Approximate Multi-Visibility Map Computation

Narcís Coll* Marta Fort* J. Antoni Sellarès*

Abstract

A multi-visibility map is the subdivision of the domain of a terrain into different regions that, according to different criteria, encode the visibility with respect to a set of view elements. We present an algorithm for computing approximate multi-visibility maps for a terrain, modeled as a TIN, with respect to a set of view segments. Our approach is based on an algorithm that reconstructs an approximation of an unknown planar subdivision from information gathered from linear probes of the subdivision.

1 Introduction

Visibility information of terrain areas is necessary in many Geographic Information Systems applications, such as path planning, mobile phone networks design and environmental modeling.

The visibility map is an structure that encodes the visibility of a terrain with respect to a view element (point, segment, triangle, ...) belonging to or above the terrain. Algorithms for computing the visibility map of a point on a Triangulated Irregular Network (TIN) are available in [3, 4]. Visibility structures for several view elements, that we generically call multi-visibility maps, can be defined by combining the visibility map of such elements according to some operators, for example intersection, union and counting.

When the representation of a terrain is a rough approximation of the underlying terrain, the approximated computation of a multi-visibility map is often sufficient.

In this paper we address the problem of computing approximate multi-visibility maps for a terrain modeled by a TIN with respect to a set of view segments.

2 Preliminaries

2.1 TIN, visibility, multi-visibility map

A terrain can be modeled as the graph of a *Triangulated Irregular Network*, $(\mathcal{T}, \mathcal{F})$, formed by a triangulation $\mathcal{T} = \{t_1, \cdots, t_n\}$ of the domain $D \subset \mathbb{R}^2$ and by a family $\mathcal{F} = \{f_1, \cdots, f_n\}$ of linear functions such that: a) function $f_i \in \mathcal{F}$ is defined on triangle t_i, $i = 1..n$; b) for every pair of adjacent triangles t_i and t_j, $f_i(x, y) = f_j(x, y)$ for all points $(x, y) \in t_i \cap t_j$.

For any triangle $t_i \in \mathcal{T}$, $\overline{t}_i = f_i(t_i)$ is a triangle in $\mathbb{R}^3$ that we will call a *face* of the TIN, and the restriction of each function f_i to an edge or a vertex of $\mathcal{T}$ is an *edge* or a *vertex* of the TIN, respectively.

Given a terrain $(\mathcal{T}, \mathcal{F})$, we will say that a point $Q = (x, y, z)$ is *above* the terrain if the domain point (x, y) belongs to some $t_i \in \mathcal{T}$ and $z > f_i(x, y)$. A *view element* (point, segment, triangle, ...) is an element belonging to or above the terrain. A point P on the terrain is called *visible* from a *view point* V if the interior points of the line segment joining V and P lie above the terrain. A point P belonging to the terrain is called *(weakly) visible* from a *view segment* v if P is visible at least from a point of v.

Given a set V of r view segments, a *multi-visibility map* is a subdivision of the domain D of the terrain into regions according to different visibility criteria. Some subdivision criteria examples are: the region visible from at least one segment and its complement; the region visible simultaneously from all the segments and its complement; the regions visible from exactly $0, 1, \cdots, r$ segments.

2.2 Planar subdivision reconstruction

Given an unknown *target* bounded planar subdivision $\mathcal{S}$, in [2] a general purpose online reconstruction algorithm is presented that reconstructs an approximation of the unknown target based on information gathered from linear probes of the target. The goal is to recover the vertices, edges and faces of $\mathcal{S}$ based on the information acquired from the probes. Online means that the algorithm maintains an approximation of the planar subdivision after processing the information from each line probe. Since the set of line probes does not provide sufficient information to reconstruct $\mathcal{S}$ exactly, the algorithm progressively refines the approximation which converges to $\mathcal{S}$ as the number of probe lines increases. How the line probes are chosen plays an important role in how accurate the reconstruction is and how quickly it converges. The probing lines are generated uniformly distributed over a bounding box B of $\mathcal{S}$ so that the probability that a line intersects a piece of the boundary of and edge of $\mathcal{S}$, independent of its location and orientation, is proportional to its length [8]. A line probe L on $\mathcal{S}$ partitions L into a finite set of segments with each segment labelled

