Abstracts of the Workshop on Computational Geometry (CG'92)

March 12–13, 1992

Utrecht, the Netherlands

Technical Report RUU-CS-92-10

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

Abstracts of the Workshop on Computational Geometry (CG'92)

March 12–13, 1992

Utrecht, the Netherlands

RUU-CS-92-10

Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454

ISSN: 0924–3275

Foreword

This document contains abstracts of the papers that were presented at the 8th European Workshop on Computational Geometry (CG '92), held in Utrecht (The Netherlands), March 12-13, 1992, and supported by EURICS, the Eindhoven-Utrecht Research Institute in Computing Science. It was the first time that this workshop—formerly known as the German Workshop on Computational Geometry—took place outside Germany or Switzerland. Moreover, the official language of the workshop has switched from German to English. We think that the increased interest in the workshop warrants this more international approach.

The increased interest also led to a problem: the number of submissions was too large to accomodate all speakers, and we had to disappoint some people. The final program contains 27 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

Mark de Berg
Annerie Deckers (secr)
Mark Overmars (chair)

Thursday March 12

Friday March 13

Path Planning Under Uncertainty (How to Look Around a Corner)

Christian Icking Rolf Klein
Praktische Informatik VI
FernUniversität Hagen
Elberfelder Straße 95
5800 Hagen 1, Germany

Abstract

Let l_1, l_2 be two halflines in the plane meeting at vertex v, and let $\sigma < \pi$ be the counterclockwise angle from l_1 to l_2 around v. Let s be the point on l_1 that has unit distance to v. Suppose that the wedge (l_1, l_2) is opaque, like the corner of a building, so that a robot standing at s cannot see any point of l_2 (exept v). In particular, the robot does not have any clue as to the size of σ.

The robot's task is to "look around corner v", i. e. to move in such a way that l_2 becomes visible as efficiently as possible. We are interested in a competitive strategy where the worst case ratio of the length of the path the robot walks before it can see l_2 and the length of the shortest path from s to a point from which l_2 is visible is as small as possible.

With naive methods, one can show that 1.15 is a lower bound for the competitive factor of this problem. Also, one can give a simple strategy that always achieves a factor less than 1.57.

In this talk we present an optimal competitive strategy for the corner problem. It involves the unique solution of a differential equation of Abel type. Approximately, its competitve factor is 1.212.

Secure Path Planning

Sabine Stifter
RISC-Linz
Johannes Kepler University
A-4040 Linz, Austria

Our investigations have been motivated by an industrial project on the automated control of tunneling: For the lining of tunnels by concrete rings an erector (a robot in its nature) is used. The erector consists of a body, an arm and a gripper. The body can move along the axis of the tunnel, the arm can rotate around this axis and can be extended. The gripper has three joints and a gripping mechanism. The gripper joints are only used for the final assembly of the ring elements.

The task of the erector is to grip the ring elements from a conveyor, bring them to the "insertion position" and install them. Typical obstacles in the region where the erector is moving are conveyor belts and cables. The critical motion – with respect to collisions – is the path form the gripping position to the insertion position.

There are several criteria with respect to which the path should be optimized: The path should be

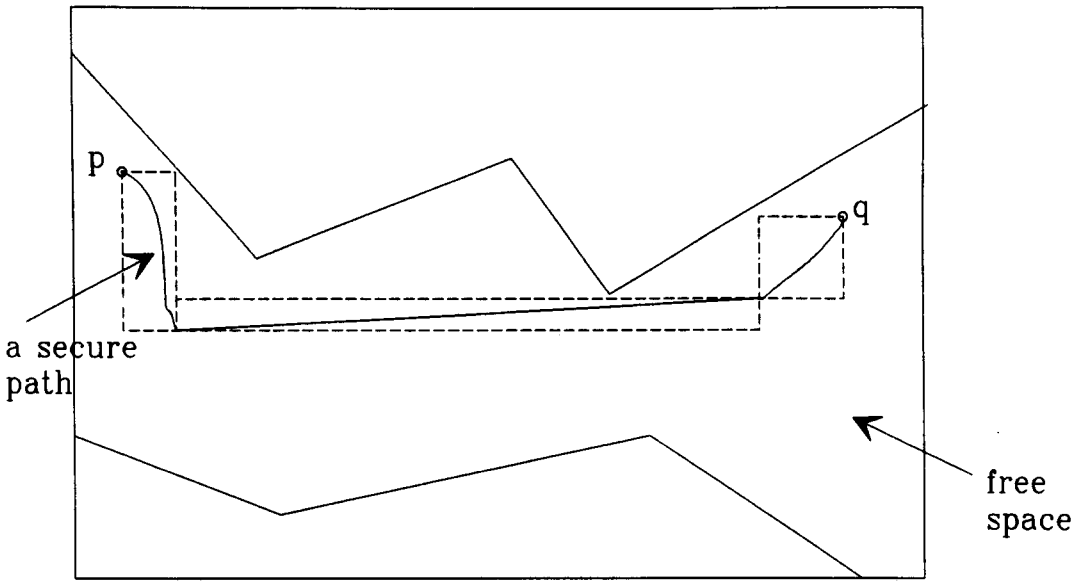
- short with respect to time,
- secure in the sense that, in case several joints are moving simultaneously, no collision occurs even if syncronizations of the joints are not implemented,
- the angle or stroke each joint is moved should be minimal.

Although the motivation comes from a very concrete example, the formulation of the problem is applicable in many areas of path planning: Consider a robot with n joints. For a start position p and a goal position q, find a path from p to q that is secure and short with respect to time and/or length.

Shortest paths with respect to time or length have already been extensively considered in the literature. However, it is usually assumed that any path can be followed by the robot, not going into details how the syncronization of the joints can be done. So we mainly stick to the problem of finding secure paths that are "short" with resepct to time (not necessarily shortest). Stopping the motion of a joint and starting it again needs a not neglectable amount of time. So there has to be a balance between shortest paths and the number of interrupts for the motion.

Formally, we define a secure path to be a sequence$(\phi_1, \ldots, \phi_k)$ of monotone (w.r.t. each coordinate) curve in configuration space such that with each monotone segment ϕ_i from p to q all curves (paths) in the box having p and q as diametral vertices are collision free, i.e. contained in free space.

Compare the following figure:



This already implies that secure paths can be constructed by partitioning (an approximation of) free space into boxes that only contain secure paths. The final path is contained in a sequence of such boxes. Moving from one box to a second one is only secure if the motion is stopped at the boundary of the box (i.e. one waits till all joints have reached the specified position) and only then all joints start a new motion.

So one is interested in having a minimal number of boxes in the sequence in order to minimize the number of times the joints have to be stopped. On the other hand, a minimal number of boxes in the sequence does not necessarily yield a shortest path with respect to time or length. The two requirements are often contradictory.

We study the complexity of finding secure paths and how to construct secure paths that are also short with respect to time or length.

ARTIBODIES: A SYSTEM FOR MODELLING ARTICULATED BODIES

J.E.Mebius
TU Delft
Vakgroep Technische Informatica
2600 AJ Delft
the Netherlands

1 Extended Abstract

1.1 Introduction

Articulated bodies consist of rigid links connected by joints. The ARTIBODIES system is restricted to articulated bodies in which 1) each joint connects exactly two links and 2) there are no kinematic loops. Restriction 1) allows a graph representation: joints correspond to edges and links to vertices. Restriction 2) amounts to allowing only free trees as articulated-body graphs. This idea is not new: see [LIEG 1985, Ch. 4.4.3] or [WITT 1977], but the present elaboration is, to the best knowledge of the author.

By assigning an arbitrary link as the **primary link** the articulated body becomes hierarchically organised, and its graph becomes a **rooted tree**. Each joint connects a **predecessor link** and a **successor link**. The lowest level of this hierarchy consists of the primary link, the next- higher level consists of the links connected to the primary link, if any, etc, until on each highest level one meets links without successor (**end links**). Each link except the primary link has a predecessor link, and the joint between them really exists; the primary link has no predecessor by nature, but it is convenient to think of the primary link as being connected to the outside world by a fictitious joint, its so-called **OSW joint**, the outside world playing the rôle of predecessor link (**OSW link**). Articulated-body trees have a finite variable unrestricted branching degree.

1.2 Coordinate systems

To each link a Cartesian coordinate system is attached: its Link Coordinate System **LCS**. The OSW link carries the Absolute Coordinate System (**ACS**).

A link is said to be in **standard position and orientation** if its LCS coincides with the ACS. A complete specification of a link in general position and orientation consists of a fixed part: its specification relative to its LCS, and a variable part: the displacement D from the ACS to its LCS. In computer graphics this displacement is known as the **modelling transformation**.

D is uniquely represented by a rotation followed by a translation, in formula: $D(r) = R(r) + T$. The rotation R introduced above is called the **absolute orientation** of the link and its coordinate system; the **relative orientation** of a coordinate system LCS' with respect to a coordinate system LCS is the rotation part of the displacement from LCS to LCS'.

1.3 Design of the ARTIBODIES software

Tree data structure: there are link nodes and joint nodes. A **link node** has a name and contains the absolute position and orientation of the link: the position as the vector T, the orientation as the Euler-Rodrigues parameters of the rotation R. The link node contains also data to draw a picture on the CRT screen. A **joint node** has a name and contains the relative position and orientation of the successor link with respect to the predecessor link, also as a vector and a set of Euler-Rodrigues parameters.

At the root we have the OSW link node. It points to a linked list of joint nodes, which represent the fictitious OSW joints of several articulated bodies. Each of these joint nodes points to a link node for the primary link. Each primary-link node points to a linked list of joint nodes which may have any length, including zero, depending upon the graph structure of the body. In general each joint node carries exactly one link node, and each link node carries zero or more joint nodes. For convenience each joint node also points back to the node of the predecessor link.

Procedures: all ARTIBODIES tree procedures perform conventional tree and list traversals. They pertain to operations on a single articulated body (**single-body procedures**) or on the entire forest of articulated bodies (**multiple-body procedures**).

Single-body procedures have the OSW joint node as a parameter, directly access the primary link, do their work and terminate. Multiple-body procedures conceptually have the OSW link node as a parameter, but are conveniently implemented with the first OSW joint node as a parameter instead of the OSW link node. They operate like single-body procedures, but instead of terminating after the first joint node, they process the entire list of joint nodes.

The single-body procedures operate depth-first: they access the successor link node and iterate through its joint node list, calling themselves recursively for each node in the list. Iterations stop when the list is exhausted, recursions stop when end link nodes are reached. The multiple-body procedures call their single-body counterparts iteratively until the joint node list is exhausted.

1.4 ARTIBODIES specification language

ASL (ARTIBODIES Specification Language) is a language for specifying the position, orientation and configuration of articulated bodies. In accordance with the hierarchical nature of the data structure to be described, ASL is a context-free non-regular language, which is interpreted by a finite-state automaton with stack.

The idea of ASL is best explained with the help of an example: the male human body. Due to lack of space only the graph structure is given.

```
JL JLJLJLZZZ JLJLJLZZZ JLZ JLJLJLZZZ JLJLJLZZZ JLZ      Z

JL           (Centre and Torso)                         Z
   JL      Z (Hip and Thigh; Shoulder and Upper Arm)
     JL    Z (Knee and Lower leg; Elbow and Forearm)
       JLZ   (Ankle and Foot; Wrist and Hand: end links)
```

Torso, hands and feet are here modelled as non-articulated parts of the body. Primary link is the torso.

The primary link is specified by the outermost J and L clauses and Z token. A J clause is always followed by an L clause; together they act like an opening parenthesis. The corresponding closing parenthesis is represented by the Z token. All joints and links at the next-higher level are specified by J and L clauses and Z tokens within the JLZ for the primary link. In our example they are the neck and head, the shoulders and upper arms, the hips and thighs, and the pubic area and penis. For the limbs we must ascend two more levels to get at the extremities.

2 Literature

LIEG 1985 Alain Liegeois, Performance and computer-aided design
Robot Technology - Volume 7
London: Kogan Paige, 1985, ISBN 0-85038-652-7

MEBI 1990 J.E.Mebius, On the rotation interpolation and animation problems in computer graphics
Delft University of Technology, Faculty of Technical Mathematics and Informatics, P.O.Box 356, 2628 AJ DELFT, The Netherlands, ISSN 0922-5641, Report 90-08, 1990

MEBI 1991 J.E.Mebius, ARTIBODIES: A system for modelling articulated bodies
Delft University of Technology, Faculty of Technical Mathematics and Informatics, P.O.Box 356, 2628 AJ DELFT, The Netherlands, ISSN 0922-5641, Report 91-57, 1991

STAN 1980 Thomas A. Standish, Data structure techniques
Addison-Wesley, 1980, ISBN 0-201-07256-4

WHIT 1924 E.T.Whittaker, Analytische Dynamik der Punkte und starren Körper. Berlin: Verlag von Justus Springer, 1924

WIRT 1987 N. Wirth, Compilerbouw. Schoonhoven: Academic Service, 1987, ISBN 90-6233-234-X

WITT 1977 J. Wittenberg, Dynamics of systems of rigid bodies
Stuttgart: B. G. Teubner, 1977, ISBN 3-519-02337-7

Computation of Free Configuration Space by Solid Modeling Techniques

Fausto Bernardini Vincenzo Ferrucci Alberto Paoluzzi

Dip. di Informatica e Sistemistica
Università "La Sapienza"
Via Buonarroti 12, 00185 Roma

We present a new method for approximate computation of the free configuration space of mobile systems. The geometrical methodology, introduced in [2], makes use of multidimensional solid modeling techniques. A mor e detailed discussion of the approach presented here can be found in [1].

Modeling degrees of freedom Consider a rigid body B which is free to translate in the plane along a given direction. Then, its position is uniquely determined by a single scalar parameter. The set of all pos sible positions that the object may assume can be represented by extruding it in 3D space. Any position of B is obtained by sectioning the extruded volume with a plane, and projecting the section back onto the original 2D space. Now, consider a system of several d-dimensional rigid bodies. For sake of concreteness, suppose we deal with two bodies embedded in $\Re^n$, B_1 with one rotational degree of freedom α, and B_2 with one translational degree of freedom t_1. A representation of all possible instances the system may assume accordingly to the variation of α and t_1 is obtained with two extrusio n steps, where each step encodes one degree of freedom. First, B_1 is screw extruded because of the dependency of its position from α, resulting in $B_1' \subset \Re^{n+1}$; since B_2 exhibits no dependency from α, a suitable linear extrusion is applied to it. Actually, sectioning B_2', the sol id obtained by extruding B_2, with a hyperplane $\alpha = \text{cost}$ and projecting back onto $\Re^n$ must yield B_2 itself. Such an extrusion is calle d a straight extrusion. The second extrusion step reflects dependency from t_1. By a simmetric argument, since B_1 does not exhibit dependence from t_1, B_1' is straight extruded yielding $B_1'' \subset \Re^{n+2}$, to guarantee sections with a hyperplane perpendicular to the x_{n+2} axis to be constant. Conversely, the body B_2 varies its position with t_1, so an appropriate linear extrusion is performed.
The whole system composed of two rigid bodies is now represented as a $(d+2)$-dimensional object in $\Re^{n+2}$, with the implicit assumption that degrees of freedom are mutually independent, i.e. the system is holonomic. The discussion generalizes easily to any number of bodies and to any mix of degrees of freedom.

Extended Configuration Space Let the moving system $R \cup E \subset \Re^d$ be composed of a set of rigid parts $R = \{R_i\}$ and a set of rigid obstacles $E = \{E_i\}$. The mobile system a nd the obstacles are usually bidimensional or threedimensional, so that $d = 2, 3$. If the obstacles move along known trajectories, the whole system can be statically modeled taking into account time as an additional dimension, thereby obtaining a $3D$ or $4D$ spacetime, respectively.

The workspace WS $\subset \Re^d$, a subset of the geometrical space embedding the moving system, identifies the region of interest. A convenient choice is an enclosing hypercube —

Solid model of the moving system

↓

Extrusion in ECS

↓

Computation of the intersection sets C_i

↓

Projection of each intersection set from ECS onto CS

↓

Union of the projected sets in CS

↓

Difference between CS and the union set

Figure: Conceptual skeleton of the ECS method

in 2, 3 or 4 dimensions. The configuration space CS = $I_1 \times \cdots \times I_k \subset \Re^k$ is the product space of range intervals I_i of configuration parameters, where k is the total number of degrees of freedom of R. If range intervals of configuration parameters are normalized CS is the standard k-dimensional hypercube.

Extended Configuration Space method: The Extended Configuration Space ECS is defined as WS x CS, and therefore is a $(d+k)$-dimensio nal hypercube.

1. **Modeling Step.** A suitable (simplicial or convex decomposition) solid model of both R and E is given within their geometrical embedding space.

2. **Extrusion Step.** Both the robot and the obstacles are extruded in Extended Configuration Space, so obtaining a set S of higher dimensional convex objects .

3. **Intersection Step.** Both auto-intersections of the robot and intersections of the latter with obstacles are checked and computed. The result is a set of quasi-disjoint convex sets, $\{C_i\}$.

4. **Projection Step.** Each convex set C_i in ECS is projected onto CS. This process can be thought of as a sequence of d elementary projections, where an elementary projection is a projection along a coordinate direction. The result is the set

$$\{\Pi_{CS}(C_i)\},$$

which is still a collection of convex sets, now in configuration space, but these are unfortunately no longer quasi-disjoint.

5. **Union Step.** The set of points in configuration space which correspond to prohibited configurations (configuration space obstacles) for the moving system is obtained as

$$\bigcup_i \Pi_{CS}(C_i).$$

6. **Difference Step.** The difference between the configuration space CS and the set obtained as in Step 5 coincides with FP, the free configuration space of the moving system $R \cup E$.

The key issue to be solved in order to obtain a practical method from the general ECS method is mainly a representational one. How do w e represent the entities which have to be manipulated in the different steps of th e method? How do we perform the operations (intersection, projection, union, difference) in the higher dimensional spaces ECS and CS?

Originally [2], the technique was presented and analyzed by using a representation based on simplicial decompositions throughout Steps 1-6. A first disadvantage is the exponential growth in the size of the representation while building the ECS representation of the moving system, whereas a second weak point is the difficulty of performing in an efficient way the projection $\Pi_{CS}(C_i)$, as w ell as of computing the result of the required union and difference operations.

Preliminary discussion We discuss a different instance of the method, which solves many of the previous difficulties, while using the same high-level conceptual scheme. The idea is to push back the computation of all geometric details to the very last minute, using an *implicit* representation scheme $\mathcal{IR}$ as long as possible and moving to an *explicit* representation scheme $\mathcal{R}$ only when strictly necessary. A further keypoint is that $\mathcal{R}$ must exploit as well as possible the peculiarity of the projection operation and the simplicity of the geometry o f the intersections in ECS — a set of quasi-disjoint convex sets $\{C_i\}$.

It is necessary to switch to an explicit representation scheme $\mathcal{R}$ when t he projection step has to be performed. The following operations must be performed with reasonable computational effort:

(a) given the implicit representation $\mathcal{IR}$ of the convex set C_i, compute its explicit representation $\mathcal{R}(C_i)$.

(b) given $\mathcal{R}(C_i)$, compute $\mathcal{R}(\Pi_{CS}(C_i))$, the represent ation of the projection of C_i onto CS.

(c) given the representation of the projections of the C_i's onto CS, compute the representation of their union

$$\mathcal{R}\left(\bigcup_i \Pi_{CS}(C_i)\right),$$

and of FP as the difference between CS and the former.

The choice of both $\mathcal{IR}$ and $\mathcal{R}$ is critical for the practicability and effectiveness of the resulting algorithm. Moreover, since the necessity of conversion arises, the two schemes can not be selected independently from each other. In the combination we propose $\mathcal{IR}$ is given by a set of linear inequalities and $\mathcal{R}$ by a bintree, a dimension-independent generalization of quadtrees and octrees. The representation conversion (a) is performed by its reformulation as a CSG to bintree conversion problem. Th e projection (b) of a bintree and the computation of union (c) deserve no addition al difficulties.

Potential for Parallelization A peculiarity of the method proposed in the previous sections is a high degree of data independence. Starting from the modeling step, consider a convex cell of the decomposition of $R \cup E$ as the basic data unit to be manipulated. Such convex cells can be distributed to different processors, and the extrusion step can be performed in parallel with no interprocessor communication. At this point, each processor owns a set of convex cells in ECS. Due to the properties of the extrusion operator, cells belonging to such a set are quasi-disjoint, so that only cells belonging to distinct sets need to be checked for intersection. However, the intersection step requires intensive interprocessor communication. After that, conversion to bintree representation and projection of each convex cell can be executed in parallel, whereas resincronization is required for the union operation in CS.

Assuming a distributed memory parallel machine, we observe that the bintree representing a convex cell might not fit entirely in local memory. An alternati ve is to subdivide the ECS space and assign a region to each processor, in an attem pt to balance the amount of geometric data to be held in their own memory by different processors. When the ECS space is subdivided in cubic cells, the same technique as before can be applied, since the conversion algorithm from $\mathcal{IR}$ to $\mathcal{R}$ does handle the case of non-convex objects.

References

[1] BERNARDINI, F., FERRUCCI, V., AND PAOLUZZI, A. Computation of free configuration space with solid modeling techniques. Tech. rep., Dip. di Informatica e Sistemistica, Università di Roma "La Sapienza", Rome, Italy, 1992. In preparation.

[2] PAOLUZZI, A. Motion planning+solid modeling=motion modeling. Tech. Rep. 17–89, Dip. di Informatica e Sistemistica, Università di Roma "La Sapienza", Rome, Italy, Nov. 1989. Submitted.

An Optimal Algorithm for the Maxima Set Problem for Data in Motion*

(Extended Abstract)

Paolo Giulio Franciosa

Università "La Sapienza", Dipartimento di Informatica e Sistemistica, via Salaria 113, I-00198 Roma, Italia.

and Carlo Gaibisso

Istituto di Analisi dei Sistemi ed Informatica, viale Manzoni 30, I-00185 Roma, Italia.

and Maurizio Talamo

Università "La Sapienza", Dipartimento di Informatica e Sistemistica, via Salaria 113, I-00198 Roma, Italia.

1 Introduction

In this paper we consider a quite new context, in which algorithms are designed in order to on-line process a sequence of queries concerning a set of moving objects with bounded speeds.

The fact that objects are *moving* instead of *changing* (i.e. appearing and/or disappearing) makes exploitable the knowledge about the positions they assumed.

If a classical dynamic approach is followed in the solution of data motion problems, a change in a piece of data will always imply some delete/insert operation, while, according to the data motion approach, if an object moves without modifying its relationship with other objects, as far as the query answer is concerned, this move could actually result in no action at all.

The paper refers to the computational model introduced by Kahan [2] for problems involving moving data. In the same paper results concerning the data motion version of some basic problems in 1-dimension are derived, such as determining maximum, median, and minimum gap of a set.

We face the well known problem of determining the maxima of a set of points, but in this context points represent the position of moving objects in $\Re^d$, for any d, each having a known speed bound. We consider the problem in 2 dimensions in the case of both uniform and non-uniform speed bounds.

In both cases we give a 3-lucky algorithm and prove that 3 is the optimal lucky ratio. Unformally speaking, the lucky ratio of an algorithm $\mathcal{D}$ is the worst case ratio between the number of objects $\mathcal{D}$ and the best non-deterministic algorithm, they must investigate for their actual position (*Update* operations) in order to answer the same query (see [2] for a formal definition of lucky ratio).

*Work partially supported by the ESPRIT II Basic Research Actions Program of the European Community under contract No. 3075 (project ALCOM) and by the Italian MPI National Project "Algoritmi e Strutture di Calcolo".

The algorithm dealing with uniform speed bounds processes a single query q in $O(\min(k \cdot \log^2 n, n \cdot \log n))$ worst case computation time, where k is the number of maximal objects in the solution plus the number of *Update* operations performed answering q.

In the non-uniform case an $O(n \cdot \log n)$ worst case computation time algorithm is given.

Both algorithms require $O(n)$ worst case space.

As far as the d-dimensional problem is concerned, we prove that for any k, not any k-lucky algorithm exists in 3 or more dimensions. This means that for any deterministic algorithm $\mathcal{D}$ there are instances that could be solved by performing only one *Update* operation, while $\mathcal{D}$ must investigate the actual position of all objects.

2 Problem

We are given a set $M = \{p_1, p_2, \ldots, p_n\}$ of point-shaped objects assuming positions in $\Re^d$ and a set $R = \{r_1, r_2, \ldots, r_n\}$ of *rates* in $\Re$ such that, if $P(p_i, t)$ denotes the position of p_i at time t, $dist^1(P(p_i, t), P(p_i, t + \Delta)) \leq \Delta \cdot r_i$, for any $t, \Delta \geq 0$ and $p_i \in M$. The problem consists in on-line processing a sequence of queries $Max(t_1), Max(t_2), \ldots, Max(t_m)$, where $Max(t_i)$ denotes the set of all *maximal* objects, i.e. objects whose position at time t_i is maximal [4] with respect to those of all other objects, and t_i is assumed to be less than or equal to t_{i+1}, $i = 1, 2, \ldots, m - 1$. Furthermore, since the sequence has to be processed on-line, t_i will not be available until $Max(t_{i-1})$ has not been answered.

3 Model and Basic Definitions

The model introduced by [2], which we will refer to in the paper, can be considered as a RAM with uniform costs [1] having an open window on the world, which consists of a set $M = \{p_1, \ldots, p_n\}$ of point-shaped moving objects assuming positions in $\Re^d$.

