# Algorithms for Visibility Computation on Digital Terrain Models

†Leila De Floriani, †Paola Magillo

†University of Genova, Computer and Information Science Department

## Abstract

In the paper, we consider the problem of computing visibility information on digital terrain models. Visibility problems on polyhedral terrains are classified according to the kind of visibility information computed into point visibility, line visibility and region visibility. A survey of the state-of-the-art of the algorithms for computing visibility is presented, according to the classification introduced.

## Introduction

Describing a terrain through visibility information, such as, for instance, the portion of the terrain surface visible from a selected point of view, has a variety of applications, such as geomorphology, navigation, terrain exploration. Problems which can be solved based on visibility are, for instance, the computation of the minimum number of observation points needed to view an entire region, the computation of paths with specified visibility characteristics, the computation of optimal location for the minimum number of television transmitters or the optimal locations for receivers [3,4]. In terrain navigation problems, the profile of the horizon is an ideal tool which can be used from an observer to locate his/her position on a map.

In this paper, we consider the problem of computing visibility information on models of natural terrains. Two points on a terrain are considered mutually visible when they can be joined by a straight-line segment lying above the terrain (and intersecting it only at its two extreme points). We present a new classification of visibility problems based on the kind of visibility information computed. We consider point visibility problems, which consist of computing intervisibility between pairs of points, and produce discrete visibility models which consist of collection of points selected in a candidate set which are visible from a predefined set of observation points. Then, we examine visibility problems related to the computation of curves with specified visibility characteristics, like the horizon with respect to a point of view. Finally, we consider the problem of computing continuous visibility models, which require the computation of that portion of a terrain visible from a point of view located on the terrain (that we will call visibility model of the terrain with respect to the specified point of view).

Visibility problems considered in this paper operate on the basis of a point of view located on the terrain (which will be termed Visibility *on* a terrain) to distinguish them from problems related to the visualization of a terrain, that we will call Visibility *of* a terrain. In this latter case, the point of view lies outside the domain of the terrain (possibly at infinity) and a projection plane (called a view plane) is given. The visibility problem of a terrain is related to the problem of computing the visible surfaces in a three-dimensional scene and has been extensively studied in the literature [20,19,14,18,15,16,5]. Moreover, some algorithms are more of a theoretical interest than of practical applicability. Here, we will focus on the problem of computing visibility on a terrain.

## Digital terrain models

A natural terrain can be described as a continuous function $z = f(x, y)$, defined over a connected subset $D$ of the $x - y$ plane. Thus, a *Mathematical Terrain Model* (MTM) can be defined as a pair $\mathcal{M} \equiv (D, f)$.

A *Digital Terrain Model* (DTM) is defined as a planar subdivision $\sum$ of the domain $D$ into a collection of planar regions $\mathcal{R} = \{R_1, R_2, \ldots R_m\}$ and by a family $\mathcal{F}$ of continuous functions $z = f_i(x, y)$, $i = 1, 2, \ldots m$, each defined on $R_i$ and such that $f_i(x, y) = f_j(x, y)$, for every $(x, y) \in R_i^* \cap R_j^*$ (where $R_i^*$ denotes the closure of region $R_i$). Thus, a DTM can be expressed as a pair $\mathcal{M} \equiv (\sum, \mathcal{F})$.

According to the above definition, a DTM is a special

ACM-SAC '93/2/93/IN, USA
© 1993 ACM 0-89791-568-2/93/0002/0380...$1.50

case of mathematical terrain model. We will call *face* of a DTM the graph of each function $f_i$, *edge* and *vertex* of the DTM the restriction of each function $f_i$ to an edge and a vertex, respectively, of $\sum$. For simplicity, we will denote by $\bar{o}$ the projection on the $x-y$ plane of a generic geometric entity $o$ in the 3D space.

DTMs can be classified into *Regular Square Grids* (RSGs), and *Polyhedral Terrain Models* (PTMs). In an RSG, the domain subdivision is a regular rectangular grid, while each function $f_i$ is a quadratic function obtained by linear interpolation along the edges of the subdivision. PTMs are characterized by a domain subdivision consisting of a straight-line plane graph and by linear interpolation functions. The graph of a polyhedral terrain model consists of a network of polygonal faces. A special class of PTMs is that consisting of *Triangulated Irregular Networks* (TINs), in which the domain subdivision is a triangulation. Often, a Delaunay triangulation is used as domain subdivision for a TIN because of its good behaviour in numerical interpolation problems [7].

