Visual Computer

Voronoi diagrams with barriers and on polyhedra for minimal path planning*

W. Randolph Franklin,
Varol Akman, and Colin Verrilli

Electrical, Computer, and Systems Engineering Dep., Rensselaer Polytechnic Institute, Troy, New York 12180, USA

Two generalizations of the Voronoi diagram in two dimensions (E^2) are presented in this paper. The first allows impenetrable barriers that the shortest path must go around. The barriers are straight line segments that may be combined into polygons and even mazes. Each region of the diagram delimits a set of points that have not only the same closest existing point, but have the same topology of shortest path. The edges of this diagram, which has linear complexity in the number of input points and barrier lines, may be hyperbolic sections as well as straight lines. The second construction considers the Voronoi diagram on the surface of a convex polyhedron, given a set of fixed source points on it. Each face is partitioned into regions, such that the shortest path to any goal point in a given region from the closest fixed source point travels over the same sequence of faces to the same closest point.

Key words: Computational geometry – Robotics – Minimal paths – Voronoi diagrams – Planar partitions

* This material is based upon work supported by the National Science Foundation under grants ECS-8021504 and ECS-8351942. The second author is also supported in part by a Fulbright scholarship

SPATIAL LOCATION AND CORRELATION ALGORITHMS are an essential aspect of computational geometry. Typical problems include determining which objects of a large set are close to one another, and the shortest path from a source to a goal, assuming some metric. As well as the practical applications of computational geometry in the area of computer graphics [19, 20], another important application to robotics that deserves attention is the determination of the most efficient way to move a robot arm from one point to another without colliding with any other objects in the workspace.

This paper presents a new construction intended to be of assistance in minimal path problems by extending the concept of the two-dimensional Voronoi diagram to handle more complicated geometries. Here we will extend the concept by allowing opaque barriers in the plane. A path, along which a distance is measured, must not pass through a barrier, but will travel alternately through free space in a straight line and bend around the end of a barrier as shown by Lozano-Perez and Wesley [45]. We will call the vertices where the path starts, stops, or bends "contact vertices", their ordered list the "contact list", and the path between two consecutive contact vertices a "leg".

Although initially the barriers are straight line segments, they may easily be generalized into polygons whose interiors are forbidden, and even into maze-like diagrams. With barriers, even the one point Voronoi diagram is of nontrivial complexity, although the total number of new edges in the diagram is linear in the number of barriers. The new edges may now be hyperbolic sections as well as straight lines. This is similar to Drysdale's generalized Voronoi diagram which can have parabolic sections [15].

Once the Voronoi diagram on the points and barriers has been calculated, the traditional operations such as closest point may be performed. However, here it is necessary to know the shortest path as well as the closest point. Thus, the one point Voronoi diagram is equivalent to a single source – multiple goals minimal path problem. There are many algorithms for determining which region of a planar graph contains a new point. They must be modified slightly to be useful here because of the hyperbolae, but the changes are expected to be minor.

An application of this construction is in robotics. Assume that we have a fixed scene where the robot always starts from the same point and must avoid the same barriers. However, we wish to move the

The Visual Computer (1985) 1:133–150

robot to a different goal point each time. Thus it is worthwhile to preprocess the scene so that the execution time of the algorithm on a new goal point is fast.

A second generalization of the Voronoi diagram is also useful. This is to use the surface of a convex polyhedron instead of a plane surface. A one point diagram is again nontrivial, and in addition to the closest point, we need the shortest path itself.

This problem arises naturally as a subproblem of the task of finding the shortest path in E^3 from a source to a goal in the presence of convex polyhedral obstructions. Such a path will alternately have two components: a straight line through free space, and a traversal along the surface of a polyhedron between where the path "lands" from free space, and where it "takes off" again.

In the following sections, we will see a review of the Voronoi diagram, a summary of the other shortest path problems, a summary of the planar point location algorithms, and then the forms of the new Voronoi diagrams along with algorithms to compute them.

Voronoi diagrams

The Voronoi diagram on a set S of n points in the plane is a partition of the plane into n regions, one containing each point [7, 29, 64]. The region around point $p \in S$ is:

$$\omega(p) = \{q \mid \forall r \in S - \{p\},\ d(p, q) < d(r, q)\}$$

that is, the part of the plane closer to p than to any other point of S. We can generalize this in two ways. First, we night partition the plane according to the k closest points, instead of the single closest point. Thus, for a subset H of S such that $|H| = k$ (cardinality of set H),

$$\omega(H) = \{q \mid \forall h \in H,\ \forall r \in S - H,\ d(h, q) < d(r, q)\}$$

In particular, for $k = n - 1$, we have the farthest point Voronoi diagram, so that for any new point we can easily determine the farthest existing point. These constructions will solve problems such as finding the smallest circle enclosing n points and finding the largest empty circle whose center is inside the convex hull of the n points. The Voronoi diagram of n points can be found in time $T = \theta(n \log n)$ as proved by Shamos [63].

A second form of generalization is to allow more general objects than points such as Drysdale's constructions [15]. Here we compute the diagram around finite line segments. This causes several complications: the regions have parabolic as well as straight edges, the regions may be concave, the dual of the diagram may be a multigraph (have more than one edge between two given vertices), and the construction time is a little slower [$T = \theta(n_\omega \exp(c\sqrt{\log n}))$ where c is a constant].

Other minimal path algorithms

There has been much previous work on this and related problems. The reader is referred to Akman [4] for a review and a long list of references. The minimal path problem is in some ways an extension of the well-known travelling salesman problem [56] (TSP) where we wish to determine the minimal cycle that traverses a given set of vertices in any order. Although the TSP is NP-complete, recent work by Kirkpatrick et al. [1] that draws inspiration from the crystallization of solids as they are slowly cooled in statistical mechanics produces good approximate optima in $\theta(n \log n)$ time. Although there is a technical difficulty arising from the distance metric, the Euclidean version of the TSP is also NP-complete [25, 26, 50]. Since the fastest known algorithms for NP-complete problems require an execution time that is exponential in the size of the input, they are too slow to use.

Reif [54] considers the existence problem of whether there is any path at all to move a polyhedral object around some barriers from a source S to a goal G. He allows the object to be rotated if necessary, just as a sofa must be rotated to move it around a corner in a narrow hall. He discretizes the problem by allowing the object's size to be slightly fuzzy, and shows that there is a polynomial time algorithm to solve it in both E^2 and E^3. If the object can have several rigid components connected at flexible joints, as in an articulated arm, he gives a polynomial space algorithm, and shows that the problem is P-space hard. Recently, Reif and Storer [55] offered several new insights into the shortest path problem.

Some useful references in the robotics field from which this problem is derived follow. Paul [51] considers modelling and trajectory calculation, and in another paper [52] gives a good summary of

robotics. Guibas and Yao [30] discuss a problem related to collision avoidance, namely, copying overlapping windows. Udupa [66] studies several practical collision detection and avoidance problems using approximation techniques.

The first method for finding the minimal path was given by Lozano-Perez and Wesley [45] who presented an algorithm for a point moving between S and G in E^2, which is illustrated in Fig. 1. It constructs a graph whose vertices are S, G, and all the vertices of the barriers. Its edges are those straight lines between the vertices that do not pass through a barrier. The algorithm simply finds a minimal path in this graph between S and G, in this case $S-V_8-V_2-G$. However, that can be slow for large scenes because if the barriers have n vertices, then the graph can have up to n^2 edges. Determining which edges do not intersect a barrier requires intersecting each one with all the barriers. This will take $\theta(n^2)$ time using the polygon visibility algorithm of El Gindy and Avis [17]. Lozano-Perez and Wesley [45] also present a method for finding minimal paths for moving a polygonal object around barriers, provided that the object is allowed to translate but not rotate. This consists of shrinking the object to a point and simultaneously enlarging all the barriers. This reduces the object minimal path problem to the point minimal path problem which they know how to solve. The method is illustrated in Fig. 2. Of course, this reduction method can also be combined with any other point minimal path algorithm. The Lozano-Perez and Wesley algorithm also provides an approximate answer in E^3 or when the object is allowed to rotate.

Lozano-Perez [43, 44] then presents the "configuration space" approach whereby a problem in d dimensions, where an object is allowed to rotate to avoid barriers, is converted to another problem in $d(d+1)/2$ dimensions where the object can only translate. He presents approximation techniques for solving the new problem. The configuration space approach works as follows. An object in E^2 that can translate and rotate has three degrees of freedom, x, y, and θ. The legal values of these are constrained by the barriers in the scene. We could also consider x, y, and θ to be three Cartesian coordinates of a point that must move while avoiding some barriers in E^3, and then use an E^3 minimal path algorithm. There are two problems with this approach. First, the straight sided barriers in E^2 become curved E^3 barriers, with curves involving trigonometrical functions and second, if S and G specify just the position and not the orientation of the object in E^2, then they will map into lines in E^3.

