Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2012)
P. Kry and J. Lee (Editors)

Occlusion-free Camera Control for Multiple Targets

M. Christie[1] and J.-M. Normand[2] and P. Olivier[3]

[1]MIMETIC, INRIA Rennes Bretagne-Atlantique, France
[2]LUNAM Université, École Centrale de Nantes, CERMA UMR 1563, Nantes, France
[3]Culture Lab, School of Computing Science, Newcastle University, UK

Abstract

Maintaining the visibility of target objects is a fundamental problem in automatic camera control for 3D graphics applications. Practical real-time camera control algorithms generally only incorporate mechanisms for the evaluation of the visibility of target objects from a single viewpoint, and idealize the geometric complexity of target objects. Drawing on work in soft shadow generation, we perform low resolution projections, from target objects to rapidly compute their visibility for a sample of locations around the current camera position. This computation is extended to aggregate visibility in a temporal window to improve camera stability in the face of partial and sudden onset occlusion. To capture the full spatial extent of target objects we use a stochastic approximation of their surface area. Our implementation is the first practical occlusion-free real-time camera control framework for multiple target objects. The result is a robust component that can be integrated to any virtual camera control system that requires the precise computation of visibility for multiple targets.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Camera control is a basic requirement for 3D computer graphics applications and in recent years a variety of techniques have been developed to automate camera control for tasks ranging from object inspection to assisted navigation. In most applications the aim of a camera control system is to maintain informational and aesthetic views of scene elements, whilst at the same time freeing the user from having to exercise low-level control of the camera parameters.

In practice, a number of factors determine the sophistication of the camera control framework required, including the nature of the user's task, the visuospatial qualities of the graphical environment, and other application domain specific constraints. Furthermore, the high dimensionality of the search space (a simple camera model has at least 7 degrees of freedom) and the intrinsic complexity of 3D environments constitute significant barriers to development of expressive real-time approaches.

An intrinsic property of any camera control system is the ability to compute and reason about the visibility of target objects in dynamic environments. However, in contrast to shadow computation and occlusion culling, the issue of visibility in camera control has received relatively little attention. Current real-time approaches to the computation of occlusion-free views of target objects (e.g. in computer games) rely almost exclusively on simple ray casting techniques.

In this paper we propose a new approach to real-time occlusion-free camera control, which addresses many of the limitations of existing approaches:

Multi-object visibility We have developed a technique that can sample the visibility of multiple targets from multiple viewpoints (see Figure 1).

Stochastic estimation of visual extent We use a stochastic approximation to compute the visibility of a whole target.

Dynamic qualities By accumulating visibility information over consecutive frames we implement the first quantitative parametrization of a camera's dynamic behavior.

2. Visibility Methods in Camera Control

Only a small number of real-time approaches for occlusion-aware camera control have been proposed (for a compre-

DOI: 10.2312/SCA/SCA12/059-064

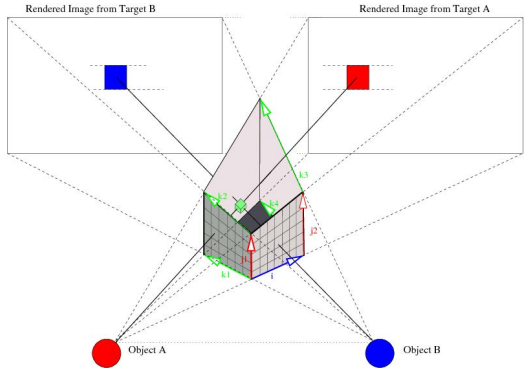

Figure 1: *Intersection of two renderings originating from target objects A and B. In this intersection region (grey box), we compose the visibility information for both targets.*

hensive survey of automated camera control see [CON08]). Crucially, existing techniques (e.g [HHS01]) cannot be easily extended for multiple target objects, and fail to capture the full spatial extent of target objects (i.e. they model target objects as points). The computation of occlusion-free viewpoints is closely related to the well known problem of visibility determination [COCSD00, Dur00] which has a bearing on a range of sub-fields in computer graphics, from hidden surface removal and occlusion culling, to global illumination and image-based modelling and rendering.

