Analyzing 3D Complex Urban Environments Using a Unified Visibility Algorithm

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 for 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. Furthermore, extended visibility analysis for complex urban environments, consisting of mass modeling shapes, is presented. Mass modeling consists of basic shape vocabulary with a box as the basic structure. Using boxes as simple mass model shapes, one can generate complex urban building blocks such as *L, H, U,* and *T* shapes. The visibility analysis is based on concatenating the analytic solution for the basic single box building structure. 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. Real urban environment approximated to the 3D basic shape vocabulary model demonstrates our approach.**

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

I. INTRODUCTION

In the past few years, the 3D GIS domain has developed rapidly, and has become increasingly accessible to different disciplines. Spatial analysis in a 3D environment appears 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 past 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. Gal and Doytsher [1] recently presented a fast and exact solution for the 3D visibility problem in urban environments based on an analytic solution. Other fast algorithms are based on the conservative Potentially Visible Set (PVS) [2]. 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 for the 3D visibility problem from a viewpoint in an urban environment, which does not suffer from approximations. We consider a 3D urban environment building modeled as a cube (3D box) and present an analytic solution for the visibility problem. The algorithm computes the exact visible and hidden parts from a viewpoint in an 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 almost 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, using approximated singular points [3]. The spatial relationship between the different objects is computed by using fast visible pyramid volumes from the viewpoint, projected to the occluded buildings.

Based on our visibility solution [1], we extend our research and introduce a fast and exact solution to the 3D visibility problem in complex urban environments, generated by mass modeling shapes and a procedural modeling method. Our solution can be carried out in a near Real Time performance. We consider a 3D urban environment, which can be generated by grammar rules. The basic entities are basic vocabulary mass modeling, such as L, H, T profile shapes that can be separated into simple boxes. Based on our visibility method, we analyze the spatial relations for each profile and compute the visible and the hidden parts. Each box is a basic building modeled as 3D cubic parameterization, which enables us to implement an analytic solution for the visibility problem, without the expensive computational process of scanning all the objects' points.

The algorithm is demonstrated by a collection of basic mass modeling shapes of an urban environment, where each shape can be sub-divided into a number of boxes. Using an extension of our analytic solution for the visibility problem of a single box from a viewpoint, an efficient solution for a complex environment is demonstrated. We also compared computation time between the presented method and the traditional "Line of Sight" (LOS) method.

II. RelatedWork

Accurate visibility computation in 3D environments is a very complicated task demanding a high computational effort, which could hardly have been performed in a very short time using traditional well-known visibility methods [4], [5]. 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], [7].

A vast number of algorithms have been suggested for speeding up the process and reducing the computation time [8]. Franklin [9] evaluates and approximates visibility for each cell in a DEM model based on greedy algorithms. An application for sitting multiple observers on terrain for optimal visibility cover was introduced in [10]. Wang et al. [11], 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 [12]. Later on, an extended method for viewshed computation was presented, using reference planes rather than sightlines [13].

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 [14]. 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 [15], [16]. 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 [17], [18].

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 [19]. 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 [5] used the aspect graph – a graph with all the different views of an object. Shadow boundaries computation is a very popular method, studied by [2],[20], [21]. 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. [22] used a ray-shooting sample to identify occluded parts. Schaufler et al. [23] use blocker extensions to handle occlusion.

Shape grammars, which are an inherent part of the procedural modeling method, have been used for several applications over the past years. The first and original formulation of shape grammar deals with arrangement and location of points and labeled lines. Therefore, this method was used for architecture applications, for construction and analysis of architectural design [24].

Modeling a 3D urban environment can be done by dividing and simplifying the environment using a set of grammar rules consisting of basic shape vocabulary of mass modeling [25]. By that, one can simply create and analyze 3D complex urban environments using computer implementation.

Automatic generation or modeling of complex 3D environments, such as the urban case, can be a very complicated task dealing with fast computations analysis. In our case, visibility computation in 3D environments is a very complicated task, which can hardly be performed in a very short time using traditional well-known visibility methods, due to the environment's complexity, modeled with or without a procedural modeling method.

III. URBAN ENVIRONMENT MODELING

A. Procedural Modeling

Procedural modeling consists of production rules that iteratively create more and more details. In the context of urban environments, grammar rules first generate crude volumetric models of buildings, named as mass modeling, which will be introduced in the next sub-section. Iterative rules can also be applied to facade windows and doors. Modeling processes of the environment also specify the hierarchical structure.

Shape grammar, which is also called Computer Generated Architecture (CGA) shape, produces buildings' shells in urban environments with high geometric details. A basic set of grammar rules was introduced by Wonka et al. [25].

Procedural modeling enables us to create fast and different three-dimensional urban models using a combination of random numbers and stochastic rule selection with different heights and widths. An example model using these four rules is depicted in Figure 1.

B. Mass Modeling

Modeling urban environments can be a very complicated task. The simplest constructions use boxes as a basic structure. By using boxes as simple mass models, one can generate basic buildings blocks such as L, H, U, and T shapes, demonstrated in Figure 2.

Figure 1. Generating Urban Environment Using CGA Shape Based on Mass Modeling (source: [26])

An extended mass modeling of roofs and facades for building models was introduced by Muller et al. [26]. In this paper, we introduce visibility analysis of the basic shape vocabulary of mass modeling using a box's basic structure, described as visibility computation of a basic shape vocabulary.

