An Efficient Algorithm for Euclidean Shortest Path with Polygonal Obstacles *

S. Kapoor and S.N. Maheshwari
Dept. of Computer Science and Engineering
Indian Institute of Technology, Hauz Khas, New Delhi.

6th January 1993
Revised August 1994.

Abstract

In this paper we present an efficient solution to finding the Euclidean Shortest Path from one point to the other in the presence of polygonal obstacles. An $O(m^2 logn + T)$ algorithm is developed where m is the number of obstacles and n, the number of vertices and T is the time for triangulation. The algorithm works by finding a restricted visibility graph in the obstacle space and is simple. It achieves the lower bound of $\Omega(n + mlogm)$ when $m = O((n/logn)^{1/2})$. We also show an output sensitive time bound of $O(m^2 log(e/m) + m^2 log(n/m) + eloge)$ for the problem where e is bounded above by m^2.

1 Introduction

The Shortest Path problem in a d-dimensional space is that of finding a shortest route from one point to another amongst the presence of obstacles. The problem has received considerable attention during the last decade not only for the obvious applications in motion planning but also for the development of a wealth of tools and techniques in computational geometry to tackle it in it's various manifestations. It should be of interest to note that in 3 dimensions under the usual Euclidean metric it is not even known whether the shortest path problem in the presence of polyhedral obstacles is in NP though the problem has been shown to be NP-Hard. These results assume finite precision arithmetic. Readers are referred to [P] and [Mi1] for a discussion on various facets of the problem and a comprehensive bibliography. Work on special cases of the problem may also be found in [GH], [GHLST], [LP], [Mi2].

We consider in this paper the following 2 dimensional Euclidean version of the problem: Find a shortest path between two specified vertices s and t in a simple polygon with at most m polygonal obstacles. There are a total of n vertices in the polygonal structures involved. We assume a model of computation where real arithmetic is allowed though the results apply even when finite precision arithmetic is used (ignoring the numerical complexity of the schemes).

The algorithms that have been proposed for this problem build the visibility graph of the point set and line segments and then apply a standard shortest path algorithm like Dijkstra's. This leads to an $O(E + nlogn)$ bound where E is the number of edges in the visibility graph and n is the number of vertices in the graph [FT]. This includes the work required for finding the visibility edges which has been shown to be of $O(E)$ complexity [AAGHI, GM,KM, EW].

*A preliminary version of this paper appeared in part in the Proc. of the 4th ACM Symposium on Computational Geometry, 1988, pp. 172-182.

However E can be $O(n^2)$. An alternative approach by Reif and Storer requires $O(mn + nlogn)$ [RS].

This paper shows that a subset of the visibility edges with cardinality of $O(m^2)$ suffices to find the shortest path. This subset gives rise to a restricted visibility graph where the vertices are points in the polygon and edges are those present in the subset of visibility edges. This subset of edges is shown to be constructable in $O(m^2 logn + T)$ operations, where $T = O(n + mlog^{1+\epsilon}m)$ is the number of operations required to triangulate the simple polygon with the m polygonal obstacles. The construction uses a sweep through a partition of the space into regions called corridors and provides a simple alternative to the circular sweep algorithm of Rohnert[R] which has been used for solving the problem when the polygonal obstacles are convex and which can be used for this problem also as described by [Mi4]. The space requirements of the algorithm is $O(m^2 + n))$ which is optimal if the entire restricted visibility graph is to be stored. The algorithm proceeds by first subdividing the polygonal region into the regions called corridors connected together by junctions. There are $O(m)$ such corridors and junctions. This subdivision is obtained from a triangulation of the polygonal region. The visibility edges to be obtained are the unobstructed tangents common to convex chains representing the corridors. These visibility edges suffice to give the shortest path. Note that when $m = O((n/logn)^{1/2})$ the algorithm is optimal. The notion of corridors was first used in [KM]. Subsequently this partition of the space has been used to obtain other results [MS].

In the next section we show how to form corridor structures. In section 3 we discuss the construction of the restricted visibility graph using the corridor structure. The construction uses a sweep of the corridor space. This technique is new and differs from Rohnhert's scheme which uses circular sweep. No sorting is used and thus the scheme appears amenable for improvements which we show in the last section. The final time complexity for construction of the restricted vsisibility graph is $O(m^2 loge/m + (e + m^2)logn/m)$ where e is the number of edges in the restricted visibility graph. The sweep scheme has also been used for construction of the relative neighborhood graph in simple polygons [SK].

The restricted visibility graph is then used for the shortest path computations.

2 Corridor Structure

In this section we will investigate the properties of the triangulation of the polygonal region and extract from it structures which we call corridors. These corridors help define useful visibility edges for shortest path computations.

Consider a triangulation of a simple polygon with obstacles. Each edge of the triangulation either connects two obstacle vertices or connects an obstacle vertex to a vertex on the outside polygonal region. Assume that the obstacles are labelled with numbers ranging from 1 to m. Label the outside region with label 0. Label each edge of the triangulation by a tuple (p, q) where p and q are the labels of the obstacles that the edge connects.

