AN OVERVIEW OF VISIBILITY PROBLEM ALGORITHMS IN 1,5D

B. Kaučič[1]

B. Žalik[2]

[1]Department of mathematics, Faculty of Education
Koroška c. 160, SI-2000 Maribor,

[2]Faculty of Electrical Engineering and Computer Science
Smetanova 17, SI-2000 Maribor

Slovenia
{branko.kaucic|zalik}@uni-mb.si

ABSTRACT

The paper gives an overview of algorithms for the terrain visibility problem. First, a comprehensive background of the problem is given. It is explained how the 2,5D problem is transformed to a 1,5D problem. Next, six algorithms (a naive approach, an approach with the height of line-of-sight (LOS), an approach with the biggest slope of LOS, an approach with the cross product, an incremental approach, and an improved incremental approach) are briefly explained and their theoretical time complexities are given. After that, run-times of the algorithms are measured for different terrain configurations and different viewpoint heights. The best algorithm is selected at the end.

Keywords: visibility problem, digital terrain model, GIS, overview

1 INTRODUCTION

Visibility analysis is a fundamental problem of many applications, which concern the geographical information systems (GIS), computer graphics, computer games, navigation, and engineering applications. Visibility problems deal with the computation of visibility information from a viewpoint, which can lie outside or inside the terrain domain, or use the visibility information to solve optimization problems. The most used visibility computations compute a viewshed and/or find a horizon. Examples of visibility optimization problems are determining the minimum number of terrain guards, calculating the minimum path with specific visibility characteristics (e.g., scenic paths and hidden paths). Applications include the location of radar facilities, observation towers, radio, TV or telephone transmitters, path planning, navigation and orientation. For an extensive survey on this subject see [Cole89, DeFlo99a, DeFlo99b].

At such applications, two contradictory problems occur:

- The quality of the terrain representation depends on the resolution of sampled terrain data. Better resolution leads to more accurate terrain analysis, which is desired by the users.
- Better resolution demands considerable amount of memory space for storing terrain data and the computational time increases significantly. As stated by Franklin, the naive (traditional) methods and algorithms require more computational capacity that is available now and will be in the near future [Frank94a, Frank94b].

Because of that, the choice of an appropriate algorithm for the visibility problem is in practice of great importance. In this paper, an extensive overview of algorithms for visibility problems in 1,5D is given. The visibility analysis is basically a 2,5D problem, but as it is shown in the paper, the analysis is actually done in 1,5D. The paper starts with a brief theoretical background. To the knowledge of the authors, all known 1,5D algorithms are presented and analyzed in the third Section. All the algorithms have been implemented and in the fourth Section, they are compared and evaluated using several terrain configurations. In the last, the fifth Section, the paper is summarized.

2 PRELIMINARIES

Terrain data are related with a 3D configuration of the Earth's surface. The geometry of a terrain is modeled as a 2,5D surface, i.e. a surface in 3D space described by a bivariate function. A model representing a terrain relief using a finite number of samples is known as a Digital Terrain Model (DTM). As its representation is extremely suitable for computer-based analysis and manipulation, it is also very frequently used for the visibility problem.

DTM is usually defined by a set of points $p \equiv (x_p, y_p, z_p)$. Two points p_1 and p_2 are mutually visible (intervisible) iff every point $q \equiv (x, y, z) = p_1 + t(p_2 - p_1)$, $0 < t < 1$, lies above the corresponding point p_q of the terrain, i.e. $z > z_q$ [DeFlo99a]. In other words, two points are mutually visible when straight line p_1p_2 lies above the terrain and touches it only at its end points. If one of these two points is a special point called viewpoint (labeled with O in the continuation), the ray from O through the second point is called a line-of-sight (LOS). If LOS Op_2 is not obstructed (with the terrain or with the objects on the terrain), point p_2 is visible.