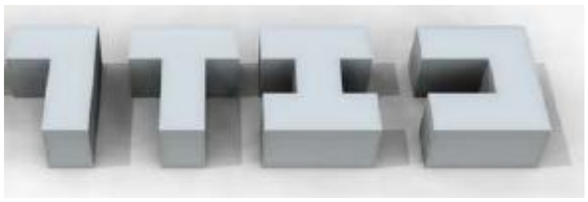


Figure 2. Basic Shape Vocabulary for Mass Modeling (source: [27])

IV. 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 would appear to be solved by conventional geometric methods, but as mentioned before, this 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.

V. 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. [3] for continuous curves. For such a curve, the silhouette points, i.e. the visibility boundary of the object, can be seen in Figure 3:

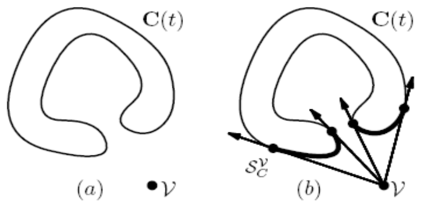


Figure 3.Visible Silhouette Points SCV from viewpoint V to curve C(t) (source: [3])

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 [3], 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:

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

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 quickly.

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) [28]. 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 one, 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:

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

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 buildings' corners, as depicted in Figure 4. Corner approximation becomes more accurate using n=350 or higher. This approximation enables us to define an analytic solution to the problem.

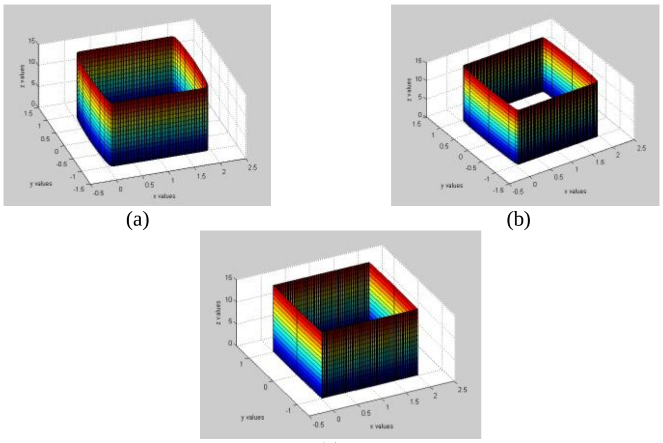

(a) (b) (c)

Figure 4. 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 5). Using a rotation matrix does not affect our visibility algorithm, and for a simple demonstration of our method we present samples of parallel buildings.

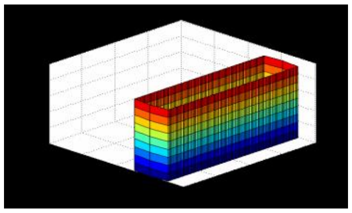


Figure 5. 3D Analytic Building Model with Equation (2), where $z_{h_{min}=0}^{h_{max}=9}$

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 noted, 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 address 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. Currently, we assume flat roof surfaces that will be extended to more complex roof models in our future work.

Two simple cases using the analytic solution from a visibility point to a building, including visible roofs, can be seen in Figure 6. 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 a very low computation effort, without scanning all the model's points, as is necessary in LOS-based methods for such a case.

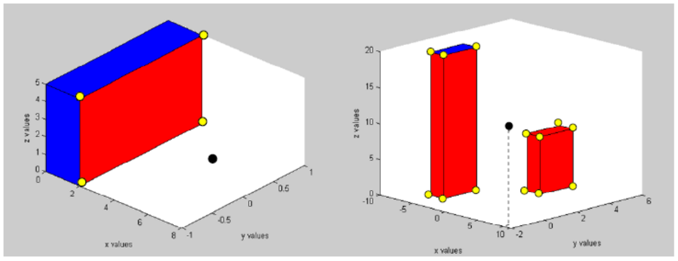


Figure 6. Visibility Volume computed with the Analytic Solution. Viewpoint is marked in black, visible parts colored in red and invisible parts in blue. VBP marked with yellow circles - (a) single building; (b) two non-overlapping buildings

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 easily defined 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.

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 7. The intersection of VPs allows us to efficiently compute the hidden surfaces in urban environments, as 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 farther buildings' VP base.

Let VP_1^i, VP_2^j be visible pyramid from a viewpoint $V(x_0, y_0, z_0)$, The Projected Surface $PS_{VP_1^i}^{VP_2^j}$ from the closer buildings' VP_1^i to the farther buildings' VP_2^j base plane consists of projection of $VBP_1^{1..N_b}$ points:

$$PS_{VP_1^i}^{VP_2^j} = \begin{bmatrix} x_{p1}, y_{p1}, z_{p1} \\ x_{p2}, y_{p2}, z_{p2} \\ .. \\ x_{pi}, y_{pi}, z_{pi} \\ .. \\ x_{pN_{bound}}, y_{pN_{bound}}, z_{pN_{bound}} \end{bmatrix} \tag{6}$$

Where the normal of VP_2^j base plane is (a,b,c) and the plane can be written as $ax + by + cz + d = 0$. The projected point $(x_{p_i}, y_{p_i}, z_{p_i})$ described in equation (6) is:

$$\begin{aligned} x_{pi} &= x_{VBP_1^i} - a\frac{ax_{VBP_1^i} + by_{VBP_1^i} + cz_{VBP_1^i} + d}{a^2+b^2+c^2} \\ y_{pi} &= y_{VBP_1^i} - b\frac{ax_{VBP_1^i} + by_{VBP_1^i} + cz_{VBP_1^i} + d}{a^2+b^2+c^2} \\ z_{pi} &= z_{VBP_1^i} - c\frac{ax_{VBP_1^i} + by_{VBP_1^i} + cz_{VBP_1^i} + d}{a^2+b^2+c^2} \end{aligned} \tag{7}$$

The Intersected Surface $IS_{VP_1^i}^{VP_2^j}$, between $PS_{VP_1^i}^{VP_2^j}$ and VP_2^j base plane can generally describe as polygons intersection:

$$IS_{VP_1^i}^{VP_2^j} = PS_{VP_1^i}^{VP_2^j} \cap \partial VP_2^J \cup \partial PS_{VP_1^i}^{VP_2^j} \cap VP_2^J \tag{8}$$

The Intersected Surface $IS_{VP_1^i}^{VP_2^j}$ is also the invisible one from a viewpoint $V(x_0, y_0, z_0)$, as can be seen in Figure 9.

