[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. Incidence and nearest-neighbor problems for lines in 3-space. Manuscript, November 1991.

[Pel91c] 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.

[PM75] M.H. Protter and C.B. Morrey. *Analytic Geometry.* Addison Wesley, 1975.

[Sha91] M. Sharir. Personal communication. November 1991.

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

[SS89] J.T. Schwartz and M. Sharir. A survey of motion planning and related geometric algorithms. In D. Kapur and J.L. Mundy, editors, *Geometric reasoning*, pages 157–169. The MIT press, 1989.

[Tol91] S. Toledo. Extremal polygon containment problems. In *Proceedings of the 7th ACM Symposium on Computational Geometry*, pages 176–185, 1991.

[vKOA90] M. van Kreveld, M. Overmars, and P. K. Agarwal. Intersection queries in sets of discs. In *Proceedings of SWAT '90*, number 447 in Lecture Notes in Computer Science. Springer Verlag, 1990.

[EGS90b] H. Edelsbrunner, L. Guibas, and M. Sharir. The complexity of many cells in arrangement of planes and related problems. *Discrete & Computational Geometry*, 5:197–216, 1990.

[FHS89] J. Friedman, J. Hershberger, and J. Snoeyink. Compliant motion in a simple polygon. In *Proceedings of the 5th ACM Symposium on Computational Geometry*, pages 175–186, 1989.

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

[GP91] L. J. Guibas and M. Pellegrini. Problems on lines and spheres in 3-space. Manuscript, November 1991.

[GPW90] J.E Goodman, R. Pollack, and R. Wenger. Geometric transversal theory. In preparation, 1990.

[GSS88] L.J. Guibas, M. Sharir, and S. Sifrony. On the general motion planning problem with two degrees of freedom. In *Proceedings of the 4th ACM Symposium on Computational Geometry*, pages 289–298, 1988.

[Gui91] L. Guibas. Personal communicationion. November 1991.

[Hal91] D. Halperlin. On the complexity of a single cell in certain arrangements of surfaces in 3-space. In *Proceedings of the 7th ACM Symposium on Computational Geometry*, pages 314–323, 1991.

[HO89] D. Halperlin and M. Overmars. Efficient Motion Planning of an L-shaped Object. In *Proceedings of the 5th ACM Symposium on Computational Geometry*, pages 156–166, 1989.

[KS88] K. Kedem and M. Sharir. An automatic motion planning system for a convex polygonal mobile robot in 2-d polygonal space. In *Proceedings of the 4th ACM Symposium on Computational Geometry*, pages 329–340, 1988.

[LS87a] D. Leven and M. Sharir. An efficient simple motion planning algorithm for a ledder moving in two-dimensional space amidst polygonal barriers. *J. of Algorithms*, 8:192–215, 1987.

[LS87b] D. Leven and M. Sharir. Planning a purely translational motion for a convex object in two-dimensional space using generalizaed voronoi diagrams. *Discrete & Computational Geometry*, 2:9–31, 1987.

[Mat91] J. Matousek. Efficient partition trees. In *Proceedings of the 7th ACM Symposium on Computational Geometry*, pages 1–9, 1991.

[Meg83] N. Megiddo. Applying parallel computation algorithms in the design of sequential algorithms. *J. of ACM*, 30:852–865, 1983.

[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.

[CEG+90a] B. Chazelle, H. Edelsbrunner, L. Guibas, R. Pollack, R. Seidel, M. Sharir, and J. Snoeyink. Counting and cutting circles of lines and rods in space. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, 1990.

[CEG+90b] K.L. Clarkson, H. Edelsbrunner, L.J. Guibas, M. Sharir, and E. Welzl. Combinatorial complexity bounds for arrangements of curves and spheres. *Discrete & Computational Geometry*, 5:99–160, 1990.

[CEGS89a] 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.

[CEGS89b] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. A singly exponential stratification scheme for real semi-algebraic varieties and its applications. In *Proceedings of the 16th International Colloquium on Automata, languages and Programming*, number 372 in Lecture Notes in Computer Science, pages 179–193. Springer Verlag, 1989.

[CK89] L.P. Chew and K. Kedem. Placing the largest similar copy of a convex polygon among polygonal obstacles. In *Proceedings of the 5th ACM Symposium on Computational Geometry*, pages 167–174, 1989.

[Cla88] K. Clarkson. A las vegas algorithm for linear programming when the dimension is small. In *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science*, pages 452–456, 1988.

[Col87] R. Cole. Slowing down sorting networks to obtain faster sorting algorithms. *J. of ACM*, 34:200–208, 1987.

[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.

[DGK63] L. Danzer, B. Grunbaum, and V. Klee. Helly's theorem and its relatives. In *Proceedings of Symposia in Pure Mathematics*, pages 100–180, 1963.

[EGS90a] H. Edelsbrunner, L. Guibas, and M. Sharir. The complexity and construction of many faces in arrangements of lines and segemnts. *Discrete & Computational Geometry*, 5:161–196, 1990.

3. In many applications of the incidence counting algorithms, we do not need the full power of a counting oracle. For example, to find the minimum point-line distance, it is sufficient to have an oracle to detect whether the number of intersections is greater than 0. Using this observation, can we improve the time bound of the algorithm for finding the minimum point-line distance?

4. The problem of finding efficiently stabbing lines for spheres in 3-space is still quite open (see [GPW90, DGK63] for a comprehensive survey).

References

[AASS90] P.K. Agarwal, B. Aronov, M. Sharir, and S. Suri. Selecting distances in the plane. In *Proceedings of the 6th ACM Symposium on Computational Geometry*, pages 321–331, 1990.

[Aga90] P.K. Agarwal. Partitioning arrangements of lines II: Applications. *Discrete & Computational Geometry*, 5:533–573, 1990.

[Aga91] P.K. Agarwal. Improved ray-shooting trade-off. Personal communication, 1991.

[AM91] P.K. Agarwal and J. Matoušek. Ray shooting and parametric search. Manuscript. To appear in STOC '92, June 1991.

[AS90] B. Aronov and M. Sharir. Triangles in space or building (and analyzing) castels in the air. *Combinatorica*, 10(2):137–173, 1990.

[AS91a] 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.

[AS91b] P. K. Agarwal and M. Sharir. Counting circular arc intersections. In *Proceedings of the 7th ACM Symposium on Computational Geometry*, pages 10–20, 1991.

[AS91c] B. Aronov and M. Sharir. On the complexity of a single cell in an arrangement of $(d-1)$-simplices in d-space and its applications. Manuscript, 1991.

[AS91d] B. Aronov and M. Sharir. On the zone of a surface in a hyperplane arrangement. In *Proceedings of the 1991 Workshop on Algorithms and Data Structures*, number 519 in Lecture Notes in Computer Science, pages 13–19. Springer Verlag, 1991.

[AW88] D. Avis and R. Wenger. Polyhedral line transversals in space. *Discrete & Computational Geometry*, (3):257–265, 1988.

[Bol78] B. Bollobas. *Extremal graph theory.* Academic Press, 1978.

[BR79] O. Bottima and B. Roth. *Theoretical Kinematics.* North-Holland, 1979.

C) $lhp(e) \notin S$ *and* $rhp(e) \notin S$ *and* $c(S) \in W(e)$ *and* $S \cap l(e) \neq \emptyset$ *imply* $e \cap S \neq \emptyset$.

