Discrete Comput Geom 12:203–221 (1994)

Discrete & Computational Geometry
© 1994 Springer-Verlag New York Inc.

On Lines Missing Polyhedral Sets in 3-Space*

M. Pellegrini

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

Abstract. We show some combinatorial and algorithmic results concerning finite sets of lines and terrains 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 that can be translated to infinity without intersecting a given finite set of n lines, where c is a suitable constant. This bound is almost tight.

(2) An $O(n^{1.5+\varepsilon})$ 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. $\varepsilon > 0$ is an arbitrary small but fixed constant.

(3) An $O(n^3 2^{c\sqrt{\log n}})$ upper bound on the worst-case complexity of the *envelope of lines* above a terrain with n edges, where c is a suitable constant.

(4) An algorithm for computing the intersection of two polyhedral terrains in 3-space with n total edges in time $O(n^{4/3+\varepsilon} + k^{1/3}n^{1+\varepsilon} + k \log^2 n)$, where k is the size of the output, and $\varepsilon > 0$ is an arbitrary small but fixed constant. This algorithm improves on the best previous result of Chazelle *et al.* [5].

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

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 three-dimensional problems. Results in three-dimensional computational geometry are still much fewer than those for similar problems cast

* A preliminary version of this work appeared in the *Proceedings of the 9th ACM Symposium on Computational Geometry*, pp. 19–28.

in two-dimensional space. One of the reasons for this state of affairs is that the combinatorial properties of lines and other one-dimensional objects in 3-space are not well understood. In a seminal paper of 1989 [5] Chazelle *et al.* use in an orginal way a range of techniques (Plücker coordinates of lines, geometric random sampling, segment trees) and obtain bounds on the complexity of certain configurations of lines in 3-space as well as efficient algorithms for solving problems on polyhedral terrains. This paper is in the same line of research and it aims at improving bounds on the combinatorial structure of sets of lines, and algorithms for three-dimensional translation and intersection problems.

1.1. Complexity of Sets of Lines Induced by a Finite Set of Lines

Using Plücker coordinates of oriented lines in projective 3-space it is quite easy to define a topology on the set of oriented spatial lines (see, e.g., pp. 24–25 [3]). Consequently, we can define paths of lines and other topological concepts for sets of lines. A given finite set L of n lines in 3-space induces a decomposition of the set of spatial lines $\mathscr{L}$ into isotopy classes. Two lines are in the same isotopy class if a continuous path (of lines) having the two given lines as extremes exists, such that any line on the path does not intersect any line of the given finite set.

We investigate worst-case upper bound on the combinatorial complexity of the following classes of lines:

(i) $M(L) \subset \mathscr{L}$ is the set of lines in R^3 that do not intersect any line in L.
(ii) $I(L) \subset M(L)$ is a connected component of $M(L)$ (i.e., an isotopy class of missing lines).
(iii) $F(L) \subset M(L)$ is the set of free lines in R^3 with respect to L (i.e., the set of lines in $M(L)$ that can be translated in some direction to infinity without collisions with elements of L).

These classes arise naturally in three-dimensional visibility and translation problems.

In the topology of lines derived by the Plücker coordinates these sets are open sets. For any of the above-mentioned sets of lines its *complexity* can be expressed as the number of *extremal lines in the closure of the set*. A line is extremal when it is incident to four lines in L. We denote with $|Q|$ the complexity of the set Q. For a given finite set L of n lines the following bounds are known: $|I(L)| = \Theta(n^2)$ and $|M(L)| = \Theta(n^4)$ [5]. There is an $O(n^3)$ upper bound for the lines *vertically above* L, which matches a cubic worst-case lower bound [5]. Clearly, a line $l \in \mathscr{L}$ vertically above any line in L can be translated to infinity without collisions with L along the vertical direction. In the definition of the set $F(L)$ we generalize this notion of a collision-free translation by considering translations in any direction. In Section 2 we prove that $|F(L)| = O(n^3 2^{c\sqrt{\log n}})$. This bound is almost tight since the $\Omega(n^3)$ lower bound in [5] holds for $|F(L)|$. This bound is proved by finding a representation for $F(L)$ as a family of $O(n^2)$ polyhedra in Plücker space which are in one-to-one correspondence with the cells of a planar arrangement of n lines.

The technique used is similar to one in [20] and the main novelty is in the reduction of the problem to this special representation. A recent method of Agarwal [1] together with the reduction shown in this paper implies a slightly tighter bound $|F(L)| = O(n^3 \log n)$.

1.2. Collision-Free Translations of Lines

Translation of polyhedral objects without collisions is an important problem in robotics and CAD/CAM in relation to the assembly of objects composed of polyhedral parts [15]. General versions of the assembly problems for polyhedra in 3-space can be PSPACE-hard [15]. In this paper we discuss some restricted types of assembly problems for which we can design efficient algorithms. We consider a finite set A of n red lines and a finite set B of n blue lines, and we ask for a direction v, if it exists, such that we can translate A simultaneously in direction v without collision with B. We give in Section 3 a nontrivial algorithm that finds a feasible direction in time[1] $O(n^{1.5+\varepsilon})$. The algorithm for this translation problem relies heavily on the upper bound obtained for $|F(A)|$. The general approach is to build a fast data structure to test whether a line $l \in B$ is in $F(A)$ and then trade off the query time and preprocessing time of the algorithm. This general approach is now somewhat standard. What makes this algorithm interesting is that the queries are not answered independently. We need to maintain a "coherence" among the different answers since in general two lines can be free for A but in different directions.

Although sets of lines are not usually found *per se* in application problems, it is important to be able to solve efficiently the problem restricted to lines as a subroutine for conceivable algorithms solving more practical versions of the problem involving polyhedra.

