Virtual camera control using dynamic spatial partitions

Christophe Lino

ANNÉE 2013

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

Ecole doctorale Matisse

présentée par

Christophe Lino

Préparée à l'unité de recherche UMR 6074 - IRISA
Institut de Recherche en Informatique et Systèmes Aléatoires
ISTIC - UFR Informatique et électronique

Virtual Camera Control using Dynamic Spatial Partitions

Contrôle de caméra virtuelle à base de partitions spatiales dynamiques

Thèse soutenue à Rennes
le 3 Octobre 2013

devant le jury composé de :

Éric MARCHAND
Professeur, Université de Rennes 1 /
Président

Daniel COHEN-OR
Professor, Tel Aviv University /
Rapporteur

Michael YOUNG
Professor, North Carolina State University /
Rapporteur

Rémi RONFARD
Chargé de Recherche HDR, INRIA /
Examinateur

Kadi BOUATOUCH
Professeur, Université de Rennes 1 /
Directeur de thèse

Marc CHRISTIE
Maître de Conférences, Université de Rennes 1 /
Co-directeur de thèse

Acknowledgments

I first thank my two advisers Pr Kadi Bouatouch and Dr Marc Christie for all the things they taught me during this thesis and even before, for being the first to trust in my capacity for doing a good PhD thesis as well as for their every day cheerfulness. It has provided me with ideal working conditions which I really appreciated. I particularly thank Marc for his patience and for all the things he has done in order for this thesis to begin and also to be completed. He has always been supportive (from any point of view) in good and even more in difficult moments. And he has been a really great adviser with a clear vision of the field and with constructive remarks, which has helped me achieve my full potential throughout the last three years. It has been a great pleasure to work with Marc and I look forward to working with him again.

I kindly thank all members of the jury for their constructive remarks, which helped me to improve this manuscript. I particularly thank Éric Marchand for accepting to chair the jury; as well as Daniel Cohen-Or and Michael Young, for having accepted the charge of reviewing this thesis.

I thank all the people who collaborated to the different works and projects which are presented and discussed hereafter. I thank Marc Christie (again) and Fabrice Lamarche for their co-supervision of my Master thesis, which has made me inexorably attracted to motion planning and particularly virtual camera control problems and is at the origin of this PhD thesis. I thank Patrick Olivier for proof reading some of the papers, and for welcoming me on many occasions in the Culture Lab Newcastle. I thank Guy Schofield for the instructive and constructive discussions we had about real cinematography. I thank William Bares, Roberto Ranon, Rémi Ronfard and Mathieu Chollet for the constructive discussions we had on various aspects of real/virtual film editing, and for their participation in some common works presented in this thesis. I thank Arnav Jhala for welcoming me at the University of California Santa Cruz and for the instructive discussions we have had about narrative and cinematic discourse.

I thank the permanent and non-permanent members of the historical (and even legendary) Bunraku research team; and particularly of the Mimetic team, for all the relevant remarks they made me before, during and after the writing of this thesis. I thank my successive roommates, and particularly Steve for the studious atmosphere of the room as well as for all the interesting discussions we had on a variety of topics. I thank all the nice people I had the privilege to meet during my time at IRISA, for the friendly atmosphere both in the corridors and during the numerous coffee breaks we had together.

Je remercie ma famille, et en particulier mes parents pour tout le soutiens qu'ils m'ont apporté tout au long de ces (longues) années d'études. Enfin, je remercie Aurélie

qui a vécu avec moi les bons et les moins bons moments de cette thèse, et qui a contribué pour une large part à sa réussite. Je la remercie pour tout le soutiens qu'elle m'a apporté durant ces trois années, et plus généralement pour tout ce qu'elle représente pour moi depuis toujours.

