ANGEWANDTE MATHEMATIK

UND

INFORMATIK

Abstracts of the 12th European Workshop on Computational Geometry (CG '96)

Klaus H. Hinrichs (ed.)

Westfälische Wilhelms-Universität, Institut für Informatik
Einsteinstr. 62, D-48149 Münster, Germany

Bericht Nr. 7/96-I

UNIVERSITÄT MÜNSTER

ANGEWANDTE MATHEMATIK

UND

INFORMATIK

Abstracts of the 12th European Workshop on Computational Geometry (CG '96)

Klaus H. Hinrichs (ed.)

Westfälische Wilhelms-Universität, Institut für Informatik
Einsteinstr. 62, D-48149 Münster, Germany

Bericht Nr. 7/96-I

UNIVERSITÄT MÜNSTER

Foreword

This document contains abstracts of the papers that were presented at the 12th European Workshop on Computational Geometry (CG '96), held in Münster (Germany), March 28 - 29, 1996 and supported by DFG, the german research foundation. This support made it possible to invite three distinguished speakers to give talks. We are grateful to Herbert Edelsbrunner, Mark Overmars and Micha Sharir for coming to Münster.

The increased interest led to a problem: the number of submissions was too large to accommodate all speakers, and we had to disappoint some people. The final program contains three invited talks and 33 presentations, leading to a diverse and interesting program. We thank everyone who contributed to the workshop, and we hope that the reader will enjoy reading the abstracts.

Organizing Committee:

Klaus H. Hinrichs (chair)
Eveline Egelkamp (secretary)
Andreas Brinkmann
Andreas Voigtmann

Table of contents

Thursday, March 28, 1996

Friday, March 29, 1996

Session 5 – Chair: M. Pocchiola (ENS Paris Cedex)

Session 6 – Chair: R. Klein (FernU Hagen)

Session 7 – Chair: G. Vegter (U Groningen)

Session 8 – Chair: F. Hurtado (U Pol. Catalunya)

Results on Fat Objects with a Low Intersection Proportion

Michael Wolfrath*
Lehrstuhl für Informatik I
Universität Würzburg

December 14, 1995

Abstract

We survey the influence of a new assumption for a set of k–fat objects, which we call *low intersection proportion*, on topics as motion–planning and arrangements. The main difference between our approach and those given by van der Stappen is that we allow the objects to intersect in a certain manner. With a low intersection proportion of the objects we can extent known results involving non–intersecting fat objects in motion–planning to k–fat objects, which now may intersect each other. Furthermore we show that the combinatorial complexity of the union of a set of n constant–complexity k–fat objects with a low intersection proportion is $O(n)$. **Keywords:** Computational Geometry, Motion Planning, Arrangements, Fat Objects.

*Email: wolfrath@informatik.uni-wuerzburg.de

Range Searching in Low-Density Environments*

Otfried Schwarzkopf† Jules Vleugels‡

Abstract

The idea of fatness is based on the observation that most worst-case bounds of geometric algorithms are achieved by artificial constructions. Many existing geometric worst-case bounds can be improved significantly if we assume the underlying set of objects to be fat. In this paper we relax the notion of fatness to the weaker definition of a low-density environment, in which at most a constant number of objects of some minimum size intersect a hypercube that is not too large compared to the objects. Fatness implies low density, that is, a set of fat objects is a low-density environment. Generalizing previous results by Overmars and Van der Stappen [?], we present data structures to store a set of arbitrary disjoint objects in $\mathbb{R}^d$ under the low-density property, such that point location and bounded-size range searching with arbitrarily-shaped ranges can be performed in $O(\log^{d-1} n)$ time. Although we achieve the same complexity bounds, our algorithm is superior from a practical point of view. Furthermore, our results apply to arbitrarily shaped objects, and the approach is robust if the objects do not satisfy the requirements for low density. Both data structures require $O(n \log^{d-1} n)$ storage after $O(n \log^{d-1} n \log \log n)$ preprocessing.

*This research was partially supported by ESPRIT Basic Research Action No. 6546 (project PROMotion) and by the Netherlands Organization for Scientific Research (NWO).

†Department of Computer Science, Postech, San 31 Hyoja-dong Pohang, Kyungbuk 790-784, South Korea. Email: otfried@vision.postech.ac.kr

‡Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, the Netherlands. Email: jules@cs.ruu.nl

Computing the Convex Hull of a Simple Polygon on the Sphere

Frank Weller Carsten Kirstein
Universität Dortmund

This talk considers simple polygons on the unit 2-sphere and their spherical convex hulls. We propose a linear-time algorithm for constructing these convex hulls.

A spherical polygon is defined by a sequence of vertices, $p_1, \ldots, p_n$, on the unit 2-sphere. No consecutive vertices p_i, p_{i+1} (indices are taken modulo n) may form an antipodal pair. The polygon is composed of the unique shortest paths (great circular arcs) between all consecutive pairs of vertices.

A subset M of the 2-sphere is called convex if, for any non-antipodal pair of points $p, q \in M$, the unique shortest path between p and q lies in M.

Except for some special cases, the convex hull of a spherical polygon is contained in an open hemisphere, and its boundary is a simple polygon all of whose vertices are convex. This "normal" case is equivalent to a planar convex-hull problem via a one-to-one, onto central projection between the open hemisphere and a suitably chosen plane. Thus, a planar convex-hull algorithm can be employed once a suitable projection plane has been found. However, it appears to be far from trivial to identify such a plane efficiently. The special cases are easily treated, once they have been recognized.

The proposed algorithm starts by splitting a sherical polygon along a great circle. The split step produces two polygons, each of which is "almost simple" and contained in a closed hemisphere. The convex hull of each of the two parts is computed seperately. If they form special cases, this is detected in the split step and treated accordingly. Otherwise, the two parts share one circular arc, along which they were split. The end vertices of this arc are guaranteed to be vertices of the two partial convex hulls, too. This fact makes it possible to employ a planar convex-hull algorithm without actually computing the central projection. In our algorithm, a modified version of Lee's planar algorithm [1] constructs the convex hull for each part. The union of the partial hulls is a star-shaped polygonal face. Its convex hull, which is also the convex hull of the original polygon, is computed by a Graham's scan [2]. Again, the scan is a modified version that works with vertices on the sphere. Furthermore, it detects and handles the special case that the polygon is not contained in a hemisphere.

The split step, both invocations of Lee's algorithm, and the Graham's scan each run in linear time. Therefore, the time complexity of the overall algorithm is $\theta(n)$. Measurements made with real-world data show that the constant factor contained in the θ-notation is low.

References

[1] D. T. Lee. On finding the convex hull of a simple polygon. *International Journal of Computer and Information Sciences*, 12(2):87–98, 1983.

[2] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133, 1972.

Three-Dimensional Restricted-Orientation Convexity[1]

Eugene Fink eugene@cs.cmu.edu
Derick Wood dwood@cs.ust.hk

Abstract A *restricted-orientation convex set,* also called an *$\mathcal{O}$-convex set,* is a set of points whose intersection with lines from some fixed set is empty or connected. The notion of $\mathcal{O}$-convexity generalizes standard convexity and orthogonal convexity. We explore basic properties of $\mathcal{O}$-convex sets in three dimensions and compare them with properties of standard convex sets.

1 Introduction

The study of convex sets is a branch of geometry that has numerous connections with other areas of mathematics, including analysis, linear algebra, statistics, and combinatorics. The application of convexity theory to practical problems led to the exploration of nontraditional notions of convexity, such as orthogonal convexity [7], finitely oriented convexity [6], and link convexity [1, 10]. These nontraditional convexities have been used in pixel graphics, locked transaction systems, VLSI design, motion planning, and other areas.

Rawlins introduced the notion of *restricted-orientation convexity* as a generalization of standard convexity and orthogonal convexity [8]. Rawlins, Wood, and Schuierer studied $\mathcal{O}$-convex sets and demonstrated that their properties are similar to the properties of standard convex sets [9, 10].

The research on nontraditional convexities has so far been restricted to two dimensions. Our goal is to study nontraditional convexities in three and higher dimensions. We have generalized two different types of restricted-orientation convexity, called *strong convexity* and *$\mathcal{O}$-convexity,* to higher-dimensional spaces and presented an extensive study of restricted-orientation sets in higher dimensions [5, 4, 2, 3].

In this paper, we describe $\mathcal{O}$-convex sets in three dimensions, give some of their basic properties, and compare them with properties of standard convex sets.

We should give a word of warning here. The results are not completely unexpected, yet their proofs are often surprisingly hard; intuition serves us badly. We present the proofs in the full paper [2].

2 $\mathcal{O}$-Convexity in two and three dimensions

We begin by reviewing the notion of $\mathcal{O}$-convexity in two dimensions [8] and presenting some of the basic properties of planar $\mathcal{O}$-convex sets.

We can describe standard convex sets through their intersections with straight lines: a set of points is convex if its intersection with every line is empty or connected. We define $\mathcal{O}$-convexity by considering the intersections of a set of points with lines from a *certain set* (rather than all lines). In other words, we select some collection of lines and say that a set is $\mathcal{O}$-convex if its intersection with every line from this collection is empty or connected.

To define this restricted collection of lines, we introduce the notion of an orientation set. An *orientation set* $\mathcal{O}$ is a (finite or infinite) closed set of lines through some fixed point o. An example of a finite orientation set is shown in Figure 1(a). A straight line parallel to one of the lines of $\mathcal{O}$ is called an *$\mathcal{O}$-line.*

Definition 1 **($\mathcal{O}$-Convexity)** *A set is $\mathcal{O}$-convex if its intersection with every $\mathcal{O}$-line is empty or connected.*

For the orientation set in Figure 1(a), the sets shown in Figures 1(b) and 1(c) are $\mathcal{O}$-convex (some $\mathcal{O}$-lines intersecting these sets are shown by dashed lines). On the other hand, the set in Figure 1(d) is not $\mathcal{O}$-convex, since its intersection with the dashed $\mathcal{O}$-line is disconnected. Unlike standard convex sets, $\mathcal{O}$-convex sets may be disconnected. We show a disconnected $\mathcal{O}$-convex set in Figure 1(e).

[1]This work was supported under grants from the Natural Sciences and Engineering Research Council of Canada and the Information Technology Research Centre of Ontario.

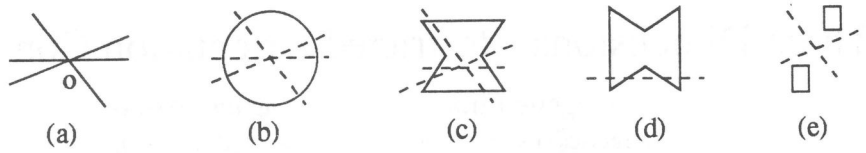


Figure 1: Planar $\mathcal{O}$-convexity.

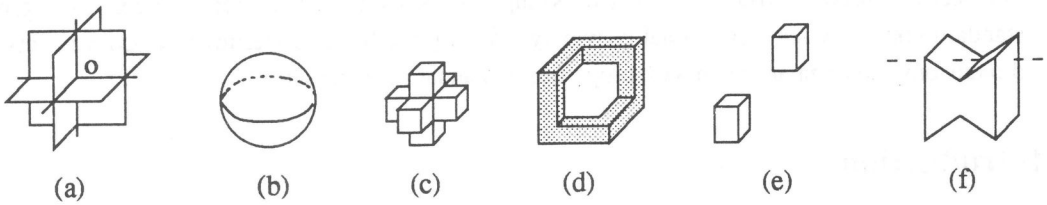


Figure 2: $\mathcal{O}$-convexity in three dimensions.

Lemma 1

1. *Every translation of an $\mathcal{O}$-convex set is $\mathcal{O}$-convex.*
2. *Every standard convex set is $\mathcal{O}$-convex.*
3. *If C is a collection of $\mathcal{O}$-convex sets, then their intersection $\bigcap C$ is also $\mathcal{O}$-convex.*
4. *A disconnected set is $\mathcal{O}$-convex if and only if every connected component of the set is $\mathcal{O}$-convex and no $\mathcal{O}$-line intersects two components.*

We now extend the notion of $\mathcal{O}$-convexity to three dimensions. An *orientation set* $\mathcal{O}$ in three dimensions is a (finite or infinite) closed set of planes through a fixed point o. A plane parallel to one of the elements of $\mathcal{O}$ is called an *$\mathcal{O}$-plane*. $\mathcal{O}$-lines are formed by the intersections of $\mathcal{O}$-planes. In other words, a straight line is an *$\mathcal{O}$-line* if it is the intersection of two $\mathcal{O}$-planes. Note that every translation of an $\mathcal{O}$-plane is an $\mathcal{O}$-plane and every translation of an $\mathcal{O}$-line is an $\mathcal{O}$-line.

We define $\mathcal{O}$-convexity in three dimensions in the same way as in two dimensions: a set is $\mathcal{O}$-convex if its intersection with every $\mathcal{O}$-line is empty or connected. For example, the sets in Figures 2(b)–(e) are $\mathcal{O}$-convex for the orientation set shown in Figure 2(a). On the other hand, the set in Figure 2(f) is not $\mathcal{O}$-convex, because its intersection with the dashed $\mathcal{O}$-line is disconnected.

The properties of $\mathcal{O}$-convex sets given in Lemma 1 hold in three dimensions as well. We next characterize three-dimensional $\mathcal{O}$-convex sets in terms of their intersections with $\mathcal{O}$-planes.

Theorem 2 *A set is $\mathcal{O}$-convex if and only if its intersection with every $\mathcal{O}$-plane is $\mathcal{O}$-convex.*

3 $\mathcal{O}$-Connectedness

We have seen that $\mathcal{O}$-convex sets may not be connected (see Figure 2e), whereas standard convex sets are always connected. We now describe a subclass of $\mathcal{O}$-convex sets that has the connectedness property: all sets of this subclass are connected, just like standard convex sets.

We define sets of this subclass in terms of path-connectedness of their intersection with $\mathcal{O}$-planes. A set is *path-connected* if every two points of the set can be connected by a curve that is wholly contained in the set. (This property is stronger than usual connectedness.)

Definition 2 ($\mathcal{O}$-Connectedness) *A set is* $\mathcal{O}$-connected *if it is path-connected and $\mathcal{O}$-convex, and its intersection with every $\mathcal{O}$-plane is empty or path-connected.*

For example, the sets in Figures 1(b) and 1(c) are $\mathcal{O}$-connected for the orientation set shown in Figure 1(a). On the other hand, the set in Figure 1(d) is not $\mathcal{O}$-connected because its intersections with some horizontal $\mathcal{O}$-planes are disconnected, the set in Figure 1(e) is not $\mathcal{O}$-connected because it is disconnected, and the set in Figure 1(f) is not $\mathcal{O}$-connected because it is not $\mathcal{O}$-convex.

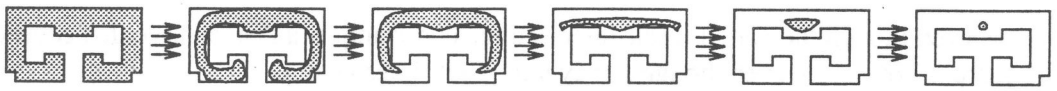


Figure 3: The contraction of a set to a point.

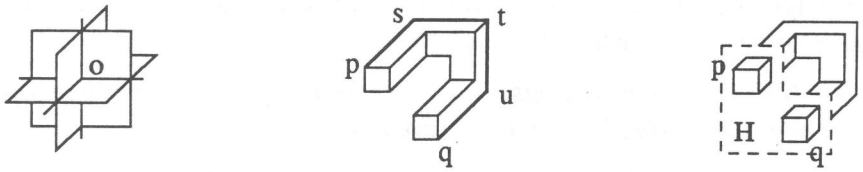


Figure 4: Generalized visibility.

Lemma 3

1. *Every translation of an $\mathcal{O}$-connected set is $\mathcal{O}$-connected.*
2. *Every standard convex set is $\mathcal{O}$-connected.*

We can characterize $\mathcal{O}$-connected sets in terms of their intersections with $\mathcal{O}$-planes, much in the same way as we characterized $\mathcal{O}$-convex sets (see Theorem 2).

Theorem 4 *A set is $\mathcal{O}$-connected if and only if it is path-connected and its intersection with every $\mathcal{O}$-plane is $\mathcal{O}$-connected.*

We next establish *contractability* of $\mathcal{O}$-connected sets. Intuitively, a set is contractable if it is connected and does not have holes. For example, lines, planes, and balls are contractable. On the other hand, a hollow sphere is not contractable, because it has a cavity inside. A doughnut (torus) is not contractable either, because it has a hole through it. To put it more formally, a set is contractable if it can be continuously transformed (contracted) to a point in such a way that all intermediate stages of the transformation are contained in the original set (see Figure 3).

Convex sets are always contractable. Connected $\mathcal{O}$-convex sets in *two dimensions* are also contractable, if the orientation set $\mathcal{O}$ contains at least one line [9, 2]. In three dimensions, on the other hand, a connected $\mathcal{O}$-convex set may not be contractable (see Figure 2d).

Theorem 5 (Contractability) *If the orientation set $\mathcal{O}$ comprises at least two planes, then every $\mathcal{O}$-connected set is contractable.*

4 Visibility

We present two notions of generalized visibility and characterize $\mathcal{O}$-convex and $\mathcal{O}$-connected sets in terms of this generalized visibility.

In standard convexity, two points of a set are *visible* to each other if the straight segment joining them is wholly contained in the set. For example, the points p and s of the set in Figure 4(b) are visible to each other, whereas p and q are not. We can characterize convex sets in terms of visibility: a set is convex if and only if every two points of the set are visible to each other.

We define a weaker visibility, which enables us to give a similar characterization of $\mathcal{O}$-convex sets, by replacing straight segments with simple $\mathcal{O}$-convex curves.

A curve c is *simple* if, for every two points p and q of c, the shortest path from p to q that is wholly contained in c is a segment of c. Informally, this definition says that the shortest way to reach p from q while remaining in c is to follow c. Self-intersecting curves are *not* simple: if p and q are points on the different sides of a loop, the shortest path from p to q does not traverse the loop. Some unusual curves are

not simple even though they are not self-intersecting. For example, a Peano curve that covers all points of a unit square is not simple even though it may not be self-intersecting.

We say that two points of a set are weakly visible to each other if there is a simple $\mathcal{O}$-convex curve joining them that is wholly in the set. For example, we can join the points p and q in Figure 4(b) by the $\mathcal{O}$-convex polygonal line (p, s, t, u, q), which is contained in the set. We characterize path-connected $\mathcal{O}$-convex sets in terms of this generalized visibility.

Theorem 6 (Visibility for $\mathcal{O}$-convex sets) *A path-connected set is $\mathcal{O}$-convex if and only if every two points of the set can be joined by a simple $\mathcal{O}$-convex curve that is wholly in the set.*

We can characterize $\mathcal{O}$-connected sets in a similar way, if we define visibility in terms of simple $\mathcal{O}$-connected curves joining points of a set. This visibility is stronger than $\mathcal{O}$-convex visibility: two points sometimes cannot be joined by an $\mathcal{O}$-connected curve even when they can be joined by an $\mathcal{O}$-convex curve. For example, the points p and q in Figure 4(b) cannot be joined by an $\mathcal{O}$-connected curve contained in the set, because the intersection of the $\mathcal{O}$-plane H (Figure 4c) with every curve joining them is disconnected.

Theorem 7 (Visibility for $\mathcal{O}$-connected sets) *A set is $\mathcal{O}$-connected if and only if every two points of the set can be joined by a simple $\mathcal{O}$-connected curve that is wholly in the set.*

5 Concluding remarks

We generalized $\mathcal{O}$-convexity to three dimensions and demonstrated that the properties of $\mathcal{O}$-convex sets are similar to the properties of standard convex sets. The results presented in the paper, except for contractability (Theorem 5), hold not only in three dimensions but also in higher dimensions [2]. The generalization of the contractability result to higher dimensions is still an open problem.

The work presented here leaves some unanswered questions. For example, we have not characterized the boundaries of $\mathcal{O}$-convex polytopes. In two dimensions, if the orientation set contains n lines, the boundary of every $\mathcal{O}$-convex polytope can be partitioned into at most n $\mathcal{O}$-convex polygonal lines [8]. We conjecture that, for every orientation set $\mathcal{O}$ in three dimensions, there is some fixed number n such that the boundary of every $\mathcal{O}$-convex polytope can be partitioned into at most n connected $\mathcal{O}$-convex regions.

We also plan to address computational aspects of $\mathcal{O}$-convexity, such as verifying $\mathcal{O}$-convexity and $\mathcal{O}$-connectedness of a polytope and computing the $\mathcal{O}$-convex hull.

References

[1] C. K. Bruckner and J. B. Bruckner. On L_n-sets, the Hausdorff metric, and connectedness. *Proceedings of the American Mathematical Society*, 13:765–767, 1962.

[2] Eugene Fink and Derick Wood. Fundamentals of restricted-orientation convexity. Unpublished manuscript, 1995.

[3] Eugene Fink and Derick Wood. Generalized halfspaces in restricted-orientation convexity. Unpublished manuscript, 1995.

[4] Eugene Fink and Derick Wood. Strong restricted-orientation convexity. Unpublished manuscript, 1995.

[5] Eugene Fink and Derick Wood. Three-dimensional strong convexity and visibility. In *Proceedings of the Vision Geometry IV Conference*, 1995.

[6] Ralf Hartmut Güting. Stabbing C-oriented polygons. *Information Processing Letters*, 16:35–40, 1983.

[7] D. Y. Montuno and Alain Fournier. Finding the x-y convex hull of a set of x-y polygons. Technical Report 148, University of Toronto, Toronto, Ontario, 1982.

[8] Gregory J. E. Rawlins. *Explorations in Restricted-Orientation Geometry*. PhD thesis, University of Waterloo, 1987. Technical Report CS-89-48.

[9] Gregory J. E. Rawlins and Derick Wood. Restricted-orientation convex sets. *Information Sciences*, 1991.

[10] Sven Schuierer. *On Generalized Visibility*. PhD thesis, Universität Freiburg, Germany, 1991.

New Greedy Triangulation Algorithms (Extended Abstract)

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

Günter Rote
Institut für Mathematik
Technische Universität Graz
Steyrergasse 30
A-8010 Graz, Austria
e-mail: rote@ftug.dnet.tu-graz.ac.at

Yin-Feng Xu
School of Management
Xi'an Jiaotong University
Xi'an Shaanxi, 710049, PRC

1 Introduction

The classical greedy triangulation (GT) of a set S of n points in the plane is the triangulation obtained by starting with the empty set (of edges) and at each step adding the shortest compatible edge between two of the points of S, where a compatible edge is defined to be an edge that crosses none of the previously added edges.

One use of the greedy triangulation is as an approximation to the minimum weight triangulation (MWT). The MWT of a point set minimizes the total length of all edges, where the length of an edge is the Euclidean distance between its two endpoints. It arises in numerical analysis [Li, Ll, PS], numerical approximation of bivariate data [Yo], or in the reconstruction of surfaces from contours [WA].

Computing a minimum-weight triangulation is an important and interesting problem, whose complexity status is unknown. It is one of the still open problems listed at the end of Garey and Johnson's book about NP-completeness [GJ]. We therefore seek efficiently computable *approximations* to the MWT.

Although neither the GT nor the Delaunay triangulation (DT) yields the MWT [MZ, Ll], the GT appears to be the better of the two in approximating the MWT. In fact, for convex polygons the GT approximates the MWT to with a constant factor while the DT can be a factor of $\Omega(n)$ larger [LL]. For general point sets a factor of $\Omega(\sqrt{n})$ is known [K, Lev] as a lower bound for the GT.

In this paper we use the greedy method as a general concept to compute a triangulation of a planar point set. As mentioned in [AART] we use either edges or triangles as basic objects. Furthermore we give different variants to compute the weight of the objects, either in a static or dynamic way, leading to a total of 156 different greedy triangulation algorithms. We investigate these algorithms in their quality of approximating the MWT for sample point sets. We check whether there are methods superior to others for all (or at least most) of these sets, which are the 'best' for uniformly distributed point sets, or which do avoid worst cases. We used a generic implementation and thus did not optimize the different algorithms.

2 The new methods

The general greedy triangulation procedure may be described as follows: First sort the 'objects' by their weight in non-decreasing order. Then start to insert the objects in this order into the (initially empty) triangulation, where only compatible objects are inserted. Repeat the last step until all compatible objects are inserted, i.e. until a triangulation is obtained. To avoid ambiguity and for simplicity, we assume all weights to be pairwise different.

To clarify this description just consider the 'usual' greedy triangulation. The objects are edges of the point set, the weight is their Euclidean length and compatible means that the edge to be inserted does not intersect any previously inserted edge.

We now consider different possibilities for alter-

native greedy algorithms.

First, objects may either be edges or empty triangles (triangles with no inner points) of S. For both types the same interpretation of 'compatible' is used, meaning intersection-free in their interior. Note that two triangles having one edge in common are compatible in this sense.

Next we distinguish between the size $s(a)$ and the weight $w(a)$ of an object a. The size of an object is a measure of the object itself and does not depend on other objects generated by the point set. For edges we just take their Euclidean length as their size:

1. $s(a)$ = length of edge a

For triangles we have different possibilities:

1. $s(a)$ = perimeter of a
2. $s(a) = \frac{\text{max edge length of } a}{\text{min edge length of } a}$
3. $s(a)$ = area of a
4. $s(a) = \frac{\text{max angle of } a}{\text{min angle of } a}$
5. $s(a)$ = radius of mindisc (smallest enclosing disc) of a
6. $s(a)$ = radius of incircle of a
7. $s(a)$ = radius of the circumcircle of a

The weight of an object is derived from its size and possibly also from the sizes of the objects intersecting it. Let a be an object and I_a be the set of objects intersecting it. E.g. a is an edge and I_a is the set of crossing edges, or a is a triangle and I_a is the set of overlapping triangles. Note that I_a does not include a itself. We use the following different methods to compute the weigth $w(a)$:

1. $w(a) = s(a)$
2. $w(a) = \frac{1}{\max\limits_{a' \in I_a} s(a')}$
3. $w(a) = \frac{1}{\min\limits_{a' \in I_a} s(a')}$
4. $w(a) = \frac{1}{\sum\limits_{a' \in I_a} s(a')}$
5. $w(a) = \frac{1}{\text{mean}\limits_{a' \in I_a} s(a')}$
6. $w(a) = \frac{s(a)}{\max\limits_{a' \in I_a} s(a')}$
7. $w(a) = \frac{s(a)}{\min\limits_{a' \in I_a} s(a')}$
8. $w(a) = \frac{s(a)}{\sum\limits_{a' \in I_a} s(a')}$
9. $w(a) = \frac{s(a)}{\text{mean}\limits_{a' \in I_a} s(a')}$

A. $w(a) = \frac{1}{|I_a|}$

B. $w(a) = \frac{s(a)}{|I_a|}$

The first weight is just the size of the object as used for the usual greedy method. Note that for weights 2) – B) we set $w(a) = 0$ if I_a is empty, witnessing that we have an unavoidable object.

There are two different ways to compute the set I_a: the static and the dynamic method. In the static method we precompute all sets I_a and the weights of all objects which can be derived from the point set. Thus the weights do not change during the insertion phase.

In the dynamic method we recompute the sets I_a and thus the weights after each insertion step if necessary: The sets I_a only contain compatible objects, i.e. objects which do not intersect any previously inserted objects and thus are still 'relevant'. This means that after inserting an object we have to remove all its intersecting objects and to update all involved sets I_a and associated weights. The dynamic method seems to be more sophisticated and we expect it to produce better results. For instance, a long crossing edge does only affect the weight of shorter edges as long as it is possible to insert the long edge into the triangulation. In other words: why should an object affect other objects when we already know that it cannot be part of the triangulation?

Counting the number of possible greedy procedures we have 11 static and 10 dynamic methods to compute the weights, one method to compute the size of an edge and 7 variants to compute the size of a triangle. Except method A) we can combine all weights with all sizes, leading to a total of 156 different greedy triangulation algorithms.

To simplify notation we give a four character identification for the different methods. The first character shows if we use edges (E) or triangles (T) as objects, the next character stands for the static (S) or dynamic (D) method. The following two characters give the number of the formula to compute the size of the objects and their weight, respectively. Thus for the usual greedy triangulation algorithm we have 'ES11'. Note that not all possible combinations are different, as for example 'TD11' which is just the same as 'TS11'.

3 Preliminary Results and Discussion

Because of the huge number of different methods our investigations and the experiments are still going on. Thus for the extended abstract we just

mention some of the preliminary results we got so far.

First let us compare a static edge method (ES11, 'usual' greedy) and a triangle method (TS11), both using the edge length of objects as size and weight.

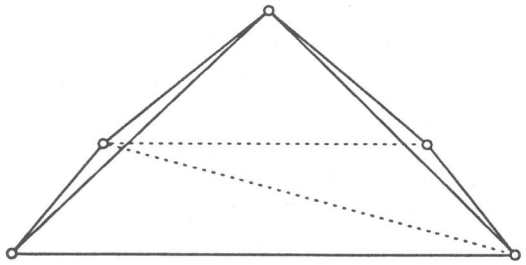


Figure 1: Optimal triangulation of a pentagon (TS11, solid) and the GT (ES11, dotted).

In Figure 1 a well known five point example shows that the GT (ES11) does not produce the MWT in general. Here method TS11 inserts first the small triangles on the left and right side, thus attaining the optimal triangulation. On the other hand ES11 is known to be local optimal while TS11 fails to be, even in the four point example of Figure 2. Thus non of these two methods is superior to the other on all point sets.

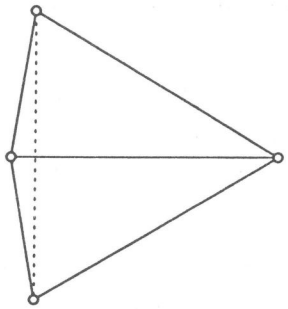


Figure 2: GT of a quadrilateral (ES11, solid) and the non local optimal TS11 (dotted).

To get a ranking we first need an appropriate cost function: For a given point set S let T be the set of triangulations produced by the mentioned greedy triangulation methods. Let $w_{max}(S)$ be the length of the longest and $w_{min}(S)$ the length of the shortest triangulation in T. To each method M we assign the cost $c_M(S) = \frac{w_M(S)-w_{min}(S)}{w_{max}(S)-w_{min}(S)} \in [0,1]$ where $w_M(S)$ is the length of the triangulation produced by method M. Thus $c_M(S) = 0$ means that method M produces (one of) the shortest triangulation in T, while $c_M(S) = 1$ shows that it achieves the upper bound. With c_M we denote the average of $c_M(S)$ over all investigated point sets S for a method M. We use this average cost for ranking the methods.

Method M	Average Cost c_M
ED17	0.00041
ES11	0.00041
TS51	0.00106
TD57	0.00106
TS59	0.00135
TS57	0.00140
ES17	0.00286
TS56	0.00298

Table 1: The eight best methods for 25 uniformly distributed point sets with costs.

Method M	Average Cost c_M
ED16	0.01104
ED17	0.02736
ES11	0.02736
ES16	0.02788
TS56	0.02992
TD56	0.03327
ES17	0.03745
TD19	0.03832

Table 2: The eight best methods for 40 'special' point sets with costs.

In a first phase we computed the cost of every method for 25 uniformly distributed point sets with up to 56 points. (This limitation steems from the huge number of methods and the generic and thus slow implementation; we plan to do experiments on larger point sets in the future). Table 1 shows the eight best methods with their costs. On the average their cost is less than 0.3% of the cost of the longest triangulation of T and 61 of the methods have an average cost of less than 0.1. This shows that for uniformly distributed point sets most of the greedy methods behave quiet well, as was already known for the GT.

To get a better classification in a second phase we added 15 special point sets, e.g. the $\sqrt{n}$–lower bound example of Levcopoulos [Lev] or the examples of Figures 1 and 2. This gave a much clearer distinction of the methods and led to the ranking shown in Table 2. Several observations may be made from this table. First of all, no mentioned greedy method is optimal for all point sets, i.e. produces always the MWT. The GT (ES11)

Group	Average Cost	Best/Groub	Cost of Best
EXXX	0.18486	ED16	0.01104
TXXX	0.28242	TS56	0.02992
XSXX	0.29056	ES11	0.02736
XDXX	0.24736	ED16	0.01104
ESXX	0.23825	ES11	0.02736
EDXX	0.12479	ED16	0.01104
TSXX	0.29803	TS56	0.02992
TDXX	0.26486	TD56	0.03327
XXXX	0.27023	ED16	0.01104

Table 3: Group of methods with 'Leader' and cost.

is among the best, witnessing its well known average behaviour. The method ED16 seems to be very stable against 'bad' constructed examples. While e.g. the GT has now approx. 65 times of its previous costs, so still being third, ED16 has increased its cost by less than 50% (being only no. 13 for random point sets).

In Table 3 we compared groups of methods to get an idea which sizes, weights etc. are good. In the identification used, an X means any valid character, i.e. EXXX are all edge-based methods. We make the following observations: Edge-based methods are significantly better than triangle-based methods. Dynamic methods prevail static ones, but only about 15%, even less for triangle-based methods. Only for edge-based triangulations they give a reasonable improvement. Most of the methods produce short triangulations and only a few behave bad, as can be seen from the small average cost of 0.27023 for all triangulations of T. It is also interesting to note that, for triangles size choice 5 gives the best results in average, while for the weight choice 6 is best not only for triangle- but also edge-based methods. Thus up to now dynamic edge-based methods seem to be preferable, especially ED16.

To refine our investigations it is necessary to consider more and larger point sets. To this end we will concentrate only on the promising subset of methods, e.g. all methods with cost less than 0.1 for the second phase (about 45). This will allow us to get a better distinction of the methods and to get some ideas how a 'optimal' algorithm should look like.

Acknowledgements: We would like to thank J. Lackinger for implementing the generic algorithm.

References

[AART] O. Aichholzer, F. Aurenhammer, G. Rote, M. Taschwer, *Triangulations intersect nicely*, Proc. 11th Ann. Symp. on Computational Geometry, Vancouver, British Columbia, June 1995.

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

[K] D. Kirkpatrick, "A note on Delaunay and optimal triangulations." *Information Processing Letters* **10** (1980) 127–128.

[Lev] C. Levcopoulos, "An $\Omega(\sqrt{n})$ Lower Bound for the Nonoptimality of the Greedy Triangulation." *Information Processing Letters* **25** (1987) 247–251.

[Li] A. Lingas, "On Approximating Behavior and Implementation of the Greedy Triangulation for Convex Planar Point Sets." *Proceedings of the Second Annual Symposium on Computational Geometry* (1986) 72–79.

[LL] C. Levcopoulos and A. Lingas, "On Approximating Behavior of the Greedy Triangulation for Convex Polygons." *Algorithmica* **2** (1987) 175–193.

[Ll] E. L. Lloyd, "On triangulations of a set of points in the plane." *Proceedings of the 18th IEEE Symp. Foundations of Computer Science* (1977) 228–240.

[MZ] G. Manacher and A. Zobrist, "Neither the greedy nor the Delaunay triangulation of the planar set approximates the optimal triangulation." *Information Processing Letters* **9** (1979) 31–34.

[PS] F. Preparata and M. Shamos, "Computational Geometry: an Introduction." Springer-Verlag, 1985.

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

[Yo] P. Yoeli, "Compilation of data for computer-assisted relief cartography." In *Display and Analysis of Spatial Data*, J. C. Davis and M. J. McCullagh, editors, John Wiley & Sons, NY (1975).

The edge-flipping distance of triangulations

Sabine Hanke Thomas Ottmann Sven Schuierer

1 Introduction

Let S be a set of points in the plane, and T a triangulation of S. An *edge-flipping operation* f in T replaces an inner edge e of T with the other diagonal of the quadrilateral Q which surrounds e if Q is convex (Fig. 1). So f transforms T into a triangulation of S that differs from T in exactly one edge. If another edge-flipping operation is used that is not invers to f, then a triangulation of S is generated that differs from T in exactly two edges, and so on. In this way a triangulation can be changed gradually by a sequence of edge-flipping operations. In the literature this method is used to construct particular triangulations from any starting triangulation, where certain criteria (like the min-max angle criterion to construct the Delaunay triangulation [2]) decide which edges are flipped.

A natural idea to compare the structures of triangulation of the same set of points is to compute the *edge-flipping distance* which is defined as the least number of admissible edge-flipping operations to transform a triangulation into another. Of course, it must be shown that such a transformation is always possible.

Let T_1 and T_2 be two triangulations of the same set of points. Every triangulation in particular T_1 and T_2 can be transformed into a Delaunay triangulation with $O(n^2)$ edge-flipping operations [1], where n is the number of points. The resulting Delaunay triangulations may be different, if more than three points lie on a circle, but then these points form a convex polygon, which triangulations can be transformed into each other with at least $2n - 6$ edge-flipping operations [3]. Since edge-flipping is reversible, it is possible to construct T_2 from T_1 with at most $O(n^2)$ edge-flipping operations.

In Section 2 we improve this rough estimate of the edge-flipping distance by showing that the number of intersections between the edges of two triangulations is an upper bound on the edge-flipping distance between these triangulations, and we present an algorithm that computes a sequence of edge-flipping operations that is no longer than the number of intersections.

2 An upper bound on the edge-flipping-distance

In the following we show that the number of intersections between the edges of two triangulations is an upper bound on the edge-flipping distance between these triangulations:

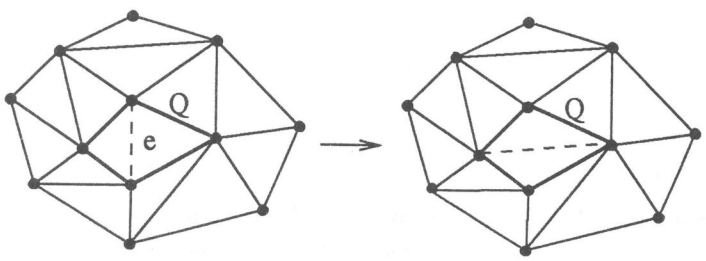


Figure 1: An edge-flipping operation replacing e

Theorem: *Let S be a set of n points in the plane, T_1 and T_2 two triangulations of S, and let $\#(T_1,T_2)$ be the number of intersections of T_1 with T_2. With $flipdist(T_1,T_2)$ denote the edge-flipping distance between T_1 and T_2. Then*

$$flipdist(T_1,T_2) \leq \#(T_1,T_2) < (3n - 2n_b - 3)^2,$$

where n_b is the number of boundary points of T_1 and T_2 respectively.

Proof: It is clear, that $\#(T_1,T_2) < (3n - 2n_b - 3)^2$, because $3n - 2n_b - 3$ is the number of inner edges of T_1 and T_2 respectively. To prove the first inequality we present an algorithm that computes a sequence of edge-flipping operations with maximal length $\#(T_1,T_2)$ that transforms T_1 into T_2. It can be described as follows:

2.1 The algorithm

$\#(e,T)$ is a function returning the number of intersections between the edge e and the triangulation T

$diag(e,T)$ is a function returning the second diagonal of the quadrilateral around the edge e in the triangulation T

```
{
 init stack S;
 T1' := T1; T2' := T2;
 while ( T1' ≠ T2' ) do
 {
  while( there exists a diagonal e of a convex quadrilateral in Ti' with #(e,Tj') = 1,
         where i,j ∈ {1,2}, i ≠ j ) do
  {
   Let e' be an edge in Tj' that intersects such an edge of Ti';
   e'' := diag(e',Tj');
   flip e' ⟶ e'' in Tj';
   if ( j = 1 ) then output( e' ⟶ e'' )
                else S.push(e'' ⟶ e' );
  }
  if ( T1' ≠ T2 ) then
  {
   Let e' be an edge of Ti', i ∈ {1,2}, so that the number of intersections decreases
   at the most by flipping e';
   e'' := diag(e',Ti');
   flip e' ⟶ e'' in Ti';
   if ( i = 1 ) then output( e' ⟶ e'' )
                else S.push( e'' ⟶ e' );
  }
 }
 while (S not empty) do  output(S.pop);
}
```

2.2 Proof of correctness

First we show that the outer while-loop terminates. Observe that:

1. By each execution of the inner while-loop a common edge of the triangulations T_1' and T_2' is generated, and so the number of intersections is decreased at least of one:
Without loss of generality let e be an edge of T_1' that has only one intersection with the edges

of the triangulation T_2', i.e. $\#(e, T_2) = 1$. Let e' be the edge of T_2' that intersect e, and $e'' = diag(e', T_2')$ be the second diagonal of the quadrilateral around e in the triangulation T_2'. The equation $\#(e, T_2) = 1$ implies that e is an edge of T_1' not containing in T_2'. Let $e = ac$, that means a and c are the endpoints of the edge e, and let $a'bc'd$ be the quarilateral around $e' = bd$ in T_2'. Because the boundary edges $a'b$, bc', $c'd$, and $a'd$ of the quadrilateral do not intersect the edge e, and because the endpoints of e can not lie in the interior of the quadrilateral, it follows, that $a'bc'd$ must be convex and $a'bc'd = abcd$, in particular $e'' = e$.

2. The if-statement is executed if and only if $T_1' \neq T_2'$ and all edges that T_1' and T_2' do not have in common are intersect by at least two edges of the other triangulation. We show that in this case there is always an edge-flipping operation that decreases the number of intersections between the triangulations T_1' and T_2':
Without loss of generality we discuss this problem only for the triangulation T_1'.

Lemma 1: *T_1' contains a convex quadrilateral $abcd$ with diagonal ac so that ac has the maximum number of intersections with T_2', i.e. $\#(ac, T_2') = max\{\#(e, T_2') \mid e \text{ is an edge of } T_1'\}$.*
Proof: Omitted

Lemma 2: *Let $abcd$ be a convex quadrilateral in T_1' with diagonal ac so that ac has the maximum number of intersections with T_2'. If T_2' contains an edge be that intersects ad or cd (or an edge dg that intersects ab or bc respectively), then the edge-flipping operation $ac \longrightarrow bd$ decreases the number of intersections of T_1' with T_2'.*
Proof: Omitted

Lemma 3: *Let $abcd$ be a convex quadrilateral in T_1' with diagonal ac so that ac has the maximum number of intersections with T_2'. If T_2' contains no edge be that intersects ad or cd and no edge dg that intersects ab or bc, then either the edge-flipping operation $ac \longrightarrow bd$ decrements the number of intersections between the triangulations T_1' and T_2', or T_1' contains another quadrilateral with maximal diagonal which complies with the conditions of Lemma 2.*
Proof: Omitted

By Lemma 1, 2, and 3 we have shown that for all triangulations T_1' and T_2' which are not equal there is an edge-flipping operation that decreases the number of intersections between these triangulations. Therefore the outer while-loop terminates after at least $\#(T_1, T_2)$ steps, because $T_1' = T_2$ iff $\#(T_1', T_2') = 0$. Then T_1 and T_2 have been transformed to the same triangulation T, where the sequence of edge-flipping operations that transforms T_1 into T has been written on the output during the transformation. Now the sequence that transforms T_2 into T is putted out in reverse order by the last while-loop of the algorithm. So the given algorithm computes a sequence of edge-flipping operations with length of at least $\#(T_1, T_2)$ that transforms T_1 into T_2. □

3 Application to the rotation distance of binary trees, Remarks, and Questions

In section 2 we have given an algorithm that computes a sequence of edge-flipping operations to transform a triangulation T_1 into a triangulation T_2 of the same set of points. This algorithm uses the heuristic, if it is possible, always to flip an edge such, that an edge of T_2 is generated. In the case of triangulations of convex polygons this strategy is very useful. Sleator, Tarjan, and Thurston [3] have shown that through such a kind of flip the edge-flipping distance between the resulting triangulation and the final triangulation decreases by one. So in the case of triangulations of convex polygons the algorithm computes the beginning and end of an optimal sequence of edge-flipping operations until the first occurence of the if-statement. Because there is a 1-1-relationship between edge-flipping operations in triangulations of convex polygons with $n + 2$ vertices and

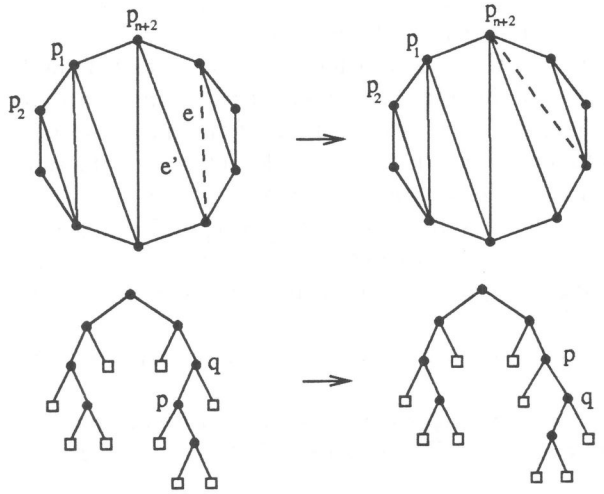


Figure 2: An edge-flipping operation and the corresponding rotation

rotations in binary trees of size n [3], this algorithm is also interesting for the computing of an upper bound on the rotation distances between two binary trees. Every binary tree with n internal nodes can be represented as a triangulation of a convex polygon $P = \{p_1, ..., p_{n+2}\}$ in the following way: The edge $p_{n+2}p_1$ represents the root, and every other boundary edge p_ip_{i+1} $(i = 1, ..., n+1)$ represents a leaf of the tree. The triangulations contains a triangle with edges e_1, e_2, and e_3 if and only if the node represented as e_1 (without loss of generality) is the father of the nodes represented as e_2 and e_3. So a rotation in the tree corresponds to exact one edge-flipping operation in the triangulation. Let p be a node in the tree with father q, and let e and e' respectively be the corresponding edges in the triangulation. If we rotate at p, we get the triangulation representing the generated tree by flipping e. Then the new edge represents q, and e' represents p (Fig. 2). Therefore, rotation distances of binary trees and edge-flipping distances of triangulations of convex polygons are equivalent.