The RAM set of primitives is extended by the *Update*(p) operation, which, if invoked while processing query $q(t)$, returns the position $P(p, t)$ of object p at that time.

A *data motion algorithm* A on-line processes sequences of queries $Q = q(t_1), \ldots, q(t_m)$, with $t_i < t_{i+1}$, $i = 1, 2, \ldots, m - 1$, and involves both RAM and *Update* operations.

The *worst case computation time* of A is defined as $\sup_Q \max_{q \in Q} C_A^T(q)$, where $C_A^T(q)$ is the number of RAM operations performed by A to process query q.

The performance of A in terms of *Update* operations is compared with that of an hypothetical *lucky* algorithm.

In this respect, let us assume that, for some $p_i \in M$, the last *Update*(p_i) operation has been performed while processing query $q(t)$: at time $t' = t + \Delta$ all is known about the position of p_i is the *free range* $[f_{l1}, f_{h1}] \times \ldots \times [f_{ld}, f_{hd}]$ in which it could lie, where $[f_{lk}, f_{hk}]$ is the projection of $\{ x \mid dist(x, P(p, t)) \leq \Delta \cdot r_i \}$ on dimension k.

The set of all the free ranges of objects in M at a certain time will be referred to as the *state of knowledge* at that time.

In order to process query $q(t)$ the state of knowledge at time t could be too vague, thus requiring some *Update* operations. The *lucky* algorithm selects in a non-deterministic way which objects to update in order to perform the minimum number of such operations, as if it already knew the actual position of objects within the corresponding free ranges and just wants to prove the correctness of its solution.

[1] $dist(a, b)$ denotes the L^∞ distance between a and b.

Algorithm A is said to be *k-lucky* if for all queries q and with respect to any state of knowledge, $C_A^U(q) \leq k \cdot C_L^U(q) + O(1)$, where $C_A^U(q)$ and $C_L^U(q)$ denote the number of *Update* operations performed respectively by A and the *lucky* algorithm to process query q.

Algorithm A has *optimal lucky ratio* k if A is k-lucky and no algorithms k'-lucky exist, with $k' < k$.

Given two free ranges $a = [a_{l1}, a_{h1}] \times [a_{l2}, a_{h2}] \times \ldots \times [a_{ld}, a_{hd}]$ and $b = [b_{l1}, b_{h1}] \times [b_{l2}, b_{h2}] \times \ldots \times [b_{ld}, b_{hd}]$, we say a *dominates* (resp., *potentially dominates*) b if and only if $(a_{l1}, \ldots a_{ld})$ (resp., $(a_{h1}, \ldots, a_{hd})$) dominates $(b_{h1}, \ldots, b_{hd})$.

A free range is said *external* if and only if is not potentially dominated by any other free range in the current state of knowledge.

A free range a is in *conflict* with b if and only if a not empty set D of indices of coordinate exists such that $(\forall i \in D)\ [a_{li}, a_{hi}] \cap [b_{li}, b_{hi}] \neq \emptyset$ and either $(\forall i \in \{1, \ldots, d\} - D)\ a_{hi} < b_{li}$, or $(\forall i \in \{1, \ldots, d\} - D)\ a_{li} > b_{hi}$.

It is worth noting that if a is in conflict with b it is not possible to decide about the dominance relation between $object_a$, i.e. the object of M corresponding to a, and $object_b$ without performing at least one *Update* operation.

A set C of free ranges is a *crucial set* if and only if either

1. $C = \{a, b\}$, a and b are external and a is in conflict with b, or
2. $C = C' \cup \{b\}$, where:
   - $C' = \{a^1, a^2, \ldots, a^k\}$ is the set of all external free ranges that potentially dominates b;
   - C' does not contain any couple of in conflict free ranges;
   - b is not potentially dominated by any other free range;
   - b is not dominated by any free range.

A free range a is said *maximal* if and only if for any free range b in the current state of knowledge $(b_{h1}, \ldots, b_{hd})$ does not dominate $(a_{l1}, \ldots, a_{ld})$, i.e. $object_a$ is guaranteed to be maximal.

4 Lower Bounds

The following lemma is crucial in proving our lower bounds:

Lemma 4.1 *Let ST be any update strategy for the data motion maxima problem and K any state of knowledge containing a crucial set C, then ST must* Update *at least one free range in C.*

Concerning the 2-dimensional version of the problem, it is possible to prove that:

Lemma 4.2 *Let ST be any deterministic strategy for the data motion maxima problem in 2 dimensions: it is possible to generate a configuration C of 3 free ranges and to locate the corresponding objects in such a way that ST must perform 3* Update *operations in order to determine which among the objects are maximal, while the lucky strategy* Lucky *needs only one* Update *to solve the same problem.*

Lemma 4.2 immediately leads to:

Theorem 4.3 *There is no update strategy for the data motion maxima problem in 2 dimensions that can lead to a k-lucky algorithm with* $k < 3$.

Let us now consider the d-dimensional case, $d \geq 3$. Lemma 4.2 becomes:

Lemma 4.4 *Let ST be any deterministic strategy for the data motion maxima problem in d dimensions,* $d \geq 3$*: for any integer* $n > 0$ *it is possible to generate a configuration C of n free ranges and to locate the corresponding objects in such a way that ST must perform n* Update *operations in order to determine which among the objects have maximal position, while* Lucky *needs only one* Update *to solve the same problem.*

Leading to the following lower bound:

Theorem 4.5 *There is no update strategy for the data motion version of the maxima problem in d dimensions,* $d \geq 3$*, that can lead to a k-lucky algorithm, for any k.*

This means that even an optimal algorithm in the worst case has very poor performances with respect to the lucky ratio.

5 Update Strategy

The update strategy ST^* we adopt in solving the data motion maxima problem is the following:

> *update all the non point-shaped free ranges belonging to some crucial set in the current state of knowledge, if any. Stop, otherwise.*

It is possible to prove that ST^* always terminates and generates a state of knowledge from which the set of maximal objects can be determined without performing any *Update* operation.

The strategy performance in terms of lucky ratio are fixed by the following theorem:

Theorem 5.1 *The above introduced strategy for the data motion maxima problem, in 2 dimensions has optimal lucky ratio 3.*

6 3-lucky algorithms in 2 dimensions

Both the algorithms we describe, which solve the 2-dimensional version of the problem for uniform and non-uniform rates, implement strategy ST^*.

In the case of non-uniform rates crucial sets are found by sweeping the plane from top to bottom. The event point schedule coincides with the order in which the sweep-line l enters free ranges in the current state of knowledge K.

In order to answer a new query, a priority queue containing the event points is built from scratch.

While sweeping the plane, the algorithm maintains the left-to-right sequence of the external free ranges in K whose north-west vertex lies above l. When a crucial set is

detected all involved free ranges are updated in such a way that there will be no crucial sets above l.

When the whole plane has been swept, the set of maximal objects corresponds to the set of external free ranges.

Obviously, since it strictly implements strategy ST^*,

Theorem 6.1 *The proposed algorithm has optimal lucky ratio 3.*

In addition it is possible to prove that:

Theorem 6.2 *the algorithm has $O(n \cdot \log n)$ worst case computation time and needs $O(n)$ worst case space to answer a single query.*

Concerning the uniform rates version of the problem, instead of organizing from scratch the information related to the current state of knowledge each time a new query has to be answered, such an information is on-line maintained while processing the whole sequence of queries.

This approach is made possible by the data structure proposed by Overmars and van Leeuwen in [3]. The current set of external free ranges is scanned, updating the data structure each time a set of *Update* operations is trigged by the detection of a crucial set.

It is possible to prove that:

Theorem 6.3 *the proposed algorithm requires $O(\min(k \cdot \log^2 n, n \cdot \log n))$ worst case computation time to process a query, where k is the number of maximal objects in the solution plus the number of* Update *operations, and needs $O(n)$ worst case space.*

7 References

1. A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithm.* Addison Wesley Publ. Co., Reading, Massachussetts, 1974.

2. S. Kahan. A model for data in motion. In *Twentythird ACM SIGACT Symp. on Theory of Computing*, pages 267-277, 1991.

3. M.H. Overmars and J. van Leeuwen. Maintainance of configurations in the plane. *Journal of Comp. and System Sc.*, 23:166-204, 1981.

4. F.P. Preparata and M.I. Shamos. *Computational Geometry, an Introduction.* Springer-Verlag, New York, 1985.

Maxima in Convex Regions (Short Abstract)

Mordecai J. Golin *
INRIA Rocquencourt
7815 Le Chesnay Cedex - France
email: golin@margaux.inria.fr

Suppose that C is a bounded convex planar region. Let $p_1, \ldots, p_n$ be drawn I.I.D. from the uniform distribution over C and let M_n^C be the number of the points which are maximal. In this note we present results showing how the asymptotic behavior of $\mathbf{E}\left(M_n^C\right)$ depends on the geometry of C.

1 Introduction

In this note we discuss $\mathbf{E}\left(M_n^C\right)$, the expected number of maximal points in a set of n Independently Identically Distributed (I.I.D.) points drawn from the uniform distribution over some bounded convex planar region C.

The corresponding question for convex hulls has been well studied. Renyi and Sulanke [5] [6] proved that if n points are chosen I.I.D. from C then, if C is a convex polygon, the expected number of convex hull points is $\theta(\log n)$ while if C is convex and has a doubly continuously differentiable boundary the answer is $\theta\left(n^{1/3}\right)$. Dwyer [3] provides a survey of more recent results.

The expected number of maxima has not been examined nearly as closely. It has been known for many years [1] that if C is the unit square then $\mathbf{E}\left(M_n^C\right) = \theta(\log n)$. Recently Dwyer [3] proved that $\mathbf{E}\left(M_n^C\right) = \theta(\sqrt{n})$ when C is a circle and also proved a general upper bound $\mathbf{E}\left(M_n^C\right) = O(\sqrt{n})$ that is valid for any bounded convex planar C (this result can also be found in Devroye [2]).

In this paper we study the asymptotics of $\mathbf{E}\left(M_n^C\right)$ in detail. Note that the condition that C be convex is important. If it is abandoned then it can be shown that, for all functions f, $f(n) \leq n/\log^2 n$, and f slowly varying at infinity, there is some C such that $\mathbf{E}\left(M_n^C\right) = \theta(f(n))$ (see [4]) for details). If C is constrained to be convex the situation is very different. We show in this paper that for convex C either $\mathbf{E}\left(M_n^C\right) = \theta\left(\sqrt{n}\right)$ or $\mathbf{E}\left(M_n^C\right) = O\left(\log n\right)$; nothing between these two functions is possible. We also give sufficient conditions for $\mathbf{E}\left(M_n^C\right) = \theta(1)$ and $\mathbf{E}\left(M_n^C\right) = \theta(\log n)$.

The rest of the paper will use the following notation: if $p = (p.x, p.y)$ and $q = (q.x, q.y)$ are planar points we say that p *dominates* q if $p.x \geq q.x$ and $p.y \geq q.y$. If $S = \{p_1, \ldots, p_n\}$ is a set of points we say that P is *maximal* in S if there is no $q \in S$, $q \neq p$, such that q dominates p. We set

$$\mathrm{MAX}(S) = \{p : p \text{ is maximal in } S\}.$$

*This work was supported by a Chateaubriand fellowship from the French Ministère des Affaires Étrangères and by the European Community, Esprit II Basic Research Action Program Number 3075 (ALCOM).

See Figures 1 (a), (b) and (c). Suppose C is a bounded planar region. Let $S = \{p_1, \ldots, p_n\}$ be a set of n points drawn I.I.D. from the uniform distribution over C. Set $M_n^C = |\text{MAX}(S)|$ to be the number of maximal points in S. We will study $\mathbf{E}\left(M_n^C\right)$, the expected number of maximal points.

We will also look at the *outer layer* of S, the set of all $p \in S$ such that one of the four quadrants of the Cartesian axes centered at p contains no point in S (see [2] for more details). A set's outer layer always contains the set's convex hull. We define $OL(S)$ to be the outer layer points of S and $OL_n^C = |OL(S)|$ to be the number of such points.

2 Results

In what follows we will always assume that C is a *closed* region. We do this to ensure that C contains its boundary, ∂C : this assumption makes our proofs slightly simpler. Notice though that the assumption is not restrictive. If C is *any* bounded convex region then $\mathbf{E}\left(M_n^C\right) = E\left(M_n^{\overline{C}}\right)$ because a point chosen from the uniform distribution over $\overline{C}$ is in ∂C with probability zero. It thus suffices to analyze $\mathbf{E}\left(M_n^C\right)$ for closed C. Our first theorem is

Theorem 1 (The Gap Theorem) *Let C be a planar convex region. We say that a point $p \in C$ is an* upper-right-hand-corner *of C if p dominates every point $q \in C$. The expected number of maxima among n points chosen I.I.D. uniformly from C is qualitatively dependant upon whether C has an upper-right-hand-corner:*

- *If C does not have such a corner then $\mathbf{E}\left(M_n^C\right) = \theta(\sqrt{n})$.*
- *If C does have such a corner then $\mathbf{E}\left(M_n^C\right) = O(\log n)$.*

Note that for all convex C $\mathbf{E}\left(M_n^C\right)$ can not behave like a function asymptotically between $\log n$ and $\sqrt{n}$. Hence the name Gap Theorem, alluding to the gap between the two possible behaviors.

Example 1 Figures 1 (b), (c), (d), (f), and (h) all have upper-right-hand-corners and thus have $\mathbf{E}\left(M_n^C\right) = O(\log n)$: figures 1 (a), (e), and (g) don't and so have $\mathbf{E}\left(M_n^C\right) = \theta(\sqrt{n})$.

Recall that OL_n^C is the number of outer-layer points among n points chosen I.I.D. uniformly from C. Theorem 1 can be shown to say quite a lot about the expected number of outer layer points.

Corollary 1 *Let C be a planar convex region. If C is a rectangle with sides parallel to the Cartesian axes then $\mathbf{E}\left(OL_n^C\right) = \theta(\log n)$. Otherwise $\mathbf{E}\left(OL_n^C\right) = \theta(\sqrt{n})$.*

Example 2 In Figure 1, (a) has $OL_n^C = \theta(\log N)$ while all of the other regions have $OL_n^C = \theta(\sqrt{n})$.

Theorem 1 tells us that when C does not have an upper-right-hand-corner then $\mathbf{E}\left(M_n^C\right) = \theta(\sqrt{n})$. When C does have such a corner then all that we know is that $\mathbf{E}\left(M_n^C\right) = O(\log n)$.

To derive tighter bounds it is necessary to have better information about the tangents to the boundary of C at the corner. We digress momentarily to introduce notation describing these tangents.

Let C be a convex region. We say that C has an upper-right-hand-corner p if $p \in C$ dominates every point $q \in C$. If C has such a corner then the boundary curve of C as it leaves p can be divided into two parts: one curve that goes down and the other that goes to the left. We define two functions $d(\alpha)$ and $l(\alpha)$:

$$\begin{aligned} d(\alpha) &= \beta \text{ such that } (p.x - \beta, p.y - \alpha) \text{ is on the down curve,} \\ l(\alpha) &= \beta \text{ such that } (p.x - \alpha, p.y - \beta) \text{ is on the left curve.} \end{aligned}$$

See Figure 2. While these functions are not defined for all real α the convexity of C ensures that there is always some $\epsilon > 0$ such that both of the functions are well defined, convex and nondecreasing in $[0, \epsilon]$ with $d(0) = l(0) = 0$. Since the functions are convex their left and right derivatives exist everywhere in the interval except for the undefined left derivative at 0 and the undefined right one at ϵ.

The down tangent to C at p is the tangent to the down curve at p. The slope of this tangent line is totally determined by the value of the right derivative $d'_+(0)$. If $d'_+(0) = 0$ the tangent line is vertical. Similarly, if $l'_+(0) = 0$ the left tangent line, the tangent to the left curve, is horizontal. This is illustrated in Figure 2.

The next two theorems discuss the behavior of $\mathbf{E}\left(M_n^C\right)$ when C has an upper-right-hand-corner .

Theorem 2 *Let C be a convex planar region with upper-right-hand-corner p. If the down tangent at p is not vertical and the left tangent at p is not horizontal then $\mathbf{E}\left(M_n^C\right) = \theta(1)$. Otherwise $\mathbf{E}\left(M_n^C\right) = \Omega(1)$.*

Example 3 Figures 1 (b) and (f) have $\mathbf{E}\left(M_n^C\right) = \theta(1)$; figures 1 (c), (d), and (h) have $\mathbf{E}\left(M_n^C\right) = \Omega(1)$.

We now present a tight lower bound for many of the cases in which the left tangent is horizontal and/or the down tangent is vertical.

Theorem 3 *Let C be a convex planar region with upper-right-hand-corner p. Suppose further that at least one of the two tangents to C at p fulfills the following (Lipschitz-like) conditions:*

1. *The down tangent is not vertical and there are positive constants δ and c such that $l(\alpha) \leq c\alpha^{1+\delta}$.*

2. *The left tangent is not horizontal and there are positive constants δ and c such that $d(\alpha) \leq c\alpha^{1+\delta}$.*

3. *There are positive constants δ and c such that $d(\alpha) \leq c\alpha^{1+\delta}$ and $l(\alpha) \leq c\alpha^{1+\delta}$.*

Then $\mathbf{E}\left(M_n^C\right) = \theta(\log n)$.

Condition (1) forces the left tangent to be horizontal and condition (2) forces the down tangent to be vertical. The conditions of the theorem can be thought of as requiring not only the left (down) tangent to be horizontal (vertical) but the curve leaving C itself to be "almost" horizontal (vertical) near p. These conditions might seem artificial but in practice are satisfied quite often. For example:

Example 4 As an application of Theorem 3 we find that if C is a convex *polygon* with an upper-right-hand-corner and a vertical down tangent and/or a horizontal left tangent at the corner then $\mathbf{E}\left(M_n^C\right) = \theta(\log n)$, e.g. Figures 1 (c), (d), and (h) have $\mathbf{E}\left(M_n^C\right) = \theta(\log n)$. Combining this with Theorems 1 and 2 we find that if C is a convex polygon then $\mathbf{E}\left(M_n^C\right)$ can have only one of three possible behaviors: $\theta(\sqrt{n})$, $\theta(\log n)$, or $\theta(1)$.

Acknowledgements: The author thanks Luc Devroye and Rex Dwyer for their emailed comments concerning the expected number of maxima. He also thanks Jon Bentley for the idea which led to the proof of Theorem 1 part (i).

References

[1] J. L. Bentley, H.T. Kung, M. Schkolnick, and C.D. Thompson, "On the Average Number of Maxima in a Set of Vectors and Applications," *Journal of the Association for Computing Machinery*, **25**(4) October 1978, 536-543.

[2] Luc Devroye, *Lecture Notes on Bucket Algorithms*, Birkhauser Verlag, Boston, 1986.

[3] Rex Dwyer, "Kinder, Gentler, Average-Case Analysis for Convex Hulls and Maximal Vectors," *SIGACT NEWS*, **21**(2), Spring 1990, 64-71.

[4] M. Golin, "Notes on Maxima in Non-Convex Regions (Extended Version)," *INRIA Rapport de recherche 1535* (1991).

[5] A. Renyi and R. Sulanke, "Ueber die konvexe hulle von n zufallig gewahlten punkten, I," *Z. Wahrschien*, **2** (1963) 75-84.

[6] A. Renyi and R. Sulanke, "Ueber die konvexe hulle von n zufallig gewahlten punkten, II," *Z. Wahrschien*, **3** (1964) 138-147.

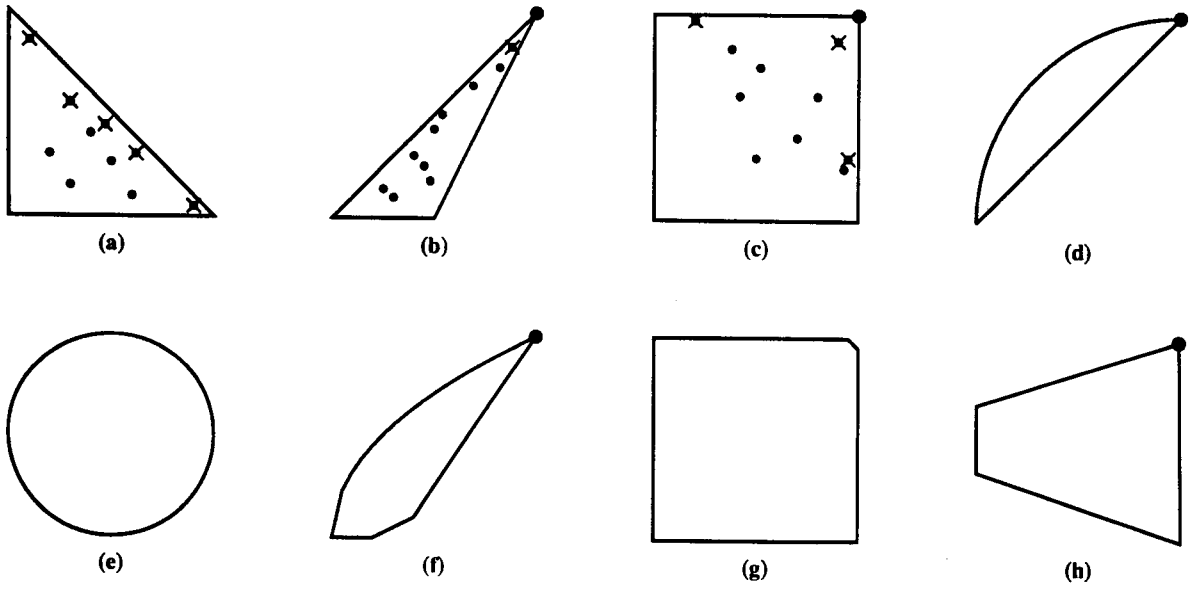


Figure 1: Figures (a), (b), and (c) each contain 10 points the maxima of which are marked by x-s; (a) contains 5 maximal points, (b) 1 maximal point and (c) 3 maximal points. Figures (b), (c), (d), (f) and (h) all have upper-right-hand-corners (marked with a large point); the other figures don't have a corner. Figures (c) and (d) have horizontal left tangents; figures (c) and (h) have vertical down ones.

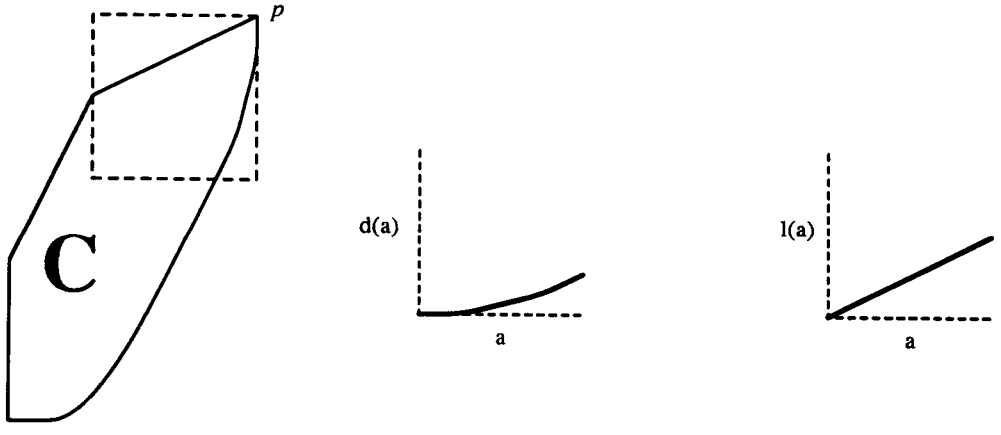


Figure 2: C has an upper-right-hand-corner p with a vertical down tangent at p and a non-horizontal left one. The middle figure portrays $d(\alpha)$, the displacement of the down boundary curve from the vertical line through p and the rightmost figure portrays $l(\alpha)$, the displacement between the left boundary curve and the horizontal line through p.

Art Gallery Problems in Integer Lattice Systems

Evangelos Kranakis[1]
Michel Pocchiola[2]

Abstract

The camera placement problem concerns the placement of a fixed number of point-cameras on the integer lattice of d-tuples of integers in order to maximize their visibility. We survey some of the combinatorial optimization and algorithmic techniques which have been developed in order to study this and other similar problems in the context of lattices and more generally, in combinations of lattice systems and tilings.

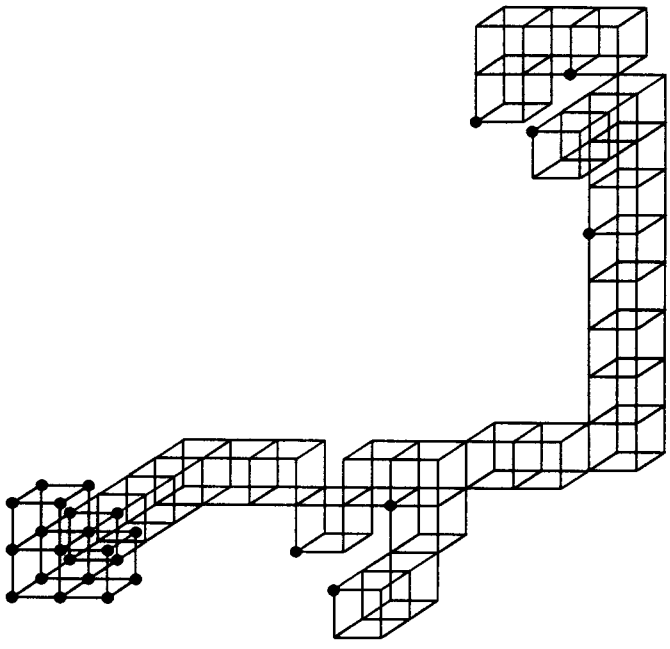


Figure 1: Optimal Configuration of 27 cameras in dimension 3

References

[1] E. Kranakis and M. Pocchiola. Enumeration and visibility problems in integer lattices. In *Proceedings of the 6th Annual ACM Symposium on Computational Geometry*, 1990.

