The Image Generalization Visibility Paradigm

withheld for double-blind review

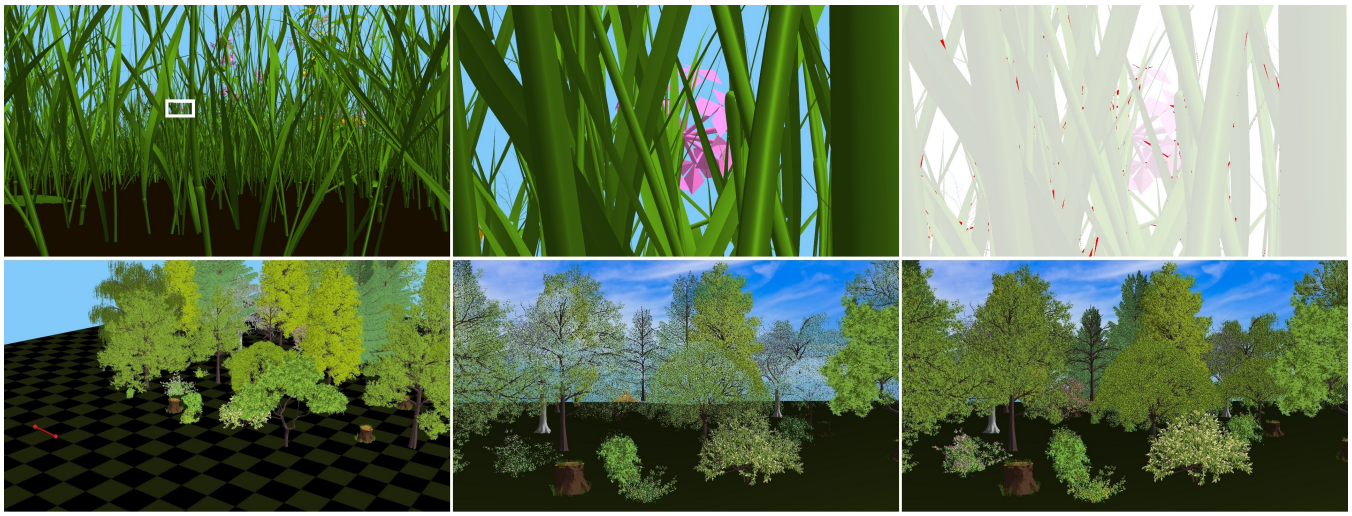

Figure 1: ***Top****: aggressive from-point visibility with quality guarantee: reference image where visible set is computed (left), frame with 17x zoom factor and 0.26% error rendered from visible set (middle), and frame error visualization (right).* ***Bottom****: from-segment visibility: view segment (left), incorrect frame rendered from the triangles visible from the view segment endpoints (middle), and correct frame rendered from the visible set computed by our algorithm directly over the entire view segment (right).*

Abstract

We extend the visibility computation capability of conventional images in three ways: (1) by adding sampling locations to sample all triangles fragments, which guarantees that all triangles with a completely visible fragment are found, no matter how small their image footprint; (2) by adding sampling locations based on a visibility subdivision of the image, which completes the set of visible triangles efficiently; (3) and by generalizing the visibility sample at a sampling location from 0D to 1D and to 2D, which supports viewpoint translations and time changes. We have used the paradigm to develop a quality-guaranteed aggressive from-point visibility algorithm, an efficient exact from-point visibility algorithm, a from-segment and a from-rectangle visibility algorithm that are exact under view translation, and an over-time-interval visibility algorithm that is exact for a constant view. We have tested our visibility algorithms on a variety of complex static and dynamic scenes. Our paradigm compares favorably to the prior art approaches of sampling the high dimensional space of visibility parameters with conventional images or with individual rays that are defined by sampling locations placed heuristically.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Visible line/surface algorithms.

Keywords: sample-based visibility, continuous visibility, aggressive visibility, exact visibility, visibility in dynamic scenes.

1 Introduction

For complex scenes, the set of triangles visible from a view region is a small fraction of the total set, and therefore visibility is a powerful tool for reducing scene complexity. Visibility benefits virtually all computer graphics applications, including interactive applications where frame rate is at a premium, applications that require expensive rendering effects such as reflections, soft shadows, or motion blur, and applications where clients running on "thin" platforms have to render complex scenes hosted on remote servers.

Visibility remains an open research problem despite decades of research. One approach evaluates visibility at a set of image plane sampling locations; the visible set is computed *aggressively*, which means that the set contains *only, but not all* visible triangles. The advantage is efficiency. However, prior sample-based visibility algorithms define sampling locations heuristically, and they do not provide any quality guarantee for the visible set they compute. A second approach performs a continuous visibility analysis of the image, subdividing it into regions where a single triangle is visible. The visibility subdivision provides the exact set that contains *only and all* visible triangles. However, prior continuous visibility algorithms are inefficient as they construct the visibility subdivision from all triangles, with the hidden triangles being unnecessarily added temporarily to the visibility subdivision.

We present a novel visibility paradigm based on image generalization, which combines the advantages and avoids the disadvantages of the two approaches described above. The paradigm is based on the observation that a conventional image is a powerful visibility computation tool: an image is computed efficiently using graphics hardware, and every image pixel finds a visible triangle. However, the visible set found by a conventional image is far from complete. First, triangles can have an arbitrarily small image footprint due to a high scene complexity, to a large distance to the eye, or to a grazing viewing angle. In Figure 2 (left), the visible set found by

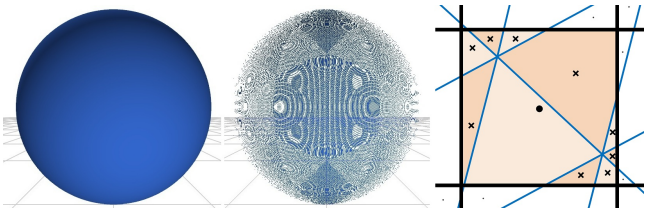

Figure 2: ***Left****: reference image and image rendered from incomplete set of visible triangles found by reference image.* ***Right****: sampling locations added (crosses) to sample all fragments.*

the reference image of the finely tessellated sphere is incomplete, which results in severe artifacts when the set is used to render an image from the same viewpoint but with a slightly different view direction. Increasing the resolution of the reference image is only palliative: the image footprint of a visible triangle can be arbitrarily small, therefore an infinite resolution would be needed to guarantee that all visible triangles are found. Second, an image can rule a triangle as a visible, but it cannot rule a triangle as hidden. Whereas a single sample can be sufficient to verify that a triangle is visible, an infinite number of samples would be needed to verify that a triangle is hidden at all its points. Third, a conventional image finds visible triangles from a viewpoint and not from a view region, missing triangles that become visible as the viewpoint translates. Moreover, a conventional image only computes visibility for a single time point, missing triangles that becomes visible as time changes. We generalize images to remove these shortcomings. Our image generalization paradigm has three elements.