Note that in a sequences of edge-flipping operations with minimal length that transform two triangulation into each other the number of intersections between the generated triangulation during the transformation and the final triangulation does not need to decrease after each step. There are a lot of examples of pairs of triangulations and optimal sequences of edge-flipping operations where the number of intersections increases by some edge-flipping operations during the transformation.

An open question is what the structure of a given sequence of edge-flipping operations must have to be of minimal length.

References

[1] M. Bern and D. Eppstein. Mesh generation and optimal triangulation. D.-Z. Du and F.K. Hwang, editors, Computing in Euclidean Geometry, Volume 1 of Lecture Notes Series on Computing, World Scientific, Singapore, 1992, 23-90

[2] C.L. Lawson. Software for C^1 surface interpolation. J.R. Rice, editor, Mathematical Software III, Academic Press, 1977, 161-194

[3] D.D. Sleator, R.E. Tarjan, and W.R. Thurston. Rotation distance, triangulations and hyperbolic geometry. Journal of the American Mathematical Society, Volume 1, Number 3, July 1988, 647-681

Flipping edges in triangulations of polygons and point sets

Ferran Hurtado Marc Noy
Universitat Politècnica de Catalunya
Barcelona, Spain

Jorge Urrutia
University of Ottawa
Ottawa, Canada

Let $P_n = \{v_1, \ldots, v_n\}$ be a set of points in the plane. Given a triangulation T of P_n and an edge $e = xy$ of T, we say that e can be *flipped* if it is contained in two triangles xyu and xyv of T whose union is a convex quadrilateral. By flipping e we mean the operation of removing e from T and replacing it by the edge uv. The remarkable fact that a simple algorithm, based on replacing a flippable edge xy by uv whenever u is contained in the circle determined by x, y and v, always reaches the Delaunay triangulation starting from any given triangulation, has attracted a lot of a attention on the operation of flipping edges (see the survey paper by Fortune in [1]). Also, when the points are the vertices of a convex polygon, flippings on triangulations are equivalent to the well known operation of rotation in binary trees. If instead of a point set we have a simple polygon Q_n, the definition of flips is the same, restricted to triangulations of Q_n.

In [2], Hurtado and Noy introduced the graph $G_T(Q_n)$, the *graph of triangulations* of a simple polyon Q_n, defined as the graph whose vertices are the triangulations of Q_n and where two triangulations are adjacent if one can be obtained from the other by flipping an edge, and proved that it is always connected. Our first result here is a new proof of this fact, togehter with a bound on the diameter of $G_T(Q_n)$ depending on the size of the visibility graph of Q_n.

Theorem 1 *The graph of triangulations $G_T(Q_n)$ of a simple polygon is connected. Moreover, the diameter of $G_T(Q_n)$ is at most the number of edges in the visibility graph of Q_n.*

Theorem 2 *For every n there is a simple polygon Q_{2n} such that the diameter of $G_T(Q_{2n})$ is exactly $(n-1)^2$.*

We also show that the diameter of the graph of triangulations of a simple polygon depends heavily on the number of its reflex vertices. In a previous paper [3], Hurtado and Noy obtained tight bounds for the number of triangulations of a simple polygon depending on the number of reflex vertices. It should also be mentioned that for a convex polyon Q_n with n vertices, Sleator, Tarjan and Thurston [4] have shown that the diameter of $G_T(Q_n)$ is equal to $2n - 10$ for sufficiently large n (see also [2] for additional results on the case of a convex polygon).

Theorem 3 *Let Q_n be a simple polygon with k reflex vertices. Then the diameter of $G_T(Q_n)$ is at most $O(n+k^2)$.*

The graph of triangulations $G_T(P_n)$ of a point set is defined as above. In this case our main result is concerned with how many edges can be flipped in a triangulation or, in other words, which is the minimum degree of a triangulation in $G_T(P_n)$.

Theorem 4 *Any triangulation of a set of points P_n contains at least $(n-4)/2$ edges that can be flipped. Moreover, this bound cannot be improved.*

Corresponding to Theorems 1 and 2, we also have:

Theorem 5 *The graph of triangulations $G_T(P_n)$ of any set of points P_n is connected and has diameter $O(n^2)$. For every n there exists a set P_{2n} of $2n$ points such that the diameter of $G_T(P_{2n})$ is $(n-1)^2$.*

And finally, we show that the diameter of $G_T(P_n)$ is, as in the case of polygons, sensible to the "complexity" of the configuration P_n. This is made precise in the following result.

Theorem 6 *Let P_n be a set of points wit k convex layers. Then the diameter of $G_T(P_n)$ is $O(kn)$.*

References

[1] S. Fortune, Voronoi diagrams and Delaunay triangulations, in *Computing in Euclidean Geometry*, D.Z. Du and F.K. Hwang eds., World Scientific, 1992, 193-234.

[2] F. Hurtado and M. Noy, The graph of triangulations of a convex polygon, Tech. Rep. MA2-IR-94-13, Universitat Politècnica de Catalunya.

[3] F. Hurtado and M. Noy, Triangulations, visibility graphs and reflex vertices of a simple polygon, to appear in *Computational Geometry, Theory and Applications.*

[4] D.D. Sleator, R.E. Tarjan and W.P. Thurston, Rotation distance, triangulations and hyperbolyc geometry, *J. Am. Mah. Soc.* 1 (1988), 647-682.

THE POLYTOPE OF ALL TRIANGULATIONS OF A POINT CONFIGURATION

(abridged version)

Jesús A. de Loera * Serkan Hoşten † Francisco Santos ‡
Bernd Sturmfels §

1 Introduction

We are interested in the set of all triangulations of the convex hull of a configuration $\mathcal{A} = \{a_1, \ldots, a_n\} \subset \mathbf{R}^d$. The subset of *regular* triangulations is well-understood thanks to its bijection with the vertices of the *secondary polytope* [9]. But non-regular triangulations remain a mystery: for instance, it is still unknown whether any two triangulations of $\mathcal{A}$ can be connected by a sequence of bistellar flips [8]. Regular triangulations seem like the tip of an iceberg. Hidden is an abundance of non-regular triangulations: if $\mathcal{A}$ is the vertex set of the cyclic polytope $C_{4n-4}(4n)$, there are at least 2^n triangulations, while the number of regular triangulations is $O(n^4)$. One approach in understanding non-regular triangulations is to replace the secondary polytope by a larger polytope $P_{\mathcal{A}}$ whose vertices are in bijection with all triangulations of $\mathcal{A}$.

Let $\mathcal{A}$ be a d-dimensional configuration of n possibly repeated points. By a *k-simplex* we mean a subconfiguration of $\mathcal{A}$ consisting of $k+1$ affinely independent points. A *triangulation* of $\mathcal{A}$ is a collection T of d-simplices whose convex hulls cover $conv(\mathcal{A})$ and intersect properly: for any σ and τ in T we have $conv(\sigma \cap \tau) = conv(\sigma) \cap conv(\tau)$. Let N be the number of d-simplices in $\mathcal{A}$. We define $P_{\mathcal{A}}$ as the convex hull in $\mathbf{R}^N$ of the set of incidence vectors of all triangulations of $\mathcal{A}$. For a triangulation T the *incidence vector* v_T has coordinates $(v_T)_\sigma = 1$ if $\sigma \in T$ and $(v_T)_\sigma = 0$ if $\sigma \notin T$. The polytope $P_{\mathcal{A}}$ is isomorphic to the *universal polytope* introduced by Billera, Filliman and Sturmfels [3].

In Section 2 we present a set of linear equations defining the affine hull $aff(P_{\mathcal{A}})$ of $P_{\mathcal{A}}$, which can be read from the oriented matroid $M_{\mathcal{A}}$ of affine dependencies of $\mathcal{A}$. Giving a set of inequalities which define the facets of $P_{\mathcal{A}}$ seems to hard and might involve too many inequalities. Instead, in Secton 3 we consider the polytope $Q_{\mathcal{A}} = aff(P_{\mathcal{A}}) \cap \mathbf{R}^N_+$, which is the linear programming relaxation of $P_{\mathcal{A}}$. In Section 4 we discuss computational issues regarding the enumeration of triangulations and how $Q_{\mathcal{A}}$ can be used in optimization of linear cost functions over $P_{\mathcal{A}}$. In Section 5 we present a duality theory relating (non-regular) triangulations of $\mathcal{A}$ with (virtual) chambers in the Gale transform of $\mathcal{A}$.

*The Geometry Center and School of Mathematics, University of Minnesota, Minneapolis MN 55455, USA

†School of Operations Research, Cornell University, Ithaca NY 14853, USA

‡Departamento de Matemáticas, Estadística y Computación, Universidad de Cantabria, Santander, E-39071, Spain

§Department of Mathematics, University of California, Berkeley, CA 94720, USA

2 Equations defining the affine span of $P_{\mathcal{A}}$

For any $(d-1)$-simplex τ of $\mathcal{A}$ consider the following linear form

$$Co_\tau := \sum_{\substack{\sigma:\tau \text{ facet of } \sigma \\ \sigma \subset H_\tau^+}} x_\sigma - \sum_{\substack{\sigma':\tau \text{ facet of } \sigma', \\ \sigma' \subset H_\tau^-}} x_{\sigma'} \qquad (1)$$

where H_τ^+ and H_τ^- are the open halfspaces defined by the hyperplane that contains τ. We call Co_τ the *cocircuit form* associated with τ. We say that τ is an interior $(d-1)$-simplex if $conv(\tau) \cap int(conv(\mathcal{A})) \neq \emptyset$. This happens if and only if neither of the two sums in (1) is void. Clearly, the equation $Co_\tau = 0$ is satisfied by the incidence vector of any triangulation of $\mathcal{A}$ and for any interior $(d-1)$-simplex τ. The following theorem shows that, in fact, these *interior cocircuit equations*, together with one non-homogeneous equation, suffice to deffine $aff(P_{\mathcal{A}})$ (and also that the affine hull of $P_{\mathcal{A}}$ coincides with the affine hull of the subpolytope defined by *regular* triangulations alone).

Theorem 2.1 *A homogeneous linear form $h = \sum c_\sigma x_\sigma$ vanishes on (the incidence vector of) every regular triangulation of $\mathcal{A}$ if and only if h is a linear combination of the interior cocircuit forms Co_τ defined in (1).*

The following non-homogeneous equations may be used for completing the description of $aff(P_{\mathcal{A}})$: let $p \in conv(\mathcal{A})$ be a point not lying in the convex hull of any $(d-1)$-simplex of $\mathcal{A}$. Every triangulation of $\mathcal{A}$ satisfies the *chamber equation*: $\sum_{\sigma: p \in conv(\sigma)} x_\sigma = 1$The simplices σ which appear in the sum depend only on the *chamber of* p, which is the intersection of all d-simplices $conv(\sigma)$ containing p. Recall that the *chamber complex* of $\mathcal{A}$ is the common refinement of all triangulations of $\mathcal{A}$ (see [1],[4]). Observe that the interior cocircuit equations and the chamber equations of chambers arising from *lexicographic extensions* (see Figure 7.2.2 on page 296 in [5]) can be read from the oriented matroid of affine dependencies of $\mathcal{A}$.

3 The relation between $P_{\mathcal{A}}$ and $Q_{\mathcal{A}}$

Let $Q_{\mathcal{A}}$ denote the polytope in $\mathbf{R}^N$ defined by the interior cocircuit equations $Co_\tau = 0$ plus an extra non-homogeneous equation satisfied on $P_{\mathcal{A}}$ (e.g., a chamber equation), and the inequalities $x_\sigma \geq 0$ for each simplex σ of $\mathcal{A}$. We shall examine the relationship between these two polytopes. Clearly, both are subpolytopes of the unit cube in $\mathbf{R}^N$ and $P_{\mathcal{A}} \subset Q_{\mathcal{A}}$. $P_{\mathcal{A}}$ has only integer vertices but $Q_{\mathcal{A}}$ may have non-integer ones. Also, the simplices in the support of any point of $Q_{\mathcal{A}}$ cover $conv(\mathcal{A})$ and do not overlap, because of the chamber equations of Section 2. Moreover:

Theorem 3.1 *Every integral point of $Q_{\mathcal{A}}$ is the incidence vector of a triangulation of $\mathcal{A}$.*

Theorem 3.2 *Let T_1 and T_2 be two distinct triangulations of $\mathcal{A}$. Then, T_1 and T_2 are neighbors in the 1-skeleton of $Q_{\mathcal{A}}$ if and only if T_1 and T_2 are neighbors in the 1-skeleton of $P_{\mathcal{A}}$.*

Because of Theorem 3.1 we say that $P_{\mathcal{A}}$ is the *integral closure* of $Q_{\mathcal{A}}$. Because of Theorem 3.2 we say that $Q_{\mathcal{A}}$ is *quasi-integral*.

The minimal configuration with $P_{\mathcal{A}} \neq Q_{\mathcal{A}}$ is provided by the five vertices $1, \ldots, 5$ of a regular pentagon together with its center of symmetry O. The polytope $P_{\mathcal{A}} \subset \mathbf{R}^{20}$ is a 10-dimensional polytope. It has a unique fractional vertex v with coordinates $v_{[123]} = v_{[234]} = v_{[345]} = v_{[145]} = v_{[125]} = v_{[013]} = v_{[024]} = v_{[035]} = v_{[014]} = v_{[025]} = 1/2$ and all other coordinates zero. This example is minimal:

Theorem 3.3 *Let $\mathcal{A} \subset \mathbf{R}^d$ be a configuration of n points. The equality $P_\mathcal{A} = Q_\mathcal{A}$ holds in the following cases: if $d = 2$ and all points lie on the boundary of a convex polygon; if $d = 1$; and if $n \leq d + 3$.*

4 Optimization and enumeration

One can show that $P_\mathcal{A}$ is a *set partitioning polytope*, using for this the chamber equations described in Section 2. This, together with the fact that $P_\mathcal{A}$ is *quasi-integral*, has important implications for enumeration and optimization purposes. Balas and Padberg [2] have given a characterization of adjacency between vertices of such polytopes, which leads to an algorithm for optimizing linear functionals over $P_\mathcal{A}$ using $Q_\mathcal{A}$. Starting from an integral vertex of $Q_\mathcal{A}$ the algorithm finds the optimal solution visiting only integral vertices of $Q_\mathcal{A}$ in a fashion similar to the simplex method in linear programming. The same adjacency characterization can be used to enumerate the vertices of $P_\mathcal{A}$ as well.

Unfortunately, no implementation of the Balas-Padberg procedure is known to us. In the other hand, enumerating the triangulations of $\mathcal{A}$ using the existing vertex enumeration packages has two major drawbacks. Either one uses an inequality presentation of $P_\mathcal{A}$, which is hard to get and might involve too many constraints, or one uses $Q_\mathcal{A}$ instead of $P_\mathcal{A}$ but then has to deal with an excessive number of fractional vertices which have to be enumerated too.

However the situation is more promising for optimization problems over $P_\mathcal{A}$ thanks to the efficient implementations of *branch-and-bound* and *branch-and-cut* algorithms for integer programming. For example, finding a triangulation of $\mathcal{A}$ with the minimal number of simplices is equivalent to minimizing the functional $\sum x_\sigma$ over $P_\mathcal{A}$. In the case of the 4-cube ($d = 4, n = 16$) the system defining $Q_\mathcal{A}$ has 1257 equations and 3008 variables. Using CPLEX 3.0 on a SPARC10 workstation, it takes only 150 seconds to verify that the 4-cube's minimal triangulation has 16 simplices. This kind of techniques were used successfully for $d = 5, 6$ in [6]. Finding a minimum/maximum cost triangulation also falls into the category of these linear optimization problems.

5 Duality

Following [4], we now consider a configuration $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ of *vectors* in $\mathbf{R}^{d+1}$. Let N be the number of simplicial cones (i.e., independent $(d+1)$-subsets) of $\mathcal{A}$. If $\mathcal{A}$ is in general position then $N = \binom{n}{d+1}$. All results for point configurations in the preceding sections can be restated for vector configurations, substituting simplices for simplicial cones and triangulations for simplicial fans.

Let $\mathcal{B} = \{b_1, b_2, \ldots, b_n\}$ be a spanning subset of $\mathbf{R}^{n-d-1}$ which is a *Gale transform* of $\mathcal{A}$ [9]. This means that the row spaces of the $(d+1) \times n$ and $(n-d-1) \times n$ matrices A and B which have the vector configurations $\mathcal{A}$ and $\mathcal{B}$ as columns are orthogonal complements. In particular, $M(\mathcal{B})$ is the oriented matroid dual to $M(\mathcal{A})$.

Let $\rho \subset \mathcal{A}$ consist of $d+2$ vectors whose linear span is $\mathbf{R}^d$ but whose positive span $pos(\rho)$ is a pointed cone (these sets correspond to the acyclic circuits of the matroid $M_\mathcal{A}$). Then, ρ has exactly two different triangulations (simplicial fans) T_ρ^+ and T_ρ^-. We define the *circuit vector*

$$Ci_\rho \quad := \quad \sum_{\sigma \in T_\rho^+} e_\sigma - \sum_{\sigma \in T_\rho^+} e_\sigma,$$

where e_σ is the standard basis vector in $\mathbf{R}^N$ corresponding to the simplicial cone σ. Let $Ci(\mathcal{A})$ denote the set of all circuit vectors of $\mathcal{A}$.

Let $\mathcal{T}(\mathcal{A}) \subset \{0,1\}^N$ be the set of incidence vectors of all triangulations of $\mathcal{A}$ and $\mathcal{T}_{reg}(\mathcal{A})$ the subset corresponding to regular triangulations. Similarly let $\Gamma(\mathcal{A}) \subset \{0,1\}^N$ be the set

of all chambers of $\mathcal{A}$. By the results in [4],

$$\mathcal{T}_{reg}(\mathcal{A}) = \Gamma(\mathcal{B}) \qquad \text{and} \qquad \mathcal{T}_{reg}(\mathcal{B}) = \Gamma(\mathcal{A}). \tag{2}$$

We fix the standard inner product $\langle \cdot, \cdot \rangle$ on $\mathbf{R}^N$.

Theorem 5.1 *For a vector $X \in \{0,1\}^N$ the following are equivalent:*
(a) $\langle X, T \rangle = 1$ for all $T \in \mathcal{T}_{reg}(\mathcal{A})$ and $\langle X, Ci_\rho \rangle = 0$ for all $Ci_\rho \in Ci(\mathcal{A})$.
(b) $\langle X, T \rangle = 1$ for all $T \in \mathcal{T}(\mathcal{A})$ and $\langle X, Ci_\rho \rangle = 0$ for all $Ci_\rho \in Ci(\mathcal{A})$.
(c) X is the incidence vector of a triangulation of $\mathcal{B}$; that is, the set of all $(n-d-1)$-subsets σ satisfying $X_{\{1,\ldots,n\}\setminus\sigma} = 1$ defines a triangulation of $\mathcal{B}$.

The chambers of $\mathcal{A}$ constitute a (generally proper) subset of the vectors $X \subset \{0,1\}^N$ characterized by the three equivalent conditions in Theorem 5.1. We propose the following interpretation for the remaining solutions:

Definition 5.2 *The solutions $X \in \{0,1\}^N$ to the system (a) in Theorem 5.3, viewed as collections of $(d+1)$-subsets in $\mathcal{A}$, are called the **virtual chambers** of $\mathcal{A}$. Writing $\Gamma_{virt}(\cdot)$ for the set of virtual chambers, we have*

$$\mathcal{T}(\mathcal{A}) = \Gamma_{virt}(\mathcal{B}) \qquad \text{and} \qquad \mathcal{T}(\mathcal{B}) = \Gamma_{virt}(\mathcal{A}). \tag{3}$$

As an application one can obtain a short proof of Carl Lee's result that every triangulation of a configuration of $d+3$ points is regular [7] and an exponential lower bound for the number of triangulations of the cyclic polytope $C_{4n-4}(4n)$ [9]:

Proposition 5.3 *If $\mathcal{A}$ is the vertex set of $C_{4n-4}(4n)$, then $\mathcal{A}$ has $O(n^4)$ regular triangulations and has at least 2^n triangulations.*

References

[1] **T.V. Alekseyevskaya, I.M. Gel'fand and A. Zelevinsky** *Arrangements of real hyperplanes and the associated partition function*, Soviet Math. Doklady 36, 1988, 589-593.

[2] **E. Balas and M. Padberg** *On the set-covering problem: II. An algorithm for set partitioning*, Operations Research, 23, 1975, 1152-1161.

[3] **L. Billera, P. Filliman and B. Sturmfels** *Constructions and complexity of secondary polytopes*, Adv. in Math. 83, 1990, 155-179.

[4] **L. Billera, I.M. Gel'fand and B. Sturmfels** *Duality and minors of secondary polyhedra*, J. Combinatorial Theory, Ser. B **57**, 1993, 258-268.

[5] **A. Björner, M. Las Vergnas, B. Sturmfels, N. White and G. Ziegler** *Oriented Matroids*, Cambridge University Press, 1992.

[6] **R.B. Hughes** *Minimum-cardinality triangulations of the d-cube for $d = 5$ and $d = 6$*, Discrete Mathematics **118**, 1993, 75-118.

[7] **C.W. Lee** *Regular triangulations of convex polytopes*, in: Applied Geometry and Discrete Mathematics-The Victor Klee Festschrift, (P. Gritzmann and B. Sturmfels eds.) Dimacs Series in Discrete Math. and Theoretical Comp. Science 4, 1991, 443-456.

[8] **J. Rambau and G. Ziegler** *Projections of polytopes and the generalized Baues conjecture*, Preprint 429/1995, TU Berlin.

[9] **G. Ziegler** *Lectures on Polytopes*, Springer-Verlag, New York, 1994.

Smooth Surfaces for Multi-scale Shape Representation

Herbert Edelsbrunner
University of Illinois at Urbana-Champaign

The *skin* of a finite set of points with real weights in $\mathbb{R}^d$ is defined as a differentiable and orientable (d–1)-manifold surrounding the points. The skin varies continuously with the points and weights and at all times maintains the homotopy equivalence between the dual shape of the points and the body enclosed by the skin. The variation allows for arbitrary changes in topological connectivity, each accompanied by a momentary violation of the manifold property. The skin has applications to molecular modeling and docking and to geometric metamorphosis.

Incidence Angle Constrained Visibility

Gregoria Blanco[1] Jesús García López[1] Ferran Hurtado[2]
Pedro Ramos[3] Vera Sacristán[2]

Abstract

Since 1973, when V. Klee proposed the problem of determining the minimum number of points (guards or cameras) that would always suffice to see any n-polygon, many variations of this problem have been studied [10], [11]. Various studies have considered restricting the class of polygons to be guarded (orthogonal, etc.), new kinds of guards have been introduced (edge-guards, mobile guards,...) and even watchman routes have been studied.

Traditionally, it is assumed that guards can see in any direction. If we imagine cameras, instead of guards, this would imply a 360° field of aperture. More recently, more realistic approaches have been proposed in which the angle of visibility is optimized (aperture angle optimization problems [9], [5]) or restricted (floodlight problems [4], [6], [12], [7]); or in which the distance to the seen object is bounded (bounded reach visibility [8], [1]).

We introduce the concept of *quality pictures* in terms of a lower bounded incidence angle visibility, and solve certain combinatorial and algorithmic classical problems with this quality requirement. Some of them are:

Decision problem: Given a polygon/polyhedron P, a set C and an angle α, is it possible to see P from C with quality α?

Optimization problem: Given a polygon/polyhedron P and a set C, find the maximum α so that P can be seen from C with quality α.

Combinatorial maximization problem: Minimize the number of photos needed in order to photograph any n-gon/dron with a given quality α.

Minimization algorithm: Give an algorithm that locates points from which a minimum number of pictures of a given polygon/polyhedron are to be taken from a set C in order to achieve a given quality α.

[1]E.U. Informática, U. Politécnica de Madrid (gblanco@eui.upm.es, jlopez@fi.upm.es).

[2]Dep. Matemàtica Aplicada II, U. Politècnica de Catalunya (hurtado@ma2.upc.es, vera@ma2.upc.es).
Partially supported by U.P.C.Project PR9410 and CICYT TIC95-0957-E.
[3]E.T.S.I. Aeronáuticos, U. Politécnica de Madrid (pramos@fi.upm.es).

References

[1] M. Abellanas, J. García and F. Hurtado: Bounded Reach Visibility, in preparation.

[2] G. Blanco, J. García, F. Hurtado, P. Ramos and V. Sacristán: Fotografías de calidad (in Spanish), *Sextos Encuentros de Geometría Computacional*, Barcelona, 5-7 July, 1995.

[3] G. Blanco, J. García, F. Hurtado, P. Ramos and V. Sacristán: Quality Pictures, in preparation.

[4] P. Bose, L. Guibas, A. Lubiw, M. Overmars, D. Souvaine and J. Urrutia: The Floodlight Problem, *Fifth Canadian Conference on Computational Geometry*, Waterloo, 5-9 August, 1993, pp. 399-404.

[5] P. Bose, F. Hurtado, E. Omaña and G. Toussaint: Aperture angle optimization problems, *Seventh Canadian Conference on Computational Geometry*, Québec, 10-13 August, 1995, pp. 73-78.

[6] J. Czyzowicz, E. Rivera-Campo and J. Urrutia: Optimal Floodlight Illymination of Stages, *Fifth Canadian Conference on Computational Geometry*, Waterloo, 5-9 August, 1993, pp. 393-398.

[7] V. Estivill-Castro and J. Urrutia: Optimal floodlight Illumination of Orthogonal Art Galleries, *Sixth Canadian Conference on Computational Geometry*, Saskatoon, 2-6 August, 1994, pp. 81-86.

[8] J. García: Problemas algorítmico-combinatorios de visibilidad, Ph.D. Thesis (in Spanish), 1995.

[9] F. Hurtado: Looking Through a Window, *Fifth Canadian Conference on Computational Geometry*, Waterloo, 5-9 August, 1993, pp. 234-239.

[10] J. O'Rourke: Art Gallery Theorems and Algorithms, Oxford U. Press, 1987.

[11] I. Shermer: Recent Results in Art Galleries, *Proceeding of the IEEE*, September, 1992.

[12] W. Steiger and I. Streinu: Positive and Negative Results on the Floodlight Problem, *Sixth Canadian Conference on Computational Geometry*, Saskatoon, 2-6 August, 1994, pp. 87-92.

The illumination problem of L. Fejes Tóth revisited

Michel Pocchiola * Gert Vegter †

December 13, 1995, 12:58

We give an algorithmic version of the following art gallery theorem, dating back to 1977 and due to L. Fejes Tóth.

Theorem 1 *[5] The boundary of a set of $n \geq 3$ pairwise disjoint convex sets can be illuminated by $4n-7$ points.* □

The proof of L. Fejes Tóth proceeds by growing the convex sets unboundedly in all directions but where the growth, in a given direction, is limited by the condition that the convex sets remain pairwise interior disjoint. In this way the convex sets will expand into convex polygons that fill the plane except for a finite number of gaps that are also convex polygons. A suitable choice of the lighting points at the vertices of the gaps leads to the $4n-7$ upper bound. It is unclear how to turn this 'growing process' into an efficient algorithm to compute a lighting set with at most $4n-7$ points. In this talk we will show that computing a lighting set with at most $4n-7$ points reduces in $O(n)$ time to computing a pseudo-triangulation of the collection of convex sets. Recall that a pseudo-triangulation is the decomposition of the plane induced by the convex sets and a maximal family of pairwise disjoint free bitangents; see [8, 9] for more details. L. Fejes Tóth provides also collections of $n \geq 3$ convex sets which cannot be illuminated by less than $4n-7$ points but leaves open a practical characterization of all cases when this number of lighting points is claimed. It turns out that this characterisation problem can be solved using the concept of visibility graph/complex (see [8]). Let us say that a visibility complex requires x lighting points if x lighting points are always sufficient and sometimes necessary to illuminate the boundary of any realization of the visibility complex. Using the Koebe-Andreev-Thurston Circle Packing Theorem' [10, page 117] or [7, pages 95–96] we will show that the visibility complexes requiring $4n-7$ lighting points are in one-to-one correspondence with the triangular planar graphs on n vertices.

The 'growing' technique was used previously by L. Fejes Tóth to show that the upper density of a packing with congruent copies of a convex disc C in the plane is bounded by $\text{Area}(C)/\text{Area}(P_6)$ where P_6 denotes a hexagon of minimum area circumscribed round C. More precisely the 'growing process' is used to show that a set of n pairwise disjoint convex sets O_i admits a polygonal cover, i.e, a set of n pairwise disjoint convex polygons R_i with $O_i \subset R_i$ for every i, such that the total number of sides of the polygons in the cover doesn't exceed $6n$ (see [7, 4, 1, 6]). This result was rediscovered, using a similar argument, by H. Edelsbrunner et *al* [3] who have shown that no more than $6n-9$ sides and $3n-6$ slopes for $n \geq 3$ are required for a polygonal cover—these bounds being optimal in the worst case. H. Edelsbrunner et *al* [3] used these bounds to study two combinatorial problems related to transversals and triangulations of collections of convex sets. Finally we mention

*Département de Mathématiques et Informatique, Ecole normale supérieure, ura 1327 du Cnrs, 45 rue d'Ulm 75230 Paris Cedex 05, France (pocchiola@dmi.ens.fr)

†Dept. of Math. and Comp. Sc, University of Groningen P.O.Box 800, 9700 AV Groningen, The Netherlands (gert@cs.rug.nl)

that polygonal covers can be used to compute efficiently depth order queries on terrains; see M. de Berg [2, pages 132–133]. In the talk we will show that computing a polygonal cover with worst case minimal size (i.e., with no more than $6n - 9$ sides and $3n - 6$ slopes, as shown in [3]) for a set of n disjoint convex sets, reduces in $O(n)$ time to computing a pseudo-triangulation. Our polygonal cover algorithm is simpler than the algorithm described by M. de Berg [2]. This latter algorithm, attributed to R. Wenger, applies only to (convex) polygons and achieves an $O(n)$ size for the cover but not worst case size optimality.

References

[1] J. Cassels. *An Introduction to the Geometry of Numbers.* Springer-Verlag, 1959.

[2] M. de Berg. *Ray Shooting, Depth Orders and Hidden Surface Removal*, volume 703 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, 1993.

[3] H. Edelsbrunner, A. D. Robison, and X. Shen. Covering convex sets with non-overlapping polygons. *Discrete Math.*, 81:153–164, 1990.

[4] L. Fejes Tóth. Some packing and covering theorems. *Acta Sciantiarum Mathematicarum (Szeged)*, 12/A:62–67, 1950.

[5] L. Fejes Tóth. Illumination of convex discs. *Acta Math. Acad. Sci. Hungar.*, 29(3–4):355–360, 1977.

[6] A. Heppes. Filling a domain by discs. *Periodica Mathematica Hungarica*, 8:363–371, 1964.

[7] J. Pach and P. K. Agarwal. *Combinatorial Geometry.* John Wiley & Sons, New York, NY, 1995.

[8] M. Pocchiola and G. Vegter. The visibility complex. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 328–337, 1993.

[9] M. Pocchiola and G. Vegter. Computing the visibility graph via pseudo-triangulation. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 248–257, 1995.

[10] G. M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics.* Springer-Verlag, 1994.

Rectangle and Box Visibility Graphs in 3D

Sándor P. Fekete* Henk Meijer†

15 December 1995

(Extended Abstract)

A *visibility representation* of a graph G maps vertices of G to sets in Euclidean space. An edge (u, v) occurs in G if and only if the objects representing u and v see each other according to some visibility rule. (In some investigations, the "if and only if" condition is relaxed to "only if".)

Application areas such as VLSI routing and circuit board layout have stimulated considerable research on visibility representations in R^2. See for example [6], [15], [21], and [24]. Recently, interest has developed in finding good 3-dimensional visualisations of graphs. See for example [5], [14], [16]. It is also of interest to develop geometric graph theory in higher dimensions.

Here we continue the study of a visibility representation, studied for example in [2], [3], [19], [10] and [1], in which the objects representing vertices are 2-dimensional sets parallel to the x, y-plane. An edge (u, v) occurs in G if and only if the objects representing u and v see each other along a line of sight parallel to the z axis. This line of sight must intersect the interiors of the objects; hence legitimate lines of sight (called thick lines) are extensible to tubes of small radius whose ends lie inside the objects. Furthermore, since G has an edge (u, v) if and only if u and v are mutually visible, the graph G is recoverable from the geometry of the representation. Throughout the paper, we use the term *ZPR* (for **Z**-**P**arallel visibility **R**epresentation) to refer to this specific model. It was shown in [3] that K_{20} has a ZPR by axis-parallel rectangles. In [10] it was shown that K_{56} does not have a ZPR by axis-parallel rectangles and that the largest complete graph with a ZPR by axis-parallel unit squares is K_7.

A different visibility representation is studied in [6] and [12]. Here vertices are represented by axis-parallel rectangles in 2-space. Two vertices are adjacent if and only if their respective rectangles can see each other parallel to the x- or y-axis.

This paper extends and generalizes these two representations by considering *box visibility representations (BR)*: vertices are represented by axis-parallel boxes in 3-space, such that edges correspond to axis-parallel thick lines of visibility between the respective boxes. It was shown in [4] that K_{42} has a BR. Box visibility graphs play a role in 3-dimensional packing algorithms, which are of importance for real-life logistics: when using certain local insertion heuristics for packing boxes into a container, the data structure for representing the packing is a box visibility graph. (See [25] and [11].)

*sandor@zpr.uni-koeln.de. Center for Parallel Computing, Universität zu Köln, D-50923 Köln, GERMANY. Parts of this work were done while visiting Utrecht University.

†henk@qucis.queensu.ca. Department of Computer Science, Queen's University, Kingston, Ont K7L 3N6, CANADA. This work was done while visiting Utrecht University, and was partially supported by the Netherlands Organisation for Scientific Research (N.W.O.) and NSERC Canada.

We derive upper and lower bounds on the size of complete graphs with a BR by using results on graphs with a ZPR. We also study complete graphs with a ZPR or a BR when the number of different objects is limited – a question that is important when it comes to packing objects. Since there are different scenarios for orienting boxes in a packing, there are two ways in which two bodies can be considered equal:

(a) They are *isothetic*, i. e. they can be made identical by translations only.

(b) They are *congruent*, i. e. they can be made identical by translations and rotations.

We say that two isothetic objects have the same *size*, while two congruent objects have the same *shape*.

The results in this paper are summarized in the following table.

ZPR						BR					
shapes	min	max	sizes	min	max	shapes	min	max	sizes	min	max
1	12	14	1	7	7	1	30	66	1	8	11
2	16	28	2	12	14	2	40	132	2	14	22
3	18	42	3	16	21	3	46	198	3	20	33
≥ 4	20	55	4	18	28	≥ 4	52	229	4	25	44
			5	20	35				5	30	55
			6	20	42				6	34	66
			7	20	49				7	38	77
			≥ 8	20	55				8	42	88
									9	44	99
									10	46	110
									11	48	121
									12	50	132
									13	51	143
									14	52	154
									..	..	..

Figure 1 gives one of the drawings which are used to establish the lower bounds. It shows a complete visibility graph of 10 rectangles using 4 different shapes or 5 different sizes. Two such sets of rectangles can be used to create a ZPR of K_{20}. If rectangle 1 is removed, we get a K_{18} with 3 shapes or 4 sizes. Also removing rectangle 3 results in a K_{16} with 2 shapes or 3 sizes.

The ZPR of K_{20} becomes a BR if we transform the 20 rectangles into thin boxes. By rotating a set of 20 boxes by 90 degrees and placing them opposite another set we can create a complete box visibility graph of size 40. By adding extra boxes on both sides of these 40, we can construct a BR of K_{52} using 4 shapes or 14 sizes.

The upper bounds are established by showing that certain structures are forbidden in the ZPR and BR of a complete graph. Details are contained in the full paper.

References

[1] H. Alt, M. Godau and S. Whitesides. Universal 3-dimensional visibility representations for graphs. To appear in *Proc. Graph Drawing 95*, Passau 1996. Lecture Notes in Computer Science, Springer 1996.

[2] P. Bose, H. Everett, S. Fekete, A. Lubiw, H. Meijer, K. Romanik, T. Shermer and S. Whitesides. On a visibility representation for graphs in three dimensions. *Proc. Graph Drawing '93*, Paris (Sèvres), 1993, pp. 38-39.

[3] P. Bose, H. Everett, S. Fekete, A. Lubiw, H. Meijer, K. Romanik, T. Shermer and S. Whitesides. On a visibility representation for graphs in three dimensions. Snapshots of Computational and Discrete Geometry, v. 3, *eds.* D. Avis and P. Bose, McGill University School of Computer Science Technical Report SOCS-94.50, July 1994, pp. 2-25.

[4] P. Bose, A. Josefczyk, J. Miller an J. O'Rourke. K_{42} is a box visibility graph, Tech. Report #034, Smith College, 1994.

[5] R. Cohen, P. Eades, T. Lin and F. Ruskey. Three-dimensional graph drawing. *Proc. Graph Drawing '94*, Princeton NJ, 1994. Lecture Notes in Computer Science LNCS #894, Springer-Verlag, 1995, pp. 1-11.

[6] A. Dean and J. Hutchison. Rectangle visibility representations of bipartite graphs. *Proc. Graph Drawing '94*, Princeton NJ, 1994. Lecture Notes in Computer Science LNCS #894, Springer-Verlag, 1995, pp. 159-166.

[7] G. Di Battista, P. Eades, R. Tamassia and I.G. Tollis, Algorithms for Automatic Graph Drawing: An Annotated Bibliography, Technical Report, Dept. of Computer Science, Brown University, 1993.

[8] G. Di Battista and R. Tamassia. Algorithms for Plane Representations of Acyclic Digraphs. *Theoretical Computer Science*, 61:175–198, 1988.

[9] P. Erdös and A. Szekeres, "A combinatorial problem in geometry, " *Compositio Mathematica* v. 2 (1935), pp. 463-470.

[10] S. P. Fekete, M. E. Houle and S. Whitesides. New results on a visibility representation of graphs in 3D. To appear in *Proc. Graph Drawing 95*, Passau 1996. Lecture Notes in Computer Science, Springer 1996.

[11] S. P. Fekete, J. Schepers and M. Wottawa. A practical insertion heuristic for 3-dimensional packing problems. Manuscript.

[12] J. Hutchinson, T. Shermer and A. Vince. On representations of some thickness-two graphs. To appear in *Proc. Graph Drawing 95*, Passau 1996. Lecture Notes in Computer Science, Springer 1996.

[13] G. Kant, G. Liotta, R. Tamassia and I.G. Tollis, "Area Requirements of Visibility Representations of Trees", *Proceedings of the 5th Canadian Conf. on Comp. Geom.*, Waterloo, Ontario, (1993) pp. 192-197.

[14] H. Koike. An application of three-dimensional visualization to object-oriented programming. *Proc. of Advanced Visual Interfaces AVI '92*, Rome, May 1992, v. 36 of World Scientific Series in Computer Science, 1992, pp. 180-192.

[15] E. Kranakis, D. Krizanc and J. Urrutia. On the number of directions in visibility representations of graphs. *Proc. Graph Drawing '94*, Princeton NJ, 1994, Lecture Notes in Computer Science LNCS #894, Springer-Verlag, 1995, pp. 167-176.

[16] J. Mackinley, G. Robertson and S. Card. Cone trees: animated 3d visualizations of hierarchical information. *Proc. of the SIGCHI Conf. on Human Factors in Computing*, 1991, pp. 189-194.

[17] I. Rival and J. Urrutia. Representing Orders by Translating Convex Figures in the Plane. *Order 4*, pp. 319–339, 1988.

[18] I. Rival and J. Urrutia. Representing Orders by Moving Figures in Space. *Discrete Mathematics*, 109, pp. 255-263, 1992.

[19] K. Romanik. Directed VR-representable graphs have unbounded dimension. *Proc. Graph Drawing '94*, Princeton, NJ, 1994, Lecture Notes in Computer Science LNCS #894, Springer-Verlag, 1995, pp. 177-181.

[20] F.J. Cobos, J.C. Dana, F. Hurtado, A. Marquez and F. Mateos On a Visibility Representation of Graphs. To appear in *Proc. Graph Drawing 95*, Passau 1996. Lecture Notes in Computer Science, Springer 1996.

[21] R. Tamassia and I.G. Tollis, "A unified approach to visibility representations of planar graphs," *Discrete Comput. Geom.* v. 1 (1986), pp.321-341.

[22] W. T. Trotter. *Combinatorics and Partially Ordered Sets: Dimension Theory.* Johns Hopkins University Press, Baltimore, MD, 1992.

[23] S. K. Wismath. Bar-Representable Visibility Graphs and a Related Network Flow Problem. University of British Columbia, Dept. of Computer Science Technical Report 89-24, August 1989.

[24] S. K. Wismath. Characterizing bar line-of-sight graphs. *Proc. ACM Symp. on Computational Geometry*, 1985, pp. 147-152.

[25] M. Wottawa. *Praktische Packungsalgorithmen.* Doctoral Thesis. Mathematisches Institut, Universität zu Köln, to appear in 1996.

