005.

[4] A. F. Cook IV and C. Wenk. Link distance and shortest path problems in the plane. *5th Algorithmic Aspects in Information and Management (AAIM)*, pages 140–151, 2009.

[5] M. Dror, A. Efrat, A. Lubiw, and J.S.B. Mitchell. Touring a sequence of polygons. *35th ACM Symposium on Theory of Computing (STOC)*, pages 473–482, 2003.

The Districting Problem

Esther M. Arkin[*] Irina Kostitsyna[†] Joseph S.B. Mitchell[*] Valentin Polishchuk[‡]
Girishkumar R. Sabhnani[§]

1 Introduction

We consider the following *districting problem*: Given a simple polygon partitioned into simple polygonal *sub-districts* (see Fig. 1) each with a given weight, merge the sub-districts into a minimum number of *districts* so that each district is a simple polygon and the total weight of the sub-districts in any district is at most a given number M. We consider also the "dual" version of the problem, in which the objective is to minimize the maximum district weight, subject to a given bound on the total number of districts.

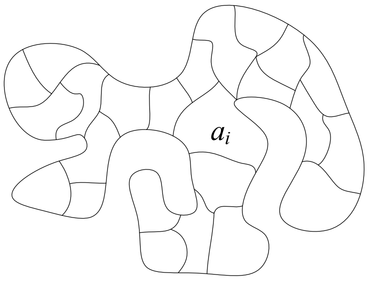


Figure 1: The input: A simple polygon P partitioned into simple polygonal sub-districts.

Related Work The problem has its roots in political districting for voting where the districts may have restrictions on the number of sub-districts, size, total population, etc.; see, for example, the survey by Tasnádi [4]. Our motivation comes from an air traffic management problem in which sub-districts correspond to Fixed Posting Areas (or "sub-sectors") with weights representing a measure of controllers' "workload", and the goal of the merging is to provide a balanced partitioning of the workload among a set of airspace sectors. In related work, Bloem et al. [1] analyze greedy heuristics for merging underutilized airspace sectors to conserve air traffic control resources. The districting problem is also related to problems in bin packing; in our case, however, the shape of the bins is not fixed (only their capacity is fixed), and the items are not allowed to be moved, but only to be grouped.

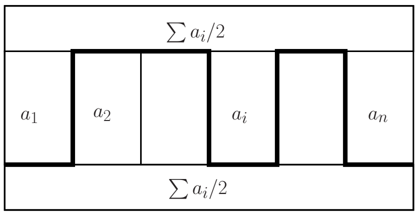


Figure 2: Integers $a_1, a_2, \ldots, a_n$ can be partitioned into two sets with equal sums if and only if the sub-districts can be merged into two districts each of weight at most $M = \sum a_i$.

2 Hardness

A simple reduction from PARTITION (see Fig. 2) shows that the problem is weakly NP-hard; in fact, it is hard to distinguish between the cases in which the optimal solution has 2 vs. 3 districts. Hence, the problem is weakly hard to approximate to within $(3/2 - \varepsilon)$, for any $\varepsilon > 0$.

Moreover, we show that the districting problem is *strongly* NP-hard by a reduction similar to the one in [3] that shows the hardness of array partitioning. The 5/4 hardness of approximation from [3] applies to the dual problem of minimizing the maximum weight of a district. With slight modifications to the construction, we also prove hardness (and 5/4 hardness of approximation for the dual version) for the following special cases of the districting problem: (i) that in which all districts are required to be convex, and (ii) that in which districts are allowed to be multi-connected (simple polygons with holes).

[*]Dept. Applied Mathematics and Statistics, Stony Brook University, {estie,jsbm}@ams.sunysb.edu
[†]Computer Science Dept., Stony Brook University, ikost@cs.sunysb.edu
[‡]Helsinki Institute for Information Technology, polishch@cs.helsinki.fi
[§]Metron Aviation, sabhnani@metronaviation.com

3 Approximation Algorithms

Our approximations are based on properties of the dual graph, G, of the input subdivision into sub-districts.

3.1 The dual graph G is Hamiltonian

Assume that G has a Hamiltonian path. Then, a simple clustering method follows the Hamiltonian path, greedily adding the sub-districts to a district, until the district's weight is about to exceed the bound M, at which point we begin a new district. The total weight of any two consecutive districts thus obtained is more than M. Thus, the average district weight is more than $M/2$, so the number of districts is at most twice the optimal.

Even though the resulting districts are not necessarily simple polygons (Fig. 3), the total number of holes is bounded; indeed, since each hole is itself a district (or a group of districts), the number of holes is at most the number of districts. For each hole we can break the surrounding district into two districts, charging the increase in the number of districts to the hole. Thus, all holes can be removed if we allow the number of districts to (at most) double. This yields a 4-approximation for the districting problem with the constraint of having simply connected districts.

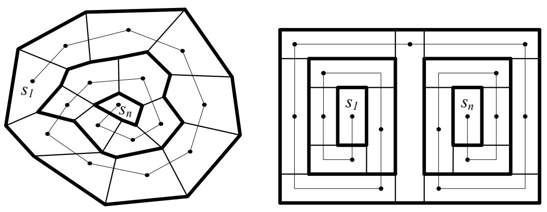


Figure 3: Clustering sub-districts along a Hamiltonain path may result in non-simple districts.

3.2 The dual graph G is not Hamiltonian

If G has no Hamiltonian path, then we turn instead to a low-degree spanning tree, T, of G. (One can compute in polynomial time a spanning tree that has degree at most 1 greater than the degree of a minimum-degree spanning tree [2].) In particular, we obtain a 2Δ-approximation, where Δ is the maximum degree of T. We root the tree T and start from the leaves, clustering sub-districts into districts as we work our way towards the root. Specifically, for each node v we merge v's children in order of increasing subtree weight, until the district weight is about to exceed the threshold M. The average weight of the resulting districts is at least M/Δ; thus, the number of districts is at most Δ times optimal. As described above, we can make all districts simply connected by removing holes, causing the number of districts to at most double.

Acknowledgements

We thank George Hart and other members of the Algorithms Reading Group at Stony Brook for helpful discussions. This research is partially supported by NSF (CCF-0528209, CCF-0729019), NASA Ames, and Metron Aviation. V. Polishchuk is supported in part by Academy of Finland grant 118653 (ALGODAN).

References

[1] M. Bloem and P. Kopardekar. Combining airspace sectors for the efficient use of air traffic control resources. In *AIAA Guidance, Navigation, and Control Conference*, Aug 2008.

[2] M. Fürer and B. Raghavachari. Approximating the minimum degree spanning tree to within one from the optimal degree. In *Proc. 3rd ACM-SIAM Symposium on Discrete Algorithms*, pages 317–324, 1992.

[3] S. Khanna, S. Muthukrishnan, and M. Paterson. On approximating rectangle tiling and packing. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*, pages 384–393, 1998.

[4] A. Tasnádi. The political districting problem: A survey. http://ssrn.com/abstract=1279030, October 2008.

On the minimum color separation circle

Steven Bitner and Yam-Ki Cheung and Ovidiu Daescu*
Department of Computer Science
University of Texas at Dallas
Richardson, TX 75080, USA
stevenbitner@student.utdallas.edu {ykcheung,daescu}@utdallas.edu

October 19, 2009

1 Introduction

Assume we are given two input point sets in $\mathbb{R}^2$, one of size n and the other one of size m. We refer to the two point sets as red set and blue set. We denote the red set by $\mathcal{R}$ and the blue set by $\mathcal{B}$, where $|\mathcal{R}|=n$ and $|\mathcal{B}|=m$. Let $\mathcal{S}$ be the set of circles that enclose all points in $\mathcal{R}$, while minimizing the number of points from $\mathcal{B}$ contained within the same circle. We show how to find the smallest circle $C_{\mathcal{B}}(\mathcal{R})$ in $\mathcal{S}$. We refer to $C_{\mathcal{B}}(\mathcal{R})$ as *the minimum separating circle*. We also give an exact solution for finding the *largest separating circle*, which we call the largest circle in $\mathcal{S}$.

In the case where $\mathcal{R}$ and $\mathcal{B}$ can be completely separated, several results exist in the literature. In [2], Fisk solves the problem of separating $\mathcal{R}$ and $\mathcal{B}$ using the smallest circle in $O(|\mathcal{R}||\mathcal{B}|)$ time and space. This result was improved upon in [3] where the time and space requirements are $O(|\mathcal{R}|+|\mathcal{B}|)$. The problem of separating two sets of polygons in the plane was also solved in linear time [1].

2 Results

We present below an algorithm for finding the smallest separating circle as well as an algorithm for finding the largest separating circle.

Lemma 1. *The center of $C_{\mathcal{B}}(\mathcal{R})$ must lie on an edge of the farthest neighbor Voronoi diagram of $\mathcal{R}$, $FVD(\mathcal{R})$.*

Proof. We know that $C_{\mathcal{B}}(\mathcal{R})$ is either the minimum enclosing circle of $\mathcal{R}$, $C(\mathcal{R})$, or passes through two points in $\mathcal{R}$ and one point in $\mathcal{B}$. Hence, $C_{\mathcal{B}}(\mathcal{R})$ must pass through at least two red points. Given the fact that the locus of the centers of enclosing circles of $\mathcal{R}$ through two red points defines an edge of $FVD(\mathcal{R})$ and the center of an enclosing circle of $\mathcal{R}$ passes through three red points and defines a vertex of $FVD(\mathcal{R})$, the proof follows. □

By applying *Lemma* 1, we can compute $C_{\mathcal{B}}(\mathcal{R})$ by executing a simple sweeping procedure on every edge of $FVD(\mathcal{R})$. For every edge e of $FVD(\mathcal{R})$, let q_i and q_j be the points in $\mathcal{R}$ defining e. That is, for any point $c \in e$, the smallest enclosing circle of $\mathcal{R}$ centered at c passes through q_i and q_j. We index these two points such that the directed line segment $\overline{q_i q_j}$ divides the enclosing circle into two regions and the smaller region A is on the left side of $\overline{q_i q_j}$, while the larger region B is on the right side of $\overline{q_i q_j}$.

We initialize our sweeping algorithm on e by constructing an enclosing circle C of $\mathcal{R}$, which passes through q_i and q_j with minimum area. Let $c \in e$ be the center of C. Notice that c is the nearest point on e to q_i and q_j. c is almost always a vertex of $FVD(\mathcal{R})$ unless $C(\mathcal{R})$ is defined by q_i and q_j. To handle the degenerate case, we can either run the sweeping procedure twice in both directions along e or divide e into two separate edges by treating c as a vertex. We expand C by sweeping c along e while still passing through q_i and q_i. See Figure 1 for an illustration. We define a point $p_e \in e$ as an event point if when c sweeps through p_e, circle C sweeps through a blue point or a red point. We compute the num-

*This research was partially supported by NSF grant CCF-0635013.

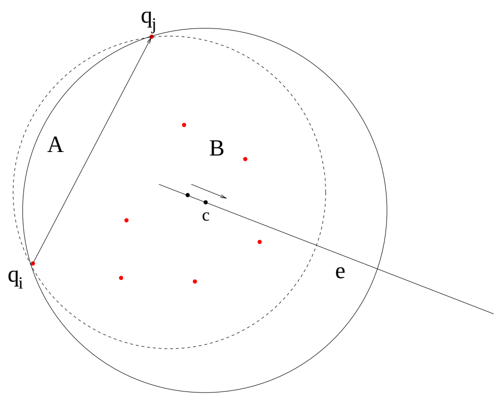


Figure 1: Expand the enclosing circle by sweeping its center c along e.

ber of blue points enclosed by C at each event point. The sweeping procedure terminates when C sweeps through a red point or all event points on e have been processed and returns the local minimum separating circle centered on e. We note that the overall algorithm can output all global minimum separating circles by maintaining a list of circles.

In order to find the largest separating circle, SEP, we first find the list of all minimum separating circles. With k circles in the list, we start with the first circle and find the points from $\mathcal{B}$ that are contained in that circle. We remove those points from $\mathcal{B}$ to obtain $\mathcal{B}_{-1}$. This leaves us with two sets of points that can be purely separated without any overlap. The maximum separating circle between $\mathcal{R}$ and $\mathcal{B}_{-1}$ can be found in $O((n+m)\log(n+m))$ time and this is optimal [3]. We then repeat this process for each of the k smallest circles running the cited algorithm on each of the sets $\mathcal{B}_{-2}$, $\mathcal{B}_{-3}$ ... $\mathcal{B}_{-k}$ and obtain our answer.

Correctness: C remains as an enclosing circle of $\mathcal{R}$ during the process, due to fact that q_i and q_j are the farthest red points to c. By executing the sweeping procedure on each edge, we compute the number of blue points enclosed by every enclosing circle of $\mathcal{R}$ defined by two red points and one blue point as well as $C(\mathcal{R})$.

The proof of the largest circle problem follows from the stated reference for the algorithm and the fact that we analyze all possible sets of blue points that are interior to a minimum circle.

Time: For the minimum separating circle problem, the construction of $FVD(\mathcal{R})$ takes $O(n \log n)$ time. Observe that during the sweeping procedure on edge $e \in FVD(\mathcal{R})$, region A contracts while region B expands. If c sweeps through an event point defined by a blue point q_k on the left (resp. right) side of $\overline{q_i q_j}$, q_k is removed from (resp. added to) C. Hence, the number of blue points enclosed by C at the current event point can be computed in constant time given the number of blue points enclosed by C at the previous event point. The number of event points on each edge is $O(m)$. The sweeping procedure takes $O(m \log m)$ time for each edge. The overall algorithm takes $O(hm \log m + n \log n)$ time, where $h = O(n)$.

For the largest separating circle problem, we must analyze the sets after removing the points contained in the minimum separating circle. Removing these points can be done in $O(m)$ time for each of the k minimum separating circles. Then, for each of the k circles, we use $O((n+m)\log(n+m))$ time for finding the largest separating circle, resulting in an overall running time of $O(k(n+m)\log(n+m))$. This running time is in addition to the running time needed to find the minimum separating circles. So, the overall running time of the largest separating circle algorithm is $O(nm \log m + k(n+m)\log(n+m) + km)$.

Lemma 2. *There are n minimum separating circles in the worst case.*

Proof. Omitted. □

Theorem 1. *Given a red set $\mathcal{R}$ and a blue set $\mathcal{B}$ of points in $\mathbb{R}^2$, one can compute the smallest separating circle, $C_{\mathcal{B}}(\mathcal{R})$, in $O(hm \log m + n\log n)$ time and $O(n+m)$ space where h is the complexity of $FVD(\mathcal{R})$ and compute the largest enclosing circle in $O(nm \log m + k(n+m)\log(n+m))$ time and $O(n+m)$ space, where k is the number of smallest separating circles.*

References

[1] J.-D. Boissonnat, J. Czyzowicz, O. Devillers, and M. Yvinec. Circular separability of polygons. *Algorithmica*, 30(1):67–82, 2001.

[2] S. Fisk. Separating point sets by circles, and the recognition of digital disks. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 554–556, July 1986.

[3] J. O'Rourke, S. R. Kosaraju, and N. Megiddo. Computing circular separability. *Discrete and Computational Geometry*, 1:105–113, 1986.

The Pencil Packing Problem

Esther M. Arkin* Sándor P. Fekete† Joondong Kim* Joseph S.B. Mitchell*
Girishkumar R. Sabhnani‡ Jingyu Zou*

Abstract

We consider the following three-dimensional packing problem that arises in multibody motion planning in an environment with obstacles: Given an $n \times n \times n$ regular grid of voxels (cubes), with each voxel labeled as "empty" or "occupied", determine the maximum number of "pencils" that can be packed within the empty voxels, where a pencil is a union of n voxels that form an axis-parallel strip (i.e., a $1 \times 1 \times n$ box). No pencil is allowed to contain an occupied voxel, and no two pencils can have a shared voxel (since they form a packing). We consider also the dual (covering) problem in which we want to find the minimum number of empty "covering" voxels such that every pencil is intersected by at least one covering voxel. We show that both problems are NP-Hard and we give some approximation algorithms. We have evaluated our approximation algorithms experimentally and found that they perform very well in practice.

Keywords: Set Packing Problem, Set Cover Problem, Voxel Cube

1 Introduction

Problem Statement: We are given an $n \times n \times n$ regular grid, C, of voxels (cubes), with each voxel labeled as "empty" or "occupied". An axis-parallel strip of n face-adjacent empty voxels is called a *pencil*; it is a $1 \times 1 \times n$ block of n empty voxels that passes all the way through C, from one facet of the cube to the opposite facet.

Our goal is to pack within C as many pairwise-disjoint pencils as possible. The problem arises in a multiagent motion planning problem for moving as many disks as possible along discrete-orientation straight trajectories among a set of possibly moving obstacles in the plane; by lifting to space-time (3D) and transforming the coordinates, we arrive at the pencil packing problem.

For convenience, we place C aligned to the x, y and z axes. We let c_{ijk} be a binary variable indicating if voxel indexed (i, j, k) is occupied ($c_{ijk} = 1$) or not ($c_{ijk} = 0$). We let x_{jk} be a binary variable indicating the presence of the x-parallel pencil at position (j, k) in (y, z)-space. We similarly define variables y_{ik} and z_{ij}. Then, the *Pencil Packing Problem* is defined as follows: For given $c_{ijk} \in \{0, 1\}$, where $i, j, k \in N = \{1, \dots, n\}$,

$$\begin{aligned} \text{maximize} \quad & \sum_{j\in N, k\in N} x_{jk} + \sum_{i\in N, k\in N} y_{ik} + \sum_{i\in N, j\in N} z_{ij} \\ \text{s.t.} \quad & x_{jk} + y_{ik} + z_{jk} + c_{ijk} \le 1 \qquad i, j, k \in N \end{aligned}$$

Related to the pencil packing problem is a covering problem, defined as follows. Let v_{ijk} be a binary variable associated with voxel (i, j, k), indicating if this voxel is part of the cover. The *Voxel Cover Problem* is, for given $c_{ijk} \in \{0, 1\}$, where $i, j, k \in N$,

$$\begin{aligned} \text{minimize} \quad & \sum_{i\in N, j\in N, k\in N} v_{ijk} \\ \text{s.t.} \quad & \sum_{i\in N} (v_{ijk} + c_{ijk}) \ge 1 \qquad j, k \in N \\ & \sum_{j\in N} (v_{ijk} + c_{ijk}) \ge 1 \qquad i, k \in N \\ & \sum_{k\in N} (v_{ijk} + c_{ijk}) \ge 1 \qquad i, j \in N \end{aligned}$$

The Voxel Cover Problem is the dual problem of Pencil Packing Problem.

