Dynamic Visibility in an Implicit Framework

Richard Tsai*†, Li-Tien Cheng‡, Paul Burchard§,
Stanley Osher*¶, and Guillermo Sapiro||

February 2002

Abstract

We investigate the problem of determining visible regions in two or three dimensional space given a set of obstacles and a moving vantage point. This is of importance in several fields of study including rendering in computer graphics, etching in materials construction, and navigation. Our approach to this problem is through an implicit framework, where the obstacles are represented by a level set function. An efficient generic multiscale level set method is developed to generate the visible and invisible regions in space. Furthermore, we study the dynamics of shadow boundaries on the surfaces of the obstacles using special level set techniques when the vantage point moves with a given trajectory. In all of these situations, topological changes such as merging and breaking occur in the regions of interest. These are automatically handled by the level set framework here proposed. Finally, we obtain additional useful information through simple operations in the level set framework.

---

[*] Research supported by ONR N00014-97-1-0027, DARPA/NSF VIP grant NSF DMS 9615854 and ARO DAAG 55-98-1-0323

[†] Department of Mathematics, University of California Los Angeles, Los Angeles, California 90095, email:ytsai@math.ucla.edu

[‡] Department of Mathematics, UCSD, La Jolla, CA 92093-0112. Email: lcheng@math.ucsd.edu

[§] Department of Mathematics, University of California Los Angeles, Los Angeles, California 90095, email:burchard@math.ucla.edu

[¶] Department of Mathematics, University of California Los Angeles, Los Angeles, California 90095, email:sjo@math.ucla.edu

[||] Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, email: guille@ece.umn.edu. Supported by ONR,NSF, PEC ASE, and CAREER.

Contents

1 Introduction

In this paper, we consider the visibility problem described as follows: given a collection of hypersurfaces representing the surfaces of objects, called the occluders, in two or three dimensional space, determine the regions of space or on the surfaces visible to a given observer. In real world applications, this problem must be solved quickly and efficiently, preferably in real time. Generalizations of the visibility problem are just as, if not more, important, such as the case of a moving rather than static observer and the determination of regions visible for all time or invisible for all time in this situation. However, we begin with the basic visibility problem for simplicity, and address parts of the dynamic problem later on. Incidentally, the visibility problem can be reformulated into a problem of determining light and dark regions given a point light source. We occasionally consider this point of view for clarification.

Under this point of view, a more precise set of assumptions we make in the visibility problem includes a space composed of a homogeneous medium and objects with nonreflecting and nondiffracting surfaces. Furthermore, we disregard interference, assuming that the distances between objects are large compared to the wavelength of light. Under these conditions, light rays travel in straight lines and are obliterated upon contact with the surface of an object. Thus a point is called visible with respect to a vantage point, the observer, if the line segment between the point and the vantage point does not intersect any of the obstructing objects or their surfaces in space.

The need for visibility information

Even under these simplifying assumptions, the visibility problem arises as a crucial part of numerous applications in different scientific fields, including rendering, visualization [16], etching [1], the modeling of melting ice [7], surveillance, navigation, and inverse problems, to name a few. In the case of computer graphics and rendering, for example, determination of the visible portions of object surfaces allows for those portions alone to be rendered, thus saving a lot of costly computation. Additionally, there are recent variational formulations for surface reconstructions that require the solution of the visibility problem [17].

While explicit surfaces, for example triangulated surfaces, are used in a majority of computer graphics and vision applications, implicitly represented surfaces are gaining more attention. The advantages can be seen in the automatic resolution of surfaces as well as the incorporation of geometric information and the handling of various surface topologies afforded, for example, by a level set framework (see, e.g., [6, 9, 15, 16, 17, 26]). Currently there are numerous algorithms for solving the visibility problem using explicit surface representations. For example, the work of [12] and [14] uses linearity to process triangulated surfaces. A detailed review of related work on the visibility problem, especially concerning explicit surfaces, can be found in [13]. Furthermore, there are a variety of visibility algorithms from computational geometry (see, e.g., [2, 3]). Visibility algorithms for implicit surfaces mostly consist of sending rays out from the point source and testing for intersections with the surfaces of the objects using information arising from the implicit formulation. Our proposed method for ray tracing is different. In essence, we send out rays in an implicit manner so as to propagate the causality relation of visibility. We describe this approach in more detail below.

Level set approach

We propose to solve the static and dynamic visibility problems in an implicit manner using a level set formulation. The level set method was first proposed by Osher and Sethian [20] as a PDE (Partial Differential Equation) based numerical device to capture moving interfaces. Over the years, a level set calculus has been developed that allows for the application of the level set method to a multitude of problems and situations. See, for example, the review paper [19] for an overview on the basics as well as recent advances in level set methods. We only recapitulate here that implicit PDE approaches such as the level set approach retain the important self interpolating property when propagating interfaces, thus obtaining good resolution of the interfaces. Furthermore, Boolean operations on sets, including finding curves of intersections and the trimming commonly required in CAD (Computer Aided Design) are easy to implement in a level set framework.

In our case of visibility, a real valued two or three dimensional function ϕ, called the level set function, is introduced. The zero level set of this function represents the surfaces of the occluding objects. Furthermore, we require that the points where ϕ is negative represent the interior of the objects. Several algorithms have been developed in the literature to efficiently obtain this representation. A level set method for visibility will use this function ϕ whenever the objects are considered.

One idea in determining whether a point is visible to a given observer is to compare the geodesic and Euclidean distances between the observer and that point. See [24] for an example of this approach. The geodesic distance between two points is the distance in the space in the presence of obstacles, namely the objects. Let $\mathbf{x}$ represent the point of interest and $\mathbf{x_o}$ represent the observer point. The geodesic distance can thus be calculated by solving the Eikonal equation

$$H(\phi)|\nabla u| = 1,$$

where H is the one dimensional Heaviside function, with condition $u(\mathbf{x_o}) = 0$. Thus the point $\mathbf{x}$ is occluded if and only if

$$u(\mathbf{x}) > |\mathbf{x} - \mathbf{x_o}|.$$

However, this algorithm is at best $O(N \log N)$, where N is the number of grid points (see, e.g., [25]). This may not be optimal, making it too slow for applications requiring real time computations. Furthermore, numerical implementation of the Heaviside function may cause problems for accuracy.

Our level set approach

We present here a few level set based algorithms for determining various types of visibility information in any dimension, though three dimensions is probably the one of interest. We first introduce a multiresolution algorithm for solving the visibility problem for a given fixed vantage point. This algorithm constructs the occlusion boundary, the interface separating visible from invisible. At each resolution level, we solve a radially defined causality relation on a given grid in one pass, obtaining not only a conservative estimate of the visible and invisible regions but a locally second order approximation of the occlusion boundary. In addition, our algorithm is independent of both the convexity of the occluders and the grid geometry, and its parallelization is straightforward.

The level set framework is especially important as it handles occluder fusion, where the occlusion boundary merges during the construction process. Furthermore, the implicit representation, though not as effective on occluders which are open surfaces, can still handle this case by considering them as very thin hypersurfaces.

Dynamic visibility

In the second part of the paper, we extend our study to the dynamic visibility problem. In this case, we consider a moving vantage point. Obviously the static visibility problem can be applied at each time to solve this problem, and our algorithm can be used to solve it efficiently enough. However, this static approach does not give us other useful information about the dynamics; for instace, how fast a point in space will become visible or invisible. In many cases, the problem can be solved even faster if the visibility at a previous time is used effectively to produce visibility at future time.

Thus we study the dynamics of curves on the occluders that separate light and dark regions on the occluders. The curves in fact can be represented using a level set approach, following the work of [5, 8, 10]. We also study other types of curves, called horizons, and their motions which form a superset of the curves separating light from dark but can be constructed quickly. We rigorously derive motion laws for all these types of curves and evolve them under the level set framework. This framework allows for topological changes which may occur in the curves and its self interpolating property automatically produces well resolved results. Finally, we derive an emergence-time estimate to predict an occluded object's emergence into view. Thus our work complements the book of Cipolla and Giblin [11] which discusses the reconstruction of shape from the perspective (orthogonal) projection of the horizons. This is exactly what we are trying to evolve.

Notations

Through out this paper, we use the following notation:

- The space in which we work will be $\mathbb{R}^d$, where $d = 2$ or 3.
- $\mathbf{x_o}$ denotes the position of the vantage point, or observer. We further assume that $\mathbf{x_o}$ never lies in the interior of the objects.
- Ω is a set of connected domains whose closure denotes the objects in question. Furthermore, let $\Gamma = \partial\Omega$.
- ϕ denotes the level set function representing the objects of interest. We may further assume that ϕ is the signed distance function to Γ. This particular level set function can be efficiently computed using fast algorithms such as the fast marching method of [25] or fast sweeping methods.
- We define the view direction vector pointing from $\mathbf{x_o}$ to $\mathbf{x}$ by $\nu(\mathbf{x_o}, \mathbf{x}) = (\mathbf{x} - \mathbf{x_o})/|\mathbf{x} - \mathbf{x_o}|$. When the context is clear, we will drop the arguments and write simply $\nu(\mathbf{x})$ or ν.
- Let $\mathbf{x_1}$ and $\mathbf{x_2}$ denote two points in space. We say $\mathbf{x_1} \preceq \mathbf{x_2}$ ($\mathbf{x_1}$ is "before" $\mathbf{x_2}$) if the conditions $\nu(\mathbf{x_o}, \mathbf{x_1}) = \nu(\mathbf{x_o}, \mathbf{x_2})$ and $|\mathbf{x_1} - \mathbf{x_o}| \leq |\mathbf{x_2} - \mathbf{x_o}|$ are satisfied. We also define the strict relation $\prec$ if the condition $|\mathbf{x_1} - \mathbf{x_o}| \leq |\mathbf{x_2} - \mathbf{x_o}|$ above is replaced by $|\mathbf{x_1} - \mathbf{x_o}| < |\mathbf{x_2} - \mathbf{x_o}|$.
- A point $\mathbf{y} \in \Gamma$ is called a horizon point if and only if $\nu(\mathbf{x_o}, \mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) = 0$, where $\mathbf{n}(\mathbf{y})$ is the outer normal of Γ at $\mathbf{y}$. The horizon thus refers to the set of horizon points.
- A point $\mathbf{y} \in \Gamma$ is a cast horizon point if and only if there is a point $\mathbf{y}^*$ such that: 1) $\mathbf{y}^* \prec \mathbf{y}$ and 2) $\mathbf{y}^*$ is a horizon point. The cast horizon thus refers to the set of cast horizon points.

- The visible contour refers to the set of visible points of the horizons and cast horizons.

2 Implicit ray tracing

We now set up the foundation of our approach and derive properties of ray tracing of a single point source in an implicit framework. The motivation is as follows: we observe that the visibility status of points sharing the same radial direction centered at the vantage point satisfy a causality condition. This means if a point is occluded, then all other points farther away from the vantage point in the same radial direction are also occluded, i.e., if $\mathbf{x_1}$ is occluded and $\mathbf{x_1} \preceq \mathbf{x_2}$, then $\mathbf{x_2}$ is also occluded.

This fact can be described more rigorously as follows. Define

$$\rho(\mathbf{p}) = \begin{cases} \min_{\mathbf{x}\in\mathbb{R}^d}\{|\mathbf{x}-\mathbf{x_o}| : \nu(\mathbf{x_o},\mathbf{x}) = \mathbf{p}, \phi(\mathbf{x}) \leq 0\} & \text{if exists} \\ \infty & \text{otherwise,} \end{cases} \tag{1}$$

giving the distance between $\mathbf{x_o}$ and the closest point on Γ in the direction of $\mathbf{p}$ from $\mathbf{x_o}$. Thus a given point $\mathbf{x}$ is invisible if $\rho(\nu(\mathbf{x},\mathbf{x_o})) \leq |\mathbf{x}-\mathbf{x_o}|$. Refer to Figure 1 for an example and clarification. Therefore, we can define the visibility indicator

$$\Xi(\mathbf{x},\mathbf{x_o}) := \rho(\nu(\mathbf{x},\mathbf{x_o})) - |\mathbf{x}-\mathbf{x_o}|,$$

so that $\{\Xi \geq 0\}$ is the set of visible regions in $\mathbb{R}^d$ and $\{\Xi < 0\}$ is the set of occluded regions.