For a survey of results on the translation problem for planar sets we refer the reader to [24]. At the moment, efficient algorithms in 3-space are known only for translating convex polyhedra. Given two convex polyhedra with n edges each, the translation problem can be solved in time $O(n)$ [16]. If instead we are given two *simple* polyhedra with n edges each, an algorithm in [17] solves the translation problem in time $O(n^4 \log n)$.

1.3. On the Line-Envelope of a Terrain

In the second part of this paper we consider a class of polyhedral objects that are useful in modeling actual physical data (e.g., geographical information, surfaces of objects): polyhedral terrains. A polyhedral terrain is the graph of a piecewise linear bivariate function. For a polyhedral terrain P with n edges, we show in Section 4 that the set $M(P) \subset \mathcal{L}$ of lines missing P has complexity $|M(P)| = O(n^3 2^{c\sqrt{\log n}})$.

[1] Here and throughout the paper we denote with ε a positive real parameter independent of n that we can choose arbirarily small. The multiplicative constants in the big-Oh notation may depend on ε and in such a case they tend to infinity when ε tends to zero.

The boundary of $M(P)$ is also called the *line-envelope* of P. We give a remarkably simple proof of this bound based on the property of lines stabbing convex polyhedra. This bound has been obtained independently in [11] by using a more general and much more complex technique. Previously, only the naive $O(n^4)$ upper bound was known. An $\Omega(n^2\alpha(n))$ lower bound for $|M(P)|$ is shown in [11]. Again, the counting method of Agarwal [1] together with the reduction shown in this paper implies a slightly tighter bound $|M(P)| = O(n^3 \log n)$.

1.4. Intersection of Polyhedral Terrains

Computing the intersection of polyhedra in 3-space is a basic problem in computational geometry [21], [9] and has immediate applications, for example, in computing the interference of polyhedral mechanical parts. In Section 5 we give an efficient output-sensitive algorithm which has two polyhedral terrains on complexity n as input and produces their intersection in time $O(n^{4/3+\varepsilon} + k^{1/3+\varepsilon}n^{1+\varepsilon} + k \log^2 n)$, where k is the number of edges, vertices, and faces of the intersection. The previously asymptotically fastest algorithm due to Chazelle *et al.* has a time bound $O(n^{3/2+\varepsilon} + k \log^2 n)$ [5]. For the range $0 < k < n^{3/2}$ the algorithm we present has a better asymptotic performance than the one in [5] and for $k > n^{3/2}$ it is as fast.

As follows from the discussion in [6] and [5], 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 gist of our algorithm is in detecting efficiently those edges that do not contribute edge–facet intersections, by using auxiliary data structures controlled by a slack parameter. This parameter controls the tradeoff between the time spent in setting up the data structures and the time spent in using them. The freedom given by the slack parameter is then used during the analysis to derive the improved time bound on the running time of the algorithm.

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

2. Free Lines Induced by Lines

2.1. Notation and Preliminaries

Plücker Coordinates of Lines. Let us fix an orthogonal reference frame in 3-space with unit vectors $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ forming a positively oriented triple according to the skew rule. A point in this real three-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 three-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) [22]. 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 $\mathscr{F}_4^2$ [23], [22]):

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

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 submatrix formed by the coordinate of the points a_1, b_1 and the minors of the submatrix formed by the points a_2, b_2, we obtain the following bilinear equation in which only Plücker coordinates are involved:

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

We introduce a mapping $\pi: l \mapsto \pi(l)$ which maps an oriented line in R^3 to a hyperplane in $\mathscr{P}^5$ (five-dimensional oriented projective space) whose plane coordinates are

$$(\xi_{23}, \xi_{31}, \xi_{12}, \xi_{01}, \xi_{02}, \xi_{03}).$$

A second mapping $p: l \mapsto p(l)$ maps a line in R^3 to a point in $\mathscr{P}^5$ whose coordinates are the Plücker coordinates of l. The incidence relation between the two lines l, l' (expressed by (3)) can be reformulated as an incidence relatioin between points and hyperplanes in $\mathscr{P}^5$. Equation (3) can be rewritten as a dot product in the form $\pi(l') \cdot p(l) = 0$, 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 techniques in [23].

Definition 1. For any given pair of lines l and l' the sign of $\pi(l') \cdot p(l)$ 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 $\pi(l)$.

The "Above" Relation. Given a line l, a vector v, and a line l' in R^3 we say that l is *above* l' in direction v (namely, *above*(l, l', v)) if moving l in direction v we eventually intersect l'. Note that an *above*(l, l', v) is true if and only if l is *free* from l' in direction $-v$. Let us define oriented lines, as before, 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 $a - b, a' - b'$, and v:

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

We introduce a mapping $\pi' \colon l \mapsto \pi'(l)$ which maps an oriented line in R^3 to a line in $\mathscr{P}^2$ (two-dimensional oriented projective space) whose plane coordinates are $(a_x - b_x, b_y - a_y, a_z - b_z)$. A second mapping $p' \colon (l', v) \mapsto p'(l', v)$ maps a line and a vector in R^3 to a point in $\mathscr{P}^2$ whose coordinates are

$$[(a'_y - b'_y)v_z - (a'_z - b'_z)v_y, (a'_x - b'_x)v_z - (a'_z - b'_z)v_x, (a'_x - b'_x)v_y - (a'_y - b'_y)v_x].$$

The triple scalar product defined in (4) is equivalent to computing the sign of the dot product $\pi'(l) \cdot p'(l', v)$, which is equivalent to checking the relative position of the point $p'(l', v)$ with respect to the line $\pi'(l)$. We denote with G the two-dimensional projective space underlying the codomains of the mappings π' and p'. In the following sections of this paper we describe geometric constructions in Plücker space and on the plane G. The reason for introducing this *parametric* plane G is embodied in the next lemma that links the predicate "above" with the relative orientation in Plücker space and the tsp-relative orientation on the plane G.

