CS-1996-05

Range Searching[1]

Pankaj K. Agarwal[2]

Department of Computer Science

Duke University

Durham, North Carolina 27708–0129

September 8, 1996

[1]Work ion this paper has been supported by National Science Foundation Grant CCR-93–01259, by an Army Research Office MURI grant DAAH04-96-1-0013, by a Sloan fellowship, by an NYI award, and by matching funds from Xerox Corporation.
[2]Department of Computer Science, Duke University, `pankaj@cs.duke.edu`

RANGE SEARCHING

Pankaj K. Agarwal

INTRODUCTION

Range searching is one of the central problems in computational geometry, because it arises in many applications and a wide variety of geometric problems can be formulated as a range-searching problem. A typical range-searching problem has the following form. Let S be a set of n points in $\mathbb{R}^d$, and let $\mathcal{R}$ be a family of subsets; elements of $\mathcal{R}$ are called *ranges*. We wish to preprocess S into a data structure so that for a query range R, the points in $S \cap R$ can be reported or counted efficiently. Typical examples of ranges include rectangles, halfspaces, simplices, and balls. If we are only interested in answering a single query, it can be done in linear time, using linear space, by simply checking for each point $p \in S$ whether p lies in the query range. However, most of the applications call for querying the same point set S several times (or sometimes we also insert or delete a point periodically), in which case we would like to answer a query faster by preprocessing S into a data structure.

Range counting and *range reporting* are just two instances of range-searching queries. Other examples include *emptiness queries*, where one wants to determine whether $S \cap R = \emptyset$, and *extremal queries*, where one wants to return a point with certain property (e.g., returning a point with the largest x_1-coordinate). In order to encompass all different types of range-searching queries, a general range-searching problem can be defined as follows. Let $(\mathbf{S}, +)$ be a semigroup. For each point $p \in S$, we assign a weight $w(p) \in \mathbf{S}$. For a query range $R \in \mathcal{R}$, we wish to compute $\sum_{p \in S \cap R} w(p)$. For example, range-counting queries can be answered by setting $w(p) = 1$ for every $p \in S$ and choosing the semigroup to be $(\mathbb{Z}, +)$, where $+$ denotes the integer addition; range-emptiness queries by setting $w(p) = 1$ and choosing the semigroup to be $(\{0, 1\}, \vee)$; and range-reporting queries by setting $w(p) = \{p\}$ and choosing the semigroup to be $(2^S, \cup)$.

Most of the range-searching data structures construct a family of 'canonical' subsets of S, and for each canonical subset C, they store the weight $w(A) = \sum_{p \in A} w(p)$. For a query range r, the data structure searches for a small subfamily of disjoint canonical subsets, $A_1, \ldots, A_k$, so that $\bigcup_{i=1}^{k} A_i = r \cap S$, and then computes $\sum_{i=1}^{k} w(A_i)$. In order to expedite the search, the structure also stores some auxiliary information. Typically, the canonical subsets are organized in a tree-like data structure, each of whose node v is associated with a canonical subset A; v stores the weight $w(A)$ and some auxiliary information. A query is answered by searching the tree in a top-down fashion, using the auxiliary information to guide the search.

MODEL OF COMPUTATION

The performance of a data structure is measured by the time spent in answering a query, called the *query time* and denoted by $Q(n, d)$; by the size of the data structure, denoted by $S(n, d)$; and by the time constructed in the data structure, called the *preprocessing time* and denoted by $P(n, d)$. Since the data structure is constructed only once, its query time and size are more important than its preprocessing time. If a data structure supports insertion and deletion operations, the *update time* is also relevant. We should remark that the query time of a range-reporting query on any reasonable machine depends on the output size, so the query time for a range-reporting query consists of two parts — *search time*, which depends only on n and d, and *reporting time*, which depends on n, d, and the output size. Throughout this survey paper we will use k to denote the output size.

We assume that d is a small fixed constant, and that the big-Oh notation hides constants depending on d. The dependence on d of the performance of all the data structures mentioned here is exponential, which makes them unsuitable for large values of d. We assume that each memory cell can store $\log n$ bits. The upper bounds will be given on *pointer-machine* or *RAM* models, which are described in [15, 107]. The main difference between the two models is that on the pointer machine a memory cell can be accessed only through a series of pointers while in the RAM model any memory cell can be accessed in constant time. Most of the lower bounds will be given in the so-called *semigroup model*, which was originally introduced by Fredman [61] and which is much weaker than the pointer machine or the RAM model. In the arithmetic model, a data structure is regarded as a set of precomputed sums in the underlying semigroup. The size of the data structure is the number of sums stored, and the query time is the number of semigroup operations performed (on the precomputed sums) to answer a query; the query time ignores the cost of various auxiliary operations, e.g., the cost of determining which of the precomputed sums should be added to answer a query. A weakness of the semigroup model is that it does not allow subtractions even if the weights of points belong to a group. Therefore, we will also consider the *group model*, in which both additions and subtractions are allowed.

The size of any range-searching data structure is at least linear, for it has to store each point (or its weight) at least once, and the query time on any reasonable model of computation (e.g., pointer machine, RAM) is $\Omega(\log n)$ even for $d = 1$. Therefore, one would like to develop a linear-size data structure with logarithmic query time. Although near-linear-size data structures are known for orthogonal range searching in any fixed dimension that can answer a query in polylogarithmic time, no similar bounds are known for range searching with more complex ranges (e.g., simplex, disks). In such cases, one seeks for a tradeoff between the query time and the size of the data structure — how fast can a query be answered using $n \log^{O(1)} n$ space, how much space is required to answer a query in $\log^{O(1)} n$ time, or what kind of tradeoff between the size and the query time can be achieved?

The chapter is organized as follows. In Section 32.1 we review the orthogonal range-searching data structures, and in Section 32.2 we review simplex range-searching data structures. Section 32.3 surveys other variants and extensions of range searching. We study intersection-searching problems in Section 32.4, which can be regarded as a generalization of range searching. Finally, Section 32.5 deals with various optimization queries.

1 ORTHOGONAL RANGE SEARCHING

In the d-dimensional orthogonal range searching, the ranges are d-rectangles, each of the form $\prod_{i=1}^{d}[a_i, b_i]$, where $a_i, b_i \in \mathbb{R}$. This is an abstraction of the 'multi-key' searching; see [21, 115]. For example, the points of S may correspond to employees of a company, each coordinate corresponding to a key such as age, salary, experience, etc. The queries of the form — report all employees between the ages of 30 and 40 who earn more than \$30,000 and who have worked for more than 5 years — can be formulated as an orthogonal range-reporting query. Because of its numerous applications, orthogonal range searching has been studied extensively for the last 25 years. A survey of earlier results can be found in the books by Mehlhorn [88] and Preparata and Shamos [99]. In this section we review the more recent data structures and the lower bounds.

GLOSSARY

EPM A pointer machine with + operation.

APM A pointer machine with basic arithmetic and shift operations.

Faithful semigroup A semigroup $(\mathbf{S}, +)$ is called faithful if for each $n > 0$, for any $T_1, T_2 \subseteq \{1, \dots, n\}$ so that $T_1 \neq T_2$, and for every sequence of integers $\alpha_i, \beta_j > 0$ $(i \in T_1, j \in T_2)$, there are $s_1, s_2, \dots, s_n \in \mathbf{S}$ such that

$$\sum_{i \in T_1} \alpha_i s_i \neq \sum_{j \in T_2} \beta_j s_j.$$

Notice that $(\mathbb{R}, +)$ is a faithful semigroup, but $(\{0,1\}, \oplus)$ is not a faithful semigroup.

UPPER BOUNDS

Most of the recent orthogonal range-searching data structures are based on *range trees*, introduced by Bentley [20]. For $d = 1$, the range tree of S is an array storing S in a nondecreasing order. For $d > 1$, let S_1 be the sequence of x-coordinates of points in S sorted in a nondecreasing order. The range tree of S is a minimum-height binary tree with n leaves, whose i-th leftmost leaf stores the point of S with the i-th smallest x_1-coordinate. For an interior node v of T, let $S(v)$ denote the set of points stored at leaves in the subtree rooted at v, let a_v (resp. b_v) be the smallest (resp. largest) x_1-coordinate of points in $S(v)$, and let $S^*(v)$ denote the projection of $S(v)$ onto the hyperplane $x_1 = 0$. The interior node v stores a_v, b_v, and a $(d-1)$-dimensional range tree constructed on $S^*(v)$. For any fixed dimension d, the size of T is $O(n \log^{d-1} n)$, and it can be constructed in time

$O(n \log^{d-1} n)$. The range-reporting query for a rectangle $q = \prod_{i=1}^{d} [a_i, b_i]$ can be answered as follows. If $d = 1$, the query can be answered by a binary search. For $d > 1$, we traverse the range tree as follows. Suppose we are at a node v. If v is a leaf, then we report the point if it lies inside q. If v is an interior node and the interval $[a_v, b_v]$ does not intersect $[a_1, b_1]$, there is nothing to do. If $[a_v, b_v] \subseteq [a_1, b_1]$, we recursively search in the $(d-1)$-dimensional range tree stored at v, with the rectangle $\prod_{i=2}^{d} [a_i, b_i]$. Otherwise, we recursively visit both children of v. The query time of this procedure is $O(\log^d n + k)$, which can be improved to $O(\log^{d-1} n + k)$, using the *fractional-cascading* technique [40, 77]. A range tree can also answer a range- counting query in time $O(\log^{d-1} n)$.

