Geometric range searching [1]

(Survey — Extended Abstract)

Jiří Matoušek
Department of Applied Mathematics
Charles University, Praha, Czech Republic

Abstract

In geometric range searching, algorithmic problems of the following type are considered: Given an n-point set P in the plane, build a data structure so that, given a query triangle R, the number of points of P lying in R can be determined quickly. Problems of this type are of crucial importance in computational geometry, as they can be used as subroutines in many seemingly unrelated algorithms. We survey some results and techniques in this area.

Geometric range searching

Jiří Matoušek
Department of Applied Mathematics
Charles University, Praha, Czech Republic

Abstract

幾何学的領域探索に関連して次のアルゴリズム上の問題を考える。平面上にn点の集合Pが与えられているとき、質問の三角形Rに対してRの中に含まれるPの点の個数が素早く求められるようなデータ構造を構築せよ。この種の問題は、一見関係がなさそうな多数のアルゴリズムにおいてサブルーティンとして使われるので、計算幾何学において極めて重要である。本文ではこの分野における結果と技法について概観する。

[1]This is a shortened version of a survey paper [Mat93a]. Part of the work on this paper was done while the author was visiting the Computer Science Institute, Free University Berlin. The work has been supported by the German-Israeli Foundation of Scientific Research and Development (G.I.F.).

1 Introduction

A basic example of a geometric range searching problem was mentioned in the abstract; let us describe a more general setting. Let $\mathcal{R}$ be a system of subsets of the d-dimensional Euclidean space $\mathbb{R}^d$. The sets of $\mathcal{R}$ are called *ranges.* Typically considered basic cases are $\mathcal{R}_{orthog}$ (the axis-parallel boxes, i.e. cartesian products of intervals), $\mathcal{R}_{halfsp}$ (the (closed) halfspaces), $\mathcal{R}_{simplex}$ (the (closed) simplices) and $\mathcal{R}_{ball}$ (the (closed) balls). Further let P be a given n-point set in $\mathbb{R}^d$. One of the geometric range searching problems is to design an efficient algorithm which, for a given range $R \in \mathcal{R}$, finds the number of points of P lying in R. The point set P is given in advance, and we can prepare some auxiliary information about it and store it in a suitable data structure (this phase is called the *preprocessing*). Then we will be repeatedly given various ranges of $\mathcal{R}$ as queries. Each such query is to be answered as soon as it appears (on-line), and as efficiently as possible.

A range searching problem of this type is usually considered with some limitations on the amount of storage for the data structure and on the time for the preprocessing. As we will see, the more space and preprocessing time we have, the faster can the queries be answered (at least in a suitable range of the parameters).

Counting points in a given range (a *range counting query*) is only one of possible range searching problems. Another natural problem is to compute a list of all points of P lying in a query range (we speak of a *range reporting query*), or we can only ask if the query range contains any point of P at all (*range emptiness query*). Also, each points of P can be assigned some weight (e.g., a real number), and we can be interested in the sum of weights of points in a given range, or in the maximum weight.

All these problems are quite convincingly motivated by direct practical applications, most often in various database systems. However, it seems that even more important than such direct applications are applications of geometric range searching as subroutines in the design of algorithms for more complicated geometric problems.

In more recent papers, one usually investigates a unifying generalization of various range searching problems. We assume that every point $p \in P$ is assigned a weight $w(p) \in S$, where $(S, +)$ is some semigroup (common to all the points). The objective of a query is to find the sum of weights of all points of P lying in a given range, $\sum_{p \in R \cap P} w(p)$. For example, for range counting queries, $(S, +)$ will be the natural numbers with addition, and all weights will be equal to 1. For queries on maximum weight, the appropriate semigroup will be the real numbers with the operation of taking a maximum of two numbers, etc. In the sequel we assume that the weights can be stored in a single computer word and that the semigroup operation can be executed in constant time.

The range reporting queries have a somewhat special position. Their query complexity is usually expressed in the form $O(f(n) + k)$, where k is the number of points in the answer, and $f(n)$ is some function of the total number of points in P.

