Occlusion-free Camera Control

Marc Christie, Patrick Olivier, Jean-Marie Normand

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Occlusion-free Camera Control

Marc Christie — Patrick Olivier — Jean-Marie Normand

N° 6640

Fevier 2008

Thème COG

ISSN 0249-6399 ISRN INRIA/RR--6640--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
RENNES - BRETAGNE ATLANTIQUE

Occlusion-free Camera Control

Marc Christie*, Patrick Olivier†, Jean-Marie Normand‡

Thème COG — Systèmes cognitifs

Équipe-Projet BUNRAKU

Rapport de recherche n° 6640 — Fevier 2008 — 27 pages

Résumé : Le calcul et le maintien de la visiblité d'objets cibles est un problème fondamental dans la conception de processus de contrôle de caméra dans des applications graphiques 3D. La majorité des algorithmes temps-réel de contrôle de caméra incorporent des mécanismes d'évaluation de la visibilité à partir d'un point de vue unique et ce faisant idéalisent la complexité géométrique des cibles. Nous présentons une nouvelle approche pour l'évaluation en temps réel de la visibilité d'objects cibles multiples qui calcule simultanément leur visibilité pour un grand nombre de points. Ce calcul de visibilité repose sur des rendus en basse résolution à partir de couples de points sur les objets cibles, sur un plan de rendu partagé, parallèle au couple de points et derrière la caméra. En combinant les informations de profondeur de ces rendus, la visibilité conjointe du couple de points est rapidement calculée pour un ensemble de configurations autour de la configuration courante de caméra. Cette combinaison par couple est étendue à trois objets cibles ou plus, et aggrégée dans une fenêtre temporelle pour stabiliser la sur-réactivité de la caméra. Pour adresser le problème d'idéalisation géométrique des objets cibles, nous proposons une approximation stochastique de l'extension visuelle de l'objet en sélectionnant des points de projection de façon aléatoire sur la surface visible de l'objet. Nous démontrons l'efficacité de l'approche pour des configurations de scènes complexes et problématiques en temps-réel, pour deux ou trois objets cibles.

Mots-clés : Contrôle de caméra, gestion de l'occultation, calcul de visibilité temps-réel

* Equipe BUNRAKU, MCF en détachement auprès de l'INRIA Rennes, France
† Senior Lecturer, Newcastle University, UK
‡ PhD, LINA CNRS UMR 6241, Nantes University, France

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Occlusion-free Camera Control

Abstract: Computing and maintaining the visibility of target objects is a fundamental problem in the the design of automatic camera control schemes for 3D graphics applications. Most real-time camera control algorithms only incorporate mechanisms for the evaluation of the visibility of target objects from a single viewpoint, and typically idealise the geometric complexity of target objects in doing so. We present a novel approach to the real-time evaluation of the visibility of multiple target objects which simultaneously computes their visibility for a large sample of points. The visibility computation step involves performing a low resolution projection from points on pairs of target objects onto a plane parallel to the pair and behind the camera. By combining the depth buffers for these projections the joint visibility of the pair can be rapidly computed for a sample of locations around the current camera position. This pair-wise computation is extended for three or more target objects and visibility results aggregated in a temporal window to mitigate over-reactive camera behaviour. To address the target object geometry idealisation problem we use a stochastic approximation of the physical extent of target objects by selecting projection points randomly from the visible surface of the target object. We demonstrate the efficiency of the approach for a number of problematic and complex scene configurations in real-time for both two and three target objects.

Key-words: Camera control, occlusion-free views, real-time visibility computation

1 Introduction

Although automatic camera control would appear to be a key problem in computer graphics it has received relatively little attention. In general, proposed approaches aim to devise both declarative formulations in which constraints can be placed on the visual properties of a shot (such as the size and position of target objects) and general purpose algorithms to find camera locations and paths that satisfy these constraints. With richer sets of properties, and powerful solving mechanisms, a camera control system with genuine cinematic qualities might be a real possibility [Ari76], [Kat91]. However, all camera control frameworks fundamentally rely on being able to compute and reason about the visibility of target objects in dynamic environments. 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 2 we review a number of more powerful techniques including sphere-based projection [BL99]. In ray casting approaches the candidate position for the camera is evaluated by casting a ray from the camera in the direction of the target object. Though inexpensive, the approach is fundamentally limited to the evaluation of the visibility of a single point, on a single target, along a single line-of-sight.

An incremental improvement (at a linear increase in cost) can be achieved by ray casting from an array of candidate camera locations, and by repeating the process for other target objects where the visibility of multiple target objects is required. However, simple ray casting, even when repeated for multiple targets and multiple candidate camera positions, only evaluates the joint visibility conditions at individual viewpoints. Deciding how to move the camera based on such collections of single point estimates has a number of undesirable consequences, for example, it is not possible to maintain partial visibility of target objects as they move behind sparse occluders such a railings. Furthermore, using a point to approximate the geometrical complexity of a target object fails to sufficiently characterise the visibility of an object in dynamic scenes.

Our approach to occlusion-free camera control addresses all these limitations as follows :

Visibility volume sampling The principal innovation is the generation of a 3D sampling of the visibility of multiple objects through the aggregation of 2D projections. Based on the dynamic properties of the camera, the possible positions of the camera in the next frame defines a region of space around the current position – the *feasible camera volume*. By projecting back from the target objects, towards the camera, two view volumes can be constructed that tightly bound the feasible camera volume. An enumeration of the intersections between rays comprising each of these projections yields a 3D sampling of the visibility properties for the two target objects (actually the visibility of the projection points on their surfaces) – we refer to this as the *visibility volume*.

Multi-object visibility evaluation The approach can be extended to more than two target objects by constructing a set of visibility volumes for object pairs, such that every target object occurs in at least one pair. By

interpolating across points in the *visibility volume* the visibility of all the target objects can be estimated.

Stability of visibility estimates and partial occlusion By aggregating the *visibility volume* information over a temporal window our approach addresses both overly reactive camera behaviour, where occlusion is momentary, and aspects of the partial occlusion problem.

Stochastic estimation of visual extent We automate the process of choosing the viewpoints on the surface of the target objects in a stochastic pre-processing step. Our estimation of the visual extent is pre-computed for a set of different camera angles and distances from a target object. By selecting the appropriate pre-computed value the visual extent of the target can be accurately estimated without re-rendering or using coarse geometric approximations (such as bounding volumes).

Efficiency As our technique uses fundamental aspects of the 3D rendering pipeline it is supported in hardware as a matter of course; no lighting is required and through modifications to the resolution of the renderings the processing demands can be adaptively controlled in an application dependent manner. There is significant scope for optimisation of our initial implementation using conventional approaches such as view frustum culling. Though we recognise that some sections of the implementation could be further optimised using a GPU, in our test cases involving moderately complex scenes, the average time to compute an occlusion-free view for two target objects was less than 3 ms per frame.

2 Occlusion-free Camera Control

A basic challenge for camera control in most application domains is the maintenance of unoccluded views of target objects. Although a number of techniques can be utilized for maintaining the view of a single target in a real-time environment (*e.g.* [HHS01]), how these might be extended to multiple objects of interest is not obvious. Nevertheless, there exist demanding applications for which the efficient and reliable computation of such occlusion-free views is essential. In particular, multiplayer computer games require camera shots that include two or more protagonists, frequently in highly cluttered environments. Similarly, there are numerous scenarios in scientific and information visualization applications that would benefit from the automatic maintenance of unoccluded views of multiple data points or regions.