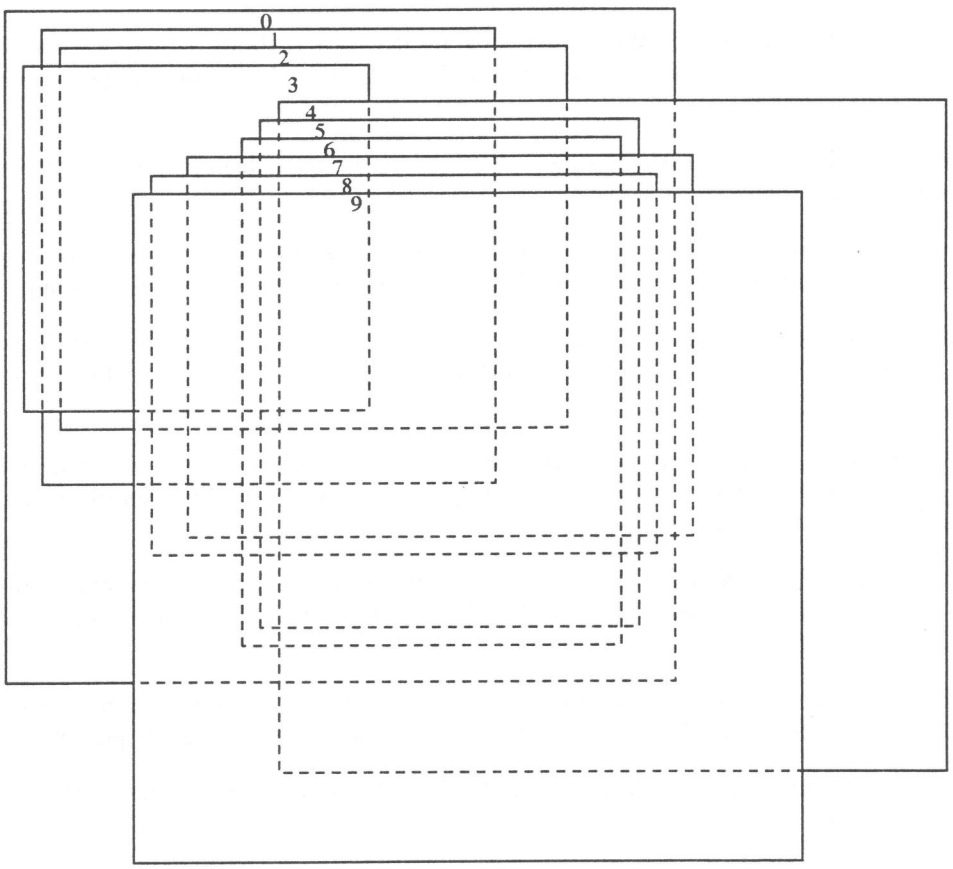


Figure 1: Complete ZPR visibility graph of size 10 using 4 shapes

Dealing with degeneracies and numerical imprecisions when computing visibility graphs

RIVIÈRE Stéphane *

December 15, 1995

1 Introduction

Most algorithms in computational geometry assume that the input is in "general position", that is that no degenerate cases occur (such as three aligned points , or three convergent lines ...). But in practice, when the size of the input becomes significant, degenerate configurations occur and programs fail. Things get worse when the program must face imprecise computations.

Some general tools have been developed to cope with these problems. For solving degeneracies symbolic perturbation schemes are employed (e.g. [EM90], [Yap90]) : coordinates of input are symbolically perturbed so that degeneracies are removed. Other techniques such as the epsilon geometry ([GSS89]) do the actual geometric computations but consider that the results have an imprecision up to ε and decisions made on these results take this imprecision into account. Finally techniques are developed to do exact computations (e.g. [FV93]) and avoid imprecision problems.

These general techniques can be used as "black boxes" for geometric computation in algorithms. This has the advantage that it can be applied to a large range of algorithms, but these black boxes have in general an extra cost which can reduce the performance of the program.

We consider here two algorithms for computing visibility graphs and study how to apply the idea of symbolic perturbation, not for black boxes creation, but for direct modifications of the algorithm, making it robust.

2 The simplicity paradigm applied to visibility

General techniques cope with degenerate cases (e.g. aligned points) by simulating a non degenerate configuration by symbolically perturbing the coordinates of the input by a quantity $f(\varepsilon)$. Symbolically means that, when computing the sign of a determinant e.g. for these points, calculation is not done with the perturbed coordinates, but symbolic calculus shows that the sign of the perturbed determinant is the one of a cofactor of the actual (null) determinant.

When dealing with visibility, perturbations must be done wisely. A degenerate situation, where three objects are aligned, is the limit between two situations : the one where all three objects can see each others, and the one where extremal objects are hidden from each other (see figure 1).

When encountering a degenerate configuration, we want to simulate the simplest situation, that is the situation where extremal objects do not see each other, so that only visibility edges $(1, 2)$ and $(2, 3)$ are computed.

We show how these ideas are applied to two sweep algorithms for computing visibility graphs.

*iMAGIS-IMAG - BP 53
38041 GRENOBLE CEDEX 09 - FRANCE
e-mail : Stephane.Riviere@imag.fr
iMAGIS is a joint project of CNRS/INRIA/INPG/UJF

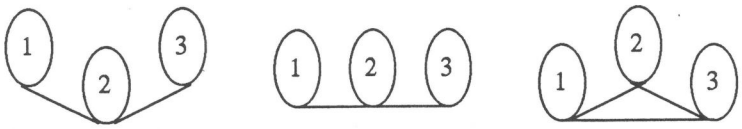


Figure 1: Visibility configurations around a degeneracy

3 Visibility graph of polygonal scenes

We study here the optimal algorithm of [Riv95] for computing visibility graphs of polygonal scenes.

We recall here the important points of the algorithm. The sweep of the visibility graph is done using two graphs : the upper and lower rotation trees. Given a direction θ, for each vertex p of the scene we associate the first vertex seen when looking forward (resp. backward) along θ and turning counterclockwise (see figure 2a). Rotation trees are subgraphs of the visibility graph and are used to sweep it by making θ vary from θ_0 to $\theta_0 + \pi$. Modifications are discrete : at each step, an edge (p, q) is passed (and reported by the program) and the new edge (p, q') (resp. (p', q)) of the upper (resp. lower) rotation tree is computed.

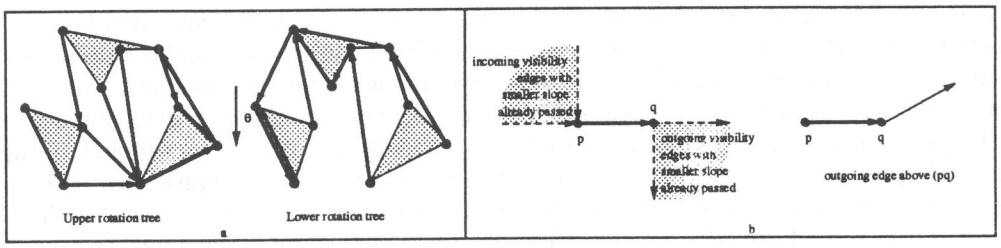


Figure 2: Sweeping the visibility graph with rotation trees

Considerations about visibility coherence show that an edge (p, q) can be passed and updated efficiently if all the visibility edges incoming at p (resp. outgoing from q) with smaller slope have already been passed.

This condition is met if the edge is in both rotation trees and if the outgoing edge of q (resp. of p) in the upper (resp. lower) rotation tree is strictly above (p, q).

Degeneracies may stop the algorithm, since an edge aligned with another can become definitively non-passable. If the "strictly above" relation is loosened to "above" then visibility coherence is lost, wrong decisions are made and generally the program aborts.

The key to cope with this problem is to consider a degenerate configuration as the corresponding non degenerate one where the extremal points are hidden from each other, as shown in figure 3.

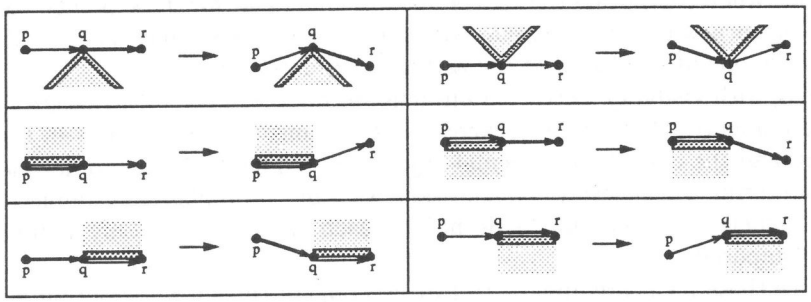


Figure 3: Simulating non degenerate situations

The test "strictly above" is then completed to handle degenerate cases. We just have to add two tests to verify that $[pq]$ is a side of a polygon and one test two check if an edge is strictly above

another one. The rest of the program is unchanged.

4 Visibility graph of convex curved objects

We study here the algorithm of [PV93a] for computing visibility graphs of curved objects, applied to scenes of convex objects (the recent algorithm of [PV95] is optimal, but more involved and not implemented). We recall the important points of the algorithm (for more details on implementation see [Riv93]). Given a direction θ, the tangents to the objects of direction θ divide the space in regions where visibility along θ is constant. An auxiliary graph is constructed : its vertices are the tangents and two tangents are linked by an edge if they bound a same region (see figure 4 a); The edge is then associated to this region. The graph is modified when θ becomes the slope of a free bitangent of the scene (see figure 4 b) : a region shrinks to the bitangent and disappears, and a new region is created.

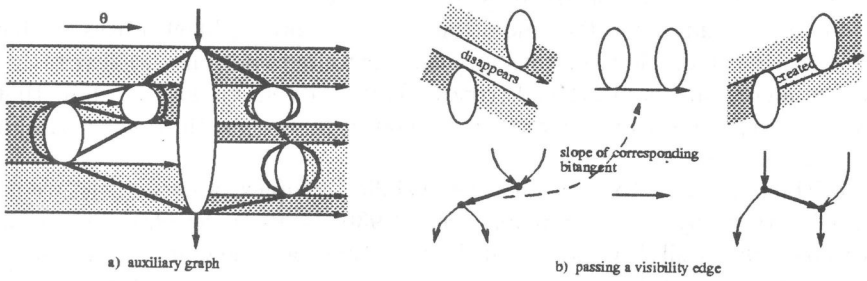


Figure 4: Auxiliary graph for computing visibility graph

The algorithm maintains the auxiliary graph from θ_0 to $\theta_0 + \pi$ reporting all the free bitangents met. Each edge has an associated "date of death" which is the slope of the bitangent causing the associated region to disappear. After initialization, edges are put in a priority queue according to their date of death. At each step, the minimum edge is taken from the queue, the graph is (locally) updated, new dates of death are computed and the priority queue is updated.

The calculation of bitangents is done numerically, and the computed slopes are accurate up to an imprecision of ε (see figure 5a).

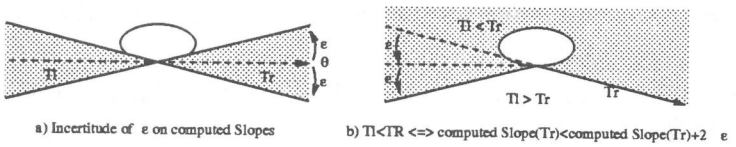


Figure 5: Numerical imprecision

This has important consequences when the program encounters a (nearly) degenerate configuration : it cannot order the bitangents of the three objects correctly according to one of the two possible schemes (see figure 6). The two schemes are mixed and the program must abort.

To cope with this problem, local comparisons are made between adjacent bitangents before putting them in the priority queue. First, the imprecision ε is taken into account when comparing the slope of two consecutive bitangents (see figure 5 b). Then a bitangent is put into the priority queue only if its slope is smaller than the one of its left and right adjacent bitangents. The modifications made when passing a bitangent involve more elements of the auxiliary graph, but these modifications remain local, and still concern only a constant number of elements.

These modifications allow us to handle degenerate cases and to cope with numerical imprecisions.

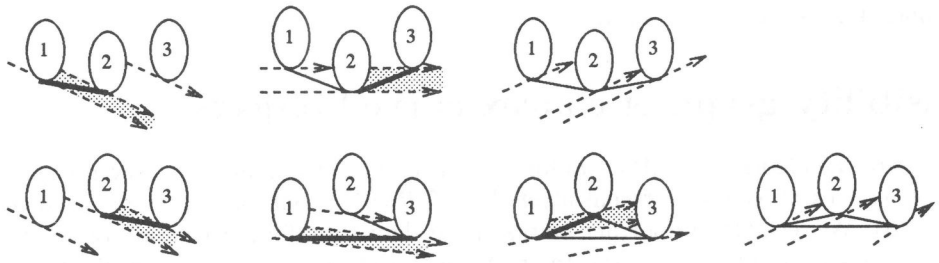


Figure 6: Sweeping schemes concerning a triplet of objects

5 Conclusion

We have shown how the idea of simulating simplicity led us to modify two algorithms for computing visibility graph so that they can handle degenerates scenes.

The advantage of studying actual modifications on applying black boxes is that in general performance is less affected. Modifications may appear to be minor ones with very little extra cost as seen with the algorithm for polygonal scenes. This study seems also to show that algorithms with topological sweep seem to be more easy to modify than algorithms with straight sweep, for which global order is important.

These algorithms are used to construct the visibility complex, a data structure encoding visibility relations between objects in the plane (see [PV93b] for more details). This data structure is then used for answering visibility queries, such as computing a view around a point in the scene. The corresponding programs face the same sort of problems when encountering degeneracies or numerical imprecisions. The principles we followed for computing visibility graphs also apply in the more general frame of visibility computation for allowing easy modifications of the programs, and making them robust and efficient.

References

[EM90] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate c ases in geometric algorithms. *ACM Trans. Graph.*, 9:66–104, 1990.

[FV93] S. Fortune and C. J. Van Wyk. Efficient exact arithmetic for computational geometry. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 163–172, 1993.

[GSS89] L. J. Guibas, D. Salesin, and J. Stolfi. Epsilon geometry: building robust algorithms from imprecise com putations. In *Proc. 5th Annu. ACM Sympos. Comput. Geom.*, pages 208–217, 1989.

[PV93a] M. Pocchiola and G. Vegter. Sweep algorithm for visibility graphs of curved obstacles. manuscrit, 1993.

[PV93b] M. Pocchiola and G. Vegter. The visibility complex. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 328–337, 1993.

[PV95] M. Pocchiola and G. Vegter. Computing the visibility graph via pseudo-triangulation. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, 1995.

[Riv93] S. Rivière. Experimental comparison of two algorithms for computing visibility graphs. manuscrit, 1993.

[Riv95] S. Rivière. Topologically sweeping the visibility complex of polygonal scenes. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages C36–C37, 1995.

[Yap90] C. K. Yap. A geometric consistency theorem for a symbolic perturbation scheme. *J. Comput. Syst. Sci.*, 40:2–18, 1990.

Experimental Comparison of Quadrangulation Algorithms for Sets of Points

*Prosenjit Bose**, Suneeta Ramaswami*, Godfried Toussaint* and Alain Turki**

ABSTRACT

The classical problem in scattered bivariate data interpolation can be stated as follows. Given a set $V = (v_1, v_2,..., v_n)$ of n points in the plane along with an elevation z_i, $i = 1,2,...n$ associated with each point v_i, determine a function f such that $f(v_i) = z_i$, for all i. There is a large body of literature on this subject describing a variety of methods that yield functions with different properties. These methods usually start with a triangulation of V which is subsequently refined in some way. Since a triangulation of a set of points in the plane always exists, there is never a problem starting in this way. Recently Lai & Schumaker [LS94] showed that there are computational advantages when starting with a quadrangulation of V rather than a triangulation. In a quadrangulation of V the basic elements are quadrangles (quadrilaterals) rather than triangles. Furthermore, for this application, the quadrangles need not be convex although it is desirable that they be *fat*. However, a set of points does not always admit a quadrangulation. This problem can be "solved" in two ways: (1) by adding additional points (Steiner points) to the data, although these points should be kept to a minimum since the data is considered too "sacred" to allow such contamination and (2) by just leaving some remaining triangles in the final "quadrangulation" which are then handled with the classical interpolation theory. Motivated by this application, Bose and Toussaint [BT95] first characterized the sets of points that admit a quadrangulation. One such characterization is surprisingly simple: a set V admits a quadrangulation if, and only if, the convex hull of V contains an *even* number of extreme points. This characterization leads to several algorithms that quadrangulate a set of points using at most *one* Steiner point in optimal $O(n \log n)$ time. Although these methods are optimal with respect to both the time complexity and use of few Steiner points they yield quadrangles which are not fat and which are frequently non-convex. A different approach is suggested by another characterization: a set V admits a quadrangulation if, and only if, the set admits a triangulation whose dual graph admits a perfect matching [RRT]. This characterization suggests algorithms that start with a "good" triangulation such as the Delaunay triangulation and then apply either maximum cardinality or maximum weighted matching algorithms to the dual graph of the triangulation. Some of these algorithms yield very "beautiful" quadrangulations with as many as 98% of the quadrangles convex and fat and very few remaining triangles on the average. On the other hand here there is no worst-case guarantee that no more than one Steiner point may be required and the time complexity of the algorithms is far from optimal. In this study we present an empirical comparison of several quadrangulation algorithms and compare them with respect to three objective measures of fatness, the number of convex quadrangles they yield, the number of Steiner points

* School of Computer Science, McGill University, Montreal, Canada.
** Department of Computer Science, University of British Columbia, Vancouver, Canada.
Research of the first author was supported by a Killam Fellowship. Research of the remaining authors was supported by NSERC Grant no. OGP0009293 and FCAR Grant no. 93-ER-0291.

they require and also the subjective beauty of the resulting quadrangulations. The results suggest that there appears to be an inherent trade-off between beauty (fatness) and the number of Steiner points needed and it is an open challenge to find efficient algorithms for obtaining "nice" quadrangulations that use at most one Steiner point in the worst case.

[BT95] P. Bose and G. T. Toussaint, "No quadrangulation is extremely odd," *Proc. 6th International Symposium on Algorithms and Computation*, Cairns, Australia, December 4-6, 1995, pp. 372-381.

[LS94] M. J. Lai and L. L. Schumaker, "Scattered data interpolation using C^2 piecewise polynomials of degree six," *Third Workshop on Proximity Graphs*, Mississippi State University, Starkville, Mississippi, December 1-3, 1994.

[RRT95] S. Ramaswami, P. Ramos and G. T. Toussaint, "Converting triangulations to quadrangulations," Proc. 7th Canadian Conference on Computational Geometry, Laval University, Quebec, August 10-13, 1995, pp. 297-302.

Calculating Voronoi Diagrams using Convex Sweep Curves

Ulrich Kühn
Institut für Informatik, Westfälische Wilhelms-Universität
Einsteinstr. 62, D - 48149 Münster, Germany
kuehn @ math.uni-muenster.de

1 Introduction

The Voronoi diagram is an important data structure in computational geometry. Given n sites in the plane, the Voronoi diagram partitions the plane into n regions. The region of a site p consists of all those points 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 [1].

A generalization of the sweep line method of Fortune [2] was developed in [4] which will be presented here. To achieve this generalization, the sweep *line* was replaced by a general sweep *curve*. In order to look for sweep curves that are feasable for such a Fortune-type algorithm, equidistant curves were introduced.

It turned out that there are two useful forms of sweep curves for the Euclidian plane, *lines* and *circles*. As a further result, a sweep algorithm based on sweep circles was implemented; this algorithms runs with $O(n \cdot \log n)$ time and $O(n)$ space requirements, which is optimal.

2 Sweep Curves

When sweeping with a curve across a Voronoi diagram, there is the problem that this curve crosses the Voronoi region of a site *before* it reaches the site that generates the region. To overcome this, the idea is that a sweep curve runs ahead of the already constructed part of the Voronoi diagram.

Points on the Voronoi edges have the same distance to two of the sites; in order to use this property in a plane sweep algorithm we take a look at the set of points that have the same distance to the sweep curve and a given site. By intersecting two of these sets we obtain points on a Voronoi edge. This leads to

Definition 1 *Let Γ be a curve in $\mathbb{R}^2$ and let $p \in \mathbb{R}^2$ be a point. Then we define $E_\Gamma(p) := \{x \in \mathbb{R}^2 | d(x,p) = d(x,\Gamma)\}$ to be the* **equidistant curve** *of p with respect to Γ.*

As the form of the equidistant curves is depending on the curve Γ we list some properties of the curves that we will need:

Definition 2 *Let $\{\Gamma_r\}_{r\in I}$ be a set of convex curves in $\mathbb{R}^2$ with $I \subset \mathbb{R}$ as the interval of the parameter r. Let each of the Γ_r be the border of a simply connected region $G(\Gamma_r)$. Either $G(\Gamma_r)$ or $\mathbb{R}^2 \setminus G(\Gamma_r)$ is defined to be the interior of Γ_r. Additionally, let the following conditions be fulfilled:*

1. *For each $r, s \in I, r \neq s$ is $\Gamma_r \cap \Gamma_s = \emptyset$. The curves Γ_r do not cross themselves.*
2. *For each $r \in I$, $s \in I$, $r < s$, interior$(\Gamma_r) \subset$ interior(Γ_s) holds.*
3. *For each $x \in \mathbb{R}^2$ there exists $r \in I$ such that $x \in \Gamma_r$.*
4. *Γ_r is continuous with respect to r.*
5. *Γ_r has a derivative for each $r \in I$ and this derivative is discontinuous at only a finite set of points.*

Then $\{\Gamma_r\}_{r\in I}$ is called a **family of sweep curves**. *Each single Γ_r is called the* **sweep curve** *of the parameter r. For each $r \in I$, Γ_r is the border of a convex region which is defined to be the interior of Γ_r. Then the set $\{\Gamma_r\}_{r\in I}$ is called* **family of convex sweep curves**.

With respect to Voronoi diagrams, the following theorem is important:

Theorem 3 *Let $\{\Gamma_r\}_{r\in I}$ be a family of convex sweep curves and let $p, q \in \mathbb{R}^2$, $p \neq q$ be two points. Then we have the following property for the bisector $b(p,q)$ of p and q: each point $u \in b(p,q)$ is an element of the intersection of $E_{\Gamma_r}(p)$ and $E_{\Gamma_r}(q)$ for a certain $r \in I$.*

Conversely, all points of the intersection of $E_{\Gamma_r}(p)$ and $E_{\Gamma_r}(q)$ lie on $b(p,q)$.

Now we consider the case where we have several sites lying in the interior of a given sweep curve. Each of the sites forms an equidistant curve, and the union of the regions bounded by these equidistant curves is bounded by parts of these curves. In the following we will call such a bounding part of an equidistant curve an **arc**. The system of these arcs is called **front**. The following theorem states, that Voronoi edges can be thought of as intersections of certain parts of equidistant curves.

Theorem 4 *Let $\{\Gamma_r\}_{r\in I}$ be a family of convex sweep curves. For each point u on a Voronoi edge e there is an $r \in I$ so that u is an element of the intersection of two parts of equidistant curves. These parts are part of the front for the given parameter r.*

3 The dynamic front

When a family of sweep curves is used in a plane sweep algorithm, it is necessary to examine the behavior of the front while the parameter r is growing. The following is a generalization of Fortune's sweep algorithm (see [2], [4]):

One kind of important changes of the front during the sweep happens when the sweep curve reaches a new site. This is called a **site event**. A new arc is inserted into the front, that is a part of the equidistant curve generated by the new site (and the sweep curve).

As the site has zero distance to the sweep curve, the equidistant curve has the same distance. In contrast, the other arcs of the front, which existed just before the new site was reached, have a positive distance to the sweep curve. Therefore, the new arc cannot be completely inside the front.

Another important kind of change of the configuration of the front happens when an arc drops out of the front. This happens when the two points of intersection of that arc with its neighbouring arcs merge. So we are considering a triplet of consecutive arcs whose middle arc vanishes. The points of intersection both lie on a Voronoi edge. The points merge where two Voronoi edges meet in a Voronoi vertex. The generating sites of the three arcs are lying on a circle whose center is the new Voronoi vertex. Therefore, this kind of event is called **circle event**.

It is important for the correctness of an algorithm based on a given family of sweep curves to show that these two kinds of events (site events and circle events) are the only ones that can happen.

4 Classifying families of sweep curves

In order to see which families of sweep curves are usable in a Fortune-type sweep algorithm it is necessary to take a look at the form of the resulting equidistant curves. It is an essential part of such an algorithm to deal with these curves and the intersections of two of them at a time.

To provide an answer we present here an explicit method of constructing a generalization of the equidistant curve of a point with respect to a given curve: for each point of the sweep curve, with the exception of points where no continuous derivative exists, we construct a candidate for the equidistant curve. These points fulfill a necessary, but not sufficient condition for this curve. Therefore, we call the curve consisting of these candidates the **local equidistant curve**.

Let p be the point generating the equidistant curve, Γ the sweep curve, and $q \in \Gamma$ a point. Further let Γ have a derivative in q, so that a normal vector exists in this point. The intersection of the line generated by this normal vector with the bisector $b(p,q)$ of p and q has the same distance from p and q. Thus, it is a point on the local equidistant curve.

The local equidistant curve for points where the sweep curve does not have a derivative or not a coninuous one, consists of several parts or has bends.

The following theoretical result shows which families of convex sweep curves are usable in the euclidian plane:

Definition 5 *Let Γ be a curve in $\mathbb{R}^2$ and $p \in \mathbb{R}^2$ a point. Let $q \in E_\Gamma(p)$ be a point which has the same distance to more than one point on Γ, thus*

$$|\{x \in \Gamma |\ d(x,p) = d(q,p)\}| > 1$$

Then q is called an **equidistant multipoint** *as the local equidistant curve crosses itself in such a point.*

With the following theorem, we will show that it only makes sense to use those families of convex sweep curves where the equidistant curves do not have equidistant multipoints.

These are the families whose curves have a constant curvature, that are **lines** and **circles**:

Theorem 6 *Let $G \subset \mathbb{R}^2$ be a non-empty, simply connected and convex region and let Γ be its boundary curve. Let Γ have a continuous second derivative nearly everywhere.*

If the curvature of Γ is non-constant, then there is at least one point $p \in \overset{\circ}{G}$ so that $E_\Gamma(p)$ contains an equidistant multipoint.

Conversely, if the curvature of Γ is constant, then there is no such point $p \in \overset{\circ}{G}$, such that $E_\Gamma(p)$ contains an equidistant multipoint.

The basic idea of the proof is to find a circle that is completely inside $\overset{\circ}{G}$ and has at least two points in common with $\Gamma = \partial G$. Its center has the same distance to at least two local parts of the curve Γ. Thus, each point on this circle generates a local equidistant curve that has an equidistant multipoint, namely the center of the circle. As these equidistant multipoints give rise to bends or breaks of the equidistant curve one wants to avoid such sweep curves in a Fortune-type algorithm.

5 The Algorithm

As a plane sweep algorithm for Voronoi-diagrams based on sweep lines already exists, we have implemented such an algorithm based on sweep circle (see [4]). These circles are concentric to one of the n given sites, which is called the *start site.*

In the algorithm based on sweep lines the equidistant curves are parabolas and the front consists of parts of these. Thus, there is a parabolic front. Here we use circles and get ellipses as equidistant curves; therefore the front consists of elliptic arcs and is called *elliptic front.* These ellipses have the property that the start site and the generating site conincide with the focus points. Thus the elliptic front is bounding a star-shaped region and is cyclic.

One can show that the site and circle events are the only events that can happen during the sweep. Also, it can be shown that arcs can enter the front only by a site event and that arcs can leave the front only by circle events. This is important for the correctnes of the algorithm.

The algorithms starts with the set of site events inserted into the sweep queue. These events are known in advance and are ordered by the distance of the site to the start site. The circle events are calculated and inserted into the queue dynamically. During each site event and each circle event, there are two new candidates that have to be checked. Each of these is based on a triplet of consecutive arcs in the front. But because of the cyclic structure of the front, we have to prevent a double insertion of the new events; therefore, we have to make sure that the arcs generating the events are pairwise different and really generate an event.

Another consequence of the front being cyclic is that the sweep tree has to be implemented in a way that realises this cyclic structure. The leaves of this tree represent the arcs while the inner nodes represent the intersections of neighboured arcs. To have a search criteria, we cut the front open and linearize it dynamically; the leftmost node of the tree defines a cut in the front, everything else in the tree then is ordered by its angle to this cut, viewed from the start site.

This algorithm runs in $O(n \cdot \log n)$ time with $O(n)$ space requirements.

References

[1] AURENHAMMER, FRANZ: *Voronoi diagrams – a survey of a fundamental geometric data structure.* ACM Computing Surveys, 23(3):345–405, September 1991.

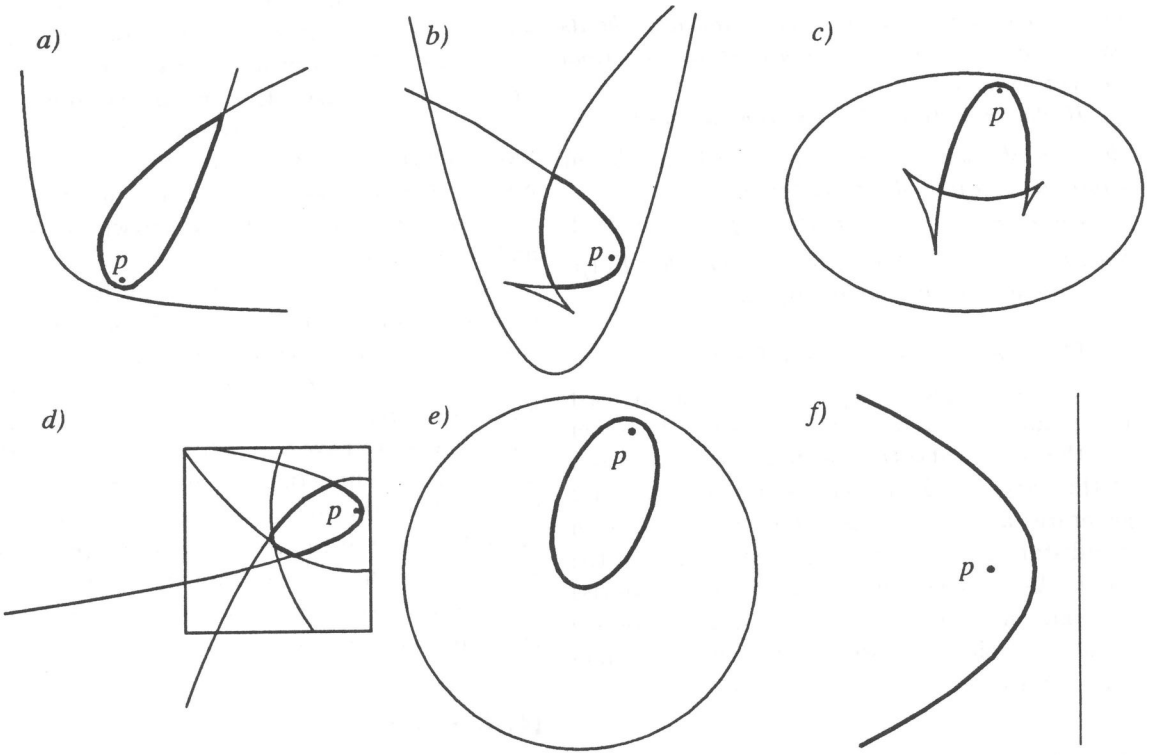
[2] FORTUNE, STEVEN: *Voronoi diagrams and delauny triangulations.* In: DING-ZHU DU, FRANK HWANG (Editor): *Computing in Euclidian Geometry*, pages 193 – 233. World Scientific, 1992.

[3] GUIBAS, L. J. and J. STOLFI: *Ruler, compass and computer: The design and analysis of geometric algorithms.* In: EARNSHAW, R. A. (Editor): *Theoretical Foundations of Computer Graphics and CAD*, pages 111 – 165. Springer Verlag, 1988.

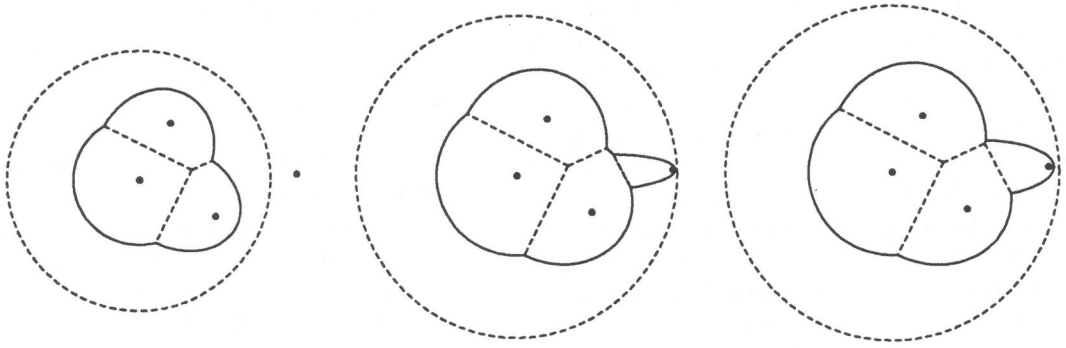
[4] KÜHN, ULRICH: *Berechnung von Voronoi-Diagrammen mit konvexen Fegekurven.* Master's thesis, April 1995.

[5] OKABE, ATSUYUKI, BARRY BOOTS, and KOKICHI SUGIHARA: *Spatial Tessallations. Concepts and Applications of Voronoi Diagrams.* Wiley, 1992.

Here we show some examples of local equidistant curves with respect to different kinds of curves: a hyperbola (a), a parabola (b), an ellipse (c), a square (d), a circle (e) and a line (f). The equidistant curves are emphasized.



Here is an example of the dynamic development of the front during the sweep:



Computing Periodic Voronoi Partitions on the Euclidean Plane

M. Mazón *, D. Bochis†
Universidad de Cantabria, Facultad de Ciencias,
Departamento de Matemáticas, Estadística y Computación,
Av. de los Castros s/n,
39071 Santander, SPAIN

Abstract

Any finite set S of distinct points induce a partition of the plane in polygonal regions , each region defined as the set of points closest to one of the points of S than to any other. This partition is called *the Voronoi partition* of S and several algorithms exist that compute this geometrical structure. The Voronoi partition can also be define for discrete infinite sets of points, but the known algorithms does not work for this case as they deal only with finite sets. In the special case of periodic sets we can prove that the infinite Voronoi partition can be obtained by computing only the Voronoi partition for a certain finite set. We discuss here some of the problems that arise in the implementation.

1 Introduction

Any finite set S of n points on the plane give rise to a cellular decomposition of the plane called the *Voronoi diagram partition.* It is one of the most useful construction associated with such a configuration of points and several algorithms that compute this geometrical structure are well-known. [5]

Infinite sets of points, if they are discrete, also define a partition of the plane ([2]), but can not, in general, be computed, as the known algorithms deal only with finite sets of points.

In the particular case that this infinite set is a of the form GS, with G a crystallographic group and S a finite set of points then, due to the periodic structure of the set, the Voronoi partition has also a periodic structure and can

*marisa@matsun1.unican.es
†dacib@matsun1.unican.es

be computed by computing only the Voronoi partition of a certain finite subset of GS [4].

This periodic Voronoi partitions are useful in modelling and simulating 2-dimensional crystal structures as well as for the computation of the Voronoi partition on the Euclidean 2-orbifolds, that include all the locally Euclidean surfaces [4].

In this work we confine ourselves to the problem of computing these periodic Voronoi diagrams.

2 Crystallographic Groups and Periodic Voronoi Diagrams

A *crystallographic group* is a subgroup G of M, where M is the group of motions of the plane, for which there exists a polygon Π such that :

1. every point in the plane is the image, by some element of G , of a point of Π or its interior, and
2. if $g_1\Pi$ and $g_2\Pi$ have an interior point in common ($g_1, g_2 \in G$), then $g_1\Pi = g_2\Pi$.

The polygon Π is called *a fundamental domain* for the group G. The whole plane can be covered by Π and its images under G, without holes or double coverings.

Every crystallographic group contains a subgroup generated by 2 linearly independent translations.

There are exactly 17 classes of non-isomorphic crystallographic groups. [3]

Given a crystallographic group G, the set $GP = \{gP : g \in G\}$ is the *orbit* of P. If $S = \{P_i\}_{i=1}^{n}$ is a finite set of points , GS denotes the union of the orbits of P_i .

If a point P has trivial stabilizer it happens that any Voronoi region of the orbit of P is a fundamental domain for the group. This particular type of fundamental domain will be called *Dirichlet domain.*

The main theorem on which it is based our procedure, conveniently stated in [4], is the following :

Theorem 1 *Let G be a crystallographic group, D_G a Dirichlet domain, S a set of points in D_G, S^* the intersection of GS with 3-linearly extended D_G. Then*

$$Vor(GS) = G(Vor(S^*) \cap D_G)$$

Note that this result can be generalized for any dimension. (see [1] for dimension 3)

3 Steps in Computing Periodic Voronoi Diagrams

Given a crystallographic group G and a set S of n *initial points* inside a Dirichlet domain corresponding to the group, our goal is to compute the Voronoi diagram for the infinite set generated from S by the action of the group.

In order to compute the Voronoi diagram for such a set we proceed as indicated in the following steps:

Step 1. Choose the group among the 17 crystallographic groups and select its parameters. The parameters one can select are the ratio $r = \frac{|a|}{|b|}$ and the angle $\alpha = \widehat{(a,b)}$ between the vectors, where a and b are translation vectors that generate the subgroup of translations. Note that in some cases we cannot change any of these parameters because of the intrinsic structure of the group.

Step 2. Find a Dirichlet domain for the chosen group. We need this type of fundamental domain in order to apply the theorem in section 2.

Step 3. Give a set of n *initial points* in the Dirichlet domain and generate the orbit of each point (inside the limit of the display). Save apart those which are inside the region generated by linearly extending the initial Dirichlet domain, by a factor of 3. (call the set of these saved points *useful set*)

Step 4. Compute the Voronoi diagram of the *useful set.*

Step 5. Get the n Voronoi regions corresponding with the *initial points* and save them apart. (call these Voronoi regions *useful Voronoi regions*)

Step 6. Generate the whole Voronoi diagram by replicating the *useful Voronoi regions* according to the the group.

References

[1] N. P. Dolbilin, D. H. Huson. *Computing the Delone Complex and Delaney-Dress Symbol of a Periodic Point Set in Euclidean Space* . Bielefeld (1994).

[2] P.E. Ehrlich and H. C. Im Hof . *Dirichlet regions in manifolds without conjugate points.* Comment. Math. Helvetici 54, 642-658 (1979).

[3] G. E. Martin *Transformation Geometry.* Springer-Verlag, (1982).

[4] M. L. Mazón . *Diagramas de Voronoi en caleidoscopios* . Ph.D Th. , Universidad de Cantabria, Spain, (1992).

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

Fixture Design

Mark H. Overmars

Manufacturing and assembly processes often require objects to be held in such a way that they can resist forces applied to it. For example, one wants to hold a piece of metal to drill a hole in it. This problem is often referred to as fixturing. Rather than building a separate solution for each fixturing task, it is desirable to use some modular fixturing techniques that can be reused. Modular fixturing toolkits consist of a fixturing table on which various fixturing elements can be placed to hold the object. Typical elements are pins, clamps and blocks.
In this talk I discuss the power of such fixturing toolkits by identifying what objects can be fixtured with them. Also algorithms are presented to efficiently compute how to fixture a given object.

Extended Abstract

REPRESENTATION OF GEOMETRIC OBJECTS AS SET OF INEQUALITIES[1]

Andrew Frank, Peter Haunold, Werner Kuhn[2], Gabriel Kuper[3]

1. Introduction

Most GIS but also CAD systems represent the objects by their boundaries. Wireframes, winged edges and similar methods are widely used. With these methods, object geometry is defined by a collection of boundaries (surfaces or edges); the relations between these boundaries are usually explicitly stored [Frank, 1987; Oliver, 1988].

As an alternative, objects can be represented by inequalities which describe half-planes (half-spaces in 3D). There are several papers [Rigaux, & Scholl, 1995] see there for earlier papers] which propagate this idea and explore the expressive power and the demands the resolution of such systems of constraints poses. Systems of Linear Constraints (Linear Constraints [Frank, & Wallace, 1995; Freeston, Kuper, & Wallace, 1995]) are relatively easy to solve and powerful methods are known. These methods have been used for a long time in Operation Research and highly efficient programs for their practical solution exist.

If the efficient methods to solve linear constraints can be applied to the geometric problem, much can be gained practically and theoretically. Many of the algorithms for constraint resolution are simpler than computational geometry algorithm and often their structure is also easier to understand and program. For example, the constraints describing an area do not depend on a particular order, but can be reordered to improve processing; the intersection of two areas is found by simply merging the two lists of constraints.

This paper explores whether this alternative is practically viable for the use in GIS. The application of this theoretically attractive representation method depends on practical conditions:

- how much storage is used,
- can large data collections be processed effectively, and
- are the most important operations of GIS implemented efficiently.

This paper takes GIS as an example and uses their typical property [Frank, & Kuhn, 1995] as a benchmark to compare a representation of geometry by inequalities.

2. Storage

Considering the description of a region as a list of inequalities seems to require large amounts of storage, but this is not correct, as will be shown here. In most GIS areal data in form of partitions of space dominate the storage requirements and these are discussed here primarily.

An inequality is represented as two signed real numbers, which is the same as in a regular vector representation (where n points represent the n edges forming a polygon, i.e. for each line in the polygon one coordinate pair is stored). For a closed polygon, there is the same number of inequalities as there are points in the boundary of the area. The data structure to store inequalities can be as simple (or complex) as for the vector representation (e.g. a linked list). For unconnected areas there is no difference in the amount of storage required between a vector representation as closed polygon or a list of inequalities.

$$ax - by \leq c \qquad (1)$$

$$x - \frac{by}{a} \leq \frac{c}{a} \qquad (2)$$

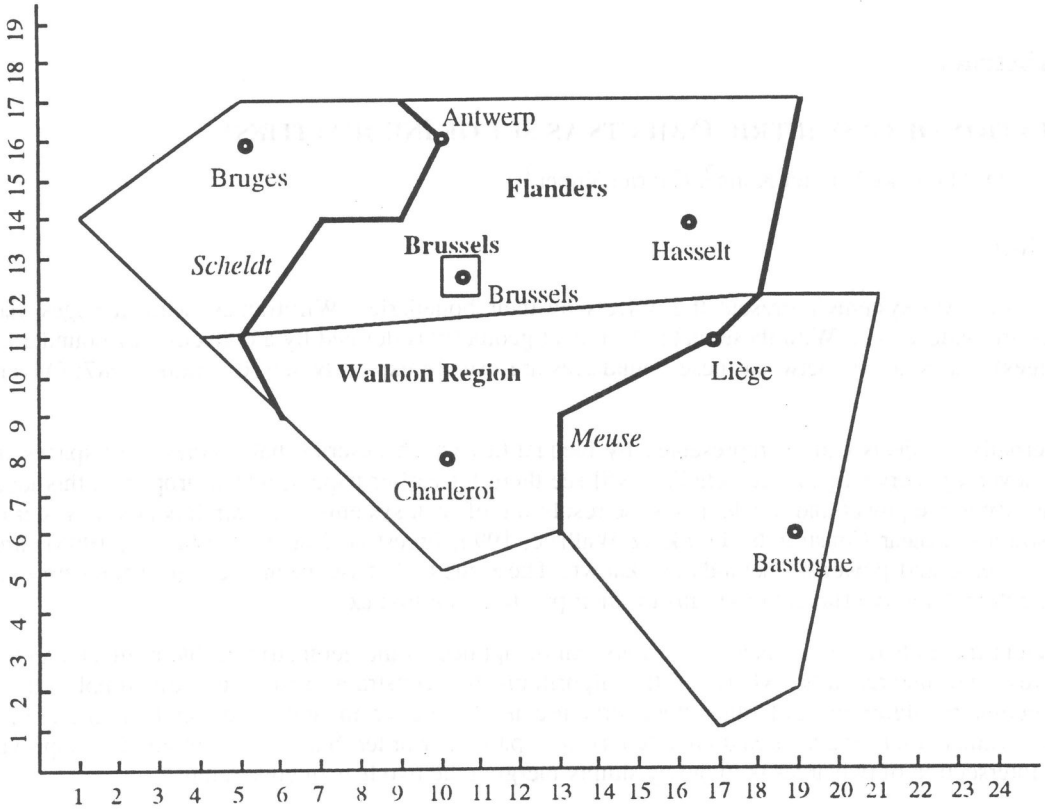
To control point precision better the form (1) may actually be easier than (2).

---

[1] This work is part of the cooperation within the Esprit CONTESSA project.

[2] Address of Andrew Frank, Peter Haunold and Werner Kuhn:
Dept. of Geoinformation, Technical University Vienna, Gusshausstrasse 27-29, A-1040 Vienna,
email: {frank, haunold, kuhn@geoinfo.tuwien.ac.at}

[3] Address of Gabriel Kuper:
European Computer Research Center, Munich, Arabellastr. 17, D-81925 Munich
email: gabriel.kuper@ecrc.de



Representation using inequalities:

Regions

Name	*Geometry*
Brussels	$(y \leq 13) \wedge (x \leq 11) \wedge (y \geq 12) \wedge (x \geq 10)$
Flandres	$(y \leq 17) \wedge (3x - 4y \geq -53) \wedge (x - 14y \leq -150) \wedge (x + y \geq 45) \wedge$ $(4x - y \leq 78) \wedge (\neg((y \leq 13) \wedge (x \leq 11) \wedge (y \geq 12) \wedge (x \geq 10)))$
Walloon Region	...