Let us remark that reporting queries and queries with weights which can be subtracted often allow a simpler and/or more efficient solution than the general case, whose prototype are the queries asking for maximum weight. For the case of subtraction, we can usually express the answers for more complicated ranges using the answers for several simpler ranges. As a simplest one-dimensional example, we note that a query interval in $\mathbb{R}^1$ (which has 2 degrees of freedom) can be expressed as the difference of two semiinfinite intervals (which have only 1 degree of freedom).

In this paper we mainly consider the simplex and halfspace range searching problems. These problems turned out to be crucial in computational geometry, they are even universal in some sense, since many other problems with more general ranges can be reduced to them, see below (for the equally important orthogonal range searching problems, with axis-parallel boxes as ranges, which are not a subject of this survey, we refer to [Cha88], [Cha90a, Cha90b] for a more recent work and references).

2 Intuition and lower bounds

Results about the computational complexity of simplex range searching can be summarized as follows:

Let us consider a simplex range searching problem for an n-point set $P \subset \mathbb{R}^d$, with weights from a semigroup $(S, +)$, and with storage and preprocessing time at most m, where m lies in the range from n to approximately n^d. Then the query time is approximately

$$\frac{n}{m^{1/d}}. \tag{1}$$

The word "approximately" in the previous sentence means "up to a multiplicative factor bounded by $O(\log^c n)$, c a constant."

In particular, for an approximately linear storage the query time is approximately $n^{1-1/d}$, and in order to achieve a polylogarithmic query time, one needs space and preprocessing approximately n^d. The complexity of halfspace range queries is believed to be very similar, except for few special cases, as e.g., halfspace emptiness queries, which can be handled more efficiently.

First we will try to give the reader some intuitive explanation where the formula (1) comes from. The explanation is quite far from a proof, and in reality the lower and upper bounds work in a more complicated manner. We consider the two extreme cases, polylogarithmic query time and roughly linear space.

Logarithmic query time. First we consider halfspace queries. It is not difficult to see that for an n-point set P in a general position there are $\Theta(n^d)$ different subsets of the form $P \cap R$, where R is a halfspace (this is best seen in the dual setting, where distinct subsets correspond to distinct cells in an arrangement of hyperplanes). Storage of the order n^d thus means that we can store the answers for all essentially different halfspaces that can ever appear. Actual algorithms are indeed based on this principle. A naive attempt on extending this idea to simplex range searching results in a much larger space than n^d. A suitable method preserving storage close to n^d is more complicated and was discovered only recently [CSW92].

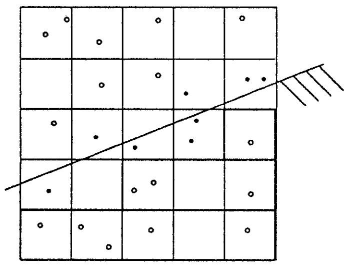
Figure 1: A simple halfspace range searching method, for uniformly distributed point sets

Approximately linear storage. Here we will assume that the set P is chosen randomly, by n independent random draws from the uniform distribution in the unit square (we consider the planar case first). We put $t = \lfloor\sqrt{n}\rfloor$ and we cover the unit square by a $t \times t$ square grid, each grid square having side $1/t$, see Fig. 1. With high probability, almost every grid square then contains only a small number of points of P (bounded by a constant).

Let R be a given halfplane. We note that the boundary line h of R intersects at most $2t$ squares of the grid (if its slope is at most 1, the it intersects at most 2 squares in every column, and for slope > 1 we apply a similar argument with rows).

For the squares intersected by h we go through all the points of P lying in them, and for each such point we test its membership in R. The uniform distribution implies that the number of points processed in this phase is $O(t) = O(\sqrt{n})$ (such points are marked as full circles in Fig. 1) .

In remains to account for the weight of points in grid squares which are completely contained in R. This can be done row by row, using the fact that such squares form a contiguous interval in every row. The total weights of points in each such segment of each row are computed in advance, thus we only need a constant time per row for the query answering. The total memory requirement is $O(n)$.

