Fast and Accurate Visibility Computation in a 3D Urban Environment

Oren Gal
Mapping and Geo-information Engineering
Technion - Israel Institute of Technology
Haifa, Israel
e-mail: orengal@technion.ac.il

Yerach Doytsher
Mapping and Geo-information Engineering
Technion - Israel Institute of Technology
Haifa, Israel
e-mail: doytsher@technion.ac.il

Abstract—**This paper presents a unique solution to the visibility problem in 3D urban environments. We shall introduce a visibility algorithm for a 3D urban environment, based on an analytic solution for basic building structures. A building structure is presented as a continuous parameterization approximating of the building's corners. The algorithm quickly generates the visible surfaces' boundary of a single building. Using simple geometric operations of projections and intersections between visible pyramid volumes, hidden surfaces between buildings are rapidly computed. The algorithm, demonstrated with a schematic structure of an urban environment and compared to the Line of Sight (LOS) method, demonstrates the computation time efficiency. The basic building structure can be modified to complex urban structures by merging together a number of basic structures.**

Keywords-Visibility; 3D; Urban environment; Spatial analysis

I. INTRODUCTION

In the last few years, the 3D GIS domain has developed rapidly, and has become increasingly accessible to different disciplines. Spatial analysis in a 3D environment seems to be one of the most challenging topics in the communities currently dealing with spatial data. One of the most basic problems in spatial analysis is related to visibility computation in such an environment. Visibility calculation methods aim to identify the parts visible from a single point, or multiple points, of objects in the environment.

The visibility problem has been extensively studied over the last twenty years, due to the importance of visibility in GIS, computer graphics, computer vision and robotics. Most previous works approximate the visible parts to find a fast solution in open terrains, and do not challenge or suggest solutions for a dense urban environment. The exact visibility methods are highly complex, and cannot be used for fast applications due to the long computation time. Other fast algorithms are based on the conservative Potentially Visible Set (PVS) [8]. These methods are not always completely accurate, as they may include hidden objects' parts as visible due to various simplifications and heuristics.

In this paper, we introduce a new fast and exact solution to the 3D visibility problem from a viewpoint in urban environment, which does not suffer from approximations. We consider a 3D urban environment building modeled as a cube (3D box) and present analytic solution to the visibility problem. The algorithm computes the exact visible and hidden parts from a viewpoint in urban environment, using an analytic solution, without the expensive computational process of scanning all objects' points. The algorithm is demonstrated by a schematic structure of an urban environment, which can also be modified for other complicated urban environments, with simple topological geometric operators. In such cases, computation time grows linearly.

Our method uses simple geometric relations between the objects and the lines connecting the viewpoint and the objects' boundaries by extending the visibility boundary calculation from 2D to a 3D environment by using approximated singular points [9]. The spatial relationship between the different objects is computed by using fast visible pyramid volumes from the viewpoint, projected to the occluded buildings.

The current research tackles the basic case of a single viewpoint in an urban environment, which consists of buildings that are modeled as cubes. More complex urban environments can be defined as a union between the basic structures of several cubes. Further research will focus on modeling more complex urban environments, and facing multiple viewpoints for optimal visibility computation in such environments.

II. PROBLEM STATEMENT

We consider the basic visibility problem in a 3D urban environment, consisting of 3D buildings modeled as 3D cubic parameterization $\sum_{i=1}^{N} C_i(x, y, z =_{h_{\min}}^{h_{\max}})$, and viewpoint $V(x_0, y_0, z_0)$.

Given:

- A viewpoint $V(x_0, y_0, z_0)$ in 3D coordinates
- Parameterizations of N objects $\sum_{i=1}^{N} C_i(x, y, z =_{h_{\min}}^{h_{\max}})$ describing a 3D urban environment model

Computes:

- Set of all visible points in $\sum_{i=1}^{N} C_i(x, y, z =_{h_{\min}}^{h_{\max}})$ from V(x_0, y_0, z_0).

This problem seems to be solved by conventional geometric methods, but as mentioned before, it demands a long computation time. We introduce a fast and efficient computation solution for a schematic structure of an urban environment that demonstrates our method.

III. Analytic Visibility Computation

A. Analytic Solution for a Single Object

In this section, we first introduce the visibility solution from a single point to a single 3D object. This solution is based on an analytic expression, which significantly improves time computation by generating the visibility boundary of the object without the need to scan the entire object's points.