Résumé en français

Avec les avancées des techniques de rendu, le réalisme des environnements virtuels ne cesse de s'améliorer. Il est donc important que ces progrès s'accompagnent d'une meilleure présentation de leur contenu. Une caméra virtuelle peut-être vue comme une fenêtre à travers laquelle le contenu d'un environnement virtuel est présenté à un utilisateur. Une configuration de caméra (*i.e.* une position fixée et une orientation fixée de la caméra) permet de fournir une vue (*i.e.* une image) montrant une partie d'un environnement ; en fournissant un flux d'images sur cet environnement, on peut alors aider un utilisateur à se construire une représentation mentale du contenu de l'environnement (*i.e.* se faire une idée de l'agencement spatial et temporel des éléments qui le composent, tels que l'existence d'un objet dans l'environnement ou sa position à un moment donné). Il est alors essentiel de bien choisir les points de vue que l'on va utiliser pour présenter un environnement. Le contrôle d'une caméra virtuelle est un composant essentiel dans un grand nombre d'applications, telles que la visualisation de données, la navigation dans les environnements virtuels (un musée virtuel par exemple), la narration virtuelle (*e.g.* un film d'animation) ou encore les jeux vidéos. Il existe trois aspects essentiels lorsque l'on contrôle une caméra virtuelle, qui sont : le choix des points de vue, le choix des mouvements de camera et le montage (*i.e.* décider du moment et de la manière d'effectuer une coupure entre les points de vue ou d'effectuer un mouvement de caméra). La nature du contrôle d'une caméra virtuelle peut varier selon les besoins, d'un contrôle semi-automatisé tel que dans les applications interactives, à un contrôle complètement automatisé. On peut cependant identifier trois problèmes transverses : (i) la composition à l'écran (*i.e.* la disposition des éléments à l'écran), (ii) la planification de trajectoires de caméra et (iii) le montage.

Le contrôle de caméra virtuelle consiste en la recherche, à chaque pas de temps, du meilleur point de vue (par rapport à un ensemble de propriétés) dans un espace à 7 degrés de liberté. Par conséquent, on peut le voir comme une sous-classe de problèmes de planification de mouvements. Ce genre de techniques est généralement utilisé pour rechercher, dans un espace de grande dimension, un mouvement ou une trajectoire (d'un bras articulé par exemple) qui évite de rentrer en collision avec les obstacles de l'environnement. Toutefois, le contrôle d'une caméra virtuelle ne se résume pas simplement à trouver une suite discrète de configurations telle qu'il n'y ait pas de collision avec des éléments de l'environnement ; cela nécessite en supplément que cette suite de points de vue satisfasse un certain nombre de contraintes, qui peuvent être liées à l'agencement des éléments à l'écran (par exemple maintenir un angle de vue ou maintenir la visibilité de sujets clés) ou bien à la trajectoire de la caméra (par exemple maintenir une certaine continuité entre les points de vue successifs).

Des chercheurs se sont intéressé aux trois problèmes principaux que nous avons mentionnés, et ont proposé des techniques qui vont d'un contrôle complètement manuel jusqu'à un contrôle complètement automatisé des degrés de liberté de la caméra. À partir de l'étude de la littérature, nous effectuons dans cette thèse trois observations. Premièrement, il n'existe aucun modèle générique prenant en compte les trois aspects du contrôle d'une caméra virtuelle (la composition visuelle, la planification de trajectoires et le montage). En effet, les travaux de recherche se sont généralement focalisés sur un seul, voire deux de ces aspects. En outre, la composition, la planification de trajectoires et le montage impliquent une bonne gestion de la visibilité, problème qui a été assez peu étudié dans le domaine. La prise en compte simultanée de ces quatre aspects constitue pourtant une base qui semble indispensable pour construire des applications graphiques plus évoluées (*e.g.* pour la narration virtuelle ou les jeux vidéo). De plus, les techniques existantes manquent d'expressivité ; elle n'offrent qu'une prise en compte limitée des styles/genres cinématographiques, et s'intéressent assez peu à fournir des moyens de les mettre en œuvre. Deuxièmement, nous pensons qu'il est nécessaire que l'utilisateur puisse interagir avec ces différents aspects. Bien que les techniques automatisées permettent d'obtenir de bons résultats, le résultat attendu par un réalisateur est souvent plus subtil et ne peut pas forcément être modélisé. Les technique automatisées existantes ne fournissent pas de réelle assistance aux utilisateurs dans la construction d'une séquence cinématographique. Enfin, la composition visuelle est un élément central dans le placement d'une caméra. Les problèmes de composition évolués sont généralement modélisés comme une fonction d'objectif, construite comme une agglomération de fonctions de qualité liées chacune à une contrainte que l'on désire satisfaire (*e.g.* la position à l'écran, la taille projetée ou encore la visibilité de certains éléments de la scène) ; les chercheurs utilisent ensuite des techniques d'optimisation numérique pour chercher un point de vue qui maximise cette fonction d'objectif. Cependant, agréger un ensemble de fonctions de qualité réduit la capacité à guider le processus de résolution à travers l'espace de recherche, ce qui conduit à l'exploration de larges zones de l'espace de recherche pour lesquels il n'existe aucune solution. Ce problème très spécifique de composition visuelle est ainsi transformé en un processus de recherche générique pour lequel il est difficile de proposer des heuristiques générales permettant d'accélérer la recherche.