For a higher dimension d we can proceed quite similarly, dividing the unit cube into a grid of cubes with sides $n^{-1/d}$. The bounding hyperplane of a given halfspace R always intersects only $O(n^{1-1/d})$ grid cubes. The cubes completely contained in the query halfspace can be processed by columns parallel to one (arbitrarily chosen) coordinate axis. In this way we get a data structure with $O(n)$ storage and $O(n^{1-1/d})$ query time for uniformly distributed point sets in the unit cube, as required by formula (1). This time also the generalization to simplex range searching is straightforward (we omit the details). For point sets which are not uniformly distributed this simple approach fails, and all known methods with query time close to $n^{1-1/d}$ are considerably more complicated.

Lower bounds. Chazelle [Cha89] proved lower bounds for the query time for a simplex range searching algorithm with a given amount of space. These bounds are formulated in the so-called *arithmetic model* which puts certain limitations on the type of algorithm used (the exact definitions are complicated so we omit them in this extended abstract). In principle, a better algorithm might exist which does not respect these restrictions (for instance, the lower bounds do not apply for algorithms using weight subtraction, that is, if $(S,+)$ is a group), although this doesn't seem very likely at present.

Chazelle proved that *for any semigroup $(S,+)$ (satisfying a certain very mild assumption), for any fixed dimension d and parameters n, m there exists an n-point set $P \subset \mathbb{R}^d$ such that the simplex range searching problem with point set P, weights from S and storage at most m has query complexity at least*

$$\Omega\left(\frac{n}{\log n \; m^{1/d}}\right) \tag{2}$$

(for $d \geq 3$), resp. at least

$$\Omega\left(\frac{n}{\sqrt{m}}\right) \tag{3}$$

for $d = 2$ in the arithmetic model.

It is quite likely that the bound (2) holds without the logarithmic factor in the denominator as well (as is the case in dimension 2). Such an improvement has an interesting relation to a generalization of a famous problem of combinatorial geometry (so-called Heilbronn problem), see [Cha89].

There are interesting particular cases where the above mentioned lower bound doesn't apply. Two such interesting cases are simplex emptiness queries and general halfspace range searching, which, however, seem to be approximately equally difficult as the general simplex range searching problem. This was partially substantiated in [BCP93] and [CR92]. A substantial improvement over the above general bound is only known for halfspace emptiness queries and for halfspace range reporting queries (see section 4).

The papers of Chazelle *et al.* on lower bounds may please the reader as a nice piece of mathematics, but the conclusions for the simplex range searching problem are quite pessimistic. For instance, if we want to improve the query complexity K-times compared to the trivial algorithm (consisting of inspection of every point of P) in dimension 10, it costs storage of the order K^{10}. This indicates that nontrivial algorithms can thus be practically useful for a really small dimension only.

3 Simplex range searching algorithms

In this extended abstract we will consider only algorithms using linear or nearly linear storage, which are usually more complicated and more interesting. The opposite extreme are algorithms with a polylogarithmic query time which, as we have seen, require about n^d storage. These are usually based on point location in hyperplane arrangments, and we refer to the literature ([Cha93], [CSW92], [Mat93c]) for more information. Algorithms with memory requirements in between these two extremes can often be obtained by a more or less straightforward combination of algorithms of the two mentioned types.

Most of nontrivial algorithms with linear space are based on the idea of *partition trees* due to Willard [Wil82]. Willard's original algorithm uses the fact that an n-point set in the plane can be partitioned into 4 parts of approximately equal size by two lines ℓ_1, ℓ_2. Given a query halfplane, its boundary only intersects 3 of the 4 regions defined by ℓ_1 and ℓ_2. If we precompute the total weight of points in each region, we can process all points of the region missed by the boundary in a single step, thus saving 25% of the work on the query answering compared to the trivial method. This alone is not significant for the asymptotic complexity, but a similar saving can be repeated recursively in each of the remaining 3 regions. By continuing in a similar manner also in a larger depth, the complexity is decreased significantly. The resulting data structure storing such a recursive partition is called a partition tree.

This simplest scheme gives query time roughly $O(n^{0.792})$ using a linear space. The history of improvements and generalizations to higher dimensions resembles athletic records somewhat. Here we only list references to steps in this progress with few remarks: [EW86] (improvement in the plane), [Yao83] (a first algorithm in dimension 3), [DE84], [EH84], [YDEP89], [Col85], [YY85] (a first nontrivial algorithm in any fixed dimension), [HW87] (introducing probabilistic methods; partition schemes with a large number of regions), [Wel88] (a first almost optimal algorithm in dimensions 2 and 3, introduces a new, nonrecursive partition scheme known as *spanning tree with low crossing number* with many applications also outside range searching; a final version of this paper is [CW89]), [MW92], [CSW92] (a first nearly optimal algorithm for every fixed dimension), [Mat92b], [Mat93c] (current asymptotically best algorithms, technically somewhat complicated).

