Lines Through Segments in Three Dimensional Space[*]

Efi Fogel[†] Michael Hemmer[†] Asaf Porat[†]
Dan Halperin[†]

Abstract

We present an efficient output-sensitive algorithm and its exact implementation to solve the following problem: Given a set $\mathcal{S}$ of n line segments in three-dimensional space, find all the lines that simultaneously intersect quadruples of line segments in $\mathcal{S}$. We refer to this problem as *the lines-through-segments problem*, or LTS for short. The algorithm properly handles all degenerate cases. Since we do not assume general position, we compute all lines that intersect *at least* four segments in $\mathcal{S}$. The algorithm runs in $O((n^3 + I) \log n)$ time, and requires $O(n + I)$ working space, where I is the output size; I is bounded by $O(n^4)$. We use CGAL arrangements and in particular its support for two-dimensional arrangements in the plane and on the sphere to efficiently compute the intersecting lines in an exact manner. We also report on the performance of our algorithm and its implementation compared to others. The source code of the LTS program as well as the input examples for the experiments can be obtained from `http://acg.cs.tau.ac.il/projects/lts`.

1 Introduction

LTS is a fundamental problem that arises in a variety of domains. For instance, solving the LTS problem can be used as the first step towards solving the more general problem of finding all lines tangent to four polyhedral objects taken from a set of polyhedral objects. The latter is ubiquitous in many fields of computation such as computer graphics (visibility computations), computational geometry (line transversal), robotics and automation (assembly planning), and computer vision.

The number of lines that intersect four lines in $\mathbb{R}^3$ is 0, 1, 2, or infinite. Brönnimann et al. [3] showed that the number of lines that intersect four arbitrary line segments in $\mathbb{R}^3$ is 0, 1, 2, 3, 4, or infinite. In addition, they showed that the lines lie in at most four maximal *connected components*.[1]

A straightforward method to find all the lines that intersect four lines, given a set of n lines, examines each quadruplet of lines using the Plücker coordinate representation. Hohmeyer and Teller [7] and Redburn [6] exploited this method. It was later used by Everett et al. [4] as a building block for the problem of finding line transversals (the set of lines that intersect all given line segments). Naturally, the running time of this method is $O(n^4)$.

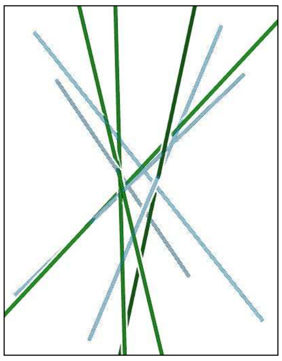


Figure 1: Four lines (drawn in green) that intersect four line segments (drawn in blue with a halftone pattern).

The combinatorial complexity of all the lines that intersect four line segments of a set of n line segments is $\Theta(n^4)$. The lower bound can be easily established by placing two grids of line segments in two parallel planes, respectively. However, in many cases the number of output lines is considerably smaller. The size of the output tends to be even smaller, when the input consists of line segments (as opposed to lines), which is typically the case in practical problems.

We present an efficient output-sensitive algorithm, and its complete and robust implementation that solves the LTS problem in three-dimensional Euclidean space. The implementation is complete in the sense that it handles all degenerate cases and guarantees exact results. Examples of degenerate cases are: A line segment may degenerate to a point, several segments may intersect, be coplanar, parallel, concurrent, lie on the same supporting line, or even overlap. To the best of our knowledge, this is the first algorithm (and implementation) for the LTS problem that is (i) output sensitive and (ii) handles all degenerate cases. The algorithm utilizes the idea of McKenna and O'Rouke [5] to represent the set of lines that intersect three lines as a rectangular hyperbola with vertical and horizontal asymptotes in $\mathbb{R}^2$. However, as opposed to their algorithm, which takes $O(n^4\alpha(n))$ time, our algorithm is output sensitive and its asymptotic time and space complexities are $O((n^3+I)\log n)$ and $O(n + I)$, respectively, where n is the input size

[*]This work has been supported in part by the 7th Framework Programme for Research of the European Commission, under FET-Open grant number 255827 (CGL—Computational Geometry Learning), by the Israel Science Foundation (grant no. 1102/11), by the German-Israeli Foundation (grant no. 969/07), and by the Hermann Minkowski–Minerva Center for Geometry at Tel Aviv University.