Visibility methods aim to calculate either the regions of a space which can be seen from a point (*from-point* visibility computation), or those that can be seem from a region (*from-region* visibility computation). In simple terms, visibility determination uses *visual events* – the boundary configurations for which the visibility changes – to partition space. Such methods can be broadly categorized according to the space in which the partitioning is performed, that is, object space, image space, viewpoint space or line-space (for a detailed presentation see [Dur99]).

The efficiency and simplicity of ray casting make it the default choice for evaluating visibility in many real-time camera control applications, in particular, computer games [Gio04]. An alternative to ray casting is to use *consistent regions* of space through the representation of the visibility of a target object in local spherical coordinates centered on the object [BL99, PBG92, DZ95].

Hardware-based approaches to real-time visibility for camera control [HO00] evaluate the degree and extent of occlusion by rendering a scene in stencil buffers using a color for each object. Such techniques have a number of attractive properties including an independence from the internal representation of the objects, and, by avoiding bounding volumes and other geometric approximations of the object, a more accurate calculation of occlusion. Approaches based on rendering also allow the use of low resolution buffers where appropriate [PBG92, HHS01]. Although efficient, such techniques are not readily extended to more that one target object and the visual extent of most target objects is not well approximated by a point. Furthermore, truly expressive approaches to camera control require the management of both partial and temporal occlusions.

3. Occlusion-free Camera Control

Our main contribution is the provision of a method for the efficient evaluation of the visibility of one or more targets for a limited region around the current camera location. The method is robust and efficient and can be integrated as a component in charge of handling visibility for many camera control systems. Real-time visibility evaluation is closely related to the problem of computing shadows for complex light sources and receivers, and a number of shadow techniques can be employed. Our system uses the principle of shadow maps to evaluate the visibility of single or multiple targets within a restricted search space (a set of candidate camera locations) and composes the resulting visibility information. Furthermore, we use multiple penumbra maps per target object to better represent the visual extent of complex objects in a manner similar to [ARHM00]. The process is divided in four steps that we present in the following sections.

3.1. Bound: computing the visibility volume

The dynamics of the camera (position, velocity and acceleration) bounds the movement of the camera within a plausible subset of space. Given the maximum possible acceleration a_{max} in any direction as determined by the application, the current camera position $\mathbf{c}_t$ at time t, and its velocity $\mathbf{v}_t$, it is possible to compute a boundary volume within which the camera must reside at the next frame (we refer to this volume as the predictive camera volume) as a sphere located at $\mathbf{c}_t + \Delta t v_t$ of radius $\Delta t^2 a_{max}$. Frustums can then be designed to fully encompass this region.

In practical cases, to reduce the locality of our search, and thereby reduce the impact of failures (*i.e.* when the whole predictive camera volume is occluded), the radius of the sphere is multiplied by a factor n which can be dynamically controlled depending on the application.

3.2. Sample: rendering from both viewpoints

While the visibility of a single target object can be computed using a depth map and a standard symmetric frustum, for multiple target objects we propose the construction of non-symmetric frustums for each pair of target objects. The intersection of these frustums defines a visibility volume inside the predictive camera volume, in which we can compose the visibility information for a selected set of candidate camera positions. By using the depth values for each projection we can obtain 3D information as to the visibility of the target

objects in the visibility volume and can address the visibility information using a trilinear interpolation (see Figure 1). Defining a common far plane for the frustums (behind the predictive visibility volume), guarantees that rays for corresponding scanlines of each frustum intersect. These intersection define a set of sample points within the predictive camera volume for which the visibility of both target objects in known precisely (i.e. no interpolation of visibility information required).