Let us outline a nearly optimal algorithm from [Mat92b]. It is based on a suitable partition scheme. Let P be an n-point set in $\mathbb{R}^d$; for simplicity we assume it is in general position. A *simplicial partition for* P is a collection $\Pi = \{(P_1, \Delta_1), \ldots, (P_m, \Delta_m)\}$, where the P_i are disjoint subsets of P (called the *classes*) forming a partition of P, and each Δ_i is a d-dimensional simplex containing the set P_i; see fig. 2.

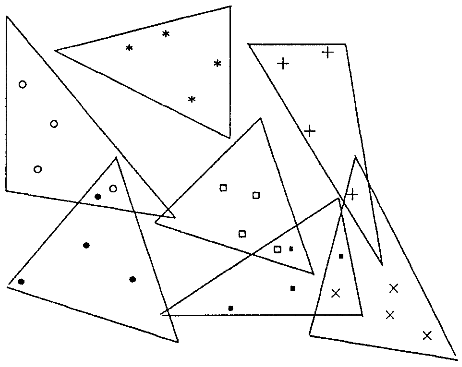

Figure 2: A simplicial partition (points of different classes are marked by different symbols).

The key geometric result is the following: *Let P be an n-point set in $\mathbb{R}^d$ ($d \geq 2$), r a parameter, $1 < r \leq n/2$. Then there exists a simplicial partition for P satisfying $n/r \leq |P_i| \leq 2n/r$ for every class P_i (thus with $O(r)$ classes), such that any hyperplane only intersects at most $\kappa = O(r^{1-1/d})$ simplices of Π.*

The value of κ in this result is asymptotically optimal. The paper [Mat92b] also gives an algorithm for constructing such a simplicial partition, with $O(n \log r)$ running time for sufficiently small r ($r \leq n^\beta$ for a certain small constant $\beta = \beta(d) > 0$).

By a recursive application of such a simplicial partition, a partition tree is created in a standard way. For a given set P we find a simplicial partition Π (with a suitable choice of the parameter r) and we store its simplices as well as the total weights of points in the classes of Π in the root of the partition tree. Each class of the simplicial partition corresponds to one subtree of the root, where the construction is used recursively for the points in the corresponding classes. When answering a query with a halfspace R, we process the simplices lying completely inside R or completely outside R directly in the current node, and for the simplices crossing the boundary of R we recursively descend into the appropriate subtrees.

The actual performance of the algorithm depends on the choice of the parameter r. If we choose $r = m^{\beta}$, where m is the number of points in the current node and $\beta < 1$ is a suitable small positive constant, we get a simple algorithm with a linear storage, $O(n \log n)$ preprocessing and $O(n^{1-1/d} \log^c n)$ query time, where c is some constant. A further small improvement of the query time, up to the probably optimal $O(n^{1-1/d})$, was given in [Mat93c] using similar ideas.

4 Extensions and examples of applications

Halfspace range reporting and other special situations. It seems that the query complexity for halfspace range reporting with space at most m might be approximately

$$\frac{n}{m^{1/\lfloor d/2 \rfloor}} + k\,, \tag{4}$$

where k denotes the number of points in the query halfspace (this is much better than the complexity of simplex range searching). In particular, in dimensions 2 and 3 algorithms exist with almost linear storage and $O(\log n + k)$ query time, see [CGL85], [CP86], [AHL90]. For a higher dimension, an algorithm achieving roughly the performance given by (4), up to a factor of n^{ε}, is obtained by combining the results of [Cla88] (in some situations one can get even closer to (4), up to logarithmic factors, see also [Sch92]). No lower bounds are known.

