

# Camera Control in Computer Graphics

Marc Christie and Patrick Olivier

---

## Abstract

*Progress in modeling, animation and rendering means that rich, high fidelity interactive virtual worlds are now commonplace. But as photographers and cinematographers know, achievement of the intended informational and aesthetic goals is highly dependent on the position and motion of the camera in relation to the elements of the scene. Camera control encompasses interactive approaches, semi-automatic camera positioning, and fully declarative approaches to the management of a user's viewpoint on a scene. Camera control is required in nearly all interactive 3D applications and presents a particular combination of technical challenges for which there have been a number of recent proposals (e.g. specific path-planning, management of occlusion, modeling of high-level communicative goals). We present classify the approaches, analyze the requirements and limits of solving techniques and explore in detail the main difficulties and challenges in automatic camera control.*

---

## 1. Introduction

One concern of photography and cinematography is the capture and communication of information. Deciding where to position a camera (as a photographer) or how to move a camera (as a cinematographer) necessarily raises questions as to what information must be conveyed and how this is to be achieved. In an attempt to transpose both photographic and cinematographic techniques to virtual environments, the computer graphics community has proposed a set of approaches by which to interact with, and automate the positioning and motion control, of virtual cameras. Camera control is a central topic in a large range of computer graphics tasks and applications, including data visualization, virtual walk-throughs, proximal or distal object inspection, virtual storytelling and 3D games. However, in contrast to other major topics in computer graphics, camera control has received little attention from the research community. This can be explained both by the difficulty of formulating generic techniques and the intrinsic complexity of computing camera paths (which is a specific case of the PSPACE-hard problem of path-planning).

The importance of camera control in applications cannot be over-emphasized as a number of implicit rules drive the location and motion of cameras and impact the users mental representation and understanding of the environment. To date contributions in the field demonstrate a clear division between three classes of approaches: interactive approaches, reactive approaches and generalized approaches. Direct interactive approaches propose a set of mappings between the

dimensions of the user input device (mouse, keyboard or specific devices) and the camera parameters. The nature and complexity of the mappings is principally dependent on the targeted application. Reactive approaches apply and extend notions used in autonomous robotics and sensor planning and the behavior of the camera is driven in direct response to properties in the current image. Finally, indirect approaches reflect a move towards a higher-level control that aim to relieve users of the need to directly manipulation of parameters, and rely on a declarative characterization of the camera and path properties by reference to which the values of the parameters are derived. The range, nature and specificity of the properties characterize the expressiveness of the approach. The underlying solving techniques that translate properties into camera parameters then determine both the nature of the results and performance. We distinguish the following solving techniques:

- algebraic systems** represent the problem in vector algebra and directly compute a solution;
- reactive real-time systems** rely on robotic mechanisms or ad-hoc solving generally driven by a single objective that is targeting a focal object in a dynamic virtual environment;
- optimization and constraint-based systems** model image properties as constraints and objective functions and rely on a large range of solution techniques.

However, while most approaches have been developed and deployed in response to the specific requirements of an application domain, there are a number of common difficul-

ties including the number of degrees of freedom, the computational complexity related to any path-planning problem, and the evaluation and avoidance of occlusion.

Our presentation of the state-of-the-art in automatic camera control reflects the progression from interactive direct approaches to fully automated computation and emphasizes the principal challenges for camera control, such as management of occlusion and expressiveness. We open with a description of what motivates the need for camera control in a number of key applications in computer graphics. Next we briefly characterize photographic and cinematographic practice that in part serves as the inspiration a number of approaches. The following sections are related to the solving techniques, including: an overview of direct interactive approaches mainly based on interaction metaphors such as camera-in-hand and world-in-hand techniques, reactive techniques, and techniques related to assisted and automated camera control from restricted algebraic systems to generalized optimization and constraint-based systems. A detailed overview of the expressive power of camera control approaches is presented before we detail the techniques related to the critical problem of occlusion evaluation and avoidance.

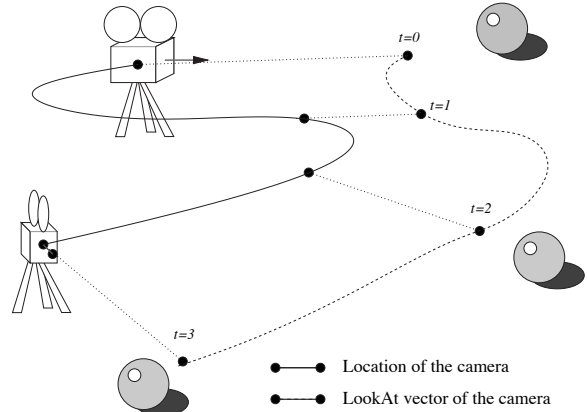
## 2. Motivations

Through an examination of the use and control of cameras in a number of classical applications we can identify the needs and issues related to modeling and solving camera control problems.

### 2.1. Conventional Modelers

Three dimensional modeling environments treat cameras and camera paths (usually splines) much the same as any other object. Virtual cameras are typically set up by providing a point which represents the location of the camera and two vectors that represents the *look-at* and *up* directions of the camera. Animation of the camera relies on classical interpolation methods such as the splines with key frames and/or control points. As illustrated in figure 1, the animation is computed by simultaneously specifying the location and the *up* and *look-at* vectors on their respective curves. The rotation matrix  $R$  in equation (1) is directly computed given the *up* vector of the camera. Fine control over the speed is provided by manipulating the velocity graphs pertaining to each curve.

A set of complementary tools allow artists to address a range of other requirements. For example, constraints can be added to fix each component of the *look-at* vector using a specific curve, tangent or to the position of a static or dynamic object in an environment environment (a tracking camera). For example, *Autodesk® Maya®* offers the possibility to interpolate the *look-at* vector between two or more



**Figure 1:** Canonical specification of a camera path in a 3D modeling environment.

points on two or more different curves. As the *look-at* vector is always centered on the screen, some modelers include offset parameters to shift the camera a small amount from the targeted object or path. Some specific metaphors are also proposed such as virtual rods that link the camera to an object to ease some target tracking issues. With the possibility to extend the functionalities of the modelers through scripting languages and plugins, new controllers on the camera are readily implemented (*e.g.* using physics-based systems). Another means of creating camera paths is to import positional sensor data for a real camera. Such approaches are more and more used as the techniques to merge real and virtual images are now robust and efficient.

In practice the naive use of modelers tend to produce stereotyped results as the underlying model (two spline curves) is not specific to the problem of modeling camera paths (that is, cameramen and the mechanical properties of real camera systems). Surprisingly, although cinematic practice has proposed a broad set of notions to describe camera positions, orientations and movement, most modelers have not attempted to explicitly incorporate such notions in their tools. For example, even simple functionality such as framing (maintaining a whole object in a user-defined frame) or maintenance of unoccluded views of a focal objects, are not apparent in commercial modeling environments.

Two considerations account for the absence of cinematic notions in current modelers. Firstly, this can be explained in terms of the generality that most modeling environments strive to achieve. Cinematic notions encompass strong references to character-based conversational shot compositions such as *over-the-shoulder shot*, *close shot* (a view a character's head) or *mid shot* (a view a character's head and torso). For a general purpose modeler to operate in such terms would in turn require the incorporation of semantic aspects to the representation of objects (which it might be

argued is likely in the near future, but is currently not the case). Furthermore, the problem of translating most cinematographic notions into controllers is a non-trivial as even the seemingly simple notion of a *shot* will encompass a large set of possible (and often distinct) solutions.

However, providing users with a high-level tools based on cinematic constructs for specification of, and interaction with, camera paths, would represent a significant advance over existing key-frame and velocity graphing controls. Even the integration of simple primitive paths such as travelings and arcs, and default (classical) camera rigs would significantly improve both the fidelity of the camera motion the interactive control afforded to the artist.

## 2.2. Games

Interactive 3D computer games serve the benchmark application for camera control techniques. Most importantly they impose the necessity for real-time camera control (*e.g.* supporting gameplay whilst following an unpredictable mobile character and simultaneously avoiding occlusions in highly cluttered complex environments). Furthermore, narrative aspects of real-time games are supported by judicious choices of shot edits both during and between (*i.e.* so-called *cut scenes*) periods of actual gameplay. It should however be noted that the camera system is only a small component of a game engine and only a very minimal (and time bounded) proportion of computation between successive frames can be devoted to camera control. Furthermore, the increasing geometric complexity of games in means that most deployed camera control algorithms in real-time 3D games rely upon fast (but fundamentally limited) heuristic occlusion checking techniques such as ray casting (see section 8 for a full discussion of occlusion).

Recent and planned console game releases demonstrate an increasing determination on the part of game studios and publishers to enhance the portrayal of narrative aspects of games and furnish players with a more cinematic experience (*e.g.* see the recently published previews of Killzone[] or Unreal Tournament 2007 [] on the PS3 and XBOX360 consoles). This move towards a more cinematic experience will necessarily rely on the rules and conventions of classical cinematography especially in games that are produced as a spin-off of a film itself where mirroring the cinematographic choices of the director is an important aspect of relating the game and the original product.

Despite the advent of near photorealistic graphics and the use of powerful and captivating story-driven plots, cameras in games have been neglected. On reflection, as John Giors (a game developer at Pandemic Studios) noted, "the camera is the window through which the player interacts with the simulated world". Film practitioners have characterized standard camera configurations and transitions in terms of a number of cinematographic principles[Ari76, Mas65, Kat91] that might be used

in games. The use of cinematographic properties and intelligent camera controls can heavily influence the look-and-feel of the game and the emotions evoked.

Camera shots are the heart of producing truly interactive visual applications. Games are inherently different to film in that the camera is usually either directly or indirectly controlled by the players (through their control of characters in the game), furthermore, they operate a dynamic environment. Therefore automated camera systems for games are considerably more complex to specify than the predominantly static camera positioning undertaken by film practitioners. Indeed game camera systems must be responsive to the action that takes place beyond the focal characters. While automation of camera shots based on cinematographic principles aim to present meaningful shots, the use of editing techniques (which are very rare indeed within games today) can preserve the gameplay by presenting jump-shots or cut-scenes to show the user only what is intended. These technologies have great potential to reduce the confusion that is evident in many games. Currently the use of automated editing and cinematographic techniques in games is rare, and where it is apparent is implemented using ad-hoc techniques.

The nature of a camera in a game can be broadly classified into three different categories:

- First person camera systems – users control the camera (giving them a sense of *being* the character in virtual environment and looking through the character's eyes). Many games use first person camera views, and the most common genre is the First Person Shooter (FPS), for example, the Doom and Quake series. Camera control is straightforward, since it is directly linked to the behavior of the character.
- Third person camera systems – here camera system tracks the characters from a set of fixed positions (generally the view is slightly above and behind the main character) and constantly modifies the camera's position and orientation based on local elements of the environment (to avoid occlusion) and the character's interactions (which are in turn controlled by the player). This mode of camera systems present a problem when the view fails to support current events in the game, *e.g.* when a character leans against a wall, such camera systems typically default to a front of the player, thereby disrupting the gameplay by effectively hiding the activity of opponents. Due to heuristic occlusion detection procedures (*e.g.* ray-casting) a common problem with the third person camera systems is the occlusion of the main character by adjacent scenery.
- Action replays camera systems – replays are widely used in modern games, particularly racing games or multicharacter games where there are significant events (*e.g.* for crashes in driving games and goals in football games). To highlight notable events it is imperative that the images generated by the camera system during the replay are meaningful. Some games also provide "in-game replays"



Figure 2: In-game screenshot of a Burnout 3 instant replay.

for which the demands are particularly onerous (*i.e.* the need to produce real-time and visually appealing summary of action as it occurs).

The problem of positioning a camera in a 3D virtual environment is a key challenge for most third person computer games, such as Prince Of Persia, World Of Warcraft and many others. The successful Tomb Raider series of computer games has been the subject of much discussion in regard to its choice of camera shots. Tomb Raider employs a third-person view for the main character, in which the camera is attached to the character, although the camera system employed was rather limited in expressing informative shots when in so called *tight spots*. This often lead to situations where the views were portrayed, significantly hindering user from playing the game as it was intended. In simple terms the Tomb Raider camera system computed its next best position without consideration of the visual properties and the ongoing action within the environment. For example Fig. 2.2 illustrates two screenshots taken from Tomb Raider: Angel Of Darkness including a *good* (*i.e.* over-the-shoulder) and a *bad* (*i.e.* front) view for Lara Croft's confrontation with an opponent.

Full Spectrum Warrior (*cf.* [Gio04]), an action war military simulator developed at Pandemic Studios, uses a more advanced camera system that facilitates a player in managing teams of soldiers. A key element is the *auto look* feature which helps the user by presenting a shot that handles occlusion to prevent the view from being blocked by an object (*e.g.* wall) through the use of ray casting. The fly-by scenes performed by the camera also avoids colliding into environmental objects by applying the same occlusion detection method. Jump cuts are used to handle situations when the only evident path is to move through a wall or an obstacle, whereby the camera jumps to the scene beyond the wall, avoiding the unnatural occurrence of many games in which cameras pass directly through solid obstacles. Full Spectrum



(a) Bad: shot from the front of Lara Croft.



(b) Good: shot from behind Lara Croft.

Figure 3: Tomb Raider: Angel Of Darkness.

Warrior is an example of a relatively advanced architecture for a game camera system, managing multiple cameras simultaneously and independently, and using the most appropriate camera based on the viewing context. Full Spectrum Warrior notable progression in the use of cameras in games, but still lacks the cinematic qualities that are apparent in film.

### 2.3. Multimodal Systems and Visualization

The generation of multimodal output (in particular natural language and graphics) involves careful coordination of the component modalities. Typically such systems have been developed in the domain of education and training and in particular need to address the problem of coordinating the choice of vantage point from which to display the objects being described or referred to linguistically.