The geometry of the projection and the resolution of the images determine the granularity of the sampling of the visibility volume. The resolution of the rendering, which defines the precision of the visibility computations, can be dynamically computed given a density parameter inside the *feasible camera volume* (number of samples per unit of volume).

Furthermore, visibility volumes for successive frames can be aggregated to stabilize the camera (i.e. make the camera robust to momentary occlusion).

3.3. Aggregate: combining visibility information

We evaluate the visibility status of each area as one value of $\{N, P_A, P_B, V\}$ such that:

N Neither of the two objects are visible
P_A Partially visible (only object A is visible)
P_B Partially visible (only object B is visible)
V Visible (both objects)

We thus define the composition operator $\otimes$ that computes the visibility state from two depth values z^A and z^B as:

$$z^A \otimes z^B = \begin{cases} N & if \quad (z^A < z^{AI}) \wedge (z^B < z^{BI}) \\ P_A & if \quad (z^A > z^{AI}) \wedge (z^B < z^{BI}) \\ P_B & if \quad (z^B > z^{BI}) \wedge (z^A < z^{AI}) \\ V & if \quad (z^A > z^{AI}) \wedge (z^A > z^{AI}) \end{cases}$$

and we can extend the composition operator $\otimes$ to vectors by building the matrix of visibility states:

$$\mathbf{z^A} \otimes \mathbf{z^B} = \begin{bmatrix} z_1^A \otimes z_1^B & \cdots & z_1^A \otimes z_m^B \\ \vdots & \ddots & \vdots \\ z_n^A \otimes z_1^B & \cdots & z_n^A \otimes z_m^B \end{bmatrix}$$

Where there is no occluder, the z-buffer defaults to the maximum depth value which is always greater than the distance to the intersection point I.

In order to efficiently compute and access the visibility states inside the visibility volume, we choose to express the intersection of the frustums as a 3D trilinear coordinate system as displayed in Figure 1. Each point I inside the visibility volume is represented in local coordinates, $I = [uvw]^T$, and expressed in global Cartesian coordinates as a linear combination of vectors $\mathbf{i}, \mathbf{j}_1, \mathbf{j}_2, \mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4$ (where $\mathbf{i}, \cdots, \mathbf{k}_4$ are defined in Cartesian coordinates). We refer to $\mathbf{I_{AB}} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ as the trilinear interpolation function related to the visibility volume for target objects A and B, that expresses local coordinates $[uvw]^T$ in Cartesian coordinates $I' = [xyz]^T$, where:

$$\begin{aligned} I' = \quad & \mathbf{I_{AB}}([uvw]^T) \\ = \quad & u.\mathbf{i} + v((1-u)\mathbf{j}_1 + u\mathbf{j}_2) + \\ & w((1-u)((1-v)\mathbf{k}_1 + v\mathbf{k}_2) + u((1-v)\mathbf{k}_4 + v\mathbf{k}_3)) \end{aligned}$$

We can now very conveniently use the local coordinates $[uvw]^T$ to address the depth values of each rendered image. Indeed, a pixel (u,v) in image B, together with a pixel $(1-w,v)$ in image A (the second component is identical as only rays in corresponding scanlines are intersected) intersect exactly at local coordinates $[uvw]^T$ inside the visibility volume. This enables both the efficient computation of all the intersection points inside the volume (only through vector algebra, at a cost of one sum per point if computed incrementally), as well as direct access to the visibility states in the corresponding 3D matrix given a pair of 2D coordinates in the image. Furthermore, it is possible to compute the inverse function $(\mathbf{I_{AB}})^{-1}$ to obtain the local coordinates of any point inside the visibility volume, and thus establish its visibility status (see Section 3.3).

By utilizing standard graphics hardware, this model enables the efficient computation of a sampling of the visibility volume. Knowledge of the visibility of the target objects, for viewpoints at and around the current camera location, is used as the basis for choosing whether to move the camera, and where to move it to (see Section 3.4). The granularity of the visibility volume is directly dependent on the resolution of the 2D renderings and is easily controlled according to the characteristics of the environment and the resource demands of the application.