We next consider the dual graph G_d obtained by assigning vertices to the triangles(regions) and edges between vertices if the corresponding triangles share an edge. Let e be an edge in the dual graph. The edge that separates the regions connected by e is defined to be, e_d, the edge dual to e. An edge e is *useful* if there exists a simple path from s to t that crosses e_d. (Fig. Partition)

Easily identified edges that are useless actually correspond to dead ends. These edges may

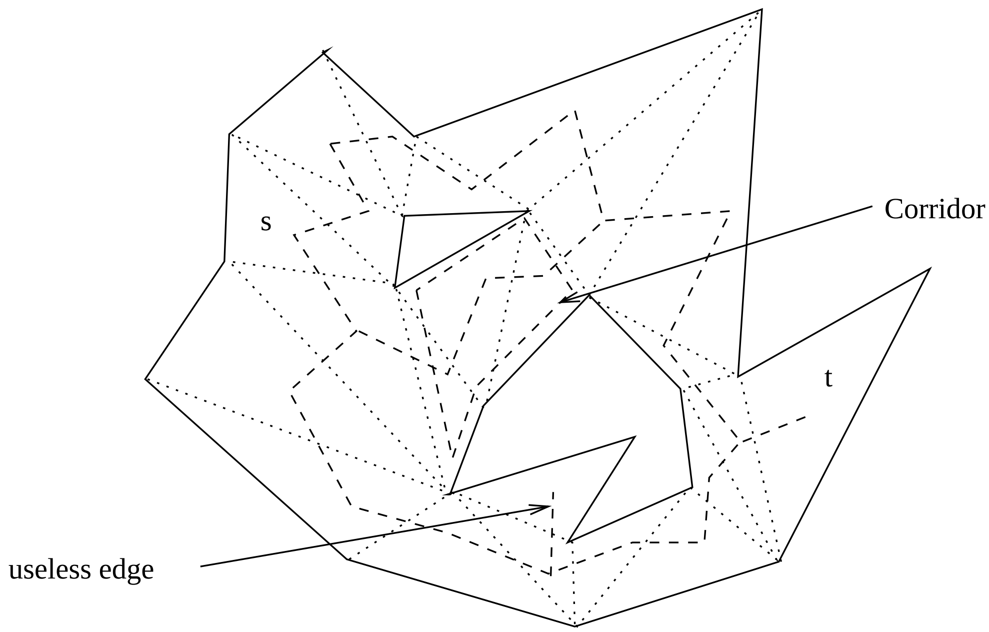


Figure 1: Partition

be identified as follows: An edge $e = (u, v)$ is useless if either u or v is a vertex of degree 1 and the corresponding triangle does not contain s or t. The dual graph is pruned by removing from it iteratively all such edges. This can be done in linear time by starting from the vertices of degree 1. The remaining edges are termed *useful* edges.

We next divide the dual graph into paths where a path is composed of maximally connected vertices of degree 2. Each such path defines a *Corridor* formed by the sequence of dual regions or triangles corresponding to the vertices of the dual graph. Each corridor has two *sides* comprising edges which are either triangulation edges or edges of obstacles. There are also two bounding edges. These edges are triangle edges. Note that a corridor may be degenerate. In this case the corridor consists of a single edge alone which serves as the two boundaries and the two vertices as the sides of the corridor.

We next bound the total number of corridors that are *useful*, where a corridor is defined to be *useful* if the triangles constituting the corridor contribute a *useful* edge in the dual graph, G_d . This will help bound the work that the algorithm does while computing the shortest path.

Theorem 1: The number of *useful* corridors in $O(m)$.

Proof: Ommitted. A proof that elucidates the structure of corridors may be found in [KM]. A simple graph theoretic proof may be found in [MS]. ■

To construct the corridor structures after triangulation, we first construct the dual graph, G_d, eliminate useless edges and define corridors in linear time.

3 The visibility graph and its construction

In this section we describe the visibility graph required for determining the shortest path and show how to construct it efficiently.

The visibility graph that is required is obtained by certain tangent constructions between corridors. To describe these tangents in more detail let us first consider the structure of the corridors.