A Delaunay triangulation has an important property related also to visibility computations. It belongs to the class of ***acyclic subdivisions*** of the plane, i.e., subdivisions on which it is possible to define a partial order relation, called ***before/behind relation***, with respect to any point inside the domain of the subdivision [6]. A DTM based on an acyclic subdivision is called an ***acyclic digital terrain model***. Such models have a special interest related to visibility application, since their faces and edges can be radially sorted around any prefixed point of view. In such a way, visibility information can be incrementally computed for each face or edge by only taking into account the configuration of the portion of the terrain formed by those faces and edges which come before them in the order, with a considerable increase in efficiency.

## Classification of visibility problems on terrains

Given a mathematical terrain model $\mathcal{M} \equiv (D, f)$, that, for brevity, we will simply call a ***terrain***, we call a ***candidate point*** any point $P \equiv (x, y, z)$ belonging to or above the terrain, i.e., such that $(x, y) \in D$ and $z \geq f(x, y)$. Two candidate points $P_1$ and $P_2$ are ***mutually visible*** (or ***intervisible***) if, for every point $Q \equiv (x, y, z) = tP_1 + (1-t)P_2$, with $0 < t < 1$, $z > f(x, y)$.

We will call ***point of view*** (or ***observation point***) any arbitrarily chosen candidate point, and ***visual ray*** any ray emanating from a point of view. Given a point of view $V$ and a spheric coordinate system centered at $V$, a visual ray $r$ is identified by the pair $(\theta, \alpha)$, called a ***view direction***, where $\theta$ is the angle between the projection $\bar{r}$ of $r$ on the $x-y$ plane and the positive $x$-axis, and $\alpha$ is the angle between $r$ and the positive $z$-axis.

Visibility problems on a terrain can be classified based on the dimensionality of their output information into ***point***, ***line*** and ***region visibility*** problems.

### Point visibility

Given a terrain, a point of view and a finite set $Q$ of candidate points, we want to compute the subset $Q'$ of $Q$ containing all the points visible from $V$. $Q'$ is called the ***discrete visible region*** of $V$ in $Q$. In practice, we consider a DTM and the candidate points form a subset of the vertices of the terrain; often also the point of view is at a vertex.

A generalization of the problem stated above consists of computing a discrete visibility model of a terrain. Given a terrain $\mathcal{M}$, a finite set $S_1$ of observation points, and a set $S_2$ of points belonging to $\mathcal{M}$, the ***discrete visibility model*** of $\mathcal{M}$ with respect to $S_1$ and $S_2$ is a collection of sets $\{Q_i \mid P_i \in S_1\}$, where $Q_i = \{P \in S_2 \mid P \text{ is visible from } V_i\}$.

### Line visibility

The well-known visible-line reconstruction problem for general 3D scenes can be formulated as a line visibility problem for a terrain, by taking a point of view inside the domain, and requiring the computation of the visible portions of the edges of the DTM. A line visibility problem of practical relevance in geographic applications consists of computing the horizon of an observation point on a terrain.

Given a terrain $\mathcal{M}$ and a point of view $V$, the ***horizon*** of the terrain with respect to $V$ is a function $\alpha = h(\theta)$, defined for $\theta \in [0, 2\pi]$ such that, for every radial direction $\theta$, $h(\theta)$ is the maximum value $\alpha$ such that each ray emanating from $V$ with direction $(\theta, \beta)$, with $\beta < \alpha$, does not intersect the terrain.

In a polyhedral terrain the horizon is a radially sorted list of labeled intervals $[\theta_1, \theta_2]$. If an interval $[\theta_1, \theta_2]$ has label $s$, then the visual ray defined by a direction $(\theta, h(\theta))$ with $\theta_1 < \theta < \theta_2$ hits the terrain at a point belonging to edge $s$. An example of horizon on a polyhedral terrain is represented in figure 1.

### Region visibility

Given a terrain $\mathcal{M} \equiv (D, f)$ and a point of view $V$, we define ***visible region*** of $V$ the subset $D'$ of $D$ formed by the points $(x, y) \in D$ such that $(x, y, f(x, y))$ is visible from $V$. The ***invisible region*** of $V$ is clearly the complement of $D'$ in $D$. An example of visible and invisible region is shown in figure 2.

In a DTM $\mathcal{M} \equiv (\sum, \mathcal{F})$, visibility with respect to a point of view can be coded as a map, called ***continuous visibility map***, which consists of a partition of the domain $D$ into maximal connected regions, each of which is labeled as invisible, if it is a subset of the invisible region, or with a face of the terrain in such a way that, if a region

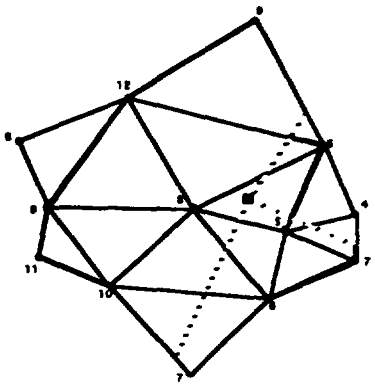

Figure 1: Horizon of an observation point on a polyhedral terrain, projected on the $x-y$ plane. The location of the view-point is the marked point.