Moreover, by using visibility volumes, our approach can be seamlessly extended to more than two target objects, cf. Figure 2.

3.4. Move: choosing the next camera position

Choosing the best next camera position within the *predictive camera volume* is application-dependent. Our task is therefore to provide parameters that capture the camera's behavior in common situations, including its responsiveness (speed of change in occluded configurations), coherence (maintenance of visual properties), and its predictive power (its ability to look-ahead and avoid occluded configurations).

Currently, we compute four different sets of camera regions: $\mathcal{S}_N$ (where both A and B are not visible), $\mathcal{S}_{PA}$ (only A is visible), $\mathcal{S}_{PB}$ (only B is visible), and $\mathcal{S}_V$ (both A and B are visible). Responsiveness is enforced by choosing camera locations that try to always ensure the visibility of the targets (*i.e.* in $\mathcal{S}_V$), generally at the cost of jumpiness and sudden

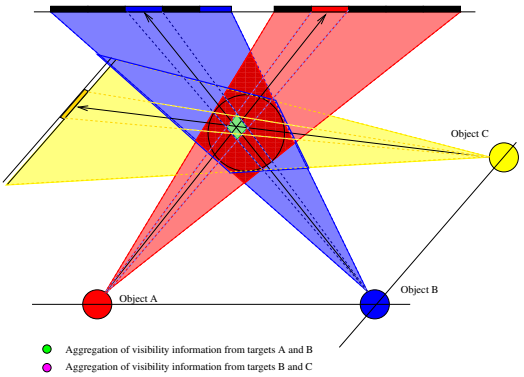

Figure 2: *Visibility computation for three targets A, B and C. Two visibility volumes V_{AB} and V_{BC} are computed and each configuration of V_{AB} is expressed in the basis of V_{BC} to evaluate its visibility status. The rendering from object B is performed only once and resampled for use by both visibility volumes.*

accelerations (although still within the acceleration bounds). Coherence is enforced by choosing areas where the visual properties on the screen are maintained over time, such as camera angles with respect to targets, camera heights, camera shots. Finally, the predictive power of the camera is achieved by selecting locations in $\mathcal{S}_V$ for which promixal neighboring regions are also unoccluded.

4. Stochastic Model of Visual Extent

In a pragmatic way to approximate the visual extent of objects (and thus approximate the visibility of the entire object, rather than a point on the object), we propose a tessellation independent stochastic method that automates the choice of target points on the surface of an object for different viewpoints and distances. We cycle between these target points in a round robin manner, selecting one point per frame, and the accumulation of this visibility information over different frames provides an improved estimate of the overall visibility of the target.

In a two-step process, we off-line distribute camera configurations evenly around the object and perform basic renderings. A random distribution of points is projected onto each rendered image, and points inside the projected extent of the object are retained. For each point, its 3D position on the surface of the object is computed by inverse projection. The density of the sampling on the rendered image is kept uniform, whatever the distance to the camera, so that objects that are larger (or closer to the camera) present more samples on their surface. These points approximate the visual extent of the target object and are stored in a view indexed sample table. In the second step, in the course of computing the real-time visibility, we use the relative orientation of the target, the visibility volume, and the distance between them, to look up the points in the sample table from which we select the particular point to be used for the projection (in a round robin manner).

5. Temporal Visibility Volume

Another important problem for real-time camera control, and one that is related to visibility, is the maintenance of shot coherency (*i.e.* avoiding abrupt and visually incongruous changes in viewpoint). If a camera only reacts to features that are 'in shot', then a sudden onset of occlusion that occurs in dynamic and complex environments will result in equivalently abrupt responses by the camera. To mitigate this we need incorporate mechanisms that stabilize the camera movements according to the spatial and temporal evolution of visibility. By monitoring the accumulation of the visibility information over the successive frames we can explicitly control the degree to which the camera is sensitive to partial and short-lived occlusions.