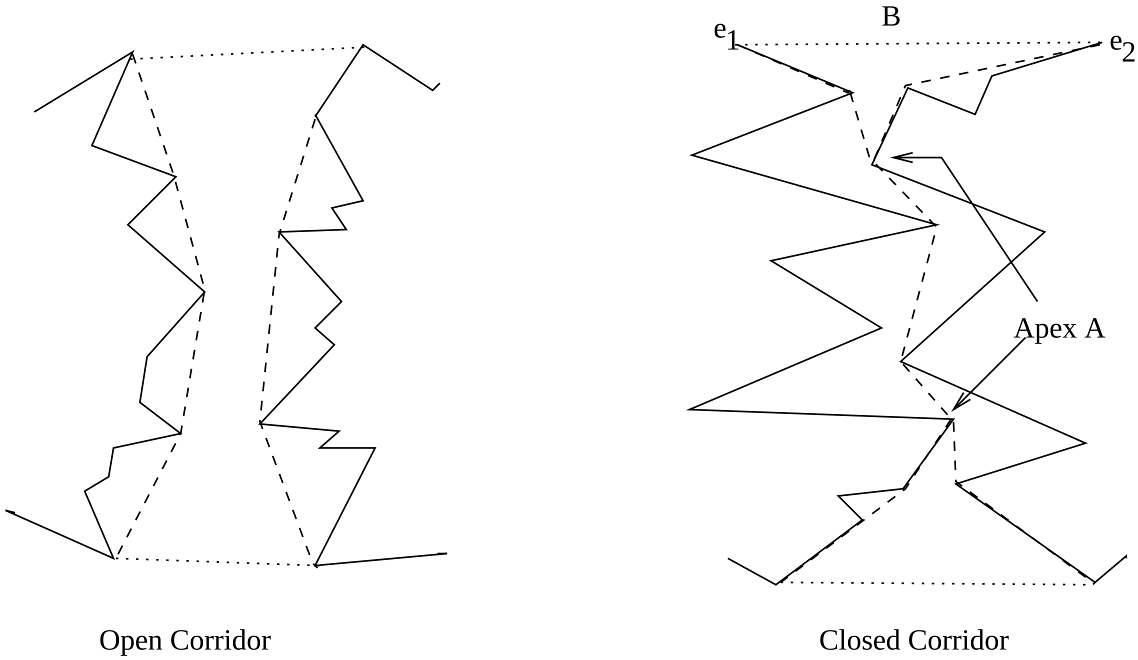


Figure 2: OC

The corridors can be classified by their structure into two types, open and closed corridors (Fig. OC). Closed corridors are those where there is no visibility from one bounding edge to the other, i.e. there does not exist a straight line that can be drawn from one bounding edge to the other without intersecting the sides of the corridor. Open corridors do allow for visibility from one boundary edge to the other. This difference reflects itself in the fact that the shortest path must use the corridor vertices if it passes through a closed corridor whereas it can pass through a open corridor without using the vertices. The classification of the corridors is aided by the convex hull of the vertices on the two sides of the corridor. Let C_1 and C_2 be the convex hull of the vertices on the two sides of the corridor respectively. We can state the following lemma which helps in classifying corridors:

Lemma 2: A corridor is closed iff C_1 intersects C_2.

Proof: First note that an edge which goes from one boundary to the other without touching any vertex in the corridor does not intersect either C_1 or C_2. Thus if C_1 intersects C_2 no such edge exists. Now suppose C_1 does not intersect C_2. Then there is a straight line which divides the two convex hulls (the common tangent suffices) and this straight line passes through the bounding edges and hence the corridor is an open corridor. ■

A closed corridor is further characterized as comprising two funnels and a path connecting these two funnels. A *funnel* is defined by two convex chains, each starting from one of the two endpoints, e_1 and e_2 of a bounding edge, B, and ending at a common endpoint, the apex A (see Fig. OC). The *apex* is the first point in the corridor common to the shortest paths from the endpoints, e_1 and e_2, to the other bounding edge. These shortest paths must intersect otherwise the corridor would be open. And all other shortest paths entering the corridor must also pass through the same point else their length can be reduced. Next consider the shortest path from s to t. It passes through the corridors and consists of straight line segments of two kinds. The first kind are line segments of the convex chains comprising the different corridors or funnels and of the shortest path connecting the apexes of the funnel while the other kind are tangents between convex chains comprising the open corridor or funnels. To construct a restricted visibility graph in which the shortest path may be found it thus suffices to construct the tangents amongst all the convex chains. We note that these convex chains can be obtained from the corridor structures in linear time.

An algorithm which uses the convex chains in a circular sweep may now be designed. In fact Mitchell has suggested that the region obtained after construction of chains may be divided into $O(m)$ convex polygons and then Rohnhert's[R] algorithm may be applied [Mi4].

3.1 Algorithm 1

Let $Q \subseteq P$ be the union of junction triangles and the open corridors of P. The boundary of Q comprises $O(m)$ convex vertices and $O(m)$ reflex chains. We show next how to compute the *restricted visibility graph* of Q, $VG(Q)$. This comprises vertices from Q and edges, connecting vertices, where the edges are either tangents, locally, to the boundary of Q at its endpoints, or are boundary edges of Q.

Consider the complementary region, $R = \mathcal{R}^2 \backslash Q$. This region comprises $O(m)$ reflex verticesand $O(m)$ convex chains. Thus R can be partitioned into $O(m)$ convex polygons using a sweep line algorithm [HM]. The convex polygons have pair-wise disjoint interiors.

Now we can apply the methods of Rohnert ([R]). For each convex polygon, CP, we need to construct tangents to other convex polygons. There are $O(m)$ tangent segments incident onto CP with each convex polygon contributing atmost 4 tangents. Each tangent segment can be computed in $O(logn)$ steps. The tangents that are not obstructed by any obstacle can be identified by a sweep in $O(mlogm)$ steps. Note that the tangent endpoints partition the boundary of the convex polygon into $O(m)$ pieces whose length can be obtained in $O(n)$ times. These pieces can be considered as single edges for purpose of computing the shortest path.

Finally, shortest paths through closed corridors can be computed in $O(n)$ time. These paths are added to $VG(Q)$ along with boundary edges of Q. The shortest path in $VG(Q)$ may now be found using the algorithm in [FT].

Note that the size of $VG(Q)$ can be reduced to $O(m^2)$ by replacng paths through closed corridors and paths along convex chains of Q by a single edge. Also the space complexity may be reduced to $O(n)$ by computing the tangents segments of $VG(Q)$ as they are required as in [R].

However below we derive a more natural scheme which is simple and scans the simple polygon to construct visibility edges and uses corrdors to guide the search for the visibility edges. The scheme is a linear sweep scheme and is based on the scheme in [KM]. It contrasts with the circular sweep scheme. Moreover the scheme avoids retriangulation to construct convex polygons and also does not require sorting which is essential in the above version of Rohnert's scheme[R] and thus appears amenable to improvement.

