Algorithms for Center and Tverberg Points*

Pankaj K. Agarwal† Micha Sharir‡ Emo Welzl§

September 18, 2006

Abstract

Given a set S of n points in $\mathbb{R}^3$, a point x in $\mathbb{R}^3$ is called *center point of* S if every closed halfspace whose bounding hyperplane passes through x contains at least $\lceil n/4 \rceil$ points from S. We present a near-quadratic algorithm for computing the *center region*, i.e., the set of all center points, of a set of n points in $\mathbb{R}^3$. This is nearly tight in the worst case since the center region can have $\Omega(n^2)$ complexity.

We then consider sets S of $3n$ points in the plane which are the union of three disjoint sets consisting respectively of n red, n blue, and n green points. A point x in $\mathbb{R}^2$ is called a *colored Tverberg point of* S if there is a partition of S into n triples with one point of each color, so that x lies in all triangles spanned by these triples. We present a first polynomial-time algorithm for recognizing whether a given point is a colored Tverberg point of such a 3-colored set S.

1 Introduction

Given a set S of n points in $\mathbb{R}^d$ and a point $x \in \mathbb{R}^d$, the *depth* of x with respect to S is the minimum number of points of S contained in a closed halfspace whose bounding hyperplane passes through x. The set of all points of depth k is called the *depth-k region* of S. The region of points of depths at least k, which we denote by $c_k(S)$, can be written as $\bigcap_{h \in \mathcal{H}_{\geq n-k+1}} h$, where $\mathcal{H}_{\geq j} = \mathcal{H}_{\geq j}(S)$ is the set of all closed halfspaces that contain at least j points from S (for a proof see Lemma 2.1 below).

Note that $c_1(S) = \mathrm{conv}(S)$, where $\mathrm{conv}(S)$ denotes the convex hull of S.

*Work by P.A. and M.S. was supported by a grant from the U.S.-Israeli Binational Science Foundation. Work by P.A. was also supported by NSF under grants CCR-00-86013 EIA-98-70724, EIA-99-72879, EIA-01-31905, and CCR-02-04118. Work by M.S. was also supported by NSF Grants CCR-97-32101 and CCR-00-98246, by a grant from the Israel Science Fund (for a Center of Excellence in Geometric Computing), and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University. A preliminary version of this paper has appeared in *Proc. 20th ACM Annu. Sympos. Comput. Geom.*, 2004, pp. 61–67.

†Dept. Computer Science, Duke University, Durham, NC 27708-0129, USA. pankaj@cs.duke.edu

‡School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel, and Courant Inst. of Math. Sci., 251 Mercer Street, NYC, NY 10012, USA. michas@post.tau.ac.il

§Inst. Theoretische Informatik, ETH Zürich, CH-8092 Zürich, Switzerland. emo@inf.ethz.ch

If $j > dn/(d+1)$, then any $d+1$ halfspaces in $\mathcal{H}_{\geq j}$ have a point of S in common, and thus Helly's theorem (cf. [13, 18]) implies $\bigcap_{h\in\mathcal{H}_{\geq j}} h \neq \emptyset$. That is, if $n-k+1 > dn/(d+1)$, or equivalently, if $k \leq \lceil n/(d+1)\rceil$, the region $c_k(S)$ is nonempty (as was first observed by Rado [22]).

Points of depth at least $\lceil n/(d+1)\rceil$ are called *center points* of S, and the region $c_{\lceil n/(d+1)\rceil}(S)$—which we know is nonempty—is called the *center region* of S, denoted by $c(S)$.

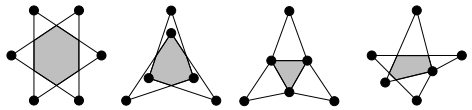


Figure 1. Sets of six points in the plane, with their center regions, the points of depth at least 2. Equivalently, these are the intersections of all halfplanes containing at least 5 points. (The meaning of the extra edges indicated will be explained later.)

Relatively little progress has been made on the algorithmic issues related to center points. Teng [25] showed that if d is part of the input, the problem of determining whether a given point is a center point of S is coNP-complete. Jadhav and Mukhopadhyay [14] gave a linear-time algorithm for computing a center point in $\mathbb{R}^2$. Matoušek [16] later developed an $O(n\log^4 n)$-time algorithm for computing the center region of a set of n points in $\mathbb{R}^2$ (see [8, Remark pg. 427] for a possible improvement and Miller et al. [20] for an $O(n^2)$-time algorithm that computes all depth regions). Naor and Sharir [21] gave an $O(n^2\,\mathrm{polylog}(n))$ algorithm for computing a center point in $\mathbb{R}^3$; very recently, Chan [8] supplied a randomized $O(n\log n + n^{d-1})$ algorithm for this problem in $\mathbb{R}^d$. For computing the center region in $\mathbb{R}^3$ there is a naïve cubic algorithm, that can be improved to $O(n^{5/2+\varepsilon})$ by running a k-level construction algorithm in the dual and computing the convex hull of the obtained vertices; (this relies on an output-sensitive algorithm by [1] for computing the k-level, and on a bound of $O(n^{5/2})$ on the complexity of a k-level in $\mathbb{R}^3$ [24]). Clarkson *et al.* [10] proposed a simple algorithm for computing an approximate center point of a set of points in arbitrary dimensions.

A variation of center points are so-called Tverberg points. Let r be a positive integer. A partition of S into r disjoint subsets $S_1,\ldots,S_r$ is called a *Tverberg partition* if $\bigcap_{i=1}^{r}\mathrm{conv}(S_i)\neq\emptyset$, and a point lying in the intersection is called an *r-divisible point*. Tverberg [26] proved that if $|S| > (d+1)(r-1)$, then a Tverberg partition always exists; $\lceil |S|/(d+1)\rceil$-divisible points, which therefore always exist, are called *Tverberg points of S*. Subsequent to his original proof, several alternative proofs have been proposed; see the book by Matoušek [18] and the survey paper by Kalai [15] for a history of the problem. Note that any halfspace containing an r-divisible point contains at least r points of S, so we get that every $\lceil n/(d+1)\rceil$-divisible point (i.e., Tverberg point) of S is also a center point of S. For $d=2$ and n a multiple of 3, the converse is also true, i.e., every center point of S is also a Tverberg point of S (i.e., $(n/3)$-divisible), but it is not true for $d\geq 3$ [3]. Teng [25] showed that if d is part of the input, then the problem of determining whether a given point is an r-divisible point of S is NP-complete. In fact no polynomial-time algorithm is known even if $d>2$ is fixed. On the other hand, if d is fixed, a polynomial-time algorithm for finding a Tverberg point

(with running time $n^{O(d^2)}$) can be obtained by modifying Tverberg's existence proof. For $d = 2$ and n a multiple of 3, a Tverberg point can be computed in linear time using the algorithm for computing a center point [14], and we can determine in $O(n \log n)$ time whether a point is a Tverberg point.

Bárány, Füredi and Lovász [5] suggested (and proved for $d = 2$) a *colored version of Tverberg's theorem*, which was then established by Živaljević and Vrećica [28] in arbitrary dimensions. In this version, the points of S are colored by $d + 1$ colors, and we require that each of the sets S_i in the partition contain at least one point of each color. The planar case allowed a quantitative improvement, provided by Bárány and Larman [4]. They showed that given a planar set S, which is the disjoint union of three sets R, B, G consisting respectively of n red points, n blue points, and n green points (in general position), there exists a partition of S into n pairwise-disjoint triples $S_1, \ldots, S_n$, where each triple consists of one point of each of R, B, G, so that $\bigcap_{i=1}^{n} \mathrm{conv}(S_i) \neq \emptyset$. In fact, their argument is constructive and yields an $O(n^6)$-time algorithm for computing such a partition. However, no polynomial-time algorithm is known for determining whether a given point is a colored Tverberg point of S, for this planar setting. We remark that, unlike the uncolored case, the set of colored Tverberg points can be a proper subset of the center region, and it does not even have to be convex, as is illustrated in Figure 2.

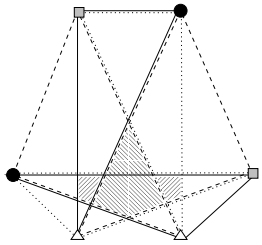


Figure 2. A non-convex region of colored Tverberg points in the plane, an example due to [12].