Lemma 1 [19]. *Considering the signs* $(+1, -1)$ *as boolean values, a line* l *is above a set* $L = \{l_i\}$ *of lines with respect to a direction* v *if and only if the following predicate is true*:

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

Our aim now 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 we have a set of planar lines $\{\pi'(l) \mid l \in L\}$ 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 half-spaces 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 $\mathscr{K}(L) = \{K_c(L) \mid c \in A_G(L)\}$.

Note that the polyhedra in $\mathscr{K}(L)$ are pairwise disjoint since they are different cells in the arrangement of hyperplanes $\{\pi(l) | l \in L\}$. A bound on $\sum_c |K_c(L)|$, where the operator $| \, |$ counts the faces of any dimension bounding a polytope, implies a bound on $|F(L)|$.

2.2. *An Upper Bound for* $|F(L)|$

We now consider a set $L$of lines in general position (i.e., only two lines meet any four lines in L) and the corresponding collection $\mathscr{K}(L)$. A standard perturbation argument [9], [5] shows that the maximum complexity of $\mathscr{K}(L)$ is attained by a set of hyperplanes in general position (i.e., any five 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 [20].

We choose in the range $[1 \,..\, n]$ an integer r large enough so that we can use the partitioning results in [12] and [7] on the arrangement $A_G(L)$. By these results, the plane G is partitioned into a set $\Sigma_G(L)$ of $O(r^2)$ triangles so that no triangle meets more than $O(n/r)$ lines. Let σ be one of these triangles on G. We partition L into two sets of lines in 3-space: $L_{\text{out}}(\sigma)$, whose hyperplanes $\pi'(l)$ *do not cut* σ, and $L_{\text{in}}(\sigma)$, whose hyperplanes $\pi'(l)$ *cut* σ. Furthermore, by the properties of the partition, $|L_{\text{out}}(\sigma)| < n$ and $|L_{\text{in}}(\sigma)| < O(n/r)$. With $K_\sigma(L_{\text{out}}(\sigma))$ we denote the polyhedron obtained in Plücker space by intersecting half-spaces supported by hyperplanes in $H_{\text{out}}(\sigma) = \{\pi(l) | l \in L_{\text{out}}(\sigma)\}$, where we choose for each such hyperplane the side whose sign is opposite to the sign of $\pi'(l) \cdot q$, for an (abstract) point $q \in \sigma$ on G. Note that we obtain the same polytope for every point $q \in \sigma$, therefore q is chosen arbitrarily in constant time.

Definition 2. Given a set L, a region σ, a set $L_{\text{out}}(\sigma)$, and a set $L_{\text{in}}(\sigma)$ defined as above, $A_\sigma(i, j, L)$ is the set of free vertices incident to i hyperplanes in $H_{\text{out}}(\sigma) = \{\pi(l) | l \in L_{\text{out}}(\sigma)\}$, incident to j hyperplanes in $H_{\text{in}}(\sigma) = \{\pi(l) | l \in L_{\text{in}}(\sigma)\}$, and contained in the closure of $K_\sigma(L_{\text{out}}(\sigma))$.

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

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

and

$$\mathscr{F}_\sigma(n) \le |A_\sigma(0, 5, L)| + |A_\sigma(1, 4, L)| + |A_\sigma(2, 3, L)| \\ + |A_\sigma(3, 2, L)| + |A_\sigma(4, 1, L)| + |A_\sigma(5, 0, L)|. \tag{7}$$

Now we bound independently each term in (7):

1. $|A_\sigma(5, 0, L)|$ represents the number of vertices touching five hyperplanes in $H_{\text{out}}(\sigma)$ and in $K_\sigma(L_{\text{out}}(\sigma))$. These vertices are the vertices of $K_\sigma(L_{\text{out}}(\sigma))$. The number of such vertices is $O(n^2)$, by the Upper Bound Theorem for polytopes (see Chapter 6, Theorem 6.12, of [9]).
2. $|A_\sigma(4, 1, L)|$. These are vertices formed by intersecting an edge of $K_\sigma(L_{\text{out}}(\sigma))$ with a hyperplane from $H_{\text{in}}(\sigma)$. 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 $H_{\text{in}}(\sigma)$ and a two-dimensional face of $K_\sigma(L_{\text{out}}(\sigma))$. We count these vertices by intersecting pairs of hyperplanes in $H_{\text{in}}(\sigma)$, thus forming $O((n/r)^2)$ three-dimensional subspaces, then we intersect each such subspace with the $K_\sigma(L_{\text{out}}(\sigma))$. Each such intersection is a three-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 $H_{\text{in}}(\sigma)$, two hyperplanes in $H_{\text{out}}(\sigma)$, and on the boundary of $K_\sigma(L_{\text{out}}(\sigma))$.