3.2 Algorithm 2

The algorithm, *Algorithm*2, which constructs the visibility information has as its input the corridor structure. Tangents are to be constructed between convex chains obtained from open corridors and funnels. We consider the convex chains in the corridor structure individually. For each convex chain in the construction, tangents are constructed to the other chains by constructing visibility information. The visibility information for a chain, C, is obtained by sweeping outwards from the convex chain and is maintained as sectors of visible regions, a sector being called a *visibility range.* Each visibility range is essentially defined by two tangents constructed between the chain, C, and two convex chains each obtained from a corridor. The visibility range represents the region from which a visible tangent may be drawn to the chain C. This boundary of this region is defined by edges of chain C and the two tangents each constructed between the chain, C, and another convex chain. As we sweep outwards from the initial convex chain C, we traverse corridors and junctions. The visibility ranges change, with a range being either restricted in scope because one of the tangents defining it strikes an edge inside a corridor or splits into two ranges if an obstacle lies within the range. A range may also be completely removed when it is completely obstructed by the sides of a corridor. To help in processing, the ranges for a chain C, denoted by $R(C)$ and defined by a pair of tangents are maintained in sorted order in a balanced tree which allows for deletion, insertion and merges of ranges along with updates and searches. Since the chain C is convex the tangents incident onto it can be sorted in order of the vertices of the chain they are incident to. Each range, defined by the pair of tangents, is stored at the leaves of the tree.

Furthermore with each corridor is associated a set of ranges. These ranges define the region within the corridor from which a visible tangent can be drawn to C. These ranges form a subsequence at the leaves of the tree storing $R(C)$.

We now describe in detail the visibility range constructions for a convex chain, C. The initial ranges in $R(C)$, are defined using the two bounding edges U and V of the corridor which contains C. The two initial sectors are defined by the tangents from the endpoints of the bounding edges to C. Let us consider the bounding edge U. Since one endpoint of U is the end of C it suffices to construct the tangent from the other endpoint. This tangent may be obstructed by the other convex chain in the corridor, C_1, and in this case the sector is defined using the tangent from C to C_1 (Fig. CC). We thus get two initial ranges. These two starting ranges associated with the two boundaries are contiguous items in the balanced binary tree storing the set of ranges $R(C)$.

At every step of the algorithm the ranges present at a boundary of a corridor are propagated either through the corridor to its other boundary, or from one corridor boundary to a boundary of another corridor (Propagate through a junction).

In the first case the process of propagating ranges through a corridor from one end to the other also results in the construction of the tangents from the corridor to the chain C, and in the second case ranges are propagated across a junction which may lead to either a split or merger of the set of ranges. The first case occurs when the boundary edge is the edge of an untraversed corridor and the second happens when the boundary edge is the edge of corridor through which ranges have been propagated.

Propagation through a corridor. The propagation through a corridor, C is done by a call to the procedure $THROUGH(C, B, E)$. This procedure assumes that a set of ranges is known at one end called the beginning boundary, B, of a corridor. This set needs to be propagated to the other boundary E. There are two cases:

Case 1: The corridor is an open corridor. Let C_1 and C_2 be the two convex chains comprising the corridor. Construct the tangents from C_1 and C_2 to C, say T_1 and T_2 using divide and conquer, stopping the recursion when the endpoint of a chain is reached. These tangents are either common tangents between C and C_1 or C_2 or are subtended at an endpoint of C.

We define a tangent to be *not useful* if it intersects with the other convex chain or does not intersect the boundary edge, B. It is easy to see that such a tangent cannot be part of the visibility graph.

