Research report 118

PARTITIONING SPACE FOR RANGE QUERIES

by
F. Frances Yao[1]
David P. Dobkin[2]
Herbert Edelsbrunner[3]
Michael S. Paterson[4]

(RR 118)

Abstract

It is shown that, given a set S of n points in R^3, one can always find three planes that form an *eight-partition* of S, that is, a partition where at most $n/8$ points of S lie in each of the eight open regions. This theorem is used to define a data structure, called an *octant tree*, for representing any point set in R^3. An octant tree for n points occupies $O(n)$ space and can be constructed in polynomial time. With this data structure and its refinements, efficient solutions to various range query problems in 2 and 3 dimensions can be obtained, including (1) half-space queries: find all points of S that lie to one side of any given plane; (2) polyhedron queries: find all points that lie inside (outside) any given polyhedron; and (3) circular queries in R^2: for a planar set S, find all points that lie inside (outside) any given circle. The retrieval time for all these queries is $T(n)=O(n^{\alpha}+m)$ where $\alpha = 0.8988$ (or 0.8471 in case (3)) and m is the size of the output. This performance is the best currently known for linear-space data structures which can be deterministically constructed in polynomial time.

Department of Computer Science
University of Warwick
Coventry, CV4 7AL, England

April 1988

[1]Xerox Palo Alto Research Center, Palo Alto, California.

[2]Department of Computer Science, Princeton University, Princeton, New Jersey.

[3]Department of Computer Science, University of Illinois, Urbana, Illinois.

[4]Department of Computer Science, University of Warwick, Coventry, UK.

[This author was supported by Xerox during visits to the Palo Alto Research Center, and by a Senior Fellowship of the Science and Engineering Research Council of the UK.]

Partitioning Space for Range Queries

F. Frances Yao[1]
David P. Dobkin[2]
Herbert Edelsbrunner[3]
Michael S. Paterson[4]

Abstract

It is shown that, given a set S of n points in R^3, one can always find three planes that form an *eight-partition* of S, that is, a partition where at most $n/8$ points of S lie in each of the eight open regions. This theorem is used to define a data structure, called an *octant tree*, for representing any point set in R^3. An octant tree for n points occupies $O(n)$ space and can be constructed in polynomial time. With this data structure and its refinements, efficient solutions to various range query problems in 2 and 3 dimensions can be obtained, including (1) half-space queries: find all points of S that lie to one side of any given plane; (2) polyhedron queries: find all points that lie inside (outside) any given polyhedron; and (3) circular queries in R^2: for a planar set S, find all points that lie inside (outside) any given circle. The retrieval time for all these queries is $T(n) = O(n^\alpha + m)$ where $\alpha = 0.8988$ (or 0.8471 in case (3)) and m is the size of the output. This performance is the best currently known for linear-space data structures which can be deterministically constructed in polynomial time.

[1] Xerox Palo Alto Research Center, Palo Alto, California.

[2] Department of Computer Science, Princeton University, Princeton, New Jersey.

[3] Department of Computer Science, University of Illinois, Urbana, Illinois.

[4] Department of Computer Science, University of Warwick, Coventry, UK. This author was supported by Xerox during visits to the Palo Alto Research Center, and by a Senior Fellowship of the Science and Engineering Research Council of the UK.

1. Introduction

Consider a database that contains a collection of records with multi-dimensional keys. Given a *range query*, which is specified by certain constraints on the value of the multi-dimensional key, the database is expected to return the set of all records (or some function of the set of all records) whose keys satisfy those constraints. Efficient solutions to range queries are important both in themselves and also as subroutines for solving other multi-dimensional search problems. In this paper we will consider solutions to range queries that use only linear space for data structure storage.

There is an extensive literature on efficient algorithms for handling *orthogonal queries*, that is, queries with constraints of the form $a_1 \leq k_1 \leq b_1, \ldots, a_d \leq k_d \leq b_d$, where the key is $(k_1, \ldots, k_d)$. Relatively little is known about solving queries of more general types, such as *half-space queries* where the constraints are linear inequalities $a_1 k_1 + \cdots + a_d k_d \leq c$. Willard [W] was the first to consider half-space queries for $d = 2$, and gave a solution with linear space and sublinear query time $O(n^\alpha)$ where $\alpha \approx 0.774$. Edelsbrunner and Welzl [EW] improved α to $\log_2 \frac{\sqrt{5}+1}{2} \approx 0.695$. Both of these results are based on the fact that a set of n points in R^2 can be partitioned by two lines so that each open quadrant contains at most $n/4$ points.

For $d = 3$, the first nontrivial time bound was $O(n^\alpha)$ for $\alpha \approx 0.98$ by Yao [Y]. The data structure is based on a partition of any point set by three planes into eight regions with the property that no seven regions together contain more than 23/24 of the points. Such a partition was obtained by making use of the concept of a *centerpoint* of a set (see [YB]).

In this paper, we prove a stronger result on partitions in R^3 by using the *Borsuk-Ulam Theorem* of topology. It is shown that, given a set S of n points in R^3, one can always find three planes that form an *eight-partition* of S, that is, a partition where at most $n/8$ points of S lie in each of the eight open regions. This theorem is used to define a data structure, called an *octant tree*, for representing any point set in R^3. Efficient solutions to various range query problems in R^2 and R^3 can be obtained by using this data structure and its refinements. For example, one can solve in time $O(n^{0.8988} + m)$, where m is the size of the output,

(1) half-space queries: find all points that lie to one side of a query plane;
(2) polyhedron queries: find all points that lie inside (outside) a query polyhedron; and
(3) circular queries in R^2: given a planar set, find all points that lie inside (outside) a query circle.

An octant tree for n points occupies $O(n)$ space and can be constructed in $O(n^6 \log n)$ preprocessing time.

The paper contains five sections. In Section 2 we define the concepts necessary for discussing space partitions in both the continuous case and the discrete case. Section 3

contains the proof of the main theorem of the paper. In Section 4 we present and analyze several data structures based on the main theorem for representing point sets. Finally we comment on open problems and related results in Section 5.