D) $e \cap S \neq \emptyset$ *implies that one and only one of* A, B *and* C *is true.*

Proof. Follows from elementary geometry. ∎

Following [AM91] we can solve a ray-shooting problem using Megiddo's technique in conjunction with an algorithm to test segments emptiness within a set of spheres. The segment emptiness algorithm is an intersection counting algorithm composed of three different tests.

1. First, we count the number of spheres containing $lhp(e)$ but not $rhp(e)$. We can do this in $O(\log n)$ query time and $O(n^{3+\epsilon})$ storage using a multi-level-data-structure approach as in [CSW90].

2. Second, we count the number of spheres containing $rhp(e)$ but not $lhp(e)$.

3. Third, we find out the spheres not containing both $lhp(e)$ and $rhp(e)$. The next level finds out those spheres whose center $c(S)$ is within $W(e)$, which is equivalent to simplex range searching. The last level decides the number of spheres S meeting $l(e)$. The multi-level data structure is built using an approach similar to one in [CSW90].

The time bound of the whole method is bound by the highest characteristic dimension $d' = 5$, which is the characteristic dimension of the last test to determine the number of spheres meeting the line $l(e)$. We can build the whole data structure in $O(m^{5+\epsilon})$ time and $O(m^{5+\epsilon})$ storage. During the query we apply Megiddo's parametric search technique which increases the query time by a logarithmic factor. Using decompositions with a non-constant parameter r we can reduce the query time. The first and the second test can be performed in $O(\log m)$. The third test too can be done in $O(\log m)$. So we conclude:

Theorem 19 *Given* m *possibly intersecting spheres in 3-space we can solve the ray-shooting problem in time* $O(\log^2 m)$ *using a data structure of size* $O(m^{5+\epsilon})$.

10 Conclusions and open problems

We have presented a list of incidence problems between lines and spheres in 3-space. Using the dual vector representation of lines in 3-space and several standard algorithmic techniques we gave non-trivial time bounds to the problem of counting incidences of lines and spheres, and to the problem of finding the minimum line-point distance. We gave an optimal randomized algorithm to find the smallest-radius stabbing sphere for a set of lines. A further research effort is needed to give satisfactory solutions to the problem of finding the largest empty sphere in a set of lines. Here is a list of open problems.

1. Extend the line-sphere intersection-counting algorithm to count intersections of segments and spheres.

2. An interesting variation of the problem of bounding the number of tangencies between lines and spheres is obtained when we impose the condition that the lines do not intersect the interior of the spheres.

Theorem 16 *Given n lines in R^3 the smallest sphere intersecting every line can be found in $O(n)$ randomized expected time.*

9.6 Finding the largest empty sphere in a set of lines

Given a set L of n lines in general position in 3-space we want to find the largest sphere s tangent to four lines and missing every other line in L. The general position assumption ensures that there is a finite number of sphere tangent to four spheres. Recalling observation 1, this problem is equivalent to finding the highest vertex in the envelope $LE(G_L)$. Using the point-location data structure tailored for the lower envelope of algebraic surfaces of Theorem 14, we get an $O(n^{4+\epsilon})$ expected time method. A variation of the approach of Section 9.3 gives an $O(n^4 \log n)$ time method.

If we are interested in the largest empty sphere which cannot be moved we just need to do local tests requiring constant time for each sphere.

9.7 Spheres and tangent lines

Let us consider a set S of m spheres in general position (i.e. no 3 centers are collinear) and a set L of n lines. We want to bound the maximum number of sphere-line tangencies.

Let us define a tangency graph $T = (S \cup L, E)$ where $(s, l) \in E$ if and only if l is tangent to s. From Observation 2 it follows that T has no subgraph isomorphic to $K_{5,32}$. Therefore, applying a well known theorem of extremal graph theory [Bol78], we obtain the following result.

Theorem 17 *The number of sphere-line tangencies is at most $O(nm^{4/5} + m)$.*

9.8 Ray-shooting in spheres

If we have a set of disjoint spheres the data structure used to prove Theorem 13 can be modified to solve a ray-shooting problem. We store within the data structure lists of spheres in stabbing order. When we detect the lists of sphere stabbed by the query lines (supporting the ray) we search for the source of the ray using a binary search schema.

Theorem 18 *There is a data structure for ray-shooting on a set of n disjoint spheres which uses $O(n^{5+\epsilon})$ storage and answer queries in time $O(\log n)$.*

The case of non-disjoint spheres is more complex. We adopt an approach which combines ideas in [AM91] and [vKOA90]. We need some definitions. Given a sphere S, $c(S)$ is its center. Given a segment e, $lhp(e)$ is its left end-point, $rhp(e)$ is its right end-point and $W(e)$ is the region of R^3 containing e and bounded by two planes both perpendicular to e and passing through its left end-point and its right end-point.

Lemma 10 *Given a sphere S and a segment e, the following holds:*

A) $lhp(e) \in S$ and $rhp(e) \notin S$ imply $e \cap S \neq \emptyset$.

B) $rhp(e) \in S$ and $lhp(e) \notin S$ imply $e \cap S \neq \emptyset$.

9.3 Finding a stabbing line for a set of spheres

Given m spheres in general position, it follows from Observation 2 that the number of lines tangent to four spheres is $O(m^4)$. If we apply the algorithm of Theorem 13 using this particular set of lines we can determine the existence of a stabbing line in time $O(m^{25/6+\epsilon}) = O(m^{4.16+\epsilon})$. This method is an improvement on the naive $O(m^5)$ time method.

A different approach, which is similar to an argument in [AW88], is the following. Considering in turns all triples of spheres, let us fix the attention on one triple. The set of lines tangent to those three spheres form a constant number of one-dimensional families of lines. We map this family of lines on a one-dimensional parametric space of real numbers. Given a fourth sphere, the lines in the family stabbing the fourth sphere are mapped into a union of intervals. In order to find a line stabbing every sphere we just compute the intersection of these unions of intervals. The intersection step can be done in time $O(n \log n)$. Repeating this procedure for each triple of spheres, we obtain a $O(n^4 \log n)$ time method.

9.4 Finding the minimum point-line distance

Given m lines and n points in 3-space we want to find the minimum pont-line distance. Initially, we use the method of Theorem 11 to find if the minimum distance is greater than 0. Then, we place around each point a sphere of radius d and check the number of incidences among the given lines and the sphere. The minimum value d for which the number of incidences in > 0 gives us the minimum distance. Using Megiddo's parametric search technique [Meg83] we just need, as a black box, an algorithm that checks whether the number of line-sphere incidences is greater than 0. The result in Theorem 12 gives us such program. Moreover, it is easy to obtain its parallel version which runs in logarithmic parallel time, as required by the parametric search technique. We obtain the following result.