Our results. We present two main results in this paper. First, we describe, for a given set S of n points in $\mathbb{R}^3$, an $O(n^{2+\varepsilon})$-time[1] algorithm for computing the center region $c(S)$ (cf. Section 2). In fact, we show that for any j, $c_j(S)$ has $O(n^2)$ complexity, and that it can be computed in $O(n^{2+\varepsilon})$ time. By performing a binary search, we can compute the nonempty region of maximum depth in $O(n^{2+\varepsilon})$ time. We use the fact that the problem amounts to computing the convex hull of a k-level Λ of the arrangement dual to S of n planes. Our approach is to first solve n subproblems on the respective planes in time $O(n^{1+\varepsilon})$ per problem, and the challenge is to find an appropriate approximation of the obvious structures in these planar arrangements (such as the intersection of the k-level with the plane, or the convex hull thereof), since each individual planar structure might have prohibitively large complexity (albeit their combined complexity is only $O(n^2)$). We build heavily on Matoušek's algorithm [16] for computing the convex hull of a level of lines in $\mathbb{R}^2$. As

[1]The meaning of bounds of this form, is that they hold for any $\varepsilon > 0$, where the constant of proportionality depends on ε, and generally tends to ∞ as $\varepsilon \to 0$.

such, heavy algorithmic tools are used and we consider the method a more theoretical contribution (although appropriate relaxations may lead to an effectively implementable procedure).

Next, given a set S of $3n$ points in $\mathbb{R}^2$, which is the disjoint union of three sets R, B, and G consisting of n points each, we present a polynomial-time algorithm to determine whether a point q is a colored Tverberg point of S (cf. Section 3). The running time of the algorithm is $O(n^{11})$. As far as we are aware, this is the first polynomial-time algorithm for this problem.

2 The Center Region

We first make sure that the problem of computing the the center region, or $\bigcap_{h\in\mathcal{H}_{\geq j}} h$ in general, is a finite discrete problem. Then we discuss the structure of the center region in the dual, before we proceed to the description of the algorithm for its construction. Most of the structural properties hold in any dimension, and we present them for arbitrary d, before restricting ourselves to $d = 3$ for the algorithmic construction.

2.1 The center region and j-facets

We assume that $|S| \geq d + 1$ and that S is in general position, i.e., no $d + 1$ points lie in a common hyperplane. A *j-facet* is an oriented simplex spanned by d points in S that has exactly j points of S on the (open) positive side of its affine hull. $\overline{\mathcal{H}}_j$ is the set of closed halfspaces that contain j points of S and have d points of S on their boundary. Clearly, these are exactly the halfspaces induced by $(j - d)$-facets. Figure 1 displays all 3-facets (orientations omitted) of the respective point sets.

Lemma 2.1 *For any set S of $n \geq d + 1$ points in general position in $\mathbb{R}^d$ and any positive integer $j \leq n$ we have:*

(1) $c_{n-j+1}(S) = \bigcap_{h\in\mathcal{H}_{\geq j}} h = \bigcap_{h\in\overline{\mathcal{H}}_j} h$.

(2) $\bigcap_{h\in\mathcal{H}_{\geq j}} h$ *is a convex polytope (not necessarily of full dimension) with at most* $2\binom{n}{d-1}$ *facets, each of which is contained in some $(j - d)$-facet of S.*

Proof: (1) For a proof of the left identity, note that if $x \notin h \in \mathcal{H}_{\geq j}$, then the depth is at most $n - j$ since the complement of h contains a closed halfspace with x on its boundary and containing at most $n - j$ points from S. On the other hand, if x lies on the boundary of a closed halfspace with at most $n - j$ points, then the complement contains a closed halfspace with at least j points and thus $x \notin \bigcap_{h\in\mathcal{H}_{\geq j}} h$.

For the right identity, consider first the set $\overline{\mathcal{H}}_{\geq j} := \bigcup_{i\geq j} \overline{\mathcal{H}}_i$. Since $\overline{\mathcal{H}}_{\geq j} \subseteq \mathcal{H}_{\geq j}$, the inclusion

$$\bigcap_{h\in\mathcal{H}_{\geq j}} h \subseteq \bigcap_{h\in\overline{\mathcal{H}}_{\geq j}} h$$

readily follows. For each $h \in \mathcal{H}_{\geq j}$, there are halfspaces in $\overline{\mathcal{H}}_{\geq j}$ whose intersection is contained in h (take the halfspaces containing $S \cap h$ and bounded by hyperplanes carrying facets of $\mathrm{conv}(S \cap h)$—we call these the *facet-supporting halfspaces* of the polytope $\mathrm{conv}(S \cap h)$). It then follows that all $h \in \mathcal{H}_{\geq j} \setminus \overline{\mathcal{H}}_{\geq j}$ are redundant. Hence, $\bigcap_{h \in \mathcal{H}_{\geq j}} h = \bigcap_{h \in \overline{\mathcal{H}}_{\geq j}} h$ and $\bigcap_{h \in \mathcal{H}_{\geq j}} h$ is a polyhedron whose facets are determined by (i.e., lie in the boundary of) some of the halfspaces $h \in \overline{\mathcal{H}}_{\geq j}$. On the other hand, every halfspace h in $\overline{\mathcal{H}}_i$, for $i > d$, contains an intersection of halfspaces in $\overline{\mathcal{H}}_{i-1}$; take the facet-supporting halfspaces of the polytopes $\mathrm{conv}((S \cap h) \setminus \{q\})$ for each of the d points $q \in S$ that lie on the boundary of h. Therefore all halfspaces in $\overline{\mathcal{H}}_{\geq j} \setminus \overline{\mathcal{H}}_j$ are redundant. Assertion (1) of the lemma follows.

(2) We have already shown that $P := \bigcap_{h \in \mathcal{H}_{\geq j}} h$ is an intersection of a finite number of halfspaces, and since it is contained in $\mathrm{conv}(S)$, it is bounded and thus a polytope. We further reduce the number of constraints in $\overline{\mathcal{H}}_j$ that are needed to determine P.

If P is empty, we are done since then it is the intersection of $d+1 \leq 2\binom{n}{d-1}$ halfspaces in $\overline{\mathcal{H}}_j$ (consult Helly's Theorem). Otherwise, consider some set K of $d-1$ points in S. Either no $(j-d)$-facet contains K, or all but two halfspaces in $\overline{\mathcal{H}}_j$ with K on their boundary are redundant. Since the number of $(d-1)$-element subsets of S is $\binom{n}{d-1}$, the asserted upper bound on the number of facets follows.

Now consider a halfspace $h \in \overline{\mathcal{H}}_j$, and the $(j-d)$-facet φ it contains in its boundary. We slightly rotate the bounding hyperplane π of h about any $d-1$ of the points of φ, while keeping φ in its halfspace. We obtain d halfspaces in $\mathcal{H}_{\geq j}$, whose intersection with π is φ, implying that no portion of π outside φ can be part of P. □

2.2 The structure of the center region

Let S be a set of n points in $\mathbb{R}^d$ in general position. A standard duality transform [13] maps a point p in $\mathbb{R}^d$ to a nonvertical hyperplane p^* in $\mathbb{R}^d$ and vice versa, so that the above/below relationships between the points and hyperplanes are preserved, i.e., if p lies below (resp., above, on) a nonvertical hyperplane h, then the dual hyperplane p^* lies below (resp., lies above, contains) the point h^*. Using this duality transform, we map S to a set S^* of n nonvertical hyperplanes in $\mathbb{R}^d$. The *level* of a point $x \in \mathbb{R}^d$ with respect to S^* is the number of hyperplanes in S^* lying strictly below x. All points on the same (relatively open) face (of any dimension) of $\mathcal{A}(S^*)$ have the same level. For a given integer $0 \leq k < n$, the *k-level* of $\mathcal{A}(S^*)$, denoted as $\Lambda_k(S^*)$, is the union of the closures of all facets of $\mathcal{A}(S^*)$ whose level is k. $\Lambda_k(S^*)$ is the graph of a continuous, piecewise-linear $(d-1)$-variate function.

By construction, the dual of a point x whose depth is k ($1 \leq k \leq n/2$) with respect to S is a hyperplane that lies between $\Lambda_{k-1}(S^*)$ and $\Lambda_{n-k}(S^*)$. More precisely if we let L_k (resp., U_k) denote the lower (resp., upper) convex hull of $\Lambda_k(S^*)$, then x^* (weakly) separates U_{k-1} and L_{n-k}. Hence, the dual of the center region $c(S)$ is the region lying between $U_{\lceil \frac{n}{d+1} \rceil - 1}$ and $L_{\lfloor \frac{dn}{d+1} \rfloor}$. In order to compute $c(S)$, it suffices to describe an algorithm for computing the upper or lower convex

hull of a level in $\mathcal{A}(S^*)$. Indeed, with $U_{\lceil \frac{n}{d+1} \rceil - 1}$ and $L_{\lfloor \frac{dn}{d+1} \rfloor}$ available, we map them back to the primal space, and intersect the two resulting unbounded polyhedra to obtain $c(S)$.

The following lemma follows from the observation that the intersection line of any $d-1$ hyperplanes of S^* intersects $\mathrm{conv}(\Lambda_k(S^*))$ in at most two points, unless it overlaps the hull in an edge (the dual analogy of the argument in the proof of Lemma 2.1(2)).