The best-known data structures for orthogonal range searching are by Chazelle [25, 27], who used compressed range trees and other techniques (such as filtering search) to improve the storage and query time. His results in the plane, under various models of computation, are summarized in Table 1; the preprocessing time of each data structure is $O(n \log n)$.

TABLE 1 Summary of planar orthogonal range-searching results

Problem	Model	$S(n)$	$Q(n)$
Counting	RAM	n	$\log n$
	APM	n	$\log n$
	EPM	n	$\log^2 n$
Reporting	RAM	n	$\log n + k \log^{\varepsilon}(2n/k)$
		$n \log \log n$	$\log n + k \log \log(4n/k)$
		$n \log^{\varepsilon} n$	$\log n + k$
	APM	n	$k \log(2n/k)$
	EPM	n	$k \log^2(2n/k)$
		$\frac{n \log n}{\log \log n}$	$\log n + k$
Semigroup	Arithmetic	m	$\frac{n \log n}{\log 2m/n}$
	RAM	n	$\log^{2+\varepsilon} n$
		$n \log \log n$	$\log^2 n \log \log n$
		$n \log^{\varepsilon} n$	$\log^2 n$
	APM	n	$\log^3 n$
	EPM	n	$\log^4 n$

All the results mentioned in Table 1 can be extended to higher dimensions at a cost of $\log^{d-2} n$ factor in the preprocessing time, storage, and query-search time. Table 2 summarizes a few additional results on higher-dimensional orthogonal range-searching results.

Overmars [96] showed that if S is a subset of a $u \times u$ grid U in the plane and the

TABLE 2 Higher-dimensional orthogonal range reporting

$S(n)$	$Q(n)$	Source	Notes
$n\log^{d-1+\varepsilon} n$	$\left(\frac{\log n}{\log\log n}\right)^{d-1} + k$	[88]	Pointer machine
m	$\left(\frac{\log n}{\log 2m/n}\right)^{d-1}$	[30]	Semigroup model
$\frac{n\log^{d-1} n}{\log\log n}$	$\frac{\log^{d-1} n}{\log\log n} + k$	[114]	Fusion trees
$n\log^{d-1} n$	$\log^{d-2} n\log^* n + k$	[101]	P^*-trees

vertices of query rectangles are also a subset of U, then a range-reporting query can be answered in time $O(\sqrt{\log u}+k)$, using $O(n\log n)$ storage and preprocessing; or in $O(\log\log u + k)$ time, using $O(n\log n)$ storage and $O(u^3\log u)$ preprocessing. The range-tree-based data structures for orthogonal range searching can be extended to handle c-oriented ranges. The performance of such a data structure is the same as that of a c-dimensional orthogonal range-searching structure. If the ranges are homothets of a given triangle, or translates of a convex polygon with constant number of edges, a two-dimensional range-reporting query can be answered in $O(\log n + k)$ time using linear space [35, 36]. If the ranges are octants in $\mathbb{R}^3$, a range-reporting query can be answered in either $O((k+1)\log n)$ or $O(\log^2 n + k)$ time using linear space [36].

LOWER BOUNDS

Fredman [59, 60, 61] was the first to prove nontrivial lower bounds on orthogonal range searching, but he considered the framework in which the points were allowed to insert and delete dynamically. He showed that a mixed sequence of n insertions, deletions, and queries takes $\Omega(n\log^d n)$ time. These bounds were extended by Willard to a group model, under some fairly restrictive assumptions.

Yao [117] proved a lower bound for the 2D static orthogonal range searching data structures. He showed that if only m units of storage is available, a query, under the semigroup model, takes $\Omega(\log n/\log((m/n)\log n))$ in the worst case. See also [108]. Later Chazelle extended the lower bound to higher dimensions and improved it slightly [30]. In particular he showed that

Theorem 1 (Chazelle [30]) *Let $(\mathbf{S},\oplus)$ be a faithful semigroup. Then for any fixed dimension d and parameters n, m, there exists a set S of n weighted points in $\mathbb{R}^d$, with weights from $\mathbf{S}$, such that the worst-case query time, under the semigroup model, for an orthogonal range-searching data structure, using m units of storage, is $\Omega((\log n/\log(2m/n))^{d-1})$.*

In fact, Chazelle's lower bound holds even for the average-case complexity. A rather surprising result of Chazelle [29] shows that the size of any data structure on a pointer machine that answers a d-dimensional range-reporting query in $O(\log^c n+k)$ time, for any constant c, is $\Omega(n(\log n/\log\log n)^{d-1})$; see also [16]. Notice that this lower bound is greater than the known upper bound on the RAM model (see Table 1).

These lower bounds do not hold for off-line orthogonal range searching, where given a set of n weighted points in $\mathbb{R}^d$ and a set of n rectangles, one wants to compute the weight of points in each rectangle. Recently, Chazelle [34] proved that the off-line version takes $\Omega(n(\log n/\log\log n)^{d-1})$ time in the semigroup model, and $\Omega(n\log\log n)$ time in the group model.

RELATED PROBLEMS

- *Partial-sum problem:* Preprocess a d-dimensional array A with n entries in an additive semigroup into a data structure so that for a d-dimensional rectangle $q=[a_1,b_1]\times\cdots\times[a_d,b_d]$, the sum $\sigma(A,q)=\sum_{(k_1,\ldots,k_d)\in q} A[k_1,\ldots,k_q]$ can be computed efficiently. In the off-line version, given A and m rectangles $q_1,\ldots,q_m$, we wish to compute $\sigma(A,q_i)$ for every $i\le m$. Yao [116] showed that, for $d=1$, a partial-sum query can be answered in $O(\alpha(n))$ time using $O(n)$ space. For $d>1$, Chazelle and Rosenberg [42] gave a data structure of size $O(n\log^{d-1} n)$ that can answer a query in time $O(\alpha(n)\log^{d-2} n)$. They also showed that the off-line version takes $\Omega(n+m\alpha(m,n))$ time for any fixed $d\ge 1$; here $\alpha(m,n)$ is the inverse Ackerman function. If points are allowed to insert into S, the query time is $\Omega(\log n/\log\log n)$ [117] for the one-dimensional case; the bounds were extended by Chazelle [30] to $\Omega((\log n/\log\log n)^d)$, for any fixed dimension d.
- *Rectangle-rectangle searching:* Preprocess a set S of n rectangles in $\mathbb{R}^d$ so that for a query rectangle q, the rectangles of S that intersect q can be reported (or counted) efficiently. Chazelle [27] has shown that the bounds mentioned in Table 1 hold for this problem also.

OPEN PROBLEMS

1. No nontrivial lower bounds are known for answering emptiness queries.
2. Chazelle's lower bound for range-reporting on the pointer-machine model does not hold if the query time is allowed to be of the form $O((k+1)\log^c n)$.
3. Better lower bounds under the group model.
4. Better lower bounds on the partial-sum problem for $d>1$.

2 SIMPLEX RANGE SEARCHING

In the last few years, simplex range searching has received considerable attention because, apart from its direct applications, the simplex range-searching data structures have provided fast algorithms for numerous other geometric problems. See the survey paper by Matoušek [86] for an excellent review of the techniques developed for the simplex range searching.

Unlike orthogonal range searching, no simplex range-searching data structure is known that can answer a query in polylogarithmic time using near-linear storage. In fact, the lower bounds stated below indicate that there is very little hope of obtaining such a data structure, for the query time of a linear-size data structure, under the semigroup model, is roughly at least $n^{1-1/d}$ (thus only saving a factor of $n^{1/d}$ over the naive approach). Since the size and query time of any data structure have to be at least linear and logarithmic, respectively, we consider these two ends of the spectrum: (i) How fast can a simplex range query be answered using a linear-size data structure, and (ii) how large should the size of a data structure be in order to answer a query in logarithmic time. By combining these two extreme cases, as mentioned below, one can obtain a space/query-time tradeoff.

GLOSSARY

Range space A range space is a set system $\Sigma = (X, R)$ where X is a set of objects and R is a family of subsets of X. The elements of R are called ranges of Σ. Σ is called a *finite range space* if X is finite.

ε-net A subset $N \subseteq X$ is called an ε-net of a finite range space Σ if $N \cap r \neq \emptyset$ for every $r \in R$ with $|r| \geq \varepsilon|X|$.

VC-dimension The VC-dimension of a range space $\Sigma = (X, R)$ is d if there is no subset $A \subseteq X$ of size $d+1$ such that $\{A \cap r \mid r \in R\} = 2^A$.

Spanning tree A spanning tree of a point S in $\mathbb{R}^d$ is a tree T whose vertices are the points of S and the edges are line segments connecting their endpoints. The *stabbing number* of T is the maximum number of its edges crossed by a hyperplane.

Arrangements The arrangement of a set H of hyperplanes in $\mathbb{R}^d$ is the subdivision of $\mathbb{R}^d$ into cells of dimensions k, for $0 \leq k \leq d$, each cell being a maximal connected set contained in the intersection of a fixed subset of H and not intersecting any other hyperplane of H.

$1/r$-cutting Let H be a set of n hyperplanes in $\mathbb{R}^d$ and let $1 \leq r \leq n$ be a parameter. A $(1/r)$-cutting of H is a set of (relatively open) disjoint simplices covering $\mathbb{R}^d$ so that each simplex intersects at most n/r hyperplanes of H.

