INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center Street • Suite 600 • Berkeley, California 94704 • 1-510-642-4274 • FAX 1-510-643-7684

On Lines Missing Polyhedral Sets in 3-Space

Marco Pellegrini*
Dept. of Computer Science, King's College London

TR-93-034

July 1993

Abstract

We show some combinatorial and algorithmic results concerning sets of lines and polyhedral objects in 3-space. Our main results include:

(1) An $O(n^3 2^{c\sqrt{\log n}})$ upper bound on the worst case complexity of the set of lines missing a star-shaped compact polyhedron with n edges, where c is a suitable constant.

(2) An $O(n^3 2^{c\sqrt{\log n}})$ upper bound on the worst case complexity of the set of lines that can be moved to infinity without intersecting a set of n given lines, where c is a suitable constant. This bound is almost tight.

(3) An $O(n^{1.5+\epsilon})$ randomized expected time algorithm that tests whether a direction v exists along which a set of n red lines can be translated away from a set of n blue lines without collisions.

(4) Computing the intersection of two polyhedral terrains in 3-space with n total edges in time $O(n^{4/3+\epsilon} + k^{1/3}n^{1+\epsilon} + k\log^2 n)$, where k is the size of the output, and $\epsilon > 0$ an arbitrary small but fixed constant. This algorithm improves on the best previous result of Chazelle at al. [8].

The tools used to obtain these results include Plücker coordinates of lines, random sampling and polarity transformations in 3-space.

A preliminary version of this work appeared in the *Proceedings of the 9th ACM Symposium on Computational Geometry* [30].

*Department of Computer Science, King's College, Strand, London WC2R 2LS U.K. marco@dcs.kcl.ac.uk

1 Introduction

It is a common pattern in computational geometry that the combinatorial structure of a problem is the main ingredient for the design of an efficient algorithm. Thus it is important to derive good combinatorial bounds for the quantities of interest in 3D problems. Still results in 3-dimensional computational geometry are much fewer than those for similar problems cast in 2-dimensional space. One of the reasons for this state of affairs is that the combinatorial properties of of lines and other 1-dimensional objects in 3-space are not well understood. Most combinatorial results for polyhedral sets in 3-space describe bounds on sets of *points* (or planes via duality) in 3-space [14, 33]. Recently, Aronov and Sharir [5, 6] could find non-trivial bounds on the complexity of one cell in an arrangement of triangles in 3-space. This is equivalent to determining the worst case complexity of one *isotopy class of points* induced by a set of triangles in 3-space[1].

In a seminal paper of 1989 [8] Chazelle, Edelsbrunner, Guibas and Sharir use in an original way a range of techniques (Plücker coordinates of lines, geometric random sampling, segment trees) and obtain new bounds on the complexity of certain configurations of lines in 3-space as well as efficient algorithms for solving problems on polyhedral terrains. This first paper has been followed by others (e.g. [27] [28] [13] [3] [31] [10] [29]) in which the original ideas have been improved or applied to a range of new problems in 3-space. This paper is in the same line of research and it aims at improving bounds on the combinatorial structure of sets of lines. We then use one of the bounds to design an efficient algorithm for a translation problem in 3-space.

1.1 Complexity of sets of lines

It is well known that a set of polyhedral objects in 3-space induces a decomposition of the set of spatial lines $\mathcal{L}$ into isotopy classes. Two lines being in the same isotopy class if we can move continuously one into the other without crossing any polyhedral object.

Let $\mathcal{Q}$ be a class of polyhedral objects and Q an element of $\mathcal{Q}$. With $n = size(Q)$ we denote the number of edges of the polyhedral objects in Q. We investigate worst case upper bounds on the combinatorial complexity of the following classes of lines: (i) $M(Q)$ the set of lines in R^3 *missing* all elements in Q, (ii) $I(Q)$ a connected component of $M(Q)$ (i.e. an isotopy class of missing lines), and (iii) $F(Q)$ the set of free lines in R^3 with respect to Q (i.e. the set of lines in $M(Q)$ that can be translated in some direction to infinity without collisions with elements of Q). These classes arise naturally in 3-dimensional visibility and translation problems

In general, given a set of polyhedral objects Q with n edges we can extend each edge into a full line and obtain a set of n lines L_Q. The number of isotopy classes induced by a set L_Q of lines in general position is $\Theta(n^4)$ and the total descriptive complexity of these classes is $\Theta(n^4)$. Sets $M(Q), I(Q)$ and $F(Q)$ can be expressed as the union of some isotopy classes (or parts thereof) induced by Q on $\mathcal{L}$. We are interested classes of polyhedral objects for which $M(Q), I(Q), F(Q)$ have worst case upper bounds that are substantially smaller then $O(n^4)$. An example in this direction is the $O(n^3\beta(n))$ bound on the complexity of the

[1] The results of Aronov and Sharir are more general and hold in d-space.

set of lines stabbing a set of convex polyhedra, where $\beta(n) = 2^{c\sqrt{\log n}}$ is a subpolynomial function [31].

For any of the above mentioned sets of lines its *complexity* can be expressed as the number of *extremal lines in the set*. A line is extremal when it is incident to four edges of polyhedra in Q, or to two vertices, or to one vertex and two edges and moreover it is tangent to the polyhedra containing those edges and vertices. With an abuse of notation we will refer to a set of lines and to its complexity using the same symbol whenever the distinction is clear from the context. Some more care is needed in definitions when the polyhedral set is composed of 1-dimensional objects.

For a compact star-shaped polyhedron P (note that a terrain is a particular star-shaped polyhedron) we show in Section 2 that $M(P) = I(P) = F(P) = O(n^3\beta(n))$. This result is proved by establishing an interesting duality relation (through a polarity transformation) between stabbing and missing lines in 3-space. A similar result is obtained in [17] using different techniques. The technique in [17] can be extended to prove a similar bound on the set of *rays* missing a terrain. An $\Omega(n^2)$ lower bound for $M(P)$ is easy to construct and a slightly more complex $\Omega(n^2\alpha(n))$ bound is mentioned in [17].

For a set of lines L: $I(L) = \Theta(n^2)$ and $M(L) = \Theta(n^4)$ [8]. In [8] a cubic upper bound is obtained for the lines *vertically above* L, which matches a cubic worst case lower bound. The set $F(L)$ is a superset of the set of lines vertically above L, since no fixed direction is involved in the definition of a free line. In Section 3 we prove that $F(L) = O(n^3\beta(n))$ which almost matches the $\Omega(n^3)$ lower bound in [8].

1.2 Collision free translations of polyhedral objects

Moving objects without collisions is an important problem in robotics and CAD/CAM in relation to assembly of objects composed of polyhedral parts [23]. General versions of the assembly problems for polyhedra in 3-space can be PSPACE-hard [23]. In this paper we discuss some restricted types of assembly problems for which we can design efficient algorithms.

The problem whether for a class of polyhedral objects a separation direction always exists is discussed in [37]. Efficient assembly algorithms have been found in the case when the direction of movement is given for all the objects [24, 16, 11] (see [37] for a survey). In the planar case Pollack et al. give algorithms for separating two simple polygons by a sequence of translations [32]. If we allow more complicated types of movement the assembly problem merges with the much studied motion planning problems (see [34] for a survey). We consider an intermediate case where the direction of movement is part of the answer to the problem, and it ranges over the whole sphere of directions, but we allow a single simultaneous translation.

Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be two classes of polyhedral objects and Q_1 (resp. Q_2) an element from $\mathcal{Q}_1$ (resp. $\mathcal{Q}_2$). With $n = \max\{size(Q_1), size(Q_2)\}$ we denote the the maximum number of edges in either set. We define $V(Q_1, Q_2)$ as the set of directions in 3-space along which there is a collision-free translation of Q_1 with respect to Q_2. A point of $V(Q_1, Q_2)$ is called a *feasible direction*. We are interested in worst case combinatorial bounds on the complexity of the set $V(Q_1, Q_2)$ and in the time needed to find one feasible direction.

When P_1 and P_2 are two simple polyhedra, $V(P_1, P_2) = \Theta(n^4)$ in the worst case [26] and

the set $V(P_1, P_2)$ can be computed in time $O(n^4 \log n)$. When C_1 and C_2 are two convex polyhedra $V(C_1, C_2) = \Theta(n)$ and $V(C_1, C_2)$ is computed in time $O(n)$ [25].

In this paper we consider the case $V(L_1, L_2)$ where L_1 and L_2 are sets of lines in 3-space. We show that $V(L_1, L_2) = \Theta(n)$. One feasible direction can be found easily in time $O(n^2)$ by adapting a technique in [25]. We give in Section 4 a non-trivial algorithm that finds a feasible direction in time $O(n^{1.5+\epsilon})$, for every $\epsilon > 0$. The upper bound on the running time of this algorithm depends on the combinatorial bound for $F(L)$.

1.3 Intersection of polyhedral terrains

Computing the intersection of polyhedra in 3-space is a basic problem in computational geometry [33, 14]. The intersection of two convex polyhedra can be compute $O(n)$ time which is optimal [7]. The intersection of two simple non-convex polyhedra can be computed in time $O(n^{8/5+\epsilon} + k \log n)$, where k is the output size [28]. For the case of polyhedral terrains (which are graphs of piecewise-linear continuous bivariate functions) Chazelle et al. [8] obtain an intersection algorithm with time bound $O(n^{3/2+\epsilon} + k \log^2 n)$.

Ideally, when computing the intersection of two objects the overhead term (i.e. the term not depending on the output size) should match the time bound of the best method to test whether two objects have empty intersection. The algorithms for convex polyhedra and simple polyhedra satisfy this ideal condition. The result for terrains in [8] does not, since the best known algorithm for testing the intersection of two terrains has complexity $O(n^{4/3+\epsilon})$ [8]. In Section 5 we combine the approach of Chazelle et al. with recent results on halfspace range queries [20] obtaining a total running time $O(n^{4/3+\epsilon} + k^{1/3+\epsilon} n^{1+\epsilon} + k \log^2 n)$. This algorithm matches the best known intersection-testing algorithm for $k < n$, and is almost optimal for $k > n^{3/2}$ (within a polylogarithmic factor). We refer to the algorithm in [8] as the CEGS-Algorithm. For the range $0 < k < n^{3/2}$ our algorithm has better asymptotic performance than the CEGS-Algorithm and for $k > n^{3/2}$ it is as fast.

As follows from the discussion in [9] and [8], the running time is dominated by the time needed to find all intersections of an edge from one terrain with a face on the other terrain. The main new idea is to detect efficiently those edges that do not contribute to the intersection. Time spent on additional computation is charged partially to the output size, thus reducing the fixed overhead term.

The paper is organized as follows: in Section 2 we discuss the relation between stabbing lines and missing lines and we derive a bound for the set of lines missing a star-shaped polyhedron. In Section 3 we derive a bound on the set of free lines induced by lines and in Section 4 we discuss the translation problem for two sets of lines in 3-space. In Section 5 we present the algorithm for intersecting two polyhedral terrains.

2 Missing lines, stabbing lines and polarity

Let us consider a well-known duality transformation between points and planes in 3-space, namely the *polarity* δ [33, 7] which maps a point $p = (a, b, c)$ distinct from the origin O into the plane $\delta(p)$ of equation $ax + by + cz = 1$. Plane $\delta(p)$ is the plane normal to the line Op and at distance $1/|\bar{O}p|$ from O, on the same side as p. Given a convex compact polytope P containing O in its interior, we define the set $\delta(P) = \{\delta(p) | p \in P\}$ of planes dual to points

in P. Considering $\delta(P)$ as a set of points in R^3, we define as the dual polytope of P the set $P^\delta = R^3/\delta(P)$. It is easy to show that P^δ is a convex compact polytope containing O in its interior. If no two facets of P are coplanar then there is a one-to-one correspondence between k-faces of P and $(2-k)$-faces of P^δ. The polarity transformation is convolutory (i.e. $(P^\delta)^\delta = P$) [33]. Given a line l as a locus of points we obtain in the dual space a locus of planes which is a 1-dimensional pencil of planes. The line l^δ dual to l is the axis of the pencil.

Lemma 1 *Given a convex compact polytope P containing the origin, a line l misses P if and only if l^δ is a stabbing line for P^δ.*

Proof. If a line l in primal space meets the convex polytope P then all planes in the pencil of axis l meet P. Therefore in the dual space l^δ does not intersect P^δ. Conversely, if a line l in primal space misses P, than there is one plane supported by l which is disjoint from P. Therefore the dual line l^δ stabs P^δ. ∎

Theorem 1 *Given a compact star-shaped polyhedron $\mathcal{P}$ of size n, $M(\mathcal{P}) = F(\mathcal{P}) = I(\mathcal{P}) = O(n^3 2^{c\sqrt{\log n}})$.*

Proof.

1) First we prove the bound for $M(\mathcal{P})$. Given a star-shaped polytope $\mathcal{P}$ with n edges and center O, we triangulate its boundary. In particular we project each edge of $\mathcal{P}$ onto the sphere at infinity obtaining a planar map. We triangulate this map and back project the new edges on the boundary of $\mathcal{P}$. This triangulation Σ has $O(n)$ triangles. We then compute the convex hull of the origin and every triangle $\sigma \in \Sigma$, thus obtaining a set of $O(n)$ tetrahedra covering $\mathcal{P}$ and such that each contains the origin. We can perturbate slightly each tetrahedron to make sure that the origin is in the interior of each tetrahedron. Let $\mathcal{P}^\delta$ be the set of dual polytopes to such tetrahedra. It is crucial to observe that, since O is common to all tetrahedra in the decomposition of $\mathcal{P}$, then dual of O, namely the plane at infinity in the dual space, is disjoint from all the dual tetrahedra in $\mathcal{P}^\delta$. Therefore we can apply the result on the set of stabbing lines described in [31] to bound the number of extremal stabbing lines for $\mathcal{P}^\delta$. From Lemma 1 this bound holds also for the set of extremal lines missing $\mathcal{P}$.

