Visibility-Based Pursuit-Evasion in Three-Dimensional Environments

Beckman CVR Technical Report 2001–01

Svetlana Lazebnik (*slazebni@uiuc.edu*)
Beckman Institute, University of Illinois at Urbana-Champaign, USA

1 Introduction

The problem of visibility-based pursuit-evasion was first introduced in 1992 by Suzuki and Yamashita [19]. Since then, it has attracted considerable attention in the communities of robot motion planning and computational geometry. Researchers have considered many variations of the problem, including curved environments[11]; pursuers equipped with one or two "flashlights", or rays of visibility[12, 15, 16]; and pursuers guarding "rooms"[13] and "corridors"[2]. The complexity of pursuit-evasion with omnidirectional visibility has remained an unsolved problem for almost a decade, and was settled only recently[16]. With all this effort focused on the two-dimensional case, the three-dimensional case has remained completely unexplored. However, many of the tools to reason about 3D visibility already exist in literature from computer vision and graphics. Data structures for maintaining global visibility information have been studied for several decades within the framework of *aspect graphs*[9, 17, 18], and, more recently, *visibility complexes*[4]. The present paper leverages this body of work to reason about visibility issues for the pursuit-evasion problem in 3D. We develop a visibility framework similar to the *conservative regions* decomposition of Guibas et. al.[7] for both polyhedral and curved environments.

Part of the reason for the apparent previous "neglect" of pursuit-evasion in 3D is that visibility issues are perceived to be a lot more complex than in 2D. As we will show in our paper, this impression is not entirely unjustified. In 2D, there are only two kinds of critical events that change the state of visibility-based information about the environment, and the changes they induce are purely local (that is, they are determined completely by the event). In the 3D case, the catalogue of visual events is vastly more complicated, and understanding these events for smooth surfaces requires sophisticated tools of differential geometry and catastrophe theory. Moreover, as we will show in the course of this paper, the interactions between different visibility events are no longer local. Because of the large conceptual gap between visibility in 2D and 3D, this paper is devoted almost entirely to establishing a coherent framework for maintaining visibility-based information, with relatively little discussion of algorithmic issues. However, once this framework is in place, it can serve as a foundation for developing algorithms and analyzing the complexity of 3D pursuit-evasion.

2 Reasoning about 3D Visibility

The problem formulation for 3D pursuit-evasion with omnidirectional visibility is essentially the same as for the 2D variant of the problem [7]. Nevertheless, we list all the relevant terminology here for completeness. The pursuer p and the evader e are points moving inside the *environment* or the *free space*, denoted F. The free space is modeled as the interior of a simple polyhedron or of a smooth solid object of genus 0 (appropriate genericity assumptions apply to both). For purposes of illustration, we may also consider free spaces with "holes" or obstacles. Both the pursuer and the evader can move with unbounded speed, but their paths must be continuous. In the rest of the paper, p and e will denote the positions of the pursuer and the evader, respectively, at a given moment in time. A *successful motion strategy* is a path of the pursuer such that, for any path of the evader, there exists a time when the positions of the pursuer and the evader can be joined by a *free segment* (that is, a segment that lies entirely inside F). In addition to free segments, we will also find it convenient to talk about *semi-free segments*, that is, segments all of whose points lie either in the interior or on the boundary of F.

The *visible space* of the pursuer, denoted $V(p)$, is the subset of the free space consisting of all points that can be joined to p by free segments. Note that $V(p)$ is open and simply connected. The *invisible* or *shadow space* of the pursuer is defined as $I(p) = \text{Closure}(F \setminus V(p))$. $I(p)$ may have several connected components that do not have to be simply connected, even if F is (see Figure 1).

It is clear that the evader can safely move anywhere within a connected component of $I(p)$, and it cannot cross between different components without being seen. Thus, a single component of $I(p)$ is either known to be entirely *cleared* (that is, the evader is known not to lie in it), or entirely *contaminated* (that is, the evader could be lurking inside). Overall, the connectivity of $I(p)$ is the topological feature of crucial interest in 3D pursuit-evasion. As the pursuer moves, the shape of $I(p)$ evolves gradually, but at certain points in time, the number of connected components of $I(p)$ changes. We will refer to these changes as *essential events*, to distinguish them from *critical events* that may change the topology of $I(p)$ without changing the number of connected components. In section 3, we will describe a scheme for tracking essential events as the pursuer moves, but first, we will examine the topology of the shadow space with the pursuer fixed in some general position (that is, the pursuer can move freely within some small neighborhood of this position without causing any topological changes in the structure of the shadow space).

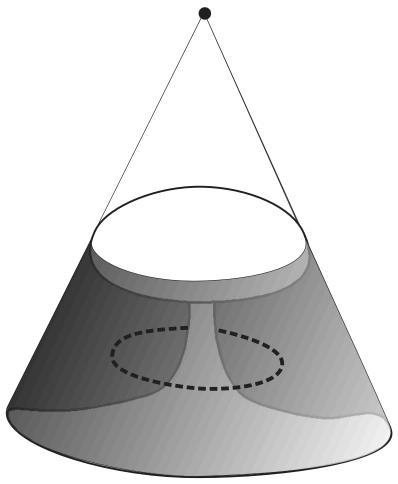

Figure 1: An example of a shadow space that is not simply connected. The shadow space under the "umbrella" has a cylindrical hole created by the stem. A thick dashed line shows a non-trivial loop around the stem.

For a particular position of p, consider a partition of ∂F into visible and invisible regions, denoted $\partial F_V(p)$ and $\partial F_I(p)$, respectively. The visible boundary region $\partial F_V(p)$ is the set of all points on ∂F that can be joined to p by a segment whose interior lies entirely in F. Then we can simply define the invisible boundary region as $\partial F_I(p) = \partial F \setminus \partial F_V(p)$. The set $\partial F_I(p)$ forms one part of $\partial I(p)$. The second part, the *occlusion surface* $O(p)$, does not lie on ∂F, but crosses the free space. To visualize $\partial F_I(p)$ and $O(p)$, it may help to imagine p as a tiny lamp illuminating the free space. Then $\partial F_I(p)$ contains the parts of the "walls" that are in shadow, and $O(p)$ separates the illuminated parts of the interior from the parts that light cannot reach. Before defining the occlusion surface, let us define its superset, the *visual cone*. The visual cone is made up of all points belonging to *critical segments*, that is, semi-free segments that originate at p, have tangential contact with ∂F in at least one point, and terminate at the point where they first exit Closure(F). In the polyhedral case, a tangent segment is defined as a semi-free segment that has one-point contact with an edge or a vertex of ∂F in its interior. Note that the two endpoints of a critical segment can never coincide since the first endpoint, p, must always lie in the open set F, and the second endpoint must lie on the boundary ∂F. Now, we can define the occlusion surface as $O(p) = C(p) \setminus V(p)$. In other words, $O(p)$ is what remains after we "chop off" the part of the viewing cone that includes the apex p and lies entirely in the free space (refer to Figure 2). There is a one-to-one relationship between segments that make up $C(p)$ and $O(p)$, and it works as follows: for a segment $s \subset C(p)$, we get the corresponding segment $s' \subset O(p)$ by taking the part of s from its first tangential contact with ∂F to its endpoint on ∂F. Note that s' can never become empty when s is not, so we can conclude that $O(p)$ has the same connectivity as $C(p) \setminus \{p\}$. At worst, s' can degenerate to one point when s is a single-contact segment that is actually tangent to ∂F at the point where it leaves the free space. This type of contact is referred to as *inflectional tangency*. In the polyhedral case, an inflectional tangent is defined as a critical segment that touches ∂F at a vertex incident on at least one *reflex edge* (an edge whose incident faces make an internal angle of more than $180°$) and at least one non-reflex edge.

