11th Europea
Computatio
CG

March 23
Hagenberg/

Organized by: RISC, Johannes Kepler U
Supported by: Land OÖ, Stadt Linz, Lin

n Workshop on
nal Geometry
G'95

3-24, 1995,
Linz, Austria

Univ., Linz, Austria
zer Hochschulfonds.

List of presentations CG'95 Hagenberg, AUSTRIA

Thursday, March 23, 1995

9:00 - 9:10	Opening (Prof. DDr. B. Buchberger, Charmain of RISC)
9:10 - 9:35	Searching for the Kernel of a Polygon - A Competitive Strategy (R. KLEIN, Ch. ICKING, Fernuniv. Hagen)
9:35 -10:00	The Graph of Triangulations of a Convex Polygon (F. HURTADO, M. NOY, Univ. Politecnica de Catalunya)
10:00 -10:25	Voronoi Diagrams for Infinite Sets Generated by Consecutive Stretches (M. ABELLANAS, Univ. Politenica de Madrid) (E. ROANES-LOZANO, Univ. Complutense de Madrid)
10:25 -10:50	Randomized Incremental Construction of Simple Voronoi Diagrams in 3-Space (Le NGOC-MINH, Fernuniv. Hagen)
10:50 -11:15	Break
11:15 -11:40	Three Dimensional Delaunay Triangulation Using a Uniform Grid (L. CUCU, M. DRAGAN, D. MANCU, V. NEGRU, Univ. Temesware Romania) VAN GEEP
11:40 -12:05	Parallel Implementation of 3D Delaunay Triangulation (L. CUCU, M. DRAGAN, V. NEGRU, Univ. Temesware Romania, T. JEBELEAN, RISC, Uni. Linz) VERBARG
12:05 -12:30	Voronoi Diagrams for Spheres and their Application in Motion Planning (M. WOLFRATH Univ. Würzburg)
12:30 -14:00	Lunch
14:00 -14:25	Overlaying Simply Connected Planar Subdivisions in Linear Time (K. HINRICHS, Westf. Wilhelms-Univ. Münster)
14:25 -14:50	Local Properties of Triangulations (O. AICHHOLZER, F. AURENHAMMER, Techn. Univ. Graz)
14:50 -15:15	Issues in Parametrization of Algebraic Curves (F. WINKLER, RISC, UNiv. Linz) concelled
15:15 -15:40	Break

Spatzenbrauer Abfart 1830

Endstation Linie 3

15:40 -16:05 Parametric Linear and Quadratic Optimization by Elimination
(V. WEISPFENNING Univ. Passau)

16:05 -16:30 Intersection Algoritms: A Comperative Study
(E. ARDELEANU, RISC, Univ. Linz)

Friday, March 24, 1995

9:15 - 9:40 Let T be Triangle, is Isoscles?
(T.GROMEX-DIAZ, Univ. de Limoges)

9:40 -10:05 Numerical problems in the plane Roider method
(S. STIFTER, E. TOMUTA, RISC, Univ. Linz)

10:05 -10:30 Codifying the Topology of Graph Embeddings on a Surface
(F. SANTOS, Univ. Cantabria)

10:30 -10:55 Break

10:55 -11:20 Some Geometric Lower Bounds Using Connected Components
(W. STEIGER, Univ. Rutgers)

11:20 -11:45 Some Space/Time Tradeoffs for Ranking and Searching
(W. STEIGER, Univ. Rutgers)

11:45 -12:10 Approximate Center Points in Dense Point Sets
(K. VERBERG, Univ. Würzburg)

12:10 -12:35 Surface Reconstruction from Unorganized Points in Space
(R. MENCL, Univ. Dortmund)

12:35 -14:00 Lunch

14:00 -14:25 Simulation Program of the Path Tracking for any Irredundant
Planar Manipulatior and the 6R Manipulator+
(M.J. GONZALEZ-LOPEZ , Univ. Cantabria)

14:25 -14:50 Computation of a Non-Uniform Grid on the Configuration Space
of a Robot Arm Amidst Obstacles
(C. VAN GEEM, RISC, Univ. Linz)

14:50 -15:15 A Hybrid Symbolic -Numerical Method for Tracing
Plane Curves and Surface Intersections
(T. QUOC-NAM, RISC, Univ. Linz)

Searching for the Kernel of a Polygon — A Competitive Strategy

Christian Icking* Rolf Klein*

Abstract

We present a competitive strategy for walking into the kernel of an initially unknown star-shaped polygon. From an arbitrary start point, s, within the polygon, our strategy finds a path to the closest kernel point, k, whose length does not exceed 5.52 times the distance from s to k. This is complemented by a general lower bound of $\sqrt{2}$. Our analysis relies on a new result about an interesting class of curves which are *self-approaching* in the following sense. For any three consecutive points a, b, c on the curve the point b is closer to c than a to c.

Key-words. Competitive strategy, simple polygon, kernel, curves with increasing chords, self-approaching curves.

1 Introduction

Suppose that a mobile robot equipped with a 360 degree vision system "wakes up" in an unknown environment. Its task is to go to some location from which the whole environment is visible. This should be achieved as efficiently as possible.

In attacking this problem we are using the following model. The robot is a point, its environment is a simple polygon, P. Clearly, we have to assume that P is star-shaped, or no location with the required property would exist. At each point p, the robot is provided with the visibility polygon $\mathrm{vis}(p)$. The cost of the robot's motion is measured by the arc length of its path; we ignore the cost of planning the path, which will in fact turn out to be negligible.

If the robot knew a map of the polygon and its start position beforehand, it could employ one of the classic algorithms for computing the kernel, see [12, 1]. Then it could determine the kernel point, k, closest to its start position, s, and go straight from s to k; note that by definition each point of the kernel can see any other point of P, so that the line segment from k to s cannot be blocked. This would result in a perfect solution at cost $d(k, s)$, where d denotes the Euclidean distance.

But since the polygon P is not known to the robot a priori, its actions are bound to contain elements of try and error which cause extra cost. For example, in the situation depicted in Figure 1 a detour cannot be avoided.

In this paper we present a strategy that guarantees a path length bounded by 5.52 times $d(k, s)$. In general, a strategy for a problem class PC is called *competitive* with competitive factor C if each instance P of PC can be solved at a cost not exceeding C times the cost of a perfect solution of P with full information in advance.

Competitive strategies for autonomous robots have recently received considerable interest in computational geometry. For example, the problem of finding a goal in an unknown environment has been studied in [9, 11, 2, 7, 6, 13]. In [3, 4] the task of drawing a map is addressed. How to decide on the robot's current location if a map is available has been investigated in [5, 10]. A special case of the present problem, where P contains only one reflex vertex, has been optimally solved in [8].

Competitive strategies offer many advantages. They are superior to heuristic approaches because they come with a proven performance guarantee. Though, they can often be made as simple as heuristic rules, as the present paper demonstrates. Sometimes competitive strategies exist even for problems whose optimal solution would be NP-hard; see [4, 5]. However, analyzing a competitive strategy is often difficult such that the proven bound is considerably bigger than the real competitive factor.

This paper is organized as follows. First we briefly recall some definitions and state a simple fact on visibility polygons. Then we describe our strategy CAB (= Continuous Angular Bisector) for finding the closest kernel point. CAB is very easy to implement. We show that each path generated by CAB consists of $O(n)$ many pieces of ellipses or hyperbolae (including circular arcs and line segments as special cases), where n denotes the number of vertices of P. See Figure 2 for an example of a complete path generated by CAB. If the polygon P is not star-shaped, the robot notices this and ends its motion. Otherwise it will arrive at the closest kernel point, k. The robot's path from s to k has the following crucial property. As the robot proceeds, it will always get closer to all the future points on its path. We call such curves *self-approaching*.

We prove that for self-approaching curves there is an upper bound of $\sqrt{4+(2+\pi)^2} \approx 5.52$ for their *maxi-*

*FernUniversität Hagen, Praktische Informatik VI, Elberfelder Str. 95, 58089 Hagen, Germany

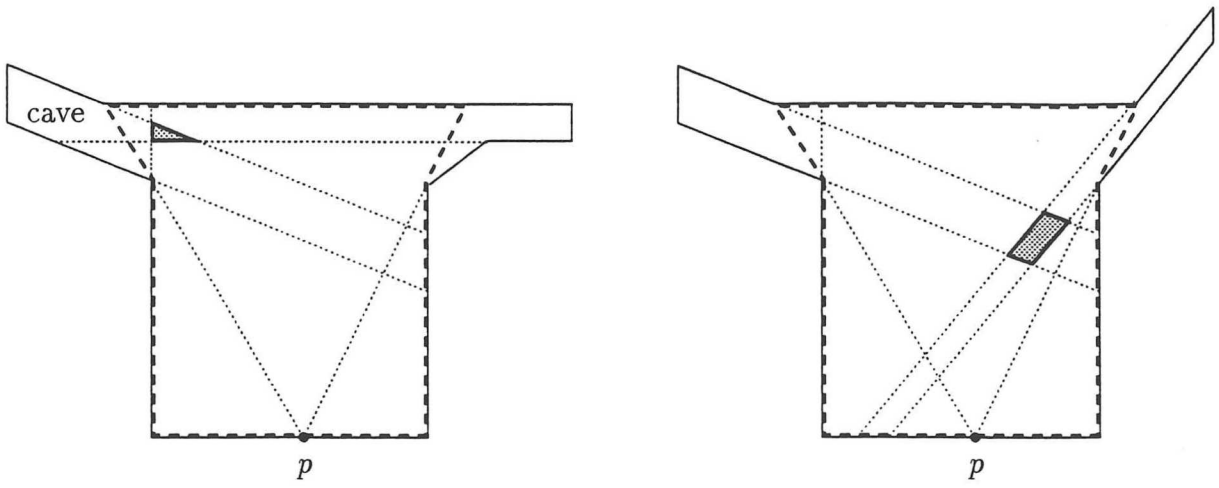


Figure 1: Identical visibility polygons but completely different kernels.

mum detour, i.e. the ratio of their arc length divided by the distance between their endpoints. Therefore, we have the same value as an upper bound for the competitive factor of strategy CAB. Curves that are self-approaching in *both* directions have been called *curves with increasing chords*. Recently, Rote [14] has solved an old open problem by proving a tight bound for this subclass of self-approaching curves. Unfortunately, it seems that his technique cannot be applied to our non-symmetric case. Our proof for the upper bound is based on the observation that the arc length of a self-approaching curve does not exceed the perimeter of its convex hull. This result is complemented by an example of a self-approaching curve whose ratio equals approximately 4.21.

Finally we prove that there is a lower bound of $\sqrt{2}$ for the competitive factor of *any* possible strategy for finding the kernel. For our particular strategy CAB, there is a situation where $\pi + 1 \approx 4.14$ is achieved (a case of only one reflex vertex). For this particular situation we have in [8] presented an optimal competitive strategy with factor 1.21218 which may be useful for further improvement of CAB.

2 A strategy for the kernel problem

References

[1] R. Cole and M. T. Goodrich. Optimal parallel algorithms for polygon and point-set problems. *Algorithmica*, 7:3–23, 1992.

[2] A. Datta and C. Icking. Competitive searching in a generalized street. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 175–182, 1994.

[3] X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment. In *Proc. 32nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 298–303, 1991.

[4] X. Deng, T. Kameda, and C. H. Papadimitriou. How to learn an unknown environment I: the rectilinear case. Technical Report CS-93-04, Department of Computer Science, York University, Canada, 1993.

[5] G. Dudek, K. Romanik, and S. Whitesides. Localizing a robot with minimum travel. In *Proc. 6th ACM-SIAM Sympos. Discrete Algorithms*, 1995.

[6] K. Ghosh and S. Saluja. Optimal on-line algorithms for walking with minimum number of turns in unknown streets. Technical Report TCS-94-2, Tata Institute of Fundamental Research, Bombay, 1994.

[7] C. Icking. *Motion and visibility in simple polygons.* PhD thesis, Department of Computer Science, FernUniversität Hagen, Germany, 1994.

[8] C. Icking, R. Klein, and L. Ma. How to look around a corner. In *Proc. 5th Canad. Conf. Comput. Geom.*, pages 443–448, Waterloo, Canada, 1993.

[9] R. Klein. Walking an unknown street with bounded detour. *Comput. Geom. Theory Appl.*, 1:325–351, 1992.

[10] J. M. Kleinberg. The localization problem for mobile robots. In *Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci.*, 1994.

[11] J. M. Kleinberg. On-line search in a simple polygon. In *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms*, pages 8–15, 1994.

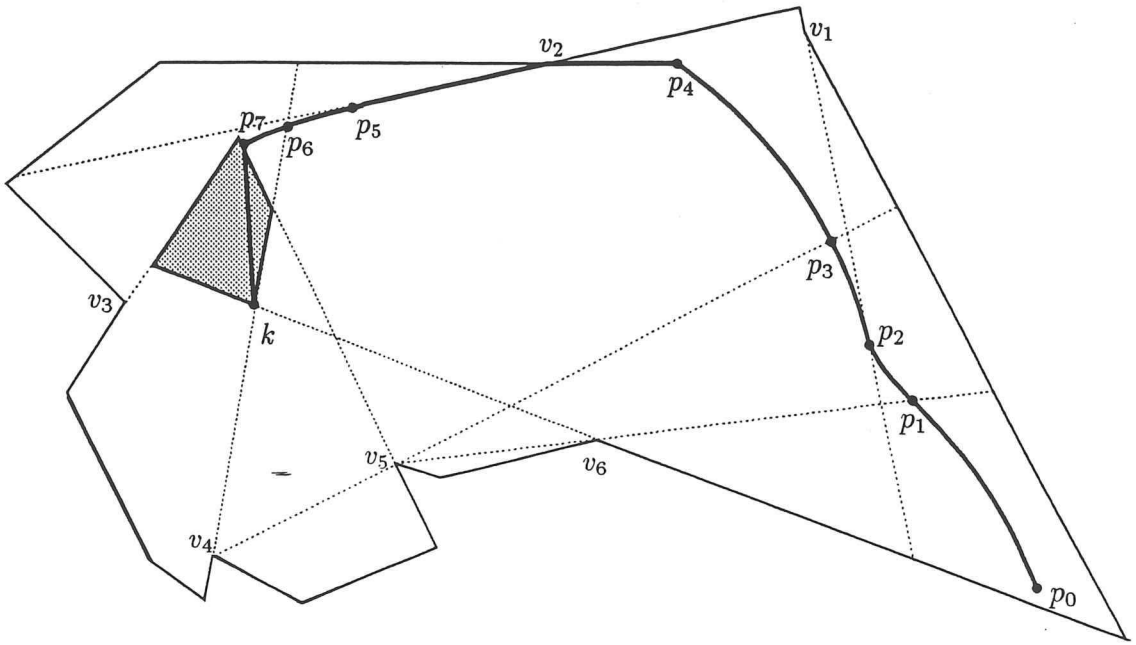
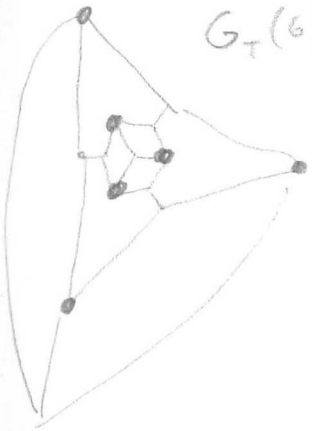


Figure 2: A path generated by strategy CAB.

[12] D. T. Lee and F. P. Preparata. An optimal algorithm for finding the kernel of a polygon. *J. ACM*, 26:415–421, 1979.

[13] A. López-Ortiz and S. Schuierer. Going home through an unknown neighbourhood. Technical report, Department of Computer Science, University of Waterloo, Canada, 1994.

[14] G. Rote. Curves with increasing chords. *Mathematical Proceedings of the Cambridge Philosophical Society*, 115(1):1–12, 1994.



The Graph of Triangulations of a Convex Polygon

F. Hurtado M. Noy

Departament de Matemàtica Aplicada II

Universitat Politècnica de Catalunya

Introduction

Let P be a simple polygon with n sides. The partition of the interior of P into triangles by means of a set of non-crossing diagonals is called a *triangulation* of the polygon. In general there are many ways of doing such a partition. Counting the number of ways of triangulating a convex polygon with n sides using $n-3$ internal diagonals is a classical problem in combinatorial geometry, which goes back to Euler, and that lied to the introduction of the Catalan numbers; now nearly ubiquitous. Some specific triangulations as well as the case for non-convex polygons have also been considered recently [HN1], [HN2], [HN3], [ES].

When two adjacent triangles form a convex quadrilateral then the shared diagonal can be *flipped* and a new triangulation of P is obtained. This is a well known process for sets of points, that allows the construction of the Delaunay Triangulation by successive flips selected with a local criterion, and that is also useful for enumerative purposes. In this work we consider for a given polygon P the *graph of triangulations of* P, denoted $G_T(P)$, having as nodes the triangulations of P, that are considered adjacent when they differ by a flip. We obtain some general properties and we focus then our attention in the special and relevant case of convex polygons. All convex polygons with n vertices have the same graph of triangulations, that will be denoted by $G_T(n)$. If v is a vertex of a convex polygon P, then P can be triangulated simply by joining v to all the other vertices. These special triangulations are called *fans*.

Properties of $G_T(n)$

i) The graph $G_T(n)$ is connected.
ii) $G_T(n)$ is triangle-free.
iii) The graph $G_T(n)$ is $(n-3)$-regular.
iv) The order of $G_T(n)$ is equal to $t_n = \frac{1}{n-1}\binom{2n-4}{n-2} \sim \gamma 4^n n^{-3/2}$. The size of $G_T(n)$ is $\frac{1}{2}(n-3)t_n$.
v) $G_T(m)$ is an induced subgraph of $G_T(n)$ $\forall m \leq n$.
vi) $girth(G_T(n)) = 4$ for $n \geq 6$.
vii) The radius of $G_T(n)$ is equal to $n-3$.
viii) The center of $G_T(n)$ consists on the n fans.
ix) $diameter(G_T(n)) \leq 2n-10$ for $n > 12$. The equality holds for all the n of the form $n = 10k^2 + 2$.
x) The automorphism group $\Gamma(G_T(n))$ is isomorphic to the dihedral group D_n of symmetries of a regular polygon with n sides.
xi) The connectivity of the graph $G_T(n)$ $(n \geq 5)$ is equal to $n-3$.
xii) $G_T(n)$ is Hamiltonian.

Comments

A graph that can be embedded in the plane such that all the vertices lie on the exterior face is called an *outerplanar graph*. An outerplanar graph is a *maximal outerplanar* graph (or a MOP) if the addition of a single edge results in a non-outerplanar graph. By traversing the (unique) Hamiltonian cycle of a MOP, labeling the nodes as the vertices on the boundary of a convex polygon, we see that MOPS of order n and triangulations of the convex n-polygon are synonymous (as graphs). So all the former results apply to MOPS.

A *binary tree* is defined to be either empty or to consist of a root node and a left and a right subtree, both of which are binary trees [K]. A tree is said to be of size n if it has n internal nodes. A *rotation* in a binary tree is the familiar operation used in maintaining balanced binary search trees. The *rotation graph* for trees of size n, denoted $G_R(n)$, has one vertex for each tree of size n, and an edge between nodes T and T' if there is a rotation that changes T into T'. By taking a fixed edge e of a convex polygon as a root, any triangle with basis e has two additional sides that can be recursively considered as roots for subtrees; in such a way we obtain a one to one correspondence between trees of size n and triangulations of an $(n+2)$-gon in which diagonal flips correspond to rotations, so $G_R(n)$ is isomorphic to $G_T(n+2)$. In [STT] Sleator, Tarjan and Thurston considered the problem of determining the diameter of $G_R(n)$ and obtained the result ix) above. In [L] Lucas proved that the rotation graph has a Hamiltonian cycle, with a long and intricate proof. The results were revisited in [LRR], where it is said that "It remains a challenge to provide a simpler proof that a cycle exists". One of our major results is a hierarchy for all triangulations of polygons with any number of vertices (that is, all MOPS of any order, or all binary trees of any size) that provides a way of *lifting* properties, in particular allowing simple proofs of xi) and xii).

References

[ES] P. Epstein and J.R. Sack, "Generating Triangulations at Random", in Proc. of the 4th Canadian Conf. on Comp. Geometry, 1992 (305-310).

[HN1] F. Hurtado and M. Noy, "Triangulations, Visibility Graph and Reflex Vertices of a Simple Polygon", Report MA2-IR-93-2, Universitat Politècnica de Catalunya, 1993.

[HN2] F. Hurtado and M. Noy, "Counting triangulations of almost-convex polygons", to appear in *Ars Combinatoria*.

[HN3] F. Hurtado and M. Noy, "Ears of triangulations and Catalan numbers", to appear in *Discrete Mathematics*.

[K] D. E. Knuth, *The Art of Computer Programming, Vol 3: Sorting and Searching*, Addison-Wesley, 1973.

[L] J. Lucas, "The Rotation Graph of binary trees is Hamiltonian", *J. Algorithms* **9** (1988), 503-535.

[LRR] J. Lucas, D. Roelants van Baronaigien and F. Ruskey, "On Rotations and the Generation of Binary Trees Hamiltonian", *J. Algorithms* **15** (1993), 343-366.

[STT] D. D. Sleator, R. E. Tarjan and W. P. Thurston, "Rotations distance, triangulations, and hyperbolic geometry", in Proc. of 18th ACM Symp. on Th. of Comp., 1986, 122-135.

degree sequence determines triangulation
distance to a fan depends only on the degree of this vertex

Voronoi Diagrams for Infinite Sets Generated by Consecutive Stretches

Manuel Abellanas * Eugenio Roanes-Lozano †

* Dep. Matemática Aplicada, Fac. Informática (Univ. Politécnica Madrid)

† Dep. Algebra, Fac. Educación (Univ. Complutense Madrid)

Abstract

In this work, the effective calculus of Voronoi diagrams for infinite set of points obtained by the action of the cyclic group generated by a certain stretch over a set of n points is studied. Mazon's work [Ma] also studies the problem of calculating Voronoi diagrams for infinite sets, but in that case, the sets are generated from a basic domain, saturating the set by the action of a discrete group of movements.
For any point P of a set S, let K_P be the degree of P in the Delaunay triangulation of S. A set obtained as mentioned above can be pre-processed in $O(n \cdot log\ n)$ time. The main result of this paper is to prove that, once the pre-process is done, it is possible to compute the Voronoi region of any point P in the set in optimal $O(K_P)$ time.

Keywords: Computational Geometry, Voronoi Diagrams, Delaunay Triangulations, Discrete Structures for Infinite Sets.