   Let us consider the family of two-dimensional subspaces $S_1, \dots, S_k$ obtained by intersecting every triple of hyperplanes from $H_{\text{in}}(\sigma)$. Let us consider the family of planar polygons $P_1, \dots, P_k$ where $P_i = S_i \cap K_\sigma(L_{\text{out}}(\sigma))$. Each vertex v of P_i can be charged to one of its incident edges so that no edge is charged more than once. Let e be an edge of P_i and let h be its generating hyperplane in $H_{\text{out}}(\sigma)$. If we choose any subset $B \subset L_{\text{out}}$ which $P'_i = S_i \cap K_\sigma(B)$. 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_{out} into r disjoint sets $B_1, \dots, B_r$ of size at most $\lceil n/r \rceil$. We form r sets $Q_i = B_i \cup L_{\text{in}}(\sigma)$, for $i = 1, \dots, r$. 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 a 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 $\mathscr{F}(2n/r)$. Therefore $r\mathscr{F}(2n/r)$ 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 $H_{\text{in}}(\sigma)$, one hyperplane in $H_{\text{out}}(\sigma)$, and incident on $K_\sigma(L_{\text{out}}(\sigma))$. We partition L_{out} into r disjoint sets $B_1, \dots, B_r$ of size at most $\lceil n/r \rceil$. We form r sets $Q_i = B_i \cup L_{\text{in}}(\sigma)$, for $i = 1, \dots, r$. We observe that *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 $\mathscr{F}(2n/r)$. Therefore $r\mathscr{F}(2n/r)$ 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 $L_{\text{in}}(\sigma)$ of size n/r within $K_\sigma(L_{\text{out}}(\sigma))$. $|A_\sigma(0, 5, L)|$ is bounded from above by $\mathscr{F}(n/r)$.

We obtain the following recurrence relation:

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

where we omit multiplicative constants. Equation (8) is solved in [20]. We obtain the bound $\mathscr{F}(n) = O(n^3 2^{c\sqrt{\log n}})$ and, as a consequence, the following theorem:

Theorem 1. *Given a set L of n lines in 3-space $|F(L)| = O(n^3 2^{c\sqrt{\log n}})$, for a suitable constant c.*

Since the lines in the construction of an upper envelope of lines in [5] are particular free lines, the lower bound $\Omega(n^3)$ in [5] holds for free lines.

3. 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 $F(L)$.

Definition 3. If a line l is free in direction v, then v is a *feasible* direction for l.

We state a simple but fundamental lemma:

Lemma 2. *For any line l the set of feasible directions for l with respect to any set of lines L is represented on the sphere of directions as a convex wedge which has the direction of l as apex.*

Proof. Consider lines l_1 and l_2. The points on the sphere of direction representing the directions for which l_1 is translated away from l_2 is a hemisphere whose bounding great circle is incident to the direction of l_1. Thus the feasibility direction of l_1 with respect to a set of lines L is an intersection of hemispheres whose great circle is incident to a fixed point. Such intersection is therefore a connected and convex spherical wedge. □

Conceptually, we want to locate the Plücker polyhedron in $\mathscr{K}(L)$ containing the Plücker point of a query line l, and then use the cell of $A_G(L)$ associated with that polyhedron to compute the feasible directions for l. On the other hand, it is not obvious how to make the construction of $\mathscr{K}(L)$ independent from the construction of $A_G(L)$, while attaining a roughly cubic time bound. For this reason we propose an approach based on the interleaved construction of $\mathscr{K}(L)$ and $A_G(L)$, together with the fast point-location data structures required.

Theorem 2. *Membership in $F(L)$ can be tested in time $O(\log n)$ using a data structure of size $O(n^{3+\varepsilon})$, which is built in $O(n^{3+\varepsilon})$ expected time. Moreover, the data structure returns a constant size representation of the set of all feasible 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 by intersecting Plücker half-spaces. The convex polyhedra in $\mathscr{K}(R)$ are pairwise 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 results in [8] on the properties of random samples, each simplex is cut by no more than $O(n/r \log r)$ of the Plücker hyperplanes corresponding to lines in L, with high probability. 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 two phases for each simplex x. Phase (i). Let $L_1(s) \subset L$ be the set of lines such that $\pi(l)$ does not meet s. For these lines the sign of $\pi(l)$ with respect to s is well defined. We can invert such sign and determine a half-plane $\pi'(l)^+$ on the plane G. We take the intersection of these half-planes with the region c, and we store such polygon P_s. The maximum size of this polygon is $O(n)$. If P_s is empty, we mark s as *not-in-F* and we finish the preprocessing of s, otherwise we go to the next phase. Phase (ii). Let $L_2(s) \subset L$ be the set of lines such that $\pi(l)$ meets s. From the above discussion this set has size $|L_2(s)| = O(n/r \log r)$. We deal with this set by a recursive call to the construction procedure we are describing.

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 to carry on the construction satisfies this recurrence:

$$T(n) \leq c_1 r^3 \beta(r)[T(n/r \log r) + n \log n + n] + c_2 n r^5,$$

where c_1 and c_2 are constants. By choosing for r a suitable constant value we obtain a solution $O(n^{3+\varepsilon})$. A similar bound holds for the storage.

(2) *Query algorithm.* Using Lemma 2 we can represent the feasible directions for l as an interval on a one-dimensional space associated with l. 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 the case when 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 $\mathscr{K}(R)$ stored at the root of $D(L)$. If $p(l)$ does not fall into 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 associated 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 interval 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, the query time is $O(\log^2 n)$. If we choose r to be a small power of n, whose exponent depends on ε, we can reduce the query time to $O(\log n)$ while keeping the preprocessing asymptotically the same function of n (e.g., see [20] and [18] for details). □

3.1. 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 asking whether

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

In turn this is equivalent to solving $|A|\,|B|$ linear inequalities, which we can solve in time $O(|A|\,|B|)$ using Megiddo's method for linear programming in linear time [14]. The discussion of the previous section gives us a first handle to produce a subquadratic algorithm. A trivial observation which will become useful is that set A is free from 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 3. *It is possible to find a feasible direction for two sets of n and m lines in time $O(n^{3/4}m^{3/4+\varepsilon} + m^{1+\varepsilon} + n^{1+\varepsilon})$, where ε is an arbitrary small positive real constant, and the multiplicative constants may depend on ε.*

Proof. Let us consider a set A of n lines and a set of B of m lines. The aim of the first part of the algorithm is to produce efficiently for each line in $l \in A$ the representation $i_f(l)$ of its wedge of feasible directions. We consider two cases.

