A SHORTEST PATH METHOD FOR HIERARCHICAL TERRAIN MODELS *

Renato Barrera
NCGIA University of Maine
120 Boardman Hall
Orono, ME 04469
e_mail RENATO@MECAN1.bitnet

Jesús Vázquez-Gómez
Department of Computer Science
UAM-Atzcapotzalco, México

Abstract

An algorithm is presented for obtaining the shortest path between two points on a terrain represented by a triangular-faced polyhedron. The terrain model is hierarchical, i.e. it has several level of precision, the representation at each level refining the previous one.

The proposed algorithm consists of two phases In the initial phase, terrain representations at increasing precision levels are searched for regions where no optimal paths can trespass; these regions are not to be considered any further. In the final phase, a standard shortest path algorithm is applied on the remaining areas

1 Introduction

A renewal of interest in path finding problems has occurred recently, spurred by several circumstances· a greater number of scientists in geometrical and combinatorial problems, the appeareance of inexpensive and powerful computers, and the coming of age of germane applications in robotics, in navigation, in CAD-CAM, etc.

This work deals with efficiently obtaining shortest paths on a triangulated terrain model with v vertices. Were the paths restricted to traversing the terrain through the triangle's edges and vertices, Dijkstra algorithm [AHO 74] would yield the solution in $(v\ log(v))$ time. Unfortunately, this approach does not necessarily render an optimal path. The best available exact solution, due to D Mount [MOUN85], solves this problem in $O(v^2 log(v))$ time.

A computational time proportional to the square of the number of datapoints is not entirely satisfactory. To that effect, efficient approximate methods that circumvent that difficulty have been devised [SMIT87] Although fast approximations are very convenient, it is also suitable to consider the improvement of *exact* path finding methods. This work describes an heuristic that prunes the search region prior to the application an exact optimization algorithm. Our heuristic considers a hierarchical terrain model and a staged elimination of unnecessary regions, each stage corresponding to a level in the hierarchy

Reduction stages use a branch-and-bound procedure. The steps involved are:

Branch All remaining triangles are subdivided; afterwards, a reasonable short source-destination path on those triangles is generated. The length of such path is an upper bound on the shortest distance, and will be called u.

Bound Let e be an edge in the triangulation. Two functions $S_e(p), D_e(p)$ defined for points $p \in e$ are computed; these functions bound from below the distance of p to the source and destination respectively. All regions circumscribed by polygons whose points obey $S_e(p) + D_e(p) > u$ can be discarded, for no shortest route can cross them.

*This work was developed while the authors were at the Electrical Engng. Dept. at CINVESTAV-IPN, México. The second author was supported by a COSNET scholarship

This paper is organized as follows: Section 2 reviews the terrain model used; Section 3 introduces Mount's algorithm while Section 4 will explain our method. Finally, the last section presents an experimental result and conclusions.

2 Terrain Models

The model used here considers a sequence of $n+1$ triangulations $\{T_0\},\cdots,\{T_n\}$, as in [BARR87]. Each triangulation is a *refinement* of its predecessors, i.e. for $0 \le k \le n$, all vertices of $\{T_k\}$ belong to the set of vertices of $\{T_{k+1}\}$ and each triangle of $\{T_{k+1}\}$ is contained in a single triangle of $\{T_k\}$.

Any triangulation $\{T_i\}$ obeys two conditions:

i) Its vertices, for whom altitudes are available, form a square grid of $(2^i+1)\times(2^i+1)$ points.