À partie des ces trois considérations, nous avons identifié trois axes de recherche :

- vers un système de cinématographie complètement intégré, c'est à dire qui permette de prendre en compte le calcul de points de vue, la planification de trajectoires, les aspects liés au montage et à la visibilité de manière interactive, tout en prenant en considération un certain nombre d'éléments de style cinématographique ;
- vers une approche interactive qui assiste l'utilisateur dans son processus de construction d'un film, et qui lui permette d'avoir un certain degré de contrôle sur le montage final, par la proposition d'une approche hybride combinant des calculs automatisées avec une interaction plus directe de l'utilisateur ;
- vers une approche efficace du problème de composition visuelle.

Nous détaillons ci-dessous les contributions de la thèse, qui répondent à ces trois

axes de recherche.

Résumé de nos contributions

Un moteur cinématographique pour les environnements 3D interactifs

Nous proposons une approche unificatrice du problème de cinématographie interactive, découlant sur un moteur cinématographique complètement intégré appelé CineSys. CineSys gère les points de vue, le montage et la planification de trajectoire dans un contexte temps-réel (ou temps-interactif). Nous présentons un modèle expressif de montage, qui s'attaque à la complexité intrinsèque de problèmes très classiques tels que la détermination de visibilité et la planification de trajectoires – qui sont des pré-requis au contrôle temps-réel d'une caméra virtuelle – ainsi qu'à des problème de plus haut niveau liés au maintien d'une certaine continuité dans la succession de plans cinématographiques. Notre moteur de cinématographie encode des idiomes cinématographiques et des règles dites de "continuity-editing", afin de produire des montages et des trajectoires de caméra appropriés à partir d'un ensemble d'événements narratifs et d'indicateurs de style. Nous présentons le concept de *Volumes Directeur*, une partition spatiale novatrice sur laquelle repose notre modèle de montage. Ces Volumes Directeurs fournissent une caractérisation des régions de visibilité (avec visibilité totale, visibilité partielle ou occultation totale) ainsi que des points de vues stéréotypiques (les Volumes Sémantiques). CineSys raisonne ensuite sur ces Volumes Directeurs afin d'identifier quand et comment les transitions entre plans (mouvements de caméra ou coupures) doivent être effectuées. Ce raisonnement sémantique et géométrique repose sur une mise en œvre basée filtrage de conventions cinématographiques. Notre processus de raisonnement est de plus suffisamment expressif pour offrir la possibilité d'implémenter une variété de styles cinématographique. Nous démontrons l'expressivité de notre modèle en faisant varier un certains nombre d'indicateurs stylistiques, tels que la visibilité des sujets, le rythme de coupures (ou pacing), la dynamicité de la camera, ou encore la dimension narrative. L'expressivité de notre modèle est en contraste très net avec les approches existantes qui sont soit de nature procédurales, soit ne permettent pas des calculs interactifs, soit ne prennent pas réellement en considération la visibilité des sujets clés.

Prise en compte de la participation d'un réalisateur dans le processus de montage

Nous présentons un cadre théorique novateur pour la cinématographie virtuelle et le montage qui ajoute une évaluation de la qualité des plans, des coupures et du rythme des coupures. Nous proposons ensuite deux approches basées classement au problème de montage d'un film, qui se basent à la fois sur notre moteur de cinématographie CineSys et sur les métriques d'évaluation que nous avons proposée, pour aider un réalisateur dans son processus créatif. Nous présentons une stratégie de recherche performante de la meilleure suite de plans parmi un grand nombre de candidats générés par les idiomes

traditionnellement utilisés, et qui permet à l'utilisateur d'avoir une certaine maîtrise du montage final. Nous permettons de plus l'application d'un rythme de coupures reposant sur un modèle fondé de durées de plans. Nous présentons ensuite un système interactif d'aide au montage, qui amène un processus novateur de création d'un film, basé sur la collaboration interactive de la créativité d'un réalisateur et du potentiel de calcul d'un système automatisé. Ce processus de création permet une exploration rapide d'un grand nombre de possibilités de montage, et une prévisualisation rapide de films d'animation générés par ordinateur.

