Algorithms for Visibility Computation on Terrains: A Survey

Article *in* Environment and Planning B Planning and Design · September 2003

DOI: 10.1068/b12979 · Source: RePEc

2 authors:

Leila De Floriani
University of Maryland, College Park
387 PUBLICATIONS **5,848** CITATIONS

SEE PROFILE

Paola Magillo
Università degli Studi di Genova
126 PUBLICATIONS **1,921** CITATIONS

SEE PROFILE

Environment and Planning B: Planning and Design 2003, volume 30, pages 709 – 728

DOI:10.1068/b12979

Algorithms for visibility computation on terrains: a survey

Leila De Floriani, Paola Magillo
Department of Computer and Information Sciences (DISI), University of Genova,
Via Dodecaneso 35, 16146 Genova, Italy; e-mail: deflo@disi.unige.it, magillo@disi.unige.it
Received 12 August 2002; in revised form 9 December 2002

Abstract. Several environment applications require the computation of visibility information on a terrain. Examples are optimal placement of observation points, line-of-sight communication, and computation of hidden as well as scenic paths. Visibility computations on a terrain may involve either one or many viewpoints, and range from visibility queries (for example, testing whether a given query point is visible), to the computation of structures that encode the visible portions of the surface. In this paper, the authors consider a number of visibility problems on terrains and present an overview of algorithms to tackle such problems on triangulated irregular networks and regular square grids.

1 Introduction

Many interesting applications involving geographic information systems (GISs) require visibility computations. Examples include the optimal placement of observation points according to certain visibility constraints, line-of-sight communication problems, and computation of paths with certain visibility properties (scenic or hidden paths). Terrain problems that can be solved by using visibility information are described elsewhere (for instance, Fisher, 1996a; Nagy, 1994; O'Sullivan and Turner, 2001).

Viewpoint-based placement problems require the distribution of several observation points on a terrain in such a way that a large part of the surface is visible. Applications include, for instance, the location of fire towers and radar sites. A dual set of problems requires one to find points that are not visible from a large part of the terrain. A typical application would be the placing of facilities in areas of natural or archaeological interest.

The objective in line-of-sight communication problems is to find a network that connects two or more sites in such a way that every two consecutive nodes of the network are mutually visible. Applications include, for instance, the location of radio, television, or telephone transmitters.

A desired property for a path on a terrain can be related either to its visibility or to its invisibility. For instance, a scenic path is desired for tourism or hiking, whereas a hidden path is appropriate for building a highway in a natural environment, and so on.

The solution to such problems requires methods to answer visibility queries efficiently and/or the development of effective data structures to encode terrain visibility. *Visibility queries* deal with finding whether a given object located on a terrain is visible from some viewpoint and, possibly, how much of it is visible. *Visibility structures* provide information about the visibility of the terrain surface itself. Effective visibility structures for a terrain can also help in answering visibility queries.

In this paper, we first introduce some basic notions concerning terrain models and visibility (section 2). Then, in section 3, we define visibility structures and visibility queries. The central part of the paper (sections 4 and 5) contains a survey of algorithms proposed in the literature to compute visibility structure and to solve visibility queries. In section 6, we address visibility problems and algorithms in the context of multiresolution terrain models, which can be used to reduce computational complexity

by adapting the level of resolution in terrain representation to the requirements of a specific visibility operation. Some conclusions are presented in section 7.

2 Preliminaries and background

A *topographic surface*, or *terrain*, can be viewed from a mathematical perspective as the image of a bivariate function f defined over a domain D in the Euclidean plane. A *digital elevation model* (DEM) is a model of terrain built on the basis of a finite set of digital data. Terrain data consist of elevation measures at a set of points S in the two-dimensional domain D. Points in S can either be scattered or distributed on a regular grid. A DEM built on S represents a surface that interpolates, or approximates, the measured elevations at the points of S. Thus, two classes of DEMs are usually considered in GISs: triangulated irregular networks (TINs) and regular square grids (RSGs).

A TIN is defined by a triangulation T of the domain D having its vertices at the points of S. Function f is generally defined piecewise as a linear function over each triangle of T. Thus, the surface described by a TIN consists of a collection of planar, triangular patches [figure 1 (a)]. An RSG is defined by a regular grid inducing a partition of the domain D into equally sized rectangles (such rectangular cells are also called *pixels* for analogy with raster images). Function f is defined piecewise over each rectangle. Depending on the desired degree of continuity, a constant function

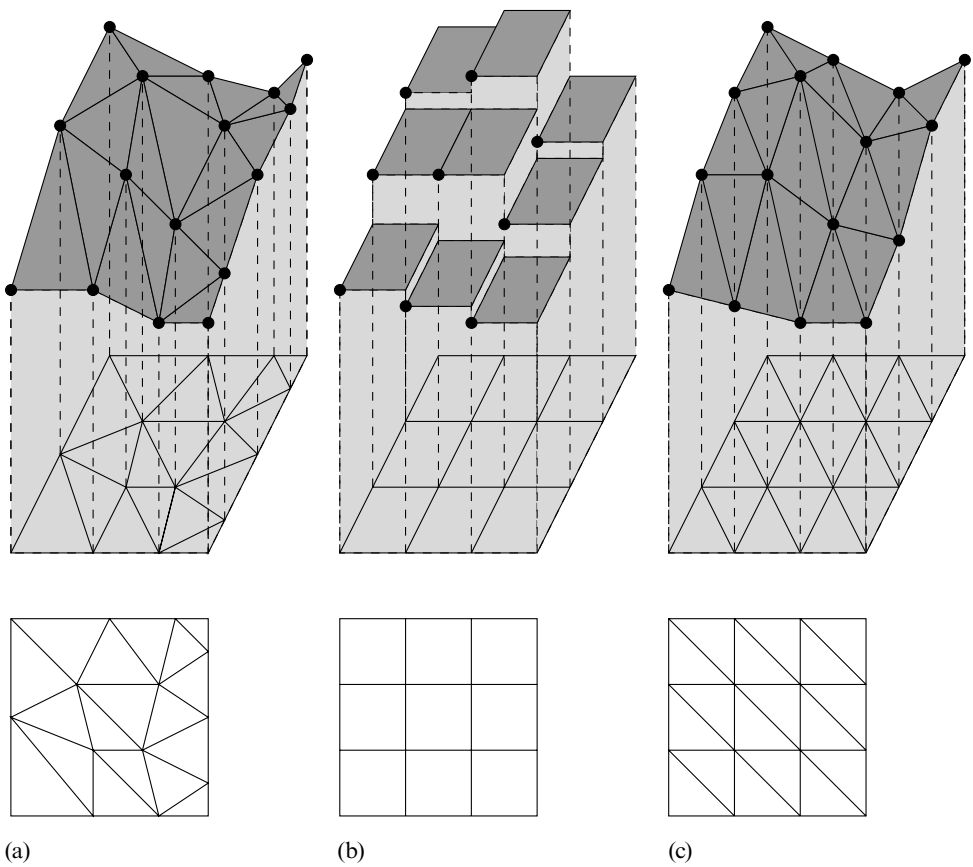


Figure 1. (a) A triangulated irregular network; (b) a stepped regular square grid (RSG); and (c) a triangulated RSG.

or a bilinear (quadratic) function can be used. Alternatively, each rectangle can be divided into two triangles, with linear interpolating functions defined on them [see figures 1(b) and (c)]. An RSG with constant functions is also called a *stepped model*. In this case, each rectangle is identified by one of its vertices (for example, the bottom left-hand vertex).

Given a topographic surface, two points, p_1 and p_2, are said to be mutually *visible* if and only if the interior of segment p_1p_2 lies strictly above the surface. Thus, a necessary condition for visibility is that both p_1 and p_2 lie on or above the surface. Any point lying on or above a terrain can be chosen as a *viewpoint*.

3 Visibility problems

Visibility information is related either to portions of the terrain surface itself or to objects located on or above the surface. Testing the visibility of an object is a query problem and must be solved 'on-the-fly'. In contrast, visibility information on the surface itself can be precomputed and stored in suitable data structures. The information contained in such data structures helps one to answer visibility queries efficiently.

3.1 Visibility structures for terrains

In this subsection we introduce the main structures used to represent the visibility of a terrain from one viewpoint or from several viewpoints. We first give an abstract definition of each data structure and then present implementations for TINs and RSGs.