[2] E. Kranakis and M. Pocchiola. A brief survey of art gallery problems in integer lattice systems. *CWI Quaterly*, 4(4), December 1991.

[3] M. Pocchiola. *Trois thèmes sur la visibilité : énumération, optimisation et graphique 2D.* PhD thesis, Université de Paris Sud, centre d'Orsay, Octobre 1990.

[1]Carleton University, School of Computer Science Ottawa, Canada, K1S 5B6. (kranakis@scs.carleton.ca)

[2]Laboratoire d'Informatique de l'Ecole normale supérieure, ura 1327 du cnrs, 45 rue d'Ulm, 75230 Paris Cédex 05, France. (pocchiol@dmi.ens.fr)

Determining contractibility of curves

Haijo Schipper
Department of Computing Science,
Groningen University,
P. O. Box 800,
9700 AV GRONINGEN,
The Netherlands.

Abstract

In the last few years a new branch of computational geometry *computational topology* has developed. It studies complexity of and algorithms on topological structures just as computational geometry studies geometrical structures. Topology, sometimes informally called *rubber sheet geometry*, is closely related to geometry; however, it deals more with relations than with realizations. That is one is i.e more interested in incidence relations of triangles than in the actual coordinates of the vertices.

When studying curves on topological surfaces, one of the basic questions is whether a curve contracts to a point. Stillwell calls this problem the *contractibility problem.* The problem was solved from a mathematical point of view more than a century ago, in the 1880s. The solution uses the universal covering space of the surface, and the following theorem :

Theorem 0.1 *A curve c on a 2-manifold $\mathcal{M}$ can be contracted to a point if and only if the lifted curve $\tilde{c}$ in the universal covering space $\mathcal{U}(\mathcal{M})$ is closed.*

The algorithm will use the theorem above and constructs a part of the universal covering space to determine if a lifted part is closed and hence will answer the question whether a given curve on a 2-manifold is contractable to a point or not.

The algorithm will work in $O(g^2k + gn)$ time, using $O(g^2k + gn)$ storage, where g is the genus of the surface, n the number of triangles of the surfaces, and k the number of edges of the curve.

Multi-scale analysis of discrete point sets.

Marcel Worring* Arnold W.M. Smeulders
Department of Computer Science
University of Amsterdam

Abstract

The computational aspects of the α-shape and the α-hull [2] [3] are well studied. In the reference, little attention is paid to the potential of the α-shape as a descriptor of the shape of a discrete point set. We pose a set of general criteria a shape descriptor for discrete point sets should statisfy and prove that the directed α-shape, which we call the α-graph, statisfies the posed criteria. It yields a multi-scale description of the point set, which is proved to be closely related to both the opening scale space known from mathematical morphology [1] as well as the pseudo-convex hull [5]. As an application we show the use of the α-graph in the multi-scale analysis of industrial objects.

1 Shape criteria.

Any geometrical structure to capture the shape of a point set should:

- **define a boundary:** The geometrical structure should define the contour of S. Such a contour in fact is an ordered subset of S and hence is a directional graph on S indicating a relation among the elements of S. The contour thus defined provides a local definition of the form of S, and permits derivation of contour parameters such as length, orientation and local curvature. We concentrate on a proper definition of the boundary of S.

The geometrical structure should have the following properties:

- **scale:** Any shape structure meant for general use should have a parameter regulating the scale. Such a parameter is indispensable when there is no a priori knowledge about the amount of detail of interest. To be precise, shape structures at different scales should be part of an one parameter family. In this one parameter family, the shape structure at a lower scale should be computable from a shape structure at higher scale [4].

- **minimal:** At every scale the shape description should contain the necessary boundary information of S, but not more than that.

- **increasing interior:** Any shape descriptor associates an area with the point set where additional points can be inserted not leading to a change in the shape structure. We define this area as the interior of S. The interior of S is a function of scale as defined above. As a consequence, the interior at lower scale should contain any interior at higher scale.

- **consistency with the continuous case:** For the case where the point set S is the result of a sampling from a continuous object $\mathcal{S}$, the boundary as defined on S should approach the boundary of $\mathcal{S}$ when the sampling is infinitely dense on $\mathcal{S}$.

*This work was supported by the SPIN projects "3D Computer Vision" and "3D Image Analysis."

2 Properties of the α-graph.

The following properties of the α-graph, unless otherwise stated, are proved in [6].

- **scale:** The number of α-extreme elements decreases with increasing α [2]. We express the amount of detail of the α-graph as the number of α-extreme elements. Then the α-graph defines a family depending on the single parameter α. In this family we have that if $\alpha'' > \alpha'$ then the α-graph for $\alpha = \alpha''$ can be computed from the α-extreme elements found for $\alpha = \alpha'$.

- **minimal:** The α-extreme of S are a necessary and sufficient point set for computing the α-graph of S. Therefore, the description is minimal. This immediately gives the clue to the observation in [2] that "Different values of α give rise to hulls that have only in some sense *essential* points on their boundary".

- **increasing interior:** The interior corresponding to the α-graph (as defined in section 1), is the α-hull. The area enclosed by the α-hull increases with increasing α. This was observed in [2] but not proved.

- **consistency with the continuous case:** The boundary of a continuous set $\mathcal{S}$ is for sufficiently small α equal to the α-hull. As a consequence, all elements of the boundary are elements of the α-extreme set of $\mathcal{S}$. Therefore, in the limit case of infinitely dense sampling, the α-graph of any sampling S converges to the boundary of $\mathcal{S}$.

- **relation with mathematical morphology:** The α-hull is equivalent to the opening scale space as defined in [1]. In fact, the α-hull is the closing of X with a disc of radius $-1/\alpha$. However, the α-hull is a more general notion, as it is defined for both negative and positive values of α. The scale space in the reference is equivalent to the case of negative α only. The case of positive α is equivalent to the pseudo-convex hull as defined in [5]. So, the α-hull combines the openings scale space and the pseudo-convex hull into an one parameter family.

The preceding properties show that the α-graph satisfies all the shape criteria raised in section 1.

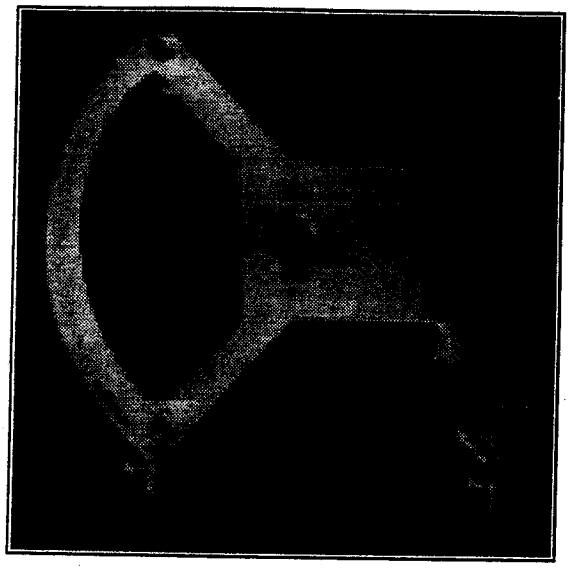


Figure 1: This shows the original image of an object taken from the Granfield benchmark.

3 Applications.

A task for which the α-graph is particularly suited is the recognition of industrial objects. The recognition is clearly multi-scale in character.

To demonstrate the performance of the method, we selected an image of one of the objects from the Granfield benchmark (see figure 1).

This image was thresholded, using an automatic thresholding algorithm, resulting in a binary image. For clarity the binary image was sampled at reduced resolution (see figure 2(a)). From there the α-graph was computed for different values of α. The ones showing detail of interest are selected (see figure 2(a-f)) for display in this paper.

In this case we used a regular grid, but this is not essential for the method. In fact, one can choose the sampling such that in highly detailed areas, sampling is more dense than in homogeneous areas.

In the first figure, the image is available in full detail as a collection of individual points. With increasing α smallest holes disappear first, related to the size of the hole with respect to the radius of the generalized disc. The larger holes also disappear and for $\alpha = 0$ the convex hull is reached. Finally elements are revealed, which define the global

shape of the whole object.

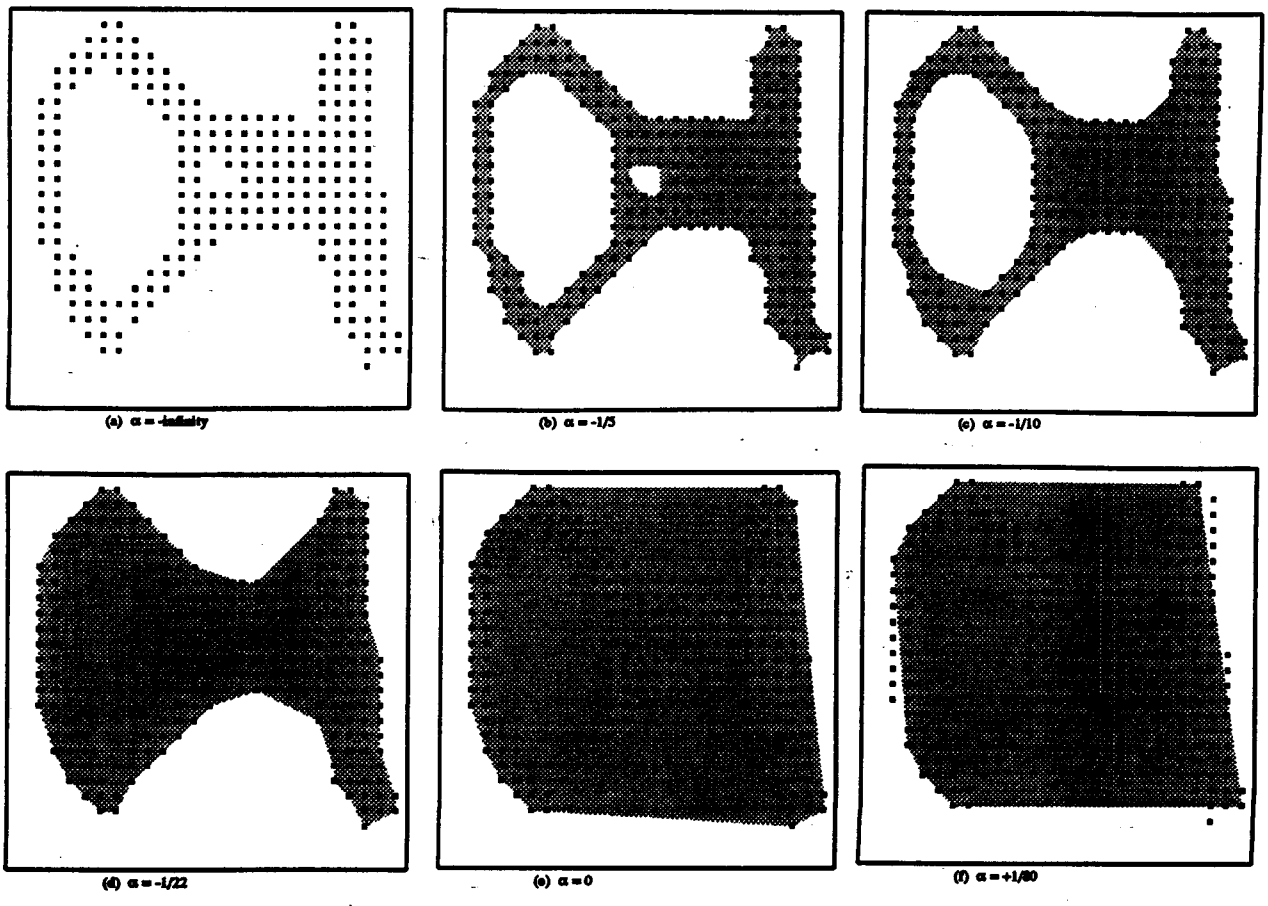


Figure 2: The α-graph is shown for increasing α. The direction of the edges is used to define an enclosed area. It is important to note that this area is *not* the interior of the point set, because that is the α-hull. Further note that the extreme elements are the elements on the boundary of this enclosed region.
For α sufficiently small the α-graph is equal to the point set S (a). In (b) the two largest holes in the objects are visible. Only one of these holes remains at a larger scale (c). An area containing no holes remains in (d). For $\alpha = 0$ the enclosed region is equal to the convex hull of S and at this specific scale the region is equal to the interior of the point set.
When $\alpha > 0$ a small set of extreme elements remain, sufficient for computation of features as global orientation etc.

4 Conclusions.

For the shape analysis of a sparse point set it is important to have a proper definition of the boundary of the point set. This definition of the boundary is not unique and must be a function of scale. At every scale it should give a minimal description. Further, the boundary should have an interior increasing with scale and its definition should be consistent with the continuous case. The α-graph satisfies all these criteria.

On the α-graph boundary features like length and curvature can be defined. A proper

definition of boundary features such that they are consistent with the continuous case is subject of further study.

Applications of the α-graph are found in areas where solid objects are studied which have a multi-scale character. Using an example from industrial vision it was shown that the α-graph has great potential in the multi-scale analysis of industrial objects.

References

[1] M Chen and P Yan. A multiscaling approach based on morphological filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):694–670, 1989.

[2] H. Edelsbrunner, D.G.Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transanctions on Information Theory*, 29(4):551–559, 1983.

[3] Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, 1987.

[4] J.J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

[5] P.W. Verbeek and F.C.A. Groen. Convex pseudo-hull algorithm for two and more dimensions. In *Proceedings of the Helsinki Converence on Image Processing and Pattern Recognition*, 1981.

[6] M. Worring and A.W.M. Smeulders. The shape of a discrete point set. Submitted for publication in "IEEE Transactions on Pattern Analysis and Machine Intelligence".

A Unified Approach to Dynamic Point Location, Ray Shooting, and Shortest Paths in Planar Maps

Roberto Tamassia
Brown University
Dept. of Computer Science
Providence, RI 02912-1910
USA

We describe a new technique for dynamically maintaining the trapezoidal decomposition of a planar map M with n vertices, and apply it to the development of a unified dynamic data structure that supports point-location, ray-shooting, and shortest-path queries in M. The space requirement is $O(n \log n)$. Point-location queries take time $O(\log n)$. Ray-shooting and shortest-path queries take time $O(\log^3 n)$ (plus $O(k)$ time if the k edges of the shortest path are reported in addition to its length). Updates consist of insertions and deletions of vertices and edges, and take $O(\log^3 n)$ time (amortized for vertex updates). (Joint work with Y.-J. Chiang and F.P. Preparata.)

A Sweepline Algorithm for Generalized Delaunay Triangulations and a Simple Method for Nearest-Neighbour Search [1]

(Extended abstract)

Sven Skyum
Computer Science Department, Aarhus University, Denmark

We give a deterministic $O(n \log n)$ sweepline algorithm to construct the generalized Delaunay triangulation. The algorithm uses no transformations and it is developed solely from the sweepline paradigm together with greediness. A generalized Delaunay triangulation can be based on an arbitrary, strictly convex Minkowski distance function (including all L_p distance functions for $1 < p < \infty$) in contrast to ordinary Delaunay triangulations which are based on the Euclidean distance function.
A direct application of the algorithm is a construction, in one phase only, of a structure for nearest-neighbour search.

Definitions and notation

The coordinates of a point $p \in \Re^2$ are denoted $(x(p), y(p))$.

For three points $p, q, r \in \Re^2$, $LT(p,q,r)$, $(RT(p,q,r))$ is true if p, q and r form a left (right) turn.

For a set M in $\Re^2$, ∂M denotes the boundary of M.

Finally $\overline{pq}$ denotes the line segment between two points p and q in $\Re^2$.

A pseudo unit disk with centre $(0,0)$ is a compact strictly convex set U with smooth boundary such that $(0,0)$ is an internal point of U. The *family of pseudo disks* given by a pseudo unit disk U is $\{c + R \cdot U \mid c \in \Re^2, R \in \Re\}$. The pseudo disk $c + R \cdot U$ is said to have centre c and radius R.

The *Minkowski distance* between p and q $(d(p,q))$ is the radius of the pseudo disk with centre p and q on its boundary. That is, $q \in \partial(p + d(p,q)U)$. Note that $d(\cdot,\cdot)$ is not necessarily symmetric.

Let a family of pseudo disks be fixed in the following. When we among other things refer to *distance*, *disk*, *cocircular*, they will always refer to the given set of pseudo disks.

For three different non-colinear points $p, q, r \in \Re^2$, D_{pqr} denotes the unique disk with p, q, r on its boundary, c_{pqr} denotes the centre of D_{pqr}, R_{pqr} denotes the radius, and t_{pqr} denotes the unique point in D_{pqr} with maximal y-coordinate.

Let $S = \{p_1, p_2, \cdots, p_n\}$ be a set of sites in $\Re^2$. We assume that $y(p_1) < y(p_2) < \cdots < y(p_n)$.

For a subset M of S and a disk D, $Empty(D, M, s)$ denotes that M is a subset of ∂D, $D \setminus \partial D$ contains no points from S, and no points in D have y-coordinate greather than s.

[1] This research was (partially) supported by the ESPRIT II Basic Research Actions Program of the EC under contract No. 3075 (project ALCOM).

Formally

$$Empty(D, M, s) \equiv (M \subset \partial D) \wedge ((D \setminus \partial D) \cap S = \emptyset) \wedge (D \subset Below_s),$$

where $Below_s$ denotes the closed half plane $\{(x, y) \mid y \leq s\}$.
All graphs involved in this paper will be planar. Hence the notion *graph* will be used to mean both an undirected graph in the normal sense and a *straight line embedding* of it.

A generalized Delaunay triangulation can be defined as follows:

For a set of sites $S = \{p_1, p_2, \cdots, p_n\}$ in $\Re^2$, the *generalized Delaunay triangulation of S* is the graph $G = (S, E)$, where $E = \{\{p_i, p_j\} \mid$ there is a disk D where $Empty(D, \{p_i, p_j\}, \infty)\}$.

Sketch of the method

The paradigm on which the algorithm is based is, apart from the sweepline paradigm, that it is greedy.

A sweepline $l_s = \{(x, y) \mid y = s\}$ will be moved upwards (increasing s). At any time a part of the Delaunay triangulation below the sweepline will have been constructed. It will be given by $G_s = (S_s, E_s)$ where $S_s = S \cap Below_s$ and $E_s = \{\{p_i, p_j\} \mid$ there is a disk D where $Empty(D, \{p_i, p_j\}, s)\}$. It turns out that G_s is allways connected.

Let $p_1 = q_1, q_2, \cdots, q_k = p_1$ be the sites on the boundary of the outer region of G_s in clockwise order. Note that there can be several instances of sites from S in the sequence.

When a new site p_i is added (a site event), that is $s = y(p_i)$, we can identify a site p_j $(j < i)$ such that $\{p_i, p_j\}$ is the edge to be added to $E_{y(p_i)-\epsilon}$ to obtain $E_{y(p_i)}$. If, for $y(p) > y(q)$, we define the v-distance between p and q, $v{-}d(p, q)$ to be the radius of the disk with top point p and q on its boundary, p_j will be the site minimizing $v{-}d(p_i, q)$ over $\{p_1, p_2, \cdots, p_{i-1}\}$. A key observation for the method to work is that p_j can be found by binary search in $\{q_1, q_2\}, \{q_2, q_3\}, \cdots, \{q_{k-1}, q_k\}$ using the test $LO(\{q_l, q_{l+1}\}, p_i)$ which is true iff p_j is to be found among $\{q_1, q_2, \cdots, q_l\}$. $LO(\{q_l, q_{l+1}\}, p_i) \equiv$

$$[(y(q_l) < y(q_{l+1})) \wedge LT(q_l, q_{l+1}, p_i) \wedge (v{-}d(p_i, q_l) < v{-}d(p_i, q_{l+1}))] \vee$$
$$[(y(q_l) > y(q_{l+1})) \wedge (RT(q_l, q_{l+1}, p_i) \vee (v{-}d(p_i, q_l) < v{-}d(p_i, q_{l+1})))]$$

New edges will be added to G_s (top point events) during the sweep without new sites encountered if a disk with three sites on its boundary that intersects the $l_{s-\epsilon}$ becomes situated entirely below l_s. The other key observation for the method to work is that the only disks to examine for this to happen are those given by consecutive sites (q_{i-1}, q_i, q_{i+1}) on the outer region of $G_{s-\epsilon}$ where $LT(q_{i-1}, q_i, q_{i+1})$.

Data structures for G_s, the sites on the boundary in clockwise order, and top points for disks through three consecutive sites on the boundary forming a left turn, are not difficult to construct so that they support an $O(n \log n)$ running time for the algorithm. It is assumed that $y(t_{pqr})$ and $v{-}d(p, q)$ can be computed in constant time.

Nearest-neighbour search

The described method is also applicable for pseudo disks which are not strictly convex. Then three points do not uniquely determine a disk. It might happen that no such disk exists or more than one exist. In the former case we can think of the disk having infinite radius and infinite y-coordinate of the top point. In the latter case we can use a limit argument to define a canonical one which will do.

Therefore the method can be applied to "half" disks. A (lower) "half" disk with centre c and radius r is the set of points p in $\Re^2$ with distance at most r to c and where $y(p) \leq y(c)$. The corresponding v-distance, $v{-}d(p,q)$, then becomes the original distance between p and q. Thus when we determine the site with minimal v-distance during the sweep, we in fact just determine the site of minimal distance situated below the new site added. By doing the same once more for the set of sites turned upside down, we get an analogous structure for search for the nearest neighbour above the query point. Thereby the all-neariest-neighbour problem can be solved. This special use of the algorithm gives an analogous method to the method presented in [2]. Here it is just generalized to arbitrary distance functions. By doing the search structures persistent, a structure (of linear size) supporting search for the nearest neighbour in time $O(\log n)$ is obtained.

References

[1] J.R. DRISCOLL, N. SARNAK, D.D. SLEATOR, R.E. TARJAN *Making Data Structures Persistent* Journal of Computer and System Sciences (1989) 38, 86 - 124

[2] K. HINRICHS, J. NIEVERGELT, P. SCHORN *A sweep algorithm for the all-nearest-neighbors problem* Computational Geometri and its applications, LNCS (1988) 333, 43 - 54

[3] S. SKYUM *A sweepline Algorithm for Generalized Delaunay Triangulations* Tech. Rep. DAIMI Pb-373 (1991)

The Space of Spheres, a Geometric Tool to Unify Duality Results on Voronoi Diagrams *

Abstract

Olivier Devillers† Stefan Meiser‡ Monique Teillaud†

Abstract

We show how duality results for generalized Voronoi diagrams can be proven by only using purely geometric interpretations, without any analytic calculations.

1 Introduction

[Bro79, ES86] presented the first results of duality for Voronoi diagrams. Many geometric transforms for generalized diagrams are now known (see [Aur91] for a very complete survey) and they have found many applications.

We do not claim to give new results, but we introduce a tool allowing to have a global view and, above all, a better understanding of a very large number of duality results.

For example, F. Aurenhammer [Aur87] introduces the same duality as ours, between spheres and points, for power diagrams, but he gives no geometric interpretation to this, and his justifications are only by analytic computations. We can give new proofs, avoiding all analytic calculations, with only geometric reasoning.

We use here a geometric interpretation of spheres as points in the *space of spheres*, which is a very powerful tool, and will probably find many applications in the future, since it allows to deal with some very general problems in a simple way (see the full paper [DMT92] for example for the case of weighted order k power diagrams, or Voronoi diagrams of general manifolds). We can also deduce algorithmic application in some cases.

We can apply our framework to different types of Voronoi diagrams, for example the hyperbolic metric (see [BCDT91] for the planar case).

It can also be used to solve problems on circles or spheres : for example, the determination of the ring with minimal surface, defined by two cocircular circles, containing a given set of points transforms in the space of spheres into a linear programming problem.

The necessary mathematical background is summarized in the full paper.