Une approche efficace du contrôle d'une caméra virtuelle : l'Espace Torique

Nous présentons une approche efficace et innovante du problème de placement d'une caméra virtuelle ayant le potentiel de remplacer un certain nombre de techniques précédentes de contrôle de caméra, et qui laisse entrevoir de grandes possibilités d'intégration de techniques de composition visuelles plus évoluées dans un nombre important d'application d'infographie. dans un premier temps, nous présentons un modèle paramétrique assez simple (la *Variété Torique*) qui résout le problème de positionnement exact de deux sujets à l'écran. Nous utilisons ensuite ce concept pour nous attaquer à des problèmes liés à la tâche de positionner une caméra virtuelle afin de satisfaire des spécifications de positionnement d'éléments à l'écran. En particulier, nous proposons la toute première méthode de résolution du problème du "vaisseau spatial" de Blinn [Bli88] utilisant une formulation purement algébrique plutôt qu'un processus itératif. De même, nous montrons comment formuler des problèmes de composition simples, généralement gérés en 6D, comme une recherche dans un espace 2D sur la surface d'une variété Dans un second temps, nous effectuons une extension de notre modèle en un espace 3D (l'*Espace Torique*) dans lequel nous intégrons la plupart des propriétés visuelles classique qui sont utilisées dans la littérature [RCU10]. Nous exprimons la taille de sujets clés (ou distance à la caméra), l'angle de vue (ou "vantage angle") et une description plus souples du positionnement des sujets à l'écran comme des ensembles de solutions en 2D dans l'Espace Torique. Nous détaillons la résolution théorique de la combinaison d'un nombre indéterminé de tels ensembles de solutions, alors que cette combinaison s'évère très difficile (voire impossible) à opérer dans l'espace classique des configurations de caméra en 6D. Nous montrons de plus comment les règles de montage ("continuity-editing") peuvent être exprimées facilement comme un ensemble de contraintes (*e.g.* distance, angle de vue) puis combinés avec les autres contraintes da,s l'Espace Torique. Nous présentons enfin une technique efficace qui génère un ensemble de points de vues représentatifs de la solution globale d'un problème de composition, et effectue une estimation intelligente de la visibilité afin de sélectionner le meilleur point de vue de cet ensemble. Du fait de la réduction de l'espace de recherche inhérente à notre Espace Torique, le bénéfice en temps de calcul donne un avantage sérieux à notre approche.

Contents

List of Figures

List of Tables

Introduction 1

With the advances in rendering, the realism of virtual environments is continuously increasing. It is therefore necessary to relate such progress to better presentation of their contents. A virtual camera is a window through which such content can be presented to a viewer. A single camera viewpoint provides a view (*i.e.* a single image) conveying a subpart of an environment; the flow of images then assists a viewer in the construction of a mental representation (spatial and/or temporal layout) of this environment. It is therefore necessary to carefully select viewpoints on these environments. Virtual camera control is an essential component for a large range of applications, such as data visualization, virtual navigation (in a museum for instance), virtual storytelling or 3D games. The three essential aspects of virtual camera control are the choice of the viewpoint, the choice of the camera motions and the editing (*i.e.* deciding when and where to perform cuts between viewpoints or motions). The nature of virtual camera control can vary *w.r.t.* needs, from semi-automated control such as in interactive applications, to fully automated approaches. However, one can identify three transversal issues: (i) the on-screen composition (*i.e.* visual arrangement of elements on the screen), (ii) the planning of camera paths and (iii) the editing.

Virtual camera control consists in searching, at each time step, for the best camera viewpoint (*w.r.t.* a set of properties) in a 7 *dof*s (degrees of freedom) search space. Consequently, it can be viewed as a sub-class of Motion Planning issues. Such techniques are generally used to search for a motion or path (of an articulated arm for instance) with no collision with obstacles of an environment in high dimensions. However, controlling a camera cannot be summarized as the search for a discrete series of configurations with no collision; it also requires the sequence of camera viewpoints to satisfy a set of constraints on the on-screen layout of elements (*e.g.* maintain a view angle or the visibility of key subjects) or on the camera path (*e.g.* enforce coherency between successive camera viewpoints).