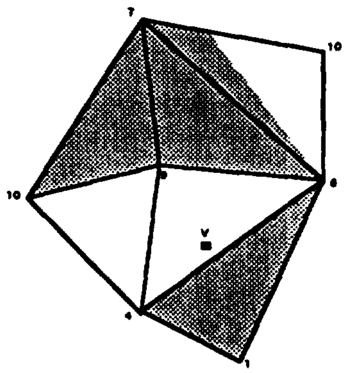

Figure 2: An example of visible and invisible region on a TIN (the invisible region is shaded).

$R$ is labeled with face $f$, then all points of $f$ whose vertical projection belongs to $R$ are visible from the point of view.

Given a set $S$ of observation points, we call the collection of visibility maps of the points in $S$ as the *continuous visibility model* of the terrain with respect to $S$.

## Algorithms for point visibility

Point visibility computations can be reduced to determining the mutual visibility of two candidate points. In this section, we present a "brute-force" approach as well as an approach based on a preprocessing of the terrain which builds a hierarchical query structure.

Given a DTM $\mathcal{M} \equiv (\sum, \mathcal{F})$ and two candidate points $P_1$ and $P_2$, the "brute-force" approach reduces to computing the intersection of the projection on the $x-y$ plane of segment $s \equiv \overline{P_1P_2}$, denoted $\bar{s}$, with the edges of $\sum$. At each intersection point $\bar{P}$ between $\bar{s}$ and an edge $e$ of $\sum$, we test whether $s$ lies above or below the edge of $\mathcal{M}$ corresponding to $e$. This process has a linear time complexity, in the worst case, in the number $n$ of vertices of $\mathcal{M}$. For a regular grid, the time complexity reduces to $O(\sqrt{n})$. To compute the discrete visibility region with respect to a single point of view $V$, we need to determine the mutual visibility of $V$ and the points belonging to a candidate set $S_2$.

The second approach preprocesses the terrain model with respect to the point of view $V$ and builds a data structure on which the problem of computing the visibility of a point $P$ from $V$ can be solved in logarithmic time. The data structure has been proposed by Cole and Sharir [4] to solve a ray shooting problem on a polyhedral terrain.

Given a DTM $\mathcal{M} \equiv (\sum, \mathcal{F})$, a point of view $V$ and a view direction $(\theta, \alpha)$, the *ray shooting* problem consists of determining the first face of $\mathcal{M}$ hit by a ray emanating from $V$ with direction $(\theta, \alpha)$.

The mutual visibility of two points $V$ and $P$ reduces to solve the ray shooting problem, since we have just to consider as view direction that defined by segment $\overline{VP}$ and, when obtained the first face of $\mathcal{M}$ hit by the corresponding visual ray, we have just to determine whether $P$ and $V$ lie on the same or on opposite sides of the plane of such a face.

The data structure of Cole and Sharir, that we call *horizon tree*, has size $O(n\alpha(n)\log n)$, where $\alpha(n)$ is the inverse of Ackermann's function, and thus almost-constant; ray shooting queries can be answered in time $O(\log^2 n)$ on such structure. Given a point of view $V$, the horizon tree can be built for any acyclic polyhedral terrain, since it needs to sort the edges of the terrain around the viewpoint.

A horizon tree is a balanced binary tree in which the whole set of edges of the terrain is associated with the root. If $S$ is the set of edges associated with a node $v$, then the closest half of the edges of $S$ are associated with the left child of $v$, the other half with its right child. The partial horizon computed on the edges associated with the left child of a node $v$ is attached to $v$. A horizon tree can be computed in optimal $O(n\alpha(n)\log n)$ time.

A ray shooting query, represented by a visual ray $r$, can be answered by descending the horizon tree $T$ until two consecutive horizons are found such that the visual ray $r$ passes above the first one, but not above the second. This identifies two edges $l_1$ and $l_2$ of the DTM such that $r$ passes above $l_1$, but below $l_2$. It can be proved that $l_1$ and $l_2$ bound the same face $f$, and that $f$ is the first face encountered by $r$. In descending $T$ one node is visited at each level. For each node $v$ the interval of the horizon associated with $v$ containing ray $r$ must be located. This leads to a time complexity of $O(log^2 n)$, which reduces to $O(logn)$ by using a fractional cascading technique.