The basic visibility structure for a terrain is the *viewshed* (see figure 2, over). Given a viewpoint v on a terrain, the viewshed of v is the set of the points on the surface that are visible from v. That is:

$$\text{viewshed}(v) = \{p \text{ in } D | p \text{ is visible from } v\}. \tag{1}$$

Sometimes, the viewshed is defined within a distance r from the viewpoint:

$$\text{viewshed}(v, r) = \{p \text{ in } D | \text{distance}(v, p) < r, \text{ and } p \text{ is visible from } v\}. \tag{2}$$

Other visibility structures are the *horizons* of a viewpoint v. The *local horizons* are the loci of points on the terrain that are visible from v and that block the view of points lying immediately beyond them (see figure 3, over):

$$\begin{aligned}\text{local horizons}(v) = \{&p \text{ in } D | p \text{ is visible from } v, \text{ and there is no point } q \text{ in } D,\\ &\text{distinct from } p, \text{ such that } p \text{ belongs to segment } vq \text{ and all points}\\ &\text{on segment } pq \text{ are visible from } v\}.\end{aligned} \tag{3}$$

The *global horizon*, also called simply the *horizon*, of v collects, for every radial direction v, the farthest point on the terrain that is visible from v. Intuitively, it corresponds to the 'distal boundary' of the viewshed (see figure 2) and is a subset of the local horizons:

$$\begin{aligned}\text{horizon}(v) = \{&p \text{ in } D | p \text{ is visible from } v \text{ and, for every point } q \text{ such that } p\\ &\text{belongs to segment } vq, q \text{ is not visible from } v\}.\end{aligned} \tag{4}$$

Some authors (Fisher, 1996a; 1996b; Van Kreveld, 1996) have proposed *extended viewsheds*, which provide, for each point p in D, richer information than a simple Boolean classification (visible – not visible). Such information can consist of:
(1) one of four values corresponding to the following meanings: p is not visible, p is on a global horizon, p is on a local horizon, p is visible and not on any horizon;
(2) a *local offset*, which is defined as the difference between the height of p and the height of the nearest local horizon lying in front of p, with a positive or a negative sign

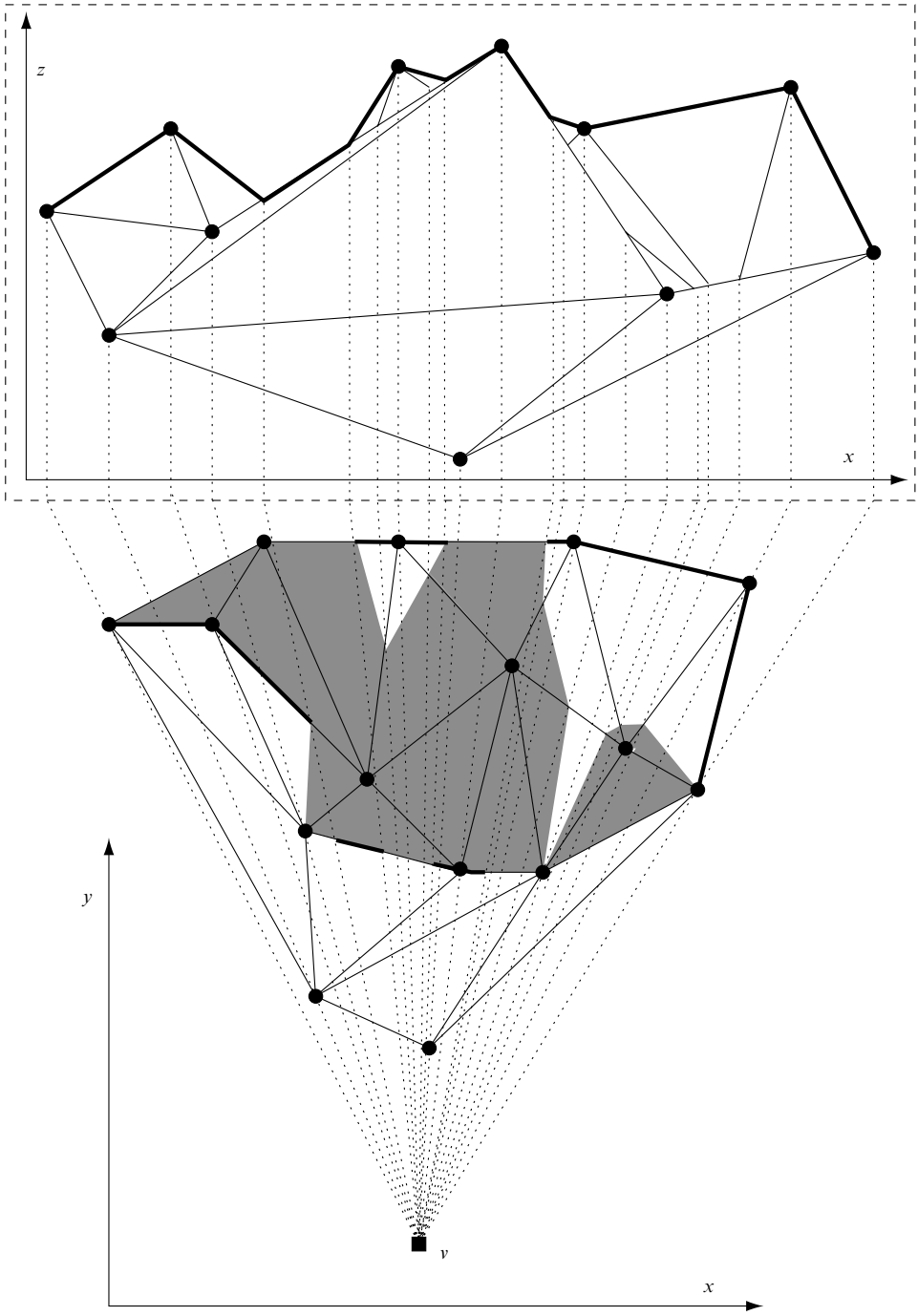


Figure 2. Lower part: the viewshed of a viewpoint v on a triangulated irregular network (TIN). Invisible areas are shaded grey. Upper part: the image of the TIN as seen from v with use of the xz plane as a viewplane for the projection. Note: the thick segments mark the horizon (on the xy plane) and the upper envelope of the TIN (on the viewplane).

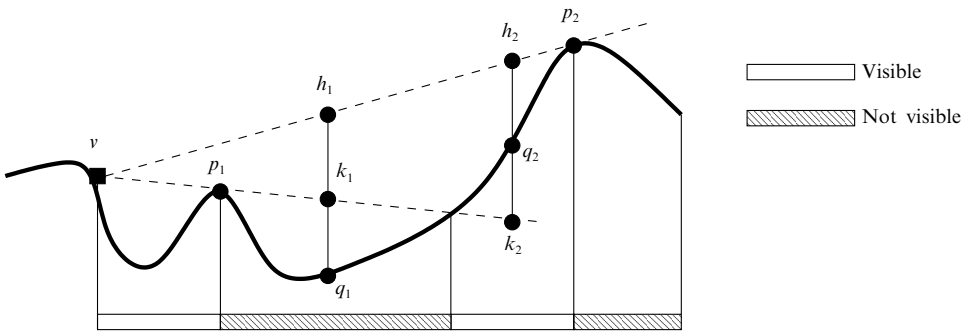


Figure 3. Cross-section of a terrain with its viewshed. Points p_1 and p_2 are local horizons; p_2 is also a global horizon. For nonhorizon points $q_i (i = 1, 2)$, height difference from k_i is the local offset, and height difference from h_i is the global offset. The difference gives a positive sign for visible point q_2 and a negative sign for q_1, which is not visible.

depending on whether p is visible or not visible; clearly, points lying on a horizon have a zero offset;

(3) a *global offset*, which is defined in a similar way to the local offset but with respect to the global horizon.

Local and global offsets are illustrated in figure 3. It is not difficult to see that extended viewsheds contain sufficient information to retrieve the viewshed and the local or global horizons (see Fisher, 1996b).