2) For a star-shaped polytope a missing line l is a free line, because l can be taken to infinity on the plane defined by l and O. The set of missing lines has only one component. We can move any line l until it is disjoint from the convex hull of $\mathcal{P}$, and then move it back to coincide with any other missing line. Thus the set of missing lines, the set of free lines and one isotopy class of missing lines are indeed the same set. ∎

Let us consider now, over all possible coverings of a star-shaped polyhedron by convex polyhedra all sharing a common point, one such covering with minimum number of edges. Let $\tilde{n}$ be such minimum number of edges. Note that by the above discussion $\tilde{n} = O(n)$. Theorem 1 implies the following corollary:

Corollary 1 *Given a compact star-shaped polyhedron $\mathcal{P}$, let $\tilde{n}$ be defined as above, then $M(\mathcal{P}) = F(\mathcal{P}) = I(\mathcal{P}) = O(\tilde{n}^3\beta(\tilde{n}))$.*

Proof. In the proof of Theorem 1 we just need to cover $\mathcal{P}$ with a collection of convex polyhedra sharing one point. Thus the real input size for the second part of the proof involving stabbing lines in dual space is the number of edges of such covering. The best bound is obtained using the covering with fewer edges. ∎

Let $P_1 ... P_k$ be a set of convex polyhedra with total n edges and sharing a point. Then the union of these polytopes $P' = P_1 \cup P_2 ... \cup P_k$ is a star-shaped polyhedron of size n' which may range from constant to $\Theta(n^2)$. For such star-shaped polyhedron P' Corollary 1 implies a bound on the set of missing lines $O(\bar{n}^3\beta(\bar{n}))$, where $\bar{n} = \min\{n, n'\}$.

3 Free lines induced by lines

3.1 Notation and Preliminaries

Let us fix an orthogonal reference frame in 3-space with unit vectors $(\vec{i}, \vec{j}, \vec{k})$ forming a positively oriented triple according to the skew rule. A point in this real 3-dimensional space has Cartesian coordinates (x, y, z) and homogeneous coordinates (x_0, x_1, x_2, x_3). The relations between the two systems of coordinates are given by the following equations: $x = x_1/x_0$, $y = x_2/x_0$ and $z = x_3/x_0$. Two points $a = (x_0, x_1, x_2, x_3)$ and $b = (y_0, y_1, y_2, y_3)$ in 3-dimensional homogeneous coordinates define a line l in 3-space. The six quantities $\xi_{ij} = x_i y_j - x_j y_i$ for $ij = 01, 02, 03, 12, 23, 31$ are called *Plücker coordinates* of the line l (oriented from x to y) [35]. These coordinates are the two-by-two minors of the two-by-four matrix formed by the coordinates of the point a (on the first row) and b (on the second row). The six parameters are not independent; they must satisfy the following equation (whose solution set constitutes the Plücker hypersurface or Klein quadric or Grassman manifold $\mathcal{F}_4^2$ [36, 35]):

$$\Pi \ : \ \xi_{01}\xi_{23} + \xi_{02}\xi_{31} + \xi_{03}\xi_{12} = 0 \tag{1}$$

The incidence relation between two lines l and l' can be expressed using the Plücker coordinates of l and l'. Let a_1, b_1 (resp. a_2, b_2) be two points on l (resp. l') oriented as l (resp. l'). The incidence between l and l' is expressed as the vanishing of the determinant of a four-by-four matrix whose rows are the coordinates of a_1, b_1, a_2, b_2 in this order from top to bottom.

$$\begin{vmatrix} a_{10} & a_{11} & a_{12} & a_{13} \\ b_{10} & b_{11} & b_{12} & b_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ b_{20} & b_{21} & b_{22} & b_{23} \end{vmatrix} = 0 \tag{2}$$

If we expand the determinant according to the two-by-two minors of the sub-matrix formed by the coordinates of the points a_1, b_1 and the minors of the sub-matrix formed by the points a_2, b_2, we obtain the following bi-linear equation in which only Plücker coordinates are involved:

$$\xi_{01}\xi'_{23} + \xi_{02}\xi'_{31} + \xi_{03}\xi'_{12} + \xi'_{01}\xi_{23} + \xi'_{02}\xi_{31} + \xi'_{03}\xi_{12} = 0 \tag{3}$$

Let us introduce two mappings: $\pi : l \rightarrow \pi(l)$ maps a line in R^3 to an hyperplane in $\mathcal{P}^5$ (5-dimensional oriented projective space) whose plane coordinates are the Plücker coordinates of l appropriately reordered. $p : l \rightarrow p(l)$ maps a line in R^3 to a point in $\mathcal{P}^5$ whose coordinates are the Plücker coordinates of the line. The incidence relation between the two lines l, l' (expressed by Equation 3) can be reformulated as an incidence relation between points and hyperplanes in $\mathcal{P}^5$. Equation 3 can be rewritten in the form $\pi_l(p_{l'}) = 0$, which is equivalent to requiring point $p(l')$ to belong to hyperplane $\pi(l)$. Computations that are standard in real spaces can be done in oriented projective spaces using a method in [36]. For any given pair of lines l and l' the sign of $\pi_l(p_{l'})$ is called the *Plücker relative orientation* of l and l', denoted by $l \diamond l'$. Computing $l \diamond l'$ is equivalent to testing the relative position of a Plücker point $p(l)$ with respect to a Plücker hyperplane $\pi(l')$.

Given a line l, a vector $\vec{v}$, and a line l' in R^3 we say that l is *above* l' in direction $\vec{v}$ (namely $above(l, l', \vec{v})$) if moving l in direction $\vec{v}$ we eventually intersect l'. Note that a $above(l, l', v)$ is true if and only if l is *free* from l' in direction $-v$. Let us define oriented lines by ordered pairs of points in 3-space: $l = (a, b)$ and $l' = (a', b')$. The *tsp-relative orientation* is the sign of the triple scalar product of $(a - b, a' - b', \vec{v})$:

$$tsp(l, l', \vec{v}) = sign \begin{vmatrix} a_x - b_x & a_y - b_y & a_z - b_z \\ a'_x - b'_x & a'_y - b'_y & a'_z - b'_z \\ v_x & v_y & v_z \end{vmatrix} \tag{4}$$

The determinant in the definition of $tsp(l, l', v)$ can be expanded as a bilinear form in the one-by-one minors of the first row and the two-by-two minors of the second and third row. We can interpret this bilinear form as testing the relative position of a point $p'(l', v)$ with respect to the hyperplane $\pi'(l)$ on a 2-dimensional projective plane G. Considering the signs $(+1, -1)$ as boolean values, the following technical lemma is proved in [29]:

Lemma 2 ([29]) *A line l is above a set $L = \{l_i\}$ of lines with respect to v if and only if the following predicate is true:*

$$\bigwedge_i [(l_i \diamond l) \ xor \ tsp(l_i, l, v)] \tag{5}$$

Our aim is to find a bound on the descriptive complexity of the lines satisfying formula (5) for some value of v, with respect to a given finite set of lines L. On the parametric plane G of the *tsp*-test we have a set of planar lines $\{\pi'(l) | l \in L\}$ induced by L, which form the arrangement $A_G(L)$. For each cell c in $A_G(L)$ the corresponding set of Plücker points of free lines is defined by an intersection of Plücker halfspaces forming a polyhedron $K_c(L)$ in Plücker space. A vertex of $K_c(L)$ is called a *free vertex*.

The set $F(L)$ is represented therefore as the intersection of the Plücker hypersurface with the collection of polyhedra in Plücker space $\mathcal{K}(L) = \{K_c(L) | c \in A_G(L)\}$. Note that the polyhedra in $\mathcal{K}(L)$ are pairwise disjoint since they are different cells in the arrangement of hyperplanes $\{\pi(l) | l \in L\}$. Thus a bound on $\sum_c \# K_c(L)$, where the operator $\#$ counts the faces of any dimension bounding a polytope, implies a bound on $F(L)$.