1. Finding all triangles with a completely visible fragment. A fragment is defined as the convex polygonal intersection between a triangle projection and a pixel. Sampling locations are added to make sure that all fragments are sampled, which guarantees finding all triangles with a completely visible fragment, and which in turn guarantees finding all triangles of a front surface, no matter how small their footprint. For our sphere example, sampling all fragments finds all visible triangles at the cost of exactly one sampling location per fragment (Figure 2, right). Sampling locations are not added heuristically but rather deterministically, based on the scene geometry and on the pixel grid, to compute efficiently a quality-guaranteed aggressive visible set.

2. Finding all visible triangles. Sampling all fragments does not guarantee finding all visible triangles since a partially visible fragment might be sampled at a point where the fragment is hidden. Starting from the aggressive visible set, sampling locations are added iteratively to determine the visibility of triangles that are not hidden by the known visible triangles. The algorithm computes and updates a visibility subdivision of the image based on the current set of visible triangles. No hidden triangle is ever added to the visibility subdivision. Because the starting set is almost complete, the visible set converges quickly to the exact set.

3. Finding all triangles visible at a sampling location. When the viewpoint translates or when time changes in a dynamic scene, multiple triangles can be visible at a given image plane sampling location. We generalize the visibility sample captured by the sampling location from 0D to 1D, for computing visibility directly from a view segment or over a time interval, and to 2D, for computing visibility directly from a view rectangle. The sampling location does *not* sample the space of visibility parameters exhaustively. The visibility changes are computed directly by solving visibility event equations, which guarantees the quality of the visibility solution, and which provides computation and storage efficiency.

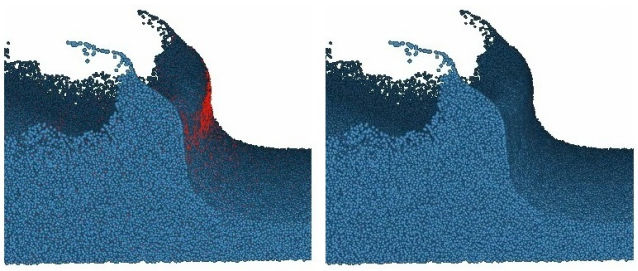

Figure 3: *Over-time-interval visibility in a dynamic scene: frame rendered from the particles visible at the time interval endpoints, with missing particles shown in red (left), and correct frame rendered from the visible particles computed by our algorithm directly over the entire time interval (right).*

We have used the image generalization paradigm to develop four visibility algorithms. The first one is a quality-guaranteed aggressive from-point visibility algorithm. In Figure 1, the algorithm computes the correct visibility over 99.93% of the reference image. The aggressive set yields high quality frames even under substantial magnification of the reference image. The few incorrect pixels are at surface boundaries, and never inside visible surfaces, so the frames are virtually indistinguishable from truth. The second algorithm computes exact from-point visibility. The algorithm extends the aggressive set from Figure 1 to the exact set in three iterations.

The third algorithm is an aggressive from-segment visibility algorithm that is exact under view translation. The algorithm computes the visible set directly over the entire view segment by computing visibility exactly for each sampling location. A sampling location stores a 1D visibility sample that records all triangles visible as the viewpoint translates. When the view translates (and does not rotate or change focal length), the sampling locations do not change, and the visible set is exact. In Figure 1, bottom, rendering an intermediate frame from the triangles visible at the endpoints of the view segment results in severe visibility artifacts, and the correct frame is obtained when using our algorithm. The same algorithm computes visibility over a time interval in a dynamic scene (Figure 3). The fourth algorithm computes visibility directly over a view rectangle, and the algorithm is exact under view translation (Figure 4).

Visibility is computed for a scene by subdividing the viewing volume into box-like cells, and the viewing time into intervals. A conventional image does not compute visibility adequately even for a single viewpoint and a single time point. Approximating from-cell and over-time visibility requires a large number of conventional images to sample the six-dimensional space of visibility parameters (two view direction rotations, three viewpoint translations, and time). Our paradigm increases the visibility computation power of images. Image generalization for exact from-point visibility alone reduces the dimensionality of the parameter space to four by eliminating the two view rotations; over-time-interval visibility eliminates the time parameter; from-rectangle eliminates two translations. From-cell visibility is approximated by aggregating the visible sets for the eight corners, the twelve edges, or the six faces of the cell. The more powerful the underlying image generalization, the more complete the resulting visible set, and the coarser the required subdivision of the viewing volume and time. We also refer the reader to the accompanying video.

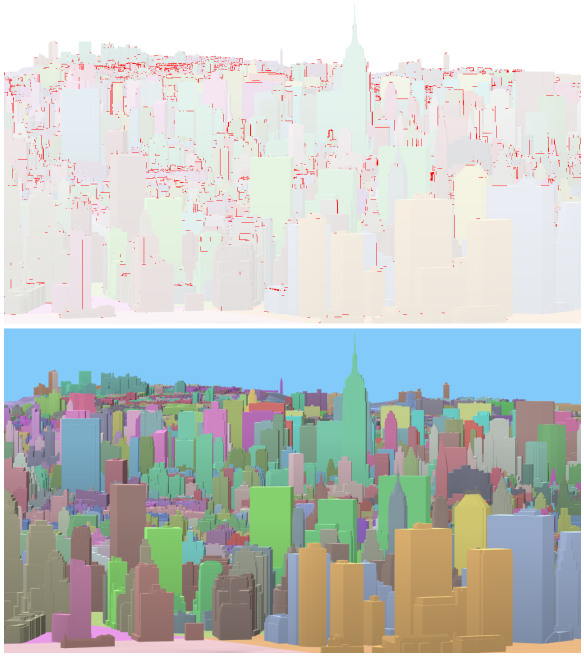

Figure 4: *From-rectangle visibility: error visualization for a frame rendered from the triangles visible from the view rectangle corners (top), and correct frame rendered from the visible set computed by our algorithm directly over the entire view rectangle (bottom).*

2 Prior Work

Visibility algorithms are classified based on the visible set they compute. *Conservative visibility algorithms* overestimate visibility, such that no visible triangle is omitted. The benefit is an accurate image, but the number of hidden triangles unnecessarily included in the solution can be substantial [Cohen-Or et al. 2003; Durand 2010]. *Aggressive visibility algorithms* underestimate the set of visible triangles, which leads to image errors. The goal of aggressive visibility research is to reduce and control the error [Nirenstein and Blake 2004; Wonka et al. 2006]. *Exact visibility algorithms* find only and all visible triangles, which avoids the cost of rendering unnecessary triangles as well as any image error.

Aggressive Visibility. We distinguish between probing visibility by casting individual rays, and by rendering entire images. Algorithms in the first category use heuristics to shoot rays that are likely to find visible triangles, and subsequent sampling is guided by what the initial rays find e.g. [Wonka et al. 2006; Bittner et al. 2009]. The advantage is the flexibility to cast precisely the rays deemed necessary, which limits sampling redundancy. However, it is difficult to place error bounds on the results. Moreover, casting individual rays can only be done efficiently with a hierarchical scene subdivision, which is difficult to extend to dynamic scenes.

