Proceedings of ROBEXS'85, The First Annual Workshop on Robotics and Expert Systems, NASA/Johnson Space Center, June 27-28, 1985, Instrument Society of America

0-87664-872-3/85/045-052/$0 + .50PP

PARTITIONING THE SPACE TO CALCULATE SHORTEST PATHS TO ANY GOAL AROUND POLYHEDRAL OBSTACLES

Wm. Randolph Franklin
Varol Akman

Electrical, Computer, and Systems Engineering Dept.
Rensselaer Polytechnic Institute
Troy, New York 12180-3590
Phone: 518-266-6077

ABSTRACT

Given a set of polyhedral obstacles we consider the problem of computing an obstacle-avoiding shortest path between two specified points, source and goal, external to the polyhedra. This problem is commonly known as FINDPATH in model-based robotics. We first show that FINDPATH is solvable in a straightforward yet inefficient way. We then consider a locus method based on an extension of the Voronoi diagram in computational geometry to solve the subproblem of determining the sequence of polyhedral edges where the shortest path bends. This method, given a source, partitions the free space around polyhedra into regions bounded by curved surfaces such that all goals inside a particular region have the same ordered list of bend edges. This reduces FINDPATH to a preprocessing step (computing the regions), plus a searching step (determining which region contains a given goal). Once the ordered sequence of edges that a shortest path must visit is known, the actual bend points are calculated after solving a system of nonlinear equations using algebraic elimination or numerical methods. Since this last computation can be done by standard packages, albeit slowly, in the common case where the goal point varies while the source point and the obstacles are fixed, the shortest path can be calculated by merely concentrating on the searching step.

Keywords: model-based robotics, computational geometry, FINDPATH, locus method, Voronoi diagram, elimination, planar and spatial point location.

INTRODUCTION

A common problem in robotics is to find a shortest path connecting two given points in the presence of polyhedral obstacles inside a workspace. This is known as FINDPATH after Winograd's renowned robot simulator SHRDLU (32). FINDPATH is a very important ingredient of "model-based" robot systems where a task is specified nonprocedurally and is expected to be accomplished automatically by referring to a geometric model of the workspace. Thus, confronted with a request such as "Move the red cube which is in front of the green pyramid to the top of the yellow prism," a model-based system would activate its geometric path planner to find an obstacle-avoiding optimal path for the given cube. Since in a model-based system human intervention is assumed to be nonexistent or minimal, this may in turn require other powerful functions from the path planner such as FINDSPACE (find a suitable location to place a given object), MAKESPACE (move or remove some obstacles to open up space), and so on. In SHRDLU all these functions do exist although their success is basically dependent on heuristics and the fact that Winograd considers simple workspaces called the "blocks world," where there are obstacles of low complexity.

In this paper we consider only the FINDPATH problem among the cited functions and give a geometric solution which always succeeds. Although we simply study the movement of a point between a source and a goal, an "object FINDPATH" version of the sort cited above can be easily simulated provided that we are only translating the given object (no rotations allowed). This is achieved using the "configuration space (C-space)" approach, first made popular by Lozano-Perez (16). In the C-space approach to an object movement problem, we transform the given obstacles to new obstacles by growing them while simultaneously shrinking the moved object to a "reference point." Once a new workspace is computed from the descriptions of the transformed polyhedra, a "point FINDPATH" version can be applied. It is

noted that the C-space method gives exact results for movements consisting of only translations and has been used with some success to obtain approximate paths when rotations are also permitted.

