Kinetic Visibility

Henry Xiao
School of Computing, Queen's University
Kingston, Ontario, Canada K7L 3N6
E-mail: xiao@cs.queensu.ca

June, 2007

Contents

Abstract

In this paper, we survey kinetic visibility problems. Unlike the research in traditional visibility problems, the researchers have also taken moving objects into consideration. These problems have been addressed from different practical aspects, such as computational graphics and robotic design. Over the years, some general combinatorial results of the visibility graph have been shown. Leveraging temporal coherence has been used in kinetic visibility studies for different scenes. Data structures that have been developed to support the kinetic operations, and kinetic algorithms that have been designed to achieve efficient visibility computation and maintanence are reviewed in this paper. Open problems related with kinetic visibility will be addressed from different perspectives. Both theoretical and practical work needs to be done in future studies.

Glossary

backward view object	the object that intersects the starting point of a directed maximal segment, 6
bitangent	a line segment that is tangent to two objects, 6
boundary segment	a free line segment from the viewpoint that is tangent to an object in the object space, 18
certificates	a set of simple geometric attributes that is chosen to validate the combinatorial structures of the attribute that we want to maintain in a KDS, 12
chipped line	a line through two points in the plane excluding the line segment between the two points, 16
chipped line arrangement (CL-arrangement)	a line arrangement constructed by drawing a chipped line through each pair of points in the plane, 16
edge (visibility complex)	a line segment connecting two vertices in the quotient space of the visibility complex that corresponds to a group of maximal segments in the original object space which are tangents to one object and have the same forward and backward view objects, 11
event	a failure of a certificate in a KDS, 12
external events	the set of events that affects the attribute of interest in a KDS, 14

face (visibility complex)	a set of all points in the quotient space of the visibility complex bounded by some vertices and edges that corresponds to a set of all maximal segments in the original object space which can be transformed into each other while keeping their forward and backward view objects, 11
forward view object	the object that intersects the ending point of a directed maximal segment, 6
gerenal position	no three objects share a common tangent line, 7
internal events	the set of events that is used to maintain the correctness of the certificates, however does not affect the attribute of interest in a KDS, 14
kinetic data structure (KDS)	a data structure that maintains an attribute of interest in a system of geometric objects undergoing continuous motion, 12
kinetic scene	a scene where every object may have its own moving trajectory in the object space, 8
kinetization	the process of taking an algorithm for computing a discrete attribute, turning it into a proof that this attribute is correct, and animating this proof through time, 13
maximal segment	a line segment in the object space that only intersects two object boundaries at its two end points, 6
object space	the space (in 2D or 3D) that contains the set of objects, 6
polyhedral complex	a finite set of closed convex polytopes, the faces of the set, in real n-space R^n, such that two conditions are satisfied: all faces of the polytopes are included; the intersection between two polytopes is either empty or is a face of both polytopes, 10
pseudo-triangle	a simple ploygon in the plane with exactly three convex vertices, 7
pseudo-triangulation	a tiling of a planar region into pseudo-triangles, 7

smooth	a well-defined tangent line through each boundary point, 7
static scene	a scene where the objects do not move except the observer, 8
upper envelope	the portion of the lines in an arrangement that is visible from the point $(0, +\infty)$, 13
vertex (visibility complex)	a point in the quotient space of the visibility complex that corresponds to a bitangent (maximal segment) in the original object space, 11
visibility complex	a mathematical structure (or a polyhedral complex) applied as a data structure that naturally captures visibility in a set of convex objects, 9
visibility graph	the set of all free bitangents of the objects in the object space, 6
visibility polygon	a polygon formed by all points of the plane directly visible to the observer (viewpoint), 6

1 Introduction

Imagine a group of intelligent robots working together on a large open ground in the near future. Each of the robots is equipped with a radar of “infinite” power, such that they can “see” others as long as there is no other robot blocking their sight. Now, just like us, the robots need to have the ability to organize themselves to accomplish a task together. Each of them needs to know which ones they can see at any time. This kind of teamwork is essential, because they need to ask others for help (again, just like us) and, of course, they do not want to bump into each other. In fact, we are grouped together to work like this all the time.

Intelligent robots need algorithms to help them keep track of other robots that are visible. In the field of computational geometry, we know that this problem is related to the visibility computation. Like all other algorithmic fields, we need efficient algorithms to help the robots calculate and update their visibility information in real time to avoid possible collisions. In the theoretical world, we usually start from the simplest case, where the robots stand still without any movement (like base stations). Decades of studies have provided us some good algorithms to calculate the visibility information in this case. However, we want the robots to have the ability to move around so that they can be greater helpers to us. The challenge is not only to calculate the visibility information, but also to maintain the visibility information with the respect of motion. A good algorithm shall provide the robot quick update ability when detecting any change in visibility. Of course, it is not hard to see that recalculating the visibility information from scratch whenever there is a change is not efficient at all.

Before we get into the formality of examining the current literature, let’s step back and take an informal look at how we (humans) process the visibility information in our daily life. Instead of a radar and a computer, we have our two eyes and a brain, which comprise our visual system. This system appears to take advantage of temporal coherence in absorbing visual information [32]. Under most circumstances, the scene changes relatively slowly with the respect to our movements, such that if we take two close snapshots on our continuous moving path, they would be very much the same. If we further restrict the question to report what things (objects) we can see, then the visibility information only needs to be updated at certain points where an object is blocked (occluded) or becomes visible to us. This is rather the case for the robots in the scenario proposed at the beginning of this section.

Clearly, in designing visibility algorithms for the future intelligent robot, we need to utilize coherence information to achieve efficiency just like our visual system does. In this paper, we take a look at current developments in deriving efficient algorithms to leverage temporal coherence in visibility computations. This topic is recognized as "kinetic visibility" to distinguish it from the traditional visibility problems where we do not consider motion. Related questions such as collision detection and room searching are also studied in this framework. We classify this framework into two categories. We start from the simple version where only one supervisor robot is moving around to check on the others, such that the environment for the supervisor is static. Then we continue to a more general problem where there is no supervisor, and every robot is trying to coordinate with others. The scene for any robot is dynamic (kinetic)[1].

We consider various formally defined models of robots and their working spaces, so that we can tackle these problems using algorithmic and analytic tools from the field of computational geometry. The working space is mainly addressed in 2D from the current developments in this field. The robot models that we consider in 2D are:

- a point set (i.e., imagine the working ground is "infinitely" big, such that a robot can be modeled as a point in the space.)
- convex polygons
- simple polygons (i.e., some vertices of the polygon may not be convex)

In 3D, we are expecting to use some well-defined geometric objects (e.g., balls or convex polytopes) as the robot models in the future studies.

We formalise the problems in Chapter 2 with technical definitions. Related data structures are discussed in Chapter 3, and algorithms are discussed in Chapter 4 and Chapter 5. Open problems are surveyed in Chapter 6.

2 Problem Statements

In this section, we formalise the visibility problems introduced by the intelligent robot above. Visibility problems in computational geometry have been intensively studied for over 30 years. The research is primarily focused on combinatorial issues, or algorithms involving problems like the art gallery theorems, illumination of convex sets, and mirror reflections. O'Rourke has provided a broad survey on many traditional visibility problems in [30] (Chapter 28). However, the set

[1]Strictly speaking, in computational geometry, "dynamic" has already been used for the set of problems where we can add/delete objects to/from the scene [9]. For the sake of clarification, we will use "kinetic" throughout this paper instead.

of problems proposed by the intelligent robot have taken continuous motion into account. Many practical problems in computational graphics and robotics share the same feature.

2.1 Technical Terms

We begin with the definitions of some common terms used in this context. Following the conventions, we also make some assumptions throughout this paper unless explicitly specified. Each robot is generalized as an object. The object space is in 2D or 3D[2], which contains the set of objects. Objects are assumed to be convex. For the ease of notation, the object space is presented in a graphic box (can be treated as another object). The boundary of the box denotes the "blue sky". The observer is an object in the object space (referred to as *viewpoint* if the observer is a point). In kinetic visibility problems, object motion is assumed to be continuous, distinguished from object motion in the dynamic set of problems concluded in [9], in which objects can be deleted or added discretely from the object space.

