# Region inter-visibility in terrains

Marc van Kreveld Esther Moet René van Oostrum

*Institute for Information & Computing Sciences*
*Utrecht University*
*P.O. Box 80.089*
*3508 TB Utrecht*
*The Netherlands*
*{marc,esther,rene}@cs.uu.nl*

**Abstract**

A polyhedral terrain is the image of a piecewise linear continuous function defined over the triangles of a triangulation in the $xy$- plane. Given a terrain with $n$ vertices, two simply-connected regions (subsets of the triangles), and any constant $\epsilon > 0$, we can determine in $O(n^{2+\epsilon})$ time and storage whether or not the two regions are completely inter-visible, which improves the $O(n^3)$ time complexity of a brute-force algorithm.

*Key words:* Terrains, visibility, GIS, data structures

## 1. Introduction

A terrain $T$ is a triangulated polyhedral surface with $n$ vertices $V(T) = \{v_1, v_2, ..., v_n\}$. Each vertex $v_i$ is specified by three real numbers $(x_i, y_i, z_i)$, which are its cartesian coordinates. Every vertical line intersects the terrain at most once, which means it can also be viewed as a piecewise linear function on $\mathbb{R}^2$.

Every triangle $t$ on a terrain $T$ has a normal vector associated with it. The terrain divides space into two parts; the outward normal on a triangle corresponds to the part above $T$, $above(T)$ and the inward normal to the part below $T$, $below(T)$. Because of the natural meaning, visibility between two points only makes sense if both points are on or above the terrain.

Given a terrain $T$ in 3D and two points $p$ and $q$ on $T$, we say that $p$ sees $q$ if the line segment $pq$ does not intersect $below(T)$, that is, visibility is not blocked by grazing contact with the terrain. A triangle $t$ in $T$ is *weakly visible* from a point $p \in \mathbb{R}^3$ if there exists at least one point on $t$ that is visible from $p$. A triangle $t$ is *strongly visible* from $p$ if every point on $t$ is visible from $p$.

A *region* is a connected subset of triangles in $T$. We develop an algorithm to decide whether two given regions $R_1$ and $R_2$ are completely *inter-visible*, which is equivalent to region $R_1$ being strongly visible from every point in region $R_2$ (if so, it automatically holds the other way around).

## 2. Related Work

Visibility computations in terrains have their main application in geographic information systems (GIS), for example computations regarding horizon pollution and signal transmission (e.g. mobile phone networks). Most of the early research considering visibility in terrains dealt with grid-based *Digital Elevation Models* (DEM), as opposed to *Triangular Irregular Networks* (TIN) which are common in recent GIS research. In algorithms research, TINs are called (polyhedral) terrains.

Algorithmic explorations of visibility in terrains were described in [4] and [10–12], for example the shortest watchtower problem. Later, more complex visibility problems were discussed: visibility computations from a moving point of view are discussed in [2], and an efficient and dynamic algorithm to maintain a visibility map for a certain viewpoint is introduced in [8].

Agarwal and Sharir [1] obtained tight bounds on the maximum number of combinatorially different views of a terrain. Finally, [6] is a recent overview of various problems concerning visbility in terrains.

## 3. Strong region inter-visibility

First, we look at the orientations of $R_1$ and $R_2$ in space. For the time being, we suppose no other part of $T$ can block visibility between $R_1$ and $R_2$ than the regions themselves.

We denote the plane in 3D that contains a given triangle $t$ as $\mathcal{P}(t)$. The half-space induced by $\mathcal{P}(t)$ that corresponds to the space above $\mathcal{P}(t)$ is denoted by $HS(t)$. Visibility is only defined above the terrain, so if a triangle $t$ sees a point $p$, then $p$ lies in $HS(t)$. If two connected sets of triangles in 3D, $R_1$ and $R_2$, can see each other, it implies that all vertices of $R_1$ lie in the intersection of the half-spaces induced by the triangles of $R_2$, and vice versa:

$$\forall v_1 \in R_1 : v_1 \in \bigcap_{t_2 \in R_2} HS(t_2) \wedge \\ \forall v_2 \in R_2 : v_2 \in \bigcap_{t_1 \in R_1} HS(t_1) \qquad (1)$$

If a vertex from one region does not lie in the half-space intersection of the other region, it is not seen by at least one of the triangles in that region. Thus, condition (1) is necessary (but not sufficient) for strong visibility between two regions in a terrain. We say regions $R_1$ and $R_2$ are *facing each other* if condition (1) is satisfied.