Researchers tackled the three main issues we have mentioned, and proposed a number of techniques ranging from manual control to fully automated control of camera parameters. From the literature, three observations can be made. Firstly, there is no general model accounting for the three aspects (visual composition, path planning and editing). Indeed, researchers generally focused on one or two of these aspects only. In complement, composition, planning and editing need to deal with visibility, which has been under-addressed in the field. The consideration of these four aspects together is a crucial basis to build more evolved computer graphics applications (*e.g.* in virtual storytelling or 3D games). Furthermore, existing techniques lack expressiveness,

i.e. they provide limited account of cinematographic styles/genres and means to enforce them. Secondly, we believe there is a necessity to let the user interact. Though automated techniques provide good results, the result intended by the user is generally more subtle and cannot be easily expressed. Existing automated techniques do not really provide assistance to users in the task of constructing a movie. Lastly, the visual composition is a central element of camera placement. Evolved composition problem are commonly modeled as an objective function constructed as the agglomeration of quality functions related to each desired visual property (*e.g.* on-screen position, size or visibility of scene elements); researchers then use optimization-based techniques to find a camera that maximizes this objective function. Aggregating quality functions for all properties together however reduces the capacities to guide the solving process through the search space, therefore leading to techniques which explore large areas of the search space without solutions. The specific problem of composition is transformed into a general search process for which it is difficult to propose efficient and general heuristics.

In this thesis, we investigate the control of a virtual camera in the context of virtual cinematography in interactive 3D environments. The thesis is organized as follows. In a first part, we present our motivations, as well as a state of the art of existing techniques.

In a second part, we present a unifying approach to the problem of interactive cinematography. Our approach handles viewpoint computation, editing and planning in a real-time context. We address the innate complexity of well understood problems such as visibility determination and path planning required in real-time camera control, while tackling higher-level issues related to continuity between successive shots in an expressive editing model. Our model relies on a spatial partitioning, the *Director Volumes*, on which we reason to identify how, when and where shot transitions should be performed. This reasoning relies on a filtering-based encoding of cinematic conventions together with the possibility to implement different directorial styles. The expressiveness of our model stands in stark contrast to existing approaches that are either procedural in character, non-interactive or do not account for proper visibility of key subjects.

In a third part, we introduce a novel framework for virtual cinematography and editing which adds a quantitative evaluation of the key editing components. We further propose two ranking-based approaches to editing a movie, that build upon the proposed evaluation metrics to assist a filmmaker in his creative process. We introduce an efficient search strategy for finding the best sequence of shots from a large number of candidates generated from traditional film idioms, while providing the user with some control on the final edit. We further enable enforcing the pace in cuts by relying on a well-founded model of shot durations. We then present an interactive assistant (*The Director's Lens*) whose result is a novel workflow based on interactive collaboration of human creativity with automated intelligence. This workflow enables efficient exploration of a wide range of cinematographic possibilities, and rapid production of computer-generated animated movies.

In a fourth part, we introduce a parametric model (the Toric Manifold) to solve a range of problems that occur in the task of positioning a virtual camera given exact on-

screen specifications. Our model solves Blinn's spacecraft problem [Bli88] by using an algebraic formulation rather than an iterative process. It casts simple camera optimization problems mostly conducted in 6D into searches inside a 2D space on a manifold surface. We further extend this model as a 3D space (the Toric Space) in which we integrate most of the classical visual properties employed in the literature [RCU10]. Because of the reduction in the search space inherent to our Toric Space, the benefits in terms of computational cost greatly favors our approach.

We finally conclude by presenting the limitations of our work, the improvements that could be made and proposing some interesting perspectives.

Publications

[ACS+11] R. Abdullah, M. Christie, G. Schofield, C. Lino, and P. Olivier. Advanced Composition in Virtual Camera Control. In *Smart graphics*, volume 6815 of *Lecture Notes in Computer Science*, pages 13–24. Springer Berlin Heidelberg, 2011. 21

[BLCR13] W. Bares, C. Lino, M. Christie, and R. Ranon. Methods, System and Software Program for Shooting and Editing a Film Comprising at least One Image of a 3D Computer-generated Animation. US Patent n°20130135315, May 2013.

[CLR12] M. Christie, C. Lino, and R. Ronfard. Film Editing for Third Person Games and Machinima. In *Workshop on Intelligent Cinematography and Editing*, Raleigh NC, USA, 2012.

[HLCO10] H. N. Ha, C. Lino, M. Christie, and P. Olivier. An Interactive Interface for Lighting-by-example. In *Smart graphics*, volume 6133 of *Lecture Notes in Computer Science*, pages 244–252. Springer-Verlag Heidelberg, 2010.