The definition of the visibility can be further extended to the points called targets. The targets are not parts of the terrain and lie above it. An example of the visibility of terrain points and targets can be seen in Fig. 1. We have a terrain,

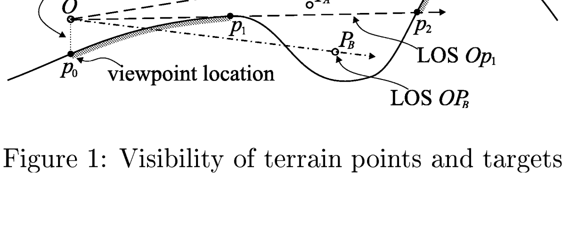

Figure 1: Visibility of terrain points and targets

viewpoint O and three targets: P_A, P_B, and P_C. From the viewpoint, all point sets of the terrain shaded with thick gray lines are visible. From the point p_1 to the point p_2 all terrain points below LOS Op_1 are invisible. Similarly, target P_B is not visible because LOS through P_B does not lie above the terrain. As seen, the point p_1 in Fig. 1 represents the highest obstacle at determining visibility of the points on its right side.

Fig. 2 considers the visibility problem in 2,5D. Let

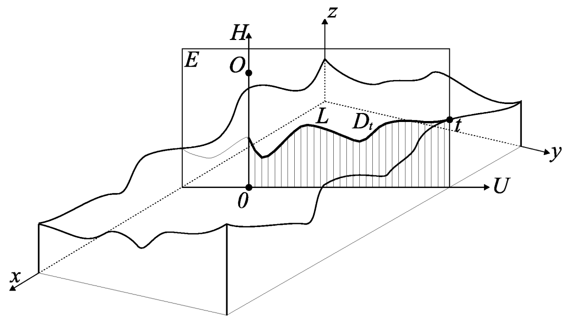

Figure 2: The visibility problem in 2,5D, translated to the visibility problem in 1,5D

E be the plane passing through the viewpoint O being perpendicular to the plane xy. Using E, a 1,5D coordinate system with axis H, U is defined. The origin of the HU coordinate system coincides with the point of viewpoint's vertical projection on the plane xy. An intersection between E and the surface of a terrain is a curve L, which visibility it is investigated. By rotating plane E around axe H for a small discrete steps, a set of discrete curves is obtained, approximating a terrain of interest. In this way, the 2,5D visibility problem is transformed to the set of 1,5D visibility problems, i.e., for each curve L. Curve L is usually represented as a discrete profile D_t (see Fig. 2). Similar discrete profiles (but parallel) are used in the more general visualization [Skala87].

3 THE ALGORITHMS

In the continuation, short descriptions of all known algorithms for the 1,5D visibility problem are given:

a) **A naive approach** (*Naive*): In this approach, only the simplest fact about the slope of the LOS is used. Namely, the point from the viewpoint is visible, if the slope of the LOS to that point is bigger than the slopes of LOSs to all previous points. Therefore, the algorithm calculates the slope of current point's LOS and checks it with the slopes of LOSs of all previous points. If a bigger slope of the LOS has been encountered the point is not visible, otherwise it is visible.

Because all previous points are checked, the algorithm exposes $O(n^2)$ time complexity. The algorithm is the slowest with the monotone increasing heights of the terrain, where all points are visible. The best case represents a terrain where only the first two points are visible and all others are invisible.

b) **An approach with the height of LOS** (*LOSHeight*): The process runs by walking on each LOS. For every point, its LOS is computed first. By walking on that LOS, the height of the LOS is compared at every step with the height of the corresponding point of the terrain. If the height of the LOS is lower then the height of the terrain, the point is visible, otherwise it is invisible. This approach has been used by Shapira and mentioned in [DeFlo94].

Because the heights of all LOSs above all previous points are checked, the time complexity remains $O(n^2)$. It is only for a constant factor faster than the previous algorithm. The worst and the best case of the algorithm performance remain the same, too.

c) **An approach with the biggest slope of LOSs** (*BiggestSlope*): The approach is an improvement of the above mentioned approaches by taking into account the knowledge about the previous point [Cohen95, DeFlo99a]. For each point, the slope of its LOS is calculated. If the slope of the current LOS is bigger then the biggest slope so far, the point is visible and the new biggest slope of all LOSs is updated. Otherwise, the point is invisible. This approach can be found in [Trobe98] and at Blelloch [DeFlo94] who called LOSs the tangents.