From another view point, the problem becomes: compute

$$\psi(\mathbf{x}) := \min_{\xi\in L(\mathbf{x_o},\mathbf{x})} \phi(\xi),$$

where $L(\mathbf{x_o},\mathbf{x})$ is the line segment connecting $\mathbf{x_o}$ and $\mathbf{x}$. Thus if $\psi(\mathbf{x})$ is negative, then $\mathbf{x}$ is occluded. In fact, our implicit ray tracing algorithm is is an approximation of this formula.

Implicit formulation

Our implicit framework encodes visibility information in a Lipschitz continuous function ψ so that a point $\mathbf{y}$ is visible if $\psi(\mathbf{y}) \geq 0$ and invisible if $\psi(\mathbf{y}) < 0$. Thus we can compute the value of $\psi(\mathbf{x})$ by

$$\psi(\mathbf{x}) = \min(\psi(\mathbf{x}'), \phi(\mathbf{x})) \tag{2}$$

where $\mathbf{x}'$ is some point "right before" $\mathbf{x}$ in the ray direction. We can therefore start from the vantage point $\mathbf{x_o}$ and update the grid points following the ray directions outwards. A simple algorithm for this reads:

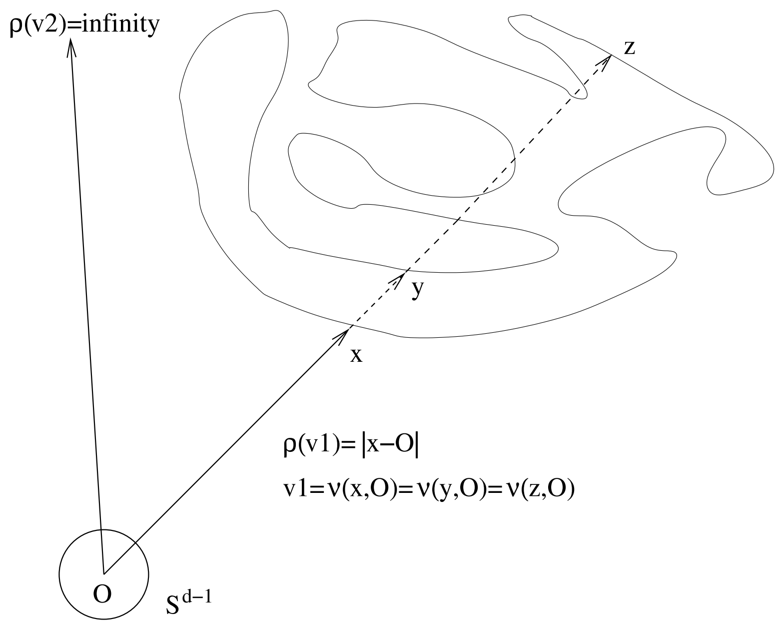

Figure 1: Illustration of ρ

1. Set $\psi(\mathbf{x_o}) = \phi(\mathbf{x_o})$.
2. Do a star-shaped[1] updating sequence on the grid.
3. For each grid point $\mathbf{x}$, choose $\mathbf{x}'$ depending on the grid geometry.
4. Compute the value of $\psi(\mathbf{x})$ via (2).

Each grid node is visited in a specified order that maintains the causality. As long as the updated grid nodes form a star-shaped region centered at the vantage point, causality is maintained. See the Appendix for an example of such an updating method. Due to the minimization and the linear interpolation (see Section 5.1) used to find $\mathbf{x}'$, the algorithm is l_∞-stable. More precisely, we have

$$\phi_{\text{int}}\left(\mathbf{x}'\right) \leq \max\{|\phi(x_l)| : x_l \text{ are the points used in the interpolation}\},$$

where ϕ_{int} is the interpolant we constructed. Please see Section 5.1. Figure 2 shows what ψ should look like in a one space dimension setting.

[1] See 5.2

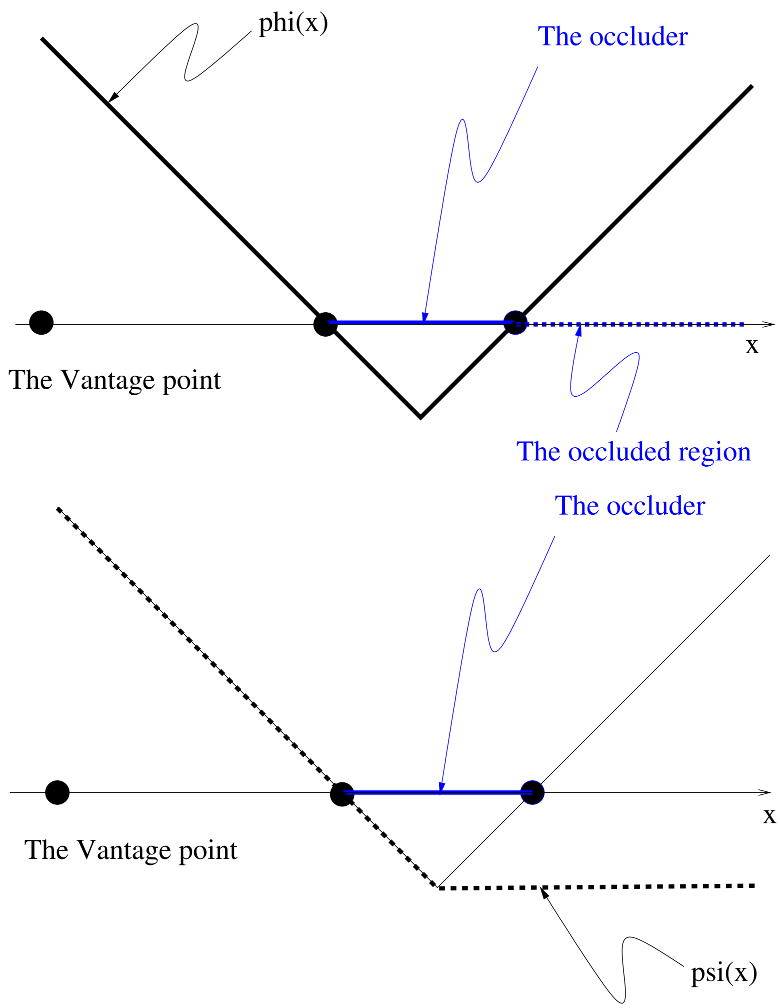

Figure 2: A demonstration of the motivation of our implicit ray tracing algorithm in one dimension.

Volumetric visibility processing

If we want to determine volumetric visibility information, i.e., we want to find regions that are occluded from a given rectangular region (view cell), the above algorithm can be modified for this purpose. For simplicity, assume that the view cell degenerates to a line with end points $\mathbf{x}_1$ and $\mathbf{x}_2$. At each point $\mathbf{x}$, we compute $\mathbf{x}'_1$ and $\mathbf{x}'_2$ as in step 2 with respect to $\mathbf{x}_1$ and $\mathbf{x}_2$. The update formula then becomes: $\psi(\mathbf{x}) = \min(\phi(\mathbf{x}), \max(\psi(\mathbf{x}'_1), \psi(\mathbf{x}'_2)))$.

2.1 Multi-resolution calculation

Finding inside/outside

Any multi-resolution approach of the visibility problem requires the skipping of large regions which we know a priori are either visible or invisible. This hinges upon the ability to determine whether any given voxel is completely "inside" or "outside" of the objects. This can be done conservatively with the help of the Lipschitz constant of the embedding level set function.

Let C be the Lipschitz constant of ϕ. Let $\mathbf{x}_c$ be the center point and $\mathbf{x}_i$ the vertices of the given voxel V. If

$$\phi(\mathbf{x}_i) + C|\mathbf{x}_c - \mathbf{x}_i| < 0 \ \ \forall\, i, \tag{3}$$

then we know $\phi|_V < 0$ ($V \subset \{\phi < 0\}$). Conversely, if

$$\phi(\mathbf{x}_i) + C|\mathbf{x}_c - \mathbf{x}_i| > 0 \ \ \forall\, i, \tag{4}$$

then we know $\phi|_V > 0$. Since our embedding function is the signed distance function, the Lipschitz constant $C = 1$.

Correspondingly, we also have:

$$\phi|_V < 0 \ \text{ if } \ \phi(\mathbf{x}_c) + C|\mathbf{x}_i - \mathbf{x}_c| < 0 \ \ \forall\, i,$$

and

$$\phi|_V > 0 \ \text{ if } \ \phi(\mathbf{x}_c) + C|\mathbf{x}_i - \mathbf{x}_c| > 0 \ \ \forall\, i.$$

A voxel V is occluded if it lies completely inside the an object or if it is "behind" an occluded voxel $\tilde{V}$.This idea can be implemented by a careful reinterpretation of formula (2). A similar condition for determining completely visible voxels can also be easily derived.

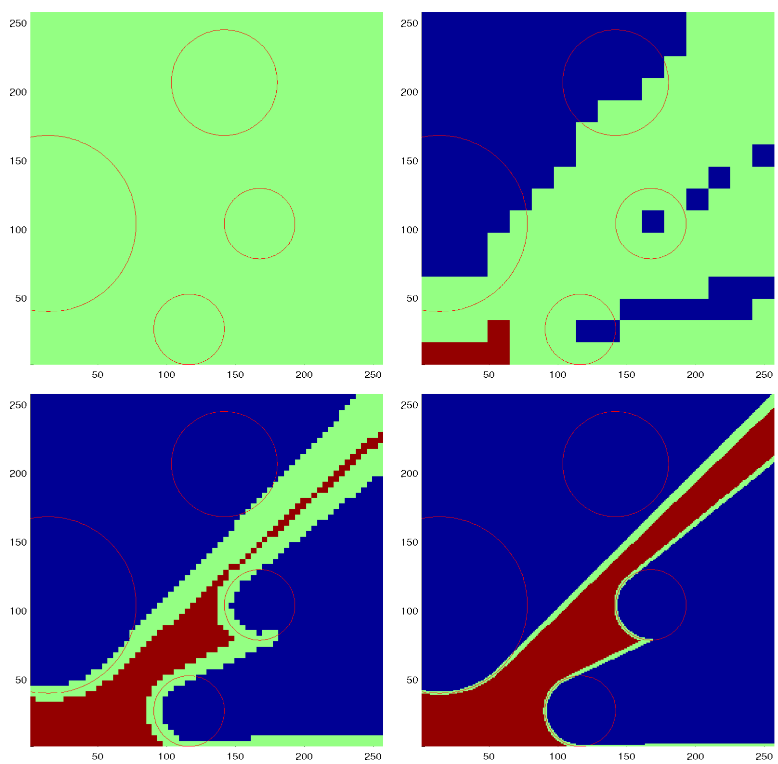

Figure 3: This is a schematic diagram for the multi-resolution algorithm. Occluded voxels are depicted in blue and visible ones in red. The regions are target for next level refinement.to be refine. The red curves represent the boundaries of the occluders, and the vantage point is positioned at $(1, 1)$. The sizes of the voxels are: 64×64, 16×16,4×4, and 1×1.

2.2 A Multi-resolution algorithm

Let h give the resolution of our grid such that smaller h means high resolution (i.e. finer grid). In practice, we can simply set h to be the mesh size. Under resolution h, we use (i^h, j^h, k^h) to denote the grid indices and V^h for a voxel in the grid. We will drop the superscripts of these indices when the context is clear.

