3D Visibility and Partial Visibility Complex

Mojtaba NouriBygi [1], Mohammad Ghodsi [1 2]

[1] *Department of Computer Engineering, Sharif University of Technology*
P.O. Box 11365-9517, Tehran, Iran

[2] *IPM School of Computer Science*
P.O. Box 19395-5746, Tehran, Iran

nouribaygi@ce.sharif.edu, ghodsi@sharif.edu

Abstract

Visibility is an important topic in computer graphics, motion planning, and computational geometry. To deal with the increasing complexity of the scenes considered, some research has been performed in visibility processing in order to accelerate the visibility determination. Two of the most studied such structures are visibility graph and visibility complex.

Visibility graph is a fundamental geometric structure which is used in many applications, including illumination and rendering, motion planning, pattern recognition, and sensor networks. While the concept of visibility graph is widely studied for 2D scenes, there is no acceptable equivalence of visibility graph for 3D space. Similarly, 3D visibility complex, proposed as an extension of visibility complex to 3D, is very complicated and cannot be used for visibility computations.

In this paper, we propose a new model for defining the visibility relations in 3D. The main idea is to replace the role of lines and segments in 2D with planes and planar polygons in 3D. Moreover, we define two new structures, namely 3D visibility graph and partial visibility complex, which we believe is the natural way to extend the earlier models. We show how to compute these structures in acceptable times. We also use partial visibility complex to compute the view around a point in 3D in $O((|V(q)|+n^2)\log n)$ time, where $|V(q)|$ is the size of the view.

1 Introduction

Problems involving the visibility of objects have arisen in several areas of computer science, like graphics, VLSI layout, motion planning, and computational geometry. Visibility computation requires an efficient method for determining the objects seen from a point in a scene or from an object in the scene. To deal with the increasing complexity of the scenes considered in computer graphics, some research has been performed in visibility processing in order to accelerate visibility determination. Frequently, the underlying structure of the visibility is critical and a graph can be created that condenses this structure information into a more usable form.

One of these structures is visibility graph. *visibility graph* is a fundamental geometric structure useful in many applications, including illumination and rendering, motion planning, pattern recognition, and sensor networks.

Consider the path planning problem in a 2D polygonal scene. The visibility graph is defined as follows: The nodes are the vertices of the scene, and an arc joins two vertices A and B if they are mutually visible, i.e., if the segment [*AB*] intersects no obstacle. It is possible to go in straight line from A to B only if B is visible from A. The start and goal points are added to the set of initial vertices, and so are the corresponding arcs. Only arcs which are tangent to a pair of polygons are necessary.

This method can be extended to non-polygonal scenes by considering bitangents and portions of curved objects. In the latter case, we will have a *tangent visibility graph (TVG)*. The set of vertices in this graph is $\mathcal{O}$, the convex objects of the scene. Furthermore, any common tangent of two objects $O_1, O_2 \in \mathcal{O}$ whose endpoints can see each other corresponds to an edge $\{O_1, O_2\}$ of the TVG. Both visibility graph of polygonal scene and tangent visibility graph of smooth objects can be computed in optimal $O(k + n \log n)$ time. Here n in the number of polygons or objects and k is the size of the visibility graph.

While the concept of visibility graph is widely studied for 2D scenes, there is no acceptable equivalence of it for 3D space. The reason is the intricate visibility relations in 3D which makes it impossible to express them as easily as

0-7695-2945-3/07 $25.00 © 2007 IEEE
DOI 10.1109/ICCSA.2007.82

was done in 2D.

Another important structure for visibility computations is *visibility complex*. Pocchiola and Vegter [7] have developed 2D visibility complex which is a topological structure encoding all the visibility relations of a 2D scene. The idea is to group rays which "see" the same objects. The central concept is that of *maximal free segments*. These are segments of maximal length that do not intersect the interior of the objects of the scene. More intuitively, a maximal free segment has its extremities on the boundary of objects, it may be tangent to objects but does not cross them. A line is divided in many maximal free segment by the objects it intersects. A maximal free segment represents a group of colinear rays which see the same objects. The visibility complex is the partition of maximal free segments according to the objects at their extremities.