ii) A triangulation similar to the ones in Fig. 1 is induced on its vertices

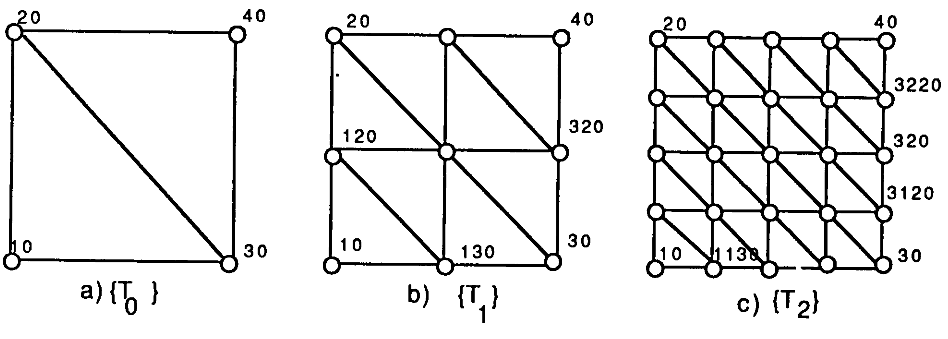


Figure 1: Representation of Triangulations

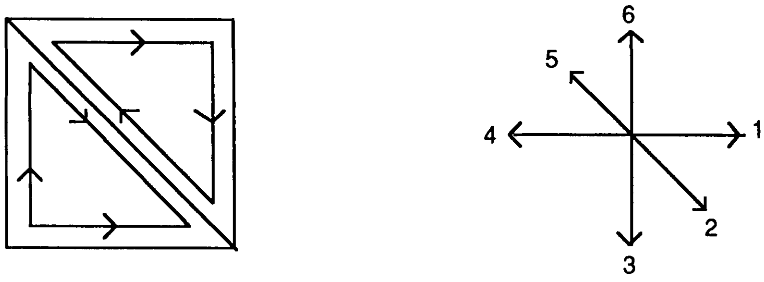
Also, at the finest level $\{T_n\}$, the altitude of any point in the terrain must be accurately described by a linear interpolation of those of the vertices of the triangle containing that point.

2.1 Positional keys for vertices and edges

A *positional key* that consists of a string of symbols of the alphabet $\{1,2,3,4\}$ terminated by a symbol $\{0\}$ is assigned to each point in the original $(2^n+1)\times(2^n+1)$ array. In our model, the positional key of a point is obtained by a Morton encoding (a.k.a bit interleaving) of its coordinates. Fig. 1.a, 1.b, 1.c show three levels of refinement with some of its corresponding keys.

Since there is a one-to-one correspondence between positional keys and coordinates, the terrain model can be stored as an ordered list of pairs of {key,altitude}. The manner in which two keys are compared is also relevant: if the comparison is lexicographical (i.e. $1230 < 140$) the points are ordered following a Peano curve; if the comparison is numerical (i.e. $140 < 1230$) the points are stored by levels of refinement: first $\{T_0\}$, then $\{T_1\}$, etc.

The edges of triangles in a model $\{T_k\}$ will be assigned a positional key as well. In order to exploit the *monotonicity property*, to be introduced in Section 3, triangles will be oriented (Fig 2.a); i.e. an edge will have a separate version or "directed edge" for each triangle it bounds. Since our model is made of isosceles rectangular triangles, directed edges can have only six directions, coded as shown in Fig 2.b. Therefore, both vertices and directed edges will be encoded in a single way prefixing the positional key of the initial node with the direction code



a) Two copies of an edge

b) Six possible directions of a directed edge

Figure 2: Oriented Edges

(a single node is supposed to have direction '0'). This unifying method of encoding is called an *e_code*.

Considering a terrain model as a hierarchy of refinements and employing a uniform coding schema for vertices and edges is advantageous: their usage favors data compression and provides our procedures with efficient search algorithms.

3 Method of Mount

The (unique) shortest path between two points on a plane is the straight line that joins them. This result has counterparts for the case of polyhedra, both convex and non_convex.

In the remainder of the work it will be assumed that the faces of the polyhedron are triangles, and that both the source s and the destination d are vertices of them. Unless otherwise stated, all paths will start at s and end at d.

This section initially presents necessary conditions for shortest paths, first for convex polyhedra and afterwards for the non_convex case. Finally it will sketch an exact shortest path algorithm due to D. Mount [MOUN85].

The concept of a *planar unfolding* is needed to proceed with the presentation. Let $F_1 \cdots F_m$ be the sequence of faces traversed by a path. As said above, s and d are vertices of F_1 and F_m respectively. A series of affine rotations $A_1, \cdots, A_m$ can be applied to the faces so that:

- $A_k(F_k)$ is on the (x, y) plane
- If e_k is the common edge between the faces F_k and F_{k+1}, then $A_k(e_k)$ and $A_{k+1}(e_k)$ coincide.

The sequence of faces $A_1(F_1) \cdots A_m(F_m)$ for a given path is called its planar unfolding, since after the transformations the path will lie flat on the horizontal surface.

The following necessary condition can be stated using unfoldings:

Condition I In a convex polyhedron, the image of an optimal path on its planar unfolding must be a straight line.

Figure 3 shows a pyramid, a shortest path and its unfolding.

Condition I is not sufficient: if the bottom of the piramid in fig 3.a is ignored, the paths between the extreme points might cross two possible sequences of faces: $F_6F_5F_1F_3$ and F_1F_2. The planar unfoldings of both sequences admit a straight line between source and destination (Fig 3.b). The first path is trully optimal. The second one is a local optimum, i.e, best among of all possible paths through that secuence of faces. Cases can be found where several optimal paths exist.