3.2 An upper bound for $F(L)$

We consider now a set L of lines in general position (i.e. only two lines meet any four lines in L) and the corresponding collection $\mathcal{K}(L)$. A standard perturbation argument [14, 8] shows that the maximum complexity of $\mathcal{K}(L)$ is attained by a set of hyperplanes in general position (i.e. any 5 hyperplanes meet in a single point). Under this general position hypothesis we have that any polyhedron $K_c(L)$ is *simple*. From Lemma 2.1 in [4] we have that the number of vertices of a simple polyhedron is an upper bound to the number of faces of any dimension bounding the polyhedron. We present a proof of a bound on $F(L)$ which follows closely the proof of an upper bound on the set of lines stabbing n triangles in 3-space [31].

Given the planar arrangement of lines $A_G(L)$ on G we use Matoušek's technique [18, 19] to partition the plane G into a set $\Sigma_G(L)$ of $O(r^2)$ triangles so that no triangle meets more than $O(n/r)$ hyperplanes. Let σ be one of these triangles on G. We partition L into two sets of lines in 3-space: $L_1(\sigma)$, whose hyperplanes $\pi'(l)$ *do not cut* σ, and $L_2(\sigma)$, whose hyperplanes $\pi'(l)$ *cut* σ. And furthermore, by the properties of the partition, $|L_1(\sigma)| < n$ and $|L_2(\sigma)| < O(n/r)$.

Definition 1 *Given a set L, a region σ and sets $L_1(\sigma)$ and $L_2(\sigma)$ as above, $A_\sigma(i,j,L)$ is the set of free vertices defined by i hyperplanes in $H_1(\sigma) = \{\pi(l) | l \in L_1(\sigma)\}$ and j hyperplanes in $H_2(\sigma) = \{\pi(l) | l \in L_2(\sigma)\}$, and are contained in the closure of $K_\sigma(L_1(\sigma))$.*

Let $\mathcal{F}(n)$ be the maximum number of *free vertices* for a set of n lines in *general position*. Let $\mathcal{F}_\sigma(n)$ be the number of free vertices in $K_\sigma(L_1(\sigma))$ for a $\sigma \in \Sigma_G$. We will show a uniform upper bound on $\mathcal{F}(n)$. Suppose without loss of generality that L, of size n, attains the maximum value $\mathcal{F}(n)$. We have

$$\mathcal{F}(n) \leq \sum_{\sigma \in \Sigma_G(L)} \mathcal{F}_\sigma(n) \tag{6}$$

and

$$\begin{aligned} \mathcal{F}_\sigma(n) \quad \leq \quad & |A_\sigma(0,5,L)| + |A_\sigma(1,4,L)| + |A_\sigma(2,3,L)| + \\ & |A_\sigma(3,2,L)| + |A_\sigma(4,1,L)| + |A_\sigma(5,0,L)| \end{aligned} \tag{7}$$

Now we bound independently each term in (7).

1) $|A_\sigma(5,0,L)|$ represents the number of vertices touching five hyperplanes in $H_1(\sigma)$ and in $K_\sigma(L_1(\sigma))$. These vertices are the vertices of $K_\sigma(L_1(\sigma))$. The number of such vertices is $O(n^2)$, by the Upper Bound Theorem for polytopes (see [14, Chapter 6,Theorem 6.12]).

2) $|A_\sigma(4,1,L)|$. These are vertices formed by intersecting an edge of $K_\sigma(L_1(\sigma))$ with an hyperplane from $H_2(\sigma)$. Since there are at most $O(n^2)$ such edges, the number of vertices is bounded by $O(n^3/r)$.

3) $|A_\sigma(3,2,L)|$ represents the number of vertices incident to two hyperplanes in $H_2(\sigma)$ and a two-dimensional face of $K_\sigma(L_1(\sigma))$. We count these vertices by intersecting

pairs of hyperplanes in $H_2(\sigma)$, thus forming $O((n/r)^2)$ 3-dimensional sub-spaces, then we intersect each such subspace with the $K_\sigma(L_1(\sigma))$. Each such intersection is a 3-dimensional polytope and thus has $O(n)$ vertices. The total number of such vertices is thus $O(n^3/r^2)$.

4) $|A_\sigma(2,3,L)|$ represents the number of vertices incident to three hyperplanes in $H_2(\sigma)$, two hyperplanes in $H_1(\sigma)$ and on the boundary of $K_\sigma(L_1(\sigma))$.

   Let us consider the family of 2-dimensional subspaces $S_1, .., S_k$ obtained by intersecting every triple of hyperplanes from $H_2(\sigma)$. And let us consider the family of planar polygons $P_1, ..., P_k$ where $P_i = S_i \cap K_\sigma(L_1(\sigma))$. Each vertex v of P_i can be charged to one of its incident edges so that no edge is charged more then once. Let e be an edge of P_i and h its generating hyperplane in $H_1(\sigma)$. If we choose any subset $B \subset L_1$ which contains h, then h will contribute an edge e' in the planar polygon $P_i' = S_i \cap K_\sigma(B)$. We can charge this edge e' to one of its incident vertices v' such that each vertex is charged no more than once.

   We partition L_1 into r disjoint sets $B_1, \ldots, B_r$ of size at most $\lceil n/r \rceil$. We form r sets $Q_i = B_i \cup L_2(\sigma)$, for $i = 1, \ldots, r$. From the above observations we have that each free vertex v in $A_\sigma(2,3,L)$ can be charged to some free vertex v' of some Q_i in such way that v' is charged only a constant number of times. The maximum number of free vertices for any set of $2n/r$ lines is $\mathcal{F}(2n/r)$. Therefore $r\mathcal{F}(2n/r)$ is an upper bound for $|A_\sigma(2,3,L)|$.

5) $|A_\sigma(1,4,L)|$ represents the number of vertices touching four hyperplanes in $H_2(\sigma)$, one hyperplane in $H_1(\sigma)$ and incident on $K_\sigma(L_1(\sigma))$. We partition L_1 into r disjoint sets $B_1, \ldots, B_r$ of size at most $\lceil n/r \rceil$. We form r sets $Q_i = B_i \cup L_2(\sigma)$, for $i = 1, \ldots, r$. We observe that a *every vertex in $A_\sigma(1,3,L)$ is a free vertex for exactly one of the sets B_i*. The maximum number of free vertices for any set of $2n/r$ lines is $\mathcal{F}(2n/r)$. Therefore $r\mathcal{F}(2n/r)$ is an upper bound for $|A_\sigma(1,3,L)|$.

6) $|A_\sigma(0,5,L)|$ counts the number of extremal free lines for the set $L_2(\sigma)$ of size n/r within $K_\sigma(L_1(\sigma))$. $|A_\sigma(0,5,L)|$ is bounded from above by $\mathcal{F}(n/r)$.

We obtain the following equation

$$\mathcal{F}(n) \leq r^2n^2 + n^3r + n^3 + r^3\mathcal{F}(n/r) + r^2\mathcal{F}(n/r) \tag{8}$$

where we omit multiplicative constants. Equation 8 is solved in [31] and we obtain the main theorem:

Theorem 2 *Given a set L of n lines in 3-space the complexity of $\sum_c K_c(L)$ is $O(n^3 2^{c\sqrt{\log n}})$, for a suitable constant c.*

and the following corollary:

Corollary 2 *Given a set L of n lines in 3-space the complexity of the set of free lines $F(L)$ is $O(n^3 2^{c\sqrt{\log n}})$, for a suitable constant c.*

Since the lines in the construction of an upper envelope of lines in [8] are particular free lines, the lower bound $\Omega(n^3)$ in [8] holds for free lines. The upper bound of Theorem 2 is almost tight. Using an idea of Agarwal [1] a slightly tighter $O(n^3 \log n)$ bound on $F(L)$ can be found.