For each point only one computation of the slope of the LOS and one comparison is needed that results in $O(n)$ time complexity. The worst and the best case of the algorithm performance remain the same as in the naive approach.

d) **An approach with the cross product** (*CrossProduct*): The LOS to p_i can also be imagined as a vector $p_i - O$ [Cohen95]. Suppose that point p_i is visible. To determine the visibility of point p_{i+1}, vector $p_{i+1} - p_i$ is observed whether it points to the right or to the left halfplane of vector $p_i - O$. Point p_{i+1} is visible when it points to the left halfplane, otherwise it is invisible (Fig. 3).

The visibility of point p_i is determined as follows.

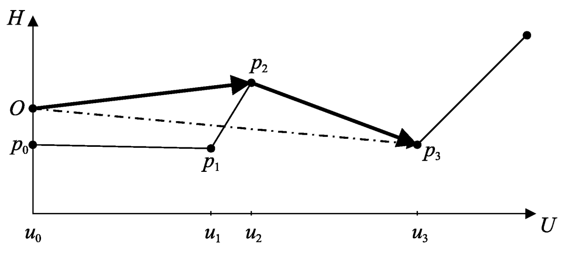

Figure 3: Determining of the visibility with the cross product

Let p_v represent the last computed visible point before p_i and let S be the value of the cross product's z component:

$$(p_v - O) \times (p_i - O) = u_v(h_i - h_O) - u_i(h_v - h_O) \quad (1)$$

If $S > 0$, point p_i is visible and a new point p_v is set.

The visibility of all points is computed in one pass over all points. Because the cross product and the comparison is done in $O(1)$, the time complexity of the algorithm remains $O(n)$. The worst and the best case of the algorithm performance stay the same as in the naive approach.

e) **An incremental approach:** (*Incremental*): The approach with the cross product can be improved with an incremental computing if the constant distance between adjacent points on axis U, $u_{i+1} - u_i = 1$, is supposed [Cohen95].

Two cases are possible. Firstly, point p_{i-1} is recognized as a visible. Then, the visibility of point p_i is computed using equation:

$$\begin{aligned}(p_{i-1} - v_O) \times (p_i - v_O) = \; & u_{i-1}(h_i - h_{i-1}) - \\ & (h_{i-1} - h_O)\end{aligned} \quad (2)$$

In the second case, point p_{i-1} is identified as invisible. If the last visible point has been p_v, $v < i - 1$, the value of the cross product for the next point p_i can be determined by adding the difference to the previous cross product, calculated for the point p_{i-1}. Let ω_{i-1} represent the value of the cross product determined at the visibility of point p_{i-1}. Then ω_i can be computed as:

$$\omega_i = \omega_{i-1} + u_v(h_i - h_{i-1}) - (h_v - h_O) \quad (3)$$

To determine the visibility of every point, $O(1)$ time is needed, resulting in an overall time complexity of $O(n)$. The worst case occurs when the visibility of points is alternating, while the best case is obtained when all points of the terrain are visible.

f) **An improved incremental approach** (*Incremental+*): In the previous approach, when the invisible point has been encountered, all further points are invisible as long as their heights remain under the LOS of the last visible point [Cohen95, Trobe98]. Fig. 4 shows such case. Suppose that point p_i is visible, and point p_{i+1}

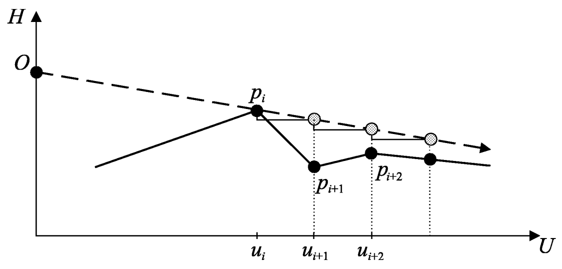

Figure 4: Determining the visibility in the invisible zone

is invisible. When point p_{i+1} is identified as invisible, the slope of the LOS to p_i (for example in variable los) is calculated. Because the length of the step over the axis U is one, the height of los above point u_{i+1} equals $h_{ray} = h_i + los$. Therefore, for every next point, starting with p_{i+2} only the value los is added to the temporary height of the ray and compared with the height of the terrain point.