When two regions $R_1$ and $R_2$ on a terrain $T$ are facing each other, we can start to take the rest of the terrain into account. Visibility between points from different regions now depends on the terrain not blocking the view.

We limit the number of points from $R_1$ and $R_2$ we have to check for inter-visibility.

**Lemma 1** *Two connected sets of triangles on a terrain $R_1$ and $R_2$ are strongly inter-visibile, if and only if*

*(i) $R_1$ and $R_2$ are facing each other, and*

*(ii) $\partial R_1$ and $\partial R_2$ are strongly inter-visible*

**PROOF.** The necessity of the first condition follows from the discussion above.

The necessity of the second condition follows easily. Because a region $R = int(R) \cup \partial R$, the boundaries of two regions must see each other if the entire regions see each other.

For the sufficiency of the two conditions, assume that $p \in int(R_1)$ and $q \in int(R_2)$ do not see each other. We assume that $R_1$ and $R_2$ are facing each other, and prove that there exist two points on $\partial R_1$ and $\partial R_2$ that cannot see each other.

Consider the vertical plane $\mu$ containing $p$ and $q$. Let $T_\mu = \mu \cap T$ be the cross-section of the terrain, which is a lower-dimensional terrain itself. The set $\mu \cap R_1$ consists of one or more connected components, and $p$ lies in the interior of one of them. The same statement holds for $R_2$ and $q$. We only have to consider visibility of $p$ and $q$ in $\mu$. Because $p$ and $q$ are facing each other, the line segment $\overline{pq}$ does not intersect $below(T_\mu)$ in a neighborhood of $p$, nor in a neighborhood of $q$. Let $r$ be the point on $T_\mu$ closest to $p$ that is in the closure of $\overline{pq} \cap below(T_\mu)$. If $p$ and $r$ are in the same component of $\mu \cap R_1$, then the triangle containing $r$ is not facing the triangle containing $q$. If $p$ and $r$ are not in the same component, then between $p$ and $r$ in $T_\mu$ there is a point $p' \in \partial R_1$ than cannot see $q$ either. We repeat the argument with $q$ and $p'$ to find a point $q' \in \partial R_2$ that cannot see $p'$. Hence, if two points interior to $R_1$ and $R_2$ cannot see each other, then there exist two boundary points of $R_1$ and $R_2$ that cannot see each other. $\square$

Checking only strong inter-visibility of the vertices on the boundary of the two regions is not sufficient, because a small peak of the terrain can block two boundary edges from being strongly inter-visible, while their endpoints can indeed see each other.

We define $S$ to be the set containing all triangles that are defined by either a vertex of $\partial R_1$ and an edge of $\partial R_2$, or by an edge of $\partial R_1$ and a vertex of $\partial R_2$. Because grazing contact with the terrain is permitted, inter-visibility is blocked if and only if there is a triangle $t \in S$ for which $t \cap below(T) \neq \emptyset$. The proof of the following lemma is straightforward and is therefore omitted.

**Lemma 2** *Given a terrain $T$ and an arbitrary triangle $t$ with vertices in $V(T)$. The intersection $t \cap below(T)$ is non-empty if and only if one of the following two situations occurs:*

*(i) a vertex of $T$ lies strictly above $t$, or*

*(ii) an edge of $T$ lies strictly above an edge of $t$.*

## 4. Algorithm and data structures

Now that we know what to compute, how can we compute it efficiently? The terrain has $O(n)$ vertices and triangles. The regions $R_1$ and $R_2$ are generally smaller, so we say they have $O(m)$ complexity. In the remainder of this paper we use $O^*(f(n,m))$ as a shorthand for a bound $O(f(n,m) \cdot n^\epsilon)$, where $\epsilon > 0$ is an arbitrarily small constant. The $O^*(..)$-notation supresses polylogarithmic factors of $n$, and factors $n^\epsilon$.