[LC12] C. Lino and M. Christie. Efficient Composition for Virtual Camera Control. In *2012 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 65–70, Lausanne, Switzerland, 2012.

[LCCR11a] C. Lino, M. Chollet, M. Christie, and R. Ronfard. Automated Camera Planner for Film Editing Using Key Shots. In *2011 ACM SIGGRAPH / Eurographics Symposium on Computer Animation (Poster)*, Vancouver, Canada, 2011.

[LCCR11b] C. Lino, M. Chollet, M. Christie, and R. Ronfard. Computational Model of Film Editing for Interactive Storytelling. In *Interactive Storytelling (Poster)*, volume 7069 of *Lecture Notes in Computer Science*, pages 305–308. Springer Berlin Heidelberg, 2011.

[LCL+10] C. Lino, M. Christie, F. Lamarche, G. Schofield, and P. Olivier. A Real-time Cinematography System for Interactive 3D Environments. In *2010 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 139–148, Madrid, Spain, 2010.

[LCLO09] C. Lino, M. Christie, F. Lamarche, and P. Olivier. A Real-time Cinematography System for 3D Environments. In *22es Journées de l'Association Francophone d'Informatique Graphique (AFIG)*, pages 213–220, Arles, France, 2009.

[LCRB11a] C. Lino, M. Christie, R. Ranon, and W. Bares. A Smart Assistant for Shooting Virtual Cinematography with Motion-tracked Cameras. In *19th*

ACM International Conference on Multimedia (Demo), pages 831–832, Scottsdale Arizona, USA, 2011.

[LCRB11b] C. Lino, Marc Christie, Roberto Ranon, and William Bares. The Director's Lens: An Intelligent Assistant for Virtual Cinematography. In *19th ACM International Conference on Multimedia*, pages 323–332, Scottsdale Arizona, USA, 2011.

State of the Art 2

Virtual camera control encompasses viewpoint computation, path planning and editing. Virtual camera control is a key element in many computer graphics applications such as scientific visualization, virtual exploration of 3D worlds, proximal inspection of objects, video games or virtual storytelling. A wide range of techniques have been proposed, from direct and assisted interactive camera control to fully automated camera control techniques.

This state of the art is divided as follows. Firstly, we describe the cinematographic background about the use of real-world cameras in Section 1. Secondly, we describe interactive camera control techniques in Section 2. Thirdly, we describe automated camera control techniques (viewpoint computation, path planning and editing) in Sections 3 to 5. Fourthly, we discuss about a transverse issue in virtual camera control, the occlusion avoidance, in Section 6. Finally, we conclude on lacks of existing camera control techniques, before detailing the contributions of the thesis in Section 7.

1 Cinematographic Background

An overview of the use of real-world cameras can be found in some reference books on photography and cinematography practice [Ari76, Mas65, Kat91]. In addition to camera placement, cinematography involves a number of issues such as shot composition, lighting and staging (how actors and scene elements are arranged), as well as the implementation of filmmaker's communicative goals. In movies, camera placement, lighting and staging are highly interdependent. However, in other situations such as in documentaries or news reports, the operator generally has no or a limited control over staging and lighting. In this state of this art, we will review cinematic camera placement with this in mind. Indeed, in computer graphics applications (*e.g.* computer games) real-time camera control is very similar to documentaries; one must generally present scene elements to a viewer without direct modification of the elements themselves [CON08a].

When taking a close look at the structure of a *movie*, one can view it as a sequence of *scenes*. A scene can, in turn, be decomposed as a sequence of *shots*, separated with *cuts*. Finally, a *shot* can be viewed as a continuous series of *frames*, and may contain camera motions. In the following, we will focus on the cinematic knowledge at the scene level. This knowledge can be classified into three categories: the shots, the camera motions, and the editing. The construction of a movie is both a creative and technical

craft, through which a director communicates his vision to the audience; shots, camera motions, and editing play a key role in the communication of the director's message. Real cinematographers have thus been elaborating a "visual" grammar of movie making for more than a century. How a director uses this grammar then determines his capacity to properly convey his vision. Indeed, two different directors, through the use of their own style, would convey two different visions of the same narrative elements; which would potentially result in the construction of two different stories.