Surprisingly, the computation of occlusion-free views in dynamic environments has received relatively little attention from the computer graphics community. Indeed many of the early approaches were proposed by researchers in computer vision and robotics. For example, work in sensor planning has developed approaches for the placement of lights and cameras based on the visibility characteristics of the edges and faces of objects [YHS95]. Important contributions relating to visibility characterisation centre on the construction of aspect graphs, boundary or CSG representations to model and compute all the distinct viewpoints in space for static polyhedral scenes [PD90, TTK96]. Likewise, target tracking has received considerable attention from computer vision, and although

tracking in the real world shares many of the problems of camera control for 3D graphics [YLPL05], the specificities of the techniques prevent their use in practice. In computer graphics and computational geometry there is also a significant related body of literature concerning the visibility characterisation of three dimensional geometries and its application in occlusion culling [GKM93] and shadow generation. Though the underlying techniques can be employed to improve some aspects of computing occlusion-free views (*e.g.* in establishing the visibility for static occluders), the nature of the problem we consider (real-time, highly dynamic, partial and temporal occlusion) requires a specific solution.

In dynamic environments, the management of occlusion may be achieved by performing ray casts from the camera to the target to evaluate the visibility and plan subsequent camera movements. The efficiency and simplicity of ray casting techniques make them the default choice for many real-time applications, in particular, for computer games [Gio04] and game-based environments [BZRL98]. However, a ray only evaluates the visibility of a single point and the adaptation of ray casting for multiple targets, and for targets that are not reasonably approximated by a point, is problematic. One alternative approach to the management of occlusion in real-time contexts is to use *consistent regions* of space through the representation of the visibility of a target object in local spherical coordinates centered on the object [BL99]. 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 then negating these to yield occlusion free viewpoints [BL99, PBG92, DZ95].

Similarly, Courty [CM01] and Marchand [MH98, MC02], avoid occlusion in a target tracking problem by computing an approximate bounding volume around 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, camera motion or motion of other object. This notion has been extended to address objects with unknown trajectories through the computation of approximate predictive volumes based on the current position and motion of an object. However, the approximate nature of the bounding volumes leads to restrictions in both expressiveness (*e.g.* an inability to handle partial occlusion) and practical application (over-estimation for complex shapes).

Other techniques have sought to exploit graphics hardware in the treatment of occlusion [HO00]. By rendering a scene in hardware stencil buffers using a color for each object, the number and extent of occluding objects can be very efficiently evaluated. Hardware rendering techniques have a number of attractive characteristics 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 allow the use of low resolution buffers when appropriate. Phillips *et al.* [PBG92] project a scene on the different faces of a cube to achieve a visibility map, and choose the closest empty area in the map by local neighbourhood exploration. Halper *et al.* [HHS01] introduced the use of geometric primitives referred to as *potential visibility volumes* (PVRs) that are configured around the current camera location. The scene is then rendered, from the target to the camera, using distinct colors for occluders and PVRs.

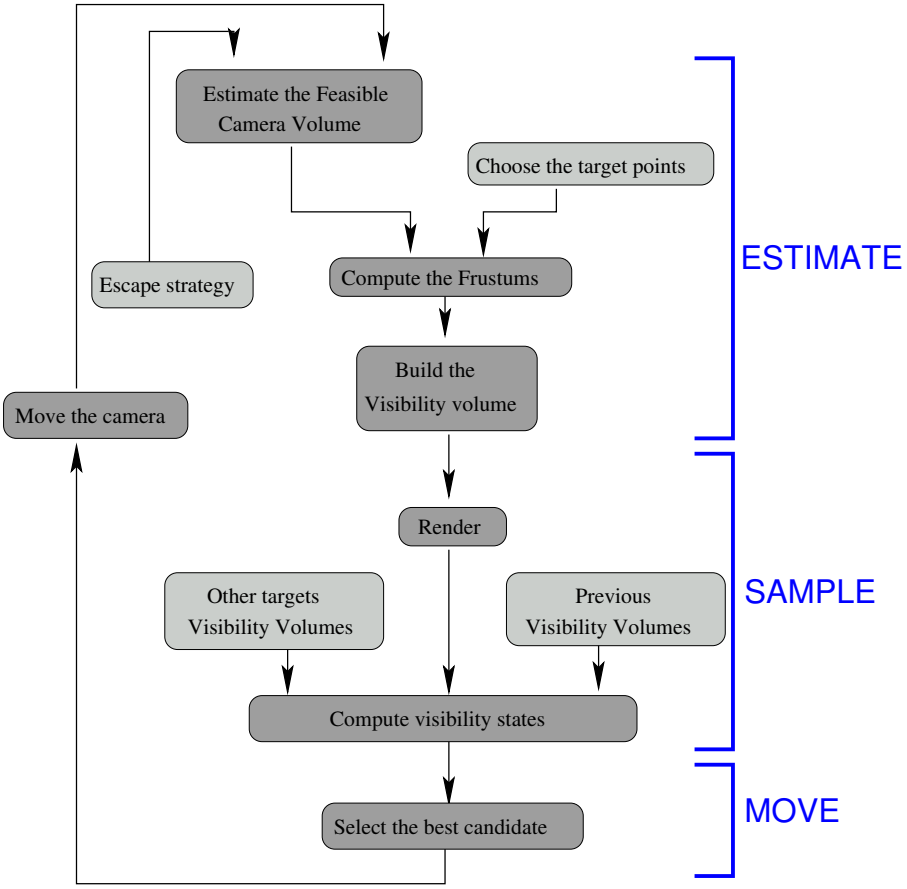

Fig. 1: *Detailed steps of the visibility computation.*

Inspection of the color and depth buffers informs the choice of the next camera movement. Although efficient, none of these techniques readily scale to cases where there is more than one target object, where the visual extent of the target object(s) is not well approximated by a point, or where the management of both partial and temporal occlusion is necessary.

3 Visibility Volume Sampling

The diagram in Figure 1 illustrates the main steps of our process : (i) estimate the feasible camera volume, (ii) sample the volume, and (iii) choose the next camera move. Our proposal builds on approaches to single target object tracking [PBG92, HHS01] that render the scene from the target back towards the actual position of the camera, and then reason over the color and the depth buffer values. The color buffer identifies the occluders and the depth buffer provides the distance to these occluders. The difficulty lies in the efficient and reliable composition of projection information so as to be able to reason about the visibility of multiple targets over the 3D space of possible camera positions. By

intersecting the target object projections we can compose the visibility information for all target objects (see Figure 2 for a 2D illustration and Figure 4 for 3D illustration). Indeed, careful selection of the geometry of the projections means that their intersection will define a *visibility volume*. Using depth values for each projection we can obtain a 3D sampling of the visibility of the targets objects in the visibility volume. The geometry of the projection and the resolution of the images determine the granularity of the sampling. Furthermore, visibility volumes for different frames can also be combined in a process that seeks to ensure camera stability. Though in what follows we only reason over depth information, the object indexed color information can be exploited in distinguishing the nature of the occluders (*e.g.* different colors for static and dynamic objects) which can inform camera motion strategies to be deployed.