Sharir and Schorr [65] mention many useful results on the nature of minimal paths on a convex polyhedron with n vertices and a fixed point on it. For example, they show that the polyhedron can be preprocessed in $T=O(n^3 \log n)$ to produce a data structure taking $O(n^2)$ space with the help of which one can find in $O(n)$ time the minimal path along the surface of the polyhedron from a new point to the fixed point. However, to arrive at this result they employ extremely complex data structures.

Brooks [8] solves the path planning problem using a good representation of free space. Like Reif, Donald [14] considers the complexity of path plan-

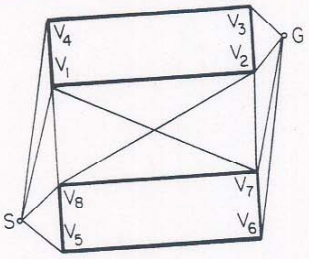


Fig. 1. Finding a minimal path by constructing a graph with edges between all pairs of visible vertices

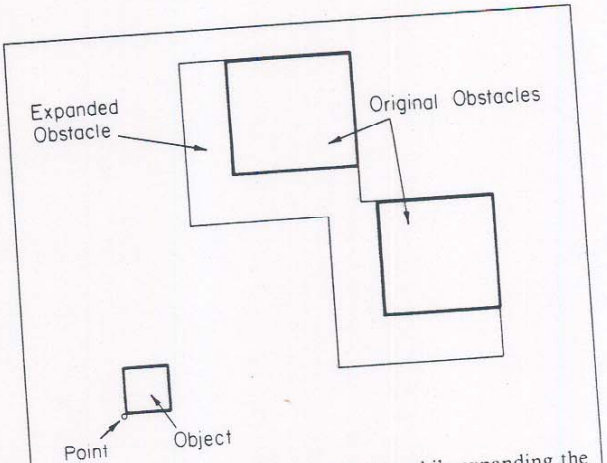


Fig. 2. Shrinking the object to a point while expanding the obstacles to maintain the same minimal path

ning and automated structural design problems. Hopcroft et al. [31] deal with a more complex problem, namely, the complexity of motion planning for multiple independent objects and prove that the general problem is P-space hard. Lee and Preparata [36] treat the specific problem of rectilinear barriers instead of general barriers and give an efficient method. Although discovered independently, their method closely resembles the method summarized here and by Franklin [21]. O'Dunlaing and Yap [48] consider the motion planning problem for a disc and give a Voronoi-based solution. A recent work by O'Rourke et al. [49] also contains results on minimal paths on polyhedral surfaces. Finally, in a series of highly interesting papers [58–62] based on algebraic and differential topology, Schwartz, Sharir, and Ariel-Sheffi solve several instances of what they call, the "Piano Mover's" problem.

Location of a point in a plane partition

Once we have preprocessed the scene for a given source S and barriers, for each goal G we must determine in which region it is. There are various algorithms, all of which however, are for planar graphs with straight edges. Lee and Preparata [35] present an algorithm that, after $\theta(n \log n)$ preprocessing time, requires $\theta(\log^2 n)$ search time. It divides the graph into an ordered list of monotonic chains of the edges of the graph. The vertices of each chain are monotonically increasing in y. If one chain is to the left of another in the list and we cut the chains with any scan line ($y=$constant), then the first chain does not intersect the scan line to the right of the second. Each edge of the graph, except some of the exterior ones, is used in one or more chains. A point is located by doing a binary search to find which two chains it lies between. Each step of that search requires determining which side of a given chain a point lies on; this requires a binary search of that chain.

Figure 3 illustrates this method for a graph with eight regions. The first level chain (the heavy solid line) separates it into two subgraphs of four regions each. Each of the two second-level chains (medium solid lines) separates one of those subgraphs into two smaller subgraphs of two regions each, while each of the four third-level chains (light solid lines) separates one of these four subgraphs into two individual regions. The dotted lines represent three exterior edges that were not used in any chain. Note that a given edge of this graph can be used by as many as $\log n$ different chains.

Lee and Yang [37], and Lee and Drysdale [34] summarize various point-location nethods including one with a $\theta(\log n)$ search time, while Shamos [63] and Brown [9, 10] also summarize various ways. Lipton and Tarjan [41, 42] describe another brilliant method with far-reaching consequences. Theirs is very complex and probably of only theoretical interest, but has $\theta(\log n)$ search time after an impressive $O(n)$ preprocessing time. Also the existence of such a method with linear preprocessing time may be a clue to more practical efficient algorithms. Kirkpatrick [33] gives an algorithm that is based on triangulating the planar graph in preprocessing time $T=\theta(n \log n)$ to give a fast search time of $T=\theta(\log n)$.

Most of the algorithms mentioned above that have a binary search time of $\theta(\log n)$ time can be improved by using an interpolation search instead [53]. This requires first, that the probability distribution of the ordered list being searched be either known or bounded in the sense that the ratio of the probabilities of two elements occurring does not increase without limit as the set size increases, and second, that the elements are not correlated in certain ways. In return, the search can be conducted in expected $\theta(\log \log n)$ time. Interpolation search differs from binary search in that the list is probed at the interpolated element's expected location instead of at the middle. It is noted however that both Kirkpatrick's algorithm, and Lipton

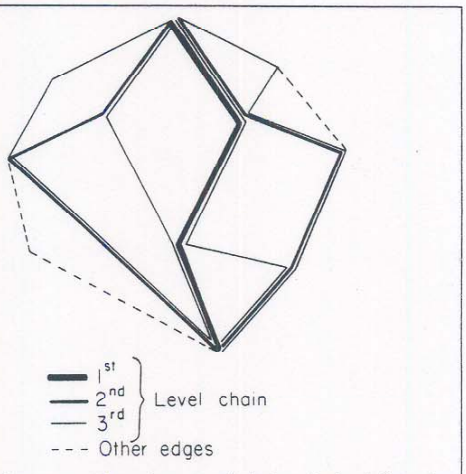


Fig. 3. A planar graph and a set of chains derived from it

and Tarian's algorithm decompose the planar point location problem in a very rigid way and make the interpolation search quite difficult to be incorporated into their algorithms.

Since the borders of the regions are hyperbolae in E^2 (or quartic surfaces in E^3), we need an efficient means of intersecting second or higher order curves. In general we can reduce a system of two second order equations in x and y to one fourth order equation, which we can solve either iteratively or with the closed form solution [3]. Levin [38–40] presents a simpler closed form method that uses second order parametric equations. It is also possible to find efficient piecewise-linear approximations and intersect them instead. Franklin and Akman [23] deal with partitioning E^3 in the presence of solid polygons so that for a goal point in a given region the sequence of obstacle edges through which the shortest path passes is the same. Converting the planar search algorithms to handle hyperbolae is conceptually simple. In general, the major use that is made of the edges' straightness is to tell which side of the line a point falls. This operation can also be performed on a hyperbola by substituting the point into the implicit equation of the hyperbola and testing the sign of the result. These planar search algorithms may also assume that a graph edge is monotonic, which hyperbola sections may not be. This can be remedied by splitting such hyperbolic edges into two parts at their maxima and observing that the chains must continue upward from the maxima (so new edges must be added). Finally, it should be remarked that Kirkpatrick's algorithm also requires convexity of the regions for triangulation to work properly.

Voronoi diagrams in the plane with barriers

Now we are ready for the first new construction. First, we will consider the Voronoi diagram on just one point, initially with one barrier, then with more. Since the barriers are assumed to consist of straight line segments, the partition of the plane caused by all the barriers will be some composition of the partition caused by one line segment. The general case is shown in Fig. 4. Here we have a source S and an barrier line λ from A to B. The distances from S to A and B are $|SA|$ and $|SB|$, respectively. The partition has three regions R_1, R_2, R_3 separated by the semi-infinite lines σ_1, ρ, and σ_2. Here σ_1 and σ_2 are the "shadow lines" of A and B respectively, and ρ, the "ridge line", is a section of a hyperbola. A general point P on ρ can be reached equally quickly from S by going around A or around B. For the minimal path that bends at A (or B) the contact lists is (S, A) [or (S, B)] and the legs are (SA, AP) [or (SB, BP)]. Accordingly,

$$d(P) = \text{length of minimum path from } S \text{ to } P$$
$$= |SA| + |AP| = |SB| + |BP|$$

so that

$$|AP| - |BP| = |SB| - |SA| = \text{constant} = c$$

This is the equation of a hyperbola with foci at A and B. If λ is horizontal, then as ρ approaches infinity, this hyperbola tends toward a straight line with slope

$$-(|AB|^2 - c^2)^{1/2}/c, \quad \text{if } c > 0$$

Some examples of scenes with multiple barriers are shown in Figs. 5 and 6. These were produced by Verrilli [68] using an interactive graphics program that he implemented to simulate the one source – multiple goals Voronoi diagrams with barriers [21]. Each region has an "associated vertex" which is the closest point of that region to S, and is also the last contact vertex before G on the minimal path to any goal G in the region. These associated vertices are either endpoints of the barrier lines ("barrier vertices"), or the source S.