For example, a direct linguistic reference to an object (*e.g. the handle on the door*) usually requires that the object (*i.e. the handle*) is no more than partially occluded in the shot. To satisfy such coordination constraints, multimodal

generation systems have relied heavily of the use of default viewpoints [SF91] from which unoccluded views of the elements of discourse are likely to be achieved, and have used ray-casting to trivially accept or reject viewpoints (although [BRZL98] address the application of constraint-based camera planning in the development of intellimedia). Alternative approaches use cutaways and ghosting, standard devices in engineering graphics, by which occluding elements of scene are removed either by direct surgery on the polygons, manipulation of the depth buffer [SF93] or object transparency.

Beyond simple object references, the coordination of language and graphics poses a number of interesting problems for camera control and planning. Indeed, such applications offer a rich source of constraints on the control of a camera as the subtle uses of spatial language can only be effectively interpreted by reference to an appropriate perspective. For example, descriptions involving spatial prepositions (e.g. *in front of*, *left of* etc) and dimensional adjectives (e.g. *big*, *wide* etc) assume particular vantage point and on-screen properties. For example, for projective prepositions the choice of deictic or intrinsic reference frame for the interpretation of a preposition such as *in front* directly depends on the viewpoint of a hypothetical viewer.

Visualisation systems are by contrast conceptually more straight forward. Multidimensional data sets may be mapped to different 3D spatial entities in an effort to furnish users with an intuitive an interactive framework to explore the underlying relations. Unfortunately, such data sets, and the resulting visualisations, are often vast landscapes of geometry for which manual interactive control is extremely difficult. By its very nature visualisation is an application for for the user requires interactive control both to explore and pursue hypotheses in the data. However, user behavior, in such applications, often reduces the problem to a number of navigational idioms, for example, the identification of a number of *interesting* points or regions in the data, and the exploration of the remaining data in relation to these. Automatic camera control, or at least augmented manual control, is likely to greatly enhance the use of such strategies.

In practice, even partially automated natural language generation, that is to be coordinated with 3D renderings of a domain, requires an interpretation and synthesis framework by which both the visuospatial properties of a viewpoint can be inspected (i.e. the interpretive framework) and viewpoint control according to the constraints arising from the semantics of the language used (i.e. the synthesis framework). Likewise, future scientific and information visualisation systems will benefit greatly from intelligent camera control algorithms that are sensitive to both the underlying characteristics of the domain and the task that the user is engaged in. Such adaptivity presupposes an ability to evaluate the perceptual characteristics of a viewpoint on a scene and the capability to modify it in a manner that is beneficial to the user.

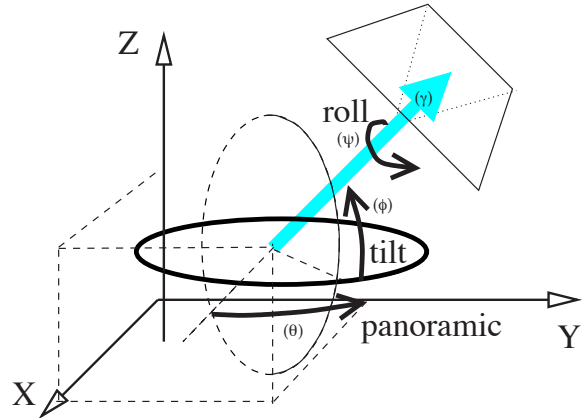


Figure 4: A simple camera model based on Euler angles.

## 2.4. Main Issues

All three application domains help us to identify the main difficulties in developing good camera control tools. First of all manipulating a virtual camera is generally a delicate task; users cannot deal simultaneously with all seven degrees of freedom. A simple pin-hole camera can be modeled using extrinsic parameters, three degrees of freedom for Cartesian coordinates, three Euler angles, and one intrinsic parameter, the focal distance (see Fig. 4). Approaches assist the instantiation of the camera parameters, either partially, in interactive applications by providing mappings through metaphors that link the user's inputs and the camera parameters, or totally, for automated presentations. Both offer solutions which are specific to a given task or set of tasks that limit any generalization.

The second issue lays in the complexity of the problem. Virtual camera control can be considered as a special case of path planning and is thus a PSPACE-hard problem in which complexity is exponential in the number of degrees of freedom. Moreover, the quality of a camera composition, although dependent of the targeted application is identified by the content of the screen, that is, what the information on the screen is and how it is laid out and organized. The mathematical relation between an object in the 3D scene and its projection on the 2D screen is strongly non-linear. If we consider a Euler-based camera model (see Fig. 4) for which parameters are  $\mathbf{q} = [x_c, y_c, z_c, \phi_c, \theta_c, \psi_c, \gamma_c]^T$ , the projection can be expressed by equation 1. This expresses the transformation of a point  $M = [x, y, z, t]^T$  in world coordinates to a point  $M' = [x', y']^T$  in screen coordinates through multiplication with matrix  $H(\mathbf{q})$ . This relation is expressed as a change from the world basis to the local camera basis: with a rotation matrix  $R(\phi_c, \theta_c, \psi_c)$  a translation matrix  $T(x_c, y_c, z_c)$  and a projection through matrix  $P(\gamma_c)$ .

$$\begin{aligned}
 \begin{pmatrix} x' \\ y' \end{pmatrix} &= P(\gamma_c).R(\phi_c, \theta_c, \psi_c).T(x_c, y_c, z_c). \begin{pmatrix} x \\ y \\ z \\ t \end{pmatrix} \\
 &= H(\mathbf{q}). \begin{pmatrix} x \\ y \\ z \\ t \end{pmatrix}
 \end{aligned} \tag{1}$$

The strong non-linearity of this relation makes it difficult to invert, *i.e.* to decide where to position the camera knowing the location of an object in the world and on the screen. Moreover, one must be able to reason about whether the object we are looking is hidden, either partially or completely, by any other object. The occlusion problem is complex and fundamental to virtual camera planning and will ultimately receive considerable attention via the solving processes.

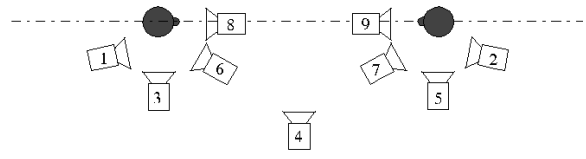
As a final and fundamental goal, we need to convey more than a simple layout of objects on the screen. The third issue we identify lays in the expressiveness of the approaches: how to model the geometric, perception and aesthetic properties in the picture. We need to convey a proper understanding of the spatial configuration of the scene, of the temporal configuration of events and of the causality between the events in order to assist users in their mental construction and understanding of the spatial environment.

### 3. Camera Control and Cinematography

Direct insight in the use of real-world cameras can be found in reports of photography and cinematography practice [Ari76, Mas65, Kat91]. Cinematography encompasses a number of issues in addition to camera placement including shot composition, lighting design and staging (the positioning of actors and scene elements) and an understanding of the requirements of the editor. For fictional film and studio photography camera placement, lighting design and staging are significantly interdependent. However, documentary cinematographers and photographers have little or no control over staging and we review accounts of camera placement in cinematography with this in mind. Indeed, real-time camera planning in computer graphics applications (*e.g.* computer games) is analogous to documentary cinematography whereby coherent visual presentations of the state and behavior of scene elements must be presented to a viewer without direct modification of the elements.

#### 3.1. Camera positioning

Whilst characterizations of cinematography practice demonstrate considerable consensus as to the nature of best practice, there is considerable variation in articulation. On the one hand accounts such as Arijon's systematically classify components of a scene (*e.g.* according to the number of



**Figure 5:** Idiom for camera placement for a two person face-to-face conversation.

principal actors) and enumerate appropriate camera positions and shot constraints [Ari76]. Not surprisingly, Arijon's procedural account of camera placement has impacted on the specification of a number of existing automatic camera planning systems. By contrast accounts such as Mascelli's [Mas65] provide less prescriptive characterizations in terms of broader motivating principles, such as narrative, spatial and temporal continuity.

It is generally considered that camera positioning for dialogue scenes can be explained in terms of broadly applicable heuristics. For example, Arijon's triangle principle invokes the notion of a line of action which for single actors is determined by the direction of the actors view and for two actors is the line between their heads. Camera positions are selected from a range of standardized shots (see figure 5), internal-reverse (6 and 7), external-reverse (1 and 2), perpendicular (3, 4 and 5) and parallel configurations (8 and 9). By ensuring that camera placements are chosen on one side of the line of action, it can be assured that viewers will not be confused by changes in the relative positions or the direction of gaze of the actors. In fact, there are a wide range of two actor configurations that vary in respect of the actors' relative horizontal position (*e.g.* close together, far apart), orientations (*e.g.* parallel, perpendicular), gaze direction (*e.g.* face-to-face, back-to-back) and posture (*e.g.* sitting, standing, lying down). As a result Arijon enumerates a large number of sets of standard camera positions, and extends the principles for filming two actors have been to three or more actors in various spatial configurations.

#### 3.2. Shot composition

Camera positioning ensures the general spatial arrangement of elements of the scene with respect to the camera, thereby placing a coarse constraint on the composition of a shot. That is, the position (and lens selection) determines the class of shot that is achievable, which can be broadly classified accordingly to the amount of the subject included in the shot as: close up (*e.g.* from the shoulders), close shot (*e.g.* from the waist), medium shot (*e.g.* from the knee), full shot (*e.g.* whole body) and long shot (*e.g.* from a distance). However, precise placement and orientation of the camera is critical to

achieving the layout of the scene elements in shot — referred to as the composition of the shot.

Composition is variously characterized in terms of shot elements including lines, forms, masses, and (in the cases of action scenes) motion. In turn, shots are organized to achieve an appropriate (usually single) center of attention, appropriate eye scan, unity, and finally compositional balance (arrangements of shot elements that affords a subconsciously agreeable picture). As psychological notions these terms are problematic to define and the empirical characterization of visual aesthetics is in its infancy. This is not question the validity or significance of the notions themselves, indeed eye tracking studies have demonstrated significant differences between the viewing behavior of observers of subjectively agreed balanced and unbalanced (through artificial modification) works of art [NLK93].

At a practical level Mascelli observes a number of compositional heuristics, for example, that “real lines should not divide the picture into equal parts” and that “neither strong vertical nor horizontal line should be centred” [Mas65]. He further categorizes forms of balance into formal (symmetrical) and informal (asymmetrical) balance. Indeed where scene objects interfere with the composition of a shot, in particular in close-ups, such objects are frequently removed for the duration of the shot. Note also that elements to be balanced do not necessarily relate to physical objects alone, critical aspects of a composition might include abstract notions such as the line of fire or direction of gaze of an actor. Composition is also constrained by the requirement to produce coherent and continuous cuts between cameras, for example, ensuring that consecutive shots have a focus of interest that is roughly collocated.

Scenes that comprise significant amounts of motion and action pose different problems for cinematographers and editors, although general heuristics such as the triangle principle, use of a line of action, and compositional rules can be extended to these more complex configurations. The challenge for camera planning is to algorithmically formulate these principles in a manner appropriate to the particular application to be addressed.

#### 4. Interactive Control

Interactive control systems provide the user with a view of a model world and modify the camera set-up in response to directions from the user. The principal issue with such systems is how an input device will map onto the properties of the camera. Ware and Osborne [WO90] published a review of possible mappings, which they referred to as camera control metaphors, and covers a broad range of approaches:

- camera in hand: the camera is directly manipulated as if it were in the user’s hand which encompasses rotational and translational movements.
- world in hand: the camera swivels around the world while

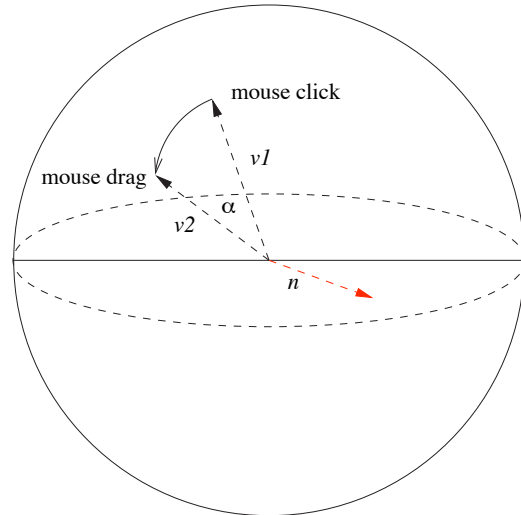


Figure 6: Shoemake’s Arcball principle for interactive visualisation.

shooting a fixed point – generally the center of the world. A left movement on the user’s device corresponds to a right movement of the scene.

- flying vehicle: the camera can be treated as a airplane. This metaphor is intuitive and has been used extensively to explore mainly large environments.

In addition is the *walking* metaphor [HW97] whereby the camera moves in the environment while maintaining a constant distance (height) from the floor.

The *world in hand* metaphor is restricted to explorations of details of an object or a group of objects (proximal exploration). Such interactive control was developed by Phillips [PBG92] for the human figure modeling system *Jack*. Despite its intuitive nature, the *Jack* system could not properly support manipulation of model figures about axes parallel or perpendicular to the view direction. The camera system prevented this from occurring by repositioning the camera. It could also reposition the camera to make a selected object visible if it was off screen. The system could also find positions from which the selected object was unoccluded. This was implemented by placing a non-viewing camera with a fish eye lens at the center of the target object looking towards the current viewing camera, then rendering. The z-buffer was examined and regions which were set to the far clip plane were mapped onto a hemicube, which was then used to select a camera position.

For similar applications, Shoemake[Sho92] introduced the concept of arcball, a virtual ball that contains the object to manipulate. His solution relies on quaternions to stabilize the computation and avoid Euler singularities while rotating around an object (gimbal lock). Occlusion is not addressed. Figure 6 presents the arcball principle. As the user drags the

mouse on the screen, the intersection between a unit sphere – the arcball – and a ray directed from the mouse pointer to the center of the sphere is computed and leads to vector  $\mathbf{v1}$ . With each user mouse movement, vector  $\mathbf{v2}$  is computed. The rotation of the scene is then computed by the quaternion  $\mathbf{q} = (\sin\theta, \cos\theta\mathbf{n})$  composed of the axis  $\mathbf{n} = \mathbf{v1} \times \mathbf{v2}$  and angle  $\theta = \mathbf{v1} \cdot \mathbf{v2}$ . This approach is in use in numerous modeling tools. For a more detailed overview of possible rotation mappings see Chen *et al.*'s [CMS88] study of 3D rotations using 2D input devices.

