Ray-Shooting on Triangles in 3-Space[1]

M. Pellegrini[2]

Abstract

We present a uniform approach to problems involving lines in 3-space. This approach is based on mapping lines in R^3 into points and hyperplanes in 5-dimensional projective space (*Plücker space*). We obtain new results on the following problems:

1. Preprocess n triangles so as to efficiently answer the query: "Given a ray, which is the first triangle hit?" (*Ray-shooting problem*). We discuss the ray-shooting problem for both disjoint and non-disjoint triangles.
2. Construct the intersection of two non-convex polyhedra in an output sensitive way with a *subquadratic* overhead term.
3. Construct the arrangement of n intersecting triangles in 3-space in an output-sensitive way, with a *subquadratic* overhead term.
4. Efficiently detect the first face hit by any ray in a set of axis-oriented polyhedra.
5. Preprocess n lines (segments) so as to efficiently answer the query "Given two lines, is it possible to move one into the other without crossing any of the initial lines (segments)?" (*Isotopy problem*). If the movement is possible produce an explicit representation of it.

Key Words: Computational Geometry, Ray-shooting on triangles, Arrangements of hyperplanes, 3-space, Plücker coordinates, Isotopy classes.

1 Introduction

1.1 Ray-shooting in graphics

Ray shooting (also called ray tracing) is a central problem in graphics. Given the internal representation of a polygonal 3-dimensional scene and

[1]Research supported by NSF grant CCR-8901484, and by Eni and Enidata within the AXL project. Results presented in this paper have appeared in preliminary form in [Pel90] and [Pel91b].

[2]New York University, Courant Institute. Present address: King's College, Dept. of Computing, Strand, London WC2R 2LS, U.K. e-mail: m.pellegrini@oak.cc.kcl.ac.uk

some light-sources, a good image rendering requires that the amount of light received by every pixel is computed. If the scene comprises reflecting surfaces the problem is complicated by the effect of the reflected light. The basic geometric content of image-rendering is computing ray/surfaces intersections. This is known to be an expensive computation and many methods are used in graphics to speed up the computation [Gla89, SML88, AK87].

A widely used approach is the substitution for complex objects of a hierarchy of objects forming a chain of inclusions [KK86, Rot82]. The external object is very simple (a cube or a sphere) and checking the ray/object intersection takes $O(1)$ time. If the ray hits the external object the next object in the hierarchy is tested. This method gives a quick way to ignore many rays that do not hit the original objects. The hierarchical approach is only heuristic and there is no gain in terms of worst case asymptotic complexity. A main drawback of the hierarchy method is that those rays that are close to but do not hit any object in the scene are not filtered out quickly and discarded.

A second approach exploits the coherence of rays. Rays are grouped into "beams" and the behavior of the beams with respect to the surfaces is considered. Arvo and Kirk [AK87] propose to represent a ray as a point in a 5-dimensional space (three parameters give the source point, two angles the direction) and a beam as an hypercube in such space. Most of the computations, though, are carried out in the original 3-dimensional space since the scene itself is not mapped in the 5-dimensional space. This is a sharp difference with respect to our *Plücker coordinate method* (Section 2.1), since we map both the polygonal scene and the rays in a 5-dimensional space.

The approach to ray tracing that is relevant to this paper is that one that considers the objects as formed by many patches of plane surfaces and tries to minimize the number of patches to test against one ray. In graphics, a second general approach [Gla89] is used: objects are modeled by few and complex analytic surfaces and the objective is to compute efficiently the intersection point of one ray with one such surface. Here we study the asymptotic complexity of the the ray-shooting problem and our results have no claim of practicality.

1.2 Summary of previous work

In computational geometry usually the ray-shooting problems are stated with some restrictions on the rays or on the polygonal scene. Also, it is important to distinguish the on-line case from the off-line case, when all rays are known in advance, because of the speed-up we can get in the latter case.

A *terrain* is a set of polygons in 3-space having a unique intersection point with any vertical line. When the polygonal scene is a terrain of complexity n, on-line ray queries can be answered in $O(\log^2 n)$ time with $O(n^{2+\epsilon})$ preprocessing time and storage [CEGS89].

If we have a terrain and we force the source of the ray to lie on a given vertical line the method of Cole and Sharir [CS86] allows us to answer on-line queries in $O(\log^2 n)$ time after $O(n^2 2^{\alpha(n)} \log n)$ preprocessing, using $O(n^2 2^{\alpha(n)})$ storage, where $\alpha(n)$ is a functional inverse of the Ackerman function and is constant for any practical purpose.

When the source of the rays is on a given *non-vertical* line but no restriction is placed on the polygonal scene, Bern, Dobkin, Eppstein and Grossman [BDEG90] build in $O((n^2+k)\log^2 n + p\log n)$ time a data structure of size $O(n^2+k)$ that answers ray-queries in $O(\log^2 n)$ time, where k is the number of *opaque topological changes* of the scene along the flight-path, p is the number of *transparent topological changes*, and, in general, $0 \leq k \leq p \leq n^3/3$. In the case of vertical line and a polygonal terrain Bern et al. [BDEG90] improve the query time of the Cole and Sharir method to $O(\log n)$.

When the rays have a unique source-point and they are known in advance, Overmars and Sharir [OS89] solve the problem with $O(n^{4/3}\log^{\gamma} n)$ time bound, for n triangles, n rays and a small constant γ. This variation of the ray-shooting problem has a strong relation with the much studied problem of hidden surface removal [OS89][RS88][Ber88] [PY89][OS90], which consists in removing from a polygonal scene all the surfaces and edges that are not visible from a (proper or improper) point.

On-line single source ray shooting queries can be solved in $O(\sqrt{n}\log n)$ time using $O(n\log n)$ space and $O(n^{3/2}\log^{\gamma} n)$ preprocessing by modifying the implicit planar point location method of Agarwal [Aga90a] [Aga90b].

Schmitt, Müller and Leister [SML88] consider the case when the objects are iso-oriented rectangles, they give several algorithms with different trade-off between query time and space. In the off-line case they use $O(m^{0.82}n^{0.59}\log^{O(1)}(n))$ time and $O(m\log^{O(1)}(m)+n)$ storage. On-line queries are solved in $O(\log^3 n)$ time with $O(n^3 \log^{O(1)}(n))$ preprocessing time and space. Their technique exploits heavily the orthogonal decomposability of the spatial problem involving iso-oriented rectangles. This method is clearly not extendible to general polygonal scenes.

One of the applications of ray-shooting presented in this paper is for computing the intersection of two polyhedra in 3-space. The intersection of two *convex* polyhedra in R^3 of total complexity n is computed in time $O(n \log n)$ by an algorithm of Muller and Preparata [MP78]. Chazelle [Cha89] gives a worst case optimal $O(n)$ algorithm. The case when only one of the two polyhedra in convex is treated in [MS85, Sha88]. When the two polyhedra are terrains with the same vertical direction Chazelle et alt. [CEGS89] give an $O(n^{1.5+\epsilon} + k \log^2 n)$ time algorithm for computing the upper envelope of the two terrains, where k is the output size and n the total input size, for any $\epsilon > 0$.

1.3 Summary of results

For the most general version of the problem involving *any polygonal scene* and *any set of rays* only the trivial $O(mn)$ bound were known before the results presented in this paper and appeared in preliminary form in [Pel90] and [Pel91b]. We give off-line ray shooting algorithms with a *substantially sublinear cost per ray, for any set of triangles and rays* (Section 5).

The size of the data structure used in this paper to answer ray-shooting queries in logarithmic time depends on the total complexity of all cells of a 5-dimensional arrangement of hyperplanes cut by a second degree algebraic surface Π, called the *Plücker hypersurface*. This set of cells is also called the *zone* of Π in the arrangement of hyperplanes $\mathcal{A}(\mathcal{H})$ (denoted as $Z_\Pi(\mathcal{H})$). Since the zone of Π can have $\Omega(n^4)$ cells in the worst case, this gives a lower bound for the method too. In [Pel90] the estimate for the complexity of $Z_\Pi(\mathcal{H})$ is $O(n^5)$, later improved in [Pel91a] to $O(n^{4.669})$. Recently Aronov, Matoušek and Sharir [AMS91] have found the bound $O(n^{4.5})$. Finally, Aronov and Sharir [APS91] have found an $O(n^4 \log n)$ upper bound. Since the logarithmic factor is going to be dominated by other components of the method, this bound matches the lower bound for the algorithmic uses presented in this paper. For a set of n disjoint triangles, a data structure of size $O(n^{4+\epsilon})$ is given in Section 3 to answer on-line ray-shooting queries in $O(\log n)$ time. In Section 4 we solve the off-line version of the ray-shooting problem for disjoint triangles. In Section 5 the ray-shooting problem for intersecting triangles is considered.