Visibility structures for several viewpoints can be defined by combining the viewsheds of such points according to some operator. Common combination operators are:

(a) *overlays*, which involve superimposing the viewsheds of all viewpoints, and labeling each region in the resulting partition of the domain D with the set of viewpoints from which that region is visible [see figure 4(c)];

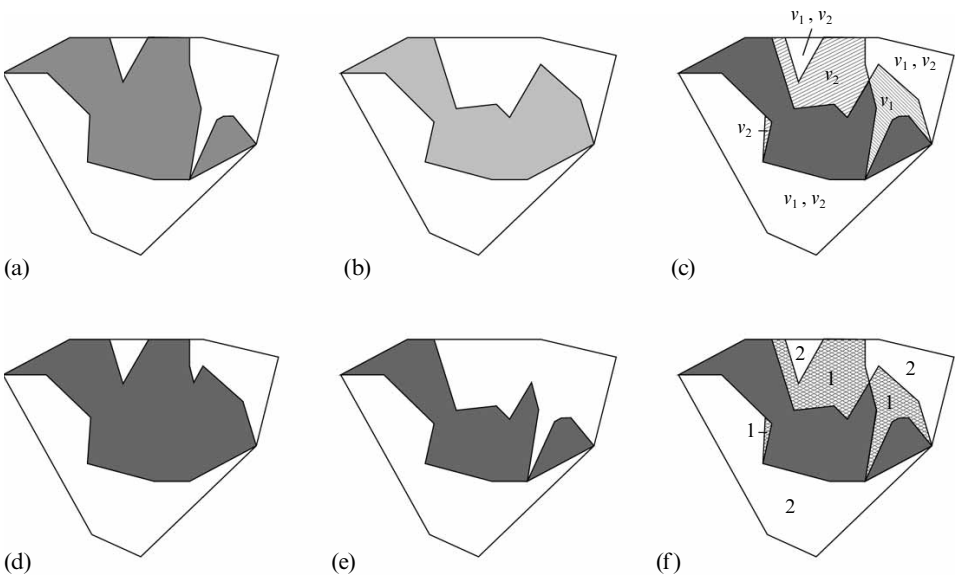


Figure 4. The viewsheds of (a) viewpoint v_1 and (b) viewpoint v_2, and some multivisiblity structures obtained through different combination operators: (c) overlay; (d) union; (e) intersection; and (f) count.

(b) *Boolean operators*, where the intersection of viewsheds gives the portions of a surface visible from all viewpoints [figure 4(e)], and the union of viewsheds gives the portions visible from at least one viewpoint [figure 4(d)], and so on;
(c) *counting operators*, which are similar to overlays but, instead of considering the set of viewpoints from which a region is visible, only the cardinality of such a set (that is, the number of viewpoints from which the region is visible) is considered; we cannot distinguish, in this case, between regions that are visible from the same number of viewpoints but from a different set of such viewpoints [see figure 4(f)].
Note that the overlay of viewsheds [figure 4(c)] contains more information than any other structure. The set of viewpoints considered is usually restricted to a subset of the vertices of the DEM. In the remainder of this paper, for brevity the visibility structures for multiple viewpoints will be called *multivisibility structures*.

Any visibility structure can be encoded either in a continuous way or in a discrete way. A continuous representation for the viewshed partitions each cell of the DEM into its visible and invisible parts. This form is called a *continuous visibility map* and it is used mainly for TINs. The continuous visibility map of a TIN with n vertices has a worst-case space complexity in $O(n^2)$, as every triangle can cast a shadow on all the triangles lying beyond it (Cole and Sharir, 1989). A *discrete visibility map* for a TIN is sometimes considered (for example, Lee, 1991). Such a map is obtained by marking each triangle or each vertex as visible or invisible. The spatial complexity is $O(n)$ for a TIN with n vertices, because only one mark is stored for each triangle or vertex.

The viewshed on RSGs is usually encoded in a discrete way, by marking every grid cell as visible or invisible. The resulting matrix of Boolean values is called a *discrete visibility map* and has a space complexity of $O(n)$ for an RSG with $n^{1/2} \times n^{1/2}$ vertices. RSGs are usually dense; thus, discrete visibility maps are accurate enough. However, a continuous visibility map would be difficult to compute for an RSG, and its size would be huge.

A continuous representation of the horizon of a viewpoint v consists of a sequence of portions of terrain edge, radially sorted around v. This form is used for TINs. The space complexity for a TIN with n vertices is $O[n\alpha(n)]$, where α is the slowly growing inverse of Ackermann's function (Cole and Sharir, 1989) and, in practice, can be considered as a constant.

Approximated representations of horizons have been proposed for TINs and RSGs, dividing the space around the viewpoint into a fixed number of radial sector and recording the maximum terrain elevation within each sector. The space requirement is linear in the number, s, of sectors.

Extended viewsheds rather than horizons are usually encoded on an RSG, as they contain more information. Discrete representations of extended viewsheds consist of matrices in which each element stores either a two-bit integer representating the four-value classification (not visible, local horizon, global horizon, visible) or a real number representing the local or global offset of the corresponding grid vertex. The space complexity is $O(n)$ for a grid with n vertices.

Multivisibility structures can be encoded in many different ways. Mainly, discrete representations are used because of the huge size of a continuous representation. The *visibility graph* (Puppo and Marzano, 1997) represents the overlay of the discrete visibility maps of several viewpoints. It consists of a graph in which each node corresponds to a vertex or to a cell of a DEM and in which every pair of mutually visible nodes is joined by an arc. The space complexity is $O(n^2)$ for a DEM with n vertices. Visibility graphs are used for RSGs and for TINs.

Visibility counts, also called *visibility indexes*, are discrete representations of multivisibility structures obtained through counting operators. Each vertex or cell of a DEM

is labeled with the number of viewpoints from which it is visible. Visibility counts are used mainly for RSGs. They can be obtained by counting all the true values in each row of the matrix implementing the discrete visibility map. A special case is the *intervisibility map* between two regions (Mills et al, 1992). Given a *source region* and a *destination region* (for example, two rectangular blocks of pixels in an RSG), each point of the destination region is labeled with the number of points of the source region from which it is visible. The size of the structure is equal to the size of the destination region. Discrete representations of multivisibility structures based on Boolean operators reduce to a stored Boolean value for each DEM vertex or cell.

3.2 Visibility queries

Visibility queries involve computation of the visibility of objects located on the terrain rather than for the surface itself. Given a viewpoint, v, a visibility query is defined by providing a query object, the problem consisting of determining the visibility of the object from v.

A visibility query for a point q simply requires one to test whether or not q is visible from v. As for the visibility of the terrain itself, the visibility of an object, which is not a point, can be encoded either in a continuous or in a discrete way. In the continuous approach, the query is answered by computing a partition of the query object into visible and invisible subsets. A discrete answer consists of marking the object with a Boolean value, according to some convention: the answer can be true either when the object is entirely visible or when it is at least partially visible, or when it is visible for more than a certain percentage of its extent, and so on.

4 Visibility computation

4.1 Computation of continuous visibility structures

Continuous representations of visibility structures are used for TINs. Thus, the algorithms to compute them exploit the fact that a TIN describes a terrain as a collection of planar faces. Objects with planar faces have received interest both from the GIS community and from computational geometry specialists working on GIS problems. Some algorithms are of theoretical interest for their good asymptotic complexity, but they are difficult to implement. Other algorithms have been successfully implemented and show good practical performance but exhibit a poor worst-case complexity, or their theoretical complexity is not known.

The computation of the continuous visibility map of a TIN is strictly related to the more general problem of *hidden surface removal* (HSR) for a three-dimensional scene. This problem requires the computation of the visible parts of each polygon in the scene projected onto a viewplane, and encompasses the viewshed of a triangulated terrain as a special case (see figure 2). In a similar way, the horizon of a TIN is equivalent to the upper envelope of the image of the terrain projected onto the viewplane (see figure 2). The *upper envelope* of a set of entities in a plane gives, for each horizontal location, the point belonging to an entity of the set that has the maximum height.

A common approach to computing the visibility map of a TIN consists of processing the faces in *front-to-back order* with respect to the viewpoint v. We say that a triangle t_1 of a TIN is in front of a triangle t_2 if there exists a ray emanating from v that intersects both triangles and if the intersection with t_1 lies nearer to v than the intersection with t_2. A front-to-back order is any total order extending the in-front relation: if t_1 precedes t_2 in such order, then t_2 cannot be in front of t_1 (see figure 5, over). A DEM is called *sortable* with respect to v if a front-to-back order exists. Sortability is not guaranteed for all TINs. Delaunay TINs, which include triangulations built from regular grids, are always sortable (De Floriani et al, 1991).

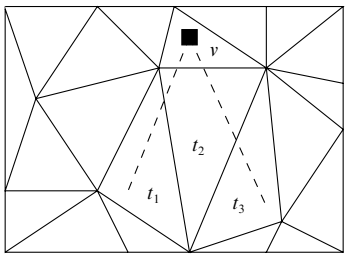


Figure 5. The in-front relation on a triangulated irregular network: triangle t_2 is in front of triangles t_1 and t_3; triangles t_1 and t_3 are not related. A front-to-back order must satisfy the inequalities $t_2 < t_1$, and $t_2 < t_3$, and one of the inequalities $t_1 < t_3$ and $t_3 < t_1$ may also hold.

A nonsortable TIN can be made sortable by splitting some of its triangles (Cole and Sharir, 1989). The front-to-back approach is based on the observation that a triangle can be hidden only by triangles lying in front of it.

