# LOCUS TECHNIQUES FOR SHORTEST PATH PROBLEMS IN ROBOTICS

Wm. Randolph Franklin and Varol Akman
Dept. of Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute
Troy, New York 12180, USA

**Abstract.** Let $P$ be a set of obstacles in a workspace. We describe two locus techniques which, given a source point $s$ and a description of $P$, partition the free space such that for subsequent goal points generically denoted by $g$, the shortest $s$-to-$g$ obstacle-avoiding path is found efficiently. In the first case $P$ is a convex polyhedron which has $s$ and $g$ on its boundary. In the second case $P$ is a set of disjoint polyhedra and $s$ and $g$ are external to them. Both techniques are based on a generalization of Voronoi diagrams to accommodate more general geometries.

**Keywords.** Unobstructed shortest paths; Voronoi diagrams; computational geometry; point location; locus method.

## INTRODUCTION

Let $P=\{P_1, \cdots ,P_n\}$ be a prescribed set of obstacles and $s,g \in R^3$ be distinct points which are *not* internal to any $P_i$. Unless noted otherwise the members of $P$ are always in the Euclidean 3-space, $R^3$. The class of rectifiable curves which have endpoints $s$ and $g$ and which do not intersect any int$P_i$ is denoted by $C(s,g;P)$. For $C$ in this class, $l(C)$ denotes the length of $C$ under the $L_2$ metric. The relevance of the following problem to robotics is obvious:

> FINDPATH
> INSTANCE: Polyhedra $P=\{P_1, \cdots ,P_n\}$ such that $P_i \cap P_j = \Phi$ for $i \neq j$, and $s,g \in R^3$ such that $s \neq g$ and $s,g$ do not belong to int$P_i$, $1 \leqslant i \leqslant n$.
> QUESTION: Which $C \in C(s,g:P)$ has the shortest length?

It is known that there exists a $C' \in C(s,g:P)$ such that for all $C \in C(s,g;P)$, $l(C') \leqslant l(C)$. In 3-space, every such $C'$ is a polygonal path with its possible bend points belonging to some edges of some members of $P$. Also note that $C'$ is not necessarily unique. (In fact, there may be exponentially many shortest paths in terms of $n$.) We shall call any such path an $s$-to-$g$ *shortest path*. Thus, FINDPATH asks for the characterization (i.e., the determination of the bend points) of an $s$-to-$g$ shortest path.

There is a wealth of material in the general area of *motion planning* which includes other familiar robotics problems such as FINDSPACE, MAKESPACE, etc. in addition to FINDPATH. A recent work by Akman[1] deals with FINDPATH in greater detail and lists numerous references. Franklin and Akman also inspect the same problem from various perspectives in [9, 10, 11, 12]. Among other relevant research, Sharir and Schorr's work[26] is a detailed study on shortest paths. They mainly consider the case of BOUNDARY FINDPATH (locus) (cf. next section) and present an algorithm which works in time $O(n)$ per query where $n$ is the measure of complexity of the polyhedron, say the number of vertices. Their algorithm is based on the idea of "ridge" points on the object. A ridge point on the polyhedron has the property that there exists at least two shortest paths to it from the source. It turns out that the set of ridge points is made of line segments and that the union of the vertices and the ridge points is a closed connected set. Defining the union of the latter with the shortest paths from the source to every vertex, one partitions the polyhedron boundary into disjoint connected regions whose interiors are free of vertices or ridge points. The partition can be constructed in $O(n^3\log n)$ time using complicated techniques. O'Rourke and coworkers[21] use Sharir and Schorr's ideas to extend the problem to an orientable polyhedral surface which is no more supposed to be convex. The shortest path that they calculate is only a "geodesic", i.e., it is confined to the surface and thus may not be the true shortest path. Their algorithm runs in $O(n^5)$ time. It does not follow a locus approach; using their algorithm on each new goal point would take $O(n^4)$ time. Finally, Mitchell and Papadimitriou[20] give an algorithm to solve this last problem in a locus setting. (It is noted however that they are still computing geodesics, not true shortest paths.) Theirs is an $O(n^2\log n)$ time algorithm for subdividing the surface of an arbitrary polyhedron so that the length of the shortest path from a given source to any goal on the surface is obtainable by simple point location. As in our algorithm to be presented in the next section, point location is achieved in time $O(\log n)$, after which the actual shortest path is backtracked in time proportional to the number of faces that it traverses on the boundary.

## PARTITIONING THE BOUNDARY OF A CONVEX POLYHEDRON

Consider the following specific instance of FINDPATH:

> BOUNDARY FINDPATH
> INSTANCE: Convex polyhedron $P$, specified points $s, g \in \mathrm{bd}P$ where $s \neq g$.
> QUESTION: Which $C \in \mathrm{bd}P$ has the shortest length?