Lemma 9 *Given m points and n lines in 3-space the minimum point-line distance can be computed in randomized expected time $O(nm^{2/3+\epsilon} + m^{1+\epsilon})$.*

Using the result of Theorem 13 and Megiddo's parametric search we get a second algorithm for finding the minimum distance, which runs in randomized expected time $O(m^{5/6+\epsilon}n^{5/6+\epsilon} + n^{1+\epsilon} + m^{1+\epsilon})$. For $m < n$ this bound is less than the bound of Lemma 9. We thus obtain this result:

Theorem 15 *Given m points and n lines in 3-space the minimum point-line distance can be computed in randomized expected time $O(\min\{(nm^{2/3+\epsilon} + m^{1+\epsilon}), (m^{5/6+\epsilon}n^{5/6+\epsilon} + n^{1+\epsilon} + m^{1+\epsilon})\})$.*

9.5 Finding the smallest sphere touching each line in L

Recalling the discussion of Section 8, given a line l of lines in R^3 we have a regions G_l^+, representing the set of spheres meeting l. For a set of lines L, let $G_L^+ = \bigcap_{l \in L} G_l^+$. G_L^+ is the set of spheres that meet every line in L. From Lemma 5 we conclude that G_L^+ is convex. This fact and an adaptation of Clarkson's Linear Programming method [Cla88] lead us to the proof of the the following result.

9.2 Counting line-sphere incidences

Fix d and let, for a given line l, S_l be the set of points in the space (p_x, p_y, p_z) such that $P(l, p, d)$ is true. Given a set of n lines L, we build a point-location data structure for the arrangements of the surfaces $\{S_l | l \in L\}$ using the method in [CEGS89b]. We obtain the following result.

Lemma 7 *Given n lines and a real number $\rho > 0$, we have an $O(n^{3+\epsilon})$ storage data structure to count in $O(\log n)$ time the number of lines intersected by any query sphere of radius ρ.*

Theorem 12 *Given n lines and m spheres with the same radius we can find the number of spheres intersected by each line in randomized expected time $O(nm^{2/3+\epsilon} + m \log n)$.*

Proof. We split the n lines into groups of size roughly $m^{1/3}$ and we use the on-line method on each group separately. ∎

If we consider spheres of different radii. For a fixed line l the locus of tangent spheres in 4-dimensional space (p_x, p_y, p_z, D) is the surface G_l. We can use the method in [CEGS89b] obtaining an $O(n^{5+\epsilon})$ size date structure to answer line-queries in $O(\log n)$ time.

Lemma 8 *Given n lines in 3-space there is a data structure of size $O(n^{5+\epsilon})$ that for any query sphere s counts the number of lines intersected by s in time $O(\log n)$.*

For a given sphere $S(p, d)$ the predicate $P(l, p, d)$ rewritten for a minimal line parameterization is a polynomial equation in four variables. The locus of zeroes of this polynomial is a surface in 4-space. We apply the space decomposition method and obtain an $O(n^{5+\epsilon})$ storage data structure that answer line queries on a set of spheres in time $O(\log n)$. By applying the batched technique of [EGS90a] we obtain the following result:

Theorem 13 *Given n lines and m spheres we can count all line-sphere intersections in randomized expected time*

$$O(n^{5/6+\epsilon} m^{5/6+\epsilon} + n^{1+\epsilon} + m^{1+\epsilon}),$$

where the constants depend on ϵ, for any $\epsilon > 0$.

If k is the number of line-sphere intersections we can report them in time $O(n^{5/6+\epsilon} m^{5/6+\epsilon} + n^{1+\epsilon} + m^{1+\epsilon} + k)$.

If we are interested in determining whether there is at least one line-sphere intersection a modified approach yields a better bound. From Observation 1, a sphere missing any line in L is mapped into a point below the lower envelope $LE(G_L)$. Using an observation of Sharir et al. [Sha91], it is possible to modify the stratification method in [CEGS89b] when we are interested in locating points in the lower envelope of algebraic surfaces in R^d, with $d > 3$. With $O(n^{2d-4+\epsilon})$ storage we can locate a point in time $O(\log n)$. Using a simple batching schema we have the following result.

Theorem 14 *Given n lines and m spheres it is possible to determine if the number of line-spheres incidences is greater than 0 in $O(nm^{3/4+\epsilon} + m^{1+\epsilon})$ randomized expected time.*

Note that, for $m > n^2$, this method is asymptotically faster than the general incidence-counting algorithm of Theorem 13.

Proof. Four spheres give us four equations similar to Equation 4. Moreover we have that $u \cdot v = 0$ and $v_x^2 + v_y^2 + v_z^2 = 1$. We have six second-degree equations in six variables. The total number of roots is at most $2^6 = 64$. If (u, v) is a solution also $(u, -v)$ is a solution, therefore we have at most 32 lines. ∎

Given a set of spheres they partition the set of all lines in R^3 into *isotopy classes*. Two lines are in the same isotopy class if they can be moved continuously one into the other without changing the set of intersected spheres. Observation 2 has the following consequence:

Theorem 10 *Given a set S of m spheres in general position all isotopy classes of lines with respect to S have complexity $O(m^4)$.*

8.1 A minimal line parameterization

Given a line l in 3-space a minimum line parameterization requires four parameters. A classic form for the equations of a line in 3-space is:

$$\frac{x - x_0}{l} = \frac{y - y_0}{m} = \frac{z - z_0}{n}$$

Where (x_0, y_0, z_0) are the coordinates of a point lying on the line and (l, m, n) are numbers proportional to the direction cosines of the line [PM75]. If we impose $z_0 = 0$ and $n = 1$ we obtain a minimal parameterization of all lines except a family of ∞^3 lines. Given a point p and a line l the polynomial describing the fact that p is at distance d form l is a polynomial of fourth degree in the parameters of the line (x_0, y_0, l, m).

For completeness we recall that, besides parameterization with four and six parameters, there is a parameterization with five parameters called *Plücker coordinates* of the line [CEGS89b, Som51]. We will switch from one parameterization to the other according to their properties in relation with the algorithm we are describing.

9 Algorithmic results

9.1 Incidences of lines and points

Given n lines and m points in R^3 the problem is to count the number of incidences between lines and points. We note that a random projection of the points and the lines keeps the same number of intersections. The problem is reduced to counting point-line intersections on a plane, for which we use the algorithm of Agarwal [Aga90]. We obtain the following result.

Theorem 11 *Given n lines and m points in R^3, there is a randomized algorithm to count the number of incidences that runs in expected time*

$$O(m^{2/3}n^{2/3}\log^{2/3} n \log^{\omega/3}(m/\sqrt{n}) + (m+n)\log n),$$

where ω is a constant < 3.33.

Lemma 4 *1. Line l is tangent to sphere $S(p,d)$ iff $Q_l(p,d) = 0$.*

2. Line l intersects sphere $S(p,d)$ iff $Q_l(p,d) < 0$.

3. Line l misses sphere $S(p,d)$ iff $Q_l(p,d) > 0$.

Let $d^2 = D$. In space (D, p_x, p_y, p_z) the tangency condition is represented by a surface G_l of equation:

$$G_l = \{(D,p) \mid D = A_l(p)^2 + B_l(p)^2 + C_l(p)^2\}.$$

We define the regions G_l^+ of spheres hitting l, and G_l^- of spheres missing l, as follows:

$$G_l^+ = \{(D,p) \mid D > A_l(p)^2 + B_l(p)^2 + C_l(p)^2\}$$

$$G_l^- = \{(D,p) \mid D < A_l(p)^2 + B_l(p)^2 + C_l(p)^2\}.$$

Lemma 5 *G_l^+ is a convex region in the space (D,p).*

Proof. $A_l(p), B_l(p)$ and $C_l(p)$ are linear functions in p_x, p_y, p_z, therefore $A_l(p)^2$, $B_l(p)^2$ and $C_l(p)^2$ are convex functions. Since $A_l(p)^2 + B_l(p)^2 + C_l(p)^2$ is a sum of convex functions, it is a convex function. ∎

Observation 1 *The set of spheres missing all lines in a set L (i.e. $G_L^- = \bigcap_l G_l^-$) is the region below the* lower envelope *of n surfaces G_l along the direction D in 4-space.*

Let us denote this lower envelope as $LE(G_L)$. It is important for any algorithmic purpose to give good bounds on the complexity of $LE(G_L)$, denoted by $|LE(G_L)|$.

