D 5.3

REAL-TIME CINEMATIC SOLVER

Project Number	FP7-ICT-231824
Project Title	Integrating Research in Interactive Storytelling (NoE)
Deliverable Number	D5.3
Title of Deliverable	Real-time Cinematic Solver
Workpackage No. and Title	WP5 - Cinematography
Workpackage Leader	UNEW
Deliverable Nature	Report
Dissemination Level	Public
Status	Final
Contractual Delivery Date	30th June 2010
Actual Delivery Date	30th November 2010
Author(s) / Contributor(s)	Guy Schofield (UNEW), Patrick Olivier (UNEW) Marc Christie (INRIA), Christophe Lino (INRIA)
Number of Pages	44

DOCUMENT HISTORY

Version	Date	Comment	Author(s)
1.1	November 30, 2010	Draft version for approval	UNEW: Guy Schofield and Patrick Oliver; INRIA: Marc Christie, Christophe Lino

TABLE OF CONTENTS

Abstract

In deliverable D5.1 (Specification of Cinematic Idioms), we have described into details the major elements of cinematography and drawn some conclusions and recommendations as to what an interactive cinematography system should handle. The deliverable has been modified to take into account the reviewers recommendations on exploring more modern cinematographic styles and studying the importance of audio.

In deliverable D5.2 (Constraint-based Cinematic Idioms), we initially detailed a real-time approach to virtual cinematography which built upon cinematographic idioms to perform camera control. Following the reviewers recommendations, we fundamentally modified our approach in order to perform a shift from idiom-based representations to the representation and formalization of continuity rules in edits. Such rules are implicitly encoded in cinematic idioms, therefore the solution we proposed is far more expressive (no need to encode restricted idioms set) and furthermore offers means to explore variations in a number of directorial styles through the control of cinematic parameters. In particular, we have studied the impact of camera dynamicity (amount of motion of the camera), camera pacing (rhythm at which cuts occur), preferred viewpoints (a selection of appropriate semantic volumes to portray specific actions) and screen composition. The style of the movie we build is then controlled through these cinematic parameters, rather than through a collection of cinematic idioms. We furthermore used this set of parameters to enforce high-level narrative aspects such as dominance, affinity and isolation of characters (i.e. designing a mapping between the narrative aspect and a collection of cinematic parameters). The revised version of D5.2 therefore describes this expressive real-time cinematography system suited for interactive storytelling.

Now, in deliverable D5.3 (initially devoted to a real-time cinematic solver, a target already reached and described in D5.2), we propose to describe a major extension to our existing D5.2 real-time cinematic solver featuring ***dynamic visibility computation*** and its integration with Director Volumes.

This extension makes our real-time cinematic solver an efficient and expressive tool to explore directorial styles in an interactive virtual storytelling context. Open questions now remain, and concern (i) the means offered to the interactive storyteller (the author) to specify the directorial styles, (ii) the way an intermediate discourse level can control the parameters of the system (eg using character dominance over a selected set of actions) in coherency with a dramatic arc.

The organization of the deliverable is structured around the ***dynamic visibility computation***. We first detail how an efficient sampling-based visibility technique can be added to the system to handle complex dynamic occluders, we then describe how visibility can be estimated for complex occluders, and how temporal visibility can be computed. We finally detail the integration of the ***dynamic visibility computation*** in the Director Volume framework detailed in D5.2.

1. Introduction

The real-time system described in D5.2 only handles static occluders along the height axis, due to a 2.5D cell-and-portal decomposition of the environment. While the system efficiently reacts to cases where characters are hidden by these static occluders (typically walls, pillars and tall pieces of furniture), the case of smaller occluders demands a particular attention. In this context, we distinguish

.

1.1. Motivations for IS

The motivation to explore techniques for ***dynamic visibility computation*** is two-fold:

(i) adapt the visibility process to complex interactive scenes where actions may not be planned in advance, and there is the necessity to maintain the visibility on the characters whatever their motions, and whatever the motions of the potential occluders. Furthermore, specific behaviors are required to handle sparse occluders (such a s fences, hedges and trees) which are weak occluders and to handle short-lived occlusions

(ii) handle more dynamic behaviors or the camera, thereby enabling the implementation of more modern cinematographic styles such as hand-held cameras, long-takes in cluttered environments, and very dynamic motions of the camera.

1.2. Contributions

- **Multi-object visibility.** In simple ray casting approaches to visibility the candidate position for the camera is evaluated by casting a ray in the direction of a target object. This can be extended, for a single viewpoint, to multiple objects by repeating the process for each target object. By contrast we have developed a technique in which the depth buffer information of simple renderings originating from the target objects can be combined to sample the visibility of multiple targets from multiple viewpoints. By carefully specifying the geometry of the frustums of these renderings, the visibility of the target objects can be precisely computed for a sample of locations within the *feasible camera volume* (the volume of possible camera positions).

- **Stochastic estimation of visual extent.** Instead of computing the visibility of the centre of a target object (which underestimates the visual extent), or using a bounding sphere (which overestimates the visual extent), or some other idealization, we use a pragmatic estimate of the visual extent of a target object. For each frame at which we compute the visibility of a target object we choose a different point on its surface. In practice the resulting visibility information is accumulated across consecutive frames. Thus our use of a distribution of points functions as an effective stochastic approximation of the visual extent of the target object.

- **Dynamic qualities.** By accumulating visibility information over consecutive frames we implement the first quantitative parameterization of a camera's dynamic behavior. This takes the form set of properties that describe: (1) how a camera responds to the sudden onset of occlusion (responsiveness); (2) its propensity to maintain shots with unchanging visual qualities (visual coherence); and (3) its anticipation of future occlusion (predictive power).

Our technique uses standard elements of the 3D rendering pipeline; no lighting is required and through modifications to the resolution of the renderings the performance can be adaptively controlled in an application dependent manner. Though some sections of the implementation could be further optimized using a GPU, in our benchmarks the average time to compute an occlusion-free view for two target objects was less than 3 ms per frame. Furthermore, our approach allows us both to integrate our visibility computation with the evaluation of other visual properties (including relative orientation, distance and size) and incorporate predictive models of target object motion.

2. Inclusion of real-time visibility computation

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. We also develop a number of metrics with which to describe camera behavior, coherence, responsiveness and predictive power, and characterize our framework in terms of these. We demonstrate the efficiency and performance of our approach for a number of well known scene configurations that are not addressed by existing camera control techniques.

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. The simplest interactive camera control approaches propose direct mappings from a mouse, keyboard or other control device, to the camera parameters, with the appropriateness of these mappings being highly dependent on the underlying application domain and the tasks required to be performed. Automating aspects of camera control has a wide range of applications, from improving user experience in computer games to viewpoint management in scientific and information visualization. Indeed, virtually any interactive 3D graphics application is likely to benefit from a system by which the desirable properties of a view can be specified and computed in real-time [Christieetal08].

The requirements of a camera control system depends on whether the user has full or partial control over the camera parameters (in the target application) or whether the application assumes full responsibility for selecting the viewpoint from which a user will view a scene. Proposals for interactive camera control vary according to whether the control is *direct*, *through-the-lens* or *assisted*. Direct techniques vary according to the reference frame and metaphor used in mapping movements of a control device to the camera parameters (e.g. *flying vehicle*, *eyeball-in-hand*, *world-in-hand* [WareO90]) whereas *through-the-lens*} control allows users to control the camera through interactive manipulation of the position of objects on the screen [GleicherW]. Assisted control utilizes knowledge of the environment in assisting user exploring objects (e.g. using repulsive fields around objects [Beckhaus]) or larger environments (e.g. using a roadmap planner to find a free path and constraining the user to a collision free path [Oskam09]).

In practice, a number of factors determine the sophistication of the camera control framework required, including the nature of the user's task, the visuo-spatial 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. In general, proposed approaches aim to devise declarative formulations in which constraints can be placed on the visual properties of a shot (e.g. specifications of the size and position of target objects), and for which optimization or constraint satisfaction algorithms can be used to find camera locations and paths that satisfy these constraints. A camera control system capable of producing a genuinely cinematic experience [Arijon76], [Katz91] requires the development of richer sets of properties, and more powerful solving mechanisms than those that exist in current systems.

In theory, 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, although in Section 3 we review a number of more involved approaches including sphere-based projection [BaresL99] and shadow mapping [Philips]. In this Deliverable, we propose a new approach to real-time occlusion-free camera control, which addresses many of the limitations of existing approaches and is well suited to the interactivity and complexity of interactive narratives.

3. Related Work

Only a small number of real-time approaches for occlusion-aware camera control have been proposed (for a comprehensive survey of automated camera control, see [Christieetal08]). Crucially, existing techniques (*e.g.* [Halper03]) 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 [Cohen-or00, Durand00] 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 modeling 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 [Cohen-or00]). For static scenes different representations have been used for this partitioning including aspect graphs and boundary CSG models of polyhedral scenes [Plantinga90,Tarabanis96]. Visibility methods in dynamic environments have mostly addressed the problem of updating these visibility representations for moving objects [Sudarsky99] and modeling moving occludees (e.g. motion volumes [Durand00]).

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 [Giors04]. 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 [BaresL99]. A consistent region can be computed by projecting the bounding boxes of nearby potentially occluding geometry onto a discretized sphere surrounding the target object, converting these projections into global coordinates, and finally negating these to yield occlusion free viewpoints [BaresL99,Philips,DruckerZ95]. Similarly, Courty [Courty01] and Marchand [Marchand98, Marchand02] avoid occlusion in a target tracking problem by computing an approximate bounding volume that encompasses both the camera and the target. Occluders (i.e. not the camera or the target objects) are prevented from entering the volume corresponding to target motion or camera motion. However, the approximate nature of the bounding volumes restricts both expressiveness (*e.g.* an inability to represent partial occlusion) and practical application (*e.g.* over-estimation for complex shapes).

Actually, the problem of multiple object visibility can be considered as tightly related to the computation of penumbra volumes [Durand00] with complex-shaped light sources: the targets represent complex light sources, and the occlusion-free views are the areas without penumbra. In such, the technique we propose reflects over some principles of real-time penumbra computation in dynamic environments [Assarsson03].

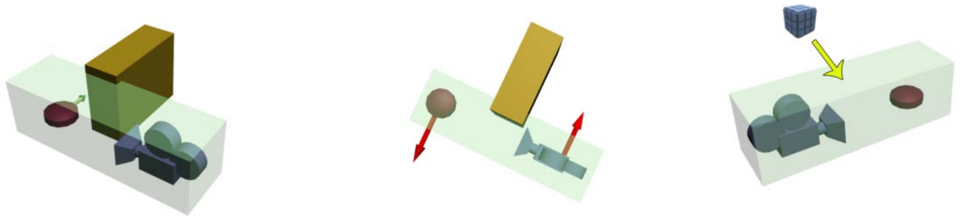


Figure 1: Characteristic situations leading to occlusion (left) target hides behind an occluder, (middle) target is occluded due to camera motion, and (c) a mobile occluder hides the target. The detection of these characteristic situations is necessary to provide the best possible anticipations.

Hardware-based approaches to real-time visibility for camera control [Halper00] 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. Phillips *etal.* [Philips] project a scene onto the different faces of a cube to achieve a visibility map, and choose the closest empty area in the map by local neighborhood exploration. Halper *etal.* [Halper03] introduced the use of geometric primitives referred to as *potential visibility regions* (PVRs) that are configured around the current camera location. By rendering, from the target to the camera, using distinct colors for occluders and PVRs, the resulting color and depth buffers can be used to inform the choice of the next camera movement. 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 occlusion.

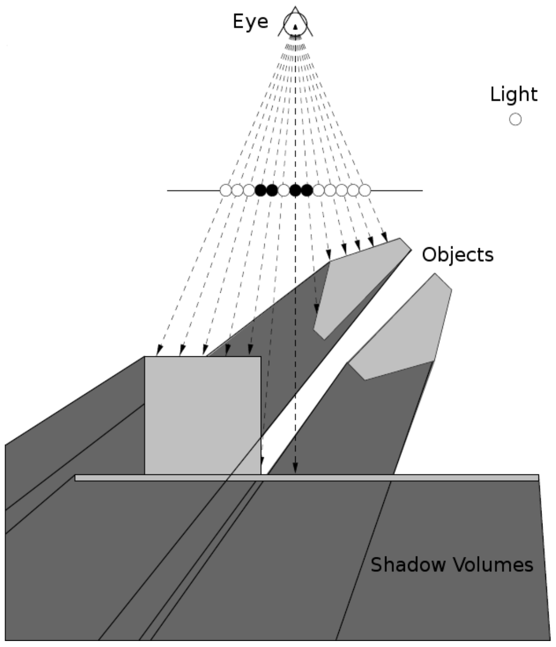


Figure 2: Shadow volumes generated by a point light source. If we consider the point light as the target object (object for which we want to maintain the visibility), the shadow volumes technique enables the computation of all the viewpoints (areas in viewpoint space) for which the target is not occluded. The extension to non point light sources generated penumbra volumes (volumes where the target is partially visible).

Where performance is not a significant issue, occlusions can be addressed using object-space techniques. For example, a shadow-volume based algorithm applied on the camera and the obstacles can be utilized to determine the parts of the scene that are unoccluded from the current viewpoint [Pickering02] (*cf.* Fig. 2). The object of interest is treated as the light source and the computed volumes represent the occluded areas.

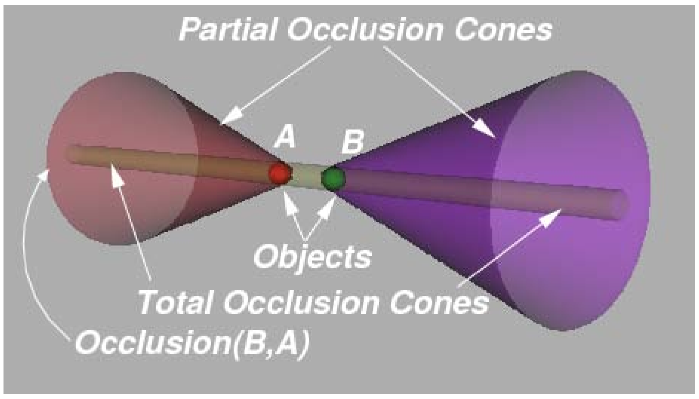


Figure 3: Boundary visibility computation using cones. When target objects and occluders are modeled with spheres, the computation of respective occluding cones between couples (target, occluder) enables the characterization of areas of full visibility, areas of partial visibility and areas of no visibility. The spatial approximations due to the spherical representations leads to large over-estimation of visibilities. Nonetheless the computational cost remains interesting (simple dot products are used characterize the visibilities).