Case 1. Let us suppose that $m < n^3$. We take a random sample $R \subset B$ of constant size r. We build the planar arrangement $A_G(R)$ on the plane G of the tsp-test, obtaining $O(r^2)$ cells. For each cell we build the corresponding Plücker polyhedron in 5-space for the lines in R. The convex polyhedra in $\mathscr{K}(R)$ 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.

Let s be a simplex in Plücker space so generated and let c be the unique corresponding cell in $A_G(R)$. For each simplex s we compute the set A_s of Plücker points of lines in A which are inside s. Let $|A(s)| = n_s$. We define $L_1(s) \subset B$ and $L_2(s) \subseteq B$ as in the proof of Theorem 2. Let $m_s = |L_2(s)|$. From results in [8] on the properties of random samples, $m_s = O(m/r \log r)$ with high probability.

For each simplex s we have two phases. As in phase (i) of the proof of Theorem 2 we construct the polygon P_s using $L_1(s)$. If P_s is empty and $n_s > 0$ we interrupt the call returing a negative answer. Otherwise, for each line $l \in A_s$, 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 interrupt the procedure and return a negative answer. Otherwise we go to the next phase. Phase (ii). We recurse the algorithm on the set A_s and the set $L_2(s)$. The recursive call either returns a negative answer or it returns an interval $i_f(l)$ for each line of A_s. We intersect this interval with the one computed previously. If any interval is empty we answer negatively.

Case 2. Let us suppose now that $m > n^3$. In this case we call the algorithm of Theorem 2, using the lines in A as input and the lines in B as queries. We use $O(n^{3+\varepsilon})$ time to preprocess A and we perform m queries in $O(m \log n)$ time. We obtain m linear constraints. The total cost of these operations is $O(m^{1+\varepsilon})$. We obtain for each line of B a feasibility interval with respect to A. Each pair in $\{(l, i_f(l)) \mid l \in B\}$ defines a wedge on the sphere of directions. By intersecting all the wedges we obtain a polygon. Each point on the polygon is a direction for which B is free from A. By reversing the sign of the directions we obtain the polygon of directions for which A is free from B. It is now easy to determine for each line $l \in A$ its interval $i_f(l)$ with respect to B. This reflection step is needed to compare the intervals returned by deep recursive calls of the algorithm with the data at the higher levels of the algorithm. Finding the polygon of free directions costs time $O(m\ log\ m)$, determining the $i_f(l)$ for lines in A_s takes time $O(n \log m)$.

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 linear programming method. The total time $T(n, m)$ needed to compute the intervals dominates the overall running time. We have

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

for some constants c, where $\mu(r) = O(r^3\beta(r))$, $m_i = O((m/r)\log r)$, and $\sum_i n_i = n$. The correctness of the algorithm comes from an argument similar to that of Theorem 2. The time bound for $T(n, m)$ is $O(n^{3/4}m^{3/4+\varepsilon} + n^{1+\varepsilon} + m^{1+\varepsilon})$, as follows from an analysis similar to one in [10]. If $n = m$, we have $T(n, n) = O(n^{1.5+\varepsilon})$. □

4. Complexity of the Line-Envelope of a Terrain

Let us consider a well-known duality transformation between points and planes in 3-space, namely, the *polarity* δ [21] 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 $1/|\overline{Op}|$ from O, on the same side as p. Given a convex compact polytope P containing O in its interior, we define the set of planes $\delta(P) = \{\delta(p) \mid p \in P\}$ in dual space. We define as P^δ the complement in dual space of the set of dual points belonging to planes in $\delta(P)$. It is easy to show that P^δ is a convex compact polytope containing O in its interior if the primal and the dual spaces are overlapped. 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., $(P^\delta)^\delta = P$) [21]. Given a line l as a locus of points we obtain in the dual space a locus of planes which is a one-dimensional pencil of planes. We define as l^δ the axis of such a pencil of planes.

Lemma 3. *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, then there is one plane supported by l which is disjoint from P. Therefore the dual line l^δ stabs P^δ. □

For convenience we establish the main result of this section for the class of star-shaped polyhedra, which includes polyhedral terrains.

Theorem 4. *Given a compact star-shaped polyhedron $\mathscr{P}$ of size n, $|M(\mathscr{P})| = O(n^3 2^{c\sqrt{\log n}})$.*

Proof. Given a star-shaped polytope $\mathscr{P}$ with n edges and center O, we triangulate its boundary as follows: we project each edge of $\mathscr{P}$ onto the sphere at infinity obtaining a planar map. We triangulate this map and back project the new edges on the boundary of $\mathscr{P}$. 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 $\mathscr{P}$ 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 $\mathscr{P}^\delta$ 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 $\mathscr{P}$, the dual of O, namely the plane at infinity in the dual space, is disjoint from all the dual tetrahedra in $\mathscr{P}^\delta$. Therefore we can apply the result on the set of stabbing lines described in [20] to bound the number of extremal stabbing lines for $\mathscr{P}^\delta$. From Lemma 3 this bound also holds for the set of extremal lines missing $\mathscr{P}$. □

5. Intersecting Polyhedral Terrains

In this section we give an improved algorithm for computing the intersection of two terrains. We start by recollecting the main features of the algorithm in [5] which is the skeleton of our algorithm. Next we discuss where the algorithm in [5] is modified. The change does not impact on the proof of correctness of the algorithm, which is derived using the same arguments as in [5], and for this reason is not repeated here. What has to be changed is the analysis of the time bound of the algorithm. This analysis is quite simple in [5]. Here we are able to prove better bounds by using a more sophisticated approach.