We define the aggregation operator $\bigoplus_\alpha$ as a low-pass filter which, applied over a set of visibility states $\{\cdots v^t, \cdots, v^{t-i}, \cdots v^{t-n}\}$, retains a visibility state v if $\frac{x_v}{n} \geq \alpha$, where x_v represents the number of occurrences of state v, n is the number of frames and $0 \leq \alpha \leq 1$. We treat cases where no state meets this condition pessimistically by considering them as occluded (value N).

As the visibility volumes evolve in space, some boundary configurations cannot be expressed in previous bases. Rather than adopting a pessimistic approach, declaring the status of out-of-bounds positions as occluded (value N), we count the occurrences of visibility states on all but out-of-bounds configurations. This *temporal visibility volume*, with which we accumulate the visibility states of past frames, can be coupled with an estimation of visual extent to provide meaningful control of the desired degree of visibility of a target.

Experimental results, as well as example camera behavior in the accompanying video, demonstrate the improved temporal stability situations that involve partial occlusion and encounters with short-lived occluders (see Section 6). The complexity of this process is bounded by $O(ng^3)$ as we only aggregate for points inside the feasible camera volume. In our implementation, the complexity is readily reduced to $O(g^3)$ by locally storing frame indices for each component visibility state.

6. Evaluation and Discussion

We have conducted a number of experiments on our prototype implementation of occlusion-aware camera control with a view to evaluating the different aspects so far described. All evaluations were run on an Intel Core 2 T7600 at 2.33 Ghz, 2GBytes of RAM and with a NVidia FX 3500 graphics card running the Ubuntu Linux system.

For each experiment, we have measured and reported the following properties:

- t_c the average time in milliseconds required to compute an occlusion-free view (this includes the hardware rendering, the computation of the intersections, and the choice of the new camera configuration).
- d_c the total distance covered by the camera during the experiment (used as a proxy for camera stability).
- d_s the distance covered in 2D by the projected centers of the target objects (used as a proxy for coherence).
- r_N the proportion of fully occluded frames (as a percentage).
- r_P the proportion of partially occluded frames (as a percentage).

The costs related to the different steps of the visibility computation process, for a number of different resolutions, are provided in Table 1. The size of the accumulation window has very little impact on the results. The extraction of the depth buffer information uses the OpenGL `glReadPixels()` function over the depth buffer which could be further improved by using Frame Buffer Objects.

An initial performance evaluation of our camera control scheme, for a moderately complex scene in Ogre3D (280K triangles), over 5000 frames, confirms that the computation of the intersections is the performance bottleneck (see Table 1). It should be noted that the expensive step of composing depth information can be entirely hardware accelerated by using vertex and fragment shaders. However, with a low resolution buffer (7×7), we can evaluate more than 300 possible camera configurations (and select the best one) in less than 3ms. Both visibility evaluation and viewpoint selection are exclusively performed on the CPU. However, in addition to the computational cost, we need to evaluate other aspects of camera behavior that our approach to occlusion-aware camera control seeks to facilitate. These include responsiveness, coherence, predictive power and temporal stability, as well as the simultaneous visibility of more than two target objects and the incorporation of supplementary properties.