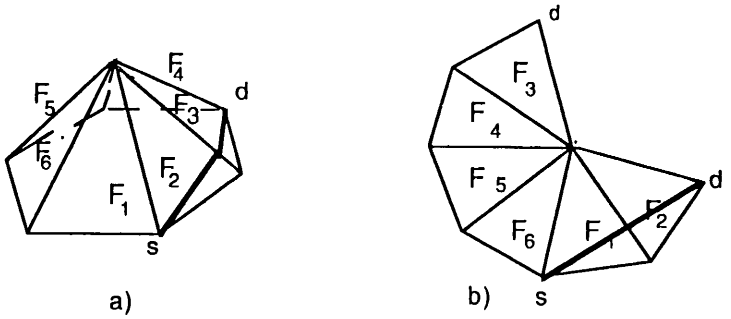


Figure 3. A shortest path on a Convex Polyhedron and its unfolding

In the case of a non-convex polyhedron, the mapping of an optimal path on its planar unfolding will not necessarily render a straight line. Condition I becomes :

Condition II In a non-convex polyhedron, the image of an optimal path on its planar unfolding must be composed of one or more straight line segments; any two consecutive segments must be joined at a vertex of a face and the angle between them must be at least 180 degrees.

Fig 4 illustrates that condition, which again is only necessary for the optimality of a path. Paths obeying it are called geodesics; they can be proved to be locally optimal, i.e. all detours inside the planar unfolding render longer distances.

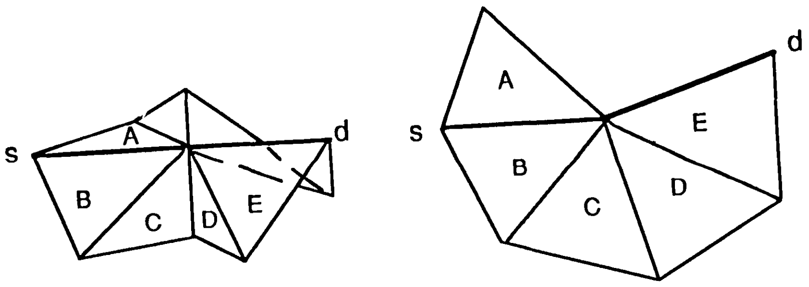


Figure 4: Planar Unfolding on a nonconvex polyhedron

Geodesics are interesting: they are uniquely specified by the sequence of its traversed faces (called "history") and the optimal paths are counted among their numbers.

The generation of histories is straightforward: Fig 5.a shows a polyhedron, and Fig 5.b the connectivity graph of its faces. Fig 5.c displays a tree whose nodes are labelled with the names of faces: Its root with *A* and its leaves with *D*. All possible histories that start with *A* and end with *D* correspond to paths between the root and a leaf of 5.c. Any initial segment of an history will be called a partial history.

The generation of histories is similar to the exploration by gradual expansion of all possible paths in a graph. Thus, the process of generating Fig 5.c from the data of Fig 5.b starts with *A* as the only partial history. Two histories *AB* and *AD* are generated by the extension of A; *AD* has reached the destination and will not be further extended, so *AB* is extended to render two partial histories ABC and ABG, which in turn are extended to $ABCD$, $ABCF$, $ABGF$, etc

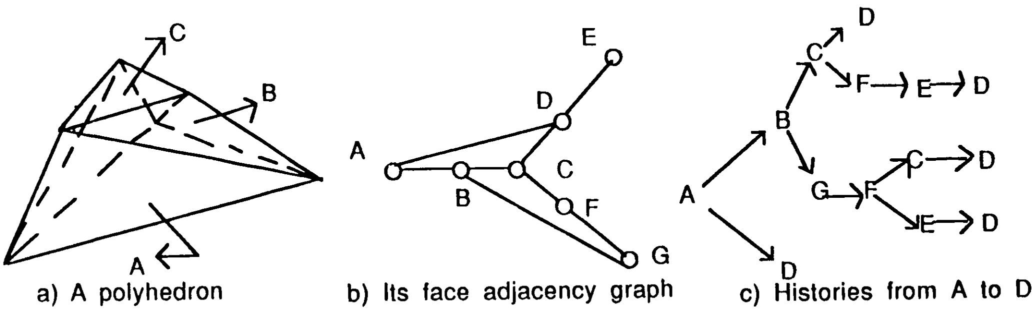


Figure 5: Faces and histories

Two things must be noted in the example of fig 5.c:

- An edge between two faces can be crossed in both directions (e.g. in Fig 5.c the edge common to C and F belongs to $ABCFED$ and $ABGFCD$). Considering an edge as made of two directed copies or d_edges simplifies greately the computations. Thus, the edge common to C and F will have two copies· a d_edge to go from C to F, and another one to go from F to C. As shown in Section 2, directed edges can be easily accessed in our model by means of its e_code.
- A d_edge might be found in more than one history (as CD, found in $ABCD$ and $ABGFCD$)

The characterization of geodesics by means of histories drastically reduces the number of paths considered for optimality, and it suggests a method based on the generation of histories. It presupposes the existence of a heap of histories ordered by the shortest distance of its shortest geodesic, of a floating point number D that measures the shortest geodesic found so far, and a function $distance(history)$ that renders the length of the shortest geodesic inside that history.