1.1 Shots

A shot is a key unit which provides information to the viewer. Each shot represents a unique way to frame one or more narrative element(s). The composition of these elements (*i.e.* their spatial arrangement within the frame) is very important. Common rules of composition [Tho98] encompass the content of the frame (what the key elements are and what their degree of visibility is), the visual aspect of key elements, as well as the arrangement and visual balance of elements. We here review a range of significant composition rules:

Head-room Head-room specifically refers to how much or how little space exists between the top of an actor's head and the top edge of the recorded frame.

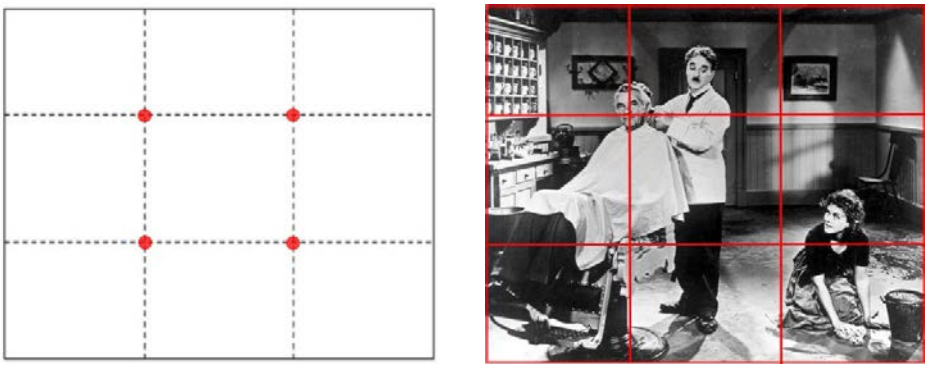
Look-room The look-room is the empty area that helps balance out a frame where, for instance, the weight of an element occupies the left of the frame and the weight of the empty space occupies the right of the frame. Here, the word "weight" really implies a visual mass whether it is an actual object, such as a head, or an empty space, such as the void filling the right of the frame.

Rule of thirds The rule of thirds is a composition rule that helps in building a well-balanced framing. The frame is equally divided into thirds, both vertically and horizontally. Key elements are then to be placed along a line (called power line) or at the intersection of two lines (called power point). See Figure 2.1 for illustration.

Shot size There is a variety of common shot sizes classified according to the proportion of a subject that is included in the frame. This rule is commonly associated with the framing of a character (see Figure 2.2). For instance, a close up shot would frame a character from the shoulders, a medium shot from the hips, a full shot would include his whole body, whereas a long shot would be filmed from a distance.

Camera angles Horizontal camera angles are stereotypical positions around one (or more) subject(s). For a one-character shot, horizontal angles can be represented as degrees of a circle (see Figure 2.3a). The 3/4 front, for instance, is the most common view angle in fictional movies; it provides a clear view of facial expressions, or hand gestures. In a two-character shot, the over-the-shoulder (or OTS, shot above and behind the shoulder of one character shooting the other character) is very often used in dialogue scenes. The sense of proximity conveyed by such a shot is used, for example, to establish affinity between the viewer and the character.

The vertical angles (see Figure 2.3b) have strong meanings. Literature distinguishes three main vertical angles: high angle (taken from above), low angle (taken from below), or neutral angle (taken from the subject's height). A high angle often conveys the feeling that what is on screen is smaller, weaker, or currently in a less powerful or compromised position. A low angle usually generates the reverse feeling. The subject seen from below appears larger, more significant, more powerful and physically higher in the film space.



(a) **(b)**

Figure 2.1 – The rule of thirds. (a) The frame is equally divided into thirds, both vertically and horizontally. Key elements are placed along a line (power line) or at the intersection of two lines (power point). (b) Example of framing, extracted from *The Great Dictator*.

1.2 Camera motions

Shots can contain be created using camera motions. Camera motions are used to change the viewer's perspective. Classical camera motions can be divided into 3 categories: (1) the mounted camera creates the motion, (2) the camera and operator move together and (3) only the camera lens moves. We hereafter review some well established camera motions.

1.2.1 Mounted camera motions

This category is characterized by a camera motion with no physical motion of the operator. This is commonly performed by mounting the camera on a tripod, for a smooth effect.

Pan (or panoramic) The camera is turned horizontally left or right. Pan is used to follow a subject or to show the distance between two subjects. Pan shots are also used for descriptive purposes.

Tilt The camera is turned up or down without raising its position. Like panning, it is used to follow a subject or to show the top and bottom of a stationary object. With a tilt, one can also reveal size of a subject.