Table 1: *Performance (ms) of the visibility computation for different resolutions r for (i) hardware rendering; (ii) intersection calculation; and (iii) next configuration selection.*

r	Samples	Render	Intersect	Select	Total
5	75	0.70	0.91	0.16	1.77
7	343	0.80	2.10	0.72	3.62
10	1000	0.80	5.20	1.81	7.83
15	3375	1.00	19.30	4.28	24.63

6.1. Predictive power

Our occlusion-aware technique detects and tries to maintain visibility within a camera's dynamic limits. Indeed, in both computer graphics applications and real-life camera motion, it is highly desirable to avoid sudden occlusion by using early anticipation. A partial solution simply involves making better choices between the candidates within the occlusion-free configurations, for example, by favoring configurations located away from occluded areas. To achieve this we gather free configurations together using their unoccluded neighbors in the visibility volume. The camera then moves towards the center of the largest connected group. In this benchmark we refer to this strategy as *RobustMove*. Our results (see Table 2) demonstrate improvements in the overall visibility during the shot. However, in the case of sharp turns and sudden occlusions, this strategy is insufficient. We use a simple prediction model (referred to as *Prediction*) that builds upon the past translational and rotational accelerations of target objects to estimate the future location.

Table 2 shows the results of our evaluation of the *RobustMove* and *Prediction* strategies, both separately and together, against the basic move strategy that selects the closest visible neighbor. For each run we computed the rate of full occlusions and the rate of partial occlusions (*i.e.* when at least one pixel is occluded) in the shot. The average ratio of occlusion (proportion of occluded pixels on the surface of the targets) is also computed. Both properties are measured in a post-process by re-rendering the shot with stored traces in medium-resolution buffers (512×512) to study pixel overlapping. The combination of *Prediction* and *RobustMove* provided the best results. The prediction time has to be carefully selected; in our example prediction times over 300ms yielded high occlusion rates due to significant differences between predicted and actual positions.

Table 2: *Comparison of occlusion models in combination with a prediction strategy as to the motion of the targets. r_N is the percentage of fully occluded frames, r_P is the percentage of partially occluded frames.*

Model	Pred. (ms)	$r_N(\%)$	$r_P(\%)$
Basic	-	12.32	8.41
RobustMove	-	7.47	7.12
Basic+Pred.	100	3.32	9.40
Basic+Pred.	300	1.12	6.33
Basic+Pred.	500	5.02	9.71
RobustMove+Pred.	100	0.59	3.08
RobustMove+Pred.	300	0.00	2.28
RobustMove+Pred.	500	1.41	5.89

6.2. Coherence

Coherence refers to a camera's ability to maintain the visual properties of a shot over time and can be measured in terms of the rate at which visual properties of a shot change. We enforce coherence by selecting cells that offer minimal change in visibility, camera orientation and the distance to the target. In evaluating camera coherence we measured the mean deviation of property satisfaction together with the sum of the camera accelerations over a time interval.

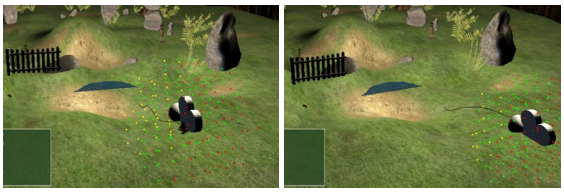

Figure 3: *Stability evaluation: a sparse occluder moves back and forth in front of two targets. Accumulating the visibility states reduces the jerkiness in the camera paths (left frame is without stability, right frame is with stability).*

6.2.1. Evaluation of temporal coherence

Temporal stability measures the responsiveness of the camera to sparse and/or fast moving occluders. In our experiment we used two target objects (characters) and a sparse occluder (fence) that moves back and forth in front of the targets for 10 seconds (see Figure 3). Table 3 shows the quantitative results.

Table 3: *Quantitative comparison of the stability of visual estimates by considering* 3 *parameters:* t_c *average time (ms),* d_c *distance covered by the camera,* d_s *the distance covered by the projected centers of the target objects. A stability of* $6/10$ *denotes that a configuration will be assigned a visibility value that has occurred at least* 6 *times in the past* 10 *frames.*

Experiment a/n	t_c (ms)	d_c	d_s
No stability $0/0$	2.69	157.0	1.7
Stability $3/5$	2.75	38.4	1.2
Stability $6/10$	2.87	4.1	0.1
Stability $8/15$	2.91	0.4	0.1

7. Conclusion

Our proposed approach to maintaining occlusion free views addresses a number of fundamental problems of camera control for 3D graphics applications. The ability to track one, two or more target objects without the imposition of significant computational cost is an important advance over both existing proposals from the research community and ray casting approaches widely deployed as 'best practice' in commercial applications. The stochastic modelling of visual extent means that objects are no longer treated as simple points or bounfing volumes.