The size of the complex is characterized by the number k of vertices which is $\Omega(n)$ and $O(n^2)$ (n being the number of convex objects). Pocchiola and Vegter [7] proposed an optimal algorithm for its construction, which runs in $O(k \log n)$ time. With this structure, the view around a point can be computed in $O(m \log n)$, where m is the size of view [7].

Durand et al. [2] have proposed a generalization of the visibility complex for 3D scenes of smooth objects and polygons. The space of maximal free segments is then a 4D manifold embedded in 5D because of the branchings. Faces of the complex are bounded by tangent segments (which have three dimensions), bitangent segments (with two dimensions), tri-tangent segments (with one dimensions) and finally vertices are segments tangent to four objects. The size k of 3D visibility complex is $\Omega(n)$ and $O(n^4)$. Although this structure encodes all visibility relations of scene, its complication prevents it to be a useful tool for visibility computations.

In this paper, we propose a new way to express the visibility relations in 3D which is both simple and intuitive and contains enough information to be used in visibility algorithms. Using our new definitions, we define a structure called 3D visibility graph, which we believe is the natural way to extend of visibility graph in 3D. The size of this structure is $O(n^3)$ and we give an $O(n^3 \log n)$ algorithm for its construction. Based on visibility graph, we also introduce a structure called partial visibility complex that is comparable with visibility complex in 3D. This structure does not encodes all the visibility relations. The size of partial complex is corresponds to the size of 3D visibility graph of scene and dominated by $\Omega(n)$ and $O(n^3)$. We give an algorithm for its construction which runs in $O(n^3 \log n)$. An example of application of this structure is computing the view $|V(q)|$ form a viewpoint q in time $O((|V(q)| + n^2) \log n)$.

The rest of the paper is organized as follows. In section 2 we specify some of the reasons that lead to complexities of visibility relations in 3D. In section 3 we define some new definitions for visibility in 3D. In section 4 we introduce a new structure called 3D visibility graph that plays the role of visibility graph in 3D scenes. In section 4.1 we give an algorithm for construction of 3D visibility graph that runs in $O(n^3 \log n)$ time. Following our new method, in section 6 we introduce the partial visibility complex which we believe is the natural way to extend the visibility complex in 3D. We propose an algorithm for its construction that runs in $O(n^3 \log n)$ time. In section 7 we present an application for partial visibility complex.

2 2D Visibility versus 3D Visibility

The visibility problems have been vastly studied and different algorithms have been proposed for various situations. Unfortunately, things are very different when we turn to 3D scenes. Consider the path planning method described above. This problem does not generalize simply to 3D where the problem has been shown to be NP-complete [1, 5]. Furthermore, in 3D, the term "visibility graph" often refers to the abstract graph where each object is a node, and arcs join mutually visible objects. This is however not the direct equivalent of the 2D visibility graph.

We enumerate here some points explaining that the difference between 2D and 3D visibility cannot be summarized by a simple increment to the dimension of the problem. This can be more easily envisioned in line space [3]. Recall that the atomic queries in visibility are expressed in line-space (i.e., first point seen along a ray; are two points mutually visible?).

- Increase in dimension of line-space is two, not one (in 2D line-space is 2D, while in 3D it is 4D). This makes things much more intricate and hard to apprehend.
- A line in 2D is a hyperplane, which is not the case any more in 3D. Thus, the separability property is lost; a 3D line does not separate two half-spaces as in 2D.
- A 4D parameterizations of 3D lines is not possible without singularities [3].
- Visual events are simple in 2D: bitangents lines or tangent to inflection points. In 3D, their locus are surfaces which are rarely planar (IEEE or visual events for curved objects) [3].

All these arguments make the sentence "the generalization to 3D is straightforward" a doubtful statement in any visibility paper.