The methods used to solve the ray-shooting problems are applicable to a wide range of 3-dimensional problems. In Section 4 we give an output sensitive method to compute the intersection of two non-convex polyhedra in time $O(n^{8/5+\epsilon} + K \log K)$ where n is the number of vertices, edges and facets of the two polyhedron and K the size of their intersection. In Section 5.1 we give an output sensitive method for constructing an arrangement of triangles in 3-space in $O(n^{8/5+\epsilon} + K \log K)$ time, where K is the output size. The interest of these two results lies in the *subquadratic* overhead. At my knowledge no previous subquadratic overhead was know for these two problems. An $O(n^2)$ overhead is trivially achieved by comparing any pair of features. Also, we sketch an algorithm that in $O(n^{8/5+\epsilon})$ expected time counts all pairs of intersecting lines.

When the query line is mapped into a point and we locate this point in a special cell complex, we have the so called *primal approach*. Ray-shooting problems can be cast also in a *dual approach* where the queries are mapped into hyperplanes, and we ask for the points in a point set that are above the query hyperplane (see [CSW90, AS91]). Using the dual approach Agarwal and Sharir [AS91] obtain an algorithm to solve decision line shooting queries that uses $O(m)$ storage and answers the queries in time $O(n^{16/15+\epsilon}/m^{4/15})$, for any $\epsilon > 0$ and $n^{1+\epsilon} \leq m \leq n^{4+\epsilon}$. As a lemma for their space/query trade-off result, Agarwal and Sharir [AS91] have a results similar to one presented in this paper for on-line ray-shooting on intersecting triangles.

A special case of particular interest is when the polygonal scene is formed by axis-oriented polyhedra. In Section 7 we give an $O(n^{2+\epsilon})$ storage and $O(\log n)$ query time method for this case. Independently, de Berg, Halperlin, Overmars, Snoeyink and van Kreveld [dBHO$^+$91] have obtained a similar (but more complicated) result for ray-shooting on axis-oriented boxes. They obtain a similar (but more complicated) result for ray-shooting on intersecting triangles (Theorem 3).

In Section 8 we discuss the *isotopy problem*: given n blue lines and two red lines decide whether the two red lines are in the same isotopy class (i.e. we can move continuously one into the other without crossing any blue line). McKenna and O'Rourke in [MO88] give an $O(n^4)$ storage and $O(n)$ query time method to solve this problem. Moreover, they use $O(n^4)$ elementary moves to find the actual movement of the two lines. At the best of my knowledge no better results were known for this problem. Here we present a method that uses $O(n^{4+\epsilon})$ storage and answers queries in $O(\log n)$ time. We use at most $O(n^{2+\epsilon})$ elementary moves to move one line into another, provided such movement is possible. These are sharp improvements over the bounds in [MO88]. We also extend the isotopy query data structure to deal with isotopy classes generated by sets of segments and polyhedra. In this case we have $O(n^{4+\epsilon})$ storage and $O(\log n)$ query time, but the movement consists of $O(n^{4+\epsilon})$ elementary moves in the worst case.

In Section 2 we give a survey of the geometric, combinatorial and algorithmic facts which are used throughout the paper.

2 Geometric preliminaries

A finite set H of hyperplanes in R^d define a decomposition of R^d into cells of various dimensions, which we call the arrangement $\mathcal{A}(H)$ of H [Ede87]. If $|H| = n$ the maximum number of cells in $\mathcal{A}(H)$ is $O(n^d)$ and the arrangement $\mathcal{A}(H)$ can be computed in optimal $O(n^d)$ time [EOS86]. One d-dimensional cell of $\mathcal{A}(H)$ is bounded by $O(n^{\lfloor d/2 \rfloor})$ cells of any dimension [Ede87].

Given a random sample R of H, with $|R| = r \leq n$, let us consider the arrangement $\mathcal{A}(R)$. A triangulation $\triangle\mathcal{A}(R)$ is a subdivision of each cell of $\mathcal{A}(R)$ into simplices such that the vertices of each simplex are vertices of $\mathcal{A}(R)$. The number of simplices in $\triangle\mathcal{A}(R)$ is $O(r^d)$. The random sampling theory of Clarkson [Cla87] states that with high probability the interior of each simplex $s \in \triangle\mathcal{A}(R)$ does not meet more than $O(n/r \log r)$ hyperplanes of H.

Given H we can build a data structure for locating the cell of $\mathcal{A}(H)$ which uses $Cn^{d+\epsilon}$ storage, for each $\epsilon > 0$, where the constant C depends on ϵ, such that a query point is located in $O(\log n)$ time [Cla87]. This data structure is built in time $O(n^{d+\epsilon})$ with high probability [Cla87].

For $d = 2$ Matoušek [Mat90] gives a deterministic method that, for a parameter $r < n$ subdivides the plane into $O(r)$ triangles in time $O(nr)$ such that the interior of each triangle meets only n/r lines in H. The results of Clarkson and Matoušek are the base of many divide-and-conquer solutions in computational geometry.

Geometric duality [Ede87, EMP$^+$82] is a pair of functions, one mapping points to hyperplanes and one mapping hyperplanes to points in R^d. Duality mappings preserve incidence and order relations. It is often convenient to transform a problem into its dual problem because this transformation preserves many important properties and for the dual problem it can be easier to find a solution.

A polyhedron P in R^3 with n facets, edges and vertices can be stored in a data structure $D(P)$ devised by Dobkin and Kirkpatrick [DK90]. $D(P)$ uses $O(n)$ storage, and can be built in $O(n \log n)$ time. Using $D(P)$, in $O(\log n)$ time it is possible to find the facets of P met by a query line. Intuitively, $D(P)$ is a hierarchy of finer and finer approximations of P. The interaction of a query line is traced down the hierarchy at a constant cost for each level.

2.1 Plücker Coordinates of lines

To solve ray-shooting on triangles with any orientation we use the *Plücker coordinates* of lines. Algorithmic uses of Plücker coordinates can be found in [CEGS89] and in [Pel91a, PS91]; a classical treatment of Plücker coordinates can be found in [Som51].

A point in 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: $x = x_1/x_0$ $y = x_2/x_0$ $z = x_3/x_0$. Two points $x = (x_0, x_1, x_2, x_3)$ $y = (y_0, y_1, y_2, y_3)$ in 3-dimensional homogeneous coordinates define a line l in 3-space. The six quantities

$$\xi_{ij} = x_i y_j - x_j y_i \quad for\ ij = 01, 02, 03, 12, 23, 31$$

are called *Plücker coordinates* of the line l (oriented from x to y). They correspond to the two-by-two minors of the two-by-four matrix formed by the coordinates of the point x (on the first row) and y (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 $\mathcal{F}_4^2$ [Sto89, Som51]):

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

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 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: $\pi : l \to \pi(l)$ 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. $p : l \to p(l)$ 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 [Sto89].

In this paper we use the notation $D(a, b, c, d)$ for the determinant formed by the coordinates of the points a, b, c, d placed on the rows in this order from top to bottom.

2.2 Characterization of line-triangle hits using Plücker coordinates

Let T be a set of n triangles. The set lines spanning edges of the triangles is denoted by by $\mathcal{L}_T$. $\mathcal{H}_T = \{\pi(l) \mid l \in \mathcal{L}_T\}$ is the set of hyperplanes in $\mathcal{P}^5$ associated with T. $\mathcal{A}(\mathcal{H}_T)$ is the *arrangement* formed by the hyperplanes in $\mathcal{H}_T$. An arrangement of Plücker hyperplanes will be called generically a *Plücker arrangement*. In this section we prove the following lemma.

Lemma 1 *Given a set of triangles T, and the arrangement $\mathcal{A}(\mathcal{H}_T)$ of the Plücker hyperplanes corresponding to lines spanning edges of T, for each cell c of $\mathcal{A}(\mathcal{H}_T)$ and any two lines l_1 and l_2, if $p(l_1) \in c$ and $p(l_2) \in c$ then l_1 and l_2 intersect the same subset of triangles in T.*

Proof. Given the point a and the triangle t in 3-space, we define as the cone $\mathcal{C}_{a,t}$ the set of rays from a intersecting t (see Figure 2). Let P be a plane below triangle t, and let $\mathcal{C}_{q,t}$ be the cone based on triangle $t = (p_1, p_2, p_3)$ with apex $q \in P$. Any two vertices of t together with q define a plane; for each such plane, the halfspace containing the third vertex of t is called *positive*.

When q belongs to the plane spanned by t the cone degenerates in a 2-dimensional object, but for simplicity of exposition we ignore degenerate cases. It is easy to check that the set of points in the rays belonging to the cone $\mathcal{C}_{q,t}$ is the intersection of the three positive halfspaces determined by q and t (see Figure (2)). If t is parallel to the reference plane P, then a variable point Q in the cone $\mathcal{C}_{q,t}$ satisfies the following system of linear inequalities:

$$\begin{aligned} D(q, p_1, p_2, Q) &\geq 0 \\ D(q, p_2, p_3, Q) &\geq 0 \\ D(q, p_3, p_1, Q) &\geq 0 \end{aligned} \tag{4}$$

If necessary relabel the vertices of t to ensure the inequalities all have the same sign.