Our analytic solution for a 3D building model is an extension of the visibility chart in 2D introduced by Elber et al. [9] for continuous curves. For such a curve, the silhouette points, i.e. the visibility boundary of the object, can be seen in Figure 1:

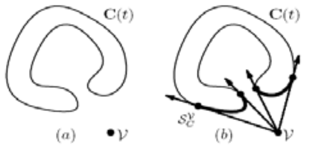

Figure 1. Visible Silhouette Points S_C^V from viewpoint V to curve *C(t)* (source: [9]).

The visibility chart solution was originally developed for dealing with the Art Gallery Problem for infinite viewpoint; it is limited to 2D continuous curves using multivariate solver [9], and cannot be used for on-line application in a 3D environment.

Based on this concept, we define the visibility problem in a 3D environment for more complex objects as:

$$C'(x,y)_{z_{const}} \times (C(x,y)_{z_{const}} - V(x_0, y_0, z_0)) = 0 \quad (1)$$

where 3D model parameterization is $C(x,y)_{z_{const}}$, and the viewpoint is given as $V(x_0, y_0, z_0)$. Solutions to equation (1) generate a visibility boundary from the viewpoint to an object, based on basic relations between viewing directions from V to $C(x,y)_{z_{const}}$ using cross-product characters.

A three-dimensional urban environment consists mainly of rectangular buildings, which can hardly be modeled as continuous curves. Moreover, an analytic solution for a single 3D model becomes more complicated due to the higher dimension of the problem, and is not always possible. Object parameterization is therefore a critical issue, allowing us to find an analytic solution and, using that, to generate the visibility boundary very fast.

1) 3D Building Model: Most of the common 3D City Models are based on object-oriented topologies, such as 3D Formal Data Structure (3D FDS), Simplified Spatial Model (SSS) and Urban Data Model (UDM) [24]. These models are very efficient for web-oriented applications. However, the fact that a building consists of several different basic features makes it almost impossible to generate analytic representation. A three-dimensional building model should be, on the one hand, simple enabling analytic solution, and on the other hand, as accurate as possible. We examined several building object parameterizations, and the preferred candidate was an extended n order sphere coordinates parameterization, even though such a model is a very complex, and will necessitate a special analytic solution. We introduce a model that can be used for analytic solution of the current problem. The basic building model can be described as:

$$x = t, y = \begin{pmatrix} x^n - 1 \\ 1 - x^n \end{pmatrix}, z = c \qquad (2)$$
$$-1 \le t \le 1, n = 350, c = c + 1$$

This mathematical model approximates building corners, not as singular points, but as continuous curves. This building model is described by equation (2), with the lower order badly approximating the building corners, as depicted in Figure 2. Corner approximation becomes more accurate using *n=350* or higher. This approximation enables us to define an analytic solution to the problem.

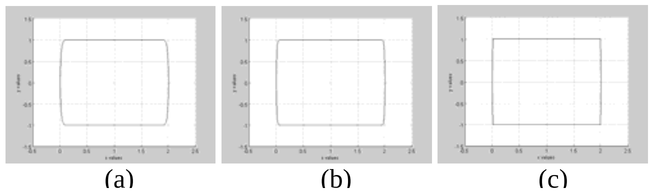

Figure 2. Topside view of the building model using equation (2) - (a) n=50; (b) *n=200*; (c) *n=350*.

We introduce the basic building structure that can be rotated and extracted using simple matrix operators (Figure 3). Using a rotation matrix does not affect our visibility algorithm, and for simple demonstration of our method we present samples of parallel buildings.

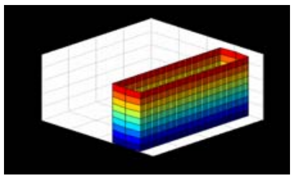

Figure 3. A Three-dimension Analytic Building Model with Equation (2), where $z_{h_{\min}=0}^{h_{\max}=9}$

2) Analytic Solution for a Single Building: In this part we demonstrate the analytic solution for a single 3D building model. As mentioned above, we should integrate building model parameterization to the visibility statement. After integrating eq. (1) and (2):

$$\begin{aligned} &C'(x,y)_{z_{const}} \times (C(x,y)_{z_{const}} - V(x_0,y_0,z_0)) = 0 \rightarrow \\ &x^n - V_{y_0} - n\cdot x^{n-1}(x - V_{x_0}) - 1 = 0 \\ &x^n + V_{y_0} - n\cdot x^{n-1}(x - V_{x_0}) - 1 = 0 \\ &n = 350, -1 \le x \le 1 \end{aligned} \tag{3}$$

where the visibility boundary is the solution for these coupled equations. As can be noticed, these equations are not related to Z axis, and the visibility boundary points are the same ones for each x-y surface due to the model's characteristics. Later on, we treat the relations between a building's roof and visibility height in our visibility algorithm, as part of the visibility computation.