3.1 Principle

The process is best described first in two dimensions. Figure 2 shows two frustums, each extending from a target object, that share a common far plane. The projected occluding elements of the scene yield two low resolution images on this plane. For all the points on the ray from target object A to pixel i, we know if occlusion occurs (resp. does not occur) for distances greater than or equal to z_i^A (resp. lower than z_i^A) where z_i^A denotes the distance from A to the occluder stored in i-th position by the depth buffer (see Figure 3). By composing this information with depth information at index j from the second rendering for target object B, we can evaluate the visibility status of each intersection as :

- 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)

For the 2D case, let the n-dimensional vector $\mathbf{z}^A = [z_1^A, \cdots, z_n^A]^T$ denote the vector of depths related to A. We define the composition operator $\otimes$ that computes the visibility state from two depth informations z^A and z^B as one value of $\{N, P_A, P_B, V\}$ as presented in Figure 3 (symmetric configurations are not displayed). Consider two rays from A and B that intersect at position I. Each z-buffer contains the depth of the closest occluder on each ray, given by values z^A and z^B, which are measured from the target to the occluder. To determine the visibility state at position I, we need to know whether the closest occluder is 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 z^{AI}) is lower than z^A, and the distance from point B to I (*i.e.* z^{BI}) is lower than z^B.

The operator can be described as follows :

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

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.

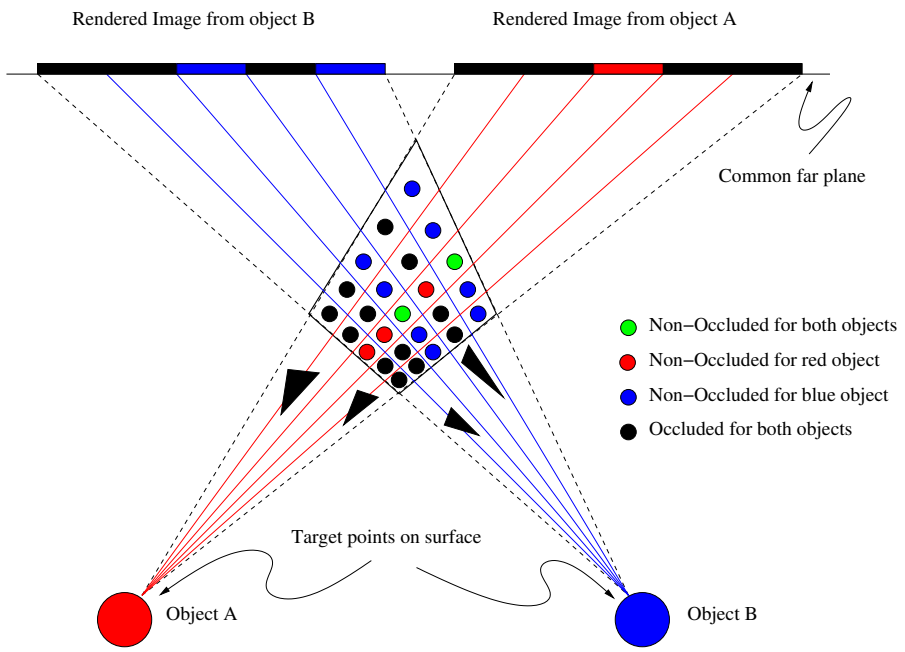

FIG. 2: *2D case : intersecting target object projections yields a sampling of the* visibility volume *; the visibility status of each point depends on the presence and depth of the occluders in both images.*

We can define the extension of this composition operator $\otimes$ to vectors by building the matrix of visibility states :

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

Extending this to three dimensions requires the specification of asymmetric frustums (one for each target object) such that the viewplanes are coplanar. This ensures that corresponding rays for the two target objects will in fact intersect (see Figure 4). In order to efficiently compute and access the visibility states inside the visibility volume, we express the intersection of the frustums (which is a distorted cube) as a 3D trilinear coordinate system as shown in Figure 5. Each point I inside the visibility volume can then be represented in local coordinates, $I = [uvw]^T$, and expressed in global Cartesian coordinates as a linear combination of vectors $\mathbf{i}, \mathbf{j}_1, \mathbf{j}_2, \mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4$ (where $\mathbf{i}, \cdots, \mathbf{k}_4$ are defined in Cartesian coordinates). We can then refer to $\mathbf{I_{AB}} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ as the trilinear interpolation function related to the visibility volume for target objects A and B, that expresses local coordinates $[uvw]^T$ in Cartesian coordinates $I' = [xyz]^T$, where :

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

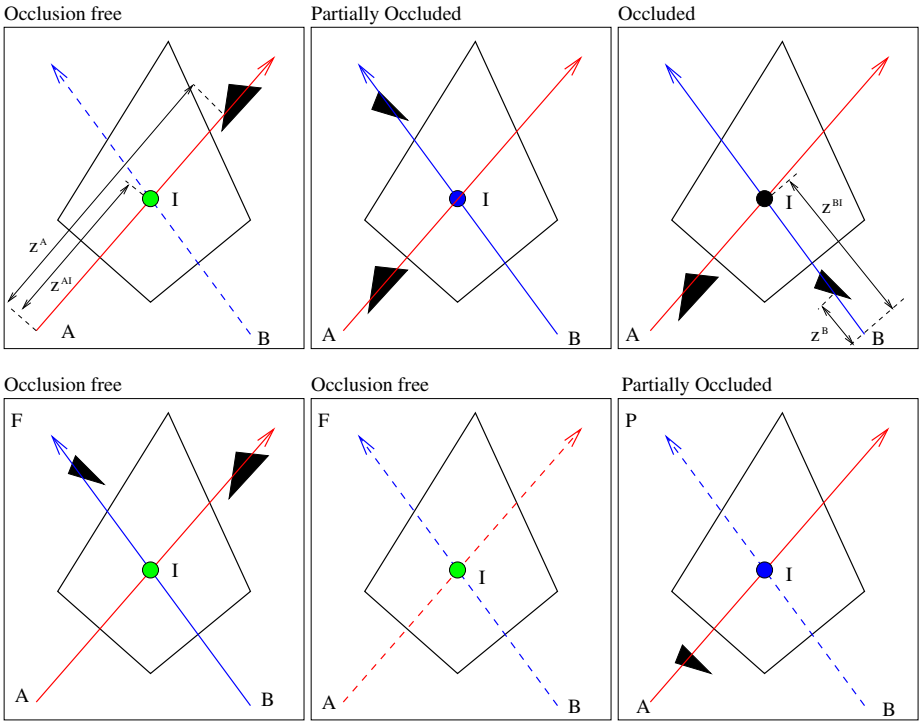

FIG. 3: *Case study of occlusion configurations w.r.t. the distance to the intersection point; dashed lines denote the absence of occluders.*

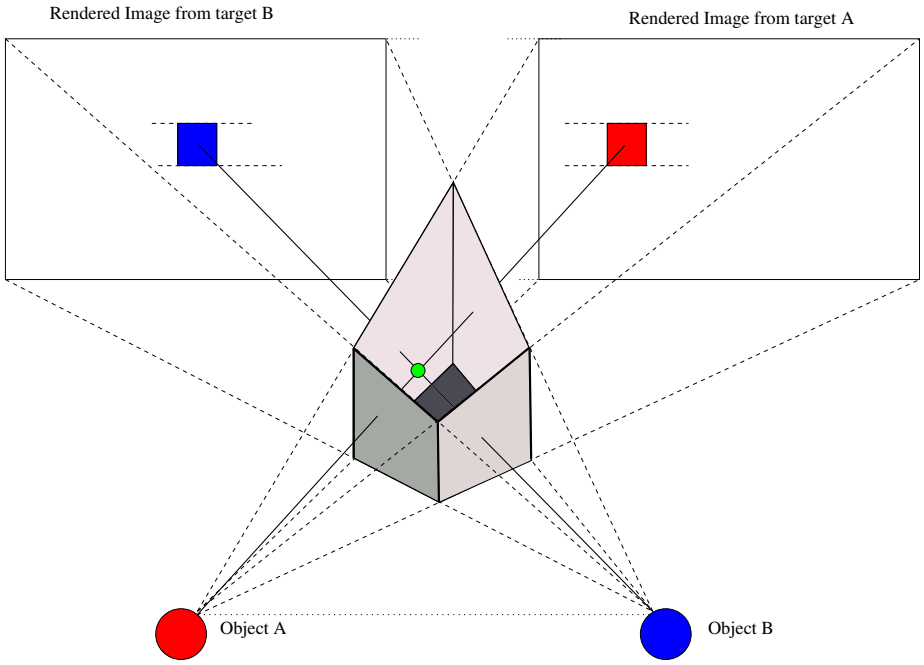

Fig. 4: *In 3D, a trilinear basis is defined by the intersection of two projection pyramids*