Algorithm: (Multi-resolution visibility sweeping)

Let $\{h_l : h_l > h_{l+1}, l = 0, \cdots, m\}$ be the set of resolutions of interest. For each resolution level h, decending from h_0 to h_m,

1. Compute ψ on the (i^h, j^h, k^h) which do not lie on a voxel marked either visible or occluded.
2. For each voxel V^h which does not lie on a voxel marked either visible or occluded, mark V^h to be visible or occluded according to formulas (3) and (4).

We remark that the Lipschitz constant of the linearly interpolated ψ is can be taken from that of ϕ. Therefore, Step 2 above is well defined. Please see Figure 3 for a demonstration of this algorithm.

As for complexity, the multiresolution algorithm for constructing visiblity information should be an $O(N^{d-1} \log N)$ algorithm in terms of speed. Here $N = 1/h$ where h is the smallest spatial stepsize used in the multiresolution framework and d is the dimension of the space. The N^{d-1} part of the complexity comes from the fact that a codimension one hypersurface in d dimensional space is being generated under fast sweeping and the $\log N$ part comes from multiresolution. The memory allocation of our algorithm is also $O(N^{d-1} \log N)$, with the $\log N$ part once again due to multiresolution. In practice, our algorithms have proven to be very fast, obtaining detailed visiblity information in almost real-time. For example, it takes less than a second to perform this algorithm running on only one resolution (meaning no multi-resolution), on a grid with 100^3 cells, on a moderate PC.

2.3 Multi-scale considerations

If we consider the visibility problem in applications related to human vision, such as 3D virtual environment rendering, it is natural to put a scale parameter into the size of the objects related to the distance of the object from the vantage point. We want to ignore certain **isolated and small** objects that are **far away** from the vantage point using this information. It is important to notice that a collection of closely positioned small objects can form a visible ensemble, seen for example in clouds and trees.

Using the level set representation of the virtual environment in conjunction with PDEs, we are able to deal with this issue easily without explicitly considering each object separately. The idea is to *dilate* the interface first so that small objects can merge to form ensembles of larger size. We then *shrink* the interfaces (one possibility is to perform curvature driven motion) such that remaining small objects will disappear. This approach follows the regularization effect of viscosity solution theory for Hamilton-Jacobi Equations. It is basic mathematical morphology, and can be done easily, see e.g. [4][23].

3 Dynamic visibility

We now consider the case in which the vantage point is moving. Naturally visibility information changes according to the position of the vantage point. We are interested in how visibility changes and when hidden objects become visible. This amounts to studying how the boundaries between visible and invisible regions move with respect to the vantage point motion. We first remark that these boundaries are hypersurfaces in regions outside of the occluders, and that the Gaussian curvature on such surfaces is 0.

For a single convex object, the horizon determines the visibility information on the surface. Therefore, tracking the motion of the horizon for all time gives us incremental information on the change of the visible portion of the object.

We formulate the visibility problem so that the points which are on the boundaries of the visible regions on the surfaces can easily be identified. The dynamics of these points are derived so that one can track the visible regions according to the motion of the vantage point $\mathbf{x_o}$.

The points forming the boundaries of visible regions on given surfaces can be placed into two categories:

- points that are part of the horizon;
- points that border *shadows cast by some surface* (cast horizon).

Thus, two types of motions need to be investigated, namely, that of the horizon and of the cast horizon. The motion of the horizon is characterized by the orthogonality constraint and it, in turn, becomes a part of the constraints of the cast horizon motion.

In a level set formulation, we want to create a level set function whose zero level set captures the points described above. We need a description relating each point on the cast horizon to a point on the horizon of the surface casting the shadow.

Furthermore, our description should be global, that is, quantities should vary "continuously" with respect to points not on the surface. This requirement is essential for the success of our level set formulation.

Assuming that there is no singularity in the velocity field of the horizon or cast horizon motion, and there are no other considerations, the level set approach of tracking the visible contours is optimal. With the fast sweeping algorithms and local storage strategies, the complexity of the level set approach to track the horizon and cast horizon curves is formally $\mathcal{O}(N)$ in operation counts and in storage. Here, the number N is the number of points used to resolve the curves. In actual applications, there are other aspects that affects the overall complexity of this approach. We will address this point in a later subsection.

3.1 Finding the horizon and the cast horizon implicitly

Horizons and cast horizons are objects of codimension 2. We may therefore categorize these objects by the intersection of the zeros of two level set functions. Furthermore, for numerical reasons, we want the two zero level sets to be more or less orthogonal to each other near their intersection.

Finding the horizon

We extend the orthogonality condition that defines the horizon and arrive at

$$\hbar(\mathbf{x}, t) = (\mathbf{x} - \mathbf{x}_0) \cdot \nabla\phi(\mathbf{x}). \tag{5}$$

$\hbar$ determines the visibility of any convex object embedded in ϕ :

$$\{\hbar(\mathbf{x}) \leq 0\} \bigcap \{\phi = 0\} \iff \text{visible}.$$

In general cases, where there are multiple objects (convex and nonconvex), $\hbar$ does not give exact visibility information anymore. It just provides local visibility information just as local extrema may not be absolute extrema. Thus clearly, for objects hiding completely behind other objects, $\hbar$ will still be non-negative on the parts of the surface facing the source. Instead, $\hbar$ gives a conservative "estimate" of the shadow:

$$\{\hbar(\mathbf{x}) > 0\} \bigcap \{\phi = 0\} \implies \text{invisible}.$$

Thus the visible horizon is a subset of $\{\phi = 0\} \bigcap \{\hbar = 0\} \bigcap \{\psi \geq 0\}$, where ψ is the visibility function coming from our static algorithm. Figure 4 gives an example of horizons found this way.

Finding the cast horizon

How do we find the cast horizon? The idea is to overshoot the shadow boundaries generated by the visible horizon when it hits another part of Γ, creating a level set piece $\tilde{\psi}$ in that neighborhood, and then propagate ψ as usual. $\{\tilde{\psi} = 0\}$ will cut through Γ on the cast horizon, therefore providing an implicit representation of it. We can even make $\{\tilde{\psi} = 0\}$ perpendicular to Γ locally around the intersection by iterating on the following PDE used in [18]:

$$\tilde{\psi}_\tau + \text{sgn}\,(\phi)\nabla\tilde{\psi} \cdot \frac{\nabla\phi}{|\nabla\phi|} = 0.$$

A more direct approach is to define $\tilde{\psi}$ on a grid point to be the "upwind" value of ϕ. This will introduce an overshoot of the size of a mesh size. Alternatively, we notice that the occlusion generated by the set $\{\hbar \geq 0\} \bigcap \{\phi \leq 0\}$ is the same[2] as $\{\phi \leq 0\}$. Therefore, we can define $\tilde{\psi}$ from the result of our algorithm under the configuration $\{\hbar \geq 0\} \bigcap \{\phi \leq 0\}$. Figure 4 shows the operations described above in a simple two circle setting.

With these characterizations and the visibility result, we can easily identify the visible contours. See Figures 5, 6, and Figures 7, 8[3] for examples. In these figures, the visible portions of the horizons and the cast horizons are depicted as cyan and yellow curves respectively. A green circle is drawn to reveal the location of the vantage point in each setting. The boundaries between visible and invisible regions are represented by blue surfaces. We observe that the blue surfaces cut through the objects exactly at the visible contours.

3.2 The dynamics of the horizon

Let $\mathbf{x}_o(t)$ be the position of the vantage point and $\mathbf{x}(t)$ be a corresponding point on the horizon at time t. We first consider a single convex occluder Ω embedded by the signed distance function ϕ. Let $n(x)$ denote the outer normal of $\partial\Omega$ at $\mathbf{x}$. This translates into the following constraints on $\mathbf{x}(t)$:

$$\begin{cases} \phi(\mathbf{x}(t)) = 0 \\ (\mathbf{x} - \mathbf{x_o}) \cdot \nabla\phi(\mathbf{x}) = 0 \end{cases} \tag{6}$$

In two dimensions, we can invert the above constraints and derive that

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \frac{1}{\kappa}\frac{\dot{\mathbf{x}}_\mathbf{o} \cdot n(\mathbf{x})}{|\mathbf{x} - \mathbf{x}_o|}\nu(\mathbf{x}). \tag{7}$$

[2] modulo a small subset of $\{\phi \leq 0\}$, which we know is invisible by definition.

[3] The terrain data is obtained from `ftp://ftp.research.microsoft.com/users/hhoppe/data/gcanyon/`.

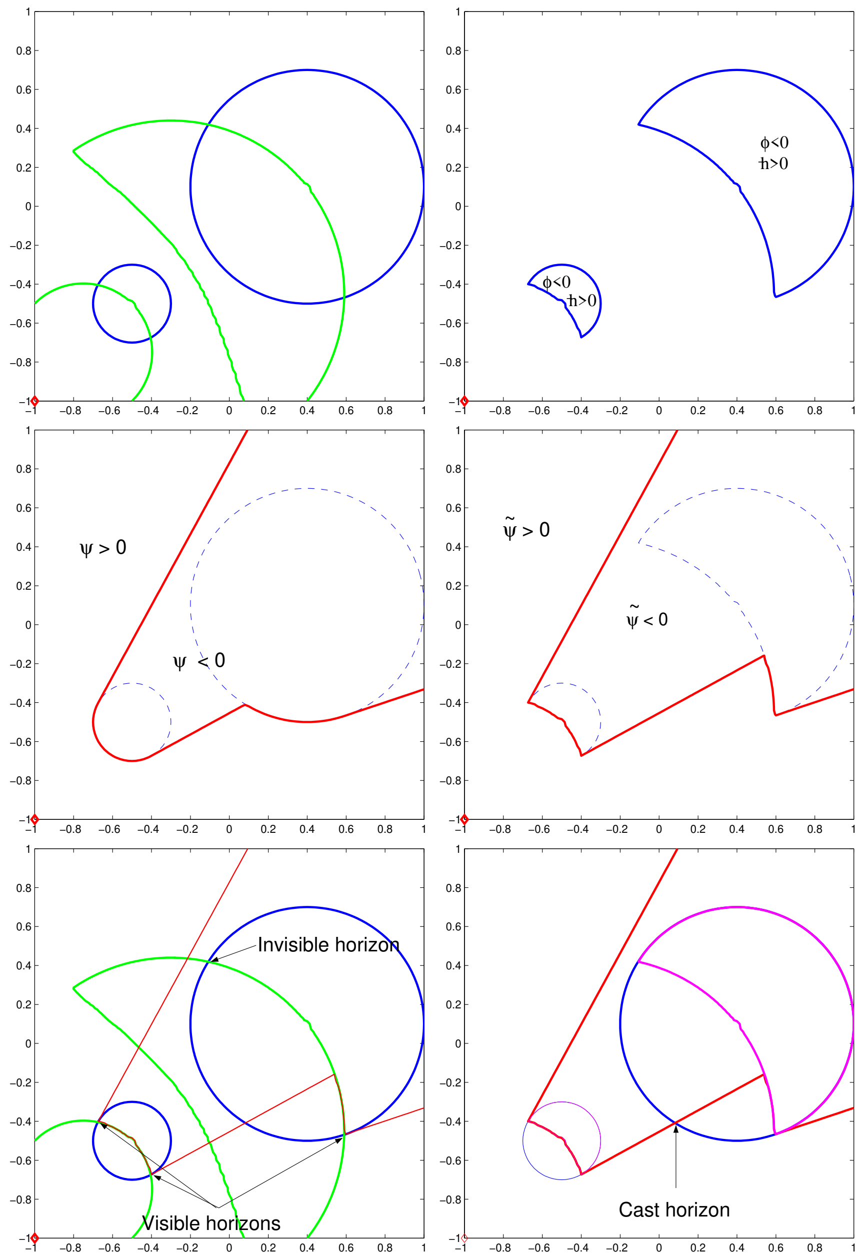

Figure 4: Finding the visible horizons and their casts. The occluders are the two circles depicted by the blue curves, and the vantage point is located at $(-1,-1)$. The green curves are the zero level set of $\hbar$. Visible horizons and their casts are characterized by the intersections of different level set functions as described in the text.