The visibility statement leads to two polynomial *N* order equations, which appear to be a complex computational task. The real roots of these polynomial equations are the solution to the visibility boundary. These equations can be solved efficiently by finding where the polynomial equation changes its sign and cross zero value; generating the real roots in a very short time computation (these functions are available in Matlab, Maple and other mathematical programs languages). Based on the polynomial cross zero solution, we can compute a fast and exact analytic solution for the visibility problem from a viewpoint to a 3D building model. This solution allows us to easily define the Visible Boundary Points.

Visible Boundary Points (VBP) - we define VBP of the object *i* as a set of boundary points $j=1..N_{bound}$ of the visible surfaces of the object, from viewpoint $V(x_0, y_0, z_0)$.

$$VBP_{i=1}^{j=1..N_{bound}}(x_0,y_0,z_0) = \begin{bmatrix} x_1, y_1, z_1 \\ x_2, y_2, z_2 \\ .. \\ x_{N_{bound}}, y_{N_{bound}}, z_{N_{bound}} \end{bmatrix} \tag{4}$$

Roof Visibility – The analytic solution in equation (3) does not treat the roof visibility of a building. We simply check if viewpoint height $V(z_0)$ is lower or higher than the building height $h_{\max_{C_i}}$ and use this to decide if the roof is visible or not:

$$V_{z_0} \ge Z = h_{\max_{C_i}} \tag{5}$$

If the roof is visible, roof surface boundary points are added to VBP. Roof visibility is an integral part of VBP computation for each building.

Two simple cases using the analytic solution from a visibility point to a building can be seen in Figure 4. The visibility point is marked in black, the visible parts colored in red, and the invisible parts colored in blue. The visible volumes are computed immediately with very low computation effort, without scanning all the model's points, as is necessary in LOS-based methods for such a case.

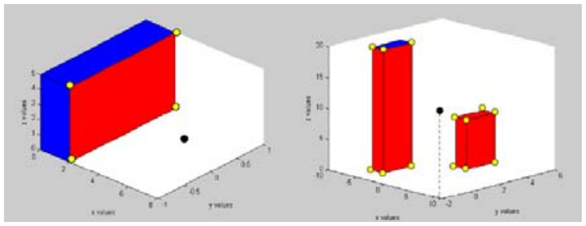
(a) (b)

Figure 4. Visibility Volume computed with the Analytic Solution. Viewpoint is marked in black, visible parts colored in red, and invisible parts colored in blue. VBP marked with yellow circles - (a) single building; (b) two non-overlapping buildings.

B. Visibility Computation in Urban Environments

In the previous sections, we treated a single building case, without considering hidden surfaces between buildings, i.e. building surface occluded by other buildings, which directly affect the visibility volumes solution. In this section, we introduce our concept for dealing with these spatial relations between buildings, based on our ability to rapidly compute visibility volume for a single building generating VBP set.

Hidden surfaces between buildings are simply computed based on intersections of the visible volumes for each object. The visible volumes are defined easily using VBP, and are defined, in our case, as Visible Pyramids. The invisible components of the far building are computed by intersecting the projection of the closer buildings' VP base to the far building's VP base as described in 4.2.2.

1) The Visible Pyramid (VP): we define $VP_i^{j=1..Nsurf}(x_0, y_0, z_0)$ of the object *i* as a 3D pyramid generated by connecting VBP of specific surface *j* to a viewpoint $V(x_0, y_0, z_0)$. Maximum number of N_{surf} for a single object is three. VP boundary, colored with green arrows, can be seen in Figure 5. The intersection of VPs allows us to efficiently compute the hidden surfaces in urban environments, as can be seen in the next sub-section.

2) Hidden Surfaces between Buildings: As we mentioned earlier, invisible parts of the far buildings are computed by intersecting the projection of the closer buildings' VP to the far buildings' VP base.