We can now very conveniently map the 2D pixel coordinates of each rendered image into this trilinear system. Indeed, a pixel of coordinates (u_1, v_1) in image B, together with a pixel of coordinates (u_2, v_1) in image A (the second component is identical as only rays in corresponding scanlines are intersected), intersect exactly at local coordinates $[u_1, v_1, -u_2]^T$ inside the visibility volume. This enables the efficient computation of all the intersection points inside the volume, as well as direct access to the visibility states in the corresponding 3D matrix given a pair of 2D coordinates in the image. Furthermore, it is possible to compute the inverse function $(\mathbf{I_{AB}})^{-1}$ to obtain the local coordinates, and thus the visibility status of any point in the visibility volume from its Cartesian coordinates (see Section 5).

To ensure that intersections occur, between rays for the same scanline in the projections of the two target objects, we need to impose the following conditions : (1) that the top and bottom planes of both frustums are coincident ; and (2) the projection planes must be coincident, parallel to the line (A, B), and such that the orthogonal projection of A (or B) on the plane represents its normal.

By utilising standard graphics hardware to perform the two renderings, this representation 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, is the basis for choosing both whether to move the camera, and where to move it to (see Section 3.4). The granularity of the sampling is directly derived from the resolution of the 2D renderings and is

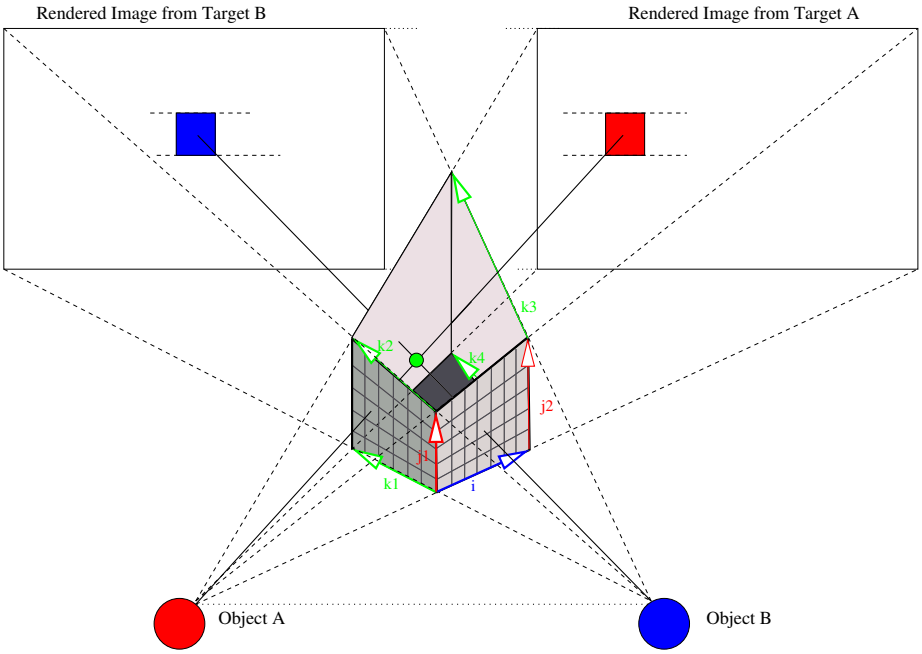

Fig. 5: *The trilinear local coordinate system enables to address the intersection points in the visibility volume.*

easily controlled according to the characteristics of the environment and the resource demands of the application.

3.2 Estimate : computing the visibility volume

The camera dynamics (position, velocity and acceleration limits) enable us to bound the next camera position within a plausible subset of space that we refer to as the *feasible camera volume*. Given that we know the current camera position $\mathbf{c}_t$ at time t, its velocity $\mathbf{v}_t$, and the maximum acceleration that we are allowed a_{max}, we can estimate the motion of the camera in all directions using Euler integration, which is bounded by the following set :

$$\{\mathbf{c}_{t+\Delta t} \in \mathbb{C} | \forall \mathbf{d} \in \mathbb{S}, \mathbf{c}_{t+\Delta t} = \mathbf{c}_t + \Delta t \mathbf{v}_t + \Delta t^2 a_{max} \mathbf{d}\}$$

where $\mathbb{C}$ is the set of camera positions, and $\mathbf{d}$ represents the direction of the acceleration inside a sphere $\mathbb{S}$ of radius 1 (here, we allow the acceleration in all the possible directions). Though notoriously imprecise for large values of Δt, this integration offers a good estimate of the region of space that can be reached by the camera. We approximate this region using a bounding sphere, centered in $c_t + \Delta t v_t + \Delta t^2 (a_{min} + a_{max})/2$ of diameter $\Delta t^2 (a_{max} - a_{min})$.

The two frustums can then be computed such that their intersection (and thus the resulting visibility volume) bounds this sphere of reachable camera positions. By having two distinct target points A and B, and the center C of the sphere, we define two planes $\top$ and $\bot$ that include the line (A, B), and are tangential

to the sphere S (above and below). The angle between the planes represent the vertical viewing angle of the frustums. The common projection plane of both views is located behind the sphere, tangential to it, orthogonal to plane (A, B, C) and parallel to line (A, B). Left and right planes of both frustums are similarly defined, tangentially to $\mathbb{S}$.

3.3 Sample : rendering from both viewpoints

The rendering is performed using these frustums which are not centered around the direction of perspective (see Figure 6). The frustum coordinates are computed by projecting the vertices of the visibility volume onto the near plane (here a plane of distance 1 to the center of projection and parallel to the rendering plane). The direction of perspective of the camera is oriented along the normal to the rendering plane. If $\mathbf{o}_1$ and $\mathbf{o}_2$ are two opposite vertices of the volume for the viewport with left, right, top and bottom parameters (l, r, t, b), then :

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

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 of basis from global coordinates to local camera coordinates in A, and $\mathbf{P}$ is a classical perspective projection matrix.

The resolution of the rendering buffers, which defines the granularity of the sampling, is dynamically computed given a required mean density parameter δ for the search space of camera positions, and the radius r of the sphere. For simplicity the resolutions are set the same value in both dimensions (u and v) for both buffers. Given a required mean density δ, the resolution g is given by :

$$g = \sqrt[3]{\delta \frac{4}{3} \pi r^3}$$

Clearly the actual density of sample points 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 the density reducing as we move away from the targets.