Algorithms in the second category leverage the fact that the amortized cost of rays in an image is lower than that of individual rays. Our aggressive visibility algorithms fall in this category. An image only captures samples visible from its viewpoint. One option is to use images from additional viewpoints [McMillan and Bishop 1995], which are highly redundant, or to eliminate redundancy as a pre-process [Max and Ohsaki 1995; Shade et al. 1998]. The challenge of these approaches is to decide which images are needed for a sufficient sampling of the visibility parameter space. The usual strategy is to sample uniformly "as densely as possible", and thus the visibility error is not bounded. Multiperspective images capture in a single shot more than what is visible from a single viewpoint through innovation at the camera model level [Yu and McMillan 2004; Cui et al. 2010], but there is no quality guarantee on the visible set they gather.

Specialized visibility algorithms have been developed for many computer graphics contexts. The algorithms are typically aggressive, focusing on finding the visible triangles of highest relevance in the particular context. *The semi-analytical visibility* algorithm [Gribel et al. 2011], developed for motion blur, samples the image with lines as opposed to points, an idea from temporal antialiasing [Jones and Perry 2000]. Visibility is analyzed continuously over time for each line sample. The algorithm is aggressive because the analysis is restricted to lines in the image. Line samples are a brute force approach for improving uniform point sampling. The line parameter adds an expensive second dimension to the 1D motion blur visibility problem. The line sample pattern is fixed, so even after solving the higher-dimension visibility problem, there is still no guarantee for the quality of the solution. We propose deterministic point sampling that guarantees a quality visible set without increasing the visibility problem dimensionality.

Exact Visibility. Early work focuses on from-point visibility for antialiasing. The solution was to compute a visibility partition for each pixel, defined by the triangle fragments visible at each pixel [Carpenter 1984; Catmull 1978; Weiler and Atherton 1977]. The solution is inefficient because fragments of hidden triangles are added and then removed from the visibility partitions. Pixel-free from-point visibility algorithms are also inefficient because occluded intersections are computed (e.g. [Goodrich 1992]); typical running times are $O((n+k)\log n)$ or $O(n\log n+k+t)$ for n triangles with k edge intersections and t triangle intersections on the image plane. Output sensitive algorithms are restricted to special input [Katz et al. 1992; Sharir and Overmars 1992]. From-point visibility was implemented on the GPU [Auzinger et al. 2013], but the quadratic running time limits applicability to simple scenes.

Beam tracing [Heckbert and Hanrahan 1984] analyzes visibility in 3D by partitioning the 2D space of rays defined by the viewpoint using conical or frustum-like beams. The unsampled gaps between rays are avoided, but beam-triangle intersection is costly. Beam-tracing has recently been used for shadow [Apostu et al. 2012; Chandak et al. 2008] and sound [Overbeck et al. 2007] rendering, using acceleration schemes based on adaptive beam splitting. Our paradigm bypasses the need for beams in from-point visibility: the beam is replaced with the smallest number of rays needed to capture the visible triangles over a solid angle. Moreover, we do not trace multi-dimensional rays but rather evaluate visibility over their sampling domain by projection.

One strategy for from-rectangle visibility is to decompose the 4D non-Euclidean space of lines in the dataset according to visibility criteria. A complete model with $O(n^4)$ complexity and output sensitive construction is yet to be practical [Durand et al. 2002]. Another strategy is to compute visibility between pairs of polygons [Haumont et al. 2005; Mora and Aveneau 2005; Charneau et al. 2007], using constructive solid geometry in the 5D ambient space of the Plucker coordinate representation of lines. The algorithms have high computational complexity and are not output sensitive.

Conservative Visibility algorithms are exact algorithms that run on a visibility problem that was conservatively simplified, e.g. through extended projections [Durand et al. 2000], or occluder erosion [Décoret et al. 2003]. Our aggressive visibility algorithms produce a visible set that is almost complete, so adding to the set all triangles not hidden by the aggressive set yields a good conservative visible set (i.e. with only a small number of hidden triangles). Per-frame occlusion culling improves rendering performance by batch

discarding triangles that are hidden in the current output frame [Bittner et al. 2004]. Triangles are grouped inside containers with simple geometry, the containers are rendered on a partial z-buffer of the output frame obtained from known big blockers, and the triangles of hidden containers are discarded. Occlusion culling methods can also be aggresive by fusing blockers [Zhang et al. 1997]. Grouping occluded and occluding geometry is heuristic, and it is particularly challenging in the case of dynamic scenes.

Irregular Framebuffers. Our paradigm advocates abandoning the uniform sampling of conventional images in favor of adding sampling locations deterministically to guarantee that all visible triangles are found. The benefits of irregular framebuffers have been noted before in contexts that include: pixel-accurate shadow mapping [Johnson et al. 2005], where the shadow map estimates light visibility precisely at the point samples captured by the output image; point-based rendering [Popescu et al. 2000], where projected reference image samples are not clamped to the output image pixel grid but rather located precisely within the output image pixel using a pair of offsets; and focus plus context visualization where focus regions are sampled at a higher rate [Furnas 1986].

3 From-Point Visibility

We have developed two from-point visibility algorithms based on our image generalization visibility paradigm: one aggressive, with a quality guarantee, and one exact.

3.1 Aggressive

The input to the algorithm is a viewpoint o, a conventional pixel grid that defines the reference image where visibility is computed, and a set of triangles. The output is a set of triangles visible from o that is guaranteed to contain all triangles with a fragment that is completely visible from o. This includes all triangles of a front surface, no matter their footprint. The algorithm has three steps.

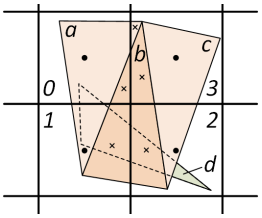

Figure 5: *Quality-guaranteed aggressive from-point visibility.*

Step 1 computes the triangles that are visible at the pixel centers using a conventional rendering pass. In Figure 5, the centers of pixels 0-3 are indicated with dots. The conventional image rendered finds triangles a and c, but not b, even though b is completely visible.

Step 2 takes a second pass over the scene triangles adding sampling locations to sample all fragments. A fragment f that does not already contain a sampling location generates a sampling location at its centroid c_f, unless the triangle of f is hidden at c_f by the triangle found at step 1. In Figure 5, a generates one and b four sampling locations (crosses); d generates no sampling locations because its fragments in pixels 1 and 2 contain sampling locations that were added for b, and because d is hidden by a at the centroid of its fragment in pixel 0.

Step 3 takes a third pass over the scene to find the triangles visible at each sampling location. Triangle projection proceeds as usual. Rasterization has to take into account that a pixel can have more than one sampling location. This implies that all pixels touched by the projection of a triangle have to be considered, and not just those whose centers are inside the projection. Furthermore, rasterization parameters (e.g. z, edge equation sidedness expressions) have to be evaluated for each sampling location of each pixel.