We shall assume that the boundary representation is used to define a polyhedron $P$. It is convenient to think of vertex labels or face labels as positive integers. The following definitions will be used in the sequel.

The *face graph (Fgraph)* of a convex polyhedron $P$ is an undirected graph $Fgraph = (FV, FE)$ with unit arc weight, $FV = \{i : F_i$ is a face of $P\}$ and $FE = \{(i,j) : F_i$ and $F_j$ are adjacent$\}$. Let $C \in \mathrm{bd}P$ be an $s$-to-$g$ polygonal path. The sequence of faces that $C$ enters defines the *face visit sequence* of $C$ which will be denoted by fvs$C$. Thus fvs$C$ is a walk in $Fgraph$ between the nodes corresponding to faces $F_s$ and $F_g$, the faces of $P$ containing $s$ and $g$, respectively. Let $C' \in \mathrm{bd}P$ be an $s$-to-$g$ shortest path. Then fvs$C'$ is clearly a simple walk in $Fgraph$. A *planar development* corresponding to a face visit sequence $1, \cdots, k$ is a union of polygons $F_1, \cdots, F_k$ of the planar polygonal schema of $P$. In the planar development two polygons $F_i$ and $F_{i+1}$, $1 \leqslant i < k$ are united along the edge that they are glued to each other, and do not overlap. The *image* of a point on a polyhedron under a planar development is the point in the plane that it ends up under the development. A planar development is *legal* if the line segment connecting the images of the source and the goal is internal to the development.

Now consider the following variant of BOUNDARY FINDPATH:

> BOUNDARY FINDPATH (locus)
> INSTANCE: Same as in BOUNDARY FINDPATH except that $g$ is not given. However, it is guaranteed that, when specified, $g$ will be on bd$P$.
> QUESTION: Preprocess $P$ so that for any specified $g \in \mathrm{bd}P$ the $s$-to-$g$ shortest path is computed efficiently.

A practical case which would benefit from BOUNDARY FINDPATH (locus) follows. Consider a truck on the surface of a mountain modeled as a convex polyhedron, say, a pyramid. Suppose that the truck is required to carry material from a fixed location on the surface to several points, e.g., construction sites. Then, it is reasonable to compute the shortest routes for the truck (approximated as a point) more efficiently than can be achieved by repeated applications of BOUNDARY FINDPATH for each specified destination. This is a powerful paradigm of computational geometry known as the *locus approach* and is studied in Overmars[22]. In solving a problem using this approach, we are allowed to spend some initial effort (preprocessing) to construct a data structure which will let us answer future requests (queries) quickly. To be effective, this assumes two things. First, the number of the query points must be large to validate such an initial effort. Second, the data structure must embody succinctly the locus of the required solution and must enjoy the existence of a fast search procedure to retrieve it.

We shall now summarize one such data structure suitable for solving BOUNDARY FINDPATH (locus). The data structure is known as the *Voronoi diagram* and was first introduced to computational geometry by Shamos[25]. Let $S = \{x_1, \cdots, x_n\}$ be a subset of $R^2$. For $1 \leqslant i \leqslant n$ let $\mathrm{regn}x_i = \{y : \mathrm{d}(x_i, y) \leqslant \mathrm{d}(x_j, y)$ for all $j\}$ be the *Voronoi region* of point $x_i$. The Voronoi diagram of $S$, denoted by vor$S$, partitions the plane into $n$ regions, one for each member of $S$. The Voronoi region of point $x_i$ consists of all points of $R^2$ closer to $x_i$ than any other point of $S$. For $1 \leqslant i, j \leqslant n$, letting $H_{ij} = \{y : \mathrm{d}(x_i, y) \leqslant \mathrm{d}(x_j, y)\}$ (the half-space defined by the perpendicular bisector of $x_i x_j$), it is seen that for $i \neq j$, $\mathrm{regn}x_i = \bigcap H_{ij}$. Thus, regn$x_i$ is a convex polygonal region and vor$S$ is equal to the union of the boundaries of all regn$x_i$. For every vertex $x$ of vor$S$ there are at least three points $x_i$, $x_j$, $x_k$ in $S$ such that $\mathrm{d}(x, x_i) = \mathrm{d}(x, x_j) = \mathrm{d}(x, x_k)$.

The Voronoi diagram of $S$ can be computed in $O(n \log n)$ time [24] and this is optimal with respect to a wide range of computational models. Yet, practical Voronoi programs which are both fast and reliable are difficult to write mainly due to the special cases in the diagram that are to be handled precisely. Among the published algorithms, those by Avis and Bhattacharya[2], Lee[16], and Guibas and Stolfi[13] seem to be more promising for practical use. On the other hand, it is possible to construct slower implementations which handle special cases without much effort.