2.1.1 Visibility

In computational geometry, we say an object O_1 is *visible* to another object O_2 if there exists a straight line segment in the object space of O_1 and O_2 that only intersects object O_1 and O_2 at its two endpoints and intersects no other objects. The *visibility graph* of the scene is defined as a vertex-edge incident graph, such that each object corresponds to a vertex, and two vertices are connected by an edge if only if the two objects defining the vertices are visible to each other. In dealing with convex objects, a more commonly used visibility graph is defined on *free bitangents*, which are line segments that are tangents to two objects at their endpoints and intersect no other objects. The (tangent) visibility graph thus contains all free bitangents of the set of objects. We will use this visibility graph throughout this paper. Figure 1 (a) depicts the visibility graph of three objects. Note that the bitangent drawn with a dashed line is not free, and thus not included in the visibility graph.

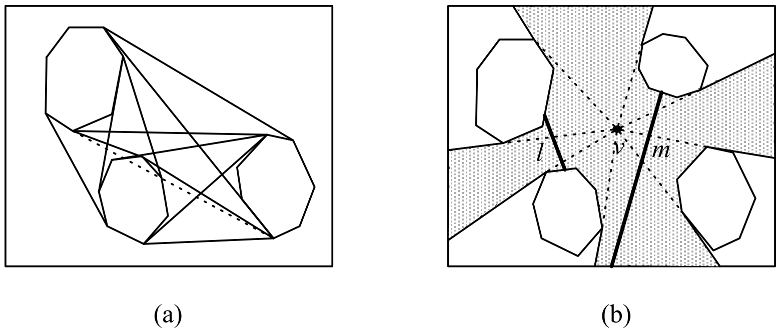

Figure 1: (a) the (tangent) visibility graph of a collection of three objects. Note that the dashed bitangent is not free, and thus not included in the visibility graph. (b) the visibility polygon (shaded area) from the viewpoint v. l and m are maximal segments in this scene.

A large set of visibility problems considers the view from an observer. The view is normally described as the visibility polygon (or visibility region in 3D). The *visibility polygon* is normally

[2]Technical definitions are addressed in 2D unless otherwise specified.

a polygon formed by all points of the plane directly visible to the observer; that is, those points that can be connected via a line segment to the observer without intersecting any object interior. The visibility region can be defined similarly. Figure 1 (b) shows the visibility polygon from the viewpoint v in the scene. Alternatively, instead of describing the exact geometry of the view, we can keep a combinatorial description of the view, which only consists of a circular ordered list of visible objects (or vertices and edges) in the order in which they appear on the boundary of the visibility polygon from the observer. This is beneficial in solving kinetic visibility problems as we shall see later. Of course, having this combinatorial description, it is easy to construct the exact visibility polygon. A related important definition is the *maximal segment*, which is a line segment in the object space that only intersects two object boundaries (or the blue sky) at its two end points. Assuming line segments are oriented, the *forward view object* of a maximal segment is the object that intersects the ending point of the directed line segment, and the *backward view object* is the object that intersects the starting point of the directed line segment. We can thus categorize a set of maximal segments by their forward and backward view objects. Line segment l and m in Figure 1 (b) are both maximal segments, where m "intersects" the blue sky (i.e., the free space) at one end. This definition motivates later developments of the visibility complex data structure.

Note that when taking bitangents of a set of convex objects, the objects are normally assumed to be *smooth* (i.e., there is a well-defined tangent line through each boundary point). This can be guaranteed by taking the Minkowski sum of the objects with an infinitesimally small circle. The *general position* (i.e, no three objects share a common tangent line) is also assumed, which can be achieved by applying some perturbation technique in practice. The following figures are drawn based on these two assumptions.

2.1.2 Pesudo-triangulation

Pocchiola and Vegter introduced a space partition method named pseudo-triangulation [47]. A *pseudo-triangulation* of a set of objects is the subdivision of the plane induced by a maximal (with respect to inclusion) family of pairwise non-crossing free bitangents [46]. A *pseudo-triangle* is then simply a connected subset T of the plane, such that (1) the boundary of T, ∂T, consists of three convex curves, with each two curves sharing a tangent at their common endpoints, and (2) T is contained in the triangle formed by the three endpoints (defined as *cusps*) of these convex chains. A pseudo-triangle is thus a simple polygon with three angles less than π. Figure 2 depicts two pseudo-triangulations of a collection of five objects. Same as the triangulation (e.g., on a set of points), it is easy to see that there are many possible ways to pseudo-triangulate a given set of objects. We also know that the bounded free regions of any pseudo-triangulation are pseudo-triangles.

Pseudo-triangulation is a powerful data structure in the computational geometry field. It has been applied to support various algorithms for different problems, such as in rigidity calculations [54] [31]. Different combinatorial results on pseudo-triangulation have been exposed. The number of pseudo-triangles (of a pseudo-triangulation of a collection of n objects) is $2n-2$, and the number of bitangents is $3n-3$ [47] [46]. The nice thing about this partition is that it has been shown we can easily construct a pseudo-triangulation by adding free bitangents in a "greedy" fashion considering some topological order based on the angles of the bitangents, such that the pseudo-triangles in the partition admit some partial order from any non-horizontal direction (see Figure 2). The resulting algorithm is called the Greedy Flip Algorithm [46] [8], which is essential to data stuctures and algorithms built on top of the pseudo-triangulation. Rote, Santos and Streinu recently provided a comprehensive survey on this subject and its various combinatorial

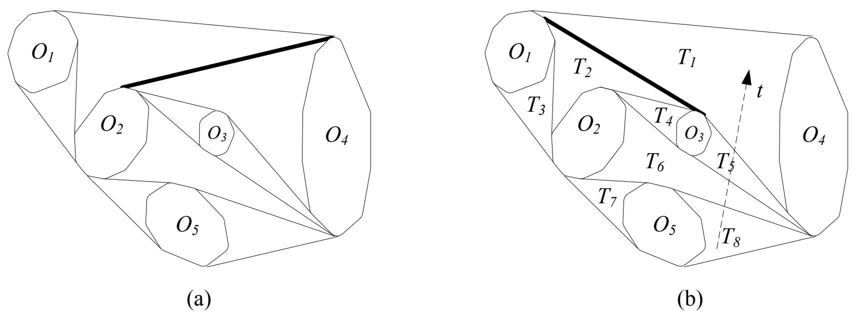

Figure 2: Two pseudo-triangulations of a given scene with five objects. The pseudo-triangles admit a partial order (as labelled) in (b) such that for any non-horizontal, free line segment t (intersects no object) in the object space, the sequence of pseudo-triangles intersected by t is ordered according to increasing y-coordinate.

properties [51].

2.2 Kinetic Visibility Problem with Static Scene

As in the case of traditional visibility problems, if the objects stand still in the object space, we call such a scene static. Considering static scenes in kinetic visiblity problems helps to simplify the underlying environment and isolate the motion to a single observer (or viewpoint). In our supervisor robot senario, only the supervisor travels around, and other robots are mounted on the ground. The supervisor needs to maintain its "sight" when it moves around. The environment to the supervisor is a static scene. The supervisor is a single observer in this case.

Many practical problems fit into this category. In computational graphics, we consider the problem of retrieving some visual space (e.g., a room). The objects in the space are static, and we walk into the space as an observer. In the mobile world, the base stations (access points) are normally fixed. A mobile device can be carried remotely as long as it can communicate with some base stations. In order to build communication channels, the mobile device may need to know which stations are "visible" to it. In recent developments on wireless sensor networks [59], each sensor has wireless communication capability and some level of intelligence for signal processing and networking of the data. Information is collected through the data received from each sensor, which has limited transmission range. In order to gather data from all the sensors, each sensor needs to have the ability to forward the data. Certain algorithmic questions can be addressed in the kinetic visibility field, such as how to get data from the sensors under different topologies. Fekete and Kröller in their recent study [26] also pointed out that it is essential to make use of the underlying geometry of wireless sensor networks to achieve distributed knowledge of global network properties with a limited amount of strictly local information among the sensors.

2.3 Kinetic Visibility Problem with Kinetic Scene

A more general set of problems that are considered in kinetic visibility has no restriction on which object(s) move. Every object in the object space may have its continuous moving trajectory. We call this environment a kinetic scene. Under kinetic scenes, we can look at kinetic visibility problems from different aspects. From a particular object point of view, it now lives in a kinetic environment. Recalling our robot example, if every robot is moving around, the supervisor has to take other robots' motion into calculation in order to know which ones are visible. Obviously, this is a harder problem than the previous senario with a static scene. If there does not exist a supervisor among the set of robots, every robot instructs itself. Then as we have discussed, everyone needs to be self-disciplined in the kinetic scene. An important task is to avoid collision. Indeed, the popular collision detection problems have been studied in this context as well.