The visibility of each point is determined similarly as in the previous algorithm. The determination of the visibility in invisible zone is improved, but before that, the slope of the LOS has to be calculated. The time complexity stays $O(n)$, only for a constant factor faster. The worst and the best case of the algorithm performance stay the same as in the incremental approach.

The first two algorithms represent the early ideas of how to determine the visibility by the computer using DTM. Other mentioned algorithms have been developed after careful study of the problem, and they reflect discovered properties. The last two algorithms work only, if all terrain points are spaced equally. This fact can be successfully used in regular square grids that seem to be increasingly used in GIS applications.

4 TESTING THE ALGORITHMS

All mentioned algorithms were implemented in C++ by the authors of the paper. They are divided in two categories, based on the time complexity: $O(n^2)$ and $O(n)$. For testing, five different types of terrain profiles were used:

a) **Random terrain** (i): The terrain heights with uniform distributed values between 0 and 1000 meters have been generated randomly.

b) **Monotone increasing terrain** (ii): Each consecutive terrain height is higher. In this way, all points are visible.

c) **Two visible points** (iii): Only the first two points of the terrain are visible, all others are invisible.

d) **Alternating visibility** (iv): Every second terrain point is visible, while all others are invisible. Fig. 5 shows an example of such terrain profile.

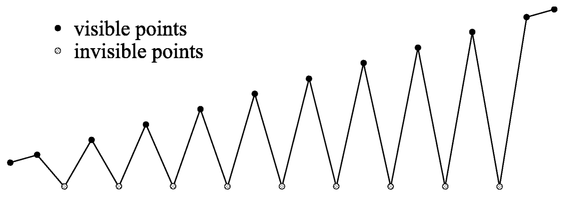

Figure 5: Alternating visibility of terrain points

e) **Fractal terrain** (v): Fractal Brownian motion has been applied for the generation of the terrain [Peitg88]. Fractal dimensions between 0,2 and 0,9, frequency factors between 0,5 and 2,0, and numbers of octaves between 5 and 10, have been generated randomly. The values between 0 and 1000 meters have been obtained.

In all cases, the viewpoint has been set on four different heights:

a) **A human view** (+1,6 m): The viewpoint was set on an approximate height of a human view, 1,6 meter above the first point of the terrain profile.

b) **An observation view I** (+25 m): A view from an observation tower was simulated, and the viewpoint was set 25 meters above the first point of the terrain profile.

c) **An observation view II** (+40 m): Similar as in the previous case, the viewpoint was set 40 meters above the first point of the terrain profile.

d) **A fixed height** (1500 m): The viewpoint was set on the height of 1500 meters to simulate a view from a flying object (e.g. a balloon or a helicopter).

Tests have been done on a portable computer with Intel Pentium III 500 MHz processor and

64 MB of memory under Windows operating system. Each test was repeated 20 times, always for 10 different numbers of terrain points (n). In the continuation, the tables for both classes of the algorithms are given in a condensed form. For all algorithms the number of terrain points and the average spent CPU time in seconds are given.

Because during the test, the results for different viewpoint heights did not differ significantly, only the results for viewpoint height +1,6 m are given in the continuation. Table 1 show the results for the first two algorithms. For the terrain types

Algorithm\\n	10000	20000	30000	40000
Naive (i)	0,00	0,01	0,02	0,00
LOSHeight (i)	0,05	0,01	0,02	0,01
visible points	2,07	2,11	3,53	0,94
Naive (ii)	4,53	17,95	49,73	102,73
LOSHeight (ii)	2,61	11,67	36,72	79,20
Naive (iii)	0,00	0,00	0,00	0,01
LOSHeight (iii)	0,01	0,00	0,00	0,02
Naive (iv)	1,77	9,14	22,83	43,29
LOSHeight (iv)	3,64	6,11	16,01	31,04
Naive (v)	3,65	17,23	42,86	83,11
LOSHeight (v)	2,12	10,18	29,97	62,01
visible points	9842,5	16261,1	27114,3	29684,0

Table 1: CPU times (s) and visible points for $O(n^2)$ algorithms at viewpoint height +1,6 m

(i) and (v), also the average numbers of visible points are included. As mentioned in the third Section, the algorithms' run-time is the shortest when only two points are visible and the worst when all points are visible. Run-times can also be seen in Fig. 6 and Fig. 7, where CPU times for the best case, the worst case and for the fractal terrain are given. From the results, it can be also seen that the fractal-generated terrains are close to the worst case distribution.