Clearly, each connected component of $O(p)$ bounds some connected component of $I(p)$. If the environment has non-trivial genus or contains holes, more than one component of $O(p)$ may bound the same region of $I(p)$. Fortunately, this situation cannot occur for simply connected environments — there is a one-to-one correspondence between the components of the shadow space and the occlusion surface. To see why, suppose that F is simply connected and there exists a single region of $I(p)$ whose boundary includes two or more distinct components of $O(p)$. Now take a closed-loop path inside F that is anchored at p, enters that region of $I(p)$ through one component of $O(p)$, and exits it through another (see Figure 3). Since F is by assumption simply

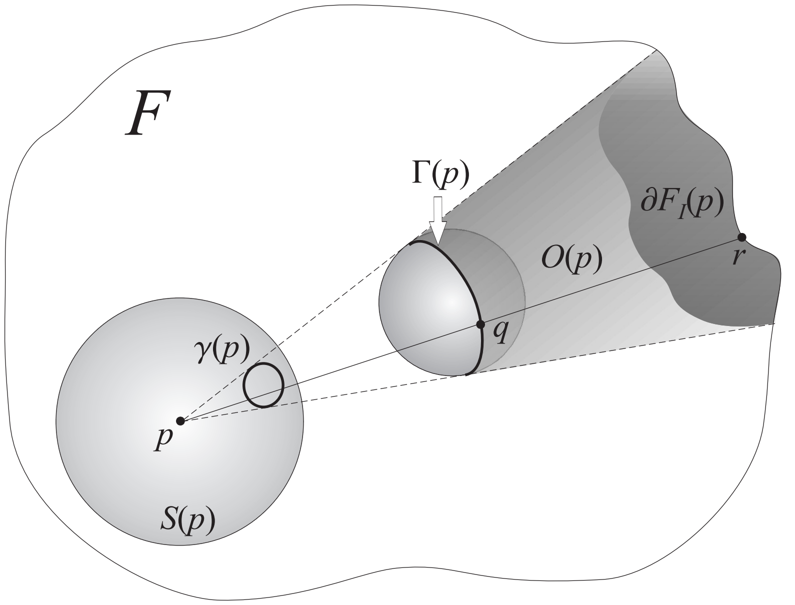

Figure 2: An illustration of the major concepts introduced in Section 2 (see text). Dashed segments indicate the boundary of the visual cone. The segment pr belongs to the visual cone, and qr is the corresponding segment on the occlusion surface.

connected, we should be able to continuously deform the loop into a single point, say p. In the course of the deformation, the part of the loop inside $I(p)$ would have to gradually shrink, and by continuity, it would have to degenerate to a single point on $O(p)$, after which all of the loop would be inside $V(p)$. However, this is impossible, because the deformation must preserve the intersections of the loop with both of the disjoint components of $O(p)$. Since we cannot shrink the loop to a point, F cannot be simply connected.

We have shown that one connected component of $O(p)$ must bound exactly one connected component of $I(p)$. This result will enable us to introduce an information graph representation for 3D pursuit-evasion that is a straightforward extension of the framework for 2D simple environments [7]. We defer a discussion of the information graph until Section 4.1, since our visibility framework is not yet complete.

The set of points where $C(p)$ is tangent to ∂F is called the *critical set*, the *contour generator*, or the *rim*[1], denoted $\Gamma(p)$. For curved environments, the rim is a set of smooth continuous curves. In

[1]Computer vision literature does not have a single standard term to describe this geometric concept. Even though we patterned our terminology of *critical segments* after the *critical set* of Giblin and Weiss [5], we prefer *rim* to *critical set*, because it is shorter.

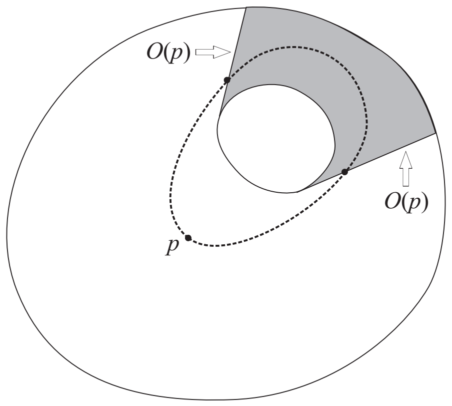

Figure 3: A 2D illustration of an environment with a hole. The shadow space (gray region) has two occlusion boundaries, lines labeled $O(p)$. The dashed line shows the loop that intersects both boundaries and passes through p. It is intuitively obvious that the loop cannot be continuously deformed into a point.

the polyhedral case, the rim is usually composed of polygonal curves (chains of edges of ∂F). It is also possible for the rim to include a whole face of the boundary, namely when p lies on the plane containing that face. However, this case is not general, and we will discuss it in Section 3. To simplify the discussion in what follows, we introduce a viewer-centered (that is, pursuer-centered) representation of visibility. The standard approach in computer vision is to project the rim onto some kind of *retinal surface*. In situations where the observer has omnidirectional visibility, a spherical retina is the natural choice. Let us denote by $S(p)$ the surface of a *viewing sphere* centered at pursuer position p. Since $p \in F$ and F is open, we can find a sufficiently small radius for the sphere such that it is contained in F. Now consider the set $\gamma(p) = S(p) \cap C(p)$ formed by the intersection of the viewing cone with the surface of the viewing sphere. Some computer vision terms for $\gamma(p)$ are *occluding contour*, *profile*, or *outline* (we will use the last term in the paper). We can think of the outline as the projection of the rim onto the retina. To visualize the connection between visual cones, rims, and outlines, refer to Figure 2.

Since each critical segment intersects $S(p)$ exactly once, there exists a one-to-one correspondence between points on the outline and critical segments. Consequently, there is a direct relationship between the topological features of the outline and of the occlusion surface. For instance, it is easy to see that the connected components of $\gamma(p)$ are exactly the connected components of $C(p) \setminus \{p\}$ or $O(p)$ projected onto the viewing sphere. In the following, we will take advantage of this close relationship by referring to topological features of the outline as a shorthand for the corresponding features of the occlusion surface. Let us emphasize, however, that we are not concerned directly with features of rims or outlines, but with features of the occlusion surface, since it is the boundary that separates the visible space from the regions of invisible space where an intruder might be lurking. We do not require the formalism of the viewing sphere to develop data structures or algorithms for the pursuit-evasion problem, since we assume that we have an exact spatial model of the environment.

We will concentrate only on *stable features* that are preserved under small arbitrary movements of the viewpoint. Stable features have been extensively studied in computer vision [8, 14]. For both polyhedral and smooth scenes, existing literature identifies the same three kinds of features on outlines, namely, *folds*, *T-junctions*, and *terminations*. Figure 4 shows the appearance of these features in the image and on the occlusion surface.