2. Preliminaries

We use S^{d-1} to denote the unit sphere $\{(x_1, x_2, \ldots, x_d) \mid x_1^2 + x_2^2 + \ldots + x_d^2 = 1\}$ in R^d. An *oriented hyperplane* (or *hyperplane* for short) h in R^d with *normal vector* $\mathbf{v} = (v_1, v_2, \ldots, v_d) \in S^{d-1}$ is defined by an equation $\sum v_i x_i = t$. If v has length 1, then the real number t is the *distance* of h from the origin; it is the unique scalar for which the point $t \cdot \mathbf{v}$ lies on h. The hyperplane h separates R^d into a *positive half-space* h^+ defined by $\sum v_i x_i \geq t$ and a *negative half-space* h^- defined by $\sum v_i x_i \leq t$. When we consider continuous functions defined on the collection of hyperplanes in R^d, we assume that the latter is endowed with the topology of $S^{d-1} \times R$ through the representation of h by $(\mathbf{v}, t)$. Corresponding to $h = (\mathbf{v}, t)$ we let $-h$ denote the hyperplane $(-\mathbf{v}, -t)$; thus, $-h$ is a hyperplane defined by the same equation as h but with opposite orientation.

We shall limit our discussions to R^d with $d \leq 3$ in this paper. A hyperplane in R^3 will simply be called a *plane*. For a set S of n points in R^3, we are interested in finding three planes h_1, h_2, h_3 so that at most $n/8$ points lie in each of the eight open regions defined by the three planes. Such a triple (h_1, h_2, h_3) is termed an *eight-partition* of S. We shall prove the existence of an eight-partition for any finite point set by first transforming the problem to a continuous framework. Thus, let A be a positive density function defined on some bounded, connected region in R^3, and let an *eight-partition* of A be a triple of planes (h_1, h_2, h_3) that partitions A into eight parts of equal mass.

Lemma 2.1. If every positive density function over a bounded connected region in R^3 has an eight-partition, then every finite point set in R^3 has an eight-partition.

Proof. We replace a set S of n points with a density function A by placing at each point $p \in S$ a small ball $b(p)$ of uniform mass $(1 - \delta)/n$ with radius ϵ and center p. We choose ϵ to be small enough so that a set of balls can intersect a common plane only if their centers are coplanar. Let C be a large sphere of volume $\|C\|$ which contains all the balls, and place additional density $\delta/\|C\|$ uniformly inside C. Thus the total mass over C is 1, and the mass outside of the union of the balls is less than δ, which is chosen to be less than $1/2n$. Suppose we find an eight-partition for A with planes (h_1, h_2, h_3). Then we can find $(\bar{h}_1, \bar{h}_2, \bar{h}_3)$ with the property that (1) $p \in \bar{h}_\ell$ if $b(p)$ intersects h_ℓ, and (2) $p \in \bar{h}_\ell^+ \cup \bar{h}_\ell$ (or $\bar{h}_\ell^- \cup \bar{h}_\ell$) if $b(p)$ lies in h_ℓ^+ (or h_ℓ^-). This is possible by the choice of ϵ. The partition $(\bar{h}_1, \bar{h}_2, \bar{h}_3)$ has the property that there are at most $n/8$ points in each of the eight open regions of the partition. $\square$

Because of Lemma 2.1, it suffices to prove the existence of eight-partitions in the continuous setting. Let A be a density function as described in the lemma. Any triple of planes (h_1, h_2, h_3) partitions A into the eight proportions denoted by $a_{xyz}(h_1, h_2, h_3)$ for $x, y, z \in \{0, 1\}$, where the l-th subscript is 0 or 1 depending on whether the region lies in h_l^+ or h_l^-. We shall abbreviate $a_{xyz}(h_1, h_2, h_3)$ as a_{xyz} whenever possible. Thus, a_{010} denotes the mass contained in $h_1^+ \cap h_2^- \cap h_3^+$, and a_{111} denotes that contained in $h_1^- \cap h_2^- \cap h_3^-$. (See Figure 1, where $\mathbf{v}_i$ is the normal vector to h_i.) The a_{xyz}'s are continuous functions of h_1, h_2 and h_3. We use $*$ as a subscript to indicate a summation of the a_{xyz}'s where that subscript can assume both 0 and 1. For example, we write a_{xy*} for $\sum_z a_{xyz} = a_{xy0} + a_{xy0}$, and a_{*y*} for $\sum_{x,z} a_{xyz} = a_{0y0} + a_{0y1} + a_{1y0} + a_{1y1}$, etc.

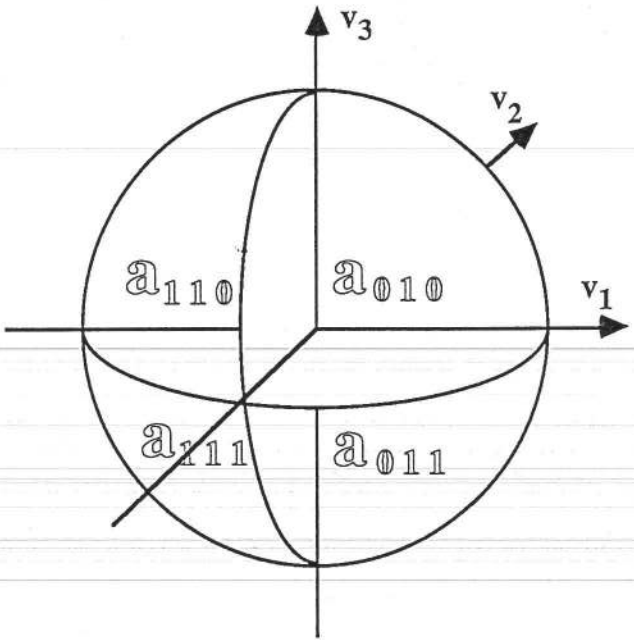

Figure 1

Definitions. Let A be a positive density function on R^3, and consider the a_{xyz}'s defined by three planes (h_1, h_2, h_3). We say that

1) h_1 is a *bisector* of A if $a_{x**} = 1/2$ for $x \in \{0, 1\}$;
2) the pair (h_1, h_2) forms a *four-partition* of A if $a_{xy*} = 1/4$ for all $x, y \in \{0, 1\}$;
3) the triple (h_1, h_2, h_3) forms an *eight-partition* of A if $a_{xyz} = 1/8$ for all $x, y, z \in \{0, 1\}$.