In the dual version of halfspace range reporting, we essentially restrict our attention to a single cell of the arrangement of the given hyperplanes. Another such special situation is when our point set lies on a fixed lower dimensional algebraic variety of bounded degree, or if all hyperplanes bounding the query ranges are tangent to such a variety; then the range searching results can sometimes be also improved, see e.g., [AM94] for a discussion. The situations with points on a surface is by no means rare — it arises e.g., when dealing with lines in 3-dimensional space, see e.g., [CEGS89].

Dynamization. Until now we have considered static range searching problems, where the point set is given once and for all. In many applications we need to insert new points or delete old ones from time to time. For a dynamic data structure storing n points we probably cannot expect a better time for one modification than a $1/n$ fraction of the time needed for building the whole (static) structure anew. For the simplex and halfspace range searching problems dynamic structures with this efficiency (up to small factors, typically of the order n^{ε}) are known, see [SO90], [AS93], [Mat92b]. Several authors investigated dynamic halfspace range reporting data structures under the assumption that the update sequence is random in a suitably defined sense and obtained very good update times for this case, see [Mul91b], [Mul91a], [Sch91]. In [AM91] a dynamic algorithm was found, which is efficient also in the worst case, for an arbitrary update sequence. This algorithm can be viewed as giving an implicit dynamic representation of the convex hull of the current point set (it allows to decide if a query point lies inside or outside the current convex hull, to compute a tangent hyperplane to the convex hull and containing a given line etc.).

Multilevel data structures. We explain the idea on an example. Let $S = \{s_1, \ldots, s_n\}$ be a set of segments in the plane. We want to construct a data structure which quickly computes the number of segments of S intersected by a query line h. We permit roughly linear space for the data structure. Let a_i, b_i be the endpoints of the segment s_i. A denotes the set of all a_i and B the set of all b_i. We consider computing the number of s_i such that a_i is above h and b_i is below h (the opposite case is solved symmetrically).

Let us consider some partition tree for the set A; for definiteness, let it be the partition tree based on simplicial partitions described in the previous section, with r being a large constant. Using such a partition tree we can determine, in roughly $\sqrt{n}$ time, the number of points of A in the halfplane R above h. This does not solve our problem yet, but we look more closely how the answer is obtained from the partition tree. The weight of every point from $A \cap R$ is accounted for in some of the visited nodes of the tree. In each such node, we find the simplices of the corresponding simplicial partition lying completely inside R, and the weights of their respective classes are accounted for as wholes.

For each node of the partition tree, let us call the classes of the simplicial partition stored there the *canonical sets.* We see that the partition tree provides a partition of the set $A \cap R$ into roughly $\sqrt{n}$ canonical sets of various sizes. The total number of canonical sets in the partition tree is $O(n)$, and the sum of their sizes is easily estimated to $O(n \log n)$.

For our problem with segments, we augment the partition tree for the set A as follows: For every canonical set $M \subseteq A$ we create a partition tree for the set $M' = \{b_i;\ a_i \in M\}$, and we store it with the corresponding node of the partition tree for A. Given a query line h in our problem, first express the set of points of A lying above h as a disjoint union of certain canonical subsets $M_1, \ldots, M_m$, and then for each such M_i we use the appropriate secondary partition tree to count the points of M_i' lying below the line h. Adding these counts together over all canonical sets M_i, we obtain the desired number of segments s_i with a_i above h and b_i below h.

A simple calculation we shows that the required space is $O(n \log n)$ only (this is because of the total size of the canonical subsets) and the query time remains still close to $\sqrt{n}$. Intuitively, this is because there are only few large canonical sets in the decomposition of $A \cap R$, and the computation on the second level is fast for small canonical sets.

The principle used in the above example is quite universal. Usually it is applicable whenever the query is a conjunction of several conditions (or, geometrically, an intersection of several regions) and for each condition (region) we already have a suitable efficient data structure. In this way e.g., an efficient simplex range searching algorithm with a polylogarithmic query time and about n^d space can be derived from a halfspace range searching algorithm with similar parameters, see [CSW92]. [Ben80], [DE87] are papers introducing data strures of this type, and from numerous recent application we mention e.g., [OSS90], [AS93].

Searching with more general ranges. So far we have considered range searching with ranges bounded by hyperplanes. Many applications naturally lead to searching in ranges with nonlinear, curved boundaries. The most important case are subsets of $\mathbb{R}^d$ defined by a conjunction of at most k polynomial inequalities of maximum degree D, where d, k and D are constants.