Fold: A fold point is the projection of a *single-contact* critical segment, that is, a segment that grazes the boundary of the free space at a unique rim point. A fold is the most common topological feature — all but a finite number of outline points are actually fold points.

T-junction or crossing: an intersection of the viewing sphere with a *double-contact* or *bitangent* critical segment, that is, a segment that touches ∂F at two distinct points in its interior. The two points of tangency are actually located on two disjoint rim branches. The more distant branch terminates at the contact point, while the nearer branch remains continuous. On the surface of the viewing sphere, this feature looks like a meeting of two outline curves, with the more distant branch becoming occluded. Note that if the boundary of the environment were transparent, neither rim branch would terminate. On the viewing sphere, we would observe a transversal crossing of two folds instead of a T-junction.

Termination or cusp: an endpoint of a single outline curve. This type of feature is a projection of an inflectional tangent. The rim branch incident on the inflectional tangent also terminates at this point (in a transparent environment, the rim remains continuous and the outline has a cusp). If the boundary ∂F is smooth, the inflectional tangent has the same direction as the corresponding rim curve.

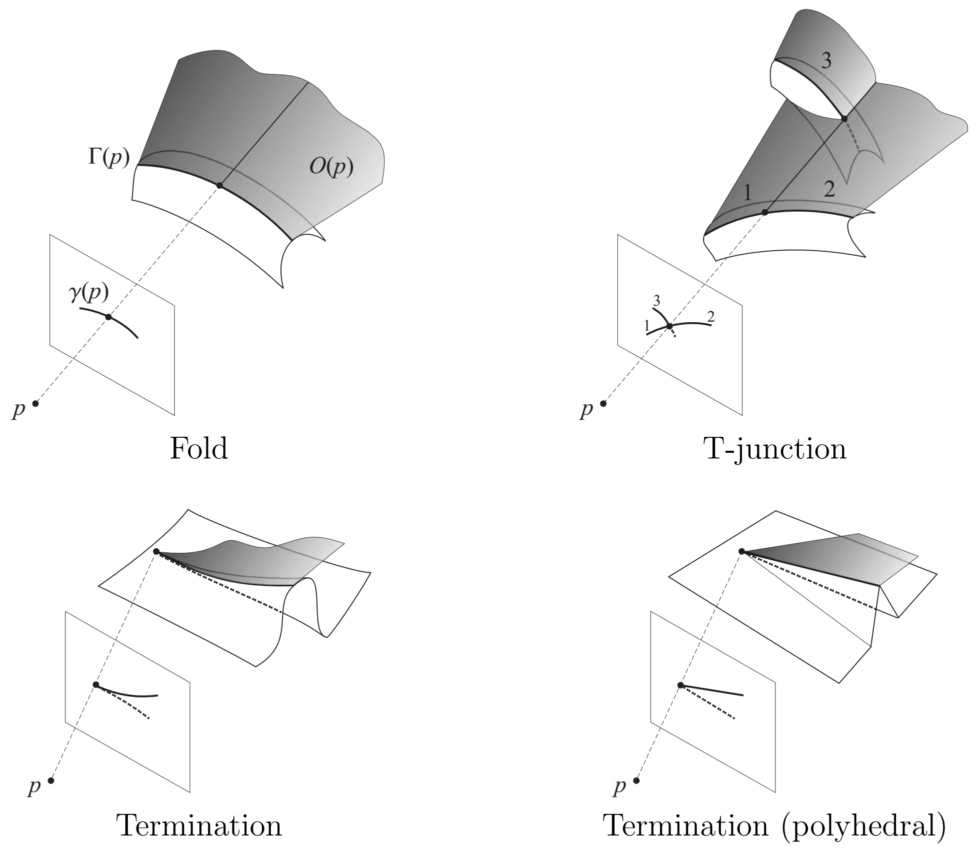

Figure 4: Generic features on the outline and on the occlusion boundary. For simplicity, the retinal surface is shown as a plane. The left top drawing labels all the important components of the setup. Folds and T-junctions look very similar in the smooth and polyhedral cases, so only the smooth case is shown. A termination is shown for the polyhedral case to illustrate the definition of inflectional tangency.

In the smooth case, the three features described above are "atoms" in a combinatorial description of the outline. Maximal connected (open) sets of fold points form a collection of disjoint curve branches. T-junctions and terminations are vertices that are incident to three or one branches, depending on their type. We will think of the occlusion surface as being made up of disjoint *sheets* that connect in threes along bitangent critical segments, and terminate at inflectional critical segments. To visualize the connectivity of these sheets at a T-junction, refer to Figure 4, top right (the sheets are labeled 1, 2, and 3). For polyhedral objects, this framework requires refinement. Recall that for general viewing position, rims of polyhedra consist of edge chains, and outlines are polygonal curves [2]. The piecewise-linear nature of fold branches is not topologically "interesting"

[2]Strictly speaking, they are a chain of great-circle arcs on the viewing sphere.

in a generalized sense, but it is significant for polyhedra. We must make a distinction between outlines that are combinatorially different, that is, outlines that are formed by projections of rims made up of different edges on the boundary of the environment. Accordingly, we must define outline branches not as continuous fold segments that end in terminations or T-junctions, but as projections of single edges on the rim. We must also introduce an additional type of isolated outline feature, namely, the projection of a vertex on the rim.

Overall, for any fixed pursuer position, we can formally specify the structure of the outline (equivalently, the occlusion surface) using the graph-like description introduced above. If the pursuer leaves its static viewpoint and begins to follow a continuous path, the occlusion surface will change smoothly most of the time, and this description will remain the same. However, there will be a discrete set of *visual events* when the combinatorial structure of the outline will change. We look at these events in the next section.

3 Critical Visibility Events and Event Surfaces

Visual events occur when the pursuer crosses certain *critical event surfaces*. Computer vision literature contains complete catalogues of visual events for both polyhedral and smooth scenes [17, 18]. The arrangement of critical event surfaces partitions the environment into three-dimensional cells inside which the topology of the shadow space does not change. The dual of this *viewpoint space partition* (VSP) is known as the *aspect graph* [9] whose nodes represent stable views of the environment and arcs represent critical events. The complexity of the aspect graph for polyhedral scenes with n faces is known to be $\Theta(n^9)$ [18]. For the smooth case, the complexity depends on the geometric representation of the environment. For example, the aspect graph of an algebraic surface of degree d is $O(d^{12})$ under orthographic projection [17]. This bound may be even higher for perspective projection, which is the relevant model in our setting.

Because of the large size of aspect graphs, practical implementation of 3D pursuit-evasion algorithms would appear to be a daunting task. Since we are in principle interested only in *essential events* that change the number of connected components of the shadow space, we might hope that it would be sufficient to build a partition of the environment using only essential event surfaces. In 2D, it is easy to identify such surfaces. For example, smooth planar objects have only two kinds of essential events, inflectional tangents and bitangents, and both of these change connectivity (passing an inflectional tangent makes an outline point appear or disappear, and passing a bitangent makes two points split or merge). When outlines go from collections of points to arrangements of curves, the situation becomes more complicated. In 3D, purely local information about a single visual event is no longer sufficient to determine whether a global change in connectivity has occurred. Refer to Figure 5, which shows two different configurations of outlines undergoing the same critical event: branch B becoming tangent to branch C. For the left configuration, this event results in a change of the number of connected components of the outline; for the right configuration, the number remains the same. From a computational point of view, the only way to detect essential events is to maintain all intermediate changes in the incidence relationships between different branches of the outline.