4 Translating sets of lines

Once we know that the set of free lines with respect to L has complexity at most $O(n^3\beta(n))$ the first algorithmic problem to be addressed is testing a line for membership in $F(L)$.

Theorem 3 *Membership in $F(L)$ can be tested in time $O(\log n)$ using a data structure of size $O(n^{3+\epsilon})$, which is built in $O(n^{3+\epsilon})$ expected time. Moreover, the data structure returns a constant size representation of the set of all free directions for the query line.*

Proof. 1) *Construction of the data structure.* We select a random sample R of L, where the size of R is $r < n$, and we build the planar arrangement $A_G(R)$ on the plane G of the tsp-test, obtaining $O(r^2)$ cells. For each such cell we build the corresponding Plücker polyhedron in 5-space for the lines in R. The convex polyhedra in $\mathcal{K}(R)$ are all disjoint and have a total of $O(r^3\beta(r))$ faces of any dimension. Therefore we can triangulate these Plücker polyhedra obtaining $O(r^3\beta(r))$ disjoint simplices. From the random sampling theory [12], each simplex is cut by no more than $O(n/r \log r)$ of the Plücker hyperplanes corresponding to lines in L. Let s be a simplex in Plücker space so generated and let c be the corresponding cell in $A_G(R)$. We have to consider 3 cases:

(i) Let $L_1(s) \subset L$ be the set of lines such that $\pi(l)$ meets s. From the above discussion this set has size $|L_1(s)| \leq n/r \log r$. We deal with it by a recursive call to the construction we are describing.

(ii) Let $L_2(s) \subset L$ be the set of lines such that $\pi(l)$ does not meets s and $\pi'(l)$ meet c. For these lines the sign of $\pi(l)$ with respect to s is well defined. We can invert such sign and determine an halfplane $\pi'(l)^+$ on the plane G. We take the intersection of these halfplanes with the region c and we store such polygon P_s. The maximum size of this polygon is $O(n)$.

(iii) Let $L_3(s) \subset L$ be the set of lines such that $\pi(l)$ does not meet s, and $\pi'(l)$ does not meet c. Region c is on a definite side of each hyperplane $\pi'(l)$, therefore we can check whether the simplex s is on the opposite side of $\pi(l)$ for each $l \in L_3(s)$. If this is the case for all lines in $L_3(s)$ then we save s for further processing, otherwise s is marked as *not-F*. The result of the construction is a search tree which we denote with $D(L)$. It is easy to see that the time needed to carry on the construction satisfies this recurrence:

$$T(n) \leq O(r^3\beta(r))[T(n/r \log r) + n \log n + n] + nr^{O(1)}$$

The solution is $O(n^{3+\epsilon})$. A similar bound holds for the storage.

2) *Query algorithm.* For any line l the set of feasible directions of l with respect to any set of lines is represented on the plane at infinity as a convex wedge which has the direction of l as apex. Therefore we can represent the feasible directions for l as an interval on a 1-dimensional space.

We keep during the query the current interval of feasible directions for l which we denote with $i_f(l)$. At the end of the query we return $i_f(l)$, which will be empty in case the query

line l is not free. Given a query line l we initialize $i_f(l)$ to $[-\infty, +\infty]$ and we locate the point $p(l)$ in Plücker space in $\mathcal{K}(R)$ stored at the root of $D(L)$. If $p(l)$ falls out of any simplex or is within a simplex marked not-F, then we set $i_f(l) = \emptyset$. If $p(l)$ is within a simplex s we find the associate polygon P_s. For a given line l and a variable v, the locus $p'(l, v)$ is a line in G. Therefore in logarithmic time we can check whether this locus meets P_s. If it does not, we set $i_f = \emptyset$. If it does, we update $i_f(l)$ by intersecting it with the intervals of values of v for which $p'(l, v)$ falls within P_s. Then we recurse the query in the data structure associated with s. We intersect $i_f(l)$ with the feasibility interval returned by the recursive call. The correctness of the query algorithm derives from the fact that it computes the value of formula (5) for the query line.

If we choose r to be a constant then the query time is $O(\log^2 n)$. If we choose $r = n^\nu$ and we use auxiliary fast point location data structure (e.g. see [31, 28]) we can reduce the query time to $O(\log n)$. ∎

4.1 Finding a feasible direction

Given two sets of lines A and B can we separate one set from the other using one translation v? This is equivalent to ask whether:

$$\exists v[\bigwedge_{i \in A, j \in B} (l_i \diamond l_j) \; xor \; tsp(l_i, l_j, v)]$$

In turns this is equivalent to solving $|A||B|$ linear inequalities, which we can solve in time time $O(|A||B|)$ using for example Megiddo's method for linear programming in linear time [22]. A quadratic method is obtained also by modifying for lines a method of Nurmi and Sack [25] for polyhedra, followed by an application of linear programming. The discussion of the previous section gives us a first handle to produce a a subquadratic algorithm. It is easy to see that for each line in A the set of feasible direction is a convex wedge on the plane at infinity. Therefore the set of feasible direction for all lines in A has worst case complexity $\Theta(n)$. A trivial observation is that if set A is free form B in direction v, if and only if B is free from A in direction $-v$. So, when comparing sets A and B we can switch the roles of A and B, but we have to take care of the fact that the set of feasible direction obtained is reflected with respect to the origin.

Theorem 4 *It is possible to find a feasible direction for two sets of n and m lines in time $O(n^{3/4}m^{3/4+\epsilon} + m^{1+\epsilon} + n^{1+\epsilon})$.*

Proof. Let us consider a set A of n lines and a set B of m lines. If $m > n^3$ we use directly the algorithm of Theorem 3. We use $O(n^{3+\epsilon})$ time to preprocess A and we perform m queries in $O(m \log n)$ time. We obtain m linear constraints and we can find a feasible direction in time $O(m)$. The total cost is $O(m^{1+\epsilon})$.

Let us suppose that $m < n^3$. The aim of this part of the algorithm is to efficiently produce for each line in $l \in B$ the representation $i_f(l)$ of its wedge of feasible directions. We take a random sample $R \subset B$ of constant size r. We build the set $\mathcal{K}(R)$ for the set R using a brute force method and the planar arrangement $A_G(R)$ on the plane G. We triangulate the polytopes in $\mathcal{K}(R)$, obtaining $\mu(r) = O(r^3\beta(r))$ disjoint simplices in Plücker space. For

each simplex s_i we compute the set A_i of Plücker points of lines in A which are inside s_i. Let n_i be the cardinality of A_i. We compute the set $B_i \subseteq B$ whose Plücker hyperplane intersect s. Let $m_i = |B_i|$. By the random sampling theory $m_i \leq cm/r \log r$, with high probability. As in the proof of Theorem 3 a simplex s is associated with a unique planar cell c of $A_G(R)$. We define $L_2^i \subseteq B$ and $L_3^i \subseteq B$ as in the proof of Theorem 3. As at step (iii) of the proof of Theorem 3 we check whether the simplex s is to be marked not-F. If yes and $n_i > 0$, we stop the algorithm and we answer negatively to the separability problem. As in step (ii) of the proof of Theorem 3 we construct the polygon P_s, and for each element of $l \in A_i$ we check whether the corresponding locus $p'(l, v)$ on the parametric plane G meets P_s and we extract the corresponding interval of directions. We update $i_f(l)$ by intersecting its old value with the new interval. If any interval $i_f(l)$ is empty we terminate the algorithm and answer negatively.