An incremental algorithm for computing the visibility map processes one triangle of the input TIN at a time according to a front-to-back order by starting from those lying nearest to the viewpoint v. The visible map of triangles already processed is maintained and updated. A current horizon is also maintained and used to compute the visible portion of each new triangle. An implementation of such algorithm (De Floriani et al, 1989) shows a nearly linear execution time, although the asymptotic complexity is $O[n^2\alpha(n)]$ in the worst case. The front-to-back traversal of the triangles is performed by incrementally building a star-shaped polygon around the viewpoint v, starting from the triangle containing v and adding one triangle at a time until all triangles have been included in the polygon. In figure 6 we show a generic stage of the process, with the current star-shaped polygon and horizon. Triangles externally adjacent to the boundary of the current star-shaped polygon are candidates to be processed next. If a candidate triangle t can be included in the polygon and can maintain that polygon as a star shape, then t is processed. The visible portions of t are determined by projecting the current horizon onto the supporting plane of t. A parallel implementation of the algorithm, achieved by partitioning data into radial sectors has been described elsewhere (De Floriani et al, 1994).

Other algorithms, still based on the incremental front-to-back approach, are *output-sensitive*; that is, their worst-case time complexity is proportional to the output size. This is achieved by storing the current horizon into more complex data structures (Preparata and Vitter, 1992; Reif and Sen, 1988). The complexity is of $O[(n+d)\log^2 n]$, where n is the number of TIN vertices and d is the output size. The interest of such algorithm is mainly theoretical.

Reif and Sen (1988) proposed an output-sensitive parallel algorithm for HSR on a sortable TIN, based on a variation of the front-to-back approach. They assumed a pool of free processors from which processors can be requested at run time. The projections of the terrain edges onto the xy plane are recursively partitioned into equal-sized subsets, lying one in front of the other, and the upper envelope of each group, within each level, is computed in parallel. Finally, for each edge in parallel, the intersections of this edge with the envelopes in front of it are computed.

General HSR for three-dimensional scenes are inefficient for TINs because they do not take advantage of the fact that a terrain does not have a 'bottom' surface (Nagy, 1994).

The HSR algorithm of Katz et al (1991) is designed for a wider class of three-dimensional scenes and is then specialised for terrains. For a sortable TIN with n vertices, the time complexity is equal to $O\{[d+n\alpha(n)]\log n\}$, where d is the output size.

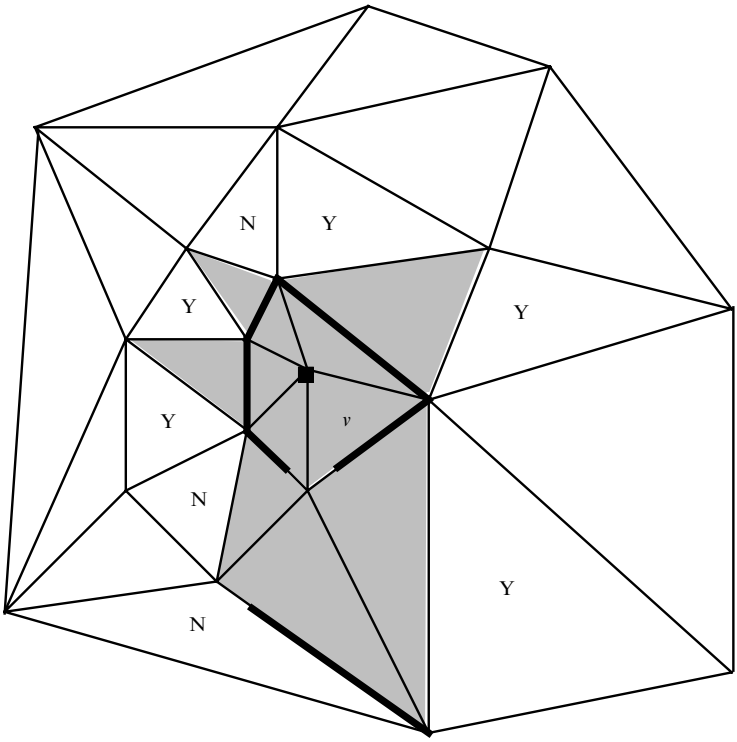


Figure 6. A generic stage of the incremental front-to-back viewshed algorithm. Triangles already processed are shaded gray. The current horizon (thick segments) is shown. Triangles adjacent to the area already processed are labeled Y if all triangles preceding them have been processed (thus they can be processed) and N otherwise.

The algorithm takes a 'divide-and-conquer' approach in which the 'divide' step is made according to a front-to-back order of the objects. Input objects are associated with the leaves of a balanced binary tree, T, in such a way that a left-to-right traversal of the leaves of T corresponds to the front-to-back order of the objects. In the 'conquer' step, tree T is traversed twice: during the first (bottom-up) traversal, for each node N, the union $U(N)$ of the images of all objects associated with the leaves of the subtree rooted at N is computed. During the second (top-down) traversal, the visible portion $V(N)$ of $U(N)$ is computed for each node N. The visible portions of each of the objects are thus the sets V associated with the leaves of T. In figure 7 (see over) we show sets U and V computed for a terrain. Teng et al (1995) have presented a parallel version of this algorithm. They consider one level of the tree at a time and perform all the operations related to the nodes at that level in parallel.

Sorensen and Lanter (1993) computed the continuous visibility map on an RSG. In order to compute the visible and invisible portions of a target cell, they found all the cells lying between the source cell containing the viewpoint and the target cell. Then, they considered the union of the shadows cast by these cells onto the target cell. However, the algorithm is not designed for efficiency but to show some limits of the discrete version of the visibility map.

The horizon of a TIN can be obtained as a side-product of algorithms for computing visibility maps. For instance, it is maintained as an auxiliary structure in the incremental front-to-back approach. The horizon can also be computed as the upper envelope of the set of segments obtained by projecting the terrain edges on a viewplane.

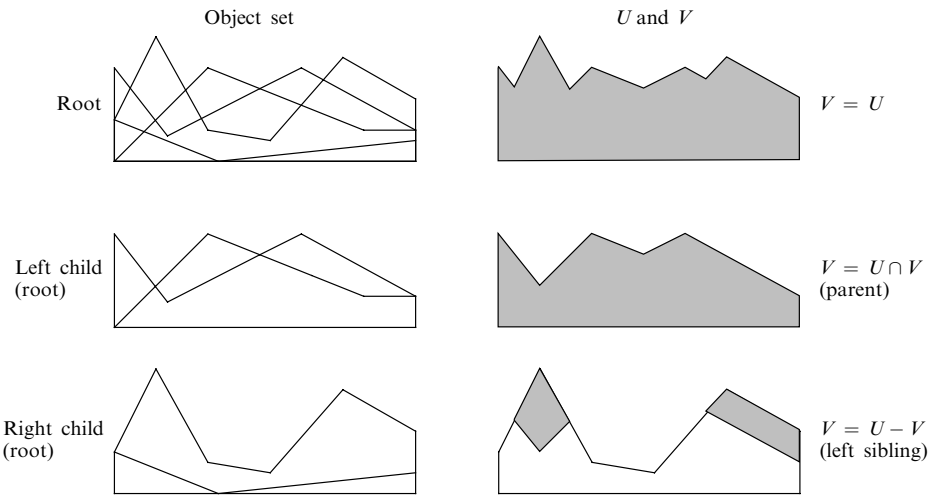


Figure 7. Computation of sets U (union) and V (visible portion; shaded) for the first level of the tree in the HSR (hidden surface removal) algorithm, by Katz et al (1991).

The inherent complexity of this problem is $\Theta(n \log n)$, as it is equivalent to a sorting problem. Algorithms for computing upper envelopes use either a divide-and-conquer or an incremental approach. Divide-and-conquer algorithms run in $O[n\alpha(n) \log n]$ time (Atallah, 1983) or in worst-case optimal $O(n \log n)$ time (Hershberger, 1989). An incremental approach has an $O[n^2\alpha(n)]$ time complexity in a straightforward implementation, whereas a more refined version (De Floriani and Magillo, 1995) provides an expected running time in $O[n\alpha(n) \log n]$, when averaging over all possible permutations of the input data.

Stewart (1998) computes an approximated horizon for all points of the DEM at the same time. In this method only the vertices of the DEM are considered, disregarding the connectivity structure of the underlying grid or triangulation. For each viewpoint, s radial sectors are considered and the horizon is assumed to have a constant elevation within each sector. The algorithm processes the ith sector for all viewpoints at the same time. The worst-case time complexity is $O(s\, n \log^2 n)$ for a terrain with n vertices. Experiments show that it is especially efficient for large terrains and that the method is suitable for a parallel implementation on a per-sector basis.