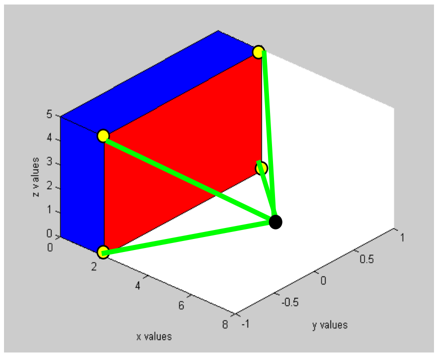


Figure 7. A Visible Pyramid from a viewpoint (marked as a black dot) to VBP of a specific surface

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).

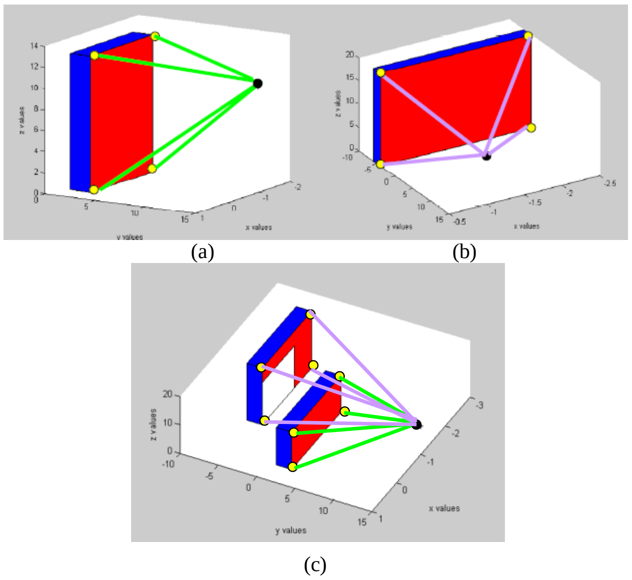


Figure 8. Generating VP - (a) VP_1^1 boundary colored in green lines; (b) VP_2^1 boundary colored in purple lines; (c) the two buildings - VP_1^1 in green and VP_2^1 in purple, and intersected surface in white

As seen in Figure 8, 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 9, if existing. At this point, we intersect the projected surface in VP_2^1 base plane, $PS_{VP_1^i}^{VP_2^i}$, and update $VBP_2^{1..4}$ and VP_2^1 (removing the intersected part). The intersected part is the invisible part of the second building from viewpoint $V(x_0, y_0, z_0)$ which is hidden by the first building $IS_{VP_1^i}^{VP_2^i}$ (marked in white in Figure 9).

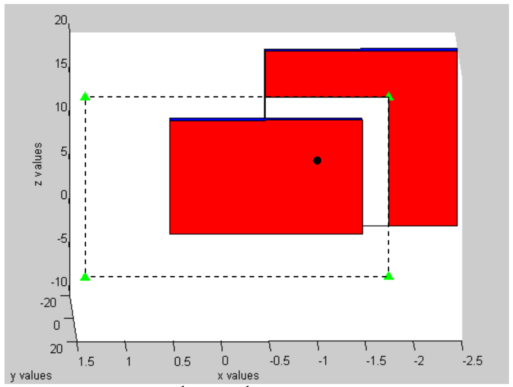


Figure 9. Projection of VP_1^1 to VP_2^1 base plane (projected surface) marked by dotted lines

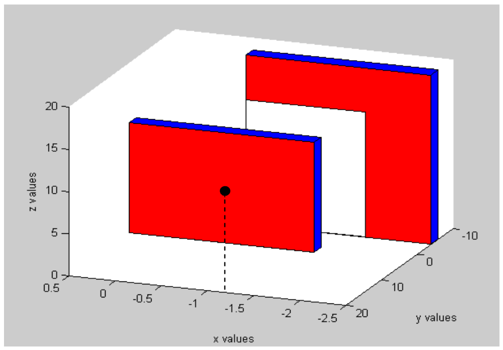


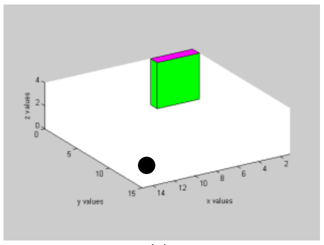
Figure 10. Computing Hidden Surfaces between Buildings by using the Intersected surface on VP_2^1 base Plane.

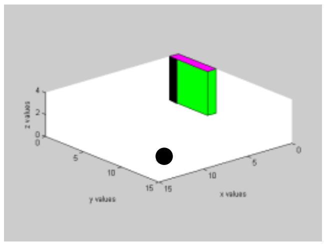
In a case of a third building, in addition to the buildings presented in Figure 10, the projected VP will only be the visible ones, and the VBP and VP of the second building will be updated accordingly (as described in the next sub-section). In cases of several buildings, the VP base would not necessarily be rectangular, due to the intersected surface profile of previous projections. We demonstrated a simple case of an occluded building. A general algorithm for a more 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.