Geometric management of occlusion has also been proposed using occlusion cones computed from the bounding spheres of the scene elements [ChristieEG05]. This method distinguishes between partial and total occlusions (*cf*. Fig. 3) but is only applicable to static scenes since each movement of an object requires an expensive re-computation of the cones.

A clear parallel can be drawn between the problem of real-time soft shadow computation and real-time visibility computation of target objects. Target objects can be treated as light sources for which we need to compute the volumes outside of the shadow and penumbra (this is an inverse volume carving problem) in which to place a camera. One technique for real-time shadow computation relies on silhouette detection (*e.g.* penumbra wedges [Assarsson03]), that use the exact silhouette of objects to compute shadow volumes. However, the complexity of silhouette detection increases with the complexity the objects casting shadows and such approaches are also not readily applicable to rasterizable entities that use alpha-textures (which are increasingly used real-time 3D graphics). Another class of techniques relies on frame-buffer approaches that construct a depth map rendered from the location of light sources using graphics hardware. This *shadow map* is then sampled in relation to the world geometry and a simple depth comparison can be used to determine the status of a point in space (i.e. whether it is in shadow).

4. Our proposal to handle dynamic visibility

In shadow maps literature, accounts either consider sampling techniques that poll all the depth maps or 3D warping techniques that map the local shadow information of each light source onto the view of the world (see [Zhang98]).

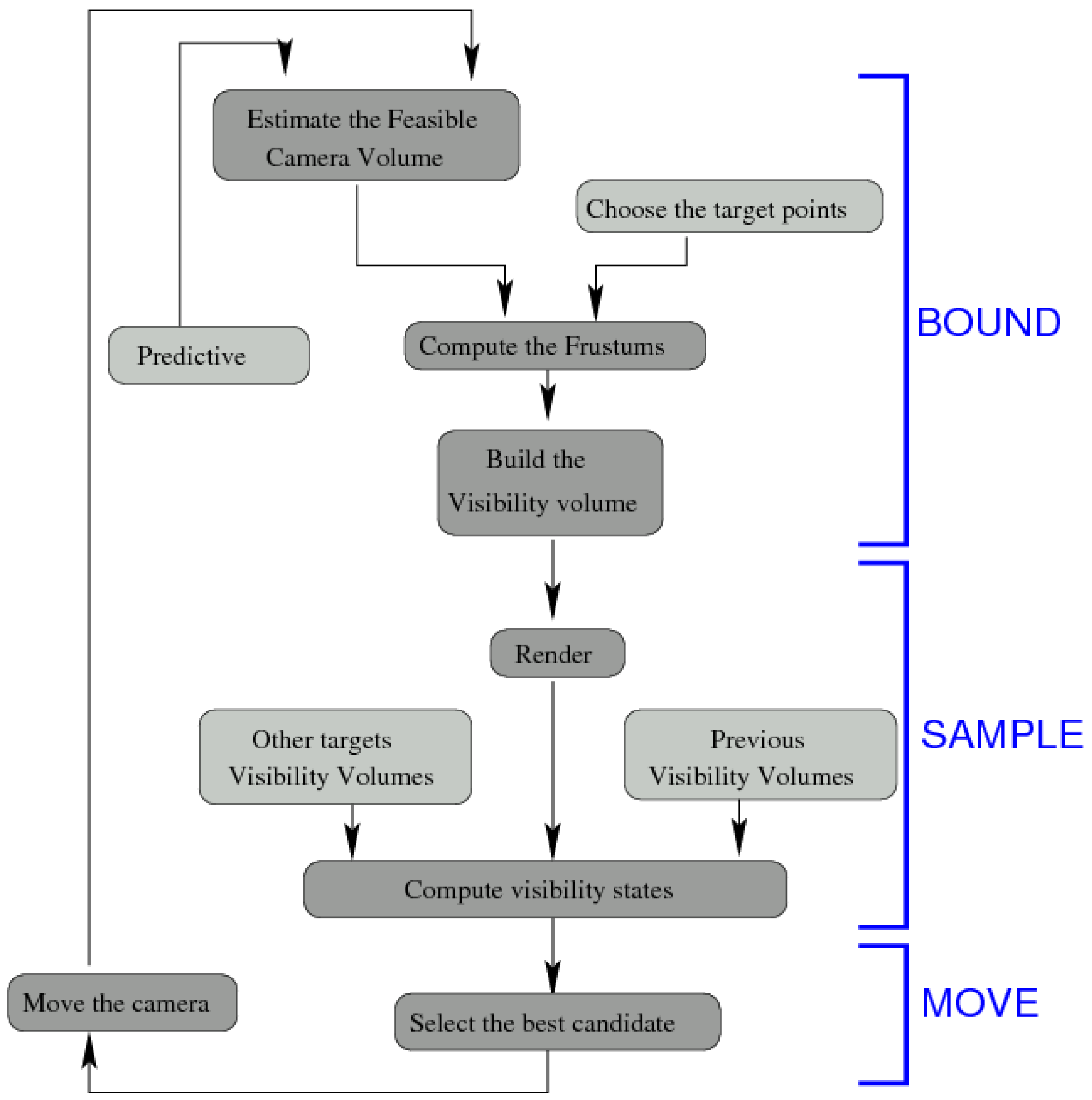

Estimate the Feasible Camera Volume
Choose the target points
Predictive
Compute the Frustums
Build the Visibility volume
BOUND
Render
Other targets Visibility Volumes
Previous Visibility Volumes
SAMPLE
Compute visibility states
Move the camera
Select the best candidate
MOVE

Figure 4: Computation of Occlusion-free Views. The process first bounds the potential region of the next camera locations (i), which is then sampled (ii) and evaluated to compute the next best move (iii).

Our proposal builds upon a frame-buffer computational model [Fournier88] that renders the scene from multiple targets back towards the actual position of the camera, and then reasons over depth buffer values [Philips,Halper03], similar to the principle of shadow maps. The difficulty lies in the efficient and reliable composition of depth information so as to reason about the visibility of multiple targets over the object space.

Our main contribution is the proposal of a method for the efficient evaluation of the visibility of one or more targets for a limited region around the current camera location. Our proposal 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 [Agrawala00].

The overall process comprises four steps (see Figure 4):

- **Bound**: the computation of the predictive camera volume. The predictive camera volume is the set of potential locations for the camera for the current frame and therefore the region of space for which visibility information needs to be computed. This volume is estimated based on the location and velocity of the camera in the previous frame. Where we estimate that all viewpoints within the predictive camera volume are at risk of being occluded (based on past visibility information) we can increase this volume.

- **Sample**: the evaluation of the visibility information for individual targets for a number of viewpoints inside the predictive camera volume. For a single target object we construct a symmetric frustum for which the centre of projection is located on the target object, and which bounds the predictive camera volume (for multiple target objects we construct pairs of asymmetric frustums). By using the frustum(s) to build depth maps we thereby compute the visibility of target object for a number of viewpoints within the predictive camera volume.

- **Aggregate**: the aggregation of visibility information across all target objects and across frames (*i.e.* over time). By aggregating the visibility information for multiple target objects we construct a single visibility representation for the scene. By accumulating visibility information across frames, we can reason about the temporal evolution of the visibility of the target objects.

- **Move**: the use of aggregated visibility information to select the next location for the camera. The choice depends on factors relating to the application, priorities between constraints (on visual properties), and the responsiveness and stability of the camera.

We now describe all four steps in detail.

4.1. 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 $\mathbf{a}_{\mathbf{max}}$ as determined by the application, the current camera position $\mathbf{c}_{\mathbf{t}}$ at time $\mathbf{t}$, and its velocity $\mathbf{v}$, it is possible to compute a volume within which the camera must reside at the next frame. If we assume a maximum acceleration of $\mathbf{a}_{\mathbf{max}}$ in any direction, the boundary can be modeled as a sphere located at $\mathbf{c}_{\mathbf{t}} + \mathbf{dt}.\mathbf{v}_{\mathbf{t}}$ of radius $\mathbf{dt}^{\mathbf{2}}.\mathbf{a}_{\mathbf{max}}$ (with $\mathbf{dt}$ the duration of the integration step). Frustums can then be designed to fully encompass this region.

However, to reduce the locality of our search, and thereby reduce the impact of catastrophic failures (*i.e.* when the whole volume is occluded), the radius of the sphere is multiplied by a factor $\mathbf{n}$ (the predictive factor) which can be dynamically controlled depending on the application. This provides a *predictive camera volume* about which we compute the frustums. Though the movement of the camera at each frame is restricted to the *feasible camera volume,* the *predictive camera volume* allows us to reason on the visibility of the targets over a larger region of space and thereby respond to total occlusion (*e.g.* resulting from the rapid onset of large or very close occluders). The size of the *predictive camera volume* is determined by: (i) the distance of the closest occluder to the target (the closer the occluder is, the further the camera is likely to need to move to find a clear line of sight); and (ii) the distance of the camera to the target objects (the closer the camera is to a target, the more it needs to move).

The sphere representation is then used to build an enclosing frustum (a pyramid which faces are tangent with the sphere), which apex is located where the target stands. The steps are the following:

- Build top (**T**) and bottom (**L**) planes tangent to the sphere and such that points **A** and **B** belong to the planes;
- For each target **A** and **B**, build left and right planes tangent to the sphere and orthogonal to planes (**T**) and (**L**)
- Finally, compute the plane (**F**), tangent to the sphere and orthogonal to plane (**ABO**). This plane represents the far plane of the frustum.

4.2. Sample: rendering from both viewpoints

While the visibility of a single target object can be computed using a standard symmetric frustum, for multiple target objects we propose the construction of non-symmetric frustums for each pair of target object. The intersection of these frustums defines a *visibility volume* in which we can compose the visibility information for a selected set of candidate camera positions (see Figure 8 for a 2D illustration and Figure 6 for a 3D illustration). In order to characterize the visibility of the targets from regions of space (instead of sample points), the renderings are performed using frustums which are not centered around the direction of perspective (see Figure 7). 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. As displayed in Figure 5, 6 and 9, defining a common far plane (**F**) 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).

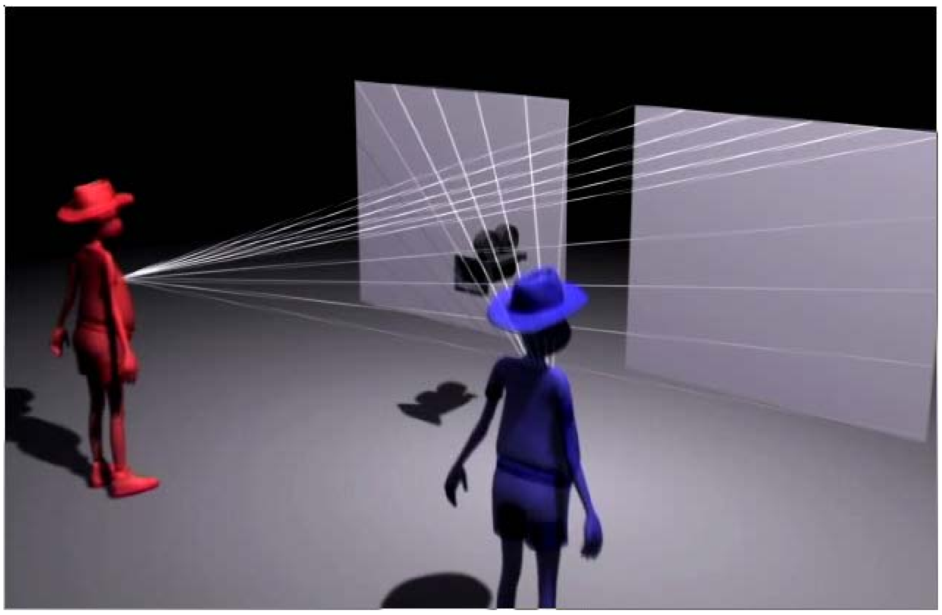


Figure 5: Two renderings are performed from the targets (blue and red characters) towards the camera. The intersection of the frustums used for the rendering provides a region in which visibility will be characterized.

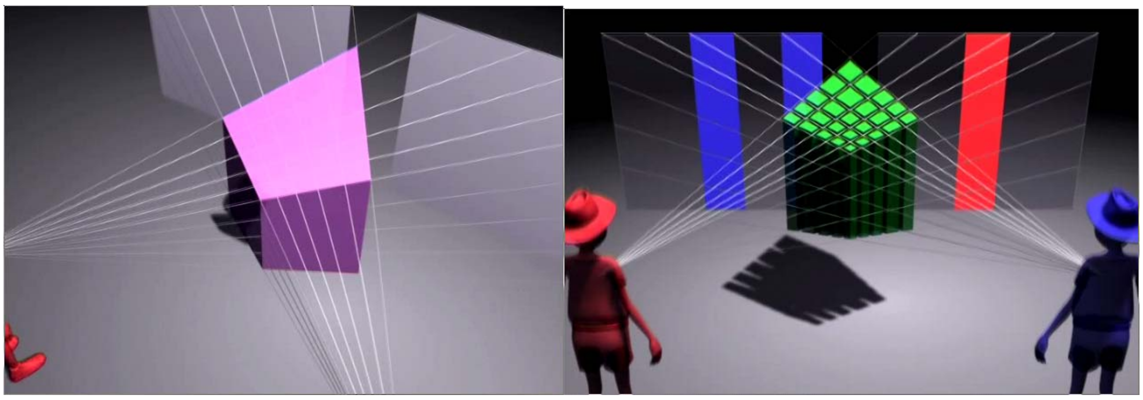


Figure 6: The intersection of asymmetric frustums (left picture) provides a volume in which we estimate the visibility of both targets. The resolution of the renderings which are performed in the frustums determine the granularity of the visibility information inside the volume (right picture).