Related Work. Our problems are geometric variants of Set Packing and Minimum Set Cover, which are known to be APX-hard even if the cardinality of each set is bounded ([2], [3], [4]). One previous work [1] shows that the maximum independent set in 3-partite graphs is NP-hard and not approximable within 26/25 unless P=NP. The pencil packing problem can be viewed as a special case of the independent set problem in 3-partite graphs.

*Dept. Applied Mathematics and Statistics, Stony Brook University, {estie,jdkim,jsbm,jasonzou}@ams.sunysb.edu
†Dept. Computer Science, Braunschweig University of Technology, s.fekete@tu-bs.de
‡Metron Aviation, sabhnani@metronaviation.com

2 Hardness & Bounds

Using reductions from 3SAT, we prove the following hardness results:

Theorem 1 *NP-Hardness of Pencil Packing: Let C be a given voxel cube and k be an integer; then it is NP-complete to determine whether k pencils can be packed in C.*

Theorem 2 *NP-Hardness of Voxel Cover: Let C be a given voxel cube and l be an integer; then it is NP-complete to determine whether l empty voxels of C exist that cover all pencils of C.*

If we have any solution for the voxel cover problem of a given C, then we know that at least one pencil is required to cover each voxel in the covering; thus, any voxel cover solution gives an upper bound for pencil packing. Also. LP relaxations of the above IP problems give upper and lower bounds on the optimal solutions. For a given cube C, let $OPT_p(C)$, $OPT_v(C)$ be the optimal solution of the Pencil Packing, Voxel Cover Problems, and let $LP_p(C)$, $LP_v(C)$ be the LP relaxations. We have the following relationship.

$$OPT_p(C) \leq LP_p(C) = LP_v(C) \leq OPT_v(C)$$

For a given cube C, the *duality gap* is the difference between $OPT_p(C)$ and $OPT_v(C)$.

3 Approximation

A very simple approximation is immediate: Restrict attention to pencils with the same orientation and pack as many as possible. This gives a 1/3-approximation, since one of the three orientations has at least (1/3)OPT pencils packed.

A better approximation comes from a layered approach. Consider each pair of orientations of pencils (e.g., x- and y-), and pack each layer (corresponding to different z-coordinates) optimally, with purely x-parallel or purely y-parallel pencils, whichever gives more pencils for that layer (which is easy to determine for each layer, by projecting onto a single axis). This gives a 2/3-approximation, since, in one choice of orientation pairs, we will be able to obtain at least as many pencils as OPT has in those two orientations. In fact, unless the number of pencils of OPT is about the same in all three orientations, this method gives better than a 2/3-approximation. (We can slightly improve the method to give at least (2/3)OPT + 1 pencils.)

4 Experiments

We have implemented several methods for experimental comparison. In particular, we implemented a simple 1/3-APX, the layered 2/3-APX, a greedy algorithm to pack pencils, and a greedy algorithm to cover pencils (for use in an upper bound). We also used CPLEX to solve an IP formulation, as well as the corresponding LP relaxation, for both the pencil packing and the pixel covering problems.

We ran the experiments on 1800 instances. We used grid sizes having $n = 5, 10, 15, 20, 25, 30$. We selected voxels to be occupied independently with probabilities $p = 0.2, 0.1, 0.05, 0.01, 0.005, 0.001$. We also considered random clusters of occupied voxels, generated as L_1 balls of radius 0, 1, 2, 3, 4 centered on a randomly selected voxel. For each choice of parameters, we ran 10 randomly generated instances. We found that the 2/3-APX works remarkably well in practice, most often yielding the optimal solution. We also found that the duality gap is usually 0. A detailed comparison is given in the full paper.

Acknowledgements. We thank Alexander Kröller and Valentin Polishchuk for helpful discussions. This work was partially supported by NSF (CCF-0528209, CCF-0729019), NASA Ames, and Metron Aviation.

References

[1] A. Clementi, P. Crescenzi, and G. Rossi. On the complexity of approximating Colored-Graph problems. In *Computing and Combinatorics*, pages 281–290. 1999.

[2] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., 1979.

[3] V. Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Inf. Process. Lett.*, 37(1):27–35, 1991.

[4] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. In *Proc. 20th ACM Symposium on Theory of Computing*, pages 229–234, 1988.

A Dynamic Algorithm for Well-Spaced Point Sets (Abstract)

Umut A. Acar[*] Benoît Hudson[*] Duru Türkoğlu[†]

The Motivation and the Problem. A set of points is well-spaced if the Voronoi cell of each point has a bounded aspect ratio, i.e., the ratio of the distance to the farthest point in the Voronoi cell divided by the nearest neighbor distance is small. Informally, a set of points is well-spaced if its density varies smoothly. Well-spaced points sets relate strongly to meshing and triangulation for scientific computing: with minimal processing, they lead to quality meshes (e.g., no small angles) in two and higher dimensions. The Voronoi diagram of a well-spaced point set is also immediately useful for the Control Volume Method.

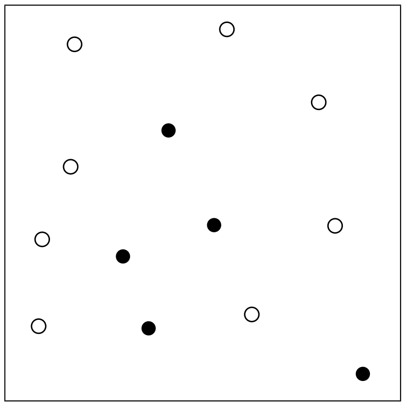


Figure 1: A well-spaced superset. Solid points (●) are input. Empty points (○) are Steiner vertices.

Given a finite set of points N in the d-dimensional unit hypercube $[0,1]^d$, the static well-spaced superset problem is to find a small, well-spaced set $M \supset N$ by inserting so called *Steiner* points. This problem has been studied extensively since the late 1980's, but efficient solutions that can generate outputs no more than a constant factor larger than optimal have been devised only recently. We are interested in the *dynamic* version of the problem. The problem requires maintaining a well-spaced superset (M) while the input (N) changes dynamically due to insertion and deletion of points. When a dynamic change takes place, a dynamic algorithm should efficiently update the output while keeping its size optimal for the new input. There has been relatively little progress on solving the dynamic problem; existing solutions are either not size-optimal or asymptotically no faster than a static algorithm.

Our Contributions. We present a dynamic algorithm that maintains an approximately optimal-sized, well-spaced output and requires worst-case $O(\log \Delta)$ time for an insertion or deletion, which we also prove to be optimal. Here, Δ is the *(geometric) spread* defined as $\frac{1}{\delta}$, where δ is the distance between the closest pair of points in the input. If the spread is polynomially bounded in n, then $O(\log \Delta) = O(\log n)$. We use dynamization to solve the problem. We first design a static algorithm that efficiently constructs an approximately optimal-sized, well-spaced superset of its input. For dynamic updates, the algorithm also constructs a *computation graph*, that represents the operations performed during the execution and the dependences between them. Then, given a modification to the input (i.e., an insertion/deletion), our dynamic algorithm updates the output and the computation graph by identifying the operations that depend on the modification and re-executing them. When re-executing an operation, the update algorithm inserts fresh operations that

[*]Toyota Technological Institute at Chicago
[†]University of Chicago

now need to be performed. Similarly, it removes invalid operations that should not be performed. We prove that the update algorithm ensures that the updated computation graph and the output are isomorphic to those that would be obtained by running the static algorithm with the modified input. As a corollary, we conclude that the output is well-spaced and has optimal size.

Proof Idea. The crux of the design and the analysis is to structure the computation in such a way that we can prove that a single modification to the input requires performing a small number of updates. First, we structure the computation into $\Theta(\log \Delta)$ levels (defined by ranks and colors) that the operations in each level depend only on the previous levels. Second, we pick Steiner vertices by making local decisions only, using clipped Voronoi cells. These techniques enable us to process each vertex only once and help isolate and limit the effects of a modification. More specifically, we prove that an insertion/deletion into/from the input causes $O(\log \Delta)$ vertices to become affected. The proof follows from a spacing-and-packing argument. The spacing argument shows that any affected vertex has an empty ball around itself whose radius is proportional to its distance to the vertex inserted or deleted (p). The packing argument shows that there can be only a constant number of affected vertices at each level, consequently, $O(\log \Delta)$ in total. Figure **??** illustrates the proof: each shade corresponds to a rank.

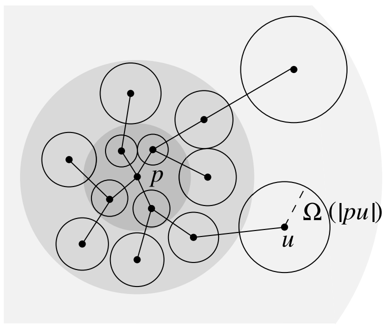


Figure 2: Illustration of the proof.

The Algorithm. We start with a few definitions. Given a vertex set $\mathcal{M}$, the *nearest neighbor distance of* v *in* $\mathcal{M}$, written $\mathrm{NN}_{\mathcal{M}}(v)$, is the distance from v to the closest other vertex in $\mathcal{M}$. The *Voronoi cell of* v *in* $\mathcal{M}$, written $\mathrm{Vor}_{\mathcal{M}}(v)$, consists of points $x \in [0,1]^d$ such that for all $u \in \mathcal{M}$, $|vx| \leq |ux|$. The *outradius* of $\mathrm{Vor}_{\mathcal{M}}(v)$ is the distance from v to the farthest point in $\mathrm{Vor}_{\mathcal{M}}(v)$ and the *aspect ratio* is outradius divided by $\mathrm{NN}_{\mathcal{M}}(v)$. For a given quality criterion $\rho > 1$, we say that a vertex v is *ρ-well-spaced* if the aspect ratio of its Voronoi cell is bounded by ρ and $\mathcal{M}$ is *ρ-well-spaced* if every point in $\mathcal{M}$ is ρ-well-spaced. For any constant $\beta > \rho$, we define the *β-clipped Voronoi cell*, written $\mathrm{Vor}^{\beta}_{\mathcal{M}}(v)$, as the intersection of $\mathrm{Vor}_{\mathcal{M}}(v)$ with the ball of radius $\beta\,\mathrm{NN}_{\mathcal{M}}(v)$ centered at v. Note that $\mathrm{Vor}^{\beta}_{\mathcal{M}}(v) \setminus \mathrm{Vor}^{\rho}_{\mathcal{M}}(v)$ is empty if and only if v is ρ-well-spaced.

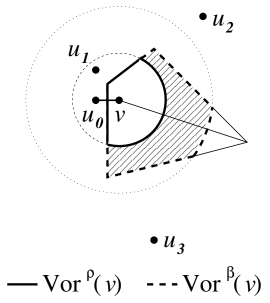


Figure 3: Nearest neighbor, outradius, ρ and β-clipped Voronoi cells.

Our static algorithm structures the computation into $\Theta(\log \Delta)$ ranks based on the nearest neighbor distances of the vertices and performs a single pass over the vertices ordered by their ranks. This technique requires selecting the Steiner vertices without affecting a previously processed vertex. Therefore, while processing a vertex $v \in \mathcal{M}$, we insert Steiner vertices only within the Voronoi cell of v, $\mathrm{Vor}_{\mathcal{M}}(v)$, but still far from v: at least at distance $\rho\,\mathrm{NN}_{\mathcal{M}}(v)$. Moreover, in order to support efficient updates, we select Steiner vertices by making local decisions only. More specifically, we select Steiner vertices within the β-clipped Voronoi cell of v; from $\mathrm{Vor}^{\beta}_{\mathcal{M}}(v) \setminus \mathrm{Vor}^{\rho}_{\mathcal{M}}(v)$ (Figure **??** shows this region). Even if we select Steiner vertices locally, these local decisions may combine and create long dependence chains which would require larger global restructuring during dynamic updates. To address this problem we partition the work in each rank into constantly many color classes. At each rank, we process the vertices in monotonically increasing order of their colors. This ensures independence: the Steiner vertices we insert while processing a vertex do not affect the decisions we make when processing other vertices of the same color. These techniques enable us to prove that dependence chains are short, of length $O(\log \Delta)$.

K-PLANE MATROIDS AND WHITELEY'S FLATTENING CONJECTURES

BRIGITTE SERVATIUS, JOINT WORK WITH H.S. AND WALTER WHITELEY

The so called *Molecular Conjecture* made by Tay and Whitely over 20 years ago states that a generically rigid (independent) body and hinge structure in 3-space with two bodies at each hinge will remain rigid (independent) under the constraint of making all hinges of a body concurrent (polar: all coplanar) Body and hinge frameworks with all hinges on a body concurrent are called molecular frameworks and their polar is called a plate framework. A proof of this conjecture and its higher dimensional analogue has recently been announced by Naoki Katoh and Shin-ichi Tanigawa [3]. We will briefly describe their proof and its relation to the work of Jackson and Jordán in [1] and [2].

An older result of Whiteley, [4], showed that a generically independent body and pin framework in the plane remains independent for realizations generic under the condition that all pins of each body are collinear. Recent work of Jackson and Jordán has confirmed that a generically rigid body pin framework, with two bodies at each pin, remains first-order rigid for realizations generic under the condition that all pins of each body are collinear.

A generalized conjecture remains:

Conjecture 1. *A generically rigid body pin framework, remains first-order rigid for realizations generic under the condition that all pins of each body are collinear, without restriction of how many bodies a pin is incident to.*

In the plane parallel drawing and plane rigidity are equivalent matroids. So the results in the plane could be restated as:

(A) A generically independent body and pin parallel drawing framework in the plane remains independent for realizations generic under the condition that all pins of each body are collinear.

(B) A generically tight body pin framework, with two parallel bodies at each pin, remains first-order rigid for realizations generic under the condition that all pins of each parallel body are collinear.

In fact, the original proof of (A) was cast in terms of parallel drawings, and the dual form for liftings and projections.

Since rigidity and parallel drawing are distinct, this suggests we could give a 3-space analog conjecture for parallel drawings. The statement below will also show how different this situation is from rigidity. It makes no sense to ask that two parallel-bodies (configurations with only trivial parallel drawings of translation and dilation) share an edge. This would, in fact, lock them both into a single body. So the analogy of a hinge remains a point shared by the two bodies, which leaves only one relative degree of freedom: different dilations for the two bodies.

A parallel-body pin framework in d-space is a collection of points (pins) and bodies (collections of the pins) and a realization of the pins as points in d-space. The body is assumed to group its pins into a parallel-tight unit, with only translations and dilations, see Figure 1.

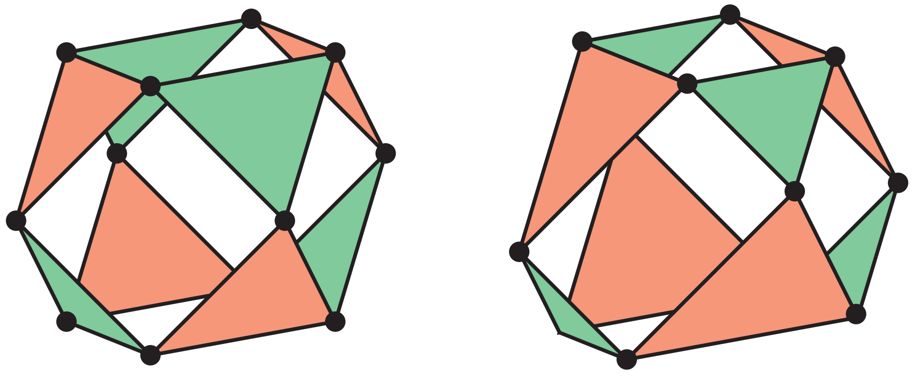


FIGURE 1. A generically tight framework with a parallel redrawing.

In this generality, it is easier to describe this as an incidence structure: $S = (V, B; I)$ where V indexes the points, B indexes the bodies, and $(i, j) \in I$ if pin i is in B. Note that we allow a pin to belong to any number of bodies, not just two bodies. A parallel-body pin framework in d-space is generic if the pins are generic in d-space. Whitely, [4], showed the following for dimension 2:

Theorem 2 (Whiteley). *If a parallel-body pin incidence structure is generically independent for generic configurations in d-space, then it remains independent under the condition that all pins of a body lie in a hyperplane of d-space.*

We offer the new general conjecture:

Conjecture 3 (Parallel Drawing Flattening Conjecture). *If a parallel body-and-pin incidence structure is generically tight for generic configurations in d-space, then it will remain tight for realizations generic under the constraint that all pins for a body are coplanar.*

REFERENCES

[1] B. JACKSON AND T. JORDÁN, *Pin-collinear body-and-pin frameworks and the molecular conjecture*, Discrete Comput. Geom., 40 (2008), pp. 258–278.

[2] ———, *The generic rank of body-bar-and-hinge frameworks*, EJC, in press (2009).

[3] N. KATOH AND S.-I. TANIGAWA, *A proof of the molecular conjecture*, in SCG '09: Proceedings of the 25th annual symposium on Computational geometry, New York, NY, USA, 2009, ACM, pp. 296–305.