Lemma 6 *$|LE(G_L)|$ is $\Omega(n^2)$.*

Proof. Build a regular $n \times n$ grid on the plane and place a circle in each square. Extend this construction to 3-space in the obvious way. Each such sphere is a vertex in the lower envelope $LE(G_L)$. ∎

A trivial upper bound for $|LE(G_L)|$ is $O(n^4)$ since this is the complexity of the whole arrangement of the surfaces G_L in 4-space.

Let us consider again the formula in Equation 3. We fix the parameters of the sphere and we consider the 6 parameters of the lines as variables. After simplifications we obtain the following equation:

$$u_x^2 + u_y^2 + u_z^2 + |p|^2 - 2(u \cdot p) - (p \cdot v)^2 = d^2. \tag{4}$$

Observation 2 *There exists a constant number of lines $c \leq 32$ tangent to four spheres in general position.*

10. Given m spheres and n lines bound the number of line-sphere tangencies (Section 9.7).

11. Given m spheres in 3-space, build a data structure to detect the first sphere hit by a query ray (Section 9.8).

12. Consider the problems for m given spheres when the radius d is the same for all the spheres.

8 Geometric preliminaries

Given a line l and the origin O we use the so called *dual vector representation* for lines in 3-space [BR79]. Let u be a vector from O to l perpendicular to l. Let v be a unit vector oriented as l. And $w = v \times u$.

Definition 1 *The pair (v, w) is the dual vector representation of l.*

Any pair of vectors such that $|v| = 1$ and $v \cdot w = 0$ represent a line in R^3. The number of independent variables is 4. Easily we have $u = -v \times w$ and $dist^2(O, l) = |u|^2 = |v \times w|^2$. Our aim is to find for a generic point p an explicit representation of the function $dist(p, l)^2$. We move the origin to p so we get a new vector $u' = u - p$. The coordinates of the line in the new reference frame are $(v, v \times (u - p)) = (v, w - (v \times p))$.

$$dist^2(p, l) = dist^2(O, (v, w - v \times p)) = |v \times w - v \times (v \times p)|^2$$

The predicate $P(l, p, d) \equiv (dist(l, p) = d)$ is equivalent to

$$|v \times w - v \times (v \times p)|^2 = d^2$$

which can be rewritten in the form

$$A(p, l)^2 + B(p, l)^2 + C(p, l)^2 - d^2 = 0 \tag{3}$$

where each A, B, C is the sum of 2 terms, one depending only on l and one depending on v and p.

Lemma 3

$$v \times (v \times p) = -[p - (v \cdot p)v]$$

Proof. Omitted. ∎

The term $A(p, l)$ is $-u_x + p_x - (p_x v_x + p_y v_y + p_z v_z)v_x$. Similar expressions hold for the terms $B(p, l)$ and $C(p, l)$. Fixing the line l, we can write a polynomial $Q_l(p, d)$ defined as follows:

$$Q_l(p, d) = A_l(p)^2 + B_l(p)^2 + C_l(p)^2 - d^2$$

where $A_l(p) = A(l, p), B_l(p) = A(l, p), C_l(p) = C(l, p)$ are linear forms in the variables p_x, p_y, p_z. The following lemma holds.

data structure has size $O(n^{2d-3+\epsilon})$ and it can be build in $O(n^{2d-3+\epsilon})$ randomized expected time or $O(n^{2d+1})$ deterministic time. The query time is $O(\log n)$.

In [EGS90a], Edelsbrunner et al. give a method for batched point-location which can be easily generalized to deal with any off-line pont location problem. Given n surfaces and m points we can compute properties of the points which depend only on the locations of these points with respect to the arrangement of the given surfaces in time $O(m^{d/(d+1)}n^{d/(d+1)} + m^{1+\epsilon} + n^{1+\epsilon})$, where d is a characteristic parameter of the problem.

Suppose we have an algorithm P', satisfying rather general conditions, to compute a predicate $P(i,t)$, which depends on an input i and a real parameter t. Megiddo's parametric search technique [Meg83, Col87] is a method that transforms P' into an algorithm P'' to find the minimum value of t for which $P(i,t)$ is true. Applications of Megiddo's parametric search to geometric problems are in [AASS90, AM91, Pel91b]. In our applications of Megiddo's parametric search the parameter t is typically the radius of the input spheres, for this reason it is of special interest to consider problems about lines and spheres in the restricted case of spheres of the same radius.

Throughout the paper we will not distinguish between a sphere S and the closed ball B whose boundary is S. The reason for this ambiguity is that a line meets B if and only if it meets S. Note that this is not necessarily true for objects different from lines (e.g. segments).

In Section 8 we discuss several representations of lines and spheres and the geometric properties that are exploited in the algorithmic results. In particular we discuss the *dual vector* representation of lines whose properties are at the base of the optimal algorithm for finding the smallest stabbing sphere for a set of lines. In the next subsection we give a survey list of the problems discussed in the paper.

7.1 A list of problems on lines and spheres

1. Given n lines and m points (degenerate spheres) count all incidences (Section 9.1).
2. Given n lines in 3-space, build a data structure to count the number of lines intersected by a query sphere (Section 9.2).
3. Given m spheres in 3-space, build a data structure to count the number of spheres intersected by a query line (Section 9.2).
4. Given n lines and m spheres count the number of line-sphere pairs with non empty intersection (Section 9.2).
5. Find a line, if exists, intersecting all spheres in a given set of m spheres (Section 9.3).
6. Given n lines and m points find the closest line-point pair (Section 9.4).
7. Given n lines find the sphere of minimum radius intersecting all lines (Section 9.5).
8. Given n lines find the maximum number of spheres tangent to 4 lines and whose interior is not intersected by any line (Section 9.6).
9. Find the sphere of maximum radius tangent to 4 lines in a set of n lines and whose interior is not intersected by any line (Section 9.6).

Part II
Lines, points and spheres in 3-space

7 Introduction

In graphics it is a common practice to enclose parts of a 3-dimensional scene in spheres [Gla89]. The goal is to solve efficiently the problem of tracing rays in the scene. The general idea is to detect, in a first phase, those rays that meet a sphere. In a second phase, only the rays meeting a sphere are traced in the portion of the scene enclosed in the sphere. It is thus of interest for applications in graphics to detect efficiently line-sphere intersections and to perform ray-shooting on spheres. In this paper we give algorithms for counting (and reporting) line-sphere incidences, and for detecting the first sphere met by a ray in a set of disjoint spheres. The algorithm for counting the number of intersections of n lines with m spheres runs in $O(n^{5/6+\epsilon}m^{5/6+\epsilon} + n^{1+\epsilon} + m^{1+\epsilon})$ randomized expected time.