Perhaps the simplest among nonlinear ranges are circular disks in the plane and balls in higher dimensions. The corresponding range searching problem arises when we are interested in points lying in at most a given distance from a given point[2]; problems of this type are often referred to as *proximity problems*, see e.g., [CCPY86], [AHL90], [CW89] and others. Other group are problems dealing with lines in 3-space, which typically lead to nonlinear range searching as well (related works are e.g., [CEGS89], [Pel92], [PS92], [Aga93]).

An important early work considering very general geometric range searching problems is [YY85]. A recent paper on the subject is [AM94], which shows how techniques developed for simplex and halfspace range searching can also be applied in this more general setting. For a lack of space, we do not formulate the specific results here; we only remark that for such techniques, combinatorial complexity bounds for arrangements of algebraic surfaces and for their decompositions into constant complexity cells become important; such problems are treated e.g., in [CEGS91], [HS93], [Sha93].

Ray shooting and linear optimization. This is another type of generalization of geometric range searching problems. In a ray shooting problem we are given some set Γ of geometric objects (planes, triangles, balls, ...) and the goal is to construct a data structure such that for a given point o and direction θ we can quickly determine the object of Γ hit by a ray ρ sent from the point o in the direction θ. This problem is very popular in computer graphics, where it arises as an auxiliary problem in determining visibility of objects, hidden surface elimination, ray tracing and in other situations.

It turns out that the ray shooting problem can be solved efficiently using data structures for suitable derived range searching problems. This idea appears in several papers, e.g., [Berg]. A systematic approach using the so-called *parametric search* (which is an important algorithmic technique due to Megiddo [Meg83]) was suggested in [AM93] and demonstrated on several examples; see also [MS93] for related results.

Another related problem is to maximize a query linear function c over a convex polytope (which is fixed and can be preprocessed for this purpose). This is in fact a linear programming problem, but in a special situation, where the constraints are given in advance, while the optimized function comes as a query. In [Mat93b] it is shown, using a multidimensional version of parametric search, that algorithms for halfspace emptiness queries can be transformed to solve also this linear programming problem, with query complexity increased by a polylogarithmic factor only (and with the same data structure). With a dynamic halfspace emptiness data structure as a basis, one may also insert and delete constraints. This can be applied in a classical computational geometry problem, that of finding extremal points. The input is an n-point set $P \subset \mathbb{R}^d$, and we want to detect which points of P are vertices of the convex hull of P. This problem can be solved by computing a combinatorial representation of the convex hull of P, but for dimensions $d \geq 4$ this method is fairly inefficient, as the convex hull of an n-point set in $\mathbb{R}^d$ can have combinatorial complexity of the order $n^{\lfloor d/2 \rfloor}$. Testing whether a point is extremal can be formulated as a linear programming problem in dimension d with n constraints. For different points these linear programs only differ by two constraints, so we can use linear programming in the preprocessing/query mode. We build a data structure for the appropriate

[2]Notice that balls do not appear in this second, "application oriented" formulation directly. This is quite typical, as range searching problems with nonlinear ranges are usually obtained by a suitable re-formulation of the original specification of the problem.

constraints, and with its help we answer n queries, thereby determining the extremal points. In this way we obtain, for example, a total time $O(n^{4/3+\epsilon})$ in dimension 4, which is the most efficient known method. This nicely illustrates a remark made in the introduction, namely that a large part of applications of geometric range searching is in problems which are not of the preprocessing/query type. Further examples are hidden surface removal problems (see e.g. [Berg]), counting circular arc intersections [APS93], problems concerning lines in space (e.g. [Pel92]) to quote only few.

Conclusion. In this extended abstract we only touched some of the main issues in geometric range searching. We have largely neglected applications, various generalized types of queries which do not fall into our framework directly, and many other important topics. Some more details can be found in the full version [Mat93a], and for other material we cannot but refer to the literature.

Acknowledgment. I would like to thank Pankaj K. Agarwal and Raimund Seidel for reading preliminary versions of this paper and numerous useful comments.

References

[Aga93] P. K. Agarwal. On stabbing lines for polyhedra in 3d. Tech. Rept. CS-1993-09, Department Computer Science, Duke University, 1993.