Since these associated vertices are points of the plane in their own right, each of them is contained in some region. Since this region has its own asso-

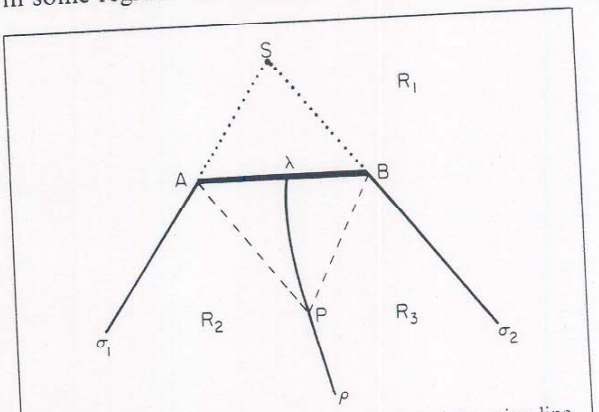


Fig. 4. The general partition caused by one obstruction line, causing a hyperbolic boundary

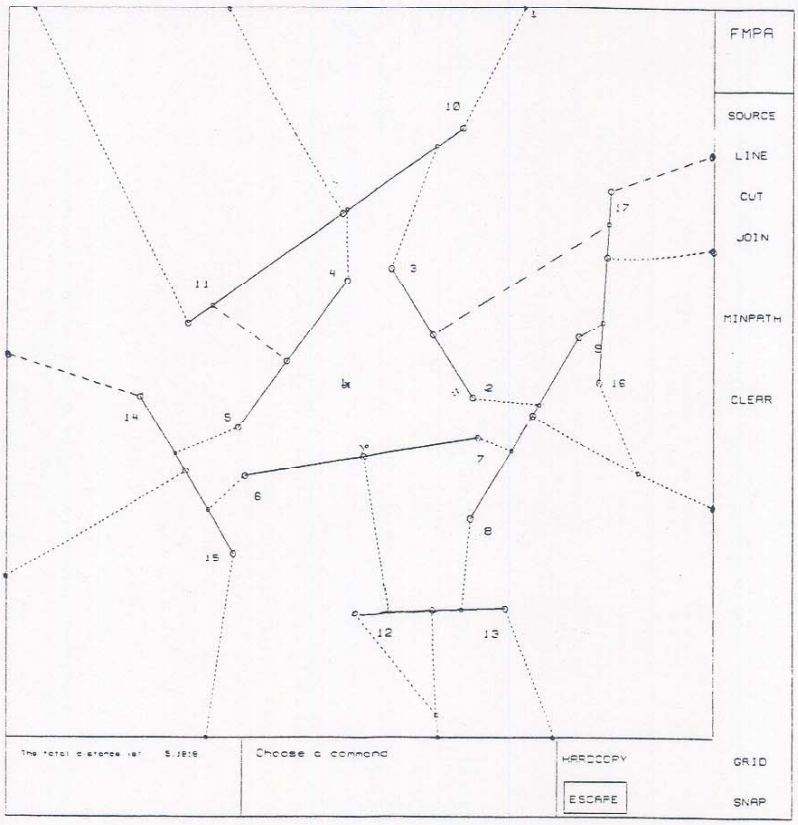


Fig. 5. A scene with multiple barriers

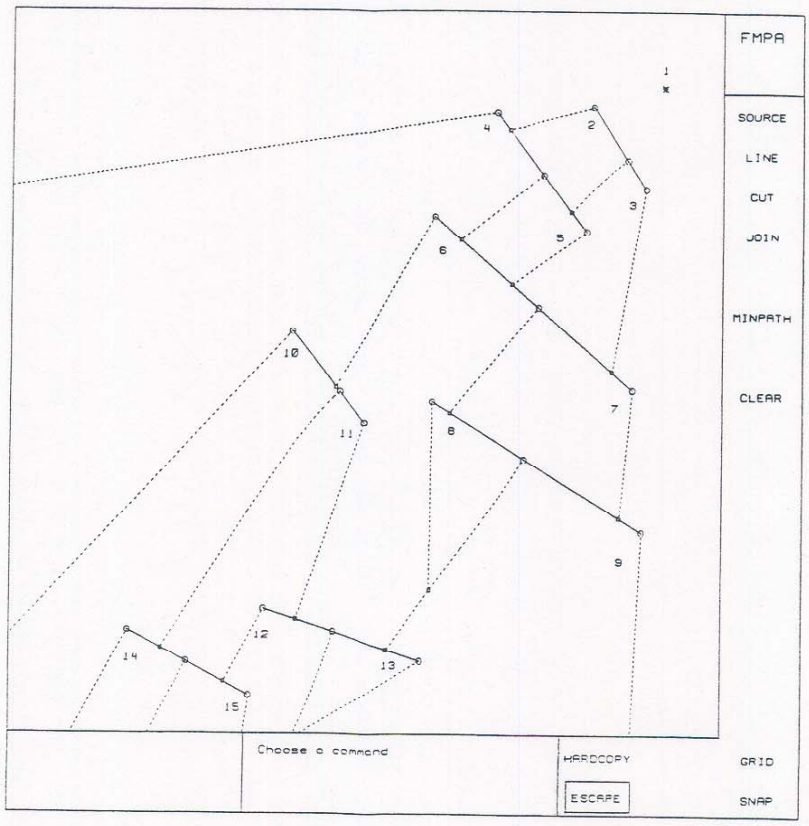


Fig. 6. Another scene with multiple barriers

ciated vertex, which is different from the vertex unless we are at S, we will get a chain of vertices from the source to the goal. This is precisely the list of contact vertices in the shortest path from S to G. Following our notation for Voronoi regions, we will represent regions with the letter ω. Thus region $\omega(V)$ has associated vertex V.

This property of regions' associated vertices being in other regions also shows us the form of the planar graph. Consider Fig. 7, which shows one end of a barrier, λ, with its barrier vertex, V. V is in region $\omega(U)$ whose associated vertex is U. There must be a shadow line, σ, separating region $\omega(V)$ from $\omega(U)$, that will be the extension of the straight line UV. σ will continue either until it intersects some other line in the graph or to infinity.

It is impossible for one region to have two different shadow lines as its borders since each shadow line has an associated vertex and one region cannot have two associated vertices, which would imply two optimal paths. In this case, there will be a ridge line that separates the original region into two smaller regions depending on which way the optimal path should go from each point in the original region. As illustrated in Fig. 8, there are two equal optimal paths from each point on the ridge line. A ridge line will start from a barrier at one edge of the region and continue until it intersects another line, if ever.

It is not necessary for the two associated vertices of a ridge line to be barrier vertices of the same barrier (Fig. 8). Here the ridge line will be an hyperbola with foci at the associated vertices of its two adjacent regions.

If a shadow or ridge line does not extend to infinity, it may meet another line such as a barrier, or another ridge or shadow line. Several examples of this are shown in Fig. 5 and 6. A line that hits a barrier simply stops. However, if two lines, each a ridge or shadow line, meet, then they both stop and a new ridge line starts from that point. See Fig. 9 where ρ_1 and ρ_2 meet at P. They separate regions $\omega(A)$ and $\omega(B)$, and $\omega(B)$ and $\omega(C)$, respectively. Note that ρ_1 and ρ_2 must be edges of the same region if they are to meet. Region $\omega(B)$ must end at P since below it, it is faster to get to the source via A or C. In fact, a goal G that is below P and to the right of ρ_1 would find it faster to go via B than via A. However since G is to the right of ρ_1, it must also be to the right of ρ_1, and so it would be even faster to go via C and then D. Thus $\omega(B)$ must terminate at P.

The new line ρ_1 will separate $\omega(A)$ from $\omega(C)$ and will be the locus of points with two equal optimal paths, via A or C. The foci of the hyperbola ρ_3 will be A and C.

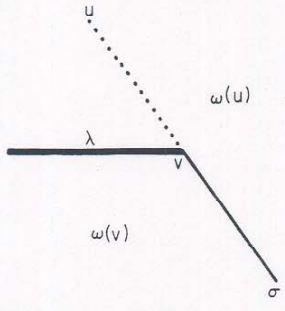


Fig. 7. The origin of a shadow line

Multiple sources and barriers

Given the Voronoi diagram with one source and multiple barriers, the next generalization is to multiple sources. A special case of this, with no barriers, will be the traditional Voronoi diagram. The diagram for many source points and barriers must continue to provide the shortest path in addition to identifying the closest source.

In the single source case, ridge lines were necessary because one region could not have two associated vertices. Remembering that a source is the associated vertex for the region from which paths from the goal go straight to the source, multiple sources may cause some region to have two associated vertices, and so it will need a new ridge line to partition it. The only generalization is that this ridge line might never touch a barrier, but might begin and end at intersections with other ridge lines. This is the case if there are no barriers at all.