There is another reason why maintaining all visual events is important: to design unambiguous motion strategies for the pursuer. Recall that the pursuer is constrained to move continuously, and the only way to do so is to pass between adjacent aspects or views of the environment. However, a sequence of essential events may not give a full description of a path, since it is not guaranteed to correspond to a sequence of neighboring cells. It may be possible to design an efficient interpolation routine to generate full paths from a partial specification; but for now, we propose a simpler strategy that will preserve the completeness properties of the motion planning algorithm. Our approach is to compute the VSP of the environment using a full catalogue of visual events, to construct the associated information graph, and to search this graph for a path from the initial state to the goal. The next step, then, is to describe the relevant visual events for polyhedral and curved environments.

3.1 Smooth Environment Boundaries

The theory of visual events for smooth surfaces is quite elegant, and it provides useful insights for the case of polyhedra. For this reason, we will consider smooth surfaces first. There are two flavors of visual events: *local* and *multilocal*. Local events result from the interaction of viewing rays or segments with a single special point on the surface of the environment, while multilocal events

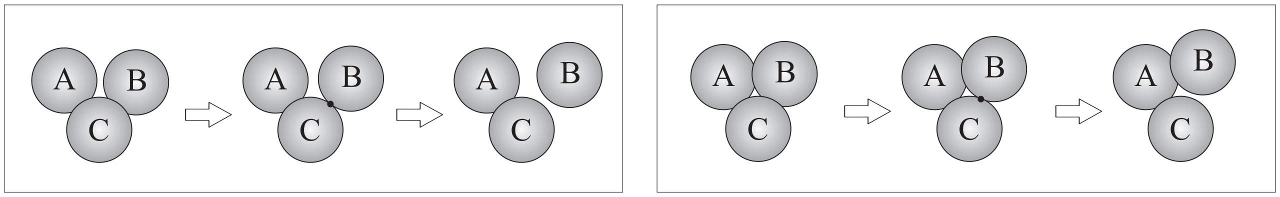

Figure 5: An illustration of the same visibility event (tangent crossing between spheres B and C) having a different effect on the connectivity of the outline. On the left, the number of connected components goes from one to two, and on the right, the number of connected components remains the same.

involve two or more points that are separated in depth. In the 2D case, there are two kinds of events, corresponding to inflections and bitangents. Intuitively, local events in 3D are analogous to 2D inflections, and multilocal events are analogous to 2D bitangents. The detailed picture, however, is considerably more complicated. A comprehensive treatment of all visual events is beyond the scope of this paper, since it heavily relies on differential geometry and catastrophe theory. We will list only the so-called "codimension-1 singularities" that have two-dimensional event surfaces. The three local events of this kind are called *lip*, *beak-to-beak*, and *swallowtail* (the names are due to the inventor of catastrophe theory, René Thom [20]). Refer to Figure 6 for an illustration of each.

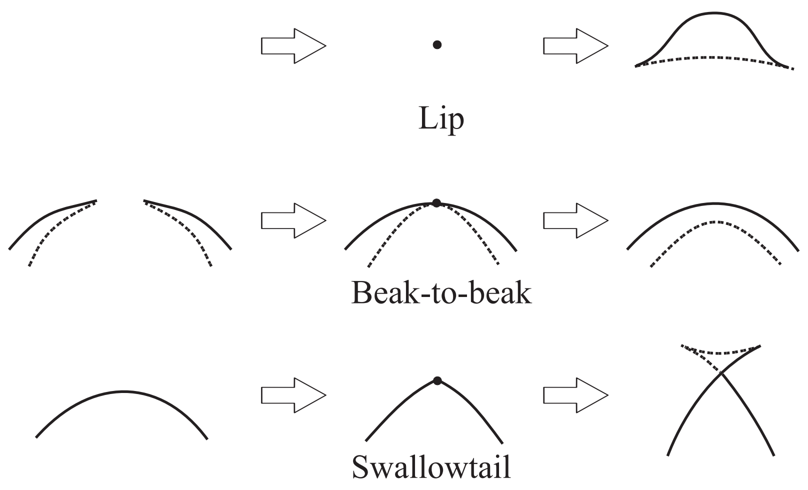

Figure 6: Three kinds of local events for smooth surfaces (dashed lines are occluded).

Lip: an isolated branch of the outline appears out of nowhere. Think of flying an airplane high over a hill, and then reducing altitude until it is possible to see the silhouette of the hill against the sky. Unlike other kinds of local events, this one gives us enough information to determine the global change in connectivity — namely, a new connected component appears. The event surface is a ruled surface defined by *asymptotic tangents* at *parabolic points*[3] on environment boundaries. Parabolic points are points of zero Gaussian curvature (the 3D analogue of inflections), and they generically occur along one-dimensional curves.

Beak-to-beak: two terminations are merged (in the transparent case, two pairs of branches come together and then move apart with different connectivity). Note that we do not have enough information to find out if the number of connected components changes (see Figure 7). The event surface is formed by asymptotic tangents at a different kind of parabolic curves.

Swallowtail: a fold morphs into two connected branches, with a creation of a T-junction and a termination. The number of connected components of the shadow space does not change. The event surface is ruled by asymptotic tangents of a special *flecnodal curve*.

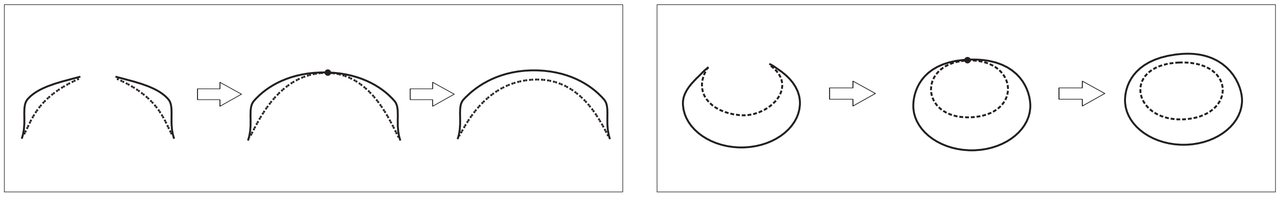

Figure 7: A beak-to-beak transition results in a local change of connectivity. Globally, the number of connected components may change (left) or remain the same (right).

Next, let us consider the three kinds of multilocal events, *tangent crossing*, *cusp crossing*, and *triple point* (see Figure 8).

Tangent crossing: two branches of the outline become tangent to each other. The tangency point is the projection of a *limiting bitangent*, or a critical segment with two points of contact that have coincident tangent planes. Figure 8 shows two kinds of tangent crossings. In the

[3]We resorted to casually dropping differential geometry terms without defining them. Complete definitions may be found in many books, e.g. DoCarmo [3] or Koenderink [10]. For our purposes, it is sufficient to note that all three kinds of singularities have ruled event surfaces.

top case, a new branch appears together with two T-junctions, and the number of connected components remains the same. In the bottom case, two locally disjoint branches merge and also create two T-junctions, but the global connectivity change cannot be determined. Note that it is possible to generate more varieties of tangent crossings, for example, by making the occluded branch "come forward" in the top case. Event surfaces corresponding to tangent crossings are *limiting bitangent developables* — that is, they are ruled by maximal semi-free segments that are also limiting bitangents.