The geometry of the projection and the resolution of the images determine the granularity of this sampling of the visibility volume. The resolution of the rendering, which defines the precision of the visibility computations, can be dynamically computed given a required mean density parameter **d** inside the *feasible camera volume*. For simplicity, the resolution is set to the same value in both dimensions (***u*** and ***v***) for both buffers. Given a required mean density **d** inside the visibility volume, one can compute the resolution ***n*** of the rendering through the following relation:

$$n = \sqrt[3]{d\frac{4}{3}\pi r^3}$$

where ***r*** represents the radius of the sphere bounding the feasible camera volume.

Furthermore, visibility volumes for successive frames can be combined to stabilize the camera (*i.e.* make the camera resilient to momentary occlusion). Although our approach only reasons over depth information, object indexed color information could also be exploited in distinguishing the nature of the occluders (*e.g.* different colors for static and dynamic objects) which has the potential to inform camera motion strategies in situations where we can model the likely motion of an occluding object.

Practical experiments have actually led us to adopt a multi-level sampling rather than a uniform sampling in figure 11. This multilevel sampling densifies the configurations closer to the center of the visibility volume -- area of most probable movement, and interpolates the pixel values for configurations further from the center. The first area represents the configurations that can be reached, whereas the second represents configurations in which the camera should move to when no solutions are available.

$$\begin{bmatrix} l \\ t \\ 0 \\ 1 \end{bmatrix} = \mathbf{P}.\mathbf{M}_{\mathbf{A}}.\mathbf{o}_1$$

and

$$\begin{bmatrix} r \\ b \\ 0 \\ 1 \end{bmatrix} = \mathbf{P}.\mathbf{M_A}.\mathbf{o}_2$$

where matrix $\mathbf{M_A}$ expresses the change in basis from global coordinates to local camera coordinates in A, and **P** is a classical perspective projection matrix.

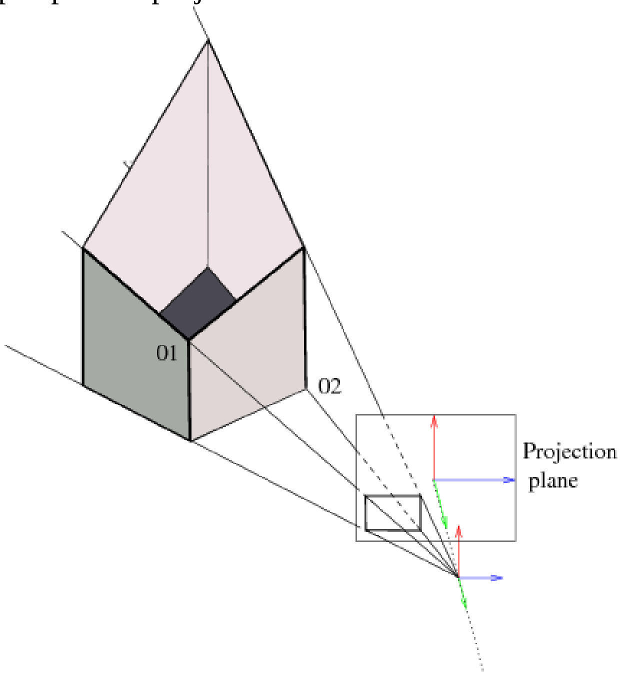


Figure 7: For each target, specific frustum coordinates are computed by projecting two vertices of the visibility volume onto the near projection plane. This leads to an asymmetric frustum with regard to the direction of projection (along z).

Figure 6 shows two viewing frustums, each extending from a target object and encompassing the current camera configuration. The renderings of the scene from both viewpoints yield two low resolution images. By reasoning over the depth-buffer information, the visibility of both targets **A** and **B** can be characterized for a number of regions inside the intersection of the frustums, as for shadow maps.

The detailed case-study is described in Figure 9.

Let the **n**-dimensional vector $\mathbf{z}^{A}=[\ \mathbf{z}^{A1}\ ,\ldots,\ \mathbf{z}^{Ai}\ ,\ \ldots\ ,\ \mathbf{z}^{An}\ \ ]^{T}$ (resp. $\mathbf{z}^{B}$ denote the vector of depths read from the depth-buffer related to target A (resp. B) where **n** is the size of the buffer. Now consider two rays ***i*** and ***j*** originating respectively from targets A and B, leading respectively to rendered pixels $\mathbf{z}^{Ai}$ and $\mathbf{z}^{Bj}$ and that intersect at position I. Each z-buffer value contains the depth of the closest occluder on the ray, denoted by values $\mathbf{z}^{Ai}$ and $\mathbf{z}^{Bj}$. To determine the visibility state at position I, we need to know whether an occluder is located in front or behind point I on each ray. That is, position I is fully visible when the distance from point A to I (which we refer to as $\mathbf{z}^{AI}$) is lower than $\mathbf{z}^{Ai}$, and the distance from point B to I (*i.e.* $\mathbf{z}^{BI}$) is lower than $\mathbf{z}^{Bj}$.

4.3. 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 $\mathbf{z}^A$ and $\mathbf{z}^B$ as:

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

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

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

In the first stamp, for all the points on the ray from point **A** to point **I**, we know that if occlusion occurs (resp. does not occur) for distances greater than or equal to $\mathbf{z}^{Ai}$ (resp. lower than $\mathbf{z}^{Ai}$) where $\mathbf{z}^{Ai}$ denotes the distance from A to the occluder stored in i-th position of the depth buffer.

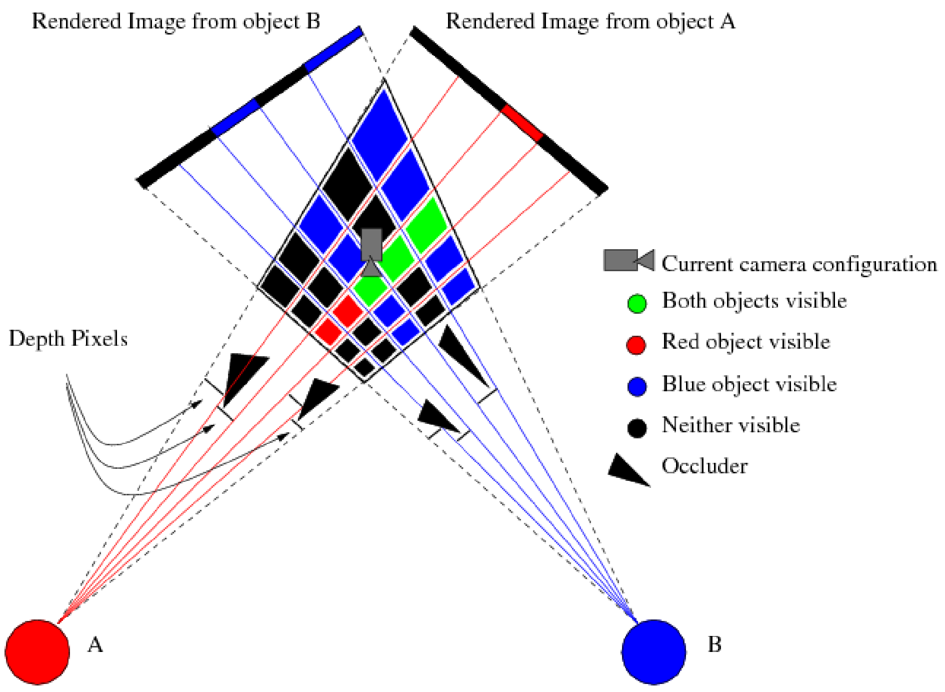


Figure 8: In a 2D case, the intersection of the respective asymmetric frustums yields a bounding volume that encloses the feasible visibility volume; using the two rendered images generated by these frustums, one can compose the visibility status of each area, thereby providing a full characterization of visibility. The visibility depends on the presence and depth of the occluders in each frustum: green areas have both targets A and B visible, black areas have no targets visible, while red or blue areas have one of the two targets visible.

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**.

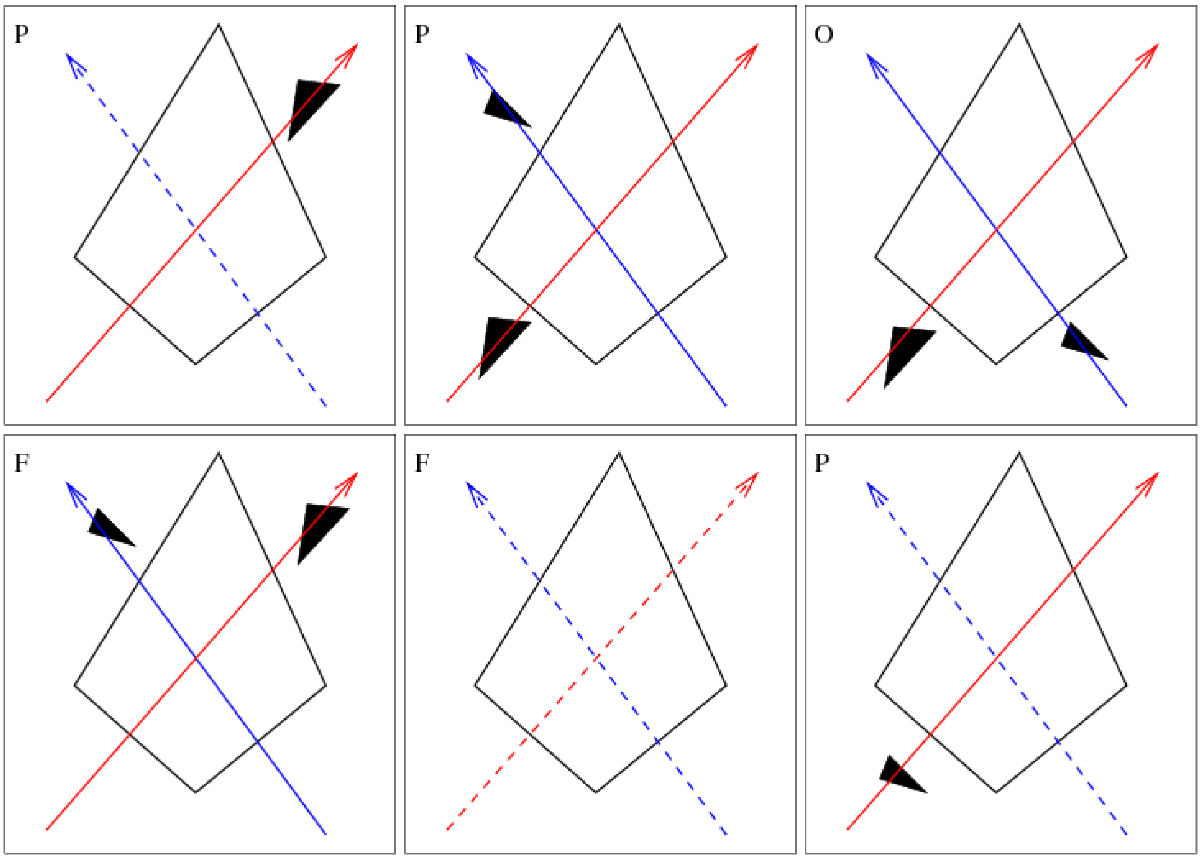


Figure 9: Case study to determine the visibility status of the intersection of two pixel lines on in the frustums. I is the intersection location at which visibility is estimated. Given the spatial arrangement of the occluder (black triangle on the figures) for first and second target object, one can compute the nature of visibility in { P_A, P_B , N, A } representing respectively A is partially visible, B is partially visible, None are visible, and All are visible.

Extending this reasoning to three dimensions requires us to specify the asymmetric frustums such that the corresponding rays for two pixels will in fact intersect (see Figure 7). This approach has a number of desirable qualities: (i) preciseness of our knowledge of visibility at the intersection point; (ii) minimal sampling of the search space; and (iii) efficiency in the representation and computation of visibility. In comparison, approaches that use standard frustums to compose depth information (*e.g.* for the computation of penumbra with multiple light sources) require re-sampling that often results in artefacts.

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 11. Each point **I** inside the visibility volume is represented in local coordinates, I=[u v w]^T, and expressed in global Cartesian coordinates as a linear combination of vectors ***i***, ***j***$_1$, ***j***$_2$, ***k***$_1$, ***k***$_2$, ***k***$_3$, ***k***$_4$ (where ***i***, … ***k***$_4$ are defined in Cartesian coordinates). We refer to $\mathbf{I_{AB}} : \mathbb{R}^3 \to \mathbb{R}^3$ as the trilinear interpolation function related to the visibility volume for target objects A and B, that expresses local coordinates [u v w]^T in Cartesian coordinates I'=[x y z]^T, where:

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

We can now very conveniently use the local coordinates $[u\ v\ w]^T$ 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 $[u\ v\ w]^T$ 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 $(I_{AB})^{-1}$ to obtain the local coordinates of any point inside the visibility volume, and thus establish its visibility status (see Section 5.2).

Thus, in a very convenient way, the intersection of a line going from point A to a pixel A_{uv} on A's rendering plane, with a line going from point B to a pixel B_{wv} on B rendering plane is located at a point **I** which local coordinates in the visibility volume are defined by $[u\ v\ (1\text{-}w)]^T$. We can then directly derive the 3D location of the intersection point $(M.\ [u\ v\ (1\text{-}w)]^T)$.