†INRIA, 2004 Route des Lucioles, B.P.109, 06561 Valbonne cedex (France), Phone : +33 93 65 77 62, E-mail : odevil or teillaud@alcor.inria.fr. This work was partly done while these authors were visiting Max Planck Institut

‡Max Planck Institut für Informatik, D-6600 Saarbrücken, Phone : +49 681 302-5356, E-mail : meiser@mpi-sb.mpg.de

* *This work has been supported in part by the ESPRIT Basic Research Action Nr. 3075 (ALCOM).*

2 The Space of Spheres

We denote by $\mathcal{P}$ the euclidean d dimensional space. $\langle .,. \rangle$ is the scalar product in $\mathcal{P}$, and $|..|$ is the euclidean distance between points. A point or a vector $(x_1, \ldots, x_i, \ldots, x_d)$ of $\mathcal{P}$ will be represented as x. Let us recall the definition of the *space of spheres* σ used in [BCDT91]. Let

$$(S) \qquad \langle x, x \rangle - 2\langle x, \Phi \rangle + \chi = 0 \tag{1}$$

be the equation of a sphere S in $\mathcal{P}$. Here Φ is a point of $\mathcal{P}$, namely the center of S. This sphere is represented by the point $S = (\Phi, \chi)$ in the $(d+1)$ dimensional space σ. $\mathcal{P}$ is identified to the d-hyperplane $\chi = 0$ of σ. Notice that the vertical projection onto $\mathcal{P}$ of a point $S = (\Phi, \chi) \in \sigma$ is the center of the sphere, and the radius r_S of S is given by :

$$\text{power}(O, S) = \chi = \langle \Phi, \Phi \rangle - r_S^2 \tag{2}$$

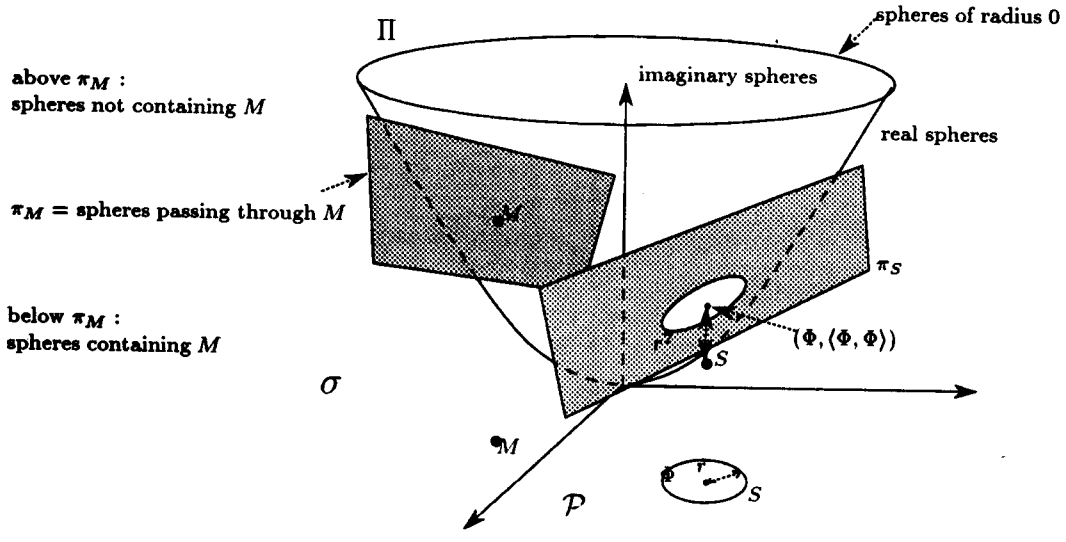


Figure 1: The space of spheres

We present in this section a few properties of σ.

The paraboloid Π. Equation (2) shows that the *point-spheres* in σ (spheres of radius 0) verify :

$$(\Pi) \qquad \langle \Phi, \Phi \rangle - \chi = 0 \tag{3}$$

Thus, the set of point-spheres is in σ the unit d-paraboloid of revolution with vertical axis (that is, parallel to the χ-axis) denoted as Π. A point (Φ, χ) on Π represents a sphere of $\mathcal{P}$ with center Φ and radius 0. Therefore points of $\mathcal{P}$ are associated with the corresponding point-spheres on Π in σ. If space $\mathcal{P}$ is identified to the d-hyperplane $\chi = 0$, the point-sphere is obtained by raising the point on Π. The exterior of Π is the set of real spheres, whereas its interior is the set of *imaginary* spheres (with negative square radius) (Figure 1). The square radius of a sphere $S = (\Phi, \chi)$ is the difference of the χ-coordinate of the vertical projection of S on Π with χ.

Hyperplanes in σ. Two spheres of $\mathcal{P}$ are orthogonal if and only if the two corresponding points of σ are conjugate with respect to Π. Thus the set of spheres of $\mathcal{P}$ orthogonal to

a given sphere $S_0 = (\Phi_0, \chi_0)$ of $\mathcal{P}$ forms in σ exactly the polar d-hyperplane π_{S_0} of point S_0 with respect to Π. Its equation is obtained by polarizing the equation of Π in S_0 :

$$(\pi_{S_0}) \qquad \chi = 2\langle \Phi_0, \Phi \rangle - \chi_0 \tag{4}$$

It is the polar d-hyperplane of the sphere with respect to Π.

The intersection of π_{S_0} and Π is precisely the set of point-spheres that are orthogonal to sphere S_0, that is the set of points lying on S_0 in $\mathcal{P}$. This shows that $\pi_{S_0} \cap \Pi$ projects in $\mathcal{P}$ onto S_0 (see Figure 1).

As a particular case, the set of spheres passing through a point $M \in \mathcal{P}$ is also the set of spheres orthogonal to the point-sphere M. But, as M belongs to Π, the polar d-hyperplane π_M is nothing else than the tangent d-hyperplane to Π at M. Each sphere in the lower half space limited by π_M (i.e. the half space which does not contain Π), contains M in its interior. For a sphere in the upper half space, M is outside (see Figure 1).

3 Duality results

We can apply our framework to different kinds of Voronoi diagrams : (weighted) power diagrams, affine Voronoi diagrams, (weighted) order k power diagrams, Voronoi diagrams of general manifolds, hyperbolic Voronoi diagrams. The resulting transformation is often already well known, however, using our framework, the transformation is straightforward and does not require any calculus. It may also lead to new efficient algorithms (see the full paper).

$\mathcal{S}$ denotes a set of spheres. An element of $\mathcal{S}$ is denoted by S or S_i. In this abstract, we only develop the simple case where $\mathcal{S}$ is a set of point-spheres. The distance we consider here is the euclidean distance in $\mathcal{P}$:

$$\delta(P, Q) = |PQ|$$

As noticed in Section 2, the set of spheres which do not contain a given point maps in σ into the upper half space, limited by the polar d-hyperplane of the corresponding point. Thus, if $\mathcal{S}$ is a set of sites in space $\mathcal{P}$, then the set of empty spheres (i.e. the set of spheres which do not surround any site of $\mathcal{S}$) of space $\mathcal{P}$ is in σ the intersection of the corresponding half spaces. Its boundary is a convex polytope $U_{\mathcal{S}}$, whose facets are tangent to Π.

For a given point $M \in \mathcal{P}$ and a site $S \in \mathcal{S}$, the intersection of the vertical line through M with π_S gives the maximum empty sphere centered at M and touching S (see Figure 2). Since the radius of the spheres on the vertical line increases with falling χ, we have :

$\mathcal{P}$ 1 The intersection of $U_{\mathcal{S}}$ with a vertical line $\Phi = M$ in σ gives the sphere with center M and whose radius is the distance to the nearest neighbor of M in $\mathcal{S}$.

In other words the projection of $U_{\mathcal{S}}$ on $\mathcal{P}$ is the Voronoi diagram of $\mathcal{S}$. This can be viewed as a new presentation of the well known correspondence between intersection of half spaces tangent to the d-paraboloid and Voronoi diagram [ES86].

Property ($\mathcal{P}$ *1*) allows to determine the extremal empty spheres of a concentric pencil, but it can also be used for any kind of pencils : a pencil of spheres in $\mathcal{P}$ is represented by a

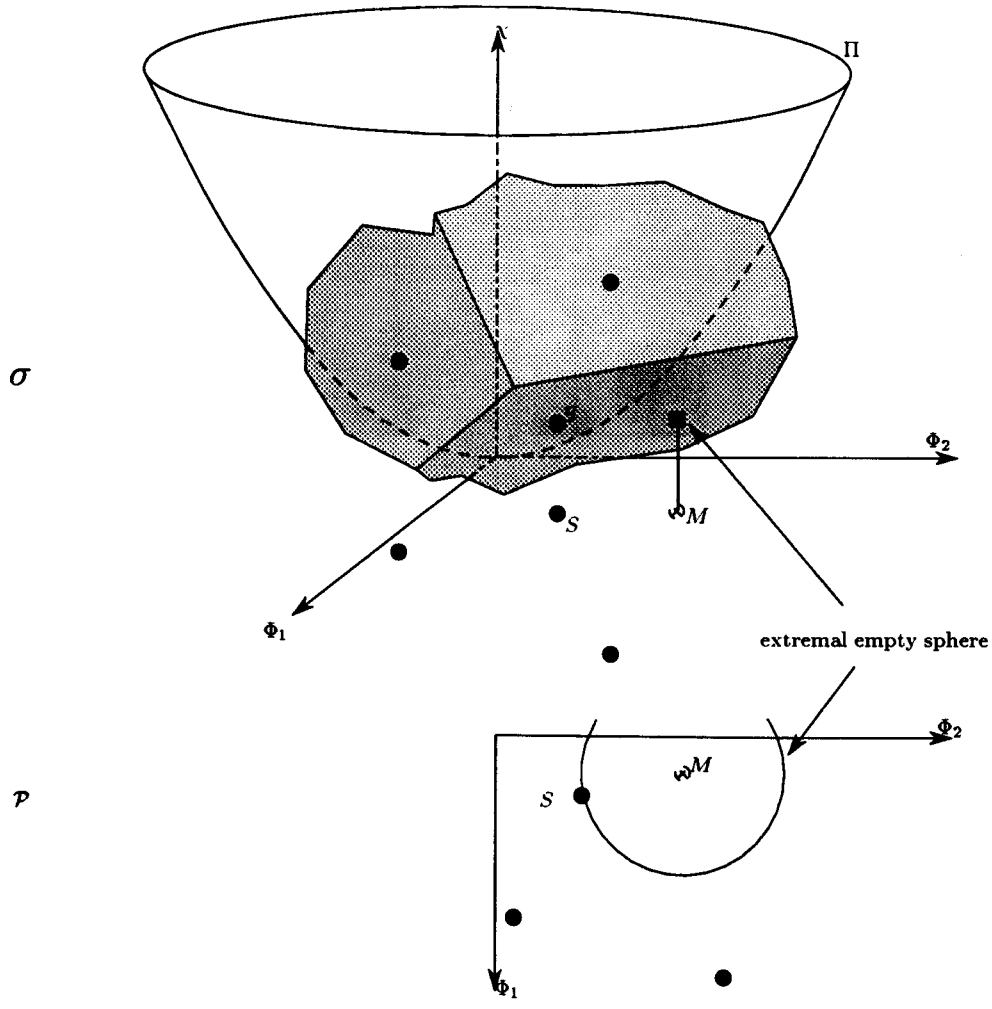


Figure 2: Empty sphere centered on M and touching S

line in σ, and the intersection of U_S with a line representing a pencil of spheres, is made up of the extremal empty spheres of this pencil.

This correspondence has been used in [BCDT91] to compute the 3D Delaunay triangulation of a set of sites lying in k different subspaces, with an output sensitive complexity.

We can generalize this approach to other kinds of sites and distances to handle all cases of generalized Voronoi diagrams [DMT92].

References

[Aur87] F. Aurenhammer. Power diagrams : properties, algorithms and applications. *SIAM Journal on Computing*, 16:78–96, 1987.

[Aur91] F. Aurenhammer. Voronoi diagrams — a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3), September 1991.

[BCDT91] J-D. Boissonnat, A. Cérézo, O. Devillers, and M. Teillaud. Output sensitive construction of 3D Delaunay triangulation of constrained sets of points. In *Third Canadian Conference on Computational Geometry in Vancouver*, August 1991. Full paper available as Technical Report INRIA 1415.

[Bro79] K.Q. Brown. Voronoi diagrams from convex hulls. *Information Processing Letters*, 9:223–228, 1979.

[DMT92] O. Devillers, S. Meiser, and M. Teillaud. *The Space of Spheres, a Geometric Tool to unify Duality Results on Voronoi Diagrams.* Technical Report 1620, Institut National de Recherche en Informatique et Automatique, (France), February 1992.

[ES86] H. Edelsbrunner and R. Seidel. Voronoi diagrams and arrangements. *Discrete and Computational Geometry*, 1:25–44, 1986.

An algorithm for constructing the convex hull of a set of spheres in three dimensional space

J.D. Boissonnat* A.Cérézo† J. Duquesne‡

Abstract

Using the "gift wrapping" approach, the algorithm computes the convex hull of a set of n spheres in three dimensional space. If h is the size of the output (i.e. of the convex hull), the algorithm's worst case running time is $O(nh)$.

1 Introduction

In the plane, the convex hull of a set of n circles can be deduced from the Voronoi diagram of these circles, which can be computed in $O(n \log n)$ time and $O(n)$ space (see [Ros88]).

In three dimensional space, the only known result is that the complexity of the convex hull of a set of n spheres is $\theta(n^2)$ in the worst case, even for collections of pairwise disjoint spheres (see [SS90]).

The convex hull of a set E of spheres is composed of three different kinds of *cells* (see Figure 1).

- Planar cells, which are triangles included in planes tangent to three spheres.
- Conical cells, which are parts of cones tangent to two spheres.
- Spherical cells, which are parts of the spheres of E.

To each point P of a cell, we associate the half line issued from P and normal to the tangent plane of the cell at P. To each cell we associate a three dimensional *region* which is the union of these half lines. The union of all the regions partition the space outside the convex hull.

*INRIA, 2004 Route des Lucioles, B.P.109, 06561 Valbonne cedex (France), Phone : +33 93 65 77 60, E-mail : boissonn@sophia.inria.fr.

†Université de Nice, Parc Valrose, 06034 Nice cedex (France).

‡INRIA, 2004 Route des Lucioles, B.P.109, 06561 Valbonne cedex (France), Phone : +33 93 65 77 49, E-mail : duquesne@sophia.inria.fr.

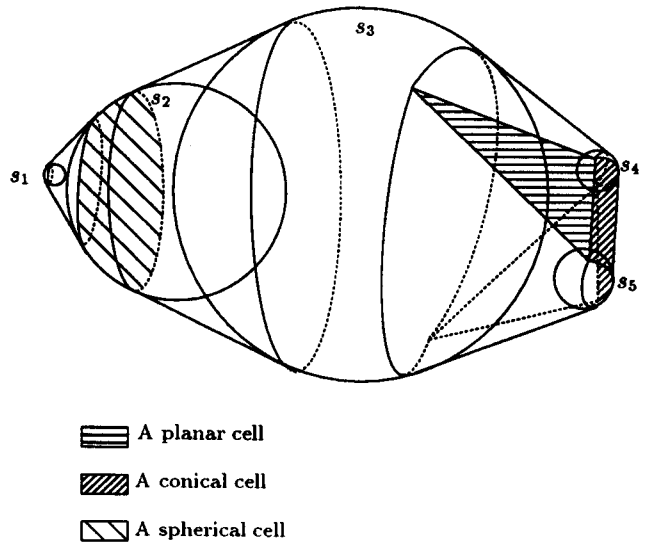


Figure 1: The convex hull of a set of spheres

2 The Algorithm

Let h be the size of the output (i.e. of the convex hull). h is $\theta(n^2)$ in the worst case.

Using the gift wrapping approach, we can compute the convex hull of the set E of spheres in worst case time $O(nh)$.

We associate to the convex hull a *three-dimensional graph* G_{CH} where a vertex stands for a spherical cell, an edge for a conical cell, and a face for a planar cell.

Let a *disconnecting vertex or edge* be a vertex or an edge such that if one cuts the graph G_{CH} at this vertex or along this edge, the graph G_{CH} is separated into γ *disjoint connected components.* For example, in Figure 1, the vertices associated with s_2 and s_3 are disconnecting, and so are the edges associated with the conical cells $s_1 - s_2$ and $s_2 - s_3$.

Let's cut G_{CH} along *all* its disconnecting vertices or edges. We separately compute the convex hull of each of the resulting connected components.

During the algorithm, we maintain two structures:

- A *graph* G representing the part of the convex hull which has already been computed.
- A list of *free edges*, i.e. the set of edges where only one of the two faces bounding the edge has already been computed.

To compute a connected component of G_{CH}, we do the following:

- We first search three spheres (if any) belonging to the convex hull which give us a face and three free edges.
- We then recurse on the next two steps until the list of free edges is empty.

- Take a free edge and compute the face bounding this edge: Let S_1S_2 be the two spheres associated with the edge and $S'S_1S_2$ the face (bounding the edge) which has already been computed. For each sphere S different from S_1 and S_2, we compute the plane tangent to the spheres S, S_1, S_2 (if it exists). We finally choose the sphere S such that the angle between the planes $S'S_1S_2$ and SS_1S_2 is closest to π.
- Update the graph G and the list of free edges.

Once we have computed one component of G_{CH}, each remaining sphere s (i.e. not belonging to the connected component already computed) either is inside the currently computed convex hull or can be located in only one region (depending of the sphere s) of the convex hull.

We separate the remaining spheres in different sets, so that all the spheres of a same set can be located in the same region. We now recursively compute the convex hull of each set.

3 Complexity Analysis

Let $T(n,h)$ be the time needed to compute the convex hull of a set of n spheres in the worst case. h is the size of the output.

If the graph G_{CH} has no disconnecting vertex or edge, then

$$T(n,h) = O(nh)$$

Else, the first connected component is composed of n_0 spheres and the size of its convex hull is h_0. The $n - n_0$ remaining spheres are separated into p sets of $n_i (1 \leq i \leq p)$ spheres as explained above. The size of the convex hull of each set is $h_i (1 \leq i \leq p)$. Thus we have:

$$T(n,h) = O(nh_0) + \sum_{i=1}^{p} T(n_i, h_i)$$

and

$$\sum_{i=0}^{p} n_i = n \qquad \sum_{i=0}^{p} h_i = h$$

The solution of the above recurrence is $T(n,h) = O(nh)$.

Thus, *the whole time needed to compute the convex hull of a set of spheres in three dimensional space is $O(nh)$ in the worst case.*

References

[Ros88] H. Rosenberger. *Order-k Voronoi Diagrams of Sites with Additive Wieghts in the Plane* . UIUCDCS-R 1431, University of Illinois at Urbana-Champaign, May 1988. This paper is a shortened version of the Master Thesis with the same title.

[SS90] J.T. Schwartz and M. Sharir. On the two-dimensional Davenport-Schinzel problem. *Journal of Symbolic Computation.* 10:371-393, 1990.

Least-Squares Partitioning (Abstract)

Franz Aurenhammer
Institut für Informatik
Freie Universität Berlin
Germany

Friedrich Hoffmann
Institut für Informatik
Technische Universität München
Germany

Boris Aronov
Computer Science Department
Polytechnic University
New York

Consider a set S of n points, called *sites*, in the Euclidean plane. S induces a partition of the plane into n polygonal regions in the following natural way. The region of a site $s \in S$, $reg(s)$, consists of all points x which are closer to s than to the remaining $n-1$ sites. This partition is known as the Voronoi diagram of S. If we fix a set X of m points in the plane, this set is partitioned by the Voronoi diagram of S into subsets $X \cap reg(s)$, called *clusters*.

Given the sets S and X, we would like to be able to change the induced clustering by varying the distance function that underlies the Voronoi diagram of S. To this end, we attach a set $W = \{w(s) \mid s \in S\}$ of real numbers, called *weights*, to the sites and replace the Euclidean distance $\delta(x,s)$ between a point x and a site s by the power function

$$pow_W(x,s) = \delta^2(x,s) - w(s).$$

The resulting partition of the plane is known as the *power diagram* of S with weights W. Each region is still a convex polygon, and has the property of shrinking (expanding) when the weight of its defining site is decreased (respectively, increased). The corresponding clustering of X now depends on the particular choice of weights.

The concepts introduced above extend to arbitrary dimensions in a straightforward manner. We can show the following:

Theorem 1: For any set S of n sites there exists a power diagram whose regions partition a given d-dimensional finite point-set X into n clusters of prescribed sizes.

Note that the power diagram of S with weights W defines an *assignment function* $A_W : X \rightarrow S$, given by

$$A_W(x) = s \iff x \in reg_W(s),$$

where $reg_W(s)$ denotes the region of site s in this diagram. For each site $s \in S$, its cluster is given by $A_W^{-1}(s)$. We may associate each s with an individual number $c(s) \in N$, called its *capacity*. Theorem 1 tells us that for any choice of site capacities there is a set W of weights for S such that the induced clusters $A_W^{-1}(s)$ have sizes $c(s)$.

The assignment function A_W has the remarkable property that it minimizes – for all possible assignments $X \rightarrow S$ satisfying the same capacity constraints as A_W does – the sum of the squares of the distances between sites and their assigned points. Such an assignment will be called a *least-squares assignment* subject to the capacity function c. More generally, the following will be shown:

Theorem 2: Let S be a set of n sites in E^d. Any assignment induced by a power diagram of S is a least-squares assignment, subject to the resulting capacities. Conversely, a least-squares assignment for S subject to any given choice of capacities can be realized by a power diagram of S.

The second part of Theorem 2 implies Theorem 1. By Theorem 2, a least-squares assignment subject to any given capacity constraints can be computed by finding weights W such that A_W satisfies these constraints.

To illustrate the usefulness of the concept of a constrained least-squares assignment $L : X \to S$, we mention some of its properties.

(1) The induced clusters $L^{-1}(s)$, $s \in S$, are pairwise *convex-hull disjoint.* Had we chosen to minimize a different function, such as the *sum* of Euclidean distances to sites, rather than the sum of squared distances, the optimum assignment would not be guaranteed to have the disjointness property. Hull-disjointness of clusters is desirable as, for example, it eases the classification of new points.

(2) It can be shown that solving a certain transportation problem is equivalent to finding least-squares clusters of prescribed sizes.

(3) L is invariant under translation and scaling of S. This property is useful, for example, for finding the best *least-squares fitting* of two n-point sets S and X if translation and scaling is allowed. It ensures that the least-squares matching between these sets ($c(s) = 1$ for all $s \in S$) already coincides with the bijection in the optimal fitting. Given the bijection, however, it is an easy task to find the optimizing translation vector and scaling factor.

Exploiting the machinery of power diagrams, we propose two algorithms for computing constrained least-squares assignments. The first algorithm proceeds incrementally. It starts with the Voronoi diagram of the n sites (all weights are zero) and with no points, and inserts the points one by one, while adjusting at each phase the weights such that the site capacities are not exceeded. In the plane, a time complexity of $O(n^2 m \log m + nm \log^2 m)$ and an optimal space complexity $O(m)$ are achieved. (m is the number of points inserted; we may assume $m \geq n$ as attention can be restricted to sites with non-zero capacities.) This is an improvement over the best known deterministic algorithm [1] that achieves time $O(nm^2 + m^2 \log m)$ and space $O(nm)$ by transformation into a minimum cost flow problem. We mention that the randomized algorithm of Tokuyama and Nakano [2] achieves expected time $O(nm + n^3\sqrt{nm})$ and space $O(nm)$; their algorithm is more general than ours in that it finds the optimum constrained assignment for any cost function, not only the square of the Euclidean distance.

The second algorithm relies on the following interesting fact: Finding a weight vector W such that A_W satisfies the capacity constraints is equivalent to finding a maximum of a concave piecewise-linear n-variate function whose domain is the weight space. We propose a gradient method for iteratively improving the weight vector; it is guaranteed to terminate after a finite number of steps. Experiments have shown that it can be expected to run quite fast in practice. Its space requirement is optimal, $O(m)$. We are exploring the possibility of combining the two approaches, in order to obtain a provably fast algorithm for computing a constrained least-squares assignment for a finite set of points.

We mention finally that a more general, continuous version of Theorem 1 can be proved.

Theorem 3: Let S be a set of n sites in E^d, let ϱ be some probability distribution on E^d which is zero outside $[0,1]^d$, say, and let $\mu(X) = \int_X \varrho(x)dx$ denote the measure of a set $X \subset E^d$ with respect to ϱ. For any capacity function $c : S \rightarrow [0,1]$ with $\sum_{s \in S} c(s) = 1$, there is a set W of weights such that $\mu(reg_W(s)) = c(s)$, for all sites $s \in S$.

By taking, for instance, ϱ as the uniform distribution in $[0,1]^d$ we get:

Corollary: For any set of n sites there exists a power diagram that partitions the unit hypercube into n polyhedral regions of prescribed volume.

This result is related to Minkowski's theorem for convex polytopes which, for our purposes, can be stated as follows: Choose any n vectors in E^{d+1}, no two parallel. There exists a convex polytope in E^{d+1} with $n+1$ facets, n of which correspond to the given vectors in that each face is normal to its corresponding vector and has d-dimensional volume equal to its length. In fact, such a polytope is unique up to translation. Since power diagrams in E^d are exactly the projections of *unbounded* convex polyhedra in E^{d+1} (which can be viewed as "polytopes" with the $(n+1)$-st facet at infinity in the direction of projection), the corollary implies a generalization of Minkowski's Theorem for unbounded polyhedra.

The proposed gradient method for iteratively approximating the weight vector also works for the continuous case. Its convergence rate is superlinear provided the probability distribution ϱ is continuous. The space requirement is optimal, $O(n)$.

References

[1] Fredman, M., Tarjan, R. Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM 34 (1987), 596 - 615.

[2] Tokuyama, T., Nakano, J. Geometric algorithms for a minimum cost assignment problem. Proc. 7^{th} Ann. ACM Symp. Computational Geometry (1991), 262 - 271.

