KEY-PROBLEMS AND KEY-METHODS IN COMPUTATIONAL GEOMETRY

Herbert Edelsbrunner
Institutes for Information Processing
Technical University of Graz
Schießstattgasse 4a, A-8010 Graz, Austria.

ABSTRACT: Computational geometry, considered a subfield of computer science, is concerned with the computational aspects of geometric problems. The increasing activity in this rather young field made it split into several reasonably independent subareas. This paper presents several key-problems of the classical part of computational geometry which exhibit strong interrelations. A unified view of the problems is stressed, and the general ideas behind the methods that solve them are worked out.

INTRODUCTION

How can the field of computational geometry be defined and what are the characteristics of key-problems and key-methods in this area? Without attempting to answer these hardly defined questions, we believe that, from the present point of view, computational geometry can be described as the discipline that is concerned with the computational aspects of geometrical questions. So far, the overwhelming majority of problems dealt with in the field are low-dimensional, that is, defined in the plane or in three dimensions. We also believe, that key-problems and key-methods in the area are those that posses mathematical beauty and simplicity coupled with computational efficiency and broad influence.

Although few earlier publications exist, computational geometry was properly started by the doctoral thesis of Shamos [Sh]. He came up with the first and most influencing classification into problems around convex hulls, closest-point problems, and intersection problems. A period of flourishing activity on these issues, on rectangle problems, and on dynamization of data structures followed. Today, we are convinced to recognize three mainstreams, however classified according to a different type of criterium: (i) The investigation of underlying mathematical principles, (ii) considerable effort to work out implementation details which have been neglected in the early days, and (iii) a constant challenge of the field by renewing activity on the borderline between computational geometry and more practical areas of computer science such as pattern recognition, cluster analysis, computer graphics, linear programming, robotics, VLSI design, database theory, computer-aided design, and others.

The primary goal of this paper is to present a small collection of problems and methods of computational geometry that are considered central by the author. Part I exhibits problems that are related to each other in various ways. We believe that these relationships allow for a reasonably consistent treatment. Part II discusses the general methods that are exploited to efficiently solve the problems of Part I. Finally, conclusions are offered.

I. KEY-PROBLEMS

Eight problems from computational geometry are defined, discussed, and briefly treated. Four of these problems have a strong geometrical flavour as they require the construction of geometric structures: order-1 Voronoi diagrams, higher-order Voronoi diagrams, convex hulls, and arrangements. The beauty of the other four problems (post-office problem, point location search, linear programming, and halfplanar range search) stems from a mathematically clear and computationally efficient solution. Throughout, emphasis is laid on an intuitively appealing presentation of the problems and their interrelations.

1. The post-offi

Let S be a set c
site s that minim
in S is called t
rest neighbour)
following kind c
point q, determir
q has a unique
nearest neighbour

This problem is
After being ment
tional geometry.
that is, O(logn)
the locus of poi
defines a subdiv
diagram of S. A
that is, by dete
it. Anticipatin
point location,
preprocessing ti

2. (Order-1) Vo

For any two poi
of points at lea
S} is called the
halfplanes, V(s)

1. The post-office problem.

Let S be a set of n points (or sites) in the Enclidean plane E^2. A site s that minimizes the distance to a query point q not necessarily in S is called the nearest neighbour of q. The post-office (or nearest neighbour) problem requires storing S such that queries of the following kind can be answered with little effort: Given a query point q, determine a nearest neighbour of q. In non-degenerate cases, q has a unique nearest neighbour. In Figure 1, the site b is the nearest neighbour of q.

This problem is motivated from database theory and cluster analysis. After being mentioned in [Kn] it became heavily examined in computational geometry. The first solution that answers a query in optimal, that is, O(logn) time was suggested in [ShH]: For each site s compute the locus of points q such that s is a nearest neighbour of q. This defines a subdivision of E^2, namely the so-called (order-1) Vo- ronoi diagram of S. A query can now be answered by locating a query point, that is, by determining the region of the subdivision that contains it. Anticipating results on constructing Voronoi diagrams and on point location, this leads to a solution with O(n) space, O(nlogn) preprocessing time, and O(logn) query time.

2. (Order-1) Voronoi diagrams.