Figure 10 is an example of two sources, S_1 and S_2, and a barrier, $\lambda = AB$. If the barrier were absent, then the Voronoi diagram would be the infinite perpendicular bisector of S_1S_2. However, λ means that some points to the right of the bisector are now closer to S_1. In fact, parts of the perpendicular bisector remain as the ridge lines ρ_1 and ρ_2, but now there are two shadow lines and three more ridge lines in between. From point

C there are three minimal paths with contact lists S_1-C, S_2-A-C, and S_2-B-C.

As the next level in complexity, the polygonal barriers are handled as follows. A polygonal barrier is more than just the set of its edges. Figure 11 shows the diagrams resulting from two adjacent barriers, λ_1 and λ_2 in the case where we either can (Fig. 11a) or cannot (Fig. 11b) travel between λ_1 and λ_2. However, the modification to the diagram is obvious: if λ_1 and λ_2 adjoin at point P with no passage between them, then if P is in region $\omega(S)$, a shadow line is drawn from P only if it would be on the same side of λ_1 and λ_2 as is S. This is not the case in Fig. 11b, but is in Fig. 12.

The concept of polygonal barriers can be extended to mazes as is shown in Fig. 13. If there is more than one path to S, we will find the shortest path from every point in the maze.

We will close our discussion of Voronoi diagrams with barriers with a general inductive method for constructing the partition of a plane:

1. Initially we have no obstacles and one region with the contact list (S) and visible source S. We refer the reader to Fig. 14 throughout this discussion.
2. To add an obstruction $\lambda = AB$ to an existing partition, assume without loss of generality that A and B are in regions R_1 and R_2 with visible sources S_1 and S_2, respectively. Assume that the two new regions with visible sources A and B are called R_3 and R_4, respectively.
3. If R_1 differs from R_2, any boundary lines that intersect λ stop there. In Fig. 14, there are three such lines.
4. Create a new straight boundary line, B_1, starting from A that is an extension of S_1A, and another, B_3, extending from B that is an extension of S_2B.
5. Create a new hyperbolic boundary line, B_2, starting at a point on λ and going down. If P is a general point on it, then we have $|AP|-|BP| = d(B)-d(A)$.
6. B_1 may eventually cut a boundary line, B_4, of

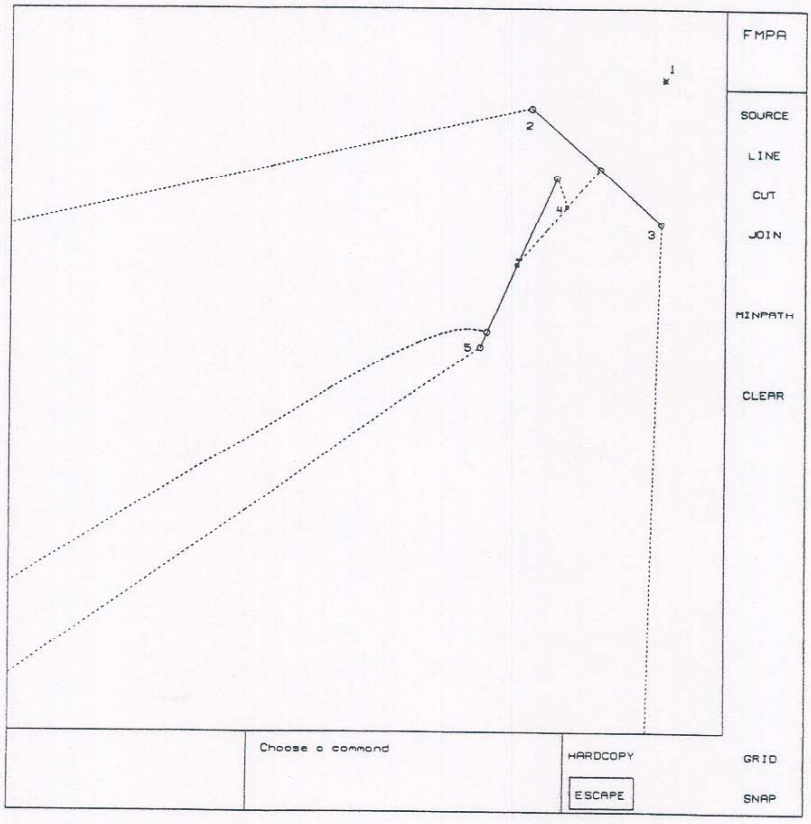
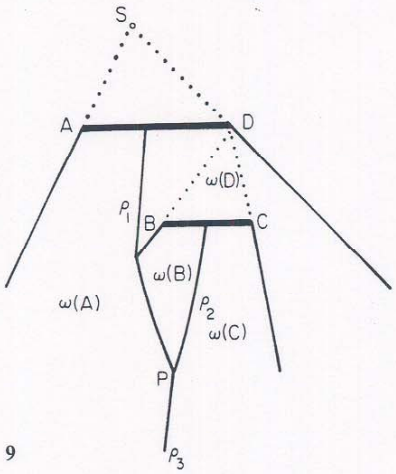
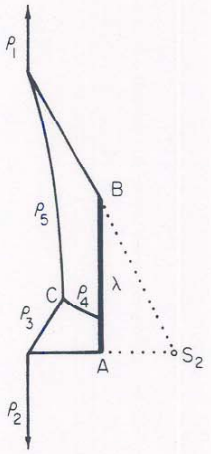


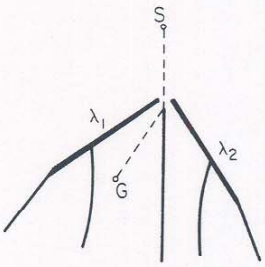
Fig. 8. The vertices adjacent to a ridge line are not always on the same barrier



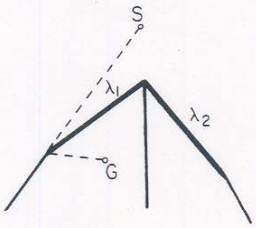
9



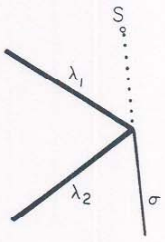
10



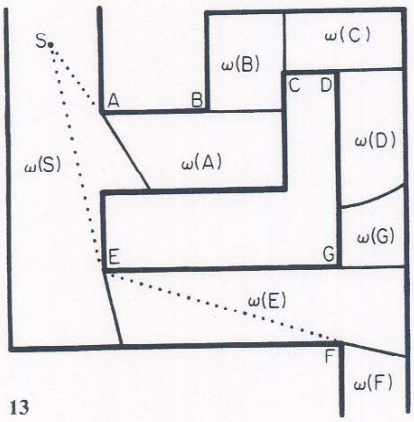
11a



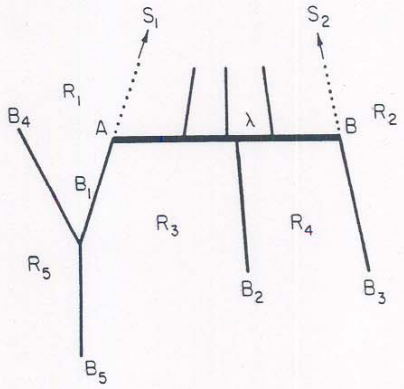
11b



12



13



14

Fig. 9. Two intersecting ridge lines in a scene with two barriers

Fig. 10. Two sources and one barrier

Fig. 11a, b. Two adjacent barrier lines with paths allowed (**a**) or not allowed (**b**) between them

Fig. 12. Two adjacent barriers where the shadow line of their common vertex is on the same side of them as the source

Fig. 13. Multiple barriers forming a maze

Fig. 14. How a general new obstruction causes new boundary lines to form

R_1. Assume that the region on the other side of B_4 is R_5. At this point, B_1, B_4, and R_1 stop and B_5, which separates R_3 and R_5 which are now adjacent, starts. Let the visible source of R_5 be S_5. If P' is a general point on B_5, then we have $|S_5P'|-|AP'|=d(A)-d(S_5)$.

7. Repeat step 2 for each obstruction.

Minimal paths on a polyhedron: instances

Now we will describe our second new construction. Let P be a convex polyhedron and S and G two points in the Euclidean space E^3. Assume that $S,G\in \mathrm{Ext}(P)\cup \mathrm{On}(P)$ throughout this paper. [$\mathrm{On}(P)$, $\mathrm{Int}(P)$, and $\mathrm{Ext}(P)$ denote the surface, the interior points, and the exterior points of P, respectively.] In the sequel, we first present algorithms to compute the minimum-length S-to-G (or vice versa) paths which do not interfere with P (i.e., at most touch P but never intersect it) for the following two instances (FP is an abbreviation for "Findpath" – a name used mostly in artificial intelligence [47] for this kind of path planning problems):

[FP1] $S,G\in \mathrm{On}(P)$ and both points are given; and

[FP2] S and/or $G\in \mathrm{Ext}(P)$ and both points are given.