4.2 Computation of discrete visibility structures

Algorithms for discrete visibility maps are based on intersecting lines of sight emanating from the viewpoint and DEM cells. A straightforward approach requires $O(n^2)$ time on a TIN, and $O(n^{3/2})$ time on an RSG with n vertices. Usually, only the edges of the cells are taken into account on an RSG, with the behaviour of the terrain inside the grid cells not taken into account. The edges are modeled as straight-line segments.

Lee (1991) computed the discrete visibility map of a TIN by using a front-to-back approach similar to one used in an earlier publication (De Floriani et al, 1989). Here, a triangle is considered to be visible if all its three edges are completely visible; otherwise, it is not considered visible.

Discrete visibility maps are mainly used for RSGs. The method by Shapira (1990) traces a line of sight from the viewpoint v to any other point p, called a *target*. The algorithm walks along the line of sight from v to p. If an intersection between the line of sight and a terrain edge is found before reaching p, then p is not visible; otherwise,

p is visible. This method performs redundant computations, as rays to different points of the grid may partially overlap.

Given a viewpoint v, a minimal set of lines of sight, covering all the pixels in the destination region, is provided by the rays from the viewpoint to pixels lying on a *far side* of the destination region. A far side is a boundary edge l of the destination region such that both the region and v lie on the same side with respect to l (see figure 8). The method by Blelloch (1990) considers only those rays joining the viewpoint v to a point on a far side of the destination region, and determines the visibility of each cell along a line of sight by using the current tangent from the viewpoint to the terrain. The complexity can be further reduced by sampling a subset of line-of-sight directions around a viewpoint, thus leading to approximate discrete visibility maps.

A very efficient, approximated, algorithm has been proposed by Franklin and Ray (1994) to compute the discrete visibility map and visibility offsets within a distance r from the viewpoint on an RSG. The algorithm explores cells in concentric rings starting from the ring adjacent to the viewpoint v. The visibility of a point p lying on the ith ring is determined by looking at the offsets of two points lying on the previous ring, namely the two points p_1 and p_2, such that line vp passes between them. The elevation of segment p_1p_2 is interpolated and used to estimate the elevation of the line of sight over p. The method is approximated because p can be obscured by a point lying on an inner ring, thus not checked by the algorithm.

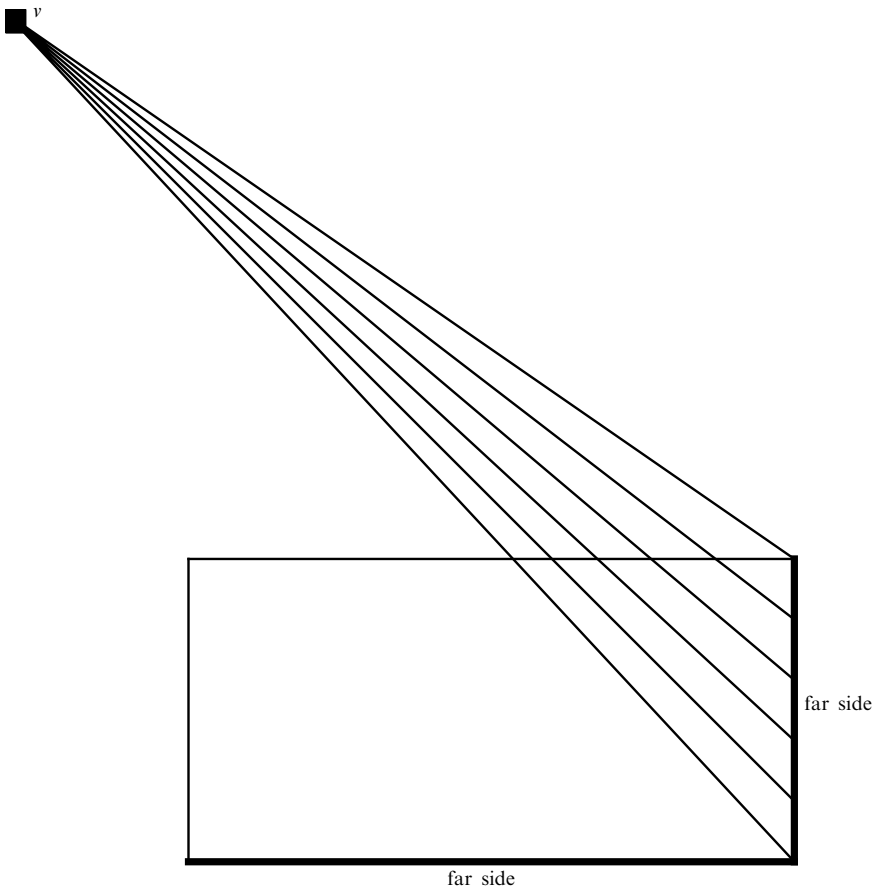


Figure 8. The far sides of a target region, and the lines of sight from the viewpoint v to one of those sides.

Fisher (1996b) showed that the line-of-sight approach can be used to compute extended viewsheds on an RSG without any computational overhead. Van Kreveld (1996) computed an extended viewshed through radial sweep. He considered a line of sight rotating around the viewpoint v. During the rotation, visibility information is computed for the cells intersected by the line. The line stops at certain events (for example, intersection of new grid cells), where information changes. The worst-case time complexity is $O(n \log n)$ for an RSG with $n^{1/2} \times n^{1/2}$ vertices.

Franklin and Ray (1994) computed approximated visibility counts on an RSG by considering a restricted number of lines of sight for each viewpoint and a restricted number of target points on each line. Another variant computes visibility counts within a distance r from the viewpoint v by testing a number of random target points uniformly and independently distributed within the circle of centre v and radius r (Franklin, 2002).

Because of their regular topology, RSGs are especially suitable for a parallel implementation on massively parallel architectures. Mills et al (1992) and Teng et al (1993) proposed parallel algorithms for computing intervisibility maps on an RSG, which directly embed the regular spatial structure of an RSG into a parallel SIMD architecture.

The method proposed by Mills et al (1992) is a parallel version of the algorithm of Shapira (1990). Every line of sight from every viewpoint of the source region is processed in parallel. An original solution is used to replace global communications (between a processor representing a line of sight and a processor representing a pixel in the grid) by much faster local communications (between processors directly representing adjacent lines of sight).

Teng et al (1993) considered only lines of sight from a point of the source region to a point on a far side of the destination region (as in Blelloch, 1990). All lines of sight corresponding to a far side are processed in parallel. The source region is divided into four sectors by drawing two lines with a 45° slope passing through the centre of gravity of the region. Each sector is swept from the centre to the boundary of the

Table 1. Summary of the algorithms reviewed.

Approach	Reference	Input DEM[a]	Output
Front-to-back	De Floriani et al (1989), [parallel version in De Floriani et al (1994)]	TIN	continuous visibility map
Front-to-back	Lee (1991)	TIN	discrete visibility map
Line-of-sight	Blelloch (1990), Shapira (1990)	RSG	discrete visibility map
Line-of-sight (parallel algorithm)	Mills et al (1992), Teng et al (1993)	RSG	intervisibility map
Line-of-sight	Sorensen and Lanter (1993)	RSG	continuous visibility map
Sector-based	Stewart (1998)	RSG or TIN (vertices only)	approximated horizon
Concentric rings (approximated algorithm)	Franklin and Ray (1994)	RSG	discrete visibility map
Line-of-sight	Fisher (1996b)	RSG	extended viewsheds
Radial sweep	Van Kreveld (1996)	RSG	extended viewsheds
Line-of-sight (parallel algorithm)	Rallings et al (1998)	RSG	visibility counts

[a] DEM, digital elevation model; RSG, regular square grid; TIN, triangulated irregular network.

source region. Coherence between pixel strips processed in consecutive stages of the sweep process is exploited in order to reduce global communications.

Rallings et al (1998) computed visibility counts for all vertices of an RSG in parallel by using a cluster of workstations. The basic approach is still the line-of-sight computation. Each processor 'knows' the entire RSG but computes the visibility only for a subset of the viewpoints. Rallings et al experimented with several criteria for partitioning the viewpoints among the processors, both static and dynamic, and compared their efficiency and load balancing.

Algorithms to compute visibility information on TINs and RSGs are summarised in table 1.

4.3 Uncertainty of visibility information

Several authors have pointed out the inadequacy of Boolean viewsheds for modeling the visibility of a terrain. Visibility information is affected by at least two sources of uncertainty. First of all, measured elevations are not error-free. Felleman and Griffin (1990) and Fisher (1995; 1996b) showed through simulation that even small errors in the data may cause relevant changes in the viewshed.