[4] W. WHITELEY, *A matroid on hypergraphs, with applications in scene analysis and geometry*, Discrete Comput. Geom., 4 (1989), pp. 75–95.

Fast Construction of the Vietoris-Rips Complex[*]

Afra Zomorodian[†]

[Extended Abstract]

1 Introduction

In this paper, we present fast algorithms for constructing the filtered Vietoris-Rips complex [2] of a point set. This complex is currently the only practical method for analyzing datasets in higher dimensions. Our software can compute large complexes for points in arbitrary dimensions in seconds. It may be applied also toward constructing other graph-based complexes, such as the weak witness complex.

Suppose we are given a finite set of d-dimensional points $S \subseteq \mathbb{R}^d$. The *Vietoris-Rips complex* (*VR complex*) $\mathcal{V}_\epsilon(S)$ of S at scale ϵ is

$$\mathcal{V}_\epsilon(S) = \{\sigma \subseteq S \mid \mathrm{d}(u, v) \leq \epsilon, \forall u \neq v \in \sigma\},$$

where d is the Euclidean metric. Our approach is to separate the construction into two phases: We first compute the 1-skeleton of the complex, which is a graph. The *VR neighborhood graph* is $(G_\epsilon(S), w)$, where $G_\epsilon(S) = (S, E_\epsilon(S))$ is an undirected graph, $w\colon E \to \mathbb{R}$ is the edge *weight* function,

$$E_\epsilon(S) = \{\{u, v\} \mid \mathrm{d}(u, v) \leq \epsilon, u \neq v \in S\},$$
$$w(\{u, v\}) = \mathrm{d}(u, v), \forall \{u, v\} \in E_\epsilon(S).$$

Given a neighborhood graph (G, w), we *expand* the graph to the Vietoris-Rips complex $(\mathcal{V}(G), \omega)$ via *Vietoris-Rips expansion*: If $G = (V, E)$, we have

$$\mathcal{V}(G) = V \cup E \cup \left\{\sigma \mid \tbinom{\sigma}{2} \in E\right\},$$

For $\sigma \in \mathcal{V}(G)$,

$$\omega(\sigma) = \begin{cases} 0, & \sigma = \{v\}, v \in V, \\ w(\{u, v\}), & \sigma = \{u, v\} \in E \\ \max_{\tau \subset \sigma} \omega(\tau), & \text{otherwise.} \end{cases}$$

Note that the definition of $\mathcal{V}(G)$ is completely combinatorial and makes no reference to the metric on the embedding space. The first phase of the construction is closely related to a classic problem in computational geometry: the *nearest neighbor problem*: For a set of n points S in a metric space X, preprocess S so that given any query $q \in X$, the closest $p \in S$ can be reported quickly [3]. We survey eight algorithms for completing this step in the full manuscript.

2 Algorithms

In this section, we give three algorithms for completing the second phase of VR construction: the VR expansion of a neighborhood graph.

Inductive. This algorithm is reminiscent of the inductive definition of the CW complex. We build the complex one dimension at a time.

```
LOWER-NEIGHBORS$(G, u)$
1  return $\{v \in G.V \mid u > v,\ \{u, v\} \in G.E\}$
```

This function simply finds all neighbors of u within G that precede it in an ordering of the vertices.

```
INDUCTIVE-VR$(G, k)$
1  $\mathcal{V} = G.V \cup G.E$
2  for $i = 1$ to $k$
3      foreach $i$-simplex $\tau \in \mathcal{V}$
4          $N = \bigcap_{u \in \tau}$ LOWER-NEIGHBORS$(G, u)$
5          foreach $v \in N$
6              $\mathcal{V} = \mathcal{V} \cup \{\tau \cup \{v\}\}$
7  return $\mathcal{V}$
```

Incremental. This algorithm adds the vertices incrementally. When a vertex is added, we construct all needed cofaces for which the vertex is maximal.

```
INCREMENTAL-VR$(G, k)$
1  $\mathcal{V} = \emptyset$
2  foreach $u \in G.V$
3      $N =$ LOWER-NEIGHBORS$(G, u)$
4      ADD-COFACES$(G, k, \{v\}, N, \mathcal{V})$
5  return $\mathcal{V}$
```

[*]Research was partially supported by DARPA HR 0011-06-1-0038, ONR N 00014-08-1-0908, and NSF CAREER CCF-0845716.

[†]Department of Computer Science, Dartmouth College, Hanover, New Hampshire.

```
ADD-COFACES($G, k, \tau, N, \mathcal{V}$)
1  $\mathcal{V} = \mathcal{V} \cup \{\tau\}$
2  if $\dim(\tau) \geq k$
3      return
4  else
5      foreach $v \in N$
6          $\sigma = \tau \cup \{v\}$
7          $M = N \cap$ LOWER-NEIGHBORS($G, v$)
8          ADD-COFACES($G, k, \sigma, M, \mathcal{V}$)
```

Maximal. The maximal algorithm is based on the following simple observation: Maximal simplices in the VR complex are maximal cliques in its neighborhood graph. The algorithm is simple: First enumerate the set of all maximal cliques C; Then, generate all $(k+1)$-combinations of the resulting cliques to get the k-skeleton.

```
MAXIMAL-VR($G, k$)
1  $C =$ IK-GX($G$)
2  $\mathcal{V} =$ GENERATE-COMBINATIONS($C, k+1$)
3  return $\mathcal{V}$
```

For enumeration, we use the greedy algorithm IK-GX [1]. In GENERATE-COMBINATIONS, we enumerate combinations lexicographically using a classic algorithm in discrete mathematics. For all complexes, we also compute a weight function $\omega\colon \mathcal{V}_\epsilon(S) \to \mathbb{R}$ on the simplices in order to compute the filtration.

3 Experiments

We have implemented all algorithms in generic C^{++} as part of a library for computational topology and compare their performance with JPlex [4]. Figure 1 compares algorithms on the dataset *bunny* containing 34,837 3-dimensional points. Our fastest algorithm is 15 times faster than JPlex at the highest scale, computing a complex with over 9.7 million simplices in 21.27 seconds, compared to 313.38 seconds for JPlex. Figure 2 gives times, for increasing dimension, on the dataset *natural* containing 10,000 8-dimensional points. On this dataset, we are at least three times faster than JPlex in any dimension. We compute the 8-skeleton with 539,627 simplices 1.68 seconds. JPlex is currently limited to 7 dimensions and computes the 7-skeleton with 472,165 simplices in 6.68 seconds.

4 Conclusion

In this paper, we present a two-phase approach to computing the Vietoris-Rips complex, present three algorithms for phase 2, implement all algorithms, and present experimental results. Our work represents the first systematic examination of all the stages of the construction of the Vietoris-Rips complex.

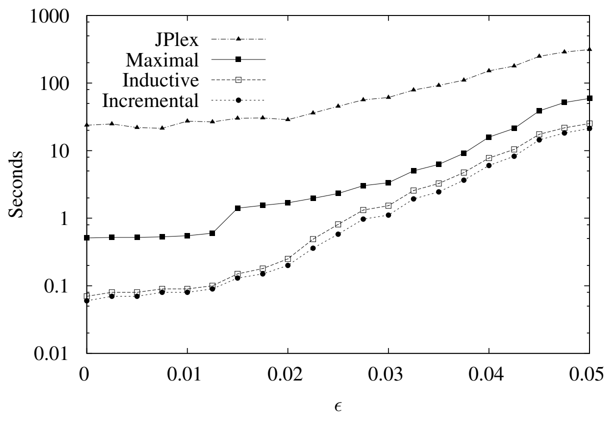


Figure 1: Comparison with JPlex on *bunny* dataset with increasing ϵ.

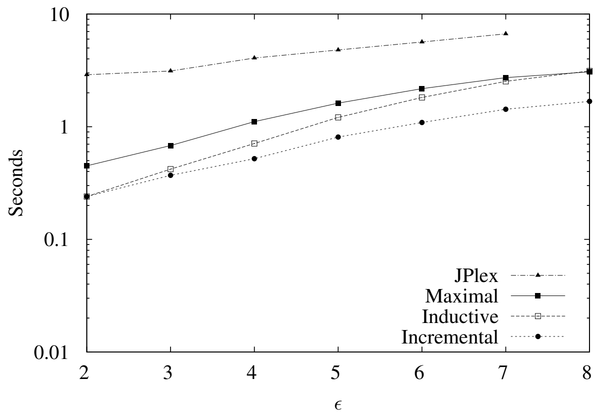


Figure 2: Comparison with JPlex on *natural* dataset with increasing dimension k.

References

[1] CAZALS, F., AND KARANDE, C. Reporting maximal cliques: new insights into an old problem. Research Report 5642, INRIA, 2005.

[2] GROMOV, M. Hyperbolic groups. In *Essays in Group Theory*, S. Gersten, Ed. Springer-Verlag, New York, NY, 1987, pp. 75–263.

[3] LIFSHITS, Y. The homepage of nearest neighbors and similarity search, 2009. `http://simsearch.yury.name/`.

[4] SEXTON, H., AND JOHANSSON, M. V. JPlex, 2009. `http://comptop.stanford.edu/programs/jplex/`.

Computing the Fréchet Distance Between Polyhedral Surfaces with Acyclic Dual Graphs[*]

Atlas F. Cook IV[†], Jessica Sherette[†], Carola Wenk[†]

I. Introduction

The Fréchet distance is a similarity metric for continuous shapes that has many practical applications. Although Godau [4] shows that computing the Fréchet distance between arbitrary surfaces is NP-Hard, Buchin et al. [3] show how to compute the Fréchet distance between simple polygons in polynomial time. Our work takes this process one step further by presenting a fixed-parameter tractable algorithm to compute the Fréchet distance between triangulated surfaces that have acyclic dual graphs.

II. Preliminaries

The *Fréchet distance* [1] is a similarity metric for continuous shapes that is defined for two surfaces $P, Q : [0,1]^2 \to \mathbb{R}^d$ as

$$\delta_F(P,Q) \;=\; \inf_{\sigma: A \to B} \sup_{p \in A} \|P(p) - Q(\sigma(p))\|$$

where σ ranges over orientation-preserving homeomorphisms that map each point $p \in P$ to an image point $q = \sigma(p) \in Q$, and $\|\cdot\|$ is the Euclidean norm. For a given constant $\varepsilon \geq 0$, *free space* is $\{(p,q) \mid p \in P,\ q \in Q,\ ||p-q|| \leq \varepsilon\}$.

In our setting, we assume that $P, Q : [0,1]^2 \to \mathbb{R}^d$ are connected triangulated surfaces. We also assume that the dual graphs of the triangulations are acyclic. We refer to the triangulation edges of P as *diagonals*, and we refer to the triangulation edges of Q as *edges*. Let m and n be the complexities of P and Q, respectively, and let k be the number of diagonals in P. In order to develop a fixed-parameter tractable algorithm, we assume that k is constant.

Previous work by Buchin et al. [3] computes the Fréchet distance between simple polygons P, Q. They show that the decision problem $\delta_F(P,Q) \leq \varepsilon$ can be computed by (1) mapping ∂P onto ∂Q such that $\delta_F(\partial P, \partial Q) \leq \varepsilon$ and (2) mapping each diagonal d in a triangulation of P to a shortest path $f \in Q$ such that both endpoints of f lie on ∂Q and such that $\delta_F(d,f) \leq \varepsilon$. Condition (1) can be enforced by finding a monotone path in the free space [1] (see Figure 1a). Such a path maps ∂P to ∂Q and defines a *placement* for each diagonal in P, i.e., a mapping of the endpoints of diagonals in P to endpoints of the corresponding image curves in Q. (Note that this assumes ∂P and ∂Q are cut open, and the algorithm eventually iterates over all combinatorially unique cuts.) Condition (2) is enforced by only considering paths in the free space that map the endpoints of a diagonal d onto an image curve f such that $\delta_F(d,f) \leq \varepsilon$. Buchin et al. [3] partition the free space into vertical slabs by the diagonal endpoints p_i on ∂P (see Figure 1a) and from this create a *reachability graph*. An edge between two consecutive diagonal endpoints is *feasible* if it represents a mapping from a diagonal $d \in P$ to a shortest path $f \in Q$ that satisfies condition (2).

III. Algorithm

In the simple polygon algorithm of Buchin et al. [3], it suffices to map diagonals onto shortest paths between two points on ∂Q. By contrast, there are polyhedral surfaces for which an optimal homeomorphism maps a diagonal $d \in P$ onto some non-shortest path in Q (see Figure 1b,c). Fortunately, it follows from a shortcutting argument that it suffices to only consider mapping a diagonal $d \in P$ onto polygonal paths in Q that cross the same sequence of edges as the shortest path between two given endpoints on ∂Q.

Observe that the points on ∂P should be mapped to points on ∂Q in a continuous fashion. This means that the endpoints of the image curves must appear on ∂Q in the same order as the respective diagonal endpoints on ∂P. We construct a reachability graph in much the same way as [3]. Incorporating condition (2), however, poses some problems because the image curves of the diagonals are no longer shortest paths and may cross each other. This means that we must explicitly ensure that the image curves of all diagonals are non-crossing. For the remainder of this discussion, we assume that we are given a placement of the diagonals (induced by a

[*]This work has been supported by the National Science Foundation grant NSF CAREER CCF-0643597.

[†]University of Texas at San Antonio, Department of Computer Science, {acook, jsherett, carola}@cs.utsa.edu

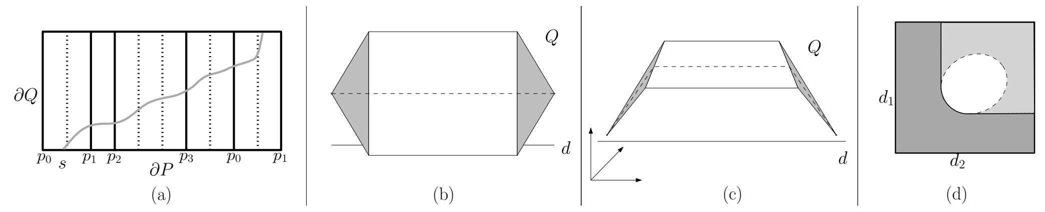


Figure 1. (a) One path through the free space (b),(c) Diagonal d is mapped to a non-shortest path in Q (d) P_e is shown in white, $C(P_e)$ is the union of the white and light grey portions

path in the free space). This defines a shortest path edge sequence in Q for each diagonal in P. Our main task is to determine whether it is possible to choose an image curve for each shortest path edge sequence such that the final set of images curves is pairwise non-crossing. We refer to this process as *untangling*.

We denote by D_e the set of diagonals whose associated shortest path edge sequences contain an edge $e \in Q$. Observe that pairwise non-crossing image curves must necessarily intersect an edge e in the same order as their end points occur on ∂Q. We refer to this as the *proper intersection order* for an edge e.

For every edge $e \in Q$, the *untangleability space* U_e consists of k-dimensional tuples of points such that a tuple is defined by fixing one point on each of the k diagonals. A tuple of points is contained in U_e if and only if the points on the diagonals in D_e can be mapped to e such that they are in the proper intersection order on e and such that the mapping of the diagonal points onto e involves distances of at most ε. In other words, U_e represents all mappings of the diagonals onto e such that the image curves can be locally untangled on e. We can use the convexity of the free space for a diagonal and an edge [1] to prove that U_e is also convex.

We now use these local structures to test whether a global untangling exists. Since we use the Fréchet distance to measure the distance between each diagonal in P and its image curve in Q, the mapping of every diagonal onto its image curve must be monotone. Unfortunately, locally untangling the diagonals on some edge $e \in Q$ may place a restriction on an edge $e' \in Q$ that invalidates the local untangling at e'. To test whether such a bad case has occurred, we first choose a boundary edge in Q to act as the root of the *edge tree* that corresponds to the dual graph of Q. We then propagate the monotonicity restrictions imposed by every local untangling up the tree to the root node. In more detail, we define for every edge e a k-dimensional *propagation space* P_e. If e is a leaf in the tree, then $P_e = U_e$. Otherwise, we assume that e is the parent of the edges e_1 and e_2 and define

$$P_e = U_e \cap C(P_{e_1} \cap C(P_{e_2})) \cap C(P_{e_2} \cap C(P_{e_1}))$$

where $C(P_{e_j})$ is the Minkowski sum of P_{e_j} with rays in each of the directions of the diagonals in D_{e_j} (see Figure 1d). $C(P_{e_j})$ contains only those points that are not excluded by the monotonicity constraints of e_j from being used to untangle on the parent of e_j. The propagation space for the root will be empty if and only if no global untangling exists. This can be computed in $O(n^{k+1})$ time as the intersection of semi-algebraic sets [2].

The free space diagram for ∂P and ∂Q contains $2k$ vertical line segments that will each contribute $O(mn)$ vertices to the reachability graph. Hence, there are $O((mn)^{2k})$ possible paths through the free space diagram. For each of these paths, we can determine whether a global untangling exists as described above in $O(n^{k+1})$ time. Similar to [3] we optimize ϵ by binary search.

Theorem 1. *The Fréchet distance for P and Q can be computed in $O(m^{2k} n^{3k+1} \log(nk))$ time.*

IV. Conclusion

We develop a fixed parameter tractable algorithm to compute the Fréchet distance between two triangulated surfaces with acyclic dual graphs. Although we account for dependencies between image curves, we have yet to find an example where the image curves could be locally untangled on every edge but a global untangling does not exist for them. If we can prove that such a case cannot happen then these dependencies would not influence the Fréchet distance, and we should be able to modify our algorithm to run in polynomial time.

References

[1] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Journal of Computational Geometry & Applications*, 5:75–91, 1995.

[2] S. Basu, R. Pollack, and M.-F. Roy. Algorithms in real algebraic geometry. *Algorithms and Computation in Mathematics*, 2006.

[3] K. Buchin, M. Buchin, and C. Wenk. Computing the Fréchet distance between simple polygons in polynomial time. *22nd Symposium on Computational Geometry (SoCG)*, pages 80–87, 2006.