The algorithm has the following steps:

i) Push all faces neighboring the source into the heap. $D = \infty$
ii) $history = pop(heap)$
iii) if $distance(history) > D$, terminate the algorithm. The shortest geodesic found so far is an optimal path.
iv) If $history$ has reached the destination, make D the minimum between the old D and $distance(history)$. Else extend $history$ and push all its descendants into the heap.
v) Go to i)

The algorithm in [MOUN85] is based on those ideas. Its implementation employs a construction called "wedge", that resumes a partial history A new wedge is generated every time a history is extended and collides with a new directed edge. Among other information, a wedge contains

- The address of the directed edge.
- The wedge's basis, i.e. a characterization of the set of points on the edge reachable by geodesics inside a planar unfolding.
- A function that renders the distance from all points on the basis to s.
- A pointer to the previous wedge in the history

The cited algorithm considers directed edges. This artifice makes the problem monotonic, i.e. makes all edges traversable in only one way. It can be proved that monotonicity guarantees the connectivity of the basis of a wedge, thus greatly simplifying the computations. The distance from a point p on a wedge's basis to s is given by a circle, that is, by a formula of the type $(p_x - x_0)^2 + (p_y - y_0)^2 + c$, where p_x, p_y are coordinates of p on the planar unfolding . The number of wedges that can coexist on a given directed edge is proportional to the total number vertices. Since the number of directed edges is proportional to that of vertices, $O(v^2)$ wedges might be generated and the algorithm needs $O(v^2)$ memory. Considering that the wedges have to be kept on a heap gives a computational complexity of $O(v^2 log(v))$

The algorithm has been only sketched, and many features related to the extension of a wedge have been omitted; e.g. how a wedge might be split, narrowed or turned around a vertex. The basis of two wedges can also collide on a directed edge; in that case at least one of the basis must be narrowed so as to keep them disjoint.

4 Proposed Method

Existing optimal algorithms for the shortest route problem require a $O(v^2 log(v))$ computational time. This means that the amount of computation grows with the square of the number of vertices, and with the fourth power of the precision.

Those figures tempt the user to decrease as much as possible the number of points under consideration and to perform the reduction at coarse scales.

That seems generally feasible. Even though there are times when an optimal solution occupies all triangles, in most of the cases only a fraction of the terrain is traversed by an optimal route. The greater percentage of the computation time is spent in fruitlessly exploring regions in which no optimal solution can exist.

Thus, a preprocessing method that reduces the area under exploration is indicated. The amount of preprocessing should be further diminished if that method is successively applied on increasing finer terrain models.

The procedure selected is one of the branch and bound type. The algorithm has the following steps:

i) Select the coarsest terrain model as the work triangulation $\{T_w\}$. Initially, all of its triangles are available.
ii) Obtain a *good* path that traverses only those triangles available from $\{T_w\}$. Let u be its length.
iii) For all points p on the boundary of the triangles of $\{T_w\}$ obtain a lower bound on the distance $S(p)$ (or $D(p)$) to s (or d)
iv) Discard all triangles enclosed by a cycle of directed edges obeying $S(p) + D(p) \geq u$
v) If the triangulation is already at its finest level, apply an exact algorithm, otherwise make $\{T_w\}$ a refinement of the available triangles, and go to step i).

An example of the application of those ideas is the following:

Suppose that s and d are located at $(-l, 0, 0), (l, 0, 0)$. Then, for all points p on the terrain, $S(p)$ and $D(p)$ can be given by the distance of the horizontal projection of p to s and d respectively. If a path of length u is available, then no optimal path can go outside the ellipse whose locus is given by $2(x/u)^2 + 2(y)^2/(u^2 - l^2) = 1$

All triangles outside the ellipse can be discarded. Fig. 5 illustrates this concept.

The elliptic formula gives a good initial reduction, but can only work on flat surfaces. In order to proceed to better approximations, a method to obtain $S(p)$ and $D(p)$ was developed that simplifies some features in an exact algorithm.