3 3D Visibility Concepts

As stated in section 2, we cannot simply extend the concept of visibility graph from 2D to 3D. The problem is that the traditional definitions of visibility cannot be used for defining a meaningful and applicable structure such as visibility graph. For one thing, the number of rays between two objects might be countless. For another, it seems impossible to represent the visibility relations of a 3D scene in a planar graph.

These problems lead us to reconsider the basic concepts of visibility in 3D space. In 2D space, we say that two points are mutually visible or see each other if there is a straight line, not intersecting any other part of the configuration, from one object to the other. If we want to use this concept in 3D, instead of using the line, we must use the plane. The reason is that the main property which explains why 3D visibility is much harder than in 2D, is the *separability* property: In 2D, a line separates the plane into two half-planes. No such property holds in 3D because lines are no longer hyperplanes. A good idea is to consider planes: in 3D, planes have this property, i.e., each plane separates the space into two half-spaces.

3.1 Some Definitions

Now we replace the role of the plane in 3D with that of line in 2D and change some basic definitions in visibility. First, we review these definitions in 2D scenes:

- A ray is a half-line starting from a point.
- A segment is a part of line bounded by two points in that line.
- The view from a point in some direction is the first object intersecting the line starting from the point in the given direction.
- We say two points are mutually visible if the line segment defining by them does not intersected with any objects except possibly at the end points.

We now give what we think is a suitable definition for visibility in 3D. As we said, we try to replace the role of line in 2D with plane in 3D:

Definition 1 A pseudo-ray *from point p in the direction of plane A is an angle surrounded by two half-line in A starting from p. Notice that this pseudo-ray is no longer unique.*

Definition 2 *We say that a pseudo-ray can* see *an object if the plane of the pseudo-ray intersects the object and the intersecting section lies within the angle of the pseudo-ray.*

Definition 3 *A* pseudo-segment *in a plane is a simple polygon in that plane. We can imagine this pseudo-segment as the region surrounded by some pseudo-rays (angles). In the simplest form, a pseudo-segment is a triangle in the plane, consisting of three pseudo-rays with mutual common edges. In this case, another way to distinguish a pseudo-segment is by giving its ordered vertices. Notice that a line is a special pseudo-segment consisting of two pseudo-rays (angles) that lie on each other.*

Definition 4 *We say that some points are mutually visible if they are all lie in a same plane and the pseudo-segment defined by them does not intersect with any object except possibly at these points.*

We will use the above definitions for building our new data structure.

3.2 Pseudo-graph

As mentioned, because of the intricate relations, it is not possible to represent the visibility relations of a 3D scene with an ordinary graph. We need a 3D structure comparable to the ordinary graph in 2D. Note that our defined visibility is hold between at least three objects.

We introduce a new structure, called the *pseudo-graph* in which each edge connects three vertices of the graph. We can think of these edges as pseudo-segments connecting its vertices. It is easy to see that the complexity of this structure for a set of n objects is $O(n^3)$.

4 3D Visibility Graph

We now define *3D tangent visibility graph* as follows. Consider a collection $\mathcal{O}$ of pairwise disjoint smooth convex objects in a 3D scene. For our structure we use pseudo-graph, the structure defined in section 3.2 in which each edge connects three vertices of the pseudo-graph. The set of vertices in this graph is $\mathcal{O}$, the convex objects of the scene. Any common tangent plane of two objects $O_1, O_2, O_3 \in \mathcal{O}$ whose tangent points are mutually visible correspond to an edge $\{O_1, O_2, O_3\}$ of the pseudo-graph. Note that there are at most 8 edges between each three vertices.)

The size of 3D visibility graph of n objects, is proportional to the number of edges of it and is limited by $O(n^3)$ and $\Omega(n)$.

4.1 Construction

Here, we present a construction algorithm for 3D visibility graph which runs in $O(n^3 \log n)$ time. It is mainly an extension of the Lee's algorithm for building visibility graphs to 3D scenes. To clear our proof, we first review the Lee's algorithm.