An approach that encompasses both rotation and translation mappings, Khan *et al.* [KKS\*05] propose an interaction technique for proximal object inspection that relieves the user from much of the camera control. In simple terms, the approach tries to maintain the camera at both a fixed distance around the object and (relatively) normal to the surface, obeying a hovercraft metaphor. Thus the camera easily turns around corners and pans along flat surfaces, while avoiding both collisions and occlusions. Specific techniques are devised to manage cavities and sharp turns. A similar approach has been proposed by Burtnyk *et al.* [BKF\*02] in which the camera is constrained to a surface defined around the object to explore (as in [HW97]). The surfaces are however restricted to *interesting* view points of the object that will guarantee a certain level of quality in the user's exploration experience. Automated transitions are built between the edges of the different surfaces.

The *flying vehicle* metaphor is popular within the computer graphics community as it is an obvious and intuitive way to explore large 3D environments that arise in scientific visualization, virtual exploration and virtual museums guide tours. The main problem lies in avoiding the *lost in space* problem encountered when the user has multiple degrees of freedom to manage in either highly cluttered environments or in an open space with a few visual landmarks. Work in this area tends to concentrate on assisting the control of the camera parameters to reduce the dimensionality of the problem, and the application of different physically-based models, vector fields or path planning to constrain possible movement and avoid obstacles.

For example, the application of a physical model to camera control has been explored by Turner *et al.* [TBGT91]. User inputs are considered as forces acting on a weighted mass – the camera – and the approach incorporates notions such as friction and inertia to ease the simultaneous control of all the degrees of freedom. Turner's approach is easily extended to manage any new set of forces and has inspired approaches that rely on vector fields to guide the camera parameters. Given knowledge of an environment the process consists of computing a grid of vectors (forces) that influence the camera. The vectors keep the user away from cluttered views and confined spaces, as well guiding him towards the objects of interest. Hanson *et al.* in [HW97] and Xiao and Hubbard in [XH98] propose good illustrations of this approach that

emphasize obstacle avoidance and gravity without any prior reference to semantic knowledge of the virtual environment.

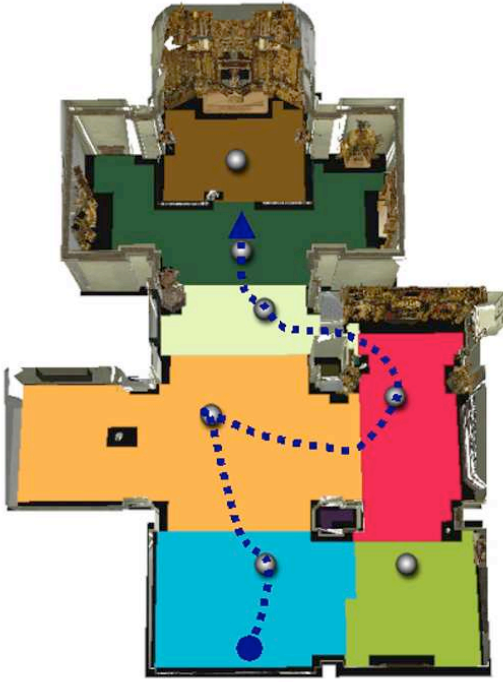
Applications of camera control to the exploration of complex environments requires specific approaches that are highly related to the more general problem of path-planning while still ensuring the continuity, smoothness and occlusion criteria. Applications are found both in navigation (looking for a precise target) and in exploration (gathering knowledge in the scene). Path planning problems in computer graphics have mostly been inspired by robotics utilizing techniques such as potential fields, cell decomposition and roadmaps.

Potential fields originated from theoretical physics and the study of charged particle interactions in electrostatic fields. Positive particles form peaks while negative particles form wells. A path-planning problem is consequently modeled by considering that obstacles are represented by peaks and targets by wells. The solving process is based on a series of local moves following the steepest descent [Lat91]. The efficiency of the method is however overshadowed by its limitations with respect to the management of local minima as well as difficulties incorporating highly dynamic environments. Nonetheless, some authors have proposed extensions such as Beckhaus [Bec02] who relies on dynamic potential fields to manage changing environments by discretising the search space using a uniform rectangular grid and therefore only locally recomputing the potentials.

Cell decomposition approaches split the environment into geographic regions (cells) and build a network that connects the regions. Navigation and exploration tasks therefore follow this cell connectivity while enforcing other properties on the camera. For example, [AVF04] proposes a cell decomposition to compute possible camera paths coupled with an entropy-based measure of the relevance of the viewpoints (see Fig. 7) to identify the critical way-points. The objective is to ease the navigation process and avoid missing relevant entities and places.

Roadmap planners operate in two phases: first the planner samples the space of possible configurations, and then builds a connectivity graph linking possible consecutive configurations. Probabilistic roadmap approaches, in which the samples are randomly chosen in the environment, have been successfully used in a number of contexts. Li and Ting [LT00], for example, compute collision-free paths to correct a user's input. [SGLM03] describes a related approach in which an expensive global roadmap enables collision-free and constrained paths to be built for an avatar that navigates in an environment. Nieuwenhuisen and Overmars provide a detailed discussion of the application of robotic techniques to planning camera movements [NO03].

A small but active field for which camera path-planning techniques are vital is virtual endoscopy. Virtual endoscopy enables the exploration of the internal structures of a patient anatomy. Difficulties arise in the interactive control of the



**Figure 7:** Cell decomposition and path planning (courtesy of Andujar, Vázquez and Fairen, Universitat Politècnica de Catalunya).

camera while maintaining it inside the structures, emphasizing details of the anatomy, and avoiding significant occlusion or confined spaces. The underlying techniques mostly rely on skeletonization of the structures and on path planning approaches or potential fields. For example, [HMK\*97] and [CHL\*98] report a technique that avoids collisions for guided navigation in the human colon. The surfaces of the colonic are modeled as repulsive fields and center line of the colon as an attractive field.

Most of the approaches discussed so far involve metaphors that address either close object inspection or large environment exploration. However, a number of techniques have concentrated on transitions between these metaphors to enable multiple interaction techniques. As early as 1992, Drucker *et al.* [DGZ92] proposed CINEMA, a general system for camera movement. CINEMA was designed to address the problem of combining the different paradigms (*e.g. eyeball in hand, scene in hand, or flying vehicle*) for controlling camera movements. It also provides a framework on which the user can develop new paradigms via a procedural interface by specifying camera movements relative to objects, events and general information on the virtual environment. Zeleznik in [ZF99] demonstrates the utility of this approach

by proposing smooth transitions between multiple interaction modes with simple gestures on a single 2D device.

Early work from Mackinlay [MCR90] provided a natural transition for movements between a set of targets allowing them to be inspected closely. This involved a three stepped process between two targets: to first view the target, then move the camera towards it at a speed relative to the distance to the target (the further away the target, the faster the movement towards it) and finally swivel around the object to propose the best view. A helicopter metaphor has been proposed by Jung *et al.* [JPK98] as a mapping function in which transitions are facilitated by a representation of the degrees of freedom as 2D planes and easily executed transitions between planes with the 2D input device. More recently, Tan *et al.* in [TRC01] utilised the locations of users' mouse dragging operations to alternate between walking around, overview and close object examination interaction metaphors. Li and Hsi [LH04] explored two adaptive methods that take the user into account and search for a personal optimal set of control parameters to improve navigation (see [BJH99] for a taxonomy of interaction techniques and evaluations).

Whilst many of these approaches go some way towards easing interactive control of camera parameters, none are concerned with a precise control of the movement of objects in the screen. The *Through The Lens Camera Control* approach devised by Gleicher and Witkin [GW92] allows the user to control a camera by manipulating the locations of objects directly on the screen. A recomputation of new camera parameters performed to match the user's desired locations. The difference between the actual screen locations and the desired locations indicated by the user is treated as a velocity and the relationship between the velocity ( $\mathbf{h}$ ) of  $m$  displaced points on the screen and the velocity ( $\dot{\mathbf{q}}$ ) of the camera parameters can be expressed through the Jacobian matrix that represents the perspective transformation:

$$\mathbf{h} = J\dot{\mathbf{q}}$$

Gleicher and Witkin propose to solve the non-linear optimization problem which minimizes a quadratic energy function  $E = \frac{1}{2}(\dot{\mathbf{q}} - \dot{\mathbf{q}}_0) \cdot (\dot{\mathbf{q}} - \dot{\mathbf{q}}_0)$  that represents a minimal change in the camera parameters ( $\dot{\mathbf{q}}_0$  representing the values of the camera's previous velocity). This problem can be converted into a Lagrange equation and solved for the value of  $\lambda$ :

$$\frac{dE}{d\dot{\mathbf{q}}} = \dot{\mathbf{q}} - \dot{\mathbf{q}}_0 = J^T\lambda$$

where  $\lambda$  stands for the vector of Lagrange multipliers. The velocity of the camera parameters is thus given by:

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_0 + J^T\lambda$$

A simple Euler integration allows us to approximate the next location of the camera from the velocity  $\dot{\mathbf{q}}$ :

$$\mathbf{q}(t + \Delta t) = \mathbf{q}(t) + \Delta t \dot{\mathbf{q}}(t)$$

The result is that the rate of change of the camera set-up is proportional to the magnitude of the difference between the actual screen properties and the desired properties set by the user. When the problem is overconstrained (*i.e.* the number of control points is higher than the number of degrees of freedom) the complexity of the Lagrange process is  $O(m^3)$ .

This formulation has been improved and extended by Kung, Kim and Hong [KKH95] with the use of a single Jacobian Matrix. A pseudo inverse of the matrix is computed with the Singular Value Decomposition (SVD) method for which the complexity is  $O(m)$ . The SVD method enjoys the property that the pseudo inverse always produces a solution with the minimal norm on the variation of  $\mathbf{q}$

This section has reviewed a number of possible mappings between the user input and camera parameters. What can clearly be seen is a progression towards high-level interaction metaphors that lay the groundwork for the process of automating the placement of a camera. Nonetheless, substantial work must be provided to express the user's desires (in terms of the qualities of a shot) and to translate these into camera parameters.

## 5. Reactive systems

Classical path-planning approaches are highly appropriate for exploration and navigation in large static environments. However, it is a very common in interactive 3D graphics applications that one wants to follow one or more mobile targets. In robotics visual servoing approaches (also called image-based camera control) [ECR93] are widely deployed in such contexts. Visual servoing relies on the specification of a task (mainly positioning or target tracking tasks) as the regulation in the image of a set of visual features. In [CM01], the authors propose a visual servoing approach that integrates constraints in the camera trajectory in order to address various non-trivial computer animation problems. A Jacobian matrix expresses the interaction between the movement of the object on the screen and the movement of the camera. The solving process consists in computing the possible values of the camera speed in order to satisfy all the properties. Let  $\mathbf{P}$  be the set of visual features used in the visual servoing task. To ensure the convergence of  $\mathbf{P}$  to its desired value  $\mathbf{P}_d$ , we need to know the interaction matrix (namely the image Jacobian)  $\mathbf{L}_P^T$  that links the motion of the object in the image to the camera motion. The convergence is ensured by [ECR93]:

$$\dot{\mathbf{P}} = \mathbf{L}_P^T(\mathbf{P}, \mathbf{p})\mathbf{T}_c \quad (2)$$

where  $\dot{\mathbf{P}}$  is the time variation of  $\mathbf{P}$  (the motion of  $\mathbf{P}$  in the image) due to the camera motion  $\mathbf{T}_c$ . The parameters  $\mathbf{p}$  involved in  $\mathbf{L}_P^T$  represent the depth information between the considered objects and the camera frame. A vision-based task  $\mathbf{e}_1$  is defined by :

$$\mathbf{e}_1 = \mathbf{C}(\mathbf{P} - \mathbf{P}_d)$$

where  $\mathbf{C}$ , called the combination matrix has to be chosen such that  $\mathbf{C}\mathbf{L}_P^T$  is full rank along the trajectory of the tracked object. If  $\mathbf{e}_1$  constrains all 6 degrees of freedom of the camera, it can be defined as  $\mathbf{C} = \mathbf{L}_P^{T+}(\mathbf{P}, \mathbf{p})$ , where  $\mathbf{L}^+$  is the pseudo inverse of the matrix  $\mathbf{L}$ . The camera velocity is controlled given the following relation:  $\mathbf{T}_c = \lambda\mathbf{e}_1$  where  $\lambda$  is a proportional coefficient.

If the primary task (following the object) doesn't instantiate all the camera parameters when solving equation 2, secondary tasks may be added (*e.g.* avoiding obstacles or occlusions, lighting optimisation, *etc.*).  $\mathbf{C}$  is then defined as  $\mathbf{C} = \mathbf{C}\mathbf{L}_P^T$  and we obtain the following task function:

$$\mathbf{e} = \mathbf{W}^+\mathbf{e}_1 + (\mathbf{I}_n - \mathbf{W}^+\mathbf{W})\mathbf{e}_2 \quad (3)$$

where

- $\mathbf{e}_2$  is a secondary task. Usually  $\mathbf{e}_2$  is defined as the gradient of a cost function  $h_s$  to minimize ( $\mathbf{e}_2 = \frac{\partial h_s}{\partial \mathbf{r}}$ ), which is minimized under the constraint that  $\mathbf{e}_1$  is realized.
- $\mathbf{W}^+$  and  $\mathbf{I}_n - \mathbf{W}^+\mathbf{W}$  are two projection operators which guarantee that the camera motion due to the secondary task is compatible with the regulation of  $\mathbf{P}$  to  $\mathbf{P}_d$ .