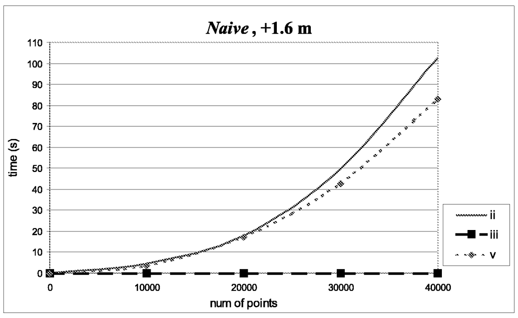

Figure 6: CPU time of *Naive* algorithm for the best case (iii), the worst case (ii) and fractal terrain (v)

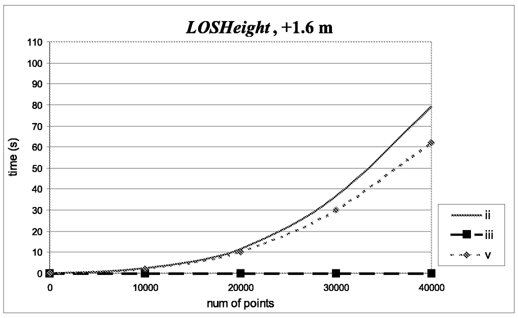

Figure 7: CPU time of *LOSHeight* algorithm for the best case (iii), the worst case (ii) and fractal terrain (v)

Similar experimental analysis has been done for the remaining $O(n)$ algorithms. Table 2 shows the results at viewpoint height +1,6 m.

Algorithm\\n	$1 \cdot 10^6$	$2 \cdot 10^6$	$3 \cdot 10^6$	$4 \cdot 10^6$
BiggestSlope (i)	0,14	0,26	0,40	0,56
CrossProduct (i)	0,16	0,30	0,48	0,65
Incremental (i)	0,15	0,29	0,47	0,60
Incremental+ (i)	0,10	0,20	0,31	0,41
visible points	2,6	4,0	3,6	2,8
BiggestSlope (ii)	0,15	0,31	0,45	0,61
CrossProduct (ii)	0,17	0,32	0,50	0,66
Incremental (ii)	0,14	0,29	0,44	0,58
Incremental+ (ii)	0,13	0,24	0,41	0,64
BiggestSlope (iii)	0,13	0,26	0,39	0,53
CrossProduct (iii)	0,15	0,29	0,46	0,62
Incremental (iii)	0,14	0,29	0,46	0,60
Incremental+ (iii)	0,10	0,19	0,28	0,40
BiggestSlope (iv)	0,14	0,30	0,44	0,60
CrossProduct (iv)	0,16	0,31	0,47	0,64
Incremental (iv)	0,18	0,34	0,51	0,73
Incremental+ (iv)	0,18	0,35	0,54	0,75
BiggestSlope (v)	0,14	0,28	0,42	0,58
CrossProduct (v)	0,16	0,30	0,47	0,63
Incremental (v)	0,14	0,30	0,44	0,59
Incremental+ (v)	0,10	0,21	0,31	0,45
visible points	154645,4	111075,0	347268,0	2525399,5

Table 2: CPU times (s) and visible points for $O(n)$ algorithms at viewpoint height +1,6 m

It can be noticed that the second group of the algorithms is relatively immune to the type of the terrain, as their best and the worst case do not differ much. The algorithm *BiggestSlope* is in the best case for 13% faster than in its worst case, the algorithm *CrossProduct* for 8%, and the algorithm *Incremental* for 17%. The significant difference exposes only the algorithm *Incremental+*, where the best case is in average for 45% faster than its worst case. Fig. 8-11 show those differences.