4.1.1 Lee's Algorithm

The algorithm attributed to D. T. Lee represents the first nontrivial solution for constructing the visibility graph, running in $O(n^2 \log n)$ time [4]. The basic idea is simple: for each vertex, sort the other points in angular order around it, then visit each one keeping track of the order of intersected edges made by the scan-line. If the visited point is associated with the first edge in this ordered list, then it can be reported. Otherwise, it must be obscured by some other edge appearing before it (with respect to the center) and so would not be reported. Of course, the edge list must handle inserts and deletes in $O(\log n)$ time which means using optimal sorting (of which many are available).

Figure 1 shows the intuitive idea. The edge-list here would be $\{5, 2, 1, 4, 3\}$ – the order of intersecting edges from the center along the scan-line. Of course, in reality, the scan-line only stops at vertices (not in the middle of edges).

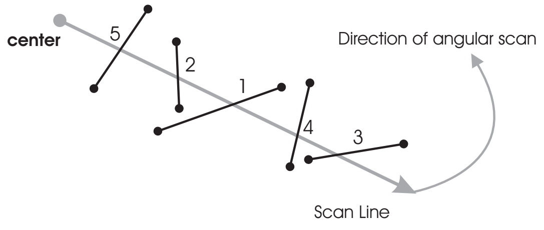

Figure 1. Example of Lee Scan with Edge-List.

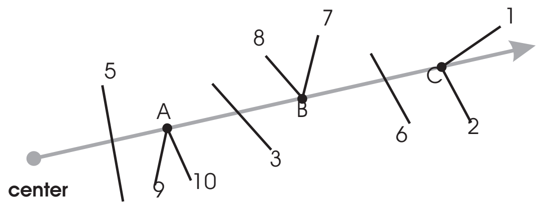

Figure 2. Basic Cases in the Lee Scan.

What happens at each vertex visit depends on the polygonal edges associated with that vertex. There may be two inserts, two deletes, or one insert and one delete. Figure 2 shows these situations with vertices marked a, b, and c and edges marked 1-10. Collinear points are handled as follows: if several points lie along the same scan-line, the order is determined by the distance from the center. In figure 2, vertex a would be visited, followed by vertex b, and then c. Before a is visited, the edge-list would be $\{5, 9, 10, 3, 6, 2\}$. When a is handled, both its edges are deleted, so the edge-list would be $\{5, 3, 6, 2\}$ afterwards. When b is handled, both its edges are inserted, so the edge-list becomes $\{5, 3, 8, 7, 6, 2\}$. When c is handled, one edge is deleted and the other is inserted. The edge-list at the end would be $\{5, 3, 8, 7, 6, 1\}$.

Time analysis: there are $(n-1)$ vertices to be visited for each of n centers. At each of these steps, it takes $O(\log n)$ for the search/insert/delete, thus making the time for one scan $(n-1)O(\log n) = O(n \log n)$. The time for all n scans would then be $nO(n \log n) = O(n^2 \log n)$.

Space analysis: in the worst case, there may be $O(n)$ edges in the edge-list at any one given time, but no more. The angularly sorted list also requires $O(n)$ storage during one scan, but can be freed after the particular center has completed. Of course, in order to store the visibility graph, it takes $O(|e|)$ space.

4.1.2 Construction of the 3D Visibility Graph

Theorem 1 *The 3D visibility graph defined above can be constructed in $O(n^3 \log n)$ where n is the number of objects of the scene.*

Proof 1 *For each pair of objects O_1 and O_2, sort all the planes which are tangent O_1 and O_2 and one other object in angular order with respect to a fixed plane tangent to O_1 and O_2. This is equivalent to sweeping the space with a rotating plane while remaining tangent to O_1 and O_2 and visiting each of the objects when the sweep-plane becomes tangent to it. We keep track of the order of intersected objects made by the sweep-plane. Like Lee's algorithm, we maintain an ordered list which has current tri-tangents. When a new tri-tangent plane appears, we check it with other tri-tangents in the list and we insert or remove it from the list.*

The time needed for sorting the tangent planes is $O(n \log n)$, as there are $O(n)$ of these planes. Each of $O(n)$ inserting and removing operations takes $O(\log n)$ time. So, each sweeping process takes $O(n \log n)$ time. As there are $O(n^2)$ pairs of objects, the total algorithms takes $O(n^3 \log n)$ time.