Computing the discrete visible region of a model $\mathcal{M}$ with respect to a point of view $V$ and a set $S_2$ of $k$ candidate points is thus equivalent to computing the horizon tree, with a computational cost equal to $O(n\alpha(n)\log n)$, plus $k$ intervisibility computations with a cost of $O(klog^2 n)$, which is clearly less than the direct intervisibility computation, if $k = O(n)$.

## Horizon computation algorithms

The horizon computation problem reduces to the computation of the upper envelope of a set of possibly intersecting segments in the plane.

Given $p$ segments in the plane, i.e., $p$ linear functions $y = f_i(x)$, $i = 1 \dots p$, each defined on an interval $[a_i, b_i]$, the *upper envelope* of such segments is a function $y = F(x)$, defined over the union of the intervals $[a_i, b_i]$, and such that $F(x) = max_{i|x \in [a_i, b_i]}(f_i(x))$.

We express the edges of the terrain in a spherical coordinate system centered at the point of view and consider only the two angular coordinates. This transformation produces a set of segments in the $\rho - \theta$ plane. By computing the upper envelope of such segments, we obtain a function which associates with each direction $\theta$ the segment (if it exists) having maximum azimuth in direction $\rho$, i.e, the horizon (according to the given definition). It has been shown [10] that the complexity of the upper envelope of $p$ segments in the plane is $O(p\alpha(p))$, and thus the complexity of the horizon of a polyhedral terrain with $n$ vertices is equal to $O(n\alpha(n))$. The upper envelope of $p$ segments in the plane can be computed either by a dynamic, or by a divide-and-conquer approach.

The dynamic approach computes the upper envelope, denoted $Env$, starting from an initially empty structure and adding a segment at a time to it, in any order. For each new segment $s$, we locate those $x$-intervals, related to $Env$, which properly intersect the $x$-interval defined by $s$, and update the upper envelope in each of these intervals. The worst-case time complexity of such algorithm is equal to $O(p^2)$.

The first divide-and-conquer algorithm for computing the upper envelope of a set of segments in the plane is due to Atallah [1] and achieves a worst-case time complexity of $O(p\alpha(p)\log p)$. The algorithm recursively partitions the set of segments in two halves, and pairwise merges the results. The merging step of two upper envelopes is performed by using a sweep-line technique for intersecting two monotone chains of segments.

The algorithm proposed by Hershelberg [13] is still based on a divide-and-conquer strategy, but computes the upper envelope of a set of $p$ segments in optimal $O(p\log p)$ time. The basic idea is to subdivide the given set of segments in such a way that the upper envelope of any subset of the generated groups is linear in the number $m$ of segments, and, thus, it can be computed in time $O(m\log m)$, by using the algorithm previously discussed. Since such subdivision generates $O(\log n)$ subsets, Atallah's algorithm is used to pairwise merge all partial envelopes according to a divide-and-conquer approach. This operation again has a $O(p\log p)$ time complexity, thus leading to $O(p\log p)$ time complexity for the entire algorithm in the worst case.

## Algorithms for region visibility

In this Section, we present first a brief survey of algorithms for computing visibility *of* a terrain, i.e., for hidden surface removal on polyhedral terrains. Then, we describe in more detail specific algorithms for computing a continuous visibility model, i.e., algorithms for visibility *on* a terrain.

### Hidden surface removal on a polyhedral terrain

The algorithms for hidden surface elimination in a 3D polyhedral scene (possibly a polyhedral terrain) give the visible portions of each object of the scene projected on the view plane, that form a *visible image*. The worst case space complexity of the visible image is $O(n^2)$ for a polyhedral scene with $n$ vertices, but in practical cases its size may be much smaller. For this reason, output sensitive algorithms are more convenient than worst case optimal algorithms. In the following, we review some output sensitive algorithms for hidden surface elimination in polyhedral terrains, which can be adapted to the computation of a continuous visibility model.

Reif and Sen [18] propose an output sensitive algorithm for hidden line elimination on acyclic polyhedral terrains, whose time complexity is equal to $O((n+d)\log^2 n)$. The point of view is located at infinity in $y$ direction, and the

$x - z$ plane is used as view plane. The algorithm consists of two basic steps. In the first step, the edges of the terrain are grouped to form $O(n)$ monotone chains with respect to the $y$ axis. In the second step, the chains are projected on the view plane one at a time, in increasing distance order. The upper envelope *Env* of the chains projected so far is maintained, representing the current horizon. When a chain $\sigma$ is projected, the intersections between $\sigma$ and *Env* are found. Then the portions of $\sigma$, that lie above the current horizon *Env*, as visible from the viewpoint, are marked. Those portions of faces adjacent to the visible portions of the edges of $\sigma$, and lying on the nearer side to the observation point are also visible. The update operation consists of the computation of the new horizon. A structure is used to store *Env* for representing monotone polygons. This makes possible to compute the intersections and the subsequent updatings in $O((m+k)\log m)$ time, where $m$ is the number of edges and $k$ the number of intersections.