3.4 Move : choosing the next camera position

The visibility values of the sampling of the volume allows the construction of four sets of potential camera positions $\mathcal{S}_N$ (both A and B occluded), $\mathcal{S}_{PA}$ (A

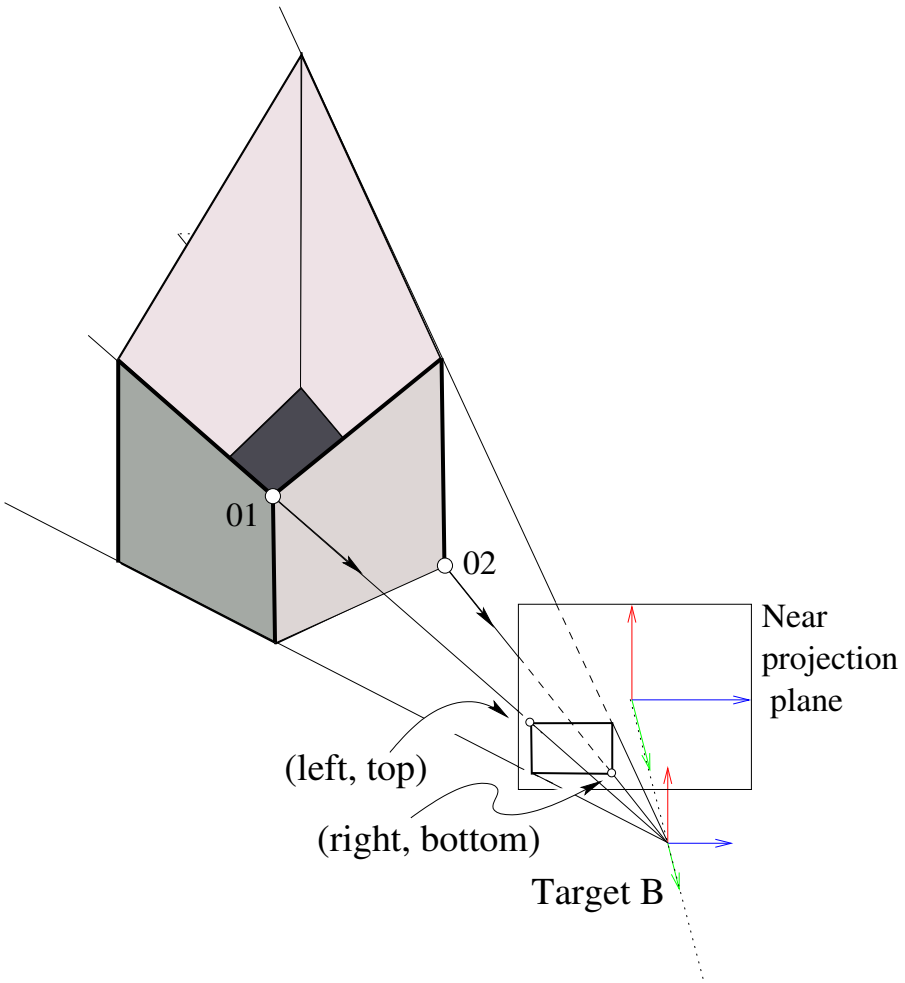

Fig. 6: *For each target, the specific frustum coordinates are computed by projecting two vertices of the visibility volume onto the near projection plane. This leads to an asymmetric frustum with regard to the direction of projection (along $-z$).*

visible only), $\mathcal{S}_{PB}$ (B visible only), and $\mathcal{S}_V$ (both A and B visible). The choice of which of these positions to move to next depends on the underlying application. 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. First, if $\mathcal{S}_V$ is not empty, we select the position which requires the smallest change in the velocity of the camera (*i.e.* avoiding sudden jumps). Whenever $\mathcal{S}_V$ is empty, we choose a position in either $\mathcal{S}_{PA}$ or $\mathcal{S}_{PB}$ depending on the visibility of targets A or B in previous frames. When all the configurations are occluded for both targets, we simply base the next position on the current dynamic properties (and constraints) of the camera.

4 Stability of Visual Estimates

In addition to occlusion, another problem for real-time camera control is the appropriate and efficient maintenance of camera stability (*i.e.* avoiding abrupt and visually incongruent changes in the velocity of the camera). Due to the sudden nature by which occlusion occurs in dynamic and complex environments, it is necessary to incorporate mechanisms that provide such camera stability (usually at the cost of allowing temporary occlusions). 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 temporary occlusions. This process is realised through the use of a sliding window over the past n visibility volumes and uses a bounded accumulation operator to aggregate the visibility information.

The aggregation operator $\bigoplus_a$ is applied over a set of visibility states $\{v^t, \cdots, v^{t-n}\}$ and acts as a filtering process by returning a visibility state v that has appeared at least a times in the last n frames (where a is greater than $n/2+1$). We treat cases where no state meets this condition pessimistically, considering these as occluded (value N). Additional control can be achieved through normalized weights which can be used to strengthen the occlusion states of the most recent visibility volumes.

When aggregating a set of visibility volumes $\{V^t, \cdots, V^{t-n}\}$, we need to take each camera position in V^t, express it in the basis of the n previous visibility volumes V^{t-i}, and apply the operator $\bigoplus_a$ over the extracted visibility states. The expression of a camera position $\mathbf{c} = [uvw]^T$ (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 position $\mathbf{c}$ in global Cartesian coordinates by applying the trilinear interpolation ($\mathbf{I}^t_{AB}(\mathbf{c})$), and (ii) re-express it in the local coordinates of V^{t-i}. That is :

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

The computation of the inverse of the trilinear interpolation $(\mathbf{I}^{t-i}_{AB})^{-1}$ requires to compute the solution of a cubic polynomial. Though the roots can be computed algebraically with Cardano's method, we propose a more efficient approach by projecting the transformed point $\mathbf{I}^t_{AB}(\mathbf{c})$ onto the rendering planes for A and B. The coordinates of the projected points can be remapped to pixel coordinates (u_A, v_A) and (u_B, v_B) on the rendering planes, which in turn can be mapped

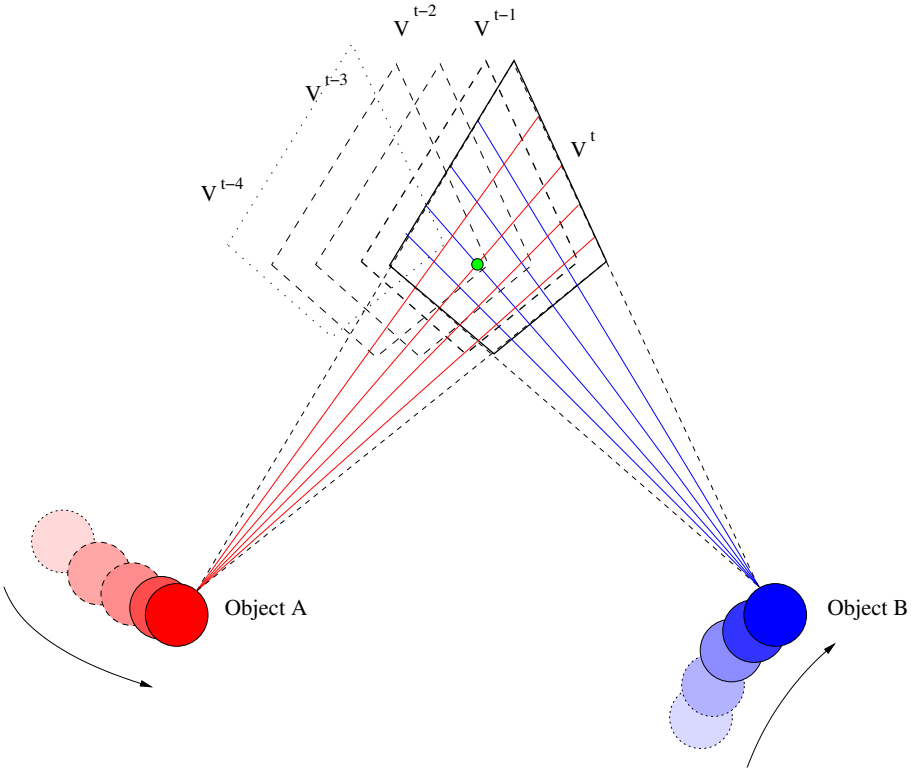

Fig. 7: *Stability is improved by aggregating the visibility information over the previous visibility volumes. The degree to which the camera reacts to occlusion is controlled by a ratio between the number of occluded states and the number of past frames to consider.*

into the 3D local coordinates $[u_B, v_B, -u_A]^T$ of $\mathbf{c}$ inside the visibility volume V_{AB}^{t-i} (see Section 3.1). From these coordinates, we get the visibility status.