Lemma 2.2 *Let S^* be a set of n hyperplanes in $\mathbb{R}^d$. For any $0 \leq k < n$, the convex hull of $\Lambda_k(S^*)$ has $O(n^{d-1})$ vertices.*

Lemma 2.3 *Let S be a set of n points in $\mathbb{R}^3$. The combinatorial complexity of $c(S)$ is $O(n^2)$, and this bound is tight in the worst case.*

Proof: The upper bound follows directly from Lemma 2.1 (2).

For the lower bound, let us first assume that n is of the form $12k^2$, for an integer $k \geq 1$. We are interested in $c(S) = c_{n/4}(S) = \bigcap_{h \in \overline{\mathcal{H}}_{3n/4+1}} h$, the intersection of halfspaces induced by $(3n/4-2)$-facets (see Lemma 2.1).

Take a triangle Δuvw in the xy-plane, pass a vertical line through each of its three vertices u, v, w, and place $n/3$ points on each line at heights $\sqrt{1}, \sqrt{2}, \ldots, \sqrt{n/3}$. This yields a set S of n points in $\mathbb{R}^3$. We fix $j = 3n/4-2$, and we consider the j-facets of S that have one point on each of the vertical lines. There are $\Theta(j^2) = \Theta(n^2)$ choices of triples (a, b, c) such that $a + b + c = j + 3$, and any such triple defines a j-facet whose vertices are $(u, \sqrt{a})$, $(v, \sqrt{b})$, $(w, \sqrt{c})$. Moreover, any j-facet either has this form, or it corresponds to a triple (a, b, c) with $a + b + c = n - j$.

Passing to the dual space, a plane containing any such j-facet becomes the intersection point of the three dual planes (where $\mathbf{x}$ denotes the vector (x, y) in the xy-plane)

$$z = u \cdot \mathbf{x} + \sqrt{a}, \quad z = v \cdot \mathbf{x} + \sqrt{b}, \quad \text{and} \quad z = w \cdot \mathbf{x} + \sqrt{c}.$$

Clearly, any such dual point lies on the ellipsoid

$$(z - u \cdot \mathbf{x})^2 + (z - v \cdot \mathbf{x})^2 + (z - w \cdot \mathbf{x})^2 = a + b + c = j + 3.$$

Since this is a convex surface, standard properties of the duality transform imply that each of the planes containing the j-facets in the primal space is tangent to a convex surface. Since each of these tangency points necessarily lies on the boundary of the intersection polytope, it follows that each of these j-facets contributes a facet to $c(S)$. The lower bound follows for the special values of n assumed so far, and it can be easily extended to other values of n. □

Remark. We are currently unable to prove a similar bound on the complexity of $c(S)$ in higher dimensions. The upper bound theorem for convex polytopes (see [27]) implies that the complexity of $c(S)$ is $O(n^{(d-1)\lfloor d/2 \rfloor})$. However, we conjecture that the actual bound is much smaller, maybe even $O(n^{d-1})$.

2.3 Computing the convex hull of a level

Let H be a set of n planes in $\mathbb{R}^3$ in general position, and let $k < n$ be an integer. We describe an algorithm for computing the upper convex hull of $\Lambda = \Lambda_k(H)$.

Lemma 2.4 $\operatorname{conv}(\Lambda) = \operatorname{conv}\Big(\bigcup_{h\in H} \operatorname{conv}(\Lambda \cap h)\Big)$.

Proof: Λ is composed of relative closures of facets of the arrangement of H, thus $\Lambda = \bigcup_{h\in H}(\Lambda\cap h)$. The claim then follows, since $\operatorname{conv}(A \cup B) = \operatorname{conv}(\operatorname{conv}(A) \cup \operatorname{conv}(B))$. □

Lemma 2.5 *For any $h \in H$, $\operatorname{conv}(\Lambda \cap h)$ has linear complexity.*

Proof: Any vertex of $\Lambda \cap h$, and thus also of $\operatorname{conv}(\Lambda \cap h)$, lies on two of the lines in $\{g \cap h \mid g \in H \setminus \{h\}\}$, and any such line $g \cap h$ can contain at most two vertices of $\operatorname{conv}(\Lambda \cap h)$. A similar reasoning is applied to the unbounded edges of $\Lambda \cap h$. □

Our algorithm processes the planes of H one at a time. For each plane h it computes a convex (generally unbounded) polygon K_h with $O(n)$ edges and with the property that

$$\operatorname{conv}(\Lambda \cap h) \subseteq K_h \subseteq \operatorname{conv}(\Lambda) \cap h. \tag{1}$$

By Lemma 2.4, $\operatorname{conv}(\bigcup_{h\in H} K_h) = \operatorname{conv}(\Lambda)$, so we simply compute and output the convex hull of $\bigcup_{h\in H} K_h$.

Although Λ is the graph of a continuous piecewise-linear totally defined function of x, y, the set $\Lambda\cap h$ within a single plane $h \in H$ is much less structured. It need not even be connected, and can in fact have quadratic complexity (in contrast with its convex hull, which has only linear complexity). Specifically, for each $g \in H \setminus \{h\}$, consider the halfplane $g^+ \cap h$ within h, where g^+ is the (closed) halfspace bounded from below by g. The level of a point $w \in h$ is the number of halfplanes $g^+ \cap h$ that contain w. These halfplanes can have a rather "erratic" structure, such as the one shown in Figure 3, which may cause $\Lambda \cap h$ to consist of up to $\Theta(n^2)$ connected components.

Fix a coordinate frame within h whose axes project vertically to the x and y-axes of the 3-dimensional frame. (This is not an orthogonal frame, but can be made so with an appropriate affine transformation within h.) Let $g \in H \setminus \{h\}$. Write the equations of h and of g as $z = a_h x + b_h y + c_h$ and $z = a_g x + b_g y + c_g$. Observe that the halfplane $g^+ \cap h$ is a upper (resp., lower) halfplane within h in this coordinate frame if and only if $b_g > b_h$ (resp., $b_g < b_h$). Thus each connected component of $\Lambda \cap h$ is bounded from below (resp. above) by halfplanes g^+ with $b_g > b_h$ (resp. $b_g < b_h$). The general-position assumption, and an appropriate choice of the coordinate frame, allow us to assume that all the coefficients b_h, for $h \in H$, are distinct.

Sort the planes $h \in H$ in decreasing order of the y-coefficients b_h of their equations. Let the sorted order be $h_1, \ldots, h_n$. We first define the polygonal region $K_j = K_{h_j}$ with the property (1) and then present an algorithm for computing each K_j.

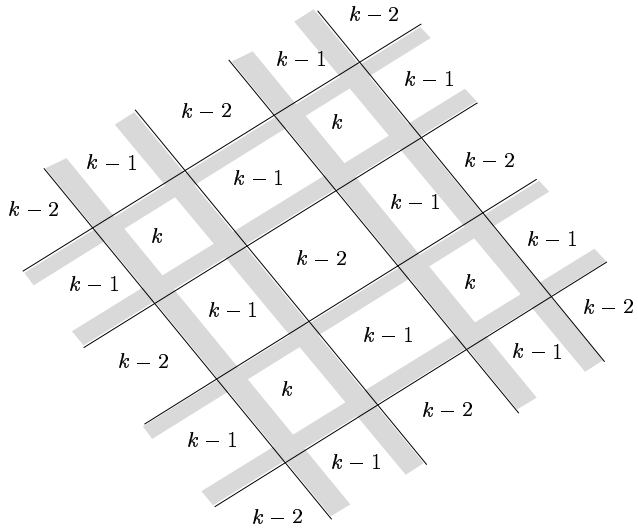


Figure 3. $\Lambda_k(H) \cap h$ may have quadratic complexity.

Definition of K_j. We give an inductive definition of K_j. All halfplanes $g^+ \cap h_1$ are upper halfplanes, implying that $\Lambda \cap h_1$ is a level of the arrangement of the lines $g \cap h_1$, for $g \in H \setminus \{h_1\}$. We set K_1 to be $\mathrm{conv}(\Lambda \cap h_1)$. Suppose $K_1, \ldots, K_{j-1}$ have been defined.

For $i < j$, let $\gamma_i = K_i \cap h_j$—these are segments, rays, or lines. Put $\Gamma_j = \{\gamma_i \mid 1 \leq i < j\}$, and refer to these segments (rays, or lines) as *red segments*. Let $R_j \subseteq h_j$ denote the convex hull of Γ_j. (Let us note right away that each γ_i can be computed in $O(\log n)$ time, by forming the line $h_i \cap h_j$ and by intersecting it, within h_i, with K_i. Thus R_j can be computed in $O(n \log n)$ time.) Refer to $h_i \cap h_j$, $i > j$, as *green lines*. We define

$$K_j = \mathrm{conv}(R_j \cup (\Lambda \cap h_j)),$$

and set $\zeta_j = \partial K_j$. Next we prove a few properties of $\Lambda \cap h_j$ and K_j that will be useful in the construction of K_j.