For simplicity, we demonstrate the method with two buildings from a viewpoint $V(x_0, y_0, z_0)$ one (denoted as the first one) of which hides, fully or partially, the other (the second one).

As can be seen in Figure 6, in this case, we first compute VBP for each building separately, $VBP_1^{1..4}$, $VBP_2^{1..4}$, based on these VBPs, we generate VPs for each building, VP_1^1, VP_2^1. After that, we project VP_1^1 base to VP_2^1 base plane, as seen in Figure 7, if existing. At this point, we intersect the projected surface in VP_2^1 base plane and update $VBP_2^{1..4}$ and VP_2^1 (decreasing the intersected part).

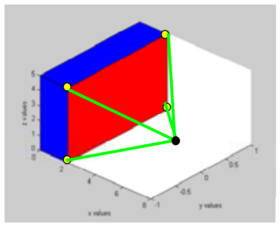

Figure 5. A Visible Pyramid from a viewpoint (marked as a black point) to VBP of a specific surface

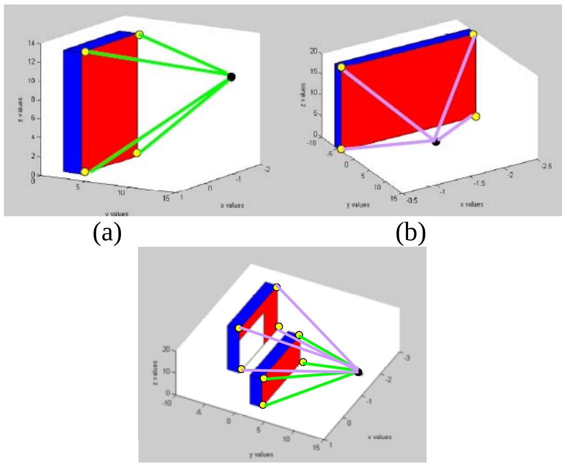
(a) (b) (c)

Figure 6. Generating VP - (a) VP_1^1 boundary colored in green arrows; (b) VP_2^1 boundary colored in purple lines; (c) the two buildings - VP_1^1in green and VP_2^1 in purple, from the viewpoint.

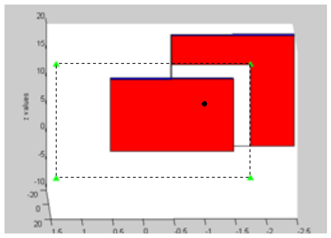

Figure 7. Projection of VP_1^1 *to* VP_2^1base plane marked with dotted lines.

The intersected part is the invisible part of the second building from viewpoint $V(x_0, y_0, z_0)$ hidden by the first building, which is marked in white in Figure 8. In the case of a third building, in addition to the buildings introduced in Figure 8, the projected VP will only be the visible ones, and the VBP and VP of the second building will be updated accordingly (as is described in the next sub-section - stage 2.3.4.3).

We demonstrated a simple case of an occluded building. A general algorithm for more a complex scenario, which contains the same actions between all the combinations of VP between the objects, is detailed in the next sub-section. Projection and intersection of 3D pyramids can be done with simple computational geometry elements, which demand a very low computation effort.

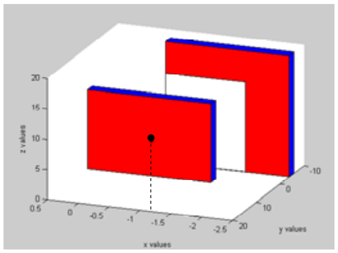

Figure 8. Computing Hidden Surfaces between Buildings by using the Visible Pyramid Colored in White on VP_2^1 Base Plane.

C. *Visibility Algorithm Pseudo - Code*

```
1. Given viewpoint $V(x_0, y_0, z_0)$
2. For $i=1:1:N_{models}$ building model
   2.1. Calculate Azimuth $\theta_i$ and Distance $D_i$ from
        viewpoint to object
2.2. Set and Sort Buildings Azimuth Array $\theta[i]$
2.3. IF Azimuth Objects $(i, 1..i-1)$ Intersect
      2.3.1. Sort Intersected Objects $j=1:1:N_{insect}$ by
             Distance
      2.3.2. Compute VBP for each intersected
             building, $VBP_{j=1..N_{\text{int sec}}}^{1..N_{bound}}$ .
      2.3.3. Generate VP for each intersected
             building, $VP_{j=1..N_{\text{int sec}}}^{1..N_{surf}}$
      2.3.4. For $j=1:1:N_{insect}-1$
             2.3.4.1. Project $VP_j^{1..N_{surf}}$ base to
                      $VP_{j+1}^{1..N_{surf}}$ base plane, if exist.
             2.3.4.2. Intersect projected surfaces in
                      $VP_{j+1}^{1..N_{surf}}$ base plane.
             2.3.4.3. Update $VBP_{j+1}^{1..N_{bound}}$ and
                      $VP_{j+1}^{1..N_{surf}}$ .
             End
      Else
             Locate Building in Urban Environment
   End
End
```