In CAD and robotics it is often important to enforce a minimum separation among 3-dimensional polyhedral objects. For this problem, in the case of general sets of polyhedra, only trivial quadratic algorithms exist, to our knowledge. We consider the minimum separation problem restricted to simple classes of objects in 3-space: planes, lines and points. In [Pel91b] Pellegrini considers the problem of finding the closet pair in a set of lines. In this paper we discuss the problem of detecting the minimum distance between a set of lines and a set of points in 3-space. It is easy to transform a proximity problem for lines and points into an incidence problem for spheres and lines. We give an algorithm to find the minimum distance between a set of n lines a set of m points which runs in $O(\min\{(nm^{2/3+\epsilon} + m^{1+\epsilon}), (m^{5/6+\epsilon}n^{5/6+\epsilon} + n^{1+\epsilon} + m^{1+\epsilon})\})$ randomized expected time.

Given a set of lines in 3-space several optimization problems for this set of lines can be rephrased as relations between lines and spheres. For example, in order to find the point in a region of R^3 with the maximum minimum distance from any line in a given set L of lines, we convert this problem into the problem of finding the sphere of maximum radius tangent to four lines and missing any other line of the set. We give an $O(n^4 \log n)$ time algorithm to solve this problem. A second example is the following: find the minimum distance d such that for some point, any line is at distance at least d. This is equivalent to finding the sphere of minimum radius stabbing every line in the given set of lines. We give a $O(n)$ randomized expected time algorithm for this problem, where n is the cardinality of the set of lines.

Understanding the interaction of two families of geometric objects is a fundamental problem in computational geometry. Several of the techniques used in this paper have been used in different contexts (see [CEG+90b, EGS90b, AS91b, AASS90]) to solve incidence problems. In this paper we explore the application of these techniques to problems on "lines and spheres" in 3-space. We use extensively the space decomposition technique in [CEGS89b], the batching technique in [EGS90a], and the parametric search technique of [Meg83], which represent, by now, standard methods for tackling problems with non-linear objects in higher dimensional spaces.

In [CEGS89b], Chazelle at al. describe a method for building a point-location data structure for the arrangement of n algebraic varieties of fixed degree in R^d. For $d \geq 3$, the

5 Conclusions and open problems

Solving problems on lines in 3-space is often a necessary ingredient for solving problems on sets of polyhedra in 3-space. In this paper we solve on-line intersection queries for lines and half-planes in 3-space, which is en essential ingredient of a fast method for free-placements queries amidst polyhedral obstacles in 3-space. In the same spirit, we give algorithms for solving a few neighbor problems on lines in 3-space. Here are a few open questions:

1. Is it possible to extend the solution of neighbor problems of lines to segments?
2. Combine neighbor information and collision free placements to plan a collision-free movement of a polyhedral object in a polyhedral environment.
3. Can the algorithms presented in this paper be modified to find the k-th shortest segment connecting two lines?
4. Can we solve efficiently this problem: given a polyhedral environment in 3-space determine if a query segment can be moved freely in direction v?
5. The collision-free query method counts the number of features of the intersection of s with the obstacles. Since this is clearly excessive for solving a decision problem, it is possible that a more sophisticated analysis yields an improved time bound.

6 Acknowledgments

Thanks to Subhash Suri for proposing the problem of finding the closest pair of lines. Thanks to Raimund Seidel for useful discussions. Thanks also to P.K. Agarwal for his comments on an early draft, and to Micha Sharir for his communications.

Lemma 2 *Given a set of n lines and a set of m cylinders of same radius, there is an algorithm that counts all line/cylinder intersections in time*

$$Am^{5/6+\epsilon}n^{5/6+\epsilon} + Bm^{1+\epsilon} + Cn\log^2 m$$

where the constants A,B and C depend on ϵ.

Proof. A line l in 3-space is representable in a canonical way as a point $p(l)$ in real parametric space R^4. A cylinder c of fixed radius t is representable in a canonical way as a point $p(c)$ in real parametric space R^4 (as a matter of fact we need only to map the axis of the cylinder).

Given a line l we define as the dual surface of l the set $S(l)$ of all the points $p(c)$ where c is a cylinder of radius t tangent to l. Clearly $S(l)$ is an algebraic variety (of co-dimension 1) in R^4. In particular there is a polynomial $q_l(.)$ such that $S(l) = \{x \in R^4 | q_l(x) = 0\}$. Symmetrically we define as the dual surface of the cylinder c the set $S(c)$ of the points $p(l)$ where l is tangent to c. Again $S(c)$ is an algebraic variety of co-dimension 1. And we know there exists a polynomial $r_c(.)$ such that $S(c) = \{x \in R^4 | r_c(x) = 0\}$. Given a line l and a cylinder c we define as their relative orientation $RO(l, c)$ as the sign of the polynomial $q_l(.)$ evaluated at the point $p(c)$. Also, we define $RO'(c, l)$ as the sign of $r_c(.)$ evaluated at $p(l)$. Since the set of all lines intersecting a cylinder is compact, knowing the relative orientation $RO(l, c)$ is enough to decide whether or not line l meets cylinder c. We have, moreover, the following facts:

1. Chazelle, Edelsbrunner, Guibas and Sharir [CEGS89b] give a data structure of size $O(n^{2d-3+\epsilon})$ to do point location in $O(\log n)$ time among n varieties of constant degree in d-space, for $d \geq 3$. We can build such data structure on n cylinders in $O(n^{5+\epsilon})$ expected time and storage, so that in $O(\log n)$ time we can answer the query: for a line l how many cylinders are intersected by l?

2. From the above discussion we can represent lines and cylinders as point or surfaces in R^4.

Using facts (1) and (2) we apply a quite standard batching argument developed in [EGS90a] and used in various other papers for locating m points in an arrangement of n surfaces. The time bound follows from the batching technique. ∎

The sequential oracle of Lemma 2 (based on the data structure in [CEGS89b]) has a tree structure of logarithmic depth in which the number of nodes is bounded by $T_S(n)$. Therefore we have $p \leq T(n)$ and $T_P \leq O(\log^c n)$ for some constant c. Applying the parametric search technique we obtain the following result:

Theorem 9 *Given n lines in space it is possible to find the shortest connecting segment in $O(n^{5/3+\epsilon})$ randomized expected time for any $\epsilon > 0$, where the multiplicative constant depends on ϵ.*

The oracle used in Theorem 9 is a counting oracle which counts the number of intersections. In order to solve the closest line problem it suffices that the oracle tests if the number of incidences is more than 0. It is likely that such algorithm can attain a better asymptotic bound than the counting oracle. Such improved bound would carry on immediately to the algorithm for finding the closest line.[2]

[2] Apparently, the improved time bound of [Sha91] is based on a similar observation.

Theorem 8 *Given a set L of n lines in R^3, the maximum length vertical segments connecting two lines in L can be found in $O(n^{4/3+\epsilon})$ randomized expected time.*

4 Finding the shortest segment

Given a set L of n lines in R^3 find the shortest segment connecting 2 lines. A more formal statement of the problem is the following: compute $M(L)$, where

$$M(L) = \min_{l',l \in L, l \neq l'} \min_{p \in l, q \in l'} |\bar{pq}|$$

For sake of completeness, we give a sketch of the general parametric search technique. For a more detailed discussion we refer to [Meg83, Col87] and [AM91, AASS90].