[4] M. Godau. On the complexity of measuring the similarity between geometric objects in higher dimensions. *PhD thesis, Freie Universität Berlin, Germany*, 1998.

Routing Parallel and Merging Lanes

Joondong Kim* Joseph S. B. Mitchell* Jingyu Zou*

1 Introduction

We study a graph embedding problem that arises in air traffic management (ATM): Given a planar polygonal domain Ω (with holes), a set of source terminals $\{s_i\}_{i=1}^k$ and a sink terminal t in Ω, and two constants $r, \epsilon > 0$, find a (r, ϵ)-"fat graph embedding" of a tree T in Ω such that the number of source terminals s_i that are connected to t by directed paths in T is maximized. Such a tree T can serve as the "fan-in" tree for merging air traffic flows within the transition airspace, when multiple flows of arriving aircraft must merge into a single flow crossing an arrival "metering fix" at the boundary of the terminal airspace.

We first consider a related problem of routing through Ω a maximum number of noncrossing paths in a particular drawing of a layered DAG in Ω, from a set of source terminals $\{s_i\}_{i=1}^k$ on the boundary of Ω (the first layer) to a set of sink terminals $\{t_i\}_{i=1}^k$ on the boundary of Ω (the final layer). We show that this "parallel lane routing" problem is NP-hard for polygonal domains Ω.

We show that the tree routing problem in a given drawing of a layered DAG is NP-hard. We then consider the special case (typical in practice) in which there are a constant number of arcs in each path from s_i to t. For this case, we give a polynomial-time dynamic programming algorithm for the tree routing problem. We have implemented the dynamic programming algorithm and have performed experiments applying it to terminal airspace instances in which the obstacles are given by hazardous weather regions or special use airspace regions.

Motivation. As flows of arriving aircraft enter the "transition airspace", from en route to the terminal airspace in the vicinity of an airport, they are merged according to a fan-in tree, with each merge point having in-degree at most a small constant (typically 2). The routes must avoid certain "constraints", which include restricted airspaces and hazardous weather cells. In order for a controller to have sufficient time to monitor the safe separation of aircraft between merges, there should be a minimum separation between one merge point and the next along a route.

Given a transition airspace cluttered with "constraints" (obstacles), and given a "demand" of desired arrival flows that need to merge to the arrival metering fix, our objective is to accommodate as many arriving flows as possible, given the constraints on the fan-in tree. This problem arising in air traffic management (ATM) motivates a tree embedding problem that we formalize next.

Problem Statement. The input to our problem is a planar polygonal domain Ω with a set of polygonal holes, $\mathcal{H} = \{H_i\}_{i=1}^h$, and a total of n vertices. There is a set of k points $\{s_i \in \Omega\}_{i=1}^k$ designated as the source terminals, and one point $t \in \Omega$ designated as the sink terminal. For any $r, \epsilon > 0$ given, our objective is to find a (r, ϵ)-"fat graph embedding" of a tree, $T = (V, E)$, in Ω connecting all of the source terminals (or as many terminals as possible) to the sink terminal. For each vertex $v \in V$, its "fat embedding" is an open disk of radius r, $D_r(v) \subset \Omega$. If we let D_ϵ be the open disk of radius ϵ centered at the origin, and for any path π in $\mathbb{R}^2$, let $(\pi)^\epsilon$ be the Minkowski sum of π and D_ϵ, then for every edge $e = (u, v) \in E$ its fat embedding is an ϵ-thickened path $(\pi_e)^\epsilon \subset \Omega$ for some reference path π_e from $D_r(u)$ to $D_r(v)$. (The path π_e is $\pi_{(u,v)} \setminus (D_r(u) \cup D_r(v))$, for a path $\pi_{(u,v)}$ that embeds the edge linking point u to point v.) We require that the disks $\{D_r(v) | v \in V\}$ be pairwise disjoint, that the fat paths, $\{(\pi_e)^\epsilon | e \in E\}$, be pairwise disjoint, and that $(\pi_e)^\epsilon$ intersects only those disks, $D_r(u)$ and $D_r(v)$, that correspond to the endpoints of edge $e = (u, v)$. It should be noted that the maximum degree of T has to bounded by some constant $O(r/\epsilon)$ because the thickened paths are disjoint, even around some disk $D_r(v)$ where they are merging into.

Related Work. Our model of the fat graph embedding is similar to the model of "bold graph drawing" in [4], except that we allow non-straight edges and require stronger separation condition than the list of conditions given in [4] for good drawings. Our problem is also related to the thick edges drawing problem in [2]. For related work on the routing of thick paths and their application to ATM, see [1, 3].

*Dept. Applied Mathematics and Statistics, Stony Brook University, {jdkim,jsbm,jasonzou}@ams.sunysb.edu

2 Routing Parallel Lanes

We begin by considering a related problem involving routing of a set of non-crossing lanes. We are given a particular drawing of a layered DAG, $G = (V, E)$, in Ω, with $\{s_i\}_{i=1}^{k} \subset V$ on $\partial\Omega$ the source terminals and $\{t_i\}_{i=1}^{k} \subset V$ on $\partial\Omega$ the sink terminals. Given any integer $l \leq k$, we want to decide if there exists in G at least l paths connecting pairs of source terminals and sink terminals such that the drawings of the paths are pairwise disjoint. We prove this problem to be NP-hard using a reduction from INDEPENDENT SET. For any given graph G and given integer l, we construct an instance of our problem, shown in Figure 1, such that there exists an independent set of size l in G if and only if there exist at least l non-crossing paths in our instance.

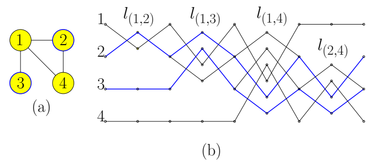


Figure 1: (a) An instance of INDEPENDENT SET, and (b) the corresponding layered DAG.

3 Routing Merging Lanes

In our motivating application, the portion of transition airspace under consideration is typically modeled as a portion of an annulus, corresponding to flights approaching an arrival metering fix. (Typically, there are four arrival metering fixes for a major airport.) Specifically, we work with a quadrant-shaped region $\Omega = \{(r\cos\theta, r\sin\theta) \in \mathbb{R}^2 | \theta_{\min} < \theta < \theta_{\max}, r_{\min} < r < r_{\max}\}$. Our application also requires that along each flight trajectory the distances between consecutive merge points be at least L and the in-degree of each merge point be at most a small constant (typically 2). We establish within Ω a set of concentric circular arcs ("layers") with radial distance increments of L and place along each such arc a set of candidate merge points (e.g., uniformly distributed). These candidate merge points are the locations where internal nodes of the merge tree (i.e., merge points) may be placed. We connect two merge points on consecutive layers with an edge if there is a feasible (obstacle-avoiding) thick route between them. (For simplicity, we consider only straight routes in the current implementation.) The resulting graph is a layered DAG (with a particular drawing), with layers corresponding to the nodes on the circular arcs (layers).

By a variant of the hardness construction for parallel lane routing (Figure 1), it is NP-hard to decide if there exists a non-crossing tree that routes at least l of the source terminals $\{s_i\}$ to the sink terminal t in the drawing of the layered DAG (assuming no transitions between embedded edges can occur except at their endpoints).

In practice, it may be that the number of layers in the DAG is bounded; such is the case for our ATM application, since there is a small upper bound (approximately 5) on the number of merges points along any one arrival route, and, further, $(r_{\max} - r_{\min})/L$ is expected to be bounded. In this case, we give an exact algorithm for computing an arrival merge tree, based on dynamic programming. The algorithm's running time is polynomial in the input size, n (the number of candidate merge points and the number of vertices describing the domain Ω), for fixed number of layers (which appears in the exponent of n in the running time). Figure 2 shows an example of a merge tree computed with our software tool, the "Tree-Based Route Planner" (TBRP). The TBRP allows for various objectives in the optimization, including maximizing the number of source terminals linked to t and minimizing the lengths of the routes from sources to sink.

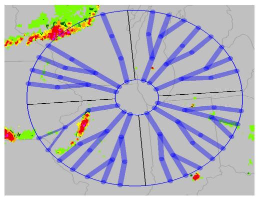


Figure 2: An example output of the implemented algorithm.

Acknowledgements. We thank Esther M. Arkin, Jimmy Krozel, and Girishkumar R. Sabhnani for helpful discussions. This work was partially supported by NSF (CCF-0528209, CCF-0729019), NASA Ames, and Metron Aviation.

References

[1] E. M. Arkin, J. S. B. Mitchell, and V. Polishchuk. Maximum thick paths in static and dynamic environments. In *Proc. 24th ACM Sympos. Computat. Geometry*, pages 20–27, 2008.

[2] C. Duncan, A. Efrat, S. Kobourov, and C. Wenk. Drawing with fat edges. In *Graph Drawing*, pages 162–177. 2002.

[3] J. S. B. Mitchell and V. Polishchuk. Thick non-crossing paths and minimum-cost flows in polygonal domains. In *Proc. 23rd ACM Sympos. Computat. Geometry*, pages 56–65, 2007.

[4] M. van Kreveld. Bold graph drawings. In *Proc. 21st Canadian Conference on Computat. Geometry*, pages 119–122, 2009.

The Gemcutting Approach to the Hirsch Conjecture (and Other Open Problems on Polytopal Graphs)

Anand Kulkarni
Department of Industrial Engineering and Operations Research
University of California, Berkeley
anandk@berkeley.edu

Sandeep Koranne
Mentor Graphics Corporation
sandeep_koranne@mentor.com

Polytopes are generalizations of polygons to higher dimensions that arise in many settings – among them, linear programming, game theory, genetics, and crystallography. A simple polytope is the bounded nondegenerate intersection of any n halfspaces in R^d. A face of a polytope is the intersection of any subset of its halfspaces; the poset of faces of a polytope ordered by inclusion form its face lattice and two polytopes are isomorphic iff their face lattices are equivalent. The graph of a simple polytope is formed by its 0- and 1-dimensional faces and also uniquely defines the polytope. A number of open combinatorial questions persist about polytopes, including questions about diameter, edge expansion, and the number of nonisomorphic face lattices [2].

We explore two new operations on combinatorial polytopes, introduced in [3]. Given the poset P of some polytope and a connected subset V of its edge-vertex graph, the *combinatorial cutting plane* problem asks for the face lattice $P' = P \cup H$, where H is a halfspace truncating V from P under some valid embedding of P in R^d. The *facet perturbation problem* takes as input a face lattice P, a $(d-1)$-dimensional face F, and a 0-dimensional face v and asks for the face lattice P' produced when F is perturbed to remove v from P in some realization of P. Although these problems are easy when restricted to a specific embedding of P as a set of halfspaces, they become nontrivial when generalized to combinatorial structures because an appropriate realization of P is not known in advance. Further, specific combinations of P, V, F, and v may be unrealizable. We present algorithms for each of these problems running in time $O(f_1 f_2)$, where f_1 and f_2 are the two largest elements of the face lattice.

These pair of operations provide value in reasoning about polytopes. We show that any combinatorial type of simple polytope may be generated from the simplex by applying a sequence of cutting planes and facet perturbations, enabling inductive proofs about combinatorial properties of polytopal graphs. The dual versions of these operations (known as bistellar flips) have been successfully applied to some of the central theorems in polytope theory, such as the g-theorem and Euler's relation. We show how cutting planes and facet perturbations can be used to provide a new line of attack on some open problems on simple polytopes – most notably, the long-standing Hirsch conjecture (see [5]), which asks whether the diameter of edge-vertex graphs of any $n-$facet polytope in R^d is at most $n - d$.

The general form of proofs using this "gemcutting" technique is always the same: we identify a sequence of cutting planes and perturbations that produce a particular class of polytopes, then show that the property is invariant under these perturbations. For the Hirsch conjecture, we've successfully proved that the diameter of a special class of maximal polytopes called Dantzig figures is almost always nonincreasing under perturbation of any facet [4]. This result reduces proof or disproof of the overall Hirsch conjecture to a new, tightly-constrained special case.

Moreover, these algorithms enable computational exploration of combinatorial properties of polytopes. Much of the general challenge in reasoning about polytopes stems from the difficulty in constructing or visualizing higher-dimensional polytopes; these objects do not always behave in the way suggested by three-dimensional examples. Using combinatorial cutting planes and facet perturbation, we can generate large families of higher-dimensional polytopes in an effort to rapidly test conjectured properties in dimensions higher than three. We present an algorithm using this strategy to enumerate a superset of the class of simple polytopes. The extreme size of this class, along with the possibility of resolving some unsolved questions by brute-force analysis, motivates some open questions about reducing the complexity of this algorithm.

Finally, we discuss the most recent progress from the Polymath project [1], a massively collaborative online effort to resolve open mathematical questions (including the Hirsch conjecture) using the contributions of several mathematicans across the web.

References

[1] T. Gowers and M. Nielsen, *Massively collaborative mathematics: The polymath project*, Nature, 461 (2009), pp. 879–881.

[2] B. Grünbaum, *Convex Polytopes*, Springer-Verlag, 2003.

[3] S. Koranne and A. Kulkarni, *Combinatorial Polytope Enumeration*, Arxiv preprint arXiv:0908.1619, (2009).

[4] A. Kulkarni and S. Koranne, *The d-step conjecture is almost true: Path preservation in polytopes*, http://ieor.berkeley.edu/ anandk/pubs/d-step.pdf, (2009).

[5] F. Santos and E. Kim, *An update on the Hirsch conjecture: Fifty-two years later*, Arxiv preprint arXiv:0907.1186, (2009).

Computing Hulls In Positive Definite Space*

P. Thomas Fletcher *fletcher@sci.utah.edu*
John Moeller *moeller@cs.utah.edu*
Jeff M. Phillips *jeffp@cs.utah.edu*
Suresh Venkatasubramanian *suresh@cs.utah.edu*

1 Introduction

There are many application areas where the basic objects of interest, rather than points in Euclidean space, are symmetric positive-definite $n \times n$ matrices (denoted by $\mathrm{P}(n)$). In diffusion tensor imaging [3], matrices in $\mathrm{P}(3)$ model the flow of water at each voxel of a brain scan. In mechanical engineering [7], stress tensors are modeled as elements of $\mathrm{P}(6)$. Kernel matrices in machine learning are elements of $\mathrm{P}(n)$ [12].

In these areas, a problem of great interest is the analysis [8, 9] of collections of such matrices (finding central points, clustering, doing regression). Since the geometry of $\mathrm{P}(n)$ is non-Euclidean, it is difficult to apply standard computational geometry tools.

The convex hull is fundamental to computational geometry. It can be used to manage the geometry of $\mathrm{P}(n)$, to find a center of a point set (via *convex hull peeling depth* [11, 2]), and capture extent properties of data sets like diameter, width, and bounding boxes (even in its approximate form [1]).

We introduce a generalization of the convex hull that can be computed (approximately) efficiently in $\mathrm{P}(2)$, is identical to the convex hull in Euclidean space, and always contains the convex hull in $\mathrm{P}(n)$. In the process, we also develop a generalized notion of *extent* [1] that might be of independent interest.

Convex Hulls in $\mathrm{P}(n)$. $\mathrm{P}(n)$ is an example of a proper CAT(0) space [6, II.10], and as such admits a well-defined notion of convexity, in which metric balls are convex. We can define the convex hull of a set of points as the smallest convex set that contains the points. This hull can be realized as the limit of an iterative procedure where we draw all geodesics between data points, add all the new points to the set, and repeat.

Lemma 1.1 ([5]). *If $X_0 = X$ and $X_{i+1} = \bigcup_{a,b \in X_i} [a, b]$, then $\mathcal{C}(X) = \bigcup_{i=0}^{\infty} X_i$.*

Berger [4] notes that it is unknown whether the convex hull of three points is in general closed, and the standing conjecture is that it is not. The above lemma bears this out, as it is an infinite union of closed sets, which in general is not closed. These facts present a significant barrier to the computation of convex hulls on general manifolds.

The *ball hull* of a set of points is the intersection of all (closed) metric balls containing the set. The ball hull has the advantage of being convex and closed. We provide an algorithm that computes an approximate ball hull.

*This research was supported in part by a subaward to the University of Utah under NSF award 0937060 to the Computing Research Association and SGER-0841185.

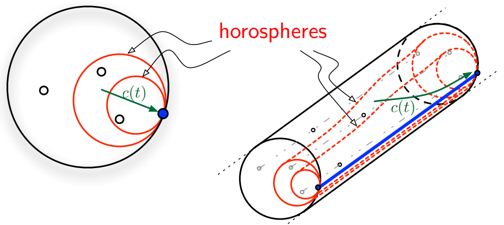


Figure 1: Left: projection of $X \subset \mathrm{P}(2)$ onto $\det(x) = 1$. Right: $X \subset \mathrm{P}(2)$. Two horospheres are drawn in both views.

2 Definitions

$\mathrm{P}(n)$ is the set of symmetric positive-definite real matrices. It is a Riemannian metric space with tangent space at point p equal to $\mathrm{S}(n)$, the vector space of symmetric matrices with inner product $\langle A, B\rangle_p = \mathrm{tr}(p^{-1}Ap^{-1}B)$. The exp map, $\exp_p : \mathrm{S}(n) \to \mathrm{P}(n)$ is defined $\exp_p(tA) = c(t) = p^{\frac{1}{2}} e^{p^{-\frac{1}{2}} tAp^{-\frac{1}{2}}} p^{\frac{1}{2}}$, where $c(t)$ is the geodesic with unit tangent A and $c(0) = p$. For simplicity, we often assume that $p = I$ so $\exp_I(tA) = e^{tA}$. The log map, $\log_p : \mathrm{P}(n) \to \mathrm{S}(n)$, indicates direction and distance and is the inverse of $\exp_p$. The metric $d(p, q) = \| \log_p(q)\| = \sqrt{\mathrm{tr}(\log(p^{-1}q)^2)}$.