Weighted Closest Pairs

Michael Formann

Institut für Informatik
Fachbereich Mathematik
Freie Universität Berlin
Arnimallee 2-6
D-1000 Berlin 33
Germany

Summary

In this talk we study the following *weighted closest pair problem*: Given a set of planar objects with centerpoints, determine the *maximal scaling factor* δ_{max}, such that the objects scaled by δ_{max} are pairwise disjoint. Clearly δ_{max} can be computed in $\mathcal{O}(n^2)$ time, by taking the minimum of all $\binom{n}{2}$ pairwise maximal scaling factors. (It is assumed, that the computation of the maximal scaling factor of two objects is a primitive operation that can be done in $\mathcal{O}(1)$ time.) The goal of this article is to beat the $\mathcal{O}(n^2)$ time bound.

We describe a method to compute the maximal scaling factor in optimal $\mathcal{O}(n \log n)$ time for a wide class of objects. The basic idea is as follows: In a preprocessing step we compute an over-estimate $\delta \geq \delta_{max}$ for δ_{max}. Clearly, if $\delta > \delta_{max}$ then there are intersections in the set of δ-scaled objects. We insure, that after the preprocessing step only a linear number of pairs of objects will intersect. We detect these intersections in $\mathcal{O}(n \log n)$ time by a standard sweep-technique and compute the related pairwise maximal scaling factors. The minimum of δ and these $\mathcal{O}(n)$ numbers will be the output δ_{max} of our algorithm.

If all the objects are unit-disks then we are faced with the ordinary *closest pair problem*, that is to determine the closest distance between any pair of points. This is a well-studied fundamental problem of computational geometry (cf. [HNS90], [HS75], [BS76]). Another related problem is finding the closest pair among a set of objects (cf. [BH91], [For87], [Sha85], [Yap87]).

In principle we could determine the maximal scaling factor via a Voronoi-Diagram approach. For example we are given a set of disks with different radii. From the center p of a disk we measure the weighted distance from p as

$$d_p(x) := \frac{|p - x|}{r_p},$$

where r_p denotes the radius of the disk around p. The bisector of two disks with centers p and q will be the set of points in the plane that have equal weighted distance to p and q. Then we could define a pair p, q as *closest pair* if it minimizes the shortest weighted distance to its bisector. Exactly the closest pairs of δ_{max}-scaled disks will touch. This approach doesn't work because the Voronoi-Diagram might have quadratic complexity (cf. [AE84]).

References

[AE84] Franz Aurenhammer and Herbert Edelsbrunner. An optimal algorithm for constructing the weighted Voronoi diagram in the plane. *Pattern Recogn.*, 17:251–257, 1984.

[BH91] Frank Bartling and Klaus Hinrichs. A plane-sweep algorithm for finding a closest pair among convex planar objects. Technical Report 91-03, Gesamthochschule Siegen, Institut für Informatik, 1991.

[BS76] J. Bentley and M. Shamos. Divide and conquer in multidimensional space. In *Proceedings 8th Annual Symp. Theory Comput.*, pages 220–230, 1976.

[For87] S. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica 2*, pages 153–174, 1987.

[HNS90] Klaus Hinrichs, J. Nievergelt, and Paul Shorn. Plane-sweep solves the closest pair problem elegantly. *Information Processing Letters*, pages 337–342, 1990.

[HS75] D. Hoey and M. Shamos. Closest-point problems. In *Proceedings 17th IEEE Annu. Symp. Found. Comput. Sci.*, pages 151–162, 1975.

[Sha85] Micha Sharir. Intersection and closest pair problems for a set of planar discs. *SIAM Journal on Computing*, 14:448–468, 1985.

[Yap87] Chee Yap. An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments. *Discrete Comput. Geom.*, 2:365–393, 1987.

On the Orders of Finitely Many Points Induced by Rotational Sweeps

Hanspeter Bieri

Universität Bern, Institut für Informatik und angewandte Mathematik, Länggassstrasse 51, CH - 3012 Bern, bieri@iam.unibe.ch

and Peter-Michael Schmidt

Friedrich-Schiller-Universität, Sektion Mathematik, Universitätshochhaus, 17. OG, D - O - 6900 Jena, PETER-MICHAEL.SCHMIDT@MATHEMATIK.uni-jena.dtp.de

Extended Abstract

Let $P = \{p_1, ..., p_n\} \subset \mathbb{R}^2$ and $\mathbf{H}(P)$ the set of all lines joining two points of P. We assume P to be *in general position* in the following sense: Two lines of $\mathbf{H}(P)$ are never parallel, three lines of $\mathbf{H}(P)$ never meet in one point. $\mathbf{H}(P)$ partitions $\mathbb{R}^2$ into relatively open *cells* of dimensions 0, 1, and 2 which form an *arrangement* $\mathbf{A}(P)$.

Let $q = (x, y)$ be a point within any one of the 2–dimensional cells i.e. not lying on any of the lines of $\mathbf{H}(P)$, τ a real number and $R(q, \tau) = \{a = (x + r\cos\tau, y + r\sin\tau); r \in \mathbb{R}\}$ a straight line passing through q. Running τ through all real numbers causes $R(q, \tau)$ to "sweep" $\mathbb{R}^2 \setminus \{q\}$, i.e. we get a *rotational sweep in* $\mathbb{R}^2$ with q as the center of rotation. The parameter τ expresses the time in the course of which the sweep takes place. Since q does not belong to any of the lines of $\mathbf{H}(P)$ a *circular ordering* on P arises (cf. [P55]).

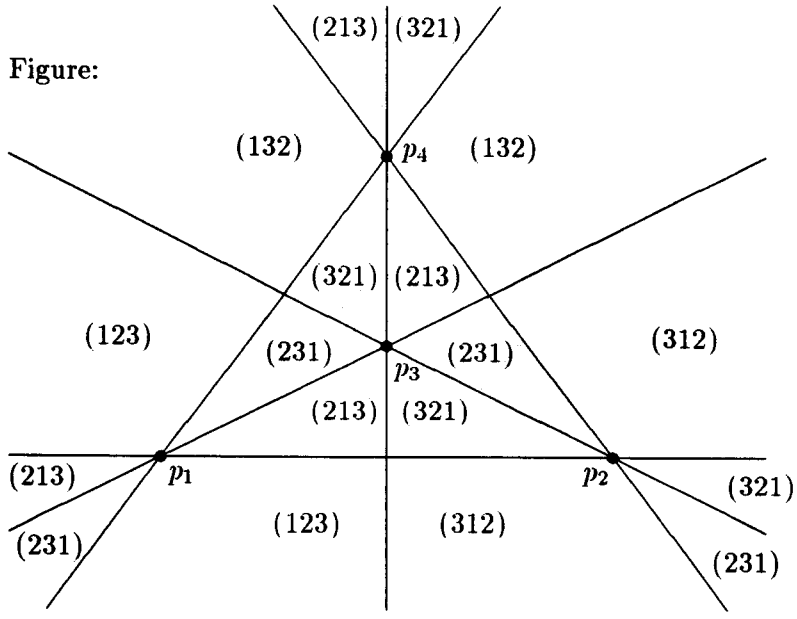
We now choose τ in the interval $[\tau_o, \tau_o + \pi)$ satisfying the condition $p_n \in R(q, \tau_o)$, in other words, the sweep starts by passing the point p_n. The rotational sweep with center q induces a *linear ordering* on the set $P^* := P \setminus \{p_n\}$ as follows: Let τ_i and τ_j satisfy the conditions $p_i \in R(q, \tau_i)$ and $p_j \in R(q, \tau_j)$, respectively. Then the order induced on P^* is defined by $p_i \ll p_j$ iff $t_i < t_j$. As the points $p_1, ..., p_{n-1}$ are associated uniquely with the indices $1, \ldots, n-1$, the rotational sweep with center q induces also a permutation π^q of the set $\{1, \ldots, n-1\}$. The number of all these permutations π^q or orders $\ll$, respectively, shall be denoted by $N(P)$.

Let E be any 2–dimensional cell of $\mathbf{A}(P)$ and $q_1, q_2 \in E$. Obviously the rotational sweeps with centers q_1 and q_2 induce the same order on P^* or the same permutation of $\{1, ..., n-1\}$, respectively, i.e. $\pi^{q_1} = \pi^{q_2}$. Furthermore let D denote the union of all 2–dimensional cells $E \in \mathbf{A}(P)$. P induces an equivalence relation on D by means of $q_1 \sim q_2$ iff $\pi^{q_1} = \pi^{q_2}$. Of course the number of equivalence classes of $\sim$ equals $N(P)$. The figure shows the arrangement $\mathbf{A}(P)$ and the equivalence classes of $\sim$ for a set $P = \{p_1, p_2, p_3, p_4\} \subset \mathbb{R}^2$. In the figure a permutation π^q of $\{1, 2, 3\}$ is indicated by $(\pi^q(1)\,\pi^q(2)\,\pi^q(3))$.

We are interested in the following two questions:

A: Given $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ in general position. Which of the $(n-1)!$ orders of $p_1, ..., p_{n-1}$ – or of the corresponding permutations of $\{1, ..., n-1\}$, respectively –

Figure:



can be induced by a rotational sweep in $\mathbb{R}^2$ i.e. by choosing a convenient center of rotation q and performing the corresponding rotational sweep ?

B: What are lower and upper bounds on the maximal number of $N(P)$ for point sets P in general position with $\text{card}(P) = n$?

Problem A can be solved by an algorithm ORDERS which makes use of a translational sweep in a similar way as the algorithm CELLS in [BN82] which finds all cells of an arrangement in $\mathbb{R}^d$:

1. All points of intersection of the lines of $\mathbf{H}(P)$ are determined, sorted with respect to their x–coordinates and put into a convenient data structure **Q**. For the sake of simplicity it is assumed that all points in **Q** have distinct x–coordinates, which can be ensured by a suitable coordinate transformation (see [NS90]).

2. The sweep line is chosen to be vertical and its initial position on the left of all points contained in **Q**.

3. For each 2–dimensional cell $E \in \mathbf{A}(P)$ which is cut by the initial sweep line, a point $q \in E$ is determined. For each center of rotation q the sweep is performed and the resulting permutation π^q is included in a list ORDS.

4. Only the points in **Q** which are not in P must be regarded as event points (cf. the figure). They are traversed in ascending order. Each such event point p is a vertex of exactly one 2–dimensional cell $E \in \mathbf{A}(P)$ completely to the right of p. Accordingly for each such p a point $q \in E$ is determined, and the sweep with center q is performed. The resulting permutation π^q is included in ORDS.

5. At the end of the translational sweep. not all elements in ORDS are unique, in general. Duplicates are removed.

To solve problem B the arrangement $\mathbf{A}(P)$ is closely examined. According to what we have stated above the number of 2–dimensional cells of $\mathbf{A}(P)$ is an upper bound on $N(P)$. In the paper it will be shown that this number is $\binom{n}{2} + 1 + n \cdot (n-2) + 3 \cdot \binom{n}{4} = O(n^4)$.

In order to obtain a lower bound on $N(P)$ it shall be assumed furthermore that the convex hull conv(P) of the n–point set P is a polygon with n vertices. Now centers of rotation lying in different 2–dimensional cells which are contained in conv(P) lead to different permutations and orders, respectively. Therefore the number of 2–dimensional cells lying inside conv(P) is a lower bound on $N(P)$. This number is $\binom{n}{4} + \binom{n-1}{2} = \Omega(n^4)$.

Finally, our results are summarized as follows:

Result 1: Given P, ORDERS finds exactly those permutations which can be induced by a rotational sweep.

Result 2: $O(n^5 \log n)$ is an upper bound for the time complexity of the algorithm ORDERS.

Result 3: $\text{Max}\{N(P); P \subset \mathbb{R}^2, \text{card}(P) = n\} = \theta(n^4)$.

References

[BN82] Bieri, H., Nef, W.: *A Recursive Sweep–Plane Algorithm Determing all Cells of a Finite Division of $\mathbb{R}^d$.* Computing **28** (1982), 189–198.

[E87] Edelsbrunner, H.: *Algorithms in Combinatorial Geometry.* EATCS Monographs on Theoretical Computer Science, Vol. **10**, Springer Verlag 1987.

[GP82] Goodman, J.E., Pollack, R.: *A Theorem of Ordered Duality.* Geometriae Dedicada **12** (1982), 63–74.

[H68] Hadwiger, H.: *Eine Schnittrekursion für die Eulersche Charakteristik euklidischer Polyeder mit Anwendungen innerhalb der kombinatorischen Geometrie.* Elem. Math. **23** (1968), 121–132.

[NS90] Nef, W., Schmidt, P.–M.: *Computing a Sweeping–Plane in Regular ("General") Position: A Numerical and a Symbolic Solution.* J. of Symbolic Computation **10** (1990), 633–646.

[P55] Pickert, G.: *Projektive Ebenen.* Springer Verlag Berlin 1955.

Miscellaneous topological algorithms[1]

Colm Ó Dúnlaing
Colum Watt
Mathematics, Trinity College, Dublin 2, Irish Republic.

Abstract

With the advent of high-resolution workstations, it becomes more desirable than ever to develop computational geometry in 3 dimensions. This development should gain from a study of various topological questions in low dimensions. These notes review a (random) assortment of algorithms relevant to such a study.

1 In-groups, out-groups, and embeddings.

In [5] Neuwirth associates with an embedding of an n-complex K in an oriented $(n+1)$-manifold M a group called the *in-group* and written $G(K, M)$.

Specifically, suppose that M is an oriented compact triangulated $(n+1)$-manifold, so its structure is defined by an $(n+1)$-complex. Let K be an n-subcomplex of its n-skeleton. Fix an orientation of M and apply arbitrary orientations to each n- and $(n-1)$-dimensional face in K. Let τ_i and σ_j respectively denote these oriented faces. Each face σ_j has codimension 2 in M, so one can form a small oriented loop ℓ_j going around it. Then ℓ_j intersects some of the faces τ_i in cyclic order $(\tau_{i(1)}, \ldots, \tau_{i(m)})$, say, and the orientations of $\tau_{i(r)}$ and ℓ_j define associated intersection numbers $\epsilon_{i(r)} = \pm 1$. In the group F freely generated by the n-faces $\tau_1, \ldots, \tau_p$ of M, let w_j be the word

$$\tau_{i(1)}^{\epsilon_{i(1)}} \cdots \tau_{i(m)}^{\epsilon_{i(m)}}.$$

The in-group $G(K, M)$ is defined as the group with generators τ_i and relators w_j.

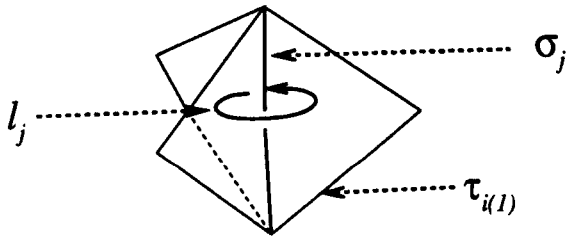


Figure 1: Oriented loop around an $(n-1)$-face.

Suppose that $M \backslash K$ has components $C_1 \ldots C_t$. Form an arc-connected topological space Y by joining these components by arcs to a centre point v (this is a formal construction: the arcs only join to Y at their endpoints). Form X by gluing K back onto Y. Let $G = \pi_1(X)$ be the fundamental group of X with basepoint v, and let H be the normal subgroup of G generated by all loops disjoint from K.

[1]Supported by the EEC under Esprit BRA 3075 (ALCOM). Electronic mail addresses: odunlain@maths.tcd.ie, colum@maths.tcd.ie

Theorem 1.1 *$G(K, M)$ is isomorphic to the quotient group G/H.* ∎

Each in-group $G(K, M)$ is defined uniquely by the cyclic ordering of n-faces τ_i around $(n-1)$-faces σ_j. Each cyclic ordering determines a group, called an *out-group* of K. This provides a necessary condition for embeddability in an oriented $(n+1)$-manifold M:

Proposition 1.2 *A necessary condition for embeddability of K in M where the complement of K has $(r+1)$ components is that at least one out-group of K should be a quotient of $\pi_1(M) * F(r)$ (free product of fundamental group with free group on r generators).* ∎

Proposition 1.3 *A 1-complex (i.e., graph) K is planar iff one of its out-groups is free.* ∎

2 References.

1. J. Bokowski and U. Brehm (1985). A new polyhedron of genus 3 with 10 vertices. *Colloquia Math. Soc. János Bolyai*, Siofok.

2. J. Bokowski and A. Eggert (1986). All realizations of Möbius' torus with 7 vertices. Preprint 1009, FB Mathematik, Technische Hochschule Darmstadt.

3. J. Bokowski and B. Sturmfels (1989). Computational Synthetic Geometry. Springer Lecture notes in Mathematics 1355.

4. S. Lefschetz (1949). *Introduction to Topology.* Princeton University Press.

5. L. Neuwirth (1965). In-groups, coverings, and embeddings. Proc. Cambridge Phil. Soc **61**, 647–656.

6. L. Neuwirth (1968). An algorithm for the construction of 3-manifolds from 2-complexes. *Proc. Camb. Phil. Soc* **64**, 603–613.

7. E. Whittlesey (1960). Finite surfaces: a study of finite 2-complexes, I: local structure, II: the canonical form. *Mathematics Magazine* **34,1:** 11–22, **2:** 67–80.

The Visibility Complex

Michel Pocchiola

LIENS, Ecole Normale Supérieure,
45 rue d'Ulm 75230 Paris Cédex 05,
France (pocchiol@dmi.ens.fr)

and Gert Vegter

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

Consider a collection of pairwise disjoint objects in the plane. We are interested in problems in which these objects arise as obstacles, either in connection with visibility problems where they can block the view from an other geometric object, or in motion planning, where these objects may prevent a moving object from moving along a straight line path.

These problems have many common features. In the solution of both the *visibility graph* often plays an important rôle. For polygonal obstacles the vertices of these polygons are the nodes of the visibility graph, and two nodes are connected by an arc if the corresponding vertices can see each other.

It is well known that a shortest path for a point moving from a given initial situation to a given final situation can be found by computing a shortest path in the visibility graph of the set of obstacles extended with the initial and final position of the point.

Another important geometric object is the *visibility polygon* of a point with respect to the set of obstacles, viz. the clockwise sequence of objects that are visible from this point. Computing the visibility polygon in time proportional to its size—possibly up to a factor that is logarithnmic in the number of obstacles—requires studying a more complicated mathematical object, the so called *visibility complex*.

The set of lines intersecting a fixed object is endowed with an equivalence relation defined in terms of the visibility from the object along these lines. The equivalence classes correspond to cells of the visibility complex. A line intersecting the visible object transversally belongs to a 2-dimensional face; if the line is a tangent of the visible object it either belongs to an edge or to a vertex of the visibility complex.

The visibility polygon of a point can be computed by maintaining the view along a line that performs a rotational sweep around the point. This rotational sweep corresponds to a one-parameter family of lines, i.e. to a curve Γ in the visibility complex. Obviously during the roational sweep the view changes if the line is in a position tangent to the visible object. In terms of the visibility complex this corresponds to an intersection of the curve Γ with an edge or a vertex, viz. with the boundary of a face.

The number of faces intersected by the curve Γ is equal to the size of the visibility polygon. To determine the intersection of this curve with a certain face efficiently we first study the geometry of the faces, and obtain a kind of *Zone Theorem* for the curve Γ. Subsequently we show how the succesive intersections of this curve with the boundaries of the faces can be determined in ouput-sensitive time.

Finally we show how the visibility complex can be constructed in time proportional to its size—again up to a factor logarithmic in the number of obstacles. Since there is a natural correspondence between the set of vertices of the visibility complex and the set of arcs of the *tangent visibility graph* of the set of objects, this construction can also be used to compute the latter graph in time proportional to the size of the output.

We should remark here that our method bears some resemblance to the optimal algorithm of Ghosh and Mount for the construction of the visibility graph of a set of polygonal obstacles—essentially considered as a collection of line segments—although it seems hard to generalize the latter algorithm to the context of more general convex objects. The richer structure of the visibility complex seems to guarantee a more natural approach to visibility problems.

We also expect that our methods can be used in various other geometric problems, e.g. in planning the motion of a rod amidst polygonal obstacles, in ray shooting (this will require a *persistent data structure for the visibility complex* and in the computation of a *sector* of the visibility polygon.

The Flintstones: Hierarchical Approximation and Localization (extended abstract)

Remco C. Veltkamp

CWI, Department of Interactive Systems
Kruislaan 413, 1098 SJ Amsterdam, the Netherlands
e-mail: remco@cwi.nl

Abstract

This paper presents a new representation scheme for closed polygonal and triangular polyhedral objects without holes. The scheme represents a hierarchical approximation to support object manipulation at various levels of detail, as well as hierarchical bounding volume information to support efficient search and intersection operations. The bounding volumes (flintstones) consist of circles in 2D and spheres in 3D, which make the representation storage efficient, and operations computationally efficient.

1 Introduction

Two facilities are often used for the manipulation of complex polygonal or polyhedral objects: approximation, and localization, or bounding volume representation. An approximation of the object can be used if it is not necessary to deal with the object in full detail. Bounding volumes can be used to support search and intersection operations. This paper presents a new representation scheme for closed polygonal and polyhedral objects without holes.

Approximation algorithms that are optimal in some sense, see e.g. [Imai and Iri, 88], are usually not hierarchical. There seem to be no efficient algorithms for optimal approximation in 3D, so that we are dependent on heuristics. Some methods are only approximation schemes, such as presented by [Duda and Hart, 73] and [Faugeras et al., 84], and some are only localization schemes, such as presented by [Ponce and Faugeras, 87]. Still others are both, but not naturally or satisfactory generalizable from 2D to 3D, for example the schemes of [Ballard, 81] and [Günther, 88].

The Flintstones scheme is hierarchical, is both an approximation and a localization scheme, is naturally generalized from 2D to 3D, and computationally efficient in hierarchical operations such as point location and object intersection.

2 Flintstones scheme — 2D

2.1 Localization

Definition 1 (Circle with signed radius) *Let a_1, a_2 and b be points not lying on one line. The radius of the circle $C(a_1, a_2; b)$ through these points is positive if the center of the circle and b lie at the same side of the line through a_1 and a_2, and negative if they lie at opposite sides.*

Definition 2 (±-operator) *Let R and S be two circles with signed radii r_R and r_S respectively. Then*

$$R \pm S = \begin{cases} R \cup S & \text{if } r_R, r_S > 0, \\ R \cap S & \text{otherwise.} \end{cases}$$

Let $H(a_1, a_2; b)$ be the half-plane containing b whose boundary passes through a_1 and a_2. We now consider an open polyline P having consecutive vertices $v_p v_{p+1}, \ldots, v_s$, $p+1 < s$.

Definition 3 (Flintstone) *Let v_q be a vertex of P such that $C(v_p, v_s; v_q)$ contains all vertices lying in $H(v_p, v_s; v_q)$. If $P \subset H(v_p, v_s; v_q)$, then the flintstone $F(P) = C(v_p, v_s; v_q) \cap H(v_p, v_s; v_q)$, otherwise $F(P) = C(v_p, v_s; v_q) \pm C(v_p, v_s; v_r)$, where v_r is a vertex of P not in $H(v_p, v_s; v_q)$ such that $C(v_p, v_s; v_r)$ contains all vertices in $H(v_p, v_s; v_r)$.*

It is easily verified that $P \subset F(P)$, so that $F(P)$ is a bounding area for P.

2.2 Approximation

Let N be the number vertices in the approximated boundary.

The hierarchical approximation of a closed polygon $v_0 \ldots v_{N-1}$ starts with the determination of the smallest circle containing all vertices, which passes through at least two vertices, say v_i and v_j, $i < j$. Line segment $v_i v_j$ is the zero-order approximation of the polygon, dividing it into two open polylines $v_i v_{i+1} \ldots v_j$ and $v_j v_{j+1} \ldots v_i$ (here and in the rest of this section the indices are taken modulo N).

The next level in the hierarchy of approximations of an open polyline $v_p, \ldots, v_s$ depends on $F(v_p, \ldots, v_s)$. If $F(v_p, \ldots, v_s) = C(v_p, v_s; v_q) \cap H(v_p, v_s; v_q)$, then $v_p \ldots v_s$ is approximated by $v_p v_q$ and $v_q v_s$. If $F(v_p, \ldots, v_s) = C(v_p, v_s; v_q) \pm C(v_p, v_s; v_r)$ and if the radius of $C(v_p, v_s; v_q)$ is larger than the radius of $C(v_p, v_s; v_r)$, then $v_p \ldots v_s$ is approximated by $v_p v_q$ and $v_q v_s$, otherwise by $v_p v_r$ and $v_r v_s$.

If there exists a circle through v_p and v_s containing $v_{p+1}, \ldots, v_{s-1}$, then the '±' in Definition 3 is a '∩'. By construction of the approximation, each approximated polyline is contained in a circle. So, all flintstones are the *intersection* of two circles or a circle and a half-plane. The shape of the intersection of two circles gave rise to the name 'flintstone'.

Theorem 2.1 *1[Time complexity] The best case time complexity to construct the full flintstones hierarchy in 2D is $\Theta(N \log N)$. The worst case complexity is $O(N^2)$.*

3 Flintstones scheme — 3D

3.1 Localization

$S(a_1, a_2, a_3; b)$ is a sphere through a_1, a_2, a_3, b, with a signed radius, defined analogously to Definition 1. The ±-operator is defined analogously to Definition 2. $H(a_1, a_2, a_3; b)$ is the half-space containing b whose boundary passes through a_1, a_2, a_3.

Let P be an open polyhedron, and v_p, v_s, and v_t three distinct vertices on the closed boundary polygon.