In order to achieve $a_{xyz} = 1/8$ for all x, y, z, it is convenient to form eight linear combinations of the a_{xyz}'s as follows. Let $f_{ijk} = \sum_{xyz} \epsilon_{ijk}^{xyz} a_{xyz}$ where $\epsilon_{ijk}^{xyz} = (-1)^b$ with $b = (i, j, k) \cdot (x, y, z) = ix + jy + kz$. For example, $f_{000} = a_{***} \equiv 1$, $f_{100} = a_{0**} - a_{1**}$, and

$$f_{110} = a_{00*} - a_{01*} - a_{10*} + a_{11*}.$$

The f_{ijk}'s, like the a_{xyz}'s, are continuous functions of h_1, h_2 and h_3. Note that f_{110} is symmetric in the first and the second arguments: $f_{110}(h_1, h_2, h_3) = f_{110}(h_2, h_1, h_3)$. Also, when we flip the sign of an argument h_l, the function f_{xyz} either changes sign or not depending on whether the corresponding subscript in f_{xyz} is 1 or 0. For example, for $l = 1$, $f_{110}(-h_1, h_2, h_3) = -f_{110}(h_1, h_2, h_3)$; while for $l = 3$, $f_{110}(h_1, h_2, -h_3) = f_{110}(h_1, h_2, h_3)$. We state these symmetry properties in the next lemma.

Lemma 2.2.
1) $f_{ijk}(h_1, h_2, h_3) = f_{jik}(h_2, h_1, h_3)$.
2) $f_{ijk}(-h_1, h_2, h_3) = (-1)^i f_{ijk}(h_1, h_2, h_3)$.

Proof. Immediate from the definition of f_{ijk}. □

In seeking an eight-partition for A, we shall make use of the following characterization in terms of the f_{ijk}'s.

Lemma 2.3.
1) h_1 is a bisector of A iff $f_{100} = 0$.
2) The pair (h_1, h_2) forms a four-partition of A iff $f_{100} = f_{010} = f_{110} = 0$.
3) The triple (h_1, h_2, h_3) forms an eight-partition of A iff $f_{ijk} = 0$ for all $(i, j, k) \neq (0, 0, 0)$.

Proof. 1) $a_{0**} = a_{1**} = 1/2$ iff $a_{0**} - a_{1**} = f_{100} = 0$ since $a_{0**} + a_{1**} = f_{000} \equiv 1$.
2) Note that

$$\begin{pmatrix} f_{100} \\ f_{010} \\ f_{110} \\ f_{000} \end{pmatrix} = \begin{pmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} a_{00*} \\ a_{01*} \\ a_{10*} \\ a_{11*} \end{pmatrix}.$$

If $a_{00*} = a_{01*} = a_{10*} = a_{11*} = 1/4$, then $f_{100} = f_{010} = f_{110} = 0$ and $f_{000} \equiv 1$. Since the 4×4 matrix is nonsingular, indeed orthogonal, the converse is also true.
3) We show that the matrix of the coefficients $\{\epsilon_{ijk}^{xyz}\}$ is orthogonal, and so nonsingular. Since

$$\begin{aligned} \sum_{xyz} \epsilon_{ijk}^{xyz} \epsilon_{i'j'k'}^{xyz} &= \sum_{xyz} (-1)^{(x,y,z)\cdot(i+i', j+j', k+k')} \\ &= 0 \text{ unless } i = i', j = j' \text{ and } k = k', \end{aligned}$$

the inner product of any two distinct rows of the matrix is zero. □

3. Eight-Partition in R^3

It is well known that a four-partition can always be found for a positive density function over a bounded connected region in R^2. (See, e.g., [Me]) We first show this as a lemma and then prove a slightly stronger version in R^3 for later use.

Lemma 3.1. (*Four-Partition*) Let A_0 and A_1 be two positive density functions on the plane whose domains are bounded, connected and separable by a line L. There is a unique (unoriented) line L' that bisects A_0 and A_1 simultaneously.

Proof. For any point p on L, let ℓ_p and r_p be the (unique) lines that go through p and bisect A_0 and A_1 respectively. We can assume that L is vertical and that A_1 is to the right of L. As p moves up L from bottom to top, the slope of r_p decreases continuously and monotonically from ∞ to $-\infty$, while that of ℓ_p increases continuously and monotonically from $-\infty$ to ∞. Hence there is a unique p for which ℓ_p and r_p coincide, giving the desired L'. □

We next consider four-partitions in R^3.

Lemma 3.2. Let A_0 and A_1 be two positive density functions in R^3 whose domains are bounded, connected and separable by a plane h. Let $S_h \cong S^1$ denote the set of unit vectors in R^3 that are parallel to h. Then,
1) for any $u \in S_h$ there is a unique plane $p(u)$ parallel to u which bisects A_0 and A_1 simultaneously and has an orientation induced by u;
2) the mapping $f : S_h \mapsto S^2$ which maps $u \in S_h$ to the normal vector of $p(u)$ gives a continuous antipodal mapping of S_h into S^2, i.e., $f(-u) = -f(u)$ for all $u \in S_h$.

Proof. For any $u \in S_h$, a plane $p(u)$ parallel to u bisects A_0 and A_1 simultaneously if and only if the projection of $p(u)$ along the direction u (onto a plane normal to u) gives a line that bisects the projections of A_0 and A_1 simultaneously. The intersection of h and $p(u)$ is a line parallel to u. The orientation of u gives it a natural orientation which can be used to define the left and right half-planes of h with respect to this line. The orientation of $p(u)$ is chosen so that these lie respectively in the positive and negative half-spaces determined by $p(u)$. By Lemma 3.1, $p(u)$ is unique.

It is easy to see that the function f defined in (2) is continuous and antipodal. □

We shall make use of the $d = 2$ case of the following topological theorem in proving that every density function A in R^3 can be eight-partitioned.

Theorem. (*Borsuk-Ulam*) Let $f : S^d \mapsto R^d$ be a continuous, antipodal map, i.e., $f(-p) = -f(p)$ for $p \in S^d$. Then there is a point $p \in S^d$ such that $f(p) = 0$.

A proof of the Borsuk-Ulam theorem can be found in textbooks on algebraic topology such as Munkres [Mu]. The theorem does not extend in general to mappings defined on direct products of spheres. However, we can establish the following extension for mappings defined

on the torus $S^1 \times S^1$ which satisfy certain additional symmetry properties. This lemma is sufficient, as we shall see, for establishing the existence of eight-partitions in R^3.

Lemma 3.3. Let $f : S^1 \times S^1 \mapsto R^2$ be a continuous map such that
1) f is symmetric, i.e., $f(u,v) = f(v,u)$,
2) f is antipodal in each argument, i.e. $f(u,-v) = f(-u,v) = -f(u,v)$, and
3) f is constant on the the diagonal $\{(u,u) \mid u \in S^1\}$.
Then there is a point $p \in S^1 \times S^1$ such that $f(p) = 0$.