TWO-DIMENSIONAL FINDPATH

Many attempts have been made to solve the two-dimensional version of FINDPATH. In this case, the obstacles become polygons whose interiors are forbidden. This is useful in workspaces where all obstacles are prisms which extend between two parallel planes (such as the columns of a Greek temple), and the source and goal are both guaranteed to be in a plane parallel to the given planes. Denoting the source by s, the goal by g, and the polygons by $P=\{P_1,\ldots,P_k\}$, it is seen that the shortest path is a polygonal path with endpoints s and g, and possible bend points at some vertices of the members of P (5). To solve the problem, one can construct a network which has s, g, and the vertices of all P_i as nodes. There will be an arc between a pair of nodes in this network if the corresponding vertices in the workspace are "visible" to each other, i.e., they can be connected by a line segment which is free of intersections with the members of P. The weight of such an arc is taken as the Euclidean length of this line segment. Once this network (which will be denoted by Vnet, the "visibility network") is constructed it is easy to search it for a shortest path between the nodes corresponding to s and g. If there is a total of n vertices in P then Vnet may have in the worst case $O(n^2)$ arcs; thus the search can be performed in $O(n^2 \log n)$ time using say, Dijkstra's algorithm on Vnet (7). The construction of Vnet can either be done naively in $O(n^3)$, or using a more sophisticated method, in $O(n^2 \log n)$ time (29).

THREE-DIMENSIONAL FINDPATH

For three-dimensional FINDPATH the above approach is not applicable. In three dimensions, shortest paths may bend at some vertices as well as some interior points of some edges. However, the fact that a shortest path will still be a polygonal path (but this time with bend points lying on some edges of the members of P) makes the following brute-force solution possible. Enumerate all possible permutations of any length of the set of edges of the given polyhedra. For each permutation of edges find a shortest path between s and g touching every edge of this permutation in the order they appear in the permutation. If a shortest path computed from a particular permutation does interfere with the objects, then discard it. (This can be seen using a line-polyhedron intersection detection function.) The shortest path between s and g is the one which has the shortest length among the shortest paths obtained from the surviving permutations.

From the above algorithm it is apparent that we need a way to compute the shortest path through a family of lines. This problem will be called the optimal line visit (Figure 1). It can be shown that, given an ordered sequence of n lines and two points, the shortest path between these points constrained to touch each line in that order is unique (29). Furthermore, the shortest path satisfies an important angular condition, namely, the entry and the exit angles the shortest path makes with each line at its bend point are always equal.

Assume that each line is given by its two distinct points and assign different coordinate systems to each line, i.e., let line L_i be parametrized by x_i. Also, for each line compute u_i which is a unit vector along L_i and denote the bend point on L_i by c_i. Then:

$$\frac{c_{i-1}c_i \cdot u_i}{|c_{i-1}c_i|} = \frac{c_i c_{i+1} \cdot u_i}{|c_i c_{i+1}|}$$

where . denotes the scalar product and $|\ |$ is the vector length. If we rewrite the above equation after putting values of c_{i-1}, c_i, c_{i+1} in terms of x_{i-1}, x_i, x_{i+1}, respectively, and eliminate the square roots due to $|\ |$ then we obtain a quartic in three variables, x_{i-1}, x_i, and x_{i+1}. Repeating this for all lines, we end up with the following system of n quartics:

$$\begin{aligned}
&Q_1(x_1,x_2)=0\\
&Q_2(x_1,x_2,x_3)=0\\
&\quad\cdots\\
&Q_i(x_{i-1},x_i,x_{i+1})=0\\
&\quad\cdots\\
&Q_n(x_{n-1},x_n)=0
\end{aligned}$$

Theoretically, the above system of equations can be solved using resultants. This is a classical method known as the "elimination theory," cf. van der Waerden (31) and Collins (6). Alternatively, one can use various numerical techniques ranging from Newton-Raphson to the homotopy methods for solving a system of nonlinear equations.

If $L_1, L_2, \ldots, L_n$ are but finite line segments then the shortest path may be bending at points located outside these line segments. In this case, Sharir and Schorr (29) hint that the shortest path will have to pass through some endpoints of these segments at which it will subtend different entry and exit angles contrary to what was stated before. The problem is thus reduced to a collection of subproblems where a shortest path passes through the interior points of a subsequence of line segments.

RELATED WORKS

There have been various developments in the area of geometric path planning, especially in the last few years. For brevity, we shall mention only a certain section of it.