[†]School of Computer Science, Tel-Aviv University, 69978, Israel. efifogel@gmail.com, mhsaar@gmail.com, asafpor@gmail.com, danha@post.tau.ac.il.

[1]Two lines tangent to the same four line segments are in the same connected component iff one of the lines can be continuously moved into the other while remaining tangent to the same four line-segments.

and I is the output size; I is bounded by $O(n^4)$.

Our algorithm is implemented on top of the Computational Geometry Algorithm Library (CGAL) [8]. The implementation is mainly based on the *2D Arrangements* package of the library [10]. This package supports the robust and efficient construction and maintenance of arrangements induced by curves embedded on certain orientable two-dimensional parametric surfaces in three-dimensional space [2, 9], and robust operations on them.[2] The implementation uses in particular 2D arrangements of rectangular hyperbolas with vertical and horizontal asymptotes in the plane and 2D arrangements of geodesic arcs on the sphere [1].

2 Representation

In this section we discuss the encoding of all lines that intersect two fixed line segments, S_1 and S_2, and a third line segment S_3. However, due to space limitation, we only discuss the generic case, where the lines underlying the input line segments are pairwise disjoint and their direction vectors are linearly independent. Then, we give one simple non-generic case.

We represent a line $L \subset \mathbb{R}^3$ by a point $p \in \mathbb{R}^3$ and a direction $d \in \mathbb{R}^3 \setminus \{\mathcal{O}\}$ as $L(t) = p + t \cdot d$, where $\mathcal{O}$ denotes the origin and $t \in \mathbb{R}$. Clearly, this representation is not unique. A segment $S \subset L \subset \mathbb{R}^3$ is represented by restricting t to the interval $[a, b] \subset \mathbb{R}$. We refer to $S(0)$ and $S(a)$ as the source and target points, respectively. We denote the underlying line of a line segment S by $L(S)$. Two lines are *skew* if they are not coplanar. Three or more lines are *concurrent* if they all intersect at a common point.

Given two lines L_1 and L_2 we define a map $\Psi_{L_1L_2}$ as follows: $\Psi_{L_1L_2}(p) = \{(t_1, t_2) \in \mathbb{R}^2 \,|\, L_1(t_1),\ L_2(t_2),$ and p are collinear$\}$. That is, $\Psi_{L_1L_2}$ maps a point in $\mathbb{R}^3$ to a set in $\mathbb{R}^2$. This set, which might be empty, corresponds to all lines that contain p and intersect L_1 and L_2. Now consider the pair $(t_1, t_2) \in \mathbb{R}^2$. If $L_1(t_1) \neq L_2(t_2)$, then this pair uniquely defines the line that intersects L_1 and L_2 at $L_1(t_1)$ and $L_2(t_2)$, respectively. Thus, for skew lines L_1 and L_2 there is a canonical bijective map between $\mathbb{R}^2$ and all lines that intersect L_1 and L_2.

The characterization of $\Psi_{S_1S_2}(S_3) = \Psi_{L(S_1)L(S_2)}(S_3) \cap [0, 1]^2$ serves as the theoretical foundation of the algorithm that solves the LTS problem presented in Section 3. In the generic case, which we sketch in the next subsection, $\Psi_{S_1S_2}(S_3)$ consists of at most three arcs lying on a rectangular hyperbola with a vertical and a horizontal asymptote. In degenerate cases, which we omit here, the base curve may be a single line, or a pair of a horizontal and a vertical line. In some cases, e.g., S_1, S_2, and S_2 are coplanar, $\Psi_{S_1S_2}(S_3)$, is a two-dimensional region that is bounded by curves of the above kind.

[2]Arrangements on surfaces are supported as of CGAL version 3.4, albeit not documented yet.

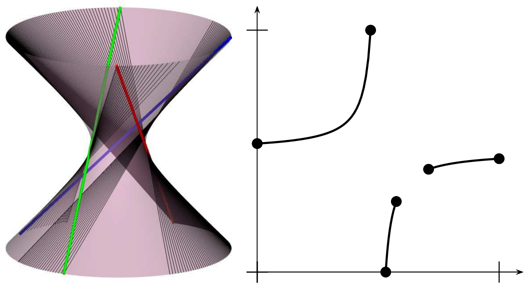


Figure 2: (a) Three surface patches the lines of which intersect three pairwise skew line segments, S_1, S_2, and S_3, in $\mathbb{R}^3$. These surface patches are contained in a hyperboloid of one sheet. (b) The point set $\Psi_{S_1S_2}(S_3)$.