The same approach is followed by Preparata and Vitter [17], who, however, use a different data structure for storing *Env*. The structure proposed by Preparata and Vitter is easier to implement, and consists of a subtree of the balanced binary tree whose leaves are the vertices of the terrain. The subtree is obtained by collecting only those leaves which correspond to vertices in *Env*. The structure allows intersections and updatings in $O((m+k)\log m)$ time, where $m$ is the number of edges and $k$ the number of intersections, thus leading to an $O((n+d)\log^2 n)$ algorithm.

De Berg, Halperin, Overmars, Snoeyink and Van Kreveld [5] propose an output sensitive algorithm which works in $O(n^{1+\epsilon}\sqrt{d})$ time, where $d$ is the output size, and $\epsilon$ is any arbitrary integer value. The algorithm computes the visible image of a generic 3D scene composed of quasi-disjoint triangular faces by using a sweepline technique (in particular, it can be applied to a TIN). A distance order is not required. The observation point is located at $z = -\infty$.

The approach proposed by Katz, Overmars and Sharir [12] is more a "schema" of an algorithm, which can be applied to three-dimensional scenes formed by quasi-disjoint objects sorted by increasing distance from a point of view (which can be in any position with respect to the scene: inside, outside or at infinity). In particular, the schema can be applied to an acyclic polyhedral terrain model, with observation point located on it. A description of this algorithm is presented in the subsection dedicated to continuous visibility algorithms.

## Computing the lower envelope of a set of triangles

An approach to the computation of the visibility model of a point of view on a digital terrain model consists of transforming the problem of computing the visibility model into the problem of computing the lower envelope of a set of disjoint triangles in the space.

The *lower envelope* of a set $T$ of triangles in space defines a partition of the $x - y$ plane into maximal connected regions, each of which is labeled with a triangle of $T$ in such a way that, if a region $R$ is labeled with triangle $t$, then $t$ is the triangle with minimum height over $R$.

It has been shown [10] that such a partition has an $O(n^2)$ complexity (since all the triangles are disjoint). When computing the visibility model with respect to a point of view $V$, we associate as "height" to a triangle $t_i$ a function $f_i(x,y)$ that maps into a point $\bar{P} \equiv (x,y)$ in the plane the distance from $V$ of the point $P$ on $t_i$ whose projection on the plane is $\bar{P}$. The same approach could be used for hidden surface removal on a terrain, provided that the projection plane is the view plane. The algorithms, which compute the upper envelope *Env*($T$), compute a first finer subdivision, denoted $A(T)$, obtained by projecting all the triangles on the $x - y$ plane and extending the projected segments to infinity. $A(T)$ has still an $O(n^2)$ complexity and is formed only by convex regions.

A "brute force" approach to determine $A(T)$ consists of computing for each region $R$, the triangle whose "height" is minimum with respect to the values of all the "height" functions associated with the triangles overlapping $R$. The worst case time complexity would be $O(n^3)$, since the projection of $n$ triangles can cover a single region.

An optimal approach to computing the lower envelope of a set of triangles has been proposed by Edelsbrunner, Guibas and Sharir [9]. Such algorithm is based on the classical divide-and-conquer paradigm, since it recursively partitions set $T$ in two equally-sized subsets $T_1$ and $T_2$, separately builds $A(T_1)$ and $A(T_2)$, and, finally, merges $A(T_1)$ and $A(T_2)$ into $A(T)$ by reassigning the labels in the regions at which $A(T_1)$ and $A(T_2)$ intersect. The worst case time complexity of the algorithm is equal to $O(n^2)$; *Env*($T$) can be computed from $A(T)$ in $O(n^2)$ time.

An incremental randomized algorithm has been proposed by Boissonat and Dobrindt [2]. The algorithm inserts a triangle at a time in the lower envelope constructed for the already examined triangles. Instead of the current lower envelope *Env*($T$), a finer subdivision, called the *trapezoidal decomposition* of *Env*($T$), and denoted *Trap*($T$), is maintained. In *Env*($T$) we can distinguish two types of vertices: projected triangle vertices and intersection points between projected triangle edges. *Trap*($T$) is obtained from *Env*($T$) by drawing through every vertex of the first type a vertical segment to the first edge above and below this point, and repeating the same process for every vertex of the second type, but only in one direction (see figure 3). *Trap*($T$) has the

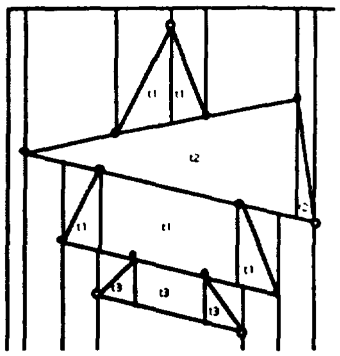
Figure 3: Trapezoidal decomposition of the lower envelope of a set of three triangles.

same space complexity than $Env(T)$, but it is formed by trapezoidal regions, i.e., regions with fixed number of edges.