Cities

Name	*Geometry*
Antwerp	$(x = 10) \wedge (y = 16)$
Bastogne	$(x = 19) \wedge (y = 6)$
Bruges	$(x = 10.5) \wedge (y = 12.5)$
...	...

Rivers

Name	*Geometry*
Meuse	$((y \leq 17) \wedge (5x - y \leq 78) \wedge (y \geq 12)) \vee$ $((y \leq 12) \wedge (x - y = 6) \wedge (y \geq 11)) \vee$ $((y \leq 11) \wedge (x - 2y = -5) \wedge (y \geq 9)) \vee$ $((y \leq 9) \wedge (x = 13) \wedge ((y \geq 6))$
Scheldt	...

Fig. 1. Example of a spatial database [Vandeurzen, Gyssens, & Van Gucht, 1995]

Representation using boundary points - relational model:

Region

Edge ID	*From Point*	*To Point*	*Left Area*	*Right Area*
a	1	2	Belgium	Flanders
b	2	3	Belgium	Flanders
c	3	4	Walloon Region	Flanders
d	4	5	Belgium	Flanders
e	5	1	Belgium	Flanders
...				

Tab. 1. Relational representation for one region only

For actual GIS applications, where the distribution of land to owners is recorded (so called parcel data), categorical coverage [Chrisman, 1987] or partitions of space (figure 1) and regions must share boundaries. For each line (or inequality) a pointer to the left and right area must be included (table 1). Edges are represented by pointers to the start and end-point. For nodes with more than two adjacent edges, the storage for the point coordinates are shared with more edges, reducing average storage requirements (compensated by a need for pointers).

Practical example: An actual parcel map at the 1:2000 scale from the Austrian cadastre shows in an area of 15 by 15 cm:

48 Parcels 1.6 Edges/Node

283 Nodes 5.9 Nodes/Parcel

180 Edges 3.8 Edges/Parcel

Formula (1) gives a table structure for the edges as in figure 1. This results in a storage requirement per edge of 2 pointers and 3 integers (corresponding to one coordinate value). This compares with the 4 pointers to the table of coordinates and the 2 coordinate values for each point (of which come on average 1.6 per edge). If pointers and coordinate values take the same amount of space, this gives 5 values per edge for inequalities and 5.6 values for the vector representation. This is for practical purposes the same storage amount.

Id	**a**	**b**	**c**	**left**	**right**
a	0	1	17	Belgium	Flanders
b	3	-4	-53	Belgium	Flanders

The most obvious difference is the lack of order information in the Linear Constraints and of explicit representation of points.

- Because Linear Constraints does not require order, an overlay operation to find the common areas for two tessalations, is only merging the two sets of Linear Constraints. The computation of an overlay does not require any floating point operations; these become only necessary at the end to calculate the boundary points (if this is required). Points for intermediate results are not calculated and the inequalities for the boundaries are not changed during calculation (this excludes the creeping of points during an overlay operation [Guevara, 1985]).
- Boundary points (corners) are very important for the human perception and cognition of parcels. They can be calculated from the Linear Constraints representation. Corners are typically necessary for the calculation of area using Gauss' formula, but there may exist more direct way to calculate area from Linear Constraints.

We conclude this section with the observation that the representation of a data set for a typical GIS application, e.g. representing cadastral parcel geometry, in Linear Constraints requires a comparable amount of storage as would a pointer-based vector representation.

3. Effective processing of large data collections

The data collection for a GIS is large. In [Frank, & Kuhn, 1995; Goodchild, 1990] a typical value of 10 - 400 Gigabytes is expected. It is thus completely excluded that for deciding a simple 'point in polygon' or a 'range' query the complete data collection can be checked. It is necessary to build a spatial access method [Ester, Kriegel, & Xu, 1995; Frank, & Kuhn, 1995; Hjaltason, & Samet, 1995].

3.1. Requirements for Spatial Access Methods

Spatial access methods allow to select quickly the data relevant for a problem and exclude the large majority of the stored data from consideration and thus from processing. The spatial access method abstract from detailed properties of the geometry of an object, but preserve enough information that one can decide quickly if an object (or large subsets of objects) can be excluded from consideration. The object geometry is typically represented as minimal bounding rectangles (MBR) or similar compact generalizations of the form and location of an object. Methods using other geometric forms, e.g. circles, were proposed in [Finke, & Hinrichs, 1995]. Processing is more selective but slower than for the rectangular methods. The advantage does not seem large enough to be tried in practice.

Spatial Access Methods require that for each object to be selected based on location a simple geometric abstraction can be supplied (i.e. a Minimal Bounding Rectangle). The selection operation is then based on this simpler geometric abstraction and must produce a superset of the desired result (i.e. false positives are allowed, but no false drops).

3.2. Minimal Bounding Rectangles for Linear Constraints

For inequalities minimal bounding rectangles do not apply, but the rectangles can be formed around the areas (i.e. a collection of inequalities). Object selection is then based on MBR and results in the collection of areas to consider.

The well known tree structures using the minimal bounding rectangle (MBR) representation of stored objects can be used, even if the objects are stored as a set of linear inequalities [Ester, Kriegel, & Xu, 1995; Güting, de Ridder, & Schneider, 1995; Hjaltason, & Samet, 1995]. Selection based on comparing the MBR retrieves all the inequalities necessary for the standard operations.

In principle, MBR could also be associated with the edges, both in the vector or in the Linear Constraints representation. The difference is only that the MBR cannot be calculated from the Linear Constraints alone, but the area it bounds and all the constraints defining it must be available. The MBR then represents the 'active' part of the constraints, namely the piece that is forming a boundary.

3.3. Generalization

Minimal bounding rectangles can be used not only for spatial access methods but generally for geometric operations. A corresponding operation is applied to the reduced representation and only objects passing this test must be included in the accurate computation. The test based on the reduced representation (e.g. MBR) is orders of magnitudes faster than the accurate test. The reduction and the two step processing is increasing performance. The accurate tests run always with very small datasets and their asymptotic behavior plays therefore a lesser role (the quick tests are nearly always linear in their time).

This generalized method of selection of relevant objects extends to the Linear Constraints representation. For most operations on objects, e.g. calculation of the intersection, the union, the boundary of a collection of areas etc., only the inequalities of the areas involved are necessary.

4. Conclusion

First considerations show that it is possible to represent the kind of data in a GIS as linear constraints. The representation as constraints affords some advantages as the representation does not depend on order as does the vector representation. Overlay processing should be faster and show less algorithmic problems. Buffer formation is much simpler and should work as well in 3D (a currently unsolved problem). Processing of inequalities generalizes to the 3D space [Raper, 1989].

Storage requirements for the Linear Constraints representation and for a vector representation have been compared for a typical case of GIS data, namely a 1:2000 cadastral plan. The storage amounts are very similar (10% more for Linear Constraints, which is an insignificant difference).

The known methods for spatial access are based on an abstract representation of geometry (typically MBR). These methods can be applied to the Linear Constraints representation as well as for a vector

representation. There are no apparent reasons, why an Linear Constraints or vector representation should perform faster.

Linear Constraints is the most significant revolutionary idea for the representation of spatial data. In contrast to several proposals to slightly improve a vector representation [Finke, & Hinrichs, 1995], Linear Constraints is a completely different concept. It promises some advantages over vector processing and there are no immediately visible drawbacks, as established here.

4.1. Future Work

Much remains to be done. Detailed algorithms for the standard operations in a GIS must be developed. Operations like

- point in polygon
- determination of boundary points of a polygon
- area of a polygon
- intersection of two polygons
- intersection of two partitions (overlay)
- construction of a buffer zone around a given object

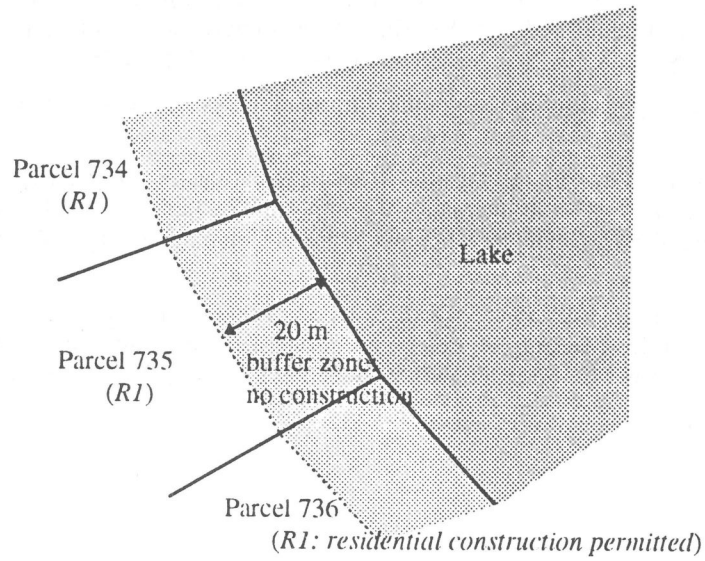


Fig. 2. Buffer zone along a lake shore

Most of these operations can be implemented efficiently using a representation with inequalities, but detailed studies considering the particular load presented by a GIS are needed. Of particular interest are buffer-zone operations (figure), which are notoriously difficult for boundary representations, but very easy for the representation with inequalities, even in 3D.

References:

Chrisman, N. 1987. Fundamental principles of geographic information systems. (Chrisman, N. R., Eds.). Auto-Carto 8, Baltimore, MA. ASPRS & ACSM. Eighth International Symposium on Computer-Assisted Cartography, pp: 32-41.

Ester, M., Kriegel, H.-P., & Xu, X. 1995. Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification. (Egenhofer, M. J., & Herring, J., Eds.). Fourth International Symposium on Large Spatial Databases - SSD 95, Portland ME. Springer Verlag, Heidelberg. Lecture Notes in Computer Science, Vol. 951, pp: 67-82.

Finke, U., & Hinrichs, K. H. 1995. The Quad View Data Structure - A Representation for Planar Subdivisions. (Egenhofer, M. J., & Herring, J., Eds.). Fourth International Symposium on Large Spatial Databases - SSD 95, Portland ME. Springer Verlag, Heidelberg. Lecture Notes in Computer Science, Vol. 951, pp: 29-46.

Frank, A. U. 1987. Overlay Processing in Spatial Information Systems. (Chrisman, N. R., Eds.). Eighth International Symposium on Computer-Assisted Cartography (AUTO-CARTO 8), Baltimore, MD, pp: 16-31.

Frank, A. U., & Kuhn, W. 1995. Specifying Open GIS with Functional Languages. (Egenhofer, M. J., & Herring, J., Eds.). Fourth International Symposium on Large Spatial Databases - SSD 95, Portland ME. Springer Verlag, Heidelberg. Lecture Notes in Computer Science, Vol. 951, pp: 184-195.

Frank, A. U., & Wallace, M. 1995. Constraint Based Modeling in A GIS: Road Design as a Case Study. Auto-Carto 12, Charlotte, North Carolina (Feb. 27 - March 1).

Freeston, M., Kuper, G., & Wallace, M. G. 1995. Constraint Databases. Conference on Information Technology and its Use in Environmental Monitoring and Protection, Holloway College.

Goodchild, M. F. 1990. Tiling a Large Geographical Database. (Buchmann, A., et al., Eds.). Symposium on the Design and Implementation of Large Spatial Databases, New York, NY. Springer-Verlag.

Guevara, J. A. 1985. Intersection Problems in Polygon Overlay. Auto-Carto 7, Washington, D.C., March 11-14, 1985. ACSM/ASPRS.

Güting, R. H., de Ridder, T., & Schneider, M. 1995. Implementation of the ROSE.

Hjaltason, G. R., & Samet, H. 1995. Ranking in Spatial Databases. (Egenhofer, M. J., & Herring, J., Eds.). Fourth International Symposium on Large Spatial Databases - SSD 95, Portland ME. Springer Verlag, Heidelberg. Lecture Notes in Computer Science, Vol. 951, pp: 83-95.

Oliver, S. G. 1988. A GIS Automation of County Tax Assessor Maps and Their Application for Siting the Superconducting Super Collider in Illinois. GIS/LIS'88, Third Annual International Conference, San Antonio, Texas. ACSM, ASPRS, AAG, URISA. Accessing the World, Vol. 1, pp: 448-458.

Raper, J. 1989. GIS: Three Dimensional Applications in Geographic Information Systems. London: Tayler and Francis.

Rigaux, P., & Scholl, M. 1995. Multi-Scale Partitions: Application to Spatial and Statistical Databases. (Egenhofer, M. J., & Herring, J., Eds.). Fourth International Symposium on Large Spatial Databases - SSD 95, Portland ME. Springer Verlag, Heidelberg. Lecture Notes in Computer Science, Vol. 951, pp: 170-183.

Vandeurzen, L., Gyssens, M., & Van Gucht, D. 1995. On the Desirability and Limitations of Linear Spatial Database Models. (Egenhofer, M. J., & Herring, J., Eds.). Fourth International Symposium on Large Spatial Databases - SSD 95, Portland ME. Springer Verlag, Heidelberg. Lecture Notes in Computer Science, Vol. 951, pp: 14-28.

Exact Volume Computation for Polytopes: A Practical Study

B. Büeler, A. Enge, K. Fukuda and H.-J. Lüthi
Institute for Operations Research
Swiss Federal Institute of Technology
CH-8092 Zurich, Switzerland

January 22, 1996

Extended Abstract

1 Introduction

A *polytope* P is the convex hull $conv(V)$ of a finite set $V = \{v_1, v_2, \ldots, v_n\}$ of points in $\mathbb{R}^d$. Equivalently, it is a bounded subset of $\mathbb{R}^d$ which is the intersection of a finite set of half spaces. When $P = conv(V)$, V is called a *vertex representation* or simply *$\mathcal{V}$-representation* of P. When $P = \{x | A\ x \leq b\}$ for some $m \times d$ real matrix A and m-vector b, the pair (A, b) is called a *halfspace representation* or simply *$\mathcal{H}$-representation* of P.

The *volume computation problem* is to compute the volume $\mathrm{Vol}(P)$ of a polytope P given by both $\mathcal{V}$- and $\mathcal{H}$-representations. One might consider the problem in which only one of the two representations is given in advance, but in order to compare different algorithms, our definition appears to be most appropriate. (We shall briefly discuss one-representation cases.) We assume that the given representations are minimal, since the removal of redundancy in a representation is essentially a linear programming problem which can be solved efficiently.

In recent years the complexity discussion of volume computation for d-polytopes (polytopes in d dimensions) has attracted many researchers. While there are many theoretical papers on this problem and some polynomial algorithms are known for certain special cases (see [2] for a comprehensive survey of important theoretical results), the complexity of the volume computation is unknown in general. In fact, it is not even clear if it is in NP, although the (binary) size of the volume itself is a polynomial function of the size of a $\mathcal{V}$-representation.

Despite the theoretical hardness of the problem, steadily growing speed of computers and efficient implementations of old and new algorithms for polyhedral computation enable us to solve relatively large scale problems in a reasonable time. In this paper we study several exact algorithms and techniques from the viewpoint of practical computation and compare them experimentally. The main goal is to acquire useful knowledge about how to choose the best algorithm for a given problem type.

The algorithms we study here fall into three classes: (A) Triangulation Methods, (B) Recursive Methods and (C) Sweeping-plane Methods. We briefly describe the algorithms in each class below.

(A) Triangulation Method

The idea behind volume computation by means of triangulation of the polytope is based on the well known formula

$$\mathrm{Vol}(\Delta(v_0,\ldots,v_d)) = \frac{|\det(v_1 - v_0,\ldots,v_d - v_0)|}{d!}$$

where $\Delta(v_0,\ldots,v_d)$ denotes the simplex in $\mathbb{R}^d$ defined by $v_0,\ldots,v_d$. Summing up all simplex volumes of a triangulation gives the volume of the polytope. There are many different ways to triangulate a polytope, each of which yields a different algorithm for the volume computation.

Triangulation by Cohen and Hickey

The triangulation method of Cohen and Hickey is based on the following theorem [1]:

Theorem 1 *Let f_k be the number of faces of dimension k of P, $0 \leq k \leq d$. Let $e_j^k, 1 \leq j \leq f_k$, be an arbitrary numbering of the faces of dimension k. Finally, let $\eta(e_j^k)$ denote the vertex e_i^0 of e_j^k with minimal i. Consider all decreasing chains of faces*

$$e_1^d \supset e_{j_{d-1}}^{d-1} \supset \cdots \supset e_{j_1}^1 \supset e_{j_0}^0,$$

such that

$$v_0 = \eta(e_{j_0}^0), \ldots, v_d = \eta(e_1^d)$$

are all distinct. Then the set of $\Delta(v_0,\ldots,v_d)$'s is a triangulation of e_1^d.

Delaunay Triangulation

Given the set of vertices V of P; add one dimension defined by the sum of the squares of the components, that is, move all vertices "up" onto the surface of a paraboloid:

$$v' = (v_1,\ldots,v_d,\sum_{i=1}^{d} v_i^2) \in \mathbb{R}^{d+1}.$$

Compute the convex hull of the thereby defined V'. When the points in V' are in general positions, all facets looking "downward" are simplices and form a triangulation of P when projected to the original d-space. Of course, lexicographic perturbation technique can be used to symbolically reallocate the points to satisfy the condition.

Boundary Triangulation by Convex Hull

To triangulate the polytope P, one can first triangulate the boundary of P and then extend the triangulation to a triangulation of P by linking all $(d-1)$-simplices with any fixed interior point. One way to find a boundary triangulation of P is by first perturbing the vertices lexicographically and then computing the convex hull. The resulting polytope is simplicial and yields indirectly a triangulation of P.

(B) Recursive Method

Lasserre's Recursive Algorithm

The volume is computed via a recursive scheme based on Euler's formula for homogeneous functions [3]. Denote by $V(d, A, b)$ the volume sought, by a_i the i^{th} row of

A, and by $V_i(d-1, A, b)$ the volume in $\mathbb{R}^{d-1}$ built by $P \cap \{a_i^T x = b_i\}$. Then the volume can be expressed through the following recursive structure

$$V(d, A, b) = \frac{1}{d} \sum_{i=1}^{m} \frac{b_i}{\|a_i\|} V_i(d-1, A, b).$$

It is of critical importance that no constraint appears more than once in (any recursively generated) $\mathcal{H}$. At the bottom of the recursion tree the volume of a line segment bounded by (at most) $m - d + 1$ constraints has to be computed.

(C) Sweep-Plane Method

Lawrence's Volume Formula

Let us consider a *simple* polytope P, i. e. each vertex lies in exactly d hyperplanes. Choose a vector $c \in \mathbb{R}^d$ and a scalar q such that the function $x \mapsto c^T x + q$ is not constant on any edge. For each vertex $v \in V$, the set of vertices, let A_v be the $d \times d$-matrix composed by the rows of A which are binding at v. Then A_v is invertible and $\gamma^v = A_v^{-1} c$ is well defined. The assumption imposed on c assures that none of the entries of γ^v is zero. Lawrence shows that under these assumptions

$$\mathrm{Vol}(P) = \sum_{v \in V} \frac{(c^T x + q)^n}{d! \,|\det A_v| \prod_{i=1}^{d} \gamma_i^v}.$$

2 Computation Codes Used

cdd (the double description method in C)

It implements the 'Double Description Method' of Motzkin *et al.* for generating all vertices (i.e. extreme points) and extreme rays of a general convex polyhedron in $\mathbb{R}^d$ given by a system of linear inequalities. By duality this implies that the program can also compute the convex hull. We used this code to compute the Delaunay triangulation of the vertex set V and to translate between the two possible representations. The author is Komei Fukuda (fukuda@ifor.math.ethz.ch).

Cohen & Hickey triangulation in C

The implementation is due to A. Enge (enge@ifor.math.ethz.ch). The algorithm itself is purely combinatorial and therefore avoids numerical instabilities: The defining hyperplanes are stored as ordered lists of the vertices they contain. The decreasing chains of faces are computed following a recursive scheme: given e^{k+1}, intersecting this face with all hyperplanes yields all possible candidates for e^k. Degeneracy does not need special handling. The following items are specific to this implementation:

- Precomputing hyperplane intersections which are needed several times speeds up the computation while increasing the memory requirements.
- The simplex volumes are computed by Gaussian elimination; it is possible to choose between partial and total pivoting and no pivoting at all.
- There is an implementation with exact arithmetic.

lrs (lexicographical reverse search in C)

This code enumerates all vertices by reverse search. Degeneracy is resolved by lexicographical perturbation. This yields at the same time a boundary triangulation when applied to compute the convex hull. If the origin lies in the interior of the polytope the program computes the volume via the boundary triangulation with the origin as the interior vertex. The program outputs all dual bases and allows thereby the application of Lawrence's volume formula. The author is David Avis (avis@cs.mcgill.ca).

qhull (quick hull in C)

This is an implementation of the beneath-beyond method for the convex hull computation which is due to C. B. Barber (bradb@geom.umn.edu) and H. Huhdanpaa (hannu@geom.umn.edu). We used this code to compute the Delaunay triangulation via lifting points to a paraboloid.

Lasserre (in C)

This implementation is due to B. Büeler (bueeler@ifor.math.ethz.ch). It contains the following elements:

- Detection and elimination of identical constraints in each recursion. This is a condition for the correctness of the volume formula.
- Elimination of some of the superfluous constraints during the generation of new systems of inequalities.
- Normalization of the constraints in each recursion to make the algorithm numerically more robust.
- The callable library of cplex could be used to detect systems with zero volume. Because this slowed down the program, it was omitted in the computation of the examples given.
- Translation of one vertex into the origin and thereby generating zeros in the right hand side: $\mathrm{Vol}(Ax \leq b) = \mathrm{Vol}(A(x + x^*) \leq b) = \mathrm{Vol}(Ax \leq (b - Ax^*))$. This speeds up computation considerably.
- Saving intermediate results speeds up computation very much.

While the last two items result in a tremendous speedup which is in the later case only limited by the memory available, the first and third item are computationally a big burden.

Lawrence's Formula Computation

Implementation of Lawrence's formula is straightforward except for the choice of c and q. As there is no obvious choice we use random values, hoping that $c^T x + q$ is not constant on any edge. Even if this is the case a bad choice may result in a disastrous numerical behavior because the entries of γ^v may be near to zero. In the case of a non-simple polytope, running lrs on the constraints will result in a simple polytope by lexicographical perturbation. Interpreted in terms of the original constraints Lawrence's formula can be applied in the same way.

3 Experimental Results

At the time of writing, we are still making various computational experiments. Computational time varies drastically from one problem type to another, even for one fixed algorithm and implementation. Thus to compare different algorithms requires extra caution.

Just to give the reader some rough idea on computational time, we shall present a partial result of our experiments below. The examples investigated fall into the following classes:

- r/c/8: eight-dimensional problems with randomly generated constraints. The constraints are constructed by generating integral vectors on a sphere with radius 1000. The resulting polytopes are (most probable) simple.
- r/v/8: eight-dimensional convex hulls of randomly generated vertices on a sphere with radius 1000. The resulting polytopes are (most probable) simplicial.
- cube: hypercubes in the appropriate dimension. Cubes are simple.
- Fm-d:
- ccp5: complete cut polytope on 5 vertices.
- cross$d$: Cross polytope of dim d; it is the dual of the d-dimensional cube.

Numerical results

All computations were done on a HP 7000/735-99 with 144 MB RAM. In case of 'cdd' the floating point implementation was used, thus the only code with precise arithmetic used in the following table is 'lrs'.

In the following table each example is presented together with its dimension d, the number of constraints m and the number of vertices n. For each volume computation method, the following items are listed:

- the CPU-time measured in seconds
- the volume computed
- the number of determinants computed, i.e. the number of simplices in the case of a triangulation method or the number of evaluations of Lawrence's formula
- a condition number which is the volume of the largest simplex divided through the volume of the smallest simplex in case of triangulations; the number shown is the exponent to the base 10. In case of Lawrence this denotes the maximal condition number of the negative and the positive summands.

The column headers denote the (abbreviated) volume computation methods:

ch	Cohen-Hickey-Triangulation with input both the vertices and incidences
chp	Cohen-Hickey-Triangulation using only the constraints as input
cdd+	Delaunay triangulation with 'cdd'
cdd-	Farthest direction Delaunay triangulation with 'cdd'
qhull	Delaunay triangulation with 'qhull'
lawd	Lawrence's formula in the general case using 'lrs' for computing all lexicographically feasible cobasis
lawnd	Lawrence's formula in the non degenerate case
lass	Lasserre's method

		Boundary tr.	Cohen & Hickey triang.		Delauney triang.			Lawrence		Lasserre
Example		lrs	ch	chp	cdd+	cdd-	qhull	lawd	lawnd	lass
r/c/8/20	time		120	127	6.83	7.58	c	45.87	0.96	13.9
d=8	vol.		37576.301	37576.301	119.081	0.05		$-9.67 \cdot 10^{10}$	37576.301	37576.301
m=20	# det		396784	411047	5	35		1115	1115	
n=1115	cond.		17	17	1	4		28	27	
r/c/8/25	time		629	579	d	d	c	127	2.17	135
d=8	vol.		785.989	785.989				$-3.27 \cdot 10^{10}$	785.989	785.989
m=25	# det		1631706	1568094				2596	2596	
n=2596	cond.		21	22				42	41	
r/v/8/10	time	0.7	0.2	2546.2			0.3			34
d=8	vol.	$1.40964 \cdot 10^{19}$	$1.40964 \cdot 10^{19}$	$1.40964 \cdot 10^{19}$			$1.40964 \cdot 10^{19}$			$1.409636 \cdot 10^{19}$
m=24	# det		4	1590735			4			
n=10	cond.		0	51			0			
r/v/8/11	time	1.4	0.2				0.3			380000
d=8	vol.	$3.04772 \cdot 10^{18}$	$3.04772 \cdot 10^{18}$				$3.04772 \cdot 10^{18}$			$3.04756 \cdot 10^{18}$
m=54	# det		16				14			
n=	cond.		1				2			
r/v/8/12	time	2.5	0.2				0.3			
d=8	vol.	$4.38579 \cdot 10^{19}$	$4.38579 \cdot 10^{19}$				$4.38579 \cdot 10^{19}$			
m=94	# det		32				28			
n=	cond.		2				2			
r/v/8/13	time	3.6	0.2				0.3			
d=8	vol.	$1.33411 \cdot 10^{20}$	$1.33411 \cdot 10^{20}$				$1.33411 \cdot 10^{20}$			
m=131	# det		35				77			
n=	cond.		4				3			
r/v/8/14	time	6.3	0.4				0.4			
d=8	vol.	$2.15655 \cdot 10^{20}$	$2.15655 \cdot 10^{20}$				$2.15655 \cdot 10^{20}$			
m=218	# det		78				110			
n=	cond.		2				2			
r/v/8/20	time	45.28	14.1				1.0			3443.30^b
d=8	vol.	$2.69179 \cdot 10^{21}$	$2.69179 \cdot 10^{21}$				$2.69179 \cdot 10^{21}$			
m=1191	# det		683				895			
n=20	cond.		3				6			
r/v/8/30	time	248.00	295.7				4.4			
d=8	vol.	$7.35016 \cdot 10^{21}$	$7.35016 \cdot 10^{21}$				$7.35016 \cdot 10^{21}$			
m=4482	# det		3262				4400			
n=30	cond.		5				6			
r/v/10/12	time	1.5	0.1				0.4			40200
d=10	vol.	$2.13595 \cdot 10^{22}$	$2.13595 \cdot 10^{22}$				$2.13595 \cdot 10^{22}$			$2.278256 \cdot 10^{22}$
m=35	# det		5				5			
n=12	cond.		1				0			
r/v/10/13	time	3.9	0.2				0.2			
d=10	vol.	$1.63289 \cdot 10^{23}$	$1.63289 \cdot 10^{23}$				$1.63289 \cdot 10^{23}$			
m=89	# det		23				19			
n=13	cond.		3				2			
r/v/10/14	time	6.3	0.3				0.4			
d=10	vol.	$2.93136 \cdot 10^{23}$	$2.93136 \cdot 10^{23}$				$2.93136 \cdot 10^{23}$			
m=177	# det		41				61			
n=14	cond.		2				4			
cube9	time	72306	103	107	e		477	2.19	0.5	0.12
d=9	vol.	512	512.000	512.000			512.000	512.000	512.000	512.000
m=18	# det		362880	362880			245093	512	512	
n=512	cond.		0	0			1	21	21	
cube10	time		1413	1470	e		a	4.7	0.52	0.25
d=10	vol.		1024.000	1024.000				1024.000	1024.000	1024.000
m=20	# det		3628800	3628800				1024	1024	
n=1024	cond.		0	0				23	23	
cube12	time				e			25.7	3.1	1.36
d=12	vol.							4096.000	4096.000	4096.000
m=24	# det							4096	4096	
n=4096	cond.							27	27	
cube14	time				e					14.1
d=14	vol.									16384.00
m=24	# det									
n=16384	cond.									
Fm-5	time	2.4	0.5		d		0.7			0.66
d=10	vol.	7109.486	7109.486				7109.486			7109.48571
m=25	# det		28				324			
n=	cond.		0				1			
Fm-6	time	210365	1811					18473		13600
d=15	vol.	286113.55	286113.55					$6.04 \cdot 10^{23}$		286113.546485
m=59	# det		997978					995422		
n=	cond.		5					46		
ccp5	time	3.0	2.8		d		0.5			3833
d=10	vol.	2.31168	2.31168				2.31168			0.00226
m=56	# det		126				124			
n=16	cond.		0				0			
cross6	time	0.3	3.3		d		0.3	12.4		
d=6	vol.	0.08889	0.08889				0.08889	50.07		
m=64	# det		32				32	1416		
n=12	cond.		0				0	11		
cross8	time	0.8			d		0.4			
d=8	vol.	0.00634921					0.00634921			
m=256	# det						128			
n=16	cond.						0			

Notes referred to in the table:

a) qhull could not finish because it ran out of memory.

b) Out of memory. Could be changed by storing fewer intermediate results, but then time usage would increase dramatically.

c) qhull reported precision errors and stopped execution.

d) cdd did not return a triangulation.

e) cdd cannot be used for computing the cubes, it does not return a triangulation.

Some observations and conclusions

- There are huge differences — several orders of magnitude — in CPU-time for the different implementations. It is therefore worthwhile to take into account the structure of the problem when choosing an algorithm.
- The computational effort needed to transform between $\mathcal{V}$- and $\mathcal{H}$-polytopes could be higher than the effort for the volume computation itself. Therefore the representation given can be an important criterion when choosing an approach. However, for most larger problems tested it was comparably cheap to change the representation with cdd.
- Generally spoken, we observed a lot of practical problems. This comprises memory problems (e.g. qhull seems to be very greedy), numerical instabilities, etc.
- Triangulation methods behave somehow dual to Lasserre's method: whereas the former behave best in the case of simplicial polytopes, simple polytopes are preferably solved by the latter. Or more generally expressed: the lower the ratio m/n (number of constraints over number of vertices) is, the better behaves Lasserre compared to triangulation methods, and the higher this ratio is the better behave triangulation methods.
- In a specific class of almost randomly chosen sets of constraints we observed considerably fewer triangles and better conditioning with the Cohen & Hickey triangulation than with the Delaunay triangulation.
- Cohen & Hickey triangulation turned out to be very robust from the numerical point of view and was in general extremely close to the exact solutions (12 digits or more) sake to its combinatorial nature.
- Delaunay triangulations exhibit numerical problems in many examples. For specially structured problems like hypercubes they even break down.
- Numerical instabilities and the unresolved problems of choosing the objective function are serious drawbacks of Lawrence's approach when applied to practical problems.
- Lasserre's method profits a lot of more memory for storing intermediate results. The same applies in a less dramatic way for Cohen & Hickey.

References

[1] Jaques Cohen and Timothy Hickey. Two algorithms for determining volumes of convex polyhedra. *Journal of the ACM*, 26(3):401–414, July 1979.

[2] Peter Gritzmann and Victor Klee. On the complexity of some basic problems in computational convexity: Volume and mixed volumes. Technical Report 94-07, Universität Trier, 1994.

[3] J.B. Lasserre. An analytical expression and an algorithm for the volume of a convex polyhedron in $\mathbb{R}^n$. *J. of Optimization Theory and Applications*, 39(3):363–377, 1983.

Answering Line Segment Intersection Queries Based On Sample Answers

(Extended Abstract)

André Hinkenjann, Markus Kukuk, Heinrich Müller
Universität Dortmund, Informatik VII, 44221 Dortmund, Germany
email: {hinkenja|kukuk|mueller}@ls7.informatik.uni-dortmund.de

The purpose of a query problem is to find out in which of several possible given classes a query subject lies. Typical examples of geometric query problems are

Point Location in d-Space.

Input. A disjoint partition of a region in d-dimensional space into a finite number of cells.

Output. For an arbitrary query point, the cell into which the point falls.

Line Segment Intersection Query.

Input. A finite scene of line segments in the plane.

Output. For an arbitrary query line segment the set of line segments intersected by the query segment.

Line segment intersection queries in a more general 3D-version are of particular interest with ray tracing techniques applied for instance in computer graphics.

There are numerous other query problems emerging from applications. For many of them efficient algorithmic solutions exist. They preprocess the input into a data structure which allows to answer arbitrary queries quickly. The data structures are designed for the particular problem. However, many of the query problems can be reduced to a special type of problem, a point location problem of suitable dimension.

For the line segment intersection query, a reduction to point location can be achieved by representing a query line segment by a 4-d point. The first two coordinates of the 4-d point represent one of the end points of the line segment, the other two define those of the other end point of the segment. The set of all 4-d points representing all those query line segments intersecting exactly those scene segments contained in a given subset of the scene define a cell in 4-d space. Each subset induces a cell which may be empty. Only the nonempty ones are further considered. These cells are disjoint. Evidently, the line segment intersection query can now be solved by point location: location of the 4-d point belonging to a line segment replies a cell which represents just those scene segments intersected by the query line segment.

This approach of unification sometimes is hard to handle in practice. The dimension of the point location space may become high, and the number of cells considerable. Also the geometric shape of the cell can be quite complex.

A possibility of complexity reduction is opened if we abandon to answer a query correctly. It might be sufficient that only about 90% of the answers are correct. For the example of line segment intersection query, it might be acceptable that sometimes the set of reported scene segments intersecting the query segment does not contain all scene segments really intersected.

Such "inexact" solutions to query problems can be obtained by choosing a set of representative sample queries, given by points $\mathbf{p}_i$ in the point location space introduced above. Each

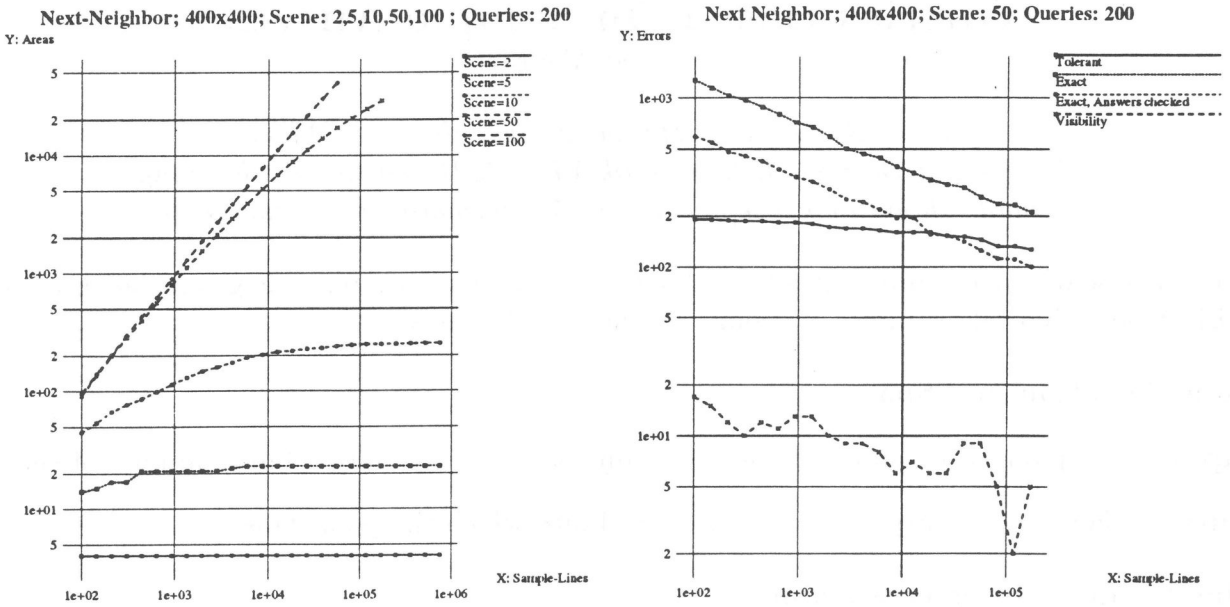


Figure 1: Left: Number of regions of identical answers found, dependent on the number of sample answers. Right: Number of wrong answers dependent on the number of sample answers.

$\mathbf{p}_i$ refer to the cell r_i containing it. Based on these sample points, an "application" query represented by a point $\mathbf{p}$ in the point location space is answered with the cell r_i of a closest sample point $\mathbf{p}_i$.

Finding a closest neighbor of an arbitrary query point in a given set of sites is a classical problem of computational geometry and efficient algorithmic solutions are known [Preparata, Shamos]. However, it does not seem to be quite clear how to get practical efficient solutions for spaces of higher dimensions and a large number of sites.

For the efficient calculation of the answers to the sample queries we can use the fact that the answers can be calculated "in batch", for all queries together. Often, this may be efficiently performed with a sweeping approach. For the line segment intersection query problem, e.g. the algorithms of Bentley and Ottmann or Chazelle and Edelsbrunner can be taken as staring point [Preparata, Shamos].

Once the answers to the sample queries are calculated, they have to be stored efficiently. For the example of line segment intersection queries, an answer may consist of up to n segments, that is all segments in a scene of size n. Thus storing the answers explicitly requires considerable space since the number of different answers may become quite high. Thus a compressed representation is required which allows random reconstruction of particular answers. For that purpose, list compression techniques might be useful [Sarnak,Tarjan].

The main question for the usefulness of this approach, however, is how many samples are necessary to obtain answers of sufficient precision. In this contribution, this question is investigated experimentally for line segment intersection queries. Test scenes of line segments, as well as sample queries and application queries, are randomly generated within a square-shaped region of the plane. Some results are displayed in the figures of this paper.

In figure 1, left, the dependency of the number of different regions of identical answers on the number of samples are displayed for scenes consisting of 2, 5, 10, 50, and 100 line segments.

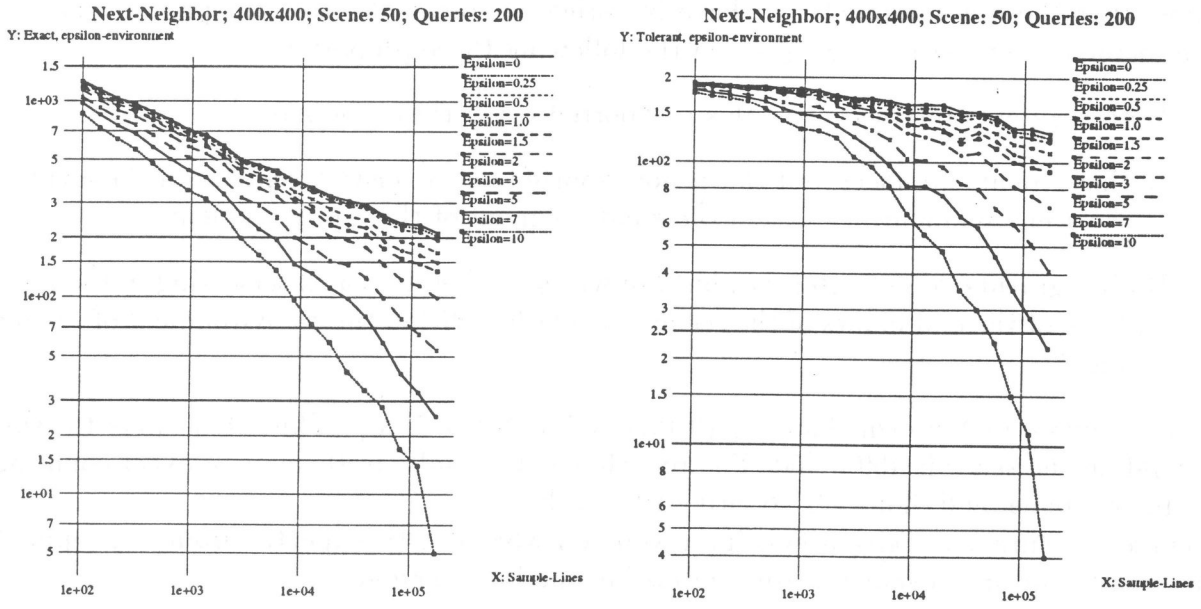


Figure 2: Error dependent on ε-tolerant counting. The left side shows the number of wrong intersections, the right side the number of wrong answers.

For low numbers of samples, the curves grow linearly. This means that for approximately each sample query, a new region is found. Later on, it becomes more and more hard to find new regions, and finally the curves are expected to become constant with value equal to the maximum number of regions. Only the latter part is of interest, since only there it can be expected that answers to application queries match with answers stored at sample queries.

Figure 1, right, shows the results achieved for 200 application queries dependent on the number of preprocessed sample queries, for a scene of 50 line segments. The curves display numbers of errors counted according to the following different rules.

Exact error. The number of existing intersections with the query segment not reported, plus the number of scene segments wrongly reported as intersecting.

Exact error with checked answers. The segments reported as intersected are checked whether this indeed holds. If not, they are removed from the answer. Thus only the existing intersections not reported are counted as error. As the curves show, a significant improvement can be achieved with that.

Tolerant error count. A query is counted if its answer is wrong, that is the degree of correctness considered in the previous two alternatives is ignored.

Visibility error. Number of wrong answers to the question whether a scene segment is intersected at all by the query segment. It turns out that reasonable answers can be obtained for less samples than in the case of the intersection reporting problem.

Another question is how wrong the wrong answers are from the geometric point of view. The sampling approach can be interpreted as either queries with "fuzzy" geometric objects like line segments with a surrounding sensitive environment in an exact scene, or as "fuzzy"

scene objects which are treated with exact queries. More about that will follow in the long version. For the measurements, we have investigated so-called ε-tolerant pairs of query and scene segments defined by having one of the following three properties:

1. Both segments intersect, and this is reported correctly in the answer.
2. Both segments intersect, but this is not reported in the answer, and one of the end points of the scene segment lies within the ε-environment of the query segment.
3. Both segments do not intersect, but are wrongly identified as intersecting in the answer, and one of the end points of the scene segment lies within the ε-environment of the query segment.

Measurements were performed for $\varepsilon \in \{0, 0.25, 0.5, 1, 2, 3, 5, 7, 10\}$. The extension of the square surrounding the scene is 400×400. Figure 2 shows the results in the case of exact counting as introduced above, and figure for counting tolerantly.

These measurements give a first impression on what can be expected from that approach. More results and improvements are compiled in the long version.

References

Preparata, Shamos, Computational Geometry – An Introduction, 2nd ed., Springer-Verlag, New York, 1988

N. Sarnak, R.H. Tarjan, Planar Point Location Using Persistent Search Trees, Communications of the ACM 29 (1986) 669–678

Discrete Simplicial Complexes

Werner Hoelbling, Werner Kuhn, Andrew U. Frank

Dept. of Geoinformation
Technical University Vienna
Gusshausstrasse 27-29/127
A-1040 Vienna (Austria)
email: kuhn@geoinfo.tuwien.ac.at
fax: +43-1-504-3535

Abstract

Simplicial complexes are a standard mathematical device underlying many data structures in computational geometry (Paoluzzi and others 1993). They serve to represent geometric objects and configurations by collections of elementary building blocks for each dimension (nodes, edges, triangles, tetrahedrons etc.), observing special intersection constraints (Giblin 1977).