Lozano-Perez (15,16) introduces the Cspace approach and its practical applications in two- and three-dimensional workspaces. Brooks (3) and Nguyen (18) report working programs for path planning in the plane although these become quite slow when applied to realistic workspaces. Reif (23), Schwartz and Sharir (24,25,26,27), Sharir and Ariel-Sheffi (28), O'Dunlaing and Yap (20), and O'Dunlaing et al (19) prove results mostly on the computational complexity of several special cases of path planning. Their work is important for showing the computational effort to accomplish certain difficult functions such as coordinating the collision-free motion of many objects, etc. Finally, shortest path computation has also been treated in recent papers such as Franklin (9), Franklin and Akman (10), Franklin et al (11), Sharir and Schorr (29), Lee and Preparata (14), O'Rourke et al (21), Papadimitriou (22), and Mitchell and Papadimitriou (17). In an upcoming work, Akman (1) discusses the problem in detail, gives several implementations for special instances of FINDPATH, and cites a wide spectrum of references on the subject. Franklin and Akman (10) deal with shortest paths on/around a single convex polyhedron. They also show how to partition the boundary of a convex polyhedron to solve the single source - all goals version of FINDPATH. Franklin et al (11) report, among other things, an implementation of a two-dimensional Voronoi-based FINDPATH algorithm which will be detailed in the next section. Sharir and Schorr (29) mention several results on the nature of shortest paths on a convex polyhedron. They also prove that given a convex polyhedron P with n vertices and a point s on it, P can be preprocessed in $O(n^3 \log n)$ time to produce a data structure with the help of which one can find in $O(n)$ time the shortest path along the surface of P from s to any g. Mitchell and Papadimitriou (17) show that the same bound holds for a general polyhedron (with concavities and holes) although they compute a path only on the surface of the polyhedron, thus introducing the possibility that the found path may not be the shortest.

The shortest path problem in some ways may be considered as an extension of the NP-complete TRAVELING SALESMAN problem where we wish to determine the shortest walk (or tour) that traverses the nodes of a given network in any order, cf. Johnson and Papadimitriou (12). Although there are some technical difficulties arising from the distance metric, the Euclidean version of TRAVELING SALESMAN is also NP-complete. It is noted that the optimal line visit problem becomes NP-complete if we relax the requirement that the shortest path should visit the lines in a specified order. To see this, assume that we have a set of points in the plane. Draw perpendicular lines to the plane at each point. If there is a polynomial algorithm to solve the optimal line visit problem for this instance then it can be used to find an optimal tour of the given points. Thus, we have shown that any approach which uses the optimal line visit as a subroutine to solve FINDPATH must be of at least exponential time complexity. It is currently an open problem whether there is a polynomial algorithm (polynomial in k, the number of obstacles, and $n_1,\ldots,n_k$, their sizes) to solve FINDPATH.

In the remainder of this paper we shall assume that FINDPATH is solved as soon as we provide an ordered sequence of edges the shortest path must visit. In the naive algorithm we presented for FINDPATH, the most annoying step was that we had to treat all sequences of edges. In the next section we shall propose a scheme to cut down the search space for FINDPATH from this to a more manageable size using a locus approach.

A LOCUS METHOD FOR FINDPATH

A common specialization of FINDPATH occurs when s and P are fixed, and new paths should be calculated as g moves around the workspace. For example, a manipulator arm may pick up a part from a pile of parts at a fixed location, and then move somewhere in the scene to work with it.

In Franklin (9), an important construction based on an extension of Voronoi diagrams in the plane is given which, for s given s in the plane, partitions the plane into a set of regions such that all g's within a given region have the same list of bend vertices. (For similar extensions of Voronoi diagrams, see Lee and Drysdale (13), and Lee and Preparata (14).) This reduces the problem of finding a shortest path to the preprocessing step (finding the regions), plus the task of determining which region contains g (searching). The last step is easy since the borders of the regions are either line segments or portions of hyperbolae. Thus, existing point location algorithms can be used after some slight modifications. In the common case where g varies while s and the obstacles are fixed, the shortest path can be found by merely repeating the search (point location) phase.