When t is not parallel to P we must consider the position of apex q with respect to the line l_t, which is the intersection of P with the spanning plane of t (we will denote this plane with *aff* (t)). The cone is then defined by the two systems of linear inequalities (5) and (6), according to the position of q relative to l_t:

$$\begin{aligned} D(q, p_1, p_2, Q) &\geq 0 \\ D(q, p_2, p_3, Q) &\geq 0 \\ D(q, p_3, p_1, Q) &\geq 0 \end{aligned} \tag{5}$$

or

$$\begin{aligned} D(q, p_1, p_2, Q) &\leq 0 \\ D(q, p_2, p_3, Q) &\leq 0 \\ D(q, p_3, p_1, Q) &\leq 0. \end{aligned} \tag{6}$$

We need two systems of inequalities because when the apex q crosses l_t the three positive halfspaces switch with the non-positive ones. In Figure (3), system (5) is valid for apex q, system (6) for apex q'.

Using the *row exchange rule* of determinants and changing the sign of the inequalities accordingly we can put all the determinants in the systems (4), (5) and (6) in the form $D(p_1, p_2, q, Q)$, $D(p_2, p_3, q, Q)$, and $D(p_3, p_1, q, Q)$. Then we expand those determinants according to minors of the first 2 rows and of the last two, and obtain a linear expression in terms of Plücker coordinates. Note that the final systems involve the Plücker coordinates of lines supporting edges of triangles in T and the (variable) line passing through q and Q (see Figure (1)).

For any line l the set of triangles in T stabbed by l is determined by the relative position of $p(l)$ with respect to hyperplanes in $\mathcal{H}_T$. ∎

3 Ray-shooting on disjoint triangles.

Theorem 1 *Given any set T of n disjoint triangles in 3-dimensional space, there is a data structure $D(T)$ that uses $O(n^{4+\epsilon})$ storage and reports the first triangle hit by any ray ρ in $O(\log n)$ time. $D(T)$ can be built in $O(n^{4+\epsilon})$ randomized expected time.*

Proof. We consider a ray ρ as composed of two elementary objects: the source point p_ρ and the line l_ρ spanning the ray. The general strategy we use to solve ray-shooting on-line queries is to find an implicit representation of the ordered list of triangles intersected by l_ρ, and to perform a binary search over this list of the source point p_ρ.

Given the set T, we consider the set $\mathcal{L}(T)$ of lines spanning edges in T and the corresponding set $\mathcal{H}_T$ of Plücker hyperplanes. We take a random sample R of $\mathcal{H}_T$ of size r and we create the zone of Π in $\mathcal{A}(R)$. This can be done, for example, by constructing the whole arrangement $\mathcal{A}(R)$ with the method of Edelsbrunner, O'Rourke and Seidel [EOS86] [Ede87] in $O(r^5)$ time and space. Within the same time bound we can check which cells of $\mathcal{A}(R)$ intersect the Plücker surface Π and we record one Plücker point in each useful cell. From [APS91] we know that the zone of Π has complexity $O(r^4 \log r)$ and it is decomposable into $O(r^4 \log r)$ simplices. As follows from the random sampling theory of [Cla87], by repeated sampling, we can make sure that only $O(n/r \log r)$ of the original hyperplanes intersect any simplex. Thus, for all triangles none of whose three corresponding hyperplanes cut the simplex s, we can detect the unique subset stabbed by all Plücker points in s. For $O(n/r \log r)$ triangles having at least one hyperplane cutting through s, we repeat the same construction recursively in the simplex s.

The main difficulty in obtaining a reporting ray-shooting algorithm at this point is the fact that a single cell in the zone of Π could contain Plücker points with different stabbing orders (see Appendix A). Therefore, we do not have yet the linear order needed for a binary search of the source of the ray.

We get around this obstacle with the following observation. If we divide each cell of $Z_\Pi(R)$ into simplices, for each simplex s, $s \cap \Pi$ has a constant number of components. In constant time we compute the components of $\Pi \cap s$ using the general approach for computing topological properties of real algebraic manifolds of Schwartz and Sharir [SS83]. Additional computational details are given in [CS88] and [Cha85]. Furthermore, we can perform point location in the resulting data structure [SS83].

Since the Zone $Z_\Pi(R)$ has complexity $O(r^4 \log r)$ we have at most $O(r^4 \log r)$ components of $\Pi \cap Z_\Pi(R)$ to deal with. Assuming that the triangles in T are pairwise disjoint, each component corresponds to a unique stabbing order.

For each simplex we compute the components of $s \cap \Pi$ and the associated lists; then we recursively proceed in each simplex. The storage required satisfies this recurrence:

$$S(n) \leq cr^4 \log r S(\frac{n}{r} \log r) + O(nr^4 \log r)$$

For r constant, we obtain $S(n) = O(n^{4+\epsilon})$, but the query time is $O(\log^2 n)$. Next we show how to improve the query time to $O(\log n)$ while maintaining the same asymptotic bound on the storage.

Setting $r = n^{\epsilon'}$ we obtain an $O(n^{4+\epsilon})$ bound for $S(n)$, and the depth of the resulting search tree is constant. We use an auxiliary point-location date structure on the the sampled hyperplanes in order to locate the simplex s where the query point lies. The cell of $\mathcal{A}(R)$ where the query point lies is found using standard point location method in [Cla87] at the expense of $O(r^{5+\epsilon})$ additional storage. In order to locate the simplex within a cell we extend each facet of a simplex into an full hyperplane and obtain a point-location problem in a set of $O(r^2)$ hyperplanes in $\mathcal{P}^5$, which, using [Cla87], requires $O(r^{10+\epsilon})$ additional storage for each cell. This is a brute force method, but, for the purpose of proving the bound, any polynomial preprocessing and logarithmic query time algorithm will do. The additional storage is dominated by the n^ϵ factor in the final bound.

To summarize, it is possible to spend time logarithmic in r to find the simplex s where the query point lies, at the expense of polynomial extra storage.

After locating the simplex we perform a binary search in each list associated with components of $\Pi \cap s$. In $O(\log n)$ time we select one triangle in each list. In constant time we determine the closest triangle and compare it with the answer to the recursive data structure associated with the simplex. The first triangle can be determined in overall $O(\log n)$ time. ∎

4 Off-line ray-shooting

In this section we apply a general method of [GOS89] and we transform an on-line query data structure into an algorithm to answer efficiently off-line queries. We describe the algorithm first, then its time analysis.

4.1 The algorithm

The general strategy is the following. When the number of rays is large compared to the number of triangles we use the on-line algorithm of Section 3. If there are many triangles compared to the number of rays we map, in a suitable way, the lines spanning triangles and the lines spanning the rays in Plücker space as Plücker points and Plücker hyperplanes. We have to test (implicitly) every point with every hyperplane. In order to do so we subdivide Plücker space into regions. In each region we have Plücker points completely contained in it, Plücker hyperplanes which cut through the region, and Plücker hyperplanes which do not cut through the region. The algorithm efficiently compares the points within the region with the Plücker hyperplanes which are outside the region. We use a recursive call to make comparisons of points and hyperplanes within a region. During this process the relative number of triangles decreases with respect to the number of rays and we reach the base case of the algorithm. The application of this schema is complicated by the presence of the non-linear Plücker surface Π.

Given m rays and n triangles, if $m \geq n^{4+\epsilon}$ we build the on-line ray-shooting data structure of Section 3 and we obtain an $O(m \log n)$ overall method. If $m \leq n^{4+\epsilon}$ we dualize the problem. The lines spanning edges of a triangle are mapped into three Plücker points and the line l_ρ, spanning a query ray ρ is mapped into a Plücker hyperplane. We select a random sample of r Plücker hyperplanes and we form its *canonical triangulation* [Cla88]. We can make sure during the process that we produce simplices having at least one vertex. We obtain $M = O(r^4 \log r)$ simplices covering the zone of Π. Each simplex is cut by $O(\frac{n}{r} \log r)$ hyperplanes [Cla87]. Now we consider every triple of simplices in turn. The number of triples is $O(M^3)$, which is constant since r will be chosen constant.

Let $\sigma = (\sigma_1, \sigma_2 \sigma_3)$ be a triple of simplices, N_σ be the set of triangles whose corresponding Plücker points are in σ, and n_σ be the cardinality of N_σ. We take all hyperplanes missing σ and we consider only those hyperplanes above σ and those below σ. These two classes represent lines stabbing N_σ, as follows easily from the discussion in Section 2.2. We repeat the following argument for each such class of hyperplanes. Let M_σ be a class of hyperplanes (without loss of generality the class of hyperplanes above σ), and m_σ its cardinality (with M^*_σ we denote the corresponding set of dual points). We want to compare the m_σ rays against the n_σ triangles in time $O((m_\sigma + n_\sigma) \log n_\sigma)$.

A correct way to proceed, but too expensive, is the following: we dualize back again, returning to the primal space. We have n_σ Plücker hyperplanes and m_σ Plücker points. Since we already know that all Plücker points represent lines stabbing N_σ and have all the same relative position we can intersect the halfspaces supported by the hyperplanes and containing the Plücker points. We obtain a polytope Q_σ of size $O(n_\sigma^2)$. We can triangulate Q_σ and apply the query technique explained in Section 3. The problem with this approach is that Q_σ is too big an object to get a good time complexity.

A better solution is the following. We dualize back to the primal space the *vertices* of the three simplices in σ (which are not necessarily Plücker points), obtaining a constant number of hyperplanes. These hyperplanes have the same relative position with respect to the Plücker points M^*_σ, therefore we can intersect the corresponding halfspaces obtaining the polytope Q'_σ.