The second source of uncertainty is the interpolation method (Felleman and Griffin, 1990; Fisher, 1993; Sorensen and Lanter, 1993). This relates not only to the classification into TIN and RGS models, but also, within the TIN class, to the type of triangulation adopted, and, within the RSG class, to the convention used for modeling relief within a single square cell. This uncertainty in the RSG case regarding the convention used is not recorded in the RSG data structure (a simple matrix of elevations) and therefore is left to the inner implementation of visibility algorithms. Other implementation choices used within algorithms (such as the conventions used for the treatment of collinear points and domain boundaries) have an impact on the result. Therefore, probabilistic versions of the discrete visibility maps have been proposed in which each point is assigned a value between 0 and 1 corresponding to its probability of being visible.

Felleman and Griffin (1990) and Fisher (1995) addressed the uncertainty deriving from measurement errors in the DEM. They computed the discrete visibility map of an RGS several times, each time adding a random noise to the elevations with a standard deviation equal to the known root mean squared (RMS) error of the data, and a mean equal to zero. Then, they took the average value as the probability that a vertex is visible. Fisher (1995) also defined rules for combining probabilistic viewsheds of different viewpoints through union and intersection. In addition, Fisher (1996b), defined probabilistic extended viewsheds.

Sorensen and Lanter (1993) proposed an approach to take into account uncertainty deriving from the interpolation scheme of stepped RSG models. They gave each pixel d a probability of being visible, equal to the percentage of those cells lying in the corridor from the source pixel (containing the viewpoint) to pixel d that obscure pixel d.

5 Answering visibility queries

The visibility of a query object from a given viewpoint v can be determined by examining the lines of sight joining v to all points of the object. The simplest situation occurs when the query object is a point p. In this case, a 'brute-force' approach that searches for the intersection of segment pv with all edges in the DEM (see figure 9, over) takes $O(n)$ time on a TIN and $O(n^{1/2})$ time on an RSG with n vertices. If several queries must be solved for the same viewpoint, visibility structures, built in a preprocessing stage, usually lead to a reduction of query times.

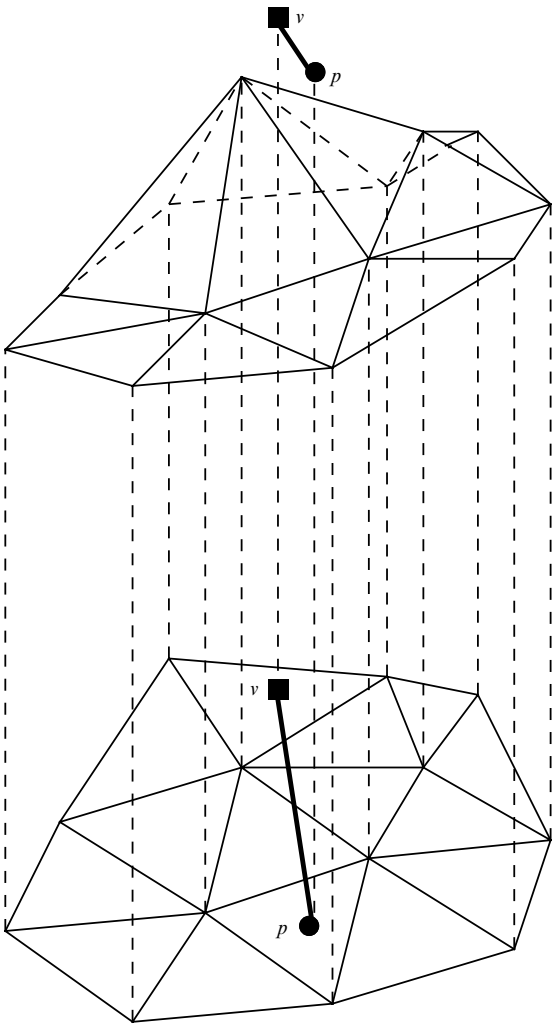


Figure 9. Finding the cells of a triangulated irregular network that intersect segment *vp*. In this example, point *p* is visible from viewpoint *v*.

Testing the visibility of a query point *p* lying on the surface reduces to locating point *p* with respect to the viewshed: *p* is visible if and only if it lies in a visible area. If the query point *p* has a nonnull height, then the same approach can be used by referring to an extended viewshed: the visibility of *p* is determined by comparing the height of *p* against the local offset at *p* (see figure 3). If the terrain is modeled as an RSG, then point location within the (extended) viewshed can be done in constant time by computing the row and column indexes of the pixel containing the query point.

In the case of a TIN, the problem of locating *p* with respect to the viewshed reduces to point location in a plane subdivision. This is true when the viewshed is encoded as a continuous visibility map (in which each triangle is divided into visible and invisible subpolygons) and also when a discrete visibility map is used (which classifies each triangle as either visible or not visible). Existing techniques for point location within a planar subdivision (see Kirkpatrick, 1983; Lee and Preparata, 1977; Preparata and Shamos, 1985; Sarnak and Tarjan, 1986) typically provide logarithmic query times.

Extended viewsheds are not generally used on TINs. Thus, if the query point p does not lie on the terrain surface, a different technique must be used in order to check its visibility. Testing whether a given point p is visible reduces to locating the projection of p in the image of the TIN on the viewplane: if the face whose image contains p lies in front of p then p is not visible. The visible image is a planar subdivision, and point location within it can be performed with the same techniques.

Cole and Sharir (1989) proposed a data structure that allows a logarithmic query time by using an amount of memory less than the $O(n^2)$ space required by a continuous visibility map or a visible image. They reduced a visibility query on a TIN to a *ray-shooting* query, that is, to the problem of determining the first face f of the terrain hit by a ray emanating from the viewpoint v and passing through the query point p. A point p is visible if an only if face f does not intersect segment vp (see figure 10). Cole and Sharir built a balanced binary tree that stores only the local horizons. The space complexity of the whole tree is only $O[n\alpha(n)\ln n]$.

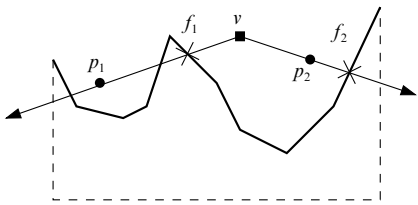


Figure 10. A two-dimensional section of a digital elevation model with two point-visiblity queries. Point p_1 is not visible from v because the face f_1, which is the first face intersected by the visual ray through p_1, intersects segment vp_1. Instead, point p_2 is visible. Note: a cross indicates the label refers to a face rather than a point.

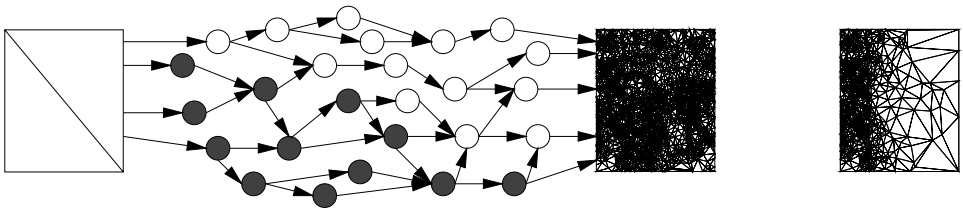
6 Visibility on multiresolution terrain models

Often, huge datasets are available for a terrain. Accurate DEMs can be built based on such data, but they entail significant storage space and long access times. As not all tasks require the same level of detail, the use of high-resolution models may affect applications for which many of the details are not relevant. Multiresolution terrain models have been developed to provide a compact representation of a terrain at different resolutions. A multiresolution model avoids redundancy of information. Also, terrain representations at any level of detail can be easily extracted from such a model.

Visibility computations are especially sensitive to errors in elevation that occur near the viewpoint, as they are amplified when constructing lines of sight. Thus, a high resolution near the viewpoint is necessary, whereas terrain portions lying far from the viewpoint can be represented with less detail. In this section, we first introduce multiresolution terrain models and then briefly review the available methods for visibility computation with such models.