If S_1 and S_2 intersect, it is impossible to encode the lines that go through the intersection point by $\Psi_{S_1S_2}(S_3)$. This case requires a special representation as discussed in Section 2.2.

2.1 Generic Case

We assume that the direction vectors of the underlying lines of the input line segments are linearly independent. Thus, we can apply a rational affine transformation, such that the three segments are given by $S_i(t_i) = p_i + t_i \cdot d_i,\ i \in \{1, 2, 3\}$, where $p_1 = (a, b, c)$, $p_2 = (d, e, f)$, $p_3 = \mathcal{O}$ and $d_i = e_i$ (where e_i denotes the unit vector along the ith axis).

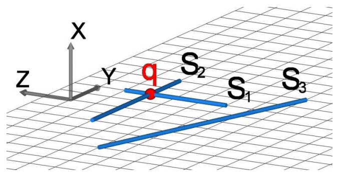
We further assume that the underlying lines are pairwise disjoint, which implies that $b \neq 0,\ d \neq 0$, and $c \neq f$. Consider the points $L_1(t_1)$, $L_2(t_2)$, and $L_3(t_3)$. These points are collinear iff $|(L_1(t_1) - L_2(t_2)) \times (L_3(t_3) - L_2(t_2))| = 0$. These are three dependent equations in three unknowns. Eliminating t_3 we obtain the following expression for t_2 in terms of t_1:

$$t_2(t_1) = \frac{e \cdot t_1 + (a \cdot e - d \cdot b)}{t_1 + a}\,. \qquad (1)$$

It implies that $\Psi_{L_1L_2}(L_3)$ is a rectangular hyperbola with a vertical asymptote and a horizontal asymptote. Nonetheless, we are interested in $\Psi_{S_1S_2}(S_3)$. Recall that all t_i, $1 \le i \le 3$, are restricted to $[0, 1]$. Similar to Equation 1, we can eliminate t_2 and solve for t_1 in terms of t_3. It is then easy to show that $\Psi_{S_1S_2}(S_3)$ consists of at most three maximal connected components, where each component represents a patch of a ruled surface as depicted in Figure 2.

2.2 S_1 and S_2 Intersect

Assume L_1 and L_2 intersect, and let $q = L_1(\tilde{t_1}) = L_2(\tilde{t_2})$ be the intersection point. The point $(\tilde{t_1}, \tilde{t_2})$ represents all lines that contain q. We represent



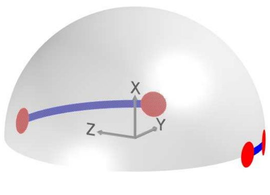


Figure 3: (a) Three line segments, S_1, S_2, and S_3, such that S_1 and S_2 intersect at q (and S_3 does not). (b) The mapping $\Xi_q(S_3)$, where $\Xi_q(S_3) = \{\Xi_q(p) \mid p \in S_3\}$, which consists of two geodesic arcs on $\mathbb{H}^2$.

these lines by points on a semi open upper hemisphere centered at q. We define the additional map $\Xi_q : \mathbb{R}^3 \setminus \{q\} \to \mathbb{H}^2$ and $\Xi_q(p) \longmapsto d = s(p-q)/|p-q|$, with $s \in \{\pm 1\}$, such that $d \in \mathbb{H}^2 = \{p \,|\, p \in \mathbb{S}^2$ and p is lexicographically larger than $\mathcal{O}\}$. In the generic case a segment S maps to one or two geodesic arcs on $\mathbb{H}^2$. If S_3 is a point, or $L(S_3)$ contains q and S_3 does not, $\Xi_q(S)$ consists of a single point. If $q \in S_3$, we define $\Xi_q(S_3) = \mathbb{H}^2$; see Figure 3.

3 The Algorithm

We describe the algorithm that handles only a reduced set of cases due to space limitation. Henceforth, we assume that none of the input line segments degenerates to a single point and no three line segments are concurrent or coplanar. The complete algorithm, which is omitted here, handles all cases.