Given a geodesic ray $c(t) : \mathbb{R}^+ \to \mathrm{P}(n)$, a *Busemann function* $b_c : \mathrm{P}(n) \to \mathbb{R}$ is defined

$$b_c(p) = \lim_{t\to\infty} d(p, c(t)) - t.$$

A Busemann function is an example of a *horofunction* (see [6, II.8]). A *horoball* $B_r(h) \subset \mathrm{P}(n)$ is a sublevel set of a horofunction h, i.e. $B_r(h) = h^{-1}((-\infty, r])$. By [6, II.8], h is convex, so any sublevel set $B_r(h)$ is convex. A *horosphere* $S_r(h)$ is its boundary; i.e. $S_r(h) = h^{-1}(r)$. See Figure 1.

The *geodesic anisotropy* [10] of a point $p \in \mathrm{P}(n)$ measures disparity in its eigenvalues, and will be useful in our analysis. It is defined as $\mathrm{GA}(p) = d(\sqrt[n]{\det(p)}I, p) = \mathrm{tr}(\log(p/\sqrt[n]{\det(p)})^2)^{\frac{1}{2}}$.

Busemann functions in $\mathbb{R}^n$. As an illustration, we can easily compute the Busemann function in Euclidean space associated with a geodesic ray $c(t) = t\mathbf{u}$, where $\mathbf{u}$ is a unit vector. Since $\lim_{t\to\infty} \frac{1}{2t}(\|p - t\mathbf{u}\| + t) = 1$, $b_c(p) = \lim_{t\to\infty} \frac{1}{2t}(\|p - t\mathbf{u}\|^2 - t^2)$ and

$$b_c(p) = \lim_{t\to\infty} \frac{\|p\|^2}{2t} - \langle p, \mathbf{u}\rangle = -\langle p, \mathbf{u}\rangle\,.$$

Horospheres in Euclidean space are then just hyperplanes, and horoballs are halfspaces.

Busemann Functions in P(n) For geodesic $c(t) = e^{tA}$, where $A \in$ S(n), the Busemann function $b_c : $ P$(n) \to \mathbb{R}$ is

$$b_c(p) = -\operatorname{tr}(A \log(\pi_c(p))),$$

where π_c is defined below [6, II.10].

There is a subgroup of GL(n), N_c (the *horospherical group*), that leaves the Busemann function b_c invariant [6, II.10]. That is, given $p \in$ P(n), and $\nu \in N_c$, $b_c(\nu p \nu^T) = b_c(p)$. Let A be diagonal, where $A_{ii} > A_{jj}$, $\forall i \neq j$. Let $c(t) = e^{tA}$. Then $\nu \in N_c$ if and only if ν is a upper-triangular matrix with ones on the diagonal[1]. If $A \in S(n)$ is not sorted-diagonal, we may still use this characterization of N_c without loss of generality, since we may compute an appropriate diagonalization $A = QA'Q^T$, $QQ^T = I$, then apply the isometry $Q^T p Q$ to any element $p \in$ P(n).

Let $A \in$ S(n) and $c(t) = e^{tA}$ as above. If we consider all elements $f \in$ P(n) that share eigenvectors Q with e^A, then $fe^A = e^A f$, and we call this space F_c, the *n-flat* containing c. If A is diagonal, $f \in F_c$ is diagonal, where the diagonal elements are positive [6, II.10]. Since we may assume that $Q \in$ SO(n), every flat F_c corresponds to an element of SO(n). Moreover, since members of F_c commute, $\sqrt{\operatorname{tr}(\log(u^{-1}v)^2)} = \sqrt{\operatorname{tr}((\log(v) - \log(u))^2)}$ for all $u, v \in F_c$ so F_c is isometric to $\mathbb{R}^n$ with the Euclidean metric.

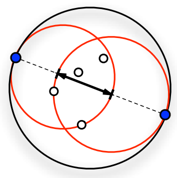
Given $p \in$ P(n), there is a unique decomposition $p = \nu f \nu^T$ where $(\nu, f) \in N_c \times F_c$ [6, II.10]. Let $p \in$ P(n) and $(\nu, f) \in N_c \times F_c$. If $p = \nu f \nu^T$, then define the *horospherical projection function* $\pi_c :$ P$(n) \to$ P(n) as $\pi_c(p) = \nu^{-1} p \nu^{-T} = f$.

3 Ball Hulls

For a subset $X \subset$ P(n), the *ball hull* $\mathcal{B}(X)$ is the intersection of all horoballs that also contain X:

$$\mathcal{B}(X) = \bigcap_{b_c, r} B_r(b_c), \; X \subset B_r(b_c).$$

We know that any horoball is convex. Because the ball hull is the intersection of convex sets, it is itself convex (and therefore $\mathcal{C}(X) \subseteq \mathcal{B}(X)$).


horoextent

Since measuring width by recording the distance between two parallel planes does not have a direct analogue in P(n), we propose the use of *horoextents*. Let $c(t) = e^{tA}$ be a geodesic ray, and $X \subset$ P(n). The *horoextent* $E_c(X)$ with respect to c is defined as:

$$E_c(X) = \left| \max_{x \in X} b_c(x) + \max_{x \in X} b_{-c}(x) \right|$$

where $-c$ is understood to mean $e^{t(-A)}$ (the ray pointing opposite c)[2]. Observe that for any c, $E_c(X) = E_c(\mathcal{C}(X)) = E_c(\mathcal{B}(X))$.

[1]For simplicity, we consider only those rays with unique diagonal entries, but this definition may be extended to those with multiplicity.

[2]While $-A$ and A share eigenvectors, the eigenvalues of $-A$ will be sorted opposite to those of A, so the projections π_c and π_{-c} will be different.

4 Algorithm for ε-Ball Hull

An intersection of horoballs is called a *ε-ball hull* ($\mathcal{B}_\varepsilon(X)$) if for all geodesic rays c, $|E_c(\mathcal{B}_\varepsilon(X)) - E_c(X)| \leq \varepsilon$.

Lemma 4.1. *For any horosphere $S_r(b_c)$, there is a hyperplane $H_r \subset \log(F_c) \subset S(n)$ such that $\log(\pi_c(S_r(b_c))) = H_r$.*

Lemma 4.2 (Lipschitz condition on P(2))**.** *Given a point $p \in P(2)$, a rotation matrix Q corresponding to an angle of $\theta/2$, geodesics $c(t) = e^{tA}$ and $c'(t) = e^{tQAQ^T}$,*

$$b_c(p) - b_{c'}(p) \leq |\theta| \cdot 2\sqrt{2} \sinh\left(\mathrm{GA}(p)/\sqrt{2}\right).$$

Algorithm. For $X \subset$ P(2) we can construct $\mathcal{B}_\varepsilon(X)$ as follows. Let $g_X = \max_{p \in X} \mathrm{GA}(p)$. We place a grid G_ε on SO(2) so that for any $\theta' \in$ SO(2), there is another $\theta \in G_\varepsilon$ such that $|\theta - \theta'| \leq (\varepsilon/2)/(2\sqrt{2}\sinh(g_X/\sqrt{2}))$. For each c corresponding to $\theta \in G_\varepsilon$, we consider $\pi_c(X)$, the projection of X into the 2-flat F_c. Within F_c, we construct a convex hull of $\pi_c(X)$, and return the horoball associated with each hyperplane passing through each facet of the convex hull, as in Lemma 4.1. Since between elements of G_ε, the points of $\pi_c(X)$ do not change the values of their horofunctions by more than $\varepsilon/2$ (by Lemma 4.2), the extents do not change by more than ε, and the returned set of horoballs is a $\mathcal{B}_\varepsilon(X)$.

Theorem 4.1. *For a set $X \subset P(2)$ of size N, we can construct an $\mathcal{B}_\varepsilon(X)$ of size $O((\sinh(g_X)/\varepsilon)N)$ in time $O((\sinh(g_X)/\varepsilon)N \log N)$.*

This can be improved by using an ε-kernel [1] on $\pi_c(X)$.

References

[1] Agarwal, P. K., Har-Peled, S., and Varadarajan, K. R. Approximating extent measures of points. *JACM 51* (2004).

[2] Barnett, V. The ordering of multivariate data. *Journal of the Royal Statistical Society. Series A 139* (1976), 318 – 355.

[3] Basser, P. J., Mattiello, J., and LeBihan, D. MR diffusion tensor spectroscopy and imaging. *Biophys. J. 66*, 1 (1994), 259–267.

[4] Berger, M. *A Panoramic View of Riemannian Geometry*. Springer, 2007.

[5] Bhatia, R. *Positive Definite Matrices*. Princeton University Press, 2006.

[6] Bridson, M. R., and Haefliger, A. *Metric Spaces of Non-Positive Curvature*. Springer, 2009.

[7] Cowin, S. The structure of the linear anisotropic elastic symmetries. *Journal of the Mechanics and Physics of Solids 40* (1992), 1459–1471.

[8] Fletcher, P. T., and Joshi, S. Principal geodesic analysis on symmetric spaces: Statistics of diffusion tensors. In *Computer Vision and Mathematical Methods in Medical and Biomedical Image Analysis* (2004), pp. 87 – 98.

[9] Fletcher, P. T., Venkatasubramanian, S., and Joshi, S. The geometric median on riemannian manifolds with application to robust atlas estimation. *NeuroImage 45* (2009), S143–52.

[10] Moakher, M., and Batchelor, P. G. Symmetric positive-definite matrices: From geometry to applications and visualization. In *Visualization and Processing of Tensor Fields*. Springer, 2006.

[11] Shamos, M. I. *Geometry and statistics: Problems at the interface*. Computer Science Department, Carnegie-Mellon University, 1976.

[12] Shawe-Taylor, J., and Cristianini, N. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

Computational Geometry Problems in CAD

Kirk Haller

Parametric solid modeling has been the defacto standard in computer aided design for mechanical engineering for the last 25 years. However, there are still fundamental issues with this paradigm that remain unresolved. Additionally, the availability of highly parallel desktop computers and cloud computing present new challenges. In this talk, we will present problems rather than solutions, focusing on issues related to computational geometry of interest to the CAD community.

Reporting flock patterns via GPU*

Marta Fort J. Antoni Sellarès Nacho Valladares
Institut d'Informàtica i Aplicacions, Universitat de Girona, Spain. Email: {mfort,sellares,ivalladares}@ima.udg.edu

I. Introduction

The study of movement is an important feature which can report very useful information in many application areas such as animal behavior, traffic management or professional sports. In most cases, moving object data sets are rather large in volume and complex in the structure of movement patterns they record. Therefore, it is necessary to develop efficient data mining algorithms in order to extract useful and relevant information. Exact and approximate algorithms, commonly based on Computational Geometry techniques, are used to compute these patterns. All approaches presented so far about pattern detection are CPU based.

The increasing programmability and high computational rates of graphic processing units (GPUs) make them attractive as an alternative to CPUs for general-purpose computing. GPU parallelism and tasks distribution considerably increases the algorithms efficiency [7].

The main aim of this paper is to use Computational Geometry techniques together with GPU capabilities to solve problems related to flock patterns detection in order to obtain reasonable running times.

II. Preliminaries

A. Computing Voronoi diagrams with the GPU

Hoff et al. [5] construct the 2D Voronoi diagram of a set S of points in the plane with the GPU. The distance function of a site in the plane is represented by a cone in space with $\pi/4$ angle discretized by triangles. Each site is assigned a unique color ID, and the corresponding distance mesh is rendered with that color using a parallel projection. They use polygon rendering hardware and the Z-buffer test to keep the closest fragments. Fischer et al. in [3] propose a technique based on an arrangement of planes, which are tangent to the unit paraboloid. In this algorithm the error produced by the discretization of cones into triangles disappears and makes the algorithm more precise. However, depth values stored in the depth buffer do not coincide with the Euclidian distance to the sites.

Both algorithms can report, not just the Voronoi diagram, but higher order Voronoi diagrams using the depth peeling technique proposed by Everitt [2]. This way we can obtain the k-order Voronoi diagram by composing the first k-levels of an arrangement of cones or planes with transparency $1/k$.

B. Computing smallest enclosing circle with the GPU

We can determine the center C of the smallest enclosing circle (SEC) of the set S via the GPU as follows. We compute the farthest Voronoi diagram keeping the farthest fragments of the arrangement of cones. This way, at every pixel q, the depth buffer stores the $\max_{s\in S} d(s,q)$ value. The pixel position corresponding to the minimum stored value in the depth buffer is the approximated center of C and his value, the radius.

C. The flock pattern

Flock pattern has been previously defined in [6] and [4]. Benkert et al. proposed in [1] a most elaborated definition of a flock and presented an algorithm, based on range counting queries in a high dimensional space.

Let P be a set of n trajectories $p_1, \ldots, p_n$, where each trajectory p_i is a sequence of τ points in the plane $p_i = \{s_1^i, \cdots, s_\tau^i\}$, where $s_j^i = (x_j^i, y_j^i)$ denotes the position of entity p_i at time t_j and $0 \leq j < \tau$. We will assume that the movement of an entity from its position at time t_j to its position at time t_{j+1} is described by the straight-line segment between the two points, and that the entity moves along the segment with constant velocity. Let I be a time interval $[t_i, t_j]$, with $j - i > m$ and $i < j < \tau$.

Definition 1 (Benkert et al.): An $(m,k,r)-Flock$ in time interval I consists of at least k entities such that for every discrete time-step $t_\ell \in I$, there is a disk of radius r that contains all the k entities.

III. Computing flocks via GPU

Definition 2: A $(k,r) - Subflock_j$ consists of at least k entities such that there is a disk of radius r that contains all the k entities at time-step t_j.

Consequently, we can think in an $(m,k,r) - Flock$ as m consecutive $(k,r) - Subflock_j$ with the same k entities.

The algorithm of Laube et al. [6] is the basis of our approach. To detect a $(k,r) - Subflock_j$ in step t_j, it constructs the k-order Voronoi diagram of the set $S_j = \{s_j^1, \cdots, s_j^n\}$ to find which groups of k entities lie closer together, and then computes the SEC of this k entities. If SEC radius is less than r, the entities define a subflock.

Our goal is to develop an algorithm that combines the robustness of Benkert et al. definition with the algorithm given by Laube et al. by using graphics hardware.

A. Voronoi diagrams via fragment shader

We propose a new algorithm for computing high order Voronoi diagrams which best meets to our needs. The main ideas are the same as the ones used by the cones algorithm. Each site has a unique color ID and the corresponding cone is rendered using this color. The goal of the algorithm is to create the cones via fragment shader in order to avoid the discretization of cones into triangles. For each site s_j^i we render a quad covering the whole screen with s_j^i as texture coordinates at each vertex. This way the interpolated value at any fragment will be the position of the site. Each quad will be rasterized in $H \times W$ fragments, where H and W are the height and width screen resolution respectively, that will be processed by the fragment shader. The fragment shader assigns to each fragment a depth value equal to the Euclidian distance between site s_j^i and the fragment position.

This algorithm avoids the problem generated by the discretization of cones into triangles and is as precise as the tangent planes algorithm. In addition, the stored depth buffer values represent the Euclidian distance so we can compute SEC as explained in Section II-B. Using the depth peeling technique we can obtain the different levels of the arrangement of cones. Moreover, in the same shader where we peel the arrangement we also compute the depth value.

B. Computing all SEC's simultaneously

We compute all smallest enclosing circles for every subset of k closest entities in one single step, once the k-level is computed.

Observation 1: The collection of minimum values associated to the farthest Voronoi diagram of groups of k closest entities coincides with the collection of minimum local values associated to the k-level of the arrangement of cones.

That means that finding the local minima of the k-level is the same as finding the minimum of each farthest Voronoi diagrams of groups of k closest entities.

* Partially supported by the Spanish MEC grant TIN2007-67982-C02-02.

C. Finding local minimum values

The intersections between cone entities are the boundaries of the Voronoi regions and the local minima must be located somewhere in these boundaries. Since the intersection between two parallel cones is a parabola, the boundaries of the Voronoi regions define an arrangement of parabolas where each parabola has a unique minimum value. The minimum values of the arrangement of parabolas are the local minima we are looking for. We can find these minimum values by passing a 3×3 mask through the depth texture. If all the values in the mask are smaller than its center value then the center value is a local minimum. In this way, just in one step we obtain all the local minima of the k-level corresponding to all the smallest enclosing circles of the groups of k nearest neighbors. We can find the entities determining the local minima by reading at the local minimum positions the color of each κ-level, $\kappa = 1, \cdots, k$. The problem is that we can not store the local minima until the peeling ends, because we need the depth of the k-level. This can be solved by storing the k levels in texture memory and once we have computed the local minima then read from each texture the colors associated to the local minima positions.

We have to take into account that the maximum number of textures in GPU memory is limited (rarely is bigger than 45). However, maximum height and width of textures is much larger than we usually need. For instance, a maximum texture size can be 4096×4097, while we typically use 1024×1024 size. Instead of creating k textures of our screen resolution size, we can create a very big texture containing several arrangement levels. A total of 512 levels of arrangement can be stored with this improvement.

D. Peeling improvement

When we paint a quad of size HW for a site s_j^i, generally most of their fragments have a depth value greater than r which are discarded according to the definition of flock. This can be avoided by rendering a smallest quad bigger enough to assure that all fragments with depth value from 0 to r will be processed (see Figure 1). For site s_j^i we take the quad determined by vertices $(x_j^i - r, y_j^i - r)$ and $(x_j^i + r, y_j^i + r)$. In this way we improve the running time of the peeling process.