Given a judicious choice of matrix  $\mathbf{W}$ , the realization of the secondary task will have no effect on the vision-based task, *i.e.*  $\mathbf{L}_P^T(\mathbf{I}_n - \mathbf{W}^+\mathbf{W})\mathbf{e}_2 = 0$ . This feature ensures that adding secondary tasks cannot affect the results obtained for the primary task and cannot invalidate the solutions. Secondary tasks as proposed in [MH98, MC02] range from tracking another mobile object, avoiding obstacles or occlusions, or more cinematographic notions (computing panning, traveling, optimizing lighting conditions, *etc.*). Specifying a secondary task comes down to defining a minimization function  $h_s$ . For example, obstacle avoidance can be handled by a cost function that will express the inverse of the distance between the camera and the obstacle, ideally the maximum (infinite) cost should be reached when the distance between the camera and the obstacle is null. The simplest cost function associated to obstacle avoidance could therefore be defined as follows:

$$h_s = \alpha \frac{1}{2\|C - O_c\|^2} \quad (4)$$

where  $C(0,0,0)$  is the camera location and  $O_c(x_c, y_c, z_c)$  are the coordinates of the closest obstacle to the camera (see [MC02] for more examples of possible cost functions associated to secondary tasks).

Since visual servoing consists in positioning a camera according to the information perceived in the image, the task is specified in a 2D space, while the resulting camera trajectories are in a 3D space. Such approaches are computationally efficient and thus suitable for highly dynamic environments such as computer games. However, one cannot determine in advance which degrees of freedom of the camera and how many of them will be instantiated by the main task  $\mathbf{W}^+\mathbf{e}_1$ . Moreover, a specific process must be added to maintain the

smoothness of the path in order to avoid too sudden modifications of the camera speed and direction while reacting to the motion of a target. Finally, since control is carried out in the image (thus in 2D), the animator has no control on the resulting camera trajectory which is computed automatically.

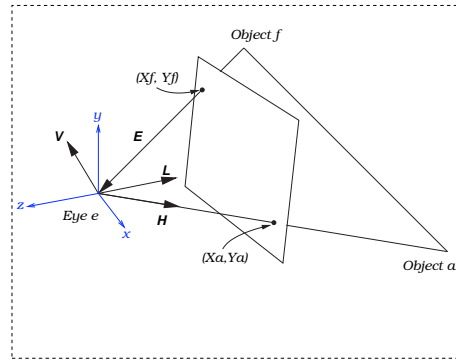
An application with similar objectives has been proposed in the domain of computer games by Halper *et al.* [HHS01]. Their approach implements a camera engine based on targeted objects both in 2D and 3D spaces. They propose a full set of properties on the objects of the game, ranging from height angles, angles of interest, size, visibility and positioning on the screen. Since the events generated by the story and actions of the game may change at each frame, constraints must be reevaluated for each new situation. Contrary to a purely reactive application of constraints like Bares [BL99], Halper *et al.* tend to avoid “jumpiness” of the camera (*i.e.* when the camera constantly jumps to global best-fit solutions). This is achieved by maintaining frame-coherence: an algebraic incremental constraint solver computes the new camera configurations from existing camera states. This ad-hoc solving process has been proposed to satisfy at each frame a set of constraints on the screen. Since not all the constraints can be fully satisfied, they propose an algorithm with relaxation capabilities that solves only certain constraints and then modifies the camera state to accommodate the remaining constraints. Moreover, lookahead techniques are used to adjust the camera parameters for future situations by approximating future object states based on their past trajectories and velocity information.

## 6. Algebraic systems

Algebraic approaches are a simple and efficient way to position the camera according based on shot properties. In an algebraic system a camera set-up is regarded as the solution to a vector algebra problem defined on the model world being viewed. Such systems comprise a method of solving one particular class of algebraic which has the advantage that the solution is usually arrived at efficiently although the necessary idealization of objects as points, and the restriction to locating two entities in the screen space, limits the application of the algebraic methods.

The earliest example of an algebraic system was Blinn’s work at NASA [Bli88]. Working on the visualization of space probes passing planets, he developed a system for setting up a camera so that the probe and planet appeared on screen at given coordinates with given sizes. The problem was expressed in terms of vector algebra for which both an iterative approximation and a closed form solution could be found.

The closed form solution consists in computing the parameters of translation matrix  $\mathbb{T}$  and rotation matrix  $\mathbb{R}$  in equation (1) that express the change from world space to view space. The input is the coordinates of the objects  $f$  and



**Figure 8:** Jim Blinn’s algebraic approach[Bli88] to camera control: two points on the screen, the field of view, and an up-vector allow to directly compute a camera position with vector algebra.

$a$  to view in the 3D space with the coordinates of their desired location on the screen  $(X_f, Y_f$  and  $X_a, Y_a)$ , the up vector, and the aperture of the camera. Blinn first computes the translation matrix to find the position of the camera ( $\mathbb{T}$ ) by studying the triangle  $(e, f, a)$  (see fig. 8) both in world space and in view space. He then computes the rotation matrix to view both objects at their desired coordinates. As most vector algebra approaches, the solution is prone to singularities that have to be carefully managed.

Interestingly the approximate method was more applicable as it produces acceptable results even when an impossible problem was posed.

The range of problems was very limited to those involving one spaceship and one planet, which illustrates the limited nature of algebraic systems. Attempts to generalize these systems have relied on the use of idioms, standard lay outs of subjects and cameras commonly used in cinematography. In these systems solution methods are devised for each lay out, and the input consists of a model world together with a list of idioms to apply. Such system have been developed by Butz [But97] and Christianson [CAH\*96].

Vector algebra approaches have also been studied in purely 2D-based applications such as cell animation (motion-picture cartoons) or virtual 2D guided tours (presentation of different artworks such as frescos, tapestries or paintings). The need for camera planning algorithms for cartoon animation has been addressed by Wood *et al.* in [WFH\*97] (in association with the Walt Disney Animation Studios). Here the problem was to generate a single 2D background image (called a *multiperspective panorama*) in order to create the visual effects expected by the director given the 3D scene and camera paths. Their system generates the panoramas by relying on some basic camera moves such as pan, tilt-pan, zoom and truck. Once the *multiperspective panoramas* and their associated “moving win-

dows” were generated, there is the possibility of incorporating other computer-animated elements that could interact with the background.

Another field of application for 2D camera planning is “multimedia guided tours” which aim is to help visitors during a visit by, for example, providing additional information on the artworks at a heritage site. Zancanaro *et al.* explored the use of PDAs in multimedia museum tours in [ZSA03, ZRS03]. The approach consists in planning camera movements on *still* images (e.g. the frescos or paintings on the walls of a room). The camera planner can help the visitor to focus on both important aspects of the artworks and small details that could be difficult to notice with an audio commentary alone. Camera movements [ZRS03] are planned on a 2D image of the artwork in a manner that is coordinated with the audio commentary that describes the artworks to the visitor. Application of observations from the cinematography literature [Ari76] avoids inconvenient transitions between camera movements. Palamidese’s algebraic approach [Pal96] admits the possibility to describe an artwork by planning camera movements that first show the details and then successively zoom-out to show the entire model.

Algebraic systems offer the first reliable solution to efficiently automate the positioning of a camera by considering simple screen configurations. It has utilized in a number of approaches where the primary cinematic goal can be decomposed into such simple configurations. The applications and potential of such methods are however limited. The representations of objects are restricted to points which can lead to contrived camera configurations if objects are either too large or too small, and the techniques are inadequate for complex shapes. The user specifies the exact projection of at most two points in the 3D scene, which is often insufficient and overly restrictive for real world camera planning problems.

## 7. Generalized Approaches

In a move towards a better automation of the process of positioning a virtual camera, we first require the identification of high-level properties on the camera, its projected image, and its motion, and then require examine solving techniques to enforce these properties. The computer graphics community has naturally relied on the rich expressiveness of cinematography to characterise declarative camera control systems. The user describes the scene in terms of properties which are in turn converted into relations (constraints and/or objective functions) between the degrees of freedom of the camera and the environment. The relations are solved by a range of techniques. These approaches are *generalized* approaches since the models are not limited in the number and nature of properties it addresses.

With respect to generalized approaches, the camera control problem can be seen as either a constraint satisfaction

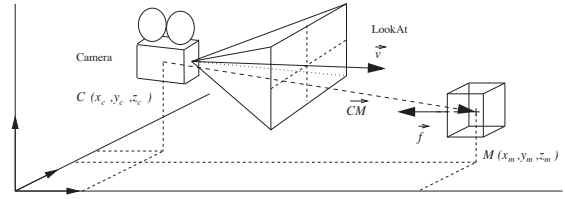


Figure 9: Modeling of the vantage angle property relative to object  $M$ .

or an optimization problem. Most approaches actually involve both. The user expresses a set of properties on the shots ranging from framing, vantage angles, to classical idioms which are in turn expressed as numerical constraints (properties that must hold) or functions to minimize (properties which do not have to be exactly enforced) on the camera parameters. The solving process is then a matter of exploring the space of camera configuration while minimizing the objective functions and/or satisfying the constraints. The principle characteristics that differentiate approaches relate to the richness of the user language and the properties and performances of the solving techniques.

### 7.1. Property modeling

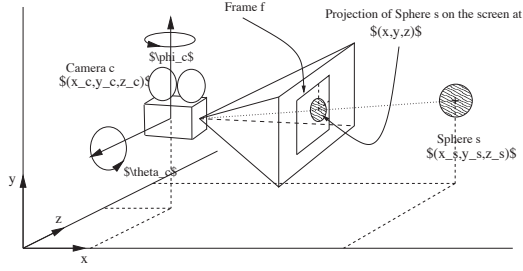
In generalized approaches, modeling the camera control problem requires a translation of the user’s descriptions into a set of fitness functions or constraints. For illustrative purpose, we describe the process of modeling of two properties: vantage angle and framing. We consider a pinhole Euler-based camera model as presented in Figure 4. The vantage angle property requires the camera to view an object from a specific orientation (e.g. front, top, rear, see section 9). As illustrated in figure 9, the fulfillment of this property can be trivially expressed as a dot product between the object orientation and the camera *lookat* vector (normalized between 0 and 1):

$$f_1(\mathbf{q}) = 1 - (v \cdot f + 1)/2$$

where  $\mathbf{q} = [x_c, y_c, z_c, \phi_c, \theta_c, \psi_c, \gamma_c]^T$  represents the camera parameters,  $f$  a unit vector representing the desired orientation of the object and  $v$  the *lookat* vector which is computed considering the camera angles  $\theta_c$  and  $\phi_c$  as polar coordinates:

$$v = \begin{pmatrix} \cos(\phi_c) \sin(\theta_c) \\ \cos(\phi_c) \cos(\theta_c) \\ \sin(\phi_c) \end{pmatrix}$$

Maximum value of function  $f_1$  is naturally reached when the vectors  $f$  and  $v$  are opposite (i.e. the camera faces the object). In optimization-based approaches, the space of camera



**Figure 10:** Modeling of the framing property.

configurations if searched for maximizing  $f_1$ . In constraint-based approaches, one can enforce some variation in the degree of respect of the property, e.g. constraint

$$v \cdot f \leq \frac{\pi}{4}$$

provides a possible variation of 45 degrees around the desired vantage angle  $f$ .

Second we consider the modeling of the framing property that constrains an object to project on a desired location  $[x_d, y_d]^T$  on the screen. An illustration is provided in figure 10.

A simple euclidian distance between the desired location  $[x_d, y_d]^T$  and the projected location of a point  $P = [x_P, y_P, z_P]^T$  of the object can be used in conjunction with  $\tanh(x)$  that converges towards 1 when  $x$  tends to  $+\infty$  in order to normalize the result:

$$f_2(\mathbf{q}) = 1 - \tanh\left(\left(H(\mathbf{q}) \cdot [x_P, y_P, z_P]^T - [x_d, y_d]^T\right)^2\right)$$

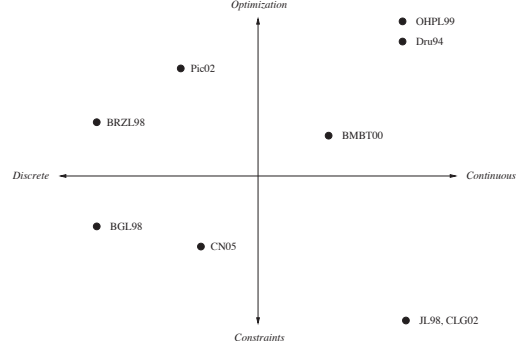
Modeling this property in a constraint-based context is a similar process. In most cases, constraints allow a certain flexibility in the translation in order to avoid systematic failures in the search process. For example in [JL98], the authors express the framing property as a projected point  $P = [x_P, y_P, z_P]^T$  that must belong to a 2D frame  $F = [[x_1, y_1]^T, [x_2, y_2]^T]^T$ :

$$\begin{cases} [x', y']^T = H(\mathbf{q}) \cdot [x_P, y_P, z_P]^T \\ x' \geq x_1 \\ x' \leq x_2 \\ y' \geq y_1 \\ y' \leq y_2 \end{cases}$$

It can be problematic to express some properties as algebraic relations and such cases require distinct techniques (for example see section 8 for the treatment of occlusion).

## 7.2. Problem solving

A broad range of techniques is available and solvers differ how they manage over-constrained and under-constrained



**Figure 11:** Classification of generalized approaches considering two axes: from discrete to continuous techniques and from constraint-based to optimization techniques.

problem formulations, both in their complete or incomplete search capacities and their discretization of continuous camera domains.

Consider the 2-dimensional classification of the approaches as presented in figure 11:

- the first axis (horizontal) corresponds to the nature of the domains considered in the solving process and spans from fully discrete to fully continuous approaches. Discrete approaches rely on testing a subset of camera configurations through a regular or stochastic subdivision of the domains to reduce the overall complexity of exploring a 7 dof search space (an incomplete process). By contrast, continuous approaches provide techniques to explore, in the worst case, the whole set of configurations. Heuristics help to reduce the complexity.
- the second axis (vertical) corresponds to the nature of the solving technique, from pure optimization techniques to pure constraint-based techniques. At one extreme, pure optimization techniques are considered as soft solving techniques in that the best (possibly local) solution is computed with respect to a function that is a measure of the violation of each property. At the other extreme, pure constraint satisfaction techniques can be considered as hard solving techniques. Such approaches perform an exhaustive exploration of the search space, thus providing the user the set of solutions to the problem, or a guarantee that the problem has no solution.