C. *Visibility Analysis for a Basic Shape Vocabulary*

In this section, we present an analysis of visibility aspects of a basic shape vocabulary, as part of the mass modeling of urban environments. Mass modeling shapes consist of boxes as a basic structure, in different shapes such as *L*, *T*, *U*, and *H*. Based on visibility analysis for a single box, and the hidden surfaces removal between overlapping boxes introduced above, we demonstrate an accurate and fast visibility solution for mass modeling buildings profiles.

1) L Shape Vsibility: We demonstrate visibility analysis for an L shape, which can be split into two separate boxes. The profile shape consists of boxes which overlap the visible surfaces, in some cases of the viewpoint location. Let the L shape be separated into two boxes A (Figure 11(a)) and B (Figure 11(b)), visible parts are colored in green, and invisible parts are colored in purple. We compute the VBP of each box $-\mathrm{VBP_A}, \mathrm{VBP_B}$. In the next phase, a visible pyramid is computed for each box $-\mathrm{VP_A^1}, \mathrm{VP_B^1}$. Projection of $\mathrm{VP_A^1}$ to $\mathrm{VP_B^1}$ base plane and intersection between pyramids are colored in black in these figures. The final visible part of L shape can be seen in Figure 11(c). A similar case, with a different viewpoint regarding the L shape, can be seen in Figure 12.


(a)


(b)

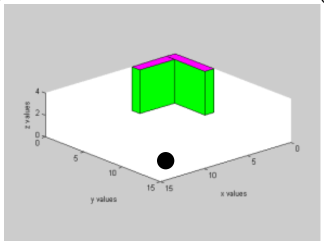
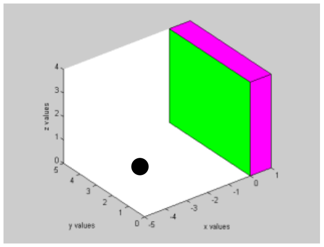
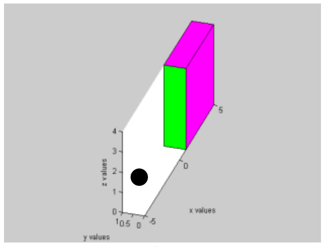

(c)

Figure 11.*L* Shape Visibility Analysis - (a) Box *A* and Viewpoint; (b) Box *B* and Viewpoint where the Hidden Surface Removal is colored in black; (c) *L* Shape with the visible and invisible parts


(a)


(b)

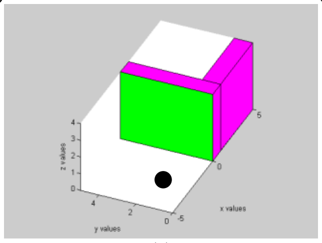
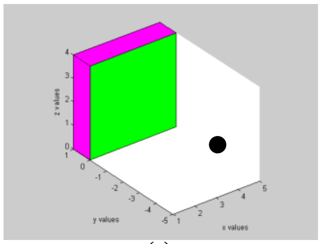
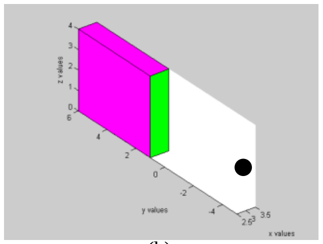
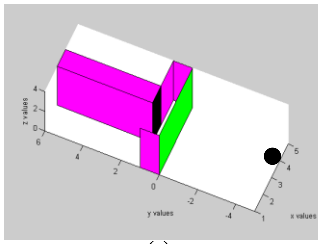

(c)

Figure 12.*L* Shape Visibility Analysis - (a) Box *A* and Viewpoint; (b) Box *B* and Viewpoint; (c) *L* Shape with the aggregated visible and invisible parts

2) T Shape Visibility: In this case, we demonstrate visibility analysis for a T shape, which can be split into two separate boxes (similar to the L shape case) – A (Figure 13(a)) and B (Figure 13(b)), the visible parts are colored in green and invisible parts in purple; and the viewpoint V colored by a black dot. We compute the VBP of each box - $\mathrm{VBP_A}, \mathrm{VBP_B}$. In the next phase, a visible pyramid is computed for each box - $\mathrm{VP_A^1}, \mathrm{VP_B^1}$. Projection of $\mathrm{VP_A^1}$ to $\mathrm{VP_B^1}$ base plane and intersection between pyramids (colored with black) can be seen in Figure 13(c). The final visible part of the T shape can be seen in Figure 13(d).


(a)


(b)


(c)

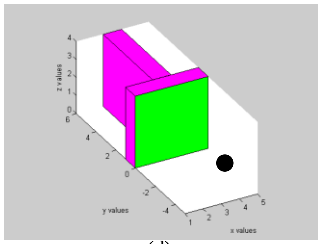
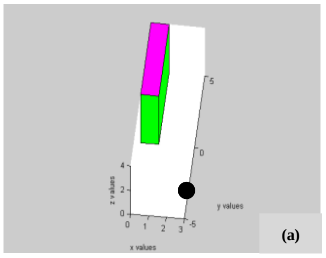
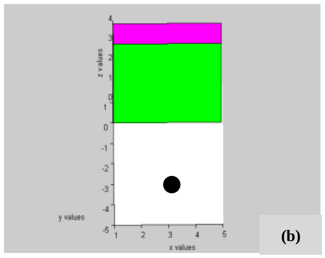

(d)

Figure 13.*T* Shape Visibility Analysis - (a) Box *A* and Viewpoint; (b) Box *B* and Viewpoint; (c) Hidden Surface Removal colored in black and visible surface colored in green; (d) *T* Shape with the visible and invisible parts

3) U Shape Visibility: In this case, we demonstrate the visibility analysis for a U shape, separated into three different boxes - A (Figure 14(a)), B (Figure 14(b)) and C (Figure 14(c)), visible parts are colored in green and invisible parts in purple; and the viewpoint V colored by a black dot. We compute the VBP of each box. In the next phase, a visible pyramid is computed for each box. The outcome of the projection and intersection between visible pyramids can be seen in Figure 14(d) and 14(e), colored with black. The final visible parts of the U shape can be seen in Figure 14(f).