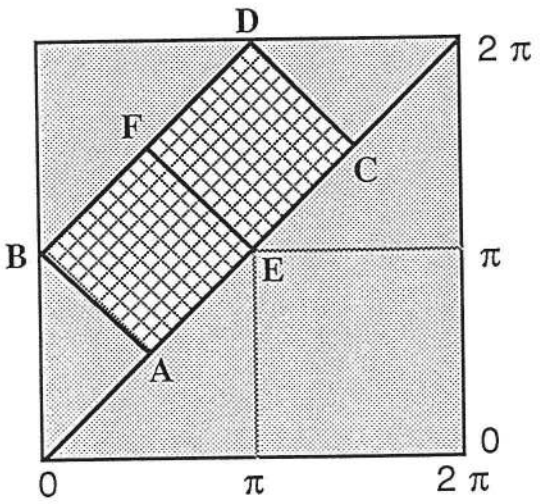

Figure 2

Proof. We can represent $S^1 \times S^1$ by the square $\mathcal{Q} = [0, 2\pi] \times [0, 2\pi]$ with opposite sides identified. Consider the rectangle $\mathcal{P} = \{(u,v) \mid \pi \leq u+v \leq 3\pi,\ u \leq v \leq u+\pi\}$ contained in $\mathcal{Q}$ with vertices $A = (\pi/2, \pi/2)$, $B = (0,\pi)$, $C = (3\pi/2, 3\pi/2)$ and $D = (\pi, 2\pi)$ (Figure 2). Note that f is constant on side AC by property (3), constant on side BD since $f(u, u+\pi) = -f(u,u)$ by (2), and defined identically on AB and CD since $f(u,v) = f(u+\pi, v+\pi)$. The involution $(u,v) \leftrightarrow (v, u+\pi)$ maps square $ABFE$ to $FECD$ and vice versa, while $f(v, u+\pi) = -f(v,u) = -f(u,v)$. Consider any map $\alpha : \mathcal{P} \mapsto S^2$ which identifies sides AB and CD, contracts BD and AC to points, and maps every pair of points of the form (u,v) and $(v, u+\pi)$ in $\mathcal{P}$ to a pair of antipodal points $(p,-p)$ on S^2. Such an α yields an induced continuous function $f' : S^2 \mapsto R^2$ where $f'(\alpha(p)) = f(p)$ for $p \in \mathcal{P}$. Since f' is an antipodal map, it follows from the Borsuk-Ulam Theorem that f', and hence f, must map some point onto the origin of R^2. □

Main Theorem. Let A be a positive density function over a bounded connected region in R^3, and let $w_0 \in S^2$ be given. Then there exists a triple of planes (h_1, h_2, h_3) which forms an eight-partition for A, and where the normal vector of h_3 is w_0.

Proof. By Lemma 2.3, this is equivalent to finding h_1, h_2, h_3 such that $f_{ijk}(h_1, h_2, h_3) = 0$ for all $(i, j, k) \neq (0, 0, 0)$. Since h_1, h_2 and h_3 must be bisectors for A, and for any $u \in S^2$ there is a unique bisector h_u for A with normal vector u, we can define functions $g_{ijk} : S^2 \times S^2 \times S^2 \mapsto R$ by

$$g_{ijk}(u, v, w) = f_{ijk}(h_u, h_v, h_w). \tag{1}$$

The g_{ijk}'s are obviously continuous. It suffices to find u_0 and v_0 such that $g_{110} = g_{101} = g_{011} = g_{111} = 0$ at (u_0, v_0, w_0). We can limit the choice of u_0 (or v_0) to the set $\{u\}$ (or $\{v\}$) for which (h_u, h_{w_0}) (respectively (h_v, h_{w_0})) forms a four-partition, i.e.,

$$g_{101}(u, v, w_0) = 0 \quad \text{and} \quad g_{011}(u, v, w_0) = 0. \tag{2}$$

By Lemma 3.2, the set of (u, v) that satisfy (2) is homeomorphic to $S^1 \times S^1$. Our goal is thus to find a point $(u_0, v_0) \in S^1 \times S^1$ where both of the functions

$$G_0(u, v) \stackrel{\text{def}}{=} g_{110}(u, v, w_0) \quad \text{and} \quad G_1(u, v) \stackrel{\text{def}}{=} g_{111}(u, v, w_0)$$

are zero. Let $G \stackrel{\text{def}}{=} (G_0, G_1) : S^1 \times S^1 \mapsto R^2$. Note that G is symmetric in its two arguments and antipodal on each S^1 by Lemma 2.2 and (1):

$$G(u, v) = G(v, u),$$
$$G(u, -v) = -G(u, v).$$

Furthermore it is easy to verify that on the diagonal we have

$$G(u, u) = (1, 0).$$

It follows from Lemma 3.3 that there exists $(u_0, v_0) \in S^1 \times S^1$ such that $G(u_0, v_0) = (0, 0)$. We conclude that $(h_{u_0}, h_{v_0}, h_{w_0})$ yields the desired eight-partition. □

Lemma 3.4. Consider a partition of R^3 into eight open regions by three mutually intersecting planes. Any plane h in R^3 can intersect at most seven of these eight regions.

Proof. Define the origin O to be the point where the three planes meet, and axes X, Y, and Z to be the lines where pairs of planes intersect. Then O divides each axis into two half-axes $\{X^+, X^-\}$, $\{Y^+, Y^-\}$ and $\{Z^+, Z^-\}$. Without loss of generality, assume that h intersects the half-axes X^+, Y^+ and Z^+. Then h does not intersect the open region bounded by X^-, Y^- and Z^-. □

The Borsuk-Ulam theorem also leads to the following well-known corollary (see, e.g. [E]), which we shall employ in defining a data structure in Section 4.2. We state it in the discrete version for convenience.

Theorem. (*Ham-Sandwich Cut*) Let $S_1, S_2, \ldots, S_d$ be d finite point sets in R^d. There exists a hyperplane which simultaneously bisects $S_1, S_2, \ldots, S_d$.

4. Data Structures and Algorithms

Let S be a finite set of points in R^3. We will describe several tree structures for representing S, based on partitions of S by planes. These partitions are obtained by recursive applications of the theorems proved in the last section, and the resulting tree structures are suitable for the purpose of half-space retrieval and related searches on S.

We first present a recursive partition scheme based on the main theorem, giving a data structure termed an *octant tree*. Two variations of this basic scheme are derived by applying recursion in more intricate ways. The retrieval time is analyzed for each scheme, with the last variant achieving a retrieval time of $(n^{0.8988})$ for half-space query.