Lemma 2 *$M^*_\sigma \subset Q'_\sigma \subset Q_\sigma$, and every component of $Q'_\sigma \cap \Pi$ maps to one component of $Q_\sigma \cap \Pi$.*

Proof. $M^*_\sigma \subset Q'_\sigma$ is given by the fact that in the dual the hyperplanes M_σ were in the same relative position with the three simplices and in defining Q'_σ we have consistently chosen the one of the two halfspaces supported by an hyperplane dual to a vertex of σ.

Suppose now that exists a point $p \in Q'_\sigma$, $p \notin Q_\sigma$. In primal space this point corresponds to an hyperplane (not a Plücker hyperplane, though) that separates points in N_σ but it does not separates vertices of the simplices enclosing N_σ. This is obviously absurd.

Since $Q'_\sigma \subset Q_\sigma$ every point of $\Pi \cap Q'_\sigma$ is in $\Pi \cap Q_\sigma$. The only possible way we can violate the second part of the lemma is by merging in Q'_σ two components of $Q_\sigma \cap \Pi$. This is possible only if we have Plücker points in Q'_σ that are not in Q_σ, which is absurd. ∎

Since Q'_σ has constant complexity, we can apply the procedure for searching the source of the ray outlined in the Section 3. Lemma 2 ensures the correctness of the procedure. The total time used to compute the comparisons of rays in M_σ and lines in N_σ is $O((m_\sigma + n_\sigma)\log n_\sigma)$. This is an essential ingredient for applying the technique in [GOS89]. We repeat the computation sketched above for each triple σ and we are able to implicitly compute many comparisons. Then, we recurse the method in each simplex.

4.2 Time analysis

The time complexity $T(m, n)$ for solving m batched ray-shooting queries over n disjoint triangles satisfies the following recurrence:

$$T(m,n) \leq \begin{cases} am \log n & \text{for } m \geq n^{4+\epsilon} \\ \Sigma_{i=1}^{M} T(m_i, n_i) + cM^3 m \log n + cM^3 n \log n & \text{for } m < n^{4+\epsilon} \end{cases} \tag{7}$$

The additional constraints are:

$$\begin{aligned} &\Sigma_{i=1}^{M} n_i \leq 3n \\ &m_i \leq c' m \log r / r \text{ for some constant } c' \end{aligned} \tag{8}$$

The following theorem states the time complexity we are aiming for.

Theorem 2 *The ray shooting problem for m rays and n disjoint triangles can be solved in*

$$[Dm^{4/5-\delta} n^{4/5+4\delta+\epsilon'} \log^2 n + Am \log^2 n + Bn \log n \log m]$$

randomized expected time, for any $\delta > 0$ and $\epsilon' > 0$, where the coefficients A, B and D depend on δ and ϵ'. The storage is bounded by $O(m + n)$.

Proof. As for the storage, $O(n+m)$ space is used in one level of the algorithm and from one level to the next we need to store only one candidate triangle for each ray. The time bound follows from solving equation (7), this solution follows a schema in [EGS88].

Fix δ and choose $r = r(\delta)$ to be sufficiently large (how large will be determined later in the proof). If $m \geq n^{4+\epsilon}$ then $T(m,n) \leq am \log^2 n$ satisfies the bound assuming $A \geq a$. Suppose $m \leq n^{4+\epsilon}$. In this case

$$m = m^{4/5-\delta} m^{1/5+\delta} \leq m^{4/5-\delta} n^{4/5+4\delta+\epsilon'} \tag{9}$$

First notice that at each level of the recursion the third term in (7) contributes $O(n \log n)$ and there are at most $O(\log m)$ levels. The overall contribution of this term is $O(n \log n \log m)$. It is sufficient to drop this term from the recursion (7) and prove the modified inequality satisfies the bound

$$T(m,n) \leq D m^{4/5-\delta} n^{4/5+4\delta+\epsilon'} \log^2 n + Am \log^2 n$$

.

By induction hypothesis we have

$$T(m,n) \leq \Sigma_{i=1}^{M} [D m_i^{4/5-\delta} n_i^{4/5+4\delta+\epsilon'} + Am_i] \log^2 n_i + cM^3 m \log n$$

But $\Sigma_{i=1,M} m_i \log^2 n_i \leq [Mc' \log r/r] m \log^2 n$, therefore we can sum up two terms in m, put $d = AMc' \log r/r + cM^3$, and use the inequality (9), obtaining

$$T(m,n) \leq [D \Sigma_{i=1}^{M} m_i^{4/5-\delta} n_i^{4/5+4\delta+\epsilon'} + d m^{4/5-\delta} n^{4/5+4\delta+\epsilon'}] \log^2 n$$

Now we eliminate the summation using first the bound on m_i and then the Hölder-Minkowsky inequality [Mit70].

$$\begin{aligned} &\Sigma_{i=1}^{M} m_i^{4/5-\delta} n_i^{4/5+4\delta+\epsilon'} \leq [(c' \log r/r) m]^{4/5-\delta} \Sigma_{i=1}^{M} n_i^{4/5+4\delta+\epsilon'} \\ &\leq [c' \log r/r]^{4/5-\delta} m^{4/5-\delta} M^{1/5-4\delta-\epsilon'} n^{4/5+4\delta+\epsilon'} \end{aligned}$$

Then we obtain, for some constant c''

$$T(m,n) \leq [Dc'' \log^{4/5-\delta} r / r^{15\delta+4\epsilon'} + d] m^{4/5-\delta} n^{4/5+4\delta+\epsilon'} \log^2 n \tag{10}$$

If we choose r sufficiently large so that the constant in the Equation (10) is less than $[D/2 + d]$, and we choose $D = 2d$, we obtain

$$T(m, n) \leq Dm^{4/5-\delta} n^{4/5+4\delta+\epsilon'} \log^2 n$$

which proofs the asserted inequality. ∎

Using this result, when $m = n$ we can answer the line shooting queries at a randomized expected cost $O(n^{3/5+\epsilon})$ per ray. When $m = n^2$ the time cost is $O(n^{2/5+\epsilon})$ per ray. Easily we have these three corollaries:

Corollary 1 *Given two non-convex polyhedra of total complexity n there is a randomized algorithm that in expected time $O(n^{8/5+\epsilon})$ decides whether they intersect.*

Proof. Given two non convex polyhedra A and B, their intersection is not empty when either $A \subseteq B$ or $B \subseteq A$, or when an edge in one polyhedron intersects a face in the second one. Once we have an answer for the third case, it is easy in linear time to test the first two. This third case dominates the time to test disjointness. We modify the method for batched ray-shooting into a method for batched segment shooting in the following way. When we have retrieved for each line spanning a segment the list of stabbed triangles we perform a binary search with both segment end-points, checking if any triangle separates them. Having this batched segment shooting method, we set in turn the faces of one polyhedron as the triangles and the edges of the second one as the query segments. The time bound follows easily from Theorem 7. ∎

The intersection of two non-convex polyhedra of total complexity n can have complexity $\Omega(n^2)$. Therefore the naive algorithm that compares every edge of one polyhedron with every face of the second one to detect the new vertices is worst-case optimal. The more sophisticated approach of the next corollary gives an output sensitive algorithm with a subquadratic overhead.

Corollary 2 *Given two non-convex polyhedra A and B of total complexity n, the intersection $A \cap B$ can be computed in time $O(n^{8/5+\epsilon} + K \log K)$, where K is the size of the intersection.*

Proof. Using the algorithm of Theorem 2, modified for segment shooting as in Corollary 1, we can detect every vertex of $A \cap B$ generated by an edge of A and a face of B and vice versa in time $O(n^{8/5+\epsilon} + k)$, where k is the number of detected intersections. Each such intersection is a vertex of $A \cap B$ and the total complexity K of $A \cap B$ is $O(k+n)$. We construct the intersection using a straightforward tracing procedure (see [MS85, Sha88]. We need to be a bit careful about the representation of the polyhedra. Let us suppose to have A (resp. B) as an incidence graph whose nodes are vertices, edges and facets of A (resp. B). We look for an equivalent representation of $A \cap B$. Once we have all vertices of $A \cap B$, which are not vertices of A or B, it is easy to find all the new edges formed by intersecting a face of A with a face of B or an edge of A with two faces of B or an edge of B and two faces of A. By ordering the new vertices along each edge of A and B we can construct the subgraph of $A \cap B$ incident to any new vertex. We traverse the graphs of A and B to complete the construction. It is easy to see that after ordering the new vertices of $A \cap B$, each search operation takes $O(\log K)$ time using suitable data structures. ∎

Corollary 3 *Given a set T of n triangles we can test their pairwise disjointness in time $O(n^{8/5+\epsilon})$.*

Proof. Partition T into two sets of equal size and test them separately. If they are both pairwise disjoint, use the batched ray-shooting method to detect intersections across the two sets using the edges one set as queries on the other set. Repeat exchanging the roles of the two sets. ∎

5 Ray shooting on intersecting triangles

Let us consider again the data structure of Theorem 1. In every cell σ of the Plücker arrangement we have an associated set of triangles T_σ such that every line dual to a Plücker point in σ meet every triangle of T_σ. It follows that extending the triangles of T_σ into full planes does not introduce new intersections. We construct the 3-dimensional arrangement of planes spanning T_σ, we process it for point location and each cell of the arrangement for fast polyhedral intersection [DK90]. The total cost is $O(n^{3+\epsilon})$ time and space. It is easy now to locate the source of the ray in the 3-dimensional arrangement and find the first plane hit along the query ray. The space and time function satisfies the following equation:

$$S(n) \leq c_1 r^4 \log r S(\frac{n}{r} \log r) + O(n^{3+\epsilon} r^4 \log r)$$

The solution is $S(n) = O(n^{4+\epsilon})$, setting $r = n^{\epsilon'}$, where $\epsilon' < \epsilon$. Using the same arguments as in the proof of Theorem 1, the query time is $O(\log n)$.