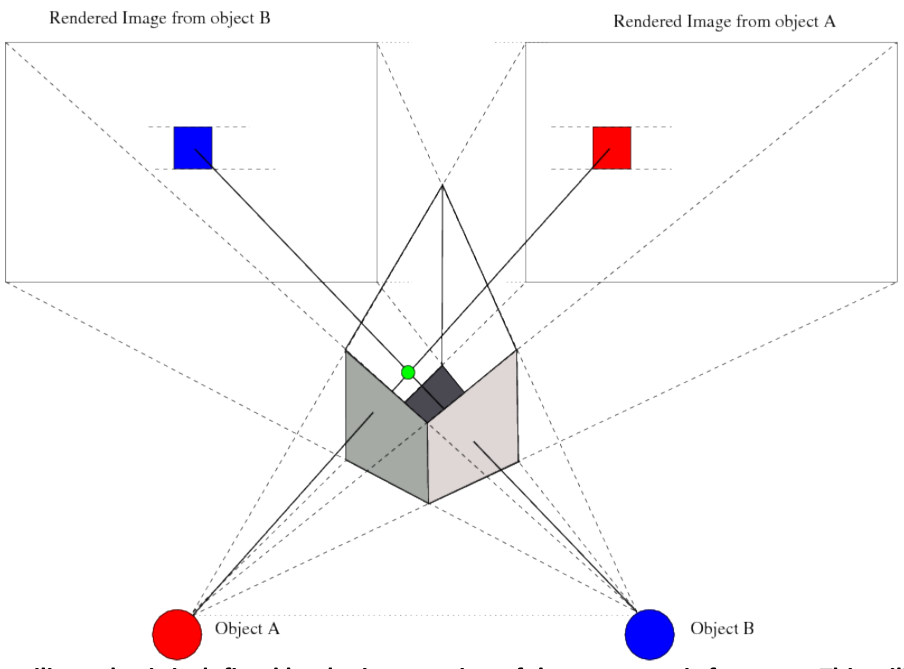


Figure 10: In 3D, a trilinear basis is defined by the intersection of the asymmetric frustums. This trilinear basis easily addresses the pixels coordinates in the rendered frame, and provides the exact location of the intersection point in the visibility volume (with intersecting rays).

We can now extend the composition operator $\bigotimes$ from vectors to matrices. Let matrix **A** contain the depth information after rendering (z^{Aij} represents the depth information at coordinates (i,j) in the image).

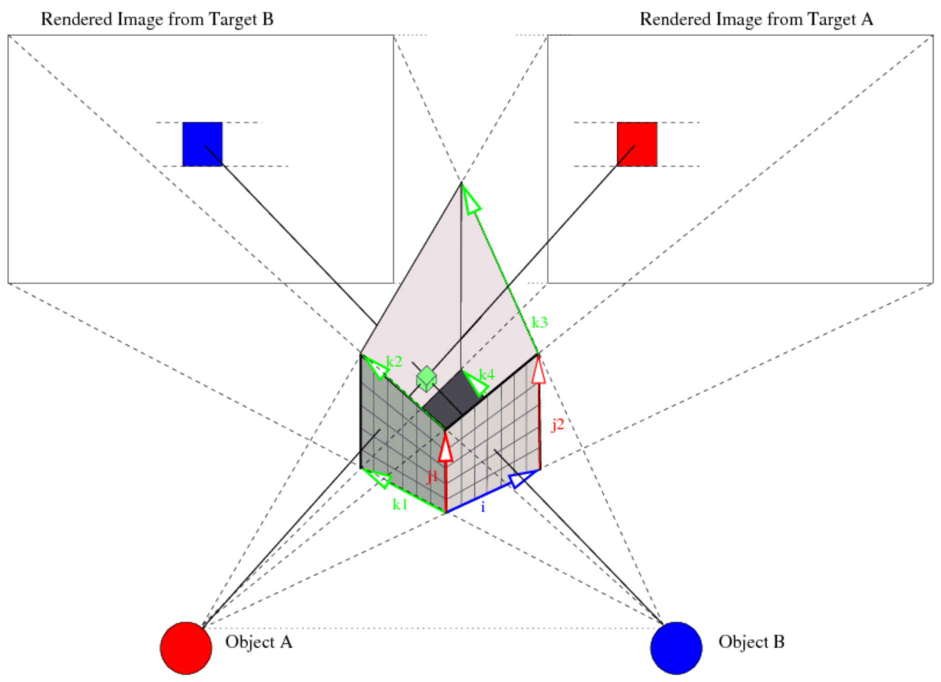
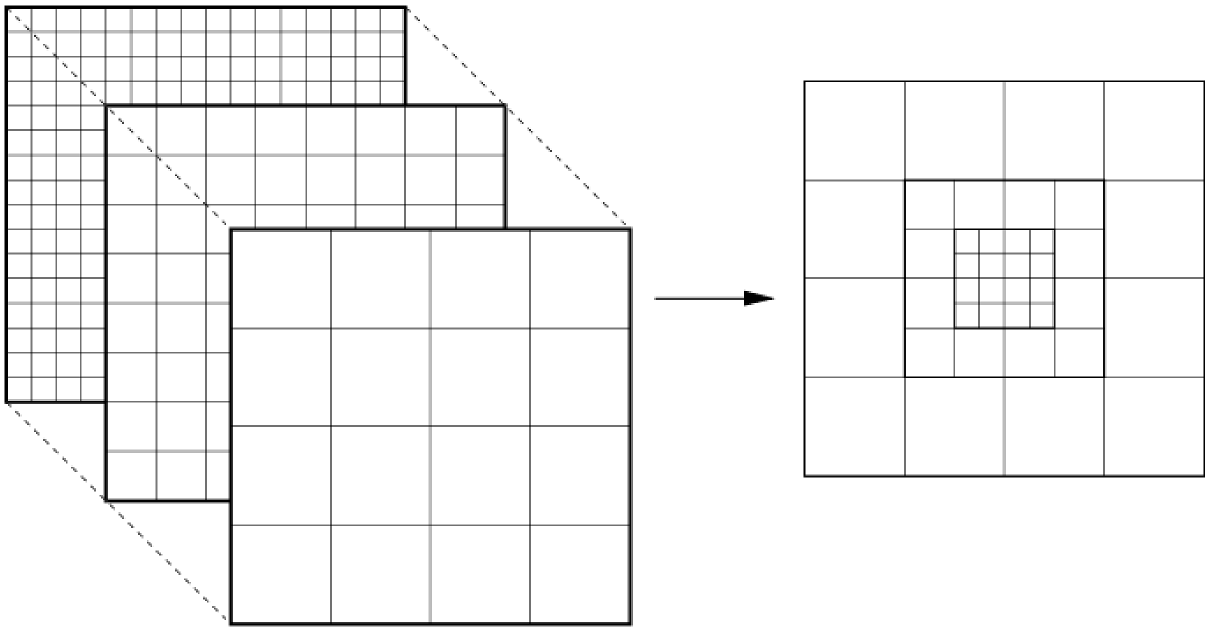


Figure 11: Intersection of the frustums in 3D: the trilinear coordinate system composed of vectors i, j_1, j_2, k_1, k_2, k_3, k_4 enables us to address the samples in an efficient manner. The local coordinates of a point (u,v,w) directly address the rendered images at coordinates (u,v) and (1-w,v).

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 4.4). 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.



(a) Hierarchical Shadow Map (b) Multi−level Density Map

Figure 12: In using a hierarchical density map rather than a regular sampling representation, our process provides a very fine resolution at the center of the rendered image, and a rough resolution on the borders (see right image). This represents an ideal balance between precise visibility information close to the expected camera location and rough visiblity information on the borders, at a reduced cost.

By using visibility volumes, our approach can be seamlessly extended to more than two target objects. Considering three targets A, B and C we can independently compute the visibility volumes V_{AB} and V_{BC} for pairs {A,B} and {B,C} (see Figure 13). To avoid two renderings from target B, we perform a resampling of the first rendering: and since both volumes enclose the predictive camera volume, the resampling has little impact on the quality of the shadow map. Combining the visibility information for the visibility volumes corresponding to each pair is performed in a manner analogous to the temporal accumulation of visibility - although here we are considering volumes that arise from different target object pairs. Each configuration inside V_{AB} is expressed in the local coordinates of V_{BC}, and both visibility states are merged using the operator $\odot$ defined as:

$\odot$	N	P_A	P_B	V
N	N	P_A	P_B	P_{AB}
P_B	P_B	P_{AB}	P_B	P_{AB}
P_C	P_C	P_{AC}	P_{BC}	P_{AC}
V	P_{BC}	P_{ABC}	P_{BC}	P_{ABC}

where, as before, P_A means that only target A is visible, and P_{ABC} that all targets A,B and C are visible. Although in most cases, we are only interested in the configurations that are visible for all the objects (state V), the storage of the individual visibility states (rather than a value for the number of visible objects) enables a finer granularity of reasoning when no fully visible configuration exists (*e.g.* by enforcing coherency in maintaining the same object occluded instead of continuously jumping from one to another). This accumulation operator can be seamlessly applied to both simple and temporal visibility volumes.

A side effect is that rendering from object B must be performed for both couples, as the projection planes are distinct; to avoid this double rendering process we simply propose to express the first rendering within the basis of the second: each point of V_{AB} is expressed in basis V_{BC} and conversely. For each point, we get the characterization of the visibility of points A, B and C.

Figure 13 illustrates the accumulation principle in 2D and for three objects.

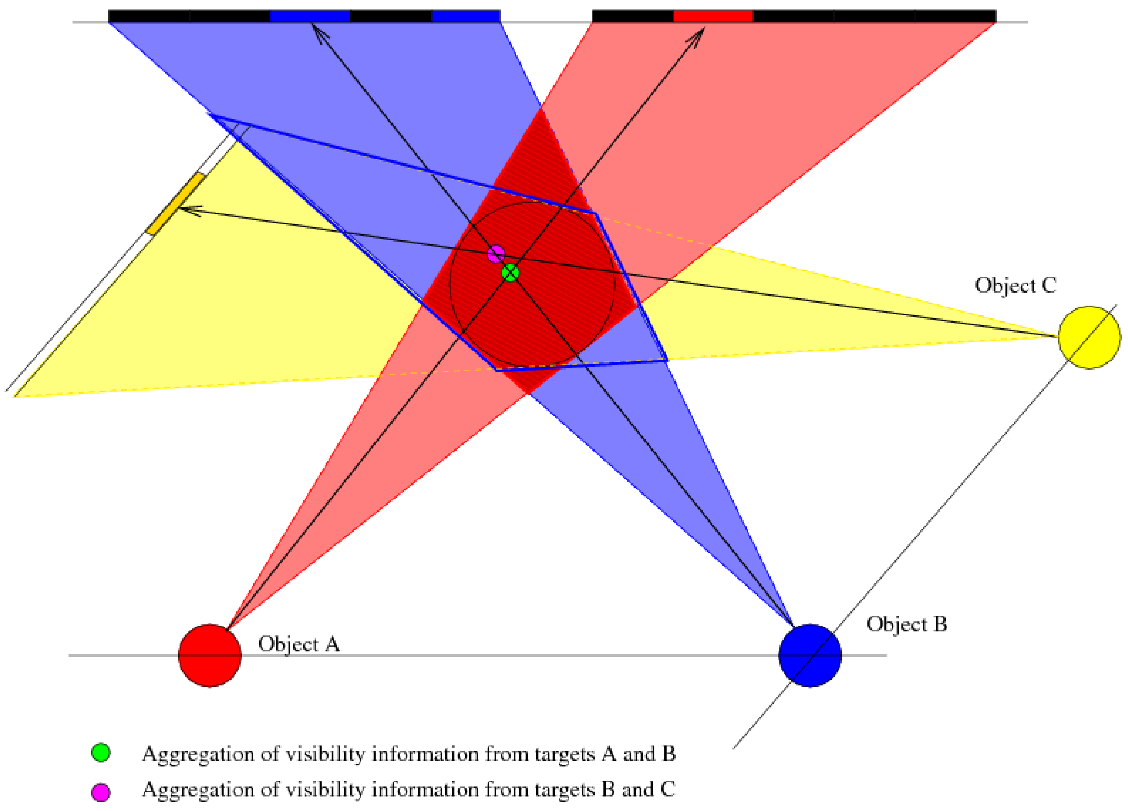

Object C
Object A
Object B
Aggregation of visibility information from targets A and B
Aggregation of visibility information from targets B and C

Figure 13: 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 re-sampled to be used by both visibility volumes (rather than performing two renderings from B).

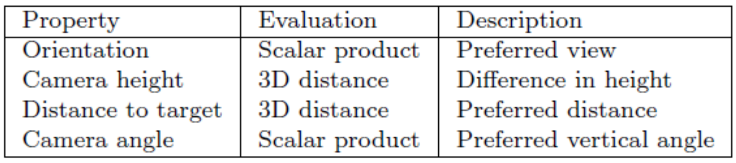
Furthermore, in balancing the ratio between the precision of visibility computation and the computational cost, we use a high density of sampling in the feasible camera volume, and progressively under-sample the shadow maps thereby reducing the density inside the remainder of the predictive camera volume. We perform this operation in a manner analogous to mipmapping technique, using a hierarchical shadow map (HSM), but replacing the depth average with the minimum depth value for a region.

Clearly, due to the non symmetric frustums the density of sampling varies within the sphere. The precise nature of the distribution strongly depends on the shape of the volume (*i.e.* the configuration of the camera and the two target objects) with lower densities while moving away from the targets.

4.4. Move: choosing the next camera position

Choosing the next camera position within the *predictive camera volume* is an aesthetic choice for the author of an application. Our task is therefore to provide tools that will afford content authors sufficient expressive power to achieve their aesthetic and communicative goals. One low-level aspect of such tools include 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: S_N (where both A and B are not visible), S_{PA} (only A is visible), S_{PB} (only B is visible), and 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 S_V), generally at the cost of jumpiness and sudden 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 (*e.g.* three quarter view), camera heights (*e.g.* low-angles, high-angles), camera shots (*e.g.* sizes of objects on the screen). All these properties can be evaluated in a straightforward manner (see Table 1). The cost of this evaluation is linear in the number of samples and has a minimal impact on performance. In this way, stylistic choices can be cast as a problem of setting priorities between the different sets of properties. In Halper *etal.* [Halper03], occlusion was considered as a weak constraint and other properties were prioritized according to a total ordering (*i.e.* height, viewing angle, distance, look at).