4.1. Octant Tree

An octant tree is a recursively defined structure for storing a finite set S of points in R^3. If S is empty the octant tree is the special node NIL, otherwise the root of the octant tree contains a plane h, and its left, middle and right children represent the subsets $S \cap h^-$, $S \cap h$ and $S \cap h^+$ respectively. More precisely, the middle child points to a 2-dimensional data structure for $S \cap h$ (such as a polygon tree [W] or a conjugation tree [EW]), while the left and right children point to the root nodes of octant trees for $S \cap h^-$ and $S \cap h^+$ respectively. We define the *domain* of any node v, denoted by $dom(v)$, to be the intersection of the 'regions' on the path from the root to v. That is, the domain of the root is the whole space, and if v is a child of w and h is the plane stored at w, then $dom(v)$ is the intersection of $dom(w)$ with h^-, h or h^+, depending on whether v is the left, middle or right child of w. The *set stored at* v, denoted by $S(v)$, is $S \cap dom(v)$.

The plane h at each node v of the octant tree is chosen so that both $|S(v) \cap h^-|$ and $|S(v) \cap h^+|$ are at most $|S(v)|/2$. There is no difficulty in finding planes with this property, but, without further conditions, such a data structure would have poor worst-case performance for half-space retrieval. (For example, Q-D trees [B] require $O(n)$ time in the worst case.) In the schemes to be described, we achieve better performance by appropriately grouping the tree nodes so that all nodes within a group can share a common plane. For this purpose, we build octant trees recursively from small *primitive trees* and define the grouping among the nodes of a primitive tree. The data structures pointed to by middle children represent the corresponding two-dimensional sets efficiently for a retrieval time of $O(t^\delta)$ for a set of size t, where $\delta \approx 0.695$ [EW]. These substructures are ignored in the recursive structures described below and contribute just $O(n^\delta)$ terms to the recurrence relations given.

Octant Tree A. In this basic scheme, the primitive tree is of height 3 and all nodes on the same level share the same plane. (See Figure 3.) The existence of such primitive trees is implied by the Main Theorem. The octant tree is obtained by applying recursion at the leaves of the primitive tree. In the figure solid nodes represent the roots of primitive trees.

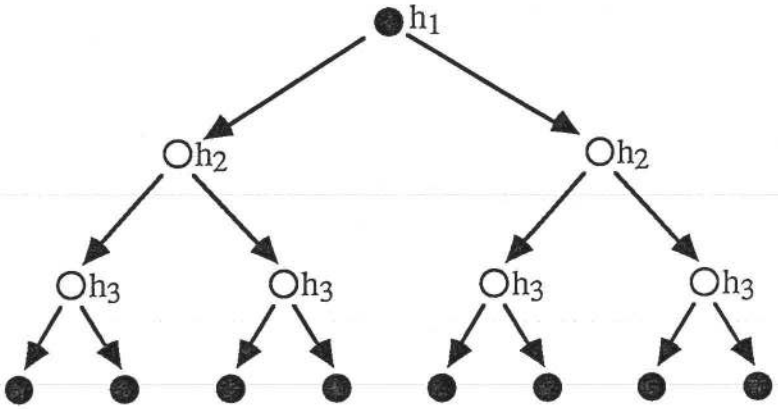

Figure 3

A search algorithm for the octant tree can be derived recursively from a search algorithm for the corresponding primitive tree. The general search strategy for primitive trees, with respect to a query plane q, is to identify those leaves in the primitive tree which need not be searched further. More precisely, a leaf v can be excluded from future search if $dom(v)$ lies completely in a half-space of q, since the entire set $S(v)$ can then be either reported or discarded. A leaf satisfying this condition is said to be *free* with respect to q.

By Lemma 3.4, the primitive tree has at least one free leaf with respect to any query q. Furthermore, the time required to identify the free leaves is bounded by a constant. Thus the search time for the derived octant tree is proportional to the total number of nodes visited. The recurrences below yield upper bounds for the search time. The reporting time, which is always linear in the size of the result, is not included here. Let $f(n)$ denote the maximum number of nodes visited in an octant tree for a set of n points. The constant δ arising from the two-dimensional subtrees has value at most 0.695.

Lemma 4.1. $f(n)$ satisfies the recurrence relation

$$f(n) \leq 7 + 7f(\frac{n}{8}) + O(n^{\delta}),$$

which gives a search time of $O(f(n)) = O(n^{\alpha})$ for $\alpha = \log_8 7 \approx 0.9358$.

Proof. In Figure 3, the seven upper nodes of the primitive tree are visited, followed by visits to the seven substructures of size at most $n/8$ corresponding to the non-free leaves.

The total number of nodes visited from middle children is at most $O(n^\delta)$ since $\delta < 1$. The linear recurrence relation is solved by standard techniques (see, e.g., Knuth [K]). □

4.2. Refined Octant Trees

Octant Tree B. In this variant of the basic scheme, we apply recursion to the four sets at level 2, while requiring that the same h_3 be used as the first plane for all four sets. The primitive tree is of depth two. (See Figure 4.) Here we take advantage of the strength of the Main Theorem, which allows one of the three partitioning planes to be an arbitrary bisector.

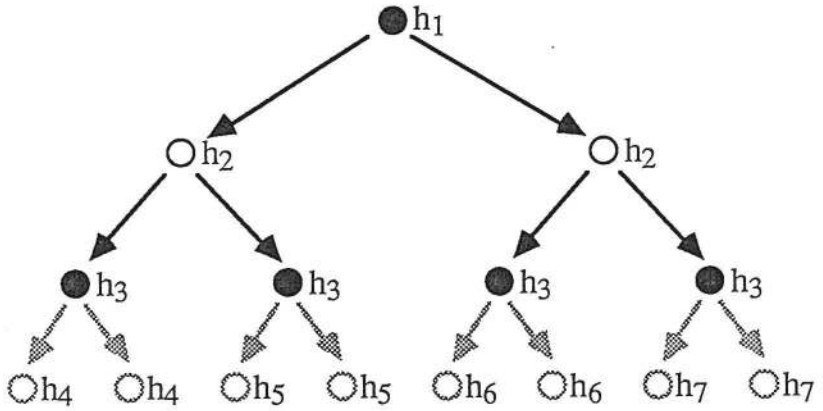

Figure 4

To estimate the search time for Scheme B, let $f(n)$ (and $g(n)$) denote the maximum number of nodes searched for any query, when the search starts at a root-level (and, respectively, second-level) node v of the primitive tree with $S(v) = n$.