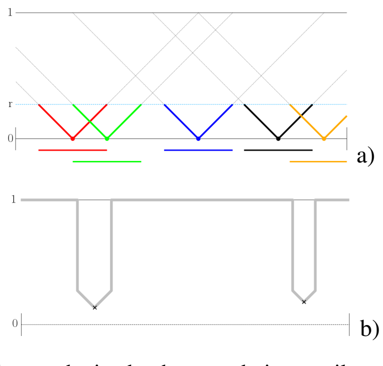


Fig. 1. a) Cones obtained when rendering until r depth value; b) Resulting depth values for 1 peel ($k = 2$) and minimum local depth values marked with crosses.

E. The algorithm

For every time step $t_j, 0 \leq j < \tau$, we peel the k levels of the arrangement of cones and store them in textures. We also store the depth buffer obtained at k-level. Then we find the local minimum values from the depth texture by discarding those values bigger than r. For each local minimum, we read the color from levels 1 to k obtaining the entities forming the corresponding subflock at t_j.

For each group of k entities we have to mark the group as subflock found at time step t_j. Then, when a subflock appears in m consecutive time steps we report a flock. We have to take into account that the consecutive time steps must contain the same entities. To assure this, we store all the obtained subflocks in a map data structure. The map index is defined by the ordered list containing the IDs of the entities conforming each subflock. The map stores for each index another ordered list containing the time steps where the subflock occurs. If we have m consecutive time steps in the same index we can guarantee that the flock is formed by the same entities.

F. Time complexity

Let $\rho \ll \phi \ll T$ be the time needed to render, copy/load a texture and read to CPU a fragment, respectively. To store the k-levels of the arrangements takes $O(\tau HW[k(n\rho + \phi) + T])$ time, determining the local minima and the entities defining subflock takes $O(\tau k[\mu T + \phi HW])$, where μ denotes the total number of obtained local minima. Map operations time is understimated because is negligible.

IV. Results

The algorithm has been tested under an Intel(R) Core(TM)2 CPU 6400 2.13GHz and a GeForce 9800 GTX+. The data sets have been artificially generated with NetLogo [8]. We have modified the NetLogo's Flocking model adjusting it to our needs.

We have normalized the problem in $[0, 1]^2$ space. In practice, we choose the normalized value of r depending on the area containing the trajectories. Table 1 shows the final results obtained with our algorithm with parameters: $m = 5$, $\tau = 30$, $r = 4 \times 10^{-3}$.

Input		*Output*	
n	k	Flocks	Time
1000	5	5	0.959
	10	0	0.983
	50	0	2.491
5000	5	43	1.025
	10	25	1.119
	50	11	3.189
10000	5	96	1.181
	10	45	1.401
	50	22	4.479

Tab. 1. Reported number of flocks and running times in seconds.

The number of flocks reported is smaller for small values of n. This is reasonable because the area covered by the entities is the same in all cases. For a fixed value of r and high values of n, it is more probably that entities are closer enough, consequently a bigger number of flocks are detected.

References

[1] M. Benkert, J. Gudmundsson, F. Hübner, and T. Wolle. Reporting flock patterns. *Computational Geometry*, 41(3):111–125, 2008.
[2] C. Everitt. Interactive order-independent transparency. Technical report, NVIDIA Corporation, 2001.
[3] I. Fischer and C. Gotsman. Fast Approximation of High-Order Voronoi Diagrams and Distance Transforms on the GPU. *Journal of Graphics, GPU, and Game Tools*, 11(4):39–60, 2006.
[4] J. Gudmundsson, M. J. van Kreveld, and B. Speckmann. Efficient detection of patterns in 2d trajectories of moving points. *GeoInformatica*, 11(2):195–215, 2007.
[5] K. Hoff, 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. ACM Press/Addison-Wesley Publishing Co., 1999.
[6] P. Laube, M. van Kreveld, and S. Imfield. Finding REMO - Detecting Relative Motion Patterns in Geospatial Lifelines. *Developments in Spatial Data Handling: 11th Int. Sympos. on Spatial Data Handling*, pages 201–215, 2004.
[7] D. P. Luebke, M. Harris, N. K. Govindaraju, et al. GPGPU: general-purpose computation on graphics hardware. In *SC*, page 208. ACM Press, 2006.
[8] U. Wilensky. NetLogo. http://ccl.northwestern.edu/netlogo.

Triangle-based Prism Mesh Generation on Interconnect Models for Electromagnetic Simulations

Shu Ye
Department of Computer Science, University of Massachusetts, Lowell
(shuy@cadence.com)

Karen Daniels
Associate Professor, Department of Computer Science, University of Massachusetts, Lowell
(kdaniels@cs.uml.edu)

Background

Serpentine line and Via are two common and important interconnect structures which can be found in today's Printed Circuit Board (PCB) and packaging design. In Gigabit per second serial data transmission, accurately modeling over multiple GHz frequency range becomes critical in order to characterize signal channel and determine the quality of transmitted signal on PCB and packaging. In recent years, the interconnect modeling on PCB and in packaging has played an important role for successful high-speed circuit design [1]. The interconnect characterization is best done through electromagnetic (EM) field solving. Among other numerical techniques, the Finite Element Method (FEM) is typically in use. However, the FEM computation relies heavily on the quality of grids and meshing generation, in terms of the accuracy and performance [2]. This paper discusses the problem of generating prism-based finite elements over 3D interconnect structures for EM modeling and simulation on PCB and electrical packaging. A practical algorithm introduced here is based on constrained Delaunay triangulation and extended into 3D triangle-based prism mesh generation. It is shown to be very suitable for today's EM modeling and simulation on multilayered PCB and electrical packaging world. Serpentine line (Figure 1: Coupled Serpentine line) and Via model (Figure 2: Coupled Via) examples are provided to illustrate the structures that are modeled.

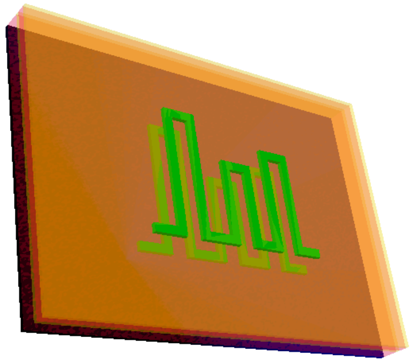


Figure 1: Coupled Serpentine line
(courtesy of Cadence Design Systems)

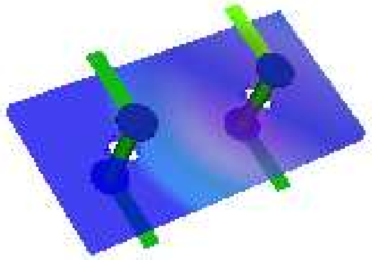


Figure 2: Coupled Via
(courtesy of Cadence Design Systems)

Approach

Due to the layered specialty of those interconnects on PCB and in packaging, on each layer, the vertical height of interconnect is thin and the shape is irregular, which is very suitable for triangular meshing. Along the vertical direction in the layer stack, the triangles can be extruded up or down to build prisms as volume elements which satisfy the FEM's needs. Due to this special property, a triangular prism is the basic element for mesh generation [3].

Our prism meshing algorithm is as follows:
1) Compress 3D model structures into the 2D *x-y* plane.

2) Apply CGAL's [4] 2D constrained Delaunay triangulation using criteria to control edge length and the angle of the mesh triangle elements.
3) Extrude all the 2D meshed triangles vertically back to the original 3D structure. Currently, the thinnest height among serpentine lines and layers is used to subdivide each existing layer.
4) A triangular prism meshing output file is generated. The number of vertices and prism elements are shown in the file.
5) Mesh generation results are displayed visually in 3D using the visualization toolkit VTK.

This algorithm has worst-case $O(n\log n)$ time complexity, where n is the number of vertices.

Experimental Results

The implementation for 3D triangular prism mesh generation and visualization utilizes several existing libraries: Nokia Qt software, CGAL with boost library and VTK. Figure 3 and Figure 4 show coupled Serpentine line and coupled Via meshing results.

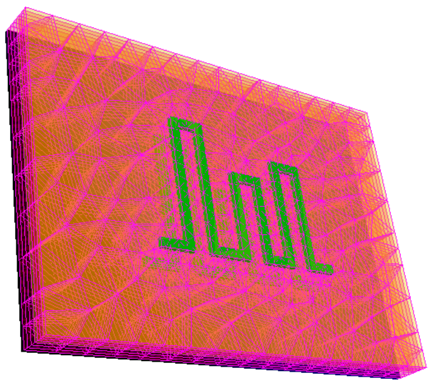


Figure 3: Meshing Results for Coupled Serpentine line
Number of 2D Triangles: 792
Number of 3D Triangular Prisms: 5180
(courtesy of Cadence Design Systems)

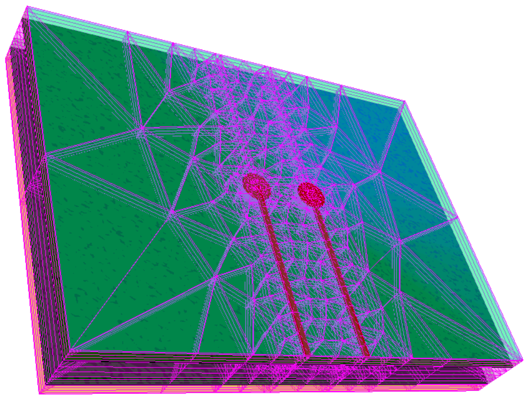


Figure 4: Meshing Results for Coupled Via

Number of 2D Triangles: 944
Number of 3D Triangular Prisms: 17404
(courtesy of Cadence Design Systems)

Future Work

In a PCB and packaging design, people usually see tens or even hundreds of Serpentine line and Via models over tens or hundreds of layers. So, parallel mesh generation [5] should be an efficient solution for this large problem. Our next step will focus on parallel constrained Delaunay mesh generation. We plan to decompose the original mesh generation domain into smaller subdomains which will be meshed in parallel. We may use a medial axis domain decomposition method to partition the original domain.

References

1. Tummala, R.R., *SOP: What Is It and Why? A New Microsystem-Integration Technology Paradigm-Moore's Law for System Integration of Miniaturized Covergent Systems of the New Decade.* IEEE Transactions on Advanced Packaging, 2004. **27**(2): p. 241-249.
2. Tsukerman, I., *A General Accuracy Criterion for Finite Element Approximation.* IEEE Transactions on Magnetics, 1998. **34**(5): p. 1-4.
3. Hwang, C.-T., et al., *Partially Prism-Gridded FDTD Analysis for Layered Structures of Transversely Curved Boundary.* IEEE Transactions of Microwave Theory and Techniques, 2000. **48**(3): p. 339-346.
4. *CGAL User and Reference Manual*. 2007.
5. Chrisochoides, Nikos, *A Survey of Parallel Mesh Generation Methods*, Technical Report BrownSC-2005-09, Brown University, (also in *Numerical Solution of Partial Differential Equations on Parallel Computers,* A.M. Bruaset, P. Bjorstad, A. Tveito Eds., Springer Verlag).

Matching Shapes Using The Current Distance[*]

Sarang Joshi sjoshi@sci.utah.edu
Raj Varma Kommaraju rajvarma@cs.utah.edu
Jeff M. Phillips jeffp@cs.utah.edu
Suresh Venkatasubramanian suresh@cs.utah.edu

1 Introduction

In this paper, we study shape matching under the *current distance*, a distance measure on shapes proposed by Vaillant and Glaunès [4]. This measure has four attractive properties. Firstly, it is global in nature, and thus does not require the determination of correspondences between features. Computing correspondences is often the most expensive part of computing distance between shapes. Secondly, although inspired by clever ideas from geometric measure theory, it can be expressed as a direct (but expensive) computation, and so is more tractable than many global shape distance measures (that typically require computation of geodesics on manifolds). Thirdly, it generalizes easily to higher dimensional structures: the current distance can be defined between pairs of point sets, curves, surfaces, and even higher-dimensional manifolds. Finally, it is defined in terms of a *norm* on the shape by the usual construction $d(S, S') = \|S - S'\|$. This norm acts as a *signature* of the shape, and is potentially useful for building data structures to answer more generalized queries about shapes (like near neighbors, clustering, etc).

We present the first algorithmic analysis of the current distance. Our main contributions in this work are (1) a fast approximation for computing the current distance that runs in near-linear time (as opposed to the quadratic bound implicit in the definition) (2) A coreset-like construction for approximating the *current norm* of a point set by a small-sized sample and (3) an FPTAS for minimizing the current distance between two point sets under translations.

2 Definitions

Let P be a set of points in $\mathbb{R}^d$. Let $K : P \times P \to \mathbb{R}$ be a *reproducing kernel* [1] i.e the matrix $M = (m_{ij} = K(p_i, p_j))$ is positive definite.

Definition 2.1. *The* current norm*[4] of a point set P is given by*

$$\|P\|^2 = \frac{1}{n^2} \sum_i \sum_j K(p_i, p_j)$$

In general, a point p may have a real-valued weight $w(p)$ associated with it. Setting $w_i = w(p_i)$, we can rewrite the current norm as $\|P\|^2 = \frac{1}{n^2} \sum_i \sum_j w_i w_j K(p_i, p_j)$. We can now associate the set sP with the set P with weights $sw(p)$. Further, we define the operator $P + Q$ as the set $P \cup Q$.

Definition 2.2. *The* current distance *between two point sets P and Q of size n and m respectively is*

$$D^2(P, Q) = \|P + (-1)Q\|^2$$
$$= \|P\|^2 + \|Q\|^2 - \frac{2}{mn} \sum_i \sum_j K(p_i, q_j)$$

3 Efficient Approximation Of Current Distance

A reproducing kernel induces a map into a Hilbert space $\mathcal{H}$: specifically, for a reproducing kernel K, there exists a map $\phi : X \to \mathcal{H}$ such that $K(x, y) = \langle \phi(x), \phi(y) \rangle$. Let $c^*(P) = \frac{1}{n} \sum_{p \in P} \phi(p)$ be the *dual centroid* of P and let $\| \cdot \|$ be the induced norm in $\mathcal{H}$.

Lemma 3.1. $D^2(P, Q) = \|c^*_P - c^*_Q\|^2$

The importance of this lemma is that it replaces a sum of quadratic terms by a computation over a linear number of terms. $\mathcal{H}$ might in general be infinite-dimensional, but we can *approximate* ϕ using a mapping $\tilde{\phi} : X \to \mathbb{R}^k$. Such a mapping can be obtained by using the fast Gauss transform and variants [5, 2] (for the Gaussian kernel) or by using a randomized method [3] for shift-invariant kernels (kernels where $K(x, y)$ can be written as $k(x - y)$). We summarize:

Theorem 3.1. *For fixed $\varepsilon > 0$ and $r = f(\varepsilon)$, we can compute a quantity $\tilde{D}(P, Q)$ in $O(nr)$ time such that $|D(P, Q) - \tilde{D}(P, Q)| \le \varepsilon$.*

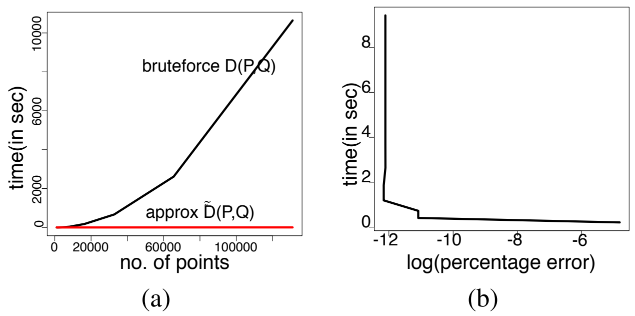


Figure 1: (a) Time performance comparison of exact ($D(P, Q)$) and approximate ($\tilde{D}(P, Q)$) distance measure. (b) Percentage error for approximate distance measure on 131072 points: $|D(P, Q) - \tilde{D}(P, Q)|/D(P, Q)$.

[*]This research was supported in part by a University of Utah Seed Grant, a subaward to the University of Utah under NSF award 0937060 to the Computing Research Association, and NIH Grant R01EB007688-01A1.

4 A Core-set For The Current Norm

Given a point set $P \subset \mathbb{R}^d$, we show that a subset $S \subset P$ can serve as a coreset of P with respect to the current norm. However, finding the smallest such set S is NP-hard, as a reduction from the partition problem.

Theorem 4.1. *Given a set $P \in \mathbb{R}$ of size n and parameters ε, k, it is NP-hard to determine if there exists a subset S of size k such that $||S - P|| \leq \varepsilon$.*

Theorem 4.2. *For a point set $P \subset \mathbb{R}^d$, a mapping $\tilde{\phi} : P \to \mathbb{R}^{\tilde{d}}$ and any kernel K such that $K(x,y) \in [0,c]$, a random sample $S \subset P$ of size $O((\tilde{d}c^2/\varepsilon)\log(\tilde{d}/\delta))$ ensures that $D(P,S) < \varepsilon$ with probability at least $1-\delta$.*

Proof. By Lemma 3.1, we just need to bound the probability that $||c_P^* - E[c_S^*]||^2 \geq \varepsilon$. This is always true if for each dimension $m \in [1, \tilde{d}]$ we have $|c_{P,m}^* - E[c_{S,m}^*]| \leq \sqrt{\varepsilon/\tilde{d}}$, so we can reduce to a 1-dimensional problem.

We can now invoke the following Chernoff-Hoeffding bound. Given a set $\{X_1, \ldots, X_n\}$ of independent random variables, such that $|X_i - E[X_i]| \leq \Delta$, then for $M = \frac{1}{n}\sum_{i=1}^n X_i$ we can bound $\mathbf{Pr}[|M - E[M]| \geq \alpha] \leq 2e^{-2n\alpha^2/\Delta^2}$.