We recurse the construction on the set A_i and the set B_i of lines relative to s. The recursive call either returns a negative answer or it returns a $i_f(l)$ for each element of A_i. We intersect this interval with the one computed previously. If any interval is empty we answer negatively.

If we have $m_i > n_i^3$ then we use the direct method of Theorem 3. We obtain for each element of B_i a feasibility interval with respect to A_i. Each pair $\{(l, i_f(l)) | l \in B_i\}$ defines a wedge on the plane at infinity. By intersecting all the wedges we obtain a polygon. Each point on the polygon is a direction for which B_i is free from A_i. By reversing the sign of the directions we obtain the polygon of directions for which A_i is free from B_i. It is easy to determine now for each line $l \in A_i$ its interval $i_f(l)$ with respect to B_i. This reflection step is needed to compare the result with intervals at the higher levels of the algorithm. Finding the polygon of free directions costs time $O(m_i \log m_i)$, determining the $i_f(l)$ for each line in A_i takes time $O(n_i \log m_i)$.

The overall algorithm generates $O(n)$ feasibility intervals. We find a common feasible direction by computing a point in the intersection of wedges corresponding to the feasibility intervals in time $O(n)$ using Megiddo's liner programming method. The total time $T(n, m)$ needed to compute the intervals dominates the overall running time. We have:

$$T(n,m) = \begin{cases} O(m^{1+\epsilon}) & \text{for } m > n^3 \\ \sum_{i=1}^{\mu(r)} T(n_i, m_i) + \mu(r)(n+m)\log m + mr^c & \text{otherwise} \end{cases} \tag{9}$$

for some constant c, where $\mu(r) = O(r^3\beta(r))$, $m_i = O(\frac{m}{r}\log r)$ and $\sum_i n_i = n$. The correctness of the algorithm comes from an argument similar to that of Theorem 3. The time bound for $T(n, m)$ is $O(n^{3/4}m^{3/4+\epsilon} + n^{1+\epsilon} + m^{1+\epsilon})$, as follows from an analysis similar to one in [15]. If $n = m$, we have $T(n, n) = O(n^{1.5+\epsilon})$. ∎

5 Intersecting polyhedral terrains

5.1 The CEGS-Algorithm

We start recollecting the main features of the *Hereditary Segment Tree* (HST) introduced in [8] [9]. Given a blue terrain $\mathcal{B}$ with m edges and a red terrain $\mathcal{R}$ with n edges, we project these edges on the xy-plane obtaining a set $B = \{b_1, .., b_m\}$ of pairwise interior-disjoint blue

segments and a set $R = \{r_1, .., r_n\}$ of pairwise interior-disjoint red segments. We assume without loss of generality that $n \geq m$.

The HST of B and R is a balanced tree $\mathcal{T}$ where we associate to each node v a pair of subsets of $B_v \subseteq B$ and $R_v \subseteq R$ with the following properties: (i) the sum of the size of all lists B_v and R_v for all v in $\mathcal{T}$ is $O(n \log^2 n)$; (ii) each intersection of a blue segment with a red segment is reported at one and only one node v of $\mathcal{T}$; and (iii) every red segment in R_v meets every blue segment in B_v. Moreover, the segments in B_v have end-points outside a vertical strip I_v on the xy-plane parallel to the y-axis. Thus, segments in B_v have a well defined vertical order within the strip I_v. On each set B_v at node v of $\mathcal{T}$ we build an auxiliary balanced tree T_v of degree δ, based on the vertical order of segments in B_v. Thus each node τ of T_v is associated with a connected part of I_v. All the intersections among an edge of $\mathcal{R}$ and a face of $\mathcal{B}$, except possibly a linear number of them, can be charged to the fact that a red segment $r \in R_v$ meets two blue segments $b_1, b_2 \in B_v$ and the corresponding red edge in 3-space is above one and below the other of the two corresponding blue edges in 3-space. Over the whole data structure each actual intersection can be charged to this event, which is called a *witness intersection*, at most $O(\log_\delta n)$ times. Those intersections not found using witness intersections can be found using easier planar techniques [8].

During the preprocessing we build $\mathcal{T}(B, R)$ with the auxiliary trees T_v. Moreover at each node τ of a tree T_v with associated list B_τ of length less than $\sqrt{n}$ we build a data structure described in [8] of size quadratic in the number of blue edges, such that any witness intersection is detected in time $O(\log n)$. The overall time to construct such data structures is $O(n^{3/2+\epsilon})$ and this cost contributes to the final overhead term of the CEGS-Algorithm.

Next, the red segments at R_v are moved down the tree T_v. At each node τ of T_v for which no query data structure has been built, the list of R_v is duplicated and sent to both children of τ. This phase produces a total of $O(n^{3/2+\epsilon})$ red segments in the algorithm. When these red segments are pushed further down the tree we can detect witness intersections at logarithmic cost. Red segments failing the test are not passed down the tree. Thus the cost of detecting witness intersection from now on is charged to the number of witness intersections $k \log_\delta n$. Choosing $\delta = n^{\epsilon'}$ for a positive $\epsilon' < \epsilon$ we obtain the overall time bound.

To summarize, in the CEGS-algorithm [8, 9] we can distinguish four main cost factors. During preprocessing:

(i) cost for setting up the HST data structure.

(ii) cost for setting up the data structure to check witness intersections.

During queries:

(iii) cost for duplicating red edges.

(iv) cost for tracing witness intersections up to the leaves of the tree.

5.2 The Relevance Test for Red Edges

The relevance test for a red edge in a set R_τ, for a node τ in T_v is failed only if this red edge is above all (or below all) the blue edges in B_τ. Since we have red edges whose projection

meet all blue edges involved in a single test we can extend the red and blue edges into full lines without changing the outcome of the test. Since the set has two symmetric parts we discuss only the first part. A red line is vertically above a set of blue lines if and only if formula (5) is satisfied for $v_z = (0, 0, -1)$. Thus for a query line l we have to test $p'(l, v_z)$ against hyperplanes in a 1-dimensional projective space and then check $p(l)$ for inclusion in a 5-dimensional polytope in Plücker space. If we interpret $p'(l, v)$ and $p(l)$ as hyperplanes we can answer equivalent problems which are an half-space range problem in 1-space and a half-space emptiness problem in 5-space.

Using a multi-level data structure approach in [2] and [20] (see also [21] for an abstract treatment of multi-level data structures) we obtain that such test can be computed in time $O(n^{1+\epsilon}/s^{1/2})$ using a data structure of size s with $|B_\tau| \leq s \leq |B_\tau|^2$. The time used to build the data structure is $O(s^{1+\epsilon})$.