Property	Evaluation	Description
Orientation	Scalar product	Preferred view
Camera height	3D distance	Difference in height
Distance to target	3D distance	Preferred distance
Camera angle	Scalar product	Preferred vertical angle

Table 1: Extending the expressiveness of our camera control system: each sample inside the visibility volume can be evaluated against different composition and cinematographic properties; the information is then composed with the visibility states when deciding the next best camera location.

Finally, the predictive power of the camera is achieved by selecting locations in S_V for which promixal neighboring regions are also unoccluded. The manner in which we have specified our frustums allows the neighbors of the current configuration to be accessed using simple table indices. Whenever S_V is empty, different strategies can be applied over sets S_{PA} and S_{PB}, for example, prioritizing aspects of coherency (*e.g.* maintaining the same object unoccluded rather than alternating between configurations in S_{PA} and in S_{PA}).

The motion of the camera is controlled by a simple mass/viscosity model: a force is applied to the current camera location to drive it towards the chosen location **p** in the visibility volume. A new location is given by:

$$\mathbf{c}_{t+\Delta t} = \mathbf{c}_t + \Delta t \mathbf{v}_t + \Delta t^2 (p - \mathbf{c}_t - v\mathbf{v}_t)/m$$

where ***m*** represents the mass of the camera and v a viscosity parameter that resists the motion (in an opposite direction to the velocity as denoted by $v\mathbf{v}_t$). When p is outside of the feasible camera volume, the location is given by:

$$\mathbf{c}_{t+\Delta t} = \mathbf{c}_t + \Delta t \mathbf{v}_t + \Delta t^2 a_{max} \left(\frac{p - \mathbf{c}_t}{|(p - \mathbf{c}_t)|} - v\mathbf{v}_t \right)/m$$

where the attraction force is normalized and restricted to the maximum acceleration $a_{\mathbf{max}}$□.

The sequence of evaluation of the factors used in selecting the next camera position has an impact on the efficiency of this selection process. In our evaluation (Section 5.4) we first select all visible configurations (or partially visible configurations where there are no visible configurations), filter those which do not have the specified cinematographic properties, and finally we apply responsiveness, predictive power and temporal coherence criteria. Where there are no visible configurations we apply an escape strategy, in practice expanding the search space, and repeat the bound, sample and aggregate processes.

A final consideration is the problem of physical barriers to the camera's motion. Where a physical obstacle is located inside the potential camera volume, the possibility exists that a path from the current to the next camera location passes through the obstacle. Checking for such collisions is a common problem is applications such a computer games and interactive environments and a range of standard methods can be used to evaluate and reject such paths.

In fine, the choice of where the camera should move to is guided by the application together with the nature environment (higly complex and dynamic) and elements of directorial style. In our implementation we use a heuristic that both incorporates a user specified dynamics for the camera, and maintains coherency with respect to partial occlusion. That is if S_V is not empty, we select the position which requires the smallest change in the velocity of the camera (*i.e.* avoiding sudden jumps). Whenever S_V is empty, we choose a position in either S_{PA} or S_{PB} depending on the visibility of targets A or B in previous frames. When all the configurations are occluded for both targets, we either perform an escape strategy or rely on the Director Volume approach to perform an appropriate cut.

5. Improving the Visibility Volume technique

We now detail a number of extensions based on the Visibility volume technique to handle the full visibility of targets, short-lived occluders and more than two targets.

5.1. Stochastic Model of Visual Extent

At this point our proposal addresses the problem of multi-object occlusions, but we still depend on a point-based abstraction of a target object, and use the visibility of these points as the basis on which to decide where to move the camera. In computer game environments, a commonly employed technique is to select a number of points on the surface of the target object when computing its visibility [Hawkins]. Although the distribution of these points on the surface helps to capture the true visual extent of an object (*e.g.* for a character choosing extremities such as the head, hands and feet) and allows us to prioritize distinctive parts of an object (*e.g.* faces, torsos and guns are salient in a gaming context), such hand crafted sampling schemes inadequately characterize differences in the visual extent of objects when they are viewed from different angles and distances.

We propose a tessellation independent stochastic approach that automates the choice of target points on the surface of an object for different viewpoints and distances. Furthermore, we provide an approach to compute the from-region visibility of areas around each viewpoint that can efficiently approximate the visual extent of the target. Rendering from a large number of viewpoints on the surface of an object would be prohibitively expensive if performed for each frame. Instead, 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 offline distribute camera configurations evenly around the object and perform basic renderings (*i.e.* without lights and using a unique color for the object). 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. Likewise, the relative importance of parts of the object (*e.g.* faces or guns for characters) can be rendered with specific colors and higher sampling densities can be employed. Deformable objects can be further indexed by key-frame with the size of sample table growing in proportion to the number of key-frames. Figure 14 illustrates the sampling process for a 3D character. 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). By considering an average of 20 points per view per object, in a real-time environment averaging 60 frames per second, we utilize the full set of points approximating the visual extent of object 3 times every second.

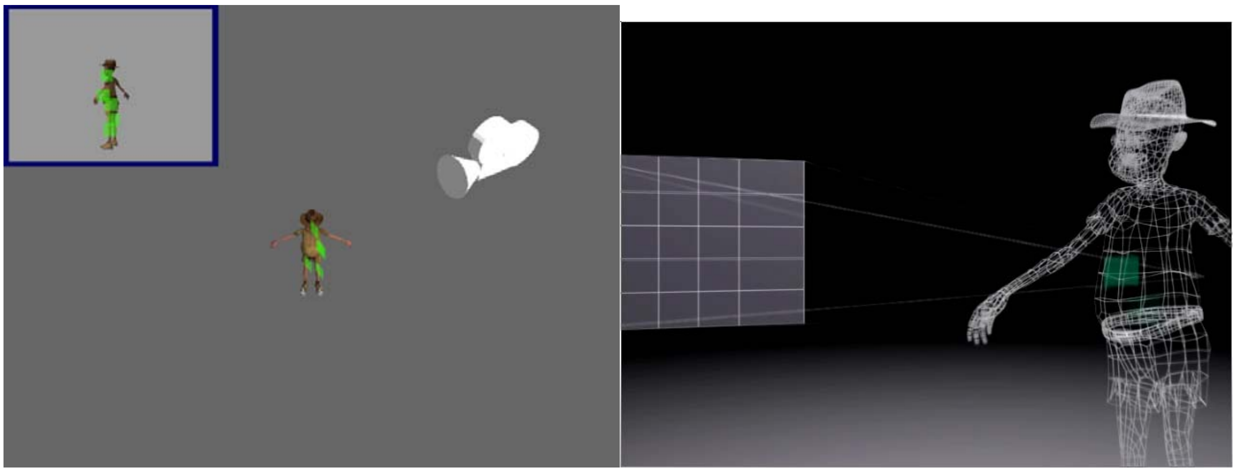


Figure 14 : Stochastic estimation of the visual extents from distinct viewing angles and distances. For each viewpoint, a number of 2D points are randomly chosen on the surface of the projected 3D shape. As the density of the sampling is constant in the 2D space, the size of the projected extent influences the number of sample points on its surface. Sample points are then extended to from-region visibility by considering a fixed size rectangular abstraction of the surface of the target.

In order to perform from-region visibility (rather than from-point visibility) for each viewpoint on the surface of the target object, we compute the shadow due to each geometerized pixel in the shadow map given a rectangular light source around the viewpoint (see Figure 15 and 16) on the surface of the target. The depth map is then updated using all per-pixel shadow volumes. Following the same composition process performed with from-point depth maps, we use the updated depth maps to characterize the cell's visibility.

From-region visibility computation is performed in three steps:

- Compute the extremal stabbing lines for each depth-pixel in the shadow map in relation to the approximated surface of the target object, which in turn defines the boundaries of the per-pixel shadow volume. For efficiency, the surface of the target is locally approximated by a rectangular area parallel to the far plane (and to the depth-pixel plane) whose borders are aligned with the frustum borders.
- Intersect the stabbing lines with the far plane of the frustum thereby providing the area for which pixel depths need to be recomputed.
- Update the depth of each of these pixels by computing a simple ray-to-plane intersection.

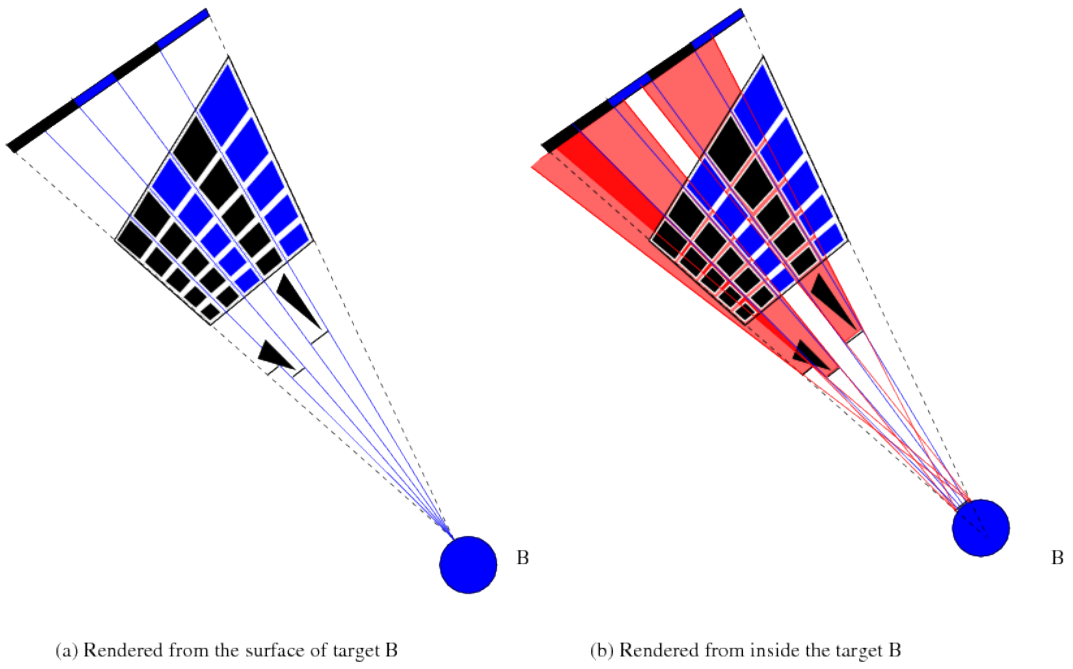

B
B
(a) Rendered from the surface of target B
(b) Rendered from inside the target B

Figure 15: Computation of the from-region visibility by updating the shadow map. For each depth-pixel: (a) a shadow pyramid is projected onto the far plane; and (b) all depth-pixels inside the shadow are updated by computing the ray-to-plane intersection.

We can reduce the average complexity of this process by first processing the depth-pixels that have not been updated by a from-region shadow computation (see Figure 16), and second, by starting the depth-pixel shadow computation using those pixels with the lowest depth values (occluders closer to the target cast larger shadows). An advantage of this approach is that the from-region visibility is not evaluated for every area in the visibility volume, but only for the individual depth maps. The computational cost is O($\boldsymbol{mr^2}$) in the worst case (where $\boldsymbol{m}$ is the number of depth-pixels, and $\boldsymbol{r}$ the resolution of the shadow map) as compared to a O($\boldsymbol{mr^3}$) cost when computed for the entire visibility volume.

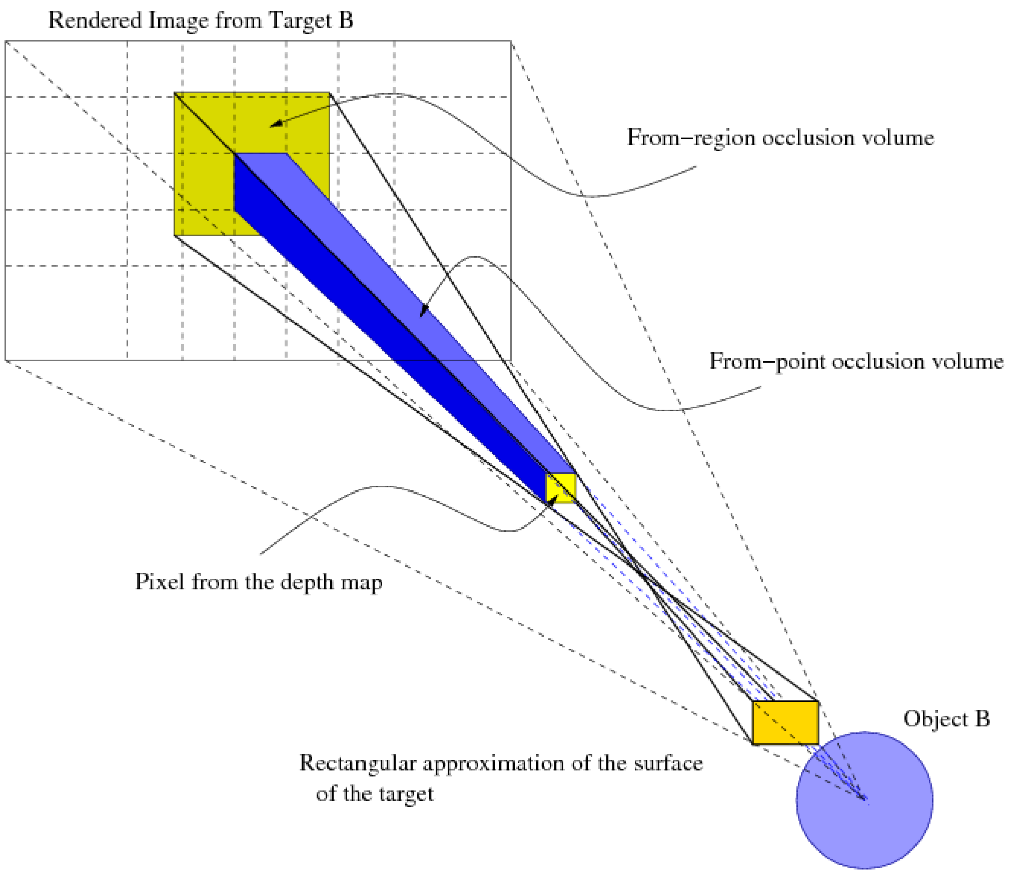

Rendered Image from Target B
From–region occlusion volume
From–point occlusion volume
Pixel from the depth map
Object B
Rectangular approximation of the surface of the target

Figure 16 : From-region visibility computation in 3D: given that the depth pixel plane, the far plane, and the approximation of the surface of the target are parallel, we can compute the from-region occlusion area with stabbing lines and only update the pixels in the shadowed area.