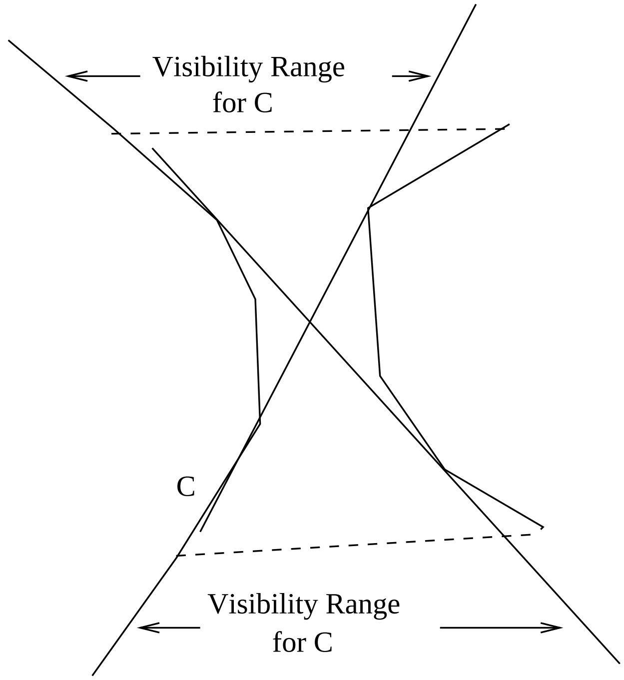


Figure 3: CC

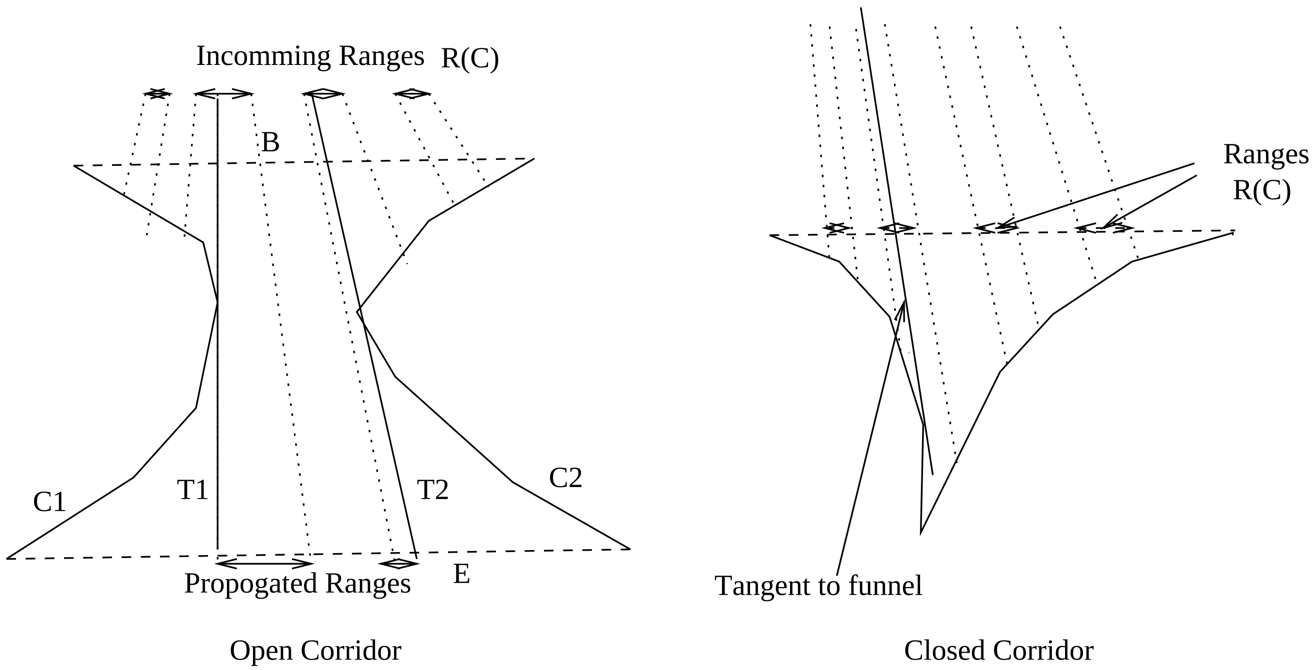
If a tangent is *useful* then locate this tangent in the set of ranges at B by finding the visibility range within which it lies or the ranges between which the tangent lies. This is done using the binary tree storing the ranges $R(C)$. Suppose the tangent lies within a visibility range. Then this tangent is included in the visibility graph since it is unobstructed. Also the range within which it lies is modified since a portion of the range is obstructed. (Fig. PCC) The tangent itself defines one end of the new range. Alternatively the tangent does not lie within a range and is not included in the visibility graph.

We now propagate the ranges through the corridor. If both the tangents constructed are *useful* then we have two tangents each of which lies either outside or on the boundary of a range, the ranges within which the tangent lies having been modified. It should be noted that all ranges lying in between the tangents are unobstructed by the sides of the corridor and can be propagated through to the other boundary of the corridor. The ranges outside are removed since they are obstructed by the corridor. This gives a set of ranges at the other end of the corridor. Note that the ranges removed are also removed from the binary tree storing the set of ranges, $R(C)$.

If, on the other hand, at least one of the tangents is not useful, then none of the ranges present at the current boundary B are propagated through since each of the ranges intersects with one of the convex chains associated with the corridor. The set of ranges associated with the other boundary edge is a null set. The ranges on boundary B are removed from the tree storing $R(C)$.

Case 2: The corridor is a closed corridor. In this case the ranges need not be propagated but tangents from C to the two chains defining the funnel are determined and added to the visibility graph if they lie within a visibility range (Fig. PCC). Again the ranges are removed from $R(C)$.

Propagation across a junction. This propagation involves either a split of a set of ranges or a merger of two sets of ranges.



Propogation Through Corridor

Figure 4: PCC

A split occurs when the ranges at one edge of the triangle, on propagation across the junction, intersect with both other boundaries. The split condition is detected as follows: Let (a_1, a_2, a_3) be the triangle and let the edge $E = (a_1, a_2)$ contain the set of ranges initially. Let T' be the tangent from a_3 to the convex chain C. If this tangent lies within the set of ranges then a split takes place else a merge is required (Fig. SM). The search procedure returns either a range within which this tangent lies or a pair of ranges between which the tangent lies or it reports that the tangent lies outside the set of ranges. If the tangent lies within the set then this set of ranges is split into two sets. There are two cases. In the first case the tangent splits a range R into two ranges R_1 and R_2. It also divides the rest of the ranges into two sets since the set of ranges is linearly ordered. The range R_1 is added to one set and R_2 to the other to complete the split. In the second case, i.e. when the tangent does not lie within any range, the tangent splits the set of ranges into two sets.