Let $P(t,i)$ be a program on an input i and a parameter t, such that parameter t is used only in tests and the program branches by evaluating the sign of polynomials in t of small and constant degree. For each input $\bar{i}$ and each value $\bar{t}$ the value $P(\bar{t},\bar{i})$ is a natural number. Moreover $P(t,i)$ is monotonically increasing in t for any fixed $\bar{i}$. Suppose $P(0,\bar{i}) = 0$. Megiddo's technique transform the program $P(t,i)$ into a program to solve the following problem: find the minimum value $t = t^*$ such that $P(t^*,\bar{i}) > 0$. The program P is called also the oracle of the computation.

In order to apply Megiddo's technique we need a parallel version of the oracle. Let T_S be the time bound on the sequential oracle. Let T_P be the parallel time bound for the parallel oracle when p processors are used. Then the overall time bound for the parametric search is $O(pT_P + T_S T_P \log p)$.

Computing $M(L)$ fits nicely into his general schema of Megiddo's parametric search. Given a set L of n lines in R^3 we partition L into two disjoint sets R and B of $n/2$ lines each. If $T(n)$ is the time of the algorithm for finding the closest pair we have the following recursive inequality:

$$T(n) \leq 2T(n/2) + T'(n/2, n/2)$$

Where $T'(n,m)$ is the time needed to solve the bichromatic version of the problem (i.e. find the shortest segment connecting a "red" line with a "blue" line).

Let us define as $P'(R,B,t)$ the program that, given a set R of red lines and a set B of blue lines, counts the number of red lines at distance less than t from any blue line. Clearly $P'(R,B,t)$ is monotone in t. We can easily check if $P'(R,B,0) = 0$ using a method in [Pel91c]. Finding the shortest bichromatic distance is equivalent to finding the minimum t such that $P'(R,B,t) > 0$.

4.1 The oracle

In this subsection we give the algorithm underlying the program $P'(R,B,t)$. The problem of counting pair of lines within distance t can be seen as the problem of counting the number of intersections among the set R of lines and the set B' of cylinders of radius t whose axis are lines in B.

3.1 Batched technique for the shortest vertical segment problem

In order to apply the batching technique in [EGS90a] (see [Pel90, Pel91c] for an adaptation of the technique to problems in Plücker space) to the problem of finding the shortest vertical segment we divide the set of lines $\mathcal{L}$ into 2 sets L_1 and L_2. One set is mapped as a set of data (surfaces) and the other set as a set of queries (points). It is essential that we obtain a correct answers when we flip the roles of the 2 sets as data and queries. The algorithm constructs a multilevel structure where a particular type of query is asked at each level. There is a subtle point to be clarified at the innermost level of the algorithm. At this stage of the algorithm we have a set of Plücker hyperplanes $H(M_\sigma)$ and a set of Plücker points $P(N_\sigma)$ such that the relative orientation is known and uniform (i.e. all lines in M_σ are above all lines in N_σ).

Using the algorithm of theorem 5 on M_σ as data and N_σ as points we obtain a set A of pairs of lines, where each line in N_σ is associated to the line in M_σ immediately below.

Reversing the role of M_σ and N_σ we obtain a set of pairs A'. The 2 sets A and A' can be very different one from the other. On the other hand if $(l, l') \in A$ is the pair of lines with minimum connecting vertical segment, then (l', l) is an element of A' and is the pair of lines with minimum connecting vertical segment in A'.

Reversing the role of the two sets of lines we do not loose the information about the minimum connecting segment. We have therefore the following theorem:

Theorem 7 *Given n lines in 3-space it is possible to find the shortest vertical connecting segment in $O(n^{8/5+\epsilon})$ randomized expected time for any $\epsilon > 0$, where the multiplicative constant depends on ϵ.*

Using the Hereditary Segment tree in [CEGS89a] we can extend the result for lines to segments.

Corollary 3 *Given n segments in 3-space it is possible to find the shortest vertical connecting segment in $O(n^{8/5+\epsilon})$ randomized expected time for any $\epsilon > 0$, where the multiplicative constant depends on ϵ.*

3.2 Finding the longest vertical segment

Suppose we want to find the longest among the vertical segments connecting 2 lines in L. This problem has a formulation similar to finding the shortest vertical segment problem, but we obtain a better solution.

We partition L into 2 subsets L_1, and L_2 of equal size and we solve the problem recursively in each set. It is thus enough to solve the bichromatic case. Suppose for the moment that the pair (l_1, l_2) achieving the maximum has l_1 above l_2, and let δ be the value of the maximum. If we shift every line in L_1 downward of a distance δ, we obtain a new set L_1'. From the fact that δ is the maximum distance follows that every line in L_1' is below every line in L_2. The two sets L_1' and L_2 satisfy the towering property (see [CEGS89a]) which can be tested in $O(n^{4/3+\epsilon})$ randomized expected time. The algorithm for testing the tower property can be used as the oracle of Megiddo's parametric search method. The parametric search approach allows us to find the value of δ which is the maximum bichromatic distance.

Theorem 4 *Given a set of polyhedral obstacles with n edges we can build a data structure using m storage with $n^{1+\epsilon} \leq m \leq n^{4+\epsilon}$ such that for any query simplex s we can determine in time $O(n^{1+\epsilon}/m^{1/4})$ whether s is collision-free.*

Proof.
Given a set of polyhedral objects with a total of n edges and a simplex s in 3-space. We have to consider the following collision cases:

1. An object is completely contained in s. In this case we select 1 point out of each object and we perform a simplex range searching to find if any point is within s. This can be done using the methods in [CSW90, Mat91] within the stated bound.
2. An edge of s meets a face of some object. This reduces to a ray-shooting problem which we can do within the stated bound using the method in [AS91a] and its improvement in [Aga91].
3. A face of s meet an edge of some object. This case is solved using the method of theorem 3.

∎

3 Finding the shortest vertical segment

We use the following theorem from [CEGS89a]:

Theorem 5 ([CEGS89a]) *Given a set $\mathcal{L}$ of n lines in R^3 and any $\epsilon > 0$, we can preprocess $\mathcal{L}$ into a data structure whose storage is $O(n^{2+\epsilon})$, so that for any query line l we can decide whether l is above all lines in $\mathcal{L}$ in $O(\log n)$ time and, if so, find the line $l' \in \mathcal{L}$ hit first by l when translated downward.*

For a query line l the line in $\mathcal{L}$ with the shortest vertical connecting segment is the line just above or below l. Each cell C the of the arrangement $\mathcal{A}(\mathcal{L})$ of Plücker hyperplanes (see [CEGS89a, Pel91a, AS91d] for more details) of lines in $\mathcal{L}$ has an associated set $b(C) \subseteq \mathcal{L}$ of lines such that for every line l whose Plücker point is in C, the lines in $b(C)$ are below l.