Lemma 2.6 *The portion of $\Lambda \cap h_j$ that lies on green lines consists of pairwise disjoint x-monotone polygonal chains, each starting and ending either at infinity or on some red segment.*

Proof: Let w be a point in $\Lambda \cap h_j$ that lies on a green line, and trace $\Lambda \cap h_j$ from w to the right (i.e., in the positive x-direction). When encountering a new green line, the level switches to the new line and continues to the right; this follows from the fact that all the green halfplanes $h_i^+ \cap h_j$ are upper halfplanes in h_j. Continuing the tracing, we either reach $+\infty$ or an intersection point u with some line $h_i \cap h_j$, with $i < j$. Then $u \in \Lambda \cap h_i$, so $u \in K_i$, and thus $u \in \gamma_i$. Tracing the level from w to the left, and repeating this analysis to each connected component of $\Lambda \cap h_j$ completes the proof of the lemma. □

Lemma 2.7 *Let u, v be two y-covertical green points in $\Lambda \cap h_j$. Then the segment uv must cross at least one red segment.*

Proof: Suppose that u lies above v (in the y-direction). As we move from v in the positive y-direction, the level increases by 1 to $k+1$. Just before reaching u, the level is restored to its original value k. Thus there had to be a point w on uv so that after crossing w the level *decreases* by 1 and becomes k. Clearly, w has to lie on some line $h_i \cap h_j$ with $i < j$ and in $\Lambda \cap h_j$. Hence, arguing as above, $w \in \gamma_i$. □

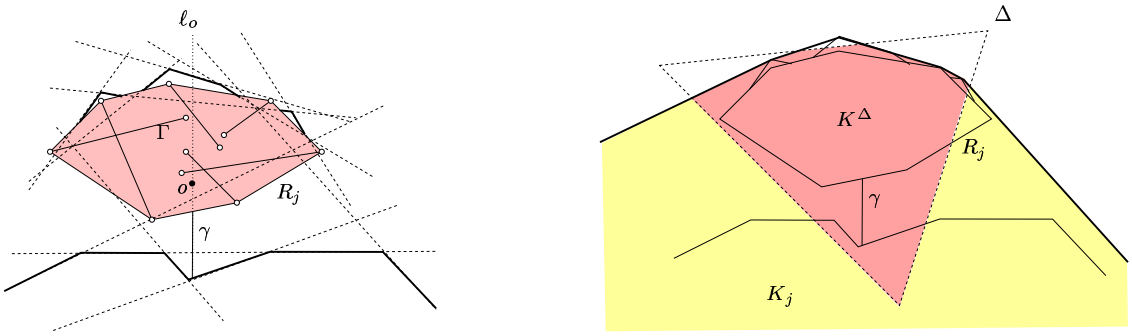
Corollary 2.8 *$R_j \cup (\Lambda \cap h_j)$ is either a connected set or the union of up to two connected components, one of which contains R_j and the other one is an x-monotone unbounded green portion of $\Lambda \cap h_j$, passing either above or below R_j.*

Proof: Suppose $R_j \cup (\Lambda \cap h_j)$ consists of three or more connected components, then *two* of them are disjoint from R_j. By Lemmas 2.6 and 2.7, R_j is unbounded and lies between these two components. Moreover, the convexity of R_j implies that the unbounded rays of ∂R_j are parallel to those of the green components. However, this is ruled out by the general-position assumption, so there is at most one connected (unbounded) green component disjoint from R_j. □

Constructing K_j. Since K_1 is the convex hull of a level in an arrangement of n lines, it can be computed in $O(n \log^4 n)$ time using the algorithm of Matoušek [16]. Assuming $K_1, \ldots, K_{j-1}$ have been computed, we construct K_j as follows. Since both R_j and $\mathrm{conv}(\Lambda \cap h_j)$ have linear complexity, so does K_j. Let $H_j = H \setminus \{h_j\}$. Since we are given R_j, we assume that we have a point $o \in K_j$ at our disposal. Let Q_j be the union of R_j, the monotone green portions of $\Lambda \cap h_j$ that terminate within R_j, and the unique monotone green portion of $\Lambda \cap h_j$ that lies above or below R_j, if it exists. In case the latter green component exists, we connect it to R_j by some vertical segment γ that lies on the vertical line ℓ_o that passes through the point o in R_j. We regard γ as being also part of Q_j, so Q_j is a connected region. By computing the intersection points of ℓ_o with the planes in H, we can compute γ in $O(n)$ time. By construction, $K_j = \mathrm{conv}(Q_j)$. See Figure 4 (i).

To construct ζ_j, we adapt the recursive technique of Matoušek [16] for computing the convex hull of a level in the plane, and slightly relax it to simplify its analysis in our new context (at the cost of making the bound on its running time slightly worse). At each recursive step, we have a triangle[2] $\Delta \subseteq h_j$, a subset $G \subseteq H_j$ of $m < n$ planes, and an integer u so that $\Lambda \cap \Delta$ coincides with the u-level of $\mathcal{A}(G)$ inside Δ. Furthermore, we assume that ζ_j intersects Δ and that we have the set Z_Δ of at most six edges of ζ_j that intersect $\partial\Delta$; if $\zeta_j \subset \Delta$, then $Z_\Delta = \emptyset$. The goal of this subproblem is to compute all μ_Δ edges of ζ_j that fully lie in the interior of Δ. Initially, $\Delta = h_j$, $G = H_j$, $u = k$, and $Z_\Delta = \emptyset$. Suppose we have at our disposal a procedure INTERSECTION (e, Δ, G, u) that computes the (at most two) edges of ζ_j that intersect a given segment $e \subset \Delta$. We

[2] In order to accommodate degeneracies, we assume each triangle to be a convex region formed by the intersection of at most three halfplanes.



(i) (ii)

Figure 4. (i) The set Q_j. Segments of Γ_j are drawn as solid lines, R_j is the shaded polygon, intersection lines $h_i \cap h_j$ are drawn as dashed lines, and the thick polygonal chains denote the green portion of $\Lambda \cap h_j$. (ii) The convex hull K_j of Q_j, a triangle Δ, and K^Δ. Thick polygonal chain is ∂K_j, shaded (unbounded) region lying below ∂K_j is K_j, and the darker region is K^Δ.

describe this procedure in Section 2.4. Let

$$K^\Delta = K_j \cap \Delta \qquad \text{and} \qquad Q^\Delta = (Q_j \cup (\bigcup Z_\Delta)) \cap \Delta. \tag{2}$$

The proof of the following simple lemma is straightforward. See Figure 4 (ii).

Lemma 2.9 $K^\Delta = \text{conv}\,(Q^\Delta)$.

We choose a sufficiently large constant r. If $m \le A_2 r \log r$, for some sufficiently large constant A_2 independent of r, we compute K^Δ directly, using Lemma 2.9. Specifically, we first compute the u-level of $\mathcal{A}(G)$ within Δ, which, by assumption, coincides with $\Lambda_\Delta = \Lambda \cap \Delta$ (recall that it can have up to $\Theta(m^2) = O(r^2 \log^2 r)$ complexity), and then compute $C_\Delta = \text{conv}\,((R_j \cup \gamma \cup (\bigcup Z_\Delta)) \cap \Delta)$. To facilitate efficient construction of C_Δ, we store the (edges of the) convex polygon R_j in a height-balanced tree. Then C_Δ can be constructed in $O(\log s)$ time, where s is the complexity of R_j, by inserting the $O(1)$ endpoints of the segments in $Z_\Delta \cup \{\gamma\}$, using an on-line procedure for updating a convex hull by insertions [6]. Finally, we compute, using the same procedure, $\text{conv}\,(\Lambda_\Delta \cup C_\Delta)$ in $O(r^2 \log^2 r \log s) = O(\log s)$ additional time. By traversing the edges of the resulting polygon, and using the fact that this polygon crosses $\partial\Delta$ in at most six points, we can report, in $O(\log s + \mu_\Delta)$ time, all μ_Δ edges of ζ_j that lie inside Δ.

Suppose then that $m > A_2 r \log r$. Let $L = \{h_i \cap h_j \mid h_i \in G\}$ and consider the following range space

$$\mathbb{X} = (L, \{\{\ell \in L \mid \ell \cap \tau \neq \emptyset\} \mid \tau \subset h_j \text{ is a triangle}\}).$$

It is well known that $\mathbb{X}$ has *finite VC-dimension* [9]. We compute, in $O(m)$ time, a $(1/r)$-net $N \subseteq L$ for $\mathbb{X}$ of size $O(r \log r)$, and a triangulation $\mathcal{A}^\nabla(N)$ of the arrangement $\mathcal{A}(N)$ inside Δ [9]. For each edge e of $\mathcal{A}^\nabla(N)$, we compute the one or two edges of ζ_j that cross e, or determine that e

does not cross ζ_j, using the procedure INTERSECTION (e, Δ, G, u). We thus obtain a collection E_Δ of some edges of ζ_j. The edges of $\mathcal{A}^\nabla(N)$ that intersect ζ_j lie in the zone of ζ_j in $\mathcal{A}(N)$, therefore $|E_\Delta|$ is asymptotically bounded by the complexity of the zone, which, since ζ_j is convex, is $O(|N|\alpha(|N|)) \leq A_1 r\alpha(r)\log r$, for an appropriate absolute constant $A_1 > 0$; see, e.g., [7].