Theorem 3 *Given any set of n (in general non-disjoint) triangles T in 3-dimensional space, there is a data structure $D(T)$ that uses $O(n^{4+\epsilon})$ storage and reports the first triangle hit by any ray ρ in $O(\log n)$ time. $D(T)$ can be built in $O(n^{4+\epsilon})$ expected time.*

5.1 Off-line ray-shooting on intersecting triangles

The batching technique of [GOS89, EGS88] allows us to to divide recursively the problem into subproblems in which every line in the set of lines M_σ, of size m_σ, intersect all the triangles in the set N_σ of size n_σ. We solve the ray-shooting problem restricted to N_σ and M_σ according to the following strategy.

1. If $m_\sigma \geq n_\sigma^{4+\epsilon}$, using the method of Theorem 3 we solve the problem in time $O(m_\sigma \log n_\sigma)$.
2. If $n_\sigma^{3+\epsilon} \leq m_\sigma \leq n_\sigma^{4+\epsilon}$, we consider, for each triangle in N_σ its spanning plane. We build the arrangement of all planes spanning triangles in N_σ, and we process this arrangement for point location and ray-shooting [DK90]. The total time spent for this preprocessing is $O(n_\sigma^{3+\epsilon})$. The total time used for answering the ray-shooting queries is $O(m_\sigma \log n_\sigma)$.
3. If $m_\sigma \leq n_\sigma^{3+\epsilon}$, we solve the problem by dualizing in 3-space the rays into double wedges and the planes into points. A double wedge is bounded by the plane dual to the source of the ray and by the plane dual to the direction of the ray (an improper point). When we locate the points in the arrangement of the wedges we are implicitly detecting the planes hit by the ray. Just for the planes intersecting a *set of rays* we generate the upper and lower cell of their arrangement and compute the first triangle hit by any ray in time $O(n'_\sigma \log n'_\sigma + m'_\sigma \log n'_\sigma)$, where n'_σ is the number of planes intersected by m'_σ rays.

   Coming back to the original subproblem, the standard batching technique [GOS89][EGS88] gives us a solution

   $$T'(m,n) = O(m_\sigma^{3/4-\delta} n_\sigma^{3/4+3\delta+\epsilon'} \log^2 n_\sigma + n_\sigma \log n_\sigma \log m_\sigma + m_\sigma \log^2 n_\sigma)$$

The partitioning technique of [GOS89, EGS88] at the external level of our construction gives us the following recursive relation:

$$T(m,n) \leq \begin{cases} 0 & for\ m = 0 \\ m \log n & for\ m \geq n^{4+\epsilon} \\ \Sigma_{i=1,M} T(m_i, n_i) + M^3 T'(m,n) & for\ m \leq n^{4+\epsilon} \end{cases}$$

Finally, solving the recurrence similarly to the solution in Section 4, we have the following theorem:

Theorem 4 *Given n triangles in 3-space, which may intersect, and m rays we can find the first triangle hit by any ray in randomized expected time:*

$$[Dm^{4/5-\delta} n^{4/5+4\delta+\epsilon'} \log^2 n + Am \log^2 n + Bn \log n \log m]$$

for any $\delta > 0$, where the coefficients A, B and D depend on δ and ϵ'. The storage is bounded by $O(m+n)$.

Theorem 4 can be easily modified to report all intersections of rays or segments with the triangles. With this modification we can built an output sensitive algorithm to construct arrangements of triangles in 3-space. A trivial method to compute this arrangement would use $O(n^2 + K \log K)$ time, we improve on the overhead term.

Given n triangles in R^3 we use them as the triangles in our method an their edges as segments we shoot. In time $O(n^{8/5+\epsilon} + k)$ we find all the intersections of edges with triangles. Note that k intersections reported will be part of the final output. We use the segments cut on each triangle as input to a planar segment intersection reporting problem. If k_i is the number of segments on triangle t_i in $O(k_i \log k_i + K_i)$ time we can find all the intersections using the algorithm in [CE88], where K_i is the size of the contribution of t_i to the arrangement. Using the same approach as in Corollary 2 we complete the construction of the arrangement. Summing up over all triangles we obtain the time complexity claimed in the following theorem:

Theorem 5 *Given a set T of n triangles in R^3 we can build the 3-dimensional arrangement $\mathcal{A}(T)$ in time $O(n^{8/5+\epsilon} + K \log K)$, where K is the size of the arrangement.*

6 A Counting problem

Theorem 6 *Given n lines in 3-space it is possible to count the number of pairs of intersecting lines in expected time $O(n^{8/5+\epsilon})$, for any $\epsilon > 0$, where the constants depend on ϵ.*

Proof sketch. We partition the set of lines $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2$ into two sets of roughly equal size and we solve recursively the problem in each set. Moreover, we locate the Plücker points of lines in $\mathcal{L}_1$ in the arrangement of the Plücker hyperplanes of lines in $\mathcal{L}_2$. We use the batching technique to perform this step in $O(n^{8/5+\epsilon})$ expected time. In order to handle the Plücker points which are on the sampled Plücker hyperplanes we also solve the problem recursively in the dimension of the space containing the query points. We repeat exchanging the roles of $\mathcal{L}_1$ and $\mathcal{L}_2$. ∎

7 Ray shooting on axis-oriented polyhedra

In this section we give a solution to a simpler case of ray-shooting where the input objects are axis-oriented polyhedra (also called axis-oriented boxes). This case is relevant for the bounding boxes method mentioned in the introduction.

Theorem 7 *Given n axis-oriented boxes in 3-space there exists a data structure of size $O(n^{2+\epsilon})$ to answer ray shooting report queries in $O(\log n)$ time. The data structure can be built in $O(n^{2+\epsilon})$ time.*

Proof. Ray-shooting problems are easily decomposable. If $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ is the set of rectangles bounding the boxes, the answer for $\mathcal{R}_1$ and the answer for $\mathcal{R}_2$ can be combined in $O(1)$ time to give the answer for $\mathcal{R}$. We partition the facets of the boxes into three sets of axis-parallel parallel rectangles, and we solve the problem on each set separately.

Given an axis-oriented rectangle R parallel to the xy-plane, a line l intersects R if and only if the projections of the line l and of the rectangle R on the xz-plane intersect, and the same holds for projections on the yz-plane. The general strategy is to build a two-level data structure where each level is a data structure to answer queries about line stabbing of segments on the plane. Given n segments on the plane, we use standard duality in [EMP+82] obtaining n *double wedges*. Each double wedge is the region bounded by the lines dual to the segment end-points and not containing a vertical line. We use Matoušek technique [Mat90] over the lines bounding the wedges to partitions the plane into $O(r^2)$ triangles. Each triangle σ is covered by $O(n)$ of the original wedges and crossed by $O(n/r)$ of the wedges, as follows from [Mat90]. A query line on the plane dualizes on the dual plane into a point, which belongs to a region σ. The list of wedges covering the region σ corresponds to segments stabbed by the query line on the plane. We keep this list sorted in stabbing order. This order is well defined since the rectangles are parallel. We store these lists of wedges in space $S(n)$, which satisfies the recurrence: $S(n) \leq r^2 S(cn/r) + r^2 n$. The solution is $S(n) = O(n^{2+\epsilon})$ for $r = n^{1/4}$, and the depth of the recursion is constant. We use the above data structure as a primary tree to store information about projections on the xz-plane. At each node we take the list of covering wedges and consider the associate rectangles and their projections on the yz-plane as an input to a similar secondary data structure. The total space used to build the secondary and the main data structures is: $S(n) \leq r^2 S(cn/r) + r^2 n^{2+\epsilon}$. The solution is $S(n) = O(n^{2+\epsilon})$ for $r = n^{\epsilon'}$, with $\epsilon' < \epsilon$. The depth of the secondary tree is constant (depending on ϵ').

The query time on the secondary data structure satisfies this recurrence: $Q'(n) \leq Q'(n/r) + \log n$ which solves in $Q'(n) = O(\log n)$ because we chose $r = n^{\epsilon'}$. Similarly the query time on the primary tree is: $Q(n) \leq Q(n/r) + Q'(n) \leq Q(n/r) + O(\log n)$, which again gives an $O(\log n)$ total query time.