In mobile ad hoc network, the network consists of a set of self-configuring mobile routers (and associated hosts) connected by wireless links, the union of which form an arbitrary topology. The routers are free to move randomly and organize themselves arbitrarily. The network's wireless topology may change rapidly and unpredictably. Various questions related with mobile ad hoc network are being studied currently [56]. From the algorithmic point of view, distributed routing algorithms for the mobile devices [17] and network coverage problems [41], where localized algorithms are needed to calculate the best path from one device to another, can be addressed in relation to kinetic visibility as well.

In the following sections, we survey some important developments in kinetic visibility starting from the data stuctures. Related algorithms are introduced and some results are reviewed. Limited by the space of this paper, we do not get into details of any particular data stucture or algorithm, but rather leave it to the original literature for the reader who is interested. We conclude with open questions and some future plans for research in this subject.

3 Data Structures

Similar to the dynamic computational geometry problems discussed by Atallah [9], to solve kinetic visibility problems, we need some supporting data structures. The difficulty of developing efficient data structures for kinetic operations is that the continous motion in this set of problems implies that some global computation is hard to avoid. Two important data structure developments (visibility complex and kinetic data structures) reviewed in this section are both built on the idea of localizing the visibility recalculation triggered by continous motion to achieve efficiency. In order to do so, we need to utilize the combinatorial stuctures of the visibility graph and take advantage of temporal coherence. In other words, instead of examining the exact visibility graph, we shall only alter the data structure for a combinatorial change (e.g., an object gets blocked or becomes visible). This also provides a natural way of leveraging temporal coherence, because not every "move" yields a combinatorial change of the visibility graph.

More or less, because the combinatorial properties of the visibility graph are hard to explore, the data structures for this set of problems are complicated. The complexity of understanding those data structures is thus high.

3.1 Visibility Complex

To overcome this difficulty and identify the combinatorial structures of the visibility graph in a given scene, Pocchiola and Vegter introduced the *visibility complex* [47], which provides some richer structures through mapping the original scene onto a quotient space. The complex is then built on some partition of the maximal segments according to the segments' views. Because a comprehensive catalogue of relationships between mutually visible objects is encoded in the visibility complex, visibility maintenance in the complex is straightforward as the objects move [50]. An optimal approach[3] using the greedy flip algorithm with linear space usage to construct the visibility complex through pseudo-triangulations is proposed by Pocchiola and Vegter as well [46]. The resulting algorithm matches the optimal time to construct the visibility graph for a set of convex objects [29] (i.e., $\Omega(n \log n + k)$, where n is the number of objects, and k is the size of the visibility graph). Thus, preprocessing the scene into a visibility complex provides numerous combinatorial properties to build efficient kinetic algorithms dealing with moving objects.

The formal definition of the visibility complex is very much involved. For the purpose of this survey, we give an intuitive explanation of the construction of the visibility complex here based on a simplified implementation in the CGAL library [7] by Angelier and Pocchiola, and on Rote, Santos and Streinu's survey of pseudo-triangulation [51]. A *ray* in the object space is a pair (p, α) consisting of a point p in free space and a direction $\alpha \in \mathbb{S}^1$ (the 1-sphere[4]). The visibility complex of a constrained scene S_H is the quotient space of the set of rays with respect to the following equivalence relation $\sim$: $(p, \alpha) \sim (q, \beta)$ if and only if both α and β are the direction of the line (p, q), and the segment $[p, q]$ lies in the free space of S_H. The orgins of the rays belonging to a same equivalence class under the $\sim$ relation define a maximal segment in the object space. The visibility complex of a 2D scene is a mathematical structure (or a polyhedral complex[5] concluded in Rote, Santos and Streinu's survey [51]). The vertices, edges, and faces of the visibility complex are depicted in Figure 3.

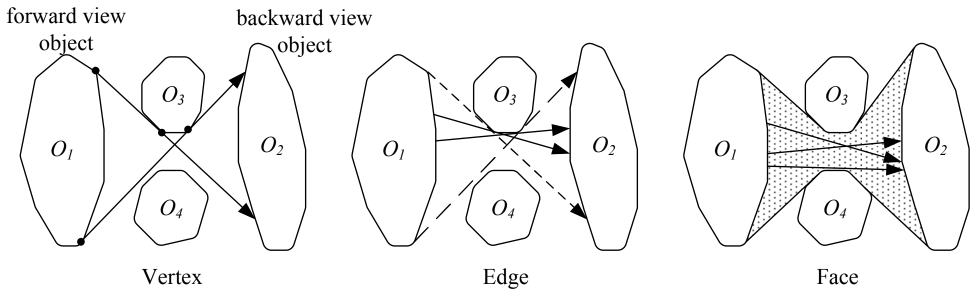

Figure 3: The vertex, edge and face are defined in the 2D visibility complex. Two vertices are defined by two bitangents to O_1 and O_3. An edge in between the two vertices corresponds to a group of maximal segments tangent to object O_3 and touching the same objects O_1 and O_2. A face is defined by the set of maximal segments with backward view object O_1 and forward view object O_2 in the shaded region.

- *Vertices*: a vertex corresponds to a maximal segment that is tangent to two objects. Each vertex thus associates with a bitangent.
- *Edges*: an edge corresponds to a group of maximal segments that tangent to one object. This group of maximal segments can be visualized by imagining a single maximal segment that rotates around the object boundary while keeping its forward and backward view objects.

[3]Under the assumption that the common tangents between two objects of constant size can be determined in $O(1)$ time.

[4]A 1-sphere is a circle of radius r.

[5]A polyhedral complex is a finite set of closed convex polytopes, the faces of the set, in real n-space R^n, such that two conditions are satisfied: all faces of the polytopes are included; the intersection between two polytopes is either empty or is a face of both polytopes. An example of a polyhedral complex is the set of all vertices, edges and two-dimensional faces of the standard three-dimensional cube. [6]

- *Faces*: a face is defined by a set of all maximal segments that can be transformed into each other while keeping their forward and backward view objects.

The visibility complex, regarded as a set of vertices, edges, and faces together with the incidence between them, forms an abstract polyhedral complex, which can be stored as a data structure. Under the general position assumption that no three objects share a common tangent, the visibility complex has a quite regular structure: every edge belongs to three faces, and every vertex belongs to four edges and six faces (see [47] for details).

The original visibility complex development suffers a critical drawback that the transformations taken to build this complex are complicated, and the whole picture of the complex is very difficult to be visualized. Pocchiola and Vegter showed that the 1-skeleton[6] of the visibility complex is isomorphic to the visibility graph [47]. Later, Durand and Puech came up a “dual arrangement” [22] to recognize this visibility complex from a different aspect that helps us to visualize its structure. Durand, Drettakis, and Puech then upgraded the visibility complex for 3D scenes [19], [20] following the same duality idea. The duality transform is demonstrated in Figure 4 adopted from the original paper [22] with two discs and a viewpoint, where (a) is the original scene, and (b) is the dual arrangement of the scene.

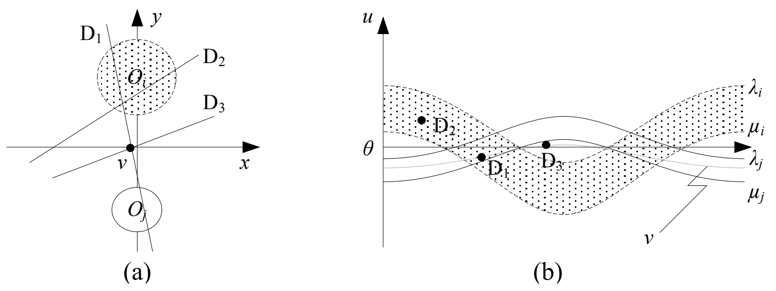

Figure 4: The dual arrangement of two discs (O_i and O_j) and a viewpoint v. (a) The scene with two discs. (b) The dual arrangement of the scene. The view around v is a sine curve in the dual space.

The mapping in Figure 4 is defined as: $-x\sin(\theta) + y\cos(\theta) = u \mapsto (\theta, u)$, such that a line in the original (primal) space is transformed into a point in the dual space, parameterized by the line's slope and the signed distance from origin. A given object has (for each θ) two tangents denoted by $(\theta, \lambda(\theta))$ and $(\theta, \mu(\theta))$. Each line (θ, u) with $\lambda(\theta) < u < \mu(\theta)$ intersects the object. According to the mapping, each point in the primal space is transformed into a sine curve. Each point on the sine curve represents a line in the primal that passes the original point. An disk is thus transformed into a strip bounded by two sine curves. For a set of discs, these strips partition the dual space into connected components corresponding to the lines in the object space. Lines from the viewpoint (v in the figure) are transformed into a sine curve in the dual space as well. For examples, line D_1 and D_3 (of v) correspond to two points on the sine curve of v in the dual. D_2 is not a line passing v, and thus not on the sine curve of v. Also, line D_1 intersects both O_i and O_j; the corresponding point of D_1 in the dual space is in a region defined by two object strips. To report the combinatorial changes (of visibility) from the viewpoint, we only need to check the sine curve of v in the dual space against different intersections defined with the object strips.