1 Introduction

It seems clear that to effectively calculate the Voronoi diagram of a certain set S, S should be finite. Anyway, if the diagram of an infinite set of points (or, what is equivalent, its Delaunay triangulation) is locally finite, we can think about the possibility of obtaining the diagram locally. To precise, we could think about the following problem: given a point $P \in S$, compute its Voronoi region V_P.

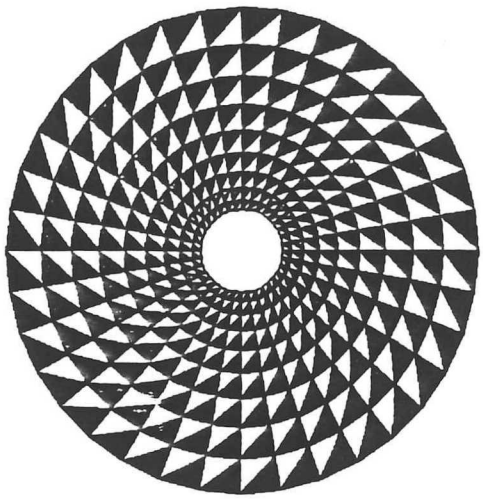
In [Ma] the computation of Voronoi diagrams of wallpaper patterns is studied (the infinite set of points is obtained by a applying a discrete group of movements to an initial set of points). In this paper we consider infinite sets obtained by applying the cyclic group generated by a certain stretch to an initial set of n points $S_0 = \{P_0^1, P_0^2, ..., P_0^n\}$ in the Euclidean plane.

We prove that with a $O(n \cdot log\ n)$ pre-process it is possible to find the Delaunay neighbours of any point P in the set in $O(K_P)$ time.

The geometrical idea for generating these sets is inspired by the roman tessellation shown below.

*mavellanas@fi.upm.es

†roanes2@eucmvx.sim.ucm.es



2 The Problem

Let $S_0 = \{P_0^1, P_0^2, P_0^3, ..., P_0^m\}$ be a finite set of points in the plane. Let O be a point interior to the convex hull $CH(S_0)$ and $C(O,r)$ an open circle of center O and radius r that does not contain any point of S_0 (neither in its interior nor in its border). Let $C(O,r')$ be an open circle of center O and such that $S_0 \subset C(O,r')$. Let $C_0 = C(O,r') \backslash C(O,r)$ (an annulus).

For any $i \geq 1 (i \in I\!N)$, let C_i be the image of C_{i-1} in the stretch, h, of ratio r'/r about O. Similarly, for any $i \geq 1 (i \in I\!N)$ let $S_i = h(S_{i-1})$. Finally, let $S = \bigcup_{i=0}^{\infty} S_i$

2.0.1 Problem.- *Find the Voronoi Diagram (or the Delaunay Triangulation) of S.*

2.0.2 Conjecture.- *Only $Vor(S_0 \cup S_1 \cup S_2)$ has to be calculated. If $i \geq 3$, for any point $P_i \in S_i$, if P_i is the image in h^{i-1} of a certain point $P_1 \in C_1$, h^{i-1} maps the Delaunay neighbours of P_1 onto the Delaunay neighbours of P_i.*

3 Results

3.1 Lemmas

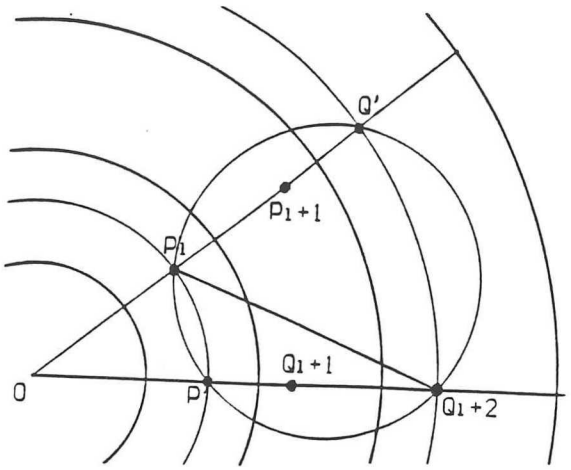
The following are well-known results and will be used in the proof of the Theorem.

3.1.1 Lemma.- *Let A, B, C, D be four points in the plane. Let us suppose that a circle exists such that A and B are in its border and C and D are points in the interior of the circle and in different sides of the segment AB. Then, any circle containing A and B contains at least one of the points C, D.*

3.1.2 Lemma.- *Given a collection of points S (in generic position), two points $A, B \in S$ are Delaunay neighbours iff a closed circle, W, exists, s.t. $A, B \in W$ and there is no other point of S in W.*

3.2 Main theorem

3.2.1 Theorem.- *If P_i is in C_i and Q is a Delaunay neighbour of P_i then $Q \in C_{i-1} \cup C_i \cup C_{i+1}$. Therefore, two Delaunay neighbours are always in the same annulus or in adjacent ones.*



Proof: Let us suppose that $P_i \in C_i$ and $Q_{i+2} \in C_{i+2}$ are Delaunay neighbours.
Let P_i' be the point on the half-line $O_{Q_{i+2}}$ s.t. $\overline{OP_i'} = \overline{OP_i}$. Let Q_{i+2}' be the point on the half-line O_{P_i} s.t. $\overline{OQ_{i+2}'} = \overline{OQ_{i+2}}$.
Then $P_{i+1} = h(P_i)$ is in C_{i+1} what implies that it is also in $\overline{P_iQ_{i+2}'}$ (as $Q_{i+2} \in C_{i+2}$). Similarly, if Q_{i+1} is the point s.t. $Q_{i+2} = h(Q_{i+1})$, then $Q_{i+1} \in \overline{P_i'Q_{i+2}}$.
The four points $P_i, P_i', Q_{i+2}, Q_{i+2}'$ are cocircular. Let W be the circumference that passes through all of them. It is clear that P_{i+1} and Q_{i+1} are points in the interior of W and in different sides of $\overline{P_iQ_{i+2}}$.
As a consequence of the Lemma, any circle containing P_i and Q_{i+2} contains at least one of the points P_{i+1}, Q_{i+1}. This is a contradiction with the assumption that P_i and Q_{i+2} are Delaunay neighbours. □

We want to effectively calculate the Delaunay triangulation of S, $S = \bigcup_{i=0}^{\infty} S_i$.

3.2.2 Notation.- *If $R \subseteq S$, let us denote by $Del(R, S)$ the set of edges of the Delaunay triangulation of the set S, which are incident with one or two points in R.*

3.2.3 Consequence.- *As a consequence of the previous Theorem, for any $i > 1$ ($i \in \mathbb{N}$),*

$$Del(S_i, S) = Del(S_i, S_{i-1} \cup S_i \cup S_{i+1})$$

and therefore, topologically,

$$Del(S_i, S) = Del(S_1, S_0 \cup S_1 \cup S_2) \ .$$

3.3 Case I: Only Stretches of Ratio Greater than 1 are Considered

3.3.1 Proposition.- *Stretches are transformations which preserve angles and shapes. Therefore, it follows from 3.2.3 that, for any $i > 1$ ($i \in \mathbb{N}$), $Del(S_i, S)$ is the image of $Del(S_1, S_0 \cup S_1 \cup S_2)$ in h^{i-1}.*

3.3.2 Corollary.- *As a consequence of the proposition above, it is enough to calculate $Del(S_0, S_0 \cup S_1)$ and $Del(S_1, S_0 \cup S_1 \cup S_2)$, because*

$$Del(S.S) = Del(S_0, S_0 \cup S_1) \cup Del(S_1, S_0 \cup S_1 \cup S_2) \cup \bigcup_{i=1}^{\infty} h^i(Del(S_1, S_0 \cup S_1 \cup S_2))$$

3.3.3 Observation.- *When looking for the Delaunay neighbours of the points in C_0 it is enough to look in $C_0 \cup C_1$. As there is no internal annulus, the case is slightly different to that of the other $C_i, i \geq 1$.*

4 Generalizations

We have proven the following results:

- The solution can be generalized for stretches of ratios smaller than 1.
- About generalizing the result to regions of other shapes:
  - Circles can not be substituted by squares
  - Moreover, circles can not be substituted by regions of any other shape.
- The problem can be generalized to $\mathbb{R}^n$.

5 Complexity

$Vor(S_0 \cup S_1 \cup S_2)$ can be calculated in time $O(n \cdot log\ n)$ [Kl]. From 3.4.1 it follows that for any point $P \in \overline{S}$, its Delaunay neighbours can be calculated in a time proportional to their number.

5.0.1 Theorem.- *For any set S^0 of n points in the plane, let S be the infinite set defined in section 2. It is possible to pre-process S in $O(n \cdot log\ n)$ time, in such a way that, for any query point $P \in S$, the Voronoi region $V(P)$ (or, equivalently, the Delaunay neighbours of P) can be computed in $O(K_P)$ time (being K_P the number of neighbours of P in $Del(S)$).*

References

[A] **F. Aurenhammer:** *Voronoi diagrams: a survey of a fundamental geometric data structure.* Fachbereich Math. Report B-90-09. Free Univ. Berlin, 1990.

[Kl] **V. Klee:** *On the complexity of d-dimensional Voronoi diagrams.* Archiv der Mathematik, 34, 1980, (75-80).

[Ke] **R. Klein:** *Concrete and Abstract Voronoi Diagrams.* Lecture Notes in Computer Science. Springer-Verlag, 1989.

[Ma] **M. L. Mazón:** *Diagramas de Voronoi en caleidoscopios.* (PhD. Thesis). Univ. de Cantabria, 1992.

[CMRS] **A.G. Corbalán, M.L. Mazón, T. Recio, F. Santos:** *On the Topological Shape of Planar Voronoi Diagrams.* In: **C. Icking, R. Klein** (editors): *Proceedings of the 9th European Workshop on Computational Geometry (CG'93).* Informatik Berichte Nr.140, 3/1993. Fern Universität Hagen, 1993.

[PS] **F.P. Preparata, M.I. Shamos:** *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[TY] **J. Torikawi, S. Yokoi:** *Voronoi and related neighbors on digitized two-dimensional space with applications to texture analysis.* In: **G.T. Toussaint:** *Computational Morphology.* North-Holland, 1991.

[Vo] **M.G: Voronoi:** *Nouvelles applications des parametres continus a la theorie des formes quadratiques.* Journal für die Reine und Angewandte Mathematik. 134. 1908 (198-287).

Randomized Incremental Construction of Simple Abstract Voronoi Diagrams in 3-Space

Ngọc-Minh Lê *

Abstract

We introduce the simple abstract Voronoi diagram in 3-space as an abstraction of the usual Voronoi diagram. We show that the 3-dimensional simple abstract Voronoi diagram of n sites can be computed in $O(n^2)$ expected time using $O(n^2)$ expected space by a randomized algorithm. The algorithm is based on the randomized incremental construction technique of Clarkson and Shor [7]. We apply the algorithm to three concrete types of such diagrams: power diagrams, diagrams under ellipsoid convex distances, and diagrams under the Hausdorff distance for sites that are parallel segments all having the same length.

1 Introduction

The Voronoi diagram ist an important data structure in computational geometry. Given n sites in d-space, the Voronoi diagram partitions d-space into n regions, one for each site. The region of a site p consists of all points in d-space that lie closer to p than to any of the other sites. For a survey on Voronoi diagrams and their applications we refer to Aurenhammer [2].

In the plane, efficient methods for computing the diagram under the Euclidean metric have been found: the divide-and-conquer method of Shamos and Hoey [20], the sweep-line method of Fortune [9], the lifting method of Brown [4] and Edelsbrunner and Seidel [8]. The Voronoi diagrams under convex distance funtions were considered by Chew and Drysdale [5], they gave a divide-and-conquer algorithm for computing these diagrams.

Klein [13] recognized that many types of planar Voronoi diagrams can be regarded as specializations of only one type of diagram, which he calls *abstract Voronoi diagrams*. He proposed an axiomatic approach that is based on the concept of bisecting curves rather than the concept of proximity derived from a concrete distance measure. For each pair of sites p and q, he assumes the existence of a bi-infinite bisecting curve that divides the plane into a p-region and a q-region. The Voronoi region of a site p is defined to be the intersection of all p-regions for different q's. Voronoi regions are assumed to be simply-connected and to partition the plane. Klein develops a divide-and-conquer algorithm to compute the planar abstract Voronoi diagrams in $O(n \log n)$ time.

Klein, Mehlhorn, Meiser, and Ó'Dúnlaing [18, 14] show how to construct the planar abstract Voronoi diagram in $O(n \log n)$ expected time incrementally by adding the sites one by one in random order and maintaining the diagram constructed so far. Their algorithm is based on the randomized incremental construction technique of Clarkson and Shor [7] and its improvement [3].

Statement of problem. In this paper we present a "low-technology" generalization of the concept of abstract Voronoi diagram to 3-space, "low-technology" in the sense that we don't try to keep the set of axioms to be minimal and we allow them to contain redundancy; moreover we assume, by analogy with the case of the Euclidean metric, that the intersection of bisectors satisfy certain specific conditions. We develop an algorithm for computing such diagrams.

Main results. Similar to the axiomatic approach of Klein we postulate, for each pair of sites p and q, the existence of a bisecting surface $B(p,q)$ that divides 3-space into a p-region and a q-region. We define the Voronoi region of a site p to be the intersection of all p-regions for different q's. Further, we assume that Voronoi regions are either empty or homeomorphic to a 3-ball and partition 3-space. Since the non-degenerate Voronoi diagram in the Euclidean metric should serve as a model for our axioms, we assume that, given 5 sites p, q, r, s, t, the set $B(p,q) \cap B(q,r)$ is a bi-infinite curve, the set $B(p,q) \cap B(q,r) \cap B(r,s)$ is a point, and the set $B(p,q) \cap B(q,r) \cap B(r,s) \cap B(s,t)$ is empty;

*Praktische Informatik VI, FernUniversität Hagen, D-58084 Hagen, Germany; ngoc-minh.le@fernuni-hagen.de.
This work was partially supported by Deutsche Forschungsgemeinschaft grant Kl 655/2-2.

in addition, we require that bisecting surfaces should intersect transversely. The diagram arising this way is called the (3-dimensional) simple abstract Voronoi diagram. We present an algorithm that constructs the simple abstract Voronoi diagram in $O(n^2)$ expected time and $O(n^2)$ expected space. The algorithm generalizes the approach for the planar case [18, 14] and is an instance of the randomized incremental construction of Clarkson and Shor, but the history graph [3] is used instead of the original conflict graph.

Significance of work. Our algorithm can be applied to computing the power diagram [1], also called Voronoi diagram in the Laguerre geometry, in $O(n^2)$ expected time using $O(n^2)$ expected space. Previously, the algorithm in [1] of Aurenhammer constructs the power diagram by computing a certain convex hull in 4-space; it runs in $O(n^2)$ time using $O(n^2)$ space.

Our algorithm computes the Voronoi diagram of points in general position under any ellipsoid convex distance within the $O(n^2)$ resource bounds. Previously, one constructs this diagram in $O(n^2)$ time using $O(n^2)$ space by employing certain space transformations to reduce it to a Euclidean diagram. In [22], Wagner develops a randomized incremental algorithm for constructing the Euclidean diagram; however, his analysis relies entirely on properties of the Euclidean metric. Inagaki *et al* [12] present a practical incremental algorithm for computing the Euclidean diagram without analysing the complexity bounds.

We give a characterization of non-ellipsoid convex distance functions which strongly indicates that, within the class of convex distance functions, exactly the ones whose unit sphere is an ellipsoid give rise to bisecting surfaces that satisfy our axioms (provided that the sites are points in general position).

Our algorithm applies to computing the diagram under the Hausdorff distance of sites that are parallel segments all of which have the same length.

The main result of our work is the following.

Theorem 1 *The simple abstract Voronoi diagram $V(S)$ of n sites in 3-space can be computed by a randomized algorithm in $O(n^2)$ expected time and $O(n^2)$ expected space. Furthermore, the r-th site can be inserted in $O(r)$ expected time.*

2 Applications

1. *Power diagrams.* Power diagrams in higher dimensions have been thoroughly investigated by Aurenhammer [1]. In 3-space, our algorithm is just another one to compute power diagrams within $O(n^2)$ resource bounds.

2. *Ellipsoid convex distances.* Previously, to compute diagrams under ellipsoid distances one first transforms the sites to new ones using a certain affine mapping, computes the Euclidean diagram of the new sites, and then inversely transforms the computed diagram to one under the corresponding symmetric ellipsoid distance. The final diagram is obtained by using a mapping analogous to [21, Proposition 6]. In contrast to this method, our algorithm constructs diagrams under ellipsoid convex distances "directly".

Lemma 2 *Let d be an ellipsoid convex distance in 3-space. Let S be a set of n points that is non-degenerate with respect to d. Then, the system of bisectors $B(p,q)$ under the distance d, with distinct $p, q \in S$, is simple.*

How large is, within the class of convex distance functions, the subclass of those that give rise to simple systems of bisectors? To give a partial answer to this question, we will need the following result of Goodey [10, Theorem 4].

Theorem 3 *(Goodey) Let K_1 and K_2 be convex bodies in D-space, with $D \geq 3$. Then the intersection of bd K_1 and bd K_2' is contained in a hyperplane for all translates K_2' of K_2 with $K_1 \neq K_2'$ if and only if K_1 and K_2 are homothetic ellipsoids.*

We are now in a position to prove the following lemma.

Lemma 4 *Let d be a convex distance in D-space whose unit-sphere is not an ellipsoid, with $D \geq 3$. Then there are $D+1$ affinely independent points $p_1, \ldots, p_{D+1}$ so that the cardinality of $B(p_1, \ldots, p_{D+1})$ is ≥ 2.*

Proof. Let K_1 and K_2 be full d-spheres. Suppose for the sake of contradiction that, for all translates K_2' of K_2 which satisfy $K_1 \neq K_2'$ and $bd\, K_1 \cap bd\, K_2' \neq \emptyset$, the set $bd\, K_1 \cap bd\, K_2'$ is contained in some hyperplane. Then, Theorem 3 implies that K_1 and K_2 are (homothetic) ellipsoids, a contradiction. Hence, there is a translate K_2' of K_2 that satisfies $K_1 \neq K_2'$ and $bd\, K_1 \cap bd\, K_2' \neq \emptyset$ so that $bd\, K_1 \cap bd\, K_2'$ is not contained in any hyperplane. Clearly, $bd\, K_1 \cap bd\, K_2'$ must contain $D+1$ affinely independent points, denoted $p_1, \ldots, p_{D+1}$. Since $bd\, K_1$ and $bd\, K_2'$ pass through these points and $bd\, K_1 \neq bd\, K_2'$, the lemma follows. □

This result strongly indicates that exactly ellipsoid convex distances give rise, *in general*, to simple systems of bisectors.

3. *Hausdorff distance.* The *Hausdorff distance* from a point $x \in \mathbb{R}^3$ to a segment s is defined to be $d_H(x, s) = \max\{|x - y| \ ; \ y \in s\}$; see [19, 3.7.3]. Let S be a set of parallel segments all having the same length, and let S be non-degenerate under d_H. Then it can be shown that the arising system of bisectors under d_H is simple (Details are omitted). Thus, our algorithm computes the Voronoi diagram of S under the Hausdorff distance in expected time $O(n^2)$ using $O(n^2)$ expected space.

3 Conclusion

Extending the ideas in [18, 14], we have developed an algorithm to compute the 3-dimensional simple abstract Voronoi diagram of n sites in $O(n^2)$ expected time using $O(n^2)$ expected space. We have presented three models for the simple abstract Voronoi diagram: power diagrams, diagrams under any ellipsoid convex distance, and diagrams under the Hausdorff distance of sites that are parallel segments all having the same length. Our results suggest some natural questions: First, is it possible to extend our approach to the case that $B(p, q, r)$ consists of an arbitrary number of bi-infinite curves, as illustrated in Figure 1, and $B(p, q, r, s)$ consists of an arbitrary number of points? Such diagrams include Voronoi diagrams of points in general position under convex distance functions that are different from the ellipsoid ones, diagrams under additive weighted distances, diagrams for segment sites, space-time diagrams for planar dynamic Voronoi diagrams, ... Finally, the case of non-crossing but touching bisectors should also be investigated in order to include degenerate systems of bisectors.

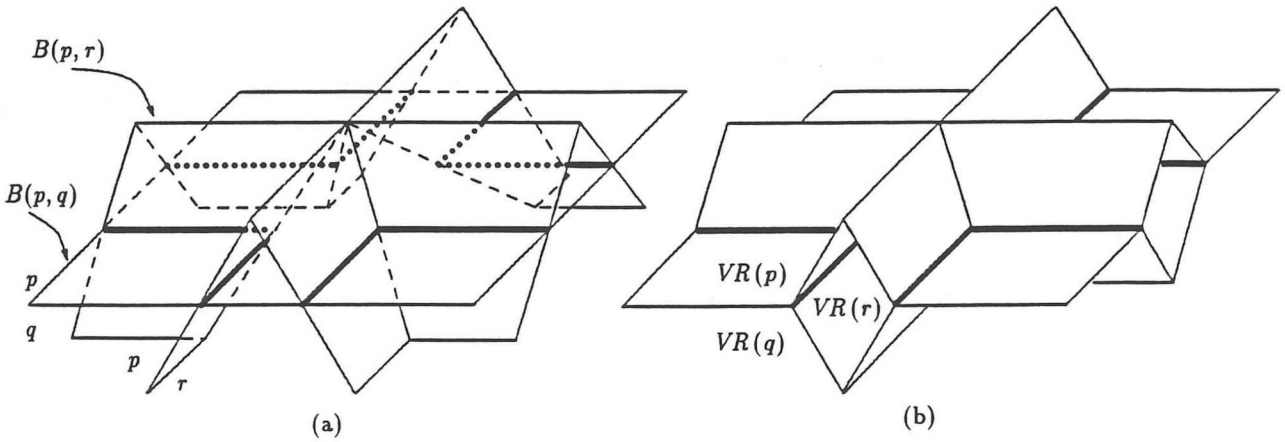


Figure 1: The connected components of $B(p, q, r)$ are shown by thick lines. (a) The bisector $B(q, r)$ is symmetric to $B(p, r)$ w.r.p. to $B(p, q)$ and is not shown. (b) The Voronoi regions of p, q, and r.

Acknowledgements. I thank Rolf Klein for helpful discussions and for the references [14] and [22]. I also thank Rolf Klein, Kurt Mehlhorn and Stephan Meiser for having directly or indirectly contributed some initial ideas that have lead to the results presented in this paper. Thanks also go to Marisa Mazon Calpena and Francisco Santos for some hints concerning ellipsoid convex distances. All errors in this paper are mine.