4) H Shape Visibility: In this case, we demonstrate visibility analysis for an H shape, separated into three different boxes – A (Figure 15(a)), B (Figure 15(b)) and C (Figure 15(c)), visible parts are colored in green and invisible parts in purple; and the viewpoint V colored by a black dot. We compute the VBP of each box. In the next phase, a visible pyramid is computed for each box. The outcome of the projection and intersection between visible pyramids are colored in black. The final visible part of the H shape can be seen in Figure 15(d) and 15(e) from two different views.





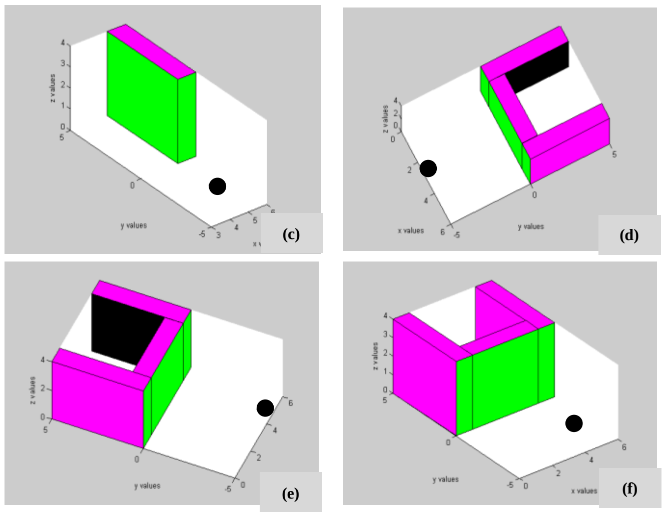


Figure 14.*U* Shape Visibility Analysis - (a) Box *A* and Viewpoint with visible part colored in green; (b) Box *B* and Viewpoint with visible part colored in green; (c) Box *C* and Viewpoint with visible part colored in green; (d) – (e) Hidden Surface Removal colored in black and visible surface colored in green; (f) *U* Shape with the visible and invisible parts

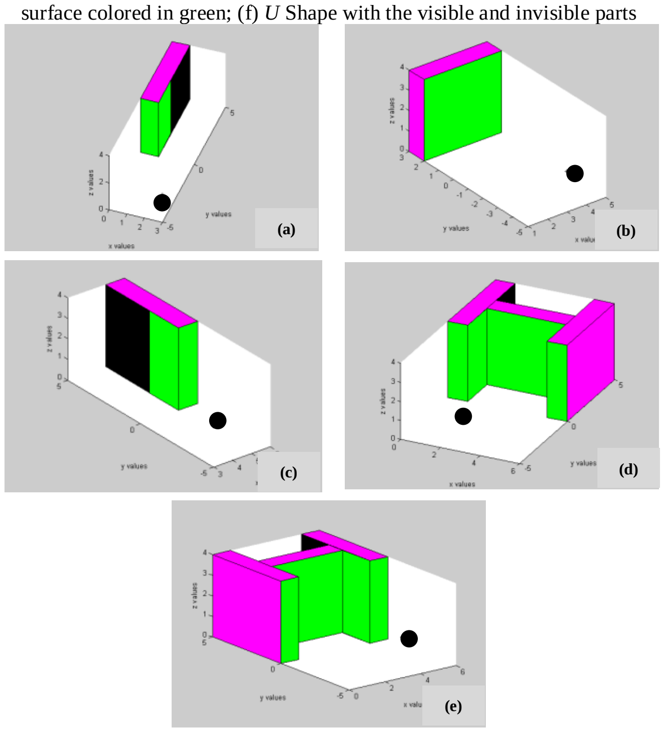


Figure 15.*H* Shape Visibility Analysis - (a) Box *A* and Viewpoint with visible part colored in green; (b) Box *B* with visible part colored in green; (c) Box *C* with visible part colored in green and Hidden Surface Removal colored in black ; (d)-(e) *H* Shape with the visible and invisible parts

D. *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
```

E. *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, $nO(1)$ complexity. Projection and intersection are also $nO(1)$ complexity.

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