D. *Visibility Algorithm – Complexity Analysis*

We analyze our algorithm complexity based on the pseudo code presented in the previous section, where *n* represents the number of buildings. In the worst case, *n* buildings hide each other. Visibility complexity consists of generating VBP and VP for *n* buildings, $n \cdot O(1)$ complexity. Projection and intersection are also $n \cdot O(1)$ complexity.

The complexity of our algorithm, without considering data structure managing for urban environments, is $n \cdot O(n)$.

We analyze the visibility algorithm complexity of the LOS methods, where *n* represents the number of buildings and *k* represents the resolution of the object. The exact visibility computation requires scanning each object and each object's points, *O(nk)* where usually *k>>n*.

IV. Results

We have implemented the presented algorithm and tested some urban environments on a 1.8GHz Intel Core CPU with Matlab. First, we analyze the versatility of our algorithm on two different test scenes with different occluded elements. After that, we compare our algorithm to the basic LOS visibility computation, to prove accuracy and computational efficiency. Test scenes can be seen in FigureT 9.

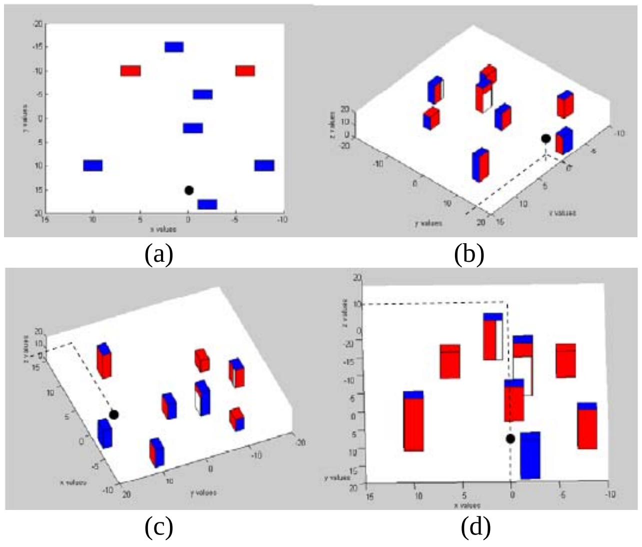

Figure 9. Scene number 1: Eight buildings in an Urban Environment, $V(x_0, y_0, z_0)$= (0,15,10) - (a) Topside view; (b)-(d) Different views demonstrating the visibility computation using our algorithm. CPU time was 0.15 sec.

A. Computation Time and Comparison to LOS

The main contribution of this research focuses on a fast and accurate visibility computation in urban environments. We compare our algorithm time computation with common LOS visibility computation demonstrating algorithm's computational efficiency.

1) Visibility Computation Using LOS: The common LOS visibility methods require scanning all of the object's points. For each point, we check if there is a line connecting the viewpoint to that point which does not cross other objects. We used LOS2 Matlab function, which computes the mutual visibility between two points on a Digital Elevation Model (DEM) model. We converted our first test scene with one to eight buildings to DEM, operated LOS2 function, and measured CPU time after model conversion. Each building with DEM was modeled homogonously by 50 points. The visible parts using the LOS method were the exact parts computed by our algorithm. Obviously, computation time of LOS method was about 500 times longer than our algorithm using analytic solution. CPU time of our analytic solution and LOS method are introduced in Figure 10.

Over the last years, efficient LOS-based visibility methods for DEM models, such as *Xdraw*, have been introduced in order to generate approximate solutions [12]. However, the computation time of these methods is at least *O(n(n-1))*, and, above all, the solution is an approximate one.