Our characterization of these approaches uses a set of user-defined cinematographic properties, of various expressiveness, motivated by the cinematography literature ([Ari76, Mas65]). These properties are formulated as constraints and/or objective functions which are used in the numerical (complete or incomplete) solving procedures.

### 7.3. Optimization-based Approaches

The category of approaches that address camera control with pure optimization techniques express a set of properties as shot objectives to be maximized. Metrics are provided to evaluate the quality of a shot with respect to the underlying graphical modeling of the scene and the user's description of the problem. The optimization solver navigates within the camera parameters in a search for a solution that maximizes the shot objectives. Classical optimization techniques encompass deterministic approaches such as gradient-based or Gauss-Seidel methods and non-deterministic approaches such as population-based algorithms (genetic algorithms in particular), probabilistic methods (Monte Carlo) and stochastic local search methods (hill climbing, simulated annealing). The problem is mathematically expressed as finding a camera configuration  $\mathbf{q} \in \mathbf{Q}$  (where  $\mathbf{Q}$  is the space of possible camera configurations) that maximizes (or minimizes) a fitness function (respectively a cost function) as illustrated by the following equation:

$$\text{maximize } F(f_1(\mathbf{q}), f_2(\mathbf{q}), \dots, f_n(\mathbf{q})) \text{ s.t. } \mathbf{q} \in \mathbf{Q}.$$

where the functions  $f_i : \mathbb{R}^7 \rightarrow \mathbb{R}$  measure the fitness of each property desired by the user, and the function  $F : (\mathbb{R}^7 \rightarrow \mathbb{R}, \dots, \mathbb{R}^7 \rightarrow \mathbb{R}) \rightarrow \mathbb{R}$  aggregates the functions  $f_i$  in a single function to maximize. In its most simple representation  $F$  is generally a linear combination of scalar weighted functions:

$$F(f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_n(\mathbf{x})) = \sum_{i=1}^n w_i f_i(\mathbf{x})$$

Olivier *et al.* [OHPL99] follow this principle to tackle the visual composition problem (*i.e.* a static camera) as a pure optimization procedure based on the use of genetic algorithms. The authors embed a large set of properties such as explicit spatial relationships between objects or on the camera encompassing (partial or total) occlusion culling, size or layout of objects. The fitness function is a linear weighted combination of the fulfilment of the properties the user desires.

The solving process consists in encoding the seven parameters of the camera into the chromosomes of genes which represent the variables of the problem. A population of cameras is then randomly distributed in the search space. Each individual of this population is evaluated with respect to a set of objective functions. The top 90% representatives of the population survive to the next generation and selection is by binary tournament. The remaining 10% are re-generated by random crossover and/or mutations of the chromosomes (small perturbations of its current value). The whole process is embedded in a tool referred to as CAMPLAN. A short example illustrates an extension of the approach to dynamic camera planning; a quadratic path joins known start and end points. The unknowns of the problem are the control points that must satisfy some temporal indexed properties.

An evolution of this approach has been reported by Halper

and Olivier in [HO00]. The optimizer is based on the same population-based method, whereas both the expressivity and the metrics pertaining to the properties have been improved. In particular a significant improvements were made to realize precise relative locations of objects and occlusion. The initial approach relies on sphere approximations of the objects, which potentially leads to erroneous and imprecise results, the authors have devised a method that first computes a convex hull of an object and then determines the extends of the hull by running along its edges. All measures are run with these extends. Therefore a property such as viewing an object A on the left of object B on the screen is naturally enhanced by a more precise evaluation. For a finer evaluation of occlusion, objects are hardware rendered and a ratio of the number of visible projected pixels use as a measure of visibility.

The computational cost, as well as the non-deterministic behavior, were identified as the main shortcomings of the pure genetic algorithm based approach with durations from three to five minutes subject to the number of degrees of freedom (from 3 to 7). A possible enhancement highlighted by the authors in [OHPL99] consists in restricting the search space to feasible regions by pruning impossible areas and the initial random distribution would thus be contained to *promising* regions. Restricting the regions in which genetic search is conducted was undertaken in [Pic02]. Following the same declarative scheme, feasible locations for the camera are abstracted from the specification of the shot. For example, if a user desires to view the front of an object, the volume of space corresponding to rear shots can be pruned. Where multiple objects and properties are concerned the final space to search is the intersection of all component feasible regions. Thus generation of the volumes comprises two steps:

- building a BSP (Binary Space Partition) tree of the search space that comprises vantage angles and shadows volumes for occlusion (*cf.* Section 8).
- generating an octree representation of the search space with a given precision such that the only retained voxels are those that fully lay in the inside volume of the BSP.

This structure exploited in the design of the chromosome which comprises a reference to a voxel with an offset inside the voxel (useful for large voxels), an orientation and a field of view. Each chromosome is subject to crossover operation and the chromosome design ensures that the search is limited to feasible regions. The solving process then follows the previous scheme. As with all multiobjective optimization approaches, the main problem is the difficulty of efficiently modeling and composing multiple components into a single objective function. The weighting is generally determined by an empirical, sometimes tedious generate and test process.

#### 7.4. Constraint-based Approaches

The CSP (Constraint Satisfaction Problem) framework has proven to succeed in a number of camera composition and motion planning approaches. The framework offers a declarative approach to model a large range of mainly non-linear constraints and proposes reliable techniques to solve them.

Jardillier *et al.* [JL98] proposed the first constraint-based approach in which the path of the camera is created by declaring a set of properties on the desired shot such as vantage angles, projection into screen frames or object sizes. The use of pure interval methods to compute camera paths in *The Virtual Cameraman* yields sequences of images fulfilling temporally indexed image properties. The path is modeled using a parameterized function (of degree 3) for each degree of freedom of the camera. The unknowns are the parameters of the parameterized function, which once instantiated provide a solution to the problem.

Interval arithmetic based solvers compute the whole set of solutions as interval boxes (a Cartesian product of intervals) – [Sny92] present a broad review of the application of interval arithmetic in computer graphics. Each unknown in the problem is considered as an interval bounded by two floating points that represent the domain within which the search should be conducted. All the computations therefore integrate operations on intervals rather than on floating point values. We present the interval extension for the classical operators:

$$\begin{aligned} [a, b] + [c, d] &= [a + c, b + d] \\ [a, b] \cdot [c, d] &= [a \cdot c, b \cdot d] \\ [a, b] \times [c, d] &= [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)] \\ [a, b] \div [c, d] &= [\min(\frac{a}{c}, \frac{a}{d}, \frac{b}{c}, \frac{b}{d}), \max(\frac{a}{c}, \frac{a}{d}, \frac{b}{c}, \frac{b}{d})] \text{ if } 0 \notin [c, d] \end{aligned}$$

In a similar way, all the other operations can be extended to interval operators in such a way that for an operator  $\top$  the following relation holds:

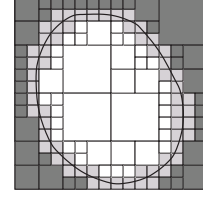
$$[a, b] \top [c, d] = \{x \top y \mid x \in [a, b] \text{ et } y \in [c, d]\}$$

The fundamental property of interval arithmetic, on which the solving process relies and correctness is guaranteed, is the containment property [Moo66]:

$$\forall \mathbf{x} \in \mathbb{R} \mid \mathbf{x} \in \mathbf{X}, f(\mathbf{x}) \subset \mathbf{F}(\mathbf{X})$$

where  $X$  is an interval,  $f(\mathbf{x})$  a unary function and  $\mathbf{F}(\mathbf{X})$  its interval extension. The interval extension of a function  $f$  is a function  $\mathbf{F}$  where all the operators on real values have been replaced by operators on interval values. This property states that an interval evaluation of an interval function  $\mathbf{F}$  always contains the evaluation of the real function  $f$ . This observation provides a basic insight as to how to solve a constraint equality  $f(\mathbf{x}) \geq 0$ , based on a trivalued logic {True, False, Unknown}:

$$\begin{cases} \text{if } \mathbf{F}(\mathbf{X}) \cap [0, +\infty] = \emptyset, & \text{return False} \\ \text{if } \mathbf{F}(\mathbf{X}) \cap [0, +\infty] = \mathbf{F}(\mathbf{X}), & \text{return True} \\ \text{if } \mathbf{F}(\mathbf{X}) \cap [ -\infty, 0] \neq \emptyset, & \text{return Unknown} \end{cases}$$



- contains no solutions
- contains possible solutions
- contains only solutions

Figure 12: Split and evaluation approach to solve  $f(\mathbf{x}) \leq 0$

and considering a constraint equation  $f(\mathbf{x}) = 0$ :

$$\begin{cases} \text{if } \mathbf{F}(\mathbf{X}) \supset 0, & \text{return Unknown} \\ \text{if } \mathbf{F}(\mathbf{X}) \not\supset 0, & \text{return False} \end{cases}$$

This leads to a dynamic programming evaluate-and-split process as presented in figure 12. Each cartesian domain  $\mathbf{X}$  of values is evaluated with respect to a function  $\mathbf{F}$ : if true (white regions), the process stops and keeps  $\mathbf{X}$ , if false (dark grey regions) the process stops and throws  $\mathbf{X}$ , and if unknown (light grey regions), the process splits  $\mathbf{X}$  and operates recursively on each half.

However computationally expensive even with small dimension problems, the interval approach shares some interesting features:

- a guaranteed approximation of all the solutions is provided,
- each box  $\mathbf{B}$  that is evaluated to True contains only solutions (*i.e.* each floating-point number in  $\mathbf{B}$  is a solution)
- if all boxes are False, the problem is guaranteed to have no solutions,
- the process manages linear, polynomial, non-polynomial and non-linear equations and inequalities.

Some enhancements, to manage the low expressivity of the camera paths as well as the important computational cost required by pure interval techniques, were reported by Christie *et al.* in [CLG02]. The path of the camera is modeled as a set of primitive camera movements sequentially linked together. These primitives are in essence cinematographic notions and include travelings, panoramics, arcings and any composition of these movements. Unlike most approaches, which only guarantee the correctness of user-defined properties for a set of points on a camera trajectory (generally the start and end points plus some key points taken on the camera path), the interval-based approach guarantees the fulfilment of properties through the whole sequence. Each primitive, referred to as a hypertube, is then treated as a separate, but related, constraint satisfaction problem. The overall solving process solves the problem in se-

quence, constraining the end of hypertube  $i$  to join the beginning of hypertube  $i + 1$ .

For reasons of computational efficiency the authors replaced the evaluation process in Jardillier's evaluate-and-split approach by a pruning process based on local consistency techniques as well as artificial intelligence propagation methods. Details, results and comparisons can be found in [BGLC04]. The main limitation of these approaches is a counterpart of the guarantee of completeness it offers; whenever the problem has no solutions, the solving process exits without any indication of the source of the inconsistency (e.g. incompatible constraints). Thus, the user must remove some constraints and solve the process again so on and so forth until a solution is found. The constraints community offer some techniques to manage overconstrained problems, but only at a significant computational cost.

Hierarchical constraint approaches are able to relax some of the constraints in order to give the user a approximate solution of the problem. Bares *et al.* proposed CONSTRAINTCAM, a partial constraint satisfaction system, which provides alternate solutions when constraints cannot be completely satisfied [BGL98]. If CONSTRAINTCAM fails to satisfy all the constraints of the original problem, the system relaxes weak constraints and, if necessary, decomposes a single shot problem to create a set of camera placements that can be composed in multiple viewports [BL99] providing an alternate solution to the user.

Inconsistencies are identified by constructing an *incompatible constraint pair graph*. When the solver fails to find a solution, the planner repetitively relaxes weak constraints until no incompatible constraint pair remains. This solution is based on a limited subset of cinematographic properties (viewing angle, viewing distance and occlusion avoidance) and restrict the application of the constraint satisfaction procedure to relatively small problems (involving two objects). Moreover, a drawback of using partial constraint satisfaction is that it requires the user to specify a hierarchy of constraints (ordering constraints on the visual appearance of a shot is not always trivial).

### 7.5. Constrained-optimization-based approaches

Pure optimization techniques enable the computation of a *possibly good* solution in that each property is satisfied to some degree. In the worst case, this partial satisfaction can lead to a contrived solution (e.g. incompatible fitness functions). In contrast, pure complete constraint-based techniques can compute the whole set of solutions, but have their own intrinsic problems such as computational cost and the challenge of overconstrained systems. An acceptable compromise can then be found through *constrained-optimization* approaches. The camera control problem is modeled through a set of properties to be enforced (the constraints) and a set of properties to maximize (fitness functions to optimize) and

the problem can be classified as a constraint satisfaction and optimization problem (CSOP).

In contrast to their early attempts at procedural camera control [DGZ92], Drucker and Zeltzer propose the CAM-DROID system which specifies behaviors for virtual cameras in terms of task level goals (objective functions) and constraints on the camera parameters. They regroup some primitives constraints into *camera modules* which represent a higher level means of interaction with the user. The constraints of each module are then combined by a constraint solver and solved by a camera optimizer based on the *CF-SQP* (Feasible Sequential Quadratic Programming) package, which has been designed to solve large scale constrained non-linear optimization problems. Constraint functions and fitness functions are restricted to be continuously differentiable. This is an implementation of the method of Lagrange undetermined multipliers in which the Lagrangian is numerically approximated and solved using Newton's method. This approach is limited to smooth functions subject to smooth constraints, conditions difficult to guarantee in computer graphics. Furthermore, the method is prone to local minimas and is extremely sensitive to the initial conditions

### 7.6. Discrete Approaches

To address the intrinsic complexity of exploring a 7 degree of freedom continuous search space, a number of approaches simply consider a regular discretization on each degree of freedom. In an approach to virtual camera composition that addresses a drawback of their CONSTRAINTCAM system, Bares *et al.* [BTMB00] propose the use of a complete search process on a discretization of the search space. The reported work follows a global optimization process in such that each configuration is provided a value representing its fitness as in classical optimization [OHPL99] but an exhaustive generate-and-test process is employed. The fitness is built as the aggregation of the satisfaction of each property provided by the user normalized between 0 and 1. The search space is covered by a  $50 \times 50 \times 50$  grid for camera locations, every  $15^\circ$  angle for orientation and 10 possible values for the field of view – 30 minutes is required (on a Pentium II 400MHz) for evaluation of over 13M shots.