```
1. O(1)
2. O(n)
2.1. O(1)
2.2. O(1) – Data structure operator
2.3. O(1) – Data structure operator
    2.3.1. O(1) – Data structure operator
    2.3.2. $n \cdot O(1)$
    2.3.3. $n \cdot O(1)$
    2.3.4. O(1) – Data structure operator
        2.3.4.1. $n \cdot O(1)$
        2.3.4.2. $n \cdot O(1)$
        2.3.4.3. $n \cdot O(1)$
```

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.

VI. 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 four different test scenes with different occluded elements. These test scenes can be seen in Figures (17)-(20).

After that, we compare our algorithm to the basic LOS visibility computation, to prove accuracy and computational efficiency.

Urban environments modeled with mass modeling of a built-up environment consisting of basic shape vocabulary were also tested. First, we analyzed the versatility of our algorithm on a synthetic test scene with different occluded elements (Figure 19) and then on real data -Gibson House Museum Region, Beacon St, MA, USA (Figure 20).

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 the common LOS visibility computation demonstrating our 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 the LOS2 Matlab function, which computes the mutual visibility between two points on a Digital Elevation Model (DEM) model. We converted our last test scene (Figure 20) with one to 58 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. The computation time of the LOS method was about 10,130 times longer than that of our analytic solutionalgorithm in this scene (4,257 sec vs. 0.42 sec). The CPU times of our analytic solution and the LOS method are depicted in Figure 16.

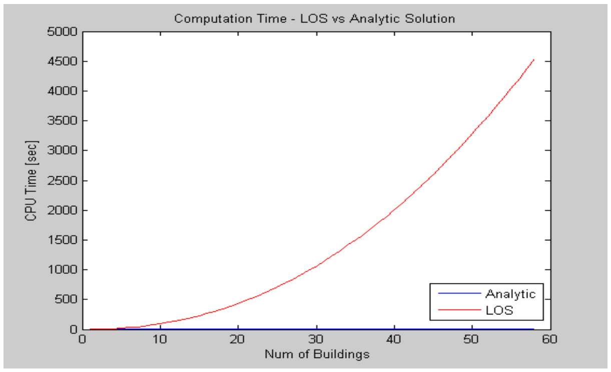


Figure 16.CPU Computation Times of the LOS and our algorithm. CPU was measured with an increasing number of buildings from one to 58. LOS method took 10,130 times longer than our algorithm

In case of mass modeling (Figure 19), computation time of the LOS method was about 6,600 times longer than that of our analytic solution (1,650 sec vs. 0.25 sec).

Over the last years, efficient LOS-based visibility methods for DEM models, such as Xdraw, have been introduced in order to generate approximate solutions [7]. However, the computation time of these methods is at least O(n(n-1)), and, above all, the solution is only an approximate one.

VII. 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.