Lemma 4.2. $f(n)$ and $g(n)$ satisfy the recurrence relations

$$f(n) \leq 4 + 3f(\frac{n}{4}) + g(\frac{n}{8}) + O(n^\delta),$$
$$g(n) \leq 1 + 2f(\frac{n}{2}) + O(n^\delta),$$

which give a search time under Scheme B of $O(f(n)) = O(n^\beta)$ where $\beta \approx 0.9163$.

Proof. Suppose without loss of generality that the rightmost of the eight lowest nodes, r, is free. The algorithm visits the three upper nodes and the parent of r, then recurses from the root level in the three left subtrees. In the fourth subtree, since r is free the search can begin at the second-level node which is r's sibling. This yields the first inequality. For the second, a search begun at a second-level node searches that node and recurses on its children, which are root-level nodes. Substituting the second inequality into the first, we have

$$f(n) \leq 5 + 3f(\frac{n}{4}) + 2f(\frac{n}{16}) + O(n^\delta).$$

The recurrence yields $f(n) = O(n^\beta)$ where $\beta \approx 0.9163$. □

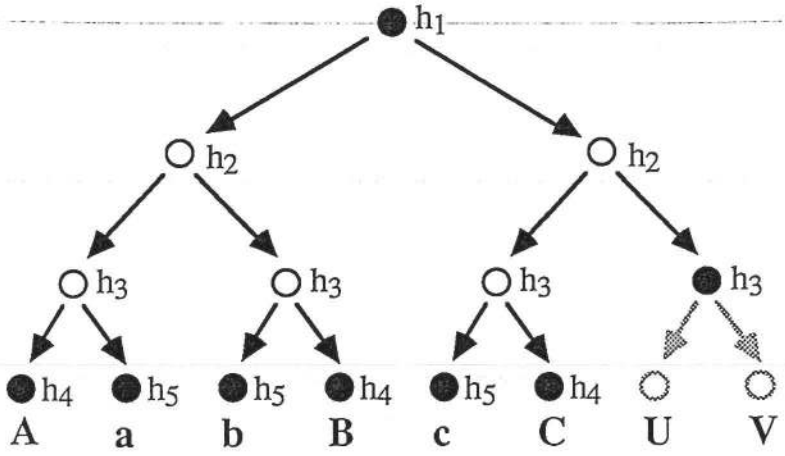

Figure 5

Octant Tree C. This is a hybrid of Scheme A and B with some further refinements. The primitive tree has six leaves on level 3 and one leaf on level 2. (See Figure 5.) The six leaves on level 3 are divided into two triplets where each triplet is to share a common first plane in the recursion. This is possible since, by the Ham Sandwich Theorem, any three point sets in R^3 can be bisected by a single plane. We choose each triplet to consist of three octants that do not share any common faces; indeed the six octants can be divided into two such triplets as shown in Figure 6, where the octants are represented as the vertices of a cube.

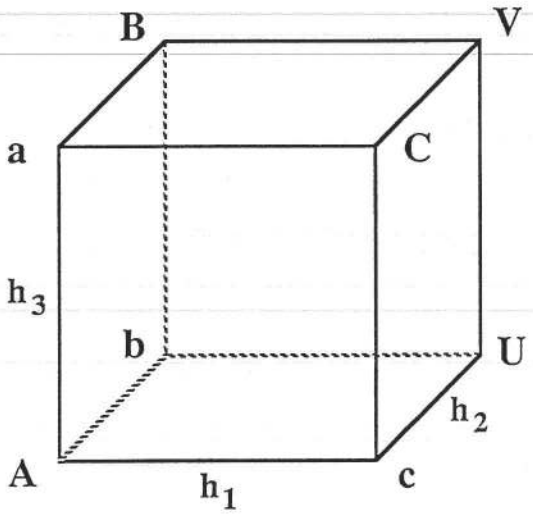

Figure 6

Again, define $f(n)$ and $g(n)$ as in in the analysis of Scheme B. We have the following bound for $f(n)$.

Lemma 4.3. Under Scheme C, $f(n)$ satisfies the recurrence relation

$$f(n) \leq 10 + f(\frac{n}{4}) + 4f(\frac{n}{8}) + 4f(\frac{n}{64}) + O(n^{\delta}),$$

which gives a search time under of $O(f(n)) = O(n^{\gamma})$ for $\gamma \approx 0.8988$.

Proof. There are a number of cases to consider depending on which of the eight nodes is free.

If A is free then the intersection of the query plane q with the other domains will be similar to Figure 7. The domains A, B and C are bisected by h_4, while h_5 bisects a, b and c. In the worst case, q will intersect both halves of B and C but, by our choice of the triplets, q cannot intersect both halves of a, b and c. (In Figure 7, the intersections of q with a, b and c form three regions which cannot be simultaneously intersected by the line representing the intersection of q and h_5.) For the case detailed in Figure 7, our algorithm is to search the six non-leaf nodes of the primitive tree, recursively search from the parent of U and V, recursively search a, b, B and C, search the node labelled c, and search from the second-level node corresponding to the child of c which is intersected by q. This yields the inequality

$$f(n) \leq 7 + f(\frac{n}{4}) + 4f(\frac{n}{8}) + g(\frac{n}{16}) + O(n^{\delta}). \tag{3}$$

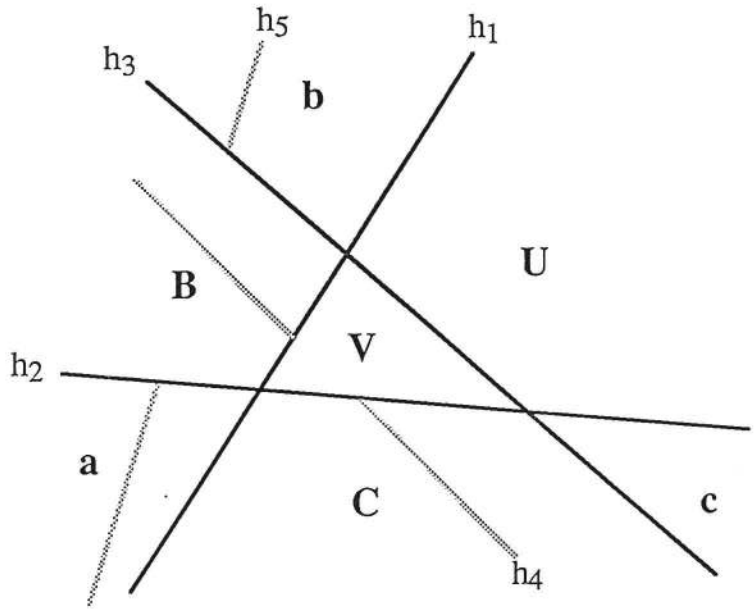

Figure 7

If U is free the intersection with q is similar to Figure 8. Here at most two of A, B and C and two of a, b and c can have both of their halves (with regard to h_4 and h_5 respectively)

intersected by q. For the case detailed in Figure 8, our algorithm is to search the upper seven nodes, search the nodes labelled B and b, recursively search a, c, A and C, recursively search the single children of B and b which are intersected by q and recursively search V. The corresponding inequality is

$$f(n) \leq 9 + 4f(\frac{n}{8}) + 2g(\frac{n}{16}) + g(\frac{n}{8}) + O(n^\delta). \tag{4}$$