Since the segments of E_Δ are in convex position, they can be sorted along ζ_j in $O(r\alpha(r)\log^2 r)$ time. Let η, η' be two consecutive edges in E_Δ. By construction, there exists a triangle τ of $\mathcal{A}^\nabla(N)$ such that each of η, η' has an endpoint inside τ, and the portion of ζ_j between η and η' is fully contained in τ and is delimited by these two endpoints. (The case where the endpoints coincide is trivial, since there is no need to fill in ζ_j between η and η'.) See Figure 5.

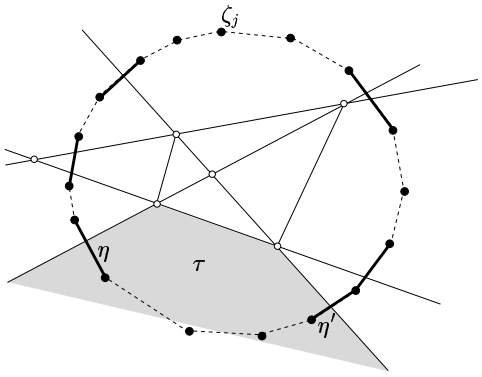


Figure 5. ζ_j (dashed polygon), E_Δ (whose edges are drawn as thick segments), and the triangles of $\mathcal{A}^\nabla(N)$. The portion of ζ_j between η and η' is fully contained in the single cell τ (shaded region).

Let Ξ be the set of triangles $\tau \in \mathcal{A}^\nabla(N)$ for which $\zeta_j \cap \tau$ contains at least one edge of ζ_j in its interior, i.e., there are two consecutive edges of E_Δ that intersect $\partial\tau$ and do not share an endpoint inside τ. We have $|\Xi| \leq A_1 r\alpha(r)\log r$. For each $\tau \in \Xi$, let $G_\tau \subseteq G$ be the set of planes that cross τ (i.e., intersect its interior). Since N is a $(1/r)$-net, $|G_\tau| \leq m/r$. If b_τ planes of G pass below τ, then $\Lambda \cap \tau$ coincides with the $u_\tau = u - b_\tau$ level of $\mathcal{A}(G_\tau)$. We also have at our disposal the set Z_τ of edges of ζ_j that intersect $\partial\tau$, so we can compute K^τ recursively, using the parameters G_τ, u_τ, and Z_τ. We thus obtain the μ_τ edges of ζ_j that lie inside τ, and insert them, in sorted order, between η and η'. Repeating this step for each of the $O(r\alpha(r)\log r)$ cells τ, we obtain K^Δ, as desired.

Suppose the procedure INTERSECTION (e, Δ, G, u) takes at most $Q(m, s)$ time, where $m = |G|$ and $s = |R_j| = O(n)$. Let $T(m, \mu)$ denote an upper bound on the running time of the recursive subproblem within Δ, where $m = |G|$ and μ is the number of vertices of ζ_j lying inside Δ (here s is considered as a global variable, since it does not change during the recursive calls). Then we obtain the following recurrence.

$$T(m,\mu) \leq \begin{cases} \displaystyle\sum_{\tau\in\Xi} T\left(\frac{m}{r}, \mu_\tau\right) + C(Q(m,s)+m), & m \geq A_2 r\log r, \\ O(\log s + \mu) & m < A_2 r\log r, \end{cases}$$

where $\sum_\tau \mu_\tau \leq \mu$, $|\Xi| \leq A_1 r\alpha(r)\log r$, A_1 and A_2 are constants independent of r, and C is a

constant that does depend on r. As Lemma 2.11 in the following subsection will show, $Q(m,s) = O(m\,\mathrm{polylog}(m+s))$. The above recurrence thus solves to

$$T(m,\mu) = O(m^{1+\varepsilon}\,\mathrm{polylog}(m+s) + \mu),$$

for any constant $\varepsilon > 0$. Initially $m, s, \mu = O(n)$, so the overall running time of the algorithm is $O(n^{1+\varepsilon'})$, for any $\varepsilon' > \varepsilon$.

We repeat this step for each of the planes h_j, for $1 < j \leq n$. We then compute $\mathrm{conv}(\bigcup_j K_j)$, a step that takes $O(n^2 \log n)$ time. We thus obtain the main result of this part of the paper:

Theorem 2.10 *Given a set S of n points in $\mathbb{R}^3$ and an integer $k \geq 0$, the region of depth-k of S, $c_k(S)$, can be computed in $O(n^{2+\varepsilon})$ time, for any $\varepsilon > 0$.*

2.4 The intersection subroutine

We now describe the main procedure INTERSECTION (e, Δ, G, u) needed for the preceding algorithm: given a triangle $\Delta \subseteq h_j$, a segment $e \subset \Delta$, a subset $G \subseteq H_j$ of m planes, and an integer u, so that $\Lambda \cap \Delta$ coincides with the level u of $\mathcal{A}(G)$ within Δ, return the edges of ζ_j that intersect e. Recall that, by assumption, ζ_j intersects Δ and we have at our disposal the set Z_Δ of edges of ζ_j that intersect $\partial\Delta$. Let K^Δ, Q^Δ be as defined in (2). Note that only the case $j > 1$ is relevant since, as noted, K_1 can be directly computed in $O(n \log^4 n)$ time [16].

We describe the overall algorithm in three stages. The first stage detects whether a query line ℓ intersects K^Δ. If the answer is positive, it also returns an interval (possibly a single point) lying in $\ell \cap K^\Delta$ that contains the first and the last intersection points of $\ell \cap Q^\Delta$. The second stage determines whether a query point $q \in \Delta$ lies in K^Δ, and computes the lines tangent to K^Δ from q if $q \notin K^\Delta$. This stage computes the tangent lines using the previous procedure and the parametric searching technique [19]. The third stage plugs the tangent-computation procedure into the parametric searching technique, to compute the edges of K^Δ that intersect the query segment $e \subset \Delta$. We now describe each stage in detail.

Intersection detection between K^Δ and a line. Let ℓ be a given line. Since Z_Δ is the set of edges of ζ_j that intersect $\partial\Delta$, $K^\Delta \cap \partial\Delta$ can be computed in $O(1)$ time; see Figure 6. We can therefore compute in $O(1)$ time the intersection points of ℓ with $K^\Delta \cap \partial\triangle$. Next, we compute the intersection points of ℓ with Q^Δ; see Figure 6. We first compute, in $O(\log s)$ time, the intersection points of ℓ with $(R_j \cup (\bigcup Z_\Delta) \cup \gamma) \cap \Delta$. Next, we compute the intersection points of ℓ with $\Lambda \cap \Delta$, as follows. We intersect ℓ with each of the planes of G and sort the intersection points along ℓ. We compute the level of an endpoint (possibly at ∞) of $\ell \cap \Delta$ with respect to $\mathcal{A}(G)$ and scan ℓ in some direction, maintaining a count of the level we are in, and updating the count by ± 1 after crossing each intersection point. If we reach a point on level u, which, by assumption, is a point on $\Lambda \cap h_j$, we conclude that ℓ intersects Λ and we compute the first and the last points of level u (with respect to $\mathcal{A}(G)$) on ℓ within Δ (see, e.g., ℓ_3 in Figure 6). This step takes $O(m \log m)$ time.

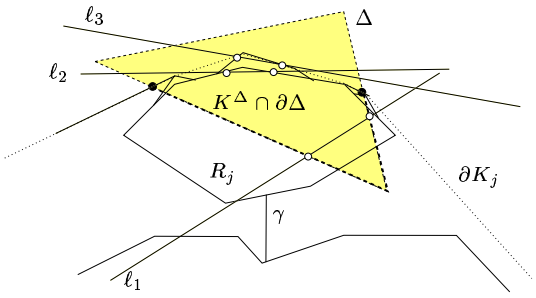


Figure 6. Detecting intersection between K^Δ and a line. Thick dashed edge belong to $K^\Delta \cap \Delta$, and thick solid lines are the edges of Z_Δ. Each of ℓ_1, ℓ_2, and ℓ_3 intersects K^Δ; ℓ_1 intersects $K^\Delta \cap \partial\Delta$, ℓ_2 intersects R_j, and ℓ_3 intersects a green edge of $\Lambda \cap \Delta$; the white circles on each line are the endpoints of the interval returned by the procedure.