[6]The 1-skeleton of a polyhedral complex is the set of edges and vertices of the surface. For example, the 1-skeleton of a cube is the set of twelve edges connecting the corners.

The dual arrangement itself does not encode the occlusion information (i.e., the maximal segment). In order to construct the complete visibility complex, we need to add another dimension to recognize the maximal segment. So the visibility complex from the dual arrangement is constructed in a 3D surface structure. The algorithm for building the visibility complex based on the concepts outlined above, does not achieve the optimal time complexity (in fact, it runs in $O(k \log n)$, where n is the number of objects, and k is the size of the visibility graph [22]). However, it is straightforward to implement in practice. A couple of applications [45] [16] on this subject applied this algorithm to build the visibility complex.

The visibility complex provides a unified way to describe the combinatorial structure of the visibility graph. A set of rays can be recognized as a subset of the complex. Sweeping the ray around a viewpoint in the scene corresponds to "walking" along a trajectory in the visibility complex. Temporal coherence is nicely encoded in the complex cell structures. Rivière has shown how to use the visibility complex to maintain the view around a moving point in a static scene [50]. His algorithm takes the advantage that local changes of visibility in the scene only yield local recomputations in the complex. Rivière also worked out an algorithm for maintaining the visibility complex for kinetic scenes, and showed its usage in global visibility computations. This job has also been done with the dual arrangement [22]. It is noticeable that most visibility algorithms do not explicitly represent the entire visibility complex, due to the complex's large size. The linear size space usage proposed with the development of the visibility complex by its original authors comes from the pseudo-triangulation [46] [47]. Hall-Holt also came up with a linear size substructure of the visibility complex (i.e., the visible zone) that can be represented and maintained explicitly [32].

Applications of the visibility complex are currently addressed in 2D. Otri et al. utilized the visibility complex in the radiosity computation to calculate the global visibility in static scenes [45]. Cho and Forsyth developed a method of producing ray-traced images of 2D environments at interactive rates based on the visibility complex [16]. Hall-Holt took the visible zone substructure, and arrived the corner arc algorithm for efficient visibility computations in kinetic scenes [32]. The 3D extension of the algorithm has also been discussed.

The 3D visibility complex has been worked out theoretically with an introduction of a fourth dimension [19]. The complexity has again been a barrier to apply this data structure to a 3D scene. In 2D, only tangent and bitangent segments are considered, while in 3D, we need to consider segments tangent to four objects or three objects (special tritangents). Also, because the lines in general are no longer hyperplanes in 3D (i.e., separability fails), some convexity or monotonicity properties do not hold in 3D accordingly. As a consequence, we cannot perform efficient sweepline algorithms like we did in 2D. For these various difficulties, there has not been any practical work done directly applying the 3D visibility complex so far to our knowledge.

3.2 Kinetic Data Structures

Kinetic Data Structures (KDSs) are closely related to data structures used in event-based simulations. The goal of a KDS is to efficiently maintain a geometric attribute in a given kinetic scene through time. The key idea is to choose a set of simple geometric attributes as *certificates*, which validates the combinatorial structures of the attribute that we want to maintain. When the objects are moving, certificates sometimes fail signaling the combinatorial changes of the attribute. Through maintaining the correctness of the certificates, the attribute is maintained. The failure of a certificate is an *event*. Effective certificates are chosen based on leveraging temporal coherence in a given problem, which provides the advantage of keeping the changes to a partial structure instead of to the full arrangement. The job of updating the certificates when an event happens is local.

Basch summarized in his Ph.D. thesis [10] that a KDS can be obtained by taking an algorithm for computing a discrete attribute, turning it into a proof that this attribute is correct, and animating this proof through time. This process is called *kinetization.* The first step in designing a KDS is to come up with a set of certificates that could be effectively used to validate the attribute of interest. If we have an efficient algorithm to calculate the attribute in the static case, we can drive out the set of certificates from the algorithm. However, not all algorithms are suitable to be kinetized. Because the certificates are going to be updated on the fly, the combinatorial structures involved in the updates should be local to guarantee efficiency. For example, if we want to maintain the Voronoi diagram for a set of moving points, we may choose certificates as the Delaunay triangulation because of the well-known duality relationship between them. The problem is that moving a point may effect the whole triangulation. In this case, we may have to recompute the triangulation globally. Therefore, the Delaunay triangulation is not suitable for maintaining kinetic Voronoi diagrams.

In terms of computing the convex hull of a set of points in the plane, various algorithms have been developed to compute the static convex hulls in the plane using different techniques [48] [49]. Combinatorial structures of convex hulls have been studied intensively, making the convex hull computation an ideal starting point for a new adventure with KDSs. Two static convex hull algorithms have been kinetized so far [12] [1].

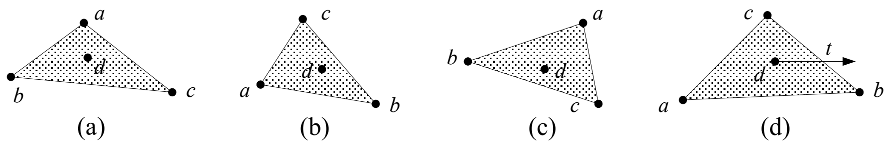

Figure 5: Four combinatorially equivalent convex hulls of four points in the plane.

The four convex hulls depicted in Figure 5 are combinatorially equivalent (i.e., point a, b, and c are in a convex position counterclockwise, and d is inside this convex hull). Considering a motion of d in the last convex hull, at time t in the figure, d is on the convex hull. Clearly, if d continuously moves in that direction, the convex hull will be changed, and d will be on the hull. Based on the definition, at time t, there is an event, where the attribute (the convex hull) needs to be updated. The next step is to find out appropriate certificates. It is generally known that we can take each triplet of the point set, and test the configurations among the set to validate the convex hull. Thus, the convex hulls in Figure 5 can be validated with the certificates in Table 2.

Certificate	*Value*
(a,b,c)	a left of $\overrightarrow{bc}$
(d,b,c)	d left of $\overrightarrow{bc}$
(b,a,d)	b right of $\overrightarrow{ad}$
(c,a,d)	c left of $\overrightarrow{ad}$

Table 2: The certificates for the convex hulls in Figure 5

These intuitive geometric relationships function well as certificates. The only problem is that the number of certificates is in $O(n^3)$ because the number of triplets in a set of n points is bounded by this complexity. The KDS for the convex hull problem can be improved by reducing the number of certificates through applying the line-point duality transformation like Hershberger demonstrated [34]. After the transformation, finding the convex hull of the set of points is equivalent to compute the upper envelope[7] of the set of lines corresponding to the points. The static algorithm for computing the upper envelope of n lines can be done optimally in $O(n \log n)$ time [34]. So, instead of maintaining the convex hull, the upper envelopes are maintained. A set of certificates can be identified to validate the relative configuration between two upper envelopes in order to merge them. The number of certificates is bounded by $O(1)$ (see Basch's thesis [12] for details). The record of the entire computation is kept in a balanced binary tree. Each node in the tree is in charge of maintaining the upper envelope of the two upper envelopes computed by its children. If an event creates a change, the event is processed through the tree. Each event can be handled in $O(\log n)$ time.

3.2.1 KDS Efficiency Measures

Measuring the complexity of a KDS is not a simple task because different continuous motion could be applied to the KDS, and the process does not terminate at some particular point. The original authors of KDS [12] propose some possible measures about the cost of the individual event update, and the relationship between the number of events and the number of changes to the attribute of interest. We list them below in Table 3 with general descriptions.

Name	*Description*
Compactness	If a KDS size is not much more than the size of the smallest certificates used to verify the correctness of the attribute of interest, the KDS is *compact*.
Responsiveness	If the worst case complexity of processing an event (update the proof) is small, the KDS is *responsive*.
Efficiency	If the total number of events processed by a KDS in the worst case is asymptotically in the same order as, or only slightly (say poly-logarithmic) larger than the number of external events in the worst case, the KDS is *efficient*.
Locality	If the maximal number of events in the event queue that depend on a single object is small, the KDS is *local*.