5.2. 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 the 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 ***n*** visibility states {$\boldsymbol{v}^{t}$, … , $\boldsymbol{v}^{t-i}$, … $\boldsymbol{v}^{t-n}$} retains a visibility state ***v*** such that:

$$\frac{X_v}{n} \geq \alpha$$

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).

Now, in order to aggregate a set of visibility volumes { V^t, …, V^{t-n} }, and given that these volumes evolve over time, we select each camera configuration c^t in V^t, express it in the basis of each visibility volume V^{t-i}, and apply the operator $\bigoplus_{\alpha}$ over the extracted visibility states. The expression of a camera position $c^t=[u\ v\ w]^T$ (defined by its local coordinates in the visibility volume V^t) in a previous visibility volume V^{t-i} requires us to: (i) express the configuration c^t in global Cartesian coordinates by applying the trilinear interpolation $\mathbf{I}^t{}_{AB}(c^t)$, and (ii) assess the visibility status of $\mathbf{I}^t{}_{AB}(c^t)$ in V^{t-i} by computing its local coordinates. That is:

$$[u'\ v'\ w'] = \llbracket(\mathbf{I}^{t-i}\rrbracket_{AB})^{-1}\llbracket(\mathbf{I}\rrbracket^{t}{}_{AB}(c^{t}))$$

The computation of the inverse of the trilinear interpolation $\llbracket(\mathbf{I}^{t-i}\rrbracket_{AB})^{-1}$ requires us to compute the solution of a cubic polynomial. Though the roots can be computed algebraically with Cardano's method, our specification of the frustums allows us to compute the local coordinates with ray-to-plane intersections and 2D local basis changes. For this, we build the intersection between the ray going through $\llbracket\ \llbracket(\mathbf{I}\rrbracket\ {}^{\uparrow}t\rrbracket\ {}_{\downarrow}AB\ (c^{\uparrow}t)$ and rendering planes for A (a similar process is performed for rendering plane B). The coordinates of the intersection points are then remapped to pixel coordinates (u_A,v_A) and (u_B,v_B) on the rendering planes, which in turn allows us to access the visibility state in 3D local coordinates $[u_B, v_B, -u_A]^T$ of c^t inside the visibility volume $V^{t-i}{}_{AB}$.

The cost of computing previous visibility states (cost of two ray-to-plane intersections per configuration) is not significant when compared to an alternative approach based on unrelated shadow maps. 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. The *temporal visibility volume* (see Figure 17) in which we accumulate the visibility states of past frames is seamlessly coupled with the estimation of visual extent to provide meaningful control of the desired degree of visibility of a target. Choosing the appropriate coupling between the window size and the density of surface sampling is a matter of both performance and desired confidence in the visibility. Furthermore, the temporal visibility volume allows us to express the requirement of partial occlusion over the target objects.

Experimental results demonstrate the improved temporal stability situations that involve partial occlusion and encounters with short-lived occluders (see Section 5.4). 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.

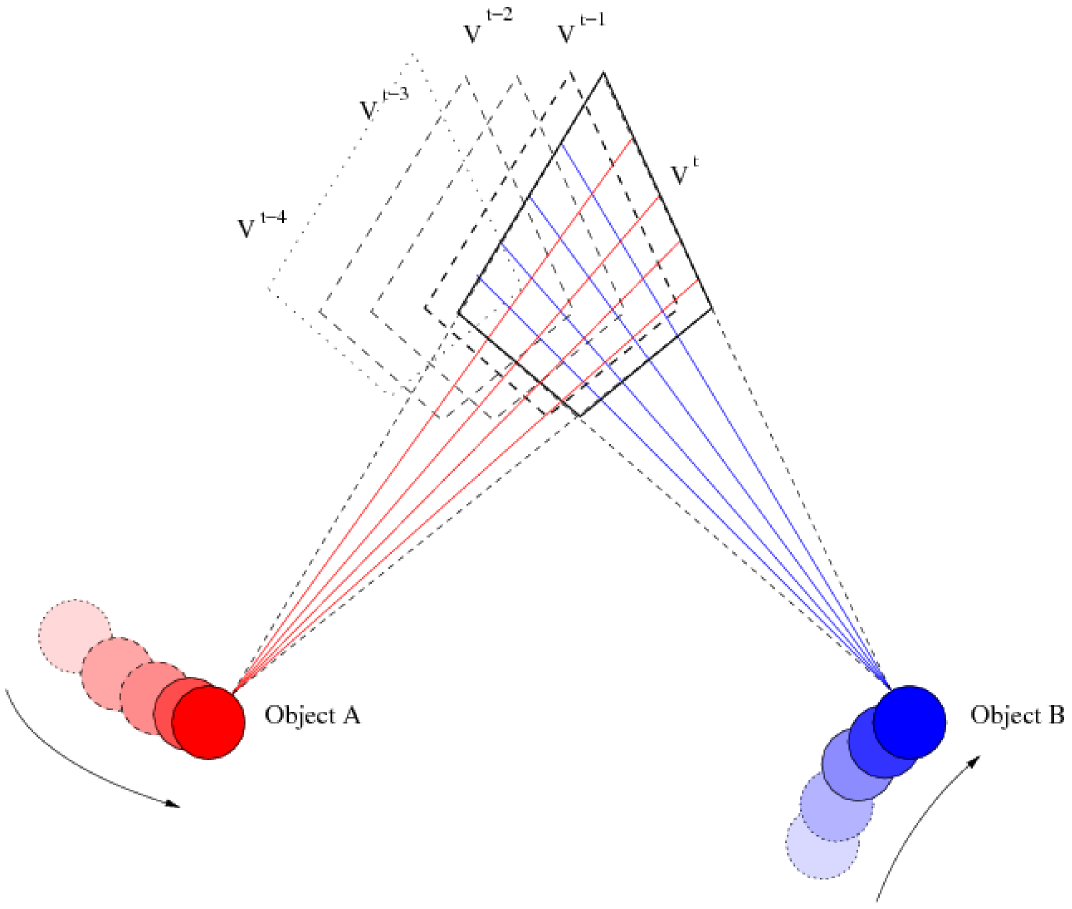


Figure 17: Stability is improved by aggregating the visibility information over the previous visibility volumes. The degree to which the camera reacts to occlusion is controlled by a ratio between the number of occluded states and the number of past frames accumulated

5.3. Multi-object visibility aggregation

By using visibility volumes, our approach can be seamlessly extended to more than two target objects. Considering three targets **A**, **B** and **C** we can independently compute the visibility volumes V_{AB} and V_{BC} for pairs {A,B} and {B,C} (see Figure 13). To avoid two renderings from target **B**, we perform a resampling of the first rendering: and since both volumes enclose the predictive camera volume, the resampling has little impact on the quality of the shadow map. Combining the visibility information for the visibility volumes corresponding to each pair is performed in a manner analogous to the temporal accumulation of visibility – although here we are considering volumes that arise from different target object pairs. Each configuration inside V_{AB} is expressed in the local coordinates of V_{BC}, and both visibility states are merged using the operator $\odot$ defined as:

$\odot$	N	P_A	P_B	V
N	N	P_A	P_B	P_{AB}
P_B	P_B	P_{AB}	P_B	P_{AB}
P_C	P_C	P_{AC}	P_{BC}	P_{AC}
V	P_{BC}	P_{ABC}	P_{BC}	P_{ABC}

where, as before, P_A means that only target A is visible, and $P_{ABC}$$ that all targets A,B and C are visible. Although in most cases, we are only interested in the configurations that are visible for all the objects (state V), the storage of the individual visibility states (rather than a value for the number of visible objects) enables a finer granularity of reasoning when no fully visible configuration exists (*e.g.* by enforcing coherency in maintaining the same object occluded instead of continuously jumping from one to another). This accumulation operator can be seamlessly applied to both simple and temporal visibility volumes.

The rendering from object B must be performed for both couples, as the projection planes are distinct; a way to avoid this double rendering process is to express the first rendering within the basis of the second. Though the frustums are distinct, the sampling is only performed in the sphere, therefore the process is conservative.

We express each point of Visibility Volume {A,B} into the basis of Visibility Volume {B, C}. For each point, we obtain the characterization of the occlusion of points A, B and C.

5.4. Evaluation and Discussion

In contexts with highly cluttered environments or with very large occluders, there may be no satisfactory configuration for the camera within its dynamic limits. This can be robustly handled by again considering the temporal evolution of the target object visibility and when necessary expanding the scope of the search for promising directions in which to move. Whenever the occlusion reaches a given threshold, the *feasible camera volume* is expanded in space without impacting the size of the renderings, and sampled for free configurations. As a result, a number samples are no more located within the dynamic limits of the camera, and the configuration to which to move to is set as target location for the camera which in turn is positioned the closest possible to this target. As the scene evolves, the target location is recomputed following the same principle. The size of the feasible camera volume is however limited by the distance between the targets and the current camera position above which the visibility volume is impossible to compute.

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. Illustrating camera behavior using static media is problematic, so for each experiment we have simply presented a global view of the environment, and a camera path computed by our system. 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. The implementation of the prototype is based on OpenGL and has been integrated into the Ogre3D engine to allow its evaluation on realistic and moderately complex scenes.

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).

Processing times have been extracted using the **gprof** profiling tool. The costs related to the different steps of the visibility computation process, for a number of different resolutions, are provided in Table 2. 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 2). 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 (7x7), we can evaluate more than 300 possible camera configurations (and select the best one) in less than 3 ms. 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.

r	Samples	Rendering	Intersecting	Selection	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

Table 2 : Performance (ms) of the visibility computation for different resolutions r: component times are reported for the three main steps of this process: (i) hardware rendering; (ii) intersection calculation; and (iii) next configuration selection.

Predictive power

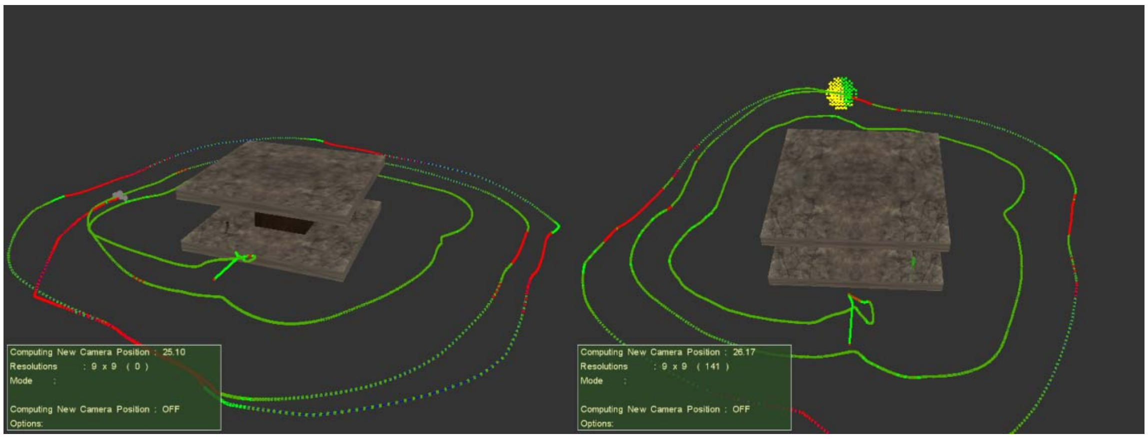
Our occlusion-aware technique detects and tries to maintain visibility within a camera's dynamic limits. Indeed, in both computer graphics applications and real-life camera motion, it is highly desirable to avoid sudden occlusion by using early anticipation. A partial solution simply involves making 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.

Simple movement prediction models for the camera, target objects and occluders can be readily integrated as proposed in [Halper03]. Although in general terms movements are unpredictable for large time steps, different generic approaches [Zhu90] or object-dependent approaches [Vasquez07] that rely on statistical techniques, provide high quality estimates of future positions. 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 3 shows the results of our evaluation of ***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 x 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. See Figure 18.

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

Table 3 : Comparison of predictive occlusion models; we measured the visibility of target objects for two models of motion: *Basic* and *RobustMove* (which selects a configuration within the largest collection of free configurations). We also evaluated each of these in combination with a prediction strategy as to the motion of the targets (*Prediction*). r_N is the percentage of fully occluded frames, r_P is the percentage of partially occluded frames.



```
Computing New Camera Position : 25.10
Resolutions       : 9 x 9  ( 0 )
Mode      :

Computing New Camera Position : OFF
Options:
```

```
Computing New Camera Position : 26.17
Resolutions       : 9 x 9  ( 141 )
Mode      :

Computing New Camera Position : OFF
Options:
```

Figure 18 : Comparison between camera paths tracking two walking targets around a central pillar with roof and floor. The color of the path represents the degree of occlusion (green is unoccluded, and red is occluded). On the

left figure (without prediction technique), there are many path segments for which the targets are occluded. On the right rigure (with our prediction technique) the amount of occlusion drops down significantly. However, occlusion still appears by moments, due to the sharp turns performed by the actors around the central pillar.

Responsiveness

In contrast to the predictive model which attempts to anticipate occlusion, responsiveness refers to a camera's ability to react in a timely fashion to an occlusion. Responsiveness can be measured both in terms of the number of frames necessary to bring the camera from a fully occluded view to a fully unoccluded view, and the quality of the camera's motion. We evaluated quality by considering the mean variation of successive accelerations (which we used as a proxy for stability).