$Trap(T)$ is stored in a special data structure which allows an easy localization of the regions which are properly intersected by the projection of the new inserted triangle on the $x-y$ plane. The data structure maintains the current trapezoidal decomposition plus the "history" of its construction in a directed acyclic graph whose nodes correspond to trapezoids that have been regions of $Trap(T)$ at some stage of the algorithm, and whose leaves are the trapezoids of the current $Trap(T)$. At each step the newly created trapezoids become children of those trapezoids, leaves at the previous step, which they properly intersect. A randomized analysis shows that, for a set of non-intersecting triangles, the $i$-th triangle can be inserted in $O(i)$ randomized expected time, leading to a $O(n^2 \log n)$ complexity for the the entire algorithm.

## An algorithm based on acyclic TINs

In the following, we briefly describe an algorithms for the computation of the visibility model on acyclic TINs (like those based on a Delaunay triangulation of the domain) [6,11]. Given an acyclic TIN $\mathcal{M} \equiv (\sum, \mathcal{F})$ and a point of view $V$ on $\mathcal{M}$, the algorithms operates in two steps:

1) sorting the triangles of $\sum$ by increasing distance with respect to the projection $\bar{V}$ of the point of view $V$ on the $x-y$ plane (*radial sorting phase*).
2) computing the visible portions of each triangle of $\mathcal{M}$ with respect to $V$ (*visibility computation phase*).

Radial sorting is performed by building a star-shaped polygon around $\bar{V}$ by incrementally adding a triangle at a time to a starting polygon formed by the triangles of $\sum$ incident in $\bar{V}$. The acyclicity property of $\sum$ ensures that at each step we can always add a new triangle while maintaining the star shape of the resulting polygon.
The visibility computation step consists of incrementally computing successive horizons, each horizon being restricted to the set of edges of the terrain examined so far.

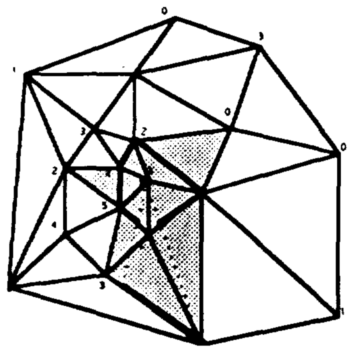
Figure 4: Configuration of ABESS and of the already examined triangles in an intermediate step at the algorithm (the viewpoint is the marked vertex).

A face $f_i$ of $\mathcal{M}$, defined by a linear function $z = a_i x + b_i y + c_i$, determines two (open) half-spaces: an upper half-space (locus of the points $(x, y, z)$ such that $z > a_i x + b_i y + c_i$) and a lower half-space (locus of the points $(x, y, z)$ such that $z < a_i x + b_i y + c_i$). A face $f_i$ is said to be *face down* with respect to $V$ if $V$ lies in its lower half-space , is said *face up* if $V$ lies in the upper half-space. An edge $e$ of $\mathcal{M}$ is a *blocking edge* when, if $f_1$ denotes the closest face incident in $e$ and $f_2$ the furthest, then $f_1$ is face up and $f_2$ is face down.

The visibility algorithm constructs an active sequence of blocking edges, called *Active Blocking Edge Segment Sequence* (ABESS). ABESS contains all those blocking edges portions belonging to triangles already visited which can cast a shadow on triangles not yet examined (see figure 4).

At the beginning of the algorithm, ABESS is initialized as an empty list of intervals. To compute the visibility of the current triangle $t$, we have just to compute the shadows cast on $t$ by the vertical trapezoids hang at segments of ABESS. If $t$ is face down with respect to $V$, then $t$ is totally invisible. Otherwise, we consider all segments in ABESS intersected by any visual ray which hits $t$. We compute the portion of $t$ hidden by each segment and insert it into the set of the invisible portions of $t$. When $t$ has been examined, ABESS is updated by considering the edges of $t$ never examined before. For every edge $b$ of $t$ which has not yet been examined, and is a blocking edge, we update ABESS with a similar procedure to the incremental updating of the upper envelope.