Pickering and Olivier [Pic02] describe how the search space can be reduced to feasible regions by pruning inconsistent possible camera locations. In a direct application of this principle, the authors propose a discretization approach that explores only feasible regions built through an intersection process [BMBT00]. Hence the overall complexity is reduced. The authors improve the previous solving approach by proposing a recursive heuristic search process that starts exploration at a coarse grid resolution. Each configuration is evaluated and the  $n$  best candidates are kept. The process is then recursively applied on each of the best candidates by considering a finer grid resolution. The process stops either

when a given quality threshold is reached when evaluating a candidate or when a minimal grid resolution is obtained.

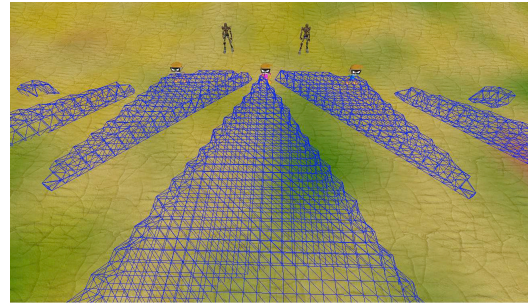
### 7.7. Hybrid Approaches.

Mixing constraint-based and optimization-based techniques together offers a comfortable solution to declarative camera control as both requirements (constraints) and preferences (fitness functions) can be embedded. However the computational cost required to solve such problems must be carefully addressed. Hybridization of different techniques can reduce this complexity. In particular, some of the constraints can be solved by relying on geometric operators that first reduce the search space before applying numerical techniques.

A novel feature is that a first step can compute geometric volumes, solely based on camera positions, to build models of the feasible space [BMBT00, Pic02]. A second step relies on a classical stochastic search as in [Pic02], or heuristic search [BMBT00] inside the resulting volumes. In [CN05], the idea has been developed further to provide identification and semantic characterization in terms of cinematographic properties of each volume (see Fig 14). The constraints are represented as *semantic volumes*, *i.e.* volumes that encompass regions of the space that fulfil a cinematographic property. This work can be considered as an extension of *visual aspects* [KvD79] and closely related to *viewpoint space partitioning* [PD90] (in the field of object recognition).

In *visual aspects* all the viewpoints of a single polyhedron that share similar topological characteristics on the image are collected. A change of appearance of the polyhedron, with changing viewpoint, gives rise to boundaries in the search space. Computing all the boundaries enables the construction of regions of constant aspect, namely *viewpoint space partitions*. This extension of viewpoint space partitions consists in dealing with multiple objects and replacing the topological characteristics of a polyhedron by cinematographic properties such as occlusions, relative viewing angles, distance shots and relative object locations on screen. A *semantic volume* is then defined as “a volume of possible camera locations that give rise to qualitatively equivalent shots with respect to cinematographic properties, *i.e.* *semantically equivalent shots*” [CN05]. This approach tends to characterize the search space in terms of cinematographic terms and, contrary to other systems, tries to offer the user different solutions fulfilling his description of the problem (*cf.* Fig. 13).

Each property offered to the user (occlusion, framing, orientation of objects, distance shots) leads to the creation of a *semantic volume* that contains the camera positions that possibly satisfy the property. Since a problem is described as a conjunction of properties, the volumes are intersected to check whether it might have a solution. Indeed if the volume representing the intersection is empty, the description has no solution. However, a solution might be found but a



**Figure 14:** Camera volumes yielding no occlusion between the characters on the screen.

numerical step is required to set the orientation parameters of the camera ( $\theta, \phi$  and  $\psi$ ), since the geometrical step only reduces search space for the position of the camera ( $x, y$  and  $z$  parameters). The optimization process within the intersection volume is based upon a continuous extension of the stochastic local search metaheuristic and tends to minimize functions that evaluate the quality of a solution. Given an initial guess inside the volume, each iteration of the algorithm generates a set of neighbors of the current configuration (*i.e.* set-ups with small randomized modifications on the camera parameters) and chooses the best one as starting point for the next iteration. The numerical stage ends when a solution is found or after a predefined number of steps. The computation of a good representative of each volume lays the groundwork for an interactive approach in which the user navigates between the different classes of solutions and interacts with the semantic information. This approach is however limited to static scenes due to the time required to compute the intersection of the volumes and to the fact that the computation of the volumes is dependent on the object’s position in space.

## 8. Occlusion

Regardless of the application domain, occlusion is nearly always a major issue in camera control. For example, computer games and animation have to deal with the non-occlusion of characters of interest and scientific and information visualisation must facilitate the maintenance of unoccluded views of significant. Many other domains like automated presentation (*e.g.* multimodal systems and knowledge-based learning environments) also face similar problems.

Occlusion occurs when the object of interest (object that must be identified to the user in an application) is hidden by another object (or a set of objects) in the scene. This can be expanded to include a range of notions associated to the expressiveness of occlusion, ranging from complete, partial to unoccluded. Indeed, in particular domains, where the relative positions of objects are important, the controlled main-

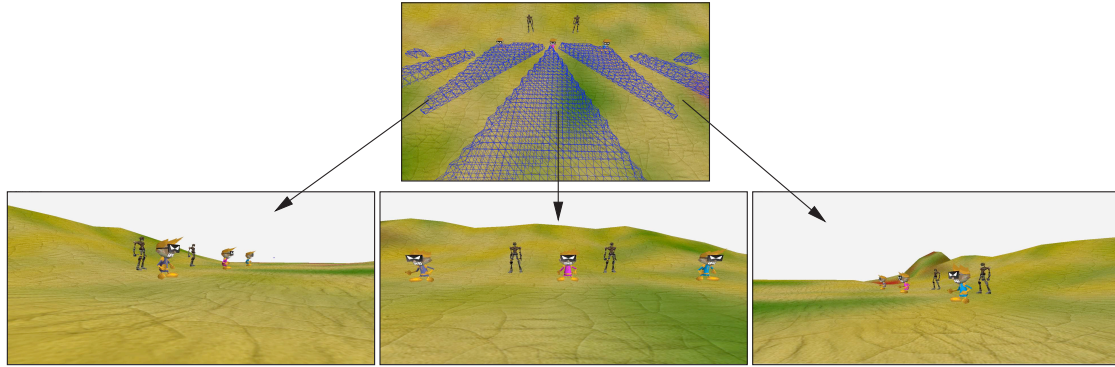


Figure 13: Illustration of the Semantic Volumes approach.

tenance of partial occlusion is desirable means of illustrating the relative depths of objects in a scene.

Occlusion is a purely geometrical relation, *i.e.* an object occluded when some region of its projection on the screen obscured by the projection of another object. Conceptualizing occlusion by reference to the screen, and thus as a discrete phenomena, we can similarly:

- partial occlusion:  $A$  partially occludes  $B$  when the projection of an object  $A$  covers at least one pixel of the projection of object  $B$  on the screen;
- total occlusion:  $A$  totally occludes  $B$  when the projection of an object  $A$  covers all the pixels of the projection of object  $B$  on the screen.;

Camera control approaches vary in the degree and character of their concerns regarding occlusion, for example, games are uniquely concerned with the maintenance unoccluded views. Some approaches address partial occlusion [CN05] and may even offer a quantification of the occlusion by providing an ability to specify the degree of occlusion as a percentage [BMBT00]), or a pixel overlap counts [HO00].

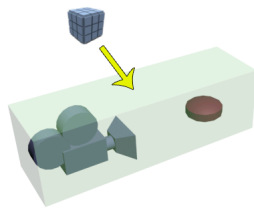
Dynamic environments require the ability to express changes in occlusion over time as is the case in many real world spatial contexts. Indeed, when a character enters a car or building, it is often important to ensure that the end of the shot includes momentary occlusion of the actor by the doorway of entrance, and likewise in the next shot might typically start with occlusion as the character emerges. Variety in expressiveness required for occlusion has an impact on its management and computation and thus the both its representation and implementation. When dealing with movement in scenes, there are two main approaches depending on the *a priori* knowledge on the scene, *i.e.* the degree of nondeterminism is the positions of objects in the scene. Techniques can be divided into two main classes: reactive and deliberative.

The reactive management of occlusion is commonly achieved by ray casts from the camera to the object of inter-

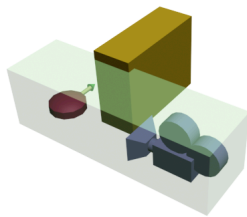
est. Based on the results of the intersections tests, the camera can be moved to avoid occlusions. The efficiency and simplicity of ray casting techniques make them a valuable choice for realtime environments. In particular this solution is used in most computer games (*e.g.* [Gio04]) and some dynamic realtime learning-based environments [BZRL98]. To improve performance, ray-object intersection is generally replaced by a ray-bounding volume intersection. The choice of the bounding model (bounding sphere, bounding box, OBB-tree, *etc.*) and the number of rays cast determine the precision of the occlusion test.

Occlusion can also be handled in real time using *consistent regions* of space, as in [BL99]. Consistent regions correspond to areas expressed in local spherical coordinates systems with origin at the center of the object upon which the occlusion constraint is applied. The consistent region satisfying the occlusion for an object  $S$  is found by projecting the bounding boxes of nearby potentially occluding obstacles onto a discretized sphere surrounding  $S$  (see [BL99, PBG92, DZ95] for further details). These projections are converted into a global spherical coordinate system and then negated to represent occlusion-free regions of space for  $S$ .

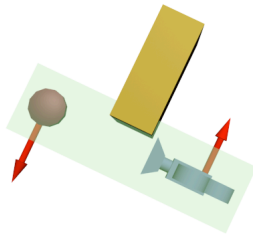
Due to their discrete and approximate nature, both ray casting and consistent region techniques can fail to detect occlusions. In address this limitations, a number of approaches use intersections between volumes and objects of the scene. For example, for target tracking purposes Courty [Cou02] and Marchand [MH98, MC02], avoid occlusion by computing a bounding volume around both the camera and the target. Objects are then prevented from entering this volume which might result from either target motion (figure 15(b)), camera motion (figure 15(c)) or motion of another object (15(a)). The bounding volumes are created around both the object and the camera according to their motion with respect to the obstacles in the scene. This notion has been extended to deal with objects with unknown trajectories by computing predictive temporal volumes based on the current position of



(a) Occlusion due to a moving object.



(b) Occlusion due to the target motion.



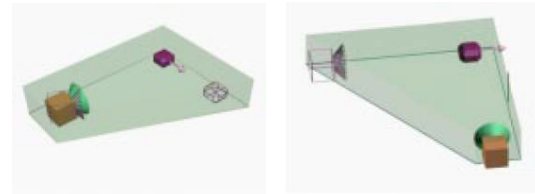
(c) Occlusion due the camera motion.

**Figure 15:** Occlusion avoidance based on bounding volumes around both the camera and the targeted object.

an object and an extrapolation of its next position based on current velocity (cf. figure 16 and figure 17).

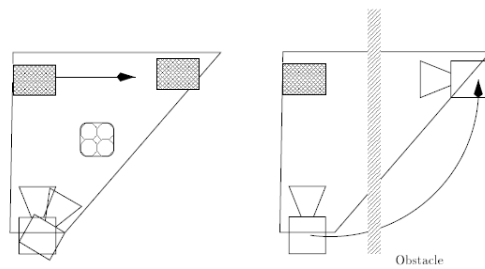
More advanced techniques [HO00, HHS01] take advantage of graphics hardware to address occlusion. By rendering the scene in hardware stencil buffers with a color associated to each object, number and extent of occluding objects can be evaluated. Whilst the major drawback of hardware-based approaches is their dependency on the underlying hardware, and thus a loss of generality (and portability), hardware rendering techniques have many advantages beyond raw performance:

submitted to EUROGRAPHICS 2006.



(a) Computing the risk of occlusion caused by motion of an object. (b) Computing the risk of occlusion caused by motion of the camera.

**Figure 16:** Evaluating the risk of occlusion caused by object or camera motion.



(a) Detection of a future occlusion. (b) Detection of a future collision.

**Figure 17:** Collision/occlusion detection with temporal volumes.

- the use of very low resolution buffers (32x32 pixels) is sufficient to deal with occlusions;
- such techniques are independent of the internal representation of the objects (no need of boundaries for the objects of the scene);
- rendering the scene without any use of bounding volume simplification ensures that exact occlusions are computed (modulo the resolution of the buffers).

Where performance is not a significant issue, occlusions can be addressed using object space techniques. For example, Pickering and Olivier [Pic02] have utilised a shadow-volume based algorithm applied on the camera and the obstacles to determine the parts of the scene that are unoccluded from the current viewpoint (cf. figure 18). The object of interest is treated as the light source and all the computed volumes represent the occluded areas. When multiple occlusions occur, the union of the shadow volumes are computed. Although effective for static scenes where it can form the basis of a secondary search for other images properties, shadow volumes are not feasible for dynamic elements. Geometric management of occlusion was proposed by Christie & Normand [CN05] whereby a subdivision of the space related to occlusion cones can be computed using bounding spheres of the objects of the scene. This method distinguishes between partial and total occlusions (cf. figure 19) but again is only

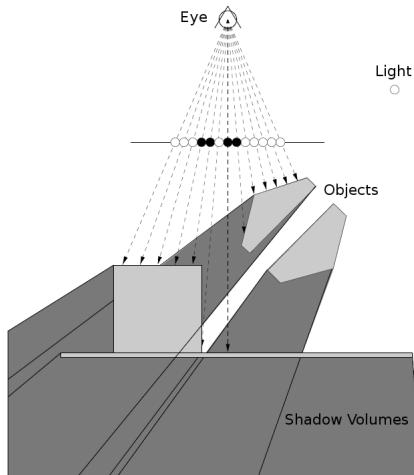


Figure 18: Shadow volumes algorithm.