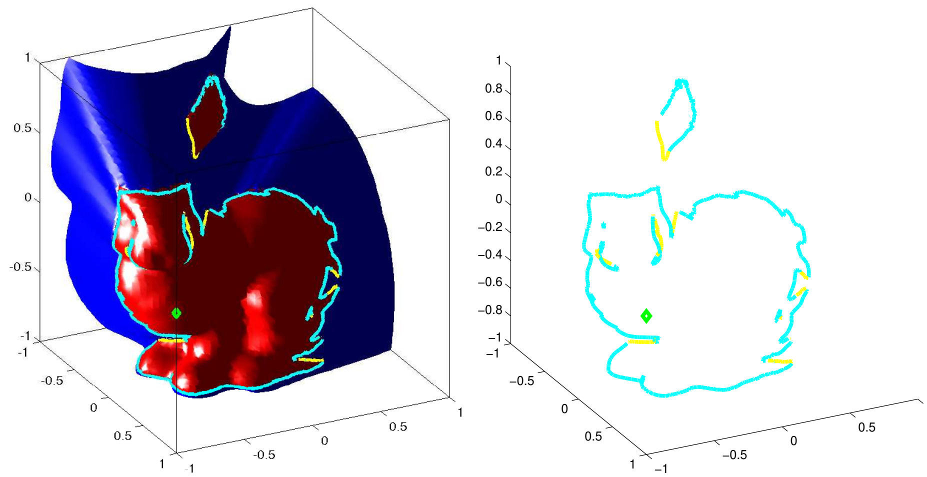

Figure 5: Visible contour (portions of horizon and cast horizon that are visible)

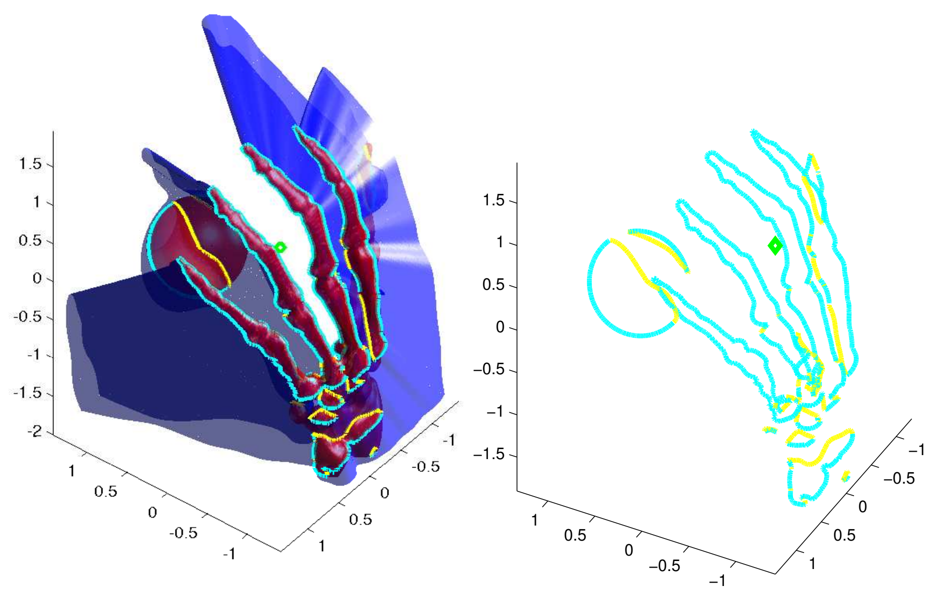

Figure 6: Visible contour (portions of horizon and cast horizon that are visible)

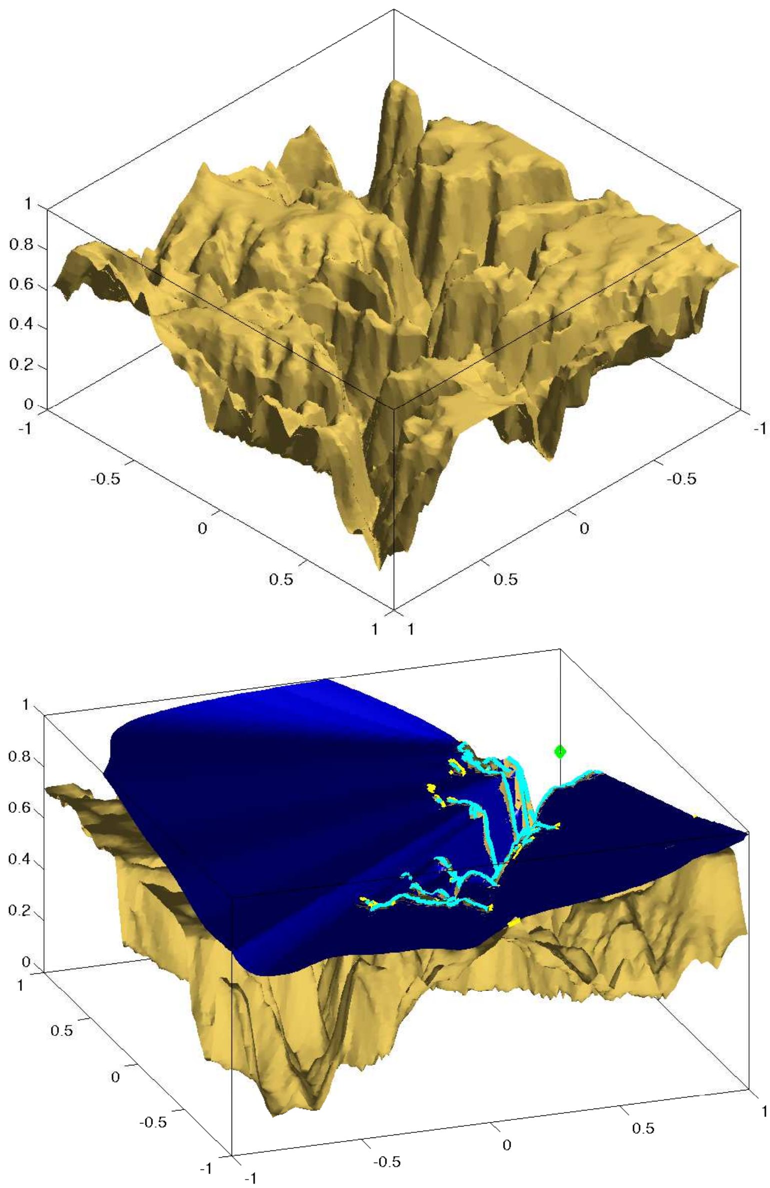

Figure 7: Horizons and cast horizons obtained from the elevation data of Grand Canyon.

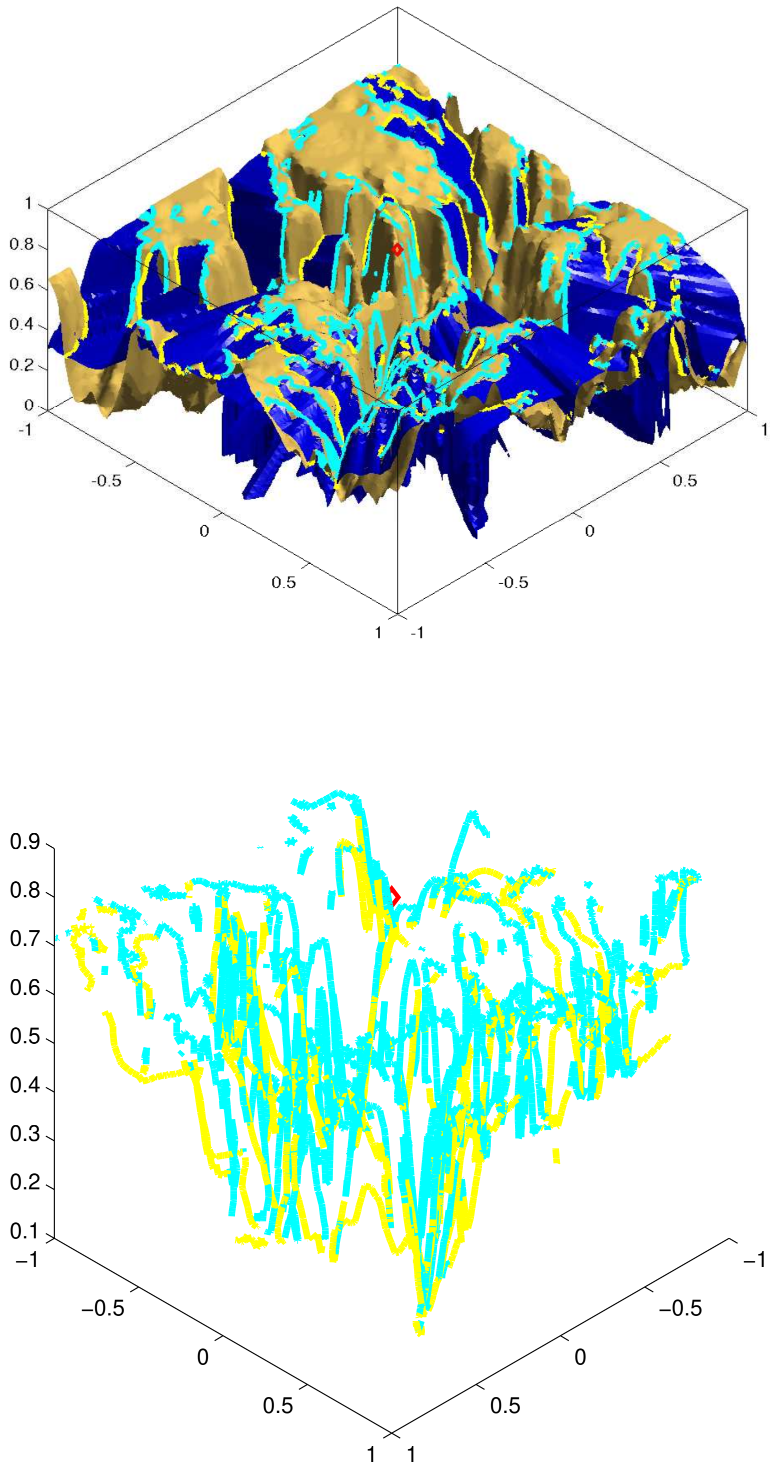

Figure 8: Horizons and cast horizons obtained from the elevation data of Grand Canyon.