A sampling location is useful only if it reveals that its triangle is visible at that point. If the triangle is not visible at the sampling location, one cannot rule the triangle as hidden, and the sampling location is wasted. Step 1 computes a preliminary set of visible triangles efficiently, then step 2 avoids creating sampling locations that are already known not to be useful based on the preliminary set. If a pixel is completely covered by a visible triangle, the pixel will have only its initial sampling location at its center; the visible triangle is found at step 1, and all candidate sampling locations are discarded by the triangle at step 2. In our experiments, the number of sampling locations was linear in the number of visible fragments.

3.2 Exact

Algorithm 1: Exact from-point visibility

input : scene S, viewpoint o, aggressive visible set V_0;
output: exact set of visible triangles V;
1: **Construct** initial visibility subdivision VS from V_0;
2: **Initialize** $V = V_0$, $U = S - V_0$;
3: **while** U is not empty **do**
4: **Remove** all sampling locations;
5: **for all** triangles t in U **do**
6: **if** t is hidden by VS **then**
7: remove t from U;
8: **else**
9: add sampling locations for t
10: **end if**
11: **end for**
12: **Render** all triangles remaining in U;
13: **Collect** newly found visible triangles V_i;
14: **Update**VS with V_i; $U = U - V_i$; $V = V + V_i$;
15: **end while**

We have developed an efficient algorithm that extends the aggressive visible set iteratively to the exact set (Algorithm 1). The algorithm first constructs an initial visibility subdivision VS of the image from the aggressive visible set V_0 received as input (1). VS has polygonal regions where one or no triangles are visible. The region boundaries are the visible portions of the projected triangle edges. Figure 6 illustrates the construction of a visibility subdivision. The visible set V is initialized to V_0 and all other scene triangles are added to the set of undecided triangles U (2). Then the algorithm iterates until there are no more undecided triangles (3-15). For each iteration, the algorithm first removes the sampling locations generated by the previous iteration (4), and then it processes the undecided triangles (5-11).

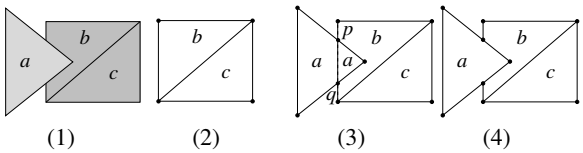

Figure 6: *Triangle a partially occluding b (1), and visibility subdivision before (2), during (3) and after (4) adding a.*

If an undecided triangle t is hidden by the visibility subdivision, t is removed from further consideration. If t is not hidden in a visibility subdivision region r, the algorithm creates sampling locations at

the vertices of t that project in r, at the vertices of r inside the projection of t, and at the intersection points of the edges of r and the projected edges of t. In Figure 7.1, t is hidden in regions r_3 and r_4, but not in r_1 and r_2, and six sampling locations are created.

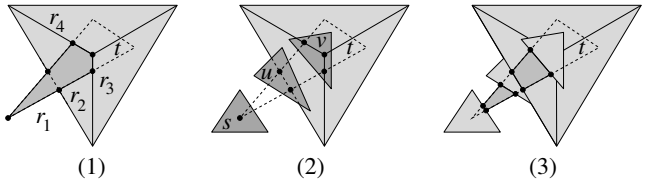

Figure 7: *(1) six sampling locations, shown with dots, created for undecided triangle t at first iteration (1), triangles s, u, and v visible at those sampling locations (2), and eight sampling locations created for t at second iteration (3).*

After all undecided triangles are processed, the remaining undecided triangles are rendered over the new sampling locations they have generated (12), a step identical to step 3 of the aggressive algorithm. The visible triangles found by the sampling locations are used to update the visibility subdivision, they are removed from the undecided set, and they are added to the visible set (14).

The algorithm is fast because the initial visible set is almost complete and most remaining triangles are hidden by the initial visibility subdivision. The visibility subdivision is built exclusively from visible triangles. Figure 7 illustrates an unlikely scenario where a triangle remains undecided after an iteration: all sampling locations created for t are won by other undecided triangles and more sampling locations are needed to decided t. In our experiments, the algorithm converged in three iterations or less, for scenes with tens of millions of triangles and with complex occlusion patterns.

4 From-Segment Visibility

We have developed an algorithm that computes visibility from a view segment directly. The input to the algorithm is a scene modeled with triangles, a view, and a view segment. The output is an aggressive approximation of the set of triangles visible from points on the view segment, with the following quality guarantees:

1. The visible set contains all triangles t that have a fragment f at a pixel p that is completely visible from any viewpoint on the segment from where f exists (i.e. the projection of t touches p). Like before, this guarantees that all triangles of a front surface are captured, no matter how small their footprint.

2. The visible set contains all triangles visible at the pixel centers as the view translates from one segment endpoint to the other. This makes the algorithm exact under view translation. As the view translates along the view segment, the visible set produces correct frames from any intermediate viewpoint.

The algorithm has two stages. In the first stage, sampling locations are added to sample all triangle fragments. This is done like before by computing the image footprint of the triangle, by computing its fragments, and by adding a sampling location at the centroid of a fragment that does not already contain one. The only difference is that now the footprint of the triangle is the union of all triangle projections as the view translates. We use a fast, tight, and conservative approximation of the triangle footprint as the convex hull of the six extremal vertex projections. In Figure 8 the projection of the triangle moves from $a_0b_0c_0$ to $a_1b_1c_1$, and the convex hull $a_0b_0c_0a_1b_1c_1$ has six fragments, each with one sampling location.

The second stage uses the third element of our visibility paradigm

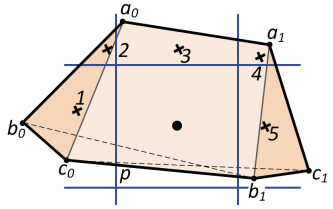

Figure 8: *Triangle footprint as the view translates.*

to find all triangles that are visible at each sampling location, as the view translates. The translation along the view segment defines a one-parameter visibility change. The visibility sample of a sampling location is generalized from 0D to 1D to store a list of visibility intervals that define which triangle is visible at the sampling location and for which view sub-segment. The algorithm for the second stage takes a pass over the scene triangles. For each triangle t, the footprint of t is approximated as described at stage 1, and then the visibility sample is updated for each sampling location inside the triangle footprint.

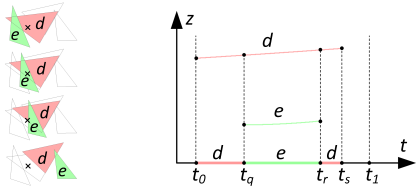

Figure 9: ***Left****: triangles d and e move over a sampling location (cross).* ***Right****: visibility intervals at the sampling location.*

In Figure 9, left, two triangle projections e and d move over a sampling location as the viewpoint translates along the view segment. The translation is linear with parameter t. The graph (right) shows the depths z from the viewpoint to the triangles, at the sampling location, as the viewpoint translates. The visibility intervals stored are shown on the abscissa. Initially the visibility sample of the sampling location is empty. After d is processed, there is one visibility interval $[t_0, t_s]$ for d. After e is processed, there are three visibility intervals: $[t_0, t_q]$ for d, $[t_q, t_r]$ for e, and $[t_r, t_s]$ for d.