*Institut d'Informàtica i Aplicacions, Universitat de Girona, Spain, {coll,mfort,sellares}@ima.udg.es. Partially supported by grant TIN2004-08065-C02-02

with the face of $\mathcal{S}$ that contains it. The reconstruction algorithm maintains a triangulation of the endpoints of the segments from which the approximation of the unknown target can be extracted. The mean computational cost of the algorithm when k lines are processed is $O(k \log k)$, where the hidden constant depends on the quotient between the sum of the length of all the edges of $\mathcal{S}$ and the perimeter of B. The algorithm is particularly suited for the setting where computing the intersection of a line with an unknown target is much simpler than computing the unknown target itself.

3 Approximate multi-visibility map computation

We have designed an algorithm for computing approximate multi-visibility maps, according to different criteria, based on the planar subdivision reconstruction algorithm mentioned in previous section.

The input of our algorithm are a TIN $(\mathcal{T}, \mathcal{F})$ and a set of view segments V. The output is an approximate multi-visibility map $\mathcal{M}$. For representing $\mathcal{T}$ and $\mathcal{M}$ we are going to use a DCEL structure. This structure allows for all necessary traversal operations efficiently.

Next we give an overview of the major steps of our algorithm.

- Determine an axis-parallel rectangular bounding box B containing the domain D of the triangulation $\mathcal{T}$. Triangulate the region of B exterior to D to extend $\mathcal{T}$ to a triangulation $\mathcal{T}'$ of B. To this purpose it suffices to add $O(n)$ triangles, where n is the number of triangles in $\mathcal{T}$, with four of them having an edge that coincides with an edge of B. Total cost $O(n)$.

- Generate a set $\mathcal{L}$ of k lines uniformly distributed over B. Each line in $\mathcal{L}$ is obtained in constant time by the following process. Let p be a point selected uniformly at random from the boundary of B, and let ϕ be an angle selected uniformly at random from the interval $[0, \pi)$. A line uniformly distributed over B will be a line l going through p and such that the angle, interior to B, between the edge s of B that contains p and l measures ϕ. Cost $O(k)$.

- For each line $l \in \mathcal{L}$ and each segment $v \in V$, compute $\mathcal{M}_{l,v}$, the restriction on l of the visibility map from v. First we compute the intersection of l with $\mathcal{T}$. Let $a_1, \cdots, a_m$ be the resulting line sections, where $a_j \in t_j$. Cost $O(m)$ (See subsection 3.1). Next we determine the visibility of the segment $\overline{a}_j = f_j(a_j)$ on the face $\overline{t}_j$ of the TIN from the view segment v. Cost $O(n^2 \log n)$ (See subsection 3.2). Finally $\mathcal{M}_{l,j}$ is obtained as the subdivision of l in segments according to the visibility of $\overline{a}_j, j = 1..m$. Cost $O(mn^2)$.

- Denote $\mathcal{M}_l$ the restriction of the multi-visibility map on l with respect to the segments of V. $\mathcal{M}_l$ is obtained by merging the information in $\mathcal{M}_{l,v}$, for each $v \in V$. Naturally, the merging operation depends on the visibility criterion chosen for defining the multi-visibility map. Cost $O(mn^2 \log r)$, where r is the number of segments in V.

- Apply the reconstruction algorithm to the k lines $\mathcal{M}_l, l \in \mathcal{L}$. Cost $O(k \log k)$.

Consequently, the total cost of the algorithm in the worst case is $O(krmn^2 \log n) + O(k \log k)$, where we have supposed that $r \leq n$.