References

[1] F. Aurenhammer: *Power diagrams: properties, algorithms and applications.* SIAM Journal of Computing 16 (1987), pp. 78–96.

[2] F. Aurenhammer: *Voronoi Diagrams — A Survey of a Fundamental Geometric Data Structure.* ACM Computer Surveys 23(3), 1991.

[3] J. D. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec: *Applications of random sampling to on-line algorithms in computational geometry.* Discrete & Comput. Geom. 8, pp. 51–71, 1992.

[4] K. Q. Brown: *Voronoi diagrams from convex hulls.* Information Processing Letters 9(5), pp. 223–228, 1979.

[5] L. P. Chew and R. L. Drysdale: *Voronoi diagrams based on convex distance functions.* Proceedings of the ACM Symposium on Computational Geometry, pp. 235–244, 1985.

[6] K. L. Clarkson, K. Mehlhorn and R. Seidel: *Four results on randomized incremental constructions.* Computational Geometry: Theory and Applications 3 (1993), pp. 185–212.

[7] K. L. Clarkson and P. W. Shor: *Applications of Random Sampling in Computational Geometry, II.* Discrete & Comput. Geom. 4, pp. 387–421, 1989.

[8] H. Edelsbrunner and R. Seidel: *Voronoi diagrams and arrangements.* Discrete & Comput. Geom. 1, pp. 25–44, 1986.

[9] S. Fortune: *A sweepline algorithm for Voronoi diagrams.* Algorithmica 2, pp. 153–174, 1987.

[10] P. R. Goodey: *Homothetic ellipsoids.* Math. Proc. Camb. Phil. Soc. 93 (1983), pp. 25–34.

[11] C. Icking, R. Klein, N.-M. Lê, L. Ma: *Convex Distance Functions in 3-Space are Different.* Proceedings 9th ACM Symposium on Computational Geometry, 1993.

[12] H. Inagaki, K. Sugihara, and N. Sugie: *Numerically Robust Incremental Algorithm for Constructing Three-Dimensional Voronoi Diagrams.* Proceedings 4th Canadian Conference on Computational Geometry, pp. 334–339, 1992.

[13] R. Klein: *Concrete and Abstract Voronoi Diagrams.* LNCS 400, Springer, 1989.

[14] R. Klein, K. Mehlhorn, and S. Meiser: *Randomized Incremental Construction of Abstract Voronoi Diagrams.* Technical Report MPI–I–93–105, Max-Planck-Institut für Informatik, Germany, 1993. Also in Comput. Geometry: Theory and Applications 3 (1993), pp. 157–184.

[15] N.-M. Lê: *On general properties of smooth strictly convex distance functions in D-space.* Proceedings 5th Canadian Conference on Computational Geometry, pp. 375–380, 1993.

[16] N. M. Lê: *On Voronoi diagrams in the L_p-metric in higher dimensions.* Proc. 11th Symp. Theoret. Aspects Comp. Sci., P. Enjalbert *et al.* (Eds.), LNCS 775, Springer, pp. 711–722, 1994.

[17] N.-M. Lê: *An axiomatic approach to Voronoi diagrams in 3-space.* Manuscript, 1994.

[18] K. Mehlhorn, S. Meiser, and C. Ó'Dúnlaing: *On the Construction of Abstract Voronoi Diagrams.* Discrete & Comput. Geom. 6, pp. 211–224, 1991.

[19] A. Okabe, B. Boots, and K. Sugihara: *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* John Wiley & Sons, 1992.

[20] M. I. Shamos and D. Hoey: *Closest-point problems.* Proc. 16th IEEE Symp. on Foundations of Computer Science, pp. 151–162, 1975.

[21] K. Sugihara: *Voronoi diagrams in a river.* Int. J. Comput. Geom. & Appl. 2(1), pp. 29–48, 1992.

[22] K. Wagner: *Ein Las Vegas-Algorithmus zur Berechnung dreidimensionaler Voronoidiagramme.* Master's thesis, CS Dept., University of Saarbrücken, Germany, 1990.

canceled

Three Dimensional Delaunay Triangulation Using an Uniform Grid

(combined with the next talk)

Lucian CUCU Mircea DRAGAN
Viorel NEGRU Dorin MANCU

Universität Temesware
Romania

Abstract

We implement of an algorithm for constructing the Delaunay Triangulation and its dual Voronoi Diagram for large sets of points in the 3 dimensional space. The main characteristics of our implementation are: efficient cell management and gradual simplification of the uniform grid used. The algorithm is designed such that a further parallel implementation is possible.

1 Introduction

We present here an extension to 3 dimensions of our previous work [Cucu *et al.*, 1993] [Cucu *et al.*, 1994] regarding the construction of Delaunay Triangulations and Voronoi Diagrams in 2 dimensions. Therefore, the approach is similar to the implementation of the 2-dimensional algorithm: we determine the Delaunay Triangulation, while the Voronoi Diagram is obtained as a side-effect. The construction of the Delaunay Triangulation is performed in a deterministic way, in that each Delaunay cell which is found is part of the final triangulation. This is due to the searching strategy used at the computational level, similar to [Fang and Piegl, 1993] and [Dwyer, 1991]. Theoretically, the algorithm requires only $O(n)$ time in the average [Dwyer, 1991] for random points in the unit ball.

2 Informal description of the algorithm

The **preprocessing** step consist in finding the smallest rectangular box which contains all the sites, and then constructing an uniform grid by dividing it in small cubes named *boxes* of same size.

The algorithm starts by finding a first tetrahedron. Its four facets are inserted into the facet dictionary as half-processed facets (i.e. facets for which only one tetrahedron is already known).

At each step, the fourth vertex for a half-processed facet is searched for. The search is performed in the opposite halfspace of the known tetrahedron. The search starts with the

points from the box which contains the center of the facet, and it is gradually extended by wrapping around the initial box. The searching process ends when the fourth point is found or the boundaries of the grid are reached. The eventually new half-facets are inserted in the dictionary, and the other found half-facets become completed-facets. This basic step is repeated until there are no more half-processed facets left in the dictionary.

The most time consuming part of the algorithm is the searching step and it is dependent of the number of points from the grid. A significant speed-up can be obtained if the sites that become interior to the already found triangulation are "hidden" from further searching.

3 Implementation

The input (real) data, consisting of the coordinates of the sites is kept in an array, each element of the array being assigned to a cell of the 3-dimensional uniform grid. The grid is designed to hold in average 4 sites (we found this blocking factor by experiments). As each Delaunay facet (2-cell) is characterized by its vertices, the data structure used to keep track of the facets holds the indices of those vertices in the vertex array. Also, each facet is part of one (if it is on the convex hull) or two tetrahedrons (Delaunay 3-cells) which can be properly characterized by the centers of their circumspheres. These are, in fact, Voronoi vertices. The coordinates of these centers are computed anyway in the process of determining the fourth vertex of a tetrahedron. Because the center of a circumsphere of a tetrahedron is related to each of the four facets, it would be a waist of storage to keep its coordinates stored with each facet. For that reason, the coordinates of each such center is kept in a separate data structure which holds also links to all the four corresponding facets, while in each facet data structure there are kept two pointers to such centers, one for each possible tetrahedron in which the facet is part.

All the facets are hashed by a function of the indices of their vertices into a facet dictionary [Dwyer, 1991], in order to keep the access times low. At each stage of the algorithm there may be two types of facets in the dictionary: facets for which both Delaunay 3-cell were already determined (hence the pointers to the two centers are non-NULL) and facets encountered so far only in one tetrahedron. These latter facets, which we call *half-processed* facets are subject to future search for the fourth vertex of their right hand tetrahedron.

When the algorithm ends, the data structures hold information on both the Delaunay Triangulation and the Voronoi Diagram.

The algorithm is designed in such a manner that it can be used as "conquer" step in our parallel algorithm on the Sequent Symmetry shared memory architecture.

4 Timings and conclusions

The tests are done using sets of random points from the unit cube. The time grows almost linear when the number of points is at most 2500. Timings are presented in figure 4.

We hope to decrease the timings by using heuristics to established the order in which the half-facets are processed. Then the simplification of the grid is optimal.

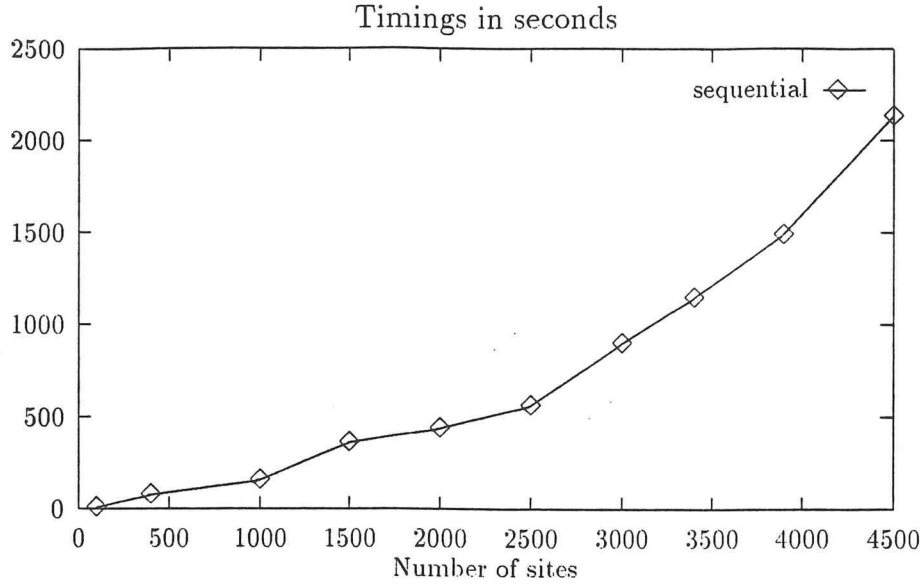


Figure 1: Timings for sequential implementations.

References

[Cucu *et al.*, 1993] L. Cucu, M. Dragan, T. Jebelean, and V. Negru. Delaunay triangulation using divide-and-conquer. Technical Report 93-62, RISC-Linz. December 1993.

[Cucu *et al.*, 1994] L. Cucu, M. Dragan, T. Jebelean, and V. Negru. Divide and Conquer Voronoi Diagram Construction. In *European Workshop on Computational Geometry, Santander*, 1994.

[Dwyer, 1991] R.A. Dwyer. Higher-dimensional Voronoi diagrams in linear expected time. *Discret Comput. Geom.*, 6:343-367, 1991.

[Fang and Piegl, 1993] T.-P. Fang and L.A. Piegl. Delaunay triangulation using a uniform grid. *IEEE Computer Graphics & Applications*, may:36–47, 1993.

EXACT ARITHMETIC

Parallel Implementation of 3D Delaunay Triangulation

EXTENDED ABSTRACT

Lucian CUCU Mircea DRAGAN
Tudor JEBELEAN Viorel NEGRU

RISC

Universität Temesware
Romania

TUDOR @risc.uni-linzac.at

Abstract

We extend our practical approach on constructing Delaunay triangulations and Voronoi diagrams (see [Cucu *et al.*, 1993]) to the three dimensional space. The sequential algorithm that we implemented is based on a facet dictionary and use an uniform grid. The parallelization scheme is based on the divide-and-conquer approach and tests have been made on large sets of uniformly distributed input points. No attempt was made to deal with degenerate cases.

1 Basics of the sequential algorithm

The study of the 3-dimensional problem is a natural extension of our previous work, and is also interesting both from the practical and the theoretical points of view. A preprocessing step is performed on the input data to assign each site to a box of a uniform grid. The sequential algorithm finds the Delaunay triangulation and then the Voronoi diagram by using some side-effect data which is computed during the triangulation process (cell adjacency information, Voronoi vertices). The triangulation is found in the "deterministic" way - all the cells which are found are part of the final triangulation (see also [Fang and Piegl, 1993]).

Facet oriented data structures, similar to the facet-edge data structure of [Dobkin and Laszlo, 1987], organized into a **"facet dictionary"** (see [Dwyer, 1991]) are the main data structure on which the algorithm is based(see fig.1). However, unlike the facet-edge data structure of [Dobkin and Laszlo, 1987], our data structure lacks explicit information about facet and edge orientation. The immediate benefit of dealing with non-oriented facets is a half-cut in the number of items that have to be stored and also an increased compactness of the data.

A linked list of structures holding information on the Voronoi vertices (which are computed during the triangulation process) and pointers to the corresponding Delaunay facets is also created.

The main difference between the two variants of the algorithm is the order in which the "half-processed" facets (facets for which only the tetrahedron on one side has been found so far) are visited. As a result of that order, in the first variant, one 3-cell wide "threads" of tetrahedrons will grow until they reach the convex hull or until the last tetrahedron will

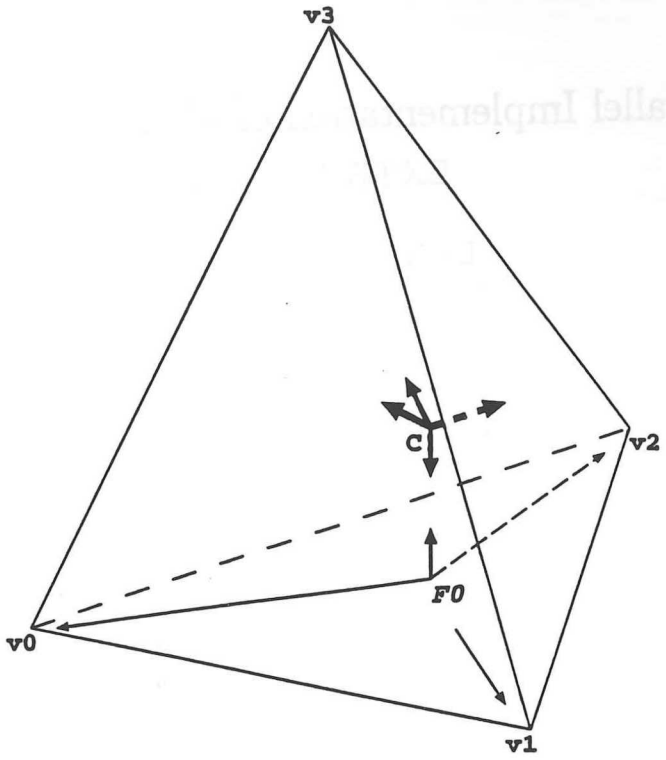


Figure 1: Data structure.

match all its facets against the facets of other already found cells of the triangulation. The facet dictionary is consulted for half-processed facets only when one of the above situations occur. In contrast, in the second variant the partial computed Delaunay triangulation will grow around a "kernel", exposing excrescences not "longer" than one tetrahedron, due to the fact that the order in which half-processed facets are first determined is "remembered" in a FIFO list.

2 Parallelization scheme

The parallelization of the algorithm is based on the **divide-and-conquer** technique - see [Cucu *et al.*, 1993]. There are three distinct steps:

1. *preprocessing* of the data (construct the grid and place the points in it);
2. *divide* the set of points into two parts by dividing the grid and *conquer*, that is construct the triangulation of each subset independently;
3. *merge* the two previous Delaunay triangulations.

Steps 2 and 3 can be applied recursively:

- at step 2 the triangulation of each subset can be done by further divide and conquer;

- at bottom-level of the recursion a sequential *local triangulation* is performed;
- at top-level of the recursion step 3 will finish the complete triangulation of the given set of points.

The *parallelization* of this algorithm is done by performing the conquering in parallel after each divide.

The advantage of this parallelization scheme is that the access to the data structure is independent for each parallel processor, hence there is no more need for monitoring. Also, interprocess communication is reduced to "start/end-of-job" messages, no other synchronization is necessary. However, a problem appears which is due to the characteristics of the data structure inherited from the sequential algorithm. This structure does not give the desired efficiency when used for irregular areas, as they occur during the merging step.

3 Conclusions and further work

The tests have been carried out for sets of points randomly distributed in the unit cube(see fig. 3).

The algorithm works good if the number of processors is small and this is the case with Sequent Symmetry shared memory architecture. To increase the speedup of parallel implementation we must reduce timings for the merging step. This can be realised by overlaping the grid partitions in the divide step. Then in the merge step we have more cell management instead of searching for new cells.

In order to deal with special cases (collinear, co-spherical points), which arise in practical data sets, an interesting direction to follow is to implement the algorithms using the symbolic computation systems SACLIB (sequential) and PACLIB (parallel) developed at RISC-Linz.

References

[Cucu *et al.*, 1993] L. Cucu, M. Dragan, T. Jebelean, and V. Negru. Delaunay triangulation using divide-and-conquer. Technical Report 93-62, RISC-Linz, December 1993.

[Dobkin and Laszlo, 1987] D.P. Dobkin and M.J. Laszlo. Primitives for the Manipulation of Three-Dimensional Subdivisions. In *3rd ACM Symp. on Comp. Geometry)*, pages 86–99. ACM, 1987.

[Dwyer, 1991] R.A. Dwyer. Higher-dimensional Voronoi diagrams in linear expected time. *Discret Comput. Geom.*, 6:343–367, 1991.

[Fang and Piegl, 1993] T.-P. Fang and L.A. Piegl. Delaunay triangulation using a uniform grid. *IEEE Computer Graphics & Applications*, may:36–47, 1993.

Segment

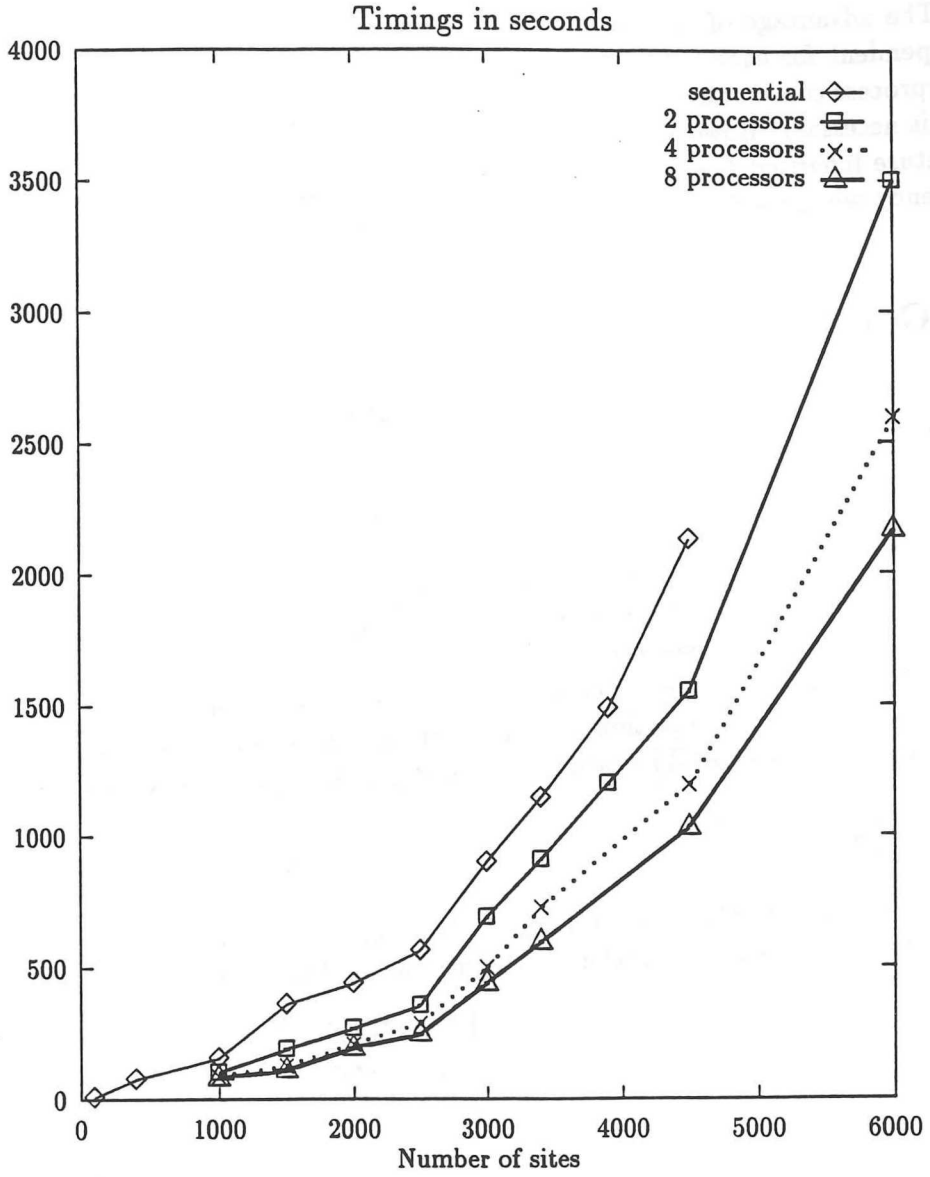


Figure 2: Timings for sequential and parallel implementations.

Voronoi Diagrams for Spheres and their Application in Motion Planning

Michael Wolfrath
Lehrstuhl für Informatik I
Universität Würzburg
Am Hubland
D-97074 Würzburg
Wolfrath@informatik.uni-wuerzburg.de

The 2-dimensional Voronoi diagram for a finite number of disjoint polygons is used in literature to plan the collision-free motion of a disc. We want to plan the motion of a ball through a scene of n disjoint spheres by using the Voronoi diagram for spheres in the Euclidean, 3-dimensional space. First we present some topological properties of the Voronoi diagram for spheres.

This analysis leads to the observation, that the diagram is not yet sufficient for the purpose of motion planning. To overcome this problem, we present an extension of the Voronoi diagram for spheres. Now we can guarantee, that there exists a safe path through the scene, if and only if there exists a safe path in the extended Voronoi diagram.

Overlaying simply connected planar subdivision in linear time

Klaus Hinrichs + Ulrich Finke
Westf. Wilhelms-Univ.
FB-15 Informatik
Einsteinerstr. 62
D-48149 Muenster
Khh@math.uni-muenster.de

Algorithms to compute the overlay of planar subdivisions are of theoretical and practical significance. They can be applied to solve the map overlay problem in spatial information systems, the hidden surface problem in computer graphics or design rule checking in VLSI layout design. We present an algorithm which computes the overlay of two simply connected planar subdivisions in linear time measured in output size. The algorithm is based on a gragh exploration technique and uses a data structure to represent planar subdivision.