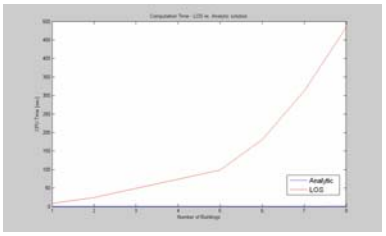

Figure 10. CPU Computation Time of LOS and our algorithm. CPU was measured in the first scene with an increasing number of buildings from one to eight. LOS method was 500 times longer than our algorithm.

V. Related Work

Accurate visibility computation in 3D environments is a very complicated task demanding a high computational effort, which can hardly been done in a very short time using traditional well-known visibility methods [1], [16]. Previous research in visibility computation has been devoted to open environments using DEM models, representing raster data in 2.5D (Polyhedral model). Most of these works have focused on approximate visibility computation, enabling fast results using interpolations of visibility values between points, calculating point visibility with the LOS method [6], [12].

A vast number of algorithms have been suggested for speeding up the process and reducing the computation time [15]. Franklin [11] evaluates and approximates visibility for each cell in a DEM model based on greedy algorithms. An application for siting multiple observers on terrain for optimal visibility cover was introduced in [13]. Wang et al. [21] introduced a Grid-based DEM method using viewshed horizon, saving computation time based on relations between surfaces and Line Of Sight (LOS), using a similar concept of Dead-Zones visibility [3]. Later on, an extended method for viewshed computation was presented, using reference planes rather than sightlines [22].

One of the most efficient methods for DEM visibility computation is based on shadow-casting routine. The routine cast shadowed volumes in the DEM, like a light bubble [17]. Other methods related to urban design environment and open space impact treat abstract visibility analysis in urban environments using DEM, focusing on local areas and approximate openness [10], [23]. Extensive research treated Digital Terrain Models (DTM) in open terrains, mainly Triangulated Irregular Network (TIN) and Regular Square Grid (RSG) structures. Visibility analysis on terrain was classified into point, line and region visibility, and several algorithms were introduced based on horizon computation describing visibility boundary [4], [5].

Only a few works have treated visibility analysis in urban environments. A mathematical model of an urban scene, calculating probabilistic visibility for a given object from a specific viewcell in the scene, has been presented by [14]. This is a very interesting concept, which extends the traditional deterministic visibility concept. Nevertheless, the buildings are modeled as circles, and the main challenges of spatial analysis and building model were not tackled.

Other methods were developed, subject to computer graphics and vision fields, dealing with exact visibility in 3D scenes, without considering environmental constraints. Plantinga and Dyer [16] used the *aspect graph* – a graph with all the different views of an object. Shadow boundaries computation is a very popular method, studied by [20], [7], [19]. All of these works are not applicable to a large scene, due to computational complexity.

As mentioned, online visibility analysis is a very complicated task. Recently, off-line visibility analysis, based on preprocessing, was introduced. Cohen-Or et al. [2] used a ray-shooting sample to identify occluded parts. Schaufler et al. [18] use blocker extensions to handle occlusion.

VI. Conclusions and Future Work

We have presented an efficient algorithm for visibility computation in an urban environment, modeling basic building structure with mathematical approximating for presentation of buildings' corners. Our algorithm is based on a fast visibility boundary computation for a single object, and on computing the hidden surfaces between buildings by using projected surfaces and intersections of the visible pyramids. Complexity analysis of our algorithm has been presented, as well as the computational running time as compared to LOS visibility computation showing significant improvement of time performance.

The main contribution of the method presented in this paper is that it does not require special hardware, and is suitable for on-line computations based on the algorithms' performances, as was presented above. The visibility solution is exact, defining a simple problem that can be a basic form of other complicated environments. Further research will focus on modeling more complex urban environments and facing multi viewpoints for optimal visibility computation in such environments, generalizing the presented building model for more complex ones.

VII. References

[1] Chrysanthou Y., "Shadow Computation for 3D Interactive and Animation," Ph.D. Dissertation, Department of Computer Science, College University of London, UK, 1996.

[2] Cohen-Or D., Fibich G., Halperin D., and Zadicario E., "Conservative Visibility and Strong Occlusion for Viewspace Partitioning of Densely Occluded Scenes," In EUROGRAPHICS'98, 1998.

[3] Cohen-Or D. and Shaked A., "Visibility and Dead- Zones in Digital Terrain Maps," Eurographics, vol. 14(3), pp. 171- 180, 1995.

[4] De Floriani L. and Magillo P., "Visibility Algorithms on Triangulated Terrain Models," International Journal of Geographic Information Systems, vol. 8(1), pp. 13-41, 1994.