5.1. *The Algorithm of Chazelle* et al.

We refer to the algorithm in [5] as the CEGS algorithm throughout this section. It is observed in [5] that the problem is easily solved by a straightforward tracing procedure once we have computed all the intersections of edges from one terrain with facets from the other terrain, and vice versa. Thus, given terrains Σ_1 with n (red) edges and Σ_2 with m (blue) edges, we concentrate our discussion on finding the intersections of red edges with blue facets. For technical reasons the facets are subdivided into trapezoids and we count edge–trapezoid incidences.

The CEGS algorithm is based on the properties of a data structure called the *Hereditary Segment Tree*, introduced in [5], which is an extension of the well-known segment-tree data structure (see, e.g., [21]). This data structure allows us to store two sets of planar segments in such a way as to make it easy to keep track of their intersections. In the CEGS algorithm this data structure is augmented with auxiliary data structures that take as input the lists of edges stored at the nodes of the hereditary segment trees. The auxiliary data structures allow us to compute quickly the above–below relation of edges in 3-space.

We take the edges of Σ_1 and we project them vertically on the xy-plane obtaining a set R of n red edges. We take the edges of Σ_2 and we project them vertically on the xy-plane obtaining a set B of m blue edges. Let $N = n + m$. Now we build an hereditary segment tree $\mathscr{T}$ with branching degree δ, whose value will be determined later, on the sets B and R. The height of the tree is $O(\log_\delta N)$ and each edge is stored in $O(\delta \log_\delta N)$ lists of red and blue edges associated with the nodes of $\mathscr{T}$ [5]. Let v be a node of $\mathscr{T}$ and let R_v, B_v be the lists of blue and red edges stored at v. We build on the list B_v a *complete binary tree* T_v and we associate to a node θ of T_v the list B_θ of blue edges stored in the subtree rooted at θ. We also associate to θ a list $R_\theta \subset R_v$ of red edges. This construction is carried out in such a way as to obtain sets B_θ and R_θ of mutually intersecting planar edges.

The Auxiliary Data Structure. Given a set of M lines in 3-space it is shown in [5] how to store them into a data structure of size $O(M^{2+\varepsilon})$, so that for a query line l we can determine in time $O(\log M)$ whether l is above all M lines or below all M lines. This data structure is called the *envelope structure* and it is built in time $O(M^{2+\varepsilon})$. Given a list B_θ we go back to the edges of Σ_1 that project onto edges of B_θ. We extend these edges into full lines and we build the envelope structure for these lines. We also extend the lines in R_θ into full lines and we use them to query the envelope data structure. Because the edges mutually intersect on the xy-plane the above/below relation among the corresponding edges of the terrains in 3-space is not changed by extending the edges into full lines.

Since we use quadratic time to build an envelope structure we cannot afford to construct it for every list. Instead we build it only if $|B_\theta| \leq N^{(1-\varepsilon)/2}$. By choosing $\delta = \lfloor N^{\varepsilon/2} \rfloor$ we have that we spend a total time $O(N^{3/2+\varepsilon})$ in the construction of envelope structures.

The Query Phase. When the auxiliary data structures are in place we start a phase of queries using the red edges in R_θ for every node θ in the tree T_v and for every

v in $\mathscr{T}$. If a red edge $r \in R_\theta$ is below or above all edges in B_θ we stop since r cannot intersect any blue trapezoid incident to edges in B_θ. Otherwise we distinguish three cases:

1. If θ is a leaf, then test r for intersections with the two trapezoids incident to the edge stored at θ. If it does, report the intersection.
2. If B_θ was too large to warrant the construction of the envelope structure, then recursively query both children of θ with r.
3. If θ is not a leaf and the list B_θ is small enough so that the envelope structure has been built, use the line spanning r as a query line. If r is above or below all edges of B_θ stop. Otherwise, recursively query both children of θ.

Let k_r be the number of intersections between $r \in R_\theta$ and a facet of Σ_2. The basic properties of the hereditary segment trees imply that each such intersection is detected exactly once [5]. On the other hand, each red edge is duplicated at most $O(\log N)$ times in the hereditary segment tree. Thus case 3 has overall cost $O(k \log^2 N)$. Each edge in Σ_1 incurs an overhead cost because it is replicated in the data structure and an overhead cost because it is duplicated during case 2 of the query phase. The total overhead per edge that cannot be charged to intersections is $O(N^{(1+\varepsilon)/2})$. The overall time bound of the algorithm is thus $O(N^{3/2+\varepsilon} + k \log^2 N)$, where k is the size of the output.

5.2. *Variations*

Agarwal and Matoušek note in [2] that Chazelle *et al.* [5] reduce the problem of detecting if a line is above or below a set of lines to the problem to answering half-space emptiness queries in Plücker space (five-dimensional projective space). This reduction, together with the method for solving half-space emptiness queries in [13], gives us a new envelope structure that, for a set of M lines and for a parameter s in the range $M \le s \le M^2$, is built in time $O(s^{1+\varepsilon})$ using $O(s)$ storage. Given a query line l we can determine in time $O(M^{1+\varepsilon}/s^{1/2})$ whether l is above all M lines or below all M lines. We call this data structure a *slack envelope structure* because we have a slack parameter s to control the preprocessing time and the query time. Now we make the following modifications to the algorithm of Chazelle *et al.*:

1. We build a normal envelope structure for nodes with $|B_\theta| \le N^{(1-\varepsilon)/3}$. We build a slack envelope structure at any other node θ. The value of the parameter s at each node θ is critical and we determine its value during the analysis in the next subsection.
2. Since now every node has a normal or a slack envelope structure, case 2 of the query phase is eliminated.