Definition 4 (Flintstone) *Let v_q be a vertex of P such that $S(v_p, v_s, v_t; v_q)$ contains all vertices lying in $H(v_p, v_s, v_t; v_q)$. If $P \subset H(v_p, v_s, v_t; v_q)$, then the flintstone $F(P) = S(v_p, v_s, v_t; v_q) \cap H(v_p, v_s, v_t; v_q)$, otherwise $F(P) = S(v_p, v_s, v_t; v_q) \pm S(v_p, v_s, v_t; v_r)$, where v_r is a vertex of P not in $H(v_p, v_s, v_t; v_q)$ such that $S(v_p, v_s, v_t; v_r)$ contains all vertices in $H(v_p, v_s, v_t; v_r)$.*

It is easily verified that $P \subset F(P)$, so $F(P)$ is a bounding volume for P.

3.2 Approximation

The hierarchical approximation of a closed polyhedron starts with the determination of the smallest sphere that passes through at least three vertices, say v_i, v_j, and v_k, and contains all vertices. Such a sphere always exists, but need not be the smallest enclosing sphere, which may pass through only two vertices.

The three shortest paths of edges in the polyhedron running from v_i to v_j, v_j to v_k, and v_k to v_i, divide the closed polyhedron into two open polyhedra. Triangle $v_i v_j v_k$ is the zero-th order approximation of the closed polyhedron.

At the next levels of the hierarchical approximation we consider an open polyhedron P and three distinct vertices v_p, v_s, and v_t on the boundary of P. If $F(P) = S(v_p, v_s, v_t; v_q) \cap H(v_p, v_s, v_t; v_q)$, then P is approximated by the three triangles $v_p v_q v_s$, $v_q v_s v_t$, and $v_p v_q v_t$. If $F(P) = S(v_p, v_s, v_t; v_q) \pm S(v_p, v_s, v_t; v_r)$ and if the radius of $S(v_p, v_s, v_t; v_q)$ is larger than the radius of $S(v_p, v_s, v_t; v_r)$, P is approximated by $v_p v_q v_s$, $v_q v_s v_t$, $v_p v_q v_t$, otherwise by $v_p v_r v_s$, $v_r v_s v_t$, and $v_p v_r v_t$.

If the triangles are always split into three new ones, the new triangles get more and more elongated, and edges remain present in all next levels of approximation. To avoid this, $v_p v_s v_t$ can be split into two triangles. The place to split is at a vertex on one of the three paths between v_p, v_s, and v_t. In that case the neighboring triangle must also be split at the same vertex. Since a triangle has three neighbors, it may have to be split at zero to three sides.

If there exists a sphere through $v_p v_s v_t$ containing P, then '$\pm$' in Definition 4 is a '$\cap$'. If no sides of this triangle are split, the new triangles are also contained in a single sphere. However, if one or more sides are split, the new triangles need not be contained in a single sphere. So, unlike the 2D case, flintstones in 3D are not always the intersection of two spheres or a sphere and a half-space, but may also be the union of two spheres.

Theorem 3.1 *2[Time Complexity] The best case time complexity to construct the full flintstones hierarchy in 3D is $O(N(\log N)^2)$. The worst case complexity is $O(N^2 \log N)$.*

4 Hierarchical operations

In this section the term sphere denotes both a circle and a 3D sphere, and the term segment denotes a line segment in 2D, and a triangle in 3D.

A point-in-polygon or -tetrahedron test can be performed efficiently if we can use an approximation of P, that yields the same result as P itself. The next lemma tells when this can be done:

Lemma 4.1 *1 Let B be a part of P, A an approximation of B having the same boundary, and F the flintstone containing B. If a point X is external to F, then X is internal/external to P if and only if X is internal/external to P with B replaced by A.*

The intersection of two objects P and Q can be computed efficiently if the computation of the intersection of two non-intersecting segments is avoided: testing for absence of intersection is easier than solving the intersection. A part of P and a part of Q do not intersect if the corresponding flintstones do not intersect. If the flintstones do intersect, testing is performed at a level of more detail.

Testing whether X is internal to a flintstone involves the calculation of the distances from X to the centers of the spheres. Testing whether two spheres intersect amounts to comparing the distance between their centers with the sum of the radii. Both tests are computationally cheap compared to tests with bounding rectangles (four lines), as in [Ballard, 81], and bounding truncated pyramids (five planes), as in [Ponce and Faugeras, 87].

5 Conclusions

The flintstones representation is boundary based: the approximation is intrinsic to the object boundary, and the bounding volumes contain the boundary segments of the object. The scheme is hierarchical, is both an approximation and a localization scheme, is naturally generalized from 2D to 3D, and computationally efficient in hierarchical operations such as point location and object intersection.

References

[Ballard, 81] D. H. Ballard. Strip trees: A hierarchical representation for curves. *Communications of the ACM*, 24(5), 1981, 310 - 321.

[Duda and Hart, 73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, 1973.

[Faugeras et al., 84] O. D. Faugeras, M. Hebert, P. Mussi, and J. D. Boissonnat. Polyhedral approximation of 3-D objects without holes. *Computer Vision, Graphics, and Image Processing*, 25, 1984, 169 - 183.

[Günther, 88] O. Günther. *Efficient Structures for Geometric Data Management*, volume 337 of *Lecture Notes in Computer Sciences.* Springer-Verlag, Berlin, 1988.

[Imai and Iri, 88] H. Imai and M. Iri. Polygonal approximations of a curve - formulations and algorithms. In G. T. Toussaint (editor), *Computational Morphology - A Computational Geometric Approach to the Analysis of Form*, volume 6 of *Machine Intelligence and Pattern Recognition.* North-Holland, Amsterdam, 1988, 71 - 86.

[Ponce and Faugeras, 87] J. Ponce and O. Faugeras. An object centered hierarchical representation for 3D objects: the prism tree. *Computer Vision, Graphics, and Image Processing*, 38(1), 1987, 1 - 28.

Computing the L_1-Diameter and Center of a Simple Polygon*

Sven Schuierer
Institut für Informatik
Universität Freiburg, Rheinstr. 10–12
D-7800 Freiburg, FRG

1 Introduction

Rectilinear paths and rectilinear obstacles play an important role in many applications. VLSI design [Wid90] is one of the most prominent examples. A natural metric to measure the distance between two points in such a setting is the L_1-metric since it corresponds to the length of a shortest rectilinear path. In this abstract we address the problem of locating the set of points inside a simple rectilinear polygon whose maximal L_1-distance, i.e., the length of a shortest rectilinear path, to any other point in the polygon is minimized. This set is known as the *center* of the polygon. A closely related problem is that of finding the maximal distance between two points inside a polygon, also known as the *diameter problem*. Note that we always measure the distance between two points according to the length of a shortest rectilinear path inside the polygon.

Center and diameter problems have been considered in various other contexts and a number of results have been obtained. Pollack latex master *et al.* [PSR89] give an $O(n \log n)$ algorithm to compute the geodesic center of a polygon if the Euclidean metric is used. The Euclidean diameter of a simple polygon can be computed within the same time bound by a result of Suri [Sur86]. It is not known up to date if the time complexities of these algorithms are optimal or whether they can be improved upon.

2 Definitions

In the following let $\mathbf{P}$ be a simple rectilinear polygon with n edges. A (*rectilinear*) *path* $\mathcal{P}$ is a curve that consists of a finite number of (axis parallel) line segments inside $\mathbf{P}$. The line segments of $\mathcal{P}$ will be called *links*. The L_1-*length* of $\mathcal{P}$, denoted by $\lambda(\mathcal{P})$, is defined as the sum of the lengths of its links. The L_1 *distance* between two points p and q in $\mathbf{P}$ is the L_1-length of the shortest rectilinear path between p and q.

We now give precise definitions of the L_1-diameter and -center of a polygon $\mathbf{P}$. The L_1-*diameter* of $\mathbf{P}$, denoted by $D(\mathbf{P})$, is defined as the maximum distance between two points in $\mathbf{P}$. Closely related is the notion of the radius $R(\mathbf{P})$ of $\mathbf{P}$ which is defined as the smallest value r such that there exists a point p in $\mathbf{P}$ with $L_1(p,q) \leq r$, for all points q of $\mathbf{P}$. The set of points p in $\mathbf{P}$ that satisfy $L_1(p,q) \leq R(\mathbf{P})$, for all $q \in \mathbf{P}$, is called the L_1-*center* of $\mathbf{P}$.

*This work was supported by the Deutsche Forschungsgemeinschaft under Grant No. Ot 64/5–4.

2.1 Chords and the L_1-distance in Polygons

Lemma 2.1 *For each point p in $\mathbf{P}$ and any axis-parallel chord c, there is one unique point on c that minimizes the distance from p to c.*

In the following we call the unique point on c that is closest to p the *projection of p on c* and denote it by $\pi_c(p)$.

Lemma 2.2 *If c is an axis-parallel chord in $\mathbf{P}$ and p a point in $\mathbf{P}$, then we have, for every point q on c, $L_1(p,q) = L_1(p,\pi_c(p)) + L_1(\pi_c(p),q)$.*

As an immediate consequence we obtain

Corollary 2.3 *For each point p in $\mathbf{P}$, all of its furthest neighbours are convex vertices of $\mathbf{P}$. In particular, there is a pair of convex vertices that spans the diameter of $\mathbf{P}$.*

It can be shown that the set of projections of all vertices of $\mathbf{P}$ can be computed in linear time.

3 A Simple Algorithm for the L_1-Diameter of a Polygon

We start off with a simple divide and conquer algorithm that takes $O(n \log n)$ time. In order to do so we choose an axis-parallel chord c that splits $\mathbf{P}$ into two parts $\mathbf{P}_l$ and $\mathbf{P}_r$ of almost equal size. In [dB91] it is shown that an axis-parallel chord can be found in linear time such that the number of edges in both $\mathbf{P}_l$ and and $\mathbf{P}_r$ is less or equal to $3/4n + 2$. The algorithm works as follows.

Algorithm L_1-Diameter

1. If $\mathbf{P}$ is a rectangle, then $D(\mathbf{P})$ is given by the L_1 length of the diagonal.
2. Compute a chord c of $\mathbf{P}$ that cuts $\mathbf{P}$ into two subpolygons $\mathbf{P}_l$ and $\mathbf{P}_r$ such that $|\mathbf{P}_l|$[1], $|\mathbf{P}_r| \leq 3/4\, n + 2$.
3. Compute $M = \max\{L_1(v_1, v_2) \mid v_1 \in \mathbf{P}_l, v_2 \in \mathbf{P}_r\}$ and remember the vertices v_1, v_2 with $L_1(v_1, v_2) = M$.
4. Compute $D(\mathbf{P}_l)$ and $D(\mathbf{P}_r)$ recursively.
5. Let $D(\mathbf{P}) := \max(M, D(\mathbf{P}_l), D(\mathbf{P}_r))$.

The bottleneck of the above algorithm is Step 3. By projecting the vertices of $\mathbf{P}_l$ and $\mathbf{P}_r$ onto c we are able to compute M in linear time which leads to a run time of $O(n \log n)$ of the overall algorithm.

The algorithm can be further improved by avoiding recursion on one side of c. If (say) $\mathbf{P}_r$ is the part of $\mathbf{P}$ such that the furthest neighbour of c of $\mathbf{P}_r$ is closer to c than the furthest

[1]The notation $|\cdot|$ is used to denote the number of edges a polygon consists of.

neighbour of $\mathbf{P}_l$, then we can disregard some parts of $\mathbf{P}_r$ since they can be shown to have a diameter of less than M. The remaining parts of $\mathbf{P}_r$ can then be processed in linear time which leads to an *overall* linear time algorithm.

4 Computing the L_1-center

The main tool for computing the L_1-center is the close relationship between the radius and the diameter of a simple polygon. More precisely, we have the following lemma.

Lemma 4.1 *If* $\mathbf{P}$ *is a simple polygon, then*

$$R(\mathbf{P}) = D(\mathbf{P})/2.$$

So once we have computed the diameter of the polygon, we immediately know its radius. But the above lemma also turns out to be very helpful in computing the center of **P**as the following lemms shows.

Lemma 4.2 *Let* v_1 *and* v_2 *be a pair of diametral vertices and a shortest path* $\mathcal{P}$ *between them. If* p *is the point in the middle of* $\mathcal{P}$*, then there is a square* $\mathbf{S}$ *containing* p *with the following properties*

1. $\mathbf{S}$ *is contained in* $\mathbf{P}$,
2. p *is contained in one diagonal* d *of* $\mathbf{S}$,
3. d *contains the* L_1*-center of* $\mathbf{P}$, *and*
4. $\mathbf{S}$ *can be computed in linear time.*

This observation gives rise to the following algorithm to compute the L_1-center. Given a diametrical pair v_1, v_2 by the diameter algorithm, we compute $\mathbf{S}$ in linear time. Let the four sides of $\mathbf{S}$ be $s_1, \ldots, s_4$ and the maximal line segments in $\mathbf{P}$ through s_i be denoted by t_i. Furthermore, let T_i be the set of all the vertices that are on the opposite side of t_i as is the square $\mathbf{S}$. We now compute the projections of T_i onto s_i. Note that s_i is the closest of the four sides of $\mathbf{S}$ to the vertices in T_i and that we consider all the vertices of $\mathbf{P}$ in this way. For each projection p_i on the boundary of $\mathbf{S}$, we then compute the interval on the diagonal of $\mathbf{S}$ that has distance $R(\mathbf{P})$ to p_i. The intersection of all these intervals then yields the L_1-center.

References

[Cha90] Bernard Chazelle. Triangulating a simple polygon in linear time. In *Proc. 31th Symposium on Foundations of Computer Science*, pages 220–230, IEEE, 1990.

[dB91] M. de Berg. On rectilinear link distance. *Computational Geometry: Theory and Applications*, 1(1):13–34, 1991.

[Lev87] Christos Levcopoulos. *Heuristics for Minimum Decompositions of Polygons.* PhD thesis, University of Linköping, Linköping, Sweden, 1987.

[PSR89] R. Pollack, M. Sharir, and G. Rote. Computing the geodesic center of a simple polygon. *Discrete and Computational Geometry*, 4(6):611–626, 1989.

[Sur86] S. Suri. *Computing the Geodesic Diameter of a Simple Polygon.* Technical Report JHU/EECS-86/08, Johns Hopkins Univesity, Department of Electrical Engeneering and Computer Science, 1986.

[Wid90] P. Widmayer. *On Shortest Paths in VLSI design.* Technical Report 19, Institut für Informatik, Universität Freiburg, Germany, 1990. Also to appear in: Annals of Operations Research, 1991.

A Simple Balanced Search Tree With $O(1)$ Worst-case Update Time[1] (Extended Abstract)

Rudolf Fleischer
Max-Planck-Institut für Informatik
W-6600 Saarbrücken
Germany

Abstract

In this paper we show how a slight modification of (2,4)-trees, called *(2,4,8)-trees*, allows us to perform member and neighbor queries in $O(\log n)$ time and updates in $O(1)$ worst-case time (once the position of the inserted or deleted element is known). Our data structure is quite natural and much simpler than previous worst-case optimal solutions. It is based on two techniques : 1) *bucketing*, i.e. storing an ordered list of $2 \log n$ elements in each leaf of a (2,4,8) tree, and 2) *lazy splitting*, i.e. postponing necessary splits of big nodes until we have time to handle them.

1 Introduction

One of the most common (and most important) data structures used in efficient algorithms is the balanced search tree. Hence there exists a great variety of them in the literature. Basically, they all store a set of n keys such that location, insertion and deletion of keys can be accomplished in $O(\log n)$ worst case time.

In general, updates (insertions or deletions) are done in the following way : First, locate the place in the tree where the change has to be made; Second, perform the actual update; And third, rebalance the tree to guarantee that future query times are in $O(\log n)$. The second step usually takes only $O(1)$ time, whereas steps 1 and 3 both need $O(\log n)$ time. But there are applications which don't need the first step because it is already known where the key has to be inserted or deleted in the tree. In these cases we would like to have a data structure which can do the rebalancing step as fast as the actual update, i.e. in constant time.

It has been well known for a long time that some of the standard balanced search trees can achieve $O(1)$ amortized update time once the position of the key is known ([O82]). But for the worst case update time the best known method had been a complicated $O(\log^* n)$ algorithm by Harel ([H79]). Only recently Levcopoulos and Overmars came up with an algorithm achieving optimal $O(1)$ update time ([LO]). They use the *bucketing technique* of [O82] : Rather than storing single keys in the leaves of the search tree, each leaf (*bucket*) can store up to $O(\log n)$ keys. In fact, the buckets in [LO] have size $O(\log^2 n)$; so [LO] also need a 2-level hierarchy of lists to guarantee $O(\log n)$ query time within the buckets. They show that buckets cannnot grow too big if after every $\log n$ insertions the biggest bucket is split into two halves; rebalancing the search tree can then be distributed over the next $\log n$ insertions for which no split occurs.

[1]This work was supported by the ESPRIT II program of the EC under contract No. 3075 (project ALCOM)

Our paper simplifies this approach considerably : We, too, distribute the rebalancing over the next $\log n$ insertions into the bucket which was split, but allow many buckets to be split at consecutive insertions (into different buckets). This seems fatal for internal nodes of the search tree : they may grow arbitrarily big because of postponed (but necessary) splits. But we show that internal nodes never have more than twice the allowed number of children. This reduces bucket size to $2 \log n$, which means that we need only an ordered list to store the elements of a bucket. Also the analysis of our algorithm seems simpler and more natural than in [LO].

2 The Data Structure

Since we can handle deletions using the *global rebuilding technique* of Overmars ([O83]) we can w.l.o.g. assume from now that the only updates to the data structure are insertions. Queries are the so-called *neighbor queries* : given a key K, if it is in the current set S report it, otherwise, report one of the two neighbors in S according to the given order.

Assume that we start with a set S_0 of n_0 keys. These keys are stored in an ordinary $(2,4)$-tree T_0 of height $\log n_0$. The number of children of a node v will be denoted by $size(v)$. Nodes v with $size(v) > 4$ will be called *big*, otherwise *small* (T_0 has only small nodes). Later the leaves of the tree will grow to buckets of size at most $2 \log n$ where n denotes the current size of the set S which is stored in the tree. Each bucket is represented as a doubly-linked ordered list. After some insertions, the tree T will loose its $(2,4)$-property but the size of nodes will always be bounded by 8. Since big nodes are still considered to be candidates for splitting, we call this kind of tree a *(2,4,8)-tree.*

Insertions into a bucket can be done in constant time if the neighbor of the inserted key is given. For rebalancing, each bucket b has an additional pointer r_b to an internal node of T; in T_0, or directly after a split, it points to the parent of the leaf which contains b. Whenever we insert a new key into b we also test whether r_b points to a big node; if so, this node is split into nodes of size at most 4; otherwise nothing happens. Then r_b is moved upwards one level towards the root, if this parent node exists, and otherwise (i.e. r_b is the root) b is split into two halves. Queries will never change the tree T.

3 The Analysis

The first Lemma can be proved easily by induction.

Lemma 3.1 *$height(T) \leq \log n$ and $size(bucket) \leq 2 \log n$ where n is the current number of keys stored in T.*

We now show that the nodes of T have bounded size. If a node is split, the edge to its parent node (the *parent edge*) is split, too; we call these edges *doubled* (we will show that nodes have size at most 8; hence they can always be split into 2 small nodes). Other edges are called *simple*. After some time doubled edges can again become simple edges (see below).

For an internal node v, let d_v be the number of pairs of split children and e_v be the number of unsplit children, i.e. $size(v) = 2d_v + e_v$. v is called *small* if $2d_v + e_v \leq 4$, *big* otherwise. v is called *normal* if it is not incident to a doubled edge, i.e. $d_v = 0$; otherwise, it is a

b-node (see below the definition of b-trees). v is called *extended* if $d_v \geq 1$, and *blocking* if $d_v + e_v \leq 2$.

For the leaves w of T we define $d_w = 1$ and $e_w = 0$, i.e. leaves are small, extended and blocking b-nodes. The parent edge of the root is defined to be simple.

If a node v is small and its parent edge is simple then it can be *normalized*, i.e. all doubled edges (to split children) become simple. Thus v becomes normal.

It is easy to prove the following relations between these definitions : Extended nodes are b-nodes, blocking nodes are small, normal nodes are small, and big nodes are extended.

At any time, the doubled edges induce a forest of subtrees of T; these subtrees are called *b-trees* (blocking trees). Since a b-tree consists only of doubled edges all its nodes are b-nodes.

We still have to say at which time b-nodes are normalized in the algorithm. If r_b is the root of a b-tree and is discovered - during the rebalancing step of an insertion - to be small, then r_b and, recursively, the whole b-tree are normalized (this can be done because the following Lemma guarantees that all non-root nodes of a b-tree are small).

Lemma 3.2 *For each b-tree T_b :*

(1) The parent edge of root(T_b) is simple.

(2) root(T_b) may be big, but $d_{root(T_b)} + e_{root(T_b)} \leq 4$.

(3) All other nodes of T_b are blocking (and hence small).

(4) T_b contains at least one leaf w of T. For each such leaf w, r_w points to root(T_b) or below.

Proof : (1) is always true by the definition of b-trees. (2)-(4) are proved by induction on the number of nodes split. From (4), it follows that a leaf (bucket) whose parent edge is doubled is not filled yet and hence can not be split. Hence splits can occur only at the root of a b-tree. From (2) we know that this root has only a few children; these children always can be distributed over the two new nodes in such a way that both new nodes are blocking afterwards. Since we created a new doubled edge, the b-tree has grown. If the father of the root was part of another b-tree previously, these two b-trees are fused together to become a big b-tree. In any case, it is easy to verify (2)-(4). □

From these two Lemmas follows immediately

Theorem 3.3 *T is always a $(2,4,8)$-tree. Hence, it supports neighbor queries in time $5 \log n$ and insertions in time $O(1)$, once the position where the key is to be inserted in the tree is known. Also, deletions can be done in time $O(1)$ using the global rebuilding technique.*

4 Conclusions

We have seen how to implement a simple (2,4,8)-tree which supports neighbour queries in time $O(\log n)$ and updates in time $O(1)$. In a real implementation one could change the rebalancing step in such a way that a bucket which normalized its b-tree before being filled, afterwards helps splitting other big nodes (which could be stored in a seperate list to guarantee quick access).

As in [LO], our data structure can not be used as a finger tree. Hence, it remains an open problem to obtain a finger tree with only $O(1)$ worst-case update time.

References

[F92] R. Fleischer
"A simple balanced search tree with $O(1)$ worst-case update time"
Technical Report MPI-I-92-101, Max-Planck-Institut für Informatik,
W-6600 Saarbrücken, Germany, January 1992

[H79] D. Harel
"Fast updates with a guaranteed time bound per update"
Technical Report, Dept. of ICS, University of California at Irvine, 1979

[LO] C. Levcopoulos, M.H. Overmars
"A balanced search tree with $O(1)$ worst-case update time"
*Acta Informatica***26** (1988), 269–277

[O82] M.H. Overmars
"A $O(1)$ average time update scheme for balanced search trees"
*Bull. EATCS***18** (1982), 27–29

[O83] M.H. Overmars
"The design of dynamic data structures"
Lecture Notes in Computer Science, Vol. 156, Springer 1983

Dynamic Point Location in General Subdivisions[1]

Hanna Baumgarten
Graduiertenkolleg Algorithmische Diskrete Mathematik, Institut für Informatik,
Freie Universität Berlin
Arnimallee 2-6, W 1000 Berlin 33, Germany

Hermann Jung
Fachbereich Informatik, Humboldt-Universität Berlin,
PSF 1297, O 1086 Berlin, Germany

Kurt Mehlhorn
Max-Plank-Institut für Informatik and Fachbereich Informatik der Universität des Saarlandes,
W 6600 Saarbrücken, Germany

Planar point location is a fundamental geometric searching problem and has been extensively studied in recent years. In this work we consider the problem of dynamically maintaining a set S of n non-intersecting (except possibly at endpoints) line segments in the plane under the following operations:

- Locate(q: point): Report the segment immediately above q, i.e. the first segment intersected by an upward vertical ray starting at q;
- Insert(s: segment): Add segment s to the collection S of segments;
- Delete(s: segment): Remove segment s from the collection S of segments.

A *connected subdivision* is defined as the graph induced by the set of line segments S (i.e. its vertices are the endpoints of the segments, the edges are the segments) if that graph is connected. Connected sudivisions include the *monotone subdivisions* defined by Preparata and Tamassia [PT89] as a special case.

The *locate* operation in connected subdivisions is usually required to return the name of the region containing the query point (and not only the segment immediately above the query point) and some papers reserve the term "dynamic planar point location problem" for this problem. However, Overmars [Ove85] has shown how to reduce the point location problem in connected subdivisions to the dynamic planar point location problem (as we defined it above) with only $O(\log n)$ additional cost per operation. Furthermore, there are applications of the dynamic point location problem, e.g. when using space sweep techniques, where the connectedness assumption is unnatural.

Table 1 summarizes our results and compares them to previous work. We achieve $O(\log n \log\log n)$ locate and insertion time and $O(\log^2 n)$ deletion time, the bounds for insertions and deletions being amortized. The space bound is $O(n \log n)$ for our first solution, and $O(n)$ for our second solution. This second solution is a refinement of the first one.

Previously, a query time below $O(\log^2 n)$ was only known for the special cases of monotone subdivisions [CT91] and horizontal line segments [MN90], respectively. In both cases non-linear space was needed.

[1]H.B. acknowledges support by the Deutsche Forschungsgemeinschaft under Grant WE 1265/2-1.