The ability to accumulate visibility information over time also provides parameterized control over the dynamic behavior of the camera, both in terms of the tolerance of the camera to partial (and temporary occlusion) and spatial scope to which the search is extended in the event that none of the targets are visible (the escape strategy).

Although our occlusion-aware framework constitutes a significant advance over the state of the art, it has a number of noteworthy limitations. Firstly, our approach is based on a local exploration around the current camera configuration. This process could be interleaved with a visibility computation for a secondary camera position (eg. one frame on five) but this is a poor substitute for a global evaluation of camera locations which would better support the incorporation of jump cuts.

However, our proposed technique has significant potential to enhance applications that require assisted interactive or automated camera control in complex environments. The technique is lightweight, utilizes ubiquitous graphics hardware, and can be readily incorporated into the rendering process of any real-time graphics application.

References

[ARHM00] AGRAWALA M., RAMAMOORTHI R., HEIRICH A., MOLL L.: Efficient image-based methods for rendering soft shadows. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 375–384. doi:http://doi.acm.org/10.1145/344779.344954. 2

[BL99] BARES W. H., LESTER J. C.: Intelligent multi-shot visualization interfaces for dynamic 3D worlds. In *Proceedings of the 4th international conference on Intelligent user interfaces (IUI 99)* (New York, NY, USA, 1999), ACM Press, pp. 119–126. doi:http://doi.acm.org/10.1145/291080.291101. 2

[COCSD00] COHEN-OR D., CHRYSANTHOU Y. L., SILVA C. T., DURAND F.: A survey of visibility for walkthrough applications. *IEEE Transactions on Visualization and Computer Graphics 9*, 3 (2000), 412Ð431. 2

[CON08] CHRISTIE M., OLIVER P., NORMAND J.-M.: Camera control in computer graphics. *Computer Graphics Forum 27*, 8 (2008), 2197–2218. 2

[Dur99] DURAND F.: *3D Visibility: Analytical Study and Application*. PhD thesis, Université Joseph Fourier, Grenoble, Grenoble, France, July 1999. 2

[Dur00] DURAND F.: *ACM SIGGRAPH Course Notes: Visibility, Problems, Techniques, and Applications*. ACM Press, New York, NY, 2000, ch. A multidisciplinary survey of visibility. 2

[DZ95] DRUCKER S. M., ZELTZER D.: Camdroid: A System for Implementing Intelligent Camera Control. In *Symposium on Interactive 3D Graphics* (1995), pp. 139–144. URL: citeseer.ist.psu.edu/drucker95camdroid.html. 2

[Gio04] GIORS J.: The full spectrum warrior camera system. In *GDC '04 : Game Developers Conference 2004* (2004). 2

[HHS01] HALPER N., HELBING R., STROTHOTTE T.: A camera engine for computer games: Managing the trade-off between constraint satisfaction and frame coherence. In *Proceedings of the Eurographics Conference (EG 2001)* (2001), vol. 20, Computer Graphics Forum, pp. 174–183. 2

[HO00] HALPER N., OLIVIER P.: CAMPLAN: A Camera Planning Agent. In *Smart Graphics 2000 AAAI Spring Symposium* (March 2000), pp. 92–100. 2

[PBG92] PHILLIPS C. B., BADLER N. I., GRANIERI J.: Automatic viewing control for 3d direct manipulation. In *Proceedings of the 1992 symposium on Interactive 3D graphics* (1992), ACM Press New York, NY, USA, pp. 71–74. 2