If we emulate Franklin's approach in space then the regions will have the following property. All g's in a given region are reached from s after visiting the same sequence of edges of the obstacles in P. To gain insight to this problem, we shall first work on a very simple case, namely, a solid triangle situated in space.

Let a, b, and c be points in space. These points describe a triangle abc if they are not colinear. Let s be any point in space external to the the plane of abc. Assuming that abc is a solid triangle we want to partition the space into regions such that if a new point g is

specified we would be able to tell whether g can be directly reached from s, and if not, which edge of the triangle (ab, bc, or ac) the shortest path must touch. We refer the reader to Figure 2 during the following discussion.

If g is outside the frustum obtained by subtracting the pyramid described by basis abc and apex s from the infinite pyramid similarly described then the shortest path is sg. Thus, one of the regions, R_o, has all the points of the space that are not obstructed by abc. Otherwise, g may belong to one of three regions: R_{ab}, R_{bc}, or R_{ac}. R_{ab} is the region such that if $g \in R_{ab}$ then the shortest path is via edge ab. R_{bc} and R_{ac} are similarly defined. When g is on the boundary of two regions there may be two or three shortest paths.

Now, we shall compute the boundaries between the pairs R_{ab} and R_{bc}, R_{bc} and R_{ac}, and R_{ac} and R_{ab}. In the sequel, s will be assumed to be the origin without loss of generality. We shall first compute the boundary between R_{ab} and R_{bc}. Take g such that the intersection of sg with triangle abc is not empty. If g belongs to the surface between R_{ab} and R_{bc} then there are at least two s-to-g shortest paths, one bending at ab and the other at bc.

Let us develop g into the plane of triangle sab by rotating g about ab until it lands on the plane of sab to the other side of ab with respect to s, and denote the obtained point by g_{ab}. Although omitted here, it is not difficult to show that

$$|g_{ab}|^2 = |a|^2 + |ag|^2 + 2((ag.u_{ab})(a.u_{ab}) + |ag \times u_{ab}||a \times u_{ab}|)$$

where x denotes the vector product, and u_{ab} is a unit vector along ab. This formula gives $|g_{ab}|^2$ in terms of known quantities (a and u_{ab}) and the unknown g (with coordinates x,y,z). The formula for $|g_{bc}|^2$ (resp. $|g_{ac}|^2$) is analogous to the above formula; just change a to b (resp. c) and u_{ab} to u_{bc} (resp. u_{ac}). The surface between regions R_{ab} and R_{bc} is computed from the condition $|g_{ab}| = |g_{bc}|$ (after squaring both sides and using the formulas for g_{ab} and g_{bc}) as a ternary quartic. Other two regions are also of the same type.

This method can be applied to a solid polygon too (Figure 3). Let P be a convex polygon with vertices a,b,... and s a point outside the plane of P. It is possible to partition P into convex regions (each completely containing an edge of P) such that if g is later specified inside one of these regions then the shortest path between s and g is via the associated edge of this region. To see this, rotate s about the lines defined by edges ab,bc,... until it is coincident to the plane of P and always on the opposite side of a particular edge compared to the interior of P. This is basically an unfolding of the pyramid with apex s and basis P to the basis plane. Thus, n image points are obtained which will be denoted by s_{ab}, s_{bc},... Draw the Voronoi diagram of these points and clip it against the window P. This partitions P into convex regions since each Voronoi polygon is convex. Figure 3 demonstrates these regions. To characterize the three-dimensional regions behind the polygon we are required to compute all the potential boundaries between pairs of regions. Although conceptually easy, this may be messy when it comes to intersect the boundaries to compute their intersection curves. In fact, even a crude drawing of the three-dimensional regions is complicated and is left aside in Figure 3. However, it is noted that the intersection of these regions with the polygon would give the same Voronoi diagram shown in Figure 3.