Subdivision	Space	Locate	Insert	Delete	Reference
hor. seg.	$n \log n$	$\log n \log\log n$	$\log n \log\log n$	$\log n \log\log n$	Mehlhorn-Näher [MN90]
monotone	n	$\log^2 n$	$\log^2 n$	$\log^2 n$	Preparata-Tamassia [PT89]
monotone	$n \log n$	$\log n$	$\log^2 n$	$\log^2 n$	Chiang-Tamassia [CT91]
monotone	n	$\log^2 n$	$\log n$	$\log n$	Goodrich-Tamassia [GT91]
connected	n	$\log^2 n$	$\log^2 n$	-	Fries-Mehlhorn [Meh84]
connected	n	$\log^2 n$	$\log^4 n$	$\log^4 n$	Fries [Fri90]
general	$n \log n$	$\log^2 n$	$\log^2 n$	$\log^2 n$	Bentley [Ben77]
general	n	$\log^2 n$	$\log n$	$\log n$	Cheng-Janardan [CJ90]
general	$n \log n$	$\log n \log\log n$	$\log n \log\log n$	$\log^2 n$	here
general	n	$\log n \log\log n$	$\log n \log\log n$	$\log^2 n$	here

Figure 1: Running Times of Dynamic Point Location Structures

In the course of deriving our results, we obtain new data structures, which combine well known structures like segment and interval trees and the method of fractional cascading introduced by Chazelle and Guibas [CG86] and dynamized in [MN90].

Let us have a look at the general philosophy of fractional cascading. Fractional cascading is a very powerful data structuring technique to improve the search and update times for structures consisting of a basic undirected frame graph and many linear lists at its nodes. The main idea of fractional cascading is based on the exploitation of the inherent structure of linear orderings. In order to speed up searches, it introduces bridges between lists of adjacent nodes, i.e. it stores the same element in several lists, such that the position of the query element in one list and a nearby bridge guide the search into the vicinity of the position of the query element in the next list. The search time depends on the number of searched nodes, the cost for an initial search part and essentially on the distance between bridges. In this way, the search time can be reduced if the distance between neighboring bridges can be kept small. In [MN90] it was shown that distance $O(1)$ between bridges can be maintained if all lists are drawn from a linearly ordered universe. In the case of general line segments we only have a partial order, the natural "above"-relation between segments. Therefore the standard fractional cascading method can not be applied directly. With respect to the partial order among the line segments we only copy segments (bridges) in root to leaf fashion. In this way, the distance between bridges can be controlled in the parent nodes but not in the children nodes. For searches this does not cause a problem, since they can be performed in leaf to root fashion.

However insertions and deletions become considerably more complex.

(1) An insertion of a segment s into a segment tree forms an insertion sequence for subsegments of s into segment lists of at most $O(\log n)$ nodes nearby the search path for the two endpoint of s. Thus binary search seems to be needed in each node resulting in $O(\log^2 n)$insertion time. We show that information about previously inserted segments can be efficiently used to achieve an amortized insertion time of $O(\log n \log\log n)$.

(2) For deletions the following problem appears. Fractional cascading copies elements from lists to neighboring lists, i.e. it creates bridges between lists. In this way an element may have copies in several lists. Suppose now that a segment with many copies is deleted from the collection S of segments. In standard dynamic fractional

cascading ([MN90]), it is possible to leave the copies as *ghost elements* in the data structure. In the case of segments, the difficulty arises that ghost segments may intersect with segments inserted later. We allow intersections, but in a carefully controlled way, e.g., we guarantee that bridges never intersect and that no segment intersects more than one bridge.

For the second solution we combine the interval-priority-search tree of Cheng and Janardan ([CJ90]) with segment trees and dynamic fractional cascading. The search algorithm proposed in [CJ90] is more complex than binary search and hence the searches in several such data structures interact in a more complex way than binary searches do. We use our previous results and obtain the same time bounds as before for insertion, deletion and query operations, but a space requirement of $O(n)$.

References

[Ben77] J. Bentley. A solution to Klee's rectangle problems. unpublished, 1977.

[CG86] B. Chazelle and L. Guibas. Fractional cascading. *Algorithmica*, 1:133–196, 1986.

[CJ90] S.W. Cheng and R. Janardan. New results on dynamic planar point location. *IEEE FOCS*, pages 96–105, 1990.

[CT91] Y.-J. Chiang and R. Tamassia. Dynamization of the trapezoid method for planar point location. *ACM Symposium on Computational Geoemtry*, 1991.

[Fri90] O. Fries. *Suchen in dynamischen planaren Unterteilungen.* PhD thesis, Univ. des Saarlandes, 1990.

[GT91] M.T. Goodrich and R. Tamassia. Dynamic trees and dynamic point location. *STOC*, 91.

[Meh84] K. Mehlhorn. *Data Structures and Algorithms.* Springer Verlag, 1984.

[MN90] K. Mehlhorn and St. Näher. Dynamic fractional cascading. *Algorithmica*, 5:215–241, 1990.

[Ove85] M. Overmars. Range searching in a set of line segments. *1st ACM Symp. on Comp.Geometry*, pages 177–185, 1985.

[PT89] F. Preparata and R. Tamassia. Fully dynamic point location in a monotone subdivision. *SIAM J. of Computing*, 18:811 - 830, 1989.

Mixed Element Trees: A Generalization of Modified Octrees for the Generation of 3-D Delaunay Meshes

Nancy Hitschfeld and Wolfgang Fichtner
Integrated Systems Laboratory, ETH-Zentrum,
8092 Zurich, Switzerland

Paolo Conti
NTT LSI Laboratories, Atsugi, Kanagawa, Japan

Abstract

We have develop an automatic grid generator for three-dimensional (3-D) devices. We have characterized the class of meshes suitable for the integration of the device equations with the usual numerical schemes as being a subclass of the class of Delaunay meshes. The generated meshes permit an exact geometrical modeling of rather general domain boundaries of modern silicon devices avoiding the "obtuse angle problem" by construction.

Software packages for the numerical solution of partial differential equations (PDEs) are important prototyping tools in several engineering disciplines. Results from numerical simulations are used to design and to optimize parts ranging from large aircraft wings to microscopic semiconductor devices and integrated circuits. The spatial discretization of the structure to be simulated, i.e. its subdivision in cells, is key to the accuracy of the computed solution. An appropriate mesh should fulfill several requirements: first, it must provide a reasonable approximation of the geometry to be modeled, in particular of its boundary and internal material interfaces. Second, it is extremely important to accurately approximate all internal quantities relevant to the solution of the PDEs, such as the doping profile in a semiconductor device. Third, the single cells must fulfill certain geometric constraints imposed by the numerical integration method: if the PDEs are solved with the finite element method, no angle must be smaller than some bound supplied *a priori*. However, if the equations are solved using a control volume discretization or box method(BM), the surfaces defining the volume discretization have perpendicular bisectors of the edges of the tessellation. It can be shown that particular Delaunay tessellations satisfy this requirement [1].

The grid generator Ω presented here has been developed for the simulation of semiconductor devices in 3-D. Several aspects must be considered when tessellating semiconductor devices. As these devices generally feature rather complex geometries the grid generator must be capable of dealing with quite general domain boundaries and material interfaces. In the simulation of mechanical parts similar problems arise. As classical finite element schemes seem inappropriate, PDEs are usually solved using the control volume or box method [2]. This leads to the requirement that the underlying mesh should be a Delaunay grid [1]. In addition in a semiconductor device featuring sizes of several μm, relevant internal quantities may vary over several orders of magnitude within 100 Å, i.e. within a few thousandths of the overall size. Hence, mesh density must vary over several orders of magnitude in the whole device.

The first version of the grid generator, Ω_{oct}(Ω octree), as presented in [3], was based on a conventional modified octree approach. It was shown that the octants of an octree could be

completed with additional edges on the surface or in the interior of each octant. However, in order to obtain an appropriate mesh, modified octrees suffer three serious drawbacks when used for discretizing semiconductor devices:

1. All octants are almost cubic; therefore the point density is locally constant along all three coordinate axes. However, relevant quantities in a semiconductor device may strongly vary along one axis, while remaining constant in the plane perpendicular to the axis. In these cases, isotropic cells, as those resulting from modified octrees, lead to a large number of redundant mesh points.
2. The algorithm used to obtain Delaunay meshes requires that material interfaces intersect an octant in the center of its edges. As a result, several refinements and a huge number of mesh points were required along material interfaces.
3. There may be inaccuracies in computing the volume discretization for interface elements.

In order to overcome these problems, the current version Ω_{me}(Ω with mixed elements) generalizes the modified octree approach in several aspects[4]. The whole device is no longer encapsulated in a single octree, but partitioned in a set of *macro-elements* consisting of cubes, rectangular prisms and rectangular pyramids (Figure 1). We use this set of

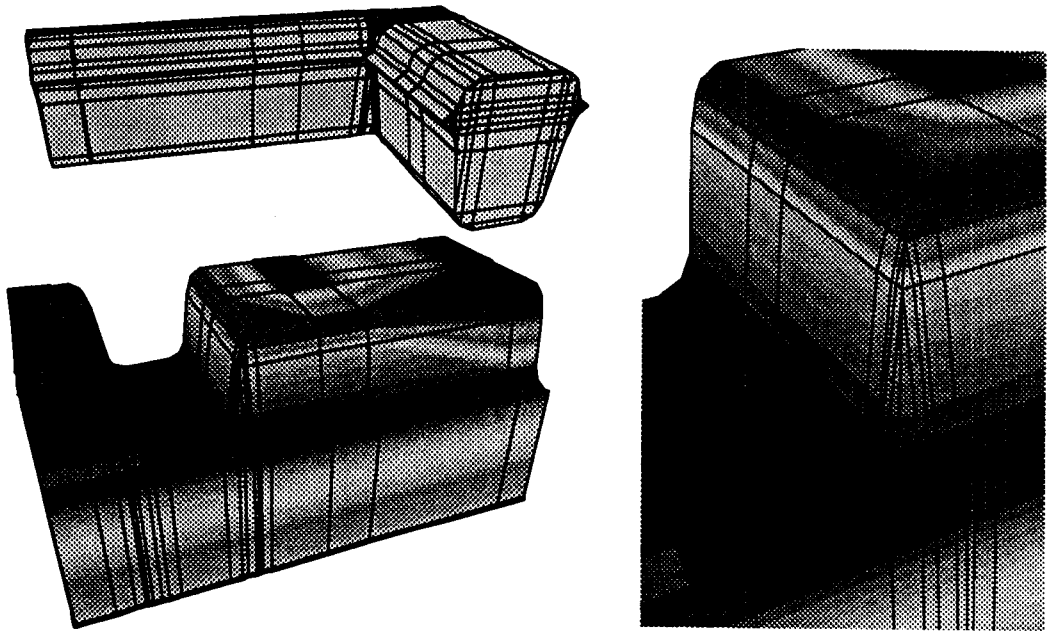


Figure 1: Macro-grid for the ECL bipolar transistor. The number of macro-elements is 2,069: 28 pyramids, 369 prisms and 1,672 cuboids.

elements because they allow the discretization of the device volume using the box method. Furthermore, they are closed under the refinement process, that is, each element obtained after a partition belongs to the same set[1].

[1] A very simple element, the rectangular tetrahedron, cannot be used as *macro-element* because it is not possible to integrate its volume using the box discretization method.

This approach permits the representation of plane-faced geometries using a reasonable number of mesh points. No refinement along material interfaces is required to fit the geometry of the device. Subsequently, the desired mesh density in the interior of the device is obtained through the recursive refinement of prisms, pyramids and cuboids. If a finer mesh is required along one, two, or three coordinate axes, cubes, for example, are subdivided into two halves, four quadrants, and eight octants, respectively. Then, the elements with additional edge mid-points are subdivided into tetrahedra, pyramids, prisms or bricks in order to get a proper finite element mesh. Figure 2 shows a zoomed view of a trench-isolated bipolar transistor as used in state-of-the-art high-speed ECL designs.

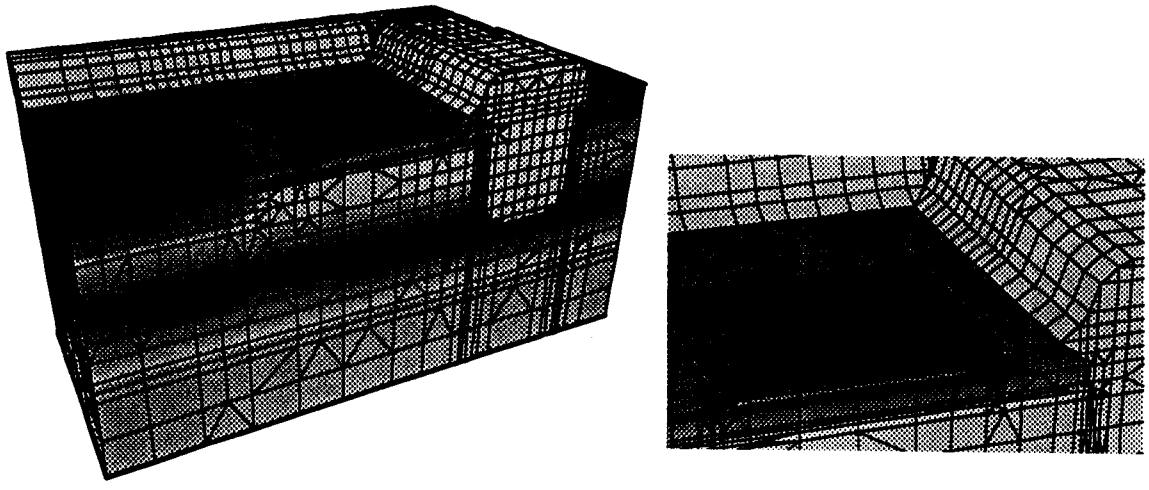


Figure 2: ECL bipolar transistor: A final grid at the left and a zoomed view at the emitter and p-channel at the right

References

[1] P. Conti, *Grid Generation for Three-dimensional Device Simulation.* PhD thesis, ETH Zürich, 1991. published by Hartung-Gorre Verlag, Konstanz, Germany.

[2] R. E. Bank, D. J. Rose, and W. Fichtner, "Numerical methods for semiconductor device simulation," *IEEE Trans. on El. Dev.*, vol. ED-30, no. 9, pp. 1031–1041, 1983.

[3] P. Conti, N. Hitschfeld, and W. Fichtner, "Ω – an octree-based mixed element grid allocator for adaptive 3d device simulation," *IEEE Trans. on CAD/ICAS*, vol. 10, pp. 1231–1241, 1991.

[4] N. Hitschfeld, P. Conti, and W. Fichtner, "Grid generation for 3-d nonplanar semiconductor device structures," in *Proceedings of the 4th International Conference on simulation of Semiconductor Devices and Processes* (W. Fichtner and A. Aemmer, eds.), pp. 165–172, Hartung-Gorre Konstanz, 1991.

Guard data structures for fat spatial objects

Peter Widmayer
Institut für Informatik
Universität Freiburg, Rheinstr. 10–12
D-7800 Freiburg, FRG

The variety of spatial data structures and retrieval algorithms developed in recent years suggests that it is difficult or impossible to design general purpose structures that perform well across the entire spectrum of objects to be stored and queries to be processed - generality comes at the cost of performance and increased algorithm complexity. Thus simple algorithms that perform well on a restricted class of problems are clearly of interest.

Guard algorithms answer stabbing and intersection queries on a dynamic collection of spatial objects embedded in space that satisfy a shape constraint. The objects stored must be fat in a technical sense that can be made precise in several ways, but always implies object convexity and a width-to-length-ratio of at least f, where the fatness f is characteristic of the class of objects to be stored.

Guard algorithms partition space in a hierarchical grid of cells and guard points. Objects are attached to cells and guard points in such a way that a stabbing query is answered in a single path from the root to a leaf of the radix tree that represents the hierarchical grid.

A plane-sweep algorithm for finding a closest pair among convex planar objects

Frank Bartling
FB 12, Informatik, Universitaet - GH - Siegen
Postfach 10 12 40, D - 5900 Siegen, Germany

Klaus Hinrichs
FB 15, Informatik, Westfaelische Wilhelms-Universitaet,
Einsteinstr. 62, D - 4400 Muenster, Germany

Given a set of geometric objects a closest pair is a pair of objects whose mutual distance is smallest. We present a plane-sweep algorithm which finds a closest pair with respect to any L_p-metric, $1 \leq p \leq oo$, for planar configurations consisting of n (possibly intersecting) compact convex objects such as line segments, circular discs and convex polygons. For configurations of line segments or discs the algorithm runs in asymptotically optimal time $O(n \log n)$. For a configuration of n convex polygons with a total of m vertices given in a suitable representation the algorithm finds a closest pair with respect to the Euclidean metric in time $O(n \log m)$.

Contour Triangulation
(Extended Abstract)

Heinrich Müller,
Institut für Informatik,
Universität Freiburg
Rheinstraße 10, W-7800 Freiburg, Germany

The input of the problem of calculating surfaces from cross sectional contours (SFCSC) consists of a sequence of parallel planes in space which are decomposed into disjoint polygonal cells. The output asked for is a spatial decomposition into polytopes whose cross sections along the given planes induce the given cell decompositions of the planes.

The SFCSC problem has found considerable interest in relation with catching data by techniques of tomography. The purpose of this contribution is to present an approach of solution for this problem, which summarizes several other suggestions made in literature in the past. For a more comprehensive survey of this topic cf. [11].

The approach presented here divides the SFCSC into two major subproblems, the *reconstruction of the topology* and the *reconstruction of geometry.*

Reconstruction of topology means mutually assigning the different contours from slice to slice. This problem is of particular difficulty if there are several contours on each plane and if it is known that the corresponding solid can have branchings. The assignment of contours can be formally described by an *assignment graph.* The vertices of the assignment graph correspond to contours. Its edges connect vertices which belong to contours of consecutive slices forming a connected subsolid. An assignment graph is called *valid* if there exists a solid which induces the assignment described by that graph.

For a sequence of contours usually more than one valid assignment graph does exist. On the other hand not every graph which assigns contours between slices is in fact valid. One type of restriction is of topological nature. In general, a planar slice is composed of nested contours. Nested contours occur if the sliced solid has holes, like for instance a torus. A valid assignment of contours has to preserve the nesting. The nesting of contours can be formally described by a *nesting tree.* Each vertex of the nesting tree corresponds to a contour. The root of the nesting tree corresponds to a virtual outer contour which envelops all other contours. The successors of a vertex correspond to the contours immediately contained into its corresponding contour.

We will present an algorithm which enumerates all assignment graphs valid from the view of topology. It is based on an analysis of the nesting trees of the given contours which is related to the *Morse theory* of mathematics (Milnor [9], Morse, Cairns [10], Shinagawa, Kunii, Kergosien [13], Kergosien [8]).

The task of *geometric reconstruction* is deriving the surface of a solid so that its slices coincide with the given contours and which corresponds with a given assignment graph. The surface results from piecewise composition of surface segments which are interpolated from assigned contours in two neighboring cutting planes. The following cases can be distinguished:

Cylindric interpolation. One simply closed polygonal curve in each of two parallel planes which are mutually assigned.

Saddle point interpolation with disconnected saddle. Two disjoint nested polygonal contours in one plane and one contour in the other. The two contours are assigned to the single contour.

Saddle point interpolation with connected saddle. Two disjoint polygonal contours in one plane which are not nested, and one contour in the other plane. The two contours are assigned to the single contour.

Extremal point interpolation. One contour in the first plane, to which no contour in the second plane is assigned.

There are two possibilities of interpolating polyhedrons. One is to allow the polyhedron to have additional vertices besides those given on the contour, and the other to have not. The usefulness of additional vertices can be seen in the saddle case where the saddle point is usually to be expected to lie between the given planes.

An immediate approach when not using additional points is to interconnect the given contours by surfaces of triangles. The case of cylindric interpolation leads to cylindric triangulation. The input of cylindric interpolation consists of two disjoint closed polygonal chains $\mathbf{p}_0, \ldots, \mathbf{p}_m$ and $\mathbf{q}_0, \ldots, \mathbf{q}_n$ in space, $\mathbf{p}_0 = \mathbf{p}_m$, $\mathbf{q}_0 = \mathbf{q}_n$. The output is a set of triangles with vertices $\mathbf{p}_i$, $\mathbf{q}_j$, so that each triangle has vertices on both polygonal chains, each edge of the polygonal chains occurs on exactly one triangle, and each edge $\mathbf{p}_i\mathbf{q}_j$ but $\mathbf{p}_0\mathbf{q}_0$ and $\mathbf{p}_m\mathbf{q}_n$, which belongs to a triangle does belong to exactly two triangles. The number of different cylindric triangulations is $N(m,n) = n \cdot \binom{m+n-1}{m-1}$. We are only interested in those of them which are not penetrating.

By Keppel [7] a description of cylindric triangulations by directed graphs was introduced. The *toroidal graph* can be graphically represented by a two-dimensional grid. It has $m \cdot n$ nodes $\mathbf{n}_{i,j}$, $i = 0, \ldots, m$, $j = 0, \ldots, n$, $\mathbf{n}_{0,j} = \mathbf{n}_{m,j}$, $\mathbf{n}_{i,0} = \mathbf{n}_{i,n}$. Each node $\mathbf{n}_{i,j}$ corresponds to a line segment $\mathbf{p}_i\mathbf{q}_j$ between the two given polygonal chains. The arcs of the toroidal graph have the form $\mathbf{n}_{i,j}\mathbf{n}_{i,j+1}$ and $\mathbf{n}_{i,j}\mathbf{n}_{i+1,j}$. Arcs of the first sort represent triangles $\mathbf{q}_j\mathbf{p}_{j+1}\mathbf{p}_i$, whereas those of the second form correspond to triangles $\mathbf{p}_i\mathbf{p}_{i+1}\mathbf{q}_j$. A cylindric triangulation is expressed by a directed closed path in the toroidal graph which visits each row and column at least once.

The trouble with the *triangulation of saddle points* is that the saddle point cannot be modeled between the two given planes as it would be natural if no additional points are allowed. A way out we suggest for triangulations without additional points is to model the saddle point on one of the given cutting planes. This can be done according to the following two alternatives:

Contour merging. The two contours on the common cutting plane are transferred into one by introducing two nonintersecting line segments. In the disconnected case both line segments totally lie in the exterior of both contours. In the connected case the line segments lie in the ring between the inner and outer contour. A special case is that both edges are identical (Shantz [12]). In the disconnected case the resulting surface consists of a cylindric triangulation an a planar polygon. In the connected case the surface is formed by two cylindric triangulations and a planar polygon. The polygon has to be triangulated.

Contour splitting. The single contour is divided by two nonintersecting line segments into two contours. In the connected case both line segments lie totally in the exterior of the contour, in the disconnected case in the inner of the contour. Both line segments are introduced so that the resulting contours are nonintersecting. In the connected case, the resulting surface consists of a cylindric triangulation and a planar polygon. In the disconnected case the surface is composed by two cylindric triangulations and a planar polygon. Again the polygon is triangulated.

The third case are the *extremal points.* They appear if on one of the neighboring cutting planes no continuing contour is assigned to a given contour. The only possibility of triangulation without additional points is to triangulate the inner of the polygon. This triangulation closes the surface as required at an extremal point.

The preceding discussion of geometric reconstruction is sufficient for a special restricted class of assignment graphs. For the talk, a treatment of the general case is envisaged.

It is not difficult to give an algorithm for enumerating all possible triangulations of these types. However, among the large number of triangulations only a considerably reduced subset will usually be acceptable for an area of application. For the case of contour triangulation, different criterions for acceptable triangulations were proposed. These criterions can be distinguished in *weight criterions* and *shape criterions.* The last part of the talk is devoted to their discussion which can be summarized as follows.

Weight criterions can be often easily expressed formally by using the toroidal graph. For this purpose, weights $w(\mathbf{e})$ respectively $w(\mathbf{n})$ are assigned to its edges $\mathbf{e}$ and nodes $\mathbf{n}$ (Ganapathy, Dennehy [4]). Typical weight criterions are minimum area of the total surface (Fuchs, Kedem, Uselton [3]), minimum biggest area of a triangle, minimum total length of the edges of the triangulation, minimum longest edge, minimum biggest angle of a triangle, maximum smallest angle of a triangle, maximum smallest slope of a triangle relative to the planes of the polygonal chains, and maximum volume of the polyhedron induced by the triangulated surface. Some of these criterions can be satisfied very efficiently since they lead to the problem of finding a shortest path in the toroidal graph.

Kaneda, Harada, Nakamae, Yasuda, Sato [6] have proposed to connect each vertex on a contour with its nearest neighbor on the other contour. However, it is not always possible to obtain a triangulation containing all these interconnecting line segments as edges. The alternative is considering those triangulations containing a maximum number of such edges. Until now, no algorithm more efficient than exhaustive search seems to be known. Shinagawa, Kunii, Nomura, Okuno, Hara [14] give an approximate solution for the maximum number of shortest edges.

Further types of triangulation were obtained by *greedy approaches.* Starting with an initial node in the toroidal graph, the greedy algorithm of Christiansen, Sederberg [1] successively constructs a path by adding that neighbor of the currently last node of the path which has the smallest weight. The weight used is the length of the connecting line segment represented by the node. Ganapathy, Dennehy [4] choose the next node in such a way that that the absolute difference between the length of the piece of the upper contour and the piece of the lower contour processed yet is minimized all the time.

