Approximating the Visible Region of a Point on a Terrain *

Boaz Ben-Moshe Paz Carmi Matthew J. Katz

Department of Computer Science
Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel
{benmoshe,carmip,matya}@cs.bgu.ac.il

Abstract

Given a terrain T and a point p on it, we wish to compute the region that is visible from p. We present a generic radar-like algorithm for computing an approximation of this region. The algorithm *extrapolates* the visible region between two consecutive rays (from p) whenever the rays are *close enough*; that is, whenever the difference between the sets of visible segments along the rays is below some threshold. Thus the density of the sampling by rays is sensitive to the shape of the visible region. We suggest a specific way to measure the resemblance (difference) and to extrapolate the visible region between two consecutive rays. We report on preliminary experimental results.

1 Introduction

Let T be a triangulation representing a terrain (i.e., there is a height (z-coordinate) associated with each triangle vertex). We are interested in the following well known problem. Given a point p on (or above) T, compute the region R_p of T that is visible from p. A point q on T is visible from p if and only if the line segment $\overline{pq}$ lies above T (in the weak sense). Thus R_p consists of all points on T that are visible from p. The problem of computing the visible region of a point arises as a subproblem in numerous applications (see, e.g., [7, 9, 11]), and, as such, has been studied extensively [2, 3, 4, 5, 7]. For example, the coverage area of an antenna for which line of sight is required may be determined by clipping the region that is visible from the tip of the antenna with an appropriate disk centered at the antenna.

Since the combinatorial complexity of R_p might be $\Omega(n^2)$ [3, 6], where n is the number of triangles in T, it is desirable to also have fast approximation algorithms, i.e., algorithms that compute an approximation of R_p. Moreover, a good approximation of the visible region is often sufficient, especially when the triangulation itself is only a rough approximation of the underlying terrain. Note that in this paper we are assuming that the terrain representation (i.e., the triangulation T) is fixed and cannot be modified. Simplifying the triangulation can of course lead to faster running times of any algorithm for computing the visible region. This approach was studied in a previous paper [1]. See, e.g., [8] for more information on terrain simplification.

We present a generic radar-like algorithm for computing an approximation of R_p. The algorithm computes the visible segments along two rays ρ_1, ρ_2 emanating from p, where the angle between the rays is not too big. It then has to decide whether the two sets of visible segments (one per ray) are *close enough* so that it can *extrapolate* the visible region of p within the wedge defined by ρ_1 and ρ_2, or whether an intermediate ray is needed. In the latter case the algorithm will now consider the smaller wedge defined by ρ_1 and the intermediate ray. Thus a nice property of the algorithm is that the density of the sample rays varies and depends on the shape of R_p.

*Research by Ben-Moshe and Katz is partially supported by grant no. 2000160 from the U.S.-Israel Binational Science Foundation, and by the MAGNET program of the Israel Ministry of Industry and Trade (LSRT consortium). Research by Carmi is partially supported by a Kreitman Foundation doctoral fellowship.

In order to use this generic algorithm one must provide (i) a measure of resemblance for two sets of visible segments, where each set consists of the visible segments along some ray from p, and (ii) an algorithm to extrapolate the visible region between two rays whose corresponding sets were found similar enough. In Section 2 we describe in more detail the generic algorithm and provide the missing ingredients.

In Section 3 we suggest a natural way to measure the error associated with an approximation of R_p. Using this error measure, we compare between our algorithm and the corresponding fixed-angle version with the same number of sample rays. According to these experiments our algorithm is much better in situations of "under sampling," where the number of sample rays is small. We are currently comparing our algorithm (and several variants of it) with other (known) algorithms for approximating the visible region. A report on these experiments will be included in the full version of this paper.

2 The Algorithm

In this section we first present our radar-like generic algorithm. Next we describe the measure of resemblance and the extrapolation algorithm that we devised, and that are needed in order to transform the generic algorithm into an actual algorithm.

The generic algorithm is presented in the frame below. The basic operation that is used is the cross-section operation, denoted $\mathit{cross\text{-}section}(T, p, \theta)$, which computes the visible segments along the ray emanating from p and forming an angle θ with the positive x-axis. Roughly speaking, the generic algorithm sweeps the terrain T counter clockwise with a ray ρ emanating from p, performing the cross-section operation whenever the pattern of visible segments on ρ is about to change significantly with respect to the pattern that was found by the previous call to cross-section. The algorithm then extrapolates, for each pair of consecutive patterns, the visible region of p within the wedge defined by the corresponding locations of ρ.