Note that the "semantic" of the algorithm has not changed and therefore the proof of correctness in [5] also holds for this mutant algorithm. In the next subsection we prove the complexity bound.

5.3. Analysis of the New Algorithm

Theorem 5. *Given a polyhedral terrain Σ_1 with n red edges and a polyhedral terrain Σ_2 with n blue edges, let $N = n + m$. All intersections between edges Σ_1 and faces of Σ_2 can be found in time $O(N^{4/3+\varepsilon} + k^{1/3}N^{1+\varepsilon} + k \log^2 n)$, where k is the number of such intersections.*

Proof. Let us consider the tree T_v and the lists R_v of size n_v and B_v of size m_v for a node v of the hereditary segment tree $\mathscr{T}$. Let $N_v = n_v + m_v$. Let $i = 0, \ldots, \log_2 m_v$ be the levels of the tree T_v, where $i = 0$ represents the root level. We denote with the pair of indices (i, j) the jth node of level i. At level i we have to build 2^i envelope data structures, where each such data structure has $m_{(i,j)} = m_v/2^i$ blue edges as input. Node (i, j) at level i receives, during the query phase, $k_{(i,j)}$ red edges, where $k_{(i,j)}$ is the number of red edges passed to node (i, j), from the *father* of node (i,j) plus the red edges stored at (i, j). Since an edge is passed only if it contributes at least one intersection we have $\sum_j k_{(i,j)} \leq (k + n_v) \log n_v$. We observe that we can keep the preprocessing time fixed on each level and thus reduce the query time for levels with higher index. This effect tending to the reduction of the total time is counterbalanced by the fact that if we discover many intersections (i.e., k is large) we have to perform many queries.

At a generic level i we have blue input size $m_{(i,j)} = m_v^\alpha$ and $l(i) = m_v^{1-\alpha}$ nodes, with $0 < \alpha < 1$. For nodes with $0 \leq \alpha < \frac{1}{3}$ we build normal envelope structures for a total preprocessing cost $O(m_v^{4/3+\varepsilon}) = O(N_v^{4/3+\varepsilon})$. Each query on these data structure costs $O(\log m_v)$ and can be charged to an intersection. It is enough to consider from now on only levels with exponent α in the range $\frac{1}{3} \leq \alpha \leq 1$. Let us redefine the slack storage parameter to be $s = m_{(i,j)}^\gamma = m_v^{\alpha\gamma}$, for a parameter γ in the range $1 \leq \gamma \leq 2$. At level i the total cost for preprocessing and queries is

$$\sum_{j=1}^{l(i)} m_v^{\alpha\gamma} + \sum_{j=1}^{l(i)} k_{(i,j)}(m_v^\alpha)^{1-\gamma/2}, \tag{10}$$

where we have omitted multiplicative constants and ε-powers in order to simplify the subsequent computations.

The first summation gives $m_v^{\alpha\gamma+1-\alpha}$. Our objective is to have the same total preprocessing time at each level, which we denote with the *exponent* η. We have $\eta = \alpha\gamma + 1 - \alpha$ and consequently $\gamma = 1 + (\eta - 1)/\alpha$. The exponent of the cost function for each query is $\alpha(1 - \gamma/2) = (\alpha - \eta + 1)/2$. The analysis is greatly simplified by the following observation:

Observation 1. *An upper bound on the running time of the algorithm is obtained when the intersections are detected at nodes with the lowest possible level number.*

Proof. For any given problem instance with n and m edges, the number of intersections is a given number k. We set the slack parameter s so that the bound on the *preprocessing* time to set up the envelope data structures at each node and the bounds on the time of a single query depend only on the total number of intersections to be discovered and on the position of the node in the tree. Consequently, s will not depend on the distribution of the intersections over the tree. On the other hand, any intersection that is detected is traced down to the leaves of the tree. As we observed before, the query time decreases with the level number, therefore we obtain an upper bound on the cost under the assumption that intersections are detected at the level with the lowest number. Equivalently, we can assume that there is a threshold level $\bar{i}$ such that for $i < \bar{i}$ every query results in a detection of an intersection, and for $i \geq \bar{i}$ no new intersection is detected and we keep on tracing the intersections detected before. □

We call the levels with $i < \bar{i}$ *satured*, and levels $i \geq \bar{i}$ *nonsaturated*. We assume for the time being that we know that the total number of intersections $k = N_v^{\beta}$ for a parameter β in the range $1 < \beta < 2$. On the saturated levels $k_{(i,j)} = n_v \leq N_v$. Let us define the real number $\bar{\alpha}$ such that $2^{\bar{i}} = N_v^{1-\bar{\alpha}}$. We have that the number of intersections k satisfies $2^{\bar{i}-1}n_v \leq k \leq 2^{\bar{i}}n_v$ and therefore $k = \Theta(N_v^{1-\bar{\alpha}}N_v)$. After easy calculations we have that $\bar{\alpha} \approx 2 - \beta$. We obtain bounds correct within constant multiplicative factors by considering the levels with $\alpha \geq 2 - \beta$ saturated, and those with $\alpha < 2 - \beta$ nonsaturated.

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

$$\sum_{j=1}^{l(i)} n_v m_v^{(\alpha-\gamma+1)/2} \leq N_v^{1-\alpha+1+(\alpha-\eta+1)/2} = N_v^{(5-\eta-\alpha)/2}. \tag{11}$$

The total cost is given by summing (11) 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 $O(\log_2 m_v)$ levels. We obtain a bound $O(N_v^{(\beta+3-\eta)/2} \log_2 m_v)$ on the cost of the queries on saturated levels.