For any two points s and t of S, let H(s,t) be the closed halfspace of points at least as close to s as to t. V(s)={q|q in H(s,t), t≠s in S} is called the Voronoi polygon of s. Being the intersection of n-1 halfplanes, V(s) is a convex polygon with at most n-1 edges. The

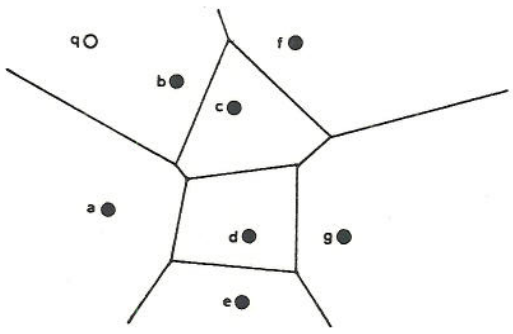

Figure 1: Order-1 Voronoi diagram.

totality of Voronoi polygons for all sites in S makes up the (order-1) Voronoi diagram 1-VOD(S) of S. Recall the equivalence of this definition to the one give in Section 1. 1-VOD(S) consists of regions, edges, and vertices (see Figure 1). Following from Euler's theorem on plane graphs, 1-VOD(S) has at most 3n-6 edges and 2n-4 vertices, for $n \geq 3$.

(Order-1) Voronoi diagrams are known in mathematics since [V]. Optimal algorithms for construction are given in [ShH] and [Br]. Borrowing from [Br] and [EOS], we propose the construction via the following geometric transform: Embed E^2 in the xy-plane of E^3 and associate each site $s=(s_x,x_y)$ with the halfspace $h(s)$: $z \geq 2s_x x+2s_y y-(s_x^2+s_y^2)$. (The plane bounding $h(s)$ is tangent to the paraboloid P: $z=x^2+y^2$ and touches P at the vertical projection of s onto P.) 1-VOD(S) can now be obtained by vertical projection of $\bigcap_{s \text{ in } S} h(s)$ onto the xy-plane.

3. Point location search.

Let G be a subdivision induced by a plane graph with m edges. G consists of regions, edges, and vertices. The point location search problem requires storing G in a data structure that supports efficient answering of the following kind of queries: For a query point q find the region (or edge or vertex) of G that contains q. An obvious application of such a data structure is to solve the post-office problem using the Voronoi diagram of the sites. Few of the numerous other applications can be found in [K] and [EGS].

The first optimal solution (that requires O(m) space and O(logm) query time) unfortunately without any practical significance was published in [LT]. Simpler yet optimal data structures well attractive for practical use were later suggested in [K] and [EGS]. Lack of space prevents us from describing the latter two substantially different approaches.

4. Higher-orc

Such as the
search, there
neighbour sea
report the k
H(t,s), t in
of non-empty
the order-k V
Voronoi diagr
the regions
nearest neighl

To the best
invented by
regions, edg
algorithm to
method derive
Section 2: F
$(s_x^2+s_y^2)$ tou
of s onto P.
we let a(e)
vertical proj
a(e)=k-1 yiel

5. Convex hu

For S a set

4. Higher-order Voronoi diagrams.

Such as the order-1 Voronoi diagram serves for nearest neighbour search, there is a generalization of the diagram to solve k-nearest neighbour search for fixed positive integer k: For a query point q report the k sites closest to q. For $T \subseteq S$, we call $V(T)=\{q \mid q \text{ in } H(t,s), t \text{ in } T \text{ and } s \text{ in } S-T\}$ the Voronoi polygon of T. The totality of non-empty Voronoi polygons of subsets T of S with $|T|=k$ is called the order-k Voronoi diagram k-VOD(S) of S. Figure 2 shows the order-3 Voronoi diagram for the point-set of Figure 1. Where space permits the regions are labelled with the three nearest sites. The three nearest neighbours of the query point q as depicted are a, b, and c.

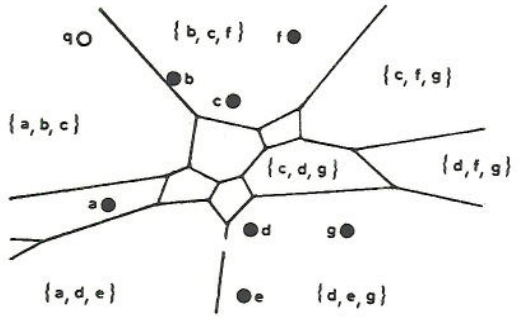

Figure 2: Order-3 Voronoi diagram.