If the procedure does not find any intersection point of ℓ that lies in K^Δ, we conclude that ℓ does not intersect K^Δ. If ℓ intersects K^Δ, then we return the first and the last intersection points computed by the algorithm that lie in K^Δ; (these are the hollow circles on ℓ_1, ℓ_2, ℓ_3 in Figure 6). The total time spent by the procedure is $O(m \log m + \log s)$.

Remarks. (1) The interval returned by the procedure may be in general a *proper* subset of $\ell \cap K^\Delta$ (see, e.g., line ℓ_2 in Figure 6).

(2) We can also use this procedure to detect an intersection between a ray ξ and Q^Δ: We apply the procedure to the line ℓ containing ξ, and check whether ξ intersects the interval returned by the procedure.

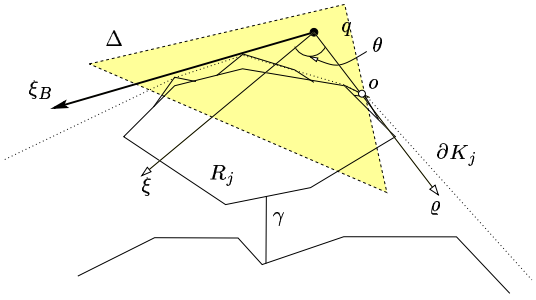


Figure 7. Tangents to K^Δ from a point q.

Computing the tangents to K^Δ from a point. Let q be a point in Δ. We wish to determine whether $q \in K^\Delta$, and if the answer is negative, we also want to compute the two tangents from q to K^Δ. We note that if Δ is unbounded then one or both of these tangents may not be real tangents, but are rather rays parallel to the unbounded rays of ∂K^Δ. We choose a point o in K^Δ. If $Z_\Delta = \emptyset$, i.e., $K_j \subseteq \Delta$, we choose o to be an arbitrary point of R_j. Otherwise, we choose o to be an intersection point of $\partial\Delta$ and an edge of Z_Δ. Let ϱ denote the line passing through q and o and oriented from

q towards o. We parametrize rays ξ emanating from q by a real number $\sigma(\xi) = \tan(\theta/2)$, where $\theta \in [0, 2\pi]$ is the angle from ϱ to ξ in the counterclockwise direction; see Figure 7.

We first run the above line-intersection procedure on ϱ. If the interval returned by the procedure contains q, we conclude that $q \in K^\Delta$ and stop. Otherwise, the ray emanating from q in direction $-\vec{qo}$ does not intersect Q^Δ. However, q may still lie in K^Δ (see the preceding Remark (1)). The definition of Q_j and its property proved in Corollary 2.8 imply that there exists an interval $I_A = [0, \sigma_A] \subset [0, \infty]$ such that a ray ξ with $\sigma(\xi) \in [0, \infty]$ intersects Q^Δ if and only if $\sigma(\xi) \in I_A$; the ray ξ_A with $\sigma(\xi_A) = \sigma_A$ is tangent to Q_A. Similarly, there exist the interval $I_B = [\sigma_B, 0] \subset [-\infty, 0]$ and the ray ξ_B, with $\sigma(\xi_B) = \sigma_B$, tangent to Q^Δ. The angular span of the wedge bounded by ξ_A and ξ_B that contains Q^Δ is less then π if and only if $q \notin K^\Delta$. Hence, it suffices to compute ξ_A, ξ_B. We describe the computation of ξ_A.

Using the parametric searching technique [19], we simulate the above line-intersection-detection procedure generically at ξ_A and maintain an interval $I = [a, b] \subseteq [0, \infty]$, such that $\sigma_A \in I$. During the simulation, we encounter comparison steps, each of which compares the unknown σ_A with some specific $\tilde{\sigma} \in I$. To resolve the comparison, we test whether the ray $\tilde{\xi}$ emanating from q, with $\sigma(\tilde{\xi}) = \tilde{\sigma}$, intersects Q^Δ (recall Remark (2) above). If the answer is positive, then $\sigma_A \geq \tilde{\sigma}$ and we set $I := [\tilde{\sigma}, b]$; otherwise $\sigma_A \leq \tilde{\sigma}$, and we set $I := [a, \tilde{\sigma}]$. Using the preceding algorithm, we can detect this intersection in $O(m \log m + \log s)$ time. If the procedure returns an interval that contains q, we conclude that $q \in K^\Delta$ and stop. If I becomes empty, we conclude that $q \in K^\Delta$, and if I becomes a singleton $[a, a]$, then $\sigma_A = a$. In both cases, the procedure terminates, and returns the indication that $q \in K^\Delta$ in the former case, or the tangent ξ_A in the latter case. Finally, if the simulation stops with an interval $[a, b]$, using the standard parametric-searching argument one can argue that $\sigma_A = a$.

To obtain an efficient implementation of the parametric search, we note that the intersection detection procedure involves three main steps: Sorting the intersections of ℓ with the planes of G, finding the intersections of ℓ with the convex polygon R_j, and testing whether ℓ intersects γ or an edge of Z_Δ. The third step involves only $O(1)$ intersection operations, and is thus trivial to simulate. The second step uses $O(\log s)$ comparisons, so its simulation takes $O(m \log^2(m + s))$ time. The first step, if implemented by a logarithmic-depth parallel sorting network [2], and enhanced with Cole's improvement [11], can also be implemented in $O(m \log^2(m + s))$ time; see [16] for details.

The intersection of ξ_A with K^Δ is a single point, an edge of K^Δ, or empty, where the third situation arises when ξ_A is parallel to an unbounded edge of K^Δ. We can easily compute this intersection, by noting that the endpoints of $\xi_A \cap K^\Delta$ must belong to Q^Δ, and we know how to compute such an intersection in time $O(m \log m + \log s)$ using the intersection-detection procedure described above.

Finally, if $q \notin K^\Delta$, we repeat the same procedure to compute ξ_B. The total time spent in computing ξ_A, ξ_B, and their intersection points with K^Δ is $O(m \log^2(m + s))$.

Computing the edges of K^Δ crossed by an edge. Armed with the tangent computation procedure, we next derive the main procedure by applying parametric searching once again. It suffices to

solve the simpler problem, where we are given a query line ℓ, and our goal is to compute the one or two edges of K^Δ that are crossed by ℓ, or to determine that $\ell \cap K^\Delta = \emptyset$. We then apply this procedure to the line containing the query edge e, and trivially retrieve the zero, one, or two edges of K^Δ that e crosses. The latter task of determining whether $\ell \cap K^\Delta = \emptyset$ can be accomplished using the basic intersection testing procedure, so we may assume that $\ell \cap K^\Delta \neq \emptyset$, and that we have computed a point $q_0 \in \ell \cap K^\Delta$. Using parametric searching once again, we slide a point q from q_0 along each of the two rays of ℓ delimited by q_0, and test whether q lies in K^Δ, using the tangent computation procedure described above. This guides our search: If q lies in K^Δ, we proceed by moving further away from q_0, and otherwise we proceed by moving towards q_0. When we home in on the actual point q of intersection between ℓ and ∂K^Δ, the tangent computation procedure yields the desired edge of K^Δ that ℓ crosses at q. We omit the routine details of the parametric searching, do not aim at the most efficient implementation, and are satisfied with the following result, which is the promised missing ingredient for the proof of Theorem 2.10.

Lemma 2.11 *Given a triangle $\Delta \subset h_j$, the set Z_j of edges of ζ_j that intersect $\partial\Delta$, a segment $e \subset \Delta$, the subset $G \subseteq H_j$ of the m planes that cross Δ, and an integer $u < m$, such that the level u of $\mathcal{A}(G)$ coincides with Λ within Δ, the procedure* INTERSECTION *(e, Δ, G, u) takes $O(m \operatorname{polylog}(m+s))$ time.*

3 Recognizing Colored Tverberg Points in the Plane

We now change gears, descend to the plane, and study the problem of recognizing colored Tverberg points. Let S be a 3-colored set, which is the disjoint union of a set R of n red points, a set B of n blue points, and a set G of n green points. We assume that the points of S are in general position.

Let q be a given point that we want to test for being a colored Tverberg point of S. Let C be the unit circle centered at q. We may assume, without loss of generality, that all the points of S lie on C; otherwise we project these points on C, centrally from q, and note that q is a colored Tverberg point of the original set if and only if it is a colored Tverberg point of the projected set. If q is generic, all projected points are distinct. Otherwise, since S is in general position, at most two pairs of points of S may project to coinciding points on C. This will require easy and straightforward modifications of the following procedure, which we omit, and assume that all projected points are distinct. Similarly, we will also assume that no two projected points are diametrically opposite on C (at most two projected pairs can consist of diametrically opposite points).