Duality The dual of a point $(a_1, \ldots, a_d) \in \mathbb{R}^d$ is the hyperplane $x_d = a_1x_1 + \cdots + a_{d-1}x_{d-1} + a_d$, and the dual of a hyperplane $x_d = b_1x_1 + \cdots + b_d$ is the point $(b_1, \ldots, b_{d-1}, -b_d)$.

LOWER BOUNDS

In a series of papers, Chazelle has proved nontrivial lower bounds on the simplex range searching, using various elegant mathematical techniques; see Table 3. The following theorem is perhaps the most interesting result on lower bounds.

TABLE 3 Lower bounds for simplex range searching.

Range	Model	$S(n)$	$Q(n)$	Source
Simplex	Semigroup ($d = 2$)	m	$\frac{n}{\sqrt{m}}$	[28]
	Semigroup ($d > 2$)	m	$\frac{n}{m^{1/d}\log n}$	[28]
	Group	$n \log n$	$\log n$	[33]
	Pointer (Reporting)	m	$\frac{n^{1-\varepsilon}}{m^{1/d}} + k$	[43]
Halfspace	Semigroup	m	$\left(\frac{n}{\log n}\right)^{\frac{d^2+1}{d^2+d}} \cdot \frac{1}{m^{1/d}}$	[23]

Theorem 2 (Chazelle [28]) *Let* $(\mathbf{S}, \oplus)$ *be a faithful semigroup. For any given parameters* n, m*, there exists a set* S *of* n *weighted points in* $\mathbb{R}^d$*, with weights from* $\mathbf{S}$*, such that the worst-case query time of any simplex range-searching data structure, under the semigroup model, using* m *units of storage, is* $\Omega(n/\sqrt{m})$ *for* $d = 2$*, and* $\Omega(n/(m^{1/d}\log n))$ *for* $d \geq 3$.

It should be pointed out that this theorem holds even for the average-case complexity and even if the query ranges are wedges or strips. Theorem 2 gives a lower bound for the simplex range-counting queries because $(\mathbb{Z}, +)$ is a faithful group, but not for emptiness queries. As we will see below, faster data structures are known for the halfspace-emptiness queries.

The lower bound under the pointer-machine model is by Chazelle and Rosenberg [43], and it holds only for range-reporting queries. No nontrivial lower bound was known under the group model until Chazelle's recent result [33].

LINEAR-SIZE DATA STRUCTURES

Most of the linear-size data structures for simplex range searching are based on the so-called *partition trees*, originally introduced by Willard [113]. His partition tree is a 4-way tree, constructed as follows. Let us assume that n is of the form 4^k for some integer k, and that the points of S are in general position. If $k = 0$, the tree consists of a single node that stores the coordinates of the only point in S. Otherwise, using the ham-sandwich theorem, we find two lines ℓ_1, ℓ_2 so that each quadrant Q_i, for $1 \leq i \leq 4$, induced by ℓ_1, ℓ_2 contains exactly $n/4$ points. The root stores the equations of ℓ_1, ℓ_2 and the value of n. For each quadrant, we recursively construct a partition tree for $S \cap Q_i$ and attach it as the i^{th} subtree of the root. The total size of the data structure is linear, and it can be constructed in $O(n \log n)$ time. A halfplane range-counting query can be answered as follows. Let h be a query halfplane. We traverse T, starting from the root, and maintain a global count. At each node v storing n_v nodes in its subtree, the algorithm performs the following step: If the line ∂h intersects the quadrant Q_v associated with v, we recursively visit the children of v. If $Q_v \cap h = \emptyset$, we do nothing. Otherwise, we add n_v to the global count. The quadrants associated with the four children of an interior node of T are induced by two lines, so ∂h intersects at most three of them, which implies that the query procedure does not explore the subtree of one of the children. Hence, the query time of this procedure is $O(n^{\log_4 3}) = O(n^{.792})$. A similar procedure can answer a simplex range-counting query within the same time bound, and a simplex range-reporting query in time $O(n^{.792} + k)$.

After a few initial improvements and extensions on Willard's data structure [55, 56, 49], a major breakthrough in simplex range searching was made by Haussler and Welzl [68]. They formulated the range searching in an abstract setting and, using elegant probabilistic methods, gave a randomized algorithm to construct a linear-size partition tree with $O(n^\alpha)$ query time, where $\alpha = 1 - \frac{1}{d(d-1)+1} + \varepsilon$ for any $\varepsilon > 0$. The constant of proportionality hidden in the big-O notation depends on the value of ε. The major contribution of their paper is the abstract framework and the notion of ε-nets. The following theorem gives a slightly stronger version of their main result.

Theorem 3 (Haussler-Welzl [68], Komlós et al. [75]) *For any finite range space (X, R) of VC-dimension d and for $0 < \varepsilon, \delta < 1$, if N is a subset of X obtained by*

$$\frac{d}{\varepsilon}\left(\log\frac{1}{\varepsilon} + 2\log\log\frac{1}{\varepsilon} + 3\right)$$

random independent draws, then N is an ε-net of (X, R) with probability at least $1 - e^{-d}$.

Theorem 3 implies that any finite range space of VC-dimension d has an ε-net of size $(1 + o(1))(d/\varepsilon)\log 1/\varepsilon$. The ε-nets have turned out to be a powerful tool in developing divide-and-conquer algorithms for several geometric problems and in learning theory; see the books by Mulmuley [94] and Anthony and Biggs [17].

Building on the theory developed by Haussler and Welzl, Welzl [111] proved that one can construct a spanning path of S of $O(n^{1-1/d}\log n)$ stabbing number; the bound was improved by Chazelle and Welzl [45] to $\Theta(n^{1-1/d})$. Preprocessing the sequence of weights of points along the path, using Yao's data structure for the partial-sum problem, one can obtain a linear-size data structure for simplex range searching, with $O(n^{1-1/d}\alpha(n))$ query time, under the semigroup model. But this technique does not give a linear-size data structure with $O(n^{1-1/d}\log n)$ query time, for $d \geq 3$, under any reasonable model of computation (e.g., pointer machine, RAM), See [1, 82, 112] for other applications of spanning trees with low stabbing number.

Matoušek and Welzl [81] gave an entirely different algorithm for the halfspace range-counting problem in the plane, using a combinatorial result of Erdős and Szekeres [57]. The query time of their data structure is $O(\sqrt{n}\log n)$, and it uses $O(n)$ space and $O(n^{3/2})$ preprocessing time. If subtractions are allowed, their algorithm can be extended to the triangle range-counting problem. This technique has also been applied to solve a number of related problems, including ray shooting and intersection searching [19].

The best-known linear-size data structure for simplex range searching, which almost matches the lower bounds mentioned above, is by Matoušek [85]. He showed that a simplex range-counting (resp. range-reporting) query in $\mathbb{R}^d$ can be answered in time $O(n^{1-1/d})$ (resp. $O(n^{1-1/d}+k)$). His algorithm is based on the following theorem.

Theorem 4 (Matoušek [83]) *Let S be a set of n points in $\mathbb{R}^d$, and let $1 < r \le n/2$ be a given parameter. Then there exists a family of pairs*

$$\Pi = \{(S_1, \Delta_1), \dots, (S_m, \Delta_m)\}$$

such that $S_i \subseteq S$ lies inside the simplex Δ_i, $n/r \le |S_i| \le 2n/r$, $S_i \cap S_j = \emptyset$ for $i \ne j$, and every hyperplane crosses at most $cr^{1-1/d}$ simplices of Π; here c is a constant. For constant values of r, Π can be constructed in $O(n \log r)$ time.

Using this theorem, a partition tree T can be constructed as follows. Each interior node v of T is associated with a subset $S_v \subseteq S$ and a simplex Δ_v containing S_v; the root of T is associated with S and $\mathbb{R}^d$. Choose r to be a sufficiently large constant. If $|S| \le 4r$, T consists of a single node, and it stores all points of S. Otherwise, we construct a family of pairs $\Pi = \{(S_1, \Delta_1), \dots, (S_m, \Delta_m)\}$ using Theorem 4. The root u stores the value of n. We recursively construct a partition tree T_i for each S_i and attach T_i as the i-th subtree of u. The root of T_i also stores Δ_i. The total size of the data structure is linear, and it can be constructed in time $O(n \log n)$. A simplex range-counting query can be answered in the same way as for Willard's partition tree. Since any hyperplane intersects at most $cr^{1-1/d}$ simplices of Π, the query time is $O(n^{1-1/d} \times n^{\log_r c})$; the $\log_r c$ factor can be reduced to any arbitrarily small positive constant ε by choosing r sufficiently large. Although the query time can be improved to $O(n^{1-1/d} \log^c n)$ by choosing r to be n^ε, a stronger version of Theorem 4, which was proved in [85], and some other sophisticated techniques are needed to obtain $O(n^{1-1/d})$ query time.

If the points in S lie on a k-dimensional algebraic surface of constant degree, a simplex range-counting query can be answered in time $O(n^{1-\gamma})$ using linear space, where $\gamma = 1/\lfloor (d+k)/2 \rfloor$ [6].

Since the query time of a linear-size simplex range-searching data structure is only $n^{1/d}$ factor faster than the naive method, researchers have developed practical data structures that work well most of the time. For example, Arya and Mount [18] have developed a linear-size data structure for answering approximate range-counting queries, in the sense that the points lying within distance $\delta \cdot \mathrm{diam}(\Delta)$ distance of the boundary of the query simplex Δ may or may not be counted. Its query time is $O(\log n + 1/\delta^{d-1})$. Overmars and van der Stappen [97] developed fast data structures for the special case in which the ranges are 'fat' and have bounded size. See [62, 70] for some other 'heuristic based' data structures.