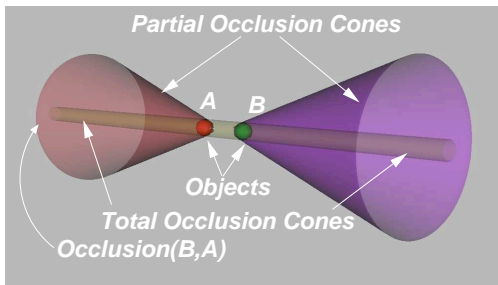


Figure 19: Total and partial occlusion cones as presented in [CN05].

applicable to static scenes since each movement of an object requires the recomputation of the cones.

## 9. Expressiveness

Expressiveness describes the possible properties offered to the user to control a virtual camera and in such characterizes the declarative power of the camera control system. Though expressiveness is strongly related to both the application and the chosen solving technique, the range and nature of properties related to applications in camera control can be explored by reference to four criteria: the range of properties, the nature of the properties, the required level of abstraction of the 3D scene, and the extensibility.

Properties are related to the nature of the entities they characterize: either directly on the camera parameters, on the path of the camera, or on the content of the projected image. Properties relative to the camera constrain (more or less directly) the intrinsic and extrinsic parameters of the camera. This encompasses properties such as preset focal lengths, high or low vantage angles that

constrain the camera orientation, distance from camera to given objects and collision avoidance. Most contributions consider vantage angles and distance to camera as important features [Dru94, HHS01, CL03, BTMB00], whereas collision is rarely managed, with the exception of some reactive techniques (e.g. with force fields or probabilistic roadmaps). Focal length is either directly fixed by the application [Bli88, HCS96], assignable through constraints [Dru94, CL03] or computed by the satisfaction of a set of properties on the screen (provided in most of the generalized approaches).

Properties on the path of camera manage low-level issues such as collision avoidance and path coherency [HHS01, HO00, CL03, NO03]. Although primarily related to reactive environments that require some predictive approaches to enforce smoothness [HHS01, MC02], path coherency aims to smooth out the camera path where sudden changes of translational or rotational velocities occur. Issues at a higher level are related to the cinematographic nature of the path [CL03, CLDM03] or to the narrative goals [HCS96, BL97]. For example, Christie *et al.* propose the composition of a camera path as a sequence of parameterized cinematographic primitives (panning, tracking, dolly, travelling, arcing and zooming) or a set of these primitives (e.g. panning+zoom).

Finally, contributions report a very large set of properties related to the content of the screen, from low level *object-is-on-the-screen* enforcement to sophisticated image composition rules (see e.g. [GRMS01]). More precisely, different levels of properties should be considered. At the lowest level we can distinguish properties that apply at purely geometrically, e.g. where to project 3D points or objects on the screen and how to organize and lay them out. This encompasses absolute location of objects [Bli88, Dru94, HCS96, HHS01, GW92], framing of objects by constraining them in a given area (in or out of the screen) [JL98, CL03, BMBT00, HO00, OHPL99, Pic02] and relative location (e.g. A is on the left of B) [OHPL99, BMBT00, CN05].

Properties can also be specified at a (higher) semantic level, *i.e.* based on the nature and the implicit meaning of the objects. For example, the orientation properties adopted in a very large set of approaches (view the front/rear/left profile of an object) and that require elaboration of the geometry using canonical views of object. Furthermore cinematography provides a precise vocabulary that can be transposed to virtual camera control such as *over-the-shoulder* or *american shot* (shoot a character above the knees). Such grammars have been used in a number of automated approaches based on idioms ([CAH\*96, HCS96]) and in declarative approaches which in turn propose modeling languages [HO00, OHPL99, Pic02, JL98, CL03].

Finally, we can think of properties at both aesthetic and cognitive levels. Gooch's *et al.* [GRMS01] work computes

slight modifications of point of views in order to satisfy some high-level composition rules that are rules of the thirds or rule of the fifths, but only considers the geometry of the objects. Color, lighting and semantics are essential in considering aesthetic goals that comprise balance, unity and composition. A number of approaches are interested in characterizing object recognizability and provide techniques to compute views that maximize the visual saliency for static scenes [HHO05, LVJ05]. Very few are interested in movement saliency (see [Roz99]).

Cognitive levels have also been addressed by a small number of approaches in the robotics community. These consider the computation of a minimal set of viewpoints (canonical views) for recognizing an object and/or detecting its features. The principal metrics are view likelihood, view stability [WW97] and view goodness [BS05], which have been computed using entropy measures [VFSH01]. Extensions to automated navigation have also been explored for specific applications: historical data visualization [SS02] and scene understanding [SP05]. However, a deep understanding on underlying cognitive issues related to a precise task as well as the necessity to conduct user-studies are prerequisites (currently unexplored) for reusable results. Finally, notions such as the emotional response that a view can evoke has been largely ignored although Tomlinson *et al.* propose camera rules to cover a set of communicative goals based on the desired emotion to be conveyed [TBN00].

The nature of the properties, is largely a matter of descriptive convenience, whether these are quantitative properties such as exact numeric values for *e.g.* location of a points on the screen or for camera-to-object distances, and qualitative properties that can either rely on specific (cinematographic) vocabulary (*e.g.* framing) or a relational semantics (*e.g.* object A is left of B on the screen). However, the underlying level of abstraction of the 3D scene significantly distinguishes the expressiveness of approaches. For example, the framing property: *Object A belongs to frame F* has been characterized by different approaches using a variety of object abstractions. Some manage point-based abstractions where only the center of A is considered to belong to F [JL98], others use bounding-based abstractions with boxes [BMBT00, MLB02, CL03], spheres [OHPL99, HO00, CN05] (and hierarchical refinements), and approximated geometry through hardware rendering techniques in low-resolution buffers [HHS01, Pic02]. The abstraction mainly depends on the solving process and the computational resources available. As presented, the abstractions are either based on geometric simplifications to reduce the computational cost (from points to single bounding and hierarchical regions) or on semantic abstractions (object, character) to aid description.

Our final criteria is the extensibility of the approaches. While some present a fixed set of properties [HCS96, HHS01] and not extensible due to choice of

the solving techniques, a number of contributions have concentrated on providing frameworks to allow the addition of new properties. Most generalized approaches rely on solving techniques that propose extensible frameworks. However, constraint-based techniques offer a natural and easy way to add new properties; although they requires the expression of properties as algebraic relations which can be problematic for complex shapes and properties such as occlusion. On the other, optimization-based techniques offer the possibility to integrate non-algebraic relations, but require hand tuning to adapt the weight related to the new properties.

## 10. Discussion

To manage complex 3D scenes, abstraction of the geometry is widely used. Currently objects are usually considered as simple primitives (points or bounding volumes such as spheres). Such approaches provide imprecise and possibly erroneous results as many complex objects cannot be represented with simple bounding volumes. Some accounts do consider the precise geometry for occlusion purposes [HHS01, Pic02], but due to the computational expense, have to rely on hardware rendering capacities. Improving the quality of abstraction of the objects is a difficult but necessary work. Sophisticated (but common) configurations, *e.g.* as shooting a character through the leaves of a tree, will require further work with respect the the management partial occlusion.

The expressiveness of the set of properties is mostly related to the application context and the retained solution mechanism. Algebraic, interactive and real-time approaches generally rely on quantitative relations (*e.g.* exact locations or orientations of objects on the screen) [Bi88, CM01, HHS01, CAH\*96, GW92, BMBT00], while optimization and constraint-based systems allow for qualitative relations through the use of square or oval frames to constrain the location of objects [OHPL99, HO00, JL98, CL03]. In [CL03] qualitative relations allow the relaxation the hardness of the properties. Expressiveness is provided through the use of property softness too, as in hierarchical approaches. For Drucker [DGZ92], the choice was hidden from the user, and usually softness of the constraints is set through scalar coefficients [BL99, BMBT00, OHPL99] which can be awkward to configure. The question of how to determine the weights of each property and the aggregation of the weights in a single objective function remains unsolved. Hard constraint-based approaches do not require user configuration [JL98, CL03] and can use constraint relaxation to compute approximate solutions [BGL98].

Computational resources constrain both the geometry abstraction and expressiveness of the approaches. However, when compared to algebraic and real-time approaches, constraints and optimization-based approaches provide a pow-

erful framework by which new constraints or objective functions relative to any specific property can be added. Naturally, much effort is currently being expended on the development of efficient implementations with appropriate properties (e.g. the avoidance local minima) and in practice, hybrid approaches that can be cast in hardware are likely to form the basis of next generation camera control systems.

This review of the state-of-the-art has presented an overview of camera control in computer graphics from interactive techniques to completely automated camera systems. An analysis of cinematic and photographic practice has helped us to develop classification on the expressiveness of camera shots, ranging from geometric to perceptual and aesthetic. This classification was undertaken with respect to both the geometric expressiveness of the approaches and the solution mechanisms, from interactive mappings of user inputs, to automatic reactive camera control.

What has become clear is that the next generation of camera control systems requires not only efficient implementations but empirical work on the characterization of higher level of properties that will facilitate the maintenance of aesthetic and emotionally evocative views. Whilst the aesthetic properties are likely to be founded on an adequate cognitive model, attempts to exploit editing rules to effectively engage the user are still very much in their infancy [FF04, TBN00]. We see this as a key area for interdisciplinary research by both computer scientists and cognitive psychologists, not only to the benefit of computer graphics, but as a window on the nature of cognition itself.

## References

- [Ari76] ARIJON D.: *Grammar of the Film Language*. Hastings House Publishers, 1976. 3, 6, 12, 13
- [AVF04] ANDÚJAR C. G., VÁZQUEZ P. P. A., FAIRÉN M. G.: Way-finder: Guided tours through complex walk-through models. *Comput. Graph. Forum* 23, 3 (2004), 499–508. 8
- [Bec02] BECKHAUS S.: *Dynamic Potential Fields for Guided Exploration in Virtual Environments*. PhD thesis, Fakultät für Informatik, University of Magdeburg, 2002. 8
- [BGL98] BARES W. H., GREGOIRE J. P., LESTER J. C.: Realtime Constraint-Based Cinematography for Complex Interactive 3D Worlds. In *Procs of AAAI-98/IAAI-98* (1998), pp. 1101–1106. 16, 21
- [BGLC04] BENHAMOU F., GOUALARD F., LANGUÉNOU E., CHRISTIE M.: Interval constraint solving for camera control and motion planning. *ACM Transactions on Computational Logic* (October 2004), 732–767. 16
- [BJH99] BOWMAN D. A., JOHNSON D. B., HODGES L. F.: Testbed evaluation of virtual environment interaction techniques. In *Proceedings of the ACM symposium on Virtual reality software and technology* (London, United Kingdom, 1999), pp. 26–33. 9
- [BKF\*02] BURTONYK R., KHAN A., FITZMAURICE G., BALAKRISHAN R., KURTENBACH G.: Stylecam: Interactive stylized 3d navigation using integrated spatial & temporal controls. In *ACM IUST Symposium on User Interface Software & Technology* (2002). 8
- [BL97] BARES W. H., LESTER J. C.: Cinematographic user models for automated realtime camera control in dynamic 3D environments. In *Proceedings of the sixth International Conference on User Modeling* (Vien New York, 1997), Jameson A Paris C T. C., (Ed.), Springer-Verlag, pp. 215–226. 20
- [BL99] BARES W. H., LESTER J. C.: Intelligent Multi-Shot Visualization Interfaces for Dynamic 3D Worlds. In *IUI '99: Proceedings of the 4th international conference on Intelligent user interfaces* (New York, NY, USA, 1999), ACM Press, pp. 119–126. 11, 16, 18, 21
- [Bli88] BLINN J.: Where am I? what am I looking at? *IEEE Computer Graphics and Applications* (July 1988), 76–81. 11, 20, 21
- [BMBT00] BARES W., MCDERMOTT S., BOUDREAUX C., THAINIMIT S.: Virtual 3D Camera Composition from Frame Constraints. In *MULTIMEDIA '00: Procs. of the eighth ACM international conference on Multimedia* (2000), ACM Press, pp. 177–186. 16, 17, 18, 20, 21
- [BRZL98] BARES W. H., RODRIGUEZ D. W., ZETTMAYER L. S., LESTER J. C.: Task-sensitive cinematography interfaces for interactive 3d learning environments. In *Proceedings Fourth International conference on Intelligent User Interfaces* (1998), pp. 81–88. 5
- [BS05] BORDOLOI U., SHEN H.-W.: View selection for volume rendering. In *16th IEEE Visualization Conference (VIS 2005)* (2005), p. 62. 21
- [BTMB00] BARES W., THAINIMIT S., MCDERMOTT S., BOUDREAUX C.: A model for constraint-based camera planning. In *Smart Graphics AAAI 2000 Spring Symposium* (Stanford, California, March 20-22 2000). 16, 20
- [But97] BUTZ A.: Animation with CATHI. In *Proceedings of American Association for Artificial Intelligence/IAAI '97* (1997), AAAI Press, pp. 957–962. 11
- [BZRL98] BARES W. H., ZETTMAYER L. S., RODRIGUEZ D. W., LESTER J. C.: Task-Sensitive Cinematography Interfaces for Interactive 3D Learning Environments. In *Intelligent User Interfaces* (1998), pp. 81–88. 18
- [CAH\*96] CHRISTIANSON D. B., ANDERSON S. E., HE L., SALESIN D. H., WELD D. S., COHEN M. F.: Declarative Camera Control for Automatic Cinematography. In *Proceedings of the American Association for Artificial Intelligence 1996* (1996), pp. 148–155. 11, 20, 21
- [CHL\*98] CHIOU R., HAUFMAN A., LIANG Z., HONG