In some situations, it is clear that a camera position $\mathbf{c}$ cannot be expressed in a previous visibility volume as these volumes evolve in space over time. Again we adopt a pessimistic approach, declaring the status of out-of-bound positions as occluded (value N). Moreover, the necessity to re-express a position $\mathbf{c}$ from one trilinear basis to another inevitably introduces an approximation.

We can now define the *temporal visibility volume* which stores the aggregations of the visibility states over a specified time interval. Experimental results, as well as example camera behaviour in the accompanying video, demonstrate the improved stability in situations that encompass partial occlusion and/or short-lived occluders (see Section 8). The complexity of this aggregation process is bounded by $O(ng^3)$ as we only aggregate for points inside the feasible camera volume (the sphere of reachable camera positions). In our implementation, the complexity is readily reduced to $O(g^3)$ by locally storing frame indices for each component visibility state.

5 Multi-object Visibility Evaluation

By using the Visibility Volumes, our approach can be seemlessly extended to multiple 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 8). As both volumes enclose the feasible camera volume, their intersection is not empty. The problem of combining the visibility information for the visibility volumes corresponding to each pair is closely related to the problem of accumulating visibility states over successive visibility volumes (as presented in Section 5) – though 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}	V

where, as before, P_A means that only target A is visible, and $P_{A,B,C}$ 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 completely visible configuration exists (*e.g.* always keeping the same object occluded instead of continuously jumping from one to another). The accumulation operator can be seamlessly applied to temporal visibility volumes.

6 Stochastic Estimation of Visual Extent

So far, our proposal addresses the problem of computing multi-object occlusion-free camera positions, but still depends on a point-based abstraction of a target object, and uses the visibility of these points as the basis on which to decide where to move the camera. We now need to consider how to capture the whole visual extent of the target object and reason as to its visiblity.

Techniques utilized in computer game environments, that build upon ray-casting, often select a number of points on a target object in computing its visibility. Although the range of these points helps to represent the visual extent of the object (*e.g.* choosing extremities such head, hands and feet) and the nature of the object (*e.g.* faces, torsos and guns are more salient than other features in the gaming context), the fixed nature of these points inadequately represents the visual extent of the object from different viewing angles and distances.

We propose a tessellation independent stochastic approach, that automates the choice of the target points on the surface of the object, and cycles between these target points on the surface of the object (choosing a different viewpoint for each frame).

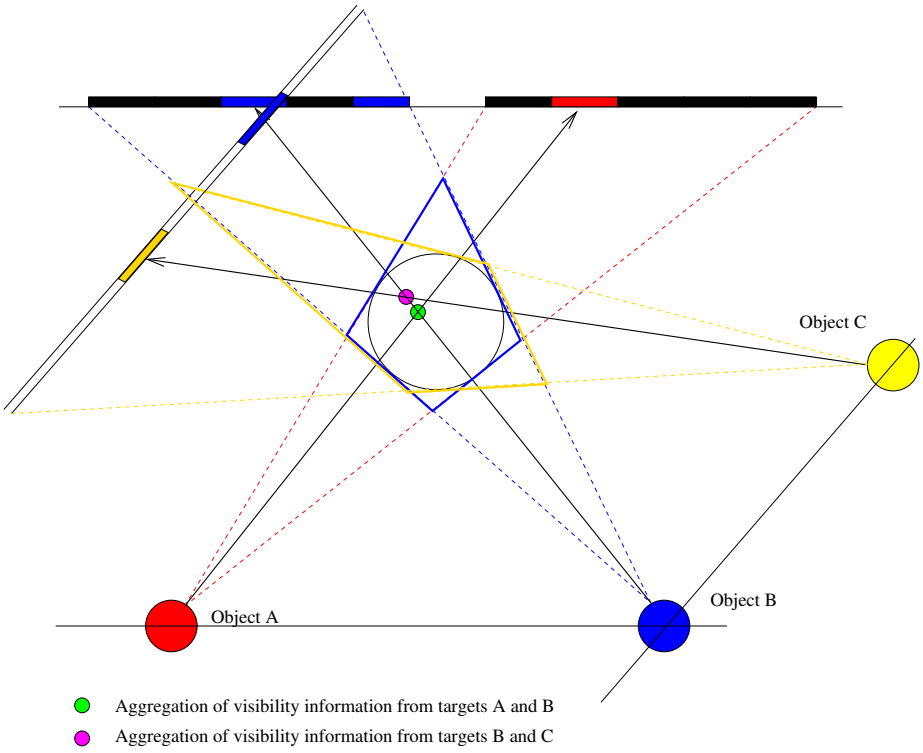

FIG. 8: *Visibility computation for three targets A, B and C. Two visibility volumes V_{AB} and V_{BC} are computed; each configuration of V_{AB} is expressed in the basis of V_{BC} to evaluate its visibility status.*

In a two-step process, we first pre-compute for each object, and for a number of different point of views, a random sample of points on the projected surface. These points approximate the visual extent of the target object from different viewpoints and are stored in a view indexed point sample table for the object. Next, in the course of computing the real-time visibility, we use the relative location of the target and the visibility volume to look up points in the sample table, and interpolate between these accordingly.

During the precomputation, an even distribution of camera configurations is computed around the target object, at different distances, from which we perform basic renderings (*i.e.* without lights using a unique color for the object). For each rendering we randomly choose a number of points on the image, keeping only those inside the projected extent of the object. For each of these points we then recompute their 3D position on the surface of the object by a simple re-projection (using the depth information). The density of the sampling 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 importance of different parts of the object (*e.g.* faces or guns for characters) can be rendered with specific colors for which higher sampling densities are applied. 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 9 illustrates the sampling process for a 3D character.

By considering 20 such points 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.

7 Practical Issues and Extensions

7.1 Failure recovery

In certain circumstances, such as highly cluttered environments or where there are 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 information and when necessary expanding the scope of the search for promising directions (*i.e.* potentially occlusion free) in which to move. Whenever the targets have been fully occluded over the last n frames, the radius of the feasible camera volume can be increased for each subsequent frame without increasing the resolution of the rendering buffers. This expands the search space of camera positions without increasing the cost. Once a possible configuration is found, it is set as a target for the future frames, and camera is set to the closest position to this target within its dynamic limits. As the scene evolves, the target is recomputed following the same principle. The radius must however be bounded by the minimum distance between the targets and the current camera position (above which the visibility volume is impossible to compute). In general, we have found this strategy helps to avoid awkward configurations, while a the same time maintaining a coherent view.