We conclude this subsection by noting that better bounds can be obtained for the halfspace range-reporting problem, using the so-called *filtering search*; see Table 4.

DATA STRUCTURES WITH LOGARITHMIC QUERY TIME

For the sake of simplicity, we first consider the halfspace range-counting problem. Using a standard duality transform, this problem can be reduced to the

TABLE 4 Halfspace range-searching.

d	$S(n)$	$Q(n)$	Source	Notes
$d=2$	n	$\log n + k$	[41]	Reporting
$d=3$	$n \log n$	$\log n + k$	[14]	Reporting
$d=3$	n	$\log n$	[52]	Emptiness
$d>3$	$n \log\log n$	$n^{1-1/\lfloor d/2 \rfloor} \log^c n$	[79]	Reporting
$d>3$	n	$n^{1-1/d} 2^{O(\log^* n)}$	[79]	Emptiness

following problem: Given a set H of n hyperplanes, determine the number of hyperplanes of H lying above a query point. Since the same subset of hyperplanes lies above all points in a single cell of $A(H)$, the arrangement of H, we can answer a halfspace range-counting query by locating the cell of $A(H)$ that contains the point dual to the hyperplane bounding the query halfspace. The following theorem by Chazelle [31] yields an $O((n/\log n)^d)$ size data structure, with $O(\log n)$ query time, for halfspace range counting.

Theorem 5 (Chazelle [31]) *Given a set H of n hyperplanes and a parameter $r \leq n$, a $(1/r)$-cutting of H of size $O(r^d)$ can be computed in $O(nr^{d-1})$ time.*

The above approach can be extended to the simplex range-counting problem as well. That is, store the solution of every combinatorially distinct simplex (two simplices are combinatorially distinct if they do not contain the same subset of S). Since there are $\Theta(n^{d(d+1)})$ combinatorially distinct simplices, such an approach will require $\Omega(n^{d(d+1)})$ storage. Chazelle et al. [44] showed that the size can be reduced to $O(n^{d+\varepsilon})$, for any $\varepsilon > 0$, using a multi-level data structure. The space bound can be reduced to $O(n^d)$ by increasing the query time to $O(\log^{d+1} n)$ [85] . Halfspace range-reporting queries can be answered in $O(\log n + k)$ time, using $O(n^{\lfloor d/2 \rfloor + \varepsilon})$ space.

A space/query-time tradeoff can be attained by combining the linear-size and logarithmic query-time data structures, as described in [44, 85]. The results are summarized in Table 5.

OPEN PROBLEMS

1. Bridge the gap between upper and lower bounds in the group model.
2. Can the lower bound on the query time in Theorem 2 be improved to $n/m^{1/d}$?
3. Can a simplex range-counting query be answered in $O(\log n)$ time using $O(n^d)$ space?
4. Can a halfspace range-reporting query be answered in $O(n^{1-1/\lfloor d/2 \rfloor} + k)$ time using linear space?

TABLE 5 Space/query-time tradeoff.

Range	Mode	$Q(m,n)$
Simplex	Reporting	$\frac{n}{m^{1/d}} \log^{d+1} \frac{m}{n} + k$
	Counting	$\frac{n}{m^{1/d}} \log^{d+1} \frac{m}{n}$
Halfspace	Reporting	$\frac{n}{m^{1/\lfloor d/2 \rfloor}} \log^c n + k$
	Emptiness	$\frac{n}{m^{1/\lfloor d/2 \rfloor}} \log^c n$
	Counting	$\frac{n}{m^{1/d}} \log \frac{m}{n}$

3 VARIANTS AND EXTENSIONS

In this section we review some extensions of range-searching data structures, including semialgebraic range searching, dynamization, and external memory data structures.

GLOSSARY

Semialgebraic set A subset of $\mathbb{R}^d$ is called a real semialgebraic if it is obtained as a finite Boolean combination of sets of the form $\{f \geq 0\}$, where f is a d-variate polynomial.

Tarski cells A real semialgebraic set is called a Tarski cell if it is defined by a constant number of polynomials, each of constant degree.

SEMIALGEBRAIC RANGE SEARCHING

So far we assumed that the ranges were bounded by hyperplanes, but in many applications one has to deal with ranges bounded by nonlinear functions. For example, a query of the form — for a given point p and a real number r, find all points of S lying within distance r from p — is a range-searching problem in which ranges are balls.

As shown below, the ball range searching in $\mathbb{R}^d$ can be formulated as an instance of the halfspace range searching in $\mathbb{R}^{d+1}$. So a ball range-reporting (resp. range-counting) query in $\mathbb{R}^d$ can be answered in time $O(n/m^{1/\lceil d/2 \rceil} \log^c n + k)$ (resp. $O(n/m^{1/(d+1)} \log(m/n))$), using $O(m)$ space; a somewhat better performance can be obtained using a more direct approach; see Table 6. But relatively little is known about range-searching data structures for more general ranges.

A natural class of more general ranges is the family of Tarski cells. It suffices to consider the ranges bounded by a single polynomial because the ranges bounded by multiple polynomials can be handled using multi-level data structures. We assume that the ranges are of the form

$$\Gamma_f(a) = \{x \in \mathbb{R}^d \mid f(x,a) \geq 0\},$$

where f is a $(d+p)$-variate polynomial specifying the type of ranges (disks, cylinders, cones, etc.), and a is a p-tuple specifying a specific range of the given type (e.g., a specific disk). We will refer to the range-searching problem in which the ranges are from the set Γ_f as the *Γ_f-range searching.*

TABLE 6 Semialgebraic range counting

d	Range	$S(n)$	$Q(n)$	Source	Notes
$d=2$	Disk	$n \log n$	$\sqrt{n} \log n$	[45]	
	Tarski cell	n	$n^{1/2+\varepsilon}$	[6]	Partition tree
$d \geq 3$	Tarski cell	n	$n^{1-\frac{1}{2d-3}+\varepsilon}$	[6]	Partition tree
	Tarski cell	n	$n^{1-\frac{1}{\lambda}+\varepsilon}$	[6]	Linearization

One approach to answer Γ_f-range queries is to use *linearization*, originally proposed by Yao and Yao [118]. We represent the polynomial $f(x,a)$ in the form

$$f(x,a) = \psi_0(a) + \psi_1(a)\varphi_1(x) + \cdots + \psi_k(a)\varphi_k(x)$$

where $\varphi_1, \ldots, \varphi_k, \psi_0, \ldots, \psi_k$ are real functions. A point $x \in \mathbb{R}^d$ is mapped to the point

$$\varphi(x) = (\varphi_1(x), \varphi_2(x), \ldots, \varphi_k(x)) \in \mathbb{R}^k.$$

Then a range $\gamma_f(a) = \{x \in \mathbb{R}^d \mid f(x,a) \geq 0\}$ is mapped to a halfspace

$$\varphi^{\#}(a) : \{y \in \mathbb{R}^k \mid \psi_0(a) + \psi_1(a)y_1 + \cdots + \psi_k(a)y_k \geq 0\};$$

k is called the *dimension* of linearization. Agarwal and Matoušek [6] have described an algorithm for computing a linearization of smallest dimension. A Γ_f-range query can now be answered using a k-dimensional halfspace range-searching data structure.

For example, a circle with center (a_1, a_2) and radius a_3 in the plane can be regarded as a set of the form $\gamma_f(a)$, where $a = (a_1, a_2, a_3)$ and f is a 5-variate polynomial that can be written as

$$f(x_1, x_2, a_1, a_2, a_3) = [a_3^2 - a_1^2 - a_2^2] + [2a_1x_1] + [2a_2x_2] - [x_1^2 + x_2^2].$$

Thus, setting

$$\begin{aligned} \psi_0(a) = a_3^2 - a_1^2 - a_2^2, \quad & \psi_1(a) = 2a_1, \quad \psi_2(a) = 2a_2, \quad \psi_3(a) = -1 \\ & \varphi_1(x) = x_1, \quad \varphi_2(x) = x_2, \quad \varphi_3(x) = x_1^2 + x_2^2, \end{aligned}$$

we get a linearization of dimension 3. In general, balls in $\mathbb{R}^d$ admit a linearization of dimension $d+1$; cylinders in $\mathbb{R}^3$ admit a linearization of dimension 9. One of the most widely used linearization in computational geometry is the so-called *Plücker coordinates*, which map a line in $\mathbb{R}^3$ to a point in $\mathbb{R}^5$; see [39, 105] for more details on Plücker coordinates.

Agarwal and Matoušek [6] have also proposed another approach to answer Γ_f-range queries by extending Theorem 4 to Tarski cells and by constructing partition trees using this extension.

Table 6 summarizes the known results on Γ_f range-counting queries; here λ is the dimension of linearization.

DYNAMIZATION

All the data structures discussed above assumed S to be fixed, but in many applications one needs to update S dynamically — insert a new point into S or delete a point from S. One cannot hope to perform insert/delete operations on a data structure in less than $P(n)/n$ time, where $P(n)$ is the preprocessing time of the data structure. If we allow only insertions (i.e., a point cannot be deleted from the structure), the static data structure can be modified, using the standard techniques [22, 95], so that a point can be inserted in time $O(P(n)\log n/n)$ and a query can be answered in time $O(Q(n)\log n)$, where $Q(n)$ is the query time of the original static data structure; in some cases the logarithmic overheard in the query or update time can be avoided. Although these techniques do not extend to deletions, many range-searching data structures, such as orthogonal and simplex range-searching structures, can handle deletions at polylogarithmic or n^ε overhead in query and update time, by exploiting the fact that a point is stored at roughly $S(n)/n$ nodes [8]. Table 7 summarizes the known results on dynamic 2D orthogonal range-searching data structures; these results can be extended to higher dimensions at a cost of $\log^{d-2} n$ factor in the storage, in the query time, and in the update time. Klein et al. [74] have described an optimal data structure for a special case of 2D range-reporting where the query ranges are translates of a polygon.