[AHL90] A. Aggarwal, M. Hansen, and T. Leighton. Solving query-retrieval problems by compacting Voronoi diagrams. In *Proc. 22nd Annu. ACM Sympos. Theory Comput.*, pages 331–340, 1990.

[AM91] P. K. Agarwal and J. Matoušek. Dynamic halfspace range reporting and its applications. Tech. Report CS-1991-43, Duke University, 1991. Extended abstract, including also results of D. Eppstein: *Proc. 33. IEEE Symposium on Foundations of Computer Science* (1992), pages 51–60.

[AM94] P. K. Agarwal and J. Matoušek. On range searching with semialgebraic sets. *Discrete Comput. Geom.* 11:1994, 393–418.

[AM93] P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. *SIAM J. Comput.*, 22(4):794–806, 1993.

[APS93] P. K. Agarwal, M. Pellegrini, and M. Sharir. Counting circular arc intersections. *SIAM Journal on Computing*, 22:778–793, 1993.

[AS93] P. K. Agarwal and M. Sharir. Applications of a new partitioning scheme. *Discr. & Comput. Geom.* 9:1993, 11–38.

[BCP93] H. Brönnimann, B. Chazelle, and J. Pach. How hard is halfspace range searching. *Discrete Comput. Geom.*, 10:143–155, 1993.

[Ben80] J. L. Bentley. Multidimensional divide-and-conquer. *Commun. ACM*, 23(4):214–229, 1980.

[CCPY86] B. Chazelle, R. Cole, F. P. Preparata, and C. K. Yap. New upper bounds for neighbor searching. *Inform. Control*, 68:105–124, 1986.

[CEGS89] B. Chazelle, H. Edelsbrunner, L. J. Guibas, and M. Sharir. Lines in space: combinatorics, algorithms, and applications. In *Proc. 21st Annu. ACM Sympos. Theory Comput.*, pages 382–393, 1989.

[CEGS91] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. A singly-exponential stratification scheme for real semi-algebraic varieties and its applications. *Theoret. Comput. Sci.*, 84:77–105, 1991.

[CGL85] B. Chazelle, L. J. Guibas, and D. T. Lee. The power of geometric duality. *BIT*, 25:76–90, 1985.

[Cha88] B. Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM J. Comput.*, 17:427–462, 1988.

[Cha89] B. Chazelle. Lower bounds on the complexity of polytope range searching. *J. Amer. Math. Soc.*, 2:637–666, 1989.

[Cha90a] B. Chazelle. Lower bounds for orthogonal range searching, I: the reporting case. *J. ACM*, 37:200–212, 1990.

[Cha90b] B. Chazelle. Lower bounds for orthogonal range searching, II: the arithmetic model. *J. ACM*, 37:439–463, 1990.

[Cha93] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9(2):145–158, 1993.

[Cla88] K. L. Clarkson. Applications of random sampling in computational geometry, II. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 1–11, 1988.

[Col85] R. Cole. Partitioning point sets in 4 dimensions. In *Proc. 12th Internat. Colloq. Automata Lang. Program.*, volume 194 of *Lecture Notes in Computer Science*, pages 111–119. Springer-Verlag, 1985.

[CP86] B. Chazelle and F. P. Preparata. Halfspace range search: an algorithmic application of k-sets. *Discrete Comput. Geom.*, 1:83–93, 1986.

[CR92] B. Chazelle and B. Rosenberg. Lower bounds on the complexity of simplex range reporting on a pointer machine. In *International Colloquium on Automata, Languages and Programming*, 1992. Also to appear in *Computational Geometry: Theory and Applications.*

[CSW92] B. Chazelle, M. Sharir, and E. Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Algorithmica*, 8:407–429, 1992.

[CW89] B. Chazelle and E. Welzl. Quasi-optimal range searching in spaces of finite VC-dimension. *Discrete Comput. Geom.*, 4:467–489, 1989.

[Berg] M. de Berg, D. Halperin, M. Overmars, J. Snoeyink, and M. van Kreveld. Efficient ray shooting and hidden surface removal. *Algorithmica.*

To appear. Extended abstract: Proc. 7. ACM Symposium on Computational Geometry, pages 21–30, 1991.