Cusp crossing: a termination and a fold locally coincide, after which a T-junction is created. The event surface is formed by maximal semi-free segments that have one regular tangential contact with the boundary of the free space, and terminate in an inflectional tangency (that is, they pass along an asymptotic direction of the associated boundary point).

Triple point: three folds become coincident. Figure 8 shows just one example of a triple point, but more examples can be generated by varying the relative depths of the three folds and their orientations (orientation tells us on which side of the fold the nearer part of the environment lies). The event surface is ruled by triple-contact maximal semi-free segments.

3.2 Polyhedral Environment Boundaries

Visual events for polyhedral scenes are simpler than for smooth scenes in some respects, and more complicated in others. On the one hand, polyhedral boundaries have fewer local features — there are no parabolic or flecnodal curves to worry about, only vertices, edges, and faces. On the other hand, polyhedral objects seem to have many "irrelevant" visual events. For instance, a convex polyhedron generates a visual event whenever the support plane of any one of its faces is crossed (the face becomes occluded, or "goes below the horizon") [18]. By contrast, a convex smooth object has no visual events at all — all its aspects are topologically equivalent. Though we seem to face an explosion in the number of events when making the transition from smooth objects to polyhedra, we can still draw useful analogies between the two cases. For instance, the distinction between local and multilocal events still applies to polyhedra.

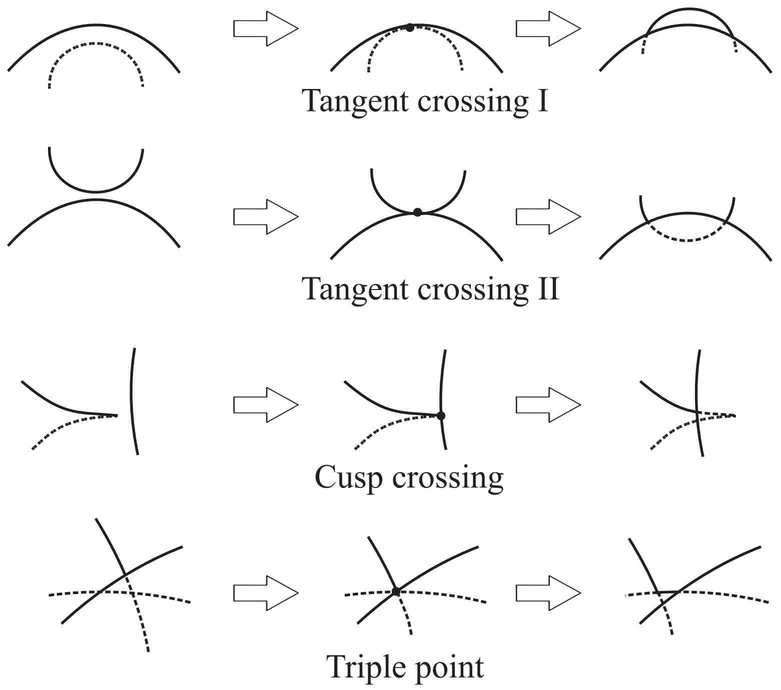

Figure 8: Three kinds of multilocal events for smooth surfaces.

Local events occur whenever the pursuer crosses the support plane of any face. For any given face, we can form the corresponding event surface by drawing all maximal semi-free segments in the plane of this face. The simplest kind of local event is crossing a support plane of a *free face*, that is, a face that can be locally extended into free space in any direction along its support plane. This type of event does not change the number of terminations or T-junctions, and it is analogous to a simple tangent in 2D. Figure 9 shows three other events (analogous to inflectional tangents in 2D) that are reminiscent of the three local events for smooth surfaces.

Multilocal events for polyhedra are very similar to those for smooth surfaces. The triple point event is essentially the same, so it is not illustrated. It is worth noting, however, that the event surface for a triple point is not planar. Instead, it is a part of a quadratic *regulus* formed by maximal semi-free segments tangent to three particular edges of the environment. This, then, is one of the chief differences between 2D and 3D: in 2D, all event surfaces are linear, even for curved environments; in 3D, even polyhedra have one class of non-planar event surfaces. In addition to triple point events, there are also events similar to tangent crossings and cusp crossings. In Figure 10, we show examples of these events. Note that a generic tangent crossing is a projection of a

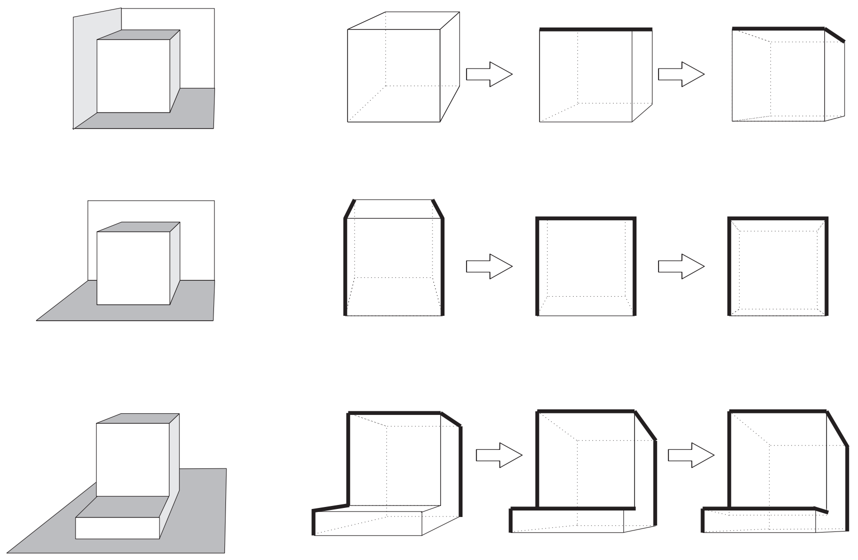

Figure 9: Local events generated by edge-vertex interactions for polyhedral surfaces. The left column shows the spatial configuration of the relevant part of the environment, and the right column shows the outlines before, during, and after the event. Thick lines are parts of outline, thin solid lines are other visible edges, and dashed lines are occluded edges. Top: a new connected component of the outline is introduced, together with two terminations. Middle: two components merge, two terminations are eliminated. Bottom: a T-junction appears. Compare with Figure 6.

bitangent critical segment that makes one contact with an edge, and one contact with a vertex. The event surface consists of all maximal semi-free segments that touch this edge and this vertex, so once again, it lies in a plane. In the literature, one often sees the two events above labeled EEE and VE respectively, V standing for vertex, and E standing for edge [4, 18].