We now summarize how to search Voronoi diagrams in logarithmic time in $n$, i.e., we cite methods which let one find the point $x \in S$ such that for a query point $y \in R^2$, $\mathrm{d}(x, y)$ is minimum. The search methods we shall review are more general than searching Voronoi diagrams in that they are based on searching a *planar subdivision*, i.e., a straight-edge embedding of a planar graph. The underlying problem is generally known as *planar point location* in computational geometry. Let us call a subset of the plane *monotone* if its intersection with any line parallel to the $y$-axis is a single interval (possibly empty). A subdivision is monotone if all its regions are monotone. Mehlorn[19] shows that a *simple* planar subdivision (a subdivision with only triangular faces) can be searched in time $O(\log n)$ after spending preprocessing time $O(n)$ and storage space $O(n)$. He also shows that if the searching problem for simple planar subdivisions with $n$ edges can be solved with search time $O(\log n)$, preprocessing $O(n)$, and space $O(n)$, then the searching problem for general subdivisions with $n$ edges can be solved with the same search time and space but preprocessing $O(n \log n)$. If all faces of the generalized subdivision are convex then $O(n)$ preprocessing is sufficient. (Lee and Preparata[15] also show that an arbitrary subdivision with $n$ edges can be refined to a monotone subdivision having at most $2n$ edges in $O(n \log n)$ time.)

Now we shall review some planar point location algorithms in the literature which either achieve the above bounds or come close. Dobkin and Lipton[5] were the pioneers to obtain an $O(\log n)$ query time but they use $O(n^2)$ space. Preparata[23] modifies their method to prove that $O(n \log n)$ space is sufficient. In an important paper Lee and Preparata[15] give an algorithm which is based on the construction of separating chains. Their algorithm achieves $O(n \log n)$ preprocessing and $O(n)$ space yet has a query time of $O(\log^2 n)$. (Their algorithm works for monotone subdivisions only.) Lipton and Tarjan[17, 18] give a method with $O(\log n)$ query time and $O(n)$ preprocessing and space (and thus optimal in all respects). Although based on a graph separator theorem which has many far-reaching consequences, they admit that their method is of only theoretical interest. Kirkpatrick[14] gives another method with the same bounds. His method builds a hierarchy of subdivisions and seems to be implementable. Finally, Edelsbrunner and coworkers[6] give a substantial improvement of the technique of Lee and Preparata and again attain the optimal bounds in all three respects. The importance of their method is that it may have an efficient implementation along with extensibility to subdivisions with curved edges.

Now we are ready to present our algorithm for BOUNDARY FINDPATH (locus). In the following we show the partitioning for only a face of $P$ (other than $F_s$) which we shall denote by $F_g$. To partition bd$P$, we apply the following algorithm for each face. (Clearly, $F_s$ is not partitioned due to convexity.)

*Algorithm* BOUNDARY FINDPATH (locus): Preprocessing
1. Find all simple face visit sequences between $F_s$ and $F_g$ using *Fgraph* of $P$.
2. For each face visit sequence found in step 1, compute the image of $s$ with respect to the planar development which starts with face $F_g$ and ends with $F_s$. (Note that we are *not* required to compute the whole development, but just the image point.)
3. Compute the Voronoi diagram of the image points calculated in step 2 using standard Voronoi programs mentioned above. It is required that with each image point (Voronoi center) we store the face visit sequence which is used to arrive it.
4. Clip the diagram obtained in step 3 with respect to "window" $F_g$ so as to preserve only those parts of it within the polygon $F_g$. (Any of the standard graphics algorithms such as the one due to Sutherland and Hodge[7] can be used for clipping.)
*End*

Assume that for a given $P$, the above preprocessing is carried out for all faces (except $F_s$). Thus, we have a family of planar subdivisions for each face such that for each region of a given subdivision we know the ordered sequence of faces to be developed to the plane if $g$ specified within that region. More specifically, we can apply the following algorithm when we are queried with a new $g$:

*Algorithm* BOUNDARY FINDPATH (locus): Querying
1. Compute the face $F_g$ holding a given $g$. If $F_g = F_s$ then the shortest path is trivially $sg$.
2. Using standard planar point location algorithms mentioned above locate $g$ inside the planar subdivision belonging to $F_g$.
3. Since we stored the face development sequence used to arrive this region we can now use it to compute the $s$-to-$g$ shortest path. Note that there may be cases where $g$ will be shared by at least two regions and thus there will be at least two shortest paths each of which is obtained via a different development sequence.
*End*

Figure 1 shows the Voronoi partitioning on a face of a cube using the above algorithm. The source face is face 1. It is noted that the given goal face (face 6) is partitioned into four triangular regions. The numbers next to each image in Fig. 1 define the face visit sequences used to arrive it. For instance, the image labeled 3 is obtained via face visit sequence 1, 3, 6. What Fig. 1 tells us can be summarized as follows. If a goal point is in the upper triangular region within face 6 then the shortest path to it is obtained by face visit sequence 1, 4, 6. For the lower triangular region this becomes 1, 2, 6, and similarly for other two regions.