We have presented an extension from a basic box to a complex urban environment model using the basic shapes vocabulary of mass modeling. Each shape of this modeling can be sub-divided into several boxes, which stand for a basic building structure.

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 presented above. The method generates an exact and quick solution to the visibility problem in relatively complex urban environments, modeled or generated by using procedural modeling consisting of basic shape vocabulary, which can be used for real urban environments, as seen in Scene no. 3. Using these basic shapes, one can create buildings having different shapes (including, for example, balconies).

Complexity analysis of our algorithm has been presented, as well as the computational running time compared to the LOS visibility computation showing significant improvement of time performance.

Further research will focus on facing multi-viewpoints for optimalvisibility computation in such environments, generalizing the presented building model such as cylinders and cones taking into account Level of Details (LOD) and roof modeling.

References

[1] O. Gal and Y. Doytsher, "Fast and Accurate Visibility Computation in a 3DUrban Environment", in Proc. of the Fourth International Conference on Advanced Geographic Information Systems, Applications, and Services, Valencia, 2012, pp: 105-110.

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

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

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

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

[6] Y. Doytsher and B. Shmutter, "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] W.R. Franklin and C. Ray, " 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.

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

[9] W.R. Franklin, "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

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

[11] J. Wang, 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.

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

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

[14] C. Ratti, "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.

[15] D. Fisher-Gewirtzman and I.A. Wagner, "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.

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

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