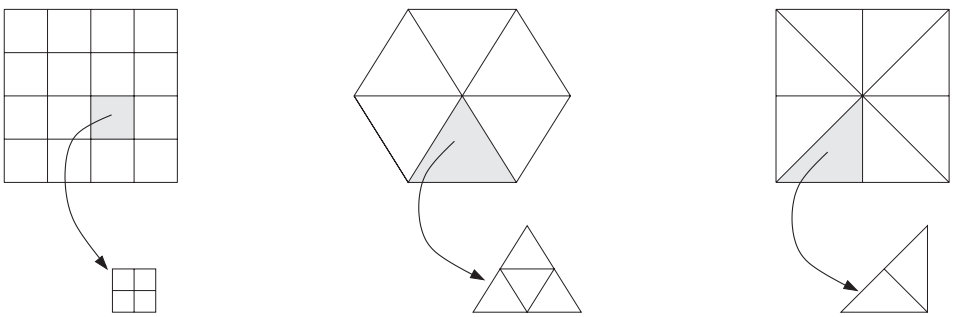
6.1 Multiresolution terrain models

A multiresolution terrain model is built from a given (huge) set of data and can be seen as a 'black box' from which it is possible to extract several DEMs at different levels of resolution. According to the general framework presented earlier (De Floriani and Magillo, 2002), a generic multiresolution model of a terrain consists of a coarse model plus a partially ordered set of modifications representing details that can be added to the initial model in order to refine it up to the maximum available resolution (see figure 11, over).



(a) (b)

Figure 11. (a) A coarse triangulated irregular network (TIN; left-hand side), a high-resolution TIN (right-hand side), and a partially ordered set of modifications forming a multiresolution terrain model between such two TINs (centre). Each modification is represented as a circle, and arrows denote dependency links. The set of grey modifications is closed with respect to the partial order. (b) The variable-resolution TIN corresponding to such a set of modifications.



(a) (b) (c)

Figure 12. Refinement rules for (a) quadtress, (b) triangles quadtrees, and (c) hierarchies of right triangles.

In models designed for gridded data (derived from RSGs), details are implicit in some fixed refinement rule. Examples are provided by quadtrees [figure 12(a), see over] and triangle quadtrees [figure 12(b)], and by hierarchies of right triangles [figure 12(c)].

In models designed for irregular data (derived from TINs), details are computed in a preprocessing step based on some operator for locally changing the resolution of a triangulation. The preprocessing stage can operate either through iterative simplification (starting from the maximum resolution and progressively eliminating details) or through iterative refinement (starting from the minimum resolution and progressively adding details). Some examples of local modification operators are:

(1) *vertex insertion*, which adds a new vertex, p, deletes a set of triangles in the neighbourhood of p, and replaces them with new triangles incident at p [see figure 13(a)];

(2) *vertex removal* (the inverse operator of vertex insertion), which deletes a vertex p together with all the triangles incident at it and creates new triangles to fill the resulting 'hole';

(3) *edge collapse*, which contracts an edge e to a vertex p, deletes the two triangles incident at e, and deforms all other triangles in the neighborhood of e [see figure 13(b)].

A dependency relation is defined among modifications based on the idea that a modification m' that deletes some triangle t cannot be performed if the modification m that has created t has not been performed (that is, m' depends on m). The dependency relation is extended to a partial order through transitive closure. In multiresolution models based on modifications that subdivide a single cell into many cells (such as the fixed-pattern modifications used for regular data), dependency implies containment between cells, and the resulting partial order embeds a tree structure. DEMs at variable

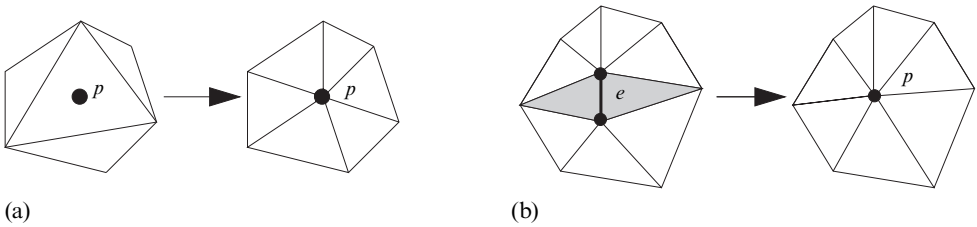


Figure 13. (a) Vertex insertion, and (b) edge collapse. Note: *e*, edge; *p*, vertex.

resolution, which can be extracted from the multiresolution model, are in one-to-one correspondence with subsets of modifications that are closed with respect to the partial order (see figure 11).

Several multiresolution models have been proposed in the literature, characterised by the data distribution they can deal with (either regular or irregular), by the local modification operator they use, and by the data structure used to encode them. A survey can be found elsewhere (De Floriani and Magillo, 2002). They all provide mechanisms to extract a specific DEM given some resolution criteria set by the user. Early multiresolution models allowed only the extraction of a DEM at a uniform level of resolution through the domain. More recent multiresolution models allow for selective refinement—that is, by locally adapting the resolution of the extracted DEM, refining it more in some interesting areas and less elsewhere (as the example shown in figure 11).

The basic algorithm for performing selective refinement starts from the initial coarse DEM and, if the resolution of some cell is insufficient, applies the modification m that refines that cell. If m depends on other modifications that have not yet been applied, then those modifications are also applied. Note that this back-tracking mechanism may cause the refinement of some cells that are already sufficiently accurate. At the end of the process, the set of all applied modifications is closed with respect to the partial order. A dynamic variant of this algorithm starts from a previously extracted DEM with the corresponding closed set of modifications, and proceeds by making modifications to refine parts of the DEM that are not accurate enough, and by undoing modifications to coarsen parts that are excessively refined. Both selective refinement algorithms are described elsewhere (De Floriani and Magillo, 2002).

6.2 Multiresolution visibility algorithms

Multiresolution terrain models do not provide an explicit terrain representation. Thus, a DEM describing the surface at a given user-defined level of resolution must be extracted to be used for visibility computations. The level of resolution may be constant (that is, a single threshold value is defined for the error over the domain) or variable (that is, the maximum error over each cell is defined according to a threshold function). In visibility applications, a resolution that decreases according to the distance from the viewpoint is especially interesting: because errors in elevations near the viewpoint are amplified in proportion to the distance, it is convenient to represent the topography more accurately near the viewpoint than on the rest of the surface (Felleman and Griffin, 1990). Once a DEM at the desired resolution has been obtained, any algorithm for visibility computation can be applied to it.

Algorithms have been presented that compute the visibility map on a hierarchical TIN by navigating the tree-like structure of the model and thus do not need the explicit construction of a TIN at the given resolution (De Floriani and Magillo, 1997). Triangles forming the DEM are extracted one by one from the multiresolution model, in front-to-back order. The key point is that resolution decreases when moving from front to back.

This guarantees that triangles that are extracted first may result in the refinement of triangles that are extracted later, but the opposite cannot happen. Thus, back-tracking to already-processed triangles is never needed. The method was originally designed for nested hierarchies of TINs, but it can be applied, with no need for change, to hierarchies of right triangles, which recently have become popular terrain representations for gridded data.

An important issue when dealing with visibility on multiresolution terrain models consists of *updating* already computed visibility structures when changing the resolution in some portion of the domain. This is necessary in applications in which the focus of attention is rapidly moving, and thus the maximum resolution is required in different areas at different times.

The visibility update problem can be solved by recomputing the visibility whenever the resolution changes, but a more efficient approach utilises dynamic algorithms. Such algorithms can update a visibility structure after the deletion of old cells and the insertion of new cells in the underlying DEM. Dynamic visibility algorithms can run in parallel to the dynamic selective refinement algorithm described above. The selective refinement algorithm manages the change in resolution of the underlying DEM, and the dynamic visibility algorithm maintains the visibility structure consistent with the current DEM at any one time. The basic operation within the selective refinement algorithm consists of modifying a current DEM by either doing or undoing a single modification among those precomputed in the multiresolution model. This implies the deletion of a small subset of cells from the DEM and replacing them with new cells. At the same time, the dynamic visibility algorithm modifies the current visibility structure by eliminating the contribution of disappearing cells and adding the contribution of new cells.

Dynamic algorithms have been proposed for horizon computation (De Floriani and Magillo, 1995) and for continuous visibility maps (Dobrindt and Yvinec, 1993; Bruzzone et al, 1995). They accept triangles in space as input—in particular, the triangles forming a TIN. The core of these algorithms is a special data structure that helps to locate the parts of the visibility map or of the horizon affected by an update. The algorithms guarantee a good expected performance for a random sequence of updates.

Finally, the 'brute-force' approach to visibility queries can be combined with a traversal of the data structure encoding a multiresolution terrain model in order to answer visibility queries efficiently at a certain level of resolution. The aim is to locate the cells at the given resolution, which may hide a query object: the visibility of the object is tested against such cells. In this context, the multiresolution structure acts as a spatial index. For instance, let us consider a query asking for the visibility of a point p from a viewpoint v. We first find the cells of the coarse DEM that intersect segment pv. Then, we apply the local modifications that refine those intersected cells that are not of a sufficient resolution. We repeat this operation until all the current intersected cells are sufficiently accurate. By storing error information together with cells (that is, by placing an upper bound on the vertical distance of each cell from the DEM at maximum resolution) we can in some cases recognize that p is not visible from v, even in the early stages of the algorithm. The major advantages in pruning the search space for the query are obtained with use of regular models that can be described by a tree data structure (De Floriani and Magillo, 1997).