Since we choose r non constant, we must build planar point location structures on the dual plane to locate the cell containing the query point. These additional structures do not asymptotically modify the space or the time needed to construct the 2-level tree. ∎

Using the above data structure and using the batching technique it is possible to solve the batched ray-shooting problem of n rays and n axis-oriented boxes in time $O(n^{4/3+\epsilon})$.

With techniques in [CSW90] and [AS91] it is possible to trade off space and query time. Using $O(m)$ storage for $n^{1+\epsilon} \leq m \leq n^{2+\epsilon}$, we obtain query time $O(n^{1+\epsilon}/\sqrt{m})$.

8 Querying isotopy classes

In this section we adapt the techniques used in the Section 3 in order to solve the following problem: Given n blue lines in R^3, determine, for a pair of red lines, whether they belong to the same isotopy class. Two red lines are in the same isotopy class if we can move one into the other without crossing or become parallel to any blue line.

For this problem we give an $O(n^2 \log n)$ time algorithm. If the problem is asked in repetitive mode after $O(n^{4+\epsilon})$ preprocessing we can answer in $O(\log n)$ time.

We check in $O(n)$ time that the Plücker points of the two red lines have the same sign with respect to any of the blue Plücker hyperplanes. If this is not the case we answer negatively. Otherwise, we construct the cell C containing the two red Plücker points in the arrangement of the blue Plücker hyperplanes. From the Upper Bound Theorem [Ede87] we have an $O(n^2)$ bound on the complexity of C. Using Seidel's algorithm [Sei86] we construct C in deterministic time $O(n^2 \log n)$ (using a recent algorithm of Chazelle [Cha91] C can be computed in $O(n^2)$ deterministic time). We subdivide C into $O(n^2)$ simplices and we compute the components of $s \cap \Pi$ for each simplex s. For any pair of adjacent simplices s and s', if a component of $\Pi \cap s$ and a component of $\Pi \cap s'$ have a common boundary point we identify those two components in the same class. Using a union find data structure we can describe any connected components of $\Pi \cap C$ in C as the union of connected components of $\Pi \cap s$, over all simplices s). A connected component of $\Pi \cap C$ represents an isotopy class of lines. The second phase consists in locating the components of Π containing the two red points, and retrieving the associated isotopy class. The union find procedure takes $O(n^2 \alpha(n))$ time [Tar83] because we have only $O(n^2)$ pairs of adjacent facets in the triangulation of the cell, and therefore only $O(n^2)$ union operations.

Theorem 8 *Given a set $\mathcal{L}$ of n lines in R^3, we can determine if any 2 given lines, not necessarily in $\mathcal{L}$, are in the same isotopy class using $O(n^2 \log n)$ time and $O(n^2)$ storage.*

We define as an *elementary path* a path on Π connecting two points on one component of $\Pi \cap s$, where s is any simplex. It is easy to see that we can compute the movement that takes one red line into the other as the concatenation of at most $O(n^2)$ elementary paths.

Given two points on any component of $\Pi \cap s$ it is possible to compute an arc completely contained in the component and connecting the two points [SS83]. This means that we can effectively compute the elementary steps required by Theorems 8, 9 and 10.

For the repetitive case we adopt the random sampling approach and we build a search tree structure of size $O(n^{4+\epsilon})$ in the first phase. The second phase consists in forming the equivalence classes starting from the bottom of the search tree and identifying components sharing boundary points.

During a query (i) we locate the two query points in the Plücker arrangement, at the bottom of the decomposition tree, (ii) we determine the component of $\Pi \cap s$, where s is a simplex stored at a leaf, to which they belong, (iii) using the auxiliary union-find structure we determine the isotopy class.

Theorem 9 *Given a set $\mathcal{L}$ of n lines we can preprocess it into a data structure $D(\mathcal{L})$ of size $O(n^{4+\epsilon})$, so that, for any given pair of lines l_1 and l_2 in $O(\log n)$ time it is possible to decide whether l_1 and l_2 belong to the same isotopy class. $D(\mathcal{L})$ can be constructed in $O(n^{4+\epsilon})$ expected time. The path connecting the 2 lines, if it exists, comprises $O(n^{2+\epsilon})$ elementary paths.*

Let us consider a segment e in R^3. Let l_e be the line spanning e. Define $S_e = \{p(l) \mid l \cap e \neq \emptyset\}$. We observe that S_e is a connected semi-algebraic set of Plücker points. From this fact it is easy to show that given a simplex s on the Plücker hyperplane $\pi(l_e)$, $s \cap S_e$ has a constant number of connected components.

In order to decide whether two lines l_1 and l_2 are in the same isotopy class with respect to a set E of edges we build the same data structure of the previous section for the set $\mathcal{L}(E)$ of lines spanning E. During the second phase of the algorithm, though, we modify the rule for identifying components of Π. Given two components $\mathcal{C}_1$ and $\mathcal{C}_2$ we identify them only if they have a common point on their boundary not in $\bigcup_{e \in E} S_e$. Clearly, if two components are separated by the hyperplane $\pi(l_e)$ we need to check only S_e. The asymptotic complexity of each check is the same as before, that is constant for each union find operation. We summarize the above discussion with this theorem:

Theorem 10 *Given a set E of n segments in R^3 we can preprocess it into a data structure $D(E)$ of size $O(n^{4+\epsilon})$, so that, for any given pair of lines l_1 and l_2 in $O(\log n)$ time it is possible to decide whether l_1 and l_2 belong to the same isotopy class. $D(E)$ can be constructed in $O(n^{4+\epsilon})$ expected time. The path connecting two lines is the concatenation of $O(n^{4+\epsilon})$ elementary paths.*

A construction in [MO88] can be easily modified to exhibit a set of n disjoint segments in R^3 with $\Omega(n^4)$ isotopy classes of lines. This is an indication that the result of Theorem 10 is almost space optimal if a label for each isotopy class has to be stored explicitly. Theorem 10 does not exploit any special property of the set of edges E. It is conceivable that, by imposing a fixed bound on the length of each segment and on the minimum distance between two segments, the result of theorem 10 could be improved.

9 Conclusions

This paper gives algorithms for on-line and off-line ray-shooting problems on triangles in 3-space. The off-line ray-shooting algorithms have applications in computing intersections of non-convex polyhedra and arrangements of triangles with an output sensitive time bound. Also, we give the first fast isotopy test for lines among polyhedral obstacles.

The main ideas behind these results are the following.

1. Complex objects are determined by a collection of algebraically simpler objects. For example, a ray is formed by a point and line; a triangle is formed by three lines and a plane.

2. Incidence properties of complex objects are converted into boolean combinations of incidence and ordering conditions among simple objects.

3. Relations among simple objects are studied and computed in a characteristic space, which depends on the nature of the objects involved. For example, lines vs. lines problems are studied in Plücker space. Point vs. planes are studied in 3-space.

4. Using standard techniques (random sampling, construction of arrangements, duality, batched queries, multilevel data-structures) we are able to efficiently compute relations among many simple objects.

It is our belief that the approach of Plücker coordinates has a potential for applications to problems involving lines, segments and polyhedra in 3-dimensional space, which stretches beyond the algorithms presented in this paper.

10 Acknowledgments

Thanks to Richard Pollack, Micha Sharir, Janos Pach, Boris Aronov, Pankaj K. Agarwal and Peter Shor for useful discussions.

References

[Aga90a] P.K. Agarwal. Partitioning arrangements of lines I: An efficient deterministic algorithm. *Discrete & Computational Geometry*, 5:449–483, 1990.

[Aga90b] P.K. Agarwal. Partitioning arrangements of lines II: Applications. *Discrete & Computational Geometry*, 5:533–573, 1990.

[AK87] J. Arvo and D. Kirk. Fast ray tracing by ray classification. In M. C. Stone, editor, *SIGGRAPH '87 Conference Proceedings*, pages 55–63, 1987.

[AMS91] 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.

[APS91] B. Aronov, M. Pellegrini, and M. Sharir. On the zone of an algebraic surface in a hyperplane arrangement. *Discrete & Computational Geometry*, 1991. To appear.

[AS91] 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.

[BDEG90] M. Bern, D. Dobkin, D. Eppstein, and R. Grossman. Visibility with a moving point. In *Proceedings of the 1st ACM-SIAM Symposium on Discrete Algorithms*, 1990.

[Ber88] M. Bern. Hidden surface removal for rectangles. In *Proceedings of the 4th ACM Symposium on Computational Geometry*, pages 183–192, 1988.

[CE88] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. In *Proc. of the 29th Ann. Symp. on Foundations of Computer Science*, 1988.

[CEGS89] 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.

[Cha85] B. Chazelle. *Fast searching in a real algebraic manifold with applications to geometric complexity*, volume 185 of *Lecture notes in computer science*, pages 145–156. Springer Verlag, 1985.

[Cha89] 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.

[Cha91] B. Chazelle. An optimal convex hull algorithm and new results on cuttings. In *Proceedings of the 32th IEEE Symposium on Foundations of Computer Science*, pages 29–38, 1991.

[Cla87] K.L Clarkson. New applications of random sampling in computational geometry. *Discrete Computational Geometry*, (2):195–222, 1987.

[Cla88] K.L. Clarkson. A randomized algorithm for closest-point queries. *SIAM J. on Computing*, 17:830–847, 1988.