This concludes our description of visual events for smooth and polyhedral environments. To summarize: given the description of a free space boundary ∂F, we want to form the VSP of F by computing the arrangement of all critical event surfaces. It is easy to show that the cells of this arrangement have the same properties as *conservative regions* in 2D [7]: any movement of the pursuer inside one of these cells does not result in topological changes of the invisible region $I(p)$, and so does not change the state of pursuer's knowledge (or ignorance) as to the evader's possible whereabouts. Note, however, that these cells are not necessarily *maximal conservative regions* — it is possible to cross boundaries between some adjacent cells without any essential event. It remains an interesting research topic to determine whether it is possible to simplify the VSP by

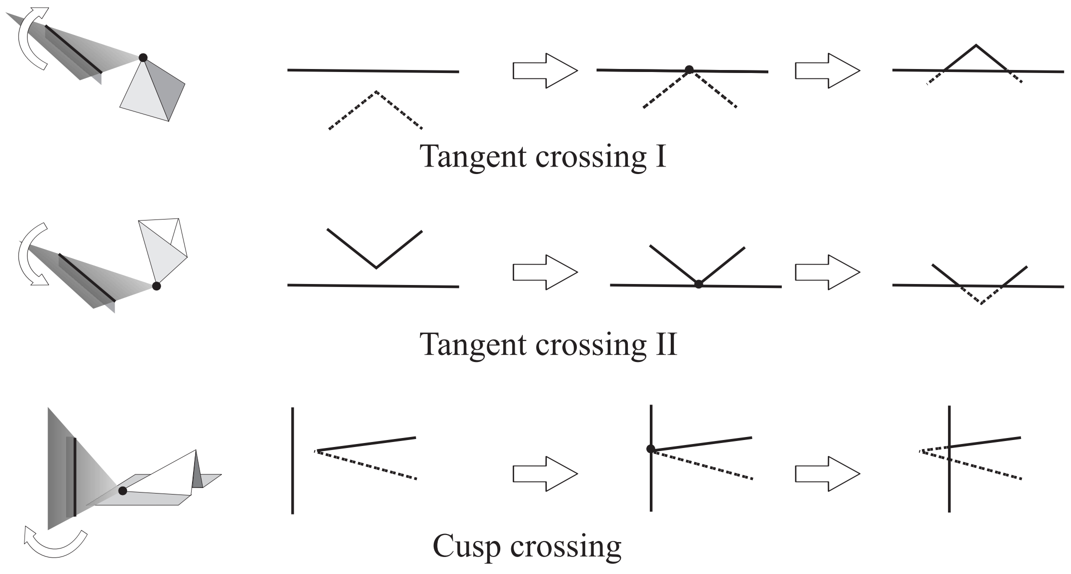

Figure 10: Three kinds of vertex-edge (VE) events for polyhedral surfaces. The left column shows the geometric configuration of the scene, together with parts of the event surface. The edge and the vertex that participate in the event are drawn in bold, and the arrow indicates the direction of movement across the event surface. Compare with Figure 8.

merging non-maximal cells. After we have computed the VSP and its dual aspect graph, we can move on to constructing the associated *information graph* and to running the motion strategy algorithm, tasks that we will address in the next section.

4 Computing Motion Strategies

4.1 The Information Graph

By analogy with the 2D case, we want to maintain the state of pursuer's knowledge as to the evader's presence or absence in each of the connected components of the invisible space. In the 2D case, this was done by associating a bit with the finite number of outline points on the viewing circle, or equivalently, with the *gap edges* of the visibility polygon inside a conservative region. There are two possible labels: 0, meaning that the corresponding component of shadow space is cleared; and 1, meaning that it is contaminated. In the 3D case, we will use the same labels, and attach them to every sheet of $O(p)$ (recall that in Section 2, we gave a different definition of sheets for smooth and polyhedral environments). Note that for a given configuration of sheets,

Visual Event	Label Update
Sheet appears in existing component	Inherit label of existing component
Sheet disappears in existing component	N/A
Component appears	Mark new component 0
Component disappears	N/A
Component splits	Inherit label of old component
Components merge	Combine old labels with logical OR

Table 1: Label changes associated with possible consequences of visual events.

not every sequence of labels is *valid*. Since sheets that belong to the same connected component of $O(p)$ bound the same region of $I(p)$, the constraint is that all their labels must be the same. Thus, we define an *information state* associated with a particular aspect as a specification of the configuration of $O(p)$ for that aspect, with binary labels (subject to the validity constraint) assigned to each sheet of $O(p)$. Then, the nodes of the information graph are all information states for all aspects. Directed arcs connect two information states if the states belong to adjacent aspects, and if the labels are related by an update associated with the visual event separating these aspects. Visual events in 3D can have several possible effects on the structure of $O(p)$. All events cause *incremental changes*, that is, appearance and disappearance of different sheets of $O(p)$. Note that both appearances and disappearances may simultaneously occur in the same event. Incremental changes may or may not affect the number of terminations or T-junctions, or the number of isolated regions of $I(p)$. When the number of connected components of $I(p)$ does change, it may be caused by the appearance or disappearance of one of the regions, splitting of one region, or merging of several regions. The updates associated with these events are essentially the same as in the 2D case, with obvious modifications to support the more complicated data structures representing the connectivity of $O(p)$. For instance, we must update labels globally for all sheets that belong to a connected component that was affected by an event. We must also allow for incremental events, that is, the appearance or disappearance of sheets that belong to the same connected component. Table 1 summarizes the possible changes and corresponding label updates.

For the sake of the motion planning algorithm, we can arbitrarily select an *initial vertex* as any information state where all the labels are 1. Our task is to search the information graph for a path to some *goal vertex*, which is any information state where all the labels are 0 (or $I(p)$ is empty). Just as in the 2D case, the resulting algorithm is complete — it will find a successful motion strategy if one exists.

4.2 Examples

To illustrate our visibility framework in action, and to get a concrete grasp on some of its subtleties, we include examples of two simple environments, one of which can be cleared by a single pursuer, and the other one cannot. The first example, shown in Figures 11 and 12, is a cube with two sides "glued" to walls of the environment. Let us imagine that the cube is sitting inside a larger rectangular box, so that no additional visual events are generated. The VSP in this example is formed by the supporting planes of the four exposed faces of the cube. By examining the aspect graph shown in Figure 12, it is easy to find a successful solution strategy for this example. Suppose we start in cell J. The shadow space at J has one connected component, and it is labeled 1. The first move is from J to I, and the visibility information remains the same. Next, we move from I to H, and this results in an appearance of a new component. We now have one component labeled 0, and one component labeled 1. Finally, a move from H to G causes a disappear event, leaving one component labeled 0. Though the strategy is simple, this example contains some non-trivial features. For instance, it cannot be reduced to a 2D environment that can be searched by a similar strategy. The aspect graph for this example actually contains both types of essential events, split/merge and appear/disappear. To look more closely at the interaction between these two kinds of events, consider the path K $\rightarrow$ J $\rightarrow$ I. None of the transitions on this path cause essential changes — all three cells have a single connected component of $I(p)$, and it changes incrementally along each transition. We may be tempted to group these cells (along with all the other cells except H, for that matter) into the same equivalence class. However, in reducing the aspect graph in this manner we lose essential information. Namely, if cells K and I are indeed equivalent, then why does the transition K $\rightarrow$ H cause a split, while the transition I $\rightarrow$ H causes an appearance? Clearly, incremental events cannot be disregarded in maintaining information states. It may still be possible to simplify the aspect graph by merging incremental transitions, but this task appears to be fraught with subtle dangers.