3.1 Line-triangulation intersection

The m line sections resulting from the intersection of l and $\mathcal{T}$ can be computed in $O(m)$ time. To facilitate the task we intersect l with the extended triangulation $\mathcal{T}'$. An edge of the first triangle of $\mathcal{T}'$ intersected by l is the edge of B containing the origin p of l. Starting from this triangle we can traverse $\mathcal{T}'$ following l through adjacent edges in constant time per edge. The whole intersection process takes $O(m)$ time.

From results of integral geometry we know that the mean number of triangles of $\mathcal{T}$ intersected by a line l uniformly distributed over B is $m = \sum_{i=1}^{n} \partial t_i / \partial B$ [8]. Consequently we can considered m as a function depending on the triangulation and the bounding box.

3.2 Segment-segment visibility

Given a view segment v and a segment s on a face of the TIN we want to compute the visible parts of s from v. We will concentrate our attention in the case in which v and s are non coplanar, so that they determine a tetrahedron $\mathbb{T}_{v,s}$. Therefore we will be just interested in the n' faces of the TIN intersecting $\mathbb{T}_{v,s}$. The study of the most simple case in which v and s are coplanar and determine a quadrilateral is omitted in this abstract.

We have adapted to our purpose an algorithm proposed by Bern et al. [1] for computing the visibility with a moving point of view in 3D polygonal scenes and we also use ideas given by S. Ghali in [5] to determine shadow points in 2D polygonal scenes. We will solve the problem by setting a moving point on v, and looking only what happens to s.

Let v be parameterized by "time" t from 0 to 1. We denote v_t the point of v with parameter value t. When points v_0, v_1 have different height we assume that v_0 is located below v_1. As well we consider the segment s parameterized by u from 0 to 1 and we denote s_u the point of s with parameter value u. We choose s_0 and s_1 in such a way that the orthogonal projections onto D of the segments $v_0 s_0$ and $v_1 s_1$ do not intersect.

While we are moving v_t along v, the triangle T_t determined by v_t and s, *the vision triangle*, sweeps the tetrahedron $\mathbb{T}_{v,s}$. During the sweep, for a given time t, the intersection point between an edge e and the vision triangle is the *pierce point*, $p_{e,t}$ (their position and existence depend on t). At time t, the ordered list of edges producing pierce points is the *transparent visibility cycle (tvc_t)*. The edges are ordered according to the angular order around v_t of the pierce points determined by them. The ordered list of edges producing visible pierce points from v_t is the *opaque visibility cycle (ovc_t)*. While we move v_t, no important changes in the visible parts of s are produced unless in a point where there is a change either in tvc or ovc. These points are the *critical points* and we say that a *transparent/opaque topology change* occurs, respectively. It is not difficult to see that a transparent topology change occurs at time t if and only if there are two edges e and e' such that there is a line segment that intersects v_t, e, e' and s. And it is an opaque topology change on s if, in addition, the line segment does not pass through the interior of any face.

We are interested in determining the opaque topology changes on s. Using ovc and tvc we are able to determine all the critical points on v, but we are not able to decide if the corresponding topology change is opaque or transparent. With a preprocess we can update ovc and determine where the opaque topology changes occur [1]. To avoid this preprocess we consider the *back face* of the edges producing pierce points. The back face of the edge e at time t, denoted $bf_{e,t}$, is the first face intersected by the ray originated in v_t passing through $p_{e,t}$. The face $bf_{e,t}$ may change when there is a transparent topology change. We will store with each edge e in tvc_t its back face $bf_{e,t}$. When moving v_t, in order to update the back face of an edge in $O(\log n')$ time, we use some "auxiliary" edges and faces. Let E be the set of segments obtained intersecting the TIN edges and $\mathbb{T}_{v,s}$. We take as auxiliary edges the vertical segments joining an endpoint of a segment in E with its orthogonal projection onto D. We take as auxiliary faces the quadrilaterals determined by each segment in E and its orthogonal projection onto D. Observe that for each TIN edge we introduce at most two new auxiliary edges and a face. From now on, faces and edges mean both, the ones from the TIN and the auxiliary ones.