Given a triangulation T representing a terrain (i.e., with heights associated with the triangle vertices) and a view point p on or above T:
$\theta \leftarrow 0$.
$\alpha \leftarrow$ some constant angle, say, $\pi/45$.
$S_1 \leftarrow \mathit{cross\text{-}section}(T, p, \theta)$.
$S_2 \leftarrow \mathit{cross\text{-}section}(T, p, \theta + \alpha)$.
while $(\theta < 360)$
    if (S_1 is **close enough** to S_2)
        extrapolate(S_1, S_2);
        $\theta \leftarrow S_2.\mathit{angle}$;
        $S_1 \leftarrow S_2$;
        $S_2 \leftarrow \mathit{cross\text{-}section}(T, p, \mathit{min}(\theta + \alpha, 360))$;
    else
        $\mu \leftarrow (S_1.\mathit{angle} + S_2.\mathit{angle})/2$;
        $S_2 \leftarrow \mathit{cross\text{-}section}(T, p, \mu)$;

In order to obtain an actual algorithm we must provide precise definitions of ***close enough*** and ***extrapolate***.

Close enough: A threshold function that checks whether two patterns S_1, S_2 are similar, where each of the patterns corresponds to the set of visible segments on some ray from p. There are of course many ways to define ***close enough***. We chose the following definition. In practice, the rotating ray is actually a rotating segment of an appropriate length. Let l denote this length. We refer to l as the *range of sight*. Now rotate the ray containing S_2 clockwise until it coincides with the ray containing S_1. See Figure 1 (a). Next compute the length of the **XOR** of S_1 and S_2, that is, the total length covered by only one of the sets S_1, S_2. This length is then divided by l. Denote by v the value that was computed, and let δ be the angle between S_1 and S_2. If $\delta \cdot v \leq C$, where C is some constant, then return TRUE else return FALSE. The role of δ in the above formula is to force ***close enough*** to return TRUE when the angle between the rays is small, even if the patterns that are being compared

differ significantly.

Extrapolate: Given two patterns S_1, S_2 which are ***close enough***, we need to compute an approximation of the portion of the visible region of p that is contained in the corresponding wedge. We do this as follows. Consider Figure 1 (b). For each 'event point' (i.e., start or end point of a visible segment) on one of the two horizontal rays, draw a vertical segment that connects it with the corresponding point on the other ray. For each rectangle that is obtained color it as follows, where grey means visible and black means invisible. If the horizontal edges of a rectangle are either both visible from p or both invisible from p, then, if both are visible, color it grey, else color it black. If, however, one of the horizontal edges is visible and the other is invisible, divide the rectangle into four pieces by drawing the two diagonals. The color of the upper and lower pieces is determined by the color of the upper and lower edges, respectively, and the color of the left and right pieces is determined by the color of the rectangles on the left and on the right, respectively.

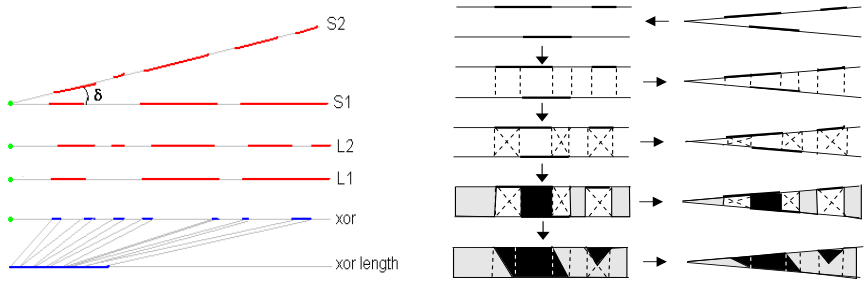


Figure 1: Grey marks visible and black marks invisible. (a) The ***close enough*** threshold function: δ times the relative length of the **XOR** of S_1 and S_2. (b) The***extrapolate*** function.