Let C_0 be a fixed semicircle of C, whose endpoints are disjoint from S. For each point u in the (open) complementary semicircle C_1, let $\bar{u}$ denote the antipodal point of u in C_0. Put $R^+ = R \cap C_0$ and $R^- = \{\bar{u} \mid u \in R \cap C_1\}$, and define similarly the sets B^+, B^-, G^+ and G^-. Sort the points of $R^+ \cup R^- \cup B^+ \cup B^- \cup G^+ \cup G^-$ in counterclockwise order along C_0, and denote the resulting (linear) sequence by E. (By our assumptions, all elements of E are distinct.) Note that a *rainbow* triangle, namely, a triangle uvw, with $u \in R$, $v \in B$, $w \in G$, contains q if and only if E contains

one of the ordered triples

$$(u, \bar{v}, w), (w, \bar{v}, u), (v, \bar{w}, u), (u, \bar{w}, v), (w, \bar{u}, v), (v, \bar{u}, w),$$
$$(\bar{u}, v, \bar{w}), (\bar{w}, v, \bar{u}), (\bar{v}, w, \bar{u}), (\bar{u}, w, \bar{v}), (\bar{w}, u, \bar{v}), (\bar{v}, u, \bar{w}),$$

as a (not necessarily contiguous) subsequence; the specific triple is determined by the locations of u, v, w along C. See Figure 8. Our goal is thus to determine whether E can be decomposed into n pairwise disjoint triples of these 12 kinds.

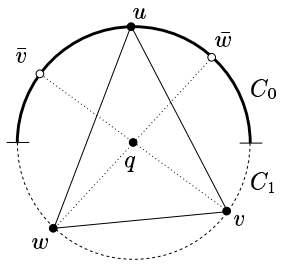


Figure 8. A triangle uvw that contains q, and the corresponding triple $(\bar{w}, u, \bar{v})$.

We first describe a less efficient (but still polynomial) algorithm for solving this problem, which is conceptually simpler, and then improve its running time. Write E as $(e_1, e_2, \ldots, e_{3n})$, and denote by E_j its prefix $(e_1, \ldots, e_j)$, for $j = 0, 1, \ldots, 3n$. The algorithm uses dynamic programming and processes the elements of E in increasing order. At the beginning of the processing of e_i, it maintains a set X_{i-1} of *configurations*, each of which is an 18-tuple of integers, which represent a partitions of $E_{i-1} = (e_1, \ldots, e_{i-1})$ into triples and prefixes of triples of the above 12 kinds. More specifically, the first six components of a configuration ξ, which we denote by $N_\xi(R^+)$, $N_\xi(R^-)$, $N_\xi(B^+)$, $N_\xi(B^-)$, $N_\xi(G^+)$, $N_\xi(G^-)$, record the number of singleton prefixes of triples that lie in E_{i-1}, where $N_\xi(R^+)$ is the number of such singleton prefixes in R^+, and similarly for the other five quantities. (Counting an element of E_{i-1} as a singleton prefix means that we expect it, in the configuration under consideration, to form a valid triple with two other elements that lie further ahead of E_{i-1}.) The next 12 components are each indexed by a pair of a positive color set and a negative color set, where the two colors are distinct, and record potential doubleton prefixes of triples in E_{i-1} (where the third element of such a triple is expected to come from the remainder of E). For example, $N_\xi(R^+G^-)$ is the number of doubleton prefixes (u, v) in the configuration, where $u \in R^+$, $v \in G^-$, and u precedes v in E_{i-1}. (To complete this pair into a valid triple, an element of B^+ will have to be chosen from the remainder of E.) The other 11 components are defined in complete analogy. We emphasize that, in each configuration $\xi \in X_{i-1}$, the counts in its components represent a *partition* of E_{i-1}. That is, each element of E_{i-1} contributes either to *one* of the components of ξ or it may drop out of the game altogether, if it is assigned to a triple that has already been completed within E_{i-1}. See below for details.

Initially, X_0 consists of only the all-zero tuple $\mathbf{0}$. When processing e_i, we produce the set X_i of configurations for E_i from X_{i-1} as follows. Suppose that $e_i \in R^+$, and let ξ be a configuration in

X_{i-1}. We generate from ξ (at most) five configurations in X_i, according to the following choices of the role of e_i:

(i) e_i is the first component of a new triple: We generate a new configuration by increasing $N_\xi(R^+)$ by 1.

(ii) e_i is the second component of a triple whose first component is in B^-: We generate a new configuration by increasing $N_\xi(B^-R^+)$ by 1, and by decreasing $N_\xi(B^-)$ by 1.

(iii) e_i is the second component of a triple whose first component is in G^-: We generate a new configuration by increasing $N_\xi(G^-R^+)$ by 1, and by decreasing $N_\xi(G^-)$ by 1.

(iv) e_i is the third component of a triple whose first component is in B^+ and whose second component is in G^-: We generate a new configuration by decreasing $N_\xi(B^+G^-)$ by 1.

(v) e_i is the third component of a triple whose first component is in G^+ and whose second component is in B^-: We generate a new configuration by decreasing $N_\xi(G^+B^-)$ by 1.

The cases where e_i belongs to any of the other five signed color classes is handled in a completely symmetric manner. We discard a new configuration if it already exists in X_i or if any of its components is negative. Then the set X_i stores configurations without repetition. Whenever a new configuration ξ is inserted into X_i (for the first time), we store with it a pointer to the configuration $\xi' \in X_{i-1}$ from which ξ is generated. More precisely, we simply store with ξ the type of incremental change, i.e., an up to three-letter string from the alphabet $\mathcal{C} = \{R^+, B^+, G^+, R^-, B^-, G^-\}$, that has produced it from ξ', using which ξ' can easily be reconstructed. For example, when $e_i \in R^+$, we store B^-R^+ at ξ in case (ii), and $B^+G^-R^+$ in case (iv). In general, there may be several configurations $\xi' \in X_{i-1}$ that can induce $\xi \in X_i$, but since we only keep the first copy of ξ we store a pointer to only the first ξ' that has generated ξ.

The number of configurations in any X_i is $O(n^{18})$, so the processing of each e_i takes $O(n^{18})$ time, using an appropriate data structure to store X_i. The total running time is thus $O(n^{19})$. After processing e_{3n}, we test whether X_{3n} contains the all-zero tuple $\mathbf{0}$. If it does not, q is not a Tverberg point. If $\mathbf{0}$ is in X_{3n} then, using the additional data stored with each configuration, we compute a Tverberg partition by tracing back a sequence of configurations $\mathbf{0} = \xi_0, \xi_1, \xi_2, \ldots, \xi_{3n-1}, \xi_{3n} = \mathbf{0}$, so that each ξ_j belongs to X_j and can be generated from ξ_{j-1} when processing e_j.

More precisely, we maintain with each configuration ξ_i that we trace, a family $\mathcal{T} = \mathcal{T}_i$ of triples. Each triple $\tau \in \mathcal{T}$ has a suffix formed by the points in E, and the remaining prefix of τ is formed by the letters in $\mathcal{C}$; the prefix indicates which types of points are needed to complete the triple. See Figure 9. For example, if a triple in $\mathcal{T}_\ell$ is (R^+, B^-, e_k), then $k > \ell$, and we need a point $e_i \in R^+$ and another $e_j \in B^-$ to form the triple (e_i, e_j, e_k) of the Tverberg partition that we are constructing, where $i < j \leq \ell$. Similarly, if the triple is (G^-, e_j, e_k), then $\ell < j < k$, and we need a point $e_i \in G^-$, with $i \leq \ell$, to form a triple (e_i, e_j, e_k). Initially, $\mathcal{T} = \mathcal{T}_{3n}$ is empty. Suppose we are currently processing ξ_i, which was constructed while adding e_i. If we stored at ξ_i a three-letter string $\chi_1\chi_2\chi_3$ to mark the change from ξ_{i-1}, with $\chi_1, \chi_2, \chi_3 \in \mathcal{C}$, then χ_3 must label the set

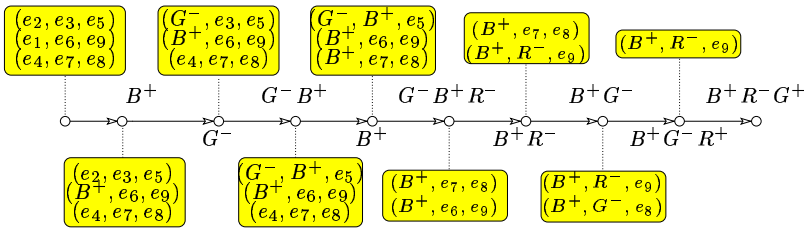


Figure 9. Constructing a Tverberg partition from a sequence of configurations, by tracing the sequence backwards. For each node in the path, the label shows the string stored at that configuration, and the sequence of triples inside the box is the family $\mathcal{T}$ just before processing the node in the backward-tracing phase. The output Tverberg partition is the one stored at the leftmost node.

containing e_i, and we add the triple (χ_1, χ_2, e_i) to $\mathcal{T}$. For example, we add the triple (B^+, R^-, e_9) to $\mathcal{T}$ after processing ξ_9 in Figure 9 (and $e_9 \in G^+$ in this case). On the other hand, if we stored a two-letter string $\chi_1\chi_2$ at ξ_i, then χ_2 must label the set containing e_i, and we extract a triple of the form (χ_1, χ_2, e_k) from $\mathcal{T}$ and replace it with (χ_1, e_i, e_k). Similarly, if the string stored at ξ_i is χ_1, we extract a triple of the form (χ_1, e_j, e_k) from $\mathcal{T}$ and replace it with (e_i, e_j, e_k). After processing ξ_1, the final value of $\mathcal{T}$ is (one possible) Tverberg partition for q. The correctness of the trace-back algorithm follows from the invariant, easily established using induction on i, that if a configuration ξ belongs to X_i then E_i admits a decomposition into pairwise disjoint prefixes of triples (and complete triples), so that the number of prefixes of each type is equal to the corresponding component of ξ.