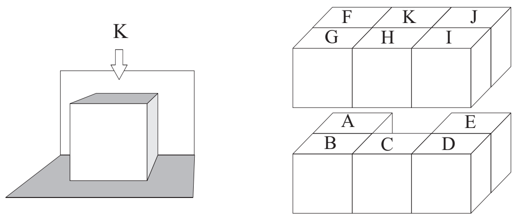

Figure 11: Example environment 1. Left: geometric setup. Right: a representation of the VSP cells. The “top” and “bottom” rows of cells are moved apart for clarity. The plane separating them is the supporting plane of the top face.

The second example, shown in Figures 13 and 14, is similar to the first, except that the cube is now “glued” only at its bottom face. A look at the aspect graph of this environment shows that it cannot be cleared by a single pursuer. None of the transitions in the graph changes the number of connected components of the outline, so the evader can continue concealing itself indefinitely. The fact that this toy example cannot be cleared by a single pursuer suggests that the class of polyhedra that are 1-searchable may be very restricted. Intuitively, pursuit-evasion is much easier in 2D. For instance, when the pursuer descends into a deep “pocket”, its visible space tends to form a narrow beam that partitions the rest of the environment. As long as the beam can be used to separate the cleared parts of the environment from the contaminated ones, the pursuer can make progress. In 3D, however, when the pursuer descends into some pocket, its visible space can merely cuts a cylindrical “hole” through the environment, without disconnecting it. As a result, the evader can freely move between several hiding places without ever being detected. This situation is depicted in Figure 15.

Because a single pursuer in 3D does not appear to have a lot of power, it may be useful to consider other variants of the pursuit-evasion problem, such as pursuit-evasion with multiple pursuers. In the next section, we focus on other possible extensions, in issues of theory as well as of practical implementation.

5 Future Work

The current paper focused on extending the visibility framework for pursuit-evasion from 2D to 3D in a manner that would permit a more or less direct application of the information graph

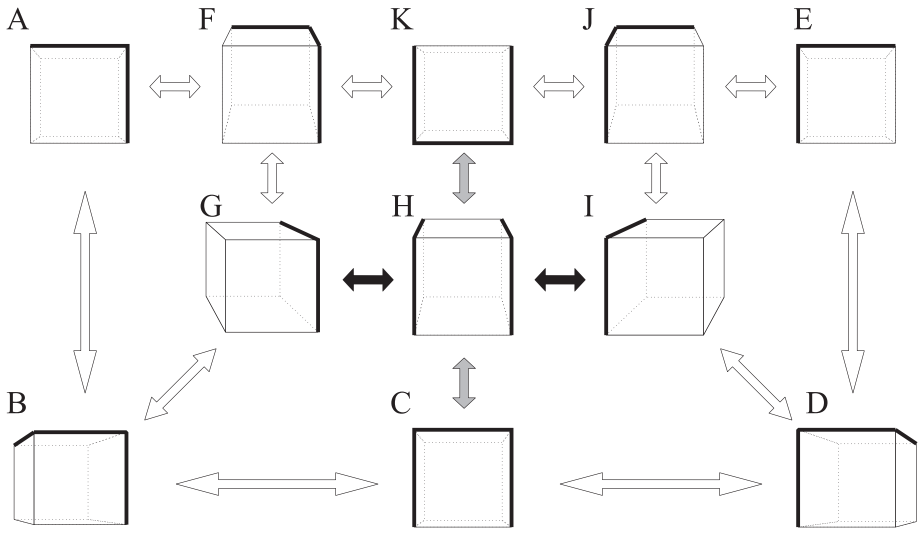

Figure 12: Example environment 1 - complete aspect graph. Aspects, or vertices, are shown using representative views from the corresponding VSP cell. The drawing conventions for the aspects are the same as in Figure 9. To simplify the graph, we have not explicitly indicated correspondences between outline segments of adjacent aspects. Gray arrows indicate merge/split events, black arrows indicate appear/disappear events, and white arrows are incremental.

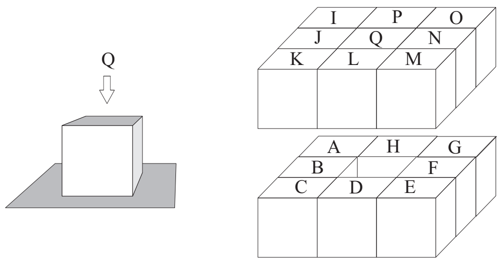

Figure 13: Example environment 2 - geometric configuration.

strategy of Guibas et. al. [7]. At this stage, most algorithmic complexity issues of this problem in 3D are still unexplored, and the no direct approach towards an efficient implementation is proposed. Therefore, the most natural way to conclude this article is by listing some promising research avenues.

- Apply recursive arguments similar to those in Guibas et. al. [7] to derive loose upper bounds on the number of pursuers in environments with or without holes. It may also be possible to use heuristic divide-and-conquer techniques to design non-optimal algorithms for multiple pursuers.

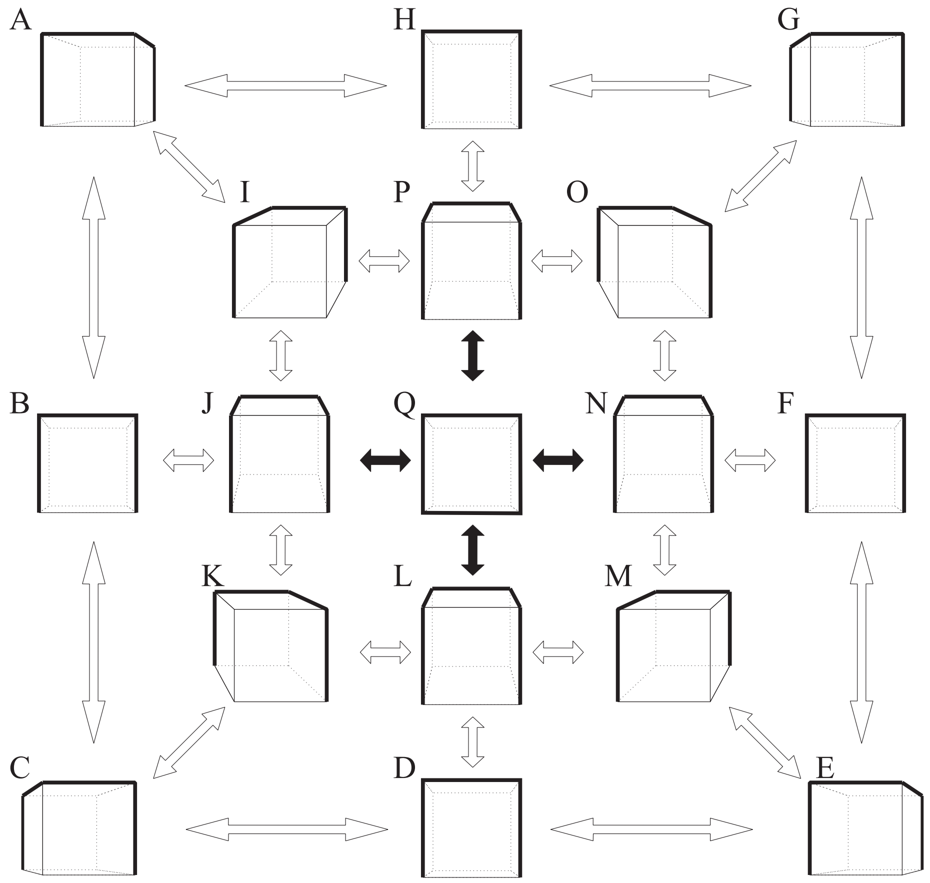

Figure 14: Example environment 2 - complete aspect graph. The pictorial conventions are the same as in Figure 12, with the exception of arrow color. Here, black arrows indicate an event that eliminates two terminations, but does not change the number of connected components.