Visibility changes at a sampling location when a moving projected triangle edge crosses the sampling location. Given two triangle vertices a and b and a sampling location s with image plane coordinates (u_s, v_s), we derive the condition that the projection of triangle edge ab contains s. The coordinates (u_a, v_a) of the projection of a are computed by writing a as a 3D point on the ray through (u_a, v_a) at w from viewpoint v_0:

$$a = v_0 + M \begin{bmatrix} u_a \\ v_b \\ 1 \end{bmatrix} w, \quad \begin{bmatrix} u_a \\ v_b \\ 1 \end{bmatrix} w = M^{-1}(a - v_0) \tag{1}$$

$$u_a = \frac{m_0 \cdot (a - v_0)}{m_2 \cdot (a - v_0)}, v_a = \frac{m_1 \cdot (a - v_0)}{m_2 \cdot (a - v_0)} \tag{2}$$

where M is the camera matrix, and the rows of M^{-1} are m_i. s is on the projection of ab if:

$$\frac{u_s - u_a}{u_s - u_b} = \frac{v_s - v_a}{v_s - v_b} \tag{3}$$

By plugging in u_a and v_a from Eq. 1 into Eq. 4, we obtain:

$$\frac{(u_s m_2 - m_0)(a - v_0)}{(u_s m_2 - m_0)(b - v_0)} = \frac{(v_s m_2 - m_1)(a - v_0)}{(v_s m_2 - m_1)(b - v_0)} \tag{4}$$

We target all possible view directions by running all visibility algorithms for each of the six faces of a cube map aligned with the world coordinate system. Consequently M is constant. Since m_i, v_s, u_s, a and b are constant, and since v_0 is linear in t, Eq. 4 is quadratic in t. Eq. 4 is solved for each of the three edges of the triangle, the solutions in $[t_0, t_1]$ are kept and sorted to define the visibility intervals of the current triangle, and the visibility intervals are z-buffered with the previous visibility intervals at the sampling location. The depth z from v_0 to the intersection of a triangle abc and camera ray r_s through s is given by:

$$z = \frac{(a - v_0) \cdot n}{r_s \cdot n}, n = (b - a) \times (c - a) \quad (5)$$

Since a, b, c, n, and r_s are constant, and since v_0 is linear in t, z is linear in t. For scenes where triangles do not intersect, z-buffering two overlapping visibility intervals is simply done by evaluating the two depth functions at a t inside the overlap. When triangles could intersect, the possible intersection between two triangles is found by solving the quadratic equation that results from setting their z's (Eq. 5) to be equal at the sampling location. The algorithm is exact per sampling location, where visibility is analyzed continuously over the view segment. Since the pixel centers are part of the sampling locations, the visible set contains all triangles visible at pixel centers as the view translates. View rotations and focal length changes modify the sampling locations defined by the view rays, therefore the algorithm is exact only under view translation.

5 Over-Time-Interval Visibility

The from-segment visibility algorithm can be used to compute the triangles visible over a time interval in a dynamic scene. The input is a dynamic scene modeled with triangles whose vertices move linearly over a time interval $[t_0, t_1]$, and a view V. The output is the set of triangles visible at the pixels of V at any time in $[t_0, t_1]$. The visible set is exact for view V.

Eq. 4 used to compute the visibility events remains a quadratic in t, as now vertices a, b, and c are linear in t and v_0 is constant. Eq. 5 used to z-buffer the overlapping visibility intervals of two triangles is now a cubic in t over a quadratic in t, as n is quadratic in t. For applications where moving triangles do not intersect, such as for the finite element analysis (FEA) and the smoothed particle hydrodynamics (SPH) simulation applications used as examples in this paper, the depth functions are trivially evaluated to arbitrate between the two triangles. When triangles intersect, depth function equality results in an order five equation in t, which would have to be solved numerically.

The visible set is exact for the given view V because visibility is computed exactly for each sampling location, including the pixel centers of V. Visibility at a sampling location is computed continuously over the time interval by solving visibility event equations. The algorithm computes not only what triangles are visible at each sampling location for $[t_0, t_1]$, but also what triangle is visible when, which provides perfect occlusion culling for any t in $[t_0, t_1]$.

6 From-Rectangle Visibility

We have used all three elements of our visibility paradigm to develop an algorithm for computing visibility directly over a view rectangle. The input to the algorithm is a scene modeled with triangles, a view, and a view rectangle. The output is an aggressive set of triangles visible from the view rectangle. The visible set provides the same guarantees as before: the set contains all triangles that have a completely visible fragment, and the algorithm is exact under view translation. The algorithm has two stages, similarly to the from-segment algorithm (Section 4).

In stage 1, sampling locations are added to a conventional image to ensure that all triangle fragments are sampled. The triangle footprint is approximated with the convex hull of the twelve extremal projections of the triangle vertices (i.e. one for each of the four corners of the view rectangle, for each of three vertices).

In stage 2 visibility is computed exactly for each sampling location with another pass over the scene triangles. For each triangle, the footprint is approximated as described for stage one, and the visibility samples of the sampling locations covered by the footprint are updated. The viewpoint translation inside the view rectangle is described with two parameters v and t. Like before, visibility changes when a projected triangle edge crosses the sampling location. These visibility events are defined by lines in the 2D visibility parameter space (v, t). The three edges of a triangle define three visibility event lines, which define a triangle in (v, t). The triangle projection moving with two degrees of freedom in the image plane corresponds to a (static) triangle in the (v, t) plane. This can also be thought of as an orthographic projection of the scene triangle with the set of parallel rays obtained by translating the sampling location ray with two degrees of freedom. We solve visibility exactly for each sampling location, by running our exact from-point visibility algorithm (Section 3.2) in the (v, t) plane.

The table in Figure 10 (left) shows two triangle projections that move over a sampling location s as the viewpoint translates over a view rectangle. The 3D graph (right) shows the depth z at the sampling location as a planar function of the two translations, for each triangle. The (v, t) plane (bottom of graph) shows the 2D visibility sample stored at the sampling location.

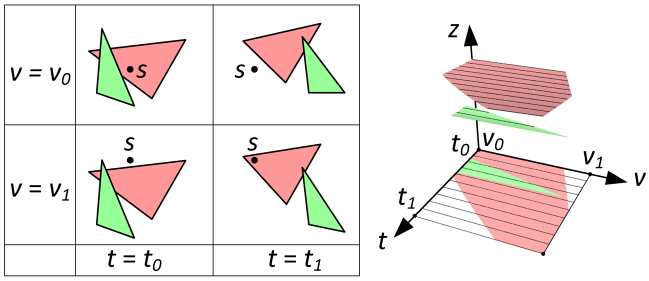

Figure 10: . *Left: two triangles projections moving over sampling location s as the view translates over a view rectangle $[v_0, v_1] \times [t_0, t_1]$. Right: z planes of the two triangles as they move over the sampling location, and 2D visibility sample in the (v, t) plane.*

7 Spherical Particles as Visibility Primitives

We have described our visibility algorithms for scenes modeled with triangles. The algorithms support any geometric primitive that can be tessellated. We have extended our from-point, from-segment, and over-time-interval visibility algorithms to support spherical particles directly, without the cost of increasing the number of primitives through tessellation. The extension has to solve three problems: (1) deciding whether a particle covers a sampling location, which is done in 3D by checking whether the distance d from the particle center to the sampling location ray is less than or equal to the particle radius r; (2) finding the visibility parameter values (translation or time) when a visibility event occurs, which is done by solving a quadratic equation that results from setting d equal to r, and (3) finding the centroid of a particle "fragment", which is the projection of the particle center if the projection is in-

side the pixel, or else the average of the intersection points between the pixel frame and the particle projection.