[DE84] D. Dobkin and H. Edelsbrunner. Organizing point sets in two and three dimensions. Tech. Report F130, Technische Universität Graz, 1984.

[DE87] D. Dobkin and H. Edelsbrunner. Space searching for intersecting objects. *Journal of Algorithms*, 8:348–361, 1987.

[EH84] H. Edelsbrunner and F. Huber. Dissecting sets of points in two and three dimensions. Report F138, Inst. Informationsverarb., Tech. Univ. Graz, Graz, Austria, 1984.

[EW86] H. Edelsbrunner and E. Welzl. Halfplanar range search in linear space and $O(n^{0.695})$ query time. *Inform. Process. Lett.*, 23:289–293, 1986.

[HS93] D. Halperin and M. Sharir. New bounds for lower envelopes in three dimensions, with applications to visibility in terrains. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 11–18, 1993.

[HW87] D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete Comput. Geom.*, 2:127–151, 1987. J. Matoušek. Reporting points in halfspaces. *Comput. Geom. Theory Appl.*, 2(3):169–186, 1992.

[Mat92b] J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8:315–334, 1992.

[Mat93a] J. Matoušek. Geometric range searching. Tech. Report B-93-09, Fachbereich Mathematik und Informatik, Free Univ. Berlin, 1993.

[Mat93b] J. Matoušek. Linear optimization queries. *J. Algorithms*, 14:432–448, 1993. The results combined with results of O. Schwarzkopf also appear in *Proc. 8th ACM Sympos. Comput. Geom.*, 1992, pages 16–25.

[Mat93c] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete & Computational Geometry*, 10(2):157–182, 1993.

[Meg83] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *Journal of the ACM*, 30:852–865, 1983.

[MS93] J. Matoušek and O. Schwarzkopf. On ray shooting in convex polytopes. *Discrete & Computational Geometry*, 10(2):215–232, 1993.

[Mul91a] K. Mulmuley. Randomized multidimensional search trees: further results in dynamic sampling. In *Proc. 32nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 216–227, 1991.

[Mul91b] K. Mulmuley. Randomized multidimensional search trees: lazy balancing and dynamic shuffling. In *Proc. 32nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 180–196, 1991.

[MW92] J. Matoušek and E. Welzl. Good splitters for counting points in triangles. *J. Algorithms*, 13:307–319, 1992.

[OSS90] M. Overmars, H. Schipper, and M. Sharir. Storing line segments in partition trees. *BIT*, 30:385–403, 1990.

[Pel92] M. Pellegrini. Incidence and nearest-neighbor problems for lines in 3-space. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 130–137, 1992.

[PS92] M. Pellegrini and P. Shor. Finding stabbing lines in 3-space. *Discrete Comput. Geom.*, 8:191–208, 1992.

[Sch91] O. Schwarzkopf. Dynamic maintenance of geometric structures made easy. In *Proc. 32nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 197–206, 1991.

[Sch92] O. Schwarzkopf. Ray shooting in convex polytopes. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 286–295, 1992.

[Sha93] M. Sharir. Almost tight upper bounds for lower envelopes in higher dimensions. In *Proc. 34th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS 93)*, pages 498–507, 1993.

[SO90] H. Schipper and M. H. Overmars. Dynamic partition trees. In *Scandawian Workshop on Algorithms Theory*, volume 2, pages 404–417. Springer-Verlag, 1990. LNCS 447; also to appear in *BIT*.

[Wel88] E. Welzl. Partition trees for triangle counting and other range searching problems. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 23–33, 1988.

[Wil82] D. E. Willard. Polygon retrieval. *SIAM J. Comput.*, 11:149–165, 1982.

[Yao83] F. F. Yao. A 3-space partition and its applications. In *Proc. 15th Annu. ACM Sympos. Theory Comput.*, pages 258–263, 1983.

[YDEP89] F. F. Yao, D. P. Dobkin, H. Edelsbrunner, and M. S. Paterson. Partitioning space for range queries. *SIAM J. Comput.*, 18:371–384, 1989.

[YY85] A. C. Yao and F. F. Yao. A general approach to *D*-dimensional geometric queries. In *Proc. 17th Annu. ACM Sympos. Theory Comput.*, pages 163–168, 1985.