Application-specific data models based on simplicial complexes have been developed for various domains. Among them is the Simplicial Data Model (Egenhofer and others 1990; Frank and Kuhn 1986), designed primarily for the management of vector data in geographic information systems (GIS). This model allows for a consistent representation of topological relationships. Inconsistencies can, however, arise from repeated intersections of line segments, due to the finite resolution of computational number systems.

The present work extends the Simplicial Data Model to discrete coordinate spaces. The result is a robust specification and implementation of plane simplicial complexes for finite representations. We start from Greene and Yao's (Greene and Yao 1986) method for line segment intersections in the discrete domain. This method guarantees that intersection points remain contained within small envelopes around the original line segments. However, it cannot resolve certain ambiguities and the resulting representation of topological relationships depends on the sequence of object insertions.

Our approach retains the idea of envelopes, avoiding the artifact of wandering lines, but replaces Greene and Yao's redrawing of line segments by a hierarchical representation of line segments and their subdivisions. In addition to the stronger consistency requirements, this also satisfies the need for retracing the history of line segment subdivisions as well as for keeping track of the collinearity of line segments.

The presentation will highlight the key ideas of our method and demonstrate an algebraic specification and prototype implementation in the functional programming language GOFER.

References

Egenhofer, Max J., Andrew U. Frank, and Jeffrey P. Jackson. "A Topological Data Model for Spatial Databases." In Symposium on the Design and Implementation of Large Spatial Databases, ed. A. Buchmann, O. Günther, T.R. Smith, and Y.-F. Wang. 271-286. 409. New York, NY: Springer-Verlag, 1990.

Frank, Andrew U. and Werner Kuhn. "Cell Graphs: A Provable Correct Method for the Storage of Geometry." In Second International Symposium on Spatial Data Handling in Seattle, WA, edited by D. Marble, IGU, 411-436, 1986.

Giblin, R.J. Graphs, Surfaces and Homology. Mathematics Series, London: Chapman and Hall, 1977.

Greene, D. and F. Yao. "Finite-Resolution Computational Geometry." In 27th IEEE Symp. on Foundations of Computer Science, 143-152, 1986.

Paoluzzi, A., F. Bernardini, C. Cattani, and V. Ferrucci. "Dimension-Independent Modeling with Simplicial Complexes." ACM Transactions on Graphics 12 (1 1993): 56-102.

DETERMINATION OF FINITE SETS BY X-RAYS

R. J. GARDNER and PETER GRITZMANN*

A new technique, based on high resolution transmission electron microscopy (HRTEM), can effectively measure the number of atoms of a crystal lying on each line in certain directions (see Schwander et al. [4]). At present, this can only be achieved for some crystals and in a constrained set of lattice directions, that is, directions parallel to a line through two points of the crystal lattice. The aim is to determine the three-dimensional crystal from information of this sort obtained from a number of different directions.

For this purpose, an *X-ray* of a finite set F in a direction u is a function giving the number of its points on each line parallel to u, essentially the projection, counted with multiplicity, of F on the subspace orthogonal to u. Motivated by Schwander's crystallographic work, we investigate the determination of finite subsets of a lattice by their X-rays in finite sets of lattice directions. The affine nature of this problem allows us to consider only the integer lattice $\mathbb{Z}^n$.

It is not difficult to see that given any prescribed finite set of m lattice directions in $\mathbb{E}^n$, there are two different finite subsets of $\mathbb{Z}^n$ with the same X-rays in these directions. Therefore, earlier results either place an a priori upper bound on the number of points in the set or focus on X-rays in coordinate directions. In this talk, however, the cardinality of the sets is completely unrestricted, and we allow arbitrary lattice directions. Instead, we work with the natural class of convex lattice sets, that is, finite subsets of $\mathbb{Z}^n$ whose convex hulls contain no new lattice points.

Our main result completely answers a question posed by Larry Shepp. We prove that there are certain prescribed sets of four lattice directions – for example, those parallel to the vectors $(1,0)$, $(1,1)$, $(1,2)$ and $(1,5)$ – such that any convex subset of $\mathbb{Z}^2$ may be distinguished from any other such set by its X-rays in these directions. This extends readily to $\mathbb{Z}^n$, $n \geq 2$. Four is the best number possible, since no prescribed set of three lattice directions has this property.

This result is a discrete analogue of the result in [3] which shows that there are prescribed sets of four directions – for example, those whose slopes yield a transcendental cross ratio – such that any convex body in $\mathbb{E}^2$ may be distinguished from any other by its continuous X-rays in these directions. Here, a continuous X-ray is a function which returns the linear measures of parallel 1-dimensional sections. Part of our technique derives from that of [3], but the discrete case is much more complicated and we find it necessary to employ methods from the theory of cyclotomic fields, in particular p-adic valuations. This allows a fine analysis which shows that uniqueness will be provided by any set of four lattice directions whose slopes (suitably ordered) yield a cross ratio not equal to $4/3$, $3/2$, 2, 3 or 4.

The theorem in [3] is, unfortunately, unstable in the sense that an arbitrarily small perturbation of a suitable set of four directions may cause the uniqueness property to be lost. The natural question arises of whether finite precision suffices to guarantee determination, that is, are there four directions that can be specified by a finite set of integers such that convex bodies are determined by continuous X-rays taken in these directions? We provide an affirmative answer.

Another result shows that *any* prescribed set of seven mutually nonparallel lattice directions has the property that any convex subset of $\mathbb{Z}^2$ may be distinguished from any other such set by its X-rays in these directions. The number seven is best possible.

In [1], Edelsbrunner and Skiena introduced an interactive technique, which we call successive determination, in which the previous X-rays may be examined at each stage in deciding the best direction for the next X-ray. It was shown in [1] that convex polygons can be successively determined by three X-rays and in [2] we proved that convex polytopes in $\mathbb{E}^3$ can be successively determined by only two X-rays. We apply this technique to finite sets of points, and find that it suffices to use orthogonal projections; the extra information granted by X-rays is superfluous. We prove that

---

* Speaker.

First author supported in part by the Alexander von Humboldt Foundation and by National Science Foundation Grant DMS-9501289; second author supported in part by the Deutsche Forschungsgemeinschaft and by a Max Planck Research Award.

Address of R.J. Gardner: Department of Mathematics, Western Washington University, Bellingham, WA 98225-9063, *gardner@baker.math.wwu.edu*

Address of P. Gritzmann: Fb IV, Mathematik, Universität Trier, D-54286 Trier, Germany *gritzman@dm1.uni-trier.de*

finite subsets of $\mathbb{Z}^n$ can be successively determined by $\lceil n/(n-k) \rceil$ projections on $(n-k)$-dimensional lattice subspaces. When $k = 1$, this means that only two projections are required. Convexity is not needed for this result, but the underlying lattice structure plays an essential role; we find that arbitrary finite subsets of $\mathbb{E}^n$ require $(\lfloor n/(n-k) \rfloor + 1)$ projections on $(n-k)$-dimensional subspaces for their successive determination. In both results, the numbers cannot be reduced, even if projections on $(n-k)$-dimensional subspaces are replaced by k-dimensional X-rays, functions which give the number of points on each translate of a given k-dimensional subspace.

In discussing inverse problems, it is important to distinguish between determination and reconstruction. The final part of the talk will briefly discuss algorithmic aspects of the problem. We give efficient algorithms for some cases, but new strong $\mathbb{NP}$-hardness results for many practically relevant variants of the problem. These hardness results motivate the use of approximation methods that are currently studied in detail jointly with D. Prangenberg.

References

1. H. Edelsbrunner and S. S. Skiena, *Probing convex polygons with X-rays*, SIAM. J. Comp. **17** (1988), 870–882.
2. R. J. Gardner and P. Gritzmann, *Successive determination and verification of polytopes by their X-rays*, J. London Math. Soc. (2) **50** (1994), 375–391.
3. R. J. Gardner and P. McMullen, *On Hammer's X-ray problem*, J. London Math. Soc. (2) **21** (1980), 171–175.
4. P. Schwander, C. Kisielowski, M. Seibt, F. H. Baumann, Y. Kim, and A. Ourmazd, *Mapping projected potential, interfacial roughness, and composition in general crystalline solids by quantitative transmission electron microscopy*, Physical Review Letters **71** (1993), 4150–4153.

Reference Points for Shape Matching
(Abstract)

Helmut Alt * Ulrich Fuchs* Günter Rote †
Gerald Weber*

This work is concerned with the problem of shape matching, i.e. given two planar shapes A, B, apply a transformation (e.g translation, rigid motion, similarity) to B so that it matches A as good as possible. We assume here that shapes are represented by simple polygons. Different distance functions have been considered to measure the quality of the match: for example the Hausdorff distance, the area of the symmetric difference, and the Fréchet distance.

Rather than finding the optimal solution, it is often much more efficient to look for solutions that are within a constant factor of the optimum by matching so called *reference points*. These are points that are not too far away from each other when the optimal matching transformation is applied. Conversely, if first the two reference points are matched and then the optimum is determined under the condition that the reference points are kept invariant, this match is "not too bad" i.e. within a constant factor of the optimal one. Usually, since matching the two reference points decreases the degrees of freedom (in the plane) by two the approximate matching is much easier to find.

It is known, for example, that the center of gravity of the boundary of the convex hull and the Steiner point are feasible reference points for the Hausdorff distance [ABB95, AAR94].

Here, for the first time we will consider reference points for the area of the symmetric difference as distance measure for shapes. We will show that

*{alt,fux,weber}@inf.fu-berlin.de, Institut für Informatik, FU Berlin, Takustr. 9 , 14195 Berlin, Germany. This research was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Association) Project No. Al253/4-1

†rote@ftug.dnet.tu-graz.ac.at, Institut für Mathematik, TU Graz, Austria

for *convex* shapes the center of gravity is a reference point, more precisely we have:

Theorem 1 *Let F_1, F_2 be convex bodies $\delta_{opt}(F_1, F_2)$ the minimal area of the symmetric difference that can be achieved when F_2 is matched to F_1 by any transformation, $\delta_h(F_1, F_2)$ the minimal area of symmetric difference when F_2 is matched to F_1 by a transformation which maps the center of gravity of F_2 to the one of F_1.*

Then

$$\delta_h(F_1, F_2) \leq \frac{11}{3}\delta_{opt}(F_1, F_2)$$

This inequality holds for translations, rigid motions, and similarities as sets of allowable transformations. There are figures F_1, F_2 where the inequality above is tight, i.e. the constant $11/3$ cannot be improved.

As was mentioned before this property of the center of gravity leads to efficient algorithms for matching convex figures with respect to the area of the symmetric difference.

References

[AAR94] H. Alt, O. Aichholzer, and G. Rote. Matching shapes with a reference point. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 85–92, 1994.

[ABB95] H. Alt, B. Behrends, and J. Blömer. Approximate matching of polygonal shapes. *Annals of Mathematics and Artificial Intelligence*, 13:251–265, 1995.

Measuring circularity of a set of points
Abstract

Jesus Garcia * Pedro A. Ramos †

December 15, 1995

1 Introduction

The problem of measuring the circularity of a shape has received a lot of attention in the past. The main motivation for this problem comes from metrology: given an object that has to be tested for circularity, take a sample of points from the object and measure the circularity of this sample set; then accept the object if the circularity is good enough and reject otherwise. Strategies for choosing such a sample and what can be deduced from the sample of points to the actual shape fall into the field of Metrology, and we are going to concentrate in the geometric problem: given a set of points in the plane S, our aim is to measure the circularity of S.

There are several possibilities in order to measure the circularity of a set $S = \{p_1, \ldots, p_n\}$. We define the *roundness* of S as the width of the annulus of minimum width containing S (both the American National Standards Institute and the International Standards Office recommends this measure to be used for testing circularity, see [3, pp. 40–42]).

If we define the function

$$\mathcal{R}_S(x) = \max_{i=i,\ldots,n} d(x, p_i) - \min_{j=1,\ldots,n} d(x, p_j)$$

the problem is to find the global minimum of $\mathcal{R}_S(x)$.

Although this problem has been studied in several works [2, 1, 6], no exact characterization of local minima of $\mathcal{R}_S(x)$ has been given. The equivalent problem for simple polygons was first treated in [4] and solved for convex polygons in optimal linear time in [7].

In this talk we shall characterize local minima of $\mathcal{R}_S(x)$, leading to two different results:

- We give an $O(n \log n)$ algorithm for solving the problem for convex sets of points, largely improving previous results.
- We rise the open problem of determine the number of local minima of the function, because the unique bounds are $\Omega(n)$ and $O(n^2)$.

*Dpto. Matematica Aplicada, E.U. Informatica, Universidad Politecnica de Madrid. e-mail:jglopez@eui.upm.es

†Dpto. Matematica Aplicada, Universidad Politecnica de Madrid. e-mail:pramos@fi.upm.es

References

[1] P. K. Agarwal, M. Sharir, S. Toledo: Applications of parametric searching in geometric optimization, *Journal of Algorithms*, Vol. 17, pp. 292-318, 1994.

[2] H. Ebara, N. Fukuyama, H. Nakano, Y. Nakanishi: Roundness algorithms using Voronoi diagrams, *Abstracts 1st Canadian Conference on Computational Geometry*, 1989.

[3] L. W. Foster: *GEO-METRICS II: The application of geometric tolerancing techniques.* Addison-Wesley Publishing Co., 1982.

[4] V. B. Lee, D. T. Lee: Out-of-roundness problem revisited. *IEEE Transactions on Pattern Analysis and Machinery Intelligence*, Vol. 13, pp. 217-223, 1991.

[5] T. J. Rivlin: Approximation by circles, *Computing* Vol. 21, pp. 93-104, 1979.

[6] M. Smid, R. Janardan: On the width and roundness of a set of points in the plane, *Proc. 7th. Canadian Conference on Computational Geometry*, pp. 193-198, 1995.

[7] K. Swanson, D. T. Lee, V. L. Wu: An optimal algorithm for roundness determination of convex polygons. *Computational Geometry: Theory and Applications* Vol. 5, pp. 225-235, 1995.

Orientation independent covering of point sets in R^2 with pairs of rectangles or optimal squares (Extended abstract)

Jerzy W. Jaromczyk* Miroslaw Kowaluk†

Abstract

We study the problem of optimal covering a finite set S of points in R^2 with squares or rectangles. Two algorithms are presented:

An $O(n^2)$ algorithm that finds optimal covering, with respect to side length, of S with two squares with mutually parallel sides.

An $O(n^2)$ algorithm that decides if there are two rectangles with sides a, b, and c, d, respectively and mutually parallel sides that cover S.

The first problem can be viewed as minimal covering with two spheres in L_1 and L_∞ with optimization both on the choice of the coordinate system and sphere radius.

The solution is based on a structure, called *silhouette*, that maintains information about the dominating points of S for each direction of the axes of coordinate systems. We show how this structure can be built in $O(n \log n)$ time and $O(n)$ space.

1 Introduction

Covering a set of points with a set of figures, such as circles or squares, is a relatively well studied topic in computational geometry. The problem is interesting in the context of set approximation for various metrics [KM93], and in the context of best matching two sets of points by means of geometric transformations [AMWW88, ISI89]. This paper considers the following problems:

Given a set S of n points in a real plane, find a pair of side-parallel squares with minimal sides (or equivalently, areas) whose union contains S.

Using L_1, or L_∞ metrics associated with a coordinate system XOY, where $d(p, q)$ denotes the distance between p and q in this metric, we can state the above problem as an optimization problem:

$$\min_{XOY} \min_{p,q \in R^2} \max_{x \in S} \min\{d(x, p), d(x, q)\}.$$

Given a set S of n points in a real plane and two pairs of positive numbers a, b, and c, d decide whether there exist two side-parallel rectangles with sides a, b and c, d whose union contains S.

Similar problems have been considered in computational geometry literature [BFG+92, HS91, MT83, Dre81, MS81, OW88]. They, however, deal mainly with the situation when the direction for the sides of squares or rectangles, i.e., the coordinate system, is fixed. Adding an additional degree of freedom for the coordinate system makes the problem even more challenging. In particular, in the context of the geometric fitting [ISI89] of matching two sets of points with geometric transformations, this leads to the dynamic computational geometry as introduced in [Ata85]. The fitting problems have strong connections with the complexity of the upper envelopes on multivariate functions and apply Davenport-Schinzel sequences or linearization techniques.

*Department of Computer Science, University of Kentucky, Lexington, KY 40506
†Institute of Informatics, Warsaw University, Warsaw, Poland

The problem of covering with squares or circles is known in literature as the p-center problem [MT83]. More specifically, the p-center problem asks whether a given set of points can be partitioned into p subsets, each of which can be enclosed in a disc of the given radius. As shown in [MS84], this problem is NP-complete if p is a part of the input. For $p = 1$ and $p = 2$ efficient algorithms exist; see for example [AS91, JK94, KM93, Dre81].

Our paper can be viewed as a generalization of the results [HS91] where partitioning with respect to a variety of measures, assuming a fixed coordinate system, is studied. For example, [HS91] solves a decision problem for a pair of squares in $O(n \log n)$ time, again, assuming a fixed coordinate system. We consider this problem independently of the coordinate system. As mentioned above, this addition of an extra degree of freedom substantially complicates this problem and calls for new techniques and data structures; we will develop them in our paper. Together with these new methods our algorithms use also classical techniques (such as Graham's scan and the depth-first-search) in a new original context.

2 Silhouette

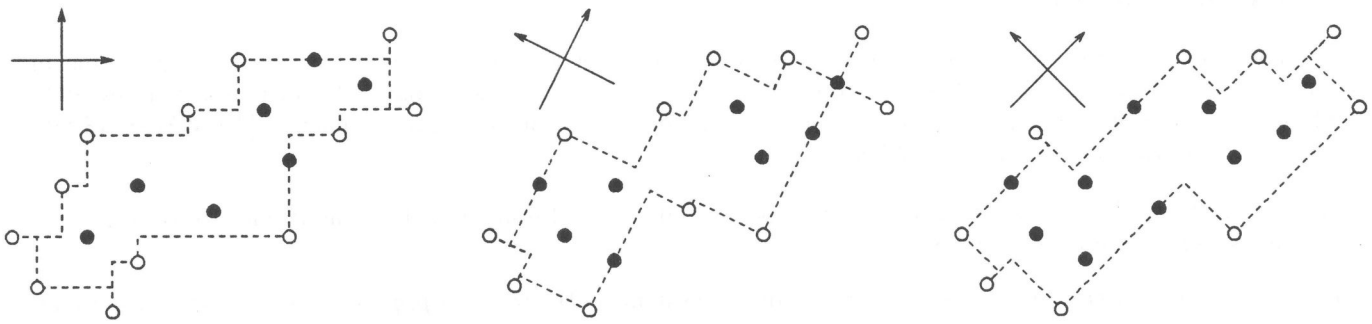
We will describe a data structure that provides fast access to a point in S that may determine optimal squares or rectangles.

Let us assume that S is endowed with an orthogonal coordinate system; we will call this system *canonical* and will denote it by XOY. A system obtained from XOY with rotation by α about O will be denoted by $XOY(\alpha)$. The angles are measured counterclockwise and $0 \leq \alpha < \pi/2$.

We say that a point $p \in S$ is *α-exposed* if at least one closed quadrant of the coordinate system obtained by centering $XOY(\alpha)$ at p does not contain any point in $S \setminus p$.

DEFINITION **2.1** *The $\alpha - silhouette(S)$ is the set of all the α-exposed points of S. Additionally, $silhouette(S) = \bigcup_\alpha \alpha - silhouette(S)$.*

The α-silhouette(S) can be visualized as a set of those points of S that block a large rectangle with sides parallel to the axes of $XOY(\alpha)$ when we move it from outside toward S.



Let $v_1, \ldots, v_k$ be vertices of the convex hull $CH(S)$ of S.

LEMMA **2.1** *Each vertex of the convex hull of S belong to $\alpha - silhouette(S)$ for each $\alpha \in [0, \pi/2)$*

PROOF:

Let v be a vertex of $CH(S)$. Then there exists a halfplane whose intersection with $CH(S)$ is v. For any orthogonal system centered at v, at least one of its closed quadrants is completely contained in the interior of this halfplane; that means that v is exposed.

□

2.1 Building α-silhouettes

As we have seen above the vertices of $CH(S)$ play a special role which will be explored in the construction of $silhouette(S)$. Our strategy is to find for each point p in S the range of α for which p is α-exposed. The process will be organized with respect to the edges of $CH(S)$. Let $v_1, \ldots, v_k$ be vertices of the convex hull $CH(S)$ of

S listed counterclockwise. We say that edge v_iv_{i+1} *does not obstruct* p for α if v_i, v_{i+1} belong to the opposite quadrants of $XOY(\alpha)$ centered at p.

LEMMA **2.2** *There exists α such that v_iv_{i+1} does not obstruct p for α if and only if the angle v_ipv_{i+1} is larger than $\pi/2$.*

PROOF: Follows directly from the definition. □

As immediate corollaries we have

COROLLARY **2.1** *A point p can belong to α-silhouette only if there exists an edge that does not obstruct p for α.*

Observe that an opposite implication is false.

COROLLARY **2.2** *For each $p \in S$ there are at most three edges of $CH(S)$ that do not obstruct p.*

PROOF: Consider the angles $v_ipv_{(i \bmod k)+1}$, $i = 1, \ldots, k$. The sum of their measures is 2π so at most three of them can be larger than $\pi/2$. □

Based on this corollary and Lemma 2.2 it is convenient to associate points in α-silhouettes with edges of $CH(S)$. That is, for each edge v_iv_{i+1} in $CH(S)$ we will find those points in S together with their ranges that are not obstructed by v_iv_{i+1} and that are in α-silhouette for some α. The first step is to associate points of S with their non-obstructing edges. This can be accomplished as follows. Based on the definition of obstruction we know that an edge does not obstruct point p for some angle if the semicircle with this edge being its diameter contains p in its interior. Hence, we can build a data structure for the arrangement of the discs associated with the edges of $CH(S)$ and then perform a point location. Based on Corollary 2.2, each point can belong to at most three circles. Using data structure from [Aur88], the points-circles location can be performed in the total $O(n \log n)$ time. This can be done easier, however. In view of Lemma 2.2, edge $e = v_1v_2$ does not obstruct p if the angle $v_1pv_2 > \pi/2$. Therefore, in order to find all the edges (at most three) that do not obstruct p, we center XOY at p and consider only those edges of $CH(S)$ that are intersected by either axis of this canonical coordinate system. Then, we check if the angle formed by the endpoints of these edges with p is larger than $\pi/2$. Clearly, the other angles must be smaller than $\pi/2$. Having $CH(S)$ the edges intersected by the axes of the coordinate system centered at p can be found in logarithmic time per point. Altogether, the entire process of finding non-obstructing edges takes $O(n \log n)$ time.

Now, for each edge e and each point p not obstructed by this edge we will find the range $(l(p), r(p))$ such that p is in α-silhouette for $\alpha \in (l(p), r(p))$. This range is taken "modulo" $\pi/2$. This angular segment will be called *range of exposure* of p for e. Intuitively, we can think about identifying those points not obstructed by e that are reachable by a large rectangle (with sides parallel to axes of $XOY(\alpha)$) that penetrates $CH(S)$ through e without touching the ends of e and without enclosing any point of S.

Let $p_1, \ldots, p_m$ be a list of all the points not obstructed by e and sorted with respect to their orthogonal projections on e (only one of many points with the same orthogonal projection can be in the silhouette). We will need the following convex hulls to utilize our process: $CH(1,j)$ - the convex hull of $v_i, p_1, \ldots, p_{j-1}$, and $CH(m,j)$ - the convex hull of $v_{i+1}, p_m, \ldots, p_{j+1}$, where $j = 1..m$. In this notation we assume that $CH(1,1) = \{v_i\}$ and $CH(m,m) = \{v_{i+1}\}$. We can find all these convex hulls in the total $O(n)$ time (when the order of vertices is known) using the Graham's scan twice, see [Gra72]. The first time the scan is run for the increasing values of j to compute $CH(1,j)$, then for the decreasing values to compute $CH(m,j)$. As a side effect of running the Graham's scan we obtain the tangent lines to $CH(1,j)$ and $CH(m,j)$ and passing through p_j, for $j = 1, \ldots, m$. If an angle between tangents is greater than $\pi/2$, these lines determine positions of the coordinate systems for which p_j belongs to the silhouette.

The pseudocode for the algorithm finding the ranges of exposure for $e \in CH(S)$ is given below (at the beginning the ranges are undefined):

---

ranges(S, e)

/* let $p_1, \ldots, p_m$ be a sorted set of points not obstructed by e sorted with respect to their projections on e */

1. Find, using the Graham's scan, $CH(1,j)$ and $CH(m,j)$, $j = 1,\ldots,m$ and lines t_j^l and t_j^r passing through p_j and tangent to $CH(1,j)$ and $CH(m,j)$, respectively. Let p_j^l and p_j^r be the corresponding points of tangency.

2.1. $\alpha(t_j^l)$:= the angle between t_j^l and a nearliest (clockwise) axis of the coordinate system obtained by centring XOY at p_j

2.2. $\beta(t_j^l, t_j^r)$:= the angle between t_j^l and t_j^r

/* range for p_j */

3. **if** $\beta(t_j^l, t_j^r) > \pi/2$ **then**

3.1. $l(p_j) := \alpha(t_j^l)$

3.2. $r(p_j) := \alpha(t_j^l) + \beta(t_j^l, t_j^r) - \pi/2$

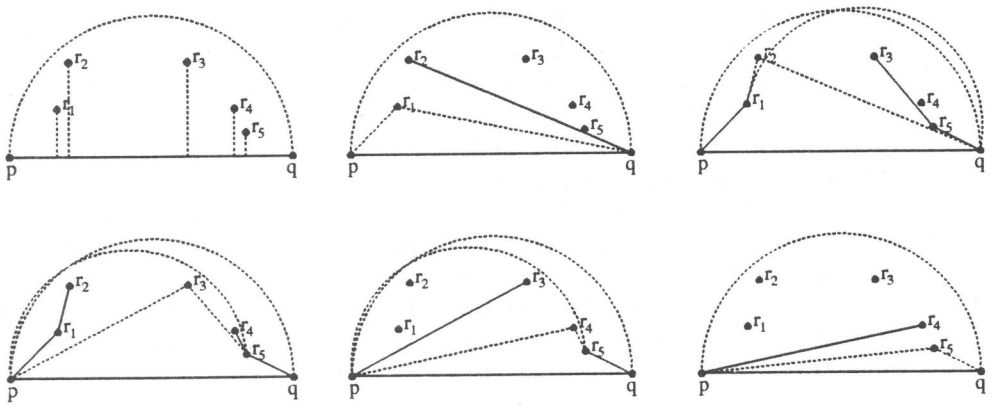
endif

Statement 3 of the above algorithm decides whether there is a coordinate system centered at p whose quadrant which intersects e does not contain any points of S. Therefore the algorithm correctly identifies ranges of exposure for points not obscured by e. Note the following:

LEMMA **2.3** *The range of exposure of p_j for e is contained in the ranges of exposure for p_j^l and p_j^r.*

PROOF: Follows from the definitions of the convex chains $CH(1,j), CH(m,j)$ and the definition of exposure. □

The algorithm is illustrated in the figure below; semicircles allow to decide if the corresponding angles are larger than $\pi/2$:



Solid lines denote portions of $CH(1,j)$ and $CH(m,j)$

For each edge and for all the points not obstructed by these edges, we can find their ranges of exposure in time $O(m \log m)$, where m is the number of points not obstructed by e. Therefore, in view of Corollary 2.2 we have:

LEMMA **2.4** *The total cost of finding ranges of exposure for all the points of S with respect to all their non-obstructing edges is $O(n \log n)$.*

2.2 Representation of the silhouette

To take the full advantage of the silhouette we need to organize points in a fashion that will support efficient traversal of the silhouette during the rotation of the coordinate system. This traversal will resemble a sweeping of the silhouette with a rotating line. The silhouette can be visualized as a directed graph. To define this graph, denoted by $\mathcal{G}(\mathcal{S})$, we need an additional notation. Consider a set of points that are not obscured by an edge e. We define $between(v,w)$ the set of these points not obscured by e that are in the silhouette for some α and whose projection on e is between the projections of v and w. Now, the graph $\mathcal{G}(\mathcal{S})$ is defined as follows: The points of S are its vertices. There is a directed edge from v to w if $((l(v), r(v)) \cap (l(w), r(w))) \setminus \bigcup_{x \in between(v,w)} (l(x), r(x))) \neq \emptyset$.

LEMMA **2.5** *The total number of the edges in $\mathcal{G}(\mathcal{S})$ is $O(n)$*

PROOF: For each obscuring edge e, the range of exposure for p is an angular segment. For each α-silhouette, p is connected in $\mathcal{G}(\mathcal{S})$ with two other points. We can count the number of edges as follows: Consider the growing

value of α. At most two edges are added when α passes through the beginning of some range. When α passes through the end of this range, at most one new edge is added (to connect the gap created by the removal of p). For each obscuring edge e we start this counting with one edge and the whole process adds and then removes each point not obscured by e at most once. The total number of points that are added or removed for all the edges of $CH(S)$ is $O(n)$. Hence the total number of edges in $\mathcal{G}(\mathcal{S})$ is also $O(n)$. □

$\mathcal{G}(\mathcal{S})$ can be represented as an array of the points of S that point to cross-indexed lists. Each entry v in the array points to a list of vertices that for some α are neighbors of v in α-silhouette. The size of this representation is linear for all the non-obstructing edges. Representation for $\mathcal{G}(\mathcal{S})$ can be built in $O(n \log n)$ time using a depth-first-search approach. More specifically, this depth-first-search is in the graph of points not obscured by e with edges that connect tangency points p_j^l with p_j and p_j with p_j^r. By Lemma 2.3 the corresponding ranges are nested and therefore the search will correctly identify neighbors in $\mathcal{G}(\mathcal{S})$. If the ranges are nested then the point with the enclosing range will appear in this representation more than once. The cost follows from the linear size of $\mathcal{G}(\mathcal{S})$. The logarithmic factor results from geometry that requires sorting: the angular order of neighbors of p in $\mathcal{G}(\mathcal{S})$ is needed.

THEOREM **2.1** *The cost of building a representation for $\mathcal{G}(\mathcal{S})$ is $O(n \log n)$.*

3 Algorithm

In this extended abstract we will focus on finding the optimal squares, and will comment only on the decision problem of the rectangles. Consider S and a coordinate system $XOY(\alpha)$ for a fixed value of α. Let $\square(\alpha, S)$ be the smallest rectangle with axis-parallel sides which encloses S. The points of S on the boundary of $\square(\alpha, S)$ will play a special role; we will denote their set by $T(\alpha)$. Now, we have an obvious lemma:

LEMMA **3.1** *For each coordinate system $XOY(\alpha)$ there is a pair of optimal squares enclosing S, with the sides axes-parallel, such that their sides contain the points of $T(\alpha)$.*

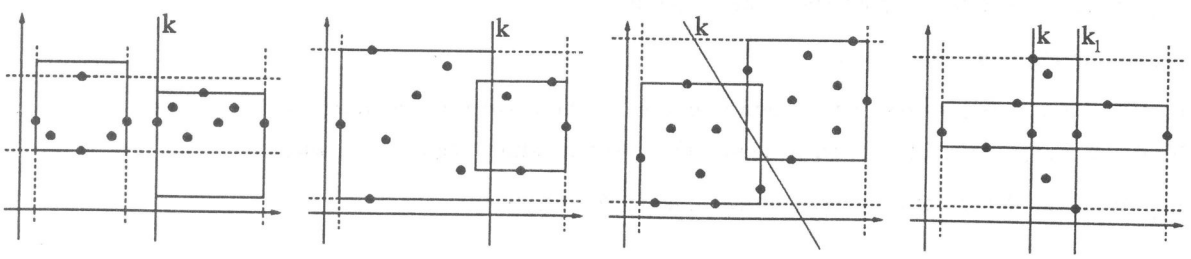
PROOF: If the points of $T(\alpha)$ are not on the sides of the optimal pair of squares, then we can slide the squares to include these points without leaving any point of S outside of the squares. □

LEMMA **3.2** *The number of different sets $T(\alpha)$ is $O(n)$.*

PROOF: The tangent points for bounding boxes correspond to the antipodal points of $CH(S)$. It is well known that the number of such points is $O(n)$; see [PS85]. □

For clear reasons the points in $T(\alpha)$ will be called *tangent determiners*. Note that without loss of generality we may assume that there are at most eight points in $T(\alpha)$ since only the outermost points of the collinear points in $T(\alpha)$ affect the size of the squares. In addition to tangent determiners there are other point of S that determine other sides of the optimal squares. They belong to α-silhouette, and we will denote them by $C(\alpha)$, or they belong to the closer boundaries of the two width-optimal stripes enclosing S, and we will denote them by $B(\alpha)$.

Distribution of $T(\alpha) \cup C(\alpha) \cup B(\alpha)$ between the squares leads to three different configurations. For rectangles we have an additional one. The configurations are illustrated below:



There are four cases:

case 1: The squares (rectangles) are disjointed and some of the determing points are not in the silhouette,

case 2: One of the squares (rectangles) is determined by three points in $T(\alpha)$,

case 3: Each of the squares (rectangles) contains two adjacent points in $T(\alpha)$,

case 4: Each of the rectangles contains two opposite points in $T(\alpha)$.

As mentioned above, in addition to $T(\alpha)$, the squares are determined by points $C(\alpha)$ on the silhouette (specifically, by points on α-silhouette), or by points in $B(\alpha)$ on the boundaries of the optimal stripes containing S with direction α. It can be easily shown that the points that can determine one or the other square (rectangle) are divided into two groups. Geometrically this division is determined by a line k; see the figure above. We can think about two pointers pl, and pr on each side of the silhouette, moving along the edges of α-silhouette in $\mathcal{G}(\mathcal{S})$ that determine this partition. For the fourth case (pertaining to rectangles) we consider two pairs of pointers that correspond to two separating lines.

Let $E(pl, pr)$ be the edges of $\mathcal{G}(\mathcal{S})$ pointed to by pl, pr, $V(pl, pr)$ be vertices of these edges. (In case when the pointers during their move point to a vertex, all three vertices will be included in $V(pl, pr)$.) The pointers move as a function α in effect of the rotating system.

Our general strategy is to determine the optimal squares based on the set of tangent determiners and points in $C(\alpha) \cup B(\alpha)$; the number of points in $C(\alpha) \cup B(\alpha)$ never exceeds twelve. The points in $T(\alpha) \cup C(\alpha) \cup B(\alpha)$ can be distributed between two squares on several ways. However, once this distribution is decided the optimal squares can be found easily. Critical for this process is set $T(\alpha) \cup C(\alpha) \cup B(\alpha)$ which is a function of α. Due to the discrete nature of the problem, the changes in $T(\alpha) \cup C(\alpha) \cup B(\alpha)$ occur for discrete values of the angles in $[0, \pi/2)$. They will be computed from $CH(S)$, the silhouette and the anchored lower chains structure introduced in [JK95]. Anchored lower chains allow us to find efficiently tangent lines to the convex hulls of points to the left (right) of a given line passing through some point of S. These tangents allows us to find the width of these sets. Typically, lines will be boundaries of a stripe.

The anchored lower chain for S can be constructed in $O(n^2)$.

In our algorithm we will traverse $\mathcal{G}(\mathcal{S})$ to simulate rotation of the coordinate system and to compute changes in $T(\alpha) \cup C(\alpha)$ as well as to find those points on the silhouette that determine the remaining sides of the squares. Changes in $B(\alpha)$ are computed based on the anchored lower chains. For each angular range in which $T(\alpha) \cup C(\alpha) \cup B(\alpha)$ is fixed, we will find in constant time the optimal squares by optimizing simple trigonometric functions of α.

LEMMA **3.3** *For each range of α where $T(\alpha)$ does not change, the set $C(\alpha)$ can change at most $O(n)$ times.*

PROOF: Each point of S belongs to the silhouette for at most three connected range of angles. Therefore there are at most $O(n)$ ranges that cover the range where $T(\alpha)$ is invariant. A change in $C(\alpha)$ occurs when some subrange is entered or left. It can happen at most $O(n)$ times. □

COROLLARY **3.1** *The number of different sets $T(\alpha) \cup C(\alpha) \cup B(\alpha)$ is $O(n^2)$.*

PROOF: Follows from Lemma 3.2, 3.3 and [JK95]. □

Now, four kinds of changes may occur as the result of rotating the system:

1. $T(\alpha)$ changes - denote the corresponding angle by $t - angle$,
2. $E(pl, pr)$ or $V(pl, pr)$ may change
   (a) if the α-silhouette does not change then denote the corresponding angle by $p - angle$,
   (b) if the α-silhouette change then denote the corresponding angle by $e - angle$,
3. $B(\alpha)$ changes - denote the corresponding angle by $b - angle$.

The algorithm can be presented as follows:

---

optimal_squares(S)
1. $\alpha := 0$
2. find $T(\alpha)$ and $B(\alpha)$
3. find pl, pr
4. find $E(pl, pr)$ and $V(pl, pr)$
5. **while** $(\alpha < \pi/2)$ **do**
5.1 find $t - angle$, $p - angle$, $e - angle$, and $b - angle$.
5.2 $nextangle := \min\{t - angle, p - angle, e - angle, b - angle\}$
5.3 call optimal_solution$(\alpha, nextangle)$ and update $best - solution$
5.4 $\alpha := nextangle$
5.5 **if** $nextangle = t - angle$ **then** find new $T(\alpha)$
5.6 **if** $nextangle = b - angle$ **then** find new $B(\alpha)$
5.7 find new pl, pr, $E(pl, pr)$, and $V(pl, pr)$
endwhile;
6. **return** $best - solution$

Remark: During the entire process the pointers move along paths in $\mathcal{G}(\mathcal{S})$, i.e., they never "jump" from one edge to another that is not adjacent. The same remark pertains to the anchored lower chains.

Remark: Procedure optimal_solution analyzes each of the four cases separately. After finding a solution we need to verify its consistency with the case being analyzed.

Steps 1-4 initiate the traversal in the silhouette and their total cost is $O(n)$. Procedure optimal_solution depends on the current case and its cost per call is $O(1)$; while the determining point are known in the given range $[\alpha, nextangle)$ the optimal squares can be found as a minimization of two trigonometric functions. Inside the while-loop the global cost of all the operations, except for statement 5.7, is constant. In particular, statement 5.5 corresponds to moving calipers, as in the width algorithm [Tou83]. Statement 5.7 involves changes in pl and pr. The total cost of statement 5.7 is proportional to the size of $\mathcal{G}(\mathcal{S})$ which is $O(n)$. The number of iterations is equal to the number of possible changes of determining points while rotating the coordinate system. This number is not larger than the total number of lines determined by n points and is, therefore, of an order $O(n^2)$.

Concluding, we have the following theorem:

THEOREM **3.1** *For a given set S a pair of the optimal squares with mutually parallel sides which include S can be found in $O(n^2)$.*

PROOF: Based on Theorem 2.1 a representation for the silhouette of S can be constructed in $O(n \log n)$ time. Building and access to the angular chains cost $O(n^2)$. The number of changes in $nextangle$, that never decrements, is $O(n^2)$. Hence, the while-loop is executed at most $O(n^2)$ times. The bound follows. □

Remark: The decision problem for rectangles can be found in the similar time bounds. The algorithm computes sets $E(pl, pr), V(pl, pr), B(\alpha)$ based on the sides of one of the rectangles. For every value of $[\alpha, nextangle)$ in constant time we can verify if the other rectangle covers the remaining points, using simple trigonometric functions and the silhouette.

4 Discussion

We have shown algorithms for covering a set of points S with a pair of rectangles or squares. The two squares problem is an instance of the p-center problem for L_1 or L_∞ metric studied in operation research. The direction for sides of the rectangles is one of unknowns in our problem. This is in contrast to many related papers that assume a fixed direction, or a fixed coordinates system. Our algorithm for the optimal pair of squares takes $O(n^2)$ time. This compares very favorably with the best known decision algorithm (see [HS91]) for a pair of squares in a fixed coordinate system, which has time complexity $O(n \log n)$. For any fixed direction our algorithm finds an optimal pair of squares in $O(n \log n)$ time.

The solution is based on a data structure, called the silhouette, that organizes the subsets of S which can determine the optimal squares. The silhouette enables us to access these determining points quickly. This data structure is interesting by itself both algorithmically and combinatorially, and we expect that it will have other applications related to rectangles and squares.

Similar methods can be used for non-orthogonal coordinate systems. Specifically, we can find two optimal rhombuses and decide if a pair of parallelograms can cover the given set.

References

[AMWW88] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. Congruence, similarity and symmetries of geometric objects. *Discrete Comput. Geom.*, 3:237–256, 1988.

[AS91] P. K. Agarwal and M. Sharir. Planar geometric location problems and maintaining the width of a planar set. In *Proc. 2nd ACM-SIAM Sympos. Discrete Algorithms*, pages 449–458, 1991.

[Ata85] M. J. Atallah. Some dynamic computational geometry problems. *Comput. Math. Appl.*, 11:1171–1181, 1985.

[Aur88] F. Aurenhammer. Improved algorithms for discs and balls using power diagrams. *J. Algorithms*, 9:151–161, 1988.

[BFG+92] B. Becker, P. G. Franciosa, S. Gschwind, T. Ohler, G. Thiemt, and P. Widmayer. Enclosing many boxes by an optimal pair of boxes. In *Proc. 9th Sympos. Theoret. Aspects Comput. Sci.*, volume 577 of *Lecture Notes in Computer Science*, pages 475–486. Springer-Verlag, 1992.

[Dre81] Z. Drezner. On a modified 1-center problem. *Manage. Sci.*, 27:838–851, 1981.

[Gra72] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Inform. Process. Lett.*, 1:132–133, 1972.

[HS91] J. Hershberger and S. Suri. Finding tailored partitions. *J. Algorithms*, 12:431–463, 1991.

[ISI89] K. Imai, S. Sumino, and H. Imai. Minimax geometric fitting of two corresponding sets of points. In *Proc. 5th Annu. ACM Sympos. Comput. Geom.*, pages 266–275, 1989.

[JK94] J. W. Jaromczyk and M. Kowaluk. An efficient algorithm for the Euclidean two-center problem. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 303–311, 1994.

[JK95] J. W. Jaromczyk and M. Kowaluk. The two-line center problem from a polar view: a new algorithm and data structure. Proceedings of the WADS'95 Conference. 1995.

[KM93] N. M. Korneenko and H. Martini. Hyperplane approximation and related topics. In J. Pach, editor, *New Trends in Discrete and Computational Geometry*, volume 10 of *Algorithms and Combinatorics*, pages 135–161. Springer-Verlag, 1993.

[MS81] A. Marchetti-Spaccamela. The p-center problem in the plane is NP-complete. In *Proc. 19th Allerton Conf. Commun. Control Comput.*, pages 31–40, 1981.

[MS84] N. Megiddo and K. J. Supowit. On the complexity of some common geometric location problems. *SIAM J. Comput.*, 13:182–196, 1984.

[MT83] N. Megiddo and A. Tamir. New results on the complexity of p-center problems. *SIAM J. Comput.*, 12:751–758, 1983.

[OW88] M. H. Overmars and D. Wood. On rectangular visibility. *J. Algorithms*, 9:372–390, 1988.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry: an Introduction.* Springer-Verlag, New York, NY, 1985.

[Tou83] G. T. Toussaint. Solving geometric problems with the rotating calipers. In *Proc. IEEE MELECON '83*, pages A10.02/1–4, Athens, Greece, 1983.

Improved p-Center Algorithms

Micha Sharir
School of Mathematical Sciences, Tel Aviv University
and
Courant Institute of Mathematical Sciences, New York University

Abstract

We present improved algorithms for several p-center problems:

(i) We give an $O(n \text{ polylog } (n))$ algorithm for the Euclidean planar 2-center problem, where we want to cover a set of n points in the plane by two congruent disks of smallest possible radius. (The best previous solution was super quadratic in n.)

(ii) We give linear and near-linear algorithms for the rectilinear planar p-center problems, for $p = 2, 3, 4, 5$ (here we want to cover a set of n points in the plane by p congruent axis-parallel squares of smallest possible size), and for several other related problems.