In general, the number of regions a given face $F_g$ is partitioned by this algorithm is expected to be small. An informal argument for this is as follows. Consider the images of $s$ in the plane of $F_g$. If the face visit sequence for a particular image is long then the image will usually end up in a point farther away from $F_g$ compared to another image obtained by a shorter visit sequence. Thus, only a small portion of the possible face visit sequences between $F_s$ and $F_g$ can render images in the plane close to $F_g$ and contribute to the Voronoi diagram on it.

## PARTITIONING THE FREE SPACE AROUND POLYHEDRA

The inspiration for the work to be described in this section is Franklin's extension of Voronoi diagrams in the plane in the presence of barriers (line segments)[8]. Here he generalizes the Voronoi concept by allowing opaque barriers that a shortest path must avoid. With this environment, a shortest path will always be a polygonal path through the free space bending at the endpoints of the barriers. The ordered list of endpoints that a shortest path passes through is called its *contact list*. Franklin's construction works as follows. Regions are constructed so that every goal point in a region has a shortest path to the source with *identical* contact lists. Each region will have an *associated vertex* which is defined as the first vertex on its contact list. When a new barrier is introduced to the environment, two half-lines extending from its endpoints are added. These boundaries are called *shadow lines* and are the projections of the endpoints by the associated vertex of the region it is in. A second type of boundary is added to separate the two new regions created. This will in general be a hyperbola and is called a *ridge curve*. A goal on a ridge curve has two shortest paths to the source. If a boundary extends to infinity nothing interesting happens. However, sometimes a boundary intersects a barrier or another boundary. In the former case, the boundary is *cut* (i.e., stops at or blocked by the barrier). In the latter case, when two boundaries hit each other they join to create a *junction*.

We now give an example to illustrate Franklin's partitioning. The reader is referred to Verrilli[27] who presents a rather complete implementation and gives several interesting computer-generated examples. In Fig. 2, there are two barriers $ab$, $cd$ in the plane and $s$ is given as shown. In this case the plane is partitioned into five regions. $R_\Phi$ holds the goals directly reachable (visible) from $s$. $R_a$ holds the goals which cause a shortest path to bend at endpoint $a$ of $ab$. $R_b$ holds the goals which cause a shortest path to bend at $b$. $R_d$ holds points which give rise to shortest paths bending at $d$. Finally, $R_{ac}$ describes shortest paths bending first at $a$, and then at $c$ while going from $s$ to $g$. It can be easily shown that the boundaries between $R_a$ and $R_b$, and $R_d$ and $R_{ac}$ are hyperbolic portions. All other boundaries are made of straight lines.

A crucial property of the diagram in Fig. 2 (and of any Franklin's partitioning in 2-space around barriers) is that a bend point *acts* as a source point for a later region. (For instance, $a$ acts as a source for the points of $R_{ac}$.) Thus the source is continuously "pushed back" and this is the underlying reason for all boundaries being either line segments or hyperbolic sections. Once the boundaries are constructed, one can use any of the standard point location algorithms (modified slightly to take care of the fact that some planar boundaries are curved) to locate a goal. As soon as the region including the given goal is known the shortest path is found immediately.

We now emulate Franklin's approach in 3-space where the regions will have the following property: all points of a given region are reached from the source after bending at the *same* edges of the obstructions in the *same* order. We shall call this problem FINDPATH (locus). Our treatment will *not* be entirely algorithmic since we have not yet answered all the questions posed by this problem. It is seen that the analogous constructs to those outlined above will be *associated edge*, *shadow plane*, and *ridge surface* in case of FINDPATH (locus). A *contact list* will now hold the ordered set of edges instead of endpoints. We shall use the terms *cut* and *junction* without modification although now they refer to surfaces instead of curves.

Let $a, b, c \in R^3$ be three points not on the same line. We shall refer to the opaque triangle $abc$ by $T$. Let $s$ be any point outside aff$T$. We shall start with the following simple case: partition the space into regions such that if a new $g \in R^3$ is specified, then we can tell if $g$ can directly be reached from $s$, and if not, the edge of $T$ where an $s$-to-$g$ shortest path bends.

When $g$ is outside the infinite *frustum* obtained by subtracting the finite pyramid described by basis $T$ and apex $s$ from the infinite pyramid described similarly, the shortest path is $sg$ itself. Thus, one of the regions, $R_\Phi$ contains all points $g \in R^3$ such that $sg \cap T = \Phi$. (In other words, $R_\Phi$ is the set of all visible points from $s$.) If $g$ is not in $R_\Phi$ then three possibilities exist. For all $g \in R_{bc}$ the shortest path bends at $bc$. For all $g \in R_{ca}$ it bends at $ca$. Finally, for all $g \in R_{ab}$ it bends at $ab$. The edges $bc$, $ca$, and $ab$ are the associated edges of regions $R_{bc}$, $R_{ca}$, and $R_{ab}$, respectively. Fig. 3 shows these regions. Intersections of $R_\Phi$ with these three regions are the shadow planes for this example.