8 Results and Discussion

We have tested our visibility algorithms on a variety of scenes: *Manhattan* (4.0Mtris, Figure 4), *Grass* (55Mtris, Figure 1), *Forest* (47Mtris, Figure 1), *Isosurface* (500Mtris, Figure 11), *Water* (2.1M spherical particles over 80 states, Figure 3), *Impact* (2Mtris over 134 states, Figure 11), *Fusion* (500K spherical particles over 100 states, Figure 11). We organize the presentation and discussion of our results in four subsections: quality, efficiency, comparison to prior art methods, and limitations.

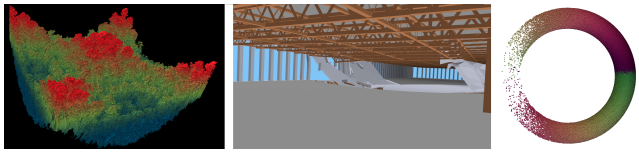

Figure 11: Isosurface, Impact, *and* Fusion *scenes*.

8.1 Quality

From-point visibility. The quality of the visibility solution computed by our aggressive from-point visibility algorithm is summarized in the top three rows of Table 1. Visibility completeness (row 1) is defined as the percentage of image area for which visibility is fully resolved by the aggressive set. Let S^* and S be the visibility subdivisions of the image induced by the aggressive and exact sets. Then visibility completeness is computed as the sum of the areas of the regions in S for which S^* provides the correct visible triangle. As seen from the table, the aggressive visible set all but completes the visibility subdivision of the image.

Table 1: *Quality and efficiency of from-point visibility algorithms.*

	Manhattan	Grass	Forest	Isosurface	Water
1. Completeness	99.9%	99.9%	98.1%	99.7%	N/A
2. Frame err. (avg)	0.03%	0.11%	2.11%	0.27%	0.09%
3. Frame err. (max)	0.08%	0.20%	3.0%	0.61%	0.13%
4. S.L. / pix (avg)	1.6	5.7	24	209	2.0
5. S.L. / pix (max)	57	1,490	1,245	1,670	27
6. Time A [s]	31	31	347	8,008	8.1
7. Iterations	2	3	3	3	N/A
8. Time E [s]	5.8	22	1,078	146	N/A

We have also estimated the quality of the algorithm by measuring the average and maximum per frame percentages of incorrect pixels over typical paths with thousands of frames. Both the reference image and the output frames have a resolution of $1,280 \times 720$. The errors are small, even though the paths include zooming in. The maximum zoom-in factors for the *Manhattan*, *Grass*, and *Forest* paths were $7\times$, $17\times$, and $10\times$. The errors only occur in between surfaces, which makes them less noticeable.

The exact from-point visibility algorithm supports rendering correct frames with any view direction and any zoom factor. We have developed a robust implementation of the algorithm based on a perturbation technique that evaluates control logic predicates correctly and without special handling of individual types of degeneracies [Sacks and Milenkovic 2014]. We start by perturbing the triangle vertex coordinates by an amount that is negligible in terms of visibility solution but sufficient to avoid degeneracies (i.e. $10^{-8}\%$ of the diameter of the scene). We evaluate predicates with floating point interval arithmetic, which provides an interval that contains the true value of the predicate. The sign of the predicate is determined unless the interval contains zero. We resolve such ambiguous cases by increasing the precision of the interval arithmetic, and thus shrinking the interval, until zero is excluded. The extended precision arithmetic is implemented with the MPFR library [Fousse et al. 2007]. Although it is costly, ambiguity is rare due to perturbation, so the overall efficiency is high.

1D visibility. We solve from-segment visibility and over-time-interval visibility with an algorithm that computes 1D visibility samples directly over the entire visibility parameter domain (i.e. translation or time). The algorithm computes visibility exactly for each sampling location, by z-buffering visibility intervals. The resulting visible set is therefore exact when the sampling locations do not change, as is the case when the view translates along the segment for from-segment visibility, or when the view does not change for over-time-interval visibility. Table 2 gives the error if instead of using our 1D visibility algorithm one approximates the visible set by unioning the two visible sets computed for the endpoints of the visibility parameter domain (i.e. view segment or time interval). The error is given as the average and maximum percentage of incorrect pixels per frame over a sequence of 1,000 frames uniformly sampling the visibility parameter domain.

Table 2: *Incorrect pixels when using union of endpoint visible sets. Our 1D visibility algorithm is exact in these cases.*

	From segment endpoints			From time interval endpoints		
	Forest	Grass	Manhattan	Impact	Water	Fusion
Avg.	26%	46%	2.5%	0.093%	0.17%	0.66%
Max.	34%	50%	5.2%	0.34%	0.35%	0.91%

The from-segment algorithm supports any segment length. The longer the segment, the bigger the benefit of the algorithm compared to simply computing visibility at the segment endpoints, but also the less efficient the algorithm becomes as discussed below. The over-time-interval algorithm is run for time intervals that are short enough for the motion of triangle vertices to be approximately linear over the interval. The scenes used here were animated by numerical simulation codes that define these intervals implicitly in between simulation states.

From-rectangle visibility. Table 3 reports the quality of the visible set computed by our from-rectangle visibility algorithm. The scene is $Manhattan$ and the output resolution is $1,280 \times 720$. The algorithm is compared to running the exact from-point algorithm from the corners of the view rectangle and unioning the resulting four visible sets. The error is given as the average and maximum number of incorrect pixels over two paths of 10,000 frames each, which sample the view rectangle. For the first path the view only translates, and for the second path the view translates and rotates. The view-rectangle algorithm is exact under view translation and it has substantially smaller errors than the four-corners algorithm when the view also rotates.

Table 3: *Quality of from-rectangle visibility algorithm.*

	From-rectangle		From rectangle corners	
	Avg.	Max.	Avg.	Max.
Tanslation only	0	0	202	362
Translation + rotation	0.01	1	274	948

8.2 Efficiency

The efficiency of our aggressive from-point visibility algorithm is summarized in rows 4-6 of Table 1. The number of sampling loca-

tions per pixel (rows 4-5) depends on the average image footprint of the triangles, and on the presence of large blockers, and therefore it is small for scenes like $Manhattan$ and large for scenes like $Isosurface$. The running times (row 6) were measured for a *serial* implementation on a 3.16GHz X5460 Intel workstation, and are all below 6min except for *Isosurface* where the average number of sampling locations per pixel is 209.