The extension of the method to several obstacles seems difficult. In this case, a desirable property of plane partitions around polygons disappears. First, a brief account of Franklin's approach is in order. (The reader is referred to Franklin (9) and Franklin et al (11) for a detailed description.) Note that in the plane, once a subdivision is formed there is only one sequence of bend points for it (provided that it is not on a boundary curve in which case there may be more). In Figure 4, there are two obstacles (line segments), ab and cd, in the plane and a source s is given as shown. Franklin's algorithm partitions the plane into five regions in this case. R_o holds g's directly reachable from s. R_a holds g's which cause a shortest path to bend at a. R_b holds g's which cause a shortest path to bend at b. The boundary between R_a and R_b is a portion of a hyperbola. R_d holds g's which give rise to a shortest path bending at d. Finally, $R_{a,c}$ holds shortest paths bending first at a and then at c. The boundary between $R_{a,c}$ and R_d is also a portion of a hyperbola. All other boundary curves are linear. A crucial property of this diagram is as follows. A bend point acts as a source point for a later region. For instance, point a acts as a source point for the points of R_a. Similarly, b acts as a source for the points of R_b. Thus, the source point is continuosly "pushed back" and this is the underlying reason for the fact that all curves are either line segments or hyperbolic sections.

In 3-space, no such property exists. When we add a new triangle to a workspace with only one triangle the new regions induced by this obstacle will be separated by surfaces of order higher than four. Thus, whereas the boundary curves remain as hyperbolae in the plane in the space they would grow with every new polygonal obstacle placed into the workspace. One practical way to get around this problem is to approximate the boundaries with more manageable surfaces (such as quadrics or planes) and to keep them as such even when new obstacles are introduced. This is possible since the boundary surfaces are generally smooth.

32-4

SPATIAL POINT LOCATION

Once a workspace is preprocessed for a given s and P, for each g, the region that it is in must be determined. There are various algorithms (known as planar point location algorithms) for this problem in the plane but they work for straight line subdivisions. Since the region boundaries may be portions of hyperbolae in Franklin's method, it is necessary to convert these algorithms to handle planar graphs where edges can be hyperbolae as well as lines. This can be done without much difficulty.

For spatial point location in the presence of high order surfaces, we can use Chazelle's method (4). His is a generalization of Dobkin and Lipton's multidimensional searching scheme for the case of arbitrary (as opposed to linear) real algebraic varieties. Dobkin and Lipton (8) showed how to represent a family of n hyperplanes, using a polynomial amount of space, so that the query "Does point x lie in any of the given hyperplanes?" can be answered in O(log n) time. Chazelle generalizes this in the following manner. Let d be a constant which is an upper bound on the degrees of a given family of n polynomials $Q=\{Q_1,\ldots,Q_n\}$ in r variables (in our case r=3). He considers the problem of preprocessing the members of Q so that for any point x, the predicate [There exists a $Q_i \varepsilon Q$ such that $Q_i(x)=0$] can be evaluated efficiently. If the predicate is true, any of the indices i for which $Q_i=0$ can be reported. (Asking for all such indices may rule out a fast response anyway.) If the predicate is false then x lies in a region over which each Q_i does not change its sign. Assuming that these regions have been labeled during the preprocessing, reporting the label corresponding to the region containing x is required. It is noted that in our case we store as a label a pointer to the sequence of edges that the shortest path bends at.

The preprocessing is based on Arnon et al's "cylindrical algebraic decomposition" (2) which is a more practical version of Tarski's decision procedure (30), and some ideas of Schwartz and Sharir (25). The main result is a data structure for answering any such predicate in O(log n) time. The space and time spent to construct the data structure are both polynomial although the involved powers of n make it doubtful if the method can be implemented at all. This is an area with only recent results and only further experimentation will show the usefulness of Chazelle's approach.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under grants ECS 80-21504 and ECS 83-51942. The second author is also supported in part by a Fulbright award.