A merge occurs when the ranges, R_1 and R_2 associated with two edges of a junction are to be combined into one, R, to compute the set of ranges for the third edge of the junction. Note that this does require that all the ranges be available at the two edges. Whenever two such sets of ranges are available the merger takes place else it is delayed until the ranges are available for merger at both edges. That no circular waiting arises will be proved in the section on correctness. If a set of ranges is not available for extension at the second triangle edge, the merger being considered when one of the triangle edges has a set of ranges available, then an attempt is made to obtain this set by extending the other existing ranges in $R(C)$. The extension needs to be carried out only if the range which is adjacent to the set of ranges at the first boundary intersects the second boundary. In such a situation the set of ranges associated with the adjacent range is extended to reach the second boundary edge. This involves repeated calls to a procedure $EXTEND$ which propagates the set of ranges across corridors or junctions until either the ranges are blocked or do not intersect the boundary or until the ranges reach the boundary. It should be noted that the set of ranges at the second boundary edge so obtained may be the null set. This happens when no point on the boundary edge is visible from C. (Fig. SM)

The algorithms for merging and extending are given more completely below.

We need the following definitions:

$S(A)$ is the set of ranges containing range A. This set corresponds to the set associated with a boundary edge.

$LAST(R)$ is the last range in the set of ranges R.

$SPLIT(C, E_1, E_2, E_3)$ is a procedure which splits the ranges in $R(C)$ at edge E_1 into two sets of ranges at edges E_2 and E_3 as discussed above.

$THROUGH(C, E_1, E_2)$ is a procedure which propagates ranges in $R(C)$ associated with edge E_1 of a corridor to the other boundary edge E_2 as discussed above.

This procedure merges the ranges at edges E_1 and E_2 onto a range at edge E_3

Procedure $MERGE(C, E_1, E_2, E_3)$
begin
  Let R_1 and R_2 denote the set of ranges
   at E_1 and E_2 respectively;
  If either R_1 and R_2 are not present
    then suppose R_2 is not present
    begin
    Let A be the range in $R(C)$ adjacent to $LAST(R_1)$;
    While $(A \cap E_3 \neq \Phi)$ **do**
     begin
      $EXTEND(C, S(A))$;
      If R_2 is not present **then**
       Let A be the range in $R(C)$ adjacent to $LAST(R_1)$;
     end;
    end;
  Merge the two sets of ranges R_1 and R_2 into a new range, R at E_3.
end.

This procedure extends the set of ranges, R

Procedure $EXTEND(C, R)$
begin
  Let R be associated with edge E_1;
  If E_1 is the beginning boundary of a corridor
   with E_2 as the other boundary
  then
    $THROUGH(C, E_1, E_2)$
  else
  If the first range in $R \cap E \neq \phi$ **then**
   begin
    Let (E_1, E_2, E_3) be the junction the range is currently at;
    If range is at a split junction

then

$SPLIT(C, E_1, E_2, E_3)$;

else

begin

If $R \cap E_2 = \phi$ **then** $MERGE(C, E_1, E_2, E_3)$

else $MERGE(C, E_1, E_3, E_2)$

end;

end;

end.

The procedure $EXTEND$ propagates the ranges associated with an edge. In one call of $EXTEND$ propagation takes place to the next possible edge which could either be the other boundary edge of a corridor or the other edge of a junction.

In the procedure $MERGE$ the calls to $EXTEND$ are repeatedly made until either the set of ranges is available at the edge E or no range can be extended to E.

As mentioned earlier the propagation of ranges in $R(C)$ starts at the two bounding edges associated with chain C. At every step the ranges are propagated either through a corridor or a junction by a call to $SPLIT$, $MERGE$ or $THROUGH$. The propagation process terminates when there are no ranges left in $R(C)$. Note that $R(C)$ at any stage contains only those ranges which can be extended further. Ranges are removed from $R(C)$ when they are obstructed. All this is done by a procedure similar to $EXTEND$.

This procedure is invoked for all the convex chains in the corridor structure to construct the entire visibility graph. If s and t are not part of an obstacle or the simple polygon then a degenerate obstacle is constructed comprising of the single point s or t as appropriate at the site of the point. Thus the requisite visibility edges from s and t are easily found.

3.3 Data Structure Details

We need simple data structures in the algorithm.

The primary data structure is a balanced binary search tree for storing the ranges at the triangulation edges. Note that for each convex chain the visibility ranges are to be maintained and propagated as described in the above algorithm. Each disjoint section of the visibility range is stored at a node of a balanced binary tree. The operation on ranges include splitting of the range into two parts during a split operation or removing a section of the current range during an extend operation. This can easily be done by operations on the nodes of the binary tree. Furthermore note that the number of ranges is $O(m)$ since there are atmost $O(m)$ corridors or junctions and each one adds atmost one new range by a split operation.

To achieve the operation above we need to maintain a set of ranges at the triangulation edges. These ranges are maintained in linked lists with doubly linked pointers to the ranges at the leaves of the binary tree. In this way the edge which the range is currently at can thus be easily determined.

3.4 Correctness.

The correctness of the algorithm follows from the fact that all relevant visibility lines are constructed.

Lemma 3: All tangents between a corridor and other corridors are correctly constructed.