- L., ACHNIOTOU M.: Interactive path planning for virtual endoscopy. In *Proc. of the IEEE Nuclear Science and Medical Imaging Conference* (1998). 9
- [CL03] CHRISTIE M., LANGUÉNOU E.: A Constraint-Based Approach to Camera Path Planning. In *Proceedings of the Third International Symposium on Smart Graphics* (2003), vol. 2733 of *Lecture Notes in Computer Science*, Springer, pp. 172–181. 20, 21
- [CLDM03] COURTY N., LAMARCHE F., DONIKIAN S., MARCHAND E.: A Cinematography System for Virtual Storytelling. In *Int. Conf. on Virtual Storytelling, ICVS'03* (Toulouse, France, November 2003), Balet O., Subsol G., Torguet P., (Eds.), vol. 2897 of *Lecture Notes in Computer Science*, pp. 30–34. 20
- [CLG02] CHRISTIE M., LANGUÉNOU E., GRANVILLIERS L.: Modeling Camera Control with Constrained Hypertubes. In *Procs of CP 2002* (Ithaca, NY, USA, September 9-13 2002), Lecture Notes in Computer Science (LNCS), Springer-Verlag. 15
- [CM01] COURTY N., MARCHAND E.: Computer animation: A new application for image-based visual servoing. In *In Proceedings of IEEE Int. Conf. on Robotics and Automation, ICRA'2001* (2001), vol. 1, pp. 223–228. 10, 21
- [CMS88] CHEN M., MOUNTFORD S., SELLEN A.: A study in interactive 3d rotation using 2d input devices. In *Computer Graphics (Proceedings SIGGRAPH '88)* (Aug. 1988), Catmull E. E., (Ed.), vol. 22-4, pp. 121–130. 8
- [CN05] CHRISTIE M., NORMAND J.-M.: A semantic space partitioning approach to virtual camera control. In *In Proceedings of the Annual Eurographics Conference* (2005), vol. 24, pp. 247–256. 17, 18, 19, 20, 21
- [Cou02] COURTY N.: *Animation référencée vision : de la tâche au comportement*. PhD thesis, INSA Rennes, soutenue à l'Université de Rennes I, November 2002. 18
- [DGZ92] DRUCKER S. M., GALYEAN T. A., ZELTZER D.: Cinema: A System for Procedural Camera Movements. In *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics* (New York, NY, USA, 1992), ACM Press, pp. 67–70. 9, 16, 21
- [Dru94] DRUCKER S. M.: *Intelligent Camera Control for Graphical Environments*. PhD thesis, School of Architecture and Planning, Massachusetts Institute of Technology MIT Media Lab, 1994. 20
- [DZ95] DRUCKER S. M., ZELTZER D.: Camdroid: A System for Implementing Intelligent Camera Control. In *Symposium on Interactive 3D Graphics* (1995), pp. 139–144. 18
- [ECR93] ESPIAU B., CHAUMETTE F., RIVES P.: A new approach to visual servoing in robotics. In *Selected Papers from the Workshop on Geometric Reasoning for Perception and Action* (London, UK, 1993), Springer-Verlag, pp. 106–136. 10
- [FF04] FRIEDMAN D. A., FELDMAN Y. A.: Knowledge-Based Cinematography and Its Applications. In *Proceedings of the 16th European Conference on Artificial Intelligence, ECAI'2004* (2004), IOS Press, pp. 256–262. 22
- [Gio04] GIORS J.: The full spectrum warrior camera system. In *GDC '04 : Game Developers Conference 2004* (2004). 4, 18
- [GRMS01] GOOCH B., REINHARD E., MOULDING C., SHIRLEY P.: Artistic composition for image creation. In *Eurographics Workshop on Rendering* (2001), pp. 83–88. 20
- [GW92] GLEICHER M., WITKIN A.: Through-the-lens camera control. In *Proceedings of ACM SIGGRAPH'92* (1992), pp. 331–340. 9, 20, 21
- [HCS96] HE L., COHEN M. F., SALESIN D. H.: The virtual cinematographer: A paradigm for automatic real-time camera control and directing. In *SIGGRAPH 96 Conference Proceedings* (Aug. 1996), Rushmeier H., (Ed.), Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 217–224. held in New Orleans, Louisiana, 04-09 August 1996. 20, 21
- [HHO05] HOWLETT S., HAMILL J., O'SULLIVAN C.: Predicting and evaluating saliency for simplified polygonal models. *ACM Trans. Appl. Percept.* 2, 3 (y 05), 286–308. 21
- [HHS01] HALPER N., HELBING R., STROTHOTTE T.: A camera engine for computer games: Managing the trade-off between constraint satisfaction and frame coherence. In *In Proceedings of the Eurographics'2001 Conference* (2001), vol. 20, pp. 174–183. 11, 19, 20, 21
- [HMK\*97] HONG L., MURAKI S., KAUFMAN A., BARTZ D., HE T.: Virtual voyage: interactive navigation in the human colon. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 27–34. 9
- [HO00] HALPER N., OLIVIER P.: CAMPLAN: A Camera Planning Agent. In *Smart Graphics 2000 AAAI Spring Symposium* (March 2000), pp. 92–100. 14, 18, 19, 20, 21
- [HW97] HANSON A., WERNERT E.: Constrained 3d navigation with 2d controllers. In *IEEE Visualization* (1997), pp. 175–182. 7, 8
- [JL98] JARDILLIER F., LANGUÉNOU E.: Screen-Space Constraints for Camera Movements: the Virtual Camera-man. In *Procs. of EUROGRAPHICS-98* (1998), Ferreira N., Göbel M., (Eds.), vol. 17, Blackwell Publishers, pp. 175–186. ISSN 1067-7055. 13, 15, 20, 21
- [JPK98] JUNG M., PAIK D., KIM D.: A camera control interface based on the visualization of subspaces of the 6d motion space of the camera. In *Proceedings of IEEE Pacific Graphics'98* (1998), pp. 198–208. 9

- [Kat91] KATZ S.: *Film Directing Shot by Shot: Visualizing from Concept to Screen*. Michael Wiese Productions, 1991. 3, 6
- [KKH95] KUNG M. H., KIM M. S., HONG S.: Through-the-lens camera control with a simple jacobian matrix. In *Proceedings of Graphics Interface '95* (1995), pp. 117–178. 10
- [KKS\*05] KHAN A., KOMALO B., STAM J., FITZMAURICE G., KURTENBACH G.: Hovercam: interactive 3d navigation for proximal object inspection. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2005), ACM Press, pp. 73–80. 8
- [KvD79] KOENDERINK J., VAN DOORN J.: The internal representation of solid shape with respect to vision. *Biological Cybernetics* 32 (1979), 211–216. 17
- [Lat91] LATOMBE J.: *Robot Motion Planning*. Kluwer Academic Publishers, 1991. 8
- [LH04] LI T.-Y., HSU S.-W.: An intelligent 3d user interface adapting to user control behaviors. In *IUI '04: Proceedings of the 9th international conference on Intelligent user interface* (New York, NY, USA, 2004), ACM Press, pp. 184–190. 9
- [LT00] LI T.-Y., TING H.-K.: An intelligent user interface with motion planning for 3d navigation. In *Proceedings of the IEEE VR2000 International Conference* (2000), pp. 177–. 8
- [LVJ05] LEE C. H., VARSHNEY A., JACOBS D. W.: Mesh saliency. *ACM Trans. Graph.* 24, 3 (2005), 659–666. 21
- [Mas65] MASCELLI J.: *The Five C's of Cinematography: Motion Picture Filming Techniques*. Cine/Grafic Publications, Hollywood, 1965. 3, 6, 7, 13
- [MC02] MARCHAND E., COURTY N.: Controlling a camera in a virtual environment. *The Visual Computer Journal* 18, 1 (2002), 1–19. 10, 18, 20
- [MCR90] MACKINLAY J. D., CARD S. K., ROBERTSON G. G.: Rapid Controlled Movement Through a Virtual 3D Workspace. *Computer Graphics* 24, 4 (Aug. 1990), 171–176. 9
- [MH98] MARCHAND E., HAGER G.: Dynamic sensor planning in visual servoing. In *IEEE Int. Conf. on Robotics and Automation, ICRA'98* (Leuven, Belgium, May 1998), vol. 3, pp. 1988–1993. 10, 18
- [MLB02] MCDERMOTT S., LI J., BARES W.: Storyboard Frame Editing for Cinematic Composition. In *IUI '02: Proceedings of the 7th international conference on Intelligent user interfaces* (New York, NY, USA, 2002), ACM Press, pp. 206–207. 21
- [Moo66] MOORE R.: *Interval Analysis*. Prentice Hall, 1966. 15
- [NLK93] NODINEM C., LOCHER J., KRUNPINSKI E.: *The role of formal art training on perception and aesthetic judgement of art compositions*. Leonardo, 1993. 7
- [NO03] NIEUWENHUISEN D., OVERMARS M. H.: *Motion Planning for Camera Movements in Virtual Environments*. Tech. Rep. UU-CS-2003-004, Institute of Information and Computing Sciences, Utrecht University, 2003. 8, 20
- [OHPL99] OLIVIER P., HALPER N., PICKERING J., LUNA P.: Visual Composition as Optimisation. In *AISB Symposium on AI and Creativity in Entertainment and Visual Art* (1999), pp. 22–30. 14, 16, 20, 21
- [Pal96] PALAMIDESE P.: A Camera Motion Metaphor Based on Film Grammar. *Journal of Visualization and Computer Animation* 7, 2 (1996), 61–78. 12
- [PBG92] PHILLIPS C. B., BADLER N. I., GRANIERI J.: Automatic viewing control for 3d direct manipulation. In *Proceedings of the 1992 symposium on Interactive 3D graphics* (1992), ACM Press New York, NY, USA, pp. 71–74. 7, 18
- [PD90] PLANTINGA H., DYER C. R.: Visibility, Occlusion, and the aspect graph. *International Journal of Computer Vision* 5, 2 (Nov 1990), 137–160. 17
- [Pic02] PICKERING J. H.: *Intelligent Camera Planning for Computer Graphics*. PhD thesis, Department of Computer Science, University of York, 2002. 14, 16, 17, 19, 20, 21
- [Roz99] ROZENHOLTZ R.: A simple saliency model predicts a number of motion popout phenomena. *Vision Research* 39, 19 (1999), 3157–3163. 21
- [SF91] SELIGMANN D. D., FEINER S.: Automated generation of intent-based 3d illustrations. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1991), ACM Press, pp. 123–132. 5
- [SF93] SELIGMANN D. D., FEINER S.: Supporting interactivity in automated 3d illustrations. In *IUI '93: Proceedings of the 1st international conference on Intelligent user interfaces* (New York, NY, USA, 1993), ACM Press, pp. 37–44. 5
- [SGLM03] SALOMON B., GARBER M., LIN M. C., MANOCHA D.: Interactive navigation in complex environments using path planning. In *SI3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics* (New York, NY, USA, 2003), ACM Press, pp. 41–50. 8
- [Sho92] SHOEMAKE K.: Arcball: a user interface for specifying three-dimensional orientation using a mouse. In *Proceedings of Graphics Interface '92* (May 1992), pp. 151–156. 7
- [Sny92] SNYDER J.: Interval analysis for computer graphics. In *J. M. Snyder, Interval analysis for computer graph-*

- ics. *Proceedings of SIGGRAPH'92, in ACM Computer Graphics 26, 2 (July 1992), 121–130.* (1992). 15
- [SP05] SOKOLOV D., PLEMENOS D.: Viewpoint quality and scene understanding. In *VAST 2005: Eurographics Symposium Proceedings*. (ISTI-CNR Pisa, Italy, November 2005), Mudge M., Ryan N., Scopigno R., (Eds.), Eurographics Association, pp. 67–73. 21
- [SS02] STOEV S. L., STRASSER W.: A case study on automatic camera placement and motion for visualizing historical data. In *VIS '02: Proceedings of the conference on Visualization '02* (Washington, DC, USA, 2002), IEEE Computer Society, pp. 545–548. 21
- [TBGT91] TURNER R., BALAGUER F., GOBBETTI E., THALMANN D.: Physically-based interactive camera motion control using 3D input devices. In *Scientific Visualization of Physical Phenomena*, Patrikalakis N. M., (Ed.). Springer Verlag, 1991, pp. 135–145. 8
- [TBN00] TOMLINSON B., BLUMBERG B., NAIN D.: Expressive autonomous cinematography for interactive virtual environments. In *Proceedings of the Fourth International Conference on Autonomous Agents* (Barcelona, Catalonia, Spain, 2000), Sierra C., Gini M., Rosenschein J. S., (Eds.), ACM Press, pp. 317–324. 21, 22
- [TRC01] TAN D. S., ROBERTSON G. G., CZERWINSKI M.: Exploring 3d navigation: combining speed-coupled flying with orbiting. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2001), ACM Press, pp. 418–425. 9
- [VFSH01] VÁZQUEZ P.-P., FEIXAS M., SBERT M., HEIDRICH W.: Viewpoint selection using viewpoint entropy. In *VMV '01: Proceedings of the Vision Modeling and Visualization Conference 2001* (2001), Aka GmbH, pp. 273–280. 21
- [WFH\*97] WOOD D. N., FINKELSTEIN A., HUGHES J. F., THAYER C. E., SALESIN D. H.: Multiperspective panoramas for cel animation. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 243–250. 11
- [WO90] WARE C., OSBORNE S.: Exploration and virtual camera control in virtual three dimensional environments. In *SI3D '90: Proceedings of the 1990 symposium on Interactive 3D graphics* (New York, NY, USA, 1990), ACM Press, pp. 175–183. 7
- [WW97] WEINSHALL D., WERMAN M.: On view likelihood and stability. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19, 2 (1997), 97–108. 21
- [XH98] XIAO D., HUBBOLD R. J.: Navigation guided by artificial force fields. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems* (1998), Wesley A., (Ed.), pp. 179–186. 8
- [ZF99] ZELEZNIK R. C., FORSBERG A. S.: Unicam - 2d gestural camera controls for 3d environments. In *Symposium on Interactive 3D Graphics* (1999), pp. 169–173. 9
- [ZRS03] ZANCANARO M., ROCCHI C., STOCK O.: Automatic video composition. In *Proceedings of the Third International Symposium on Smart Graphics* (2003), vol. 2733 of *Lecture Notes in Computer Science*, Springer, pp. 192 – 201. 12
- [ZSA03] ZANCANARO M., STOCK O., ALFARO I.: Using cinematic techniques in a multimedia museum guide. In *Proceedings of Museums and the Web 2003* (March 2003). 12