(The results in (ii) are joint with Emo Welzl.)

Hierarchical Motion Planning Using a Spatial Index

K. Verbarg* A. Hensel*

Abstract

We investigate the problem of constructing a *shortest path* of a point–like robot between two configurations in the euclidean plane cluttered with (intersecting) convex polygonal obstacles. One common approach is to construct the *visibility graph* and search within this graph in a total time of $\mathcal{O}(n^2)$. We show that in general it is not necessary to construct the entire visibility graph. In contrast, we develop two hierarchical motion–planning techniques based on the *monotonous bisector tree* and the visibility graph, which are shown to be more efficient in scenes of *low object density*. We show that in our setting the visibility graph can be *incrementally* constructed in time $\mathcal{O}(n^2 \log n)$, where n is the complexity of the scene. A shortest path can then be constructed in time $\mathcal{O}(l^4 \log l + n \log l)$, if a shortest path of length l exists. The dependency on n can be further reduced by the use of a spatial data structure complying a query assumption to $\mathcal{O}(l^4 \log l + l^2 \log n)$. If l is not small compared to the diameter D_S of the scene, the algorithm still respects an $\mathcal{O}(n^2(\log n + \log D_S))$ upper bound. If no path exists, then similar estimations for the diameter of a set of obstacles certifying the failure can be stated. We report on experimental results verifying the efficiency of the methods.

Keywords: Shortest path, visibility graph, spatial data structure, computational geometry.

1 Introduction

A fundamental problem in constructing autonomous robots is to compute a collision–free path through a scene of obstacles. Here, we treat the case of finding a *shortest* path between two arbitrary query points in the euclidean plane cluttered with *many* polygonal obstacles. Since we permit intersections of obstacles, we may assume without loss of generality that the obstacles are convex and of constant complexity. Many methods have been proposed to solve this problem — a survey can be found in the textbook of Latombe [1]. The basic concept to solve this special case is the *visibility graph* [2, 3, 4]. It consists of straight–line segments between polygon vertices, which do not cross the interior of any polygon.

The complexity of motion–planning algorithms is usually expressed in terms of the complexity n of the underlying scene. The complexity of a scene of polyhedrons is the number of faces of any dimension. By constructing the visibility graph and ap-

*Department of Computer Science, University of Würzburg, Am Hubland, 97074 Würzburg, Germany, Email: {verbarg,hensel}@informatik.uni-wuerzburg.de, Phone: ++49 – 931 888 5058, FAX: ++49 – 931 888 4600, WWW: http://www-info1.informatik.uni-wuerzburg.de/index_e.html.

plying Dijkstra's algorithm we obtain an $\mathcal{O}(n^2)$ shortest path algorithm for non–intersecting obstacles in the plane. There is a strong indication that there is no planar motion planning algorithm (without the demand to compute the *shortest* path) running in time $o(n^2)$ in the algebraic decision tree model, since this problem was shown to be 3Sum–hard [5]. Our idea now is to speed up the construction of a shortest path by *reducing this input complexity* for the motion–planning procedure using a spatial index.

In large scenes, we can observe that the shortest path is often determined by only few obstacles. Therefore, it seems sensible to support the access to locally relevant obstacles with the index so that we do not have to construct the entire visibility graph to find the shortest path. In this paper, we present two different methods which realize this idea: The *ellipse method* and the *segment method*. Our methods can be viewed as *hierarchical* motion–planning techniques, since we do not only use an arbitrary index as a tool for querying objects, but our index is an integral part of the motion–planning procedure. The index we use is the *monotonous bisector tree* [6]. Other attempts to constitute a hierarchical approach are the space–time octree [7], cell decomposition with a binary space partition [8] and hierarchical approximate cell decomposition [9].

To obtain our results we adopt the property of *low object density* [10], which states that an area of constant size may be intersected only by a constant number of objects. The preprocessing is the construction of the monotonous bisector tree (MBT) in time $\mathcal{O}(n \log n)$. When obstacles must be inserted in or deleted from the scene, our data structures can be updated in amortized time $\mathcal{O}(\log^2 n)$. A new obstacle can be inserted in the visibility graph in time $\mathcal{O}(n \log n + k)$, where k is the size of the change. This provides an $\mathcal{O}(n^2 \log n)$ effort to incrementally construct the graph. The ellipse method constructs a shortest path in time $\mathcal{O}(l^4 \log l + n \log l)$, if a shortest path of length l exists. Moreover, if the assumption concerning the query performance of the MBT hold, we obtain an improved running time of $\mathcal{O}(l^4 \log l + l^2 \log n)$. If l is not small compared to the diameter D_S of the scene, the algorithm still respects an $\mathcal{O}(n^2(\log n + \log D_S))$ upper bound. If no path exists, then similar estimations depending on a diameter of a set of obstacles certifying the failure can be stated.

The paper is organized as follows: The next section gives the problem formulation, the model of the scene of obstacles and their properties demanded. Section 3 introduces the monotonous bisector tree and Section 4 the visibility graph with the result on its incremental construction. In Sections 5 and 6 we describe and analyze the two motion–planning methods. Section 7 reports on some experimental results.

2 Problem Formulation

We want to compute a *shortest path* from a start s to a target t in a scene of polygonal obstacles. We could model the obstacles by disjoint simple polygons, but this implies the following disadvantage: Assume, we want to compute a shortest path between s and t in a scene consisting of a simple polygon with complexity $n = 4t+4$, where t is the number of "teeth" on top of the "comb", as displayed in Figure 1. Using this single obstacle, we would have to take into account all n vertices to compute a shortest path, independent of the length of such a path. Contrary to this, we will model the obstacles by *possibly intersecting convex polygons of constant complexity*, as indicated by the dotted lines in

the figure. Then, it is sufficient to compute the shortest path only under consideration of the shaded base of the "comb". Moreover, it is more likely to establish a spatial data structure for the potentially smaller convex parts of constant complexity. On the other hand, this setting will prove to be more complex for the construction of the visibility graph.

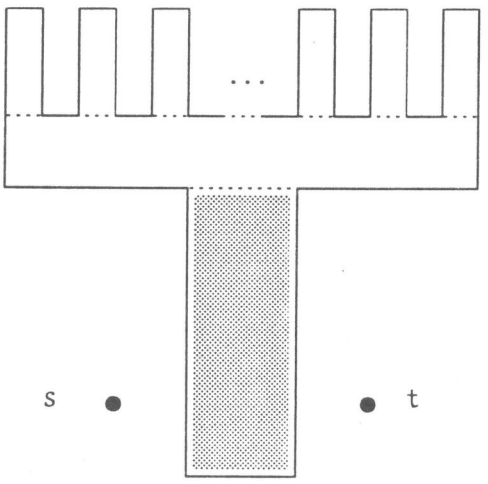


Figure 1: Modeling the obstacles with simple polygons is disadvantageous.

Nevertheless, we have to pose some properties of the scene to avoid degenerate cases. To this end we adopt Property 2.2 of [10] (formulated in d–dimensions):

Property 2.1 Let $\mathbb{E}^d$ be a space with a set O of polygonal objects, where each object has a minimal enclosing hypersphere diameter of at least ρ. Then the set O is of *low (object) density* if any region with minimal enclosing hypersphere diameter $D \cdot \rho$, for some constant $D \geq 0$, intersects no more than a constant number, say $\delta_1(D\rho)^d$, of objects in O.

This property prevents the vertices to admit an accumulation point. This assumption seems to be quite realistic in many practical applications. Van der Stappen showed that e.g. a set of non–intersecting k–fat objects is of low object density.

To be able to construct the visibility graph within sensible time bounds we additionally have to demand another property.

Property 2.2 For a constant $\sigma \in \mathbb{N}$, each object of O intersects at most σ other objects in O. The set of obstacles is then said to be of *low intersection*.

This property does not seem to be a severe restriction for our purposes. If we aim to model scenes of obstacles, as described before, a low intersection of the obstacles with e.g. $\sigma = 8$ (by an orthogonal grid) will be sufficient to cover any simple polygon in the plane. For the sequel we assume that the scenes under consideration fulfill the low object density and low intersection property.

We assume a point–like robot $\mathcal{R}$ which can move in the free space, i.e. $\mathcal{R}$ must not intersect any obstacle. To handle also motions along the boundary of an obstacle $o \in O$, we permit $\mathcal{R}$ to move on obstacle edges or vertices, *as long as they do not intersect another obstacle.* Therefore, it is not possible to pass a vertex, which is a common vertex of two obstacles.

References

[1] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, 1991.

[2] T. Lozano-Pérez and M. Wesley. An Algorithm for planning collision–free Paths among polyhedral Obstacles. *Communications of the ACM*, 22(10), 560–570, 1979.

[3] H. Rohnert. Shortest Paths in the Plane with Convex Obstacles. *Information Processing Letters*, **23**, 71–76, 1986.

[4] H. Rohnert. Time and Space Efficient Algorithms for Shortest Paths between Convex Polygons. *Information Processing Letters*, 27, 175–179, 1988.

[5] A. Gajentaan and M.H. Overmars. On a Class of $\mathcal{O}(n^2)$ Problems in Computational Geometry. Technical Report RUU–CS–93–15, Dept. of Computer Science, University of Utrecht, 1993.

[6] H. Noltemeier, K. Verbarg, and C. Zirkelbach. Monotonous Bisector* Trees – A Tool for Efficient Partitioning of Complex Scenes of Geomtric Objects. In B. Monien and Th. Ottmann, editors, *Data Structures and Efficient Algorithms*, volume 594 of *Lecture Notes in Computer Science*, 186–203. Springer, 1992.

[7] K. Fujimura and H. Samet. Path Planning among Moving Obstacles using Spatial Indexing. In *IEEE International Conference on Robotics and Automation*, 1662–1667, 1988.

[8] C. Ballieux. Motion Planning using Binary Space Partition. Technical Report 93-25, Utrecht University, 1993.

[9] D. Zhu and J.-C. Latombe. New Heuristic Algorithms for Efficient Hierarchical Path Planning. *IEEE Transactions on Robotics and Automation*, 7(1), 9–20, 1991.

[10] A.F. van der Stappen and M.H. Overmars. Motion Planning in Environments with Low Obstacle Density. Technical Report 33, Utrecht University, 1995.

[11] K. Verbarg and H. Noltemeier. Dynamic Environmental Modeling by the C–Tree. In H. Bunke, T. Kanade, and H. Noltemeier, editors, *Modelling and Planning for Sensor–Based Intelligent Robot Systems.* World Scientific, 1995.

[12] G. Vegter. Dynamically maintaining the visibility graph. In *Proc. 2nd Workshop Algorithms Data Struct.*, volume 519 of *Lecture Notes in Computer Science*, 425–436. Springer-Verlag, 1991.

Optimal Robot Localization in Trees

Kathleen Romanik * Sven Schuierer†

1 Introduction

The problem of *localization*, i.e. of a robot finding its position on a map, is an important task for autonomous mobile robots. It has applications in numerous areas of robotics ranging from aerial photography to autonomous vehicle exploration. In this paper we present a new strategy for a robot to find its position on a map where the map is represented as a geometric tree. Our strategy exploits to a high degree the self-similarities that may occur in the environment.

The environment of the robot is usually assumed to be a simple or multiply connected polygon P and the robot is assumed to have access to its local visibility polygon V, i.e. all the points that are visible from its position, via an on-board camera system. The robot is also assumed to have a compass so that it knows its orientation. The first task of the robot is to determine its possible placements within P; that is, all $p \in P$ such that the visibility polygon of p equals V. Guibas, Motwani, and Raghavan provide a data-structure that allows efficient enumeration of all such positions [2].

If more than one placement exists, then the robot has to travel to certain points of the polygon that allow it to distinguish between the different placements. Ideally it should not travel much farther than necessary. In order to measure the performance of a localization strategy we employ the framework of competitive analysis [4]. If $L(p, P)$ is the length of a shortest path to localize a robot "waking" in polygon P at position p, a strategy S is called *c-competitive* if the length of the path traveled by a robot using strategy S is at most c times $L(p, P)$, for all possible polygons P and points $p \in P$. The value c is called the *competitive ratio* of S.

A slightly different approach, proposed by Kleinberg, leaves aside all concerns raised by the visibility structure of P and abstracts the combinatorial nature of the problem [3]. In this context two types of environments are considered: bounded-degree trees embedded in $I\!E^d$ (called geometric trees) and rectangle packings in the plane. In these environments the robot has no use of vision other than to know the orientation of all edges incident to its current location. For geometric trees Kleinberg provides a strategy with a competitive ratio of $O(n^{2/3})$, where n is the number of vertices of degree greater than two, and for rectangle packings a strategy with a competitive ratio of $O(n\sqrt{\frac{\log\log n}{\log n}})$, where n is the number of rectangles.

Kleinberg also provides a lower bound of $\Omega(\sqrt{n})$ for the localization problem in geometric trees, which is illustrated by the example in Figure 1. If the true placement s of the robot is at the bottom of the d^{th} spike to the right of spike t, where $\sqrt{n} < d \leq 2\sqrt{n}$, then a localization strategy either has to explore all spikes between t and s or to travel to one of the end points q_1 or q_2 of the base b of the tree. In each case the total distance traveled by the robot is $\Omega(n)$ while the shortest path from s to t, which is of length at most $3\sqrt{n} + 1$, is the shortest path to localize the robot.

*DIMACS, Rutgers University, Piscataway, NJ 08855-1179, email: romanik@dimacs.rutgers.edu

†Institut für Informatik, Universität Freiburg, Am Flughafen 17, Geb. 051, D-79110 Freiburg, Fed. Rep. of Germany, e-mail: schuiere@informatik.uni-freiburg.de

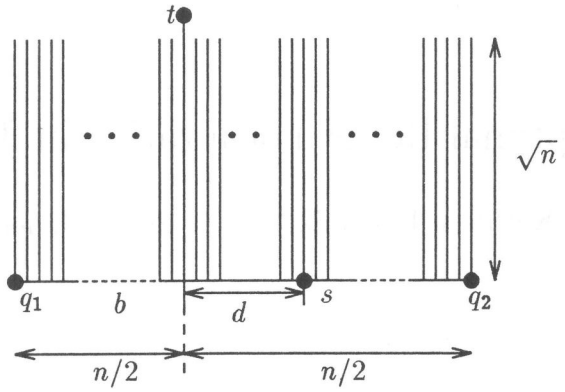


Figure 1: A geometric tree for which every on-line localization strategy is no better than $\Omega(\sqrt{n})$-competitive.

2 A New Localization Strategy

Our localization strategy consists of four main steps. The following pseudo-code gives a rough impression of how the strategy works.

Strategy *Localize-by-Placement-Separation (LPS)*
Input: A geometric tree $T = (V, E)$ with n non-degenerate vertices;
Output: The position of the robot in T;

Variables: The set of placements P;
The tree $\overline{T}_{ex}$ that has been explored;
The radius ϱ of the largest ball around p_0 that is contained in $\overline{T}_{ex}$;

$P := V$; $\overline{T}_{ex} := \{\mathbf{0}\}$; $\varrho := 0$;
Step 1: Perform a spiral search until either $|\overline{T}_{ex}| > 2\sqrt{n}$ or $|P| = 1$;
Step 2: Identify a critical path of $\overline{T}_{ex}$;
Step 3: Eliminate close placements until $d_T(p, p') > \varrho$, for all $p, p' \in P$;
Step 4: Eliminate the remaining placements using a greedy strategy;
end *Localize-by-Placement-Separation*;

2.1 Step 1: Spiral Search

In Step 1 of Strategy LPS the robot performs a spiral search [1] as follows: starting at the origin it performs successive depth-first searches, each time visiting all points within distance d of the origin (where initially d is set to 1) and then doubling d if it has not localized itself. During the search the robot keeps track of the size $|\overline{T}_{ex}|$ of the explored subtree $\overline{T}_{ex}$ and the set P of all possible placements of the robot, i.e., the set of all $p \in T$ with $p + \overline{T}_{ex} \subseteq T$. If $|P| = 1$, then the robot stops its search and outputs $p \in P$ as its position in T.

2.2 Step 2: Identifing a critical path of $\overline{T}_{ex}$

If after Step 1 there are at least two placements p_1 and p_2 with $d_T(p_1, p_2) < \varrho$, then in Step 2 the robot identifies a "critical path" of $\overline{T}_{ex}$. The critical path can be found using the following lemma.

Lemma 2.1 *[3] If p is a placement in P, then the set of all placements that are contained in $(p + \overline{T}_{ex})$ induces either a simple periodic path or a comb-tree.*

A tree $\mathcal{T}$ is called a *comb-tree* if there is a simple path $\mathcal{Q}$ with start point s and end point t and a simple path $\mathcal{S}$ also with start point s such that $\mathcal{T}$ is given by the union of the k-fold concatenation of $\mathcal{Q}$ with itself (the *base* of $\mathcal{T}$) and the sets $k_i(t-s)+\mathcal{S}$ (the *spikes* of $\mathcal{T}$), for $1 \leq i \leq m$ and $0 = k_1 < \cdots < k_m = k$.

If the placements in $(p + \overline{T}_{ex})$ induce a simple periodic path $\mathcal{P}$, then $\mathcal{P}$ is the critical path. If the placements in $(p + \overline{T}_{ex})$ induce a comb-tree, then the base of the comb-tree forms the critical path.

2.3 Step 3: Eliminating Close Placements

If after Step 2 there are pairs (p_1, p_2) of placements with $d_T(p_1, p_2) \leq \varrho$—so-called *critical pairs*, then in Step 3 the robot explores branches emanating from the critical path $\overline{C}_{ex}$ until either it has found a periodic tree $\overline{\mathcal{P}}_{ex}$ with $\sqrt{n}$ vertices or there are no more critical pairs. If it succeeds in finding a periodic tree $\overline{\mathcal{P}}_{ex}$ with $\sqrt{n}$ vertices, then there is some point where the tree $\overline{\mathcal{P}}_{ex}^{2\sqrt{n}}$ of $2n$ vertices formed by concatenating together $2\sqrt{n}$ copies of $\overline{\mathcal{P}}_{ex}$ differs from $\overline{T}$. The robot can find one such mismatch by exploring only a small portion of $\overline{\mathcal{P}}_{ex}^{2\sqrt{n}}$. It then propagates this initial mismatch towards $\mathbf{0}$ until all critical pairs are eliminated.

2.4 Step 4: Greedy Elimination

Since $2\sqrt{n}$ non-degenerate vertices of T have been visited by the robot at the end of Step 1 and there are no critical pairs left at the end of Step 3, we can show that at the beginning of Step 4 the number of remaining placements is at most $2\sqrt{n}$.

Lemma 2.2 *If $|\overline{T}_{ex}| \geq 2\sqrt{n}$ and there are no critical pairs, then $|P| \leq 2\sqrt{n}$.*

In order to eliminate the remaining placements the robot visits the closest point $\overline{q}$ such that at least one placement is eliminated. Since $d_{\overline{T}}(\mathbf{0}, \overline{q})$ is no more than the shortest distance to localize the robot, a repeated application of this step guarantees that Step 4 has a competitive ratio of at most $2\sqrt{n}$.

References

[1] R. Baeza-Yates, J. C. Culberson, and G. J. E. Rawlins. Searching in the plane. *Information and Computation*, 106:234–252, 1993.

[2] L. Guibas, R. Motwani, and P. Raghavan. The robot localization problem in two dimensions. In *Proc. 3rd ACM-SIAM Symp. on Discrete Algos.*, pp. 259–268, 1992.

[3] J. M. Kleinberg. The localization problem for mobile robots. In *Proc. 35th IEEE Symp. Found. Comput. Sci.*, pp. 521–531, 1994.

[4] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, 1985.

Subquadratic algorithms for the general collision detection problem

Elmar Schömer* Christian Thiel†

December 12, 1995

Abstract

We present the first subquadratic collision detection algorithm for simultaneously moving geometric objects which works in a fairly general setting. Geometric objects are regarded as rigid bodies in 3-space and are represented by unions of triangles (polyhedra) or unions of spheres (molecules). The motions of all objects are specified by polynomial functions which describe their position and orientation at any point in time. The general framework we develop for the solution of our specific problem is interesting of its own because it may be applicable for a wide range of other problems which require the solution of systems of polynomial (in)equalities.

Classification: algorithms and data structures, computational geometry

1 Introduction

Collision detection is prerequisite for simulating the physically correct behaviour of real world processes. It is an important tool in the field of "mechanical computer aided engineering" and in the field of "computational molecular biology". There it is essential to detect unintentional interferences between objects as early as possible. Moreover real time collision detection is still a major bottleneck in most virtual reality applications. That is the reason why efficient collision detection algorithms must be developed.

Let us consider the problem of collision detection abstractly.

> Given two simultaneously moving objects B_1 and B_2 with well defined geometric form and trajectory, decide whether their motion is collision-free.

To be more specific we assume that the objects B_i are rigid bodies represented by a set of triangles or by a set of spheres. The complexity of B_i is simply measured by the cardinality of the defining set. The trajectory of each B_i is specified in advance by a polynomial function $\mathcal{C}_i(t)$ which describes its configuration at time $t \in [0,1]$. A collision between two molecules/polyhedra occurs if two spheres/triangles from different objects collide. Thus the trivial algorithm to detect a collision simply compares every pair of spheres/triangles and runs in time $O(N^2)$. Here $N = |B_1| + |B_2|$ denotes the total complexity of both objects. The contribution of this paper consists in the proof that it is possible to decide in time $o(N^2)$ whether the objects collide. This result can be seen as a generalization of the results obtained in [6].

This paper is organized as follows. First we describe the mathematical model which underlies our approach. Next we summarize previous results for some special cases of the collision detection problem, which build the kernel of the general strategy. In particular we introduce the concept of linearization. In section 4 we show how to reduce the collision detection problem for two spheres or two triangles to the task to determine the number of real roots of a polynomial satisfying several polynomial inequalities.

In contrast to the solutions proposed in [1, 2], which are based on Sturm sequences, we apply some long known theorems of Jacobi, Hermite and Sylvester which deal with the existence of real roots of a system of univariate polynomial equations/inequalities (see sections 5, 6). It turns out, that this technique is especially suitable to find an appropriate linearization. The combination of the data structures in section 3 and this linearization yields a subquadratic algorithm.

2 Preliminaries

The configuration of a rigid body B_i in 3-space can be easily specified by the position and orientation of its local coordinate frame. Let vector $\mathbf{o}_i \in \mathbb{R}^3$ denote the position of B_i's reference point and quaternion $\mathfrak{r}_i \in \mathbb{R}^4$ its orientation. Quaternion calculus provides an elegant

*Universität des Saarlandes, Fachbereich 14, Informatik, Im Stadtwald, D-66041 Saarbrücken, Germany. E-mail: schoemer@cs.uni-sb.de.

†Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany. E-mail: thiel@mpi-sb.mpg.de. This author was supported by the ESPRIT Basic Research Actions Program, under contract No. 7141 (project ALCOM II).

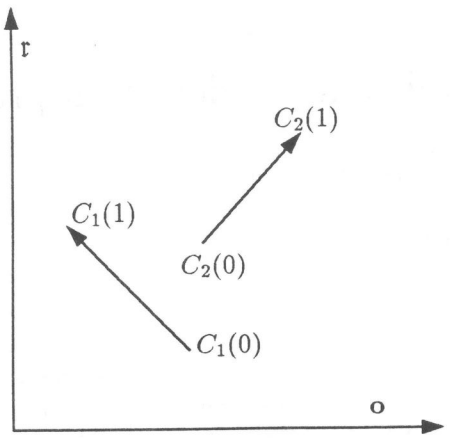


Figure 1: Configuration space

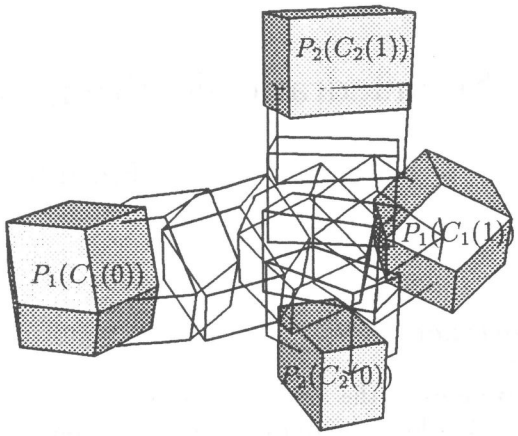


Figure 2: Physical space

way of algebraically specifying orientations in 3-space analogous to complex numbers in 2-space. (For a short review of quaternion calculus see appendix A.) Thus the configuration $\mathcal{C}_i(t)$ at time t can be represented as a tuple $(\mathbf{o}_i(t), \mathfrak{r}_i(t)) \in \mathbb{R}^3 \times \mathbb{R}^4$. For a moving body B_i the configuration $\mathcal{C}_i(t)$ varies with time and defines a unique trajectory. For simplicity we assume that the motion is confined to the time interval $[0, 1]$.

$$\begin{aligned} \mathcal{C}_i(t): \quad & [0,1] \mapsto \mathbb{R}^3 \times \mathbb{R}^4 \\ & \mathcal{C}_i(t) = (\mathbf{o}_i(t), \mathfrak{r}_i(t)) \\ & \text{where } \mathbf{o}_i(t) \in \mathbb{R}[t]^3 \text{ and } \mathfrak{r}_i(t) \in \mathbb{R}[t]^4 \end{aligned}$$

I.e. the components of $\mathbf{o}_i(t)$ and $\mathfrak{r}_i(t)$ are polynomials in t. Let d_i and d_i' denote their degree. In vector-matrix notation the trajectory of a single point $\mathbf{x} \in B_i$ is given by

$$\mathbf{x}(t) \;=\; \mathbf{R}_i(t)\mathbf{x} + \mathbf{o}_i(t). \tag{1}$$

where $\mathbf{R}_i(t) \in \mathbb{R}(t)^3 \times \mathbb{R}(t)^3$ is the rotation matrix corresponding to the quaternion $\mathfrak{r}_i(t)$ (see equation (3)).

The following table shows the simplest kinds of motion of a rigid body B_i for the case of constant or linearly changing positions and orientations and in figure 1 and 2 such motions are depicted in configuration and in physical space.

$d_i = 0$	$d_i' = 0$	B_i remains stationary
$d_i = 1$	$d_i' = 0$	B_i is translated in a fixed direction
$d_i = 0$	$d_i' = 1$	B_i rotates about a fixed axis
$d_i = 1$	$d_i' = 1$	B_i's motion is a superposition of a translation and a rotation

3 General approach

For the collision detection problem for a stationary and a translationally moving polyhedron respectively a polyhedron rotating about a fixed axis subquadratic algorithms were presented in [6]. In this paper we will use the same data structures for solving the general version of the problem. For completeness we give a short overview of the approach in [6]. The basis of our algorithm is an efficient solution of the following subproblem:

> Let $\mathcal{S}$ be a set of rigid bodies of the same type and let $\mathcal{Q}$ be a second set of rigid bodies of another type. Build a data structure that, given a query object $Q \in \mathcal{Q}$ decides quickly whether the object Q collides with an object from the set $\mathcal{S}$ during its motion. We call this the *on-line collision problem for $\mathcal{Q}$ with respect to $\mathcal{S}$.*

Our strategy is to reduce the collision problem to a problem for other objects that do not move and then solve the latter by known techniques. This is done by the concept of linearization. To find a *linearization* of the collision problem means to establish the equivalence

$$\begin{aligned} & [\exists t \in [0,1]: \; S(t) \cap Q(t) \neq \emptyset] \\ & \iff \bigvee_{i=1}^{dis} \bigwedge_{j=1}^{con} \left[\sum_{k=1}^{dim} \sigma_k^{ij}(S)\, \delta_k^{ij}(Q) \bowtie 0 \right], \end{aligned} \tag{2}$$

where $\bowtie \in \{<, >, =\}$, *dis*, *con*, *dim* are positive constants, and $\sigma_k^{ij}(S)$ respectively $\delta_k^{ij}(Q)$ are rational functions of constant degree depending on the kind of motion and kind of objects.

Having such a linearization we map the objects $S \in \mathcal{S}$ into the points $p^{ij} := (\delta_1^{ij}(S), \delta_2^{ij}(S), \ldots, \delta_{dim}^{ij}(S))$ in $\mathbb{R}^{dim}$ and the query object Q into the hyperplanes $h^{ij} := (\sigma_1^{ij}(Q), \sigma_2^{ij}(Q), \ldots, \sigma_{dim}^{ij}(Q))$ in the same space. Then we can think of any $\sum_{k=1}^{dim} \sigma_k^{ij}(Q)\,\delta_k^{ij}(S) \bowtie 0$ as the condition, that (depending on $\bowtie$) the point p^{ij} lies on the hyperplane h^{ij} respectively in a halfspace bounded by h^{ij}.

Therefore the linearization (2) leads to a combination of several halfspace range searching problems. A general notation for such combined search problems was first introduced in [4]:

Let $\mathcal{P} = \{p_1, p_2, \ldots, p_N\}$ be a set of N points in $\mathbb{R}^{dim}$, let $\mathcal{R}$ denote the set of all simplices in $\mathbb{R}^{dim}$, let $\mathcal{S} = \{s_1, \ldots, s_N\}$ be a set of N objects, and let $\mathcal{Q}$ denote a set of queries on $\mathcal{S}$. The *composed query problem* $(\mathcal{S}', \mathcal{Q}')$ is defined as follows: $\mathcal{S}' = \{(p_i, s_i); 1 \le i \le N\}$, $\mathcal{Q}' = \mathcal{R} \times \mathcal{Q}$ and the answer set for a query $(R, Q) \in \mathcal{Q}'$ is given by $\{(p, s);\ (p, s) \in \mathcal{S}' \text{ and } p \in R \text{ and } s \in Q\}$. We also say that $(\mathcal{S}', \mathcal{Q}')$ is obtained from $(\mathcal{S}, \mathcal{Q})$ by *simplex composition.*

Simplices in *dim*-space are the intersection of at most $dim + 1$ many halfspaces. Therefore we can w.l.o.g. consider simplex compositions where the simplices are halfspaces. In this case we also use the term *halfspace composition.*

Because each conjunction of (2) can be interpreted as the composition of *con* halfspace range searching problems we can find the objects in $\mathcal{S}$ satisfying a particular conjunction by applying halfspace composition *con* times. The disjunctions of (2) correspond to the union of ranges.

In his Ph.D. thesis [4] Marc van Kreveld investigated efficient solutions for simplex composition[1] of query problems:

Theorem 1 ([4]) *Let $\mathcal{P}$ be a set of N points in dim-space, and let $\mathcal{S}$ be a set of N objects in correspondence with $\mathcal{P}$. Let T be a data structure on $\mathcal{S}$ having building time $b(N)$, size $s(N)$ and query time $q(N)$. For an arbitrary small constant $\epsilon > 0$, the application of simplex composition on $\mathcal{P}$ to T results in a data structure D of building time $O(M^{\epsilon}(M + b(N)))$, size $O(M^{\epsilon}(M + s(N))$ and query time $O(N^{\epsilon}(q(N) + N/M^{1/dim}))$ for every fixed M such that $N \le M \le N^{dim}$, assuming that $s(N)/N$ is non-decreasing and $q(N)/N$ is non-increasing. Reporting takes additional $O(K)$ time if there are K answers.*

Assume we have N objects $Q \in \mathcal{Q}$ instead of only one and we want to decide whether there occurs any collision between any pair Q, S, for $Q \in \mathcal{Q}$ and $S \in \mathcal{S}$. We apply the solution to the on-line problem and query the data structure of theorem 1 with each element in $\mathcal{Q}$.

Using this approach we get the following result.

Corollary 1 *Given a set $\mathcal{S}$ of N objects and a set $\mathcal{Q}$ of N objects. Assume that there is a linearization of the collision problem for $\mathcal{Q}$ with respect to $\mathcal{S}$ in the form of (2). Then we can solve in $O(N^{\frac{2dim}{dim+1}+\epsilon})$ time and space the problem of collision detection between any elements of $\mathcal{Q}$ and $\mathcal{S}$.*

We have reduced the collision problem to the task of the formulation of an appropriate linearization. In [6] the linearization is derived from an explicit computation of the collision times. We could do this because the equations of the motions had degree at most two. The natural question is how we can proceed if the motion of the objects is more complicated, i.e. if the equations have degree greater than five (then no explicit formulation of the roots exists).

Actually we do not need an explicit representation of the collision times for the linearization. We only need to know whether two particular objects collide during the specified time period.

4 Collision of two molecules/polyhedra

In the following we deal either with the collision between two molecules or between two polyhedra.

A collision between two molecules occurs if a sphere of one molecule collides with a sphere of the other one.

A collision between two polyhedra is a little bit more difficult to characterize. A collision occurs if a vertex of one polyhedron hits a vertex/edge/face of the other one or if two edges collide. We want to derive polynomial formulas for the different types of collision on the condition that each point $\mathbf{x}$ of body B_i moves according to equation 1. We begin with the discussion of the collision of two spheres. After that we derive a necessary condition for the collision of two edges. By extending the edges to infinite lines we get a set of potential collision times as roots of a univariate polynomial. With the help of additional polynomial inequalities we can restrict this set to those roots which actually represent a collision of the edges (see 4.2-4.3). In order to take care of the restricted duration of the motion we introduce the inequality $g_0(t) := t(1 - t) > 0$.

In sections 4.4 and 4.5 we proceed in an analogous way in order to deal with the collision of a vertex and a face. The extension of the face to a plane enables us to find a superset of the desired collision times.

[1] Actually we use only halfspace composition

4.1 Collision of two spheres

If a moving sphere $S_a(t)$ (center $\mathbf{a}$, radius a_0) collides with an other moving sphere S_b it holds:

$$\begin{aligned}
&|\mathbf{a}(t) - \mathbf{b}(t)| = a_0 + b_0 \\
\iff &(\mathbf{R}_1(t)\mathbf{a} + \mathbf{o}_1(t) - \mathbf{R}_2(t)\mathbf{b} - \mathbf{o}_2(t))^2 = (a_0 + b_0)^2 \\
\iff &(\mathbf{o}_1(t) - \mathbf{o}_2(t))^2 - 2\mathbf{a}^T\mathbf{R}_1(t)^T\mathbf{R}_2(t)\mathbf{b} \\
&+2(\mathbf{o}_1(t) - \mathbf{o}_2(t))^T(\mathbf{R}_1(t)\mathbf{a} - \mathbf{R}_2(t)\mathbf{b}) \\
&-(a_0 + b_0)^2 + \mathbf{a}^2 + \mathbf{b}^2 = 0
\end{aligned}$$

By multiplying with the denominators we get a polynomial $f(t)$ of degree $2(d_1' + d_2') + 2\max\{d_1, d_2\}$. Its zeros correspond to the collision times.

4.2 Collision of lines

If a line $L_{ab}(t)$ collides with an other line $L_{cd}(t)$ the points $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$, $\mathbf{d}$ lie in a common plane. This can be expressed as the vanishing of the following determinant

$$\begin{aligned}
&\det\begin{bmatrix} 1 & 1 & 1 & 1 \\ \mathbf{a}(t) & \mathbf{b}(t) & \mathbf{c}(t) & \mathbf{d}(t) \end{bmatrix} = 0 \\
\iff\ &(\mathbf{d}(t) - \mathbf{c}(t))^T(\mathbf{a}(t) \times \mathbf{b}(t)) \\
&+(\mathbf{c}(t) \times \mathbf{d}(t))^T(\mathbf{b}(t) - \mathbf{a}(t)) = 0 \\
\iff\ &(\mathbf{d} - \mathbf{c})^T\mathbf{R}_2(t)^T\mathbf{R}_1(t)(\mathbf{a} \times \mathbf{b}) \\
&+(\mathbf{c} \times \mathbf{d})^T\mathbf{R}_2(t)^T\mathbf{R}_1(t)(\mathbf{b} - \mathbf{a}) \\
&+(\mathbf{o}_1(t) - \mathbf{o}_2(t))^T(\mathbf{R}_1(t)(\mathbf{b} - \mathbf{a}) \\
&\times\mathbf{R}_2(t)(\mathbf{d} - \mathbf{c})) = 0
\end{aligned}$$

Here the resulting polynomial $f(t)$ has degree $2(d_1' + d_2') + \max\{d_1, d_2\}$.

4.3 Collision of line segments

A collision between two open line segments $l_{ab}(t)$ and $l_{cd}(t)$ (not involving one of the end points) occurs only if the corresponding lines collide and if the following inequalities are fulfilled for $\mathbf{s} = \mathbf{e}_1$ or $\mathbf{s} = \mathbf{e}_2$ or $\mathbf{s} = \mathbf{e}_3$.

$$\begin{aligned}
&D_s(\mathbf{b} - \mathbf{a}, \mathbf{c})(t) < D_s(\mathbf{b}, \mathbf{a})(t) < D_s(\mathbf{b} - \mathbf{a}, \mathbf{d})(t) \\
\wedge\ &D_s(\mathbf{d} - \mathbf{c}, \mathbf{a})(t) < D_s(\mathbf{d}, \mathbf{c})(t) < D_s(\mathbf{d} - \mathbf{c}, \mathbf{b})(t) \\
\vee\ &D_s(\mathbf{b} - \mathbf{a}, \mathbf{c})(t) > D_s(\mathbf{b}, \mathbf{a})(t) > D_s(\mathbf{b} - \mathbf{a}, \mathbf{d})(t) \\
\wedge\ &D_s(\mathbf{d} - \mathbf{c}, \mathbf{a})(t) > D_s(\mathbf{d}, \mathbf{c})(t) > D_s(\mathbf{d} - \mathbf{c}, \mathbf{b})(t)
\end{aligned}$$

with the abbreviation

$$D_s(\mathbf{u}, \mathbf{v})(t) = \det[\mathbf{s}, \mathbf{u}(t), \mathbf{v}(t)]$$

We want to examine the first inequality in the first row. Substitution of the motion equation for the points yields:

$$\begin{aligned}
&\mathbf{s}^T(\mathbf{R}_1(t)(\mathbf{b} - \mathbf{a}) \times (\mathbf{R}_2(t)\mathbf{c} + \mathbf{o}_2(t)) \\
&+\mathbf{R}_1(t)(\mathbf{a} \times \mathbf{b}) + \mathbf{o}_1(t) \times \mathbf{R}_1(t)(\mathbf{b} - \mathbf{a})) < 0
\end{aligned}$$

4.4 Collision of a point and a plane

Since all points of a moving plane $H(t) = \{\mathbf{x} \mid \mathbf{n}(t)^T\mathbf{x} = n_0(t)\}$ fulfill equation (1), the normal vector $\mathbf{n}(t)$ and the parameter $n_0(t)$ change as follows:

$$\begin{aligned}
\mathbf{n}(t) &= \mathbf{R}_i(t)\mathbf{n} \\
n_0(t) &= n_0 + \mathbf{o}_i(t)^T\mathbf{n}(t)
\end{aligned}$$

If a point $\mathbf{a}(t)$ hits the plane $H(t)$ it holds:

$$\begin{aligned}
&\mathbf{n}(t)^T\mathbf{a}(t) = n_0(t) \iff \\
&\mathbf{n}^T\mathbf{R}_2(t)^T\mathbf{R}_1(t)\mathbf{a} + \mathbf{n}^T\mathbf{R}_2(t)^T(\mathbf{o}_1(t) - \mathbf{o}_2(t)) = n_0
\end{aligned}$$

4.5 Collision of a point and a triangle

A moving point $\mathbf{a}(t)$ hits the interior of a moving triangle $\Delta_{bcd}(t)$, if it collides with the supporting plane $H_n : \mathbf{n}^T\mathbf{x} = n_0$ of this triangle.

$$\begin{aligned}
\mathbf{n} &= \mathbf{b} \times \mathbf{c} + \mathbf{c} \times \mathbf{d} + \mathbf{d} \times \mathbf{b} \\
n_0 &= \mathbf{b}^T(\mathbf{c} \times \mathbf{d})
\end{aligned}$$

For $\mathbf{s} = \mathbf{e}_1$ or $\mathbf{s} = \mathbf{e}_2$ or $\mathbf{s} = \mathbf{e}_3$ the following inequalities must additionally hold:

$$\begin{aligned}
&D_s(\mathbf{d}, \mathbf{b})(t) > D_s(\mathbf{a}, \mathbf{b} - \mathbf{d})(t) > D_s(\mathbf{c}, \mathbf{b} - \mathbf{d})(t) \\
\wedge\ &D_s(\mathbf{d}, \mathbf{c})(t) < D_s(\mathbf{a}, \mathbf{c} - \mathbf{d})(t) < D_s(\mathbf{b}, \mathbf{c} - \mathbf{d})(t) \\
\vee\ &D_s(\mathbf{d}, \mathbf{b}(t))(t) < D_s(\mathbf{a}, \mathbf{b} - \mathbf{d})(t) < D_s(\mathbf{c}, \mathbf{b} - \mathbf{d})(t) \\
\wedge\ &D_s(\mathbf{d}, \mathbf{c})(t) > D_s(\mathbf{a}, \mathbf{c} - \mathbf{d})(t) > D_s(\mathbf{b}, \mathbf{c} - \mathbf{d})(t)
\end{aligned}$$

4.6 Collision of a point and a line

A moving point $\mathbf{a}(t)$ collides with a moving line $L_{cd}(t)$ iff

$$\begin{aligned}
&u_1(t)^2 + u_2(t)^2 + u_3(t)^2 = 0 \\
&\mathbf{u}(t) = (\mathbf{d}(t) - \mathbf{c}(t)) \times \mathbf{a}(t) + \mathbf{c}(t) \times \mathbf{d}(t)
\end{aligned}$$

4.7 Collision of a point and a line segment

A moving point $\mathbf{a}(t)$ collides with a moving line segment $l_{cd}(t)$ if it collides with the line $L_{cd}(t)$ and the following conditions hold:

$$\begin{aligned}
&c_1(t) < a_1(t) < d_1(t) \vee c_1(t) > a_1(t) > d_1(t) \\
\vee\ &c_2(t) < a_2(t) < d_2(t) \vee c_2(t) > a_2(t) > d_2(t) \\
\vee\ &c_3(t) < a_3(t) < d_3(t) \vee c_3(t) > a_3(t) > d_3(t)
\end{aligned}$$

4.8 Collision of two points

Two moving points $\mathbf{a}(t)$ and $\mathbf{b}(t)$ collide iff

$$\begin{aligned}
&u_1(t)^2 + u_2(t)^2 + u_3(t)^2 = 0 \\
&\mathbf{u}(t) = \mathbf{b}(t) - \mathbf{a}(t)
\end{aligned}$$

5 Existence of real roots of a polynomial satisfying several polynomial inequalities

Wanted:

$$\#\{t \in \mathbb{R} \mid f(t) = 0 \wedge g_1(t) > 0 \wedge \ldots \wedge g_l(t) > 0\},$$
where $f(t), g_i(t)$ are polynomials

$Z_f = \{t \in \mathbb{R} \mid f(t) = 0\}$ denotes the set of real roots of polynomial f and $\chi_g(t)$ the characteristic function of the predicate $[g_1(t) > 0 \wedge \ldots \wedge g_l(t) > 0]$. It holds:

$$\begin{aligned}&\#\{t \in \mathbb{R} \mid f(t) = 0 \wedge g_1(t) > 0 \wedge \ldots \wedge g_l(t) > 0\}\\ &= \sum_{t \in Z_f} \chi_g(t)\end{aligned}$$

$\chi_g(t)$ can be expressed as follows

$$\begin{aligned}\chi_g(t) &= 2^{-l} \prod_{i=1}^{l} (1 + \operatorname{sgn} g_i(t))\\ &= 2^{-l} \sum_{I \in 2^{\{1,\ldots,l\}}} \prod_{i \in I} \operatorname{sgn} g_i(t)\end{aligned}$$

This implies, that

$$\begin{aligned}\sum_{t \in Z_f} \chi_g(t) &= 2^{-l} \sum_{I \in 2^{\{1,\ldots,l\}}} \sum_{t \in Z_f} \prod_{i \in I} \operatorname{sgn} g_i(t)\\ &= 2^{-l} \sum_{I \in 2^{\{1,\ldots,l\}}} \Big(\#\{t \in Z_f \mid g_I(t) > 0\}\\ &\qquad - \#\{t \in Z_f \mid g_I(t) < 0\}\Big)\end{aligned}$$

where $g_I(t) = \prod_{i \in I} g_i(t)$ and $g_\emptyset(t) = 1$.