Finally, after these preparations, as a third and more interesting instance, we solve the following single source – multiple goals version. Let S be fixed and G be varied inside the workspace. This corresponds to a manipulator consuming a pile of objects by continuously moving them to different locations in the scene. Clearly, the symmetrical case of a manipulator picking up objects from several piles in order to assemble them in a fixed location is transformable to this instance by simply trading the roles of S and G. Assume that S and the particular face F_G of P on which G is guaranteed to be located are given. It is emphasized that the exact location of G on F_G is not yet known. Clearly, the aim is to do some preprocessing using this limited information (i.e., P, S, and F_G) so as to compute the S-to-G minimal paths efficiently for specified "query" points which will generically be denoted by G. In our terminology:

[FP3] $S,G\in \mathrm{On}(P)$; S is given while G is not (but F_G, the face of P that G will belong, is known). We are allowed to preprocess P to quickly answer later queries that specify many $G\in F_G$ and ask for the S-to-G minimal paths.

Parts of the following material can also be found in Franklin and Akman [22].

Solution of FP1

In the following, α_0, α_1, and α_2 denote the number of vertices, edges, and faces of P, respectively. We reserve the terms "edge", "vertex", and "path" only for polygons and polyhedra; for graphs we use the less common terms "link", "node", and "sequence" in order to avoid potential confusions [12]. For polyhedral definitions, we follow Grunbaum [28].

Throughout this paper, we assume that the line segment SG is not the shortest path since there is a fast algorithm due to Chazelle [11] to detect this, that is, to intersect a line segment with a convex polyhedron P. In fact, Chazelle's algorithm first detects whether a given line λ intersects P. The output is data giving the common point(s) of λ and P if they do intersect or null if they are disjoint. The algorithm works in time $O(\log^2 \alpha_0)$ and requires a special representation of P which demands $O(\alpha_0^2)$ operations to reach from its standard representation.

In the following, we summarize a simple but useful fact about the shortest path on a dihedral angle. This lemma can be found in various places including a book by Lyusternik [46] on variational calculus and many mathematical puzzle books [6, 13].

Lemma (Dihedral principle). *If S and G are on different faces of a dihedral angle D, then the shortest path between them is $SX\cup XG$ where X is a point on the common line CC' of the faces such that $<SXC=<C'XG$ where "$<$" denotes a planar angle.*

Proof. Trivial. Algorithmicly, X is found as follows. Rotate G about CC' until it touches the plane of S (but specifically to the hale-plane described by CC' not including S). Thus G is transformed to G'. This is called a "development" of D into the plane. Draw SG' and intersect it with CC' to find X. When we fold the faces back to their origi-

nal position the minimal path is placed into three dimensions on D again.

To solve FP1 we will develop a partly geometrical and partly graph-theoretical machinery. By definition, a polyhedron is a figure in space formed by a finite number of polygons situated in a certain mutual relationship. This can be represented by a system of polygons ("evolvent" or "planar polygonal schema") in the plane as shown by Efimov [16], as well as Frechet and Fan [24].

Given a polyhedron P, associate a polygon with each of its faces, this polygon being subject to the single condition of having the same metrical form as the associated face of P. We thus obtain a finite number of polygons in the plane each of which is separated from the others. We will couple the edges of these polygons in pairs in such a way that two "coupled" edges in the plane lead to the same edge of P. Now denote each coupled edge in the plane by the same letter. This does not yet suffice to describe the mutual relationship of the faces of P. To make it precise, each edge in the plane must be oriented by placing an "arrow" on it in an arbitrary sense, except that for two coupled edges the heads of the two arrows on them coincide when they are identified in forming the corresponding edge of P.

The evolvent Δ of P, obtained as described above, defines the intrinsic geometry of P in the sense that for a given Δ it is possible to determine the minimal paths between two points on P [5]. In our solution of FP1, Δ is the main geometrical tool. It is noted that Δ satisfies the following conditions:

I) The coupling ("gluing" in Aleksandrov's terminology) of polygons may occur either at edges or vertices

II) If two polygons F_i and F_j are glued at vertex v, then they are either glued together along edges containing v or there exists a sequence of polygons, starting at F_i and ending at F_j, which are glued to one another along edges containing v

III) Each edge is glued to exactly one edge

IV) Any two polygons are joined by a chain of polygons glued along edges (the "connectivity" condition)

In addition to Δ, we need another simple but useful tool. Define the "face adjacency" (or shortly, face) graph $\Omega=(V, E)$ as an undirected graph with unit link length where $V=\{v|F_v$ is a face of $P\}$ and $E=\{(v1, v2)|F_{v1}$ and F_{v2} are adjacent along an edge of $P\}$.

Let t be a minimal path on P from S to G. The faces that t touches while going from S to G define the "face visit sequence", $F_S, \ldots, F_G$, where each face in this sequence is adjacent to its preceding and following neighbors.

Lemma. *A face visit sequence in Ω associated with a minimal path on P is simple (loop-free).*

Proof. After recalling the fact that each face of P is also convex, one would further shorten the portion(s) of the path corresponding to the cyclic sequence(s). But then the given path is not minimal.

Lemma. *There may be $O(n!)$ simple sequences between two specified nodes of an undirected graph with n nodes.*

Proof. In the worst-case the graph is complete. (But see the argument below.)

In practice, things are more encouraging than what is predicted by the above lemma. The number of links in Ω is much less than the number of links in a complete graph with the same number of nodes. This is due to the observation that the surface of a convex polyhedron has the structure of a planar graph; hence it can have only a linearly growing number of links in terms of its nodes. This $O(n)$ factor of reduction in the number of links renders less freedom in moving from one node to another and helps reduce the number of simple sequences considerably.

Before we present our algorithm for FP1 we emphasize that Δ is a geometrical data structure. Hence it is possible, for instance, to obtain from Δ such information as "F_1 and F_4 share edge e", or "F_6 and F_3 are disjoint." With these examples, a convenient internal representation for storing an evolvent should be evident (cf. Abelson and DiSessa [2]). On the other hand, Ω contains only topological information; that is, it is less informative of shape. (A parallelepiped has the same Ω as a cube.) Normally, an adjacency list structure is the best representation scheme for storing Ω.

After above preparations we can now present our algorithm for FP1:

Algorithm A1

1. Given S and G, find the faces F_S and F_G holding them. Assume that they are different; otherwise we are finished with path SG.
2. Mark these two faces in Ω and enumerate all simple sequences between these nodes in any order.
3. For each sequence (which may be thought of as a face visit sequence) compute the associated development of the relevant faces. This is accomplished using the information provided by Δ.
4. For each development compute the minimal path between the images of S and G and finally choose the smallest one(s) among them.

In Figs. 15–17 we demonstrate this algorithm on a simple polyhedron – a cube. Figure 15 depicts the evolvent and the face graph of the given cube, Fig. 16 lists the first few of the simple face visit sequences, and finally, Fig. 17 shows the corresponding developments for those sequences given in Fig. 16.

To the best of our knowledge, counting the number of simple sequences between two nodes of a graph is a difficult problem. Our belief is due to the fact that given graph $\varphi=(V, E)$, $|V|=n$, $|E|=m$, length $l(e)\in Z^+$ for each $e\in E$, specified nodes $S, G\in V$, and positive integers b and k, it is stated by Garey and Johnson [25] that the problem "Are there k or more distinct simple paths from S to G in φ, each having total length b or less?" is NP-hard. Yet, this problem can be solved in pseudo-polynomial time (polynomial in n, k and $\log b$) and hence in polynomial time for any fixed value of k. Although this problem is not known to be in NP, the corresponding enumeration problem is $\#P$-complete as proved by Valiant [67]. $\#P$ is the class of functions that can be computed by counting Turing machines of polynomial time complexity. This class has computationally equivalent counting problems that are at least as difficult as

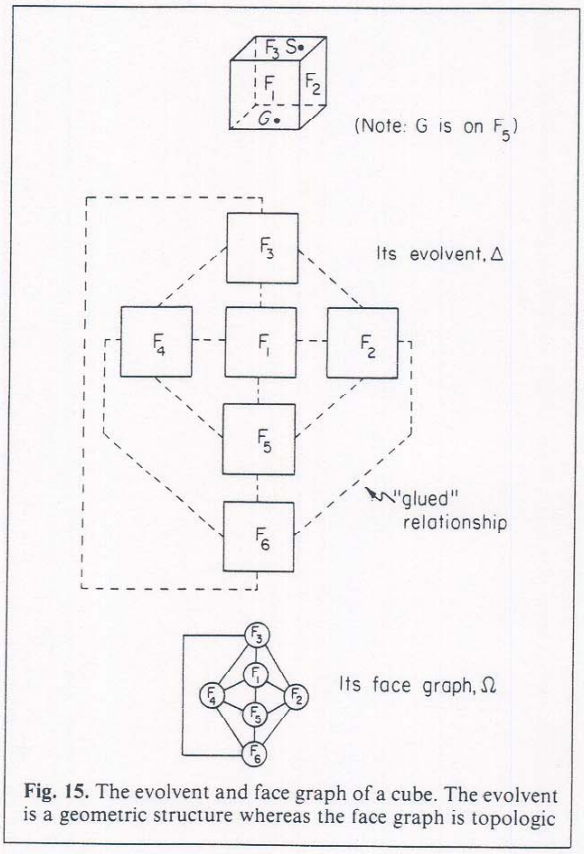


Fig. 15. The evolvent and face graph of a cube. The evolvent is a geometric structure whereas the face graph is topologic