Combining this observation, theorem 5 and a random sampling approach we obtain a data structure that for every query line l gives us the line in $\mathcal{L}$ immediately above and the line immediately below. The storage $S(n)$ used by the data structure satisfies the following equation:

$$S(n) \leq r^4 S(n/r \log r) + r^4 n^{2+\epsilon}$$

The solution is $S(n) = O(n^{4+\epsilon})$. We have the following theorem:

Theorem 6 *Given a set $\mathcal{L}$ of n lines in R^3 and any $\epsilon > 0$, we can preprocess $\mathcal{L}$ into a data structure whose storage is $O(n^{4+\epsilon})$, so that for any query line l we can find in $O(\log^2 n)$ time he line $l' \in \mathcal{L}$ hit first by l when translated downward (or upward).*

Proof. If we take $\mathcal{L}$ itself as the set of queries, the batched version of Theorem (6) is an algorithm for finding for each line in L the line immediately above and below. We have just a linear number of such pairs and in linear time we find the shortest connecting vertical segment. ∎

A pair (l', v) determines an halfplane h in 3-space. We can think of $p'(l', v)$ as an hyperplane in $\mathcal{P}^2$ and $\pi'(l)$ as a point in $\mathcal{P}^2$. We build the data structure of Theorem 1 reversing the roles of hyperplanes and points in $\mathcal{P}^2$. For a query line l we can detect efficiently if any of the lines stored in the data structure hits l when moved in a specified direction. This is equivalent to detect an intersection of l' with the half-planes stored in the data-structure.

Corollary 2 *Given a set of halfplanes H of size n, there is a data structure $D(H)$ of size $O(n^{2+\epsilon})$ that in time $O(\log n)$, for a query line l, determines if l intersects 0 (or all) halfplanes in H.*

The data structure of Corollary 2 is a generalization of the data structure in [CEGS89a] which can detect intersections of lines and *vertical* halfplanes only. A data structure solving the same problem of Corollary 2 is in [AM91].

2.1 Querying sets of segments with triangles

Let S be a set of segments in 3-space we build a multilevel data structure (see [CSW90]) to answer the following question: count the number of segments in S intersected by a query triangle. Let t be a query triangle and aff(t) be the plane spanning t. The first level of the data structure detects the segments of S which are intersected by aff(t). We dualize every segment in S into a double spatial wedge using a standard point/plane duality D. Detecting the segments intersected by the plane aff(t) is equivalent to locate the dual point $D(\text{aff}(t))$ in the arrangement $\mathcal{A}(D(S))$ of the wedges.

We consider now t as the intersections of 3 half-planes in 3-space. In the next 3-levels of the data structure we store the lines supporting the segments and we build half-plane/line intersection data structures. At the last level we can count the number of segments which intersect t. We obtain the following result:

Theorem 2 *Given n segments there is a data structure that with storage $O(n^{4+\epsilon})$ allow to count all segments intersected by a query triangle in time $O(\log n)$.*

We can also report the intersections in time $O(\log n + k)$, using $O(n^{4+\epsilon})$ storage, where k is the output size.

We can rebuild the whole multilevel data structure using an half-space range searching approach [AS91a, CSW90, Mat91, Aga91] obtaining an almost linear size data structure which answers queries in time $O(n^{3/4+\epsilon})$. We obtain a space/time trade-off using the techniques in [AS91a] and the improvements of [Aga91]. With m unit of storage the query time is $O(n^{1+\epsilon}/m^{1/4})$.

Theorem 3 *Given a set of n segments we can build a data structure using m storage with $n^{1+\epsilon} \leq m \leq n^{4+\epsilon}$ such that for any query triangle t we can count in time $O(n^{1+\epsilon}/m^{1/4})$ the number of segments intersected by t.*

Theorem 3 is an essential ingredient in the proof of the next theorem:

of l and l' on the xy-plane is given by the sign of the following determinant:

$$mix(l,l',v) = sign \begin{vmatrix} a_1 - b_1 & a_2 - b_2 & a_3 - b_3 \\ a'_1 - b'_1 & a'_2 - b'_2 & a'_3 - b'_3 \\ 0 & 0 & 1 \end{vmatrix} \quad (1)$$

which is the mixed product of 3 vectors, namely $p_1 - p_2$, $p'_1 - p'_2$ and $(0,0,1)$. If the vertical direction is specified by a variable vector v, the relative orientation is still the mixed product of 3 vectors:

$$mix(l,l',v) = sign \begin{vmatrix} a_1 - b_1 & a_2 - b_2 & a_3 - b_3 \\ a'_1 - b'_1 & a'_2 - b'_2 & a'_3 - b'_3 \\ v_1 & v_2 & v_3 \end{vmatrix} \quad (2)$$

We define $\Phi(l,l')$ as the relative orientation of 2 lines in Plücker space ([CEGS89a]). We have the following fundamental lemma:

Lemma 1 *$above(l,l',v) = mix(l,l',v)$ xor $\Phi(l,l')$.*

Proof. Omitted. ∎

The determinant in (2) can be expanded in a bilinear form in terms of the 2-by-2 minors of the last two rows and the 1-by-1 minors of the first row. Let us define $\pi'(l) = (a_1 - b_1, a_2 - b_2, a_3 - b_3)$ and $p'(l',v) = (u_{23}, u_{31}, u_{12})$, where $u_{ij} = (a'_i - b'_i)v_j - (a'_j - b'_j)v_i$.

We can think of $p'(l',v)$ as a point in oriented projective 2-space $\mathcal{P}^2$ and of $\pi'(l)$ as an hyperplane in the same space. The value of $mix(l,l',v)$ is just the sign of the inner product of $\pi'(l)$ and $p'(l',v)$. The value of $mix(l,l',v)$ is also the sign of the polynomial defining $\pi'(l)$ evaluated at $p'(l',v)$.

In order to test a line l' for "aboveness" in direction v with respect to a set of lines L we construct the arrangement $\mathcal{A}'(L)$ of the hyperplanes in $\{\pi'(l) \mid l \in L\}$ and we associate with each region an appropriate Plücker polyhedron. First we locate the cell c in the arrangement $\mathcal{A}'(L)$ containing $p(l',v)$. Then we test for inclusion of the Plücker point $p(l')$ in the Plücker polyhedron associated with c. Using a plane partition approach to construct the arrangement, the storage $S(n)$ needed to construct the data structure satisfies the following equation:

$$S(n) \leq cr^2 S(n/r) + cr^2 O(n^{2+\epsilon})$$

We obtain the following theorem:

Theorem 1 *Given a set of lines L of size n, there is a data structure $D(L)$ of size $O(n^{2+\epsilon})$ that in time $O(\log n)$, for a query pair (l',v), determines if l' is above L in direction v.*

If a line l is above all lines in L along a direction v, then the half-space defined by l and $-v$ does not intersect any line in L. We obtain the following corollary:

Corollary 1 *Given a set of lines L of size n, there is a data structure $D(L)$ of size $O(n^{2+\epsilon})$ that in time $O(\log n)$, for a query halfplane h, determines if h intersect 0 (or all) lines in L.*

extensively studied in computational geometry. Much less is known about neighbor problems in higher dimensional spaces for objects different from points. Neighbor problems for polyhedral objects in 3-space such as lines, segments and polyhedra are important because many applications in Robotics and 3D-VLSI require a minimum separation among 3-dimensional objects.