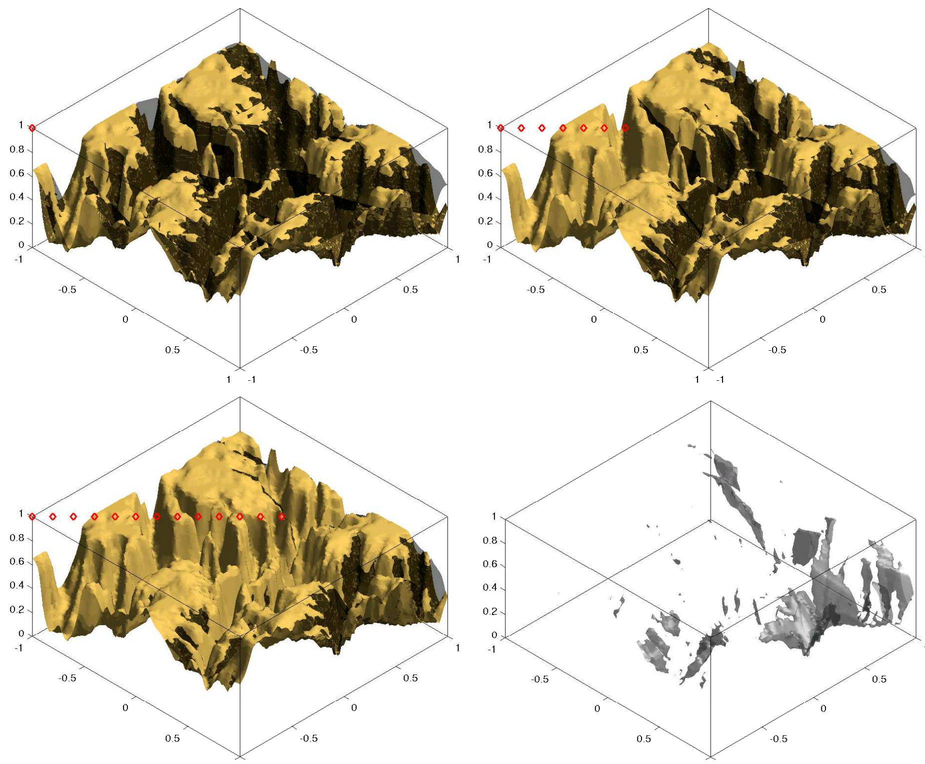

Figure 9: By taking the intersection of the occlusion during a trajectory of the observer, we can find the cumulative occlusion easily and efficiently. The following pictures show a progression of the cumulative occlusion subject to an observer ("spy plane") moving across a region of Grand Canyon.

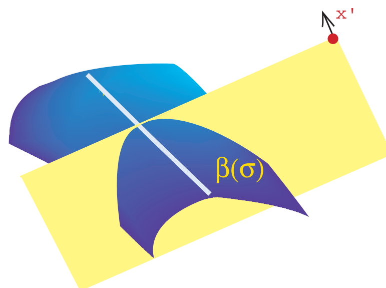

Figure 10:

Here, κ is the curvature of the occluding surface at $\mathbf{x}$. In three space dimensions, the horizon becomes a closed curve $\Gamma(s) = \mathbf{x}(s,t)$, where s is the arc length of $\Gamma(s)$. Let P be the plane tangent to $\dot{x}_o$, passing through $\Gamma(s)$ and x_o. Let $\beta(\sigma)$ be the curve on the intersection of P and $\partial\Omega$. Then, locally at t and x, we have a two dimensional visibility problem on the plane P, in which $\beta(\sigma)$ defines the boundary of the objects. Following this reasoning, κ should naturally be taken from $\beta(\sigma)$. See Figure 10.

Alternatively and more naturally under our level set formulation, we rederive the above motion law as

$$\dot{x} = \frac{II(x-x_0)}{|II(x-x_0)|^2}\left(\dot{x}_0 \cdot \frac{\nabla\phi}{|\nabla\phi|}\right), \tag{8}$$

where II is the *second fundamental form,* which can conveniently be extended to the other level sets and takes the form:

$$II = \frac{1}{|\nabla\phi|} P_{\nabla\phi} \nabla^2 \phi P_{\nabla\phi}.$$

Here, $P_{\nabla\phi}$ is the orthogonal projection matrix projecting vectors to the plane with normal vector parallel to $\nabla\phi$.

For a detailed derivation and implementation, please see sections 5.4, 5.3 and 5.7. Figure 11 shows a result of horizon motion on a nonconvex body.[4]

---

[4]For more exmaples on the horizon motion, please see `http://www.math.ucla.edu/ ytsai/math_page`

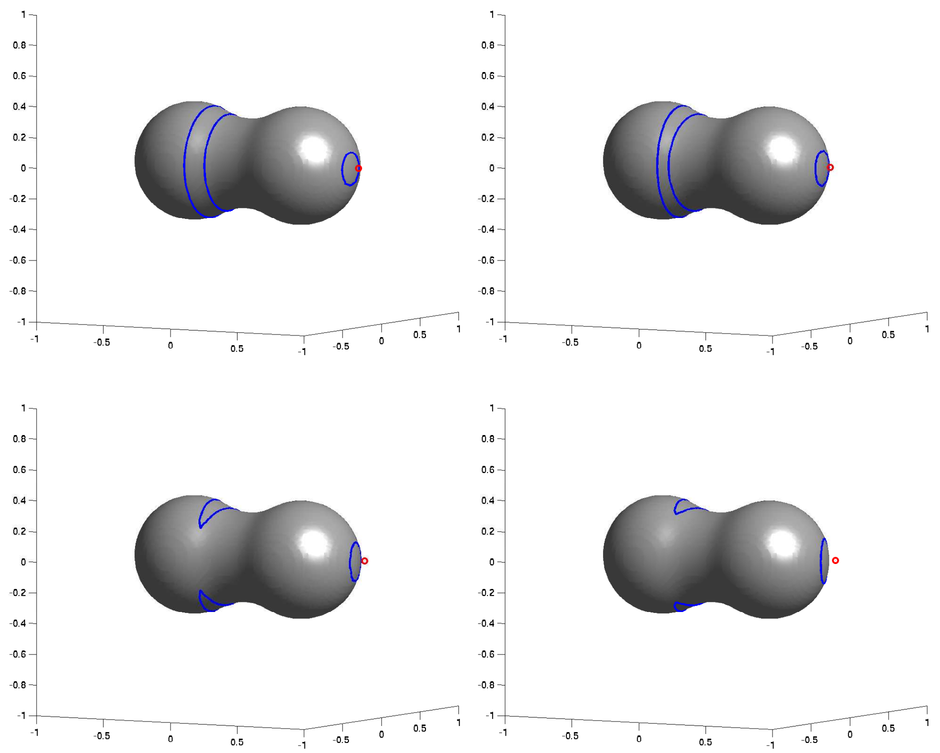

Figure 11: A example of moving horizon around a nonconvex occluder. Observe that the horizon curves break and change topology.

3.3 The dynamics of the cast horizon

Assume that $\mathbf{x}$ is a cast horizon point and $\mathbf{x}^*(\mathbf{x})$ is its generator. In two dimensions, the motion of $\mathbf{x}$ is determined by the following constraints:

$$\begin{cases} \phi(\mathbf{x}) = 0, \\ \frac{\mathbf{x}-\mathbf{x}_o}{|\mathbf{x}-\mathbf{x}_o|} = \frac{\mathbf{x}^*-\mathbf{x}_o}{|\mathbf{x}^*-\mathbf{x}_o|}. \end{cases} \tag{9}$$

Inverting, we find that the motion of the cast horizon can be written as follows:

$$\dot{\mathbf{x}} = \frac{1}{\nu \cdot \mathbf{n}(\mathbf{x})}\left(\frac{|\mathbf{r}|}{|\mathbf{r}^*|}\dot{\mathbf{r}}^* \cdot (\nu^*)^\perp + \dot{\mathbf{x}}_\mathbf{o} \cdot \nu^\perp\right)\mathbf{n}^\perp(\mathbf{x}), \tag{10}$$

where

$$\mathbf{n}^\perp(\mathbf{x}) := \begin{pmatrix} \phi_{x_2}(\mathbf{x}) \\ -\phi_{x_1}(\mathbf{x}) \end{pmatrix} / |\nabla\phi(\mathbf{x})|,$$

and similarly for $\nu^\perp$. See Section 5.5 for a detailed derivation. See Figure 12 for a computational result using this formula. We notice that these constraints also tell us how the shadow boundaries should move.

In three dimensions, we can reduce the instantaneous motion to a two dimensional problem on the "right" section of the surface following the reasoning given in the previous subsection.

Motions of the shadow boundaries

How does the shadow move in space? We can constrain a point on the shadow boundary to move only normal to the viewing direction (ergo, the shadow boundary):

$$\begin{cases} \mathbf{x}' \cdot \nu = 0, \\ \frac{\mathbf{x}-\mathbf{x}_o}{|\mathbf{x}-\mathbf{x}_o|} = \frac{\mathbf{x}^*-\mathbf{x}_o}{|\mathbf{x}^*-\mathbf{x}_o|}. \end{cases} \tag{11}$$

Motions of horizons and cast horizons of dynamic surfaces

We remark that we we are able to derive the motion laws of the visible contours even when the occluders are changing shapes. In this case, the embedding level set function ϕ is a function of space and time, $\phi(\mathbf{x}, t)$ and differentiating formulas (6) and (9) with respect to t will bring $\phi(\mathbf{x}, t)$ into the equations.

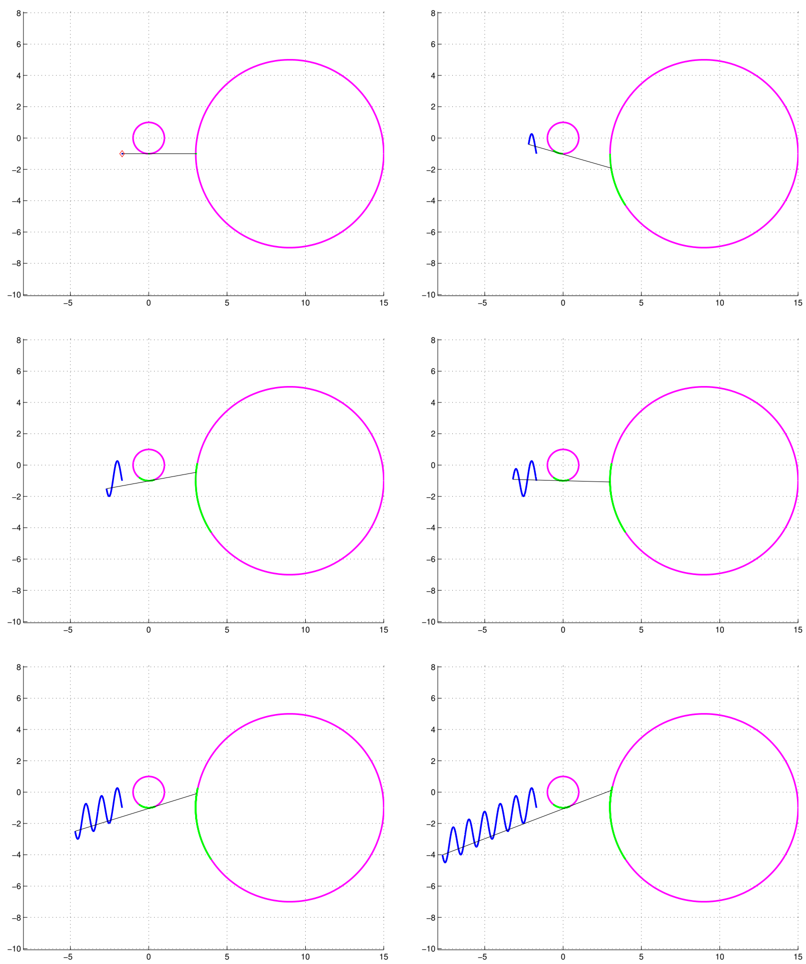

Figure 12: A result of tracking the horizon and cast horizon motion using the formulas derived in this paper. The blue curve represents the trajectory of the vantage point and the green curves represent the paths of the horizon and cast horizon. The black line links the current position of the vantage point and the cast horizon; it shows that the colinearity of the vantage point, the horizon and its cast is preserved.