Although the above algorithm is not naive, it is not optimum. We note that it might be possible to improve the running time to $O(k + n^2 \log n)$, where k, $O(n^3)$, is the number of visibility edges of pseudo-graph, by using pseudo-triangulation based on the work of [8].

4.2 Application of 3D visibility Graph

We believe that this pseudo-graph has most of the properties that visibility graph is reputed for. For example, consider the convex hull of a set of 3D objects. As we know, in 2D, the convex hull of a set of objects lies on the visibility graph of these objects. It can be easily shown that the same property holds for the 3D visibility graph as well.

5 Parameterization

We use a parameterization which maps a plane in 3D space to a point in $\mathbf{R}^3$.

Note that this parameterization will be used mainly for illustration purposes. The relations we study could be expressed using only topological notions, although we believe that visualizing in $\mathbf{R}^3$ greatly helps their understanding.

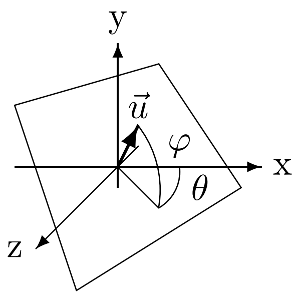

Figure 3. Plane parameterization using two spherical angles for the direction and the distance of plane from origin.

We map a plane p of the scene to a point $p^\star$ in dual space. As every plane in 3D space has three degree of freedom to move in the space, the dual space is 3D. Consider the line perpendicular to p that passes throw origin (Figure 3). Let (θ, φ) be the spherical coordinates of the direction vector of this line, and u be the distant of the plane from origin. We map the plane p to the point (θ, φ, u) in dual space. Our mapping is thus into $[0, 2\pi] \times [-\pi/2, \pi/2] \times \mathbf{R}^2$.

6 Partial Visibility Complex

Base on our definitions of visibility, we introduce a structure which can be thought as the extension of visibility complex for 3D scenes. We define this structure, which we call partial visibility complex, for basic states, then we present a complete definition for it and propose an algorithm for its construction.

Consider a sphere and a plane in space and assume that the plane can move continously in space. In this case the boundary between times that the plane intersects with the sphere and times that it does not, is when the plane is tangent to the sphere. This gives the idea that we can compute the boundary in dual space and partition the planes of space according to their intersection status with the sphere.

In what follows, we set the direction of a plane to be the direction of line perpendicular to it and we show it by its spherical coordinates (θ, φ).

For each direction (θ, φ) in 3D space, there are two planes $(\theta, \varphi, \lambda(\theta, \varphi))$ and $(\theta, \varphi, \mu(\theta, \varphi))$ which are tangent to the object.

If we represent the planes in a dual space, for each (θ, φ), a given object has two tangents $(\theta, \varphi, \lambda(\theta, \varphi))$ and $(\theta, \varphi, \mu(\theta, \varphi))$. $\lambda(\theta, \varphi)$ and $\mu(\theta, \varphi)$ describe two surfaces in the dual space. Each plane (θ, φ, u) such that $\lambda(\theta, \varphi) < u < \mu(\theta, \varphi)$ crosses the object. For a scene of objects, these surfaces partition the dual space into connected components corresponding to planes intersecting the same objects. we call this partition the dual arrangement.

As we mentioned, the set of planes tangent to one object is a 2-D set in the three-dimensional plane space. This means, more intuitively, that a plane has 2 degrees of freedom while staying tangent to one object. We will call the set of planes tangent to an object the tangency surfaces of that object.

Visualizing 3D space is hard. One approach is to use slices or cross sections. In this paper we will fix $\varphi =$ ct. Such a slice will be called a φ-slice. We will obtain a 2D slice where only θ and u vary, composed of all the planes that have φ as their second parameter of their direction. Such a slice will be called a φ-slice. These two-dimensional φ-slices are easier to visualize. They justify in part the choice of the parameterization because they can be interpreted as orthographic projections of the scene.