Now we shall compute the intersections of the pairs of regions. (These will be the ridge surfaces.) For clarity, we take $s$ as the origin of the coordinate system without loss of generality. We shall compute the intersection of regions $R_{ab}$ and $R_{bc}$ and generalize it to the other two cases. If $g \in R_{ab} \cap R_{bc}$ then there exists a shortest path to $g$ either via $ab$ or $bc$. Label the bend points of this shortest path with $ab$ (resp. $bc$) by $d_{ab}$ (resp. $d_{bc}$). These points should satisfy:

$$\mathrm{d}(s, d_{ab}) + \mathrm{d}(d_{ab}, g) = \mathrm{d}(s, d_{bc}) + \mathrm{d}(d_{bc}, g) \qquad (1)$$

The left (resp. right) hand side of (1) is equal to $\mathrm{d}(s, g_{ab})$ (resp. $\mathrm{d}(s, g_{bc})$) where $g_{ab}$ (resp. $g_{bc}$) is the point obtained by rotating $g$ about aff$ab$ (resp. aff$bc$) until it coincides with aff$sab$ (resp. aff$sbc$) and is in the opposite half-plane compared to $s$. In the sequel, we shall call any such point obtained by rotating a point about an axis an *image* point. If $p$ is a point in 3-space and $p'$ is its image due to rotation by an angle $\Theta$ about axis $u$ passing through the origin then:

$$p' = \langle p, u \rangle + (p - \langle p, u \rangle u)\cos\Theta + (u \times p)\sin\Theta \qquad (2)$$

Applying (2) for the image of $g$ about aff$ab$, we obtain:

$$g_{ab} = a + f\, u_{ab} + (ag - f\, u_{ab})\cos\alpha + (u_{ab} \times ag)\sin\alpha \qquad (3)$$

where $\alpha$ is the dihedral angle between aff$sab$ and aff$gab$, $u_{ab} = ab \,/\, \mathrm{d}(a, b)$, and $f = \langle ag, u_{ab} \rangle$. After some simplifications in (3), which are given in full in Franklin and Akman[12], the following expression for $g_{ab}$ is obtained:

$$\mathrm{d}^2(g_{ab}) = \mathrm{d}^2(a) + \mathrm{d}^2(a, g)$$
$$+ 2(\langle ag, u_{ab} \rangle \langle a, u_{ab} \rangle + \mathrm{d}(ag \times u_{ab})\mathrm{d}(a \times u_{ab})) \qquad (4)$$

To have a geometric interpretation of (4), we expand the scalar and vector products and simplify the resulting expression to get:

$$\mathrm{d}^2(g_{ab}) = \mathrm{d}^2(a) + \mathrm{d}^2(ag) - 2\mathrm{d}(a)\mathrm{d}(ag)\cos(\alpha_1 + \alpha_2) \qquad (5)$$

Here $\alpha_1$ and $\alpha_2$ are the angles $<gab$ and $<sab$, respectively. Thus (5) is a statement of the cosine theorem on triangle $sag_{ab}$.

Up to this point, we found an equation (i.e., (4)) that gives $\mathrm{d}^2(g_{ab})$ in terms of known quantities $a$ and $u_{ab}$, and unknown $g$. The equations for $\mathrm{d}^2(g_{bc})$ and $\mathrm{d}^2(g_{ca})$ are found completely analogously with obvious modifications. To compute the intersections of $R_{ab}$ and $R_{bc}$ we solve the equation $\mathrm{d}(g_{ab}) - \mathrm{d}(g_{bc}) = 0$, or equivalently, $\mathrm{d}^2(g_{ab}) - \mathrm{d}^2(g_{bc}) = 0$. In [12] it is shown that the intersection surfaces are in general ternary (in $x, y, z$, the coordinates of $g$) quartics which may degenerate to planes in some cases.

If we want to partition the space behind a solid polygon instead of a triangle then we are required to compute all the potential boundaries between the pairs of regions. It is obvious that the intersections of the regions with the polygon (or the triangle in the previous case) are made of straight edges. In fact the subdivision of the polygon by these edges can be found more simply. Let $P$ be a convex polygon with vertices $v_1, \cdots, v_n$ and $s$ a point outside aff$P$. The invisible side of $P$ to $s$ is partitioned into at most $n$ convex regions (each completely containing an edge of $P$) such that for a goal $g$ specified inside one of the convex regions the $s$-to-$g$ shortest path is via the associated edge of this region. To see this rotate $s$ about aff$v_1v_2$, aff$v_2v_3$, $\cdots$ until it is coincident to aff$P$ and always on the opposite side of a particular edge with respect to int$P$. This is basically a planar polygonal schema of the pyramid with basis $P$ and apex $s$ in aff$P$. Denote the $n$ images obtained in this way by $s_{12}$, $s_{23}$, $\cdots$ and construct their Voronoi diagram. When clipped by window $P$ the diagram partitions $P$ into at most $n$ regions which are necessarily convex since $P$ is convex. Fig. 4 illustrates this process.