TABLE 7 Dynamic 2D orthogonal range-searching

Mode	$S(n)$	$Q(n)$	$U(n)$	Source
Counting	n	$\log^2 n$	$\log^2 n$	[27]
Reporting	n	$k\log^2(2n/k)$	$\log^2 n$	[27]
	n	$n^\varepsilon + k$	$\log^2 n$	[104]
	$n\log n$	$\log n \log\log n + k$	$\log n \log\log n$	[89]
	$\frac{n\log n}{\log\log n}$	$\frac{\log^{2+\varepsilon} n}{\log\log n} + k$	$\frac{\log^2 n}{\log\log n}$	[104]
Semigroup	n	$\log^4 n$	$\log^4 n$	[27]

Since an arbitrary sequence of deletions is difficult to handle in general, researchers have examined whether a random sequence of insertions and deletions can be handled efficiently; see [92, 93, 102]. Mulmuley [92] has shown that there exists a dynamic halfspace range-reporting data structure that can process a random update sequence of length m in expected time $O(m^{\lfloor d/2 \rfloor + \varepsilon})$ and can answer a halfspace range-reporting query in time $O(k \log n)$. Agarwal and Matoušek [7] developed a dynamic data structure for halfspace range-reporting that can process an arbitrary update sequence efficiently; its update time is $O(n^{\lfloor d/2 \rfloor - 1 + \varepsilon})$, and it can answer a query in time $O(\log n + k)$. If we allow only $O(n \log n)$ space, then the query and update time become $O(n^{1 - 1/\lfloor d/2 \rfloor + \varepsilon} + k)$ and $O(\log n)$, respectively.

SECONDARY MEMORY STRUCTURES

If the input point set is rather large and does not fit into the main memory, then the data structure is stored in the secondary memory, and portions of it are moved to the main memory, as required. In this case the bottleneck is the time spent in transferring the data between main and secondary memory. We assume that the data is stored in the secondary memory in blocks of size B, where B is a parameter. Each access to the secondary memory transfers one block (i.e., B words), and we count this as one I/O operation. The size of a data structure is the number of blocks required to store it, and the query (resp. preprocessing) time is defined as the number of I/O operations required to answer a query (resp. to construct the structure). I/O-efficient orthogonal range-searching structures have received much attention recently, but most of the results are known only for the planar case.

Table 8 summarizes the known results on secondary-memory structures for orthogonal range searching; here $\beta(n) = \log\log\log_B n$. The data structure by Subramanian and Ramaswamy [106] for 3-sided queries supports insertion/deletion of a point in time $O(\log_B n + (\log_B n)^2/B)$. Extending the lower-bound proof by Chazelle [43], they also proved that any secondary-memory data structure that answers a range-reporting query in time $O(\log_B^c n + k/B)$ requires $\Omega((n/B)\log(n/B)/\log\log_B n)$ storage.

TABLE 8 Secondary memory structures

d	Range	$Q(n)$	$S(n)$	Source
$d=1$	Interval	$\log_B n + k/B$	n/B	
$d=2$	Quadrant	$\log_B n + k/B$	$(n/B)\log\log B$	[100]
	3-sided rect.	$\log_B n + k/B + \log^* B$	n/B	[106]
	3-sided rect.	$\log_B n + k/B$	$(n/B)\log B \log\log B$	[100]
	Rectangle	$\log_B n + k/B + \log^* B$	$(n/B)\log(n/B)/\log\log_B n$	[106]
$d=3$	Octant	$\beta(n,B)\log_B n + k/B$	$(n/B)\log(n/B)$	[110]
	Rectangle	$\beta(n,B)\log_B n + k/B$	$(n/B)\log^4(n/B)$	[110]

OPEN PROBLEMS

1. Can a ball range-counting query be answered in $O(\log n)$ time using $O(n^2)$ space?
2. Can a Γ_f range-counting query be answered in time $O(n^{1-1/d+\varepsilon})$ using near-linear space?
3. A solution to the following problem, which is interesting in its own right, will result in a better semialgebraic range-searching data structure: Given a set Γ of n algebraic surfaces in $\mathbb{R}^d$, each of constant maximum degree, decompose the cells of the arrangement into $O(n^d)$ Tarski cells.
4. If the hyperplanes bounding the query halfspaces satisfy some property, e.g., all of them are tangent to a given sphere, can a halfspace range-counting query be answered more efficiently?
5. Simple dynamic data structure for halfspace range reporting.
6. Efficient secondary-memory structures for higher dimensional orthogonal range searching and for simplex range searching.

4 INTERSECTION SEARCHING

A general intersection-searching problem can be formulated as follows: given a set S of objects in $\mathbb{R}^d$, a semigroup $(\mathbf{S}, +)$, and a weight function $w : S \to \mathbf{S}$; we wish to preprocess S into a data structure so that for a query object γ, we can compute the weighted sum $\sum w(p)$, where the sum is taken over all objects of S that intersect γ. Range searching is a special case of intersection-searching in which S is a set of points.

An intersection-searching problem can be formulated as a semialgebraic range-searching problem by mapping each object $p \in S$ to a point $\varphi(p)$ in a parametric space $\mathbb{R}^k$ and every query range γ to a semialgebraic set $\psi(\gamma)$ so that p intersects γ if and only if $\varphi(p) \in \psi(\gamma)$. For example, let S be a set of segments in the plane and the query ranges be also segments in the plane. Each segment $e \in S$ with left and right endpoints (p_x, p_y) and (q_x, q_y), respectively, can be mapped to a point $\varphi(e) = (p_x, p_y, q_x, q_y)$ in $\mathbb{R}^4$ and a query segment γ can be mapped to a semialgebraic region $\psi(\gamma)$ so that γ intersects e if and only if $\psi(\gamma) \in \varphi(e)$. A shortcoming of this approach is that k, the dimension of the parametric space, is typically much larger than d, and therefore, it does not leads to an efficient data structure. The efficiency can be significantly improved by expressing the intersection test as a conjunction of simple primitive tests (in low dimensions) and then using a multi-level data structure to perform these tests. For example, a segment γ intersects another segment e if the endpoints of e lie on the opposite sides of the line containing γ and vice-versa. We can construct a two-level data structure — the first level sifts the subset $S_1 \subseteq S$ of all the segments whose endpoints lie on the opposite side of the line supporting the query segment, and the second level reports those segments of S_1 whose supporting lines separate the endpoints of γ. Each level of this structure can be implemented using a two-dimensional simplex range-searching searching structure, and hence a reporting query can be answered in $O(n/\sqrt{m}\log^c n + k)$ time using $O(m)$ space.

It is beyond the scope of this survey paper to cover all intersection-searching problems. Instead, we discuss a few basic ones, which have been studied extensively. All intersection-counting data structures described here can answer intersection-reporting queries, at an additional cost that is proportional to the output size. In some cases, an intersection-reporting query can be answered faster. Moreover, using intersection-reporting data structures, intersection-detection queries can be answered in time proportional to their query-search time. Finally, all the data structures described in this section can be dynamized at an expense of $O(n^\varepsilon)$ factor in the storage and query time.

POINT INTERSECTION SEARCHING

Preprocess a set S of objects (e.g., balls, halfspaces, simplices, Tarski cells) in $\mathbb{R}^d$ into a data structure so that all the objects of S containing a query point can be reported (or counted) efficiently. This is the inverse of the range-searching problem. Moreover, it can also be viewed as locating a point in the subdivision induced by the objects in S. Table 9 gives some of the known results.

TABLE 9 Point intersection searching

d	Objects	$S(n)$	$Q(n)$	Source	Notes
$d=2$	Disks	m	$(n^{4/3}/m^{2/3})\log(m/n)$		Counting
	Disks	$n\log n$	$\log n + k$	[14]	Reporting
	Triangles	m	$\frac{n}{\sqrt{m}}\log^3 n$	[8]	Counting
	Fat triangles	$n\log^2 n$	$\log^3 n + k$	[73]	Reporting
	Tarski cells	$n^{2+\varepsilon}$	$\log n$	[37]	Counting
$d=3$	Functions	$n^{1+\varepsilon}$	$\log n + k$	[4]	Reporting
$d \geq 3$	Simplices	m	$\frac{n}{m^{1/d}}\log^{d+1} n$		Counting
	Balls	$n^{d+\varepsilon}$	$\log n$	[6]	Counting
	Balls	m	$\frac{n}{m^{1/\lceil d/2\rceil}}\log^c n + k$	[79]	Reporting
	Tarski cells	$n^{2d-3+\varepsilon}$	$\log n$	[37]	Counting

Point location in arrangement of surfaces, especially determining whether a query point lies above a given set of regions of the form $x_{d+1} \geq f(x_1, \ldots, x_d)$, has found many applications in computational geometry. For example, the *collision-*

detection problem — given a set O of obstacles and a robot B, determine whether a placement p of B is free — can be formulated as a point intersection-detection query amid a set of regions. If B has k degrees of freedom, then a placement of B can be represented as a point in $\mathbb{R}^k$, and the set of placements of B that intersect an obstacle $O_i \in m$ is a region $K_i \subseteq \mathbb{R}^k$. If B and the obstacles are semialgebraic sets, then each K_i is also a semialgebraic set. A placement p of B is free if and only if p does not intersect any of K_i's. See [76] for a survey of known results on the collision-detection problem and [11, 37, 38] for a few other applications of point intersection-searching structures.