To the best knowledge of the author, order-k Voronoi diagrams are invented by [ShH]. [L] shows that k-VOD(S) consists of $O(k(n-k))$ regions, edges, and vertices, and also gives an $O(k^2 n \log n)$ time algorithm to construct it. A different and intuitively more appealing method derives from using the geometric transform also outlined in Section 2: For each site $s=(s_x,s_y)$, the plane $p(s)$: $z=2s_x x+2s_y y-(s_x^2+s_y^2)$ touches the parapoloid $P: z=x^2+y^2$ in the vertical projection of s onto P. For each edge e in the arrangement A of planes obtained, we let a(e) denote the number of planes strictly above e. The vertical projection onto the xy-plane of the skeleton of edges e with $a(e)=k-1$ yields the order-k Voronoi diagram ([EOS]).

5. Convex hulls.

For S a set of n points in E^d, the smallest convex body that con-

tains S is called the convex hull CH(S) of S ([G]). In E^2, CH(S) is a bounded and convex polygon with at most n edges, in E^3 it is a convex polytope with at most n vertices, 3n-6 edges, and 2n-4 faces, for $n \geq 3$. Several worst-case optimal (that is O(nlogn) time) algorithms are known in E^2 ([Gr], [PH], and others), while the divide-and-conquer approach described in [PH] yields the only O(nlogn) time method in E^3. On a rather coarse level, it reads as follows:

If |S|=1 then CH(S)=p with S={p}
else DIVIDE: Let S_L contain the n/2 leftmost points of S and let $S_R=S-S_L$.
RECURSION: Compute $CH(S_L)$ and $CH(S_R)$.
MERGE: Derive CH(S) from $CH(S_L)$ and $CH(S_R)$ by "wrapping paper" around both polytopes.

Besides being an interesting problem for itself, constructing convex hulls can be exploited to compute intersections of halfspaces as e.g. required in Section 2 to build order-1 Voronoi diagrams: Each halfspace $h: z \geq ax+by+c$ is transformed into the dual vertical ray r(h) extending downwards from the point top(h)=(a/2,b/2,-c). Computing the lower half of the convex hull of all points top(h) is equivalent to constructing the forbidden space FS for the planes that intersect all rays r(h). Since these planes correspond to the points in the intersection of the halfspaces h, FS is dual to the required intersection of halfspaces. As a consequence, the O(nlogn) time algorithm for convex hulls implies an O(nlogn) time algorithm for FS and for order-1 Voronoi diagrams. Figure 3 illustrates the dual transform and the correspondence between FS and the intersection of halfplanes in E^2.

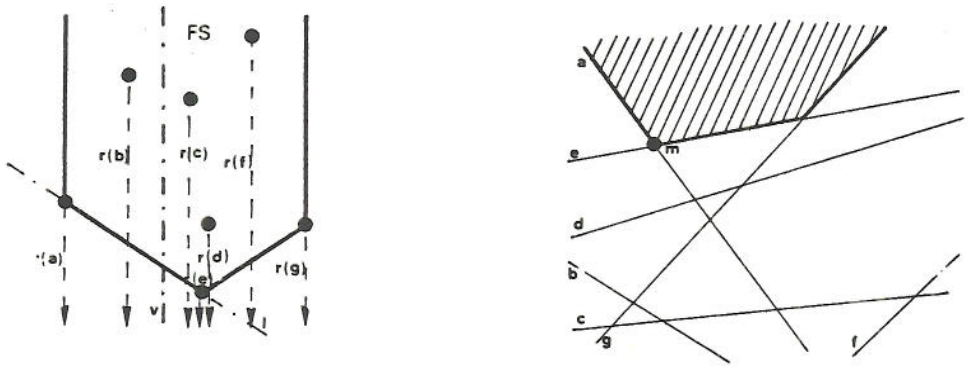

Figure 3: Lower convex hull and intersection of halfplanes in E^2.

6. Linear pro

The linear pr
straints in E
linear target
considered to
For our prese
bounded below
the y-coordina

If |H|
else

Although linea
sion, the gec
time.