[CS86] R. Cole and M. Sharir. Visibility problems for polyhedral terrains. Technical Report 266, CIMS, December 1986.

[CS88] B. Chazelle and M. Sharir. An algorithm for generalized point location and its applications. Technical Report 153, Courant Institute of Mathematical Sciences, Robotics Lab., May 1988.

[CSW90] B. Chazelle, M. Sharir, and E. Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. In *Proceedings of the 6th ACM Symposium on Computational Geometry*, pages 23–33, 1990.

[dBHO+91] 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*, 1991.

[DK90] D. Dobkin and D. Kirkpatrick. Determining the separation of preprocessed polyhedra: a unified approach. In *Proc. of the 17th International Colloqium on Automata, Languages and Programming*, pages 400–413, 1990.

[Ede87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer Verlag, 1987.

[EGS88] H. Edelsbrunner, L. Guibas, and M. Sharir. The complexity of many faces in arrangements of lines and segments. In *Proceedings of the 4th ACM Symposium on Computational Geometry*, pages 44–55, 1988.

[EMP+82] H. Edelsbrunner, H. Mauer, F. Preparata, E. Welzl, and D. Wood. Stabbing line segments. *BIT*, (22):274–281, 1982.

[EOS86] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM Journal of Computing*, pages 341–363, 1986.

[Gla89] A.S. Glassner, editor. *Ray tracing.* Academic Press, 1989.

[GOS89] L. Guibas, M. Overmars, and M. Sharir. Ray shooting, implicit point location and related queries in arrangements of segments. Technical Report TR433, Courant Institute, 1989.

[KK86] T.L. Kay and J.T. Kajiya. Ray tracing complex scenes. *Computer Graphics*, 20:269–278, 1986.

[Mat90] J. Matoušek. Cutting hyperplane arrangements. In *Proceedings of the 6th ACM Symposium on Computational Geometry*, pages 1–9, 1990.

[Mit70] D.S. Mitrinovic. *Analytic inequalities.* Springer Verlag, 1970.

[MO88] M. McKenna and J. O'Rourke. Arrangements of lines in 3-space: A data structure with applications. In *Proceedings of* *the 4th annual Symposium on Computational Geometry*, pages 371–380, 1988.

[MP78] D.E. Muller and F.P. Preparata. Finding the intersection of two convex polyhedra. *Theoret. Comput. Sci.*, 7:217–236, 1978.

[MS85] K. Mehlhorn and K. Simon. Intersecting two polyhedra one of which is convex. In *Proc. of Fundamentals of Computation Theory*, number 199 in Lecture Notes in Computer Science, pages 534–542. Springer Verlag, 1985.

[OS89] M. Overmars and M. Sharir. Output-sensitive hidden surface removal. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, 1989.

[OS90] M.H. Overmars and M. Sharir. Merging visibility maps. In *Proceedings of the 6th ACM Symposium on Computational Geometry*, pages 168–176, 1990.

[Pel90] M. Pellegrini. Stabbing and ray shooting in 3 dimensional space. In *Proceedings of the 6th ACM Symposium on Computational Geometry*, pages 177–186, 1990.

[Pel91a] M. Pellegrini. *Combinatorial and Algorithmic Analysis of Stabbing and Visibility Problems in 3-Dimensional Space.* PhD thesis, New York University–Courant Institute of Mathematical Sciences, February 1991. Courant Institute, Robotics Lab. Report Number 241.

[Pel91b] M. Pellegrini. Ray shooting and isotopy classes of lines in 3-dimensional space. In *Proceedings of the 1991 Workshop on Algorithms and Data Structures*, number 519 in Lecture Notes in Computer Science, pages 20–31. Springer Verlag, 1991.

[PS91] M. Pellegrini and P. Shor. Finding stabbing lines in 3-dimensional space. In *Proceedings of the Second SIAM-ACM Symposium on Discrete Algorithms*, pages 24–31, 1991.

[PY89] M.S. Paterson and F.F. Yao. Binary partitions with applications to hidden surface removal and solid modelling. In *Proceedings of the 5th ACM Symposium on Computational Geometry*, pages 23–32, 1989.

[Rot82] S.D. Roth. Ray casting for modeling solids. *Computer Graphics and Image Processing*, 18:109–144, 1982.

[RS88] J.H. Reif and S. Sen. An efficient output-sensitive hidden-surface removal algorithm and its parallelization. In *Proceedings of the 4th ACM Symposium on Computational Geometry*, pages 193–200, 1988.

[Sei86] R. Seidel. Constructing higher-dimensional convex hulls at logarithmic cost per face. In *Proceedings of the 18th annual Symposium on Theory of Computing*, pages 404–413, 1986.

[Sha88] M. Sharir. The shortest watchtower and related problems for polyhedral terrains. *Information processing Letters*, 29:265–270, 1988.

[SML88] A. Schmitt, H. Muller, and W. Leister. Ray tracing algorithms - theory and practice. In R.A. Earnshaw, editor, *Theoretical Foundations of Computer Graphics and CAD*, pages 997–1030. NATO ASI, Springer Verlag, 1988.

[Som51] D. M. H. Sommerville. *Analytical geometry of three dimensions.* Cambridge, 1951.

[SS83] J.T. Schwartz and M. Sharir. On the piano mover's problem: II. General techniques for computing topological properties of real algebraic manifolds. *Adv. in Appl. Math*, 4:298–351, 1983.

[Sto89] J. Stolfi. Primitives for computational geometry. Technical Report 36, Digital SRC, 1989.

[Tar83] R.E. Tarjan. *Data Structures and Network Algorithms*, volume 44 of *CBMS-NSF Regional Conference Series in Applied Mathematics.* SIAM, 1983.

A Different geometric permutations in the same Plücker cell

Figure 4 shows two lines l' and l'' which have two different stabbing orders (also called *geometric permutations*) with respect to two intersecting segments on the plane. Lifting this two-dimensional construction into a 3-dimensional one we obtain two lines realizing two different geometric permutations, whose Plücker points are in the same cell of the Plücker arrangement.

Recalling the terminology of section 2.1, lines l' and l'' intersect the base line P in the same region of the partition induced on P by the lines spanning the segments t_1 and t_2, therefore, by results in subsection 2.2, we conclude that the two Plücker points of l' and l'' belong to the same cell of the Plücker arrangement.

We take the same two segments on the plane, we extend them in 3-space, and we transform them into thin triangles in space so to obtain on the base plane a map corresponding to the partition of the line P. Now we move t_1 and t_2 slightly apart so that they become disjoint, keeping all other features of the picture the same. Two stabbing lines with different stabbing orders, which on the plane must intersect, can be twisted so that they are disjoint and still stabbing the two triangles.

The next lemma shows which geometric feature of the set T affects the presence of different stabbing orders in the same cell. Given two triangles t_1 and t_2, let $\mathit{aff}(t_1)$ and $\mathit{aff}(t_2)$ be the two spanning planes and let $l_{12} = \mathit{aff}(t_1) \cap \mathit{aff}(t_2)$ be their intersection.

Lemma 3 *A necessary condition for two triangles t_1 and t_2 to have stabbing lines belonging to the same cell of $\mathcal{A}(\mathcal{H}_T)$ with different stabbing orders is that both t_1 and t_2 intersect l_{12}*

Proof. Let us consider the set of triangles $T = \{t_1, t_2\}$ and the planar map $\mathcal{M}$ generated on the reference plane P below T, by $\mathit{aff}(t_1)$ and $\mathit{aff}(t_2)$. A line l that intersects region σ of $\mathcal{M}$ and intersects T in the order (t_1, t_2), necessarily hits the plane $\mathit{aff}(t_1)$ before the plane $\mathit{aff}(t_2)$. This fact is equivalent to the fact that l is above (or below) the line l_{12}. Two lines l' and l'' realizing two different stabbing orders hit points belonging to the triangles above and below the line l_{12}, therefore, for convexity, both triangles intersect the line l_{12}. ∎