SEGMENT INTERSECTION SEARCHING

Preprocess a set of objects in $\mathbb{R}^d$ into a data structure so that all the objects of S intersected by a query segment can be reported (or counted) efficiently. See Table 10 for some of the known results on segment intersection searching. For the sake of clarity, we have omitted polylogarithmic terms from the query-search time whenever it is of the form n/m^α.

TABLE 10 Segment intersection searching

d	Objects	$S(n)$	$Q(n)$	Source	Notes
$d = 2$	Simple polygon	n	$(k+1)\log n$	[69]	Reporting
	Segments	m	$n/\sqrt{m}$	[8, 47]	Counting
	Circles	$n^{2+\varepsilon}$	$\log n$	[13]	Counting
	Circular arcs	m	$n/m^{1/3}$	[13]	Counting
$d = 3$	Planes	m	$n/m^{1/3}$	[5]	Counting
	Triangles	m	$n/m^{1/4}$	[6]	Counting
	Spheres	m	$n/m^{1/4}$	[6]	Counting
	Spheres	$n^{3+\varepsilon}$	$(k+1)\log^2 n$	[2]	Reporting

A special case of segment intersection searching, in which the objects are horizontal segments in the plane and query ranges are vertical segments, has been widely studied. In this case a query can be answered in time $O(\log n + k)$ using $O(n \log n)$ space and preprocessing [109]. If we also allow insertions and deletions, the query and update time are $O(\log n \log\log n + k)$ and $O(\log n \log\log n)$ [89], or $O(\log^2 n + k)$ and $O(\log n)$ using only linear space [46]; if we allow only insertions, the query and update time become $O(\log n + k)$ and $O(\log n)$ [71].

A problem related to segment intersection searching is the *stabbing problem*. Given a set S of objects in $\mathbb{R}^d$, determine whether a query k-flat $(0 < k < d)$ intersects all objects of S. Such queries can also be answered efficiently using semialgebraic range-searching data structures. A line-stabbing query amid a set of triangles in $\mathbb{R}^3$ can be answered in $O(\log n)$ time using $O(n^{2+\varepsilon})$ storage [98]. The paper by Goodman et al. [63] is an excellent survey of this topic.

COLORED INTERSECTION SEARCHING

Preprocess a given set S of colored objects in $\mathbb{R}^d$ (i.e., each object in S is assigned a color) so that the we can report (or count) the colors of the objects that intersect the query range. This problem arises in many contexts where one wants to answer intersection-searching queries for nonconstant-size input objects. For example, given a set $P = \{P_1, \ldots, P_m\}$ of m simple polygons, one may wish to report all the simple polygons that intersect a query segment; the goal is to return the index, and not the description, of these polygons. If we color the edges of P_i by the color i, the problem reduces to colored segment intersection searching in a set of segments.

If an intersection-detection query for S with respect to a range γ can be answered in $Q(n)$ time, then the colored intersection-reporting query with γ can be answered in time $O((1 + k\log(n/k))Q(n))$. Therefore logarithmic query-time intersection-searching data structures can easily be modified for colored intersection-reporting, but very little is known about linear-size colored intersection-searching data structures, except in some special cases [12, 65, 66, 67, 72].

Gupta et al. [65] have shown that the colored halfplane-reporting queries in the plane can be answered in $O(\log^2 n + k)$ using $O(n \log n)$ space. Agarwal and van Kreveld [12] presented a linear-size data structure with $O(n^{1/2+k} + k)$ query time for colored segment intersection-reporting queries amid a set of segments in the plane, assuming that the segments of the same color form a connected planar graph, or if they form the boundary of a simple polygon; these data structures can also handle insertions of new segments. Gupta et al. [65, 67] present segment intersection-reporting structures for many other special cases.

OPEN PROBLEMS

1. Faster algorithms for point intersection searching in Tarski cells.
2. An $O(\log n+k)$ query-time and linear-size segment intersection-reporting data structure for a simple polygon.
3. Faster segment intersection-detection structures for (possibly intersecting) Jordan arcs in the plane, and for triangles and spheres in $\mathbb{R}^3$.
4. Linear-size, $O(\sqrt{n}\log^c n + k)$ query-time data structures for colored triangle range reporting.

5 OPTIMIZATION QUERIES

In the optimization queries, we want to return an object that satisfies certain condition with respect to the query range. The most common example of optimization queries is, perhaps, the ray-shooting queries. Other examples include segment-dragging and linear-programming queries.

RAY-SHOOTING QUERIES

Preprocess a set S of objects in $\mathbb{R}^d$ into a data structure so that the first object intersected by a query ray (if there exists one) can be reported efficiently. This problem arises in ray tracing, hidden-surface removal, radiosity, and other graphics problems. Recently, efficient solutions to many other geometric problems have also been developed using ray-shooting data structures.

A general approach to the ray-shooting problem, using segment intersection-detection structures and Megiddo's parametric searching technique [87], was proposed by Agarwal and Matoušek [5]. The basic idea of their approach is as follows. Suppose we have a segment intersection-detection data structure for S, based on partition trees. Let ρ be a query ray. Their algorithm maintains a segment $\vec{ab} \subseteq \rho$ such that the first intersection point of $\vec{ab}$ with S is the same as that of ρ. If a lies on an object of S, it returns a. Otherwise, it picks a point $c \in ab$ and determines, using the segment intersection-detection data structure, whether the interior of the segment ac intersects any object of S. If the answer is yes, it recursively finds the first intersection point of $\vec{ac}$ with S; otherwise, it recursively finds the first intersection point of $\vec{cb}$ with S. Using the parametric searching, the points c at each stage can be chosen in such a way that the algorithm terminates after $O(\log n)$ steps.

In some cases, the query time can be improved by a polylogarithmic factor, using a more direct approach.

Table 11 gives a summary of known ray-shooting results. For the sake of clarity, we have ignored the polylogarithmic factors from the query time whenever it is of the form n/m^{α}. The ray-shooting structures for d-dimensional convex polyhedra assume that the source point of the query ray lies inside the polytope. All the ray-shooting data structures mentioned in Table 11 can be dynamized at a cost of polylogarithmic or n^{ε} factor in the query time. Goodrich and Tamassia [64] have developed a dynamic ray-shooting data structure for connected planar subdivisions, with $O(\log^2 n)$ query and update time.

Like simplex range searching, many practical data structures have been proposed that, despite having bad worst-case performance, work well in practice. One common approach is to construct a subdivision of $\mathbb{R}^d$ into constant-size cells so that the interior of each cell does not intersect any object of S. A ray-shooting query can be answered by traversing the query ray through the subdivision until we find an object that intersects the ray. The worst-case query time is proportional to the maximum number of cells intersected by a segment that does not intersect any object in S. Hershberger and Suri [69] showed that a triangulation with $O(\log n)$ query time can be constructed when S is the boundary of a simple polygon in the plane. See [3, 91, 54, 78] and the references therein for other ray-shooting results using this approach. Agarwal et al. [3] proved worst-case bounds for many cases on the number of cells in the subdivision that a line can intersect.

The *nearest-neighbor query* problem is defined as follows: preprocess a set S of points in $\mathbb{R}^d$ into a data structure so that a point in S closest to a query point ξ

TABLE 11 Ray shooting

d	Objects	$S(n)$	$Q(n)$	Source
$d = 2$	Simple polygon	n	$\log n$	[69]
	s disjoint simple polygons	n $(s^2 + n) \log s$	$\sqrt{s}$ $\log s \log n$	[9, 69]
	s convex polygons	$sn \log s$	$\log s \log n$	[9]
	Segments	m	$n/\sqrt{m}$	[8, 47]
	Circlular arcs	n	$n/m^{1/3}$	[13]
	Disjoint arcs	n	$\sqrt{n}$	[13]
$d = 3$	convex polytope	n	$\log n$	[53]
	c-oriented polytopes	n	$\log n$	[51]
	s convex polytopes	$s^2 n^{2+\varepsilon}$	$\log^2 n$	[10]
	Halfplanes	m	$n/\sqrt{m}$	[5]
	Terrain	m	$n/\sqrt{m}$	[5, 39]
	Triangles	m	$n/m^{1/4}$	[6]
	Spheres	$n^{3+\varepsilon}$	$\log^2 n$	[2]
$d > 3$	Hyperplanes	m	$n/m^{1/d}$	[5]
	Hyperplanes	$\frac{n^d}{\log^{d-\varepsilon} n}$	$\log n$	[5]
	Convex polytope	m	$n/m^{1/\lfloor d/2 \rfloor}$	[5, 80]
	Convex polytope	$\frac{n^{\lfloor d/2 \rfloor}}{\log^{\lfloor d/2 \rfloor - \varepsilon} n}$	$\log n$	[80]

can be reported quickly. This query can be formulated as an instance of the ray-shooting problem in a convex polyhedron in $\mathbb{R}^{d+1}$, as follows. We map each point $p = (p_1, \dots, p_d)$ in S to a hyperplane in $\mathbb{R}^{d+1}$, which is the graph of the function

$$f_p(x_1, \dots, x_n) = 2p_1 x_1 + \cdots + 2p_d x_d - (p_1^2 + \cdots + p_d^2).$$

Then p is a closest neighbor of a point $\xi = (\xi_1, \dots, \xi_d)$ if and only if

$$f_p(\xi_1, \dots, \xi_d) = \max_{q \in S} f_q(\xi_1, \dots, \xi_d).$$

That is, if and only if f_p is the first hyperplane intersected by the vertical ray $\rho(\xi)$ emanating from the point $(\xi_1, \dots, \xi_d, 0)$ in the $(-x_{d+1})$-direction. If we define $P = \bigcap\{x_{d+1} \geq f_p(x_1, \dots, x_d) \mid p \in S\}$, then p is the nearest neighbor of ξ if and only if the intersection point of $\rho(\xi)$ and ∂P lies on the graph of f_p. Thus a nearest-neighbor query can be answered in time roughly $n/m^{1/\lfloor d/2 \rfloor}$ using $O(m)$ space. This approach can be extended to answer farthest-neighbor and k-nearest-neighbor queries also.