This method for the calculation of $\#\{t \in \mathbb{R} \mid f(t) = 0 \wedge g_1(t) > 0 \wedge \ldots \wedge g_l(t) > 0\}$ goes back to [5]. In this way the original problem is reduced to the calculation of the number of real roots of $f(t)$, which satisfy a single polynomial inequality $[g(t) > 0]$.

6 Hermite's method for the calculation of $\#\{t \in \mathbb{R} \mid f(t) = 0 \wedge g(t) > 0\}$

Let us consider two polynomials

$$\begin{aligned}f(t) &= u_0 t^n + u_1 t^{n-1} + \ldots + u_n \quad \text{and}\\ g(t) &= v_0 t^m + v_1 t^{m-1} + \ldots + v_m\end{aligned}$$

Let $\lambda_1, \ldots, \lambda_n$ denote the roots of $f(t)$ and s_k the Newton sum $\sum_{i=1}^n \lambda_i^k$. In addition we define $h_k = \sum_{i=1}^n g(\lambda_i)\lambda_i^k$. Since s_k and h_k are symmetrical polynomials in $\lambda_1, \ldots, \lambda_n$ they can be expressed as rational functions in the coefficients of f and g. It holds:

$$s_k = \begin{cases} n & \text{for } k = 0\\ -\dfrac{u_1}{u_0} & \text{for } k = 1\\ -\dfrac{u_1 s_{k-1} + \ldots + u_{k-1} s_1 + k u_k}{u_0} & \text{for } 2 \le k \le n\\ -\dfrac{u_1 s_{k-1} + u_2 s_{k-2} + \ldots + u_n s_{k-n}}{u_0} & \text{for } k > n\end{cases}$$

$$h_k = v_0 s_{k+m} + v_1 s_{k+m-1} + \ldots + v_m s_k$$

In the following the two Hankel matrices $\mathbf{S}$ and $\mathbf{H}$ play a decisive role. They are composed of the values s_k and h_k.

$$\mathbf{S} = [s_{i+j}]_{i,j=0}^{n-1} = \begin{bmatrix} s_0 & s_1 & s_2 & \ldots & s_{n-1}\\ s_1 & s_2 & s_3 & \ldots & s_n\\ \vdots & \vdots & \vdots & & \vdots\\ s_{n-1} & s_n & s_{n+1} & \ldots & s_{2n-2}\end{bmatrix}$$

$$\mathbf{H} = [h_{i+j}]_{i,j=0}^{n-1}$$

Let S_i (and analogous H_i)

$$S_i = \det \begin{bmatrix} s_0 & s_1 & \ldots & s_{i-1}\\ s_1 & s_2 & \ldots & s_i\\ \vdots & \vdots & & \vdots\\ s_{i-1} & s_i & \ldots & s_{2i-2}\end{bmatrix}$$

The following theorems hold:

Theorem 2 (Jacobi) *The number of distinct roots of $f(t)$ equals the rank r of matrix $\mathbf{S}$ and the number of distinct real roots equals $r - 2V(1, S_1, S_2, \ldots, S_r)$, where the function $V(.)$ counts the number of sign changes of a sequence.*

Theorem 3 (Hermite,Sylvester) *The number of distinct real roots of $f(t)$ satisfying the condition $g(t) > 0$ equals $r - V(1, S_1, S_2, \ldots, S_r) - V(1, H_1, H_2, \ldots, H_r)$.*

If there are no degeneracies the rank r equals n. If one of the sequences $S_1, \ldots, S_r$ respectively $H_1, \ldots, H_r$ contains zeros, the rule of Frobenius (see [3]) can be applied.

S_i and H_i are rational functions in the coefficients of f and g with denominators u_0^{2i-2} and u_0^{2i+m-2} respectively. The degree of their numerator fulfills: $\deg(\text{numer}(S_i)) = 2i - 2$ and $\deg(\text{numer}(H_i)) = 2i + m - 2$. Especially it holds (see [7]):

$$\begin{aligned}S_n &= \mathcal{D}(f)/u_0^{2n-2}\\ H_n &= \mathcal{R}(f,g)\mathcal{D}(f)/u_0^{2n+m-2}\end{aligned}$$

where $\mathcal{R}(f,g)$ denotes the resultant of f and g and $\mathcal{D}(f) = (-1)^{n(n-1)/2}\mathcal{R}(f,f')$ the discriminant of f. It holds:

$$\mathcal{R}(f,g) = \det \begin{bmatrix} u_0 & u_1 & \dots & u_n & & & \\ & u_0 & u_1 & \dots & u_n & & \\ & & \ddots & \ddots & & \ddots & \\ & & & u_0 & u_1 & \dots & u_n \\ v_0 & v_1 & \dots & v_m & & & \\ & v_0 & v_1 & \dots & v_m & & \\ & & \ddots & \ddots & & \ddots & \\ & & & v_0 & v_1 & \dots & v_m \end{bmatrix} \begin{matrix} \leftarrow 1 \\ \\ \\ \leftarrow m \\ \leftarrow m+1 \\ \\ \\ \leftarrow m+n \end{matrix}$$

7 Linearization

S_i and H_i are rational functions of the coefficients of f and g. S_i can be expressed as follows:

$$S_i = \sum_{\sum^{\nu_1,\dots,\nu_n}_{\nu_j=2i-2}} \alpha^{(i)}_{\nu_1,\dots,\nu_n} \prod_{j=1}^{n} u_j^{\nu_j} / u_0^{2i-2}$$

Since we are only interested in $\text{sign}(S_i)$ it suffices to examine the numerator $\text{numer}(S_i)$ because $u_0^{2i-2} > 0$.

The coefficients u_j can be rewritten as linear functions in d variables $x_1, \dots, x_d$ which uniquely specify the corresponding object. We call these the *linearized coordinates* of the object (E.g. for a sphere with center $\mathbf{a}$ and radius a_0 we set $x_i = a_i\,(i = 1,\dots,3)$, $x_4 = a_0$ and $x_5 = a_0^2 + \mathbf{a}^2$. For a line we use its Plücker coordinates.)

$$u_j(x_1,\dots,x_d) = \sum_{k=1}^{d} \gamma_{jk} x_k$$

This substitution transforms the numerator of S_i into the following form.

$$\text{numer}(S_i) = \sum_{\sum^{\mu_1,\dots,\mu_d}_{\mu_k=2i-2}} \zeta^{(i)}_{\mu_1,\dots,\mu_d} \prod_{k=1}^{d} x_k^{\mu_k}$$

If we introduce a new variable y_j for each product of the form $\prod_{k=1}^{d} x_k^{\mu_k}$ with $\sum_{k=1}^{d} \mu_k = 2i-2$ we get a linearized expression for $\text{numer}(S_i)$ with at most $\binom{2i-d-3}{d-1}$ terms.

$$\sum_{j=1}^{\binom{2i-d-3}{d-1}} \zeta_j y_j$$

Analogously we get a linearized expression for $\text{numer}(H_i)$ with at most $\binom{2i+m-d-3}{d-1}$ terms.

$$\sum_{j=1}^{\binom{2i+m-d-3}{d-1}} \xi_j z_j$$

where z_j is of the form $\prod_{k=1}^{d} x_k^{\mu_k}$ with $\sum_{k=1}^{d} \mu_k = 2i + m - 2$.

Whether the predicate $[\#\{t \in Z_f | g_1(t) > 0 \wedge \dots \wedge g_l(t) > 0\} > 0]$ is true or not only depends on the signs of the expressions S_i and $H_i(g_I)$ for $I \in 2^{\{1,\dots,l\}}$. Therefore we can write it as a $\{0,1\}$-valued function $P(.)$ of these signs. I.e. we can arrange all possible sign values for S_i and $H_i(g_I)$ in a function table with $n2^l$ input columns. If we express the function value in disjunctive normal form we get

$$\#\{t \in Z_f | g_1(t) > 0 \wedge \dots \wedge g_l(t) > 0\} > 0$$
$$\iff \bigvee_{\varepsilon \in \{-1,0,1\}^{n2^l}} P(\varepsilon) \bigwedge_{j=1}^{n2^l} \text{sign}(X_j) = \varepsilon_j$$

where each X_j corresponds to $\text{numer}(S_i)$ or $\text{numer}(H_i(g_I))$.

Lemma 1 *We get a linearization of dimension* $dim = \binom{2n+m+d-3}{d-1}$ *consisting of* $dis = O(3^{n2^l})$ *disjunctions and* $con = O(n2^l)$ *conjunctions. Thereby* n *denotes the degree of the polynomial* f, l *the number of inequalities* g_i, m *the maximal degree of all polynomials* g_I, *and* d *the number of the linearized coordinates of the involved objects.*

8 Summary

In our model of computation we only assume that the basic arithmetic operations $+$, $-$, and $*$ can be done in constant time. For a given problem instance the terms *dim*, *dis* and *con* are constants. Therefore we can combine Lemma 1 with Theorem 1 and get the following result.

Theorem 4 *The collision detection problem for two polyhedra or molecules, whose trajectories are described by polynomials, can be solved in subquadratic time.*

A Appendix

A.1 Orientations Described by Quaternions

The orientation of a rigid body in 3-space can be described by a quaternion

$$\mathfrak{r} = \begin{bmatrix} r_0 \\ \mathbf{r} \end{bmatrix} \in \mathbb{R}^4.$$

This orientation results from a rotation of the world frame about the axis with direction $\mathbf{r} \in \mathbb{R}^3$. The rotation angle is determined by $|\mathbf{r}|$ and the scalar r_0.

The 0-th component of a quaternion is called the scalar part, the other components comprise the vector part. Vectors in 3-space are interpreted as quaternions with scalar part 0. Quaternions form a vector space with an associative multiplication defined by

$$\mathfrak{q} \cdot \mathfrak{p} = \begin{bmatrix} q_0 \\ \mathbf{q} \end{bmatrix} \cdot \begin{bmatrix} p_0 \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} q_0 p_0 - \mathbf{q}^T \mathbf{p} \\ q_0 \mathbf{p} + p_0 \mathbf{q} + \mathbf{q} \times \mathbf{p} \end{bmatrix}.$$

The quaternion product is linear in $\mathfrak{p}$ and $\mathfrak{q}$. The conjugate quaternion $\mathfrak{q}^*$ of $\mathfrak{q}$ is formed by negating the vector part of $\mathfrak{q}$. The product $\mathfrak{q} \cdot \mathfrak{q}^*$ yields the scalar value $q_0^2 + \mathbf{q}^2$, which corresponds to the length of $\mathfrak{q}$ under the Euclidean metric in $\mathbb{R}^4$. Quaternions which satisfy $\mathfrak{q} \cdot \mathfrak{q}^* = 1$ are called unit quaternions. Let $\mathfrak{r}$ be a given quaternion. Then the mapping

$$\mathfrak{a} = \begin{bmatrix} 0 \\ \mathbf{a} \end{bmatrix} \rightarrow \mathfrak{a}' = \begin{bmatrix} 0 \\ \mathbf{a}' \end{bmatrix} = \frac{\mathfrak{r} \cdot \mathfrak{a} \cdot \mathfrak{r}^*}{\mathfrak{r} \cdot \mathfrak{r}^*}$$

describes a rotation of the vector $\mathbf{a}$ about the axis $\mathbf{r}$ about the angle $\varphi = 2 \arctan(|\mathbf{r}|/r_0)$. In matrix notation, this amounts to $\mathbf{a}' = \mathbf{R}(\mathfrak{r}) \cdot \mathbf{a}$, with

$$\mathbf{R}(\mathfrak{r}) = \frac{(r_0^2 - \mathbf{r}^2)\mathbf{I} + 2\mathbf{r}\mathbf{r}^T + 2r_0\mathbf{r}^\times}{r_0^2 + \mathbf{r}^2}. \qquad (3)$$

Here, $\mathbf{r}^\times$ denotes the canonical skew-symmetric matrix corresponding to $\mathbf{r}$. It can be easily verified that $\mathbf{R}(\mathfrak{r}) \cdot \mathbf{R}(\mathfrak{r})^T = \mathbf{I}$ and $\det(\mathbf{R}(\mathfrak{r})) = 1$.

Suppose the orientation $\mathfrak{r}(t)$ of a rigid body depends on a time parameter t in the following way

$$\mathfrak{r}(t) = \mathfrak{p} + t(\mathfrak{q} - \mathfrak{p}) \quad \text{for } t \in [0, 1].$$

The motion of the body induced by this varying orientation is a simple rotation about an axis the direction of which is given by the vector part of the quaternion product of $\mathfrak{q} \cdot \mathfrak{p}^*$. This axis of rotation can be determined by calculating the instantaneous angular velocity $\omega(t)$ of the vector $\mathbf{a}(t) = \mathbf{R}(\mathfrak{r}(t)) \cdot \mathbf{a}$.

$$\begin{aligned} \frac{d\mathbf{a}(t)}{dt} &= \omega(t) \times \mathbf{a}(t) = \omega^\times(t) \cdot \mathbf{a}(t), \\ &= \frac{d\mathbf{R}(t)}{dt} \cdot \mathbf{a} = \omega^\times(t)\mathbf{R}(t) \cdot \mathbf{a} \end{aligned}$$

This implies that

$$\omega^\times(t) = \frac{d\mathbf{R}(t)}{dt} \cdot \mathbf{R}(t)^T.$$

It turns out that the direction of $\omega(t)$ is time invariable and only its magnitude changes with time.

References

[1] M. Ben-Or, D. Kozen, and J. Reif: *The complexity of elementary algebra and geometry*, J. Comp. and Sys. Sci., Vol. 32: 251–264, (1986)

[2] J. Canny: *An improved sign determination algorithm*, AAECC-91 (1991), New Orleans

[3] F.R. Gantmacher: *The Theory of Matrices*, Chelsea, New York, (1959)

[4] M. van Kreveld: *New Results on Data Structures in Computational Geometry*, Ph.D. Thesis, University of Utrecht, The Netherlands, (1992)

[5] P. Pedersen: *Counting real zeros*, New York University, NYU Technical Report 545-R243, (1991)

[6] E. Schömer and C. Thiel: *Efficient collision detection for moving polyhedra*, Proc. 11th Annu. ACM Sympos. Comput. Geom.: 51–60, (1995)

[7] A. Y. Uteshev and S. G. Shulyak: *Hermite's method of seperation of solutions of systems of algebraic equations and its applications*, Linear Algebra and its Applications 177: 49–88, (1992)

Dynamic Collision Detection Algorithms in Computational Geometry

Marina Gavrilova[1], Jon Rokne[1] and Dmitri Gavrilov[2]

[1] Department of Computer Science, University of Calgary, Calgary, AB, Canada
[2] Department of Mechanical Engineering, University of Calgary, Calgary, AB, Canada

Introduction

A survey of dynamic data structures and computational geometry algorithms for collision detection problem is presented in this paper. The performance of methods for predicting and resolving collisions in the system of moving objects in two and three dimensional space is investigated in terms of efficiency of construction and maintenance of the data structure, space requirements, flexibility and portability of the algorithm. This work is a part of a research project for simulation of dynamics of granular-type materials, which combines concepts from the fields of computational geometry, data structures and mechanics, and develops the computational framework for simulation of discrete dynamic systems. The software system is currently in the implementation phase.

Problem definition

A set of N moving disks is given in the plane (or spheres in 3D space). Each of the disks moves along a straight line trajectory with a constant velocity. The disks move in a space limited by boundaries represented by M straight line segments (or plane segments in 3D). We will assume that $M << N$. It is required to detect and handle collisions between disks and between disks and boundaries. All collisions are considered to be instantaneous and one-on-one only. Handling of a collision requires updating of the velocities of the disks participating in the collision and detecting new collisions for the updated disks.

The determination of the time of collision for two disks moving along straight-line trajectories requires the solution of a quadratic equation with real coefficients. If it does not have a root, or if all roots are negative, then the two disks will not collide. We will assume that the determination of the collision time for a pair of disks is implemented by a function CollisionTime.

The review of some existing approaches for the collision detection problem can be found in [Kamat 1993]. In this paper we suggest and perform the comparison analysis for the five following data structures and algorithms for collision detection: the dynamic Delaunay triangulation under power and Manhattan metrics; the regular spatial subdivision; the regular spatial tree and a set of segment trees.

Straight-forward algorithm

This is the simplest approach to solve the problem. At the start of the simulation, all collisions are detected by checking each pair of disks for collision. This requires $O(N^2)$ CollisionTime calls. After each velocity update (due to a collision) the disk participating in the collision is checked for collision with each of the other disks. This requires $O(N)$ CollisionTime calls. There is no time or space overhead associated with the algorithm. However, for a large system the application of this method is computationally expensive. In the following algorithms we will try to decrease the required number of CollisionTime calls by introducing various geometric data structures.

Dynamic Delaunay triangulation

Dynamic Delaunay triangulation can be used for optimization of collision detection. We consider Delaunay triangulations in two metrics: power metrics and Manhattan metrics (L_∞ or L_1). When the Delaunay triangulation is maintained for the system of moving disks, it is possible to only check collisions between those disks, which are neighbors in the Delaunay triangulation. At the time of collision the distance between two disks becomes equal to zero (in any metric), and therefore the disks constitute a nearest-neighbor pair at this moment of time. Hence, there must be an edge in the Delaunay triangulation connecting these two disks, i.e. the collision will not be missed. Each time a new edge appears in the Delaunay triangulation, the pair of disks connected by the edge must be checked for a collision.

The dynamic maintenance of Delaunay triangulation of moving objects has been considered in [Roos 1991]. The algorithm is based on the fact, that the topological structure of Delaunay triangulation changes only at specific moments of time (called topological events). A topological event occurs when four sites become cocircular, according to the metrics being used. Handling of a topological event requires swapping of the diagonal in a quadrilateral consisting of two neighboring Delaunay triangles and scheduling the new topological events for the new quadrilaterals. This takes $O(\log N)$ since the events are stored in a priority queue.

Determination of the time of topological event requires finding the minimal root t_0 of the equation $INCIRCLE(P_1, P_2, P_3, P_4) = 0$, where $P_i = P_i(t), i = 1..4$ are the coordinates of moving sites. The form of the *INCIRCLE* function depends on the metrics being used. For the power metrics we calculate function as a 4x4 determinant [Gavrilova & Rokne 1995]. For the straight-line disk trajectories this requires the solution of a 4^{th} order polynomial equation, which usually can not be performed analytically.

In the Manhattan metric the solution of the equation is implemented by a set of conditional statements with linear formulas. We introduce the concepts of *internal* and *external* SWAPs and compute the time of these events using linear formulas of bodies coordinates and velocities. Hence, this approach shows much better performance than the one using power metrics.

The initial construction of the Delaunay triangulation can be performed in $O(N \log N)$ time. In practice, we develop and implemented two algorithms for this task: incremental construction (similar to the algorithm presented in [Guibas et al. 1992]), and the sweep-line algorithm, based on the original idea suggested by [Fortune 1987]. The expected running time for both algorithms is $O(N)$.

We can not bound the number of collisions between disks during the simulation because we do not limit the simulation time and because the disks move in a bounded space (i.e. unless the disks stop moving, the collisions must keep occurring). Hence, we will estimate the overhead associated with the use of a particular algorithm by the maximum number of topological events which can occur before a collision happens. This number is can be as large as $O(N^3)$ according to [Guibas et. al. 1991]. However, for a densely packed disk systems, this number can be as low as $O(1)$.

When a topological event happens, we spend $O(\log N)$ time on scheduling new topological events, and one CollisionTime calculation is performed. The space required to store the required data structures for the dynamic Delaunay triangulation maintenance is $O(N)$.

Regular spatial subdivision

The proposed approach is based on the partitioning of the whole simulation space into subdomains. In the simplest form, the space is subdivided into axis-parallel rectangles in 2D or cubes in 3D (see Figure 1). These are generically called boxes in the sequel. At any particular moment of time, each disk knows in which of the boxes it resides. Then, it is possible to locate regions of possible collisions only with disks residing in the neighboring boxes of the map. When the center of a disk moves from one box to another (we will call this situation a topological event), then the disks in the new neighboring boxes must be checked for collisions.

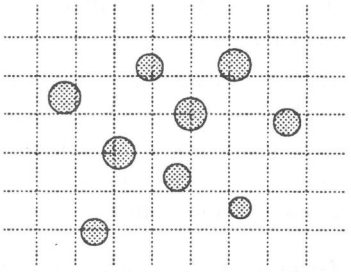


Figure 1. Regular subdivision

Each of the boxes can contain only a constant number of disks, when the size of the boxes is comparable to the size of disks. Hence, only a constant number of CollisionTime checks is required for each topological event.

Let us assume that the size of the map is such that there are K boxes in each direction. The value of K can be determined as the diameter of the simulation space divided by the maximum diameter of disk. Then the space requirement for the algorithm is $O(K^2)$ (or $O(K^3)$ for 3D). Each box must contain a list of disks, which currently reside in this box (the length of list can be bounded by a constant). A topological event

requires a constant number of operations in order to update the data structure: changing the box coordinates of the disk, and moving a disk from one sector to another. However, it is also required to schedule the next "box change" event for the disk. This requires $O(\log N)$ operations.

Since a disk can change only $O(K)$ boxes without collisions (because it must eventually collide with a boundary), the maximum number of topological events before a collision happens is $O(NK)$. For a densely packed system, this number is usually just a constant. The data structure is initialized and constructed in $O(N)+O(K^2)$ time.

The efficiency of this data structure is strongly dependent on the distribution of the radii of disks. We require that the size of a sector must be larger than the diameter of the largest disk. If a system contains one large disk and many small disks, then it is possible that a sector contains all of the disks, and the efficiency of the collision detection is no better than that of the straight-forward algorithm (when each pair of disks must be checked for a collision).

However, when all of the disks in the system have approximately the same size, then the maximum number of disks in a sector is bounded by a constant, and each topological event requires only a constant number of CollisionTime checks.

Regular spatial tree

It is possible to improve the space requirement of the previous approach by storing only those boxes which contain disks. For this purpose we will use a 2-level tree, where on the first level the boxes are sorted by the first coordinate, and on the second level boxes with the same first coordinate are sorted by the second coordinate. Then the total space requirement is $O(N)$, since the number of boxes stored can not exceed the total number of disks in the system. However, each sector access now will require $O(\log N)$ time compared to $O(1)$ direct array access in the previous approach. Hence, each topological event requires $O(\log N)$ operations on the tree. The initial construction of the data structure requires $O(N \log N)$ time.

Set of segment trees

In this approach, we use a set of sorted segment trees for optimization of the collision detection. Each of the disks is projected onto a segment on each of the coordinate axis (see Figure 2). The sets of segments are maintained in a sorted order during the motion of the disks. A topological event defined as a situation, when two segments on one of the axis start intersecting with each other. When a topological event happens, the CollisionTime is calculated if and only if the bounding squares of two disks intersect.

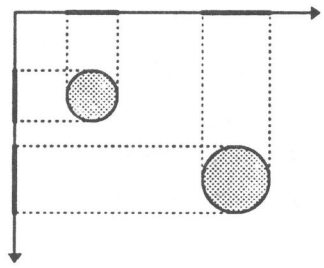


Figure 2. Segment trees

When a topological event happens, new topological events must be scheduled to maintain the segment trees in sorted order. This requires $O(\log N)$ time. The segment trees are constructed initially in $O(N \log N)$ time. The space requirement is $O(N)$. The maximum number of topological events before a collision happens is $O(N^2)$.

Performance analysis

The performance of the described algorithms is presented in the following table:

Algorithm	Checks/TE	Time/TE	TE/collision	Initialization	Space
Straight-forward	$O(N)$	-	-	-	-
Dynamic DT	1	$O(\log N)$	$O(N^3)$	$O(N \log N)$	$O(N)$
Spatial subdivision	$O(1)$	$O(\log N)$	$O(NK)$	$O(N)+O(K^2)$	$O(K^2)$
Subdivision tree	$O(1)$	$O(\log N)$	$O(NK)$	$O(N \log N)$	$O(N)$
Segment trees	0 or 1	$O(\log N)$	$O(N^2)$	$O(N \log N)$	$O(N)$

The CollisionTime call is more computationally expensive than operations on queues, because it involves floating-point arithmetic. Therefore, the minimization of the number of CollisionTime calls is the primary concern for selecting the best algorithm. As it can be seen from the table, the last two data structures match or outperform the other suggested algorithms. When the system is not very densely packed, i.e. the sizes of disks are small compared to the size of the simulation space, the segment tree approach outperforms the subdivision tree, which is based on the partitioning concept, since under these conditions the size of the map K has to be much larger than N.

Application of the algorithms to the dynamics simulation project

Two of the described structures: the subdivision tree and the segment trees are implemented as a part of the project for simulation of dynamics of granular-type material system. The simulated system is considered as a dynamic multibody system. For simplicity, each body is represented by a disk in a planar model or a sphere in 3D model. The motion of bodies is described by a system of ordinary differential equations [Vinogradov 1993]. The topology of the system is represented by a set of *clusters* (a number of objects, which move in contact with each other). An object-oriented dynamic data structure is described in [Sun et. al. 1994]. The data management problem, associated with the need to identify the topology after any change in the geometrical arrangement and to predict the time of the next topology change, is resolved by applying the described above the subdivision tree and the segment trees data structures.

Open problems and future directions

Currently, the fact that the bodies are connected into clusters is not used for optimization of collision detection. Collisions can occur either between bodies in different clusters, or between bodies which belong to the same cluster. These two different types of situations can be handled by different algorithms. The collision detection between clusters may require the application of one of the methods for collision detection for two randomly shaped bodies [Ganter & Isarankura 1993], since clusters may have a complex shape.

References

Fortine, S. 1987. "A sweepline algorithm for Voronoi diagrams," Algorithmica, Vol. 2, pp.153-174.

Ganter, M.A. and Isarankura, B.P. 1993. "Dynamic collision detection using space partitioning." Journal of Mechanical Design, Vol. 115, No. 1, pp. 150-155.

Gavrilova, M. and Rokne, J. 1995. "SWAP conditions for dynamic Voronoi diagrams for circles and line segments", submitted.

Guibas, L., Knuth, D. E. and Sharir, M. 1992. "Randomized incremental construction of Delaunay and Voronoi diagrams," Algorithmica, Vol. 7, pp.381-413.

Guibas, L., Mitchell, J.S.B. and Roos, T. 1991. "Voronoi diagrams of moving points in the plane," Lecture Notes in Computer Science, No. 570, pp.113-125.

Kamat, V.V. 1993. "Survey of techniques for simulation of dynamic collision detection and response." Computer Graphics, Vol. 17, No. 4, pp.379-385.

Roos, T. 1991. *Dynamic Voronoi Diagrams*. Ph.D. Thesis, Bayerische Julius-Maximilians-University, Würzburg, Germany.

Sun, Y., Vinogradov, O., Gavrilova, M. and Rokne, J. 1994. "An algorithm of updating system state in simulation of dynamics of granular-type materials." In *Proceedings of 1994 Summer Computer Simulation Conference*, 45-50.

Vinogradov, O.G. 1993. "Explicit equations of motion of interacting spherical particles." Recent Advances in Structural Mechanics, PVP - Vol. 248, pp.111-115.

An Efficient Competitive Strategy for Learning a Polygon (Short Version)

Frank Hoffmann* Christian Icking** Rolf Klein** Klaus Kriegel*

1 Introduction

A basic task of autonomous mobile robots is to explore an unknown environment, i.e. to walk around until each point of the environment has been visible at least once.

In our model, the unknown environment is a simple polygon, P, the edges of which are considered as opaque walls. The robot is a point, initially situated on a polygon vertex x. Through its vision system, it gets, at each time, the visibility polygon with respect to its current standpoint. Starting and ending at x, its task is to move in such a way that each point of the polygon has been visible at some time.

For this *online* problem, a strategy for moving on such a closed path within an unknown polygon is said to be *competitive*, if for any polygon its path length does never exceed a constant factor times the optimal watchman route.

Though one might think that the corresponding *offline* problem, i.e. computing the optimal watchman route, is difficult, it has been shown [1] to be solvable in time $O(n^4)$, n being the number of polygon vertices, which was later improved [5] to $O(n^2)$.

For the *online* problem, a paper [2] has appeared in FOCS '91 which claimed a competitive strategy with factor 2016. Unfortunately, the paper contains only some very vague ideas (which we do not use here) and no proof has appeared since then except for the rectilinear case [3] where a factor of $\sqrt{2}$ is easily achievable. In our opinion, the non-rectilinear case has been open until now.

2 Our approach

In this paper, we present a new strategy with a proven factor of 136. For brevity, only some ideas of the strategy are discussed without going into the details.

For seeing all points of the polygon, a path is sufficient which can see all edges which are adjacent to reflex vertices. So while moving in an unknown polygon, we only concentrate on reflex vertices and distinguish the *explored* and the *unexplored* vertices, i.e. those of which we have already seen the two adjacent edges, and those where at least one edge is still hidden.

There is a result [4] which says that even for a polygon with only one reflex vertex, walking directly to this vertex is not competitive. Instead, we have to approach it on a curved path, in our case we may chose a halfcircle (between the current position and the vertex) until the vertex is explored.

A first idea for exploring all vertices of an unknown polygon could be to attack all unexplored vertices in this way, one after the other, in the order as they appear on the boundary. This does not lead to a competitive factor, even if we happened to walk directly to the respective closest points on the prolongation of the invisible edges, as Figure 1 indicates. Here, the problem is that different kinds of reflex vertices force the path to oscillate from left to right, which is inefficient. But this will not happen, if we group these vertices accordingly.

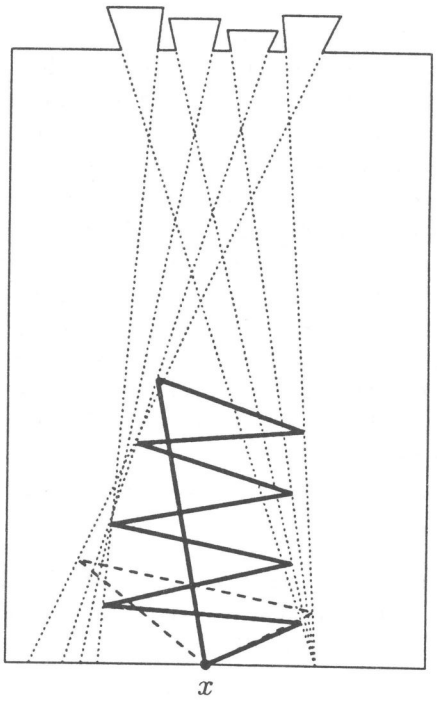


Figure 1: A naive strategy which makes a large detour, the dashed line indicates the optimal watchman route.

A better idea is the following. We label a reflex vertex *left* resp. *right* if, at that vertex, the shortest path from x to some other vertex makes a left resp. right

*Freie Universität Berlin, Institut für Informatik, Takustr. 9, D-14195 Berlin.

**FernUniversität Hagen, Praktische Informatik VI, Elberfelder Str. 95, D-58084 Hagen.

email: Hoffmann@Inf.FU-Berlin.de, Icking@FernUni-Hagen.de, Klein@FernUni-Hagen.de, Kriegel@Inf.FU-Berlin.de

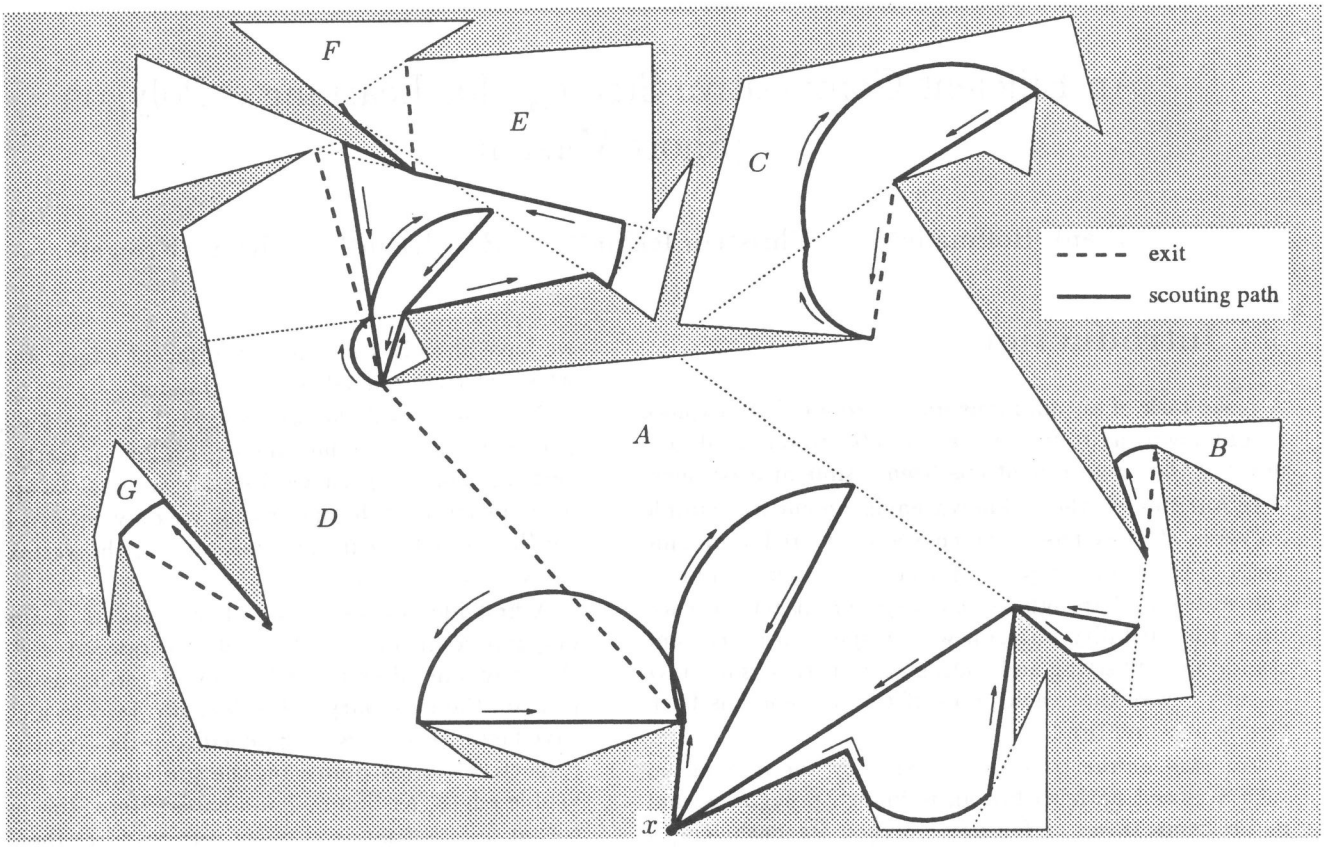


Figure 2: The polygon is partinioned into rooms $A, \ldots, G$, each room is treated separately.

turn. We want to explore the left vertices first, then the right vertices. This is not possible for the whole polygon, since right vertices may well be hidden behind left ones, etc., but it works if we partition the polygon into elementary parts, the so-called *rooms*.

Rooms are certain subpolygons of P, each room has an entrance and some exits. Entrances and exits are certain chords between a left and a right vertex such that the shortest watchman route is guaranteed to visit one of them. Every exit is at the same time the entrance of a next room, so the rooms are structured like a tree, rooted at the room which contains x, see Figure 2 for an example. For brevity, we will not give the exact definition here.

Now, our strategy roughly proceeds as follows, using two basic procedures.

Scout: On entering a room, a halfcircle strategy is used to find all exits, two such phases are necessary for the left and right vertices of the room.

Pass: After recognizing all exits, these are visited recursively in the order they appear on the boundary.

For the proof of the competitive factor, we remark that the scouting phase is competitive with factor $c = (\pi + 2)^2$ against the local shortest watchman route inside the room, and that scouting and passing from one room to another, all in all, do not use more than $5c + 3 < 136$ times the length of the shortest watchman route.

References

[1] W.-P. Chin and S. Ntafos. Watchman routes in simple polygons. *Discrete Comput. Geom.*, 6(1):9–31, 1991.

[2] 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.

[3] 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.

[4] 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.

[5] X. Tan and T. Hirata. Constructing shortest watchman routes by divide-and-conquer. In *Proc. 4th Annu. Internat. Sympos. Algorithms Comput. (ISAAC 93)*, volume 762 of *Lecture Notes in Computer Science*, pages 68–77. Springer-Verlag, 1993.

Optimization Problems Related to Zigzag Pocket Machining (Abstract*)

Esther M. Arkin[(1)] † Martin Held[(2)] ‡ Christopher L. Smith[(1)] §

[(1)]Department of Applied Mathematics and Statistics
State University of New York, Stony Brook, NY 11794-3600, USA
[(2)]Institut für Computerwissenschaften
Universität Salzburg, A-5020 Salzburg, Austria

A standard task in the production cycle of mechanical parts is to manufacture them from a billet or from a forging by removing material within given boundaries. Depending on whether or not the machining (milling) takes place in a plane (normal to a coordinate axis), one distinguishes between 2D/$2\frac{1}{2}$D machining and 3D/5D machining. Here the dimension[1] 'D' refers to the number of axes of a milling machine which need to be driven simultaneously in order to accomplish a machining task. The particular task of $2\frac{1}{2}$D milling came to be known as *pocket machining*. The ultimate goal of pocket machining is the generation of correct and somehow optimal tool paths in an automated manner, thus reducing the amount of human interaction required.

Basically, there exist two main types of tool-path topologies according to which tool paths for (pocket) machining are generated in practice:

- contour-parallel machining (see Fig. 1a),
- direction-parallel machining (see Fig. 1b).

Contour-parallel machining uses offset segments (of the boundary elements of the pocket) as tool path segments. Thus, the pocket is machined in a spiral-like fashion by driving the

*An extended abstract of this paper will be published at SODA'96; a (preliminary) full version is available on the WWW at http://www.cosy.sbg.ac.at/~held/papers/soda96.ps.Z.

†Partially supported by NSF Grant CCR-9204585. Part of this work was done while the author participated in the Special Semester on Computational and Combinatorial Geometry at Tel-Aviv University. E-mail: estie@ams.sunysb.edu.

‡Partially supported by grants from Boeing Computer Services, and by NSF Grants DMS-9312098 and CCR-9504192. This work was carried out while this author was with the AMS Dept. of SUNY Stony Brook. E-mail: held@cosy.sbg.ac.at.

§Supported by NSF Grant CCR-9204585. E-Mail: smithc@ams.sunysb.edu.

[1]The '$\frac{1}{2}$' in the term $2\frac{1}{2}$D emphasizes that only two axes, the x- and y-axis, are used for machining whereas the third axis, the z-axis, is only used for tool retractions, i.e. for lifting the tool for non-cutting movements.

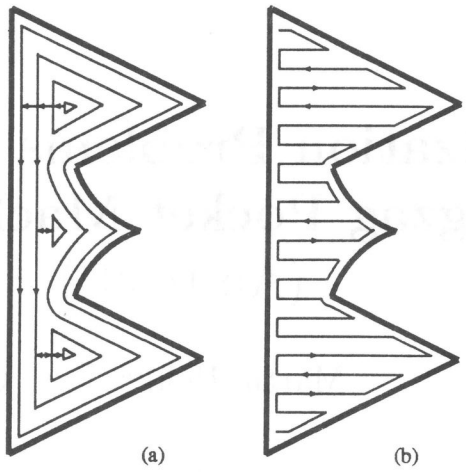


Figure 1: (a) Contour-parallel and (b) direction-parallel machining

tool(s) along curves that are at constant distances from the pocket's boundary. With direction-parallel machining, the tool(s) are moved along line segments which are parallel to a reference line selected initially. Based on this strategy a connected tool path[2] is obtained by linking these parallel segments such that they are either all traversed from right to left (resp., from left to right) or, as depicted in Fig. 1b, alternatingly from left to right and from right to left. The latter instance of direction-parallel machining also came to be known as 'zigzag' machining. Which type of tool-path topology is applied in practice highly depends on the NC machine used and on the particular machining task to be performed.

The common goal of all machining applications is to generate a tool path which is valid – i.e., which does not over-cut (gouge) or under-cut the part to be machined – and which yields the desired quality (roughness) of the machined surface, and which is efficient. What exactly constitutes an 'efficient path' yielding the desired surface quality is hard to quantify since it depends on several aspects such as the particular application, the material to be machined, the machine tool, etc. However, there exist a few generally accepted requirements which a good and efficient tool path has to fulfill:

1. No regions of the part should be machined repeatedly because repeated machining increases machining time, and also may decrease the surface quality.

2. Excessive numbers of tool retractions have to be avoided. Tool retractions may be necessary because (flat-ended) milling tools cannot simply plunge down onto the material but rather need to have holes pre-drilled at the starts of their machining paths, or otherwise require a careful handling of re-starts. However, the drilling of holes may largely extend the manufacturing time both directly (due to the actual time elapsed during drilling) as well as indirectly (due to time, costs, and more complex logistics caused by the set-up for another production-cycle).

In this paper we present a general algorithm for zigzag pocket machining which guarantees that no parts of the pocket are re-machined, thereby attempting to minimize the number of

[2]In addition, the tool may be moved along the pocket boundary once.

tool retractions needed. (It is assumed that the tool radius together with a possible inclination of the zigzag lines are pre-specified, and that a black box for offsetting the pocket boundary by the tool radius is available.) Our algorithm can handle arbitrary, multiply-connected pockets.

We transform the original pocket-machining problem into a graph-theoretic problem – of traversing a so-called 'machining graph' under certain restrictions – which is then analyzed and solved. To our knowledge, this is the first approach that does not purely rely on heuristics which hopefully yield a tool path with a small number of tool retractions, but rather applies an algorithm which is provably good, i.e., where constant-factor bounds on the maximum number of tool retractions compared to the (unknown) minimum number of retractions for any particular pocket are known. We currently achieve paths with at most $5 \cdot OPT$ retractions for a simply-connected pocket, where OPT is the minimum number of retractions required by any algorithm, and with $5 \cdot OPT + 6 \cdot h$ retractions for a multiply-connected pocket with h holes.

It is fairly easy to transform our machining graph into another graph such that a solution to the TRAVELING-SALESMAN PROBLEM (TSP) on this graph yields an optimal solution for our problem. (Of course, this transformation does not yield a practical algorithm because the computation of TSP-paths is $\mathcal{NP}$-hard.) We also show that the well-known PLANAR 3-SATISFIABILITY PROBLEM can be reduced to traversing a machining graph corresponding to a multiply-connected pocket such that a minimum number of retractions is used, thus establishing the $\mathcal{NP}$-hardness of our problem for a pocket with holes.

We implemented our algorithm, enhanced by some heuristics, and ran a few practical experiments. We compared our results to simple lower bounds on the number of tool retractions. For some small examples, for which we could afford to wait for the exact solution of the corresponding TSPs by a TSP-package, we also compared our results to the optimum (minimum) numbers of retractions. In both cases, our experiments clearly highlighted the fact that the approximation factors proved are pessimistic. In practice, our algorithm's performance is about $1.5 \cdot OPT$ retractions when applied to simply-connected pockets.

Computing the Minkowski sum of monotone polygons

Antonio Hernández Barrera*

Abstract

This paper presents algorithms for computing the Minkowski sum of two polygons P and Q for a family of problems. For P being convex and Q being monotone, an algorithm is given with $O(nm)$ time and space complexity. For both P and Q being monotone polygons, an $O(nm \log nm)$ time algorithm is presented and it is shown that the complexity of the sum is $\theta(nm\alpha(\min(n,m)))$ in the worst case, where $\alpha(\cdot)$ is the inverse of Ackermann's function. Finally, an $O((nm+k)\log nm)$ algorithm is given when P is monotone and Q is simple, where k in the worst case could be $\theta(n^2m)$. Here, m and n denote the number of edges of P and Q, respectively.

1 Introduction

Let A and B be two arbitrary sets in $\Re^d$ space. The Minkowski addition of A and B, denoted by $A \oplus B$, is defined as the set $\{a+b \mid a \epsilon A, b \epsilon B\}$, where "+"means vector sum.