$\min(t_1, t_2) = t_{10}$

Local Properties of Triangulations (Extended Abstract) *

Oswin Aichholzer†
Institute for Theoretical Computer Science
Graz University of Technology
Klosterwiesgasse 32/2, A-8010 Graz, Austria
e-mail: oaich@igi.tu-graz.ac.at

1 Introduction

In this paper we study local properties of two well known triangulations of a planar point set S, both of which are defined in a non-local way. The first one is the greedy triangulation (GT) that is defined algorithmically: it can be obtained by starting with the empty set and at each step adding the shortest compatible edge between two points of S, where a compatible edge is defined to be an edge that crosses none of the previously added edges. For this triangulation recent investigations have shown how to compute it in expected time $O(n \log n)$ [DDMW] and most recently in $O(n)$ [DRA].

One key observation for this algorithms was the so called *exclusion region* for a possible edge pq (throughout this paper let $p, q \in S$) of the triangulation: If each half-region of the exclusion region of two points p and q contains at least one point in S (other than p and q) then the edge pq cannot be in the desired triangulation of S. We will survey some different exclusion regions for the GT and derive some upper bounds for their size. With the help of this exclusion regions the computation of the GT for uniformly distributed points is possible in linear expected time.

Next we investigate *inclusion regions* for an edge pq: If the inclusion region of two points p and q does not contain any point from S (other than p and q) then the edge pq is in the considered triangulation of S. For the GT we give some examples and discuss their properties. Note that for the Delaunay triangulation exclusion and inclusion regions can be derived directly from its definition.

The other triangulation we deal with is the minimum-weight triangulation (MWT) which minimizes the total length of all edges among all possible triangulations of S. It arises in many fields, e.g. numerical analysis, numerical approximation, or in the reconstruction of surfaces from contours [WA]. Though it has been shown how to compute the MWT in time $O(n^3)$ for the special case of n-vertex polygons [Kl], there are no known efficiently computable algorithms for the MWT in the general case [GJ]. For a better understanding of the structure of the MWT it is necessary to know some local properties of this global defined triangulation. Recent results [AART] use a local edge property (the so called *light edge*, see below) to give better lower bounds for the weight of the MWT and to derive special point sets for that the MWT can be computed in polynomial time.

*This paper benefited from discussions with Franz Aurenhammer, Scot Drysdale and Günter Rote.

†This research has been supported by the Spezialforschungsbereich F003 "Optimierung und Kontrolle".

2 Overview of Results

2.1 Exclusion Regions

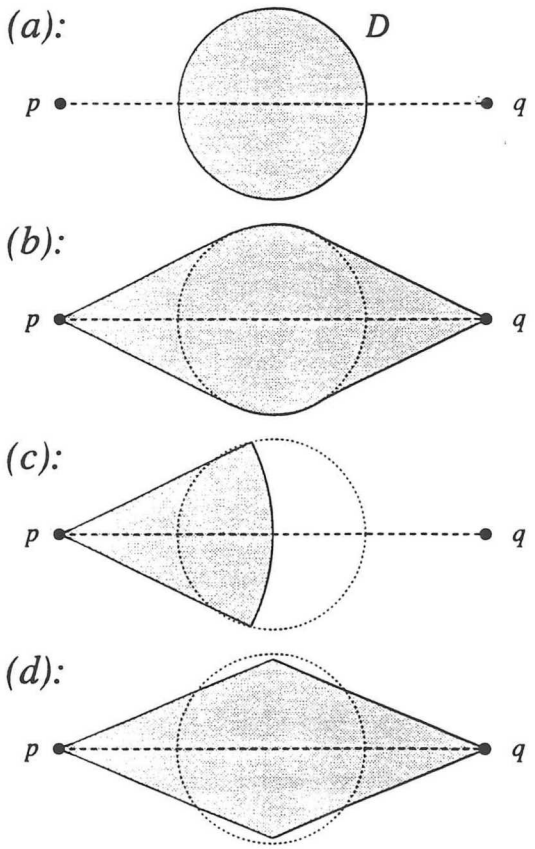


Figure 1: Exclusion regions for the greedy (a-d) and minimum-weight triangulation (d).

In the literature four different exclusion regions for a possible edge pq of the GT can be found (see Figure 1). The first was given in [DDMW] and was just a circle D of radius $|pq|/(2\sqrt{5})$ centered at the midpoint of the

edge pq (Fig. 1(a)). With this exclusion region Dickerson et al. developed a generate-and-test algorithm for uniformly distributed point sets where they generate a set of only $O(n \log n)$ candidate edges and then compute the GT from this edge set in $O(n \log n)$ expected time. Basically the same idea was used by Drysdale et al. [DRA] but they used an improved exclusion region: the so called eye-shaped region, cf. Figure 1(b). This is just the same circle extended by the sectors build by the tangent lines from p and q, respectively. Using this region their algorithm generates only $O(n)$ candidate pairs and then they show how to build the GT from this subset in linear expected time. Thus the usefulness of the exclusion region for the GT is to reduce the number of possible edges by local tests which can be done very efficiently.

In [DDMW] it was conjectured that the central circle D may have a radius up to $|pq|/(2\sqrt{2})$, motivated by Figure 2.

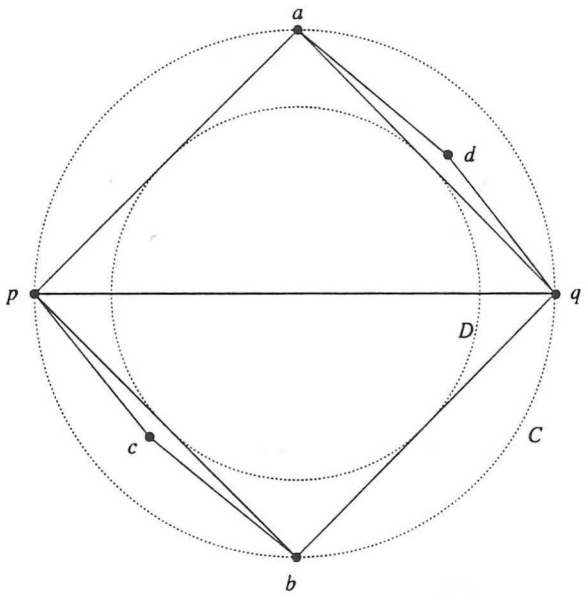


Figure 2: A simple upper bound.

Note that a and b are ϵ outside of the circle C with diameter pq and that the points c and d may be moved almost arbitrary as long as they are inside C and 'outside' the lines pb and aq, respectively. Thus the diamond provides an upper bound for any symmetric exclusion region for the GT.

The following example disproves the above conjecture:

Starting with Figure 2 move the points a and b counter clockwise along C and move the points c and d closer to p and q, respectively. Note that we do this in a symmetric way. Also note that for illustration the movement in Figure 3 is much larger than it is possible for the real example (we skip the exact locations for this abstract). At last we add the points e and f which may be as closed to the line aq (pb) as desired, e.g. only ϵ away from the center of the line. Thus e (f) falls into the circle of radius $|pq|/(2\sqrt{2})$, disproving the conjecture. The new upper bound is now a circle with radius $\sqrt{2/17}\ |pq| = 0.343|pq|$ that is only about 3% better than $|pq|/(2\sqrt{2}) = 0.354|pq|$. Computing the angle qpf gives approx. 43.31° as an upper bound.

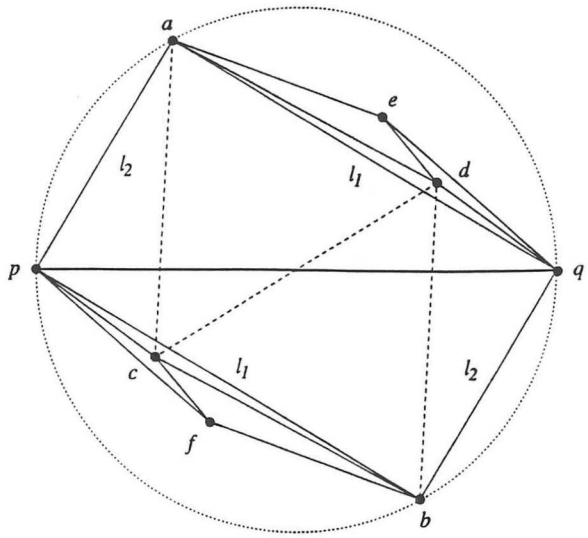


Figure 3: Disproving the conjecture of [DDMW].

Note that as for the diamond the positions of e and f are not fixed, thus they provide a stronger bound along the line aq (pb) for symmetric exclusion regions.

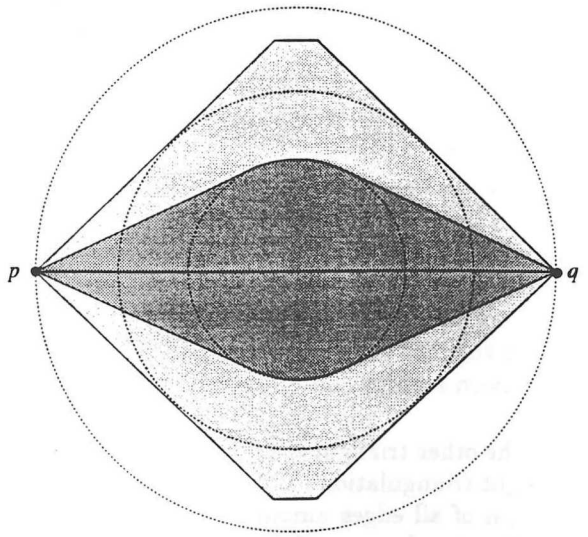


Figure 4: Summary for eye-shaped regions.

The dark shaded region of Figure 4 shows the largest known exclusion region so far while the light shaded region gives the possible area, bounded by the presented and other examples. The inner dotted circles have a radius of $|pq|/(2\sqrt{5})$ and $\sqrt{2/17}\ |pq|$, thus the angles on the left side are 26.57° and 43.31°.

Some questions are: If we assume the exclusion re-

gion to be symmetrical to both axes, is there a unique exclusion region with max. area? Does it have to be convex? What is the shape (are the possible shapes) of the exclusion region(s)?

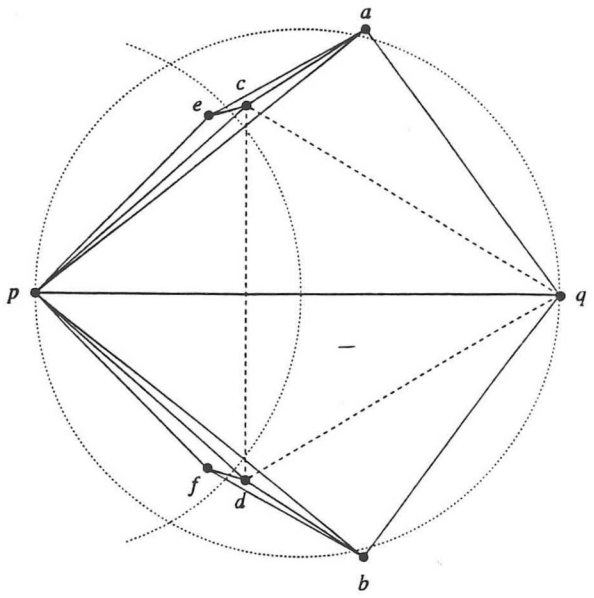


Figure 5: The circular sector has $< 43.87°$.

Another possible exclusion region is a circular sector centered at p with radius $|pq|/2$ which is just a part of the eye-shaped region (cf. Fig. 1(c) and Fig. 5). Actually Drysdale et al. [DRA] only use this region instead of the eye-shaped region: For each point $p \in S$ they arrange 30 circular sectors around p and test the possible edges emanating from p. They prove that a constant number of tests per point is sufficient, thus in linear expected time one can produce all $O(n)$ candidate edges.

Since the previous examples do not bound the circular sector we give a new construction, using the same ideas as for Figure 3. We start with a very similar example to Figure 2 but place all critical points in the left half. For the abstract we skip the detailed description but note that to improve the example we move the points a and b horizontal to the right, see Figure 5.

Computing the angle apq we have $43.87°$, which is slightly greater than the upper bound for the eye-shaped region. This difference gives rise to the question if there is a difference between the max. possible angle for the eye-shaped region and the 'half-sided' circular sector.

Another interesting exclusion region was studied by Das et al. [DJ], namely the diamond with angle apq of $\pi/8$ (see Figure 1(d)). For the GT this region is included in the eye-shaped region but interesting enough it is also an exclusion region for the MWT. Unfortunately it seems that up to now there are no algorithms using this fact for the MWT.

2.2 Inclusion Regions

As for exclusion regions there are several different inclusion regions for the GT. Figure 6 shows four of the 'more regular' regions. Note that all regions have the square with side length $|pq|$ in common, either symmetrical (a-c) or on one side (d). Despite the tricky proofs for exclusion regions it is almost obvious how to construct this inclusion regions for the GT. The simplest region (a) is just a circle of radius $\sqrt{2}|pq|/2$ centered at the middle of the edge pq.

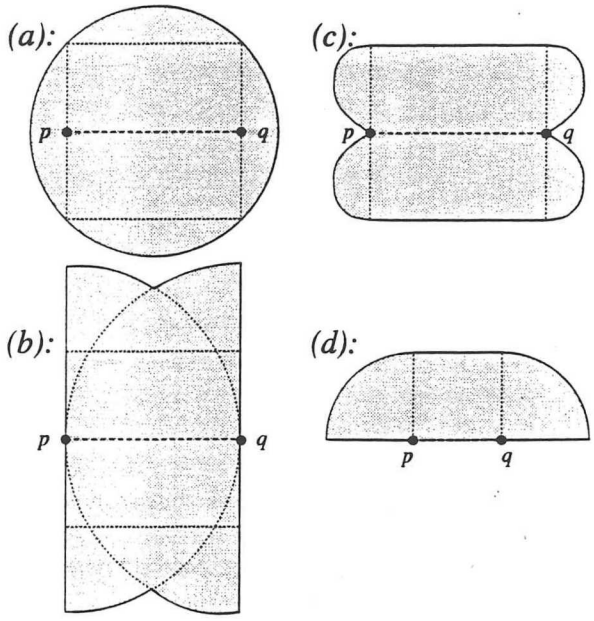


Figure 6: Inclusion regions for the greedy triangulation.

For region (b) we suppose the left and right side of the square to build the boundary and thus we have to add the two half circles of radius $|pq|$ centered at p and q, respectively. The most complicated region is (c) where we suppose the upper and lower edges of the square to bound it. To get a correct inclusion region we have to add four small regions on the left and right side. This 'extensions' can be described by $x = \pm\left(\frac{1}{2} + \frac{|y|\sqrt{3-4y^2-4|y|}}{2|y|+1}\right)|pq|$ for $y \in [-|pq|/2...|pq|/2]$, assuming that the origin is at the midpoint of the edge pq. We conjecture this example to be the minimum area inclusion region for the GT. For algorithmic purposes it is possible and might be more comfortable to approximate the additional area by four half circles of radius $|pq|/4$ and appropriate centers.

Region (d) is just a simple example for a non symmetric region. Note that from this examples it is easy to derive an infinite number of inclusion regions, either symmetrically or not. But there is one common type of edge in all of this regions: the light edge. Let us call an edge pq *light* if there is no other edge between two points of the point set S that crosses pq and is shorter than pq. Note that by definition the light edges form

a subset of the greedy edges. Although, in general, not all light edges occur in a minimum-weight triangulation recent results [Ke, YXY] imply that certain subsets of light edges always have to be there. Keil [Ke] shows that the MWT always includes the so-called $\sqrt{2}$-skeleton of P. This structure consists of all edges pq such that the two circles through both p and q and with diameter $\sqrt{2}\,|pq|$ are empty of points in S; see Figure 7. It is not

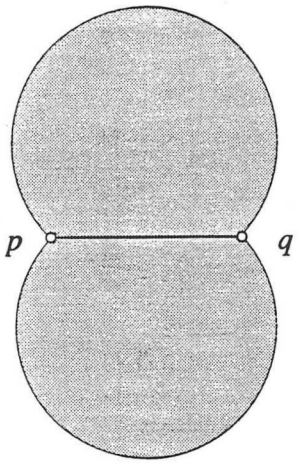


Figure 7: Keil's empty region defining the $\sqrt{2}$-skeleton.

difficult to verify that this condition forces $|pq|$ to be light. The $\sqrt{2}$-skeleton is a subset of the Gabriel graph (1-skeleton) which itself is known to be not part of the minimum-weight triangulation.

Yang et al. [YXY] prove that a condition stronger than lightness implies inclusion of pq in MWT: For any edge xy crossing pq, $|pq| \leq \min\{|px|, |py|, |qx|, |qy|\}$. By the quadrilateral inequality, xy then has to be the longest of the six edges involved. Hence pq is a light edge, and we obtain the region in Figure 8 whose emptiness implies $pq \in MWT$. With this inclusion regions it is e.g. possible to give lower bounds on the weight of the MWT of S.

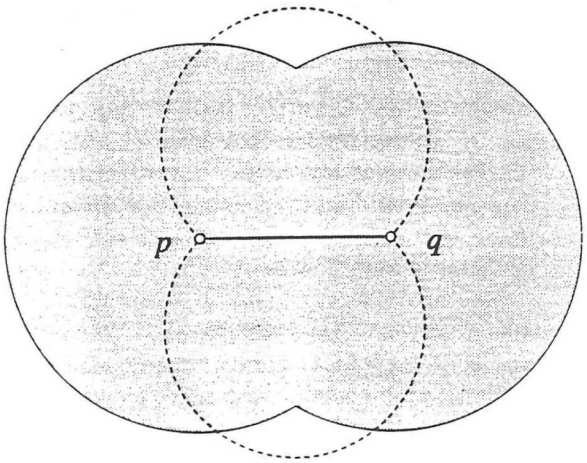


Figure 8: Empty region of Yang et al.

Very recent results give further important properties of light edges for the MWT. In [AART] the following is shown: The weight of the MWT of a point set S is bounded below by the sum of the length (weight) of all light edges of S. Note that, despite the fact that not all light edges occur in a minimum-weight triangulation in general, summing up the light edges provides a valid lower bound. This bound covers the bounds given by the above inclusion regions as the subsets of the MWT considered there are built of light edges only. Moreover they show: If a planar point set S admits a light triangulation (i.e. a triangulation consisting only of light edges) then this is the minimum-weight triangulation for S.

Thus beside new lower bounds for the MWT we have now a local edge-property which allows us to compute the MWT for special point sets in polynomial time.

References

[AART] O. Aichholzer, F. Aurenhammer, G. Rote, M. Taschwer, *Triangulations Intersect Nicely*, to appear in Proc. 11th Ann. Symp. Computational Geometry (1995).

[DDMW] M. Dickerson, R. L. Drysdale, S. McElfresh, and E. Welzl, *Fast greedy triangulation algorithms*, Proc. 10th Ann. Symp. Computational Geometry (1994), 211–220.

[DJ] G. Das and D. Joseph, "Which Triangulations Approximate the Complete Graph?", *Optimal Algorithms, Int. Symp. Proc.*, LNCS **401**, Springer-Verlag, Berlin 1989, pp. 168–192.

[DRA] R. L. Drysdale, G. Rote, O. Aichholzer, *A simple linear time greedy triangulation algorithm for uniformly distributed points*, Manuscript, 1994.

[GJ] M. Garey and D. Johnson, *Computers and Intractability. A Guide to the Theory of NP-completeness*, Freeman, 1979.

[Ke] M. Keil, *Computing a subgraph of the minimum weight triangulation*, Computational Geometry: Theory and Applications **4** (1994), 13–26.

[Kl] G. Klincsek, "Minimal triangulations of polygonal domains." *Ann. Discrete Math.* **9** 121–123.

[WA] Y. F. Wang and J. K. Aggarwal, "Surface reconstruction and representation of 3-D scenes." *Pattern Recognition* **19** (1986) 197–207.

[YXY] B.-T. Yang, Y.-F. Xu, and Z.-Y. You, *A chain decomposition algorithm for the proof of a property on minimum weight triangulations*, In: Algorithms and Computation, Proc. Conf., Beijing 1994, LNCS 834, Springer-Verlag, 1994, pp. 423–427.

Issues in Parametrization of Algebraic Curves *)

Franz Winkler
Institut für Mathematik and RISC-LINZ
Johannes Kepler Universität
A-4040 Linz, Austria
e-mail: winkler@risc.uni-linz.ac.at

The problem of rational parametrization of plane algebraic curves is best explained by an example. Suppose we consider the curve $\mathcal{C}_1$ in the complex plane, whose points are the zeros of the polynomial

$$f_1(x,y) = (x^2 + 4y + y^2)^2 - 16(x^2 + y^2).$$

See Figure 1.

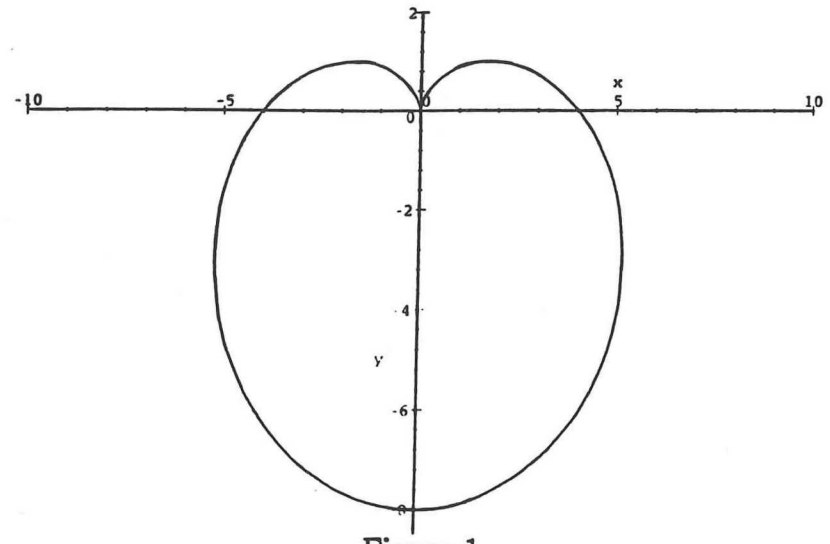


Figure 1

$\mathcal{C}_1$ has a double point at the origin $(0,0)$ as the only affine singularity. But if we move to the associated projective curve $\mathcal{C}_1^*$ defined by the homogeneous polynomial

$$f_1^*(x,y,z) = (x^2 + 4yz + y^2)^2 - 16(x^2 + y^2)z^2,$$

we see that the singularities of $\mathcal{C}_1^*$ are

$$O = (0:0:1), \quad P_{1,2} = (1:\pm i:0).$$

*) Supported by *Österr. Fonds zur Förderung der wissenschaftlichen Forschung*, Proj. POSSO, Nr. P9181-TEC. (ESPRIT BRA No. 6846).

$P_{1,2}$ is a family of conjugate algebraic points on C_1^*. All of these singularities have multiplicity 2, so the genus of C_1^* is 0, i.e. it can be parametrized. This also means that the affine curve C_1 is parametrizable.