Nonsaturation case. When we reach the last saturated level no new intersections are discovered. The cost of each query decreases when the level number increases. The preprocessing cost is N_v^{η} at each level. Therefore the total query cost of each nonsaturated level is bounded by the cost of the last saturated level. The cost for all the queries on the nonsaturated levels is $O(N_v^{(\beta+3-\eta)/2} \log_2 m_v)$. Now we choose η to balance the preprocessing and query costs. Setting $\eta = (\beta + 3 - \eta)/2$ we obtain $\eta = \beta/3 + 1$. Summarizing the above discussion, and taking into account the effect of ε factors, the cost of tracing the intersection on slack envelope structures is $O(N_v^{4/3+\varepsilon} + k^{1/3}N_v^{1+\varepsilon})$.

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 preprocessing of the envelope structures. 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: $k_u = 2k_{u-1}$. The number of guesses is $O(\log_2 N_v)$. If we exceed the time allowed by the bound we stop and start again doubling our estimate. For the first and the second terms of the bound this extra logarithmic factor is absorbed by the n^ε factor. For the third term we note that the guessed values for k_u form a geometric progression. Therefore the sum of the terms is proportional to the last term in the summation, which in turn is no more than twice the actual number of intersections k. We obtain the bound stated in Theorem 5, by summing over the whole primary tree $\mathscr{T}$ the bound for a single node v. □

6. Conclusions

We have shown some combinatorial bounds on the complexity of sets of lines missing polyhedral sets in 3-space. We have applied one of 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. Many natural questions on lines in 3-space are still left unanswered. For example, which is the complexity of $F(Q)$ and $I(Q)$ for Q a simply 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 these combinatorial bounds effectively for solving algorithmic problems on polyhedral objects in 3-space.

References

1. P. K. Agarwal. On stabbing lines for convex polyhedra in 3d. Technical Report CS-1993-09, Department of Computer Science, Duke University, Durham, NC 27706, April 1993.
2. P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 517–526, 1992.
3. P. S. Alexandrov. *Combinatorial Topology*. Graylock Press, Rochester, NY, 1956.
4. B. Aronov, J. Matoušek, and M. Sharir. On the sum of squares of cell complexities in hyperplane arrangements. *Proceedings of the 7th ACM Symposium on Computational Geometry*, pages 307–313, 1991.
5. B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. Lines in space: combinatorics, algorithms, and applications. *Proceedings of the 21st ACM Symposium on Theory of Computing*, pages 382–393, 1989.
6. B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. Algorithms for bichromatic line-segment problems and polyhedral terrains. *Algorithmica*, 11(2):116–132, 1994. Also Technical Report UIUCDCS-R-90-1578, Department of Computer Science, University of Illinois at Urbana Champaign, IL, 1990.
7. B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10(3):229–249, 1990.
8. K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2:195–222, 1987.
9. H. Edlesbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, New York, 1987.
10. H. Edelsbrunner, L. Guibas, and M. Sharir. The complexity and construction of many faces in arrangements of lines and segments. *Discrete Comput. Geom.*, 5:161–196, 1990.
11. D. Halperin and M. Sharir. New bounds for lower envelopes in three dimensions, with applications to visibility in terrains. *Proceedings of the 9th ACM Symposium on Computational Geometry*, pages 11–18, 1993.
12. J. Matoušek. Construction of ε-nets. *Discrete Comput. Geom.*, 5:427–448. 1990.
13. J. Matoušek. Reporting points in half-spaces. *Comput. Geom. Theory Appl.*, 2(3):169–186, 1992.

14. N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. Assoc. Comput. Mach.*, 31(1):115–126, 1984.
15. B. K. Natarajan. On planning assemblies. *Proceedings of the 4th ACM Symposium on Computational Geometry*, pages 299–308, 1988.
16. O. Nurmi and J. R. Sack. Separating a polyhedron by one translation from a set of obstacles. In J. van Leeuwen, editor, *Proceedings of the Workshop on Graph-Theoretic Concepts in Computer Science*, pages 202–212. Lecture Notes in Computer Science, volume 344. Springer–Verlag, Berlin, 1988.
17. D. Nussbaum and J. R. Sack. Translation separability of polyhedra. *Abstracts of the First Canadian Conference on Computational Geometry*, *Montreal*, page 34, 1989.
18. M. Pellegrini. Ray shooting on triangles in 3-space. *Algorithmica*, 9:471–494, 1993.
19. M. Pellegrini. On collision-free placements of simplices and the closest pair of lines in 3-space. *SIAM J. Comput.*, 23(1):133–153, 1994. Preliminary version in *Proceedings of the 8th ACM Symposium on Computational Geometry*, pages 130–137.
20. M. Pellegrini and P. Shor. Finding stabbing lines in 3-space. *Discrete Comput. Geom.*, 8:191–208, 1992.
21. F. P. Preparata and M. I. Shamos. *Computational Geometry: an Introduction.* Springer-Verlag, New York, 1985.
22. D. M. H. Sommerville. *Analytical Geometry of Three Dimensions.* Cambridge University Press, Cambridge, 1951.
23. J. Stolfi. Primitives for computational geometry. Technical Report 36, Digital Systems Research Center, 130 Lytton Avenue, Palo Alto, CA, 1989.
24. G. Toussaint. Movable separability of sets. In G. Toussaint, editor, *Computational Geometry*, pages 335–375. North-Holland, Amsterdam, 1985.

Received March 31, 1993, and in revised form December 23, 1993.