3.4 Analysis of the motions

PDEs for moving the level set functions

Once we have the motion laws, we can extend the velocity to the domain near the surfaces and obtain the corresponding velocity field $\mathbf{v}(\mathbf{x})$. We then evolve the level set function(s) u in question by

$$u_t + \mathbf{v} \cdot \nabla u = 0.$$

Furthermore, the velocity fields for horizon and cast horion motions do not depend on the function u. In horizon motion, the velocity is a function of postion, time, $\dot{\mathbf{x}}_\mathbf{o}$, and the derivatives of ϕ, i.e. $\mathbf{v} = \mathbf{v}(t, \mathbf{x}, \mathbf{x_o}, \nabla\phi, D^2\phi)$. Furthermore, the level set function to be evolved is $\hbar$. In cast horizon motion, we have $\tilde{\mathbf{v}} = \tilde{\mathbf{v}}(t, \mathbf{x}, \mathbf{x_o}, \nabla\phi, D^2\phi, \hbar)$, and the level set function to be evolved is $\tilde{\psi}$. Therefore, we are evolving the following two level set equations:

$$\hbar_t + \mathbf{v}(\mathbf{t}, \mathbf{x}, \mathbf{x_o}, \nabla\phi, D^2\phi) \cdot \nabla\hbar = 0,$$

$$\tilde{\psi}_t + \tilde{\mathbf{v}}(\mathbf{t}, \mathbf{x}, \mathbf{x_o}, \nabla\phi, D^2\phi, \hbar) \cdot \nabla\tilde{\psi} = 0.$$

These are simple convection equations whose viscoity solutions are well studied, provided that the velocity fields are bounded. We only have to be careful near singularities.

Singularities in the velocity fields

Formula (7) reveals a few interesting facts. First, we notice that the speed of the horizon motion is inversely proportional to the normal curvature in the viewing direction and to the distance between the horizon and the vantage point. If the vantage point is moving in the tangent direction ν, the horizon will not move (since $\dot{\mathbf{x}}_o \cdot \mathbf{n} = 0$). The speed of the horizon motion becomes singular if the curvature of the surface at the horizon location becomes zero. On strictly convex objects, this will never happen. If we restrict our analysis to a single connected smooth non-convex object, we will see easily that at the instance in which a horizon point moves into the location where $\kappa = 0$, a neighborhood of this location becomes completely visible. This signifies the disappearance of the horizon point. If the course of the vantage point is reversed, we get the genesis of a new horizon point.

Formula (10) tells us that the motion of a cast horizon point becomes singular when it is a horizon point ($\nu \cdot \mathbf{n} = 0$). On a single non-convex smooth surface, this happens precisely when a horizon point and its cast across the concavity collide into each other at the location where $\kappa = 0$. In the setting where there are multiple strictly convex objects, this also describes the changing of the cast horizon into a visible horizon point which is previously invisible. Therefore, the singularities of the horizons and cast horizons describe a part of their genesis. A complete genesis of the visible contours includes another part, in which a hidden object suddenly becomes visible. We shall discuss this point in a later subsection.

3.5 Relating horizon and its shadow

To move the cast horizon, following the notation used in the previous section, we need to find $\mathbf{x}^*(\mathbf{x})$ for each point $\mathbf{x}$ on the cast horizon.

3.5.1 Explicit formula

$\mathbf{x}$ and $\mathbf{x}^*(\mathbf{x})$ are related by

$$\mathbf{x}^*(\mathbf{x}) := \mathbf{x} - r(\mathbf{x})\nu(\mathbf{x});$$

$r(\mathbf{x})$ can be computed by

$$r(\mathbf{x}) = |\mathbf{x} - \mathbf{x}_o| - \rho(\nu(\mathbf{x})),$$

where ν and ρ are defined as previously.

3.5.2 Implicit formulation

We follow the spirit of formula (2) introduced in Section 2 and propagate the link between the horizon and its cast shadow implicitly. Define $\vartheta : \mathbb{R}^d \mapsto \mathbb{R}^d$ to be

$$\vartheta(\mathbf{x}) := \begin{cases} \mathbf{x} & \text{if } \tilde{\psi}(\mathbf{x}) = \phi(\mathbf{x}) \\ \mathbf{x}' & \text{if } \tilde{\psi}(\mathbf{x}) = \tilde{\psi}(\mathbf{x}'). \end{cases}$$

$\tilde{\psi}$ is the function defined in Section 3.1. When we move the points $\mathbf{x}$ near the cast horizon, we also move the points $\vartheta(\mathbf{x})$, which are points near the horizon. By continuity around the cast horizon, we will have the right motion of the cast horizon.

3.6 Reinitialization and emergence-time estimate

It can be easily seen from figure 13 that a completely hidden object may suddenly become visible at a later time during the journey of the vantage point. At the time of emergence, we need to reinitialize our algorithm, i.e., we need to find the apparent contours on the newly emerged surfaces to get the correct visibility information.

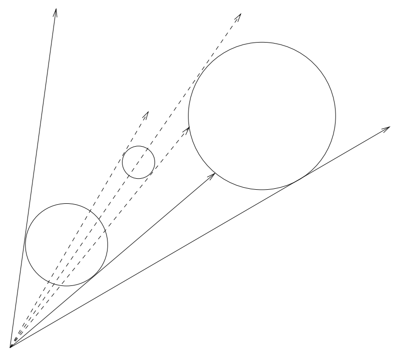

Figure 13: Model scenario I

An explicit reinitialization criterion

Assuming that we are merely tracking the visibility boundaries on the objects. How do we know when to initialize? We can formalize the reasoning as follows. We define the map $\mathcal{G}^{-1} : S^{d-1} \mapsto \{\mathbf{x}_i\}_{i=0}^{N(\theta)}$ such that

$$\mathcal{G}^{-1}(\theta) := \{\mathbf{x}_i \in \mathbb{R}^d : \nabla\phi(\mathbf{x}_i)/|\nabla\phi(\mathbf{x}_i)| = \theta \,\text{and}\, \phi(\mathbf{x}_i) = 0\}.$$

This map is the inverse of the Gauss map in the case that $\{\phi = 0\}$ is a strictly convex hypersurface. Let $\mathcal{S}$ be the set containing x and x^*. We reinitialize whenever there exists an $x \in S$ such that $\exists\, y \in \mathcal{G}(\nu(\mathbf{y}))$ with $\mathbf{x}^*(\mathbf{x}) \prec y \prec \mathbf{x}$.

This provides an explicit criterion for reinitialization. However, we can do better with our implicit formulation. This amounts to knowing 1) how the shadow moves 2) how far a hidden surface is from the shadow boundaries.

Emergence-time estimate

Given current vantage point position and its motion, we want to estimate the emergence time for an object that is occluded. We begin by assuming the the curvatures of the surfaces locally around the regions of interest are constant. The diagram in Figure 14 shows a model configuration: The small circle is initially occluded by the larger circle on the left. We want to estimate the time interval δt between this instance t_0 and the time $t_1 = t_0 + \delta t$ when the small circle first emerges into the scene.

Following the discussion above, consider a point $\mathbf{y}$ on the shadow boundaries away from the horizon such that $\nabla\phi(\mathbf{y}) \perp \nu(\mathbf{y})$ and $\nabla\phi(\mathbf{y}) \cdot \nabla\phi(\mathbf{y}(\mathbf{x})) > 0$. This is the point closest to some hidden part of the objects.

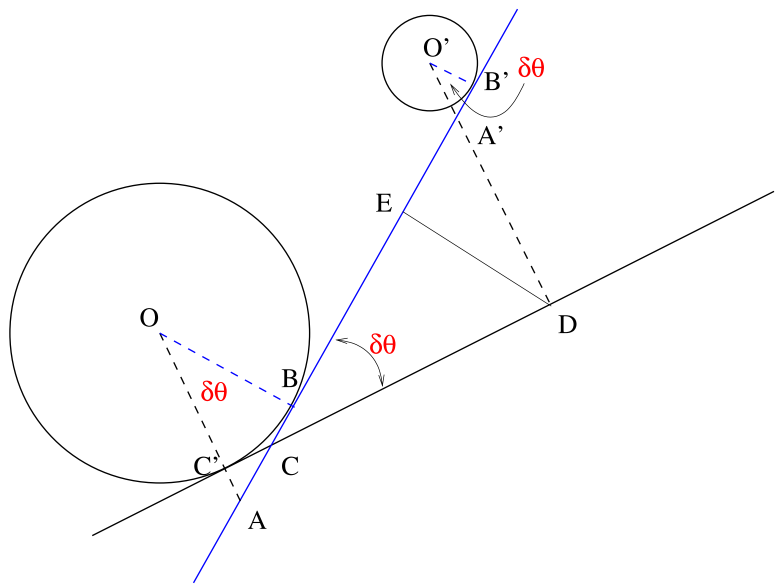

Figure 14: A diagram for emergence time estimate.

For consistency of notation, we will use D in place of $\mathbf{y}$. Let d be the distance between D and the circle centered at O'. Let ρ and ρ' be the radius of the circle centered at O and O' respectively. Let r denote the distance between D and C'. By elementary Euclidean geometry, we have the following identities:

$$CC' = \rho \tan \frac{\delta\theta}{2}, \;\; CD = r - CC' = r - \rho \tan \frac{\delta\theta}{2}, \;\; \Longrightarrow DE = CD \sin \delta\theta,$$

$$O'A' = \frac{1}{\cos \delta\theta} \rho'.$$

Therefore, we can find $\delta\theta$ from the last two equalities. Since we know how fast the horizon is moving, we can then determine δt.

Let $\dot{\mathbf{x}} = |\dot{\mathbf{x}}|_\infty$,

$$\frac{DE}{\delta t} = |\dot{\mathbf{x}}|_\infty, \;\; \delta t = \frac{DE}{|\dot{\mathbf{x}}|_\infty}.$$

Further considerations

We have mentioned in the beginning of this paper that the approach of moving the visible contour may not be more efficient than simply performing implicit ray tracing in general "large world" configurations. Here we construct such a case to

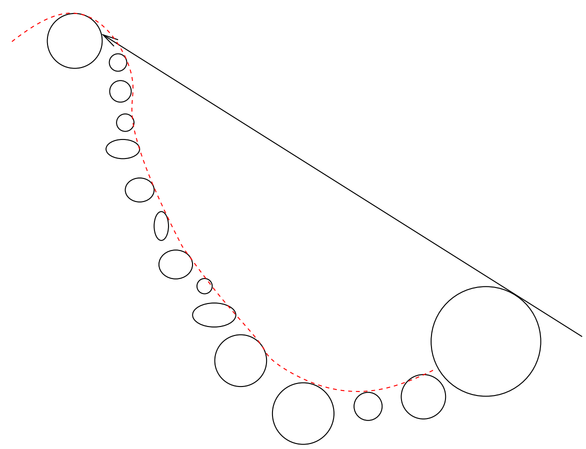

Figure 15: bad case for moving curves

validate our arguments. Consider a nonconvex part of an object as shown by the dashed red curve in Figure 15. Suppose the red curve is broken down into dense small disconnected components. In the case where the red curve is the nonconvex part of a connected component and with the viewing direction being depicted in Figure 15, the cast horizon will move continuously on the nonconvex part of the object without need for reinitialization. However, in the second case, we have to reinitialize very often because the cast horizon will "jump" from component to component.

4 Conclusion and future directions

In this article, we introduced a fast implicit ray tracing algorithm independent of grid geometry and easily parallelizable. This is then extended to a multi-resolution algorithm for near optimal efficiency. Furthermore, we showed that the implicit framework captures accurately the shadow boundaries, which include the horizon and cast horizon curves. We studied how these objects move when the source point is moving. Explicit formulas which reveal the relations between the motions and the local/global geometry of the given configuration are derived and are tightly coupled with our level set framework for implementation. Also, questions such as "how soon will this hidden object appear" can be answered as a result of our algorithm.

There is a rich pool of applications related to the visibility problem described in this paper. Currently we are working on problems related to navigation, visibility with occluders changing shapes in time, in non-uniform media. Our solutions

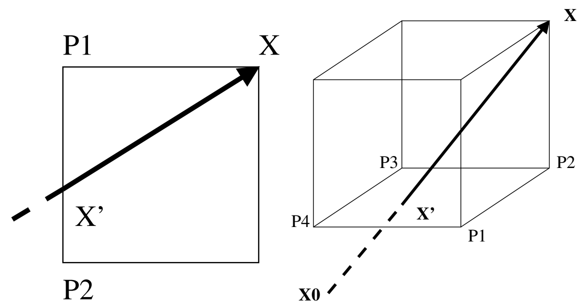

Figure 16: A demonstration of 2D and 3D interpolation

will combine approaches both from the PDE formulation and the algorithms in computational geometry.