In order to achieve a parametrization, we need a simple point on C_1^*. Intersecting C_1^* by the line $y + \frac{40}{9}x = 0$, we get of course the origin as a double intersection point. The other two intersection points are also rational, namely

$$Q_1 = (-36 : 160 : 1681), \quad Q_2 = (2916 : -12960 : 1681).$$

So now we construct the linear system L^2 of curves of degree 2, having $O, P_{1,2}$ and Q_1 as base points of multiplicity 1.

The affine version L_a^2 of L^2 is defined by

$$h_a(x, y, t) = tx^2 + ty^2 + x + (-\frac{1}{10}t + \frac{9}{40})y.$$

Now we determine the free intersection point of L_a^2 and C_1. The non–constant factors of $\mathrm{res}_x(f_2(x, y), h_a(x, y, t))$ are

$$\begin{aligned}
&y^2,\\
&1681y - 160,\\
&(430336t^4 - 94464t^3 + 58976t^2 - 5904t + 1681)y+\\
&\quad(-40960t^4 + 184320t^3 - 204800t^2 - 11520t + 12960).
\end{aligned}$$

The first two factors correspond to the affine base points of the linear system L^2, and the third one determines the y–coordinate of the free intersection point depending rationally on t.

Similarly, the non–constant factors of $\mathrm{res}_y(f_1(x, y), h_a(x, y, t))$ are

$$\begin{aligned}
&x^2,\\
&1681x + 36,\\
&(430336t^4 - 94464t^3 + 58976t^2 - 5904t + 1681)x+\\
&\quad(-9216t^4 + 62464t^3 - 141696t^2 + 108864t - 2916).
\end{aligned}$$

The first two factors correspond to the affine base points of the linear system L^2, and the third one determines the x–coordinate of the free intersection point depending rationally on t.

So we have found a rational parametrization of C_1, namely

$$\begin{aligned}
x(t) &= \frac{9216t^4 - 62464t^3 + 141696t^2 - 108864t + 2916}{430336t^4 - 94464t^3 + 58976t^2 - 5904t + 1681},\\
y(t) &= \frac{40960t^4 - 184320t^3 + 204800t^2 + 11520t - 12960}{430336t^4 - 94464t^3 + 58976t^2 - 5904t + 1681}.
\end{aligned}$$

Definition 1: In general we say that a curve C has a *rational parametrization*, iff there are (nontrivial) rational functions $x(t), y(t)$ such that the defining polynomial f of C vanishes on them,

$$f(x(t), y(t)) = 0. \qquad \square$$

The parametrizable curves are exactly the irreducible curves of genus 0, i.e. they achieve the upper bound on the number of singularities. If such a parametrization exists, then it is by no means unique, even if we require that the degrees of the rational functions involved are minimal. In fact, one of the main problems is to determine a minimal extension of the field of definition, in which a parametrization can be expressed.

For this purpose we assume that K is a computable field of characteristic 0, we call it the *field of definition*. $\overline{K}$ denotes the algebraic closure of K. The defining polynomial of the curve C that we want to parametrize has coefficients in K. The curve C itself, however, is considered to be a curve over $\overline{K}$. In the parametrization process we might have to extend K algebraically. We will ultimately construct a parametrization (if one exists) over some field $K(\gamma) \subseteq \overline{K}$, where γ is algebraic over K. Of course, we want to keep the degree of γ as low as possible.

We call a point, or a curve, or a system of curves *rational*, if they involve only coefficients from the field of definition K.

Points on algebraic curves occur only in full conjugacy classes. If we choose all the points in such a conjugacy class as base points of a linear system, then the coefficients of the system will still be rational.

Lemma 1: *Let $f(x, y, z)$ be a homogeneous polynomial in $K[x, y, z]$ defining an algebraic curve C in $\mathbb{P}^2(\overline{K})$. Let $(a_1(\alpha) : a_2(\alpha) : a_3(\alpha))$ be a point of multiplicity r on C, α algebraic over K with minimal polynomial $p(t)$, and $a_1, a_2, a_3 \in K[t]$. Then for every conjugate β of α, also the point $(a_1(\beta) : a_2(\beta) : a_3(\beta))$ is a point of C.* $\square$

Definition 2: If $p(t) \in K[t]$ is irreducible and $a_1, a_2, a_3 \in K[t]$ with $p \not| \gcd(a_1, a_2, a_3)$, then we call

$$\{ (a_1(\alpha) : a_2(\alpha) : a_3(\alpha)) \,|\, p(\alpha) = 0 \}$$

a *family of conjugate algebraic points.* $\square$

Lemma 2: *Let L be a rational linear system of curves of degree d. Let $P_\alpha = \{(a_1(\alpha) : a_2(\alpha) : a_3(\alpha) | p(\alpha) = 0\}$ be a family of conjugate algebraic points. Then also the subsystem $\hat{L}$ of L, having all the points in P_α as base points of multiplicity r, is a rational linear system of curves.* $\square$

So we can proceed as in the introductory example. An algorithm along these lines is presented in [SeW 91]. The only step which might force us to extend the field of definition is the determination of simple points on the curve C.

From the work of Nöther [Nöt 1884] and Hilbert and Hurwitz [HiH 1890] we know that it is possible to parametrize any curve C of genus 0 over the field of definition K, if $\deg(C)$ is odd, and over some quadratic extension of K, if $\deg(C)$ is even. An algorithm which actually achieves this optimal field of parametrization is presented in [Sendra,Winkler 94]. Moreover, if the field of definition is $\mathbb{Q}$, we can also decide if the curve can be parametrized over $\mathbb{R}$.

The resulting parametrization algorithm is of polynomial complexity if we assume that the curve C has only ordinary singularities [MSW 94].

References

[HiH 1890] D. Hilbert, A. Hurwitz, "Über die Diophantischen Gleichungen vom Geschlecht Null", *Acta math.* **14**, 217–224 (1890)

[MSW 94] M. Mňuk, J.R. Sendra, F. Winkler, "On the Complexity of Parametrizing Curves", manuscript (1994)

[Nöt 1884] M. Nöther, "Rationale Ausführung der Operationen in der Theorie der algebraischen Functionen", *Math. Annalen* **23**, 311–358 (1884)

[SeW 91] J.R. Sendra, F. Winkler, "Symbolic Parametrization of Curves", *J. Symbolic Comp.* **12**, 607–631 (1991)

[SeW 94] J.R. Sendra, F. Winkler, "Optimal Parametrization of Algebraic Curves", RISC-Report 94-65, J. Kepler Univ. Linz, Austria (1994)

linear progr. $\#J \leq (\frac{s}{2})^n$ ↗ # cells ~~Matrices~~ Matrices of bandwidth k → factor of $(\frac{k}{n})^n$

Parametric Linear and Quadratic Optimization by Elimination

Volker Weispfenning
Universität Passau, D-94030 Passau, Germany
Tel: 0851-509-3120, Fax: 0851-509-1802
e-mail: weispfen@alice.fmi.uni-passau.de

Abstract. We present a new elimination method for linear, quadratic, and fractional linear optimization involving parametric coefficients. In comparison to the classical Fourier-Motzkin method (compare [Dantzig, Eaves & Rothblum, Williams]) (that we show to be of doubly exponential worst-case complexity) our method is singly exponential in the worst case. For problems with additive parameters only the method is worst-case optimal. For parameter-free inputs the algorithms run in polynomial space. Moreover the complexity decreases significantly for sparse inputs. The method can also be used to solve systems of parametric linear inequalities by a recursive computation of intervals for all variables. It generalizes ideas in [Weispfenning, Loos & Weispfenning]. Examples computed in the REDUCE–implementation [Kappert, Sturm] confirm the superiority of the method over Fourier-Motzkin and its applicability to problems of interesting size.

s constraints
n variables

weakly parametric.
→ parameters in the right-hand-side

References

[Dantzig] G. B. Dantzig, Lineare Programmierung und Erweiterungen, Springer-Verlag, 1966.

[Eaves & Rothblum] B.C. Eaves, U.G. Rothblum, Dines-Fourier-Motzkin quantifier elimination and an application of corresponding transfer principles over ordered fields, Mathematical Programming **53**, North-Holland, 1992, pp. 307-321.

[Kappert] M. Kappert, Diplomarbeit, Universität Passau, 1995.

[Loos & Weispfenning] R. Loos, V. Weispfenning, Applying linear quantifier elimination, The Computer Journal 36 (1993), pp. 450-462.

[Sturm] Th. Sturm, REDUCE-implementation of the algorithms in [Loos & Weispfenning], Universität Passau, 1993.

[Weispfenning] V. Weispfenning, The complexity of linear problems in fields, J. Symb. Comp. 5 (1988), pp. 3-27.

[Williams] H.P. Williams, Fourier's method of linear programming and its dual, American Mathematical Monthly 93 (1986),), pp. 681-695.

Intersection Algorithms: A Comparative Study *Extended Abstract*

in 2d and 3d

Eugen Ardeleanu*
Research Institute for Symbolic Computation
Johannes Kepler University Linz, A-4040 Austria
Sabine.Stifter@risc.uni-linz.ac.at
Eugen.Ardeleanu@risc.uni-linz.ac.at

Intersection tests are ubiquitous in computational geometry and in a wide range of applications to robotics, computer graphics, simulation, etc. We review here some of the most important ones, pointing out the basic strategies and time and space complexity. A hybrid $2D$ Roider method mixing some of the most efficient techniques is suggested.

There are wellknown optimal $\mathcal{O}(n \log n)$ algorithms for finding the convex hull of n points in the plane:

- *Graham scan*: a local method method based on $< \pi$ *property of interior angles in a convex polygon* that determines the convex hull in linear time after the vertices were sorted;
- *Jarvis march*: a method based on the property according to which all the points lie on one side of each edge of the convex hull, $\mathcal{O}(n^2)$ worst-case complexity, but *linear* if the convex hull has a constant number of edges;

*Supported by the Austrian National Science Foundation (FWF) project no. P8620-TEC.

- *Quickhull*: a *divide-and-conquer* method whose *merge* step is based on calculation of *supporting lines* (lines passing through a vertex of the convex polygon such that the interior of the polygon lies entirely on one side of the line).

The idea of supporting lines can be further exploited to obtain an on-line $\mathcal{O}(n \log n)$ time, real-time ($\mathcal{O}(\log n)$ update time) algorithm for computing the convex hull of n points in the plain, and furthermore, through Kallay's *beneath-beyond method*,to obtain an on-line $\mathcal{O}(n^{\lfloor d/2 \rfloor + 1})$ time algorithm for computing the convex hull of n points in d-dimensional space.

We treat first convex planar figures. First of all, there are optimal $\mathcal{O}(n \log n)$ algorithms for finding the intersections of n segments in the plane based on the *sweep-line method*. As a consequence, the intersection of n half-planes can be computed in $\mathcal{O}(n \log n)$ time, and this is optimal. (Hence the kernel of a polygon can be found in $\mathcal{O}(n \log n)$ time; there exist however optimal linear time algorithms.) Furthermore, the common intersection of n convex k-gons can be found in $\mathcal{O}(nk \log n)$ time. Similar sweep-line techniques are used in geometric searching problems:

Point-location in an n-vertex planar subdivision can be performed in $\mathcal{O}(\log n)$ optimal query time using $\mathcal{O}(n^2)$ storage and $\mathcal{O}(n^2)$ preprocessing time.

Passing now to 3-dimensional intersection tests we mention Muller-Preparata method, which computes the intersection of two convex polyhedrons in $\mathcal{O}(n \log n)$ time if one knows a point in the intersection (*witness to intersection*. The method is based on duality. We remark here that such a witness can be found via the Roider method. Reciprocally, the original Muller-Preprata algorithm for finding a witness to intersection (or reporting disjointness), simplified by Dyer, and which is based on minimizing a certain function representing the distance between relevant parts of the boundary of the two polyhedrons, will be used in our design of a hybrid Roider method.

The standard formulation of the linear programming problem in 2 variables is similar to the problem of intersecting n planes (giving the *feasible region*):

$$\min(ax + by) : a_i x + b_i y + c_i \leq 0, i = 1, ..., n$$

The *objective function* defines a family of parallel lines $ax + by + c$, where c is a real parameter. The lines of this family that support the feasible

$h = xB$ $B = \pm B^T$ $xBx^T = 0$ Kegelschnitt

pol $p_0 \rightarrow$ Polare $p_0 B x^T = 0$

region pass through the (optimum) vertices corresponding to the extreme values of the objective function. Since support lines of a convex polygon can be constructed in $\mathcal{O}(\log n)$ time, the 2-variable linear programming can be transformed to half-planes intersection, which can be solved in $\mathcal{O}(n \log n)$ time.

Building on the observation that the construction of the feasible region is not necessary to solve the linear programming problem, the Megiddo-Dyer optimal $\mathcal{O}(n)$ algorithm (adaptable also to 3 and more dimensions) discards redundant constraints (relative to the half-plane intersection) and constraints not containing an optimum vertex by applying the linear transformation:

$$ax + by \mapsto y, x \mapsto x$$

of the objective function into one of the coordinates

$$\min y : (a_i - \frac{a}{b} b_i)x + \frac{b_i}{b} y + c_i \leq 0, i = 1, ..., n$$

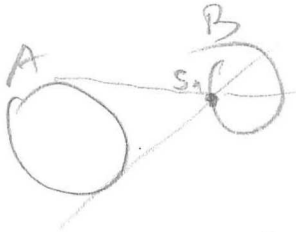
It can easily be shown that the problem of *linear separability* of two sets of points can be stated as a linear programming problem, being thus solvable in linear time in 2 or 3 dimensions.

There is a close relationship between finite sets of points and finite sets of hyperplanes: for each finite set of points P in a d-dimensional euclidian space E^d there is a set of hyperplanes H in a bijective correspondence such that certain statements about P are true if and only if their corresponding dual statements about H are true, e.g. for d=2, three points in P are colinear if and only if the corresponding lines in H are concurrent. The correspondence is realized through projective transformations. Thus let B be a $(d+1) \times (d+1)$ matrix of the projective group on $d+1$ homogeneous coordinates. Then the transformation mapping points to points and lines to lines in the plane

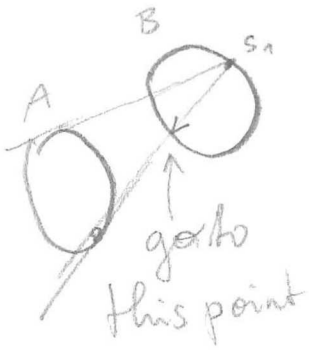
$$\mathbf{h} = \mathbf{x}B$$

can be interpreted in E^3 as transforming a line (represented by x) to a plane with h as normal passing by the origin. If D is the inverse duality transform then preservation of incidence implies $D = k(B^{-1})^T$ and the involutory condition $B = \pm B^T$.

Roider Method for Intersection Detection



- *A general method for finding supporting lines*: To find the supporting lines from a point p_1 to the unit parabola construct first the polar $D(p_1)$ of p_1 with respect to the parabola and then construct the polars of the intersection points $D(p_2)$.



- *Point-location test*: A point $p_1(p_3)$ lies "outside (inside)" the conic depending upon its corresponding polar $D(p_1)(D(p_3))$ intersecting (or not) the conic.

A simple and elegant method for testing intersection of convex planar polygons was discovered by Roider and Stifter, and was subsequently generalized to 3-dimensional general convex sets and *collision detection.*

We will now outline a hybrid Roider method mixing the basic idea of the original and duality and linear programming following a divide-and-conquer strategy.

This reduction of the problem is especially useful in dealing with objects whose boundaries consist of "curve segments". In this case the Roider method can serve as a preprocessing step determining the two segments of the boundary that are "closest" to each other. It then suffices to determine whether these two segments intersect or not. For this purpose, the minimum of the unimodal function can be computed. (For objects with boundaries not composed from segments (i.e. defined by a single equation) this kind of reduction might not be as useful).

B convex polygon

→ may not terminate

From the complexity viewpoint, whether or not we can improve upon the pure Roider method, it is crucial which specific method is used for computing the minimum of the unimodal function; e.g. using an iterative method based on the construction of lines normal to the boundaries of the object renders a less efficient technique than finishing the search by the pure Roider method.

These variations of the Roider method concern only the technique of determining a witness to disjointness or intersection, and not the properties of such a witness. (Note that the computed witness to disjointness will always lie between the last two intersection points found by the Roider method when applied until termination.) Thus, the witness to disjointness found by the hybrid Roider method is especially a point for which the tangents to A separate A and B. This is the prerequisite for using the results provided by the hybrid Roider method for collision (and not only intersection) test.

Let T be a triangle, is it isosceles?

Teresa Gómez-Díaz
Laboratoire d'Arithmétique, Calcul formel et Optimisation *
123 Av. Albert Thomas, 87060 Limoges Cedex

Abstract

We study mechanical geometry theorem proving for the example proposed as title. In this way we present a different method to look this kind of problems. This method is an application of dynamic evaluation.

Introduction

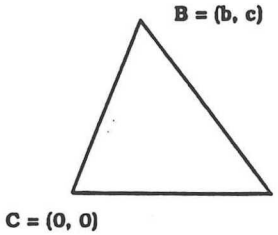
Do you think that this is a naive example? This is one of the examples (the simplest one) used in [CY] to ilustrate some problems in different formulations for mechanical geometry theorem proving: some methods can prove that *every triangle is isosceles* [CY], thus it is a good example to examinate.

There are two main methods in geometry theorem proving, they use either Gröbner basis (see [BCK] for a presentation of the method and further references) or characteristic sets (see [Ch, CY, BCK]).

Here we present a different method to study this problem, it is based on *dynamic evaluation* [DDD, DR]. We explain this method applied to the proposed example, and show its flexibility to deal with hypothesis.

1 Let T be a triangle

We would like to build a triangle, and to study some of its properties. To build the figure we start by choosing a coordinate system, and in this way we can represent the triangle ABC with three symbols – a, b, c – corresponding to its vertices.



These symbols represent elements in a field. We call this symbols *parameters*.

To study properties about figures we can express these properties in algebraic way. For example, in order to study whether the triangle ABC is isosceles, we examine the equation

$$a^2 = (b-a)^2 + c^2$$

*URA CNRS 1586. e-mail: tgomez@marie.polytechnique.fr

This equality represents $\overline{AC} = \overline{AB}$. In fact we will do a bit more:

to study in which cases, depending on parameters, this equality is true or false.

Let us analyse the triangle further. We have choosen *any* triangle, that is we have no constraints over the vertices. But sometimes we are interested in work over a triangle that is not a point or a segment. We can express this in the following way

a and c *must be different to* 0.

Or in other words, we do not admit from thir point the zero value for parameters a and c, and so we have then *constraints* over the parameters. In the next sections we will see how to deal with constraints.

2 Dynamic constructible closure

Previous section shows that we need to know how to do computations with parameters, to know their possible values at any moment, to be able to impose constraints over they, and moreover, if it is possible to impose a constraint. Finally we need to answer any *equality test* over these parameters.

All this is possible in the *dynamic constructible closure* [Go]. It is a program in the computer algebra system Axiom [JS]. It uses over the *dynamic evaluation* principle [DDD, DR].

To use the dynamic constructible closure program one needs a ground field **K**. For the kind of problems proposed here, we could take any field of zero characteristic. One can build its dynamic constructible closure in this way:

```
CL:= DynamicConstructibleClosure(K)
```

This provides a function to introduce parameters:

```
newElement: Symbol -> CL
```

In addition, there are two functions to impose or forbid values for the introduced parameters, i.e. imposing constraints over the parameters:

```
areEqual: (CL,CL) -> Boolean
areDifferent: (CL,CL) -> Boolean
```

The boolean result of these operators says whether a new constraint is (or is not) compatible with the previous ones. Constraints over parameters express the possible values for a parameter. At the moment of their introduction, they are reduced into a standard form in a recursive way. Suppose that we have a parameter a over the ground field **K**. At every moment, constraints over a are in one of following forms:

- `anyElement`: there is no constraint over a (this means that a can take any value)
- `algebraic`: there is a constraint of type $P(a) = 0$ with P monic univariate polynomial of positive degree with coefficients in **K** (this means that a can take as value any zero of P).
- `exception`: there is a constraint of type $P_1(a) \neq 0$ and ... and $P_k(a) \neq 0$ with $P_1, \ldots, P_k$ monic univariate polynomials of positive degree with coefficients in **K** and pairwise coprime (this means that a can take any value different to any zero of any of the polynomials $P_1, \ldots, P_k$)

In addition, when the characteristic of the ground field is zero, we can suppose that polynomials $P, P_1, \ldots, P_k$ above are square-free.

The main point now is that parameters can take different values, so that it is in general impossible to answer `true` or `false` to an *equality test* over parameters. When both answers are possible, it is essential to distinguish the values of the parameters corresponding to `true` from the values corresponding to `false`. This is called a *splitting*.

The principal algorithm used to answer an equality test is gcd over polynomials in one variable and coefficients in a field. It is also used in the introduction of new constraints. Dynamic evaluation takes care of the different branches that are possible in splitting points.

3 Is T isosceles?

We will consider two examples about our triangle in the dynamic constructible closure. The first one corresponds to the title:

Let T be a triangle, is it isosceles?

For that we need to write:

```
RN:= Fraction Integer
CL:= DynamicConstructibleClosure(RN)

triangle1():Boolean ==
  a:CL:= newElement('a)
  b:CL:= newElement('b)
  c:CL:= newElement('c)
  (a**2 = (b - a)**2 + c**2)$CL

allCases(triangle1)
```

In this way we say that our ground field is the field of rational numbers. We build its dynamic constructible closure and we write a little function with the computation to be done. This consists of introducing first the parameters and then an equality test for the problem. Finally we call this function with `allCases` — the procedure that implements dynamic evaluation.