Dualizing two
following pro
vertical line
it such that
lower edge of
these observa
constructing (

7. Arrangeme

If H denotes
complex A(H)
combinatorial
ber of k-dime
Thus, $\Omega(n^d)$ i
that is, of
pointers to

6. Linear programming.

The linear programming problem involves a set H of n linear constraints in E^d and asks for finding the point m that maximizes a linear target function while satisfying all constraints in H. If d is considered to be a constant then O(n) time algorithms exist ([M]). For our presentation, we assume that d=2, all halfplanes in H are bounded below by a non-vertical line, and m is required to minimize the y-coordinate. Then the strategy of [M] is roughly as follows:

> If $|H| \leq 3$ then use a trivial algorithm
> else organize the halfplanes in pairs, and find the median x-coordinate x^* of the thus defined n/2 apices. Examining the slope of $f(x)=\min\{y|(x,y) \text{ in } h, h \text{ in } H\}$ at x^* allows us to decide whether m is to the left of, on, or to the right of $x=x^*$. Based on this decision, $m=(x^*,f(x^*))$ or n/4 of the halfplanes of H can be eliminated. In the latter case, the process is repeated recursively.

Although linear time is needed to carry out one level of the recursion, the geometric regression of |H| guarantees O(n) overall runtime.

Dualizing two-dimensional linear programming allows us to solve the following problem in O(n) time: For a set S of points in E^2 and a vertical line v, find the line l that has all points of S above or on it such that $l \cap v$ has maximal y-coordinate. Obviously l defines the lower edge of CH(S) that intersects v (see Figure 3). Building on these observations, [KS] developed an O(nlogV) time algorithm for constructing CH(S) if V is the number of its vertices.

7. Arrangements of hyperplanes.

If H denotes a set of n hyperplanes in E^d, then we call the cell complex A(H) induced by H the arrangement of H. Classical results in combinatorial geometry (see [G]) give exact upper bounds on the number of k-dimensional faces of A(H), for $0 \leq k \leq d$, which are in $\Theta(n^d)$. Thus, $\Omega(n^d)$ is a lower bound for the explicit construction of A(H), that is, of the incidence lattice that provides each k-face with pointers to the incident (k-1)-faces and (k+1)-faces. This lower

bound matches the upper bound $O(n^d)$ which is achieved in [EOS] following an incremental strategy to build A(H), $H=\{h_1,h_2,\ldots h_n\}$:

For i=1 to n do
construct $A(\{h_1,\ldots h_i\})$ by inserting h_i into $A(\{h_1,\ldots h_{i-1}\})$.

To insert a hyperplane h into an arrangement A of i hyperplanes needs only to check and update those faces of A that are incident with a face intersecting h. Figure 4 depicts an arrangement in E^2 with the solid edges to be checked at the insertion of a new line h shown as broken line.

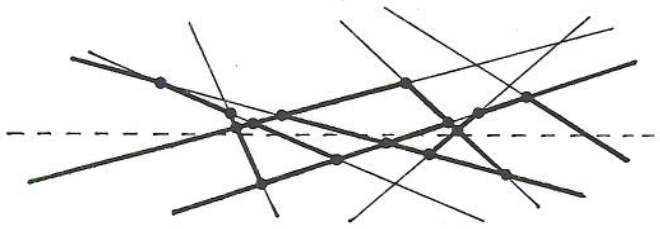

Figure 4: Two-dimensional arrangement.

As the number of such faces is in $O(i^{d-1})$ ([EOS]), h can be inserted in $O(i^{d-1})$ time implying the $O(n^d)$ time bound for the sketched construction of A(H).

Arrangements in E^3 can be used to construct order-1 and higher-order Voronoi diagrams, convex hulls in E^3 (which is, however, quite a detour), and other structures. It also proves useful as a thinking paradigm for problems on finite sets of points.

8. Halfplanar range search.

Given a set S of n points in E^2, a halfplanar range query specifies a halfplane h and asks for the number N(h) of points of S in h. The halfplanar range search problem requires storing S in a data structure that supports halfplanar range queries. The most efficient O(n) space solution (give in [EW]) builds on the following geometric fact:

Let b be a bisecting line of S, that is, $2|S^+| \leq |S|$ and $2|S^-|) \leq |S|$, for S^+ (S^-) the set of points strictly above (below) b. Then there exists a conjugate line c, that is, c bisects S^+ and S^-.

Recursively dividing two sets separated by a line using a conjugate line defines a tree that stores S in O(n) space. Since the boundary of a query halfplane intersects at most three of four sectors defined by two crossing lines, the query time in this tree is governed by $Q(n)=Q(n/2)+Q(n/4)+O(1)=O(n^{\emptyset.695})$ (see [EW]).