Acknowledgement

The author YT wishes to thank the organizers of IPAM Geometrically Based Motion Workshop and IPAM staff.

5 Appendix

5.1 Interpolation schemes

Since the majority of visibility applications benefit from the simplicity of Cartesian grids, we need to adapt the algorithm in order to take advantage of this. As described in the algorithm, at each grid point $x = (i, j, k)$, we need to determine an upwind neighbor x' and find the value of $\psi(x')$. In most cases, x' does not lie on the grid. Therefore, we need to interpolate the values of ψ from the grid points closest to x'. For simplicity and speed considerations, we choose to perform linear interpolation in 2D and bilinear interpolation in 3D. In Figure 16, we use $\psi(P_1)$ and $\psi(P_2)$ for linear interpolation in the 2D case and use $\psi(P_i)$, $i = 1, 2, 3, 4$, for bilinear interpolation of ψ.

We note that a fast marching or fast sweeping strategy for determining distance from the source point and passing values can be used in place of this interpolation.

Let ψ_{int} be the interpolant near x', we know that $\psi_{\text{int}}(x') = \psi(x') + \mathcal{O}(h^2)$. Thus, the discrete visibility equation (2) is in effect

$$\psi(x) = \min(\psi_{\text{int}}(x'), \phi(x)).$$

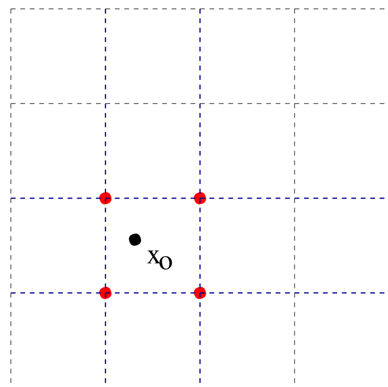

Figure 17: The red points denote the cell vertices.

5.2 Examples of star-shaped updating sequence (sweeping)

There are many different ways of implementing a star-shaped updating sequence. One approach is to use the algorithm based on the heap sort strategy [25] to find grid nodes for update based on their distance to the vantage point. However, due the complexity involved with heap sort, this algorithm is not optimal.

Alternatively, we use a sweeping approach in our simulation. For example, let us consider a Cartesian grid in 2D and assume that the vantage point lies on a grid node; we can then consider separately the visibility problem in each of the four quadrants centered at the vantage point. For simplicity, let us assume that the vantage point is at the origin and the grid is represented by the lattice $[-n_x, n_x] \times [-n_y, n_y] \subset \mathbb{Z}^2$. A compact way of writing this sweeping sequence in C/C++ is:

```
for(s1=-1;s1<=1;s1+=2)
for(s2=-1;s2<=1;s2+=2)
for(i=0;(s1<0?i>=-nx:i<=nx);i+=s1)
for(j=0;(s2<0?j>=-ny:j<=ny);j+=s2)
  update ψ_{i,j}.
```

In the case where $\mathbf{x_o}$ does not lie on a grid node, we describe an easy modification to the updating sequence above. Let $\mathbf{x_o} \in I_o := [x_{i_0}, x_{i_0+1}) \times [y_{j_0}, y_{j_0+1})$. Update the values of ψ on the vertices of I_o. Then update the grid nodes in the strips $\{(x_i, y_j) : i = i_0, i_0 + 1$ and $j = -n_y$ to $n_y\}$ and $\{(x_i, y_j) : i = -n_x, n_x$ and $j = j_0, j_{0+1}\}$. Finally, update the remaining four quadrants independently. See Figure 17 for a depiction of this approach.

5.3 Finding the curvature of a specified direction

As we argued in Section 3.2, the three dimensional problem of determining the motion of the horizon can be reduced to an instantaneous two dimensional problem. In order to move the horizon in this manner, we need to evaluate the curvature of the surface in the specified direction. Here we present a way to do that.

Let τ be the tangent vector being specified. We want to find the curvature on $\partial\Omega$ in this direction. First let $p(\mathbf{x};\tau)$ be the plane passing through $\mathbf{x}$, spanned by $\mathbf{n}(x)$ and τ, and let P be the level set function that embeds this plane. Then

$$\tilde{\tau} = \frac{\nabla\phi \times \nabla P}{|\nabla\phi \times \nabla P|},$$

where $\tilde{\tau}(\mathbf{x}) = \tau$, and the curvature is

$$k_\tau \mathbf{n} = \vec{\nabla}\tilde{\tau} \cdot \tilde{\tau}.$$

5.4 Derivation of the dynamics of horizon

We follow the constraints (6):

$$\begin{cases} \phi(\mathbf{x}) = 0 \\ (\mathbf{x} - \mathbf{x}_0) \cdot \nabla\phi(\mathbf{x}) = 0 \end{cases}$$

and differentiate with respect to t, we have

$$\nabla\phi(\mathbf{x}) \cdot \dot{\mathbf{x}} = 0 \tag{12}$$

$$(\mathbf{x} - \mathbf{x}_o) \cdot D^2\phi(\mathbf{x})\dot{\mathbf{x}} = \dot{\mathbf{x}}_o \cdot \nabla\phi(\mathbf{x}) \tag{13}$$

In 2 space dimensions, these two relations uniquely determine the motion of x with given initial conditions. Writing $(\mathbf{x} - \mathbf{x}_o) = |\mathbf{x} - \mathbf{x}_o|\, n^\perp(\mathbf{x}) = |\mathbf{x} - \mathbf{x}_o|\,(-\phi_y, \phi_x)/|\nabla\phi|$, we have

$$\begin{aligned} &\frac{|\mathbf{x} - \mathbf{x}_o|}{|\nabla\phi|} \begin{pmatrix} -\phi_y \\ \phi_x \end{pmatrix} \cdot \begin{pmatrix} \phi_{xx} & \phi_{xy} \\ \phi_{yx} & \phi_{yy} \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \\ &= \frac{|\mathbf{x} - \mathbf{x}_0|}{|\nabla\phi|} \begin{pmatrix} -\phi_y\phi_{xx} + \phi_x\phi_{yx} \\ -\phi_y\phi_{xy} + \phi_x\phi_{yy} \end{pmatrix} \cdot \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix}, \end{aligned}$$

and

$$\begin{aligned} &\frac{|\mathbf{x} - \mathbf{x}_o|}{|\nabla\phi|} \begin{pmatrix} \phi_x & \phi_y \\ -\phi_y\phi_{xx} + \phi_x\phi_{yx} & -\phi_y\phi_{xy} + \phi_x\phi_{yy} \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \\ &\qquad = \begin{pmatrix} 0 \\ \dot{\mathbf{x}}_o \cdot \nabla\phi(\mathbf{x}) \end{pmatrix} \end{aligned}$$

$$
\begin{aligned}
\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} &= \tfrac{|\nabla\phi|}{|\mathbf{x}-\mathbf{x}_o|}\tfrac{1}{D} \begin{pmatrix} -\phi_y\phi_{xy}+\phi_x\phi_{yy} & -\phi_y \\ \phi_y\phi_{xx}-\phi_x\phi_{yx} & \phi_x \end{pmatrix} \begin{pmatrix} 0 \\ \dot{\mathbf{x}}_o\cdot\nabla\phi(\mathbf{x}) \end{pmatrix} \\
&= \tfrac{|\nabla\phi|}{|\mathbf{x}-\mathbf{x}_o|}\tfrac{\dot{\mathbf{x}}_o\cdot\nabla\phi(\mathbf{x})}{D} \begin{pmatrix} -\phi_y \\ \phi_x \end{pmatrix},
\end{aligned}
$$

where

$$
\begin{aligned}
D &= \det \begin{pmatrix} \phi_x & \phi_y \\ -\phi_y\phi_{xx}+\phi_x\phi_{yx} & -\phi_y\phi_{xy}+\phi_x\phi_{yy} \end{pmatrix} \\
&= -\phi_x\phi_y\phi_{xy}+\phi_x^2\phi_{yy}+\phi_y^2\phi_{xx}-\phi_x\phi_y\phi_{xy} \\
&= \phi_x^2\phi_{yy}+\phi_y^2\phi_{xx}-2\phi_x\phi_y\phi_{xy}.
\end{aligned}
$$

Since the curvature of $\partial\Omega$ at x is

$$
\begin{aligned}
\kappa &= \nabla\cdot\frac{\nabla\phi}{|\nabla\phi|} \\
&= \frac{1}{|\nabla\phi|^3}(\phi_x^2\phi_{yy}+\phi_y^2\phi_{xx}-2\phi_x\phi_y\phi_{xy}),
\end{aligned}
$$

the motion of x is

$$
\dot{\mathbf{x}} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \frac{1}{\kappa}\frac{\dot{x}_o\cdot n(\mathbf{x})}{|\mathbf{x}-\mathbf{x}_o|}n^{\perp}(\mathbf{x}). \tag{14}
$$

We define $n^{\perp}(\mathbf{x}) = (\mathbf{x}-\mathbf{x}_o)/|\mathbf{x}-\mathbf{x}_o|$.

Alternatively, we can write $\dot{x}$ in a slightly different form:

$$
\dot{x} = \frac{P_{\nabla\phi}\nabla^2\phi(x-x_0)}{|P_{\nabla\phi}\nabla^2\phi(x-x_0)|^2}(\dot{x}_0\cdot\nabla\phi),
$$

where P_v is the orthogonal projection matrix projecting vectors to the plane with normal vector v. Let us check this expression for the velocity of the curve. Note $\nabla\phi\cdot\dot{x} = 0$ since $v\cdot P_v w = 0$ for all vectors v and w. Also, $\nabla^2\phi(x-x_0)\cdot\dot{x} = \nabla\phi\cdot\dot{x}_0$ is satisfied since $P_v w\cdot w = |P_v w|^2$ for all vectors v and w. Thus this velocity is valid and is the first form in our alternate derivation.

Geometric interpretation

If x indeed represents the position of the curve, then $x-x_0$ is tangent to the object surface at x and so $x-x_0 = P_{\nabla\phi}(x-x_0)$. Making this replacement above gives our second form for the velocity,

$$
\dot{x} = \frac{P_{\nabla\phi}\nabla^2\phi P_{\nabla\phi}(x-x_0)}{|P_{\nabla\phi}\nabla^2\phi P_{\nabla\phi}(x-x_0)|^2}(\dot{x}_0\cdot\nabla\phi).
$$

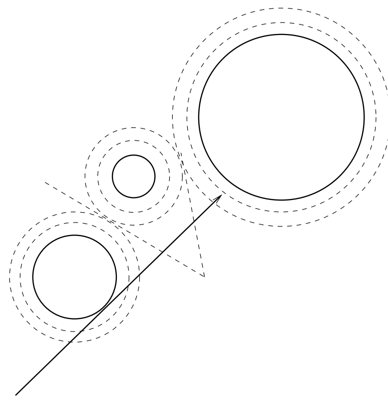

Figure 18: Model cast horizon scenario

This form is particularly nice because we know the second fundamental form in a level set framework is transformed to

$$II = \frac{1}{|\nabla\phi|} P_{\nabla\phi} \nabla^2 \phi P_{\nabla\phi}.$$

Thus we can rewrite the velocity in its final form,

$$\dot{x} = \frac{II(x - x_0)}{|II(x - x_0)|^2} \left(\dot{x}_0 \cdot \frac{\nabla\phi}{|\nabla\phi|} \right).$$

IIv evaluated at a point represents the change in the normals of the object surface in the direction of v at that point.