3 Experimental Results

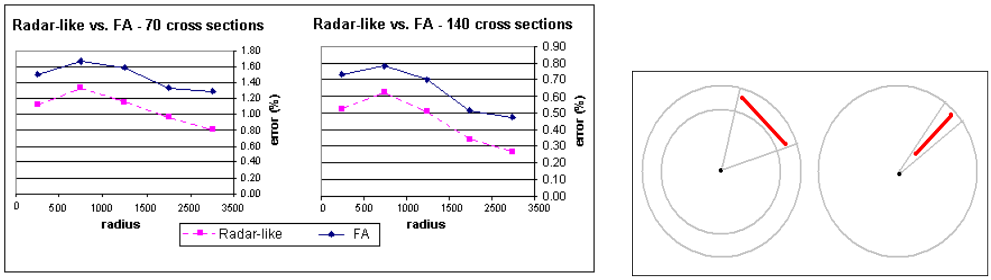
In our experiments we use the following natural error measure. Let R'_p be an approximation of R_p obtained by some approximation algorithm, where R_p is the region visible from p. Then the error associated with R'_p is the area of the **XOR** of R'_p and R_p, divided by the area of the disk of radius l, where l is the range of sight that is in use. See Figure 2.

Figure 2: Left: the exact region R_p; Middle: the approximate region R'_p computed by our algorithm; Right: $\mathbf{XOR}(R'_p, R_p)$.

We compared between our radar-like algorithm and the corresponding fixed-angle version. That is, we ran our algorithm, with several values of α, on a collection of terrains, view points, and ranges of sight. For each application of our algorithm, we also ran the fixed-angle version with angle $360/n$, where n is the number of sample rays (i.e., cross-section operations) that were used in this application. We then computed the errors for the two regions that were obtained. Figure 3(a) shows some typical results. From these results it is clear that our

algorithm is better is situations of "under sampling," where the number of sample rays is small. Both algorithms and the error computation were implemented in Java. (The error computation is actually a grid-based approximation of the error measure defined above.)

We are currently comparing our algorithm (and several variants of it) with other (known) algorithms for approximating the visible region, including the z-buffer algorithm [10] and an algorithm (see Figure 3(b)), that is in some sense orthogonal to our algorithm, that uses circles of increasing radii instead of cross-sections [5]. A report on these experiments will be included in the full version of this paper.



(a) (b)

Figure 3: (a) Our algorithm is more accurate than the corresponding fixed-angle version. (b) Circles of increasing radii approach vs. Radar-like approach.

Acknowledgment. The authors wish to thank Ofir Ganani and Maor Mishkin who helped implementing the radar-like algorithm.

References

[1] B. BenMoshe, M.J. Katz, J.S.B. Mitchell and Y. Nir. Visibility preserving terrain simplification. *Proc. 18th ACM Sympos. Comput. Geom.* 303–311, 2002.

[2] D. Cohen-Or and A. Shaked. Visibility and dead-zones in digital terrain maps. *Computer Graphics Forum* 14(3):171–179, 1995.

[3] R. Cole and M. Sharir. Visibility problems for polyhedral terrains. *Journal of Symbolic Computation* 7:11–30, 1989.

[4] L. De Floriani and P. Magillo. Visibility algorithms on triangulated digital terrain model. *International Journal of GIS* 8(1):13–41, 1994.

[5] L. De Floriani and P. Magillo. Representing the visibility structure of a polyhedral terrain through a horizon map. *International Journal of GIS* 10:541–562, 1996.

[6] F. Devai. Quadratic bounds for hidden line elimination. *Proc. 2nd ACM Sympos. Comput. Geom.* 269–275, 1986.

[7] R. Franklin, C.K. Ray and S. Mehta. Geometric algorithms for siting of air defense missile batteries. *Technical Report Contract No. DAAL03-86-D-0001*, 1994.

[8] P. S. Heckbert and M. Garland. Fast polygonal approximation of terrains and height fields. Report CMU-CS-95-181, Carnegie Mellon University, 1995.

[9] M.F. Goodchild and J. Lee. Coverage problems and visibility regions on topographic surfaces. *Annals of Operation Research* 18:175–186, 1989.

[10] N. Greene, M. Kass and G. Miller. Hierarchical z-buffer visibility. *Computer Graphics Proc. Annu. Conference Series* 273–278, 1993.

[11] A.J. Stewart. Fast horizon computation at all points of a terrain with visibility and shading applications. *IEEE Trans. Visualizat. Compt. Graph* 4(1):82–93, 1998.