We also have two inequalities for $g(n)$, depending on whether the second-level node where the search starts is the left child or the right child of a root node.

$$g(n) \leq 3 + 4f(\frac{n}{4}) + O(n^\delta), \tag{5}$$

$$g(n) \leq 2 + 2f(\frac{n}{4}) + f(\frac{n}{2}) + O(n^\delta). \tag{6}$$

The worst case is obtained by substituting (5) into (3), resulting in

$$f(n) \leq 10 + f(\frac{n}{4}) + 4f(\frac{n}{8}) + 4f(\frac{n}{64}) + O(n^\delta).$$

This yields $f(n) = O(n^\gamma)$ where $\gamma \approx 0.8988$. □

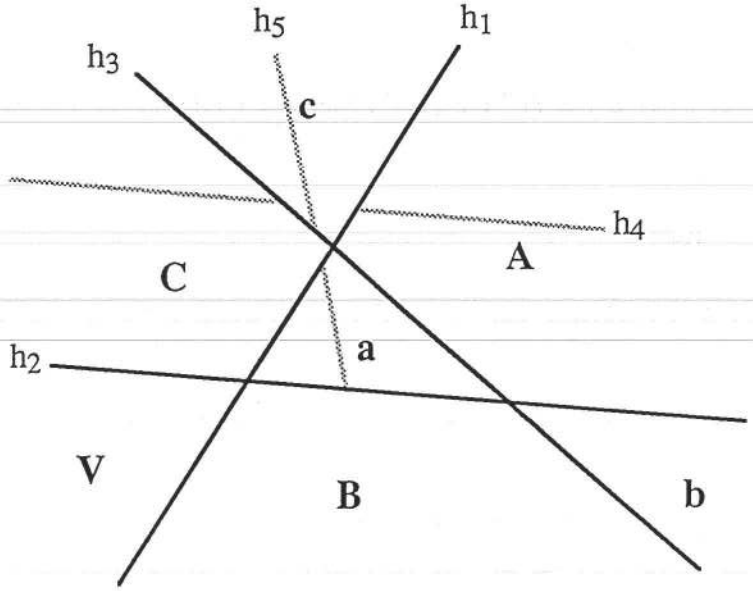

Figure 8

4.3. Preprocessing Cost

We look at the time it takes to construct an octant tree for a set S of n points. First consider the computation of an eight-partition (h_1, h_2, h_3) for S. The first bisector h_1 can be found in $O(n)$ time by a median-finding algorithm. We may assume that h_2 and h_3 are each determined by three points of S. For each of the $O(n^6)$ possible choices of (h_2, h_3), we decide

in $O(n)$ time whether it forms an eight-partition together with h_1 by explicitly counting the number of points in each octant. This amounts to a total time of $O(n^7)$ with linear space. The time can be reduced to $O(n^6)$ by lowering the cost due to counting as follows. For any two fixed points a, b of S, order the remaining points as $c_1, c_2, \ldots$ by projecting the points of S onto a plane perpendicular to $\overline{ab}$ and sorting them radially. Coplanar sets of more than three points introduce some complication but no significant difficulty. With such orderings imposed on h_3 during the search, the task of counting associated with each pair (h_2, h_3) becomes that of doing simple updates, with constant cost per pair on average. The total cost for finding an eight-partition, with sorting included, is thus $O(n^6)$ time using $O(n^3)$ storage. The storage could be made $O(n)$ by repeating the sorting operations whenever needed, but at a cost of $O(n^6 \log n)$ time.

The octant trees of schemes A and B can be constructed by applying the above procedure recursively, in total time $O(n^6)$ for n points with linear space. The same bounds hold for scheme C since a 'ham-sandwich cut' can be computed in $O(n^3 \log n)$ time.

4.4. Circular queries

The problem of finding all points (x, y) in a planar set S which lie inside a query circle C with center (a, b) and radius r can be transformed to a 3-dimensional half-space problem in the following way. Since

$$\begin{aligned}(x, y) \text{ lies inside } C &\iff (x-a)^2 + (y-b)^2 < r^2 \\ &\iff -2ax - 2by + (x^2 + y^2) < r^2 - a^2 - b^2,\end{aligned}$$

if we represent each point $(x, y) \in S$ as a 3-vector $\mathbf{v} = (x, y, x^2 + y^2)$ then the query with respect to circle C can be expressed by the half-space query:

$$(-2a, -2b, 1) \cdot \mathbf{v} < (r^2 - a^2 - b^2).$$

A geometrical interpretation of this is that, when the xy-plane is projected upwards onto the paraboloid $z = x^2 + y^2$, the image of any circle in the plane is the intersection of the paraboloid with a suitable plane. The same technique is applicable to other 'algebraic' queries, but the dimension required is the number of degrees of freedom of the defining polynomial.

Rather than transforming the circle query problem to three dimensions we can also recast our three-dimensional results in two dimensions. An 8-partition of the n points on the paraboloid can be projected down to the xy-plane yielding a partition by three circles. Our main theorem implies the following.

Corollary 4.4. For any finite point set and any bisecting circle in the plane, there are two circles such that each open region defined by the three circles contains at most one eighth of the set.

As observed by Marshall Bern, a query circle intersects at most 6 of the 8 regions since it meets the three circles in at most 6 points. Using (the two-dimensional projection of) octant tree A we therefore get $O(f(n))$ query time, where

$$f(n) \leq 7 + 6f(\frac{n}{8}) + O(\log n) = O(n^{\alpha}),$$

for $\alpha = \log_8 6 \approx 0.8617$. Here the $O(\log n)$ term takes care of the one-dimensional queries needed for points lying on the partitioning circles.