The input is a set $\mathcal{S} = \{S_1, \ldots, S_n\}$ of n line segments in $\mathbb{R}^3$. The output is a set of at most $O(n^4)$ lines in $\mathbb{R}^3$, such that each line intersects four line segments in $\mathcal{S}$. The intersected line segments are provided as part of the output.

The idea is to process each pair, (S_i, S_j) with $i < j-2$, and construct the planar arrangement, $\mathcal{A}_{ij}$, induced by the set $\{\Psi_{S_i S_j}(S_k) \,|\, i < k < j\}$. As we limit ourself to the generic case here, $\mathcal{A}_{ij}$ is induced only by hyperbolic arcs. Each intersection point of two such arcs represents a line that intersects S_i and S_j as well as the two segments that are mapped through $\Psi_{S_i S_j}$ to the hyperbolic arcs. Considering only pairs of segments S_i and S_j with $i < j-2$ and constructing the corresponding arrangement using only the line segments of the set $\{S_k \,|\, i < k < j\}$, ensures that every output line is reported exactly once.

If S_i and S_j intersect, we also construct an arrangement of geodesic arcs on the half-sphere $\mathbb{H}^2$ centered at their intersection point. The arrangement is induced by the central projections of the other segments on the sphere. Again, an intersection of two geodesic arcs encode lines that intersect four segments.

Using the sweep line algorithm, each arrangement, $\mathcal{A}_{ij}$, is constructed in $O(n \log n + I_{ij})$ time, where I_{ij} in the number of intersections in $\mathcal{A}_{ij}$. Since there are $O(n^2)$ arrangements, the total runtime is $O(n^3 \log n + I)$. As only one arrangement must be retained at a time, the required storage space is $O(n \log n + J)$, where J is the maximum number of intersections in a single arrangement. J is bounded by $O(n^2)$.

We remark that the complete algorithm, which is implemented, is necessarily more involved. In particular, in some degenerate cases $\Psi(S_i, S_j)(S_k)$ is a two dimensional patch that is bounded by hyperbolic arcs and line segements. In these cases we use the overlay algorithm of CGAL to add the two dimensional patches to the arrangement generated by the sweep.

4 Experimental Results

We have conducted several experiments on three types of data sets. The first comprises random input. The second produces the worst-case combinatorial output and has many degeneracies. The third consists of transformed versions of the former and has many near-degeneracies. We report on the time consumption of our implementation, and compare it to the implementation of J. Redburn [6]. All experiments were performed on a Pentium PC clocked at 2.40 GHz.

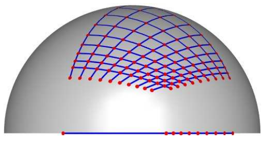
4.1 Random Input

Input	Time		Lines
	LTS	Red	
Short	1.06	300.4	0
Medium	2.82	314.0	20,742
Long	5.15	327.0	64,151

The Random data set consists of 50 line segments drawn uniformly at random. In particular, the endpoints are selected uniformly at random within a sphere. We experimented with three different radii, namely, **Short**, **Medium**, and **Long** listed in increasing lengths. We verified that the line segments are in general position. The table above shows the number of output lines and the time in seconds it took to perform the computation using our implementation, referred to as **LTS**, and an instance of a program developed by J. Redburn [6] that relies on unlimited precision, referred to as **Red**. One can clearly observe how the time consumption of our implementation decreases with the decrease of the output size, which in turn decreases with the decrease in the line-segment lengths. Adversely, the time consumption of Redburn's implementation hardly changes.

4.2 Grid

The Grid data set, which attains the maximal number of output lines, comprises 40 line segments arranged in two grids of 20 lines segments each lying in two planes parallel to the yz plane. Each grid consists of ten vertical and ten horizontal line segments. Thus, each plane contains 100 intersection points, which implies that the output includes exactly $100^2 = 10,000$ lines, each containing one intersection point of each plane. Due to the degenerate nature of the input, the output also includes several planar patches each lying in one of the two planes that contain the input grids, which we also compute but not elaborate on any further here. Using our implementation it took 20.74 seconds to compute the output. Unfortunately, Redburn's implementation could not handle this case.



We remark that every output line is represented by a vertex of an arrangement on the sphere. The figure to the right depicts one sample arrangement on the sphere centered at the intersection point of two line segments, S_1 and S_{40}, lying in the same plane. The arrangement is induced by the set of geodesic arcs $\{\Xi_{S_1 \cap S_{40}}(S_i) \,|\, i = 2, \ldots, 39\}$.