Proof: We first show that the visibility ranges corresponding to a corridor C are propagated properly across corridors and junctions. The split at a junction and extension along a corridors is correct by construction as described above. The correctness of the merge procedure is by induction on the number of ranges as follows:

Suppose a merge of two sets of ranges is to be done. There are two cases. When both sets of ranges are completely present the merge is simple. When only one set is present the ranges on the adjacent edge, E_2, must be obtained. This is done by an iterative process which extends all relevant ranges upto the edge E_2. The iterative procedure starts with a set of ranges which includes the range, A, adjacent to the last range on E_1 and which has a non-zero intersection with either E_1 or E_2 or equivalently E_3. In the while loop calls are made to $EXTEND$ repeatedly. At each call to $EXTEND$ the set $S(A)$ is propagated through either a corridor or a junction. This may involve a recursive call to the $MERGE$ procedure. This is done correctly by induction because the number of ranges to be considered is less at the recursion. This is because the ranges to be considered do not circle around as they cannot intersect E_1 and E_2 and thus do not include a range in R_1. At the end of the call either a set of ranges is available at E_2 in which case the merge can be carried out. Otherwise $EXTEND$ needs to be called again to obtain a set of ranges for merger. However during the last call A may have been obstructed together with other ranges in $S(A)$. Thus A and $S(A)$ needs to be recomputed. A is the range in $R(C)$ which is now adjacent to the last range which has reached the junction, i.e. $LAST(R_1)$. Note that all the intermediate ranges which are obstructed have been removed from $R(C)$. The iterative call is invoked with the new set after checking that A has a nonzero intersection with edge E_2. Alternatively a proof may be obtained that no range can be extended to E_2 when $A \cap E_3 = \Phi$. The iterations end since the ranges can only be propagated through the corridors and junctions once only. At the end of the iterations either a set of ranges is available at E_2 or we have a proof that no range can be extended. The merge procedure then combines the ranges at E_1 and E_2. This proves the correctness of the $MERGE$ procedure by induction. The basis is easily established.

Finally note that the entire visibility range for C is correctly computed, i.e. for all corridors it is determined which portions of the corridor have a visible tangent to C as the main program extends ranges until they are blocked.

The tangents associated with the corridor C are computed when the ranges are extended across the corridors and all the tangents are correctly computed since the ranges are correctly computed as proved above.

And lastly since all corridors are considered as starting points all the $O(m^2)$ tangents are correctly computed. ■

This leads to the following result

Theorem 2: The Shortest Path Visibility Graph is correctly constructed.

Proof: This follows from the fact that the tangents required for the shortest path are correctly computed from all the corridors.■

3.5 Timing Details.

Lemma 4: The number of operations required to construct the visibility information for a corridor is $O(mlogn)$.

Proof. First consider the work done in the $SPLIT$, $MERGE$, $THROUGH$ and $EXTEND$ procedures. In the procedure $EXTEND$ either a junction or a corridor is processed through further calls to $SPLIT$, $THROUGH$ or $MERGE$ or the algorithm returns having done a constant amount of work. In the $MERGE$ procedure the algorithm processes a junction but in doing so calls the $EXTEND$ procedure a number of times during the while loop. In each call to the $EXTEND$ procedure from the while loop in the $MERGE$ procedure atleast one corridor is processed. The $SPLIT$ and $THROUGH$ procedures process a corridor.

The processing of a corridor or junction involves finding tangents to the chain C and searching for these tangents in the balanced binary tree storing the ranges $R(C)$. The entire processing requires $O(logn)$ work.

Moreover each corridor or junction is processed only once. The other work done in the procedures besides the processing of the corridors or the junctions requires $O(1)$ work. Also note that the procedures which propagate are called whenever there is a range in $R(C)$. This range is present only if it is extendible since ranges which reach dead ends are removed. Thus each call in the main algorithm leads to the processing of a corridor or a junction. Since there are a total of $O(m)$ corridors and junctions to be processed the time bound of $O(mlogn)$ follows. ■

Theorem 3: Determining the shortest path requires $O(n + T + m^2 logn)$ operations.

Proof: This is done by applying the shortest path algorithm to the graph obtained after constructing the visibility edges from each of the $O(m)$ chains in the corridors. ■

4 Improvements

We next show how to improve on the time complexity of constructing tangents so that the time complexity is dependant on $e, e \leq m^2$, the number of edges in the restricted visibility graph.

The first modification is in the construction of common tangents when propagating ranges through a corridor. A tangent is constructed only if it falls within a visibility range. To find this out each range in the set of ranges present at the boundary edge of the corridor is checked for propagation in sorted order. Let C be the chain to which the common tangent is to be constructed. Assume that C has n_c vertices. And let $R(C)$ be the set of ranges of chain C, at boundary $B = (u, v)$ of a corridor comprising chains C_1 and C_2. We consider construction of tangents to chain C_1. Let u be the first vertex of C_1. Consider successively the ranges present at boundary B starting with the one closest to u. Either a range propagates through or is blocked by the chain C_1 of the corridor. This can be determined by binary search in time $O(logn_i)$ when there are $O(n_i)$ edges in the chain C_1. If the range is blocked then the time for determining this is charged to the range. Otherwise the time for propagating is charged to the chain. A common tangent between C and the chain,C_1, is constructed if the chain of the corridor falls inside a visibility range. This happens when one of the tangents defining the range is obstructed by C_1 and the other propagates to the other boundary. Thus each tangent is charged $O(logn_i + logn_c)$ steps when the visibility range it defines is blocked or a chain is charged $O(logn_i)$ steps when there are $O(n_i)$ vertices in the chain. Thus the total time for constructing tangents common to a chain C is $O(e_c logn/e_c + e_c logn_c + mlogn/m)$ using the fact that $\sum_{1 \leq i \leq m} logn_i = mlogn/m$ when $\sum_{1 \leq i \leq m} n_i = n$ where e_c is the number of tangents constructed using chain C.

Furthermore merges and splits of ranges require $O(loge_c)$ operations. Thus the total time to construct all tangents is $O(m^2 loge/m + elogn/e + (e + m^2)logn/m)$ which is an improvement when $e = o(m^2)$.