LINEAR-PROGRAMMING QUERIES

Let S be a set of n halfspaces in $\mathbb{R}^d$. We wish to preprocess S into a data structure so that for a direction vector $\vec{v}$, we can determine the first point of $\bigcap_{h \in S} h$ in the direction $\vec{v}$. For $d \leq 3$, such a query can be answered in $O(\log n)$ time using $O(n)$ storage, by constructing the normal diagram of the convex polytope $\bigcap_{h \in S} h$ and preprocessing it for point-location queries. For higher dimensions, Matoušek [84] showed that, using multidimensional parametric searching and the data structure for answering halfspace emptiness queries, a linear-programming query can be answered in $O((n/m^{1/\lfloor d/2 \rfloor}) \log^c n)$ with $O(m)$ storage. Recently Chan [24] has described a randomized procedure whose expected query time is slightly faster.

SEGMENT DRAGGING QUERIES

Preprocess a set S of objects in the plane so that for a query segment e and a ray ρ, the first position at which e intersects any object of S as it is translated (dragged) along ρ can be determined quickly. This query can be answered in $O((n/\sqrt{m}) \log^c n)$ time, with $O(m)$ storage, using segment intersection-searching structures and the parametric-search technique. Chazelle [26] gave a linear-size, $O(\log n)$ query-time data structure for the special case in which S is a set of points, e is a horizontal segment, and ρ is the vertical direction. Instead of dragging a segment along a ray, one can ask the same question for dragging along a more complex trajectory (along a curve and allowing both translation and rotation). These problems arise quite often in motion planning and manufacturing. See [90, 101] for a few such examples.

OPEN PROBLEMS

1. Ray shooting amid a set of intersecting arcs in the plane.
2. Ray shooting amid triangles in $\mathbb{R}^3$ in $n/m^{1/3}$.
3. Can a ray-shooting query in a nonconvex polytope in $\mathbb{R}^3$ be answered any faster than a ray-shooting query amid triangles?
4. No nontrivial lower bounds are known for the ray-shooting problem.

6 SOURCES AND RELATED MATERIAL

BOOKS AND MONOGRAPHS

- Mehlhorn [88]: A text book on computational geometry. The first part of the book covers multidimensional searching.

- Mulmuley [94]: A text book on randomized techniques in computational geometry. Chapters 6 and 8 cover range-searching, intersection-searching, and ray-shooting data structures.
- Preparata and Shamos [99]: A text book on basic topics in computational geometry. Chapters 2 includes earlier results on orthogonal range searching.
- Foley et al. [58]: A text book on graphics. Discusses practical data structures for ray tracing and intersection searching.
- de Berg [50]: A monograph on ray shooting and related problems.
- Schwarzkopf [103]: This PhD thesis includes many results on randomized dynamic data structures.

SURVEY PAPERS

- Bentley and Friedman [21]: A survey of earlier results on orthogonal range searching.
- Chazelle [32]: A general survey of recent developments in computational geometry. It contains most of the references on simplex and semialgebraic range searching.
- Chiang and Tamassia [48]: A survey of dynamic data structures.
- Goodman et al. [63]: A survey of stabbing problems and related topics.
- Matoušek [86]: A comprehensive survey of simplex range searching and related topics.

References

[1] P. K. Agarwal, Ray shooting and other applications of spanning trees with low stabbing number, *SIAM J. Comput.*, 21 (1992), 540–570.

[2] P. K. Agarwal, B. Aronov, and M. Sharir, Computing envelopes in four dimensions with applications, *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, 1994, pp. 348–358.

[3] P. K. Agarwal, B. Aronov, and S. Suri, Line stabbing bounds in three dimensions, *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, 1995, pp. 267–276.

[4] P. K. Agarwal, A. Efrat, and M. Sharir, Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications, *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, 1995, pp. 39–50.

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

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

[7] P. K. Agarwal and J. Matoušek, Dynamic half-space range reporting and its applications, *Algorithmica*, 14 (1995), 325–345.

[8] P. K. Agarwal and M. Sharir, Applications of a new partition scheme, *Discrete Comput. Geom.*, 9 (1993), 11–38.

[9] P. K. Agarwal and M. Sharir, Ray shooting amidst convex polygons in 2D, *J. Algorithms*, XX (1996), to appear.

[10] P. K. Agarwal and M. Sharir, Ray shooting amidst convex polytopes in three dimensions, *SIAM J. Comput.*, 25 (1996), 100–116.

[11] P. K. Agarwal, M. Sharir, and S. Toledo, Applications of parametric searching in geometric optimization, *J. Algorithms*, 17 (1994), 292–318.

[12] P. K. Agarwal and M. van Kreveld, Connected component and simple polygon intersection searching, *Proc. 3rd Workshop Algorithms Data Struct.*, *Lecture Notes in Computer Science*, Vol. 709, Springer-Verlag, 1993, pp. 36–47.

[13] P. K. Agarwal, M. van Kreveld, and M. Overmars, Intersection queries in curved objects, *J. Algorithms*, 15 (1993), 229–266.

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

[15] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[16] A. Andersson and K. Swanson, On the difficulty of range searching, *Proc. 4th Workshop Algorithms Data Struct.*, *Lecture Notes in Computer Science*, Vol. 955, Springer-Verlag, 1995, pp. 473–481.

[17] M. Anthony and N. Biggs, *Computational Learning Theory*, Cambridge University Press, Cambridge, 1992.

[18] S. Arya and D. Mount, Approximate range searching, *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, 1995, pp. 172–181.

[19] R. Bar-Yehuda and S. Fogel, Variations on ray shooting, *Algorithmica*, 11 (1994), 133–145.

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

[21] J. L. Bentley and J. H. Friedman, Data structures for range searching, *ACM Comput. Surv.*, 11 (1979), 397–409.

[22] J. L. Bentley and J. B. Saxe, Decomposable searching problems I: static-to-dynamic transformation, *J. Algorithms*, 1 (1980), 301–358.

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

[24] T. Chan, Fixed-dimensional linear programming queries made easy, *Proc. 12th ACM Symp. Comput. Geom.*, 1996, p. to appear.

[25] B. Chazelle, Filtering search: a new approach to query-answering, *SIAM J. Comput.*, 15 (1986), 703–724.

[26] B. Chazelle, An algorithm for segment-dragging and its implementation, *Algorithmica*, 3 (1988), 205–221.

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

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

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

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

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

[32] B. Chazelle, Computational geometry: A retrospective, *Proc. 26th Annu. ACM Sympos. Theory Comput.*, 1994, pp. 75–94.

[33] B. Chazelle, A spectral approach to lower bounds, *Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci.*, 1994, pp. 674–682.

[34] B. Chazelle, Lower bounds for off-line range searching, *Proc. 27th Annu. ACM Sympos. Theory Comput.*, 1995, pp. 733–740.

[35] B. Chazelle and H. Edelsbrunner, Optimal solutions for a class of point retrieval problems, *J. Symbolic Comput.*, 1 (1985), 47–56.

[36] B. Chazelle and H. Edelsbrunner, Linear space data structures for two types of range search, *Discrete Comput. Geom.*, 2 (1987), 113–126.

[37] 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 (1991), 77–105.

[38] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir, Diameter, width, closest line pair and parametric searching, *Discrete Comput. Geom.*, 10 (1993), 183–196.

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

[40] B. Chazelle and L. J. Guibas, Fractional cascading: I. A data structuring technique, *Algorithmica*, 1 (1986), 133–162.

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

[42] B. Chazelle and B. Rosenberg, Computing partial sums in multidimensional arrays, *Proc. 5th Annu. ACM Sympos. Comput. Geom.*, 1989, pp. 131–139.

[43] B. Chazelle and B. Rosenberg, Lower bounds on the complexity of simplex range reporting on a pointer machine, in: *Proc. 19th International Colloquium on Automata, Languages, and Programming*, *Lecture Notes in Computer Science*, Vol. 623, Springer-Verlag, 1992, pp. 439–449. Also to appear in *Comput. Geom. Theory Appl.*

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

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

[46] S. W. Cheng and R. Janardan, Efficient dynamic algorithms for some geometric intersection problems, *Inform. Process. Lett.*, 36 (1990), 251–258.

[47] S. W. Cheng and R. Janardan, Algorithms for ray-shooting and intersection searching, *J. Algorithms*, 13 (1992), 670–692.

[48] Y.-J. Chiang and R. Tamassia, Dynamic algorithms in computational geometry, *Proc. IEEE*, 80 (1992), 1412–1434.