The extension of the technique outlined for FINDPATH (locus) to several obstacles creates difficult problems. In this case we cannot find a similar property to that of Franklin's partitioning mentioned above, i.e., the junctions of the partition no more stay as quartics but grow in degree for each new obstacle causing new regions. We shall now imagine that for a given $s$ we managed to compute a subdivision of the space into $n$ regions bounded either by planes or quartic (or higher-order) surfaces using the above approach. With each region we assume the existence of a stored label which is simply equal to the contact list for this region. If we are given a new point $g$, we should first locate the region that $g$ is in, and then use the contact list for this region to compute the $s$-to-$g$ shortest path. Since the latter operation can be done by an optimal visit of the involved lines we shall concentrate on the former operation.

Let $S$ be a family of $n$ planes in $R^3$. Dobkin and Lipton[5] show how to represent $S$ in a polynomial amount of space so that whether a given point belongs to any of the given planes can be tested in $O(\log n)$ time. If we approximate the boundaries of our regions with planes then their algorithm is applicable. Chazelle[3] generalizes this for the case where $S$ is a family of algebraic varieties. His algorithm is an adaptation of Collins' quantifier elimination procedure[4]. Let $d$ be a positive constant and let $S=\{Q_1, \cdots, Q_n\}$ be a set of polynomials of degree at most $d$ in three variables with rational coefficients. He considers the problem of preprocessing $S$ so that, for any $(x,y,z) \epsilon R^3$, the predicate "There exists an $i$, $1 \leqslant i \leqslant n$, such that $Q_i(x,y,z)=0$" can be evaluated efficiently. This problem (and its generalization to spaces higher than $R^3$) is known as *spatial point location*. If the predicate is true then *any* of the indices $i$ for which $Q_i(x,y,z)=0$ is reported. Otherwise, $(x,y,z)$ is seen to lie in a region over which each $Q_i$ has a constant sign. Assuming that these regions have associated labels, his algorithm retrieves the label corresponding to the region containing $(x,y,z)$. The algorithm uses $O(n^{512})$ preprocessing time (and space) and computes the above predicate in $O(\log n)$ time.

## CONCLUSION

We described two locus techniques to compute the shortest obstacle-avoiding path between two points in 3-space. Only one of the points (source) is known prior to preprocessing and it is required that for later (goal) points the path must be evaluated quickly. Both techniques are based on generalizations of Voronoi diagrams to partition the free space and use standard point location algorithms to search a partition efficiently.

## ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under grants ECS 80-21504 and ECS 85-51942.

## REFERENCES

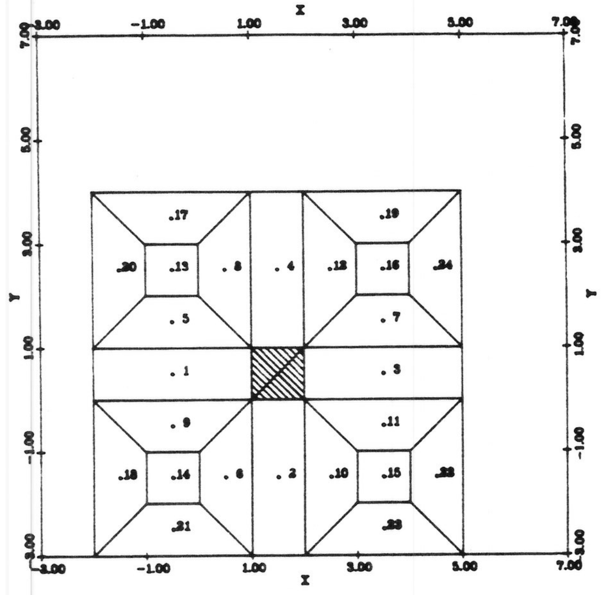
1. V. Akman, "Shortest Paths Avoiding Polyhedral Obstacles in 3-Dimensional Euclidean Space," Ph.D. dissertation, Dept. of Electrical, Computer, and Systems Eng., Rensselaer Polytechnic Institute, Troy, N.Y., Jun. 1985.
2. D. Avis and B. K. Bhattacharya, "Algorithms for Computing the d-Dimensional Voronoi Diagrams and Their Duals," in *Advances in Computing Research*, ed. F. P. Preparata, pp. 159-180, JAI Press, 1983.
3. B. M. Chazelle, "Fast Searching in a Real Algebraic Manifold with Applications to Geometric Complexity," Tech. Rep. CS-84-13, Dept. of Computer Science, Brown Univ., Providence, RI, Jun. 1984.
4. G. E. Collins, "Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition," *Proc. of the 2nd GI Conf. on Automata Theory and Formal Languages (Lecture Notes in Computer Science 33)*, pp. 134-183, Springer-Verlag, Berlin, 1975.
5. D. P. Dobkin and R. J. Lipton, "Multidimensional Searching Problems," *SIAM Journal on Computing*, vol. 5, no. 2, pp. 181-186, 1976.
6. H. Edelsbrunner, L. J. Guibas, and J. Stolfi, "Optimal Point Location in a Monotone Subdivision," Tech. Rep. 2, DEC Systems Research Center, Palo Alto, CA, Oct. 1984.
7. J. D. Foley and A. van Dam, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, Reading, MA, 1982.
8. W. R. Franklin, "Partitioning the Plane to Calculate Minimal Paths to any Goal around Obstructions," Manuscript, Dept. of Electrical, Computer, and Systems Eng., Rensselaer Polytechnic Inst., Troy, N.Y., Nov. 1982.
9. W. R. Franklin and V. Akman, "Shortest Paths between Source and Goal Points Located on/around a Convex Polyhedron," *Proc. of the 22nd Allerton Conf. on Communication, Control, and Computing*, Monticello, IL, Sep. 1984.
10. W. R. Franklin, V. Akman, and C. Verrilli, "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).
11. W. R. Franklin and V. Akman, "Partitioning the Space to Calculate Shortest Paths to any Goal around Polyhedral Obstacles," *Proc. of the 1st Annual Workshop on Robotics and Expert Systems (Houston, TX, Jul. 1985)*, Instrumentation Society of America, 1985 (to appear).
12. W. R. Franklin and V. Akman, "Euclidean Shortest Paths in 3-Space, Voronoi Diagrams with Barriers, and Related Complexity and Algebraic Issues," *Proc. of the NATO Advanced Study Institute on Fundamental Algorithms for Computer Graphics (Ilkley, Yorkshire, Apr. 1985)*, Springer-Verlag, 1985 (to appear).
13. L. J. Guibas and J. Stolfi, "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams," *Proc. 15th Annual ACM Symp. on Theory of Computing*, pp. 221-234, 1983.
14. D. G. Kirkpatrick, "Optimal Search in Planar Subdivisions," *SIAM Journal on Computing*, vol. 12, no. 1, pp. 28-35, Feb. 1983.
15. D. T. Lee and F. P. Preparata, "Location of a Point in a Planar Subdivision and its Applications," *SIAM Journal on Computing*, vol. 6, no. 3, pp. 594-606, Sep. 1977.
16. D. T. Lee, "On k-Nearest Neighbor Voronoi Diagrams in the Plane," *IEEE Trans. on Computers*, vol. 31, no. 6, pp. 478-487, Jun. 1982.
17. R. J. Lipton and R. E. Tarjan, "A Separator Theorem for Planar Graphs," *SIAM Journal on Applied Mathematics*, vol. 36, no. 2, pp. 177-189, Apr. 1979.
18. R. J. Lipton and R. E. Tarjan, "Applications of a Planar Separator Theorem," *SIAM Journal on Computing*, vol. 9, no. 3, pp. 615-627, Aug. 1980.
19. K. Mehlorn, *Data Structures and Algorithms (vol. 3: Multi-dimensional Searching and Computational Geometry)*, 3 vols., Springer-Verlag, Heidelberg, Berlin, 1984.
20. J. S. B. Mitchell and C. H. Papadimitriou, "The Discrete Geodesic Problem," Manuscript, Dept. of Computer Science, Stanford Univ., Stanford, CA, 1985.
21. J. O'Rourke, S. Suri, and H. Booth, "Shortest Paths on Polyhedral Surfaces," *Proc. of the 2nd Annual Symp. on Theoretical Aspects of Computer Science (Lecture Notes on Computer Science 182)*, pp. 243-254, Springer-Verlag, New York, Jan. 1985.
22. M. H. Overmars, "The Locus Approach," Tech. Rep. RUU-CS-83-12, Computer Science Dept., Univ. of Utrecht, Utrecht, the Netherlands, Jul. 1983.
23. F. P. Preparata, "A New Approach to Planar Point Location," *SIAM Journal on Computing*, vol. 10, no. 3, pp. 473-482, Aug. 1981.

24. M. I. Shamos and D. Hoey, "Closest-point Problems," *Proc. of the 16th IEEE Annual Symp. on Foundations of Computer Science*, pp. 151-162, Oct. 1975.

25. M. I. Shamos, "Computational Geometry," Ph.D. dissertation, Dept. of Computer Science, Yale Univ., New Haven, CT, 1978.