### 4.1. *Facing the correct way*

For each region $R_i$, $i = 1,2$, we compute $\bigcap_{t \in R_i} HS(t)$, which is the intersection of $O(m)$ half-spaces. Because half-spaces are convex, this intersection can be computed in $O(m \log m)$ time [5]. Next, we check for all $O(m)$ vertices of the other region whether they are contained in this intersection. We preprocess the convex volume for point location in $O(m \log m)$ time and then query with $O(m)$ points in $O(\log m)$ time per query. Concluding, we can check if two $O(m)$ regions on the terrain are facing each other in $O(m \log m)$ time using $O(m)$ storage.

### 4.2. *Visibility blocked by the terrain*

We have to check for intersections between a set of $O(n)$ terrain triangles, and the elements of $S$, a set of $O(m^2)$ triangles between the edges and vertices of $\partial R_1$ and $\partial R_2$. We check all triangles in $S$ for intersection with the terrain. In a brute-force algorithm, this leads to $O(nm^2)$ time. But can we do better?

By Lemma 2, we have to check two things to decide whether a triangle $t$ from $S$ intersects $\mathcal{T}$: either a vertex of $T$ lies above $t$, or an edge of $T$ lies above an edge of $t$. We discuss these two situations next.

#### 4.2.1. *Terrain vertices*

To reduce the time complexity, it is necessary to take a different look at the geometry. For a given vertex $v$ of the terrain, we want to check whether there is any triangle in $S$ that lies strictly below $v$. Obviously, we want to limit the number of triangles to check in height against $v$.

If $v$ lies above a triangle $t$ in $S$, then the projection of $v$ on the $xy$-plane lies in the projection of $t$. $S$ contains $O(m^2)$ triangles that possibly overlap each other; we want to retrieve exactly those triangles $S(v) \subseteq S$ that contain the projection of $v$ in their projections. Then the remaining question is: does $v$ lie on or below all triangles of $S(v)$?

This question can also be simplified by looking at the geometry. If a point $p$ in 3D lies above a set of triangles $S(v)$, it lies in the polyhedron $\bigcup_{t \in S(v)} HS(t)$. These geometric observations lead us to the data structure described next.

To find the triangles from $S$ that contain a given point in their projections on the $xy$-plane, we construct a partition tree of $O(m^2)$ size [5]. Given a query point $p$, it returns $O^*(m)$ canonical subsets of triangles from $S$ that contain $p$ in their projection. We give every node $\mu$ in the partition tree an associated data structure of linear complexity in the cardinality of the canonical subset $c(\mu)$. This structure is used to determine whether the query point lies in the intersection of the half-spaces induced by all triangles in $c(\mu)$. This query can be answered in $O(\log m)$ time after $O(m \log m)$ preprocessing time and with linear storage, by using the Dobkin-Kirkpatrick hierarchy [7,9].

The construction time for a partition tree of size $O(m^2)$ is $O(m^{2+\epsilon})$ for any constant $\epsilon > 0$ [5]. Fortunately, the space required for the partition tree's associated structure does not increase the total storage space much, in particular, nothing at all in $O^*(..)$-notation. The total data structure requires $O(m^2 \log m)$ space. The construction time is $O^*(m^2)$ and the total query time is $n$ times $O^*(m)$, which is $O^*(mn)$ [5].

#### 4.2.2. *Terrain edges*

The second situation that must be tested to discover whether the terrain blocks visibility is if a terrain edge lies above an edge of a triangle from $S$. We project the terrain edges onto the $xy$-plane and for every edge from $S$, we want to find the terrain edges whose projections intersect the projection of the query edge. We can treat these terrain edges as full lines in 3D, and the same is true for the query edge. The objective is to find out whether the line supporting the query edge lies above all lines supporting the selected terrain edges.