In practice, during the sweep, we have three types of critical points characterized by the following events: (1) an endpoint of an edge is reached by T_t; (2) a non-auxiliary edge intersects the boundary of $\mathbb{T}_{v,s}$ (3) two edges exchange their order in tvc, what means that the pierce points they determine exchange their angular position. As $\mathbb{T}_{v,s}$ intersects with $O(n')$ edges, the number of critical points is at most $O(n'^2)$.

Let us focus in the algorithm. The input is the TIN $(\mathcal{T}, \mathcal{F})$, the view segment v and the segment s on a face of the TIN. The output is the set of the endpoints of the visible sub-segments of s ordered by u.

The algorithm has two main parts:

1. *Opaque topology changes on s:* We move v_t along v sweeping the tetrahedron $\mathbb{T}_{v,s}$ with the vision triangle T_t. During the sweep we determine the events corresponding to critical points along v. These points are characterized by 4-tuples (t, e, e', u) such that s_u is the intersection point between s and the line passing through v_t, e and e'. We only store the 4-tuples of critical points that corresponds to opaque topology changes on s, the *shadow points*, in an ordered list $\mathcal{C}_s$.

2. *Visible parts of s:* First we obtain the visible parts of v looking from point $s_0 \in s$. Next we determine the visible parts of s by traversing the set of 4-tuples (t, e, e', u) that characterize the opaque topology changes on s in an ordered way from $u = 0$ to $u = 1$.

3.2.1 Opaque topology changes on s

Events determination. The coordinates t and u that correspond to the two events of type (1) and (2) determined by and edge e can be easily computed in constant time. We will characterize them by a 4-tuple (t, e, e, u).

To obtain the events of type (3) we can consider all the pairs of edges intersecting $\mathbb{T}_{v,s}$ and compute the time were they would exchange their angular position. In spite of using brute force, we only consider the pairs of edges which are adjacent in tvc_t at some time t. For this reason we will obtain them during the sweep process. The coordinates t and u for the events of type (3) are obtained using the skew projection [1].

Events are stored in a priority queue ordered by t. Different events can occur at the same t, in this case the last events to handle are the events of type (2) with e as an auxiliary edge and $u \neq 1$. During all the process we will only store in the priority queue events with t and u between 0 and 1.

Obtaining tvc_0 and ovc_0. Once we have the priority queue initialized with events of type (1) and (2), we can obtain tvc_0 and ovc_0.

We take the set E of edges e corresponding to the events $(0, e, e, u)$ and we delete these events from the priority queue. The transparent visibility cycle tvc_0 contains the edges of E ordered by increasing u. We determine ovc_0 and the back face $bf_{e,0}$ of each edge $e \in E$ by an angular sweep around v_0 in $O(n' \log n')$ time. We store in the list $\mathcal{L}_0$ the faces intersecting the line $v_0 s_1$ by increasing distance from v_0.

Once tvc_0 has been computed we can obtain the events of type (3) by traversing tvc_0. For each pair of adjacent edges in tvc_0 we compute a 4-tuple (t, e, e', u) and we store it in the priority queue.

Handle events. Assume that for a critical point occurred at t' we have $tvc_{t'}$, $ovc_{t'}$, and $\mathcal{L}_{t'}$ (the ordered set of faces intersecting the line $v_{t'}s_0$). To handle the next event $(t, e, e', u), t \geq t'$, found during the sweep we must proceed as follows:

1) Obtain tvc_t and ovc_t. The updating process depends on the type of the event handled, there is one method for events of type (1) or (2) and another for events of type (3). We use the auxiliary edges in order that this update can be done in $O(\log n')$ time. The details are not given in this abstract.

2) Find new events of type (3). We check for new adjacencies in tvc_t and we add the new events in the priority queue.