5.3 The overall algorithm

In this section we prove the following main result:

Theorem 5 *Given a polyhedral terrains $\mathcal{B}$ with m edges and a polyhedral terrain $\mathcal{R}$ with $n \geq m$ edges, all intersections between edges of $\mathcal{R}$ and faces of $\mathcal{B}$ can be found in time $O(n^{4/3+\epsilon} + K^{1/3}n^{1+\epsilon} + K \log^2 n)$ where K is the number of such intersections.*

As in the CEGS-Algorithm we build the tree $\mathcal{T}$ and we discover witness intersections at each node v of $\mathcal{T}$. The total cost of the algorithm after the construction of $\mathcal{T}$ is the sum of the cost at each node v of $\mathcal{T}$. Since the total duplication of segments over $\mathcal{T}$ is quite small the time bound for a node v hold essentially for the all tree, modulo some polylogarithmic factors which will be covered in the final bound by $O(n^\epsilon)$ factors. To simplify the notation we denote the size of B_v with m and of R_v with n. Moreover, we assume without loss of generality that $m \leq n$.

We set the threshold for tracing intersections, at logarithmic cost for intersection, at nodes τ in T_v whose associated blue list has size less than $n^{1/3}$ (in the CEGS-Algorithm the threshold is at $n^{1/2}$!). Below the level of nodes for which we have sets of size $n^{1/3}$ the tracing of the intersections is made as in the CEGS-Algorithm with on-line queries of logarithmic cost. Setting up these data structures has a total cost $O(n^{4/3+\epsilon})$.

The main new idea is to avoid duplicating red edges that do not contribute some witness intersection. We consider the nodes of T_v for which an data structure has not been built yet. Let τ be one such node. At such node we have an input sets B_τ which is used in the preprocessing phase. On-line we obtain a red set R_τ of red edges that reach τ. During the query we use the data structure for the relevance test stored at τ to detect the red edges that do not contribute any intersection. These edges are not passed to the children of τ. Red edges passing the test are replicated and sent to all the children of τ. Note that in the structure of T_v, the blue lists at each node τ are split between the two children of τ, in such way that each children covers a connected portion of I_v. In the CEGS-Algorithm the red edges were passed down the tree and replicated at every level without any test.

Let $i = 0, .., \log_\delta n$ be the levels of the tree T_v, where $i = 0$ represents the root. At level i we have to build δ^i data structures for the relevance tests, where each such data structure has $m_{ij} = m/\delta^i$ blue edges as input. The node ij at level i receives, during the query phase,

$k_{i-1,j}$ red edges where $k_{i-1,j}$ is the number of red edges at the father of node ij passing the relevance test. Since an edge is passed only if contributes a (witness) intersection we have $\sum_j k_{ij} \leq K \log_\delta n$.

We observe that the higher the level of the node, the smaller the set of *blue* edges becomes and we can spend less time in preprocessing. So, while keeping the total preprocessing time of each level the same function of m, we can reduce the query time for levels with higher number. This effect tending to the reduction of the total time is balanced by the fact that if we discover many intersections (K is large) we have to perform many queries.

It is enough to consider form now on only levels with input blue size $m_{ij} \geq n^{1/3}$. At a generic level i we have blue input size $m_{ij} = m^\alpha$ and $N(i) = m^{1-\alpha}$ nodes, with $1/3 < \alpha < 1$. We use $(m^\alpha)^\gamma$ storage for the data structure at node ij, for $1 \leq \gamma \leq 2$. At level i the total cost for preprocessing and queries is

$$\sum_{j=1}^{N(i)} [(m^\alpha)^\gamma + k_{i-1,j}(m^\alpha)^{1-\gamma/2}]. \tag{10}$$

The summation on the first term gives $m^{\alpha\gamma}m^{1-\alpha}$ which we set to be equal to m^π, where π is a parameter to be tuned later. We obtain: $\pi = \alpha\gamma + 1 - \alpha$ from which $\gamma = 1 + (\pi - 1)/\alpha$. The exponent of the cost function for each query is $Q = \alpha(1 - \gamma/2) = (\alpha - \pi + 1)/2$.

Lemma 3 *An upper bound on the running time of the algorithm is obtained when the intersections are detected at nodes with lower possible level number, and when the intersections are spread evenly on the tree.*

Proof. For any given problem instance with n and m edges, the number of intersections is a given number K. The bound on the *preprocessing* time of the algorithm and on the time of a single query depends on the total number of intersections to discover and not on their distribution over the tree. On the other hand any intersection that is discovered (witness intersection [8]) must be traced down to the leaves of the tree. As we observed the query time decreases with the level, therefore we obtain the maximum cost under the assumption that all intersections are discovered at the highest level possible, which amounts to saturating the high levels of the tree. Moreover, if no red segment reaches some node we can avoid the preprocessing step for that node thus reducing the total time. ∎

Using the above lemma we can distinguish levels of the tree with low number which are *saturated* (i.e. all red edges contribute an intersection) and those with high number that are not saturated (i.e. no new intersections are discovered).

We assume for the time being that we know that the total output size $K = n^\beta$ for $1 < \beta < 3/2$. On the saturated levels $k_{i-1,j} = n$. Since we cannot obtain more than n^β intersections we have that the last saturated level must satisfy: $n^{1-\alpha}n = n^\beta$. So, for $\alpha > 2 - \beta$ we have saturation, for $\alpha < 2 - \beta$ we do not have saturation and we assume we do not find new intersections either.

Saturation case. We have $2 - \beta < \alpha < 1$. The summation on the second term of (10) is:

$$\sum_{j=1,N(i)} nm^Q \leq \sum_{j=1,N(i)} nn^Q = n^{1-\alpha+1+(\alpha-\pi+1)/2} = n^{(5-\pi-\alpha)/2}.$$

The total cost is given by summing this cost over all the saturation levels. The level with the highest cost is the last saturated level, which is attained at the lower end of the α range, for $\alpha = 2 - \beta$. There are at most $O(\log_\delta n)$ levels. We obtain a bound $n^{(\beta+3-\pi)/2} \log_\delta n$ on the cost of the queries on saturated levels.

Non-saturation case. When we reach the last saturated level there are no new intersections discovered. The cost of each query decreases when the level number increases. The preprocessing cost is $m^\pi \leq n^\pi$ at each level. Therefore the total cost of each non-saturated level is bounded by the cost of the last saturated level.

The total cost we obtain is $n^\pi \log_\delta n + n^{(\beta+3-\pi)/2} \log_\delta n$. Now we choose π so to balance the two terms. Setting $\pi = (\beta + 3 - \pi)/2$ we obtain $\pi = \beta/3 + 1$. Summarizing the above discussion, we have obtained that the cost of tracing intersection while descending the tree from the root up to the threshold level is $O(n^{4/3+\epsilon} + K^{1/3} n^{1+\epsilon})$. From the threshold level to the leaves each query has logarithmic cost and can be charged directly to witness intersections.

When we run the algorithm we do not know the value of K and as a consequence we do not know how much time to allocate for the prepocessing of the data structures for the relevance test. To overcome this situation we guess a value for K and we run the algorithm. The initial guess is $K_0 = n$, then guess K_u is obtained by doubling the preceding one: $K_u = 2K_{u-1}$. We have only a logarithmic number of guesses to do at most. If we exceed the time allowed by the bound we stop and start again doubling our estimate. For the first and the second term of the bound this logarithmic extra factor is swallowed by the n^ϵ factor. For the third term we note that the guessed values for K form a geometric progression. Therefore the sum of the terms is proportional to the last term in the summation, which in turns is no more than twice the actual number of intersections. Also, we choose $\delta = n^{\epsilon'}$. This completes the proof of Theorem 5.