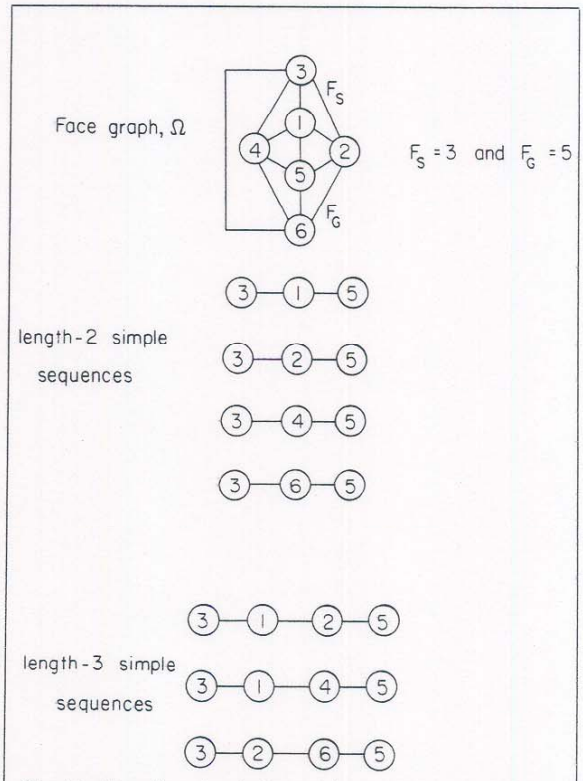


Fig. 16. First few simple face visit sequences on the face graph of the cube depicted in Fig. 15

the NP-complete problems. Valiant's # P-completeness proof is for the following:

Problem: S-T paths (i.e., self-avoiding walks) in graphs.
Input: Undirected φ (V, E) and $S, T \in V$.
Output: Number of paths from S to T that visit every node at most once.

Basically, the proof consists of showing that if this problem were not # P-complete then counting the number of Hamiltonian circuits in a graph would be a polynomial process. It is emphasized that # P-completeness of S-T paths still holds even when φ is planar.

In a paper by Yen [69] an $O(kn^3)$ time algorithm is given to enumerate the first k simple sequences in increasing path length. Recently, Katoh et al. [32] gave a new improved algorithm which works in time $O(kf(n, m))$ if the shortest sequences from one node to all the other nodes are obtained in $f(n, m)$ time. Since $f(n, m)$ is at most $\min[\theta(n^2), \theta(m \log n)]$ this algorithm is more efficient than Yen's.

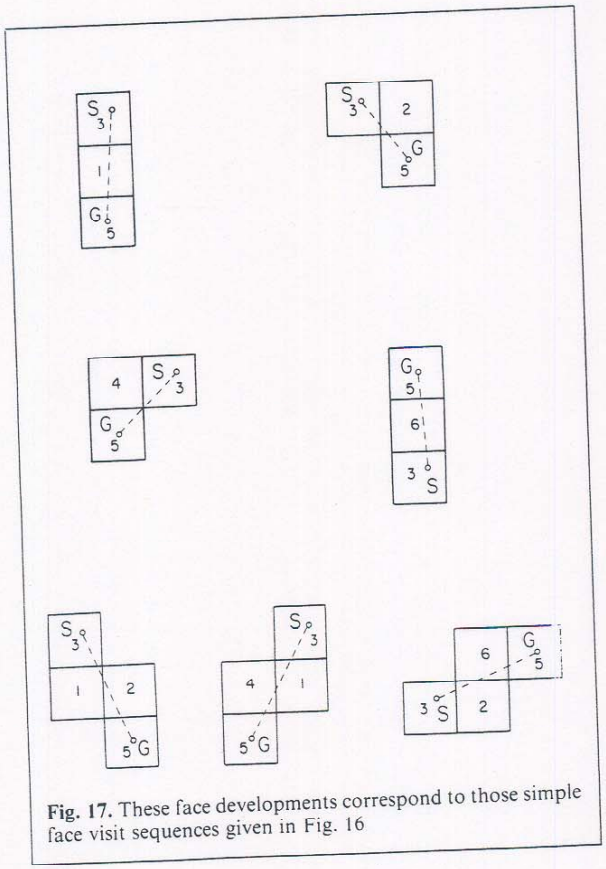


Fig. 17. These face developments correspond to those simple face visit sequences given in Fig. 16

Solution of FP2

When S and/or G are off a convex polyhedron we obtain the instance FP2. In this case, it is still possible to apply the method of solution summarized in the previous section but only after a simple transformation of P to another polyhedron P' which now has S and G necessarily on its surface. The transformation is given as follows:

Algorithm A2

1. Compute the two visible outlines Z_S and Z_G of P from the viewpoints S and G, respectively. These are generally nonplanar polygons which have as edges some edges of P.
2. Construct a new polyhedron P' which is the combination of three polyhedra, $P' = Q_S \cup P \cup Q_G$. Here Q_S is the pyramid having S as apex and Z_S as its base outline, and Q_G is the pyramid having G as apex and Z_G as its base outline. Note that these pyramids have generally concave bases consisting of the visible faces of P from S and G, respectively.

The correctness of Algorithm A2 follows from the following properties:

Lemma. *The planar projections of Z_S and Z_G are convex polygons.*

Proof. Due to fact that a convex polyhedron casts a convex shadow when illuminated by a point source.

Lemma. *P' is also a convex polyhedron.*

Proof. Take two points X and Y inside P'. If $X \in \text{Int}(Q_S)$ and $Y \in \text{Int}(P)$ then XY is always inside $Q_S \cup P$ for Y will always be inside the infinite pyramid originating from S. A similar argument shows that the union of Q_G and P is also convex. The only unproved case is when $X \in Q_S$ and $Y \in Q_G$. Let us check the dihedral angles of P' where Q_S and P, and Q_G and P are joined. Clearly, those angles are all nonreflex implying that XY must be totally contained inside P'.

The following result shows that after the problem-reduction step accomplished by Algorithm A2 (i.e., the computation of P' from P) the complexity of this instance is the same as with FP1.

Lemma. *Size* $(P') = O[Size\ (P)]$.

Proof. The "size" of a polyhedron can be defined as the number of its vertices or edges or faces depending on choice. In the case of a convex polyhedron they are all asymptotically equivalent since a convex polyhedron can be embedded in the plane as a graph and since α_1 is at most equal to $3\alpha_0 - 6$ in a planar graph. Hence, let us choose α_1 as the size measure without loss of generality. From the previous lemma, Q_S and Q_G may contribute $\theta(\alpha_1)$ new edges in the worst-case.
How fast can we compute Z_S and Z_G? Below we summarize the computation of Z_S; Z_G is computed similarly. Find a point C_i inside every face of P; this can be done in $O(1)$ time per point. Using Chazelle's method, compute the intersections of P and SC_i, $i=1,\ldots,\alpha_2$. If the intersection corresponding to a particular C_i is null then F_i is visible from S; otherwise it is obstructed by other faces of P. In this manner all visible faces of P from S are found. This operation clearly takes $O(\alpha_2 \log^2 \alpha_0)$, or equivalently $O(\alpha_0 \log^2 \alpha_0)$ time.
To compute Z_S from this set of visible faces, we make a list consisting of the edges of these faces. Any edge which appears twice in this list cannot belong to Z_S. Thus, the edges of Z_S are determined in $O(\alpha_0 \log \alpha_0)$ time via sorting this list and eliminating both elements of duplicate pairs. To get Z_S with ordered edges from this unordered set, take any point Q inside any of the visible faces and sort the vertices of this set in angular order about Q. As soon as we find Z_S, we know all the faces of Q_S.
In Fig. 18, we show the problem reduction step for a simple scene. After this reduction P' must be submitted to Algorithm A1 summarized in the preceding section.