Table 3: Efficiency measures for KDSs proposed in [12] [10]

The efficiency definition makes a distinction between *external events*, which affect the attribute of interest, and *internal events*, which are needed to maintain the correctness of the certificates, but do not change the attribute. The external events are unavoidable given the moving trajectory. Efficient KDSs shall have a small number of internal events. In the definitions, some terms like "not much more" and "small" are used instead of actual complexity bounds. In KDSs, the actual bound depends a lot on the underlying structure. For example, in kinetic sorting, it is efficient to use binary heaps [10]. The update operation with binary heaps is bounded in $\Omega(\log n)$, which means that we cannot expect our KDSs based on binary heaps to do better than that. In most cases, like the traditional complexity analysis, $O(\log n)$ is taken as the meaning of "small" asymptotically.

[7]Regarding lines as opaque obstacles, the *upper envelope* is defined as the portion of the lines visible from the point $(0, +\infty)$.

3.2.2 KDSs in Geometry

KDSs have been developed for some basic geometric problems so far. Kinetic sorting [12] [10] is introduced as an 1D example. The same designers also worked out KDSs for the closest pair problem in 2D. Recently, Abam and de Berg studied kinetic sorting and kinetic convex hulls [1], and proved tight lower bounds for the sorting problem. They showed that even for linear motion, the worst case maintenance cost is $\Omega(n^2)$ if one wants to be able to reconstruct the sorted list in $o(n)$ time. Let m be a parameter, $2 \leq m \leq n$, then if we relax the time to $o(n \log m)$ for reconstruction (between $\Omega(n)$ and $O(n \log n)$), the lower bound on maintenance cost is roughly $\Omega(n^2/m)$. Their KDS for the convex hull problem kinetizes a different algorithm of the static convex hull calculation, which gives the ability to answer extreme point queries and convex hull containment queries.

In visibility computations, we have seen the similar idea being used in the visibility complex. We know that the coherence boundaries are some free bitangent lines crossing which will trigger the combinatorial changes of visibility. Thus, the visibility complex is built as a data structure to contain those boundaries. What different is that a KDS further requires some partial structure built by the set of chosen certificates to achieve update efficiency when an event happens. Hall-Holt's visible zone [32] follows this idea, which derives a substructure from the visibility complex to help the KDS design. The difficulty of applying the KDS approach in visibility is that identifying combinatorial structures of the certificates is not straightforward like in the previous convex hull problem. More work needs to be done to address kinetic visibility problems with KDSs.

3.3 Other Developments

Various other data structures have been proposed for visibility related problems. Although they have not been directly applied to kinetic visibility problems so far, it is interesting to see different properties that have been unveiled through them. The visibility skeleton introduced in computer graphics and computer vision[21] has been applied to compute global visibility information. The visibility skeleton is a graph structure, in which an arc is defined by a set of lines that incident on four polygon edges in the scene and a node is where two sets of lines (i.e., two arcs) merge. The visibility skeleton encodes all global visibility information for polygonal scenes. In comparison with the visibility complex, it is much simpler to be constructed and visualized in practice. Durand, Drettakis, and Puech also implemented this data structure [21]. It is noticeable that their development and implementation are targeted on 3D polygonal scenes.

Room searching problems have captured some recent interests in the visibility area. Kameda et al. developed a data structure named the visibility diagram[13][38] to encode the occlusion information inside a 2D polygon. The essential idea of this data structure is to transform the visibility computation into a path searching in the visibility diagram. The duality explored by this data structure is elegant, and with its help, we can report whether a room is searchable in linear time. Further development with the visibility diagram is possible as one of the authors pointed out in his Ph.D. thesis [58].

Kinetic visibility problems are a set of very practical problems requiring highly efficient data structures to support kinetic operations. At the data structure level, we still need to find new ways of targeting these problems from different aspects. Along with various new problems being brought into this set, we will also need to study the underlying problem domain more specifically.

4 Static Scene Algorithms

From a theoretical point of view, a successful kinetic visibility algorithm shall leverage temporal coherence through recognizing some combinatorial properties. The goal is to recognize the combinatorial equivalence of the properties throughout the motion. Only update the visibility information when the equivalence breaks and avoid global visibility recomputation for every change. Algorithms designed for simple static scenes like a set of points [18] or a set of line segments [27] [28] [43] follow this approach, although the specific usages in the algorithms are different.

Ghali and Stewart have developed an efficient algorithm for maintaining the view around a moving viewpoint in a static scene consisting of line segments [27] [28]. The same problem was studied by Nechvile and Tobola [42] [43], who improved the original algorithm with a topological map on the local structure of the line segments, which identifies the visibility changes through cases. The original algorithm [27] [28] is built on the standard duality transform that maps a set of points to a set of lines, and vice visa, in a one-to-one manner. The mapping is given as $(m, b) \leftrightarrow y = mx - b$. It is known that this transformation preserves the incidence and order among lines and points between the primal and the dual spaces. The visibility maintenance from a viewpoint in the scene is solved by checking whether any point in the dual space (corresponding to a line connecting two segment end points in the primal space) changes its position with the respect of the line (corresponding to the viewpoint in the primal space). Ghali and Stewart applied the method of maintaining two dynamic convex hulls, one each side of the line in the dual space, to check whether the visibility from the viewpoint has been changed combinatorially. The problem of this approach, as Nechvile and Tobola pointed out, is that the convex hulls need to be constantly maintained even though the set of visible segments does not change. In other words, the first algorithm keeps a global configuration of the convex hulls and tries to maintain it through time. The improvement follows logically from this observation. The second algorithm [42] [43] with a local structure thus eliminates some unnecessary updates by exploiting temporal coherence locally. This problem was studied before the development of the KDS framework. Applying the KDS knowledge, we can see that Nechvile and Tobola actually built a KDS based on the relative geometries among the line segments to validate the view from the viewpoint.

Nechvile and Tobola's algorithm [42] [43] does not improve the theoretical bound from Ghali and Stewart's algorithm [27] [28]. Both algorithms solve the problem spending $O(n \log n)$ on preprocessing, and $O(\log n)$ on an update, where n is the number of line segments. However, Nechvile and Tobola also implemented the two algorithms and compared them from a practical aspect. Their algorithm showed some significant improvement in the experiments.

The algorithm developed by Devillers et al. [18] addresses the problem of given a set of n points in the plane, computing the circular ordering of the points around a viewpoint v and efficiently maintaining this order list as v moves. The main result is that, after an $O(n \log n)$ preprocessing, a moving point query along a specified line segment can be answered in $O(k \log n + s)$ amortized time or $O(k \log^2 n + s)$ worst case time[8] where k is the number of different views and s is the output size. If we further simplify to report only the changes (not the whole point set) on the moving trajectory, then each update can be done in constant time, outputting the two points that swapped their order on the list.

Temporal coherence in this problem is explored through observing the chipped line arrangement. A *chipped line* is a line through two points in the plane excluding the line segment (chipped line

[8]The $O(\log n)$ or $O(\log^2 n)$ factor for each update comes from the cost of maintaining a dynamic convex hull.

segment) between the two points. The chipped line arrangement (CL-arrangement) induced by a set of points is constructed by drawing a chipped line through each pair of points (see Figure 6 for an example). The claim is that this arrangement encodes temporal coherence. It is not hard to see that crossing any chipped line segment does not trigger any circular order change, while crossing any chipped line does. Furthermore, the change of crossing a chipped line is combinatorially equivalent to swapping the circular order of the two points defining that chipped line. For example, in Figure 6, the viewpoint v has the initial view of (a, b, c, d, e) counterclockwise. When it moves to v' crossing two chipped line segments on the way, the view (order) does not change. However, from v' to v'', the viewpoint crosses a chipped line $\overline{cd}$. The view at v'' becomes (a, b, d, c, e) where c and d are swapped.

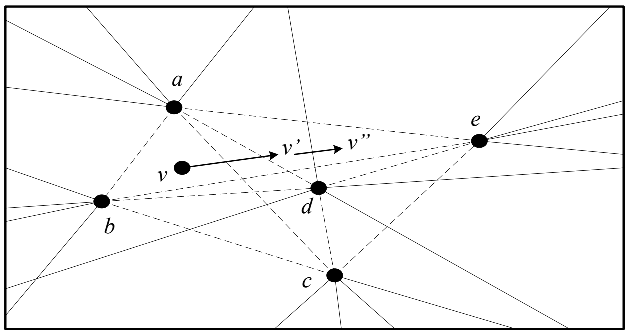

Figure 6: The CL-arrangement of a set of five points in the plane.