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

27. C. Verrilli, "One Source Voronoi Diagrams with Barriers, a Computer Implementation," Tech. Rep. IPL-TR-060, Image Processing Lab., Rensselaer Polytechnic Inst., Troy, N.Y., Feb. 1984.



```
Source face no.:                  1
Goal face no.:                    6
Source point coords.:              0.000     0.500     0.500
No. of developments:             24

Development sequences:
  1:   6   5   1                 13:   6   4   5   2   1
  2:   6   2   1                 14:   6   5   2   3   1
  3:   6   3   1                 15:   6   3   2   5   1
  4:   6   4   1                 16:   6   4   3   2   1
  5:   6   5   4   1             17:   6   5   4   3   2   1
  6:   6   2   5   1             18:   6   2   5   4   3   1
  7:   6   3   4   1             19:   6   3   4   5   2   1
  8:   6   4   5   1             20:   6   4   5   2   3   1
  9:   6   5   2   1             21:   6   5   2   3   4   1
 10:   6   2   3   1             22:   6   2   3   4   5   1
 11:   6   3   2   1             23:   6   3   2   5   4   1
 12:   6   4   3   1             24:   6   4   3   2   5   1
```

**Fig. 1.** BOUNDARY FINDPATH (locus) on a cube. Only the partition of one face is shown.

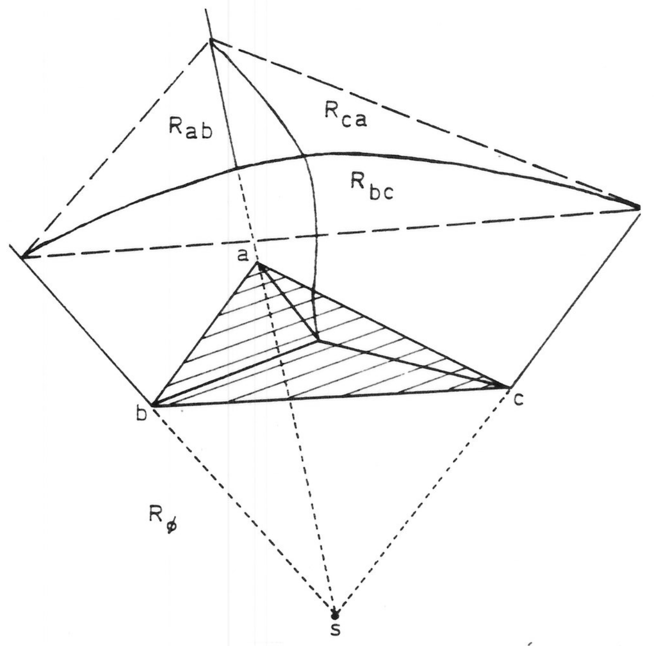
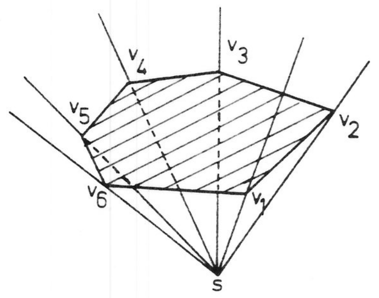


**Fig. 3.** Partitioning 3-space around a triangle. The curved surfaces are quartics.



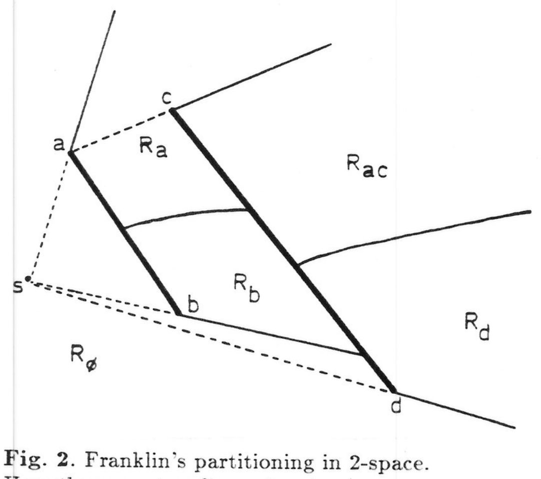


**Fig. 2.** Franklin's partitioning in 2-space. Here there are two linear barriers.

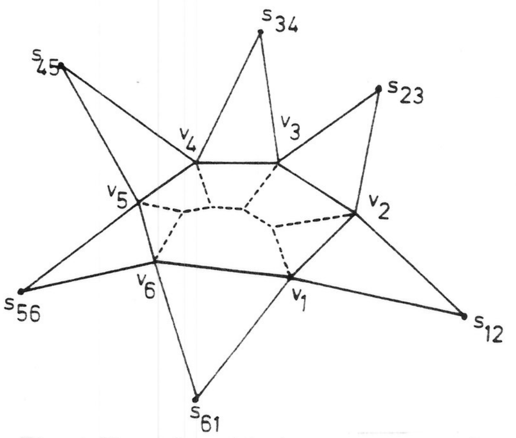


**Fig. 4.** Voronoi partitioning on a convex polygon. This is done by unfolding above pyramid.