- Show non-trivial examples with recontamination (that is, examples that weren't constructed by extruding 2D environments along the extra dimension). In the 2D case, it is known that some solution strategies require a linear number of recontaminations. Is the same true in 3D, or could the required number be, say, quadratic?

- Characterize the class of environments that can be cleared by one pursuer with 360° visibility. Recently, Park et. al.[16] presented such a characterization for the 2D case (the problem was open for almost a decade). It is possible that the 2D characterization can be extended to 3D directly. For instance, the first non-searchability condition has a direct counterpart in 3D. Figure 15, which we discussed earlier, is actually a non-trivial example of an environment for which this condition (called *unguarded triple*, or *u-triple*) holds. Overall, counterparts

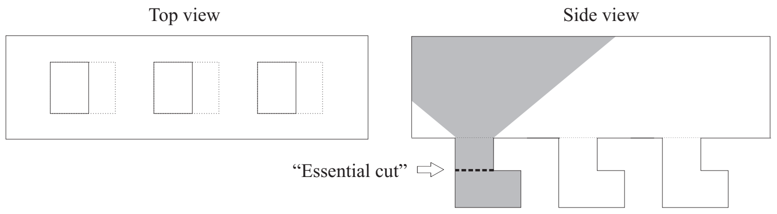

Figure 15: A "u-triple" configuration in 3D [16].

of all three non-searchability conditions should also be necessary in 3D, although they may not be sufficient.

- Develop more "direct" motion strategy algorithms. The upper bound on the complexity of the information graph algorithm as presented in this paper is exponential. While this approach is provably complete, it is difficult to reason about its practical performance. Moreover, VSP-based solution strategies may not offer an appropriate framework in which to derive lower bounds on the complexity of the problem. In 2D, the decomposition of the environment into conservative regions has complexity $O(n^3)$, while it was shown that there exists a $O(n^2)$ algorithm for computing a successful strategy [16].

- Study alternative formulations of 3D pursuit-evasion. In 2D, we know a lot about pursuers with k flashlights. The analog of k flashlights in 3D is a pencil of half-planes. The development of efficient algorithms may be easier for a pursuer sweeping a plane through the environment.

- Consider special cases of 3D environments, for instance, polyhedral terrains. The complexity of aspect graphs is known to be lower for terrains than for general polyhedra — almost $O(n^8)$, as opposed to $O(n^9)$ [1]. It is more likely that a practical implementation could be developed for a restricted class of environments.

- Develop an efficient implementation of a VSP-based motion strategy algorithm. To this day, aspect graphs are notoriously difficult to implement. We may be able to take advantage of recent advances in the study of visibility complexes [4]. Visibility complexes exist in four-dimensional line space, but their complexity, $O(n^4)$, is substantially lower than that of aspect graphs. It must be noted that visibility complexes are still a relatively new research area: so far, they have only been defined for convex objects, and there no complete implementation. It is interesting to study the ways in which visibility representations in line and segment space can be applied to pursuit-evasion.

Overall, three-dimensional visibility-based pursuit-evasion is a problem worth studying, since it offers opportunities to integrate research topics that have recently generated interest in many fields, including motion planning, computational geometry, computer vision, and computer graphics. Whatever the focus of future research, practical implementation will be one of the biggest challenges, and resolving this challenge successfully will offer the biggest rewards.

References

[1] P. Agarwal and M. Sharir, "On the Number of Views of Polyhedral Terrains", In Proc. 5th Canad. Conf. Comput. Geom., 1993, pp. 55-60.

[2] D. Crass, I. Suzuki, and M. Yamashita, "Searching for a Mobile Intruder in a Corridor — the Open-Edge Variant of the Polygon Search Problem", Int. J. of Comp. Geom. and Appl., 5(4), 1995, pp. 397-412.

[3] M. do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

[4] F. Durand, G. Drettakis and C. Puech, "The 3D Visibility Complex", submitted to ACM Transactions on Graphics.

[5] P. Giblin and R. Weiss, "Reconstruction of Surfaces from Profiles", In Proc. 1st Int. Conf. on Computer Vision, pp. 136-144, 1987.

[6] Z. Gigus and J. Malik, "Computing the Aspect Graph for the Line Drawings of Polyhedral Objects", IEEE Trans. Pattern Analysis and Machine Intelligence, 12(2), 1990, pp. 113-122.

[7] L. Guibas, J.-C. Latombe, S. LaValle, D. Lin and R. Motwani, "Visibility-Based Pursuit-Evasion in a Polygonal Environment", International Journal of Computational Geometry and Applications, 9(5), 1999, pp. 471-494.

[8] J. Koenderink and A. Van Doorn, "The Singularities of the Visual Mapping", Bological Cybernetics 24, 1976, pp. 51-59.

[9] J. Koenderink and A. Van Doorn, "The Internal Representation of Solid Shape with Respect to Vision", Biological Cybernetics 32, 1979, pp. 211-216.

[10] J. Koenderink, *Solid Shape*, MIT Press, Cambridge, Massachusetts, 1990.

[11] S. LaValle and J. Hinrichnsen, "Visibility-Based Pursuit-Evasion: The Case of Curved Environments", Proc. of IEEE International Conference on Robotics and Automation, 1999.

[12] S. LaValle, B. Simov, and G. Slutzki, "An Algorithm for Searching a Polygonal Region with a Flashlight", Proc. of 16th ACM Symp. on Comp. Geom., 2000, pp. 260-269.

[13] J.-H. Lee, S.-Y. Shin, and K.-Y. Chwa, "Visibility-Based Pursuit-Evasion in a Polygonal Room with a Door", Proc. 15th ACM Symp. on Comp. Geom., 1999, pp. 281-290.

[14] V. Nalwa, "Line-Drawing Interpretation: A Mathematical Framework", Int. J. Computer Vision, 2(2), 1988, pp. 103-124.

[15] S.-M. Park, J.-H. Lee and K.-Y. Chwa, "A Characterization of the Class of Polygons Searchable by a 1-searcher", Technical Report TR-2000-160, CS Department, KAIST, 2000.

[16] S.-M. Park, J.-H. Lee and K.-Y. Chwa, "Visibility-Based Pursuit-Evasion in a Polygonal Region by a Searcher", Automata, Languages and Programming, 2001, pp. 456-468.

[17] S. Petitjean, J. Ponce, and D. Kriegman, "Computing Exact Aspect Graphs of Curved Objects: Algebraic Surfaces", Int. J. of Computer Vision, 9(3), 1992, pp. 231-255.

[18] H. Plantinga and C. Dyer, "Visibility, Occlusion, and the Aspect Graph", Int. J. of Computer Vision, 5(2), 1990, pp. 137-160.

[19] I. Suzuki and M. Yamashita, "Searching for a Mobile Intruder in a Polygonal Region", SIAM J. Comput., 21(5), 1992, pp. 863-888.

[20] R. Thom, *Structural Stability and Morphogenesis*, Benjamin, New York, 1972.