The connection of this data structure to the material in Section 7 is provided by the requirement to find conjugate lines. By duality, conjugate lines that contain at least two points correspond to particular vertices in the two-dimensional arrangement dual to S. At the time, the most efficient strategy to find conjugate lines is given in [CSY] running in $O(n\log^2 n)$ time.

II. KEY-METHODS

Several rather general ideas and methods are used to solve the problems presented in Part I. It is the goal of this part to explicate these paradigms in a reasonably general way. The first two paradigms (locus approach and geometric transformation) are rather general ideas that turn out to be useful in approaching a large number of computational problems. Different in nature and closer to actually writing programs are the divide-and-conquer paradigm, the elimination method, and the incremental approach treated in the final three sections.

9. Locus approach.

The locus approach is the idea of subdividing some space into domains of constant answer ([O2]). A necessary assumption is therefore that the answer depends on a point in the same space; in other cases a geometric transform might be used to obtain this situation. This principle is responsible for the close relationship between nearest and k-nearest neighbour search and Voronoi diagrams (see Sections 1,

2, and 4). The importance of the locus approach can also be recognized by its independent development in other areas: e.g. [LP] calls the same principle the configuration approach to motion and location planning. The significance of the locus approach implies the importance of the point location search problem (see Section 3), since locating a point is the obvious operation one needs to perform in a subdivided space.

10. Geometric transformation.

Geometric construction gains surprisingly often from transforming the problem into some space and problem different from the originally given ones. The advantages that a suitable geometric transformation can provide are:

1. additional insight into the problem which might be well-obscured in the original setting, and
2. the possibility to use one program to solve several problems.

The transformations that turn out to be useful a good number of times are the dual transform and the inversion used to embed a d-dimensional problem in E^{d+1}. For both types of transformation, various different formulas exist that realize the desired properties. The motivation for using the particular dual mapping described in Sections 5 and 6 is based on its independence from the origin (as opposed to the classical formula [G]) and the fact that it is an involution (unlike the mapping most often used in computational geometry [Br]). Similar reasons lead is to use the particular embedding of two-dimensional problems in E^3 outlined in Sections 2, 4 and 7.

11. Divide-and-conquer.

The divide-and-conquer paradigm is applicable to problems that are, in some little understood sense, decomposable. If the principle applies then it implies a fairly large part of the algorithm's structure. We provide a loose outline of this invariant structure. S denotes a set of data and c a positive constant which can often be chosen equal to one.

> If $|S| \leq c$ then solve the problem using a trivial algorithm
> else DIVIDE: Split S into two roughly equally large sets S_1 and S_2.

RECURSION: Solve the problem recursively for S_1 and S_2.
MERGE: Combine the solutions for S_1 and S_2 to compute the solution for S.

Section 5 provides an example of this classical paradigm ([AHU]).

12. Elimination.

Equally concrete as the divide-and-conquer paradigm is the elimination method that implies the following kind of algorithm:

If $|S| \leq c$ then solve the problem using a trivial algorithm
else ELIMINATE: Find a constant proportion of S that is irrelevant and eliminate it.
RECURSION: Solve the problem recursively for the reduced set S.

This method is applicable to problems whose solution is determined by at most c of the data-items, the other items being irrelevant or redundant. An example is provided in Section 6 where redundant or for the target function not significant constraints are eliminated. The key-step in the algorithm is the elimination part. To achieve efficiency in this step, computing some kind of a median [AHU] turns out to be extremely useful. The geometric regression of |S| implies overall efficiency.

13. Incremental approach.

Geometric properties of certain problems favour a rather straightforward method called the incremental approach. Recently, it was shown to lead to optimal algorithms for constructing convex hulls in E^d (if d is even, [S]) and for constructing arrangements of hyperplanes in E^d ([EOS]). Let $S=\{d_1,\ldots d_n\}$ be a set of data and let C(S) denote the structure to be constructed. The incremental approach suggests the following strategy:

For i=1 to n do
construct $C(\{d_1,\ldots d_i\})$ by inserting d_i into $C(\{d_1,\ldots d_{i-1}\})$.

So the insertion of an item into an already existing structure is the crucial step which determines the efficiency of the algorithm. An example for this approach is given in Section 7.

DISCUSSION

Eight interrelated central problems from computational geometry are discussed, and the general ideas that lead to efficient solutions for these problems are explicated. By no means we claim any exhaustive treatment of computational geometry. In fact, large classes of problems like rectangle problems [E] or hidden line problems [Sch], and several design techniques like the plane-sweep technique [NP] or dynamization methods for static data structures [O1], are not mentioned. Our primary goal thus remains to offer an introduction to an appealing part of where geometry and algorithms meet.