REFERENCES

1. Akman, V., "On Shortest Paths Avoiding Polyhedral Obstacles in Three-Dimensional Euclidean Space," Ph.D. dissertation, ECSE Dept., Rensselaer Polytechnic Inst., Troy, NY, Summer 1985, to appear.

2. Arnon, D., Collins, G.E. and McCallum, S., "Cylindrical Algebraic Decomposition I: the Basic Algorithm," SIAM Journal on Computing, Vol. 13, No. 4, Nov. 1984, pp. 865-877.

3. Brooks, R.A., "Solving the Findpath Problem by Good Representation of Free Space," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 13, No. 3, Mar./Apr. 1983, pp. 190-197.

4. Chazelle, B.M., "Fast Searching in a Real Algebraic Manifold with Applications to Geometric Complexity," Tech. Rep. CS-84-13, Computer Science Dept., Brown Univ., Providence, RI, Jun. 1984.

5. Chein, O. and Steinberg, L., "Routing Past Unions of Disjoint Linear Barriers," Networks, Vol. 13, 1983, pp. 389-398.

6. Collins, G.E., "The Calculation of Multivariate Polynomial Resultants," Journal of the ACM, Vol. 18, No. 4, Oct. 1971, pp. 515-532.

7. Dijkstra, E.W., "A Note on Two Problems in Connection with Graphs," Numerische Mathematik, Vol. 1, 1959, pp. 269-271.

8. Dobkin, D.P. and Lipton, R.J., "Multidimensional Searching Problems," SIAM Journal on Computing, Vol. 5, No. 2, 1976, pp. 181-186.

9. Franklin, W.R., "Partitioning the Plane to Calculate Minimal Paths to any Goal around Obstructions," Tech. Rep., ECSE Dept., Rensselaer Polytechnic Inst., Troy, NY, Nov. 1982.

10. Franklin, W.R. and Akman, V., "Minimal Paths between Source and Goal Points Located on/around a Convex Polyhedron," Proc. 22nd Allerton Conf. on Communication, Control, and Computing, Monticello, IL, Sep. 1984.

11. Franklin, W.R., Akman, V. and Verrilli, C., "Voronoi Diagrams with Barriers and on Polyhedra for Minimal Path Planning," The Visual Computer - An International Journal on Computer Graphics, Springer-Verlag, 1985, to appear.

12. Johnson, D.S. and Papadimitriou, C.H., "Computational Complexity and the Traveling Salesman Problem," The Traveling Salesman Problem, Chap. 3, Eds. E.L. Lawler, J.K. Lenstra and A.G. Rinooy Kan, Wiley, New York, 1981.

32-5

13. Lee, D.T. and Drysdale, R.L., "Generalizations of Voronoi Diagrams in the Plane," SIAM Journal on Computing, Vol. 10, No. 1, Feb. 1981, pp. 73-87.

14. Lee, D.T. and Preparata, F.P., "Euclidean Shortest Paths in the Presence of Rectilinear Barriers," Networks, Vol. 14, 1984, pp. 393-410.

15. Lozano-Perez, T., "Automatic Planning of Manipulator Transfer Movements," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 11, No. 10, Oct. 1981, pp. 681-698.

16. Lozano-Perez, T., "Spatial Planning: a Configuration Space Approach," IEEE Transactions on Computers, Vol. 32, No. 2, Feb. 1983, pp. 108-120.

17. Mitchell, J.S.B. and Papadimitriou, C.H., "The Discrete Geodesic Problem," Tech. Rep., Computer Science Dept., Stanford Univ., Stanford, CA, 1985.

18. Nguyen, V-D., "The Findpath Problem in the Plane," Tech. Rep., Artificial Intelligence Lab, Massachusetts Inst. of Technology, Cambridge, MA, Feb. 1984.

19. O'Dunlaing, C., Sharir, M. and Yap, C.K., "Retraction: a New Approach to Motion Planning," Proc. 15th ACM Symp. on Theory of Computing, Apr. 1983, pp. 207-220.