[5] De Floriani L. and Magillo P., "Intervisibility on Terrains," In P.A. Longley, M.F. Goodchild, D.J. Maguire & D.W. Rhind (Eds.), Geographic Information Systems: Principles, Techniques, Management and Applications,1999, pp. 543-556. John Wiley & Sons.

[6] Doytsher Y. and Shmutter B., "Digital Elevation Model of Dead Ground," Symposium on Mapping and Geographic Information Systems (Commission IV of the International Society for Photogrammetry and Remote Sensing), Athens, Georgia, USA, 1994.

[7] Drettakis G. and Fiume E., "A Fast Shadow Algorithm for Area Light Sources Using Backprojection," In Computer Graphics (Proc. of SIGGRAPH '94), 1994, pages 223–230.

[8] Durand F., "3D Visibility: Analytical Study and Applications," PhD thesis, Universite Joseph Fourier, Grenoble, France, 1999.

[9] Elber G., Sayegh R., Barequet G. and Martin R., "Two-Dimensional Visibility Charts for Continuous Curves," Shape Modeling International 05, MIT, Boston, USA, 2005, pp. 206-215.

[10] Fisher-Gewirtzman D. and Wagner I.A., "Spatial Openness as a Practical Metric for Evaluating Built-up Environments," Environment and Planning B: Planning and Design vol. 30(1), pp. 37-49, 2003.

[11] Franklin W.R., "Siting Observers on Terrain," in D. Richardson and P. van Oosterom, eds, Advances in Spatial Data Handling: 10th International Symposium on Spatial Data Handling. Springer-Verlag, 2002, pp. 109–120

[12] Franklin W.R. and Ray C., " Higher isn't Necessarily Better: Visibility Algorithms and Experiments," In T. C. Waugh & R. G. Healey (Eds.), Advances in GIS Research: Sixth International Symposium on Spatial Data Handling, 1994, pp. 751–770. Taylor & Francis, Edinburgh.

[13] Franklin W.R. and Vogt C., "Multiple Observer Siting on Terrain with Intervisibility or Lores Data," in XXth Congress, International Society for Photogrammetry and Remote Sensing. Istanbul, 2004.

[14] Nadler B., Fibich G., Lev-Yehudi S. and Cohen-Or D.,"A Qualitative and Quantitative Visibility Analysis in Urban Scenes," Computers & Graphics, 1999, pp. 655-666.

[15] Nagy G., "Terrain Visibility," Technical report, Computational Geometry Lab, ECSE Dept., Rensselaer Polytechnic Institute, 1994

[16] Plantinga H. and Dyer R., "Visibility, Occlusion, and Aspect Graph," The Int. Journal of Computer Vision, vol. 5(2), pp.137-160, 1990.

[17] Ratti C, "The Lineage of Line: Space Syntax Parameters from the Analysis of Urban DEMs'," Environment and Planning B: Planning and Design, vol. 32, pp. 547-566, 2005.

[18] Schaufler G., Dorsey J., Decoret X. and Sillion F.X., "Conservative Volumetric Visibility with Occluder Fusion," In Computer Graphics, Proc. of SIGGRAPH 2000, pp. 229-238.

[19] Stewart J. and Ghali S., "Fast Computation of Shadow Boundaries Using Spatial Coherence and Backprojections," In Computer Graphics, Proc. of SIGGRAPH 1994, pp. 231-238.

[20] Teller S. J., "Computing the Antipenumbra of an Area Light Source," Computer Graphics, vol. 26(2), pp.139-148, 1992.

[21] Wang, J., G.J. Robinson, and K. White, "A Fast Solution to Local Viewshed Computation Using Grid-based Digital Elevation Models," Photogrammetric Engineering & Remote Sensing, vol. 62, pp.1157-1164, 1996.

[22] Wang, J., G.J. Robinson, and White K., "Generating Viewsheds without Using Sightlines," Photogrammetric Engineering & Remote Sensing, vol. 66, pp. 87-90, 2000.

[23] Yang, P.P.J., Putra, S.Y. and Li, W., "Viewsphere: a GIS-based 3D Visibility Analysis for Urban Design Evaluation," Environment and Planning B: Planning and Design, vol. 43, pp.971-992, 2007.

[24] Zlatanova S., Rahman A., and Wenzhong S., "Topology for 3D Spatial Objects," Int. Sym. and on Geoinformation, 2002.