Finally, Cook, Batnitsky [2] suggest to construct the triangles in such a way that their orientation is a s close as possible to the orientation of the line joining the centroids of the two contours.

Shape criterions take into consideration the shape properties of the given polygonal chains known from the area of image processing (cf. e.g. Gonzalez, Wintz [5]). An example is shape analysis based on curvature. Curvature allows to identify changes between left and right windings, and regions of extremal curvature. The shape of the polygonal chains is transferred onto the shape of the triangulation by inserting edges between both contours which connect points with the same property in suitable order. By the new edges, the given stripe is subdivided into substripes which are triangulated according to some further criterion.

References

[1] H.N. Christiansen, T.W. Sederberg (1978). *Conversion of complex contour line definition into polygonal element mosaics. Computer Graphics*, 12(3):187–192.

[2] P.N. Cook, S. Batnitsky (1981). *Three-dimensional reconstruction from serial sections for medical applications. Proceedings of the 14th Hawaii International Conference on System Sciences*, 2:358–389.

[3] H. Fuchs, Z. Kedem, S.P. Uselton (1977). *Optimal surface reconstruction from planar contours. CACM*, 20:693–702.

[4] S. Ganapathy, T.G. Dennehy (1982). *A new general triangulation method for planar contours. Computer Graphics*, 16(3):69–75.

[5] R.C. Gonzalez, P. Wintz (1987). *Digital image processing, 2nd ed.* Addison-Wesley, Reading, Mass.

[6] K. Kaneda, K. Harada, E. Nakamae, M. Yasuda, A.G. Sato (1987). *Reconstruction and semi-transparent display method for observing inner structure of an object consisting of multiple surfaces.* In T.L. Kunii, editor, *Computer Graphics 1987*, pp. 367–380. Springer-Verlag, New York.

[7] E. Keppel (1975). *Approximating complex surfaces by triangulation of contour lines. IBM J. Res. Devel.*, 19:2–22.

[8] Y. L. Kergosien (1991). *Generic sign systems in medical imaging. IEEE CG & Appl.*, 11(9):46–65.

[9] J. Milnor (1963). *Morse Theory.* Princeton University Press, New Jersey.

[10] M. Morse, S.S. Cairns (1969). *Critical point theory and differential topology.* Academic Press, San Diego.

[11] H. Müller . *Surface interpolation from cross sections.* In H. Hagen, H. Müller, G. Nielson, editors, *Scientific Visualization Seminar Book.* Springer-Verlag, Berlin, to appear.

[12] M. Shantz (1981). *Surface definition for branching contour-defined objects. Computer Graphics*, 15(2):242–270.

[13] Y. Shinagawa, T.L. Kunii, Y.L. Kergosien (1991). *Surface coding based on Morse theory. IEEE CG & Appl.*, 11(9):66–78.

[14] Y. Shinagawa, T.L. Kunii, Y. Nomura, T. Okuno, M. Hara (1989). *Reconstructing smooth surfaces from a series of contour lines using a homotopy.* In R.A. Earnshaw, B. Wyvill, editors, *New Advances in Computer Graphics*, pp. 147–161. Springer-Verlag, Berlin.

Parallel Triangulation of Nonconvex Polytopes

W. Preilowski
Department of Mathematics and Computer Science,
University of Paderborn

Abstract

A new parallel algorithm for the triangulation of a nonconvex polytop $\mathcal{P}$ is presented. It will be shown that $\mathcal{P}$ can be decomposed into $O(n + r^2)$ tetrahedra within time $O(log(n) \cdot max\{log^*(n), log(r)\})$ with $O(n + r^2)$ processors, where r is denoting the number of reflex edges of $\mathcal{P}$.

Introduction

Look at the following problem: given a nondegenerated simple nonconvex polytope $\mathcal{P} \subset \mathbb{R}^3$ with r reflex edges, we have to partition the interior of P into a small set of tetrahedra. The partition of geometric objects into a small number of convex pieces is an important problems in computational geometry, as can been seen in many applications in computer graphics and pattern recognition.
In the plane the parallel triangulation-problem for nonconvex polygones is optimally solved by Goodrich [Go 89] in time $O(log(n))$ with $O(\frac{n}{log(n)})$ processors. For triangulation of 3-dimensional nonconvex objects only sequential algorithms and complexity results by Chazelle [Ch 84],[Ch 90], Ruppert and Seidel [RuSe 89] and Lingas [Li 82] are publicated. In [RuSe 89] it is shown that if no Steiner points are allowed in the decomposition, the problem of deciding whether a given polytope is decomposible into tetrahedra is NP-complete. From Lingas we have learned that the problem of finding a decomposition with a minimum number of tetrahedra is NP-hard.
Chazelle has given an $O(n^2)$ lower bound for the number of tetrahedra used in the triangulation by using Steiner points in [Ch 84]. In [Ch 90] he gave a sequential algorithm running in time $O((n + r^2) \cdot log(r))$ for a nondegenerated simple nonconvex polytope $\mathcal{P}$ containing $O(r)$ reflex edges.
In this paper a parallel algorithm based on Chazelle's result is given. Because of Ruppert's and Seidel's result it is not always possible to triangulate in a way that all extrempoints of the tetrahedra used for the triangulation are also endpoints of P. So it is possible that the algorithm presented here could use up to $O(r^2)$ Steiner points for the triangulation.
It will be shown that a simple nondegenerated nonconvex polytope $\mathcal{P}$ of genus 0 with n vertices and r reflex edges can be decomposed into $O(n + r^2)$ tetrahedra within time $O(log(n) \cdot max\{log^*(n), log(r)\})$ using $O(n + r^2)$ processors.

The Triangulation Algorithm:

The algorithm is based on Chazelle's sequential approach, which uses time $O((n + r^2) \cdot log(r))$.
It consists of two phases the Pull-Off- and the Fence-Off-Phase. In the Pull-Off-Phase one has to search many vertices on the boundary of $\mathcal{P}$ whose cups are unhindered. These

cups can be replaced by their domes. Triangulation of the unhindered cups is easy by drawing straight lines from the apex of the cup to all vertices of the crown, as they are all star-shaped polytopes. Mark, that in this phase no Steiner-points are needed for the triangulation. Arbitrary replacing of unhindered cups by their domes could lead to a large number of tetrahedra, namely $O(n \cdot r)$ in the triangulation. So the order of replacing must be determined very carefully.(See figure 1) After the Pull-Off-Phase the number of nodes of the polytope $\mathcal{P}$ is $O(r)$. In the Fence-Off-Phase the resulting polytope will be triangulated into a partition of tetrahedra. Therefore vertical walls called fences are placed through all reflex edges.(See figure 2) This leads to a partition of the interior into $O(r^2)$ cylindric polytopes. These polytopes are not all necessarily convex. They can be made convex by triangulating their horizontal base polygon and set up vertical walls onto the new edges, too. Each of the resulting pieces can be triangulated with a constant number of tetrahedra. The Fence-Off-Phase can be done in time $O(log(r))$ with $O(r^2)$ processors.[1]

References

[Ch 84] B. Chazelle. *Convex Partition of Polyhedra: A Lower Bound and Worst-Case Optimal Algorithm. SIAM J. Comput., Vol. 13, No. 3, 488-507, 1984.*

[Ch 89] B. Chazelle. *An Optimal Algorithm for Intersecting Three-Dimensional Convex Polyhedra. IEEE, pages 586-591, 1989.*

[Ch 90] B. Chazelle, L. Palios. *Triangulating a Nonconvex Polytope. Discrete Comput. Geometry 5, 505-526,1990.*

[CoZa 90] R. Cole, O. Zajicek. *An optimal Parallel Algorithm for Building a Data Structure for Planar Point Location. Journal of Parallel and Distributed Computing, no.8, pages 280-285, 1990.*

[GoPlSh 87] A. V. Goldberg, S. A. Plotkin, G. E. Shannon. *Parallel Symmetry-Breaking in Sparse Graphs. Journal Discrete Mathematics, No. 1, pages 434-446, 1988.*

[Go 89] M. T. Goodrich. *Triangulating a Polygon in Parallel. Journal of Algorithms, No. 10, pages 327-351, 1989.*

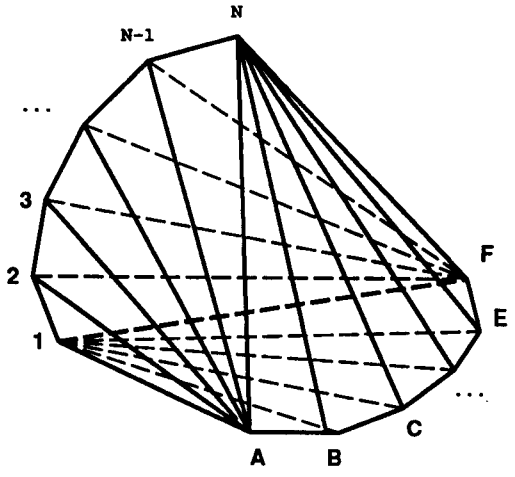
[Li 82] A. Lingas. *The Power of Non-Rectingular Holes. Proc. 9th Colloq. Automata, Languages and Programming, 369-383, 1982.*

[RuSe 89] J. Ruppert, R. Seidel. *On the Difficulty of Tetrahedralizing 3-Dimensional Non-Convex Polyhedra. Proceedings of the fifth annual Symposium on ACM - Computational Geometry, pages 380-392, 1989.*

[1] The main time $O(log(r))$ is used for the triangulation of the base poygons.

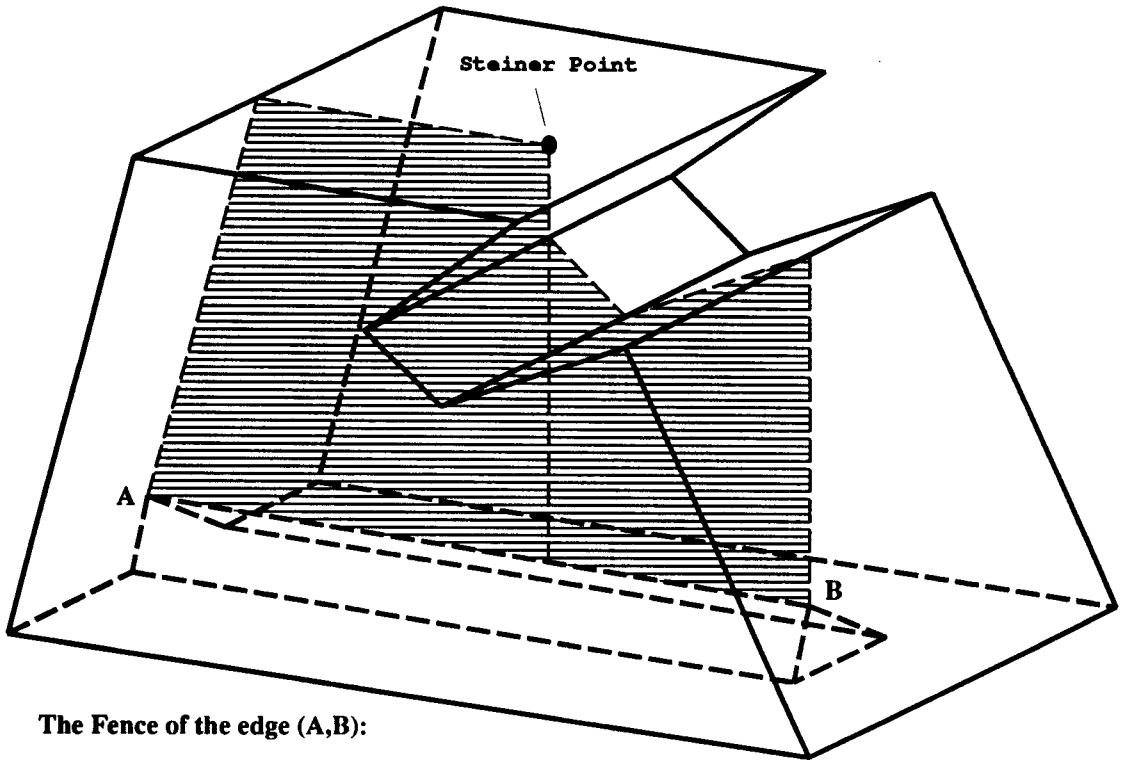
Figures

figure 1:



The removement-order N-(N-1)-...-3-2-1 of unhindered cups leads to a number of O(n· r) tetrahedra, whereas the reverse order guarantee O(n) pieces for the triangulation.

figure 2:



The Fence of the edge (A,B):

Author's index

List of Participants

Manuel Abellanas
Facultad de Informatica
Buadilla del Monte
28660 Madrid
Spain
tel: +34-1-336-7426
fax: +34-1-352-4645
mavellanas@fi.upm.es

Helmut Alt
Freie Universität Berlin
FB Mathematik
Arnimallee 2-6
D-1000 Berlin 33
Germany
tel: +49-30-838-5911
fax: +49-30-838-5913
alt@tcs.fu-berlin.de

Franz Aurenhammer
Freie Universität Berlin
Institut für Informatik
Arnimallee 2-6
D-1000 Berlin 33
Germany
tel: +49-30-838-3775
fax: +49-30-838-5913
aurenham@tcs.fu-berlin.de

Frank Bartling
Universität-GH Siegen
FB 12 / Informatik
Hölderlinstr 3
5900 Siegen
Germany
tel: +49-271-7402304

Hanna Baumgarten
Freie Universität Berlin
Institut für Informatik
Graduiertenkolleg Algorithmische Diskrete Mathematik
Arnimallee 2-6
D-1000 Berlin 33
Germany
tel: +49-30-838-6649
fax: +49-30-838-5913
baumgart@tcs.fu-berlin.de

Mark de Berg
Utrecht University
Dept. of Computer Science
P.O. Box 80.089
3508 TB Utrecht
the Netherlands
tel: +31-30-533922
fax: +31-30-513791
markdb@cs.ruu.nl

Hanspeter Bieri
Universität Bern
Institut für Informatik
Länggassstrasse 51
CH-3012 Bern
Switzerland
tel: +31-658681
fax: +31-653965
bieri@iam.unibe.ch

Ton Dammers
Katholieke Universiteit Nijmegen
Fac. Natuurwetenschappen
Experimentele Vaste Stof Fysica 2
Toernooiveld 1
6525 ED Nijmegen
the Netherlands
tel: +31-80-653094
fax: +31-80-553450
dammers@sci.kun.nl

Ir. Victor J. Dielissen
Technische Universiteit Eindhoven
Faculteit Wiskunde en Informatica
Vakgroep Informatica
Postbus 513
Den Dolech 2
5600 MB EINDHOVEN
the Netherlands
tel: +31-40-474317
fax: +31-40-436685

Jacqueline Duquesne
INRIA-Sophia Antipolis
BP 109
2004 Route des Lucioles
06561 Valbonne
France
tel: +33-93657749
fax: +33-93657643
duquesne@alcor.inria.fr

Vincenzo Ferrucci
University of Rome "La Sapienza"
Dept. of Computer Science
Via Buonarroti 12
00185 Roma
Italy
tel: +39-6-4873676
fax: +39-6-4873628
ferrucci@iasi.rm.cnr.it

Rudolf Fleischer
Max-Planck-Institut für Informatik
Im Stadtwald
W-6600 Saarbrücken
Germany
tel: +49-681-3025359
fax: +49-681-3024421
rudolf@mpi-sb.mpg.de

Michael Formann
Freie Universität Berlin
Institut für Informatik
FB Mathematik
Arnimallee 2-6
W-1000 Berlin 33
Germany
tel: +49-30-838-6589
fax: +49-30-838-5913
formann@tcs.fu-berlin.de

Paolo Giulio Franciosa
Università di Roma "La Sapienza"
Dipartimento di Informatica e Sistemistica
Via Salaria, 113
00198 Roma
Italy
tel: +39-6-8841820
fax: +39-6-8442383
franciosa@vaxrma.infn.it

Carlo Gaibisso
IASI-CNR
Viale Manzoni 30
00185 Rome
Italy
tel: +39-6-770031
fax: +39-6-770031
gaibisso@iasi.rm.cnr.it

Michael Godau
Freie Universität Berlin
Institut für Informatik
Arnimallee 2-6
W-1000 Berlin 33
Germany
tel: +49-30-838-5924
fax: +49-30-838-5913
godau@tcs.fu-berlin.de

Mordecai Golin
INRIA
Domaine de Voluceau-Rocquencourt
B.P. 105
F-78153 Le Chesnay Cédex
France
tel: +33-1-39635658
fax: +33-1-39635330
golin@margaux.inria.fr

Mark de Groot
P.J. Blokstraat 29
2313 ES Leiden
the Netherlands
tel: +31-71-126271

Johann Hagauer
Technische Universität Graz
IICM
Klosterwiesg. 32/II
8010 Graz
Austria
tel: 316-810063
fax: 316-8100655
JHAGAUER@IICM.TU-GRAZ.AC.AT

Klaus Hinrichs
Westfälische Wilhelms Universität
FB-15 / Informatik
Einsteinstraße 62
D-4400 Münster
Germany
tel: +251-833752
hinrichs@ti.e-technik.uni-siegen.dbp.de

Nancy Hitschfeld
ETH Zurich
Integrated Systems Labs.
Gloriastr. 35
CH-8092 Zurich
Switzerland
tel: +41-1-256-5746
fax: +41-1-256-0994
nancy@iis.ethz.ch

Dr. Ulrich Huckenbeck
Universität Würzburg
Lehrstuhl I für Informatik
Am Hubland
8700 Würzburg
Germany
tel: +49-931-8885025
fax: +49-931-8884600
hu@informatik.uni-wuerzburg.de

Ferran Hurtado
Universitat Politecnica Catalunya
Dept. Matem. Applicada II
Pau Gargallo 5
08028 Barcelona
Spain
tel: +34-3-4017280
fax: +34-3-4017040
hurtado@ma2.upc.es

Goos Kant
Utrecht University
Dept. of Computer Science
P.O. Box 80.089
3508 TB Utrecht
the Netherlands
tel: +31-30-534092
fax: +31-30-513791
goos@cs.ruu.nl

Prof.Dr. Rolf Klein
Fernuniversität Hagen-Gesamthochschule
FB Inform, Prakt. Inform. VI
Postfach 940
Elberfelderstraße 95
D-5800 Hagen 1
Germany
tel: +49-2331-804-8365
fax: +49-2331-332904
klein@fernuni-hagen.de

Marc van Kreveld
Utrecht Unievrsity
Dept. of Computer Science
P.O. Box 80.089
3508 TB Utrecht
the Netherlands
tel: +31-30-533922
fax: +31-30-513791
marc@cs.ruu.nl

P.W.H. Lemmens
Rijksuniversiteit Utrecht
Vakgroep Wiskunde
Postbus 80.010
Budapestlaan 6
3508 TA Utrecht
the Netherlands
tel: 030-531426
fax: 030-518394
lemmens@math.ruu.nl

Marisa Mazón
Universidad de Cantabria
Facultad de Ciencias
Avde. Los Castros s/n
39071 Santander
Spain
tel: 34-42-201522
fax: 34-42-201402
recio@ccucvx.unican.es

J.E. Mebius
Technische Universiteit Delft
Vakgroep Technische Informatica
Postbus 356
2600 AJ Delft
the Netherlands
tel: +31-15-783072
fax: +31-15-787141

H. Müller
Universität Freiburg
Institut für Informatik
Rheinstraße 10-12
D-7800 Freiburg
Germany
tel: +49-761-203-3892/3891
fax: +49-761-203-3889
mueller@informatik.uni-freiburg.de

Prof.Dr. Hartmut Noltemeier
Universität Würzburg
FB Informatik
Am Hubland
8700 Würzburg
Germany
tel: +49-931-8885055
fax: +49-931-8884600
noltemei@informatik.uni-wuerzburg.de

Dr. C. O'Dunlaing
Dublin University
Trinity College, School of Mathematics
Dublin 2
Irish Republic
tel: +353-1-702-1948 / 1949(messages)
fax: +353-1-702-2694
odunlain@maths.tcd.ie

Mark H. Overmars
Utrecht Unievrsity
Dept. of Computer Science
P.O. Box 80.089
3508 TB UTRECHT
the Netherlands
tel: +31-30-533736
fax: +31-30-513791
markov@cs.ruu.nl

Alberto Paoluzzi
University of Rome "La Sapienza"
Dept. of Computer Science
Via Buonarroti 12
00185 Roma
Italy
tel: +39-6-4873676
fax: +39-6-4873628
paoluzzi@iasi.rm.cnr.it

F. Pistorius
Universiteit Twente
Faculteit Informatica
Postbus 217
7500 AE Enschede
the Netherlands
tel: +31-53-893763
fax: +31-53-339605
pistorius@cs.utwente.nl

Michel Pocchiola
LIENS, Laboratoire d'Informatique de l'ENS
45, rue d'Ulm
75230 Paris
France
tel: +33-1-43266158
fax: +33-1-46340531
pocchiol@dmi.ens.fr

E.J. Pol
Philips Natuurkundig Laboratorium
WAY 1.69
Postbus 80.000
5600 JA EINDHOVEN
the Netherlands
tel: +31-40-742174
fax: +31-40-744004
evert@prl.philips.nl

Waldemar Preilowski
Universität-GH Paderborn
FB 10 - Informatik
Warburgerstraße 100
D-4790 Paderborn
Germany
tel: +49-5251-603326
fax: +49-5251603342
waldi@uni-paderborn.de

Haijo Schipper
Rijksuniversiteit Groningen
Vakgroep Informatica
Postbus 800
9700 AV Groningen
the Netherlands
tel: +31-50-633957 / 3939
fax: +31-50-633800
haijo@cs.rug.nl

Stefan Schirra
Max-Planck-Institut für Informatik
Im Stadtwald
W-6600 Saarbrücken
Germany
tel: +49-681-302-5359
fax: +49-681-302-5401
stschirr@mpi-sb.mpg.de

Dr. Peter-Michael Schmidt
Mathem. Fakultät der FSU-Jena
Universitätshochhaus 17.OG
O-6900 Jena
Germany
tel: 8224765

Sven Schuierer
Universität Freiburg
Institut für Informatik
Rheinstraße 10-12
D-7800 Freiburg
Germany
tel: +49-761-203-3894
fax: +49-761-203-3889
schuierer@informatik.uni-freiburg.de

B.C. Schultheiss
Universiteit Twente
Faculteit Informatica
Postbus 217
7500 AE Enschede
the Netherlands
tel: +31-53-893763
fax: +31-53-339605

Christian Schwarz
Max-Planck-Institut für Informatik
Im Stadtwald
D-6600 Saarbrücken
Germany
tel: +49-681-3025355
fax: +49-681-3025401
schwarz@mpi-sb.mpg.de

Otfried Schwarzkopf
Utrecht University
Dept. of Computer Science
P.P. Box 80.089
3508 TB Utrecht
the Netherlands
tel: +31-30-533977
fax: +31-30-513791
otfried@cs.ruu.nl

Sven Skyum
Aarhus University
Dept. of Computer Science
Ny Munkegade, Building 540
DK-8000 Aarhus C
Denmark
tel: +45-86202711/5227
fax: +45-86135725
sskyum@daimi.aak.dk

A.W.M. Smeulders
Universiteit van Amsterdam
Faculteit Wiskunde en Informatica
Kruislaan 403
1098 SJ Amsterdam
the Netherlands
tel: +31-20-5257563
fax: +31-20-5257490
smeulder@fwi.uva.nl

Michiel Smid
Max-Planck-Institut für Informatik
Im Stadtwald
D-6600 Saarbrücken
Germany
tel: +49-681-302-5354
fax: +49-681-3025401
michiel@cs.uni-sb.de

Frank van der Stappen
Utrecht University
Dept. of Computer Science
P.O. Box 80.089
3508 TB Utrecht
the Netherlands
tel: +31-30-534095
fax: +31-30-513791
frankst@cs.ruu.nl

Sabine Stifter
Johannes Kepler Universität
RISC-Linz
A-4040 Linz-Auhof
Austria
tel: +7236-323160
fax: +7236-333830
stifter@risc.uni-linz.ac.at

Roberto Tamassia
Brown University
Dept. of Computer Science
P.O.Box 1910
Providence, RI 02912-1910
USA
fax: 401-863-7657
rt@cs.brown.edu

Monique Teillaud
INRIA-Sophia Antipolis
BP 109
2004 Route des Lucioles
06561 Valbonne
France
tel: +33-93657762
fax: +33-93657643
teillaud@alcor.inria.fr

Gert Vegter
Rijksuniversiteit Groningen
Vakgroep Informatica
Postbus 800
9700 AV Groningen
the Netherlands
tel: +31-50-633930
fax: +31-50-633800
gert@cs.rug.nl

Remco Veltkamp
CWI
Afdeling Interactieve Systemen
Kruislaan 413
1098 SJ AMSTERDAM
the Netherlands
tel: +31-20-5924128
fax: +31-20-5924199
Remco.Veltkamp@cwi.nl

Emo Welzl
Freie Universität Berlin
Institut für Informatik
Arnimallee 2-6
W-1000 Berlin 33
Germany
tel: +49-30-838-6635
fax: +49-30-838-5913
emo@tcs.fu-berlin.de

Peter Widmayer
Universität Freiburg
Institut für Informatik
Rheinstraße 10-12
D-7800 Freiburg
Germany
tel: +49-761-203-3885 / 3886
fax: +49-761-203-3889
widmayer@informatik.uni-freiburg.de