Radial sorting the triangles of $\sum$ requires $O(n)$ time. The visibility computation step has an $O(n^2\alpha(n))$ worst-case time complexity, since $O(n\alpha(n))$ is the size of ABESS in the worst case. Implementation issues and experiments on real data are discussed in [11].

## A general algorithm for sortable scenes

We describe now an approach to the computation of the visibility model on a polyhedral terrain that consists of the application of the already mentioned "schema", proposed by Katz, Overmars and Sharir [12], to the specific case of an acyclic polyhedral terrain with point of view located on the terrain.

The processing of a scene composed of $n$ objects is performed as follows:

1) The $n$ objects are sorted by distance from the viewpoint.
2) The sorted objects are stored in a balanced tree $T$ with $\log n$ levels, by associating the $n$ objects to the root, the $n/2$ closest objects to its left child, the $n/2$ farthest object to its right child, and so on.
3) For each node $v$ of $T$, the union $\mathcal{U}(v)$ of the projections on the view plane plane of the objects associated with $v$ is computed. The computation is performed by ascending $T$ from the leaves to the root, since $\mathcal{U}$ can be easily obtained by merging the regions corresponding to the two children of $v$.
4) For each node $v$, the visible portion $\mathcal{V}(v)$ of $\mathcal{U}(v)$ is computed; $\mathcal{V}(v)$ is the portion of $\mathcal{U}(v)$ not hidden by objects closest to the viewpoint than those associated with $v$. The objects, which can possibly hide portions of $\mathcal{U}(v)$, are those associated with the left children of the nodes on the path from the root to $v$. This computation proceeds by descending $T$ from the root to the leaves, and each region $\mathcal{V}(v)$ is obtained by considering $\mathcal{U}(v)$ and the region $\mathcal{V}$ of its parent.
5) The regions $\mathcal{V}(v)$ associated with the leaves are "collected" to find the visible portion of each object in the scene.

It can be shown that the global size of all regions $\mathcal{U}$, stored at any level of $T$, is equal to $O(U(n))$ and, similarly, the total size of all regions $\mathcal{V}$ stored at any level of $T$ is less or equal to $d$ (= size of the final result). The space complexity of $T$ is $O((U(n) + d)\log(n))$, and, for a polyhedral terrain, $O((n\alpha(n) + d)\log(n))$, since in this case $U(m) = m\alpha(m)$. The storage cost can be reduced to $O(U(n)\log(n))$ by computing regions $\mathcal{V}$ with a sweepline technique [12].

The worst case time complexity of the algorithm for acyclic polyhedral terrains can be evaluated as follows. The time complexity is dominated by the cost of steps 3 and 4, which involve boolean operations among polygonal regions. For a polyhedral terrain, such regions are monotone polygons with respect to any radial direction from $\bar{V}$. Thus, the boolean operations on such regions can be performed in linear time in the output size, i.e., $O(n\alpha(n)\log n)$ for step 3, and $O(d\log n)$ for step 4. The complexity of the whole algorithm is $O((n\alpha(n) + d)\log(n))$, where $d$ is the output size.

| HORIZON ALGORITHMS | | |
|---|---|---|
| ALGORITHM | AUTHORS | TIME COMPLEXITY |
| Incremental | | $O(n^2)$ |
| Divide-and-conquer | Atallah | $O(n\alpha(n)\log n)$ |
| Optimal, divide-and-conquer | Hershelberg | $O(n\log n)$ |

| HIDDEN SURFACE ALGORITHMS | | |
|---|---|---|
| AUTHORS | TIME COMPLEXITY | TERRAIN |
| DeFloriani, Falcidieno, Nagy, Pienovi | $O(n^2\alpha(n))$ | Acyclic triangulated Terrain |
| Reif, Sen | $O((n+d)\log^2 n)$ | Acyclic |
| Preparata, Vitter | $O((n+d)\log^2 n)$ | Acyclic |
| DeBerg, Halperin, Overmars, Snoeyink, VanKreveld | $O(n^{1+\epsilon}\sqrt{d})$ | Triangulated |

| TRIANGLE ENVELOPE ALGORITHMS | | |
|---|---|---|
| ALGORITHM | AUTHORS | TIME COMPLEXITY |
| Brute force | | $O(n^3)$ |
| Incremental, randomized | Boissonat, Dobrindt | $O(n^2\log n)$ |
| Optimal, divide-and-conquer | Edelsbrunner, Guibas, Sharir | $O(n^2)$ |

| REGION VISIBILITY ALGORITHMS | | |
|---|---|---|
| AUTHORS | TIME COMPLEXITY | TERRAIN |
| DeFloriani, Falcidieno, Nagy, Pienovi | $O(n^2\alpha(n))$ | Acyclic triangulated |
| Katz, Overmars, Sharir | $O((n\alpha(n) + d)\log(n))$ | Acyclic |

Tabel 1

Comparative table of the algorithms for terrain visibility presented in the paper.