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

[50] M. de Berg, *Ray Shooting, Depth Orders and Hidden Surface Removal*, Springer-Verlag, Berlin, 1993.

[51] M. de Berg and M. Overmars, Hidden surface removal for c-oriented polyhedra, *Comput. Geom. Theory Appl.*, 1 (1992), 247–268.

[52] D. Dobkin, J. Hershberger, D. Kirkpatrick, and S. Suri, Implicitly searching convolutions and computing depth of collision, *Proc. 1st Annu. SIGAL Internat. Sympos. Algorithms*, *Lecture Notes in Computer Science*, Vol. 450, Springer-Verlag, 1990, pp. 165–180.

[53] D. P. Dobkin and D. G. Kirkpatrick, A linear algorithm for determining the separation of convex polyhedra, *J. Algorithms*, 6 (1985), 381–392.

[54] D. P. Dobkin and D. G. Kirkpatrick, Determining the separation of preprocessed polyhedra – a unified approach, *Proc. 17th Internat. Colloq. Automata Lang. Program.*, *Lecture Notes in Computer Science*, Vol. 443, Springer-Verlag, 1990, pp. 400–413.

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

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

[57] P. Erdős and G. Szekeres, A combinatorial problem in geometry, *Compositio Math.*, 2 (1935), 463–470.

[58] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*, Addison-Wesley, Reading, MA, 1990.

[59] M. L. Fredman, Inherent complexity of range query problems, *Proc. 17th Allerton Conf. Commun. Control Comput.*, 1979, pp. 231–240.

[60] M. L. Fredman, The inherent complexity of dynamic data structures which accommodate range queries, *Proc. 21st Annu. IEEE Sympos. Found. Comput. Sci.*, 1980, pp. 191–199.

[61] M. L. Fredman, A lower bound on the complexity of orthogonal range queries, *J. ACM*, 28 (1981), 696–705.

[62] J. H. Friedman, J. L. Bentley, and R. A. Finkel, An algorithm for finding best matches in logarithmic expected time, *ACM Trans. Math. Softw.*, 3 (1977), 209–226.

[63] J. E. Goodman, R. Pollack, and R. Wenger, Geometric transversal theory, in: *New Trends in Discrete and Computational Geometry* (J. Pach, ed.), Springer-Verlag, Heidelberg–New York–Berlin, 1993, pp. 163–198.

[64] M. T. Goodrich and R. Tamassia, Dynamic ray shooting and shortest paths via balanced geodesic triangulations, *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, 1993, pp. 318–327.

[65] P. Gupta, R. Janardan, and M. Smid, Efficient algorithms for generalized intersection searching on non-iso-oriented objects, *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, 1994, pp. 369–378.

[66] P. Gupta, R. Janardan, and M. Smid, On intersection searching problems involving curved objects, *Proc. 4th Scand. Workshop Algorithm Theory*, *Lecture Notes in Computer Science*, Vol. 824, Springer-Verlag, 1994, pp. 183–194.

[67] P. Gupta, R. Janardan, and M. Smid, Further results on generalized intersection searching problems: counting, reporting and dynamization, *J. Algorithms*, 19 (1995), 282–317.

[68] D. Haussler and E. Welzl, Epsilon-nets and simplex range queries, *Discrete Comput. Geom.*, 2 (1987), 127–151.

[69] J. Hershberger and S. Suri, A pedestrian approach to ray shooting: Shoot a ray, take a walk, *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, 1993, pp. 54–63.

[70] P. Houthuys, Box sort, a multidimensional binary sorting method for rectangular boxes, used for quick range searching, *Visual Comput.*, 3 (1987), 236–249.

[71] H. Imai and T. Asano, Dynamic orthogonal segment intersection search, *J. Algorithms*, 8 (1987), 1–18.

[72] R. Janardan and M. Lopez, Generalized intersection searching problems, *Internat. J. Comput. Geom. Appl.*, 3 (1993), 39–69.

[73] M. J. Katz, 3-D vertical ray shooting and 2-D point enclosure, range searching, and arc shooting amidst convex fat objects, Unpublished manuscript, 1995.

[74] R. Klein, O. Nurmi, T. Ottmann, and D. Wood, A dynamic fixed windowing problem, *Algorithmica*, 4 (1989), 535–550.

[75] J. Komlós, J. Pach, and G. Woeginger, Almost tight bounds for ϵ-nets, *Discrete Comput. Geom.*, 7 (1992), 163–173.

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

[77] G. S. Lueker, A data structure for orthogonal range queries, *Proc. 19th Annu. IEEE Sympos. Found. Comput. Sci.*, 1978, pp. 28–34.

[78] J. MacDanold and K. Booth, Heuristics for ray tracing using space subdivision, *The Visual Computer*, 6 (1990), 153–166.

[79] J. Matoušek, Reporting points in halfspaces, *Comput. Geom. Theory Appl.*, 2 (1992), 169–186.

[80] J. Matoušek and O. Schwarzkopf, On ray shooting in convex polytopes, *Discrete Comput. Geom.*, 10 (1993), 215–232.

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

[82] J. Matoušek, E. Welzl, and L. Wernisch, Discrepancy and ε-approximations for bounded VC-dimension, *Combinatorica*, 13 (1993), 455–466.

[83] J. Matoušek, Efficient partition trees, *Discrete Comput. Geom.*, 8 (1992), 315–334.

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

[85] J. Matoušek, Range searching with efficient hierarchical cuttings, *Discrete Comput. Geom.*, 10 (1993), 157–182.

[86] J. Matoušek, Geometric range searching, *ACM Comput. Surv.*, 26 (1994), 421–461.

[87] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, *J. ACM*, 30 (1983), 852–865.

[88] K. Mehlhorn, *Multi-dimensional Searching and Computational Geometry*, Springer-Verlag, Heidelberg, West Germany, 1984.

[89] K. Mehlhorn and S. Näher, Dynamic fractional cascading, *Algorithmica*, 5 (1990), 215–241.

[90] J. S. B. Mitchell, Shortest paths among obstacles in the plane, *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, 1993, pp. 308–317.

[91] J. S. B. Mitchell, D. M. Mount, and S. Suri, Query-sensitive ray shooting, *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, 1994, pp. 359–368.

[92] K. Mulmuley, Randomized multidimensional search trees: dynamic sampling, *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, 1991, pp. 121–131.

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

[94] K. Mulmuley, *Computational Geometry: An Introduction Through Randomized Algorithms*, Prentice Hall, Englewood Cliffs, NJ, 1993.

[95] M. H. Overmars, *The design of dynamic data structures*, Springer-Verlag, 1983.

[96] M. H. Overmars, Efficient data structures for range searching on a grid, *J. Algorithms*, 9 (1988), 254–275.

[97] M. H. Overmars and A. F. van der Stappen, Range searching and point location among fat objects, *Algorithms – ESA'94* (J. van Leeuwen, ed.), LNCS 855, September 1994, pp. 240–253.

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

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

[100] S. Ramaswamy and S. Subramanian, Path caching: A technique for optimal external searching, *Proc. 13th ACM Symp. on Principles of Database Systems*, 1994, pp. 25–35.

[101] E. Schömer and C. Thiel, Efficient collision detection for moving polyhedra, *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, 1995, pp. 51–60.

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

[103] O. Schwarzkopf, *Dynamic Maintenance of Convex Polytopes and Related Structures*, Ph.D. Thesis, Fachbereich Mathematik, Freie Universität Berlin, Berlin, Germany, June 1992.

[104] M. Smid, Maintaining the minimal distance of a point set in less than linear time, *Algorithms Rev.*, 2 (1991), 33–44.

[105] D. M. H. Sommerville, *Analytical Geometry in Three Dimensions*, Cambridge University Press, Cambridge, 1951.

[106] S. Subramanian and S. Ramaswamy, The p-range tree: A new data structure for range searching in secondary memory, *Proc. 6th ACM-SIAM Symp. on Discrete Algorithms*, 1995, pp. 378–387.

[107] R. E. Tarjan, *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.

[108] P. M. Vaidya, Space-time tradeoffs for orthogonal range queries, *SIAM J. Comput.*, 18 (1989), 748–758.

[109] V. K. Vaishnavi and D. Wood, Rectilinear line segment intersection, layered segment trees and dynamization, *J. Algorithms*, 3 (1982), 160–176.

[110] J. Vitter and D. Vengroff, Efficient 3-d range searching in external memory, *Proc. 28th ACM Symp. Theory of Computing*, 1996, p. to appear.

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

[112] E. Welzl, On spanning trees with low crossing numbers, in: *??*, *Lecture Notes in Computer Science*, Vol. 594, Springer-Verlag, 1992, pp. 233–249.

[113] D. E. Willard, Polygon retrieval, *SIAM J. Comput.*, 11 (1982), 149–165.

[114] D. E. Willard, Applications of the fusion tree method to computational geometry and searching, *Proc. 3rd ACM-SIAM Sympos. Discrete Algorithms*, 1992, pp. 286–296.

[115] D. E. Willard, Applications of range query theory to relational data base join and selection operations, *J. Comput. Sys. Sci.*, 52 (1996), 157–169.

[116] A. C. Yao, Space-time trade-off for answering range queries, *Proc. 14th Annu. ACM Sympos. Theory Comput.*, 1982, pp. 128–136.

[117] A. C. Yao, On the complexity of maintaining partial sums, *SIAM J. Comput.*, 14 (1985), 277–288.

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