Theorem 5 and the reduction in [8, 9] lead to the following corollary:

Corollary 3 *The intersection of two terrains of total size n can be found in time $O(n^{4/3+\epsilon} + K^{1/3} n^{1+\epsilon} + K \log^2 n)$, where K is the size of the intersection.*

6 Conclusions

We have shown some combinatorial bounds on the complexity of sets of lines missing polyhedral sets in 3-space. We have applied these bounds to the design of an algorithm for solving a translation problem for lines in 3-space. We have also discussed an improved algorithm for computing the intersection of polyhedral terrains.

Still, many natural questions on lines in 3-space are left unanswered. For example, which is the complexity of $F(Q)$ and $I(Q)$ for Q a simple polyhedron or a set of rods in R^3? We conjecture that a complexity close to cubic is the right answer.

A related challenge is to use effectively these combinatorial bounds for solving algorithmic visibility problems in 3-space. In a preliminary version of this paper [30] the repeated computation of the shadow of a star-shaped polyhedron from view-points given on-line is considered. Unfortunately the solution sketched in [30] has a flaw. The difficulty is that the set of missing lines is described in Plücker space as a collection of interior-disjoint convex polyhedra at the cost of using additional hyperplanes not related to the initial edges.

These additional constraints do not influence the bound on $F(P)$ but introduce spurious break-points in the on-line computation of the shadow. At the moment it is not clear thus whether the time and storage complexity claimed in [30] on this problem can be achieved.

References

[1] P. K. Agarwal. Personal communication. January 1992.

[2] P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 517–526, 1992.

[3] P. K. Agarwal and M. Sharir. Applications of a new space partitioning technique. In *Proceedings of the 1991 Workshop on Algorithms and Data Structures*, number 519 in Lecture Notes in Computer Science, pages 379–391. Springer Verlag, 1991.

[4] B. Aronov, J. Matoušek, and M. Sharir. On the sum of squares of cell complexities in hyperplane arrangements. In *Proceedings of the 7th ACM Symposium on Computational Geometry*, pages 307–313, 1991.

[5] B. Aronov and M. Sharir. Triangles in space or building (and analyzing) castles in the air. *Combinatorica*, 10(2):137–173, 1990.

[6] B. Aronov and M. Sharir. Castles in the air revisited. In *Proceedings of the 8th ACM Symposium on Computational Geometry*, pages 146–256, 1992.

[7] B. Chazelle. An optimal algorithm for intersecting three-dimensional convex polyhedra. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 586–591, 1989.

[8] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. Lines in space: combinatorics, algorithms and applications. In *Proc. of the 21st Symposium on Theory of Computing*, pages 382–393, 1989.

[9] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. Algorithms for bichromatic line segments problems and polyhedral terrains. To appear in *Algorithmica*. Also Tech. Rep. UIUCDCS-R-90-1578, University of Illinois at Urbana Champaign, Dept. of Comp. Sci., 1990.

[10] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. Diameter, width, closest line pair and parametric search. In *Proceedings of the 8th ACM Symposium on Computational Geometry*, pages 120–129, 1992.

[11] B. Chazelle, T. Ottman, E. Soisalon-Soininen, and D. Wood. The complexity of decidability of separation. In *Proceedings of the 11th International Colloquium on Automata, Programming, and Languages*, pages 125–134, 1984.

[12] K. Clarkson. New applications of random sampling in computational geometry. *Discrete & Computational Geometry*, 2:195–222, 1987.

[13] M. de Berg, D. Halperlin, M. Overmars, J. Snoeyink, and M. van Kreveld. Efficient ray-shooting and hidden surface removal. In *Proceedings of the 7th ACM Symposium on Computational Geometry*, pages 21–30, 1991.

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

[15] H. Edelsbrunner, L. Guibas, and M. Sharir. The complexity and construction of many faces in arrangements of lines and segments. *Discrete & Computational Geometry*, 5:161–196, 1990.

[16] J. Guibas and F. Yao. On translating sets of rectangles. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing*, pages 154–160, 1980.

[17] D. Halperin and M. Sharir. New bounds for lower envelopes in three dimensions, with applications to visibility in terrains. In *Proceedings of the 9th ACM Symposium on Computational Geometry*, 1993.

[18] J. Matoušek. Construction of ϵ-nets. In *Proceedings of the 5th ACM Symposium on Computational Geometry*, pages 1–10, 1989.

[19] J. Matoušek. Cutting hyperplane arrangements. In *Proceedings of the 6th ACM Symposium on Computational Geometry*, pages 1–9, 1990.

[20] J. Matoušek. Reporting points in halfspaces. In *Proceedings of the 32th IEEE Symposium on Foundations of Computer Science*, pages 207–215, 1991.

[21] J. Matoušek. Range searching with efficient hierarchical cuttings. In *Proceedings of the 8th ACM Symposium on Computational Geometry*, pages 276–285, 1992.

[22] N. Megiddo. Linear programming in linear time when the dimension is fixed. *Journal of ACM*, 31(1):115–126, 1984.

[23] B. Natarajan. On planning assemblies. In *Proceedings of the 4th ACM Symposium on Computational Geometry*, pages 299–308, 1988.

[24] O. Nurmi. On translating a set of objects in 2- and 3-dimensional space. *Computer Vision, Graphics, and Image Processing*, 36:42–52, 1986.

[25] O. Nurmi and J. Sack. Separating a polyhedron by one translation from a set of obstacles. In J. van Leeuwen, editor, *Proc. Workshop on Graph-Theoretic Concepts in Computer Science*, volume 344 of *Lecture Notes in Computer Science*, pages 202–212, 1988.

[26] D. Nussbaum and J. Sack. Translation separability of polyhedra. In *Abstracts of the first Canadian Conference on Computational Geometry*, page 34, 1989.

[27] M. Pellegrini. Stabbing and ray shooting in 3-dimensional space. In *Proceedings of the 6th ACM Symposium on Computational Geometry*, pages 177–186, 1990.

[28] M. Pellegrini. Ray shooting on triangles in 3-space. *Algorithmica*,9:471–494, 1993.

[29] M. Pellegrini. On Collision-free placements of simplices and the closest pair of lines in 3-space. To appear in *SIAM J. on Computing*. Preliminary version in *Proceedings of the 8th ACM Symposium on Computational Geometry*, pages 130–137, 1992.

[30] M. Pellegrini. On Lines Missing Polyhedral Sets in 3-Space. In *Proceedings of the 9th ACM Symposium on Computational Geometry*, 1993.

[31] M. Pellegrini and P. Shor. Finding stabbing lines in 3-space. *Discrete & Computational Geometry*, 8:191–208, 1992.

[32] R. Pollack, M. Sharir, and S. Sifrony. Separating two simple polygons by a sequence of translations. *Discrete & Computational Geometry*, 3:123–136, 1988.

[33] F. Preparata and M. Shamos. *Computational Geometry: an Introduction*. Springr Verlag, 1985.

[34] J. Schwartz and M. Sharir. A survey of motion planning and related geometric algorithms. In D. Kapur and J. Mundy, editors, *Geometric Reasoning*, pages 157–169. The MIT press, 1989.

[35] D. M. H. Sommerville. *Analytical geometry of three dimensions*. Cambridge University Press, 1951.

[36] J. Stolfi. Primitives for computational geometry. Technical Report 36, Digital SRC, 1989.

[37] G. Toussaint. Movable separability of sets. In G. Toussaint, editor, *Computational Geometry*, pages 335–375. North-Holland, 1985.