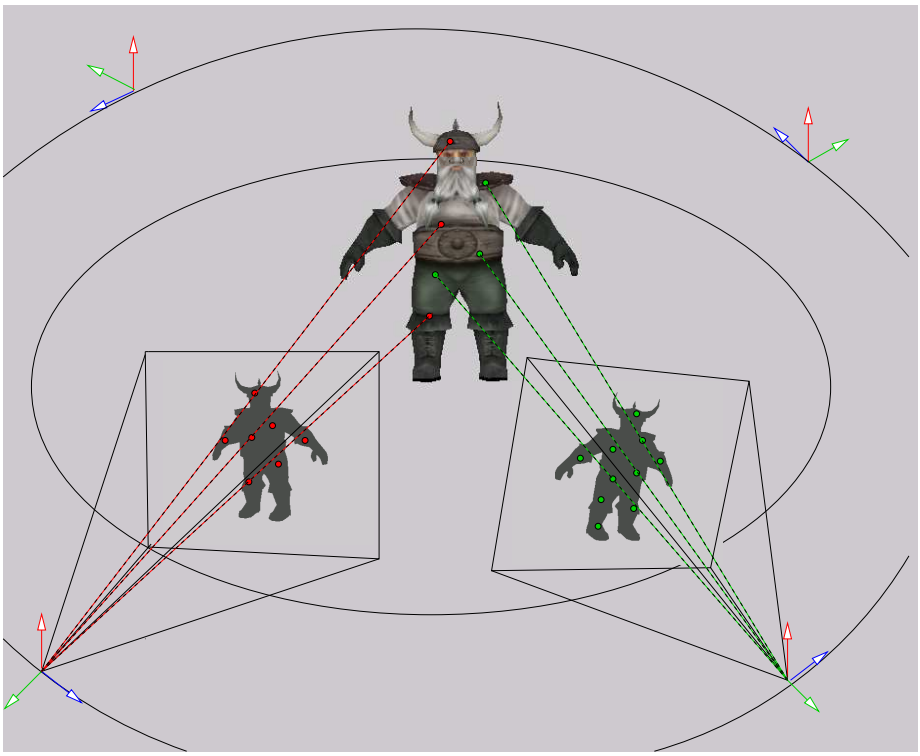

Fig. 9: *Stochastic estimation of the visual extents from distinct viewing angles and distances. For each viewpoint, 2D points are randomly scattered over the projected extent of the target and are projected onto the 3D shape. As the density of the sampling is constant in the 2D space, the size of the projected extent influences the number of sample points.*

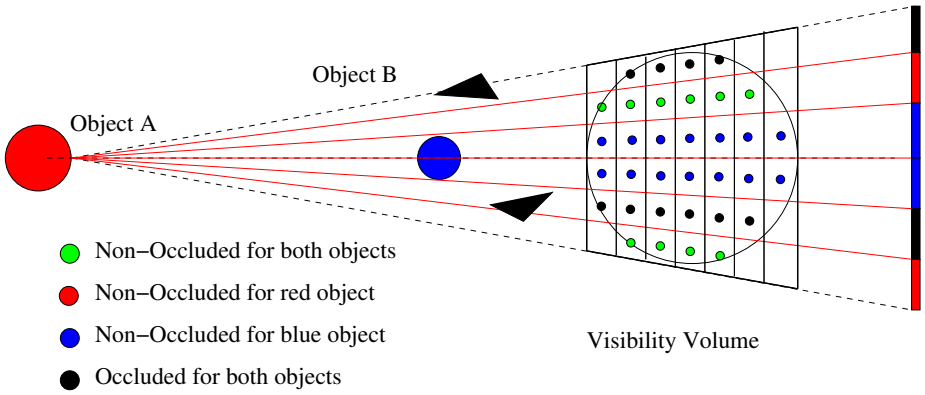

Fig. 10: *The specific case of mutual occlusion : only one projection is performed, and samples are automatically generated to compose a visibility volume.*

7.2 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 10). 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 δ this is trivial to construct.

7.3 Extension to compute orientation

Occlusion only requires us to consider camera location, not its orientation. A number of techniques, of which those proposed by the robotics community are best suited, can be incorporated to provide meaningful control over the camera's orientation. In particular, *visual servoing* can be used to maintain visual properties in screen space, such as the relative orientation of objects, screen velocities, and the positions of objects [MH98]. Where $\mathbf{P}$ is a set of visual properties on the screen, then in order to ensure the convergence of $\mathbf{P}$ to a desired value $\mathbf{P_d}$, we need to know the interaction matrix (the image Jacobian) $\mathbf{L}_\mathbf{P}^T$ that links the motion of the object in the image to the camera motion and orientation. Convergence is ensured by :

$$\dot{\mathbf{P}} = \mathbf{L}(\mathbf{P})\mathbf{T_c} \tag{1}$$

where $\dot{\mathbf{P}}$ is the time variation of $\mathbf{P}$ (the motion of $\mathbf{P}$ in the image) due to the camera motion $\mathbf{T_c}$ [ECR93]. By using Singular Value Decomposition (SVD) to compute the inverse of the image Jacobian ($\mathbf{L}^+$), we can obtain the camera motion that corresponds to the desired value $\mathbf{P_d}$:

$$\mathbf{T_c} = \mathbf{L}^+(\mathbf{P})\dot{\mathbf{P}}$$

As SVD requires at least a square matrix, and since we have 4 degrees of freedom available (rotation and aperture of the camera), we need the visual properties to be defined over 4 parameters (*e.g.* on-screen velocities of two points is sufficient).

8 Experiments

In order to highlight the various characteristics of our technique, a number of experiments are described. For each, we present a global view of the environment, with the camera path computed by our system. All evaluations were run on an Intel Core 2 T7600 at 2.33 Ghz, 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 evaluation, we have measured and report the following data :

- t_c the average time in ms spent computing an occlusion-free view (this encompasses 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 (which is a proxy for camera stability).
- d_s the distance covered in 2D by the projected centers of the target objects.
- r_N the proportion of fully occluded frames.
- r_P the proportion of partially occluded frames.

Processing times have been extracted using the `gprof` profiling tool. The costs related to the different steps of the visibility computation process are provided in Table 1 for a number of different resolutions. The size of the accumulation window has very little influence 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 the use of 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 strong bottleneck (see Table 1). However, with a low resolution buffer (7×7), the process can evaluate the state of more than 300 possible camera configurations (and chooses the best one) in less than 3 ms.

8.1 Evaluation of the stability of visual estimates

The stability of the visual estimates is illustrated using two target objects (cubes) and a sparse occluder that moves back and forth in front of the targets. The left frame in Figure 11 displays the camera path without any stabilising

Tab. 1: *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 calculations and (iii) next configuration selection.*

r	Samples	Rendering	Intersecting	Selection	Time
5	75	0.70	0.91	0.16	1.07
7	343	0.80	2.10	0.72	2.82
10	1000	0.80	5.20	1.81	7.03
15	3375	1.00	19.30	4.28	23.63

temporal window, and right frame shows the camera movement for a sliding window of size 10 and a threshold value $a = 6$ (meaning that states are aggregated when they appear at least 6 times in the past 10 frames). The sphere shows the feasible camera volume (according to the minimum and maximum acceleration). The trace of camera path is displayed as a white line in both figures. The left figure shows significant changes in the path as soon as the occluder hides a target object. The right figure shows the advantage of accumulation with the camera maintaining a stable position despite the temporary partial occlusion.

Table 2 quantifies the camera stability in the experiment. We can assess the impact of our stability heuristic by measuring both the global camera motion, using d_c the total distance covered by the camera, and the motion likely to observed by the viewer, using d_s the distance covered in 2D by the projected centers of the target objects. The additional cost of accumulating the visibility information can be seen in the difference in average time spent computing an occlusion-free view (t_c). For these experiments, the resolution of the buffers is set to $r = 7$.