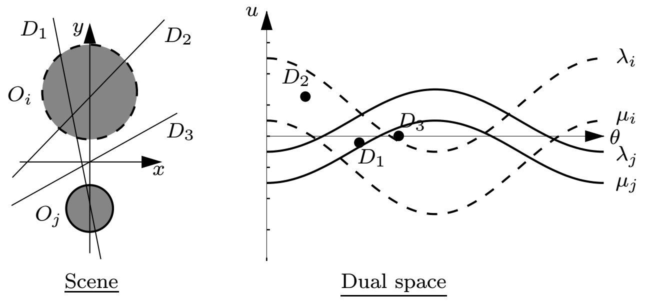

Figure 4. A slice form scene and dual space [6].

Figure 4 shows a z-slice of the scene and a φ-slice of the dual space. The scene is composed of two objects O_i and O_j and three planes D_1, D_2 and D_3. We assumed that that the dual of these planes lies in our slice (they have the same φ direction parameter). As we can see, the dual of D_1 that intersects both objects lies in the tangency surfaces of both of them, and D_3 that does not intersect the objects, is outside the tangency surfaces. We remind that the domain of θ is $[0, 2\pi]$ and the domain of φ is $[-\pi/2, \pi/2]$. In this case, the intersection of the tangency surfaces of two objects form a one-dimensional curve that represents the planes that are tangents to both objects.

If there are three objects in the scene, there will be a partition similar to above, in which the tangency surfaces of the objects have intersection in some points (at most eight points). Each of these points corresponds a plane tangent to the three objects.

We can see from above cases that the tangency surfaces of objects, partition the dual space to continuous sections. For a plane, belonging to such a section determines the objects that it crosses.

Now we give a formal definition of partial visibility complex. Consider a collection $\mathcal{O}$ of pairwise disjoint objects in the space. We limit ourselves to convex objects and add an infinite "blue sky" object to the scene for the sake of coherence. We consider the set of maximal free pseudo-segments, which are pseudo-segments that do not intersect with any objects except at there vertices. We define the partial visibility complex as partition of these pseudo-segments according to the objects they touch.

Like visibility complex, partial visibility complex is composed of several elements. These elements are 0 to 4 dimensional, and in the increasing order of dimension are: vertex, 0-D elements that correspond to planes tangent to three objects; edge, 1-D elements that correspond to a set of planes that are tangent to two objects and touch another object; face, 2-D elements that correspond to a set of planes that are tangent to a plane and touch two other planes; and volume, 3-D elements, that correspond to a set of planes that cross three objects.

In dual space, each type of elements is incident to, or delimited by a number of other elements:

- A vertex is incident to six edges and twelve faces.
- An edge is delimited by two vertices.
- A face is delimited by four vertices and incident to four edges.
- A volume is delimited by two chains of vertices, edges and faces.

For example, for the first statement, consider three objects A, B, and C, and their mutual tangent plane p. We can see the number of incident edges to this vertex in dual space (the plane p) by the following argument: If p remains tangent to A and B and move it, we get two edges, one is the set of planes that are tangent to A and B and intersect with C, and another one is the set of planes that are tangent to A and B and do not intersect with C. With similar argument for remaining tangent to B and C, and to A and C, we can see that there are six edges incident to p. For the third statement, As a face is the set of planes that are tangent to an object and cross two other objects, for each of the two objects, there can be two type of tangent planes (above and below the objects), and each of them gives an edge.

6.1 Structure

The size of the partial visibility complex is characterized by the number k of vertices which is $\Omega(n)$ and $O(n^3)$ (with n the number of convex objects). As we can see, the size of partial complex is equal to the size of 3D visibility graph of the scene.