We now construct a restricted graph RG to compute the shortest path. The vertices in the graph are constructed as follows: A vertex is added for every point in the polygonal scene if a tangent is incident onto it. The tangents form edges in the restricted graph with weight equal to the euclidean length of the tangents. Moreover, consider a chain of points of degree two in the corridors of the original polygonal scene. An edge replaces this chain with edge weight being the sum of the lengths of the edges on the replaced chain. Thus the number of edges and vertices in the graph are $O(e)$. This graph suffices to compute the shortest path.

Moreover the space requirements for constructing tangents can be reduced to $O(n)$ by only constructing tangents when required in the shortest path tree. This technique is used in [R][Mi4].

We can thus state the following result:

Theorem 4: The shortest path in a polygon with obstacles with a total of n vertices and m obstacles can be computed in $O(n + eloge + m^2log(e/m) + m^2log(n/m))$ operations and $O(n)$ space.

5 Conclusions

The method for finding shortest paths amongst polygonal obstacles that we have described is simple and efficient. After the results obtained in this paper, a number of $o(n^2)$ results have been shown for the shortest path problem [KM], [K], [Mi3] but these results though efficient, (in fact either close to optimal or optimal) are fairly complicated. A more recently published result claims a bound of $O(nlog^2n)$ [HS].

References

[1] [AAGHI] "Visibility of Disjoint Polygons", T. Asano, T.Asano, L. Guibas, J. Hershberger and H. Imai, Algorithmica 1 (1986), 49-63.

[2] [BC] "Triangulating a Set of Non-Intersecting and Simple Polgonal Chains", R. Bar-Yehuda and B. Chazelle, Tecnical Report, Princeton University, 1992.

[3] [FT] "Fibonacci heaps and their uses in Network Optimization Algorithms", M. Fredman and R.E. Tarjan, Proc. 25th Annual IEE Symposium on FOCS, pp.338-146, 1984.

[4] [GH] "Optimal Shortest Path Queries in a Simple Polygon", L.J. Guibas and J.Hershberger, J. Comput. Syst. Sci. 39 (1989),126-152.

[5] [GHLST] "Linear time algorithms for visibility and shortest path problems inside simple polygons" by L.Guibas, J.Hershberger, D.Leven, M.Sharir and R.Tarjan, Algorithmica 2 (1987), 209-233.

[6] [GM] "An output Sensitive Algorithms Algorithm for Computing Visibility Graphs," S.K. Ghosh and D. Mount, SIAM J. Comput. 20 (1991), 888-910.

[7] [HS] "Hershberger J. and S. Suri", On computing Euclidean Shortest paths in the plane, Proc. 34th Annu. IEEE Symposium on FOCS, pp. 508-517.

[8] [K] "An Efficient Wavefront method," S. Kapoor, Tech Report 93/10 (manuscript 1992), CSE Dept. IIT Delhi.

[9] [KM1] "Efficient Algorithms for Euclidean shortest paths and visibility problems amongst polygonal obstacles", Kapoor Sanjiv and S.N. Maheshwari, Proc. ACM Symposium on Computational Geometry, 1988.

[10] [KM2] "Efficient Algorithms for Maximum Flows and Shortest Paths in Polygonal Domains", Sanjiv Kapoor and S.N.Maheshwari, Technical Report 93/5 (manuscript 1991), CSE Dept. IIT Delhi.

[11] [LP] "Euclidean Shortest Paths in the Presence of Rectilinear Barriers", D.T. Lee and F.P. Preparata., Networks, vol.14 (1984), pp.393-41O.

[12] [Mi1] "Planning Shortest Paths", J.S.B. Mitchell, Ph.D dissertation, Dept. of Operations Research, Stanford University, 1986.

[13] [Mi2] "A new algorithm for shortest paths among obstacles in the plane." J.S.B. Mitchell, Annals of Math. and Artificial Intelligence, 3:83-106,1991.

[14] [Mi3] "Shortest Paths among obstacles in the plane," J.S.B. Mitchell, Proc. 9^{th} ACM Symposium on Computational Geometry, pp. 308-317, 1993.

[15] [Mi4] Manuscript on shortest paths, J.S.B. Mitchell (personal communication).

[16] [MS] "Separation and Approximation of Polyhedral Surfaces", J.S.B. Mitchell and S. Suri, Proc. 3rd Annual ACM-SIAM Symposium on Discrete Algorithms, Orlando, FL, 1992, pp.296-306.

[17] [P] "Shortest Path Motion," C.H. Papadimitrou, Proceedings 6^{th} FST & TCS, New Delhi, India, LNCS 241, pp.144-153.

[18] [RS] "Shortest Paths in Euclidean Space with Polyhedral Obstacles," J.Reif and J. Storer, Technical Report CS-85-121, C.S.Dept., Brandeis University, 1985 (to appear in J.A.C.M.).

[19] [R] "Shortest paths in the plane with convex polygonal obstacles," H. Rohnert, Inform. Process. Lett., 23:71-76, 1986.

[20] [SK] "Constructing Vornoi Diagrams and Relative Neighborhood graphs of simple polygons," V. Sarin and S. Kapoor, Proc. of the 4th Canadian Conference on Computational Geometry, 1992.

[21] [W] "Constructing the Visibility Graph for n Line Segments in O(n) time," E.Welzl, I.P.L., Vol 20 (1985), pp.167-171.