REFERENCES

[AHU] Aho,A.V., Hopcroft,J.E., and Ullman,J.D. The design and analysis of computer algorithms. Addison-Wesley, Reading, Mass., 1974.

[Br] Brown,K.Q. Geometric transforms for fast geometric algorithms. Ph.D.Thesis, Rep. CMU-CS-80-101, Dep.Comp.Sci., Carnegie-Mellon Univ., Pittsburgh, Penn., 1980.

[CSY] Cole,R., Sharir,M., and Yap,C.K. Convex k-hulls and related problems. Manuscript.

[E] Edelsbrunner,H. Intersection problems in computational geometry. Ph.D.Thesis, Rep.F93, Inst.Inf.Proc., TU Graz, Austria, 1982.

[EGS] Edelsbrunner,H., Guibas,L.J., and Stolfi,J. Optimal point location in a monotone subdivision. Manuscript.

[EOS] Edelsbrunner,H., O'Rourke,J., and Seidel,R. Constructing arrangements of lines and hyperplanes with applications. Proc. 24th Ann. Symp. Found. Comp. Sci. (1983), 83-91.

[EW] Edelsbrunner,H. and Welzl,E. Halfplanar range search in linear space and $O(n^{0.695})$ query time. Rep.F111, Inst.Inf. Proc., TU Graz, Austria, 1983.

[Gr] Graham,R.L. An efficient algorithm for determining the

convex hull of a finite planar set. Inf. Proc. Lett. (1972), 132-133.

[G] Gruenbaum,B. Convex polytopes. Pure and applied math. XVI, Interscience, London, 1967.

[K] Kirkpatrick,D.G. Optimal search in planar subdivisions. SIAM J. Comp. 12 (1983), 28-35.

[KS] Kirkpatrick,D.G. and Seidel,R. The ultimate planar convex hull algorithm? Proc. 20th Ann. Allerton Conf. Comm., Contr., Comp. (1982), 35-42.

[Kn] Knuth,D.E. Retrieval on secondary keys. Chapter 6.5 in Knuth, D.E.: Sorting and searching - the art of computer programming III. Addison-Wesley, Reading, Mass., 1973.

[L] Lee,D.T. On k-nearest neighbor Voronoi diagrams in the plane. IEEE Trans. Comp. C-31 (1982), 478-487.

[LT] Lipton,R.J. and Tarjan,R.E. Application of a planar separator theorem. Proc. 18th Ann. Symp. Found. Comp. Sci. (1977), 162-170.

[LP] Lozano-Perez,T. Spatial planning: a configuration space approach. IEEE Trans. Comp. C-32 (1983), 108-120.

[M] Megiddo,N. Linear-time algorithms for linear programming in R^3 and related problems. Proc. 23rd Ann. Symp. Found. Comp. Sci. (1982), 329-338.

[NP] Nievergelt,J. and Preparata,F.P. Plane-sweep algorithms for intersecting geometric figures. Comm. ACM 25 (1982), 739-747.

[O1] Overmars,M.H. The design of dynamic data structures. Lect. Notes Comp. Sci. 156, Springer, 1983.

[O2] Overmars,M.H. The locus approach. Rep. RUU-CS-83-12, Dep. Comp.Sci., Univ. Utrecht, the Netherlands, 1983.

[PH] Preparata,F.P. and Hong,S.J. Convex hulls of finite sets of points in two and three dimensions. Comm. ACM 20 (1977), 87-93.

[Sch] Schmitt,A. On the time and space complexity of certain exact hidden line algorithms. Rep. 24/81, Fak.Inf., Univ. Karlsruhe, Germany, 1981.

[S] Seidel,R. A convex hull algorithm optimal for point sets in even dimensions. Rep. 81-14, Dep.Comp.Sci., Univ. British Columbia, Vancouver, 1981.

[Sh] Shamos,M.I. Computational geometry. Ph.D.Thesis, Dep.Comp. Sci., Yale Univ., New Haven, Conn., 1978.

[ShH] Shamos,M.I. and Hoey,D. Closest-point problems. Proc. 16th Ann. Symp. Found. Comp. Sci. (1975), 151-162.

[V] Voronoi,G. Sur quelques proprietes des formes quadratiques positives parfaites. J. reine angew. Math. 133 (1907), 97-178.