For this experiment, we created occluders of quite different natures which appear suddenly in front of one, two or all targets, for a range of distances to the targets, and we evaluated the responsiveness for different values of the *responsiveness factor* which controls the size of the search space around the current camera location. The maximum allowable acceleration was the same in all cases. The responsiveness factor $\boldsymbol{f}$ is governed by the ratio between the distances of the camera $\boldsymbol{d_c}$ and the closest occluder $\boldsymbol{d_o}$ to the target, such that: $\boldsymbol{f}=\mathbf{c_r d_c/d_o}$ where c_r is a constant. Table 4 shows the resulting performance for different combinations of strategies, in terms of the time to find an unoccluded view (t), and the mean acceleration variation $\boldsymbol{M_a}$. We also considered the impact of accumulation and our results demonstrate that responsiveness improves with the size of the search space, though for the higher values the camera undertakes many large and needless movements.

c_r	Accum	$t(ms)$	M_a
1.0	no	1.61	3.41
1.0	yes (3)	1.89	3.29
3.0	no	0.38	2.11
3.0	yes (5)	0.51	1.89
5.0	no	0.21	2.48
5.0	yes (5)	0.43	2.17

Table 4 : Evaluation of the responsiveness in the camera behavior for the Temple benchmark. Variation in responsive factor and accumulation.

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 [Halper03] (although Halper *etal.* refer to this ability as **frame coherency**). We enforce coherence by selecting cells that offer minimal change in visibility, camera orientation and the distance to the target, from within the possible candidates. In evaluating camera coherence we measured the mean deviation of property satisfaction together with the sum of the camera accelerations over a time interval.

Evaluation of temporal coherence

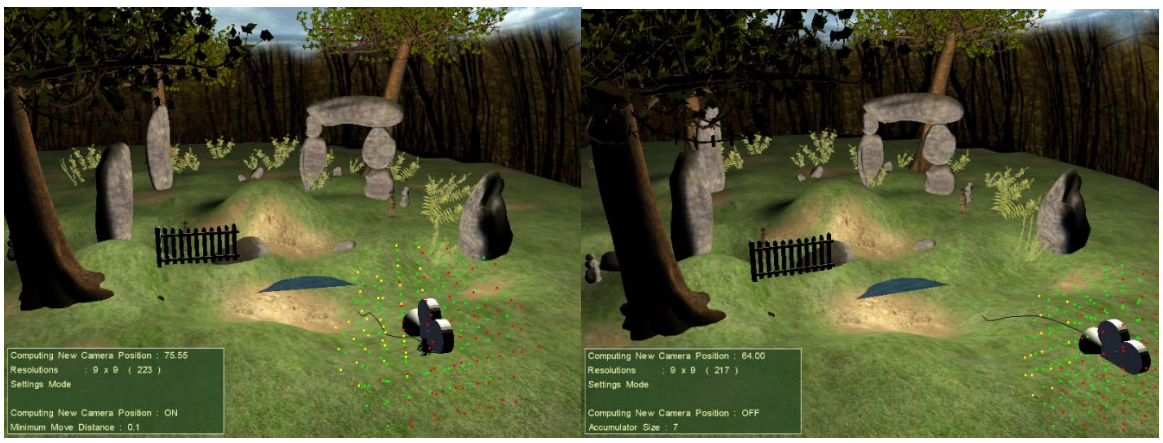
Temporal stability measures the responsiveness of the camera to sparse and/or fast moving occluders. In our experiment we used two target objects (characters) and a sparse occluder (fence) that moves back and forth in front of the targets for 10 seconds (see Figure 19). The left frame

shows the camera motion (with white traces) without any temporal stability (*i.e.* the size of the sliding window is set to zero), and the right frame shows the camera motion for a sliding window of size 10 and a threshold value of ***a***=6 (meaning that a configuration is considered occluded if it is occluded at least 6 times in the past 10 frames). Both targets were abstracted as points (with no representation of the visual extent). The figure on the left shows the large changes in the path that occur as an occluder hides a target object. The figure on the right shows the significant improvement in the maintenance of a stable position despite the temporary partial occlusion.
The cost of computing the accumulation over the 10 past frames is not significant due to the fact that we store previous accumulation state (rather than recomputing them).

Table 5 shows the quantitative results. We evaluated the impact of our stability heuristic by measuring both the camera motion (d_c stands for the total distance covered by the camera), and the motion likely to be observed by the viewer, using d_s the distance covered in 2D by the projected centers of the target objects. The cost due to the enforcement of temporal stability is given by the time spent computing an occlusion-free view (t_c). For these experiments, the resolution of the buffers was set to ***r***=7. As described in Section 5.1, the threshold value, the size of the sliding window, and the number of sample points on the surface are strongly related. In this experiment, the stability dampened the movements of the camera. In fact, when considering multiple samples on the surface of an object, the ratio between the threshold value and the number of samples defines the confidence in the degree of visibility.

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

Table 5 : 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.


Computing New Camera Position : 75.55
Resolutions : 9 x 9 (223)
Settings Mode
Computing New Camera Position : ON
Minimum Move Distance : 0.1

Computing New Camera Position : 64.00
Resolutions : 9 x 9 (217)
Settings Mode
Computing New Camera Position : OFF
Accumulator Size : 7

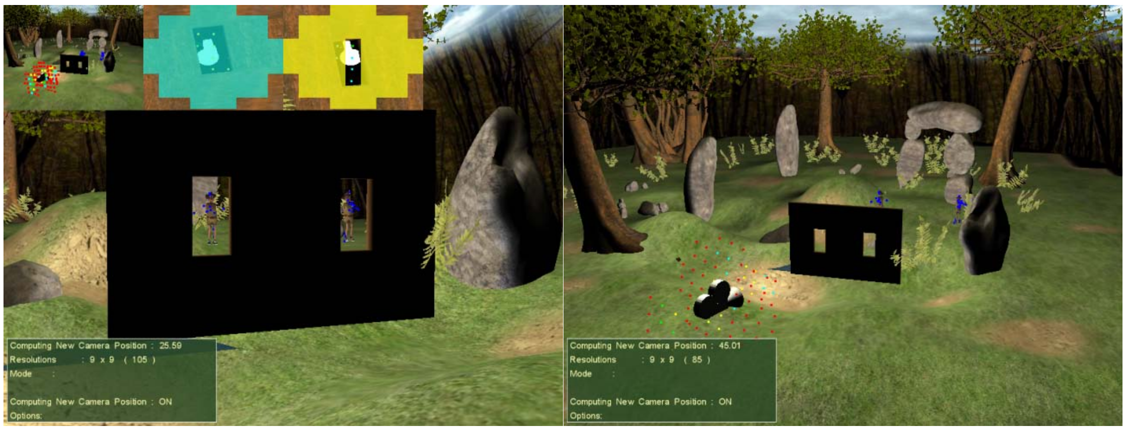
Figure 19 : Stability evaluation: a sparse occluder moves back and forth in front of two targets. The left figure shows significant movement by the camera (black knots at the basis of the camera), whereas the right frame shows how accumulating visibility states leads to significantly less camera movement.

Confidence in the degree of visibility

Since we cycle between viewpoints on surface of the target object over successive frames, the degree of confidence we have as to the full visibility of an object depends of the number of frames over which we conduct the estimation and the total number of viewpoint samples. In this experiment we use two target objects and a plane containing two holes. The surface of each object has been sampled with an average of 10 target positions for each viewpoint, and we compared the camera behaviors for different values of α , confidence in the degree of occlusion, which in turn determines the size of the sliding window ($\alpha = \frac{a}{n}$ where a is the threshold and n the number of samples). Results are reported in Table 6: r_N and r_P measure the rates of fully and partially occluded frames respectively (as a percentage of the total number of frames), while r_S reports the average percentage of occlusion for both targets (measured post process). See Figure 20.

α	r_N	r_P	r_S
0.1	7.47	7.12	77.4
0.5	5.42	8.5	58.2
0.8	3.21	8.9	15.7
1.0	0.9	1.7	11.3

Table 6 : Confidence α in the degree of visibility for the Temple benchmark with the RobustMove strategy. High values of $\alpha = \frac{a}{n}$ where a is the threshold and n the number of samples on the surface, guarantee a good visibility of target objects.


```
Computing New Camera Position : 25.59
Resolutions       : 9 x 9  ( 105 )
Mode       :

Computing New Camera Position : ON
Options:
```
```
Computing New Camera Position : 45.01
Resolutions       : 9 x 9  ( 85 )
Mode       :

Computing New Camera Position : ON
Options:
```

Figure 20 :Evaluation of the visual extent: each character presents a set of 10 target positions that have to be shot. The left view is a screenshot of the benchmark environment where camera configurations are displayed accordingly to their occlusion status: red if occluded for both characters, yellow or cyan if occluded for one of the characters, and green if unoccluded. The right view a subjective viewpoint taken from a solution camera configuration for which all 20 target points are in the shot.

Mutual occlusion

Mutual occlusion occurs when one of the target object hides the other. In most cases this does not require special treatment as each target object is always considered as a potential occluder of the other. However, in the specific case where there is no occlusion, but the target points A and B and the current camera configuration **O** are very close to being aligned (or are exactly aligned), the visibility volume cannot be computed as the left and right planes of the frustums do not intersect. This spatial configuration can be readily identified by checking the angle between **OA** and **OB**, and addressed by constructing a single perspective frustum from the furthest object of interest (see Figure 21). Using a single rendering allows the measurement of mutual occlusion. To maintain the capability to accumulate visibility information in temporal visibility volumes (and for accumulating visibility states over more than two objects), we can build a visibility volume in which candidate points are computed on the rays. Given the desired density ***d*** this is trivial to construct. See Figure 21.

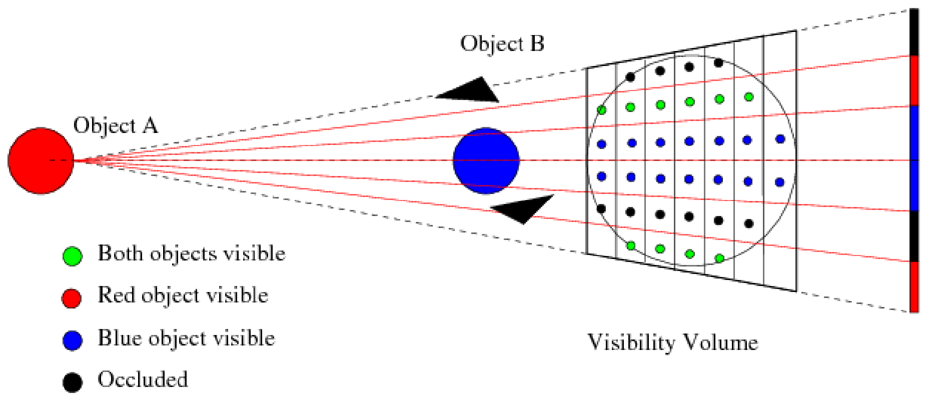


Figure 21 : The specific case of mutual occlusion: only one projection is performed, and samples are automatically generated to compose a visibility volume.

6. Results

This section demonstrates some results of our dynamic visibility computation process in a medium-to-complex geometric environment. The aim consists in tracking two dynamic targets (characters) in an environment with many sparse occluders (eg trees and furns) and a number of hard occluders (large stones, tree trunks). The camera path is a continuous path (plan-séquence). The focus is clearly not positioned on the cinematographic quality of this path, but directly guided the visibility and stability of the camera. A deeper integration of this visibility process with the Director Volumes framework is necessary to fully demonstrate the capacities. We describe this integration in section 7. However, on its own, the technique presents the interesting capacity of simulating a class of more modern styles related to hand-held camera motions. By directly constraining the height of the camera, as well as reproducing the motion of a virtual cameraman (ie by simply constraining the velocity of the camera), our system easily reproduces Steadycam styles or hand-held camera styles. Artefacts due to human motions (budging camera related to walking/running motions) can simply be expressed as new forces applied on the camera control process. Sinusoids and modified sinusoids have been empirically used and their effectiveness demonstrated in virtual environments for games [Lecuyer06].

Figure 22 displays a sequence of snapshots from our results.

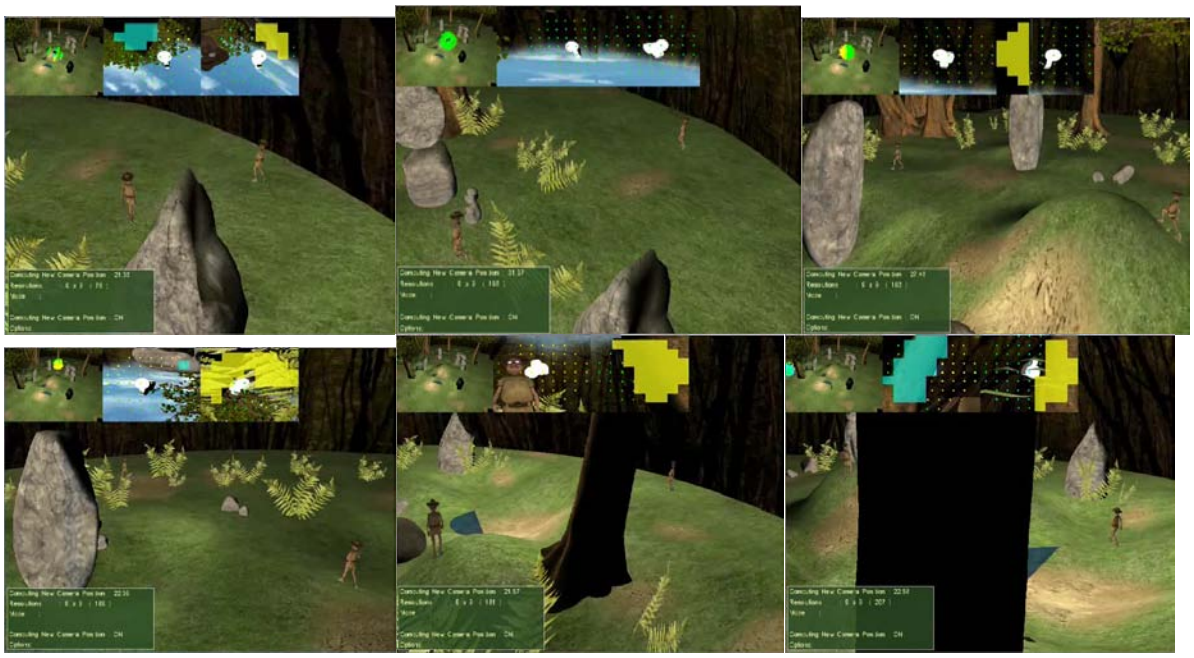


Figure 22: Simultaneous tracking of two characters in a complex 3D environment with both solid and sparse occluders, using a dynamic birds' eye view. The camera tracks both characters using the stochastic sampling to ensure an appropriate degree of visibility and temporal accumulation to ensure temporal coherence (avoid over-reactivity of the camera).