Tab. 2: *Quantitative comparison of the stability of visual estimates by considering* 4 *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 occured at least* 6 *times in the past* 10 *frames.*

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

8.2 Escape strategy evaluation

The problem of escaping from situations where there is no occlusion-free position for the camera (within the dynamic limits of the camera) is illustrated using an environment in which the two targets are behind a plane in which there are two wholes, and the plane moves back and forth. The plane is constructed such

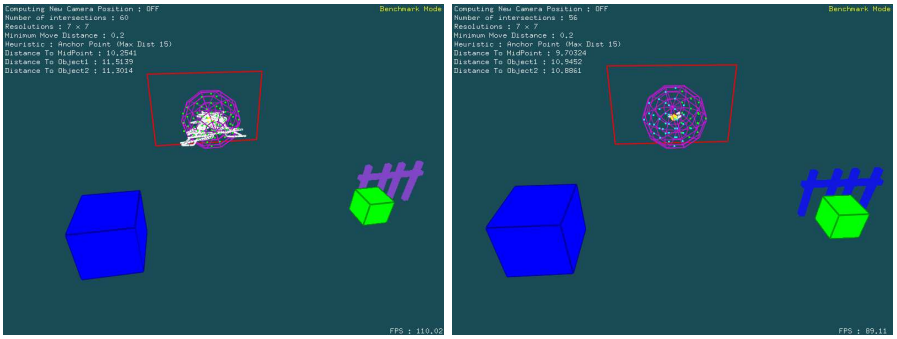

Fig. 11: *Stability evaluation – a sparse occluder moves back and forth in front of two targets. The left figure shows significant movement by the camera (white line), whereas the right frame shows how accumulating visibility states leads to significantly less camera movement.*

that for a number of positions of the plane there is no reachable location for the camera from which both target objects will be visible. The experiment consists of comparing the escape strategy to the default strategy of continuous camera motion (*i.e.* when both objects are occluded compute the next camera position on the basis of the current camera velocity and acceleration).

For each configuration, the number of frames where objects are both occluded is computed. Table 3 illustrates the benefits of this strategy. The escape strategy leads to a reduction in the proportion of frames in which the targets are both occluded (r_N) from 43% to 24% (a reduction of 44%) without any increase in the average time spent computing an occlusion-free view (t_c).

Tab. 3: *Quantitative comparison of the escape strategy; the ratio of fully occluded frames r_N has significantly decreased using the escape strategy in critical occlusion conditions.*

Strategy	t_c (ms)	r_N	r_P
No escape	2.80	43%	31%
Escape	2.82	24%	34%

8.3 Occlusion-free views for three objects

In Figure 12 we present the trace of camera location for three target objects in a game-like environment. The average time spent in the computation of occlusion-free views (t_c) is approximately 6 ms per frame, with a 7×7 resolution and stability set to 6/10. This is close to twice the average time required for two targets. The visual servoing technique related in section 7.3 is employed to control the camera orientation.

Fig. 12: *Following three targets in a moderate complex environment. We present the evolution of the samples over time ; green points represent fully visible camera locations, and red points, fully occluded ones. Only the motion of the target objects (in grey and white) are represented here (occluders hiding too much of the scene).*

9 Conclusion

Our proposed approach to maintaining occlusion free views addresses the fundamental problems of camera control for interactive graphics applications. The ability to track two or more target objects without the imposition of significant computational costs is a significant enhancement over both existing proposals from the research community and ray casting approaches widely deployed 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 means objects are no longer treated as point-like abstractions, avoiding the attendant anomalies of such approaches.

The ability to accumulate visibility information over time also provides parameterized control over the dynamic behaviour 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). We have demonstrated the utility of these enhancements over standard approaches to control. Of equal importance is the fact that since our approach is based on the generation of sets of sample camera positions, with different classes of visibility (within the dynamic limits of the camera), means few constraints are placed on its extension to incorporate other declarative aspects of camera planning. Indeed, we have demonstrated this through the incorporation of Jacobian based techniques to control camera orientation.

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, utilises ubiquitous graphics hardware, and can be readily incorporated into the rendering process of any real-time graphics application.

Références

[Ari76] D. Arijon. *Grammar of the Film Language*. Hastings House Publishers, 1976.

[BL99] W. H. Bares and J. C. Lester. Intelligent multi-shot visualization interfaces for dynamic 3D worlds. In *Proceedings of the 4th international conference on Intelligent user interfaces (IUI 99)*, pages 119–126, New York, NY, USA, 1999. ACM Press.

[BZRL98] W. H. Bares, L. S. Zettlemoyer, D. W. Rodriguez, and J. C. Lester. Task-sensitive cinematography interfaces for interactive 3D learning environments. In *Intelligent User Interfaces (IUI 98)*, pages 81–88, 1998.

[CM01] N. Courty and E. Marchand. Computer animation : A new application for image-based visual servoing. In *Proceedings of International*

Conference on Robotics and Automation, (ICRA 2001), pages 223–228, 2001.

[DZ95] S. M. Drucker and D. Zeltzer. Camdroid : A System for Implementing Intelligent Camera Control. In *Symposium on Interactive 3D Graphics*, pages 139–144, 1995.

[ECR93] Bernard Espiau, François Chaumette, and Patrick Rives. A new approach to visual servoing in robotics. In *Selected Papers from the Workshop on Geometric Reasoning for Perception and Action*, pages 106–136, London, UK, 1993. Springer-Verlag.

[Gio04] John Giors. The full spectrum warrior camera system. In *GDC '04 : Game Developers Conference 2004*, 2004.

[GKM93] Ned Greene, Michael Kass, and Gavin Miller. Hierarchical z-buffer visibility. In *SIGGRAPH '93 : Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 231–238, New York, NY, USA, 1993. ACM.

[HHS01] N. Halper, R. Helbing, and T. Strothotte. A camera engine for computer games : Managing the trade-off between constraint satisfaction and frame coherence. In *Proceedings of the Eurographics Conference (EG 2001)*, volume 20, pages 174–183. Computer Graphics Forum, 2001.

[HO00] N. Halper and P. Olivier. CAMPLAN : A Camera Planning Agent. In *Smart Graphics 2000 AAAI Spring Symposium*, pages 92–100, March 2000.

[Kat91] S. Katz. *Film Directing Shot by Shot : Visualizing from Concept to Screen*. Michael Wiese Productions, 1991.

[MC02] E. Marchand and N. Courty. Controlling a camera in a virtual environment. *The Visual Computer Journal*, 18(1) :1–19, 2002.

[MH98] E. Marchand and G.D. Hager. Dynamic sensor planning in visual servoing. In *Proceedings of the International Conference on Robotics and Automation (ICRA 98)*, volume 3, pages 1988–1993, Leuven, Belgium, May 1998.

[PBG92] C. B. Phillips, N. I. Badler, and J. Granieri. Automatic viewing control for 3d direct manipulation. In *Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 71–74. ACM Press New York, NY, USA, 1992.

[PD90] Harry Plantinga and Charles R. Dyer. Visibility, occlusion, and the aspect graph. *Int. J. Comput. Vision*, 5(2) :137–160, 1990.

[TTK96] Konstantinos Tarabanis, Roger Y. Tsai, and Anil Kaul. Computing occlusion-free viewpoints. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(3) :279–292, 1996.

[YHS95] Seungku Yi, Robert M. Haralick, and Linda G. Shapiro. Optimal sensor and light source positioning for machine vision. *Comput. Vis. Image Underst.*, 61(1) :122–137, 1995.

[YLPL05] Tao Yang, Stan Z. Li, Quan Pan, and Jing Li. Real-time multiple objects tracking with occlusion handling in dynamic scenes. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 970–975. IEEE, 2005.

Table des matières

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399