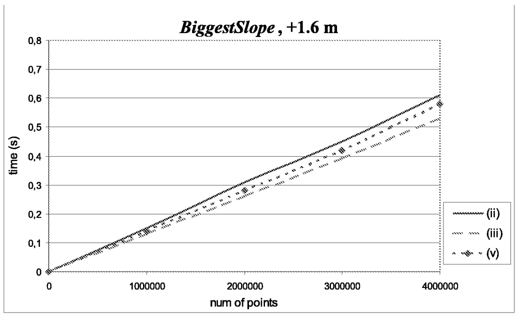

Figure 8: CPU time of *BiggestSlope* algorithm for the best case (iii), the worst case (ii) and fractal terrain (v)

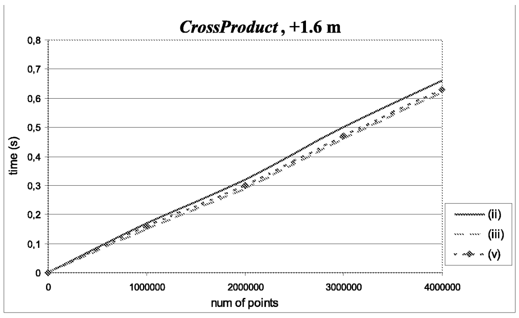

Figure 9: CPU time of *CrossProduct* algorithm for the best case (iii), the worst case (ii) and fractal terrain (v)

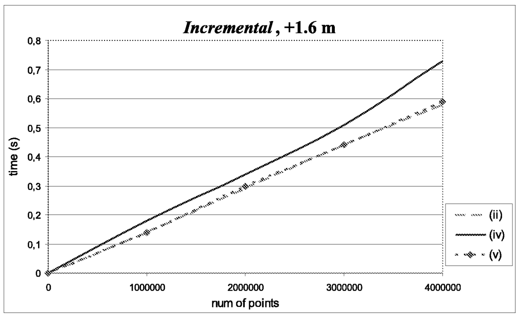

Figure 10: CPU time of *Incremental* algorithm for the best case (ii), the worst case (iv) and fractal terrain (v)

Fig. 12 shows a comparison of the worst case run-times of all $O(n)$ algorithms. Surprisingly, the algorithm *Incremental+* has the worst behavior. However, in real terrain data the worst case for that algorithm is not expected.

A similar analysis can be seen in Fig. 13, where all $O(n)$ algorithms are compared against their best run-times. The algorithm *Incremental+* gives the

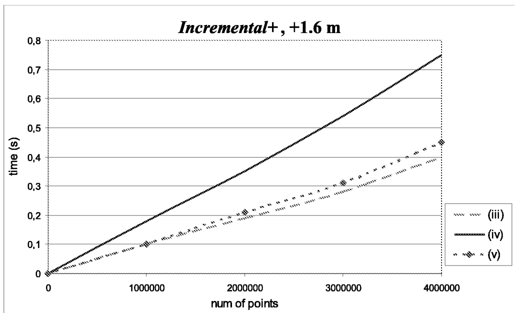

Figure 11: CPU time of *Incremental+* algorithm for the best case (iii), the worst case (iv) and fractal terrain (v)

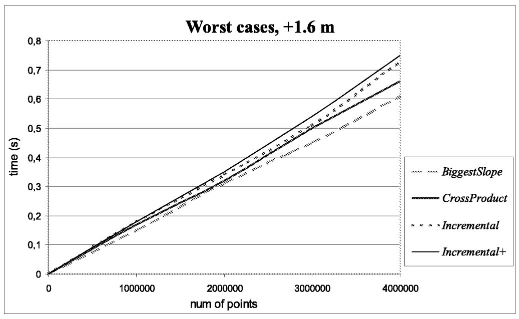

Figure 12: Comparison of $O(n)$ algorithms at their worst cases

best result. The same algorithm shows the best

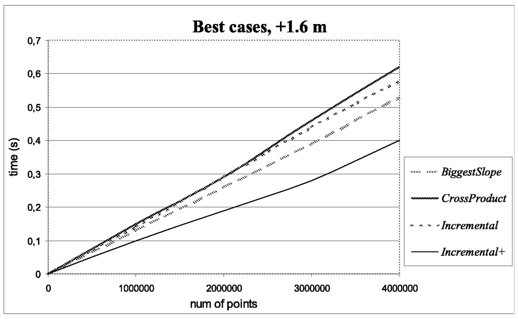

Figure 13: Comparison of $O(n)$ algorithms at their best cases

performance also with the fractal terrain type (see Fig. 14). Moreover, it is in average 25% faster than the second best algorithm, *BiggestSlope*.