20. O'Dunlaing, C. and Yap, C.K., "The Voronoi Method for Motion Planning: I. The Case of a Disc," Tech. Rep., Courant Inst. of Mathematical Sciences, New York Univ., New York, Mar. 1983.

21. O'Rourke, J., Suri, S. and Booth, H., "Shortest Paths on Polyhedral Surfaces," Proc. 2nd Annual Symp. on Theoretical Aspects of Computer Science, Saarbrucken, W. Germany, Jan. 1985.

22. Papadimitriou, C.H., "An Algorithm for Shortest Motion in Three Dimensions," Information Processing Letters, 1985, to appear.

23. Reif, J.H., "Complexity of the Mover's Problem and Generalizations," Proc. 20th IEEE Conf. on Foundations of Computer Science, 1979, pp. 421-427.

24. Schwartz, J.T. and Sharir, M., "On the Piano Mover's Problem: I. The Case of a Two-Dimensional Rigid Polygonal Body Moving amidst Polygonal Barriers," Communications on Pure and Applied Mathematics, Vol. XXXVI, 1983, pp. 345-398.

25. Schwartz, J.T. and Sharir, M., "On the Piano Mover's Problem: II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds," Advances in Applied Mathematics, Vol. 4, 1983, pp. 298-351.

26. Schwartz, J.T. and Sharir, M., "On the Piano Mover's Problem: III. Coordinating the Motion of Several Independent Bodies: The Special Case of Circular Bodies Moving amidst Polygonal Barriers," International Journal of Robotics Research, Vol. 2, No. 3, Fall 1983, pp. 46-75.

27. Schwartz, J.T. and Sharir, M., "On the Piano Mover's Problem: V. The Case of a Rod Moving in Three-Dimensional Space amidst Polyhedral Obstacles," Communications on Pure and Applied Mathematics, Vol. XXXVII, 1984, pp. 815-848.

28. Sharir, M. and Ariel-Sheffi, E., "On the Piano Mover's Problem: IV. Various Decomposable Two-Dimensional Motion Planning Problems," Communications on Pure and Applied Mathematics, Vol. XXXVII, 1984, pp. 479-493.

29. Sharir, M. and Schorr, A., "On Shortest Paths in Polyhedral Spaces," Proc. 16th ACM Symp. on Theory of Computing, 1984, pp. 144-153.

30. Tarski, A., A Decision Method for Elementary Algebra and Geometry, Univ. of California Press, 1948.

31. van der Waerden, B.L., Algebra, Vol. 1 and Vol. 2, Ungar, New York, 1970.

32. Winston, P.H., Artificial Intelligence, Addison-Wesley, Reading, MA, 1977.

32-6

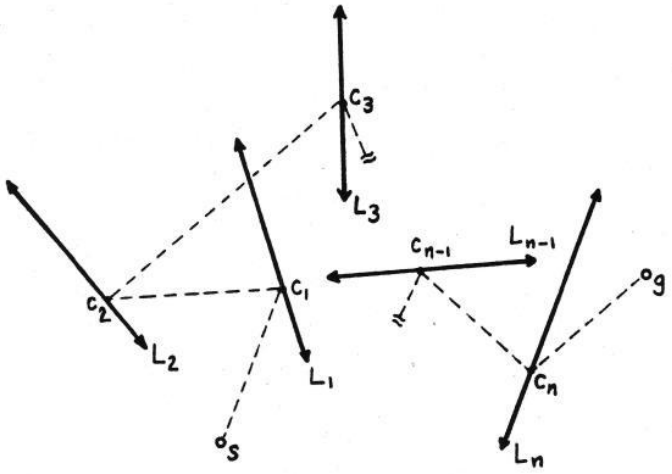

Figure 1. The Optimal Line Visit Problem.