We get the same worst-case time using scheme C, but using octant tree B yields a slight improvement. The query time is $O(f(n))$ where

$$f(n) \leq 7 + 2f(\frac{n}{4}) + 4f(\frac{n}{16}) + O(\log n),$$

which solves to $f(n) = O(n^{\beta})$, with $\beta = \frac{\log_2 \sqrt{5}+1}{2} \approx 0.8471$.

4.5. Polyhedron queries

Based on our half-space query schemes, we may derive a generalization to queries for convex polyhedra defined by the intersection of r hyperplanes. For fixed r, the query time will be of the same order as for half-space queries, but the constant increases with r. Since every (not necessarily convex) polyhedron can be decomposed into (possibly unbounded) tetrahedra (see, e.g., [Ch]), it would suffice to consider at most tetrahedral queries, i.e., $r \leq 4$.

Let $f_r(n)$ be the number of nodes of the data structure for n points which may be searched in a query with a polyhedron C which is the intersection of r half-spaces, where we have already by our Scheme C above that $f_1(n) = O(n^{\gamma})$ for some $\gamma \leq 0.8988$. We prove by induction on r that $f_r(n) = O(n^{\gamma})$ for all r.

Suppose $f_{r-1}(n) = O(n^{\gamma})$ and consider a query with respect to the intersection of r half-spaces. In the case that some level 3 node is free with respect to all r planes, the recurrence relations considered above hold for f_r. In the alternative case, there are two level 3 nodes each free with respect to some plane or planes. Now the recurrence terms in f_r are diminished and so correspond to some exponent $\gamma' < \gamma$, while the remaining terms are of size $O(f_{n-1}) = O(n^{\gamma})$. The result is that $f_r(n) = O(n^{\gamma})$ and the induction is complete.

Of course the same general argument is valid for any similar scheme in any finite dimension.

5. Conclusion and Related Results

We showed that an eight-partition with three planes exists for any finite point set in R^3. It was brought to the authors' attention that a continuous version of this theorem was

proved earlier by H. Hadwiger with a more complicated argument [H]. As far as generalization to higher dimensions is concerned, David Avis [A] showed that 2^d-partitions are not always possible in dimen sions $d \geq 5$. A different and simpler proof is as follows (stated here for $d = 5$ but adaptable to any larger d). Take thirteen small balls of equal mass and place them in general position so that no hyperplane in R^5 can intersect more than five of the balls. Now, in any 2^5-partition, each ball must be cut by at least two hyperplanes, otherwise some orthant will contain at least one half of a ball, with at least 1/26 of the total mass, which is larger than the 1/32 required. Therefore, for all thirteen balls, at least 26 instances of hyperplane-ball intersections are needed. Since the balls are in general position, five hyperplanes can provide at most 25 such instances and we have a contradiction.

The case of $d = 4$ still remains an intriguing open question, that is, given any finite point set in R^4, whether one can always partition it with four hyperplanes such that each orthant contains at most 1/16 of the points. Our proof for $d = 3$ makes use of the Borsuk-Ulam Theorem of algebraic topology; the case of $d = 4$ is likely to draw further upon classical mathematics for its resolution.

Generalizations of eight-partition to higher dimensions have also been studied along a different line, by relaxing the number of hyperplanes used in the partition (see Cole [Co] and Yao and Yao [YY]). Deterministic partition schemes in three dimensions that are different from those described in this paper can be found in [EH]. Their main construction is based on the existence of a 6-partition for every planar point set, that is, a partition by three concurrent lines so that every wedge contains at most one sixth of the points. The best such scheme achieves $O(n^\alpha)$ query time, where $\alpha \approx 0.9089$.

Haussler and Welzl [HW] used random sampling to demonstrate the existence of partitions in R^d which afford the best query time currently known. In particular, for $d = 3$ their scheme gives query time $O(n^\alpha)$ for $\alpha \approx 0.857$. As their algorithms are probabilistic, it is an interesting open question to find deterministic algorithms for constructing partitions to realize similar or even better query time in R^d.

Chazelle [Ch2] established a lower bound under a rather general model for range search in d dimensions. His bound in three dimensions assuming $O(n)$ space is $\Omega(n^{\frac{3}{4}})$ query time.

References

[A] D. Avis, Non-partitionable point sets, *Inf. Proc. Letters* 19 (1984), 125-129.

[B] J. L. Bentley, Multidimensional binary search trees used for associative searching, *CACM* 18 (1975), 509-516.

[Ch1] B. Chazelle, Convex partitions of polyhedra: a lower bound and worst-case algorithm, *SIAM J. on Computing* **13** (1984), 488-507.

[Ch2] B. Chazelle, Polytope range searching and integral geometry, *Proc. 28th Ann. IEEE Sympos. Found. Comput. Sci.* (1987), 1-10.

[Co] R. Cole, Partitioning point sets in arbitrary dimensions, Report 184, Dept. of Computer Science, New York University, New York, NY.

[DE] D. P. Dobkin and H. Edelsbrunner, Space searching for intersecting objects, *J. of Algorithms* **8** (1987), 348-361.

[E] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag (1987).

[EH] H. Edelsbrunner and F. Huber, Dissecting sets of points in two and three dimensions, Report F 138, Inst. Informationsverarb., Techn. Univ. Graz, Austria, 1984.

[EW] H. Edelsbrunner and E. Welzl, Halfplanar range serch in linear space and $O(n^{0.695})$ query time, *Inf. Proc. Letters* (1986).

[H] H. Hadwiger, Simultane Vierteilung zweier Körper, *Arch. Math. (Basel)* **17** (1966), 274-278.

[HW] D. Haussler and E. Welzl, ϵ−nets and simplex range queries, *Discrete and Computational Geometry* **2** (1987), 127-151.

[K] D. E. Knuth, *Fundamental Algorithms: The Art of Computer Programming I* Addison-Wesley, Reading, MA, 1968.

[Me] N. Megiddo, Partitioning with two lines in the plane, *J. of Algorithms* **3** (1985), 430-433.

[Mu] J. R. Munkres, *Elements of Algebraic Topology,* Addison-Wesley, 1984.

[W] D. E. Willard, Polygon retrieval, *SIAM J. on Computing* **11** (1982), 149-165.

[YB] I. M. Yaglom and V. G. Bolyanski, *Convex Figures,* (English translation) Holt, Rinehart and Winston, New York, NY (1961).

[Y] F. F. Yao, A 3-space partition and its applications, *Proc. 15th Ann. ACM Sympos. on Theory of Computing* (1983), 258-263.

[YY] A. C. Yao and F. F. Yao, A general approach to d-dimensional geometric queries, *Proc. 17th Ann. ACM Sympos. on Theory of Computing* (1985), 163-168.