5 CONCLUSION

New applications employing the visibility information arise daily. Common to most of them is

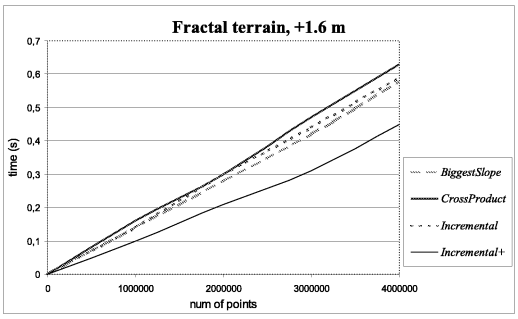

Figure 14: Comparison of $O(n)$ algorithms at the fractal type of the terrain

that the visibility analyses last long and their fundamentals lie in 1,5D. Therefore, the selection of the most appropriate algorithm in 1,5D is crucial.

The paper gives an overview of all known algorithms for the 1,5D visibility problem. All algorithms have been implemented and compared using different configurations of the digital terrain model. From the results, it can be concluded that two algorithms are appropriate to use. The best algorithm is the improved incremental algorithm [Cohen95]. However, it is useful only when the points are equally spaced. In the general distribution of terrain data, the algorithm with the biggest slope of LOSs [Trobe98] is preferred.

At solving the 1,5D visibility problem, two interesting properties occurred:

- Run-times of the algorithms at different viewpoint heights did not differ much.
- The algorithms with linear time complexity behaved similar at different terrain configurations.

The challenge to improve those algorithms remains.

REFERENCES

[Cohen95] Cohen-Or D., Shaked A.: *Visibility and Dead-Zones in Digital Terrain Maps*, Computer Graphics Forum, 14(3), pp. 171–179, 1995

[Cole89] Cole R., Sharir M.: *Visibility Problems for Polyhedral Terrains*, J. Symbolic Computation, 7, pp. 11–30, 1989

[DeFlo94] De Floriani L., Magillo P.: *Visibility Algorithms on Triangulated Digital Terrain Models*, International Journal of Geographic Information Systems, 8(1), pp. 13–41, Taylor and Francis, London, 1994

[DeFlo99a] De Floriani L., Magillo P.: *Intervisibility on Terrains*, Chapter 38 in Geographic Information Systems: Principles, Techniques, Management and Applications, Longley P.A., Goodchild M.F., Maguire D.J., Rhind D.W. (eds), John Wiley & Sons, pp. 543–556, 1999

[DeFlo99b] De Floriani L., Puppo E., Magillo P.: *Applications of Computational Geometry to Geographic Information Systems*, Chapter 7 in Handbook of Computational Geometry, Sack J.R., Urrutia J. (eds), Elsevier Science, pp. 333–388, 1999

[Frank94a] Franklin Wm.R., Ray C.K., Mehta S.: *Geometric Algorithms for Sitting of Air Defense Missile Batteries*, Research Project for Battelle, Columbus Division, Contract Number DAAL03-86-D-0001, Delivery Order Number 2756, 1994

[Frank94b] Franklin Wm.R., Ray C.K.: *Higher isn't Necessarily Better: Visibility Algorithms and Experiments*, Advances in GIS Research: Proceedings of the 6th International Symposium on Spatial Data Handling, Waugh T.C., Healey R.G., (eds), Edinburgh, UK, pp. 751–770, 1994

[Peitg88] Peitgen H.O.: *The Science of Fractal Images*, Springer-Verlag, ISBN 0-387-96608-0, 1988

[Skala87] Skala V.: *An Intersecting Modification to the Bresenham Algorithm*, Computer Graphics Forum, 6(4), pp. 343–347, North Holland Comp., 1987

[Trobe98] Trobec T., Žalik B., Guid N.: *Two Algorithms for Visibility Determination of Raster Relief Models*, Spring Conference on Computer Graphics SCCG 1998 - CSCG'98, Kalos L.S. (ed), Budmerice, Slovak Republic, pp. 247–256, 1998