7. Integration in the Director Volume approach

The previous section has presented a new technique to efficiently handle complex occlusion in dynamic environments (moving targets with moving occluders). The benefits of this approach lay in its ability to be included in a standard camera control process to enhance it with visibility computation possibilities. We therefore detail in the following section how we can extend the results of Deliverable D5.2 by adding the capacities of this real-time dynamic occlusion avoidance. The process we propose to follow consists adding the Visibility Volume technique as a component to compute a precise estimation of key-subjects inside the Director Volumes. This component with be used at two stages:

(1) when selecting the next Director Volume to which a cut needs to be performed, to estimate the quality of the viewpoints inside the Director Volume

(2) when planning a camera path between two Director Volumes, to estimate the next best move to be performed (closest non occluded viewpoint to the camera path)

As detailed in D5.2, Director Volumes can be computed in real-time with dynamic targets. The visibility process which consists in using a cell-and-portal representation provides a semantic characterization of the space of viewpoints around the targets in terms of partial/total/no visibility. However such a technique does not handle dynamic occluders (occluders other than key-subjects moving in the environment). Thus, merging the techniques of Director Volumes with those of Visibility Volumes represents an important milestone in providing a full expressive and reactive camera control technique for interactive storytelling.

The process we propose to follow consists in using the Visibility Volume technique inside the Director Volumes. Four steps are required to complete the process:

- adapt the location and extent of a Visibility Volume to a specified Director Volume, or to a specified camera path
- compute the dynamic visibility inside the Director Volume or along the path
- adapt the local optimization process with a new criteria which computes the best composition for a given frame, together with the best visibility.

All three steps are detailed in the following paragraphs.

We first consider separately the problems of adapting the Visibility Volume to (1) a Director Volume and (2) a path between Director Volumes.

Adapt the location to a specified Director Volume

The first case occurs when a target Director Volume is selected (this will be the next shot to which the camera will cut). We first compute an enclosing sphere of the Director Volume from which we build the Visibility Volume. The sampling process is performed inside the visibility volume. Depending on the number of key subjects framing in the Director Volume, the process either builds a single frustum, or the intersection of multiple frustums. Temporal accumulation (duration over which occlusion is considered) is a parameter of the system and its value is guided by responsiveness one expects from the camera system.

Compute the dynamic visibility along the path

In the second case, a path is provided between to Director Volumes (the path is interactively recomputed as the target volume may move and evolve in time). The sampling process is then performed along the path.

Adapt the local optimization process

Finally, the last step merges the screen composition process with the visibility information. The general local search process (described in D5.2) is unchanged, only the fitness function is changed to include the visibility information. We first provide a visibility fitness function f which, given a location in x the Director Volume and a visibility threshold v , returns an estimate of how much the location x is close to the expected threshold v :

$$f(x,v) = \begin{cases} if\ V_{AB}\left(I_{AB}^{-1}(x)\right) > v\ then\ 1 \\ \square \\ else\ [1 - (v - V]_{AB}\left(I_{AB}^{-1}(x)\right)) \end{cases}$$

The fitness function is then expressed as a weighted sum between composition and visibility, where α and β represent the weights:

$$F(x) = \alpha.c(x) + \beta.f(x,v)$$

Weights have been fixed through a series of experiments. Since the composition is generally easy to achieve (all shots in a Director Volume present a similar composition).

When considering a camera moving from one Director Volume to another, the process is more complex. Composition is only enforced in the initial and final shot (and interpolated on orientation is performed between shots). For both shots, the fitness function $F(x)$ is used (and since the destination Director Volume may evolve, the composition is recomputed at each step). In between initial and final shots, the process searches the camera configuration which is closest to the computed path, and which maximizes the visibility of targets which are on screen.

8. Concluding remarks

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 two or more complex 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. Using the target object centered projections, which lie at the core of our approach, means that we not only exploit widely available hardware rendering capabilities, but that developers are able to control the cost of the occlusion computation through the resolution of these projections (using the sampling density). The stochastic modelling of visual extent and the use of from-region visibility estimates, means that objects are no longer treated as point-like abstractions.

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). In our experiments we have demonstrated the utility of these enhancements. Of equal importance is the fact that since our approach is based on the generation of sets of candidate camera positions (within the dynamic limits of the camera) few constraints are placed on its extension to incorporate other declarative aspects of camera planning.

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. For this reason, the process is integrated in an approach which shares a global knowledge of the scene (typically the Director Volumes proposed in D5.2). Resampling of the shared shadow map, when handing three objects (or more), also introduces a degree of approximation, as does resampling in the processes that accumulate visibility information. However, our proposed technique has significant potential to enhance applications that require assisted interactive or automated camera control in complex environments. The ability to reliably and efficiently target two or more objects is of particular importance in 3D computer games, for which the limitations of existing approaches impacts significantly on the expressive use of the camera. The technique is lightweight, utilizes ubiquitous graphics hardware, and can be readily incorporated into the rendering process of any real-time graphics application.

In the second part of this deliverable we presented how this novel approach can be integrated within our cinematography framework (described in D5.2), in particular how Director Volumes can exploit the sampling process performed by the Visibility Volume. The close integration of these two approaches yields one of the very first fully interactive cinematography camera control system, able to react to high-level inputs such as new events occurring and changes in editing style, together with reaction to low-level geometric issues such as occlusion which often occurs in interactive contexts.

In such, this approach enables both the encoding of more traditional cinematic styles by relying on viewpoint selection and classical editing rules, together with more modern cinematographic styles in that the camera can freely move around the environment while maintaining the elements visible and portraying the narrative (eg. hand-held camera devices, steadycam, …).

9. Perspectives

Staging the camera in an interactive context and encoding some degree of directorial style only represents one of the many dimensions to support and convey a narrative. Elements such as audio tracks, lighting and staging are as essential as framing and cutting, and need to be considered simultaneously. For example, relation between audio tracks (speech and music) and the way cuts are performed is very complex. A difficult problem in particular is selecting the appropriate moment for performing the cut (making the unnoticeable cut). While literature identifies which are the appropriate moments when cuts can be performed due to actions, speech and music (on a characters reaction, on a characters action, in anticipation to a characters reaction, or just after a character has started talking), there are in a symmetric way moments where cuts cannot be performed (on key words of the speech, on small changes in posture that portray changes in affinity, intimacy, dominance, on slow trackings which convey a character's personal thoughts). In trying to encode such notions, we need to further explore temporal probabilistic models for cutting which mix elements of audio and character actions to provide hints for appropriate cuts (a sort of temporal map of possible cuts along both the action and audio tracks).
A second key problem lies in the strong relation that exists between directorial style and staging. In the current state of our progress, we consider staging as an input (provided by WP4 Intelligent Virtual Agents) and not as a parameter we can play with. As described in the literature (see Bordwell for a detailed review on staging [Bordwell]), staging and directorial style are linked to a point that and a characterization of style necessarily encompasses a characterization of staging. Thought the shift from 1910-1920 cinema with mostly static camera shots and elaborate staging, to more recent 1960-1980 Hollywood styles with dynamic cameras and a large number of cuts has reduced the importance and complexity of staging, it remains a way to complement and instrument the camera work. It provides elegant ways of appropriately guiding the spectators gaze through blockage (a character blocking the view over another one) and revelation (a character moving to let another one appear). Therefore the next axis of our research will focus on how staging and cinematography can be considered as the same problem. We will follow an empirical approach based on film analysis to extract elements of both dimensions and provide computational models to enforce them in an interactive context.
A third key problem is related to composition and how the correct layout of elements on the screen can support the conveyance of specific narrative elements (eg. relations between characters). In our current approach, we adopt a rough model which enforces elements of dominance and affinity through application of rules of the thirds and the fifths. We have not considered elements of perception (eg. lighting of the characters), influence of setting and balance which improve the composition, and incidental elements which may ruin the composition (secondary characters who may be incidentally cut by the edges of the frame, meaningless objects which make an unbalanced composition balanced, etc.). This problem is currently under study through the identification of key properties in composition and the provision of appropriate numerical optimization techniques.

By building on the current results (already a step ahead on the state of the art in interactive storytelling cinematography techniques) and in providing practical and computational models to these key problems, we believe our contribution will represent a solid scientific foundation on which to build the future of interactive virtual storytelling.

Bibliography

[Agrawala00] Agrawala, M., Ramamoorthi, R., Heirich, A., and Moll, L. 2000. Efficient image-based methods for rendering soft shadows. In SIGGRAPH ’00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co.,New York, NY, USA, 375–384.

[Assarsson03] Assarsson, U., and Akenine-M¨oller, T. 2003. A geometry-based soft shadow volume algorithm using graphics hardware. In SIGGRAPH ’03: ACM SIGGRAPH 2003 Papers, ACM, New York, NY, USA, 511–520.

[Arijon76] Arijon, D. 1976. Grammar of the Film Language. Hastings House Publishers.

[BaresL99] Bares, W. H., and Lester, J. C. 1999. Intelligent multi-shot visualization interfaces for dynamic 3D worlds. In Proceedings of the 4th international conference on Intelligent user interfaces (IUI 99), ACM Press, New York, NY, USA, 119–126.

[Beckhaus] Beckhaus S.: Dynamic Potential Fields for Guided Exlporation in Virtual Environments. PhD thesis, Fakultät für Informatik, University of Magdeburg, 2002.

[Bordwell] Bordwell D., Figures traced in light: On Cinematic Staging, University of California Press, 2005

[ChristieEG05] Marc Christie, Jean-Marie Normand A Semantic Space Partitionning Approach to Virtual Camera Control in Proceedings of the Annual Eurographics Conference, Computer Graphics Forum, Volume 24-3, pp 247-256, 2005

[Christieetal08] Christie, M., Oliver, P., and Normand, J.-M. 2008. Camera control in computer graphics. Computer Graphics Forum 27, 8, 2197–2218.

[Cohen-or00] Cohen-Or, D., Chrysanthou, Y. L., Silva, C. T., and Durand, F. 2000. A survey of visibility for walkthrough applications. IEEE Transactions on Visualization and Computer Graphics 9, 3, 412431.

[DruckerZ95] Drucker, S. M., and Zeltzer, D. 1995. Camdroid: A System for Implementing Intelligent Camera Control. In Symposium on Interactive 3D Graphics, 139–144.

[Durand00] Durand, F., Drettakis, G., Thollot, J., and Puech, C. 2000. Conservative visibility preprocessing using extended projections. In SIGGRAPH ’00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., New York, NY,USA, 239–248.

[Giors04] Giors, J. 2004. The full spectrum warrior camera system. In GDC ’04 : Game Developers Conference 2004

[GleicherW] Gleicher, M. and Witkin, A. P. 1992. Through-the-lens camera control. In SIGGRAPH (2006-02-10). ACM Computer Graphics, 331–340.

[Halper03] HALPER, N., HELBING, R., AND STROTHOTTE, T. 2001. A camera engine for computer games: Managing the trade-off between constraint satisfaction and frame coherence. In Proceedings of the Eurographics Conference (EG 2001), Computer Graphics Forum, vol. 20, 174–183.

[Hawkins] Hawkins, B. 2004. Real-Time Cinematography for Games (Game Development Series). Charles River Media, Inc., Rockland, MA, USA.

[Katz91] KATZ, S. 1991. Film Directing Shot by Shot: Visualizing from Concept to Screen. Michael Wiese Productions.

[Lecuyer06] Lecuyer, A., Burkhart, J-M., Henaff J-M., Donikian S. Camera Motions Improve Sensation of Walking in Virtual Environments, In Proceedings of IEEE VR, 2006

[Oskam09] Oskam T., Sumner R. W., Thueray N., Gross M., 2009: Visibility transition planning for dynamic camera control. In Proceedings of Symposium on Computer animation (SCA)

[Philips] Phillips, C. B., Badler, N. I., and Granieri, J. 1992. Automatic viewing control for 3d direct manipulation. In Proceedings of the 1992 symposium on Interactive 3D graphics. ACM Press New York, NY, USA, 71–74.

[Pickering02] Pickering J. H.: Intelligent Camera Planning for Computer Graphics. PhD thesis, Department of Computer Science, University of York, 2002.

[Plantinga90] Plantinga, H. and Dyer, C. R. 1990. Visibility, occlusion, and the aspect graph. Int. J. Comput. Vision 5, 2, 137–160.

[Sudarsky99] SUDARSKY, O., AND GOTSMAN, C. 1999. Dynamic scene occlusion culling. IEEE Transactions on Visualization and Computer Graphics 5, 1, 13–29

[Vasquez07] Vazquez P.-P., Feixas M., Sbert M., Heidrich W.: Automatic view selection using viewpoint entropy and its application to image-based modelling. Computer Graphics Forum 22, 4 (2003), 689–700.

[WareO90]Ware, C. and Osborne, S. 1990. Exploration and virtual camera control in virtual three dimensional environments. In Proceedings of the 1990 symposium on Interactive 3D graphics (SI3D 90). ACM Press, New York, NY, USA, 175–183.

[Zhu90] Zhu, Q. 1990. A stochastic algorithm for obstacle motion prediction in visual guidance of robotmotion. IEEE International Conference on Systems Engineering, 216–219.