We could thus build a cell structure based on the CL-arrangement. Within each cell, temporal coherence is captured. The edges of a cell are determined by consecutive points on the circular ordered list from the perspective of any point within the cell. This observation implies that, in order to compute a cell, it suffices to know the view (the circular order) from a point in that cell. We can thus construct the initial view and the cell around the viewpoint. As the moving trajectory is given online, we calculate the edge of the cell that will be crossed by the moving viewpoint. Then, we update and output the view at the crossing. The new cell is "updated" by deleting two half-planes and inserting two new ones. The same procedure is repeated if more intersections are conducted.

Hall-Holt studied a more general kinetic visibility problem with a set of static convex objects at the beginning of his Ph.D thesis [32]. He adopted the traditional radial sweep technique and developed the visible zone. Figure 7 depicts the visible zone for a viewpoint v in a static scene. The important design is on the update, such that we only update the visible zone when two radial sweep lines coincide. In the figure, the arrow marks the moving direction of v in (a). The first update is going to be conducted at (b) where two sweep lines overlap (the dashed line). Note that other "unrelated" radial sweep lines are untouched at the update.

The visible zone is essentially a KDS constructed from the radial sweep, and attempts to minimize the number of internal update events which do not effect the visibility. Based on this data structure, Hall-Holt did case analysis on all possible situations where an update needs to be configured. An online algorithm follows his case studies directly. Given a static scene and the viewpoint, the algorithm first runs a radial sweep from the viewpoint to construct the initial visible zone. At each event point, it analyses which update case should be applied. It is possible to have recursive updates at some event because the radial sweep lines are not required to be aligned with

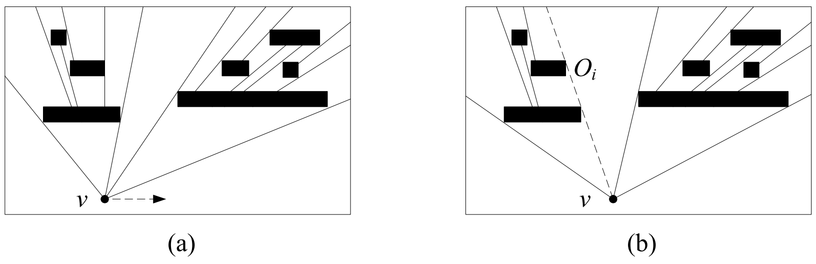

Figure 7: The visible zone from the viewpoint v. The update only happens when two radial sweep lines coincide. (a) the original visible zone from v. (b) first update on the moving path of v.

the viewpoint on the moving trajectory. However, Hall-Holt managed to show that the worst event can still be done in linear time [32]. He later paired with Rusinkiewicz to apply this algorithm on the real-time occlusion culling [33].

In some broader sense, more algorithms can be put into this category for problems like room searching[39][40][38][13] and visibility maintenance inside a polygon[57]. Recent data structure developments[38][13], as we have reviewed in the previous section, have derived some interesting algorithms which solve the 2D room searching problems in optimal time under different settings. The newly arrived algorithm for the visibility maintenance problem inside a polygon [57] applies the KDS design approach. An interested reader can follow the references to get more details.

5 Kinetic Scene Algorithms

General kinetic visibility problems are studied with kinetic scenes of a set of convex objects [35] [32] [33]. There are not many algorithms developed so far in this category, considering the difficulty of dealing with various motions in the object space. An important related question addressed more often is the collision detection problem among a set of moving objects [15] [36] [23]. Some KDSs have been used on this set of problems as well [53] [3] [11]. Speckmann explained how KDSs can be used in solving collision detection problems in her Ph.D. thesis [53]. She also provided a KDS for detecting collision among a set of simple polygons. Agarwal et al. built a KDS [3] for maintaining a pseudo-triangulation of a point set, and they extended the KDS to support an efficient algorithm on detecting collision among a set of simple polygons. Basch et al. designed a more efficient KDS for detecting collisions between two simple polygons in motion [11]. Their KDS uses fewer certificates.

In kinetic scenes, Hall-Holt's corner arc algorithm [32] is a major development, which solves the visibility maintenance problem for a set of moving convex objects. This algorithm is explained through three problem transformations. The first step is based on temporal coherence in visibility such that maintaining a list of visible objects from a viewpoint is combinatorially the same as maintaining a description of the boundary segments. A *boundary segment* is a free line segment from the viewpoint that is tangent to an object in the object space (e.g., the dashed line segments in Figure 8). When motion is taking place, we transform the problem of maintaining the boundary segments into the maintenance of a topologically equivalent structure, named a corner arc. Figure 8 (a) depicts a corner arc for a boundary segment. Given a boundary segment θ, let O_3 be the tangent object (a is the tangent point). Extend the boundary segment θ to intersect object O_1 (i.e., the far object) at b and c where c is at the far side. Let σ be the shortest path homotopic to θ from a to c. The path σ can be partitioned into the part incident to O_1 (i.e., the partial boundary of

O_1 from d to c, where d is the first incident point on O_1), and the part not incident to O_1. The corner arc of θ is the part from a to d on σ. Alternatively, the corner arc can be understood as turning θ into an elastic band and pulling the end of θ along the far object away from the visibility polygon, until it becomes tangent to the far object. A special kind of corner arc is defined as the shared corner arc. In order to avoid intersections of bitangents as depicted in Figure 8 (b), where bitangent t_1 and t_2 intersect each other, we use bitangent t_3 instead. The final transformation is an attempt to localize the maintenance of corner arcs through some partition of the object space. From the success of the visibility complex, pseudo-triangulation is chosen as the partition for the spatial subdivisions.

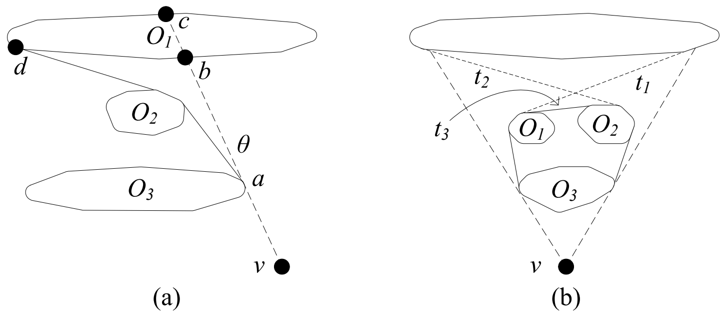

Figure 8: (a) A corner arc that is defined by the two bitangents. (b) The special kind corner arc that is shared by two boundary segments when they face each other along the same far object.

The above description actually gives the basic idea of how this algorithm works. Executing the problem transformation steps backward, we construct a pseudo-triangulation of the objects; sweep within the pseudo-triangulation from the viewpoint to get boundary segments; and calculate the corner arcs for the boundary segments. Visibility maintenance is now a matter of fact through maintaining the pseudo-triangulation bitangents, corner arcs, and boundary segments. To compute the event time of a tangency event, it suffices to compute the time that a boundary segment will become tangent to the first object along its corner arc. There are also pseudo-triangle events that need to be processed. Details of handling different types of events are omitted. The critical point is that the corner arc captures temporal coherence as some predicate for the next combinatorial change. For a boundary segment, the change is going to take place along its corner arc, and thus, through maintaining the corner arc, we can predict when the coherence needs to be updated.

There are some subtleties on handling different events as proposed in the original paper [32]. However, the proof for the correctness of the update procedure is quite straightforward from the development of the pseudo-triangulation. Events are kept in a heap based on the processing time of the event.

Hornus and Puech proposed another algorithm for this kinetic problem considering some weak radial decomposition with a similar event identification mechanism [35]. They also extended their algorithm to some simple polygonal objects (i.e., object boundary may contain some concave vertex). The main idea is to consider vertices on the boundary of the object, and categorize them based on whether they are convex or concave. So, instead of working with objects of a scene, we are dealing with vertices from each object in the plane. However, the details of their work did not appear in the publications.

Interestingly, the corner arc algorithm shows that the intrinsic complexity of maintaining visibility in a temporally coherent manner for static and kinetic scenes is asymptotically the same, up to a poly-logarithmic factor in the number of visible objects. Obviously, the algorithms for visibility maintenance in kinetic scenes solve the same problem of the static version as well. However, the reverse is not necessarily true from the first intuition. Hornus and Puech also claimed in their paper [35] that the complexity for visibility maintenance in simple polygonal object scenes is the same as for convex smooth object scenes.