As said, partial visibility complex is composed of vertices, edges, faces, and volumes. Each vertex have pointers to objects (at least three) that is tangent to them. Also it has six pointers to its incident edges and two pointers to its incident volumes. Each edge has two pointers to objects that is tangent to them, one pointer to its crossing object, and some pointers to its incident elements. We can similarly obtain the structure of faces and volumes. We ought to mention that partial visibility complex, like visibility complex is a topological structure.

The construction of the partial complex can be achieved by a sweep of its vertices. When a vertex is swept, the relations between the edges and the faces have to be updated. We can use the construction algorithm for 3D visibility graph and sweep the vertices of partial complex in $O(n^3 \log n)$. As the relation between different element types are $O(1)$, we can build the partial visibility complex in $O(n^3 \log n)$.

7 Applications in Visibility Computations

Computing the view around a viewpoint amounts to computing changes of visibility occurring along a ray pivoting around the viewpoint. Because of its applications, computing the view has been widely studied for various scenes. In 3D space, the proposed algorithms are mainly for graphical computation of view and are designed to fulfill special needs. A overview of these algorithms is given in [3]. In [2] a method for computing the view with 3D visibility complex is given, which is unpractical due to its complications.

In this section, we present an algorithm for computing the view form a viewpoint with partial visibility complex. Unless otherwise stated, by visibility relations we mean classical definition of them.

We assume that the viewpoint is outside the convex hull of the objects. Consider a plane p passing throw the viewpoint that do not cross any objects. We call this plane the sweeping plane. If we rotate this plane around a line passing throw the viewpoint, their corresponding points in dual space form a curve p_v. This curve will cross with the partial visibility complex of the scene at some points. Crossing the partial complex with a face corresponds to a plane tangent to an object, crossing with an edge (intersection of two faces) mean its tangent to two objects, and finally, crossing with a vertex (intersection of three faces) corresponding to tri-tangent plane.

While p is rotating, the cross sections of objects with it form a dynamic shape in it. Having the 2D visibility complex of this cross sections we can compute the view around viewpoint in the plane (remember that the viewpoint is always inside p). On the other hand, while sweeping, if the topological structure of cross sections do not change, its visibility complex and view will not change too. We can conclude that to compute the view around the viewpoint, we just have to maintain the visibility complex inside the sweeping plane, and upon each change in it, update the view as well.

As we said, while the sweeping do not cross any face (similarly, edge and vertex) of partial complex, the number of crossing objects with it remains constant and maintaining the view while sweeping is equal to maintaining the view in a plane with moving objects. This can be done in each visibility event at $O(\log m)$, which m is the number of crossing objects with the sweeping plane [9]

Continuing the rotation of the sweeping plane, the curve p_v will cross the partial complex in some point. If p_v crosses i faces ($i = 1, 2, 3$, corresponding to face, edge, and vertex, respectively), it means that k objects start crossing the sweep plane and l objects stop their crossing ($k + l = i$). Starting crossing of an object means that we must put a new object in 2D visibility complex. We can do this in $O(v \log v)$ time where v is the view size [9]. When an objects finishes its crossing with the sweeping plane, its corresponding faces in 2D visibility complex must be deleted. This deletion can be done in $O(v)$, where v is, again, the view size in visibility complex.

The time complexity of the described algorithm is as follows. We assume that the partial visibility complex of the scene is computed in preprocessing time. The initial 2D visibility complex can be built in $O(k + n \log n)$ [7], where k is $O(n^2)$.

Firstly, we consider the changes in crossing points of p_v with partial complex. The number of these events are $O(n)$ which we can store them in a queue with $O(n \log n)$ cost. Each object starts and stops intersecting with the sweeping plane exactly once, and the size of view in cross sections is determined by the number of objects intersecting the plane at that moment. So, the time needed to process these events in $O(n^2 \log n)$ in total.