The efficiency of our exact from-point visibility algorithm is summarized in rows 7-8 of Table 1. The algorithm converges in at most three iterations (row 7), even for the *Grass* and *Forest* scenes which have tens of millions of triangles and complex occlusion patterns. The sampling locations generated based on the visibility subdivision (Figure 7) are very likely to resolve the visibility of an undecided triangle t by verifying that t is visible or by finding the triangle that occludes t. The running time (Time E, row 8) does not include the time for computing the starting aggressive visible set (Time A), and it was measured for a parallel implementation executed on 32 2.27GHz X7560 Intel cores. The algorithm is parallelized by subdividing the $1,280 \times 720$ reference image into 16×16 tiles, and by assigning the tiles to cores in round robin fashion.

The running times of our from-segment and our over-time-interval visibility algorithms are given in Table 4. For the over-time-interval algorithm the times given are the sum of all times over all intervals (i.e. 134, 80, and 100 intervals for the *Impact*, *Water*, and *Fusion* scenes). The times were measured for a parallel implementation that tiles the reference image, run on 4 3.16GHz X5460 Intel cores. Interval z-buffering uses a binary search tree and has an $O(n \log n)$ running time for a sampling location with n intervals.

Table 4: *Running times in minutes for our 1D visibility algorithm.*

	From-segment			Over-time-interval		
	Forest	**Grass**	**Manhattan**	**Impact**	**Water**	**Fusion**
Time	34	120	20	34	12	75

The running time of our from-rectangle visibility algorithm was 23h for the *Manhattan* scene for a parallel implementation that tiles the reference image, run on 20 2.53GHz E5649 Intel cores.

8.3 Comparison to Prior Art Methods

We compare the visibility computation capability of our image generalization paradigm to two prior art approaches.

The first prior art approach is the use of conventional images to aggregate an approximate visibility solution by uniformly sampling the high-dimensional space of visibility parameters. We compared our from-point visibility algorithms to computing visibility with a conventional image of ultra-high resolution, i.e. $32 \times 1,280 \times 32 \times 720$. This corresponds to a 32×32 uniform supersampling of the reference image used by our visibility algorithms. Even at the prohibitive cost of 1,024 samples per pixel, the conventional image fails to find all visible triangles: the average pixel errors for the cases used in Table 1 are 0.015%, 0.0078%, 0.11%, 0.25%, and 0.0001%. Our exact from-point visibility algorithm produces correct frames. For the *Manhattan* scene our aggressive algorithm yields a 0.03% frame error with 1.6 sampling locations per pixel, compared to the 0.015% error for the high resolution image with 1,024 sampling locations per pixel.

We compared our 1D visibility algorithm to sampling the visibility parameter domain, i.e. the view segment or the time interval. Even when the domain is sampled densely, the quality of the aggregate visibility solution is low. For example, for the *Forest* (Table 3), aggregating the visibility solution from 40 points on the view segment reduces the average error only to 4.9%, whereas our algorithm yields error-free frames. Similarly, we have compared our from-rectangle visibility algorithm to aggregating visibility from 50×50 conventional images supersampled by a factor of 32×32, rendered from viewpoints that sample the view rectangle uniformly. Again, the extensive uniform sampling of the visibility parameter domain failed to find all visible triangles (average and maximum frame errors of 2.5 and 41 pixels). Our algorithms do not search for the visible triangles blindly, through uniform sampling of the visibility parameter domain, but they rather find the visible triangles directly, by computing the visibility parameter values where visibility changes occur, through visibility event equations.

The second prior art approach to which we compare our image generalization paradigm for visibility is *adaptive global visibility sampling* (AGVS) [Bittner et al. 2009], which relies on individual rays to sample the space of visibility parameters heuristically. Table 5 gives the average and maximum frame errors obtained for a path with 10,000 frames through a box-like view cell in the *Manhattan* scene. Our paradigm performs substantially better even when we only compute visibility at the eight corners of the view cell. The errors are further reduced when our more powerful visibility algorithms are used for the edges and the faces of the view cell.

Table 5: *Incorrect pixels per frame for prior and our approach.*

	AGVS	**Image Generalization paradigm**		
		8 corners	**12 edges**	**6 faces**
Avg.	435	21	4.9	0.56
Max.	3,198	210	124	21

8.4 Limitations

The current implementation of our aggressive from-point visibility algorithm has a running time quadratic in the number of sampling locations per pixel, as a new sampling location is created only after checking that none of the existing sampling locations is inside the current fragment. This is acceptable when the average number of sampling locations is small, but can become a bottleneck for scenes where the average triangle image footprint is small (e.g. *Isosurface*). In all our experiments the resolution of the reference image where visibility was computed was always the same as the resolution of the output image. Choosing a reference image resolution commensurate with the average triangle footprint will control the average number of sampling locations per pixel. Another option is to subdivide pixels hierarchically, e.g. with a quadtree, to find the sampling locations inside a fragment in logarithmic time.

Our over-time-interval visibility algorithm relies on the assumption that triangle vertices move with a constant velocity vector over the time interval, which reduces the complexity of the visibility event equations. The assumption requires subdividing time into multiple intervals and running the visibility algorithm for each interval. Generalizing our from-rectangle visibility algorithm to a from-segment *and* over-time-interval algorithm is non-trivial. When one visibility parameter is translation and the other one is time, the visibility event loci are not lines anymore, but rather curves, and the visibility subdivision becomes not polygonal.

Our algorithms rely on sampling locations to eliminate two visibility parameters. For example, from-rectangle visibility is reduced to solving a from-point visibility problem at each sampling location. However, this comes at the cost of redundancy–the same triangle is found as visible by many sampling locations. The bigger the span of the visibility parameter domain (e.g. view segment, time interval, view rectangle), the higher the redundancy.

9 Conclusions and Future Work

We have described a novel approach to visibility based on image generalization. The image is enhanced with sampling locations defined by scene geometry. A small number of sampling locations are sufficient to reveal most visible triangles. Our approach couples a sample-based and a continuous visibility analysis of the image plane to complete the visible set efficiently, in a remarkably small number of iterations. 1D and 2D visibility domains are handled directly, by solving visibility event equations, which reveals visibility changes without trial and error.

So far we have used our visibility algorithms to precompute visibility for complex scenes off-line. One direction of future work is to accelerate our visibility algorithms to support applications where visibility has to be computed in real time, such as antialiasing of minified geometry, motion blur, or soft shadow rendering. One option is to leverage the programmability of current graphics hardware (e.g. through CUDA); another option is to devise hardware extensions that bring native support to rendering over framebuffers with a variable number of sampling locations per pixel, and with more complex sampling locations (e.g. a list of intervals, a 2D map).

A second direction of future work is to use the image generalization paradigm to develop more general visibility algorithms. For example, an exact from-rectangle visibility algorithm, which also provides exact from-box visibility when applied to the six faces of the box, requires the construction of a two-parameter dynamic visibility subdivision of the image; a quality-guaranteed aggressive from-rectangle *and* over-time-interval visibility algorithm requires 3D visibility samples, which could be implemented by voxelizing the translation by translation by time visibility parameter volume.