The data structure we use stores the projections of the terrain edges in a cutting tree of size $O(n^2)$

[5]. We perform $m^2$ queries on this tree with edges from $S$, each taking $O(\log n)$ query time. The query returns all intersecting terrain edges in $O(\log n)$ canonical subsets. We give each node $\mu$ in the cutting tree an associated structure of size $O(n^{2+\epsilon})$ for the canonical subset $c(\mu)$ of edges as the supporting lines in space [3]. Consequently, we can decide whether a query line lies above all lines in the canonical subset in $O(\log n)$ query time.

Because the storage required for the cutting tree dominates the storage of the associated structure, the total data structure requires $O(n^{2+\epsilon})$ storage and can be constructed in $O^*(n^2)$ time. The total query time is $O(m^2 \log n)$, or $O^*(m^2)$, and thus the total time complexity to determine if any terrain edge is above any edge of $S$ is $O^*(n^2 + m^2)$.

### 4.3. *Total time and space complexity*

In the previous section, we investigated the running times and storage space needed to determine whether two regions in a terrain completely see each other. We now summarize the time and storage complexities for the partition tree with its associated structures (PT) and the cutting tree with its associated structures (CT).

| | Preprocessing | Total queries | Storage |
|---|---|---|---|
| PT | $O^*(m^2)$ | $O^*(mn)$ | $O(m^2)$ |
| CT | $O^*(n^2)$ | $O^*(m^2)$ | $O^*(n^2)$ |

Because $m$ is in the worst case $O(n)$, we have a total time complexity of $O^*(n^2)$ which is a considerable improvement over the $O(n^3)$ time of a brute-force algorithm.

## 5. Concluding remarks

We developed an algorithm to determine in $O^*(n^2)$ time whether two regions in a terrain are strongly inter-visible. The algorithm uses as data structures a partition tree and a cutting tree, both with associated structures, leading to a total of $O^*(n^2)$ storage.

Another natural problem would be determining weak inter-visibility between two regions, which means that every point in one region sees at least one point in the other region. Because weak visibility has less constraints than strong visibility, it is more difficult to compute. We are currently working on algorithms and data structures for this problem.

## References

[1] P. K. Agarwal and M. Sharir. On the number of views of polyhedral terrains. In *Proc. 5th Canad. Conf. Comput. Geom.*, pages 55–60, 1993.

[2] M. Bern, D. Dobkin, D. Eppstein, and R. Grossman. Visibility with a moving point of view. *Algorithmica*, 11:360–378, 1994.

[3] B. Chazelle, H. Edelsbrunner, L. J. Guibas, M. Sharir, and J. Stolfi. Lines in space: Combinatorics and algorithms. *Algorithmica*, 15:428–447, 1996.

[4] R. Cole and M. Sharir. Visibility problems for polyhedral terrains. *J. Symbolic Comput.*, 7:11–30, 1989.

[5] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, Germany, 2nd edition, 2000.

[6] L. De Floriani and P. Magillo. Intervisibility on terrains. In P.A.Longley, M.F.Goodchild, D.J.Maguire, and D.W.Rhind, editors, *Geographic Information Systems: Principles, Techniques, Managament and Applications*, chapter 38, pages 543–556. John Wiley & Sons, 1999.

[7] D. P. Dobkin and D. G. Kirkpatrick. Determining the separation of preprocessed polyhedra – a unified approach. In *Proc. 17th Internat. Colloq. Automata Lang. Program.*, volume 443 of *Lecture Notes Comput. Sci.*, pages 400–413. Springer-Verlag, 1990.

[8] P. Magillo and L. De Floriani. Computing visibility maps on hierarchical terrain models. In *Proc. 2nd ACM Workshop on Advances in GIS*, 1994.

[9] D. Mount. Geometric intersection. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 33, pages 615–632. CRC Press LLC, Boca Raton, FL, 1997.

[10] G. Nagy. Terrain visibility. Technical Report, Rensselaer Polytech. Inst., Troy, NY, 1982.

[11] M. Sharir. The shortest watchtower and related problems for polyhedral terrains. *Inform. Process. Lett.*, 29(5):265–270, 1988.

[12] B. Zhu. Improved algorithms for computing the shortest watchtower of polyhedral terrains. In *Proc. 4th Canad. Conf. Comput. Geom.*, pages 286–291, 1992.