3) Select opaque topology changes on s. If there has been a change in ovc and the face containing s belongs to $\{bf_{e,t'}, bf_{e',t'}, bf_{e,t}, bf_{e',t}\}$ we store the 4-tuple (t, e, e', u) in the list $\mathcal{C}_s$ of shadow points on s.

Complete the list $\mathcal{C}_s$ of shadow points. Finally we must complete $\mathcal{C}_s$ with the shadow points produced by v_0 and v_1. These points are obtained by traversing ovc_0 and ovc_1. For each edge e such that $s \subset bf_{e,j}, j = 0, 1$ we store in $\mathcal{C}_s$ the 4-tuple $(0, e, v_j^s, u)$ where v_j^s is the vertical segment joining v_j with its orthogonal projection onto D.

3.2.2 Visible parts of s

Visible parts of v from s_0. By working in the triangle defined by v an s_0 we partition v in segments according to the visibility from s_0. Each visible segment of v is determined by a pair of edges (e, e'). The visibility changes in v are obtained by making a sweep around s_0 similar to the sweep made to obtain tvc_0 and ovc_0 but considering v as a face. It takes at most $O(n' \log n')$ time (details omitted in this abstract). At the end of the process the pairs (e, e') are stored in the set of visible parts of v from s_0, $\mathcal{V}_{v,0}$.

Visible parts of s from v. For each 4-tuple in $\mathcal{C}_s$ we update in $O(\log n')$ time $\mathcal{V}_{v,u}$, the set of visible parts of v from s_u, and we obtain the set of endpoints of the the visible segments of s according to a process similar to the one given in [5].

4 Visibility from a polygon and inter-regions visibility

If we are interested in a multi-visibility map from a view polygon placed on or over a terrain we just have to consider the visibility from the segments forming the boundary of this polygon [9] and then merge the visibility information obtained for each single boundary segment to obtain information of the whole view polygon.

A region on a TIN is the subset of faces of the TIN determined by a connected subset of triangles of the domain triangulation. Our algorithm can be easily adapted to determine the approximate weak visibility from region R to region R' [7]. Just consider the region R as a set of view polygons and use the projection onto D of the region R', instead of the whole D, to take random lines.

References

[1] M. Bern, D. Dobkin, D. Eppstein, and R. Grossman. Visibility with a moving point of view, In *Algorithmica* 11, pages 360-378, 1994.

[2] N. Coll, F. Hurtado and J.A. Sellarès. Approximating planar subdivisions ans generalized Vornoi diagrams from random sections, In *19th European Workshop on Computational Geometry*, pages 27-30, 2003.

[3] L. De Floriani, P. Magillo, Visibility Algorithms on Triangulated Digital Terrain Models, In *International Journal of Geographic Information Systems* 8(1), pages 13-41, 1994.

[4] L. De Floriani, E. Puppo, P. Magillo, Applications of Computational Geometry to Geographic Information Systems, In *Handbook of Computational Geometry, J.R. Sack, J. Urrutia (Eds.)*, pages 333-388, Elsevier Science, 1999.

[5] S. Ghali, Computation and Maintenance of Visibility and Shadows in the Plane, In *Sixth Int. Conf. in Central Europe on Computer Graphics and Visualization*, pages 117-124, 1998.

[6] Marc J. van Kreveld, Digital Elevation Models and TIN Algorithms, In *Algorithmic Foundations of Geographic Information Systems, LNCS 1340*, pages 37-78, 1997.

[7] Marc J. van Kreveld, E. Moet, R. van Ostrum, Region inter-visibility in terrains, In *Proc. 20th European Workshop on Computational Geometry*, pages 155-158, 2004

[8] L.A. Santalò, Geometric Probability, In *Society for Industrial and Applied Mathematics*, 1976.

[9] C.An Wang,B.Zhu, Three-dimensional waeak visibility: Complexity and applications, In *Theoretical Computer Scinece* 234, pages 219-232, 2000.