We now analyze the time needed for maintaining the 2D visibility complex and its view when the number of intersecting objects with the sweeping plane is constant. Consider a segment of p_v that does not cross the partial complex except at its endpoints, and assume that the number of crossing object in this part of p_v is m. The visibility complex of cross sections of objects in sweeping plane or the view in it only changes when a bitangent (a line that is tangent to two objects) becomes tangent to another object (changing in the 2D visibility complex), or the tangent line from viewpoint to an object becomes tangent to another object (changing in the view). The number of bitangents in the sweeping plane is $O(m^2)$ and the number of tangents from the viewpoint is $O(m)$. As the sweep plane rotates, the cross section of an object makes at most $O(n)$ bitangents with other cross sections in the plane. As we assumed that objects are convex and disjoint, these bitangents can cross other objects at most $O(n)$ times. For all objects, the number of these events is $O(n^2)$. At each such event, the topological structure of visibility complex changes and must be updated. The cost of each update is $O(\log m)$ [9] and the total cost will be $O(n^2 \log n)$. As said, another event type is when a tangent line from view point to a cross section of an object becomes tangent to cross section of another object. In this case the visibility of crossing object from viewpoint is changed. Because in each of these events, the view will change, the number of these events is in the order of view size. The cost of updating the view in visibility complex in these events is $O(\log m)$ [9].

Overall, we conclude that, if we denote the size of view from a viewpoint q by $|V(q)|$, the output sensitive cost of our proposed algorithm is $O((|V(q)| + n^2) \log n)$.

8 Conclusion

In this paper, we proposed a new model for defining visibility relations in 3D. This model is a new step towards obtaining a generalized model for visibility graphs in three (or even higher) dimensions. One of the properties that is checked to be satisfied in the new model is that the convex hull of the 3D objects lie on the 3D visibility graph of the objects, as in the 2D case. We presented an algorithm for its construction in time $O(n^3 \log n)$. We believe that 3D visibility graph can be used in many 3D geometric problems.

We also defined an extension of visibility complex for 3D space, that, while reserving the definitions of visibility in 2D, reduce the computation times and makes it applicable to visibility computations. Unlike the 2D visibility complex, the partial visibility complex is based on tangent planes and the concept of visibility based on to them. The size of this structure for n convex objects in 3D space is $O(n^3)$ and can be built in $O(n^3 \log n)$. We proposed an algorithm for its construction.

The proposed structures can be used in visibility computations. As an example, we showed that the view around a viewpoint q can be computed in $O((|V(q)| + n^2) \log n)$ time.

As mentioned in section 4.1, it may be possible to improve the running time of computing the 3D visibility graph, and as a result, the partial visibility complex, to $O(k+n^2 \log n)$, where k is the number of visibility edges of pseudo-graph, by using pseudo-triangulation based on the work of [8].

References

[1] J. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *Proc. IEEE Symp. on Foundations of Computer Science*, 1987.

[2] D. G. Durand, F. and C. Puech. The 3d visibility complex. In *ACM Transactions on Graphicss*, pages 176–206. ACM Press, April 2002.

[3] F. Durand. *3D Visibility: Analytical Study and Applications*. PhD thesis, University of Joseph Fourier, 1997.

[4] D. T. Lee. Proximity and reachability in the plane. Technical report, University of Illinois at Urbana-Champaign, 1978.

[5] J. S. B. Mitchell and M. Sharir. New results on shortest paths in three dimensions. In *Annual Symposium on Computational Geometry*, pages 124 – 133, 2004.

[6] R. Orti, F. Durand, S. Rivière, and C. Puech. Using the visibility complex for radiosity computation. In *Applied Computational Geometry (ACM Workshop on Applied Computational Geometry, Philadelphia)*, pages 177–190, May 1996.

[7] M. Pocchiola and G. Vegter. The visibility complex. In *Proceedings of the Ninth Annual Symposium on Computational Geometry)*, pages 328 – 337, 1993.

[8] M. Pocchiola and G. Vegter. Computing the visibility graph via pseudo-triangulation. In *Proceedings of the 11th Annual ACM Symposium on Computation Geometry, Vancouver)*, pages 248 – 257, May 1995.

[9] S. Rivière. Dynamic visibility in polygonal scenes with the visibility complex. In *13th Annual ACM Symposium on Computational Geometry*, 1997. http://www.