By setting $\alpha = \sqrt{\varepsilon/\tilde{d}}$, we can then state that

$$\begin{aligned}
&\mathbf{Pr}[||c_P^* - E[c_S^*]||^2 \geq \varepsilon] \\
&\leq \tilde{d}\,\mathbf{Pr}\left[|c_{P,m}^* - E[c_{S,m}^*]| \geq \sqrt{\varepsilon/\tilde{d}}\right] \\
&\leq \tilde{d}2e^{-2k\varepsilon/(\tilde{d}\Delta^2)} \leq \tilde{d}2e^{-k\varepsilon/(2\tilde{d}c^2)}
\end{aligned}$$

where the last step follows because $\Delta = \max_{p,q\in P} ||\tilde{\phi}(p) - \tilde{\phi}(q)|| \leq 2c$ since $\max_{p\in P} ||\tilde{\phi}(p)|| \leq c$. The proof follows by setting $\delta \geq \tilde{d}2e^{-k\varepsilon/(2\tilde{d}c^2)}$ and solving for k. □

5 An FPTAS for Minimizing The Current Distance Under Translation

A translation $T \in \mathbb{R}^d$ is a vector so for point set $Q \subset \mathbb{R}^d$, we have $Q \circ T = \{q + T \mid q \in Q\}$. The translation $T^* = \arg\min_{T\in\mathbb{R}^d} D^2(P, Q\circ T)$, applied to Q minimizes the current norm. We describe, for any parameter $\varepsilon > 0$, an algorithm for a translation $\hat{T}$ such that $D^2(P, Q\circ\hat{T}) - D^2(P, Q\circ T^*) \leq \varepsilon$.

In minimizing $D^2(P, Q\circ T)$ under all translations, we can reduce this to solving for

$$T^* = \arg\max_{T\in\mathbb{R}^d} \sum_{p\in P}\sum_{q\in Q} K(p, q+T),$$

since the first two terms of $D^2(P, Q\circ T)$ have no dependence on T. For convenience let $\kappa(P,Q) = \sum_{p\in P}\sum_{q\in Q} K(p,q)$.

Lemma 5.1. $D^2(P, Q\circ T^*) \geq 1$.

Proof. If $T \in \mathbb{R}^d$ aligns $q \in Q$ so $q + T = p$ for $p \in P$ ensures $K(p,q) = 1$. All other subterms in $\kappa(P, Q\circ T)$ are at least 0. Thus $\kappa(P, Q\circ T) \geq 1$. □

When $K(p,q) = e^{-||p-q||^2}$ then if $||p-q|| < \sqrt{\ln(1/\gamma)}$, it implies $K(p,q) < \gamma$. Thus, if $\kappa(P, Q\circ T^*) \geq 1$, then some pair of points $p \in P$ and $q \in Q$ we must have $K(p, q+T^*) \geq 1/n^2$. Otherwise, if all n^2 subterms $K(p, q+T^*) < 1/n^2$, then $\kappa(P, Q\circ T^*) < 1$. Thus some pair $p \in P$ and $q \in Q$ must satisfy $||p - (q+T^*)|| \leq \sqrt{\ln(n^2)}$.

Lemma 5.2. *Let $K(p,q) = e^{-||p-q||^2}$. For any $\delta \in \mathbb{R}^d$ where $||\delta|| < \varepsilon$, then for any points $p, q \in \mathbb{R}^d$ we have $|K(p,q) - K(p, q+\delta)| \leq \varepsilon$.*

Proof. The slope for $\phi(x) = e^{-x^2}$ is maximized where $x = 1/2$ as $\phi(1/2) = e^{-(1/2)^2} = e^{-1/4} < 1$. Thus $|\phi(x) - \phi(x+\varepsilon)| < \varepsilon$. And since translating by δ changes $||p-q||$ by at most ε, the lemma holds. □

Let G_ε be a grid on $\mathbb{R}^d$ so that when any points $p \in \mathbb{R}^d$ is snapped to the nearest grid point $p' \in G_\varepsilon$, we guarantee that $||p - p'|| \leq \varepsilon$. Let $\mathcal{G}[\varepsilon, p, \Delta]$ represent the subset of the grid G_ε that is within a distance Δ of the point p.

These results imply the following algorithm. For each point $p \in P$, for each $q \in Q$, and for each $g \in \mathcal{G}[\varepsilon/2, p, \sqrt{\ln(n^2)}]$ we consider the translation $T_{p,q,g}$ such that $q + T_{p,q,g} = g$. We return the translation $T_{p,q,g}$ which maximizes $\kappa(P, Q\circ T_{p,q,g})$, by evaluating κ at each such translation of Q.

Theorem 5.1. *The above algorithm runs in time $O((1/\varepsilon)^d n^4 \log^{d/2} n)$, for fixed d, and is guaranteed to find a translation $\hat{T}$ such that $D^2(P, Q\circ\hat{T}) - D^2(P, Q\circ T^*) \leq \varepsilon$.*

References

[1] ARONSZAJN, N. Theory of reproducing kernels. *Transactions of the American Mathematical Society 68* (1950), 337 – 404.

[2] GREENGARD, L., AND STRAIN, J. The fast gauss transform. *SIAM Journal on Scientific and Statistical Computing 12* (1991).

[3] RAHIMI, A., AND RECHT, B. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems,* (2007), MIT Press.

[4] VAILLANT, M., AND GLAUNÈS, J. Surface matching via currents. In *Proc. Information processing in medical imaging* (January 2005), vol. 19, pp. 381–92.

[5] YANG, C., DURAISWAMI, R., GUMEROV, N., AND DAVIS, L. Improved fast gauss transform and efficient kernel density estimation. In *Proceedings Ninth IEEE International Conference on Computer Vision* (2003), IEEE, pp. 664–671 vol.1.

Segmented Height Field and Smoothed Particle Hydrodynamics in Erosion Simulation

Christopher S. Stuetzle*, Zhongxian Chen*, Katrina Perez‡, Jared Gross‡, Barbara Cutler*, W. Randolph Franklin†, and Thomas Zimmie‡
**Computer Science †Electrical, Computer, and Systems Engineering ‡Civil and Environmental Engineering*
Rensselaer Polytechnic Institute

1 Motivation

The New Orleans area levee failures during Hurricane Katrina drew media attention to an important problem in Civil Engineering. The emphasis of our work is on earthen levees, dams, and embankments. A major cause of failures of such structures is overtopping, which causes erosion to the point of breaching the crest. Our research focuses on simulating the initial small-scale features of erosion – the formation of rills and gullies on the embankment. We wish to study and eventually be able to simulate the way earthen embankments erode, with respect to the formation of these rills and gullies. Validation of computer simulations is the primary focus of our research. We will utilize RPI's geotechnical centrifuge to perform high-g erosion experiments on small-scale models to predict and validate the model for full scale simulations.

2 Review of Literature

Erosion Models and Erodibility A variety of existing erosion models calculate the overall soil loss during the overtopping of an earthen embankment. For example, Wang and Kahawita present a two-dimensional mathematical model of erosion of the profile of an earthen embankment during overtopping (Wang and Kahawita 2003). The "erodibility" of soil is generally defined as the ratio of the rate the soil erodes to the velocity of the water causing the erosion. In his work, Jean-Louis Briaud defines erodibility as a function of hydraulic shear stress, or pull of the water on the soil and presents measurements of soil samples collected from many of the affected earthen levees in the New Orleans area (Briaud, Chen, Govindasamy, and Storesund 2008).

Erosion Simulations Kristof et al., present an erosion simulation using smoothed particle hydrodynamics (SPH). The soil, water, and soil-water boundaries are represented by particles – the soil and water particles have mass and velocity while the boundary particles are designed solely for the two phases to interact (Kristof, Benes, Krivanek, and St'ava 2009).

Data Structures Three data structures dominate erosion research in computer graphics: the height field (or height map), the layered height field, and the voxel grid (see Figure 2). Height fields are a regular 2D grid array of heights, where each grid space allows for a single height value. Height fields, like the one used in (Benes and Arriaga 2005), are easily compressible, and allow for fairly simple surface extraction. However, they do not allow for layered terrains.

A voxel based terrain representation allows for immediate coupling with many fluid simulation techniques. Voxel-based terrain representations allow for multiple soil layers, different soil parameters, caves and undercuts, and inherent volume information. Tetrahedral meshes are another volumetric representation that builds the model out of tetrahedra. This representation uses the points along the surface of the volume, and thus is more accurate in representing the model's surface than a voxel grid, which is limited to the grid resolution in three dimensions.

Benes et al. present a layered height field that combined several advantages of each of height fields and voxel grids (Benes and Forsbach 2001). The terrain is divided into a two dimensional grid, like a height field, but each grid space contains an array of heights. This representation allows for several different soil types, a surface can be easily extracted for visualization and simulation purposes, the precision is arbitrary, and caves and undercuts are possible.

3 Segmented Height Field

Our new data structure is the Segmented Height Field (SHF), shown in Figures 1&2d. It is similar to the layered height field, described earlier, in that it is a grid of cells, and each cell contains a list of the layers of soil found in that column. In our structure, each cell (referred to as a **column**) contains a list of soil **segments** (a rectilinear block of soil). Segments of the same soil type in adjacent grid columns that have overlapping intervals in height create a **layer**.

Several important differences exist between the layered height field and SHF. SHF allows for layers to be dynamically added and removed, making accurate re-deposition easier. Second, a layered height field does not allow for two layers of the same soil

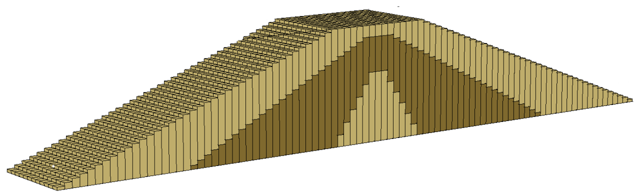


Figure 1: Earthen embankment with multiple soils.

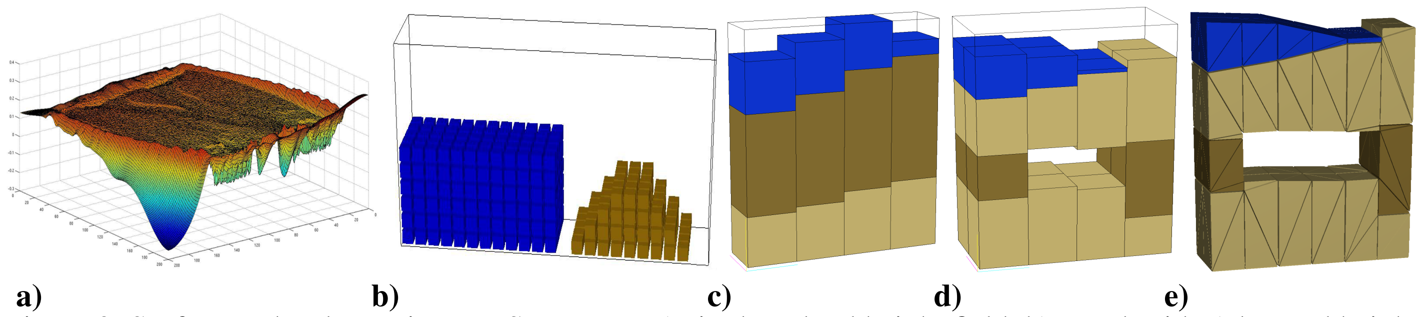


Figure 2: Surface and Volumetric Data Structures: a) single-valued height field, b) voxel grid, c) layered height field d) our new Segmented Height Field (SHF), e) interpolated tetrahedral mesh based on the SHF.

type to exist in one list in one grid cell, whereas in our representation these layers are added automatically. Because there are no restrictions with regards to the location of a soil segment within the column SHF naturally supports overhangs and air pockets.

Furthermore, each SHF segment can have spatially varying soil parameters; for example, moisture content. Yet our SHF retains a key advantage of height fields: the layer height is not limited in resolution as it is for a voxel grid. Finally, the SHF can be more memory efficient than either a voxel grid or a layered height field. Voxel grids store redundant data when representing layers that are thick relative to the underlying grid. And layered height fields are inefficient for accurate representation of complex soil re-deposition, because the addition of new layers is global rather than local.

4 Tetrahedralization

For visualization and simulation purposes, it is necessary to be able to generate an interpolated volume from our SHF. To do this, we construct, from the segmented height field, a tetrahedral mesh that explicitly captures the slope and surface normals of the boundary surfaces between air and soil and between different soil types. These not only improve the quality of the resulting visualization, but also will yield more accurate physical simulations by allowing water to flow smoothly down the inclined slope of the embankment and through channels cut within the soil.

To convert the SHF to a tetrahedral mesh, we define a set of rules to govern the placement of pivot points. The points are either placed at T-junctions in the data or interpolated along layer surfaces. The points are then connected in a manner that ensures the formation of a closed, water-tight tetrahedral mesh. The resulting volume has a smooth, water-tight surface on which erosion simulations may be run.

5 Smoothed Particle Hydrodynamics

The simulation of water and its erosion on the levee will be based on Smoothed Particle Hydrodynamics (SPH). In the SPH method, the state of a system is represented by a set of particles, which possess individual material properties and move according to some governing conservation equations. The most significant advantage of this mesh-free method lies in its adaptability and its ability of handling problems of extremely large deformation. Unlike the widely used voxel-based fluid simulation methods, the mess-free SPH method is better suited to adapt to changing soil-fluid boundary conditions. Therefore, we believe it is most suitable for simulating levee erosion. In the simulation, water particles and soil particles will first be handled independently based on SPH theories, and then coupled to simulate the interaction between water and soil.

6 Results and Future Work

To date, we have collected LIDAR scan data of rill and gulley formation on a small earthen embankment and loaded it into our new SHF data structure. In the future, we will be able to simulate water flowing over the embankment, using SPH, and eroding the soil away to form rills and gulleys. We will then perform high-g erosion experiments on RPI's centrifuge in order to statistically validate our simulation results. This research will lead to new software tools for civil engineers that help analyze and predict the performance of earthen embankments.

REFERENCES

Benes, B. and X. Arriaga (2005). Table mountains by virtual erosion. In *Eurographics Workshop on Natural Phenomena 2005*, pp. 33–40.

Benes, B. and R. Forsbach (2001, April). Layered data representation for visual simulation of terrain erosion. In *Proceedings of the 17th Spring conference on Computer graphics*.

Briaud, J. L., H.-C. Chen, A. V. Govindasamy, and R. Storesund (2008). Levee erosion by overtopping in new orleans during the katrina hurricane. *Journal of Geotechnical and Geoenvironmental Engineering 134*(618), 618–632.

Kristof, P., B. Benes, J. Krivanek, and O. St'ava (2009). Hydraulic erosion using smoothed particle hydrodynamics. *Computer Graphics Forum 28*(2), 219–228.

Wang, P. and R. Kahawita (2003). Modeling the hydraulics and erosion process in breach formation due to overtopping. In *Sedimentation and Sediment Transport, Proceedings*, pp. 211–220.

Orthogonal Morphing of Orthogonal Drawings

Sergey Bereg*

1 Introduction

A *morph* between two drawings of a graph is a continuous transformation of one drawing into the other. We consider *intersection-free* [3] or *planarity-preserving* [1, 4] morphs that preserve non-crossing edges. Morphing of graph drawings was addressed a long time ago by Cairns [2] who proved that there is a planarity-preserving morph between two drawings of a planar triangulation such that they have the same boundaty triangle.

Two straight-line drawings P and Q of a graph $G = (V, E)$ are *parallel* if, for every edge $(u, v) \in E$, the vector from u to v has the same direction in both P and Q. For a planar graph G and simple straight-line drawings P, Q of G, a morph of P into Q is called *parallel* if it preserves planarity and edge directions.

Biedl *et al.* [1] studied parallel morphing of orthogonal drawings of a planar graph. A morph between two drawings of a graph is *linear* if every vertex move along a straight line from its position in the source to its position in the target. Note that a linear morph may not preserve simplicity even for morphing simple orthogonal polygons. Biedl *et al.* [1] proved that a parallel morph between two parallel orthogonal drawings of a planar graph can be found in polynomial time; however, when edges are allowed to be in one of three or more slopes the problem becomes NP-hard.

Theorem 1 (Biedl, Lubiw and Spriggs [1]) *Any two simple parallel orthogonal drawings P, Q of a connected graph (V, E) admit a parallel morph that is composed of $O(|V|)$ linear morphs.*

*Dept. of Computer Science, University of Texas at Dallas, Box 830688, Richardson, TX 75083, USA.

We call a morph of a graph drawing *orthogonal* if either the x-coordinates or the y-coordinates of its vertices do not change during the morph. If a morph changes only x-coordinates (resp. only y-coordinates) of vertices, then we call it an x-morph (resp. a y-morph). Obviously, every orthogonal morph is a linear morph. It turns out that all linear morphs except perhaps one (which can be realized as two orthogonal morphs) in Theorem 1 are orthogonal. Theorem 1 uses the *zig-zag elimination*, see Fig. 1.

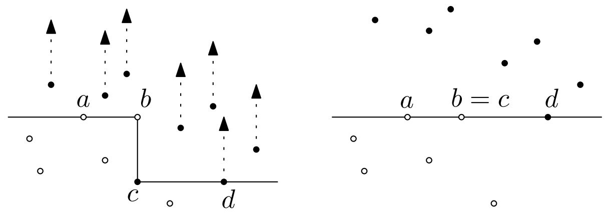


Figure 1: Zig-zag elimination by a linear morph.

In some examples, P can be transformed into Q with only $o(|V|)$ orthogonal morphs. Figure 2 shows P and Q that can be morphed using one x-morph.

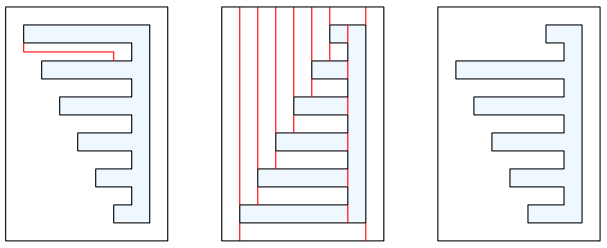


Figure 2: (a) Drawing P. (b) Drawing Q with rectangilar faces. (c) Straightening the path from (a).