In the second part of this paper we describe algorithms to solve a few neighbor problems for lines in 3-space. First, we are interested in finding the shortest segment connecting 2 lines in a set of lines, under a variety of constraints. All the neighbor problems considered in this paper are easily solved in $O(n^2)$ time. For some of the problems slightly sub-quadratic algorithms were known. Here we obtain algorithms that are substantially sub-quadratic.

For the special case of the shortest vertical distance between two sets of edges of non-intersecting polyhedral terrains Chazelle et al. [CEGS89a] give an $O(n^{4/3+\epsilon})$ randomized expected time algorithm. The problem of the minimum vertical separation for segments was solved in [CS88] in time $O(n^{1.99987})$. The method in [CS88] maps a segment into a point in 6-dimensional space. Using the representation in [CS88] and the more recent decomposition technique of [CEGS89b] it is possible to obtain a time bound of roughly $O(n^{2-1/9}) = O(n^{1.8889})$.

In this paper we give algorithms to find the shortest vertical segment connecting two elements in a set of lines or segments which runs in time $O(n^{8/5+\epsilon})$. Surprisingly, we prove a time bound for finding the *longest* vertical segment that is only $O(n^{4/3+\epsilon})$. We find the shortest segment connecting 2 lines in a set of n lines in time $O(n^{5/3+\epsilon})$.

For some of the results mentioned above we use Megiddo's parametric search technique [Meg83, Col87]. Roughly speaking, this technique transforms an algorithm to test a property into an algorithm to find the minimum value of a parameter for which the property is true, under quite general conditions. Our use of the parametric search technique is similar to to that [AASS90], where they use it to select the k-th distance in a set of points on the plane.

Independently, Leonidas Guibas, Micha Sharir and others [Gui91] have considered the closest pair of lines problem, obtaining results similar to those presented in this paper. More recently, Micha Sharir et al. [Sha91] have improved the time bound for this problem to $O(n^{8/5+\epsilon})$.

The paper is organized as follows: in Section 2 we discuss the problem of detecting efficiently whether a placement of a simplex is collision-free. Section 3 deals with the shortest and longest vertical segment. Section 4 presents the algorithm for finding the shortest segment in a set of lines.

2 Free-placement query problem for polyhedral obstacles

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 intersect eventually l'.

When v is a vector parallel to the z-axis we have a query data structure of [CEGS89a] to decide whether a line is above a set of lines. In this section we generalizing that data structure by making the "vertical direction" part of the query. In [CEGS89a] the condition for a line to be above another line was given by the relative orientation of the 2 lines in Plücker space and by the relative orientations of the projections of the 2 lines on the xy-plane. Let us define $l = (a, b)$ and $l' = (a', b')$, then the relative orientation of the projections

Part I
Incidence, order and neighbor problems for lines

1 Introduction

Lines in 3-space is a recent topic of research in computational geometry (e.g. [CEGS89a, Pel90, Pel91c, AS91a, dBHO+91, CEG+90a]). The first type of problems concerning lines that have been considered can be classified as "incidence" or "order" problems. For example, to solve ray-shooting questions as in [CEGS89a, Pel91c, dBHO+91] we look at the relative orientation of lines in space. Other problems can be classified as "neighbor" problems because they involve a metric on the set of lines in R^3.

In the first part of the paper we address some "incidence" problems for lines in 3-space. In [Pel91c, AS91a] algorithms for the following problems are presented: given a set of triangles, count (or report) the triangles intersected by any query ray or segment. Here we address the dual problem: given a set of segments in 3-space count (or report) the segments intersected by any query triangle.

The solution of this problem allows us to determine efficiently whether a placement of a polyhedral object of constant complexity within polyhedral obstacles is collision-free. Finding collision-free placements for objects is a basic problem for applications in Robotics. There are many results (e.g. [LS87b, LS87a, GSS88, KS88, HO89, CK89, FHS89, Tol91], see [SS89] for a survey) for several variations of the free-placement problem (e.g. finding a free path, finding the placement of the largest copy, etc.) for a polygonal object amidst polygonal obstacles *in the plane*. In [Hal91] some motion planning problems for systems with 3 degrees of freedom are considered. In this case the robot is mapped as a point in a suitable 3-dimensional parametric space and the obstacles are mapped as surfaces within this space. Also, recent work of Boris Aronov and Micha Sharir [AS90, AS91c] on the complexity of a cell in an arrangement of triangles in 3-space has applications for this kind of problem. The result presented in this paper is of a slightly different nature. In 3-dimensional space, free-placement problems have a higher intrinsic combinatorial complexity, therefore we tackle the simplified problem of building a data structure for free-placements queries. On the other hand, our data structure is powerful because *both* the placement (6 degrees of freedom) *and* shape (as long as it is of constant complexity) of the robot are part of the query.

We consider first a solution which, at the expense of a large storage, answers the queries in logarithmic time. Applying the methods in [CSW90, Mat91, Aga91] (see also [AS91a]) we obtain a space/query-time trade-off.

We obtain the following main result (Theorem 4): Given a polyhedra set of obstacles with n edges we can build a data structure using m units of storage, with $n^{1+\epsilon} \leq m \leq n^{4+\epsilon}$, such that for any query simplex we can determine in time $O(n^{1+\epsilon}/m^{1/4})$ whether the simplex is collision-free[1].

Once we know that our simplex does not intersect any obstacle we are interested in finding the closest obstacle. Neighbor problems for sets of points on the plane have been

[1] All the bounds presented hold for every $\epsilon > 0$ and the multiplicative constants depend on ϵ.

Contents

INTERNATIONAL COMPUTER SCIENCE INSTITUTE

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

New algorithmic results for lines-in-3-space problems

Leonidas J. Guibas* Marco Pellegrini†

TR-92-005

January 1992

Abstract

In the first part of the report we consider some incidence and ordering problems for lines in 3-space. We solve the problem of detecting efficiently if a query simplex is collision-free among polyhedral obstacles. In order to solve this problem we develop new on-line data structures to detect intersections of query halfplanes with sets of lines and segments.

Then, we consider the nearest-neighbor problems for lines. Given a set of n lines in 3-space, the shortest vertical segment between any pair of lines is found in randomized expected time $O(n^{8/5+\epsilon})$ for every $\epsilon > 0$. The longest connecting vertical segment is found in time $O(n^{4/3+\epsilon})$. The shortest connecting segment is found in time $O(n^{5/3+\epsilon})$.

Problems involving lines, points and spheres in 3-space have important applications in graphics, CAD and optimization. In the second part of the report we consider several problems of this kind. We give subquaratic algorithms to count the number of incidences between a set of lines and a set of spheres, and to find the minimum distance between a set of lines and a set of points. We show that the sphere of minimum radius intersecting every line in a set of n lines can be found in optimal expected time $O(n)$. Given m possibly intersecting spheres we solve ray-shooting queries in $O(\log^2 m)$ time using a data structure of size $O(m^{5+\epsilon})$.

This technical report collects part of the second author's work at I.C.S.I. form September 1991 to January 1992. It comprises results in [Pel91b] and [GP91].

*Stanford University and DEC System Research Center.

†International Computer Science Institute, 1947 Center St., Berkeley, CA 94704 U.S.A. and Dept. of Computer Science, King's College, Strand, London WC2R 2LS U.K. e-mail: m.pellegrini@oak.cc.kcl.ac.uk