In summary, we have shown that determining whether a given point q is a colored Tverberg point of S can be done in $O(n^{19})$ time.

We next proceed to optimize the algorithm. In the revised version we apply the same general approach, but maintain configurations with fewer components. First we note that there is no need to maintain the two separate quantities $N_\xi(R^+B^-)$, $N_\xi(B^+R^-)$, and it suffices just to maintain their sum. Indeed, both quantities are accessed only when a further element of E that belongs to G^+ "decides" to become the last element of a triple, in which case it has to be matched with a doubleton that is counted in one of these two quantities, but it does not matter which of the two kinds of doubletons is being used. A symmetric argument applies to other pairs of doubleton components. Hence, a configuration needs to record only 12 counts, six singleton counts, as above, and the six doubleton counts

$$\begin{aligned}
N_\xi(R^+B^-) &+ N_\xi(B^+R^-),\\
N_\xi(R^-B^+) &+ N_\xi(B^-R^+),\\
N_\xi(R^+G^-) &+ N_\xi(G^+R^-),\\
N_\xi(R^-G^+) &+ N_\xi(G^-R^+),\\
N_\xi(B^+G^-) &+ N_\xi(G^+B^-),\\
N_\xi(B^-G^+) &+ N_\xi(G^-B^+)\,.
\end{aligned}$$

This already yields an algorithm that runs in $O(n^{13})$ time (there are $O(n^{12})$ different configurations,

and there are n iteration steps). The path-tracing procedure can be adapted in a straightforward manner to construct a Tverberg partition.

We can further reduce the number of components in a configuration to 10, as follows. Suppose that $\xi \in X_j$. Denote by $M_\xi(R)$ the sum of all components of ξ that record prefixes of tuples that involve an element of R (i.e., of $R^+ \cup R^-$), and define similarly $M_\xi(B), M_\xi(G)$. Let $K_j(R), K_j(B), K_j(G)$ denote the number of elements of E_j that are red, blue, and green, respectively. Let t denote the number of complete triples (contained in E_j) that have been generated by the incremental construction recorded in ξ (or, more precisely, in the unique sequence of configurations in $X_1, X_2, \ldots, X_j$ that terminates at ξ and whose reverse is obtained by following the stored back pointers, starting from ξ). Then we have

$$\begin{aligned} K_j(R) &= t + M_\xi(R) \\ K_j(B) &= t + M_\xi(B) \\ K_j(G) &= t + M_\xi(G)\ . \end{aligned}$$

That is,

$$\begin{aligned} M_\xi(B) - M_\xi(R) &= K_j(B) - K_j(R) \\ M_\xi(G) - M_\xi(R) &= K_j(G) - K_j(R)\ . \end{aligned}$$

This gives us two independent linear relations among the 12 components of a configuration, showing that it suffices to store and maintain only 10 of them. The number of tuples generated by the algorithm is thus $O(n^{10})$, and the total running time is $O(n^{11})$. That is, we have:

Theorem 3.1 *Let S be a set of $3n$ points in the plane, n of which are red, n blue, and n green. For a given point q, we can determine whether q is a colored Tverberg point of S in time $O(n^{11})$.*

4 Open Problems

The paper raises several open problems for further research. We mention only a few:

(i) Obtain a tight bound for the maximum possible complexity of the center region $c(S)$ of a set S of n points in general position in $\mathbb{R}^d$, $d > 3$. Then provide an efficient algorithm for constructing $c(S)$.

(ii) Can the center region in $\mathbb{R}^3$ be constructed in $O(n^2 \mathrm{polylog}(n))$ time?

(iii) Can colored Tverberg points in the plane be recognized in a more efficient manner? What about colored Tverberg points in higher dimensions?

Acknowledgments. The authors thank Jirka Matoušek and Boris Aronov for useful discussions concerning this problem, and the anonymous referees for many helpful comments for improvements and corrections. The authors also thank the GWOP'03 working group [12] for providing the construction in Figure 2.

References

[1] P. K.Agarwal and J. Matoušek, Dynamic half-space range reporting and its applications, *Algorithmica* 13 (1995), 325–345.

[2] M. Ajtai, J. Komlós, and E. Szemerédi, Sorting in $c \log n$ parallel steps, *Combinatorica* 3 (1983), 1–19.

[3] D. Avis, The m-core property contains the m-divisible points in space, *Pattern Recognition Letters* 14 (1993), 703–705.

[4] I. Bárány and D. Larman, A colored version of Tverberg's theorem, *J. London Math. Soc., Ser. II*, 45 (1992), 314–320.

[5] I. Bárány, Z. Füredi and L. Lovász, On the number of halving planes, *Combinatorica* 10 (1990), 175–183.

[6] M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 2nd Edition, Springer Verlag, Heidelberg, 2000.

[7] M. Bern, D. Eppstein, P. Plassman, and F. Yao, Horizon theorems for lines and polygons, in *Discrete and Computational Geometry: Papers from the DIMACS Special Year* (J. Goodman and R. Pollack, and W. Steiger, eds.), American Mathematical Society, Providence, RI, 1991, 45–66.

[8] T. Chan, An optimal randomized algorithm for maximum Tukey depth, *Proc. 15th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2004, 423–429.

[9] B. Chazelle, *The Discrepancy Method*, Cambridge University Press, 2000.

[10] K. Clarkson, D. Eppstein, G. L. Miller, C. Sturtivant, and S.-H. Teng, Approximating center points with iterative Radon points, *Intl. J. Comput. Geom. Appls.* 6 (1996), 357–377.

[11] R. Cole, Slowing down sorting networks to obtain faster sorting algorithms, *J. ACM* 34 (1987), 200–208.

[12] P. Csorba, K. Fischer, M. John, Y. Okamoto, J. Solymosi, M. Stojanković, Cs.D. Tóth, and U. Wagner, A nonconvex colored-Tverberg region, Working Group at the 1st GWOP'03 workshop, Switzerland, 2003.

[13] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer Verlag, Heidelberg, 1987.

[14] S. Jadhav and A. Mukhopadhyay, Computing a centerpoint of a finite planar set of points in linear time, *Discrete Comput. Geom.* 12 (1994), 291–312.

[15] G. Kalai, Combinatorics with a geometric flavor: Some examples, in *Visions in Mathematics Toward 2000 (Geometric and Functional Analysis, Special Volume)*, 742–792.

[16] J. Matoušek, Computing the center of a planar point set, in *Discrete and Computational Geometry: Papers from the DIMACS Special Year* (J. Goodman, R. Pollack, and W. Steiger, eds.), American Mathematical Society, Providence, RI, 1991, 221–230.

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

[18] J. Matoušek, *Lectures in Discrete Geometry*, Springer Verlag, Heidelberg, 2002.

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

[20] K. Miller, S. Ramaswami, P. Rousseeuw, T. Sellarés, D. Souvaine, I. Streinu and A. Struyf, Efficient computation of location depth contours by methods of computational geometry, *Statistics and Computing* 13 (2003), 153–162.

[21] N. Naor and M. Sharir, Computing a point in the center of a point set in three dimensions, *Proc. 2nd Canadian Conf. Comput. Geom.*, 1990, 10–13.

[22] R. Rado, A theorem on general measure, *J. London Math. Soc.* 21 (1947), 291–300.

[23] M. Sharir and P. K. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications*, Cambridge University Press, New York, 1995.

[24] M. Sharir, S. Smorodinsky and G. Tardos, An improved bound for k-sets in three dimensions, *Discrete Comput. Geom.* 26 (2001), 195–204.

[25] S.-H. Teng, *Points, Spheres, and Separators: A Unified Geometric Approach to Graph Partitioning,* Ph.D. Thesis, School of Computer Science, Carnegie Mellon University, PA, 1992.

[26] H. Tverberg, A generalization of Radon's theorem, *J. London Math. Soc.* 41 (1966), 123–128.

[27] G.M. Ziegler, *Lectures on Polytopes*, Springer Verlag, New York, 1995.

[28] R.T. Živaljević and S.T. Vrećica, The colored Tverberg's problem and complexes of injective functions, *J. Combin. Theory, Ser. A,* 6 (1992), 309–318.