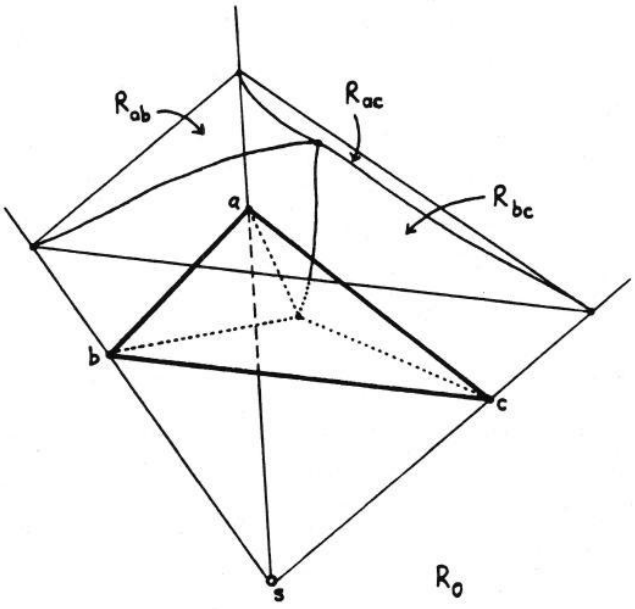

Figure 2. Partitioning the Space around a Solid Triangle.

32-7

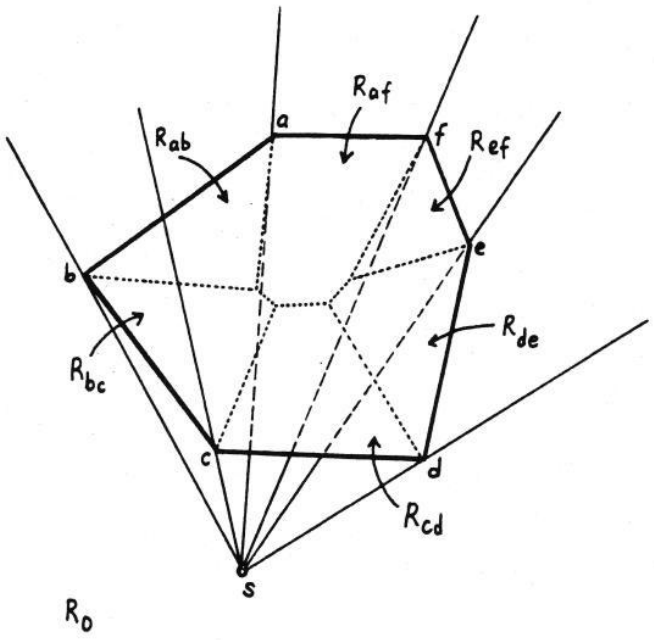

Figure 3. Partitioning the Space around a Solid Polygon.

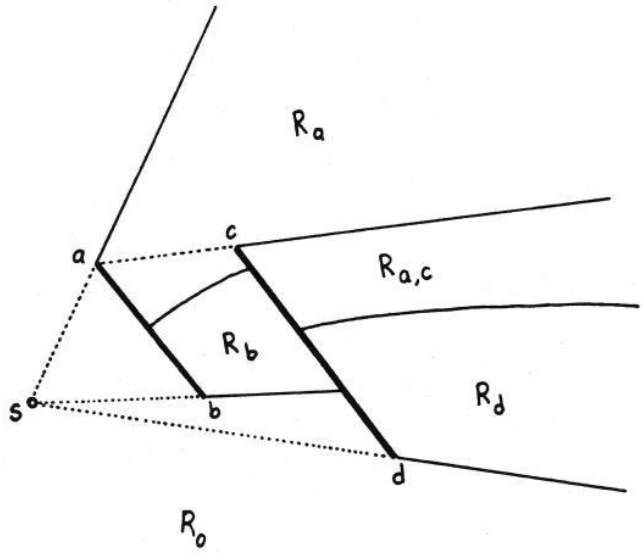

Figure 4. Partitioning the Plane around Two Linear Barriers.

32-8