A number of researchers ([1],[2], [6], [7], [8], [12]), have studied the efficient computation of the Minkowski operations (addition and decomposition) as well as the applications of these operations which include, among others, the polygon containment and the motion planning problems. For example, an algorithm is given in [8] to solve this problem: given a convex polygon B, of relatively simple shape, free to translate, but not to rotate, in a two dimensional open region bounded by a collection of convex obstacles with n corners in total, and an initial and final configurations of B, determine whether there exists a continuous obstacle-avoiding motion of B between the two given configurations, and if so, plan such a motion. The algorithm runs in time $O(n \log^2 n)$ (later the problem was solved using $O(n \log n)$, [9]) and is based on a property of the union of certain Minkowski sums of convex 2-dimensional objects. Also, in [6] the problem of computing sums of polygons that are path-connected, bounded and regular in $\Re^2$ is considered, and an $O(m^2n^2(m+n)\log(m+n)$ worst case complexity algorithm is given, where the input consists of $n+m$ edges.

Here, we present algorithms for computing the Minkowski sum of two polygons P and Q with m and n edges, respectively, for a family of problems. Specifically, we deal with the cases where (1) P is convex and Q is monotone, (2) both polygons are monotone and (3) P is monotone and Q is simple. We propose an $O(nm)$ worst-case optimal algorithm in the first case, an $O(nm \log nm)$ in the second and an $O((nm+k)\log nm)$, where k is $O(n^2m)$, in the last one.

2 Preliminaries

A *chain* $C = (u_1, u_2, ..., u_n)$ is a planar straight-line graph with the vertex set $\{u_1, u_2, ..., u_n\}$ and the edge set $\{(u_i, u_{i+1}), i = 1, ..n-1\}$. C is said to be *monotone* with respect to a straight line ℓ if any line orthogonal to ℓ intersects C at no more than one point. A simple polygon, ie. with no holes and nonintersecting edges, is considered to be monotone if the boundary can be decomposed into two chains that are monotone with respect to the same line ℓ.

$T_u(P)$, $(P)_{\min x}$, $(P)_{\max x}$ and $bd(P)$ denote the translation by vector u, the x-coordinate of the leftmost vertex, the x-coordinate of the rightmost vertex and the boundary of polygon P, respectively.

3 The monotone-monotone case

Suppose we are given polygons P and Q, with n and m edges, respectively, where P and Q are both monotone with respect to some line ℓ. Our goal is to efficiently compute $P \oplus Q$. Without loss of generality we will consider ℓ to be the X axis and we will assume that 0 is an interior point of P.

We begin by giving some results which shall be used in the sequel:

Theorem 3.1 *Let P and Q be monotone polygons. $P \oplus Q$ is also a monotone polygon.*

*Department of Mathematics, Faculty of Science, Hiroshima University

Proof Let I_t^A be the intersection of polygon A and a vertical line at $x = t$. Note that a polygon R is monotone if and only if I_t^R is a connected interval for $t\epsilon[(R)_{\min x}, (R)_{\max x}]$. Take any s_0 in the interval $[(P \oplus Q)_{\min x}, (P \oplus Q)_{\max x}]$. Notice that $I_{s_0}^{P\oplus Q}$ is the union of connected intervals $I_t^P \oplus I_{s_0-t}^Q$, i.e. $I_{s_0}^{P\oplus Q} = \bigcup_{t\epsilon I} I_t^P \oplus I_{s_0-t}^Q$, where $I = [\max\{P_{\min x}, s_0 - Q_{\max x}\}, \min\{P_{\max x}, s_0 - Q_{\min x}\}]$. It is not difficult to see that $\bigcup_{t\epsilon I} I_t^P \oplus I_{s_0-t}^Q$ is a connected interval by a topological consideration. □

Proposition 3.2 *Let A and B be planar sets where B is the result of the union of the sets $B_1, ...B_k$. Then $A \oplus B = A \oplus (\bigcup_{i=1}^k B_i) = \bigcup_{i=1}^k (A \oplus B_i)$.* ([10])

Proposition 3.3 *If one polygon is convex with m edges and a second polygon is non-convex with n edges, the output size of the Minkowski sum has a tight asymptotic bound of $O(nm)$.* ([6])

With respect to Proposition 3.3, we want to point out that the bound is tight even if the non-convex polygon is restricted to be monotone as the reader may easily verify.

3.1 The convex-monotone case

We now proceed to consider the case where one of the polygons, say P, is convex and the other is monotone.

Proposition 3.2 allows us to design a simple divide and conquer algorithm to compute the Minkowski addition of P and Q. Consider the decomposition of Q into vertical trapezoids $Q_1, Q_2, ..., Q_k, k < n$, obtained by extending a vertical line segment from each vertex of Q until it reaches the opposite side of Q's boundary. From Proposition 3.2 we know that $P \oplus Q = \bigcup_{i=1}^k P \oplus Q_i$. It has been noted in [2] that the Minkowski sum of two convex polygons with r and s edges respectively, can be computed by merging their edges in slope order to obtain in this way an $O(r + s)$ time complexity algorithm which is optimal since the number of vertices of the sum set is, at most, $r + s$. We can compute $P \oplus Q$ by means of the following algorithm, which we adapt from [8]: Partition the set $\{Q_i \mid 1 \leq i \leq k\}$ into two families $Q\prime$ and $Q\prime\prime$ of roughly $\frac{k}{2}$ trapezoids each. Recursively calculate $P \oplus Q\prime$ and $P \oplus Q\prime\prime$. Then compute the union of $P \oplus Q\prime$ and $P \oplus Q\prime\prime$. The splitting process stops whenever $Q = Q_i$ for some $i, 1 \leq i \leq k$, where an algorithm for computing the Minkowski sum of two convex polygons is applied.

It is not difficult to prove that the complexity of this squeme is $O(nm \log n \log nm)$. However, with it we are not making full use of the particular characteristics of our input. We will show that a worst-case optimal time complexity algorithm can be obtained.

The algorithm proceeds incrementally, that is, the Minkowski sum is built by adding $P \oplus Q_i$ polygons, $1 \leq i \leq k$, one at a time to an already constructed sum. The $P \oplus Q_i$ are added following the left right order of the Q_i, i.e. first $P \oplus Q_1$ is considered, after $P \oplus Q_2$ is added, and so on.

Suppose $\bigcup_{i=1}^{j-1} P \oplus Q_i$ has been constructed after the $j - 1$th step has been performed. $\bigcup_{i=1}^{j} P \oplus Q_i$ is a monotone polygon (Theorem 3.1) thus its boundary consists of a lower chain and an upper chain. We will now describe how the lower one is constructed. A symmetric procedure can be applied to obtain the upper one. Let $\mathcal{L}$ and $\mathcal{L}_j$ be the lower chain of $\bigcup_{i=1}^{j-1} P \oplus Q_i$ and $P \oplus Q_j$, respectively. It is not difficult to see that the lower chain of $\bigcup_{i=1}^{j} P \oplus Q_i$ can be computed in the j th step by simply computing the lower envelope of $\mathcal{L} \cup \mathcal{L}_j$. The point here is the complexity of the envelope. We claim that the number of intersections between $\mathcal{L}$ and $\mathcal{L}_j$ is exactly one. By an intersection we mean "crossing", i.e. the point where the boundary of one of the chains "disappears" locally into the other set, since the "touching" case is not relevant to the complexity analysis.

Lemma 3.4 *The number of intersections of $\mathcal{L}$ and $\mathcal{L}_j$ is exactly one.*

Proof A rapid analysis of the possible interactions between the boundaries of $\bigcup_{i=1}^{j-1} P \oplus Q_i$ and $P \oplus Q_j$ shows us that $\mathcal{L}$ and $\mathcal{L}_j$ must intersect. To prove that there is at most one intersection, consider a vertical line which is swept from right to left over the two chains. Let t be the first intersection of $\mathcal{L}$ and $\mathcal{L}_j$ found by the sweep line. Notice that t must belong to the lower chain $\mathcal{L}_i$ of some $P \oplus Q_i$, $1 \leq i \leq k$. We classify the intersections being of *first type*, when $\mathcal{L}_j$ crosses $\mathcal{L}_i$ to enter $P \oplus Q_i$ and of *second type*, when $\mathcal{L}_j$ crosses $\mathcal{L}_i$ to abandon $P \oplus Q_i$. Clearly, t is a first type intersection. The intersections between $\mathcal{L}_j$ and $\mathcal{L}_i$ must occur on $\mathcal{L}_i$ ($\mathcal{L}_j$) along the subchain formed by, clockwise, the rightmost (leftmost) vertical edge (possibly a point), the edges of $T_p(P)$ ($T_q(P)$) belonging to $\mathcal{L}_i$ ($\mathcal{L}_j$), and the portion lying to the right (left) of the vertical line at $(T_p(P))_{\min x}$ ($(T_q(P))_{\max x}$), where p (q) is the lower right (left) vertex of Q_i (Q_j). See Figure 1 (a).

Suppose that t occurs between the sections of $bd(T_p(P))$ and $bd(T_q(P))$ lying on $\mathcal{L}_i$ and $\mathcal{L}_j$, respectively. Let us first observe that the boundaries of a convex polygon and a translated copy of it cross in at most two points. For a second type intersection to exist, $\mathcal{L}_j$ should cross again the lower chain of

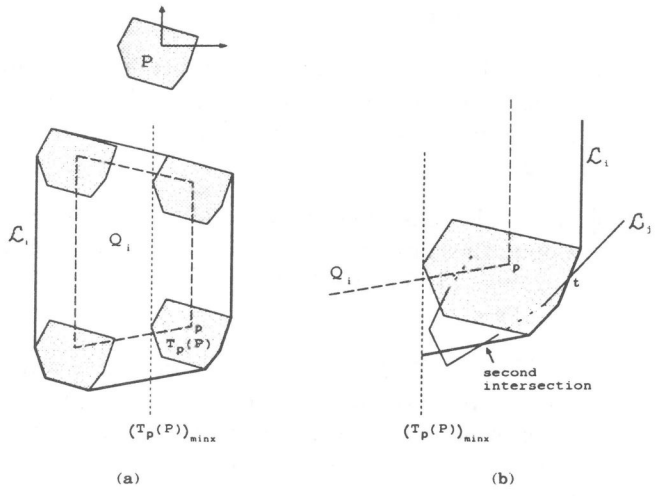


Figure 1: (a) The intersection on $\mathcal{L}_i$ must occur to the right of $(T_p(P))_{\min x}$. (b) A second intersection is not possible.

$T_p(P)$. But, notice that $(T_q(P))_{\min x}$ is to the right of $(T_p(P))_{\min x}$, thus $\mathcal{L}_j$ would have to cross for a third time the lower chain of $T_p(P)$. See Figure 1 (b). Thus, such a second type intersection cannot exist. Very similar or simpler arguments can be applied for the different cases of t, i.e. when t is an intersection of the section of $bd(T_p(P))$ lying on $\mathcal{L}_i$ and the leftmost vertical edge of $\mathcal{L}_j$, etc. to prove that such a second type intersection does not exist. Thus, no other intersection between $\mathcal{L}_j$ and $\mathcal{L}_i$ is found by the sweep line after it goes past t. Hence, no other intersection exists between $\mathcal{L}_j$ and $\mathcal{L}$. □

Therefore, we can compute the lower envelope of $\mathcal{L} \cup \mathcal{L}_j$ by traversing both chains from right to left until $\mathcal{L}_j$ hits $\mathcal{L}$. In this way, each vertex of $\mathcal{L}$ that is passed over will not be looked at during the examination of the next $P \oplus Q_j$'s. Thus, each vertex of $\bigcup_{i=1}^{n} P \oplus Q_i$, as well as each intersection point obtained in the process, is visited a constant number of times. In short,

Theorem 3.5 *Let P and Q be a convex and a monotone polygon with m and n edges, respectively. $P \oplus Q$ can be computed using $O(nm)$ time and space.*

3.2 The monotone-monotone case

We also use Proposition 3.2 to develop a simple algorithm in case both P and Q are monotone polygons. First, partition Q in linear time into vertical trapezoids $Q_1,...,Q_k$, $k < n$. Second, compute $P \oplus Q_i$, for $i = 1, 2, ..., k$. Due to Theorem 3.5, this step is performed in $O(nm)$ time and space. Finally, compute the union of monotone polygons $\bigcup_{i=1}^{k} P \oplus Q_i$. Let $\mathcal{L}_i$ be the lower chain of polygon $P \oplus Q_i$. From Theorem 3.1 $P \oplus Q$ is a monotone polygon, thus its lower chain can be obtained by computing the lower envelope of the set of $\mathcal{L}_i$, $i = 1, 2, ...k$. The upper chain is obtained similarly.

The remaining problem is to find the envelope of n chains, each with, at most, m edges. The problem of computing the envelope of N line segments has been studied by some researchers and these results are well known: the envelope may consist of $\theta(N\alpha(N))$ edges, where $\alpha(N)$ is the inverse of Ackermann's function ([4], [13]), and it can be computed in optimal $O(N \log N)$ time [5]. Then, as we have $O(nm)$ line segments in the chains, the lower envelope can be obtained in $O(nm \log nm)$ time by the algorithm in [5].

We may ask ourselves if the complexity of the envelope (and hence $P \oplus Q$'s) could be $\Omega(nm\alpha(nm))$, since we are not dealing with $O(nm)$ line segments in general position, but they are the edges of the lower chains of the monotone polygons $P \oplus Q_i$. Recently, it was proven in [3] that the number of edges and vertices bounding a single face of the complement of the Minkowski sum $A \oplus B$ is $\theta(ab\alpha(a))$ where a and b are the number of vertices of the polygonal sets A and B, respectively, and $a < b$. Since $P \oplus Q$ is monotone, only one face exists, the outer one, therefore its complexity is $O(nm\alpha(\min(n, m)))$. The polygons constructed in [3] to show that a single face could have $\Omega(ab\alpha(a))$ edges are, precisely, monotone. Thus, in the worst case $P \oplus Q$ complexity is $\theta(nm\alpha(\min(n, m)))$. In summary,

Theorem 3.6 *Let P and Q be two monotone polygons, with m and n edges respectively. The number of edges and vertices bounding the Minkowski sum $P \oplus Q$ is $\theta(nm\alpha(\min(n, m)))$ in the worst case and it can be computed in $O(nm \log nm)$ time.*

4 The monotone-simple case

Let P and Q be monotone and simple polygons, respectively. As in the previous sections, an algorithm for computing $P \oplus Q$ is obtained by partitioning Q into simpler parts, say, convex or monotone, and computing the union of the partial Minkowski sums. The contour of the union of a set of polygons can be obtained in $O((N + k) \log N)$ time, where N is the number of edges and k is the number of intersections between the edges [11]. In our case, the number of edge intersections is easily seen to be $O(n^2 m)$, hence the complexity of $P \oplus Q$ is $O(n^2 m)$. The bound is tight

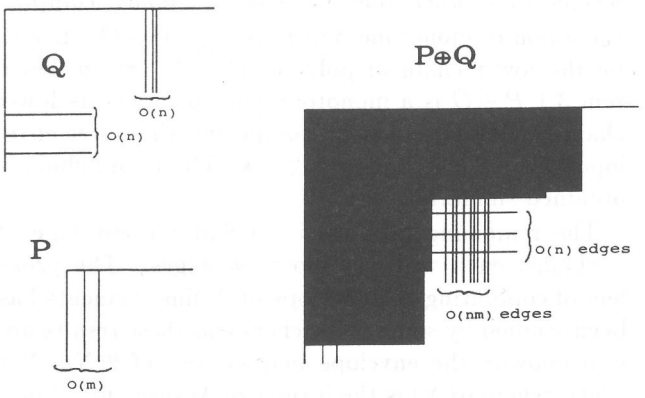


Figure 2: A worst case with $O(n^2m)$ edges

in the worst case. See Figure 2 for an example with $\Omega(n^2m)$ edges. Thus,

Theorem 4.1 *Let P and Q be monotone and simple polygons with m and n edges, respectively. $P \dotplus Q$ can be computed in $O((nm+k)\log nm)$ time, where k is $O(n^2m)$.*

5 Conclusions

We have proposed algorithms for computing the Minkowski sum of two polygons in a variety of cases involving monotone polygons. A natural direction for extension of these results is to consider the generalization to the three dimensional space.

References

[1] P.K. Ghosh. A solution of polygon containment, spatial planning, and other related problems using Minkowski operations. *Computer Vision, Graphics, and Image Processing*, **49** (1990), pp. 1–35.

[2] L. Guibas, L. Ramshaw, and J. Stolfi. A kinetic framework for computational geometry. In *Proc. 24th Annual IEEE Symposium on Foundations of Computer Science*, pp. 100–111, 1983.

[3] S. Har-Peled, T. M. Chan, B. Aronov, D. Halperin, and J. Snoeyink. The complexity of a single face of a Minkowski sum. In *Proc. 7th Canadian Conf. Comp. Geom.*, 1995.

[4] H. Hart and M. Sharir. Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes. *Combinatorica*. **6** (1986), pp. 151–177.

[5] J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Information Processing Letters*, Vol. 33, No. 4, pp. 169–174, 1989.

[6] A. Kaul, M.A. O'Connor, and V. Srinivasan. Computing Minkowski sums of regular polygons. In *Proc. 3rd Canada Conf. Comp. Geom.*, pp. 74–77, 1991.

[7] K. Kedem, R. Livne, J. Pach, and M. Sharir. On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete and Computational Geometry*, **1** (1986), pp. 59–71.

[8] K. Kedem and M. Sharir. An efficient algorithm for planning collision-free translational motion of a convex polygonal object in 2-dimensional space amidst polygonal obstacles. In *Proc. 1th ACM Annual Symposium on Computational Geometry*, pp. 75–80, 1985.

[9] D. Leven and M. Sharir. Planning a purely translational motion for a convex object in two-dimensional space using generalized Voronoi diagram. *Discrete and Computational Geometry*, **2** (1987), pp. 9–31.

[10] G. Matheron. *Random sets and integral Geometry*. Wiley, New York, 1975.

[11] T. Ottmann, P. Widmayer, and D. Wood. A fast algorithm for the boolean masking problem. *Computer Vision, Graphics, and Image Processing*. **30** (1985), pp. 249–268.

[12] M. Sharir. Efficient algorithms for planning purely translational collision-free motion in two and three dimensions. *IEEE International Conference on Robotics & Automation*, Vol. 3, pp. 1326–1331, 1987.

[13] A. Wiernik and M. Sharir. Planar realizations of nonlinear Davenport-Schinzel sequences by segments. *Discrete and Computational Geometry*, **3** (1988), pp. 15–47.

Morphing Fields of Directions defined on Triangulations to Morph Simple Polygons

Antonio Oliveira
Sara do Nascimento
Sonja A. L. Meerbaum
Programa de Engenharia de Sistemas
COPPE-Universidade Federal do Rio de Janeiro
CP:68511 - 21945-970 - Rio de Janeiro
e-mail: Oliveira@cos.ufrj.br

Consider the problem of continuously transforming a polygon L(0) into another L(1) with the same number of vertices (n) taking each vertex of L(0) into a corresponding one of L(1). We call a morphing of L(0) into L(1) to a solution of that problem wich in addition satisfies the two following conditions:
i) Only simple polygons are generated during the transformation. This, in particular means that creating loops or whiskers is not allowed.
ii) The shape of any intermediate polygon generated is a blend of L(0) and L(1) shapes. We do not dare to propose any objective measure of that blend quality. This condition is only to eliminate processes generating in between polygons whose shape is completely uncorrelated with those of L(0) and L(1).

To conciliate the two conditions above is far from being easy. On the one hand, all the best known Contour Shape Interpolation approaches used in Computer Graphics cannot guarantee the topology preservation condition. A list of those approaches include very simple instances as linear interpolating pairs of corresponding vertices or Turtle-Geometry descriptions of the polygons[SED93]. Other examples are the Minkowsky Sum Interpolation Method and related ones using Set Operators[ROS91] , Contour Morphing methods inspired in Active Contour or based on other Physical Models [SED92] and Skeleton transforming methods[WOL90].[1]
On the other hand, approaches that are successful in mantaining the topology during the transformation can generate polygons whose shape is much simpler than those of L(0) and L(1). An example that ilustrates well this point is given by the following approach. At first morph L(0) into its convex hull. Then, this convex hull into the one of L(1) and finally this last hull into L(1). In any of these three stages, topology preserving transformations can be obtained by methods requiring only O(n) pre-processing time and also O(n) time for constructing an intermediate polygon. Thus the time complexity of the approach is optimal.The blend quality of the morphing obtained, however, is clearly deficient. In the extremely interesting approach for morphing parallel polygons presented in [HER94] there is not exactly a simplification in shape but contractions of parts of the polygons, during the transformation.

In an attempt to get a better compromise between these two apparently conflicting objetives, we propose another methodology to the problem. We will refer to that methodology by Marionnettes Morphing(in short: MM) because the way the deformation process is controled in its most naive version reminds somewhat an operator controling the movement of a Marionnette puppet by pulling and loosing strings. MM can be laborious and has some other drawbacks commented in the article. Nevertheless, it has also some advantages and our option for it has been mainly formulated in function of the three arguments below:

1. Intermediate polygons generated by MM can, of course, be considerably simpler than the L(i)s. However, in general that simplification process is less drastic than those determined by other shemes to obtain topology preserving transformations. In consequence the morphings generated by MM seem more natural.

2. The methodology can be extended to get a morphing of a set of polygons into another if there is an

[1] Some of those methods have been originally proposed for raster versions of the problem and we are considering here crude adaptations of them to the continuous environment.

homeomorphism preserving orientations which takes each polygon in the first set into a corresponding one in the second[OLI95]. That extension however, can require an worky pre-processing. Indeed, this more general version of the problem is much more common in practice than the restricted one studied here. We observe that, countour morphing methods which only uses information relative to shape and dimensions and do not consider the two polygons imbedded in a common background cannot be extended to that case.

3.The user can furnish polygonal curves indicating in an approximate way , how some or all L(0) vertices have to be moved during the transformation. That information can be used not only to improve the blend quality but is particularly helpful when the contours to be morphed are line drawings representing an articulated object in motion . If a trajectory is not provided for a vertex(v_{j0}) of L(0), the method determines one in an initialization step. A common option is to make that trajectory be a polygonal approximation of a circle arc with extremities at (v_{j0}) and at the L(1) vertex corresponding to it.

In the strategy employed by MM the fundamental steps are finding for each L(i), i = 0,1; a field of directions transversal to it and continuously transforming one these fields into the other. Instead of fields of directions defined in R^2 or a subset we can use functions associating a direction to every triangle of a triangulation. We refer to that functions as Tfds. The Tfds that we use in this article are defined on triangulations of an annular region U whose borders(U_{in} and U_{out}) are polygonal approximations of circles centered in the same point s .The outer circle must contain the given polygons and the inner one must be contained in the interior of both of them. We will consider that adequately position a polygon in relationto the other and even choose s is either an user's attributtion or is done in a preparation step not discussed in the work.

Let D be a Tfd, defined on a triangulation T containing a triangle t. Consider the straight lines parallel to D(t) passing through the vertices of t. Only one of these straigth lines intersects t in more than one point. A vertex of t on that straight line will be said assigned to t by D. This vertex is uniquely defined if D(t) is not parallel to an edge of t. If v is assigned to t by D and $v+\lambda D(t)$, $\lambda > 0$ intersects t, we say that the sign of v in t is positive. Otherwise it is negative. See figure 1. An Assignment of D is a function associating each triangle t of T to a pair consisting of a vertex assigned to t by D and the sign of that vertex in t.

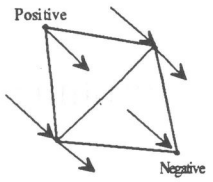


Fig. 1 - The Vertex-types of a triangle.

A Tfd is called admissible if any trajectory determined by it is disjoint to any other, start on U_{in} and end on U_{out}.See figure 2. An admissible Tfd D is said

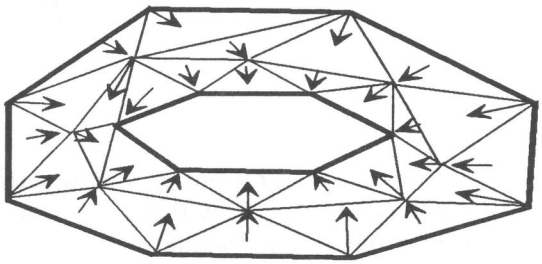


Fig. 2 - An Admissible Tfd.

to be a convexer for a set $S \subseteq U$, iff the intersection of any trajectory determined by D with S is connected. As $L(i) \supseteq U_{in}$, i=0,1; a convexer for it, is a Tfd whose trajectories cross the L(i) contour exactly once. See figure 3 below. A convexer for L(i) can be obtained in O(n) time.

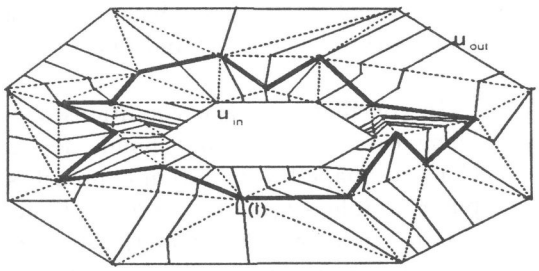


Fig. 3 - A convexer of the polygon L(i).

Given two triangles t_1 and t_2 of T, we say that $t_1 \ _D \ t_2$ iff some trajectory meets t_2 after t_1. D is said to be ordering if $_D$ is a partial order.
Given an admissible Tfd D, consider the system of reference where the coordinates of a point p in U are the distance from p to s along the D-trajectory through p and the point where this trajectory crosses U_{in}. Refer to this system of coordinates as the system induced by D.

Let D(i),i=0,1 be a convexer for L(i) and suppose we know how to obtain a continuous transformation: w ε [0,1] $\rightarrow$ D(w) of D(0) into D(1) which generates only

admissible Tfds in between. Call that transformation admissible. Also, let S(w) be the system of reference induced by D(w).Finally, define L(w) as the region encircled by the curve whose points have coordinates in S(w) which are the result of a continuous interpolation parametrized by w between the S(0)-coordinates of L(0) border points and the S(1)-coordinates of their corresponding points on L(1) border. w$\rightarrow$L(w) defines a topology preserving continuous deformation of L(0) into L(1). That is the essence of the approach of MM but applying it, in that raw version, may determine that intermediate polygons with $O(n^2)$ edges are constructed. To guarantee that L(w) is a simple polygon witn an O(n) number of edges, less straightforward schemes indicated in the article are required.

Let us concentrate on the problem of getting an admissible transformation of D(0) into D(1). First of all let us consider the possibility of any D(w) have closed trajectories. This cannot happen if both D(0) and D(1) are ordering because in this case it is possible to make that property hold for every D(w).
Now suppose for a while that D(0) and D(1) are defined on the same triangulation T.Refer by A(w) to an Assignment of D(w). If during an admissible transformation, for a triangle t ε T , $A(w^+) \neq A(w^-)$ then, supposing that D(·) is locally 1-1 at w, we have that:
i) D(w)(t) is parallel to an edge e=[v_1, v_2] of t.
ii) If t′ is the other triangle adjacent to e then D(w)(t) = D(w)(t ′), $A(w^+)(t) = A(w^-)(t') = (v_i, \text{sign})$ and $A(w^+)(t') = A(w^-)(t) = (v_{3-i}, -\text{sign})$, i = 1 or 2. See figure 4 below.
In view of ii, during an admissible transformation,the

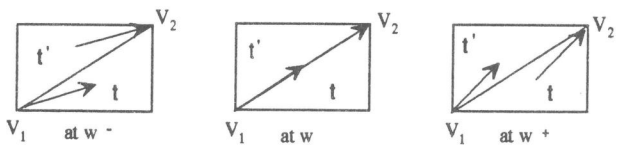


Fig. 4 - An Assignament change during an Admissible Transformation.

Assignment of a triangle can only change through an Assignment swap with an adjacent one. We prove that given two admissible Tfds defined on T, the assignment of one can be turned into the assignemnt of the other through a sequence of $O(n\lambda)$ admissible swaps where n is the size of T and λ is $\max_{t \epsilon T}$\{minimun number of triangles crossed by a path from t to the U border\}. The precise number of times any pair of adjacent triangles exchange their assignments during the transformation can be determined by the simple observation of D(0) and D(1).

Now, imagine that each D(i),i=0,1; is defined on a different triangulation T(i), both T(i)s having the same number of vertices. The graph associated to T(0) can be transformed into that of T(1) by means of a sequence of $O(n^2)$ quadrilateral diagonals swaps. However, these swaps cannot be made independently of the current Tfd definition. Let T(w) be the triangulation current at a moment w ε [0,1] and Q, a convex quadrilateral composed of triangles t_1 and t_2 belonging to T(w). A diagonal swap in Q can only be made if $D(w)(t_1) = D(w)(t_2)$.(1) This condition is necessary to assure the continuity of D(.) at w. It also guarantees that $D(w^+)$ will be well defined in the triangles composing Q after the swap. In the article we prove that condition 1 can always be satisfied after O(n) admissible assignments exchanges. Multiplying it by the worst case complexity of the diagonal swaps numnber we end up with an $O(n^3)$ bound for the number of assignments swaps.

In the last part of the article we tackle the problem of morphing two Tfds defined on the same 3D triangulation. Some preliminary results are presented and some specific problems which we consider open are formulated. 2D morphings obtained by MM are also presented.

.REFERENCES

[HER94] J.Hershberg and L.Guibas; Morphing Simple Polygons: Proc of the 10th ACM Symposium on Computational Geometry; 267-276,1994

[OLIV95] A.A.F. Oliveira; Morphing Polygonal Scenes(in preparation).

[ROS91] J.Rossignac and A.Kaul; Solid Interpolating Deformations: Construction and Animation of PIPs, Proc.Eurographics 91,493-505(1991).

[SED92] T.Sederberg and E.Greewood; A Physically Based Approach to Shape Blending; Computer Graphics 26(2), 25-34(1992).

[SED93] T.Sederberg, P. Gao, G. Wang, and H. Mu; 2-D Shape Blending: An Intrinsic Solution to the Vertex Path Problem; Computer Graphics Proceedings, 1993.

[WOL90] G.Wolberg; Digital Image Warping; Computer Society Press, 1990.

Fast Stabbing of Boxes in High Dimensions

Franck Nielsen*

December 11, 1995

We present in this talk a very simple and efficient algorithm for stabbing a set $\mathcal{S}$ of n axis-parallel boxes in d-dimensional space with c points in output-sensitive time $O(dn + n \log c)$ and linear space. Let c^* be the minimum number of points required to stab $\mathcal{S}$, then we can prove that $c \leq \min\{\frac{c^{*\overline{d}}}{d!} + \frac{c^{*\overline{d-1}}}{(d-1)!} - 1, c^* \frac{(\log n+1)^{\overline{d-1}}}{(d-1)!}\}$, where $x^{\overline{m}}$ is the rising factorial power: $x^{\overline{m}} = \prod_{i=0}^{m-1}(x+i)$. Since finding a minimal set of c^* points is NP-complete as soon as $d > 1$, we obtain a fast heuristic for stabbing $\mathcal{S}$ in output-sensitive time and linear space. Moreover, we show that the bounds we get on c are tight. This implies that we get surprisingly a polynomial dependence on d for small values on c^*. We also prove that our algorithm guarantees that $c \leq 2^{d-1}c^*$ ($c \leq O_d(c^*)$) when considering identical boxes (respectively for bounded aspect ratio boxes). We elaborate on the hardness of approximation of any algorithm.

This algorithm may be of practical interest in VLSI, image processing, facility location, point location in $\mathcal{E}^d$, etc. Finally, we corroborate our theoritical results with experiments and give further insights for future research.

*INRIA, BP93, 06902 Sophia-Antipolis cedex (France) E-mail: Franck.Nielsen@sophia.inria.fr — http://www.inria.fr:/personnel/nielsen/frank.html, and University of Nice Sophia-Antipolis, Valrose (France).

Sequential and Parallel Construction of $(1/r)$-Approximations

Petra Knieper * Anand Srivastav **

(Abstract, December 1995)

1 Introduction

One central concept in applications in computational geometry is the efficient computation of ϵ-nets and ϵ-approximations. Two different algorithms have resolved the problem of computing ϵ-approximations of size independent of the the set system in deterministic polynomial time: Matoušek's (1991) algorithm in case of polynomially bounded primal shatter function and the algorithm of Matoušek, Welzl and Wernisch (1991) for polynomially bounded dual shatter function. For bounded primal shatter function Chazelle and Matoušek (1993) simplified the proof of Matoušek and Brönnimann, Chazelle, Matoušek (1993) achieved a slightly better running time. Matoušek's algorithm is based on the computation of ϵ-approximations for arbitrary set systems and thus can be accelerated, if such basic computations can be done as quickly as possible.

In this paper we give the presently fastest algorithm for the computation of ϵ-approximations for arbitrary set systems, derive its parallel version and parallelize Matoušek's algorithm for set systems with polynomially bounded primal shatter function. We leave open the problem of finding a parallel counterpart of Matoušek, Welzl and Wernisch's algorithm for set systems with polynomially bounded dual shatter function.

2 Preliminaries

A range space is a pair $(X, \mathcal{R})$, where X is a set of points and $\mathcal{R}$ is a set of subsets of X. The elements of $\mathcal{R}$ are called ranges. We assume that X itself is a range, i.e. $X \in \mathcal{R}$. For $Y \subseteq X$ let $(Y, \mathcal{R}_Y)$ be the range space restricted to Y, where $\mathcal{R}_Y = \{R \cap Y; R \in \mathcal{R}\}$. A subset $A \subseteq X$ is called ϵ - **net** for $(X, \mathcal{R})$, if for all $R \in \mathcal{R}$ with $|R| > \epsilon|X|$, $A \cap R \neq \emptyset$.

Presently, the only way to compute ϵ-nets deterministically is via ϵ-approximations. ϵ-approximations are special ϵ-nets with nice properties not shared by arbitrary ϵ-nets: for example ϵ-approximations are stable under divide-and-conque arguments.

*Institut für Informatik; Humboldt Universität zu Berlin; Unter den Linden 6, 10099 Berlin, Germany; e-mail: kieper@informatik.hu-berlin.de

**Institut für Informatik; Humboldt Universität zu Berlin; Unter den Linden 6, 10099 Berlin, Germany; e-mail: srivasta@informatik.hu-berlin.de

A subset $A \subseteq X$ is called an ϵ - approximation for $(X, \mathcal{R})$, if for all $R \in \mathcal{R}$

$$\left| \frac{|A \cap R|}{|A|} - \frac{|R|}{|X|} \right| \leq \epsilon.$$

As mentioned in the introduction range spaces with bounded Vapnik-Chervonenkis dimension allow the construction of ϵ - approximations and ϵ-nets of sizes independent of the set system (see Vapnik, Chervonenkis (1971) and Haussler, Welzl (1987))

Let $(X, \mathcal{R})$ be a range space and $Y \subseteq X$.

- Y is called shattered, if $\mathcal{R}_Y = 2^Y$.
- The VC - dimension (Vapnik-Chervonenkis dimension) of $(X, \mathcal{R})$ is the maximal cardinality of a shattered subset of X.

There are many natural examples of range spaces in computational geometry with bounded VC-dimension. Unfortunately, sometimes it is hard to determine the VC- dimension. But it is easier to compute functions closely related to the VC- dimension, the shatter functions.

Let $(X, \mathcal{R})$ be a range space and $A \subseteq X$. Define $\Pi_{\mathcal{R}}(A) = \{A \cap R : R \in \mathcal{R}\}$. The primal shatter funktion $\pi_{\mathcal{R}}$ is

$$\pi_{\mathcal{R}}(n) := \max \{|\Pi_{\mathcal{R}}(A)| : A \subseteq X,\ |A| = n\}.$$

Vapnik and Chervonenkis (1971) and Sauer (1972) showed that for a range space with VC-dimension d, $\pi_{\mathcal{R}}(n) \leq \sum_{i=0}^{d} \binom{n}{i}$.

3 Sequential Construction

Let $(X, \mathcal{R})$ be a range space with $|X| = n$ and $|\mathcal{R}| = m$. For $m = n$ the six-standard-deviation result of Spencer (1986) ensures the existence of a 2-coloring with disrepancy $6\sqrt{n}$. It can be proved that such a 2-coloring guarantees also the existence of a $(1/r)$-approximation of size $O(r^2)$. And, since the discrepancy of the set system associated to a Hadamard matrix is $\Omega(\sqrt{n})$, $\Omega(r^2)$ is the lower bound for the size of $(1/r)$-approximations. But for arbitrary range spaces the algorithmically reachable size presently is only $O(r^2 \log m)$.

In this section we give for any parameter $r \geq 1$ an algorithm which computes an $(1/r)$-approximation $R \subset A$ of size $O(r^2 \log m)$ in $O(n^2 + \sum_{R \in \mathcal{R}} |R|)$ time. Thus our running time is optimal for all range spaces with $n^2 \leq \sum_{R \in \mathcal{R}} |R|$. (Note that the $O(n^2 + nm)-$time algorithm of Matoušek's (1991) as well as our algorithm are optimal for $\sum_{R \in \mathcal{R}} |R| = O(mn)$ and $n = O(m)$.)

The algorithm is a variation of the conditional probability method which derandomizes the simple probabilistic algorithm where we generate A by picking points of X independently and with probability $p = s/n$ where $s \geq 8r^2(\lceil \log 4m \rceil + 4)$.

Our improvement on the running time makes use of the special form of the upper bounds on the conditional probabilities which are sums of products of the exponential function taken over all ranges. The idea is: First, precompute the Taylor approximation of all factors in these products. This can be carried out in $O(n^2)$ time. Then run the standard conditional probability method, but with an "update trick": It turns out that in the l-th step of the method $(1 \leq l \leq n)$ it is necessary to update only one factor in each

product corresponding to a range R with $l \in R$. Because all possible factor have been precomputed, the update operation for each such range costs only constant time, and for all such ranges the required time is $O(\sum_{l \in R \in \mathcal{R}} |R|)$ time. After n steps the procedure terminates and the required time is $O(n^2 + \sum_{R \in \mathcal{R}} |R|)$. The result is:

Theorem 3.1 *Let $(X, \mathcal{R})$ be a range space with $|X| = n$ and $|\mathcal{R}| = m$. Then for every $r \geq 1$ and $s \geq 8(\lceil \log(4m) \rceil + 4)$ we can compute in $O(n^2 + \sum_{R \in \mathcal{R}} |R|)$ time a $(1/r)$-approximation A of size $|A| = s$.*

4 Parallel Construction

The parallelization of the sequential construction of ϵ-approximatons is based on the following close relationship between approximations and discrepancies due to Matoušek, Welzl and Wernisch (1991): If the discrepancy of a n point range space is δ, then there is a $\frac{2\delta}{n}-$ approximation A of size $|A| = \lceil \frac{n}{2} \rceil$. But two-colorings with discrepancy roughly $0(n^{1/2+\tau}\sqrt{\log m})$ can be computed for any $\tau \geq \frac{1}{n}$ by the metod of limited dependency in parallel (Berger, Rompel (1989), Motwani, Naor, Naor (1989)). Our algorithm is an interplay between parallel discrepancy computation and approximation construction by divide-and -conquer. The result is:

Theorem 4.1 *Let $(X, \mathcal{R})$ be a range space with $|X| = n$ and $|\mathcal{R}| = m$. Let $r \geq 1$, $0 < \tau \leq \frac{1}{4}$ and $\alpha = 6 \cdot 10^4$. We can compute using $O(m^{2+\frac{1}{\tau}})$ EREW parallel processors an $(1/r)$-approximation A of size*

$$\alpha \cdot (r^2 \log m)^{\frac{1}{1-2\tau}} \leq |A| \leq \alpha \cdot (r^2 \log m)^{\frac{1}{1-2\tau}} + 2$$

in $O(\log^3 m \log n)$ time.

Finally, using Theorem 4.1 we give a parallel counterpart of Matoušek's ϵ-approximation algorithm for polynomially bounded primal shatter function.

Theorem 4.2 *Let $(X, \mathcal{R})$ be a range space with $|X| = n$ and $|\mathcal{R}| = m$. Let the primal shatter function $\pi_{\mathcal{R}}(m)$ be bounded, i.e. $\pi_{\mathcal{R}}(l) = O(l^d)$ for a contant d and all $l = 1, \ldots, n$. Suppose that we are given a parallel subspace oracle which for a set $Y \subset X$ returns a list of all distinct $R \cap Y$, $R \in \mathcal{R}$ using at most $O(|Y|\pi_{\mathcal{R}}(|Y|))$ processors in $O(\log n)$ time. Let $1 \leq r \leq n$ and $0 < \tau \leq \frac{1}{4}$. Then an $(1/r)$-approximation of $(X, \mathcal{R})$ of size $O((r^2 \log r)^{\frac{1}{1-2\tau}})$ can be computed using at most $O(n^{4d+\frac{1}{\tau}d+3})$ processors in $O(\log^5 n)$ time.*

5 Open Problems

There are some natural questions arising from the previous discussion:

a) Can one show an parallel counterpart of the Matoušek, Welzl, Wernisch (1991) algorithm for polynomially bounded dual shatter function?

b) Is it possible to compute a $(1/r)$-net of size $O((r \log r)^{\frac{1}{1-2\tau}})$ in parallel. If so, how small must τ be so that such an algorithm is applicable to concrete range spaces, where the computation of a net of size $O(r \log r)$ seems to be necessary? The first step in this direction would be the parallelization of the random sampling approach of Chazelle and Friedmann (1990).

Preprints
"Angewandte Mathematik und Informatik"

9/95 - S G. Alsmeyer: Superposed Renewal Processes: A Markov Renewal Approach

10/95 - I W.-M. Lippe, Th. Feuring, Th. Büscher: A Fuzzy-Neural Network Based on the Backpropagation Algorithm.

11/95 - I C. Fahrner, G. Vossen: Transforming Relational Database Schemas into Object-Oriented Schemas according to ODMG-93.

12/95 - I W.-M. Lippe, Th. Feuring, L. Mischke: Supervised Learning in Fuzzy Neural Networks.

13/95 - I D. Laurent, N. Spyratos, G. Vossen: Optimization and Serializability of Update Transactions in Marked Databases.

14/95 - N A. Klawonn: An Optimal Preconditioner for a Class of Saddle Point Problems with a Penalty Term, Part II: General Theory.

15/95 - I L. Becker, K. Hinrichs, A. Voigtmann: An Object-Oriented Data Model and a Query Language for Geographic Information Systems.

16/95 - I J. Ebert, G. Vossen: I-Serializability: Generalized Correctness for Transaction-Based Environments.

17/95 - I G. Vossen: Old and New Models for Database Transactions.

18/95 - I J. Döllner, K. Hinrichs: Geometric, Chronological, and Behavioral Modeling.

19/95 - I J. Döllner, K. Hinrichs: The Virtual Rendering System -- a Toolkit for Object-Oriented 3D-Graphics.

20/95 - I D. Lammers: A Study on Dynamic Load Balancing Systems.

21/95 - S F. Harten: Prophetenregionen bei zeitlichen Bewertungen im unabhängigen Fall.

22/95 - S F. Harten: Remark on ''A difference prophet inequality for bounded i.i.d. variables, with cost for observations''

23/95 - I J. Meidanis, G. Vossen, M. Weske: Using Workflow Management in DNA Sequencing.

1/96 - I A. Brayner, M. Weske: Einsatz von FlowMark™ in der Molekularbiologie.

2/96 - I C. Bauzer Madeiros, G. Vossen, M. Weske: GEO-WASA -- Supporting Geoprocessing Applications using Workflow Management.

3/96 - I M. Weske, G. Vossen, C. Bauzer Madeiros: Scientific Workflow Management: WASA Architecture and Applications.

4/96 - I L. Becker, A. Voigtmann, K. Hinrichs: Developing Applications with the Object-Oriented GIS-Kernel GOODAC

5/96 - I A. Voigtmann, L. Becker, K. Hinrichs: A Query Language for Geo-Applications

6/96 - I A. Voigtmann, L. Becker, K. Hinrichs: Temporal Extensions for an Object-Oriented Geo-Data-Model.