5.5 Derivation of the dynamics of the cast horizon

We assume that the level sets of ϕ near $\mathbf{x}$ are smooth curves and are not tangent to ν. In two space dimension, we have two equations that determine the dynamics of $\mathbf{x}$:

$$\begin{cases} \phi(\mathbf{x}) = \text{constant} \\ \nu = \tilde{\nu} \end{cases} \tag{15}$$

Let $\mathbf{r}$ and $\tilde{\mathbf{r}}$ denote $(\mathbf{x} - \mathbf{x_o})$ and $(\tilde{\mathbf{x}} - \mathbf{x_o})$ respectively. Differentiating these equations, we arrive at:

$$\nabla\phi(\mathbf{x}) \cdot \mathbf{x}' = 0,$$

$$\frac{\mathbf{r}'}{|\mathbf{r}|} - \frac{\mathbf{r}}{|\mathbf{r}|^2} \nu \cdot \mathbf{r}' = \frac{\tilde{\mathbf{r}}'}{|\tilde{\mathbf{r}}|} - \frac{\tilde{\mathbf{r}}}{|\tilde{\mathbf{r}}|^2} \tilde{\nu} \cdot \tilde{\mathbf{r}}'.$$

Notice that the term

$$\nu \cdot \mathbf{r}' \frac{\mathbf{r}}{|\mathbf{r}|} = (\nu \cdot \mathbf{r}')\nu$$

is $\mathbf{r}'$ projected onto the unit vector ν. Therefore, the left hand side denotes the projection of $\mathbf{r}'$ onto the unit vector $\nu^{\perp}$:

$$\frac{1}{|\mathbf{r}|}(\mathbf{r}' - \nu \cdot \mathbf{r}'\nu) = \frac{\mathbf{r}' \cdot \nu^{\perp}}{|\mathbf{r}|} = \frac{1}{|\mathbf{r}|} P_{\nu}\mathbf{r}'.$$

Similarly, with the right hand side, we have the equation:

$$\frac{1}{|\mathbf{r}|} P_{\nu}\mathbf{r}' = \frac{1}{|\tilde{\mathbf{r}}|} P_{\tilde{\nu}}\tilde{\mathbf{r}}'.$$

Keeping in mind that we want to solve for $\mathbf{x}'$, we move every other term to the right hand side and arrive at

$$P_{\nu}\mathbf{x}' = \frac{|\mathbf{r}|}{|\tilde{\mathbf{r}}|} P_{\tilde{\nu}}\tilde{\mathbf{r}}' + P_{\nu}\mathbf{x}'_{\mathbf{o}},$$
$$\nabla\phi(\mathbf{x}) \cdot \mathbf{x}' = 0.$$

In two dimensions, $P_{\nu}w = (w \cdot \nu^{\perp})\nu^{\perp}$, and $P_{\nu}w \cdot \nu^{\perp} = w \cdot \nu^{\perp}$, therefore, we have

$$\begin{pmatrix} \nu_2 & -\nu_1 \\ \phi_{x_1} & \phi_{x_2} \end{pmatrix} \begin{pmatrix} x'_1 \\ x'_2 \end{pmatrix} = \begin{pmatrix} \frac{|\mathbf{r}|}{|\tilde{\mathbf{r}}|}\tilde{\mathbf{r}}' \cdot v^{\perp} + \mathbf{x}'_{\mathbf{o}} \cdot \nu^{\perp} \\ 0 \end{pmatrix},$$

and consequently,

$$\begin{aligned}
\begin{pmatrix} x'_1 \\ x'_2 \end{pmatrix} &= \frac{1}{\nu \cdot \nabla\phi(\mathbf{x})} \cdot \\
&\quad \begin{pmatrix} \phi_{x_2} & \nu_1 \\ -\phi_{x_1} & \nu_2 \end{pmatrix} \begin{pmatrix} \frac{|\mathbf{r}|}{|\tilde{\mathbf{r}}|}\tilde{\mathbf{r}}' \cdot \nu^{\perp} + \mathbf{x}'_{\mathbf{o}} \cdot \nu^{\perp} \\ 0 \end{pmatrix} \\
&= \frac{1}{\nu \cdot \nabla\phi(\mathbf{x})} \\
&\quad (\frac{|\mathbf{r}|}{|\tilde{\mathbf{r}}|}\tilde{\mathbf{r}}' \cdot \nu^{\perp} + \mathbf{x}'_{\mathbf{o}} \cdot \nu^{\perp}) \begin{pmatrix} \phi_{x_2} \\ -\phi_{x_1} \end{pmatrix} \\
&= \frac{\mathbf{n}^{\perp}(\mathbf{x})}{\nu \cdot \mathbf{n}(\mathbf{x})} (\frac{|\mathbf{r}|}{|\tilde{\mathbf{r}}|}\tilde{\mathbf{r}}' + \mathbf{x}'_{\mathbf{o}}) \cdot \nu^{\perp} \\
&= \frac{(\frac{|\mathbf{r}|}{|\tilde{\mathbf{r}}|}\tilde{\mathbf{r}}' + \mathbf{x}'_{\mathbf{o}}) \cdot \nu^{\perp}}{\nu \cdot \mathbf{n}(\mathbf{x})} \mathbf{n}^{\perp}(\mathbf{x})
\end{aligned}$$

where

$$\nabla^{\perp}\phi(\mathbf{x}) := \begin{pmatrix} \phi_{x_2} \\ -\phi_{x_1} \end{pmatrix}, \text{ and } n^{\perp} := \frac{\nabla^{\perp}\phi(\mathbf{x})}{|\nabla^{\perp}\phi(\mathbf{x})|}.$$

5.6 An algorithm to project $\{\psi = 0\}$ to S^{n-1}

We discretize S^{n-1} over a Cartesian grid: $\rho : [0, 2\pi) \times [-\pi, \pi) \mapsto x_o + S^{n-1}/r$, with r large enough so that $x_o + S^{n-1}/r$ lies completely outside of ψ. $\Delta\theta_1$, $\Delta\theta_2$ are set to be the smallest angle possible on the grid that discretizes ψ. Therefore, we can set $\Delta\theta_1 = \tan^{-1}(\min(\Delta x, \Delta y)/L)$, where L is the diagonal length of the grid.

For each $\rho_{l,m}$, we then shoot a ray, starting from $p_0 = \rho_{l.m}$ outward according to the angle determined by (l, m), and use the bisection method to find p_1, the intersection of the ray and $\{\psi = 0\}$.

Note that the complexity is $N^2 \log N$ for a grid of size N^3.

5.7 Numerics

We computed the quantities describe in this paper using standard level set technologies. Please refer to [19, 20, 21, 22, 10] for details.

5.8 A list of level set functions used in this paper

We provide a comprehensive list of the level set functions we construct in this paper:

ϕ embeds the objects

$\hbar(\mathbf{x}) := (\mathbf{x} - \mathbf{x_o}) \cdot \nabla\phi(\mathbf{x})$ characterizes the horizon

$\tilde{\phi} := \max(\phi, -\hbar)$ $\{\tilde{\phi} \leq 0\} = \{\phi \leq 0\} \setminus \{\hbar < 0\}$, defines the same visibility as ϕ

ψ the visibility map resulting from the implicit ray tracing on ϕ

$\tilde{\psi}$ the visibility map resulting from the implicit ray tracing on $\tilde{\phi}$, characterizes the cast horizon

ϑ: $\mathbb{R}^d \mapsto \mathbb{R}^d$ links horizon to its cast implicitly

References

[1] D. Adalsteinsson and J.A. Sethian. An overview of level set methods for etching, deposition, and lithography development. *IEEE Transactions on Semiconductor Devices*, 10(1), February 1997.

[2] Pankaj K. Agarwal and Micha Sharir. Ray shooting amidst convex polygons in 2D. *J. Algorithms*, 21(3):508–519, 1996.

[3] Pankaj K. Agarwal and Micha Sharir. Ray shooting amidst convex polyhedra and polyhedral terrains in three dimensions. *SIAM J. Comput.*, 25(1):100–116, 1996.

[4] Luis Alvarez, Frédéric Guichard, Pierre-Louis Lions, and Jean-Michel Morel. Axioms and fundamental equations of image processing. *Arch. Rational Mech. Anal.*, 123(3):199–257, 1993.

[5] M. Bertalmio, G. Sapiro, and G. Randall. Dynamic visibility in an implicit framework. *IEEE Trans. Medical Imaging*, 18:448–451, 1999.

[6] Marcelo Bertalmio, Li-Tien Cheng, Stanley Osher, and Guillermo Sapiro. Variational problems and partial differential equations on implicit surfaces. *J. Comput. Phys.*, 174(2):759–780, 2001.

[7] M. D. Betterton. Theory of structure formation in snowfields motivated by penitentes, suncups, and dirt cones. *Physical Review E*, 63, 2001.

[8] Paul Burchard, Li-Tien Cheng, Barry Merriman, and Stanley Osher. Motion of curves in three spatial dimensions using a level set approach. *J. Comput. Phys.*, 170:720–741, 2001.

[9] J. C. Carr, R. K. Beatson, J.B. Cherrie T. J. Mitchell, B. C. McCallum W. R. Fright, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH*, 2001.

[10] Li-Tien Cheng, Paul Burchard, Barry Merriman, and Stanley Osher. Motion of curves constrained on surfaces using a level set approach. *J. Comput. Phys.*, 175:604–644, 2002.

[11] Roberto Cipolla and Peter Giblin. *Visual motion of curves and surfaces*. Cambridge University Press, 2000.

[12] Satyan Coorg and Seth Teller. Real-time occlusion culling for models with large occluders. In *ACM Symposium on Interactive 3D Graphics*, 1997.

[13] Fredo Durand. 3d visibility: Analysis study and applications. *PhD Thesis, MIT*, 1999.

[14] Fredo Durand, George Drettakis, Joëlle Thollot, and Claude Puech. Conservative visibility preprocessing using extended projections. In *SIGGRAPH*, 2000.

[15] Memoli Facundo and Guillermo Sapiro. Fast computation of distance functions and geodesics on implicit surfaces. To appear, Journal of Computational Physics (2001).

[16] Aaron Hertzmann and Denis Zorin. Illustrating smooth surfaces. In *SIGGRAPH*, 2001.

[17] Hailin Jin, Anthony Yezzi, and Stefano Soatto. Stereoscopic shading: Integating multi-frame shape cues in a variational framework. *In Preparation*.

[18] Stanley Osher, Li-Tien Cheng, Myungjoo Kang, Hyeseon Shim, and Yen-Hsi Tsai. Geometric optics in a phase space based level set and eulerian framework. 2001. Submitted to Journal of Computational Physics.

[19] Stanley Osher and Ronald P. Fedkiw. Level set methods: an overview and some recent results. *J. Comput. Phys.*, 169(2):463–502, 2001.

[20] Stanley Osher and James A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.

[21] Stanley Osher and Chi-Wang Shu. High-order essentially nonoscillatory schemes for Hamilton-Jacobi equations. *SIAM J. Numer. Anal.*, 28(4):907–922, 1991.

[22] Danping Peng, Barry Merriman, Stanley Osher, Hongkai Zhao, and Myungjoo Kang. A PDE-based fast local level set method. *J. Comput. Phys.*, 155(2):410–438, 1999.

[23] G. Sapiro, R. Kimmel, D. Shaked, B. B. Kimia, and A. M. Bruckstein. Implementing continuous scale morphology via curve evolution. *Pattern Recognition*, 26(9):1363–1372, 1993.

[24] J. A. Sethian. *Level set methods and fast marching methods*. Cambridge University Press, Cambridge, second edition, 1999. Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science.

[25] John Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.

[26] Hong-Kai Zhao, Stanley Osher, Barry Merriman, and Myungjoo Kang. Implicit and non-parametric shape reconstruction from unorganized points using variational level set method. *Computer Vision and Image Understanding*, 80:295–319, 2000.