We get the answer:

```
[value is true in case c = 0 and b = 0 and a = 0,

 value is false in case c /= 0 and b = 0 and a = 0,
                         2    2
 value is true in case c  + b  = 0 and b /= 0 and a = 0,
                        2    2
 value is false in case c  + b  /= 0 and b /= 0 and a = 0,
                              2
 value is true in case c = 0 and b  - 2a b = 0 and a /= 0,
                                   2
 value is false in case c /= 0 and b  - 2a b = 0 and a /= 0,
                         2    2                 2
 value is true in case c  + b  - 2a b = 0 and b  - 2a b /= 0 and a /= 0,
                        2    2                  2
 value is false in case c  + b  - 2a b /= 0 and b  - 2a b /= 0 and a /= 0]

Time: 2.26 sec
```

← general case

Note that $a^2 = (b-a)^2 + c^2$ is equivalent to $c^2 + b^2 - 2ab = 0$. Also note that in this answer several cases correspond to a segment or a point. In the other cases the theorem is *true* if $a \neq 0$ and $b^2 - 2ab \neq 0$ and $c^2 + b^2 - 2ab = 0$ (that is, for two precise values of c depending of a and b) and the theorem is *false* if:

- $a \neq 0$ and $b^2 - 2ab = 0$ and $c \neq 0$
- $a \neq 0$ and $b^2 - 2ab \neq 0$ and $c^2 + b^2 - 2ab \neq 0$ (*general case*)

The function `areDifferent` allows us to avoid cases corresponding to triangles reduced to a point or a segment. For example:

```
triangle2():Boolean ==
  a:CL:= newElement('a)
  b:CL:= newElement('b)
  c:CL:= newElement('c)
  if areDifferent(a,0) and areDifferent(c,0)
  then (a**2 = (b-a)**2 + c**2)$CL
  else "failed"

allCases(triangle2)
```

with answer:

```
[value is false in case c /= 0 and b  - 2a b = 0 and a /= 0,
                        2   2               2
 value is true in case c  + b  - 2a b = 0 and b  - 2a b /= 0 and a /= 0,
                             2   2                2
 value is false in case c /= 0 , c  + b  - 2a b /= 0 and b  - 2a b /= 0 and a /= 0]

Time: 2.71 sec
```

4 Method

To summarize, the method proposed using Dynamic constructible closure consists of the following steps:

1. Introduce symbols corresponding to the points in the figure by `newElement`.

   In particular we do not distinguish between *dependent or independent* variables, at it is proposed in other methods [BCK, Ch, CY]. Nevertheless the order in the introduction of parameters will be important in computations.

2. Introduce hypothesis with `areEqual` or `areDifferent`.

   Note that the set of hypothesis can be empty. This is not possible in methods that use Gröbner basis or characteristic sets because they work over the ideal generated by the polynomials in hypothesis.

   `areEqual` and `areDifferent` allows us to avoid some types of figures that can be consider as *degenerate* cases, and also study only some special cases. Moreover, `areDifferent` allows us to work with the constraint *must be different* without introducing new variables, contrasting with methods using Gröbner basis.

   The program works automatically over the introduced constraints in order to be presented as in section 2. If hypothesis are not consistent our system will detect it in this step.

3. Introduce the conclusion to be proved as an equality test.

   Our system will work automatically in order to differenciate the cases where conclusion is true and where it is false.

   Finally we note that factorization is not done but gcd computations instead.

Conclusion

We have presented here a different method for studying mechanical geometry theorem proving. This method is based on the principle of dynamic evaluation.

We have studied the example of the isosceles triangle and we have showed that this method is able to avoid some weak points appearing in other methods, as for example the distinction between dependent or independent variables or the question of degenerate cases.

Finally we have presented this method in relation with the kind of problems we are able to solve using dynamic constructible closure. But the method derived from dynamic evaluation is general and it will be able to work also in other frameworks, for example dynamic real algebraic closure, where questions related with order will be possible [DGV].

References

[BCK] B. Buchberger, G.E. Collins, B. Kutzler. – *Algebraic Methods for Geometry Reasoning.* Ann. Rev. Comput. Sci. 3, (1988).

[Ch] S. C. Chou – *Mechanical Geometry Theorem Proving* Reidel Publishing Company (1988).

[CY] S. C. Chou, J. G. Yang. – *On the Algebraic Formulation of Certain Geometry Statements and Mechanical Geometry Theorem Proving.* Algorithmica 4, p. 237–262 (1989).

[DDD] J. Della Dora, C. Dicrescenzo, D. Duval. – *About a New Method for Computing in Algebraic Number Fields.* Eurocal'85, vol.2, Springer Lecture Notes in Computer Science 204, ed. G. Goos, J. Hartmanis, p. 289–290 (1985).

[DGV] D. Duval, L. González-Vega. – *Dynamic Evaluation and Real Closure.* To appear in Mathematics and Computers (transactions of IMACS Conference, Juin 1993).

[DR] D. Duval, J.-C. Reynaud. – *Sketches and Computation (Part I): Basic Definitions and Static Evaluation* et *(Part II): Dynamic Evaluation and Applications.* Mathematical Structures in Computer Science vol. 4, pp. 185-271 (1994).

[Go] T. Gomez-Diaz. – *Quelques applications de l'évaluation dynamique.* Thesis, Université de Limoges (1994). Available from Atelier National de Reproduction des Thèses, Université de Grenoble 2.

[JS] R. D. Jenks, R. S. Sutor. – *Axiom, The Scientific Computation System.* NAG, Springer-Verlag (1992).

Numerical problems in the plane Roider method

Sabine Stifter, Elena Tomuta

RISC
Johannes Kepler University
Schloß Hagenberg
A-4232 Hagenberg, Austria

Extended Abstract

1 Introduction

The paper is motivated by experiments done with a floating point implementation of the Roider method.

The method was invented for deciding collision of two planar convex objects. Therefore the speed of the method is important and it is meaningfull to try to handle the numerical problems appearing in a floating point implementation, rather then to use rational arithmetic that would eliminate these problems but would also drastically increase the time complexity of the algorithm.

The method proceeds by constructing a sequence of points on one object, such that after a finite number of steps one either finds a point in the intersection — if the objects intersect — or a witness to disjointness — if the objects are disjoint. For details on the method see [STIFTER 1988].

The algorithm is formulated in terms of required subprocedures, namely, intersecting an object with a line and computing the touching points from a given point to an object.

We investigate the impact of the numerical error of these subprocedures on the precision of the Roider method.

For this we first introduce some concepts necessary for handling the numerical aspects of the algorithm.

We give bounds for the error in one Roider step.

The information supplied by these bounds can be used to find sufficient conditions for the termination of the Roider method.

Finaly we suggest how the results we obtained can be further extended to completely characterize the numerical behaviour of the method.

In the following we assume the reader is familiar with the basic strategy and terminology of the Roider method as given in [STIFTER 1989].

2 Notations and Terminology

In the follwoing, unless otherwise specified, $\mathcal{A}$ and $\mathcal{B}$ are convex, compact objects in the plane. Let $\epsilon_I^{\mathcal{A}}$ and $\epsilon_I^{\mathcal{B}}$ be the errors for the intersection operations for the objects $\mathcal{A}$ and $\mathcal{B}$. That is, if the procedure $intersect-line(\mathcal{A}, P1, P2)$ returns the intersection points S_1 and S_2, then there exist S_1' and S_2' in $\delta\mathcal{A}$ such that

$$dist(S_i', S_i) \leq \epsilon_I^{\mathcal{A}}, \quad i = 1, 2$$

In an anlogous way we define $\epsilon_T^{\mathcal{A}}$ as the absolute error of the touching point operation *touch* for the object $\mathcal{A}$. For the exact definitions of $intersect-line$ and *touch* see [STIFTER 1989], pp. 9-10.

We call *exact Roider sequence for* $\mathcal{A}$ the sequence $P_0, \ldots, P_i, \ldots$ constructed by the Roider method on the object $\mathcal{A}$, by exact elementary operations.

We call *computed Roider sequence for* $\mathcal{A}$ the sequence $\bar{P}_0, \ldots \bar{P}_i \ldots$ constructed on the object $\mathcal{A}$ by the Roider method with approximate elementary operations.

Given a point $P \in \delta\mathcal{A}$, the next point in the exact Roider sequence starting with P is denoted by $\mathcal{R}(P)$. The next point in the approximate Roider sequence is denoted by $\bar{\mathcal{R}}(P)$.

For given $\mathcal{A}$, $\mathcal{B}$ we define an associated coordinate system $C(\mathcal{A}, \mathcal{B})$ as follows:

- if $\mathcal{A} \bigcap \mathcal{B} = \phi$, the x-axis of C is a separating line for $\mathcal{A}$ and $\mathcal{B}$ and the points in $\mathcal{A}$ have positive y-coordinates.
- if $\delta(\mathcal{A}) \bigcap \delta(\mathcal{B})$ consists of two or more points, then the x-axis of C is the line determined by those points; the points in $\mathcal{A} - \mathcal{B}$ have positive y-coordinates.
- if $\delta(\mathcal{A}) \bigcap \delta(\mathcal{B})$ consists of one point then the x-axis of C is a touching line for $\mathcal{A}$ passing through that point.

For a given pair $\mathcal{A}$, $\mathcal{B}$, an extension of $\mathcal{B}$, denoted by $\bar{\mathcal{B}}$ is an x-extension of $\mathcal{B}$ w.r.t. an associated system of coordinates $C(\mathcal{A}, \mathcal{B})$.

3 Bounds for the error in one Roider step

In this section we give an upper bound for the error in a Roider step.

For $\mathcal{A}$ and $\mathcal{B}$ we define $\mathcal{L}(\mathcal{A})$ the lateral region of $\mathcal{A}$ w.r.t. $\mathcal{B}$ as the union of two curves l and r such that

- l and r are maximal connected subsets of $\delta\mathcal{A}$;
- the upper endpoint of l is the left exterior touching point of $\mathcal{A}$ and $\mathcal{B}$; the lower endpoint of l is the left interior touching point of $\mathcal{A}$ and $\mathcal{B}$ in case the objects are disjoint and the leftmost intersection point in case the objects intersect;
- the upper endpoint of r is the right exterior touching point of $\mathcal{A}$ and $\mathcal{B}$; the lower endpoint of r is the right interior touching point of $\mathcal{A}$ and $\mathcal{B}$ in case the objects are disjoint and the rightmost intersection point in case the objects intersect;

The lateral region of $\mathcal{B}$ w.r.t. $\mathcal{A}$ is defined analogously.

In the above definition the notions of *left*, *right*, *upper*, *lower* are formulated w.r.t. an associated system of coordinates.

Lemma 3.1 *For all points $i \geq 1$ the points P_i in an exact Roider sequence on $\mathcal{A}$ lie in $\mathcal{L}(\mathcal{A})$. The touching points from the points in the Roider sequence to $\mathcal{B}$ lie in the $\mathcal{L}(\mathcal{B})$.*

This lemma enables us to only consider cases when the points in the (computed and exact) Roider sequence lie on the lateral region. In the following, unless otherwise specified, this will be the case.

Lemma 3.2 *Let $\mathcal{A}$ and $\mathcal{B}$ be more then $\epsilon_T^{\mathcal{A}}$ apart, $\bar{P}$ and P points, with $P \in \mathcal{L}\mathcal{A}$ and $d(P, \bar{P}) \leq \epsilon_I^{\mathcal{A}}$. Let $S = \mathcal{R}(P)$ and $\bar{S} = \bar{\mathcal{R}}(\bar{P})$.*

Then the following holds:

$$d(S, \bar{S}) < \epsilon_I^{\mathcal{A}}(1 + \frac{1}{\sin(\alpha)})$$

where α is the angle minimum angle between the interior touching lines to $\mathcal{A}$ and the line defined by one touching point and the opposite vertex of $\bar{B}$.

A similar lemma holds for the case of intersecting objects, but here the role of the inner touching points is played by the intersection points.

4 Suggestions for future work

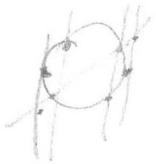
Our intention is to use these bounds to give sufficient conditions for termination of the approximate Roider method. Furthermore, the non-terminating cases should be characterized. In practice some of the non-terminating cases arise if the objects differ greatly in size. These cases however can be solved by swapping the objects and performing the Roider method again. Other non-terminating cases arise if the objects are "close to the critical cases" given in [STIFTER 1989] for the exact algorithm. These can be solved by returning from the algorithm with the answer "objects not more then <some value> apart". The problem is to find this apropriate value in terms of the errors of the elementary operations.

References

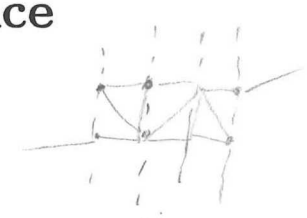
[STIFTER 1988] Stifter S., 1988: A Medley of Solutions to the Robot Collision Problem in Two and Three Dimensions, Ph. D. thesis, Univ. Linz, RISC-Linz, series no. 88-12.0

[STIFTER 1989] Stifter S., 1989: The Roider Method: A Method for Static and Dynamic Collision Detection, Advances in Computer Research, vol. 6, Issues in Robotics and Non-Linear Geometry, Gh. Hoffmann (ed.), JAI Press, 1992, pp 31-52

canonical ! encoding



Codifying the Topology of Graph Embeddings on a Surface
(extended abstract)



F. Santos †,‡

Abstract: We consider the problem of codifying the topology of a certain class of objects which includes graph embeddings, Voronoi diagrams and real algebraic plane curves. By codifying we mean giving a *canonical* combinatorial code associated to the object so that two such objects have the same code if and only if they are topologically equivalent.

Throughout this paper we will use the word *surface* meaning a topological manifold of dimension 2, compact and connected. Our methods and results will also concern the Euclidean plane but, for convenience, the plane will just be considered a sphere with one distinguished point (the infinity point).

By a *diagram* $\mathcal{D}$ on a surface S we will mean a subset $\mathcal{D} \subset S$ homeomorphic to a graph $G_{\mathcal{D}}$. Thus, our diagrams are almost the same thing as *graph embeddings*, except that we are going to forget the graph structure of $G_{\mathcal{D}}$ and study the subset $\mathcal{D}$ with its intrinsic topological graph structure. In this graph structure, *vertices* of $\mathcal{D}$ are the points of $\mathcal{D}$ where $\mathcal{D}$ is not locally a line and the *edges* are the connected components of $\mathcal{D} \setminus V_{\mathcal{D}}$, where $V_{\mathcal{D}}$ is the set of vertices. Edges will usually be homeomorphic to lines, but they can be homeomorphic to circles. Loops and multiedges between vertices are permitted. The connected components of $S \setminus \mathcal{D}$ will be called *faces*.

Two diagrams $\mathcal{D}_1$ and $\mathcal{D}_2$ on surfaces S_1 and S_2 will be said to be *topologically equivalent* (or just *equivalent*) if there is an homeomorphism $h : S_1 \longrightarrow S_2$ such that $h(\mathcal{D}_1) = \mathcal{D}_2$. If the surfaces S_1 and S_2 are *orientable*, then we will normally consider them with a certain orientation and require the homeomorphism h to be *orientation preserving*.

Special mention need diagrams in the plane. A diagram in the plane will be a graph embedded in the plane which may have some edges going to infinity. With this definition diagrams include Voronoi diagrams, triangulations (where the diagram itself is the 1-skeleton of the triangulation) and real algebraic plane curves. Two diagrams $\mathcal{D}_1$ and $\mathcal{D}_2$ in the plane are equivalent if there is an orientation preserving homeomorphism from the plane into itself sending $\mathcal{D}_1$ to $\mathcal{D}_2$. This is the same as saying that they are equivalent as diagrams in the sphere and the homeomorphism from the sphere into the sphere sending one to the other fixes the distinguished point.

If we are interested in recognizing, comparing or cataloging the topological types (up to equivalence) of diagrams, we will need two different things. First, we need a data structure that captures all the topological properties of the diagram and, second, we need a procedure to decide whether two such data structures represent equivalent diagrams. This is so because, in building the data structure associated to a diagram, some ambiguities will normally have to be solved by a certain choice not depending on the topology itself. We can call this two tasks *encoding* and *equivalence testing*, respectively.

Our purpose here is twofold. First we will describe a general way to encode the topology of any diagram and then we will give a procedure to choose, in quadratic time in the most interesting cases,

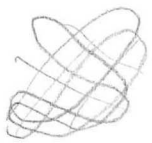
---

† Departamento de Matemáticas, Estadística y Computación. Facultad de Ciencias. Universidad de Cantabria. 39071-SANTANDER. Spain. E-mail: santos@matsun1.unican.es

‡ Partially supported by CICyT PB 92/0498/C02/01 (Geometría Real y Algoritmos and Esprit/Bra 6846 (Posso).

implementation: $X_{topo} + (\#\text{cells})^2 \log n$ ($X_{topo} \hookrightarrow n^{15}$)

$= \text{poly}(\text{degree})$ ($\hookrightarrow O(d^3)$)



n double points can be realized with degree $2n$

a special representative from all the possible *equivalent codes* representing the same topological type. The existence of such a *canonical code* makes equivalence testing trivial.

Graph embeddings are usually represented on a computer by a list of edges with, for any given edge e some information saying which other edges are incident to the end-points of e and which edge follows e in, for example, cous [PrSh85, GuSt85, Baum75, MuPr78]. Sometimes some additional information is added (e.g. on the faces of the diagram) which, although being redundant, makes the data structure more efficient for specific purposes. Also, special care is needed when dealing with non-orientable surfaces (which is done in [GuSt85]) because "counterclockwise" cannot have a global meaning on them.

In all the references above the diagram $\mathcal{D}$ is assumed to be connected and its faces to be simply-connected. We want to admit non-connected diagrams and non-simply-connected faces not only to seek generality, but in order to apply our methods to real algebraic plane curves. Actually, an implementation of our algorithms for recognizing and comparing the topology of (affine or projective) real algebraic plane curves is under developement (see [EnHoSa94]), based on the Xtopo program ([Hong93]) which computes the topology of an algebraic curve.

Let us also remark that our main goal when encoding the topology of a diagram will not be computational efficiency but simplicity. Because of this almost all redundant information will be omitted and we will not make use of pointers at all. This is why we prefer the term "code for the topology of a diagram" instead of "data structure representing the topology of a diagram" for our encodings. In this sense our codes will be closer to the Gauss codes which are used to represent simple curves (cf. [KMPS92]) or to the vertex-lists proposed in [GoRe90, GoSa92] than to the data structures referenced above.

For codifying connected diagrams —and for codifying separately each connected component of a non connected diagram— we will do the following. Vertices of the diagram will be labeled by integer numbers and edges adjacent to a particular vertex will also be labeled in counterclockwise direction. Then, each edge e of the diagram will be represented in the form $(A.a, B.b)$ where A and B are the labels for its endpoints and a and b are the places of e in the counterclockwise ordering of edges around A and B respectively. In non-orientable surfaces we will have to define "counterclockwise" locally at each vertex, i.e. choose a local orientation, and each edge will be marked with an additional bit saying whether the two orientations agree or disagree along that edge. The code associated to a connected component of the diagram will be the list of the edges, represented in that way. It can be shown that this list completely captures the topology for connected diagrams with simply connected faces.

After codifying each connected component separately, the whole diagram will be represented with a graph of connected components. This graph will be bipartite, having some *component-nodes* representing connected components of the diagram and some *face-nodes* representing faces. A face-node will be joined to a component-node in the graph if the corresponding face and component are adjacent in the diagram. Apart from the code of each connected component (which will be attached to each component-node of the graph) some information specifying the topology of the face will be attached to each face-node and some information specifying where and how the component is incident to the face will be attached to each edge in the graph. With that information the topology of the diagram will be completely characterized.

The number of equivalent codes representing the same topology is exponential on the topological complexity of the diagram (which equals, roughly speaking, the number of edges). In order to decide equivalence of codes we will choose among them the one which is lexicographically lower, and call it the *canonical* code of the diagram. Somehow paradoxically this canonical code can be found in quadratic time in the most interesting cases.

HILBERT XVI. What different topological types can an algebraic plane curve of given degree have?
(currently open for $d \geq 8$)

Consider for example a connected diagram $\mathcal{D}$ with simply-connected faces, whose code is just a list of edges. If an "oracle" would tell us which vertex is labeled "1" and which edge is labeled the first one at that vertex in the canonical code for $\mathcal{D}$, then the whole canonical code could be recovered straightforwardly. In fact, the first edge in the canonical code will be precisely that one, and its code will be either (1.1, 1.2) if it is a loop at vertex "1" or (1.1, 2.1) if it goes to another vertex, necessarily labeled "2". The second edge will be a (1.3, ...) or a (1.2, ...) depending on the case and the code of its second part will again be easy to guess. This process will only terminate when the code of the whole diagram is computed.

With some preprocessing permitting us to move along the code in constant time the process described can be performed in linear time. To compute the true canonical code for $\mathcal{D}$ we will just need to compute all the candidates —which are two times the number of edges in the diagram— and choose among them the one which is lexicographically the lowest.

For general diagrams with complicated graphs of connected components it will not be true that a canonical code can be found in quadratic time, because testing isomorphy of the graphs themselves will be involved. Nevertheless, in the most important cases for us —which are those of the surface S being the Euclidean plane or the projective plane— the graph of connected components will necessarily be a tree. Moreover, this tree will be canonically rooted, the root being the unbounded or non-orientable face or connected component. In this case, the canonical code can still be computed in quadratic time. The algorithm for that will be recursive: to make a certain node of the tree-code canonical we will first make canonical its sons, then reorder the sons lexicographically, and then make canonical the code for the node itself using the technique described above. Applying this recursive process to the root, the whole tree will be made canonical.

References

[Baum75] B.G. Baumgart, A polyhedron representation for computer vision. In *1975 National Computer Conference.* AFIPS Conference Proceedings, vol. 44. AFIPS Press, Arlington, Va., 1975, pp. 589–596.

[EnHoSa94] M.J. Encarncion, H. Hong, F. Santos, Computing a canonical encoding of the topology of a real algebraic plane curve, in preparation.

[GuSt85] L. Guibas and J. Stolfi, Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams, *ACM Transactions on Graphics*, Vol. 4, No. 2, April 1985, pp. 74–123.

[GoRe90] A. González-Corbalán and T. Recio, Shape invariant lists and realization as plane real algebraic curves with double points. In *Real analytic and Algebraic Geometry. Proceedings, Trento 1988*, Lect. Notes in Math. 1420, Springer-Verlag, 1990.

[GoSa92] A. González-Corbalán and F. Santos, Representation of Curves in the Real Plane, and Construction of Algebraic Curves with Given Topology. In: *Seminaire DDG sur les Structures Algebriques Ordonnees*, Publ. Math. de l'Université de Paris VII, (1992).

[Hong93] H. Hong, Efficient method for analyzing topology of plane real algebraic curves. In *IMACS SC-93*, Lille, France.

[KMPS92] L. Kari, S. Marcus, Gh. Paun and A. Salomaa, In the prehistory of formal language theory: Gauss languages. *Bull. EACTS* **46**, (1992), 124–139.

[MuPr78] D.E. Muller and F.P. Preparata, Finding the intersection of two complex polyhedra, *Theor. Comput. Sci.*, 7, 1978, pp. 217–236.

[PrSh85] F.P. Preparata and M.I. Shamos, *Computational Geometry, An Introduction*, Springer-Verlag, 1985.