References

APOSTU, O., MORA, F., GHAZANFARPOUR, D., AND AVENEAU, L. 2012. Analytic ambient occlusion using exact from-polygon visibility. *Computers & Graphics 36*, 6, 727–739.

AUZINGER, T., WIMMER, M., AND JESCKE, S. 2013. Analytic visibility on the gpu. In *Computer Graphics Forum*, vol. 32, Wiley Online Library, 409–418.

BITTNER, J., WIMMER, M., PIRINGER, H., AND PURGATHOFER, W. 2004. Coherent hierarchical culling: Hardware occlusion queries made useful. In *Computer Graphics Forum*, vol. 23, Wiley Online Library, 615–624.

BITTNER, J., MATTAUSCH, O., WONKA, P., HAVRAN, V., AND WIMMER, M. 2009. Adaptive global visibility sampling. In *ACM Transactions on Graphics (TOG)*, vol. 28, ACM, 94.

CARPENTER, L. 1984. The a-buffer, an antialiased hidden surface method. *ACM SIGGRAPH Computer Graphics 18*, 3, 103–108.

CATMULL, E. 1978. A hidden-surface algorithm with anti-aliasing. In *ACM SIGGRAPH Computer Graphics*, vol. 12, ACM, 6–11.

CHANDAK, A., LAUTERBACH, C., TAYLOR, M., REN, Z., AND MANOCHA, D. 2008. Ad-frustum: Adaptive frustum tracing for interactive sound propagation. *Visualization and Computer Graphics, IEEE Transactions on 14*, 6, 1707–1722.

CHARNEAU, S., AVENEAU, L., AND FUCHS, L. 2007. Exact, robust and efficient full visibility computation in plücker space. *The Visual Computer 23*, 9-11, 773–782.

COHEN-OR, D., CHRYSANTHOU, Y. L., SILVA, C. T., AND DURAND, F. 2003. A survey of visibility for walkthrough applications. *Visualization and Computer Graphics, IEEE Transactions on 9*, 3, 412–431.

CUI, J., ROSEN, P., POPESCU, V., AND HOFFMANN, C. 2010. A curved ray camera for handling occlusions through continuous multiperspective visualization. *Visualization and Computer Graphics, IEEE Transactions on 16*, 6, 1235–1242.

DÉCORET, X., DEBUNNE, G., AND SILLION, F. 2003. Erosion based visibility preprocessing. In *Proceedings of the 14th Eurographics workshop on Rendering*, Eurographics Association, 281–288.

DURAND, F., DRETTAKIS, G., THOLLOT, J., AND PUECH, C. 2000. Conservative visibility preprocessing using extended projections. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 239–248.

DURAND, F., DRETTAKIS, G., AND PUECH, C. 2002. The 3d visibility complex. *ACM Transactions on Graphics (TOG) 21*, 2, 176–206.

DURAND, F. 2010. *3D Visibility: analytical study and applications*. PhD thesis, Université Joseph Fourier.

FOUSSE, L., HANROT, G., LEFÈVRE, V., PÉLISSIER, P., AND ZIMMERMANN, P. 2007. MPFR: A multiple precision binary floating point library with correct rounding. *ACM Transactions on Mathematical Software 33*, 13.

FURNAS, G. W. 1986. *Generalized fisheye views*, vol. 17. ACM.

GOODRICH, M. T. 1992. A polygonal approach to hidden-line and hidden-surface elimination. *CVGIP: Graphical Models and Image Processing 54*, 1, 1–12.

GRIBEL, C. J., BARRINGER, R., AND AKENINE-MÖLLER, T. 2011. High-quality spatio-temporal rendering using semi-analytical visibility. In *ACM Transactions on Graphics (TOG)*, vol. 30, ACM, 54.

HAUMONT, D., MÄKINEN, O., AND NIRENSTEIN, S. 2005. A low dimensional framework for exact polygon-to-polygon occlusion queries. In *Proceedings of the Sixteenth Eurographics conference on Rendering Techniques*, Eurographics Association, 211–222.

HECKBERT, P. S., AND HANRAHAN, P. 1984. Beam tracing polygonal objects. In *ACM SIGGRAPH Computer Graphics*, vol. 18, ACM, 119–127.

JOHNSON, G. S., LEE, J., BURNS, C. A., AND MARK, W. R. 2005. The irregular z-buffer: Hardware acceleration for irregular data structures. *ACM Transactions on Graphics (TOG) 24*, 4, 1462–1482.

JONES, T. R., AND PERRY, R. N. 2000. Antialiasing with line samples. In *Rendering Techniques 2000*. Springer, 197–205.

KATZ, M. J., OVERMARS, M. H., AND SHARIR, M. 1992. Efficient hidden surface removal for objects with small union size. *Computational Geometry 2*, 4, 223–234.

MAX, N., AND OHSAKI, K. 1995. Rendering trees from precomputed z-buffer views. In *Rendering Techniques 95*. Springer, 74–81.

MCMILLAN, L., AND BISHOP, G. 1995. Plenoptic modeling: An image-based rendering system. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, 39–46.

MORA, F., AND AVENEAU, L. 2005. Fast and exact direct illumination. In *Computer Graphics International 2005*, IEEE, 191–197.

NIRENSTEIN, S., AND BLAKE, E. H. 2004. Hardware accelerated visibility preprocessing using adaptive sampling. *Rendering Techniques 2004*, 15th.

OVERBECK, R., RAMAMOORTHI, R., AND MARK, W. R. 2007. A real-time beam tracer with application to exact soft shadows. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, Eurographics Association, 85–98.

POPESCU, V., EYLES, J., LASTRA, A., STEINHURST, J., ENGLAND, N., AND NYLAND, L. 2000. The warpengine: An architecture for the post-polygonal age. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 433–442.

SACKS, E., AND MILENKOVIC, V. 2014. Robust cascading of operations on polyhedra. *Computer-Aided Design 46* (Jan.), 216–220.

SHADE, J., GORTLER, S., HE, L.-W., AND SZELISKI, R. 1998. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, 231–242.

SHARIR, M., AND OVERMARS, M. H. 1992. A simple output-sensitive algorithm for hidden surface removal. *ACM Transactions on Graphics (TOG) 11*, 1, 1–11.

WEILER, K., AND ATHERTON, P. 1977. Hidden surface removal using polygon area sorting. In *ACM SIGGRAPH Computer Graphics*, vol. 11, ACM, 214–222.

WONKA, P., WIMMER, M., ZHOU, K., MAIERHOFER, S., HESINA, G., AND RESHETOV, A. 2006. Guided visibility sampling. *ACM Transactions on Graphics (TOG) 25*, 3, 494–502.

YU, J., AND MCMILLAN, L. 2004. General linear cameras. In *Computer Vision-ECCV 2004*. Springer, 14–27.

ZHANG, H., MANOCHA, D., HUDSON, T., AND HOFF III, K. E. 1997. Visibility culling using hierarchical occlusion maps. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 77–88.