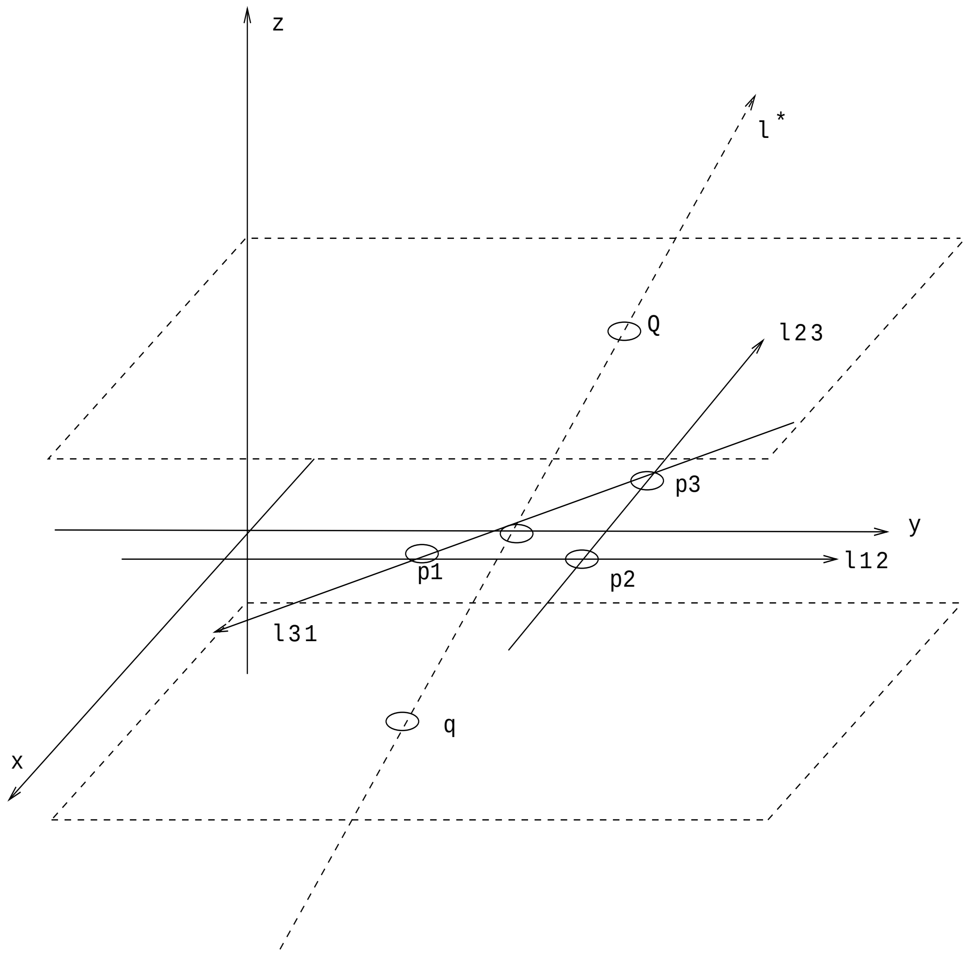

Figure 1: Stabbing line and triangle

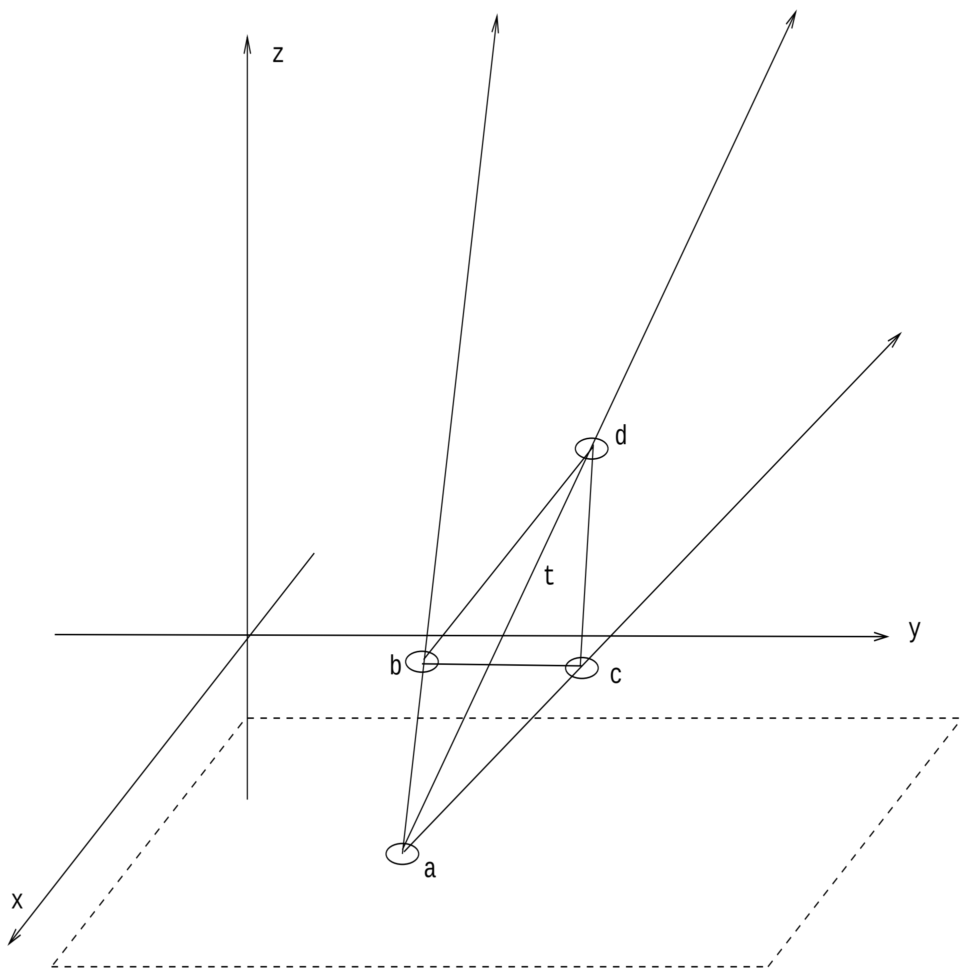

Figure 2: Cone with apex a based on t

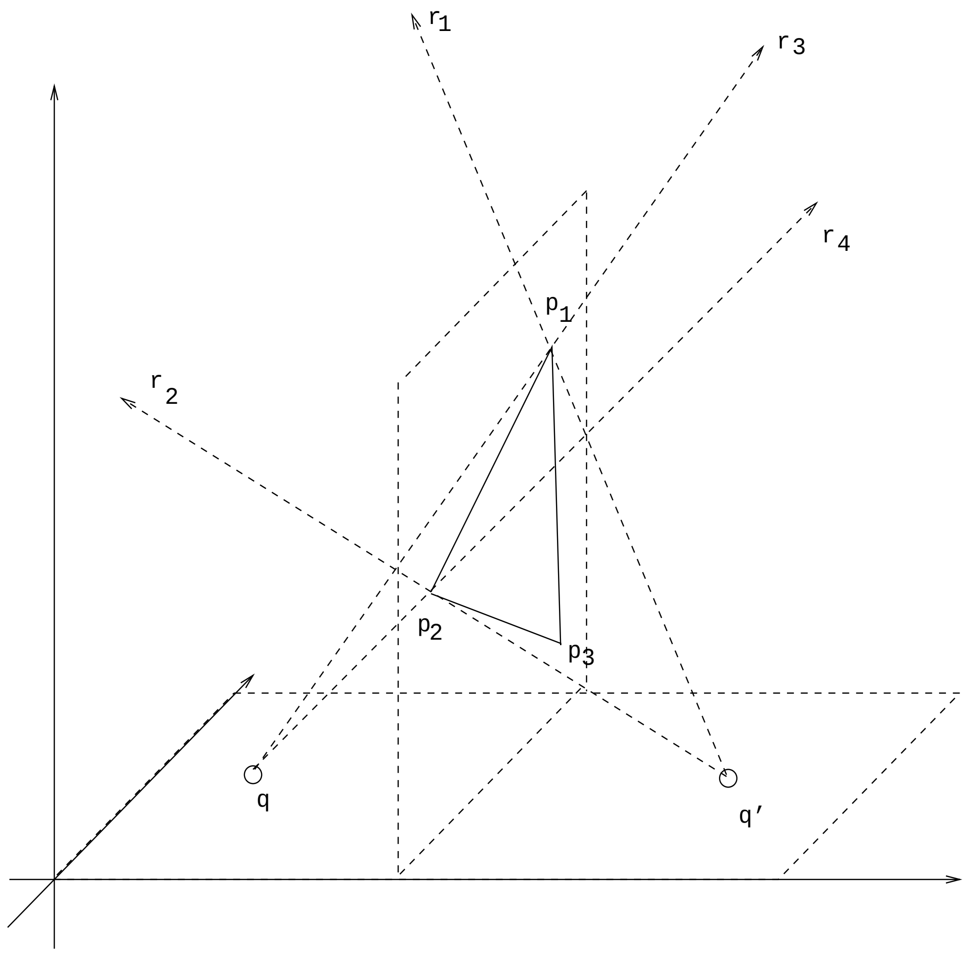

Figure 3: Different cones based at q and q'

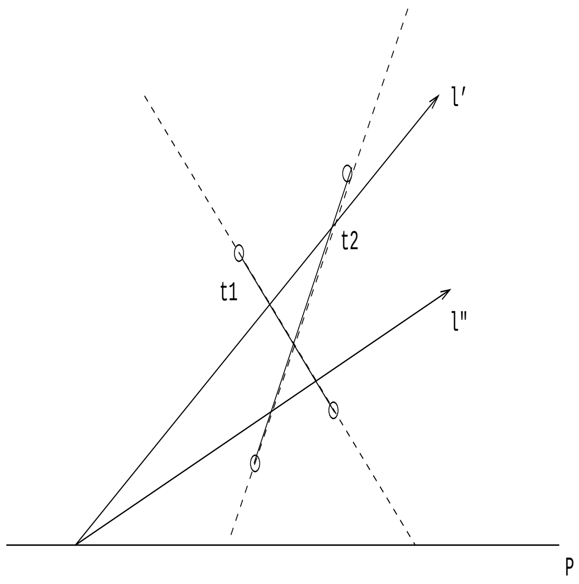

Figure 4: Two segments with the different stabbing orders from the same cell.