(real) algebraic cylindrical decomposition

$\exists i<j : |a_i - a_j| = x$ (input $a_1, \ldots, a_n, x$)

Some Geometric Lower Bounds Using Connected Components

William Steiger*
Dept Matematica Aplicada II
Universität Politecnica de Catalunya
Barcelona
Steiger@cs.princeton.edu

Dobkin and Lipton introduced the connected components technique to deduce lower bounds for element uniqueness in the comparison model of computation. This work applies the same idea to obtain lower bound statements for a variety of problems, each having the flavor of element uniqueness. In fact one of these problems is a parametric version of element uniqueness which asks, given n inputs $a_1, \ldots, a_n$ and a query $x \geq 0$, whether there is a pair of inputs satisfying $|a_i - a_j| = x$; the case $x=0$ IS element uniqueness. This result combines with some others to establish the fact that *search can be easier than uniqueness*: specifically two examples are given (one is the planar ham-sandwich cut) where finding or constructing a geometric object - known to exist - is less complex than answering the question about whether that object is unique. A final result is a nontrivial lower bound for the complexity of the two dimensional minimum-median-residual regression problem: Given n points (x_i, y_i) in the plane, it is required to compute the minimizer (m^*, b^*) of the function $g(m,b) = \text{median} \{ |y_i - (m x_i + b)| \mid i=1, \ldots, n \}$. Souvaine and Steele gave on $O(n^2)$ algorithm for this task. Using some of the previous results and a reduction argument, an $(n \log n)$ lower bound is derived.

"Standard" Lowerbound methods

- Inf. Theory
- Reduce to Sorting
- Reduce to Element Uniqueness
- Combinatorics

*) Joint work with H. Chien

Some Space/Time Tradeoffs for Ranking and Searching

William Steiger
Dept Matematica Aplicada II
Universität Politecnica de Catalunya
Barcelona
Steiger@cs.princeton.edu

Given an arrangement of n lines in general position in the plane, the rank of a vertical line $x=T$ is the number of vertices of the arrangement with x-coordinates $\leq T$. If the lines are sorted by slope (charge n storage locations to retain the results of this preprocessing) the query can be answered in time n log n. On the other hand if the preprocessing step computes all the vertices of the arrangement, sorts them by x-coordinate, and saves the result (charge $O(n^2)$ storage locations for this), then the query can be answered in time O(log n). For some time it had been a question as to whether it is possible to have anything intermediate; i.e., using $o(n^2)$ storage to retain information computed in a preprocessing step, can the ranking query be answered in time $o(n \log n)$? A similar space/time question arises when, given numbers $a_1, \ldots, a_n$ and a value T, it must be decided whether there is a pair whose difference is T; if preprocessing sorts all differences $a_i - a_j$ ($O(n^2)$ storage to retain them) the answer takes O(log n) time while if just the a_i's are sorted ($O(n)$ storage to hold this information) the answer takes $O(n)$ time. Again, the question is whether it is possible to answer the query in sublinear time using sub-quadratic storage. Space/time tradeoffs are presented for both these problems. Although there seems to be less structure to exploit in the latter case, the results obtained are stronger than in the geometric problem.

Space $\frac{n^2}{2T}$ $\frac{n^2}{T^2 \log T}$

Time $O(n \log T)$

↑ for finding

v. d. Stappen (fat)
equally distributed sets

Approximate Center Points in Dense Point Sets

Knut Verbarg Was verbarg Knut ?
Lehrstuhl für Informatik I
Universität Würzburg
Am Hubland
D-97074 Würzburg
Verbarg@informatik.uni-wuerzburg.de

We describe a simple, yet efficient algorithm to compute an approximate center point for a ∂-dense set of n points in $\mathbf{R}^d$ in $O(n)$ time . The quality of the algorithm depends on ∂ and the dimension d.

center of gravity is a $\frac{1}{2\delta^d \lambda}$ center

with $\lambda = d^{(d-1)/2}$

Add weight according to the distance from the center $\frac{1}{dist}$

Was verbarg Knut ?

previous

iterate

fixed point satisfies $\sum \frac{\overrightarrow{p_j}}{\|p_j\|} = 0$

$d=1$ median

↑ Fermat-Weber point

Graph Theoretic Approach

Surface Reconstruction from Unorganized Points in Space

(Extended Abstract)

Robert Mencl
Fachbereich Informatik
Lehrstuhl VII (Computer Graphics)
University of Dortmund
Germany

Introduction

Digitizing natural shapes is a topic of increasing importance in computer graphics. Examples can be found in mechanical engineering (digitizing clay models for the purpose of CAD), but also in computer animation or in interactive virtual environment applications (with the high end application of digitizing human actors). The data produced by the digitizing process can be more or less structured. In this work we assume that the only information about the digitized shape is an unorganized set of sample points. Our goal is to derive a (triangulated) surface interpolating the sample points, for example for further processing in a surface-based CAD system or for surface based rendering.

Several solutions are known for this problem. In [1], Boissonnat describes two approaches. One approach is surface-oriented. Starting with an edge the surface is built up incrementally by successively adding triangles lying close to an estimated tangent plane. A problem with this approach is that the quality of reconstruction depends on the order in which the surface is constructed. It might happen that the extension in a specific direction may not lead to a reasonable reconstruction even if it would be possible for a different direction.

Boissonnat's second approach is volume-oriented in the sense that the object is obtained from the spatial Delaunay triangulation of the given set of points by removing certain tetrahedrons.

Edelsbrunner and Mücke [4, 3] also extract an object from the spatial Delaunay triangulation on the basis of so-called α-balls. A problem with volume-oriented approaches is to apply them on open surfaces not bounding a solid.

The method suggested by Hoppe et al. [6, 7] is in some sense hybrid. In a first step, the surface is described by a set of locally constructed tangent planes. The tangent planes are used to divide the vertices of a regular spatial grid into an inner and an outer region of the object. From that, the surface is obtained by a modified marching cubes algorithm. In difference to the other methods, the resulting surface is approximating, not interpolating. As for volume-oriented approaches the surface is supposed to form the boundary of a solid.

Our solution is surface–oriented. The algorithm consists of three main phases [8]:

1. Computation of the euclidean minimum spanning tree (EMST) of the given set of points and extraction of the component surfaces based on the EMST.
2. Extension of the EMST to a more dense surface description graph (SDG) defining contours on the surface.
3. Filling the contours of the SDG with triangles.

The advantage of this approach is that the EMST leads to skeletons which are often quite similar to the structures a human observer recognizes in a given set of points. For that reason the EMST has found considerable interest in the area of point set clustering. Further, the EMST adapts quite natural to point sets of varying density which may cause difficulties for the other approaches. As a surface oriented technique, the new approach can also reconstruct surfaces with boundaries.

Phase 1: Computation of the Euclidean Minimum Spanning Tree

Let $P = \{p_1, \ldots, p_n\}$ be a set of points describing a surface. An *euclidean minimum spanning tree* (EMST) is a tree connecting all points of P with line segments so that the sum of its edge lengths is minimized. A tree is an acyclic connected graph.

Evidently, the edges of an euclidean minimum spanning tree connect points that lie close together in space. On the other hand, it can be expected for a reasonable sampled surface that the point density on the surface is higher than anywhere else in the surrounding space. In particular for non-convex surfaces and objects consisting of more than one component, points lying far apart form each other in space are unlikely to be neighboring on the surface. This leads to a first of several assumptions on which our approach is based:

Assumption 1 *The edges of an euclidean minimum spanning tree* EMST(P) *are lying on the surface to be reconstructed if the surface consists of one component. If the point set describes more than one surface, the different surfaces are connected pairwise by single edges. These edges are characterized by an exceptional high length in comparison to the majority of the edges.*

As we see in Figure 1 the euclidean minimum spanning tree (upper left) is a fairly adequate skeleton describing the shape of the surface.

There are several possibilties for identifying connected components in the given point set. Some sophisticated methods of point clustering may be applied for that purpose [2, 5, 10].

For a set of n points, an EMST in space can be calculated in time $O(n^2)$.

Phase 2: Computation of the Surface Description Graph

The surface description graph (SDG) is obtained by connecting the leaves of the (possibly reduced) EMST to points in their environment. The SDG is constructed for each component separately.

The selection of a point to be connected to a leaf is based on the assumption that the edges of the graph EMST(P) lie almost in parallel to the true tangent plane of the surface (which is of course unknown). The idea is to do some kind of graph–extension to the leaves of the EMST(P) by adding new edges to the leaves. This is not easy, because it is necessary to avoid degenerated areas in the reconstructed surface. Therefore we have to choose edges which helps our algorithm to reconstruct the surface correctly in its third phase. These edges will be determined by some restrictions for the search, like the search cone (directed towards the leaf) the search length.

Figure 1 shows a SDG (upper right) for the example with the bust. The skeleton looks quite natural.

The concrete SDG depends on the choice of the search cone and the search length. Experiments have shown that for a wide range of data sets these parameters can be chosen so that the SDG leads to a intuitively reconstructed surface.

Based on these observations it is reasonable to assume the following property of the SDG:

Assumption 2 *The edges of the surface description graph SDG(P) are lying on the reconstructed surface, and define the contours of a well-formed surface.*

Phase 3: Contour Filling with Triangles

The purpose of steps 1 and 2 is to build up a global skeleton of the assumed surface. In step 3, the skeleton is extended to a triangular mesh defining the reconstructed surface. A basic assumption for that step is that triangles induced by pairs of incident edges of the SDG enclosing a small angle are likely to belong to the surface. Based on this assumption the triangulation is constructed incrementally by adding edges. The edge to be added next is selected by choosing a pair of incident edges in the current graph in a greed fashion, with respect to the angle enclosed. Among all pairs of incident edges not yet inducing a triangle, one with the smallest enclosing angle is chosen. If the selected edge pair and the triangle induced by it satisfy some additional constraints, the edge closing the edge pair to a triangle is added to the current graph. This process is continued as long as suitable pairs of edges do exist.

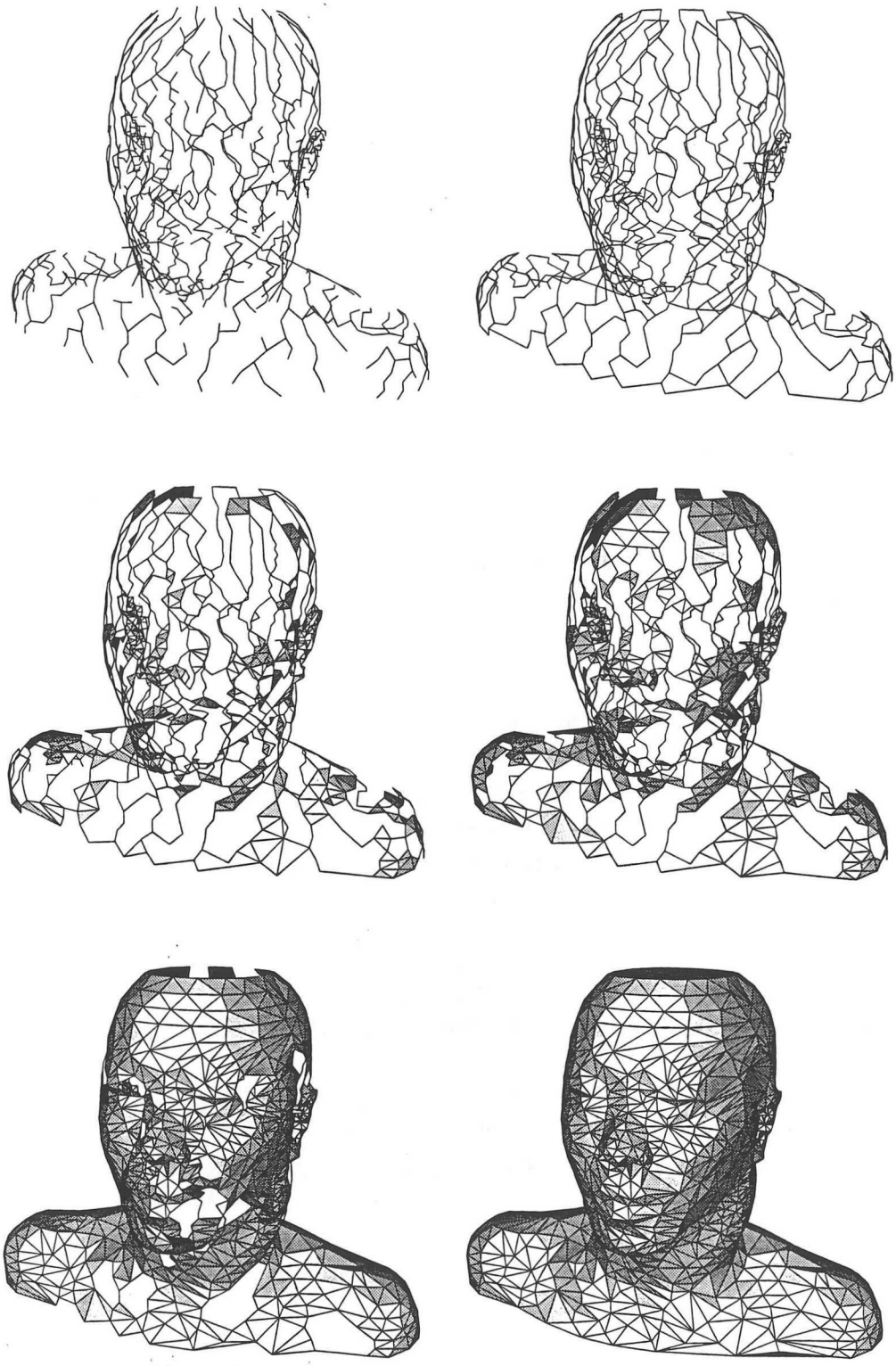


Figure 1: The stepwise reconstruction of a bust. Shown are the euclidean minimum spanning tree, the surface description graph, the reconstruction after 500, 1000, 2000 computed faces and the complete reconstruction with 2730 faces.

These constraints are the following:

1. At most two triangles may be incident to an edge.
2. Triangles may not intersect each other.

The purpose of the next two constraints is to avoid foldovers in the surface.

3. Faces may not overlay each other.
4. The angle between two faces at their incident edge must not be less than a given constant bound.

The next constraint helps us to detect holes and borders of the surface with a determined precision.

5. The angle enclosed by a triangle–inducing pair of incident edges must not exceed a given constant bound.

The overall time complexity of the algorithm is bounded by $O(n^2)$, n the number of given points. The particular expensive intersection and neighborhood tests in the contouring algorithms can be speeded up by using space subdivision techniques.

We are currently working on the extension of the concepts of this algorithm to improve the reconstruction safety as well as the run time of the algorithm.

Experimental results

The algorithm was implemented and applied to several data sets. Some of them were taken from the software package in [4, 9].

A realistic situation is the bust in Figure 1. The number of points in that case is 1436. The times required for this example are about two minutes for phases 2 and 3 and more than one minute for phase 1.

References

[1] Jean-Daniel Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics*, 3(4):266–286, October 1984.

[2] William H. Day and Herbert Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering. *Journal of Classification*, 1:113–135, 1984.

[3] Herbert Edelsbrunner. Weighted alpha shapes, 1992. Technical Report UIUCDCS-R-92-1760, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois.

[4] Herbert Edelsbrunner and Ernst Mücke. Three-dimensional alpha shapes. *Proceedings to the Boston Volume Visualization Workshop*, 1992. Also Technical Report UIUCDCS-R-92-1734, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois.

[5] Tomás Feder and Daniel H. Greene. Optimal algorithms for approximate clustering. *ACM Symposium on Computational Geometry*, pages 434–444, 1988.

[6] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, july 1992.

[7] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. *Computer Graphics Proceedings*, pages 21–26, 1993.

[8] Robert Mencl. Triangulierungen von ungeordneten Punktmengen auf Flächen und Körpern, 1994. Diploma work (in german), Institut für Betriebs- und Dialogsysteme, Fakultät für Informatik, University of Karlsruhe, Germany.

[9] Ernst P. Mücke, 1993. Manual for the 3d delaunay-triangulation program `detri`, (`detri/README.tex`), theory and practice.

[10] Roderick Urquhart. Graph theoretical clustering based on limited neighbourhood sets. *Pattern Recognition*, 15(3):173–187, 1982.

Simulation program of the path tracking for any irredundant planar manipulator and the 6R manipulator

D. F. Bochis, M. J. González-López, T. Recio[1]
Dpt. Matemáticas, Estadística y Computación
Facultad of Ciencias, Universidad de Cantabria
Santander 39071, Spain

The general problem of planifying the trajectory of a robot between two given (initial and final) configurations (position and orientation) q_i and q_f, consists in the description of a continuous mapping f from the interval $[0,1]$ into the set of positions and orientations of all the bodies of the robot (the configuration space C of the robot), such that the images of the extremes 0 and 1 are the given points q_i and q_f, respectively. When dealing in practice with robot manipulators, we usually want, not only to specify the initial and final state of the whole robot, but also the way in which a particular body of the robot (the tip) execute its task. Thus, the input of the problem could be the given positions q_i and q_f, and also a continuous mapping $g : [0,1] \to \Pi(C)$ describing the trajectory of the tip, where Π denotes the projection onto the tip variables. The problem now is to find f as before, and such that $\Pi \circ f = g$. This problem, known as *path tracking* or *path following* in motion planning terminology, can be regarded (see [González-Recio 1]) as a generalization of the *inverse kinematic* problem in which, for a given position and orientation q_{tip} of the tip we have to compute the configurations q_{robot} of the whole robot that place the tip in the given configuration q_{tip}, i.e. such that $\Pi(q_{robot}) = q_{tip}$.

The inverse kinematic problem, which can be considered as the particular instance of the "tracking of a point", has been reduced, from a computer algebra viewpoint, to solve a system of equations (see [Buchberger]) describing the robot's configuration space, in which we want to express the configuration parameters (joint parameters) of all the bodies of the robot as a function of the tip parameters. Following some kind of heuristic methods (see [Paul] and [Hintenaus]) we can obtain closed formulae

$$q_{robot} = F(q_{tip})$$

solving this problem for some robots; for example, in a typical 3R manipulator in the space (composed by a rotatory base over which we place two arms, one of them connected to the base by a rotational link, and the second one connected to the first also by a rotational link in the same plane), we could obtain four formulae to solve the inverse kinematic problem, each one corresponding to a kind of specific configuration between the elbow up/down and left/right arm possibilities.

Once we have determined one closed formula $q_{robot} = F(q_{tip})$, for any given path $g(t) := q_{tip}(t)$ $(t \in [0,1])$ of the tip the path the trajectory for the whole robot is defined as

$$f(t) := q_{robot}(t) = F(q_{tip}(t)).$$

1 Partially supported by CICyT PB 92/0498/C02/01, Esprit/Bra 6846 "POSSO" and TIC-1026-CE.

We could think that this approach could solve the general path tracking problem, but following the formulae suffers, in general, from continuity problems (see [González-Recio 2], [Baker]). To show an easy example, detailed in [González-Recio 2], let us consider the 2R planar manipulator which has two rotational degrees of freedom, performed by the joint variables (θ_1, θ_2). Let us denote the length of its arms by l_1 and l_2, respectively. We consider that the tip variables are the coordinates of the extreme of the second arm, denoted by (x, y). The system relating joint and tip variables is:

$$\begin{aligned} x &= l_1 cos\theta_1 + l_2 cos(\theta_1 + \theta_2) \\ y &= l_1 sin\theta_1 + l_2 sin(\theta_1 + \theta_2) \end{aligned} \tag{I}$$

The inverse kinematic problem consists in determine the angles (θ_1, θ_2) placing the extreme point of the second arm in a given position (x, y). Manipulating the equations in (I) we obtain two closed formulae, one of which is the following ((II):

$$\theta_2 = F_2(x, y) := \arctan(\sin\theta_2, \cos\theta_2), \text{ where } \begin{cases} \cos\theta_2 = \dfrac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1 l_2} \\ \sin\theta_2 = +\sqrt{1 - \cos^2\theta_2}, \end{cases} \tag{II}$$

$$\theta_1 = F_1(x, y) := \arctan(y, x) - \arctan(l_2 \sin\theta_2, l_1 + l_2 \cos\theta_2),$$

Now, if the two arms have equal length, and for simplicity let $l_1 = l_2 = 2$. Consider the parametrization of the segment between the points (1, 1) and (-1,-1):

$$g(t) := (2t - 1, 2t - 1), \quad , t \in [0, 1].$$

Then, the trajectory for the first angle given by (II) is

$$\theta_1(t) = \arctan(2t - 1, 2t - 1) - \arctan\big(2\sin\theta_2(t), 2(1 + \cos\theta_2(t))\big)$$

where $\cos\theta_2(t) = \dfrac{(2t-1)^2 - 4}{4}$ and $\sin\theta_2(t) = +\sqrt{1 - \cos^2\theta_2(t)}$, but the mapping $\theta_1(t)$ is not continuous in $t = 1/2$:

$$\lim_{t \to (1/2)^+} \cos\theta_1(t) = \frac{\sqrt{2}}{2} \neq -\frac{\sqrt{2}}{2} = \lim_{t \to (1/2)^-} \cos\theta_1(t)$$

In fact, we cannot find one unique formula independent of the extreme points of the tip path to describe the trajectory of the robot. This is related to the results shown by Baker in [Baker], where there are studied the relations between the kinematic singularities (the point (0, 0) in the above case) and the existence of closed formulae to describe the tracking of a trajectory.

The aim of this work is to present a path tracking algorithm for any irredundant planar manipulator, i.e., any robot in the plane composed by two arms and the tip, connected

among them by rotational or prismatical joints. These restrictions produce a collection of eight classes of robots, depending on the types of the joints. We study the kinematic equations of each one of these classes, the configuration space, the singularities and the projection of the configuration space onto the tip variables. We also propose a tip trayectory which is tracked by the whole robot (avoiding the continuity problems that could appear). It is also studied the case of the 6R general manipulator.

These path tracking algorithms have been implemented in C for Unix, using the graphics library SPHIGS. The program has an interface that allows to choose the robot class parameters (lengths of the arms, type of joints, etc.). Once these parameters are fixed, the program displays the projection of the configuration space into the position tip variables; this allows to choose the the initial and final configurations inside the set of reachable points. The output is the motion of the robot from the initial to the final given configurations.

References

[Baker] D. R. Baker: *Some topological problems in robotics.* The Mathematical Intelligencer, Vol.12, n. 1. Springer Verlag. (1990)

[Buchberger] B. Buchberger:*Applications of Groebner Bases in non-linear Computational Geometry.* Trends in Computer Algebra. Lecture Notes in Computer Science 296. Springer Verlag. Ed. R.Jansen (1989).

[González-Recio 1] M. J. González-López y T. Recio: *Path tracking in motion planning.* The Computer journal, vol. 36, No. 5, pp.515-524 (1993).