In an exact algorithm:

- The distance to s from points on the base of a wedge is given by a circle.

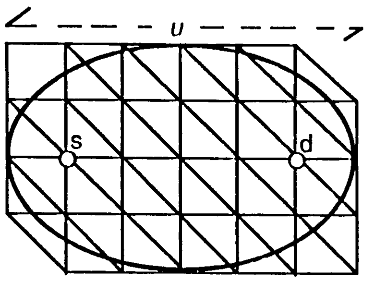


Figure 6: The elliptical approximation

- $O(r)$ wedges with mutually exclusive basis can coexist on an edge.
- The wedge closest to s is expanded, making it possible for $O(r)$ wedges to be expanded on a single edge.

In the algorithm for obtaining lower bounds, a construction *called approximated wedge* (ap_wedge) will be used. For any given directed edge:

- The distance from points in an ap_wedge base to the s or to d is given by a circle.
- Up to a constant number of ap_wedges can coexist on an edge.
- Only one ap_wedge can be expanded per edge. Several ap_wedges can coexist on an edge prior to expansion When expansion is to be performed, a subtending ap_wedge, i.e. one whose distance to the origin is not greater than any of the existing ap_wedges, should be obtained.

Lower bounds $S()$, $D()$ are obtained by the following algorithm, that consider only those non-discarded edges:

i) Obtain ap_wedges for the edges neighboring s (or d). Expand them. Push their expansions into the heap.
ii) If the heap is empty, terminate.
iii) $wedge = pop(heap)$. Let e be the corresponding edge, $\{W_e\}$ be the set of wedges residing on e. Complement $\{W_e\}$ with the expansion of those ap_wedges that have not affected it yet.
iv) Let $wedge'$ be a wedge subtending $\{W_e\}$. Expand $wedge'$, and push into the heap those new ap_wedges that are not incident on an expanded edge.
v) go to ii)

The previous method for evaluating $S(p)$. $D(p)$ takes care of all those regions enclosed into polygons whose points obey $S(p)+D(p) \geq u$ The respective internal edges will not be expanded and, therefore, their interior regions will not be considered.

5 Example

Fig 7 shows the results of our algorithm in a 9×9 terrain model, with three stages of reduction. Overall terrain reduction shown in Fig. 7.a was 40%.

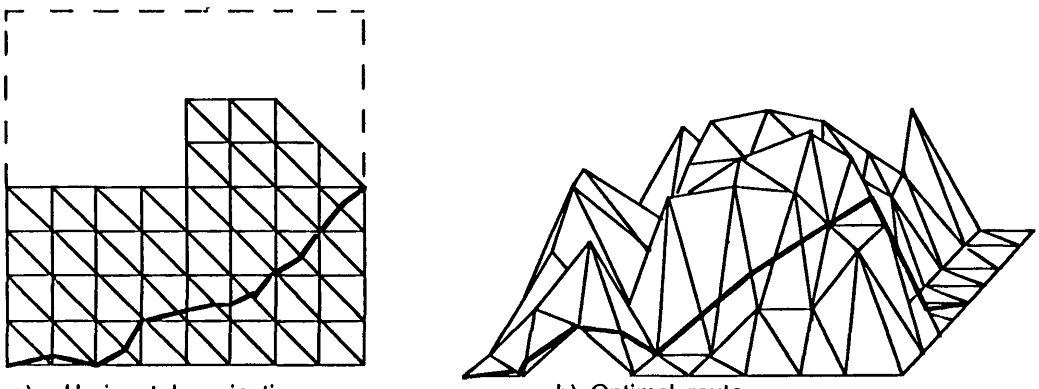


Figure 7: An Example

The computation time was 6 minutes in a PC-XT without floating point accelerator. The exact method lasted 8 minutes.

References

[AHO 74] A.V.
Aho, J.E. Hopcroft, J.D. Ullman The Design and Analysis of Computer Algorithms. Addison-Wesley, 1974.

[BARR87] R. Barrera, A. Hinojosa "Compression methods for terrain relief". Intl. Colloquium in Progress in Terrain Modelling, Copenhagen, May 20-22, 1987.

[MOUN85] D.M. Mount "Voronoi Diagrams on the Sufface of a Polyhedron". Rept. CS-TR-1496, Computer Science Technical Report Series, U. of Maryland, College Park MD., May 1985.

[SMIT87] T.R. Smith, R.E.Parker "An analysis of the efficacy and efficiency of hierarchical procedures for computing trajectories over complex surfaces" European Journal of Operational Rsch. Vol 30, pp 327-338, 1987.