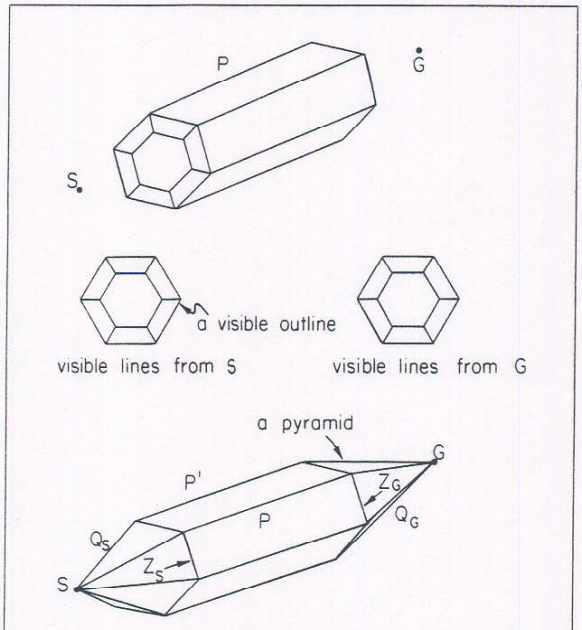


Fig. 18. The two pyramids are computed after a visibility computation from viewpoints S and G. In this case the bases of the pyramids are hexagonal

Solution of FP3

The methods outlined in the last two sections are used in this case. To solve FP3, we follow a Voronoi-based strategy [29]. We want to partition the face F_G into subregions via a planar subdivision made of straight edges such that if a query point G is discovered to be belonging to a particular subregion of this subdivision, it must be possible to list the face visit sequence that should be taken by an S-to-G minimal path. The following algorithm accomplishes this:

Algorithm A3

I. Preprocessing

1. Develop into the plane of F_G all simple visit sequences in Ω between nodes F_S (the face of P holding S) and F_G using Δ. At the end of this process, one obtains, using the Dihedral principle, a set of points in the plane which are the "images" of S under given visit sequences.
2. Store with each image point, the visit sequence used to arrive it.
3. Using standard algorithms construct the Voronoi diagram Γ which has these images as Voronoi centers.
4. "Clip" Γ against F_G. This simply means [18] finding the subset Γ' of Γ included within the polygon F_G.

II. Querying

1. Given a new query point G, first make sure that $G \in F_G$.
2. Using standard point location algorithms locate the subregion R including G in subdivision Γ'.
3. Using the stored sequence in the Voronoi center corresponding to R develop the faces specified by this sequence and find the minimal path with straightforward application of the Dihedral principle.

Once we preprocess the given scene for a given S and F_G, for each goal G we must determine to which region of the subdivision Γ' it belongs. Assume that Γ' has y edges. This can be done using any of the point-location algorithms mentioned in the previous sections directly.

The size y of Γ' depends on two things: the number of simple visit sequences between F_S and F_G in Ω, and the shape and relative location of F_G. If F_G is very large it will probably include most of Γ. If it is almost coincident with the most crowded parts of Γ then Γ' will again have many subregions. Similarly, if there are many visit sequences then the Voronoi diagram Γ will have many edges. In general, the complexity of constructing Γ on the surface of P for many specified source points will depend on the average number of regions that each face is partitioned into.

In Fig. 19, we demonstrate this preprocessing method for a cube. The lists written next to each image of S in this figure specify the order of unfolding applied to arrive that point. Not all images of S are shown. For example, the simple sequences (3, 4, 5, 2, 1, 6) and (3, 4, 5, 1, 2, 6) are missing, but they do not really alter anything. Notice that if G is inside the subregion with vertical hatching applied then the minimal path is via the unfolding of simple sequence (3, 2, 6). On the other hand, it is given via the unfolding of (3, 6) when G is inside the horizontally hatched subregion.

A dynamizing technique which would allow good insertion time for an incoming point in the Voronoi diagram proves very useful in this context. Thus, we start with only a few image points (commonly corresponding to the first few short simple sequences in the face graph) and easily work our way to an incomplete Voronoi diagram. When we want to see the effect of another image point not yet tried we use a dynamic data structure [57] to insert it. In a recent paper by Gowda et al. [27], it is proved that this is achievable in $O(y)$ time if the Voronoi diagram has y points prior to insertion.

Finally, the solution of FP3 offers a new and interesting question regarding Voronoi diagrams. Given y points and a polygon P in the plane, how quickly can one find the subset of the Voronoi diagram on these y points lying inside P? In other words, can one gain anything (i.e., save unnecessary computations which are wasted by the clipping process in Algorithm A3) by promising to look at the diagram only through a window? Although an adversary can always force a worst-case we conjecture that on the average this with take time less than $O(y \log y)$ as long as the convex hull of these points and P are not close to each other under some reasonable distance measure.

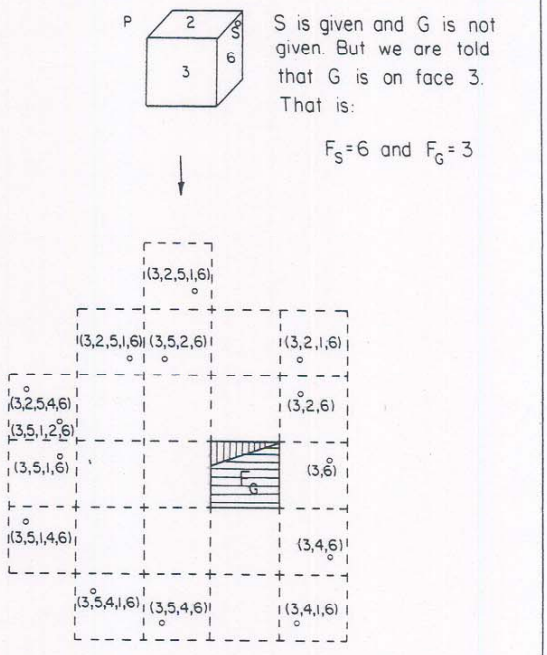


Fig. 19. Voronoi partitioning on face 3 of a cube for the specified location of S. Only the clipped Voronoi diagram is shown

Complexity of the diagrams

The complexity of the diagrams in the plane with barriers, as measured by the number of new points and lines added, is linear in the number of sources

plus barrier edges. In fact, the number of regions is equal to the total number of associated vertices, which is the number of sources plus twice the number of barrier lines. Since every new vertex (at the intersection of new ridge or shadow lines) has three adjacent lines, and since this is a planar graph, albeit with some edges extending to infinity, then the number of vertices, edges, and regions are each linearly related to the other. Thus the graph is of linear complexity in the size of the input.

Although we do not yet have an efficient algorithm for constructing the graph, one appears to be possible along these lines: Assume a wave front starting from each source. As it hits a barrier vertex, label that vertex, start a shadow line, and "diffract" part of the wave front behind the barrier. If two wave fronts meet, then start a ridge line. If two lines, which means three wave fronts, meet, then stop one wave front and start one new ridge line. Continue until all barrier vertices have been reached, and the remaining lines are heading towards infinity at angles such that they will never intersect each other.

The key to the above algorithm's efficiency lies in the choice of data structures since there may be many possible objects that each might be reached next by any of several wave fronts. Avoiding a quadratic time requires a spatial data structure such as Drysdale's generalized Voronoi diagrams, modified to be dynamic. Then as we insert new elements, we will immediately know all the relevant adjacency relations.

The complexity of constructing Voronoi diagrams on the surface of a polyhedron depends on the average number of regions that each face is partitioned into. It is suspected that this small in practice, although an adversary could find some very large cases.

Summary

We have demonstrated two extensions of the Voronoi diagram to handle either opaque barriers in the plane, or the surface of a convex polyhedron. These constructions are useful in the single source – multiple goals problem with barriers in robotics, and in finding minimal paths in E^3 in the presence of impenetrable polyhedral obstructions.

References

1. (Anon) (1982) Statistical mechanics algorithm for Monte Carlo optimization. Physics Today: pp 17–19
2. Abelson H, DiSessa A (1982) Turtle geometry. The Computer as a medium for exploring mathematics. MIT Press, Cambridge, MA
3. Abramowitz M, Stern IA (1964) Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, pp 17–18
4. Akman V (1984) Findminpath algorithms for task-level (model-based) robot programming. Manuscript, ECSE Dep., Rensselaer Polytechnic Inst., Troy, NY
5. Aleksandrov AD (1958) Konvexe Polyeder (German, translated from Russian). Akademie-Verlag, Berlin
6. Bakst WW (1941) Mathematics, its magic and mastery. D. Van Nostrand, New York
7. Bentley JL, Shamos MI (1976) Divide and conquer in multidimensional space. In Proceedings of the 8th ACM Annual Smyposium on Theory of Computing, pp 220–230
8. Brooks RA (1983) Solving the Findpath problem by good representation of free space. IEEE Systems Man Cybernet 13:190–197
9. Brown KQ (1979) Geometric transforms for fast geometric algorithms, Ph.D. Thesis, also Dep. of Computer Science, Tech. Rep. CMU-CS-80-101, Carnegie-Mellon Univ., Pittsburgh, PA
10. Brown KQ (1979) Voronoi diagrams from convex hulls. Information Processing Letters 9:223–228
11. Chazelle BM (1980) Computational geometry and convexity. Ph.D. Thesis, Computer Science Dep., Yale Univ., New Haven, CT, 1980. Also Tech. Rep. CMU-CS-80-150, Computer Science Dep., Carnegie-Mellon Univ., Pittsburgh, PA
12. Christorides N (1975) Graph theory. An algorithmic approach. Academic Press, New York
13. Davis PJ, Chin WG (1966) 3.1416 and all that. Simon and Schuster, New York
14. Donald BR (1983) The Mover's problem in automated structural design. In: Proceedings of Harvard Computer Graphics Conference, Cambridge, MA
15. Drysdale RL (1975) Generalized Voronoi diagrams and geometric searching, Ph.D. Thesis, also Computer Science Dep., Tech. Rep. STAN-CS-79-705, Stanford Univ., Stanford, CA
16. Efimov NV (1962) Qualitative problems of the theory of deformation of surfaces. In: Differential Geometry and Calculus of Variations (translated from Russian), AMS Translations Series 1, 6:274–423
17. El Gindy H, Avis D (1981) A linear algorithm for computing the visibility of a polygon from a point. J Algorithms 2:186–197
18. Foley J, and Van Dam A (1982) Fundamentals of Interactive Computer Graphics. Addison-Wesley, Reading, MA
19. Franklin WR (1980) Efficiently computing the haloed line effect for hidden line elimination. Image Processing Lab, Tech. Rep. IPL-81-004, Rensselaer Polytechnic Inst., Troy, NY
20. Franklin WR (1981) An exact hidden sphere algorithm that operates in linear time. Comp Graph Image Processing 15:364–379
21. Franklin WR (1982) Partitioning the plane to calculate minimal paths to any goal around obstructions. Image