[18] L. De Floriani and P. Magillo, "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.

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

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

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

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

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

[24] F. Dowing and U. Flemming, "The bungalows of buffalo" Environment and Planning B 8, 1981, 269–293.

[25] P. Wonka, M. Wimmer, F. Sillion, and W. Ribarsky, "Instant architecture". ACM Transactions on Graphics 22, 3, 2003, 669–677.

[26] P. Muller, P. Wonka, S. Hawgler, A. Ulmer, and L.C. Gool, "Procedural Modeling of Buildings" In Proceedings of ACM, SIGGRAPH 2006, pp. 614-623.

[27] G. Schmitt, Architectura et machina. 1993, Vieweg&Sohn.

[28] S. Zlatanova, A. Rahman, and S. Wenzhong, "Topology for 3D Spatial Objects," International Symposium and Exhibition on Geoinformation, 2002, pp. 22-24.

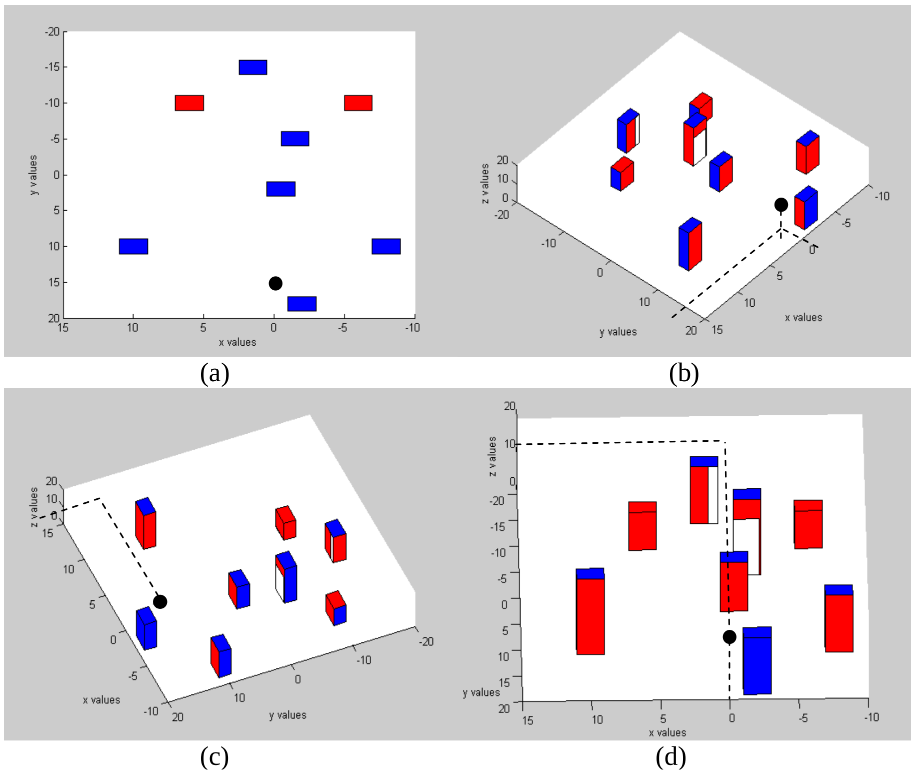


Figure 17.Scene number 1 - Eight buildings in an Urban Environment, $V(x_0, y_0, z_0)$= (0,15,10) - (a) Topside view; (b)-(d) Different views demonstrating the visibility computation using our algorithm. CPU time was 0.15 sec

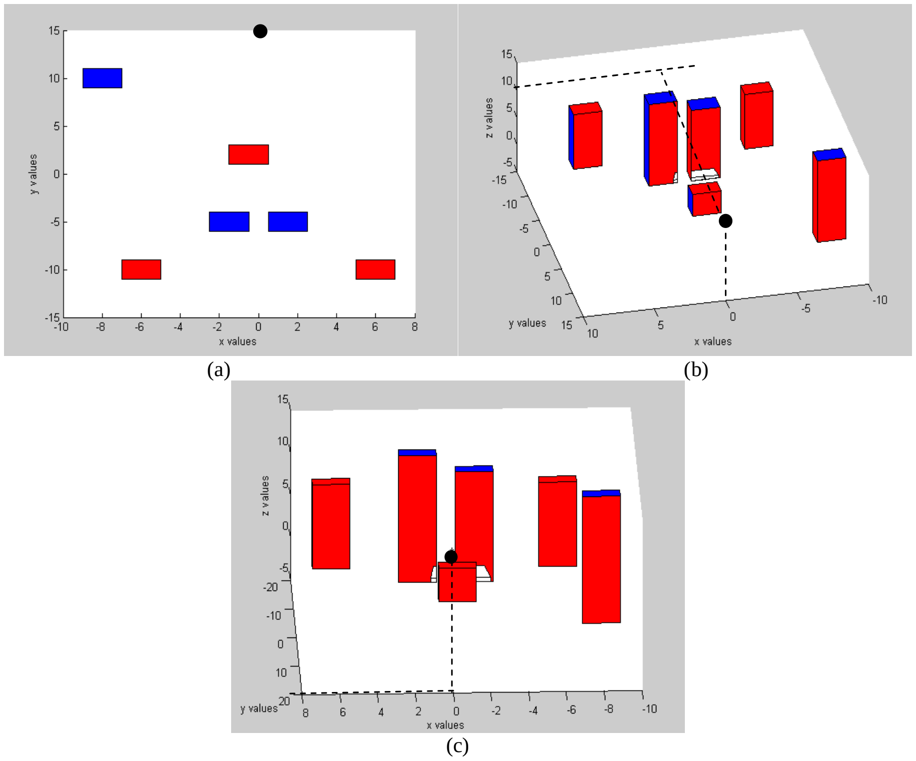


Figure 18.Scene number 2 - Six Buildings in an Urban Environment, where viewpoint is higher than the projected building, $V(x_0, y_0, z_0)$= (0,15,10) - (a) Topside view; (b)-(c) Different views demonstrating visibility computation using our algorithm. CPU time was 0.14 sec

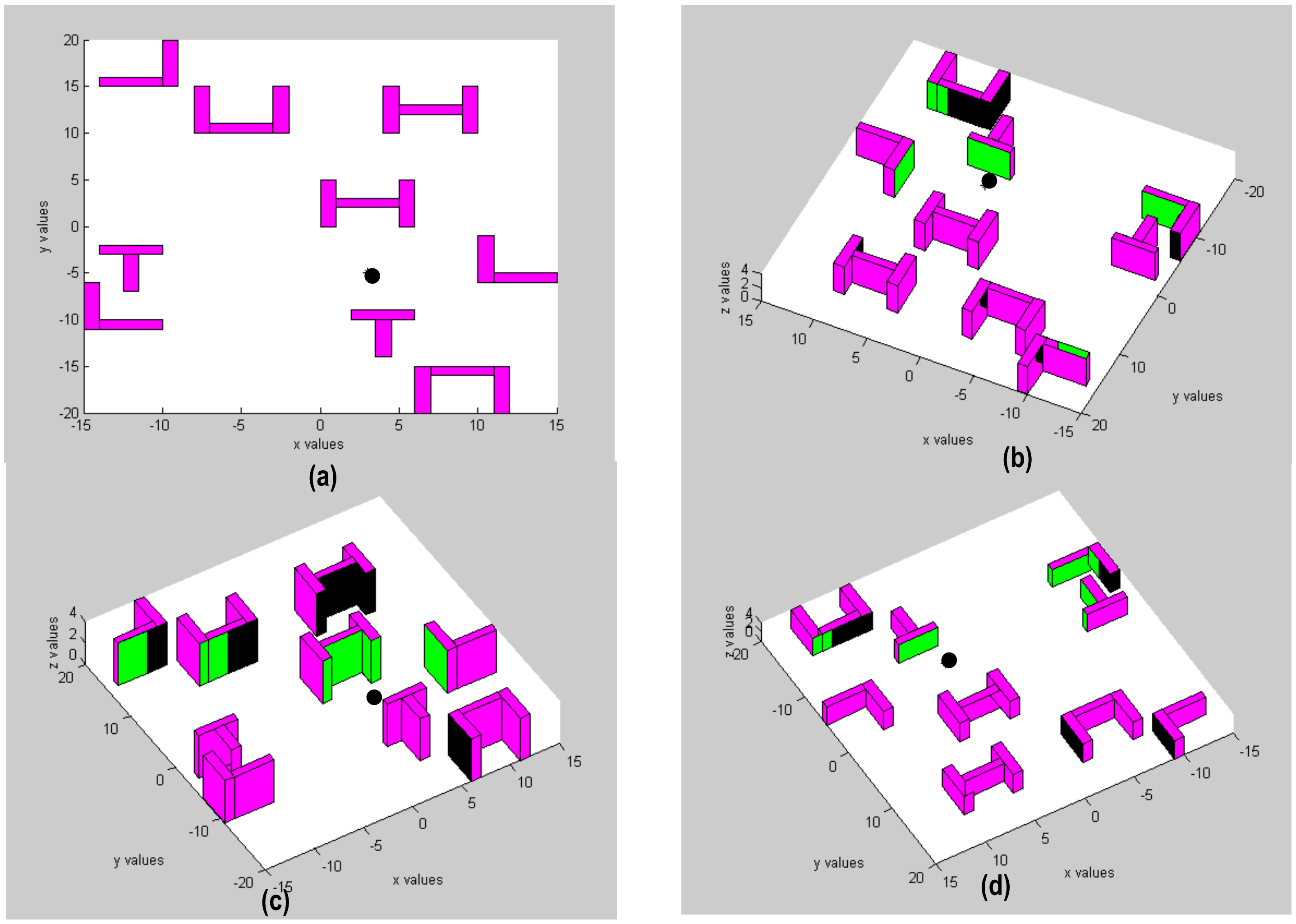
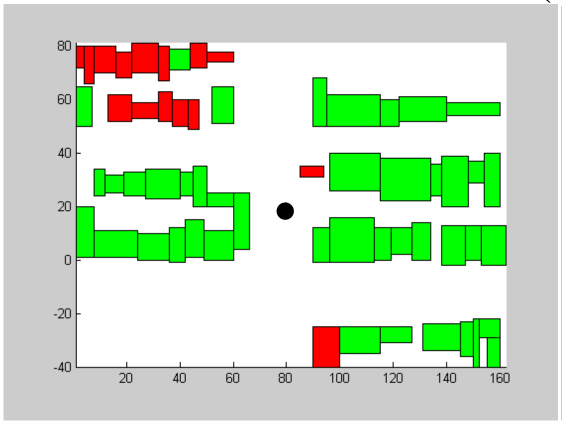
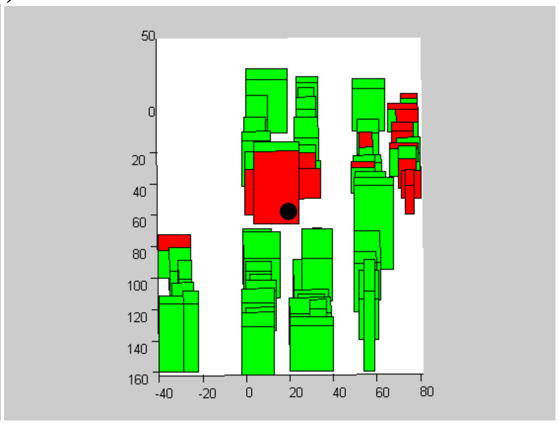


Figure 19.Scene number 3 - Nine basic shape structures of buildings in an Urban Environment, $V(x_0, y_0, z_0)$= (3,-5,2) - (a) Topside view; (b)-(d) Different views demonstrating the visibility computation using our algorithm. CPU time was 0.25 sec

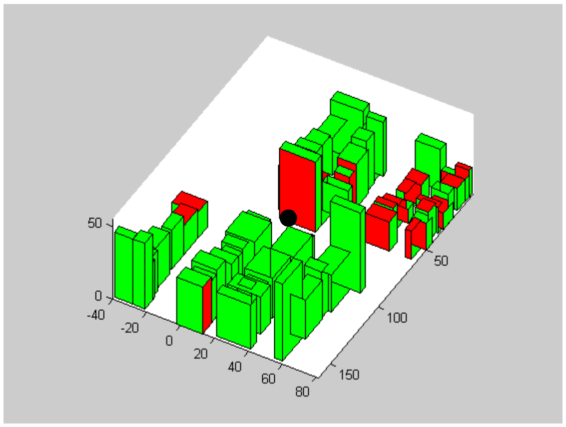
מפה
תנועה
100 רגל
50 מ'
Imagery ©2012 Google, Sanborn, Bluesky, DigitalGlobe, GeoEye, Getmapping plc, Infoterra Ltd & Bluesky, The GeoInformation Group, Map data ©2012 Google

(a)





(b) (c)



(d)

Figure 20. (a) Scene number 4 - Real Data of Urban Environments. (a) Gibson House Museum Region, Beacon St, MA, USA (Google Maps); Visible parts colored in red and invisible parts with green (b) Topview Modeling; (c)-(d) Sideviews. $V(x_0, y_0, z_0)$ = (80,20,20). CPU time was 0.42 sec