6 Open Problems

There are various open problems in kinetic visibility and related fields. In general, recognizing the combinatorial structures of the visibility graph has long been an open question.[9] Different communities have been looking at this set of problems from different perspectives. In this section, we try to describe open problems from both theoretical and pragmatic points of view.

6.1 Combinatorial Issues

We have seen numerous efforts to discover the combinatorial structures of the visibility graph in different scenes. The visibility complex is one such attempt that provides a natural context for understanding and developing visibility algorithms for kinetic visibility problems. The main focus from the theoretical aspect is on exploring temporal coherence in the scenes. In order to achieve this, we have to address the scene specifically. Throughout this survey, we have seen different properties being shown for different scenes. However, building the visibility complex for a convex scene is complicated; the current implementation [7] and experiment [25] simplify the objects into discs, which suggests that we need to find other ways to leverage temporal coherence more effectively. Although Hornus and Puech claimed that their algorithm [35] can deal with objects having non-convex boundaries, understanding the properties of a simple polygonal scene is still far from clear.

The combinatorial difference between 2D and 3D is dramatic. Temporal coherence may be especially vulnerable in this case. Most of the literature that we have reviewed so far leaves the 3D version of the problem open. The lack of realization of the combinatorial structure of the visibility graph prevents algorithm designers from working in 3D or any higher dimensional spaces. The hope of solving this problem seems to depend on some mathematical system such as the oriented matroids to build some combinatorial model for organizing the combinatorial data, as Björner et al. pointed out [2] [14]. Nevertheless, it would be interesting to explore initially the connections between the kinetic visibility problems and the oriented matroids.

6.2 Complexity Analysis

Analyzing the complexity of a kinetic data structure or algorithm is another challenging aspect. We usually address the complexity by evaluating the number of events along the motion path and the time complexity of processing the event(s). For instance, Davenport-Schinzel sequences [52] are used to bound the number of events in designing a KDS [10]. This method could be generally applied. However, in kinetic visibility computation, there is some difficulty in using this approach because the events may be cascaded (recursively defined). The complexity of the corner arc algorithm [32]

[9] One version is that given a graph G and a Hamiltonian circuit C, determine in polynomial time whether there is a simple polygon whose vertex visibility graph is G, and whose boundary corresponds to C. [44]

is thus derived from some probability analysis. In order to work with the statistical model, the distribution (density) of the objects in the space has to be assumed. This approach is much more involved than using Davenport-Schinzel sequences.

For KDSs, the original authors have proposed some efficiency measures [12] (Table 3), but the validity of applying these measures seems to be case based. Because the algorithm for a kinetic problem does not terminate at certain point in time, the time complexity analysis has to be addressed with some given moving trajectories. In some instances, the worst case complexity measure may not reflect the true performance of the algorithm and/or the data structure. Above all, complexity analysis for kinetic visibility data structures and algorithms is an ongoing topic.

6.3 Visibility Applications

Visibility has various applications. The application questions are addressed in relation to computational graphics, robotic design, and some new fields such as network discovery in wireless networks [59] [56]. Here, we focus our discussion on the theory behind the application. Some applications like radiosity computation [45] have already been studied in kinetic visibility. A large number of applications in graphic rendering, collision detecting, and network discovery can also be addressed in the field of kinetic visibility. On the other hand, many practical problems involving 3D visibility computation are waiting to be solved.

In computational graphics, few algorithms utilize the prior or future positions of the observer, particularly when the objects are moving. Although the pontential value of temporal coherence for visibility computations is well-recognized, few methods are known for leveraging it effectively. Sutherland et al. have showed different ways of calculating visibility information from a practical point of view with hidden-surface removal algorithms [55]. Hubschman and Zucker further applied the frame-to-frame coherence to improve the hidden-surface removal computation [37]. Currently, ray tracing is an active topic in the graphics community. We have seen one application of the visibility complex on this subject[16]. With the current developments on kinetic visibility, it may be possible to apply other techniques to the ray tracing problem in order to gain further computational efficiency.

Collision detecting is an important algorithmic topic in robotic design. It has also been studied in the kinetic visibility domain. The KDS approach has been generally applied. Building on the current developments, we may be able to look at this problem from a more general perspective with fewer restrictions on the shape of the object or the consistency of the scene.

Different groups have been focusing on the sensor network design problems. From the computational geometry point of view, we ask questions related to visibility and range searching. Because there could be different network configurations, we may have a set of interesting problems addressing different aspects such as sensor communication and network topology.

6.4 Other Interests

Validation of an algorithm or a data structure could also be done from the implementation, which is a practical approach to evaluate data structures and algorithms. Nechvíle and Tobola proved their local enhanced algorithm [43] from the original algorithm [28] by comparing the testing results. Implementing and testing of the visibility complex have also been done recently [7] [25]. Implementation is probably a necessary step in proving and validating the algorithms and data structures in kinetic visibility problems, because most developments aim at solving practical visibility computations. At the time of writing, many implementations and empirical studies still need to be done.

Numerous other problems arise from visibility computations. Everett et al. studied the problem of the complexity of predicates for line transversals in 3D [24]. Although in theory, we generally assume each predicate can be computed in constant time, the constant factor could be very high in 3D as Everett et al. demonstrated in their paper. Their eventual goal is to find efficient kinetic visibility algorithms in practice. So, when evaluating kinetic visibility algorithms, we may have to take the complexity of predicates into consideration as well.

Finally, in addition to the traditional computation model, kinetic visibility problems may be good candidates to be addressed in the parallel computation world. In our intelligent robot case, we indeed have a parallel or distributed model where all the robots process information simultanously. Akl and Lyons studied some computational geometry problems in parallel models in the 1980s [5]. Their studies could be extended here as well.

7 Conclusion

Akl wrote in his recent technical report that "tracking moving objects becomes harder as they travel away from the observer (for example, a spaceship racing towards Mars)" [4] as an example of unconventional computing problems. The eventual goal of studying kinetic visibility problems is to explore enough combinatorial properties of the visibility information in different scenes such that efficient visibility computations can be achieved. The computations here involve not only calculating the visibility information at the starting point but also always maintaining such information.

Currently, our ability to solve such kinetic problems is very limited. General problems are always simplified by proposing constraints to the object and the scene. We have not been able to satisfy the requirements of building the intelligent robot so far. However, we have seen significant progress from different research communities targeting kinetic problems from different aspects. And, of course, they have been proposing new problems at the same time.

We conclude this survey with rather more questions than we had at the beginning in a certain sense. Our research plan would be to continuously work on this set of problems based on the previous developments. We believe that, gradually, we will get to know more about this complex world, and be able to build our intelligent robot!

References

[1] M. A. Abam and M. de Berg. Kinetic sorting and kinetic convex hulls. In *Symposium on Computational Geometry*, pages 190–197. SCG '05, ACM, 2005.

[2] J. Abello and K. Kumar. Visibility graphs and oriented matroids. In *DIMACS International Workshop on Graph Drawing*, pages 147–158, London, UK, 1995. GD '94, Springer-Verlag.

[3] P. K. Agarwal, J. Basch, L. J. Guibas, and L. Zhang. Deformable free space tiling for kinetic collision detection. In *4th International Workshop on Algorithmic Foundations of Robotics*, pages 83–96, 2002.

[4] S. G. Akl. Unconventional computing problems. Technical report, School of Computing, Queen's University, Kingston, November 2006.

[5] S. G. Akl and K. A. Lyons. *Parallel Computational Geometry.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[6] P. S. Aleksandrov. *Combinatorial Topology.* Graylock, Rochester, 1956.

[7] P. Angelier and M. Pocchiola. CGAL-based implementation of visibility complexes, 2003. Technical Report ECG-TR-241207-01, Effective Computational Geometry for Curves and Surfaces (ECG).

[8] P. Angelier and M. Pocchiola. A sum of squares theorem for visibility complexes and applications. In B. Aronov, S. Basu, J. Pach, and M. Sharir, editors, *Discrete and Computational Geometry*, volume 25 of *Algorithms and Combinatorics*, pages 79–139. Springer-Verlag, 2003.

[9] M. J. Atallah. Dynamic computational geometry. In *Proc. 24th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 92–99, 1983.

[10] J. Basch. *Kinetic Data Structures.* Ph.D. dissertation, Stanford University, 1999.

[11] J. Basch, J. Erickson, L. J. Guibas, J. Hershberger, and L. Zhang. Kinetic collision detection between two simple polygons. *Computational Geometry Theory and Application*, 27(3):211–235, 2004.

[12] J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. *J. Algorithms*, 1:1–28, 1999.