Processing Lab, Tech. Rep., Rensselaer Polytechnic Inst., Troy, NY
22. Franklin WR, Akman V (1984) Minimal paths between source and goal points located on/around a convex polyhedron. In: Proceedings of the 22nd Allerton Conference on Communication, Control, and Computing, Allerton IL
23. Franklin WR, Akman V (1984) Euclidean shortest path in 3-space. Voronoi diagrams with barriers, and related complexity and algebraic issues (Extended abstract). ECSE Dep., Tech. Rep., Rensselaer Polytechnic Inst., Troy, NY
24. Frechet M, Fan K (1967) Initiation to Combinatorial Topology (translated from French). Prindle, Weber, and Schmidt, Complementary Series in Mathematics, Vol. 7, Boston, MA
25. Garey MR, Johnson DS (1976) Computers and intractability. A Guide to the theory of NP-Completeness. WH Freeman, San Francisco, CA
26. Garey MR, Graham RI, Johnson DS (1976) Some NP-complete geometric problems. In: Proceedings of the 8th ACM Annual Symposium on Theory of Computing, pp 10–22
27. Gowda IG, Kirkpatrick DG, Lee DT, Naamad A (1983) Dynamic Voronoi diagrams. IEEE Trans Inf Theory 29:724–731
28. Grunbaum B (1967) Convex Polytopes, Wiley Interscience, New York
29. Guibas L, Stolfi J (1983) Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. In: Proceedings of the 15th ACM Annual Symposium on Theory of Computing, pp 221–234
30. Guibas L, Yao FF (1980) On translating a set of rectangles. In: Proceedings of the 10th ACM Annual Symposium on Theory of Computing, pp 154–160
31. Hopcroft JE, Schwartz JT, Sharir M (1984) On the complexity of motion planning for multiple independent objects: *P*-space hardness of the "Warehouseman's Problem". Computer Science Div., Tech. Rep., Courant Inst. of Mathematical Sciences, New York Univ., New York
32. Katoh N, Ibaraki T, Mine H (1982) An efficient algorithm for *k* shortest simple paths. Networks 12:411–427
33. Kirkpatrick DG (1983) Optimal search in planar subdivisions, SIAM J Computing 12:28–35
34. Lee DT, Drysdale RL (1981) Generalization of Voronoi diagrams in the plane. SIAM J Computing 10:73–87
35. Lee DT, Preparata FP (1977) Location of a point in a planar subdivision and its applications. SIAM J Computing 6:594–606
36. Lee DT, Preparata FP (1984) Euclidean shortest paths in the presence of rectilinear barriers. Networks 14:393–410
37. Lee DT, Yang CC (1979) Location of multiple points in a planar subdivision. Information Processing Letters 9:190–193
38. Levin JZ (1976) A parametric algorithm for drawing pictures of solid objects composed of quadric surfaces. Communications of the ACM 19:555–563
39. Levin JZ (1979) Mathematical models for determining the intersections of quadric surfaces. Comp Graph Image Processing 11:73–87
40. Levin JZ (1980) Implementation of two hidden-line algorithms. Comput Graph 5:31–40
41. Lipton RJ, Tarjan RE (1979) A separator theorem for planar graphs. SIAM J App Mathematics 36:177–189
42. Lipton RJ, Tarjan RE (1980) Applications of a planar separator theorem. SIAM J Computing 9:615–627
43. Lozano-Perez T (1981) Automatic planning of manipulator transfer movements. IEEE Systems Man Cybernet 11:681–698
44. Lozano-Perez T (1983) Spatial planning, a configuration space approach. IEEE Trans Computers 32:108–120
45. Lozano-Perez T, Wesley MA (1979) An algorithm for planning collision-free paths among polyhedral objects. Communications of the ACM 22:560–570
46. Lyusternik LA (1964) Shortest Paths. Variational Problems (translated from Russian). Macmillan Co., New York
47. Nguyen VD (1984) The Findpath problem in the plane. A.I. Memo No. 760, Artificial Intelligence Lab, Massachusetts Inst. of Technology, Cambridge, MA
48. O'Dunlaing C, Yap CK (1983) The Voronoi method of motion planning I: the case of a disc. Computer Science Div.,Tech. Rep., Courant Inst. of Mathematical Sciences, New York Univ., New York
49. O'Rourke J, Suri S, Booth H (1985) Shortest paths on polyhedral surfaces. Proceedings of the 2nd Annual Symposium on Theoretical Aspects of Computer Science. Saarbrücken, W. Germany
50. Papadimitriou CH, Steiglitz K (1982) Combinatorial Optimization. Algorithms and Complexity. Addison-Wesley, Reading, MA
51. Paul RC (1972) Modeling, trajectory calculation, and servoing of a computer controlled arm. Ph.D. Thesis, Dep. of Computer Science. Stanford Univ., Stanford, CA
52. Paul RC (1979) Robots, models, and automation. IEEE Comput 12:19–27
53. Perl Y, Itai A, Avni H (1978) Interpolation search – a log log N search. Communications of the ACM 21:550–553
54. Reif JH (1979) Complexity of the Mover's problem and generalizations (Extended abstract). In: Proceedings of 20th IEEE Annual Symposium on Foundations of Computer Science, pp 421–427
55. Reif JH, Storer JA (1985) Shortest paths in Euclidean space with polyhedral obstacles. Computer Science Dep., Tech. Rep. CS-85-121, Brandeis University, Waltham, MA
56. Saaty T (1978) Optimization in Integers and Related Extremal Problems, McGraw-Hill, New York
57. Saxe JB, Bentley JL (1979) Transforming static data structures to dynamic data structures. In: Proceedings of the 20th IEEE Annual Symposium on Foundations of Computer Science. pp 148–168
58. Schwartz JT, Sharir M (1983) On the "Piano Movers" problem: I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. Communications on Pure and Applied Mathematics XXXVI:345–398
59. Schwartz JT, Sharir M (1983) On the "Piano Movers" problem: II. General techniques for computing topological properties of real algebraic manifolds. Adv Appl Mathematics 4:298–351
60. Schwartz JT, Sharir M (1983) On the "Piano Movers" problem: III. Coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers. Int J Robotics Res 2:46–75
61. Sharir M, Ariel-Sheffi E (1984) On the "Piano Movers" problem: IV. Various decomposable two-dimensional motion-planning problems. Communications on Pure and Applied Mathematics XXXVII:479–493
62. Schwartz JT, Sharir M (1984) On the "Piano Movers" problem: V. The case of a rod moving in three-dimensional space amidst polyhedral obstacles. Communications on Pure and Applied Mathematics XXXVII:815–848

63. Shamos MI (1978) Computational geometry. Ph.D. Thesis, Dep. of Computer Science, Yale Univ., New Haven, CT
64. Shamos MI, Hoey D (1975) Closest-point problems. In Proceedings of 16th IEEE Annual Symposium on Foundations of Computer Science, pp 151–162
65. Sharir M, Schorr A (1984) On shortest paths in polyhedral spaces. In: Proceedings of the 16th ACM Annual Symposium on Theory of Computing, pp 144–153
66. Udupa SM (1977) Collision detection and avoidance in computer controlled manipulators. Ph.D. Thesis. Dep. of Electrical Engineering, California Inst. of Technology, Pasadena, CA
67. Valiant LG (1979) The complexity of enumeration and reliability problems. SIAM J Computing 8:410–421
68. Verrilli C (1984) One source Voronoi diagrams with barriers, a computer implementation. Image Processing Lab, Tech. Rep. IPL-TR-060. Rensselaer Polytechnic Inst., Troy, NY
69. Yen JY (1971) Finding the k shortest loopless paths in a network, Management Science 17:712–716