4.3 Transformed Grid

We conducted three additional experiments using a transformed version of the Grid data set. First, we slightly perturbed the input line segments, such that every two line segments became skew and the directions of every three line segments became linearly independent (input **I**). Secondly, we translated the (perturbed) horizontal line segments of one grid along the plane that contains this grid (input **II**), increasing the distance between the (perturbed) vertical and horizontal line segments of the grid. This drastically reduced the number of output lines. We then applied a similar translation to the other plane (input **III**), further reducing the size of the output. Table 1 shows the number of output lines and the time it took to perform the computation using our implementation. The output sensitivity of our algorithm is prominent. The table also shows the time it took to perform the computation using two instances of the program developed by Redburn. One instance, relies on a number type with unlimited precision, while the other resorts to double-precision floating-point numbers. As expected, when limited precision numbers were used, the output was only an approximation. Notice that the influence of the output size on the time consumption of Redburn's implementation is negligible.

Table 1: Perturbed Grid. Time is measured in seconds.

Input	**Unlimited Prec.**			**Double Prec.**	
	Time		**Lines**	**Time**	**Lines**
	LTS	**Red**		**Red**	
I	23.72	140.17	12,139	0.70	12,009
II	11.83	132.80	5,923	0.69	5,927
III	6.90	128.80	1,350	0.70	1,253

5 Acknowledgement

We thank Michael Hoffmann for helpful discussions on assembly partitioning, which inspired us to conduct the research discussed in this article. We also thank Linqiao Zhang who provided us with Redburn's code that was used for the experiments. Zhang used it as part of an implementation of an algorithm that constructs the visibility skeleton [11].

References

[1] E. Berberich, E. Fogel, D. Halperin, M. Kerber, and O. Setter. Arrangements on parametric surfaces II: Concretizations and applications. *Math. in Comput. Sci.*, 4:67–91, 2010.

[2] E. Berberich, E. Fogel, D. Halperin, K. Mehlhorn, and R. Wein. Arrangements on parametric surfaces I: General framework and infrastructure. *Math. in Comput. Sci.*, 4:45–66, 2010.

[3] H. Brönnimann, H. Everett, S. Lazard, F. Sottile, and S. Whitesides. Transversals to line segments in three-dimensional space. *Disc. Comput. Geom.*, 34:381–390, 2005. 10.1007/s00454-005-1183-1.

[4] H. Everett, S. Lazard, W. Lenhart, J. Redburn, and L. Zhang. On the degree of standard geometric predicates for line transversals. *Comput. Geom. Theory Appl.*, 42(5):484–494, 2009.

[5] M. McKenna and J. O'Rourke. Arrangements of lines in 3-space: a data structure with applications. In *Proc. 4th Annu. ACM Symp. Comput. Geom.*, pages 371–380, New York, NY, USA, 1988. ACM Press.

[6] J. Redburn. *Robust computation of the non-obstructed line segments tangent to four amongst n triangles.* PhD thesis, Williams College, Massachusetts, 2003.

[7] S. Teller and M. Hohmeyer. Determining the lines through four lines. *j. of graphics, gpu, and game tools*, 4(3):11–22, 1999.

[8] The CGAL Project. *CGAL User and Reference Manual.* CGAL Editorial Board, 3.9 edition, 2011. `http://www.cgal.org/Manual/3.9/doc_html/cgal_manual/title.html`.

[9] R. Wein, E. Fogel, B. Zukerman, and D. Halperin. Advanced programming techniques applied to CGAL's arrangement package. *Comput. Geom. Theory Appl.*, 38(1–2):37–63, 2007. Special issue on CGAL.

[10] R. Wein, E. Fogel, B. Zukerman, and D. Halperin. 2D arrangements. In *CGAL User and Reference Manual.* CGAL Editorial Board, 3.9 edition, 2011. `http://www.cgal.org/Manual/3.9/doc_html/cgal_manual/packages.html#Pkg:Arrangement2`.

[11] L. Zhang, H. Everett, S. Lazard, C. Weibel, and S. Whitesides. On the size of the 3D visibility skeleton: Experimental results. In *Proc. 16th Annu. Eur. Symp. Alg.*, volume 5193/2008 of *LNCS*, pages 805–816, Karlsruhe Allemagne, 2008. Springer.