[13] B. Bhattacharya, J. Z. Zhang, Q. Shi, and T. Kameda. An optimal solution to room search problem. In *18th Canadian Conference on Computational Geometry (CCCG'06)*, pages 55–58, August 2006.

[14] A. Björner, M. Las Vergn, B. Sturmfels, N. White, and G. M. Ziegler. *Oriented matroids.* Cambridge University Press, Cambridge, 2nd edition, 1999.

[15] S. Cameron. Collision detection by four-dimensional intersectin testing. *IEEE Transaction on Robotics and Automation*, 6(3):291–302, 1990.

[16] F. S. Cho and D. Forsyth. Interactive ray tracing with the visibility complex. *Computers and Graphics*, 23(5):703–717, 1999.

[17] S. Datta, I. Stojmenovic, and J. Wu. Internal node and shortcut based routing with guaranteed delivery in wireless networks. *icdcsw*, 00:0461, 2001.

[18] O. Devillers, V. Dujmovic, H. Everett, S. Hornus, S. Whitesides, and S. Wismath. Maintaining visibility information of planar point sets with a moving viewpoint. In *17th Canadian Conference on Computational Geometry*, pages 291–294, August 2005.

[19] F. Durand, G. Drettakis, and C. Peuch. The 3d visibility complex, a new approach to the problems of accurate visibility. In *Eurographics Workshop on Rendering*, pages 245–257, 1996.

[20] F. Durand, G. Drettakis, and C. Peuch. The 3d visibility complex: a unified data-structure for global visibility of scenes of polygons and smooth objects. In *9th Canadian Conference on Computational Geometry*, pages 153–158, 1997.

[21] F. Durand, G. Drettakis, and C. Peuch. The visibility skeleton: A powerful and efficient multi-purpose global visibility tool. In *SIGGRAPH'97, Computer Graphics Proceedings, Annual Conference Series*, pages 89–100, 1997.

[22] F. Durand and C. Puech. The visibility complex made visibly simple. 11th Annual ACM Symposium on Computational Geometry, 1995.

[23] J. Erickson, L. J. Guibas, J. Stolfi, and L. Zhang. Separation-sensitive collision detection for convex objects. In *10th ACM-SIAM Symposium on Discrete Algorithms*, pages 327–336. SODA'99, January 1999.

[24] H. Everett, S. Lazard, W. Lenhart, J. Redburn, and L. Zhang. Predicates for line transversals in 3d. In *18th Canadian Conference on Computational Geometry*, pages 43–46. CCCG'06, 2006.

[25] H. Everett, S. Lazard, S. Petitjean, and L. Zhang. An experimental assessment of the 2d visibility complex. In *17th Canadian Conference on Computational Geometry (CCCG'05)*, pages 295–298, 2005.

[26] S. P. Fekete and A. Krőller. Geometry-based reasoning for a large sensor network. In *SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry*, pages 475–476, New York, NY, USA, 2006. ACM Press.

[27] S. Ghali and J. A. Stewart. Incremental update of the visibility map as seen by a moving viewpoint in two dimensions. In *7th Eurographics Workshop on Computer Animation and Simulation*, pages 1–11, August 1996.

[28] S. Ghali and J. A. Stewart. Maintenance of the set of segments visible from a moving viewpoint in two dimensions. In *12th Annual Symposium on Computational Geometry (ISG' 96)*, pages V3–V4. ACM Press, May 1996.

[29] S. K. Ghosh and D. M. Mount. An output-sensitive algorithm for computing visibility graphs. *SIAM J. Comput.*, 20(5):888–910, 1991.

[30] J. E. Goodman and J. O'Rourke. *Handbook of Discrete and Computational Geometry.* CRC Press LLC, Boca Raton, FL, second edition edition, April 2004.

[31] R. Haasa, D. Ordenb, G. Rotec, F. Santosd, B. Servatiuse, H. Servatiuse, D. Souvainef, I. Streinu, and W. Whiteley. Planar minimally rigid graphs and pseudo-triangulations. In *Special Issue on the 19th Annual Symposium on Computational Geometry*, volume 31, pages 31–61, May 2005.

[32] O. Hall-Holt. *Kinetic Visibility.* Ph.D. dissertation, Stanford University, 2002.

[33] O. Hall-Holt and S. Rusinkiewicz. Visible zone maintenance for real-time occlusion culling. In *DIMACS Workshop on Computational Geometry.* DIMACS Center, Rutgers University, Piscataway, NJ, November 2002.

[34] J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Inf. Process. Lett.*, 33(4):169–174, 1989.

[35] S. Hornus and C. Puech. A simple kinetic visibility polygon. In *18th European Workshop on Computational Geometry*, pages 27–30, 2002.

[36] P. M. Hubbard. Collision detection for interactive graphics applications. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):218–230, 1995.

[37] H. Hubschman and S. W. Zucker. Frame-to-frame coherence and the hidden surface computation: constraints for a convex world. *ACM Trans. Graph.*, 1(2):129–162, 1982.

[38] T. Kameda, J. Z. Zhang, and M. Yamashita. Characterization of polygons searchable by a boundary 1-searcher. In *18th Canadian Conference on Computational Geometry (CCCG'06)*, pages 113–116, 2006.

[39] S. M. LaValle, B. H. Simov, and G. Slutzki. An algorithm for searching a polygonal region with a flashlight. In *Symposium on Computational Geometry*, pages 260–269, 2000.

[40] S. M. LaValle, B. H. Simov, and G. Slutzki. An algorithm for searching a polygonal region with a flashlight. In *International Journal of Computational Geometry and Applications*, volume 12, pages 87–113, 2002.

[41] X. Li, P. Wan, and O. Frieder. Coverage in wireless ad hoc sensor networks. *IEEE Transactions on Computers*, 52(6):753–763, 2003.

[42] K. Nechvíle and P. Tobola. Dynamic visibility in the plane. In *15th Spring Conf. Computer Graphics*, pages 187–194. SCCG '99, 1999.

[43] K. Nechvíle and P. Tobola. Local approach to dynamic visibility in the plane. In *7th Int. Conf. in Central Europe on Computer Graphics and Visualization*, pages 202–208. WSCG '99, 1999.

[44] J. O'Rourke. Visibility graph recognition. The Open Problems Project, 2001. http://maven.smith.edu/ orourke/TOPP/P17.html.

[45] 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.

[46] M. Pocchiola and G. Vegter. Topologically sweeping visibility complexes via pseudo-triangulations. In *Discrete and Computational Geometry*, volume 16, pages 419–453, 1996.

[47] M. Pocchiola and G. Vegter. The visibility complex. *Internat. J. Comput. Geom. Appl.*, 6(3):279–308, 1996.

[48] F. P. Preparata. An optimal real-time algorithm for planar convex hulls. *Commun. ACM*, 22(7):402–405, 1979.

[49] F. P. Preparata and S. J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Commun. ACM*, 20(2):87–93, 1977.

[50] S. Rivière. Dynamic visibility in polygonal scenes with the visibility complex. In *Proc. 13th ACM Sympos. Comput. Geom.*, pages 421–423. ACM, 1997.

[51] G. Rote, F. Santos, and I. Streinu. Pseudo-triangulations - a survey. *ArXiv Mathematics e-prints*, December 2006.

[52] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications.* Cambridge University Press, New York, United States, 1995.

[53] B. Speckmann. *Kinetic Data Structures for Collision Detection.* Ph.D. dissertation, University of British Columbia, British Columbia, Canada, 2001.

[54] I. Streinu. Pseudo-triangulations, rigidity and motion planning. *Discrete and Computational Geometry*, 34(4):587–635, 2005.

[55] I. E. Sutherland, R. F. Sproull, and R. A. Schumacker. A characterization of ten hidden-surface algorithms. *ACM Computing Surveys*, 6(1):1–55, March 1974.

[56] O. K. Tonguz and G. Ferrari. *Ad Hoc Wireless Networks: A Communication-Theoretic Perspective.* John Wiley, 2006.

[57] A. Zarei, A. A. Khosravi, and M. Ghodsi. Maintaining visibility polygon of a moving point observer in polyons with holes. In *11th International CSI Computer Conference*. CSICC '2006, January 2006.

[58] Z. Zhang. *The Applications of Visibility Space in Polygon Search Problems.* Ph.D. dissertation, Simon Fraser University, British Columbia, Canada, 2005.

[59] Feng Zhao and Leonidas Guibas. *Wireless Sensor Networks: An Information Processing Approach.* Elsevier/Morgan-Kaufmann, 2004.