7 Conclusions

We have provided a survey of visibility problems on terrains and of algorithms that have been proposed for tackling such problems. From a practical point of view, the most useful algorithms are those that have low computational complexity and that at the same time are easy to implement.

As terrain models tend to be huge in size, parallel visibility algorithms give interesting results. The most promising approaches are based on a domain partition, which can be implemented in a distributed environment such as a network of workstations.

Multiresolution terrain models can also be used to reduce the size of DEMs that are processed by visibility algorithms and thus to reduce the complexity of visibility computations. This approach is particularly interesting because an accurate representation of the surface is needed mainly in the neighbourhood of the viewpoint, whereas too many details in far areas are redundant. Thus, variable-resolution DEMs extracted 'on the fly' from a multiresolution terrain model can play a fundamental role in visibility computation and analysis.

Visibility structures are extremely sensitive to data error as well as to conventions used both in DEM construction and in implementing visibility algorithms. For this reason, a probabilistic approach to visibility, which gives a point a probability of being visible, is perhaps more suitable than the traditional formulation, in which a point is sharply classified as either visible or not visible.

Acknowledgements. This work has been supported by the project funded by the Italian Ministry of University, Education and Research [Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR)] on Representation and Processing of Spatial Data in Geographic Information Systems. This work was partially performed while Leila De Floriani was on leave at the Department of Computer Science at the University of Maryland, College Park, MD, USA.

References

Attallah M, 1983, "Dynamic computational geometry", in *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science* (IEEE Computer Society, Washington, DC) pp 92 – 99

Blelloch G E, 1990 *Vector Models for Data-parallel Computing* (MIT Press, Cambridge, MA)

Bruzzone E, De Floriani L, Magillo P, 1995, "Updating visibility information on multiresolution terrain models", in *Proceedings of the Conference on Spatial Information Theory (COSIT 95)* Eds A U Frank, W Kuhn (Springer, Berlin) pp 279 – 296

Cole R, Sharir M, 1989, "Visibility problems for polyhedral terrains" *Journal of Symbolic Computation* **17** 11 – 30

De Floriani L, Magillo P, 1995, "Horizon computation on a hierarchical terrain model" *The Visual Computer: An International Journal of Computer Graphics* **11** 134 – 149

De Floriani L, Magillo P, 1997, "Visibility computations on hierarchical triangulated terrain models" *Geoinformatica* **1**(3) 219 – 250

De Floriani L, Magillo P, 2002, "Multiresolution mesh representation: models and data structures", in *Tutorials on Multiresolution in Geometric Modelling* Eds A Iske, E Quak, M S Floater (Springer, Berlin) pp 363 – 418

De Floriani L, Falcidieno B, Nagy G, Pienovi C, 1989, "Polyhedral terrain description using visibility criteria", TR, Institute for Applied Mathematics, National Research Council, Genova

De Floriani L, Falcidieno B, Nagy G, Pienovi C, 1991, "On sorting triangles in a Delaunay tesselation" *Algorithmica* **6** 522 – 532

De Floriani L, Montani C, Scopigno R, 1994, "Parallelizing visibility computations on triangulated terrains" *International Journal of Geographical Information Systems* **8** 515 – 532

Dobrindt K, Yvinec M, 1993, "Remembering conflicts in history yields dynamic algorithms", in *Algorithms and Computation* Eds K W Ng, P Raghavan, N V Balasubramanian, F Y L Chin (Springer, Hong Kong) pp 21 – 30

Felleman J P, Griffin C, 1990, "The role of error in GIS-based viewshed determination—a problem analysis", TR EIPP-90-2, State University of New York, Institute for Environmental Policy and Planning, Albany, NY

Fisher P F, 1993, "Algorithms and implementation of uncertainty in viewshed analysis" *International Journal of Geographic Information Systems* **7** 331 – 347
Fisher P F, 1995, "An exploration of probable viewsheds in landscape planning" *Environment and Planning B: Planning and Design* **22** 527 – 546
Fisher P F, 1996a, "Extending the applicability of viewsheds in landscape planning" *Photogrammetric Engineering and Remote Sensing* **62** 1297 – 1302
Fisher P F, 1996b, "Reconsideration of the viewshed function in terrain modeling" *Geographic Systems* **3** 33 – 58
Franklin W R, 2002, "Siting observers on a terrain", in *Proceedings of the International Symposium on Spatial Data Handling 2002* Eds D Richardson, P van Oosterom (Springer, Berlin)
Franklin W R, Ray C, 1994, "Higher is not necessarily better: visibility algorithms and experiments", in *Advance in GIS Research, Proceedings of the 6th International Symposium on Spatial Data Handling* Eds T C Waugh, R G Healey (Taylor and Francis, London) pp 751 – 770
Hershberger J, 1989, "Finding the upper envelope of n line segments in $O(n \log n)$ time" *Information Processing Letters* **33** 169 – 174
Katz M J, Overmars M H, Sharir M, 1991, "Efficient hidden surface removal for objects with small union size", in *Proceedings of the 7th ACM Symposium on Computational Geometry* (ACM Press, New York) pp 31 – 40
Kirkpatrick D G, 1983, "Optimal search in planar subdivision" *SIAM Journal of Computing* **12**(1) 28 – 33
Lee J, 1991, "Analyses of visibility sites on topographic surfaces" *International Journal of Geographical Information Systems* **5** 413 – 429
Lee J, Preparata F P, 1977, "Location of a point in a planar subdivision and its applications" *SIAM Journal of Computing* **6** 594 – 606
Mills K, Fox G, Heimbach R, 1992, "Implementing an intervisibility analysis model on a parallel computing system" *Computers and Geosciences* **18** 1047 – 1054
Nagy G, 1994, "Terrain visibility" *Computer and Graphics* **18** 763 – 773
O'Sullivan D, Turner A, 2001, "Visibility graphs and landscape visibility analysis" *International Journal of Geographic Information Science* **15** 221 – 237
Preparata F P, Shamos M I, 1985 *Computational Geometry: An Introduction* (Springer, Berlin)
Preparata F P, Vitter J S, 1992, "A simplified technique for hidden-line elimination in terrains", in *STACS '92* Eds A Finkel, M Jantzen (Springer, Berlin) pp 135 – 144
Puppo E, Marzano P, 1997, "Discrete visibility problems and graph algorithms" *International Journal of Geographical Information Systems* **11** 139 – 161
Rallings P J, Ware J A, Kidner D B, 1998, "Parallel distributed viewshed analysis", in *Proceedings of ACMGIS '98* (ACM Press, New York) pp 151 – 156
Reif J, Sen S, 1988, "An efficient output-sensitive hidden surface removal algorithm and its parallelization", in *Proceedings of the 4th ACM Symposium on Computational Geometry* (ACM Press, New York) pp 193 – 200
Sarnak N, Tarjan R E, 1986, "Planar point location using persistent search trees" *Communications of the ACM* **29**(7) 669 – 679
Shapira A, 1990, "Visibility and terrain labelling", masters thesis, Department of Electrical, Computer, and System Engineering, Rensselaer Polytechnic Institute, Troy, NY
Sorensen P, Lanter D, 1993, "Two algorithms for determining partial visibility and reducing data structure induced error in viewshed analysis" *Photogrammetric Engineering and Remote Sensing* **59** 1129 – 1132
Stewart A J, 1998, "Fast horizon computation at all points of a terrain with visibility and shadow applications" *IEEE Transactions on Visualization and Computer Graphics* **4**(1) 82 – 93
Teng Y A, De Menthon D, Davis L S, 1993, "Region-to-region visibility analysis using data parallel machines" *Concurrency: Practice and Experience* **5** 379 – 406
Teng Y A, Mount D, Puppo E, Davis L S, 1995, "Parallelizing an algorithm for visibility on polyhedral terrain" *International Journal of Computational Geometry and Applications* **7**(1 – 2) 75 – 84
Van Kreveld M, 1996, "Variations on sweep algorithms: efficient computation of extended viewsheds and class intervals", in *Proceedings of the Symposium on Spatial Data Handling '96* (Eds M J Kraak, M Molenaar, Faculty of Geodetic Engineering, Delft University of Technology, Delft) pp 13A.15 – 13A.27