In this paper we address the problem of finding a smallest sequence of orthogonal morphs transforming P into Q.

2 Straigthening Paths by Orthogonal Morphs

We generalize zig-zag morphs to morphs straigthening monotone paths, see Fig.3.

Labeling. We label the vertices of a path with L (left turn) and R (right turn). A path is *balanced* ([1]) if the number of vertices labeled L is the same as the number of vertices labeled R. We also label the horizontal segments of a path by X and the vertical segments by Y.

Path contraction. We define an operation on orthogonal paths called *contraction* corresponding to straigthening monotone paths (details in full version). The *length* $l(\phi)$ *of a path* ϕ is defined as the number of its segments. We denote by ϕ_C the path obtained by applying a contraction C to ϕ. An x-contraction (resp. y-contraction) C of a path ϕ is called *maximal* if no non-trivial x-contraction (resp. y-contraction) can be applied to ϕ_C.

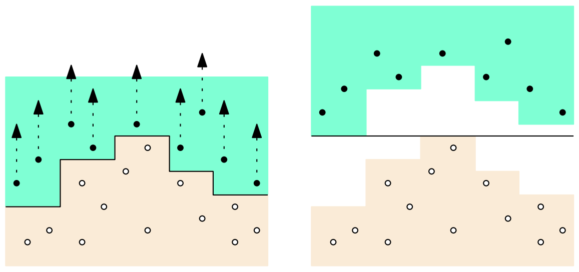


Figure 3: Straightening an x-monotone path.

Lemma 2 *Any contraction of a balanced path* ϕ *can be realized by an orthogonal morph. Specifically, any* x*-contraction (*y*-contraction) of* ϕ *can be realized by an* y*-morph (resp.* x*-morph).*

Contractions can be used to reduce a balanced path to a single segment. We define an x*-length* (resp. y*-length*) of a balanced path as the smallest number of contractions such that the first contraction is horizontal (resp. vertical).

Lemma 3 x*-length and* y*-length of any balanced path can be computed using maximal contractions.*

Theorem 4 *A minimum number of orthogonal morphs transforming* P *into* Q *can be found in polynomial time.*

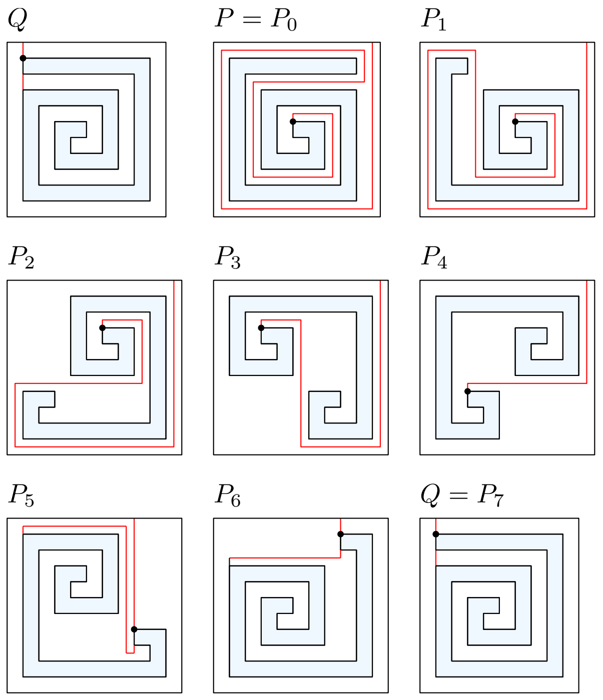


Figure 4: Example of morphing. The red path in P_0 corresponds to the upper vertical segment in Q. It is straightened in P_5. The lower vertical segment in Q is morphed in P_6, P_7.

References

[1] T. Biedl, A. Lubiw, and M. J. Spriggs. Morphing planar graphs while preserving edge directions. In *Proc. 13th Internat. Sympos. on Graph Drawing*, LNCS 3843, pp. 13–24. Springer-Verlag, 2005.

[2] S. S. Cairns. Deformations of plane rectilinear complexes. *American Math. Monthly*, 51:247–252, 1944.

[3] C. Erten, S. G. Kobourov, and C. Pitta. Intersection-free morphing of planar graphs. In *Proc. 11th Internat. Sympos. on Graph Drawing*, LNCS 2912, pp. 320–331. Springer-Verlag, 2003.

[4] A. Lubiw, M. Petrick, and M. J. Spriggs. Morphing orthogonal planar graph drawings. In *Proc. 17th ACM-SIAM Sympos. Discrete Algorithms*, pp. 222–230, 2006.

A Promising Approach to Pseudo-triangulations and Their 3D Generalization

Field Cady, Seth Goldstein, Gary Miller

October 20, 2009

Abstract

We discuss the problem of generalizing pointed pseudo-triangulations to higher dimensions. We present a brief overview of pointed pseudo-triangulations, their importance in rigidity theory, and their expansive motions, focusing particularly on ways to create pointed pseduo-triangulations using Henneberg constructions. Henneberg constructions are interesting because, unlike pseudo-triangles, they lend themselves easily to higher dimensional generalizations. We suggest that rather than generalizing pseudo-triangles, it may be more fruitful to generalize the methods of their construction.

We outline our ideas and present several hypotheses which generalize generalize classical results in the plane to higher dimensions.

1 Background

Pseudo-triangulations have received much attention in recent years, particularly in studying rigidity and expasive motions in the plane. Pointed pseudo-triangulations of convex regions are minimally rigid. If the region has a single concave vertex, the pseudo-triangulation is a framework with a single expansive degree of freedom. The biggest open problem in pseudo-triangulations is widely considered to be generalizing them to higher dimensions.

Laman graphs are generically minimally rigid graphs; it is a standard result that any Laman graph can be embedded as a pointed pseudo-triangulation. Laman graphs can be built up from a single edge by means of Henneberg constructions, shown in Figure 1. A Henneberg addition is a way to add a new vertex to a Laman graph and produce another Laman graph. Type-I Henneberg additions simply add a "hinge" to the framework. Type-II additions replace an existing edge with a hinge, but add another edge which re-establishes rigidity. All planar Laman graphs can be constructed recursively by Henneberg additions starting from a single edge.

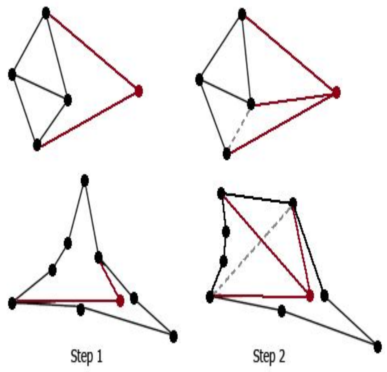


Figure 1: Henneberg constructions add a new vertex while keeping a graph Laman. For pointed pseudo-triangles, a type-II construction becomes a type-I construction plus an edge flip. Figure taken from "Henneberg construction" by Andrei Haralevich.

2 Solutions in the Plane

We define a special type of Henneberg addition for planar graphs, which we call a *Henneberg** addition, which turns pointed pseudo-triangulations in other pointed pseudo-triangulations. For type-I, we require that it maintain pointedness of all vertices and that the arc enclosed by the added hinge be convex. For type-II, we require that it be equivalent to a type-I addition followed by an adjacent edge flip.

The following lemmas are straightforward corollaries of the results discussed in the previous section:

Lemma 2.1. *Let F be a pointed pseudo-triangulation of a region R. If R is convex, then F can be built up by Henneberg* additions starting from an edge. If R has exactly one concave vertex, F can be constructed by Henneberg* additions starting from a hinge.*

Lemma 2.2. *Let F be a pointed pseudo-triangulation. If F is rigid, then a Henneberg* addition to a F maintains minimaly rigidity. If F has a single expansive degree of freedom, a*

Henneberg* addition to F maintains expansiveness.

Lemma 2.2 is fairly intuitive, at least for type-I additions. Combined with lemma 2.1, it provides an alternate way to prove the classical results about pointed pseudo-triangulations regarding expansiveness and minimal rigidity: a triangle is minimally rigid and a hinge expansive, and as we build these into any pointed pseudo-triangulation by *Henneberg** addition, these properties are preserved. We ask whether this approach can be generalized to higher dimensions by using the higher dimensional analogs of *Henneberg** additions.

As an aside, while in general type-I and type-II additions are both needed to construct a given pointed pseudo-triangulation, type-I additions are still quite general, as shown in the following lemma:

Lemma 2.3. *Every pointset in general position has a pointed pseudo-triangulation constructed using only type-I Henneberg* additions.*

The lemma is desirable because Lemma 2.2 is intuitively straighforward and simpler to prove for type-I additions.

3 Problems in 3D

We let *Henneberg*$^+$ additions be the 3D generalization of *Henneberg** additions. The topology of a *Henneberg*$^+$ addition is clear; type-I adds a "tripod" to a framework, while a type-II addition replaces one edge with a hinge, and adds 2 additional edges. Consideration of the degrees of freedom shows these will make *Henneberg*$^+$ additions preserve rigidity. The generalization of pointedness is also straightforward.

What is less obvious is how the convexity of the enclosed arc generalizes; this property is critical for proving expansiveness. The best approach to take is unclear, but we are currently examining the following approach. In 2 dimensions, one can choose the *Henneberg** construction such that no point or edge is ever placed where it can see the inside of the starting hinge. A 3D expansive framework can start with 2 triangles sharing a side, forming an "origami hinge". If no vertex or edge is ever added that can see the inside of this hinge, that may be sufficient.

Our central hypothesis is that *Henneberg*$^+$ additions, properly defined, to an expansive framework preserve expansiveness. his lemma would characterize a broad class of rigid and expansive frameworks in 3D.

4 Concluding Remarks

Our work suggests a new approach to the difficult problem of expansiveness and rigidity in higher dimensions. Perhaps our most novel contribution is the idea that it is constructive processes which generalize, rather than geometric shapes.

Potential applications for these ideas include robot motion planning and understanding physical structures in 3 dimensions.

The work is also relevant to distributed systems (that is actually what led us to the topic), in that a large ensemble of vertices and edges, interacting only with their neighbors, give rise to global properties. In the plane, this is most clear in frameworks made only from type-I additions; there is a natural partial ordering of edge pairs, where the motion (if any) of a pair is determined only by the motion of those less than it. This decoupling of motions in the framework simplifies calculating the actual motion.

References

[1] Rote, G., Santos, F., Streinu, I. Pseudo-triangulationsa survey. In: Goodman, J.E., Pach, J., Pollack,R. (eds.) Surveys on Discrete and Computational Geometry: Twenty Years Later. Contemporary Mathematics, vol. 453, pp. 343410. Am. Math. Soc., Providence (2008)

[2] I. Streinu, "A combinatorial approach to planar non-colliding robot arm motion planning," focs, pp.443, 41st Annual Symposium on Foundations of Computer Science, 2000

[3] Haas et. al., "Planar Minimally Rigid Graphs and Pseudo-triangulations',' Proceedings of the 19th Annual Symposium on Computational Geometry, 2003, Pages 154 - 163

[4] Ordena, D., Santos, F., BServatiusc, B., and Servatius, H., "Combinatorial Pseudo-triangulations," Discrete Mathematics Volume 307, Issues 3-5, 6 February 2007, Pages 554-566

[5] Jack E. Graver, Brigitte Servatius, Herman Servatius, "Combinatorial Rigidity," Graduate Studies in Mathematics 1993

CRA-W/CDC Workshop on Computational Geometry (11/15)

08:30	Gathering and registration
09:00	*Joseph O'Rourke* (Smith College) **Curve Shortening, Geodesics, and the Poincaré Conjecture**
09:45	*Godfried Toussaint* (McGill Univ. and Harvard) **The Geometry of Musical Rhythm**
10:30	Break
10:45	*Anna Lubiw* (Univ. of Waterloo and MIT) **Finding Shortest Paths**
11:30	*Jack Snoeyink* (UNC Chapel Hill) **Exploring the Geometry of Molecular Interactions**
12:15	Lunch break
13:15	*Gill Barequet* (Technion and Tufts University) **Half-a-Century of Counting Polycubes and Estimating their Growth Rate**
14:00	*Nancy M. Amato* (Texas A&M University) **Randomized Motion Planning:** **From Intelligent CAD to Group Behaviors to Protein Folding**
14:45	Break
15:00	*Roberto Tamassia* (Brown University) **Graph Drawing and Information Visualization**
15:45	*Matthew T. Dickerson* (Middlebury College) **Understanding Voronoi Diagrams as Lower Envelopes**
16:30	Break
16:45	Open-problems session
17:30	Adjourn

Abstracts of the

CRA-W/CDC Workshop on Computational Geometry

Joseph O'Rourke **(Smith College)**

Curve Shortening, Geodesics, and the Poincaré Conjecture

A time series can be smoothed by averaging nearby samples to aggregate data and remove noise. This is a simple and obvious transformation of one curve to another. Despite its simplicity, when viewed as "curve shortening," it is connected to two deep and decidedly non-obvious theorems: the existence of simple, closed geodesics on a closed surface, and the recently settled Poincaré conjecture. Exploring this connection elucidates the rich interplay between continuous and discrete geometry.

Godfried Toussaint **(McGill Univ. and Harvard)**

The Geometry of Musical Rhythm

Many problems concerning the theory and technology of musical rhythm are fundamentally geometric in nature. Such problems include measuring the similarity and complexity of rhythms, as well as the automatic generation of 'good' rhythms. The interaction between computational geometry and music yields new insights into the theories of rhythm as well as new problems for research in several areas, ranging from computational geometry in computer science to music theory, music perception, and musicology. Recent results on the geometric and computational aspects of rhythm are reviewed, connections to established areas of computer science, mathematics, statistics, computational biology, and crystallography are pointed out, and some open problems are discussed. Rhythm is considered from the symbolical point of view, and no knowledge of music is expected of the audience.

Anna Lubiw **(Univ. of Waterloo and MIT)**

Finding Shortest Paths

We will begin by reviewing algorithms for some geometric shortest path problems, such as finding shortest paths among obstacles in the plane, and finding shortest paths on terrains like the Earth's surface (which, as any bicyclist knows, should not be modelled as a flat plane).

The rest of the talk will be on extensions and variations of these problems. We will look at some touring problems, where you want to visit certain sites with the shortest possible route. In general this is the Travelling Salesman Problem, which is hard, but there are more tractable versions of the problem with such colourful names as the Safari Problem, the Zoo-keeper Problem, and the Watch-man Route Problem.

We will also look at some problems of finding shortest paths on terrains when there are constraints about slope. For example, is there a route from A to B that allows you to coast downhill on your bicycle the whole way? And what if you're afraid of steep hills?

Jack Snoeyink **(UNC Chapel Hill)**

Exploring the Geometry of Molecular Interactions

The models used in structural molecular biology for protein folding and drug design are surprisingly geometric, and many of the questions that they ask, such as “how many unsatisfied hydrogen bond donors are found in the core of a proposed protein structure” are approached by developing geometric algorithms. This tutorial will introduce problems related to hydrogen bond geometries, and show how the exponentially-increasing number of known native structures, and the ability to modify structure prediction code to make new “decoy” structures gives a path for computational experiments and validation.

Gill Barequet **(Technion and Tufts University)**

Half-a-Century of Counting Polycubes and Estimating their Growth Rate

A d-dimensional polycube of size n is a (d-1)-face-connected set of n cubes in d dimensions. Among many interesting questions related to polycubes, one can ask how we can count them efficiently (say, for specific values of n and d), and, for a fixed dimension d, what the asymptotic growth rate of d-dimensional polycubes is. In the past 50 years there have been two (sometimes parallel) research tracks of these questions: one in the mathematical literature and one in the statistical-physics literature. In this talk I will attempt to provide an overview of both tracks.

Nancy M. Amato **(Texas A&M University)**

Randomized Motion Planning: From Intelligent CAD to Group Behaviors to Protein Folding

Motion planning arises in many application domains such as computer animation (digital actors), mixed reality systems and intelligent CAD (virtual prototyping and training), and even computational biology and chemistry (protein folding and drug design). Surprisingly, a single class of planners, called probabilistic roadmap methods (PRMs), have proven effective on problems from all these domains. Strengths of PRMs, in addition to versatility, are simplicity and efficiency, even in high-dimensional configuration spaces.

In this talk, we describe the PRM framework and give an overview of several PRM variants developed in our group. We describe in more detail our work related to virtual prototyping, group behaviors, and protein folding. For virtual prototyping, we show that in some cases a hybrid system incorporating both an automatic planner and haptic user input leads to superior results. For group behaviors, we describe new PRM-based techniques for herding, pursuit/evasion and evacuation planning. Finally, we describe our application of PRMs to simulate molecular motions, such as protein and RNA folding. More information regarding our work, including movies, can be found at http://parasol.tamu.edu/~amato/

***Roberto Tamassia* (Brown University)**

Graph Drawing and Information Visualization

Information visualization is increasingly used in tools for analyzing data in business, engineering, and scientific applications. Constructing effective visual representations of graphs and related combinatorial structures presents significant algorithmic and user interface challenges. In this talk, we survey fundamental techniques for drawing general graphs as well as specialized methods for drawing trees, planar graphs, and directed graphs. We discuss combinatorial and geometric concepts related to graph embedding and address computational issues associated with layout optimization. We also present applications of graph drawing methods to the visualization of security and privacy in computer systems and networks.

***Matthew T. Dickerson* (Middlebury College)**

Understanding Voronoi Diagrams as Lower Envelopes

Voronoi Diagrams are one of the most important objects of study in the field of computational geometry. They are a surprisingly simple geometric concept, and yet they let appear in a variety of different and diverse applications and contexts including: astronomy and physics, chemistry and the study of crystals, modeling rainfall, and archaeology. They also can be extended in countless different interesting and non-trivial ways. This presentation will explore a geometric understanding of 2D Voronoi diagrams by looking at collections of 3D surfaces that defined distances.