[González-Recio 2] M. J. González-López y T. Recio: *Display of a straigh line path for a 2R planar manipulator.* Proceedings CEIG'94, pp. 1-8, 1.994.

The Computer journal, vol. 36, No. 5, pp.515-524 (1993).

[Hintenaus] P. Hintenaus: *The inverse kinematics system.* RISC Tech. report 87-18.0. University of Linz (1987).

[Paul] R. P. Paul: *Robot Manipulators: Mathematics, Programming and Control.* The MIT Press Series in Artificial Intelligence (1981).

Computation of a Non-Uniform Grid on the Configuration Space of a Robot Arm Amidst Obstacles: Intersection Problem.

Carl Van Geem
Research Institute for Symbolic Computation (RISC-Linz)
Joh. Kepler University
A-4040 Linz, Austria
Internet: Carl.Van.Geem@risc.uni-linz.ac.at

1 Specification of the Problem.

Robot Motion Planning using the Configuration Space (Cspace) consists of two subproblems: the construction of Cspace and planning a path for a point in Cspace. This contribution tackles the problem of the construction of Cspace. We concentrate on a robot with rotational joints only, and to Cspaces of at least 6 dimensions. Recently, we have developed an algorithm for path planning in Cspace, using a particular representation (so called GET-representation) of a non-uniform grid on Cspace and a 'potential field' on the grid (see [VG. 94a] and [VG. 94b]).

The cells of the non-uniform grid, used in our planning algorithm, can be classified in two categories: 'white cells' and 'black cells'. 'White cells' are those cells in which all configurations correspond to free positions in workspace; 'black cells' are cells in which some (or all) configurations are forbidden positions in workspace. Here we consider the computation of the GET-representation (GET-Rep.) of such a non-uniform grid on Cspace, when given a robot arm and a number of obstacles in the workspace of the robot.

2 Modelling Scheme.

The GET-Rep. of a non-uniform grid is a compact representation of the generalised quadtree (2^k-tree) corresponding to the grid. It consists of two binary lists. Due to the characteristics of 2^k-trees and their GET-Rep., it is sufficient to consider only the problem of constructing the Cspace for one obstacle in the workspace. For the computation of the GET-Rep. for two obstacles, we can combine the two GET-Rep. where we considered each of the obstacles seperately. We can parallelise these computations efficiently.

We use the same hierarchical solid modelling technique for modelling the robot arm as the one described in [T. & F. 88]. On the j-th hierarchical level, the links $A_0,...,A_{j-1}$ are modelled by j solid bodies. The other links are modelled by their swept volume with respect to the joints following link A_{j-1}. The cells on the free space in Cspace are computed stepwise. In the j-th step, the j-th dimension of the Cspace is added. The $j-1$-dimensional white cells computed in the previous step are expanded to j-dimensional white cells; additional free space is computed for the $j-1$-dimensional black cells.

For simplicity reasons, we consider the obstacle in 3D workspace to be a convex polyhedron. We assume that the links of the robot arm are modelled by cylinders. Firstly we have to consider the intersection of the polyhedral obstacle with the cylinder modelling the j-th link and secondly with the swept volume of the following links. In the following we only consider the cylinders. In a first approximation the swept volumes can be overestimated as cylinders.

3 Intersecting a Polyhedron with a Cylinder.

We concentrate on the case where the cylinder rotates around an axis through the center of one of its circular faces.

Consider the center of this circle to be the origin of a fixed coordinate frame, and the rotation axis to be the Y-axis. We choose the symmetry axis of the cylinder to be the Z-axis and we choose the X-axis in such a way that we get a righthand oriented coordinate frame.

We then consider the interval along the Y-axis that intersects the cylinder. We preform a subdivision strategy along this interval in order to find the extreme values for the angles of rotation for which intersection occurs. For each step in the subdivision, we look for intersections in a plane perpendicular to the Y-axis.

3.1 The 2D Intersection Problem.

In each step of the subdivision, we choose some particular y-value, and we only consider what happens in plane Φ parallel to the Z-axis and through the point on the Y-axis with the chosen y-value.

The intersection of plane Φ with the cylinder is a segment S parallel to the Z-axis. (Due to symmetry, we only consider wlog. the halfplane with positive x-coordinates.) We compute the distance in plane Φ of the endpoints of S to the origin. For endpoint b lying in the XY-plane, this is just its x-coordinate x_b. We denote the other value by r_{max}.

When rotating the cylinder, each point s of segment S discribes a circle $C(o,r)$ with center in the origin and radius the distance r of s to the origin. All possible values for r are between x_b and r_{max}.

Observe that for a given point w in the circular region swept by S when rotated, there is exactly one point s on the segment that collides with a given point w during the sweep. Indeed, this point s has coordinates $(x_b, y_b, \sqrt{r^2 - x_b^2})$, where r is the radius of the circle with center in the origin and through w. So, for any points w on the polyhedron, we can easily determine the unique angle α_w for which the segment, rotated over that angle, intersects with the polyhedron at w. This computation is the Basic Operation in our approach.

3.2 Subdivision.

Let C denote the cylinder in its initial position (angle of rotation $\alpha = 0$), C_α for the cylinder rotated over angle α, P the polyhedron and e an edge of P. Let us write R for the radius of the circle at the basis of cylinder C. Then we start off with the interval [-R,+R] along the Y-axis.

We get the following algorithm.

Step 1: Subdivide [-R,+R] in intervals [y_i,y_{i+1}] such that all edges of P with at least one point with y-coordinate in [-R,+R] are intersected by at least one plane Φ_i (parallel to the XZ-plane, with y-coordinate y_i).

Step 2: Compute for each Φ_i the corresponding α_i with the Basic Operation.

Step 3: Change α_i until the smallest and largest value when sweeping a plane Φ along the corresponding edge e_i is reached.

Step 4: Find among all these 'extreme values' the smallest and largest value for α.

We will now concentrate on Step 3.

We propose a discretisation approach for this step. Plane Ψ through edge e and the over α rotated segment S is tangent to the cylinder iff. α is an extreme value along e. Thus, for each edge e, we can proceed in the following way. We write w for the point with y-coordinate y_i on edge $e = e_i$ and α_w for α_i.

Step i: Rotate points p_a and p_b over angle $-\alpha_w$, resulting in points e_1 resp. e_2.

Step ii: We now check whether the line through e_1 and e_2 is tangent to cylinder C. We can do this check in the XY-plane by projecting the points e_1 and e_2 and determining whether the line through their projections is tangent to the circle at the bottom of the cylinder. So, project e_1 and e_2 to the XY-plane, compute the square of the distance of the line E through these projected points to the origin and compare the result with R^2. If these values are identical, α_w is the extreme value for angle α along edge e.

Step iii: If these values are different, then the extreme value for α along edge e occurs at another point w on e.

We now concentrate on this last step. If indeed point w, chosen as above, is not the good one, we adapt α_w by a factor $\sigma > 0$. We can consider the angle $\beta_1 = \alpha_w - \sigma$ as a new candidate for the extreme value. We compute the projection E_i of edge e after it has been rotated over $\beta_i = \beta_{i-1} - \sigma$ (where $\beta_0 = \alpha_w$), and check for the number of intersection points with the circle. If there is one intersection point, then we reached the extreme value for the angle. If there are two intersection point, then we repeat the adaptation of α_w (we consider $\beta_{i+1} = \beta_i - \sigma$). If there is no intersection, we repeat the adaptation of α_w in the opposite direction by $\frac{1}{2}\sigma$ (we consider $\beta_{i+1} = \beta_i + \frac{1}{2}\sigma$). Obviously, we again use the technique of computing the distance from the origin to E_i, in order to determine which of the above cases apply. We should keep in mind that here we compute numerically the distance and the comparison with radius R, so some inaccuracy in the data may occur. We can now compute the distance of the origin $(0,0)$ to E_i with the formula:

$$d((0,0),E_i) = \frac{y_{pa} - s_i\,(x_{pa}\cos\kappa - z_{pa}\sin\kappa)}{\sqrt{(y_{pb}-y_{pa})^2 + ((x_{pb}-x_{pa})\cos\kappa - (z_{pb}-z_{pa})\sin\kappa)^2}},$$

where $s_i = \frac{y_{pb}-y_{pa}}{(x_{pb}-x_{pa})\cos\kappa-(z_{pb}-z_{pa})\sin\kappa}$ and κ the angle of rotation. The value of σ will be chosen appropriately to get an adequate subdivision for a 2^k representation of the Cspace and thus suited for the GET-Rep..

References

[T. & F. 88] B. Faverjon and P. Tournassoud, Motion Planning for Manipulators in Complex Environments, in 'Lecture Notes in Computer Science, No. 391, Geometry and Robotics'.

[VG. 94a] C. Van Geem, Towards a Fast Solution Method for the General Robot Motion Planning Problem using a Manhattan-like Distance Function on a Non-Uniform Grid in Configuration Space, 10th European Workshop on Computational Geometry, Santander, March 17-18, 1994.

[VG. 94b] On Using a Manhattan Distance-Like Function for Robot Motion Planning on a Non-Uniform Grid in Configuration Space, Technical Report, RISC-Linz, 94-74, October 1994.

A Hybrid Symbolic-Numerical Method for Tracing Curves and Surface Intersections

Trần Quốc Nam *
Research Institute for Symbolic Computation
RISC–Linz, Johannes Kepler University
A–4040, Linz, Austria
E-mail: tqnam@risc.uni-linz.ac.at

Abstract

We present a hybrid symbolic-numerical algorithm for approximation and representation of surface-to-surface intersections – the fundamental and difficult problem in Computer Aided Geometric Design (CAGD) and solid modeling. Reliability and efficiency of intersection algorithms are two basic prerequisites for their effective use in any practical system. Typically, a numerical algorithm is efficient, but it is not fully robust and may fail in certain cases. On the other hand, algorithms based on exact arithmetic are fully robust and accurate, but are normally slow and require a lot of memory space. Perhaps the goals of efficiency and reliability cannot be met simultaneously without some compromises.

In this paper, we negotiate those compromises judiciously. The key step of the algorithm is based upon the new technique of the author, namely the "Extended Newton Method" for determining the roots of an arbitrary system of equations iteratively, where the equations can be nonlinear algebraic or transcendental. The number of equations of the system can be greater than, less than or equal to the number of variables. Our method for the surface-to-surface intersection problem does not require the expensive computation of birational mapping, projections, re-constructions, etc. Further it utilizes floating-point arithmetic whenever possible. Experiments from surface-to-surface intersection examples found in the literature show the dramatic speed up of the method.

Keywords: Hybrid Symbolic-Numerical Method, Computer Aided Geometric Design (CAGD), Algebraic Geometry, System of Equations Solving.

*Supported by the Austrian Fonds zur Förderung der wiss. Forchung, Project Posso, Nr. P9181-TEC.

1 Introduction

In this paper, we describe a hybrid symbolic-numerical algorithm for approximation and representation of surface-to-surface intersections – the fundamental and difficult problem in Computer Aided Geometric Design (CAGD) and solid modeling. Given two tri-variate polynomials, the algorithm produces a space graph which is an approximation and homeomorphic to the real algebraic set of the intersection (In fact, a space curve) in Euclidean space. Due to its importance, there have been persistent efforts in devising algorithms for this problem (see [BFJP87, BHLH88, Hof88, Hof89, KPP90, W+91, Pat92, Pat93]). The main issue in the intersection problem is the efficient discovery and description of all features of the solution with high precision commensurate with the tasks required from the underlying geometric modeler. Reliability and efficiency of intersection algorithms is a basic prerequisite for their effective use in any geometric modeling system. It is closely associated with the way the algorithm handles such features as near singular cases, small loops, etc.

Roughly speaking, surface intersection methods can be classified in two main categories: symbolic approach and numerical approach.

Symbolic methods (as in [Win93, Wal93]) are based upon the primitive element theorem of Algebraic Geometry: every irreducible d-dimensional algebraic set in n-dimensional space is, after a suitable linear transformation of coordinates, birationally projectable onto an irreducible d-dimensional algebraic set in $d+1$-dimensional space, which can be specified by a single polynomial in $d+1$ variables together with the birational mappings. The general structure of algorithms using symbolic approach is: (1) Decompose the algebraic set of the intersection into the union of irreducible ones by using, for instance, Ritt–Wu's method (see [Wan92]). (2) For each irreducible algebraic set, project it onto a hyper-surface getting a plane curve and a set of birational mappings. (3) Analyzing the topology of plane real algebraic curves. (4) Project plane real algebraic curves back into Euclidean space using their birational mapping. The method is reliable. It can handle the features near singular cases, small loops, or even partial surface overlap. However, the algorithm is complicated, inefficient and requires a lot of memory space, especially when we have to deal with points with real algebraic number coordinates.

On the other hand, we can trace surface-to-surface intersections numerically (see [BHLH88, BK90, PP90, KP91, KPPW92]). The method involves generating point sequences of each separated branch of the algebraic set by stepping from a given point on the required branch in a direction prescribed by the curve's local differential geometry. However such methods are by themselves incomplete in that they require starting points for every branch of the solution. Finding the starting points is a difficult task. Moreover, the method is not reliable and have great difficulties when tracing through singularities. The reason stems from this fact: a numerical tracing scheme technically requires formulating and solving a linear system of equations. Such a system is formulated both for determining an approximation to the curve at a given points, as well as when refining an estimate of the location of a point on the curve with Newton iterations. At a curve singularity, the linear system is singular,

and nearby the condition number of the system is so large that roundoff errors destroy all accuracy of the estimate.

Our method is a compromise between those two methods which uses exact computation or numerical computation when needed. The method overcomes the main difficulties of numerical tracing, namely finding a sample point for every real branch and marching through singularities by using techniques for real root isolation of polynomials with real algebraic number coefficients and the continuation method for solving polynomial systems. The key step of the algorithm is based upon the new technique of the author, namely "Extended Newton Method" for determining the roots of an arbitrary system of equations iteratively, where the equations can be nonlinear algebraic or transcendental (see [Nam94a, Nam94b]). The number of equations of the system can be greater than, less than or equal to the number of variables. The algorithm for surface-to-surface intersections does not require the expensive computation of birational mapping, projections, re-constructions, etc. as in a pure symbolic method. Further it utilizes floating-point arithmetic whenever possible. Experiments from surface-to-surface intersection examples found in the literature shows the dramatic speed up of the method.

The structure of the paper is as follows. In Section 2, we give basic definitions and theorems of the new method for determining the roots of an arbitrary system of equations iteratively; we recall some basic facts for real root isolation of polynomials with real algebraic number coefficients and the continuation method for solving polynomial systems. In Section 3, we elaborate the main algorithm. In Section 4, we prove some theorems which are concerned with the correctness and termination of the algorithm. In Section 5, we show how to achieve further efficiency by the use of floating point and interval arithmetic. In Section 6, we report experimental results.

References

[BFJP87] R.E. Barnhill, G. Farin, M. Jordan, and B.R. Piper. Surface/surface intersection. *Comp. Aided Geom. Design*, 4:3–16, 1987.

[BHLH88] C. L. Bajaj, C. M. Hoffmann, R. E. Lynch, and J. E. H. Hopcroft. Tracing Surface Intersections. *Computer Aided Geom. Design*, 5: pages 285–307, 1988.

[BK90] R. E. Barnhill and S. N. Kersey. A marching method for parametric surface/surface intersections. *Computer Aided Geometric Design*, 7(1–4):257–280, 1990.

[Hof88] C.M. Hofmann. A dimensionality paradigm for surface interogations. Technical Report CSD-TR-837, Purdue Univ., Dept. of Comp. Sci., West Lafayette, Indiana, USA, 1988.

[Hof89] C. M. Hoffmann. *Geometric and Solid Modelling – an Introduction.* Morgan Kauffmann Publisher, San Mateo, California, 1989.

[KP91] G. A. Kriezis and N. M. Patrikalakis. Rational polynomial surface intersections. In *Proc. 17th ASME Design Automation Conf.*, 1991.

[KPP90] G. A. Kriezis, P. V. Prakas, and N. M. Patrikalakis. A method for intersecting algebraic surfaces with rational polynomial patches. *Computer Aided Design*, 22(10):645–654, 1990.

[KPPW92] G. A. Kriezis, P. V. Prakas, N. M. Patrikalakis, and F. E. Wolter. Topological and differential equation methods for surface intersection. *Computer Aided Design*, 24(1):41–55, 1992.

[Nam94a] **Tran Quoc** Nam. Extended Newton's method for finding the roots of an arbitrary system of equations and its applications. In M. H. Hamza, editor, *Proceedings of the Twelfth IASTED International Conference: Applied Informatics*. IASTED, 1994.

[Nam94b] **Tran Quoc** Nam. Extended Newton's method for finding the roots of an arbitrary system of equations and its applications. Technical Report 94–49, RISC-Linz, J. Kepler Universität, Linz, Austria, 1994.

[Pat92] N. M. Patrikalakis. Interrogation of surface intersections. In R. E. Barnhill, editor, *Geometry Processing for Design and Manufactoring*, pages 161–185. SIAM, Philadelphia, 1992.

[Pat93] N. M. Patrikalakis. Surface-to-surface intersections. *IEEE Computer Graphics & Applications*, 1(1):89–95, 1993.

[PP90] N. M. Patrikalakis and P. V. Prakash. Surface intersections for geometric modeling. *J. Mechanical Design*, 112(1):100–107, 1990.

[W+91] Y. Wang et al. Intersection of parametric surfaces for next generation geometric modeling systems. In J. Turner, J. Pegna, and M. Wozny, editors, *Product Modeling for Computer Aided Design and Manufacturing*, pages 75–96. Elsevier, 1991.

[Wal93] Bernhard Wall. *Symbolic Computation with Algebraic Sets*. PhD thesis, RISC-Linz, Universität Linz, 1993.

[Wan92] D. Wang. Irreducible Decomposition of Algebraic Varieties via Characteristic Sets and Gröbner Bases. *Computer Aided Geom. Design*, 9: pages 471–484, 1992.

[Win93] F. Winkler. Constructive Algebraic Geometry with CASA. Technical report, RISC-Linz series no. 93-26, Research Institute for Symbolic Computation, University of Linz, Austria, June 1993.

List of participants CG'95, Hagenberg AUSTRIA

M. ABELLANAS	Univ. Politecnica de Madrid
E. ARDELEANU	RISC, Univ. Linz
O. AICHHOLZER	Techn. Univ. Graz
F. AURENHAMMER	Techn. Univ. Graz
D. BOCHIS	Univ. de Cantabria
A. BRINKMANN	Univ. Münster
B. BUCHBERGER	RISC, Univ. Linz
L. CUCU	Univ. Temesware Romania
T. GOMEX-DIAZ	Univ. de Limoges
M. DRAGAN	Univ. Temesware Romania
W. FREISEISEN	RISC, Univ. Linz
C. VAN GEEM	RISC, Univ. Linz
T. GRAF	WWU Münster
D. HEITZINGER	Techn. Univ. Wien
K. HINRICHS	Univ. Münster
F. HURTADO	Univ. Politecnica de Catalunya
R. KLEIN	Fernuniv. Hagen
M.J. GONZALEZ-LOPEZ	Univ. de Cantabria
E. ROANES-LOZANO	Univ. Complutense de Madrid
R. MENCL	Univ. Dortmund
Le NGOC-MINH	Fernuniv. Hagen
H. MÜLLER	Univ. Dortmund
W. MÜLLER	Salzburg-Aigen
T. QUOC-NAM	RISC, Univ. Linz
M. NEUHAUSER	Techn. Univ. Wien
S. NEUWIRTH	Han Dataport
H. NOLTEMEIER	Univ. Würzburg
E. OBERMAYR	VAI
N. PFEIFER	Techn. Univ. Wien
T. RECIO	Univ. de Cantabria
G. ROTE	Techn. Univ. Graz
F. SANTOS	Univ. Cantabria
W. SIEBERER	PHi-Tech
H. STACHEL	Techn. Univ.Wien
W. STEIGER	Univ. Rutgers
S. STIFTER	RISC, Univ. Linz
W. STÖGER	Han Dataport
E. TOMUTA	RISC, Univ. Linz
P. URAY	Graz
K. VERBERG	Univ. Würzburg
V. WEISPFENNING	Univ. Passau
F. Winkler	RISC, Univ. Linz
M. WOLFRATH	Univ. Würzburg
K. YOKOYAMA	RISC, Univ. Linz

RISC – LINZ
Research Institute for Symbolic Computation
Johannes Kepler University
A-4040 Linz, Austria

Confirmation of participation

We confirm that Mister *Günter Rote, Techn. Univ. Graz,* has participated at

11 the European Workshop on
Computational Geometry CG' 95

March 23.-24. 1995, Hagenberg/Linz, Austria

RISC – LINZ
Research Institute for Symbolic Computation
Johannes Kepler University
A-4040 Linz, Austria

Hagenberg, 23. 3.1995

for the organizer

Telephone: Austria (7236) 32 31 – 41 B. Buchberger, Chairman
– 22 F. Lichtenberger, Manager
– 20, 21 Secretary
Telefax: Austria (7236) 32 31-30
Electronic Mail: buchberg @risc.uni-linz.ac.at (Chairman)
lichtenberg @risc.uni-linz.ac.at (Manager)
Bank-Account: Raiffeisenkasse Hagenberg - Pregarten (BLZ 34151) 25320

Call for Papers & Participation

11th European Workshop on Computational Geometry CG'95

March 23-24, 1995, Hagenberg/Linz, Austria

The European Workshop on Computational Geometry will this year be held in Hagenberg (a village close to Linz), Austria, organized by RISC-Linz, Johannes Kepler University. The goal of the Workshop is to bring together the European researchers in Computational Geometry.

Organizing Committee: Sabine Stifter (chair), Anita Zwettler (secr.)

Collection of Abstracts

No proceedings will be published but extended abstracts of presentation will be collected and distributed at the workshop.

Participation

People interested in participating are asked to fill in the enclosed form and to send it as soon as possible (before February 28, 1995) to

Anita Zwettler
Research Institute for Symbolic Computation
SchloßHagenberg
4232 Hagenberg

Phone: (07236) 3231 - 60 Fax: (07236) 3231 - 30
e-mail: zwettler @risc.uni-linz.ac.at

A preliminary program and information about hotel reservation is enclosed.

Registration fee

A registration fee of ATS 800,-- will include the copy of the abstracts, coffee breaks and lunches.

Please indicate on the registration form whether you need bus transfer from hotel to conference localization and back. No public transportation is available.

Please send a cheque payable to "RISC, CG'95" in Austrian shillings or transfer the amount to our account no. 25 320 at RAIKA Hagenberg, Blz. 34 151.