## Concluding Remarks

We have presented an overview of visibility problems on terrains, and a survey of algorithms for solving such problems. We have classified visibility computations in point, line and region visibility, and we have followed such classification in the description of the algorithms which compute them. Table 1 describe algorithms for horizon computation, algorithms for visibility of a terrain, algorithms for computing the lower envelope of triangles and region visibility algorithms.

An open research problem consists of computing visibility information on hierarchical terrain models. Such models have encountered a lot of interest in the GIS community because of their capability of representing a surface at levels of different specification. It will be important to develop algorithms for point, line and region visibility on such a model [7].

## References

[1] M.Atallah, Dynamic Computational Geometry, *Proceedings 24th Symposium on Foundations of Computer Science*, 1989, pp.92-99.

[2] J.D.Boissonnat, K.Dobrindt, On-Line Construction of the Upper Envelope of Triangles in $\Re^3$, *Acts of 4th Canadian Conference on Computational Geometry*, August 1992.

[3] M.Cazzanti, L.De Floriani, E.Puppo, G.Nagy, Visibility Computation on a Triangulated Terrain, *Proceedings 8th International Conference on Image Analysis and Processing*, Como, September 1991.

[4] R.Cole, M.Sharir, Visibility Problems for Polyhedral Terrains, *Technical Report 32, Courant Institute*, New York University, 1986.

[5] M.De Berg, D.Halperin, M.Overmars, J.Snoeyink, M.van Kreveld, Efficient Ray Shooting and Hidden Surface Removal, *Proceedings 7th ACM Symposium on Computational Geometry*, 1991, pp.21-30.

[6] L.De Floriani, B.Falcidieno, G.Nagy, C.Pienovi, Polyhedral Terrain Description Using Visibility Criteria, *Institute for Applied Mathematics, National Research Council, Technical Report n.17*, October 1989.

[7] L.De Floriani, E.Puppo, A hierarchical triangle-based model for terrain description, *Proceedings International Conference on GIS*, Pisa, 21-23 September 1992.

[8] F.Devai, Quadratic Bounds for Hidden Line Elimination, *Proceedings 2nd ACM Symposium on Computational Geometry*, 1986, pp.269-275.

[9] H.Edelsbrunner, L.J.Guibas, M.Sharir, The Upper Envelope of Piecewise Linear Functions: Algorithms and applications, *Discrete and Computational Geometry*, 4, 1989, pp.311-336.

[10] H.Edelsbrunner, The Upper Envelope of Piecewise Linear Functions: Tight bounds on the number of faces, *Discrete and Computational Geometry*, 4, 1989, pp.337-343.

[11] D.M.Jung, Comparisons of Algorithms for Terrain Visibility, *Master Thesis, Rensselaer Polytechnic Institute*, Troy, New York, August 1989.

[12] M.J.Katz, M.H.Overmars, M.Sharir, Efficient Hidden Surface Removal for Objects with Small Union Size, *Proceedings 7th ACM Symposium on Computational Geometry*, 1991, pp.31-40.

[13] J.Hershelberg, Finding the Upper Envelope of $n$ Line Segments in $O(n \log n)$ time, *Information Processing Letters*, 33, 1989, pp.169-174.

[14] M.McKenna, Worst Case Optimal Hidden Surface Removal, *ACM Trans. on Graphics*, 1987, n.6, pp.19-28.

[15] K.Mulmuley, An Efficient Algorithm for Hidden Surface Removal, *Computer Graphics*, Vol. 23, n.3, July 1989, pp.379-388.

[16] M.Overmars, M.Sharir, Output-Sensitive Hidden Surface Removal, *Proceedings 30th IEEE Symposium on Foundations of Computer Science*, 1989, pp.598-603.

[17] F.P.Preparata, J.S.Vitter, A Simplified Technique for Hidden-Line Elimination in Terrains, *Proceedings STACS'92*, Paris, February 1992.

[18] J.H.Reif, S.Sen, An Efficient Output-Sensitive Hidden-Surface Removal Algorithm And Its Parallelization, *Proceedings 4th ACM Symposium on Computational Geometry*, 1988, pp. 193-200.

[19] A.Schmitt, Time and Space Bounds for Hidden Line and Hidden Surface Algorithms, *Proceedings Eurographics*, 81, 1981, pp.43-56.

[20] K.Weiler, P.Atherton, Hidden Surface Removal Using Polygonal Area Sorting, *Proceedings SIGGRAPH '77*, July 1977.