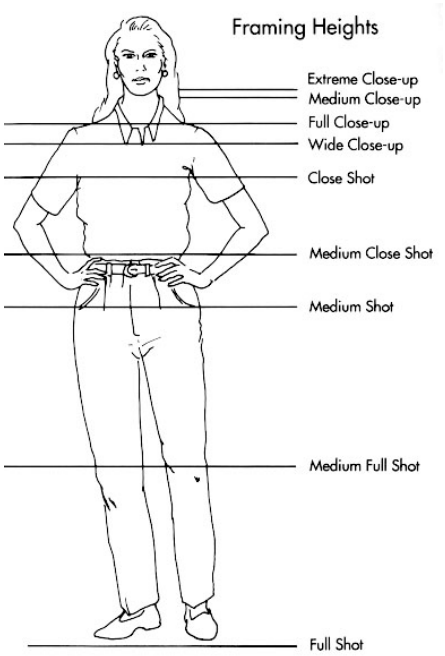
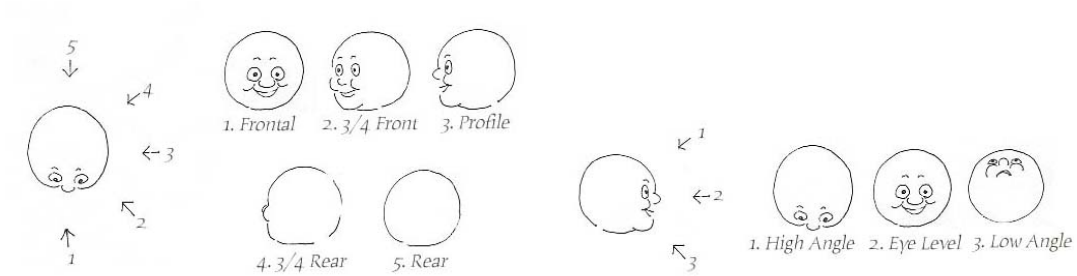


Figure 2.2 – Framing heights describe the proportion of a subject that is included in the frame.



(a) **(b)**

Figure 2.3 – Shot angles: (a) the different horizontal camera angles, (b) the different vertical camera angles.

Pedestal The camera is physically moved up or down (in terms of position), without changing its orientation.

1.2.2 Camera and operator motions

This category is characterized by a physical motion of both the operator and the camera. Some of them may be performed by mounting the camera on a tripod.

Dolly The camera is set on tracks or wheels and moved toward or back from a subject. It can be used to follow a subject smoothly.

Crane or Boom This works and looks similar to a construction crane. It is used for high sweeping shots or to follow the action of a subject. It provides a bird's eye view, that may be used for example in street scenes so one can shoot from above the crowd and the traffic, then move down to the level of individual subjects.

Hand held The camera is held without a tripod, which leads to a shaking motion to the camera. It may introduce more spontaneity in the unfolding action (*e.g.* in documentaries), or introduce a feeling of discomfort.

1.2.3 Camera lens motions

This category is characterized by a change in intrinsic parameters of the camera only (*i.e.* the field of view angle). There is no physical motion of the operator, and no change of the extrinsic camera parameters (*i.e.* position and orientation).

Zoom The field of view is narrowed or widened, in order to zoom in or out. It is used to bring subjects at a distance closer to the lens, or to show size and perspective. It also changes the perception of depth. Zooming in brings the subject closer, with less visible area around the subject and distant objects are compressed. Zooming out shows more of the subject and make surrounding areas visible.

Rack Focus Depth of field determines the range of depth at which elements are in focus (*i.e.* appear sharp). Elements before and behind are out of focus (*i.e.* appear blurred). In a rack focus, the focal length is changed so that the depth of field changes. Elements at a the initial depth of field become more and more blurred, while elements at the final depth of field appear more and more clearly. Rack focus makes a transition similar to an edit by constructing two distinct shots. It is often used in dramas, changing focus from one character's face to another during a conversation or a tensed moment.

Simple camera motions can be combined to construct more complex camera motions.

1.3 Editing

The process of editing – the timing and assembly of shots into a continuous flow of images – is a crucial step in constructing a coherent cinematographic sequence. Some general rules can be extracted from the literature. For instance, a new shot should provide new information which progresses the story, and each cut should be motivated.

1.3.1 Continuity Editing

In order not to get the audience disoriented, editing should not break continuity. Here are some of the basic continuity rules listed in the literature.

Continuity of screen direction To help maintain lines of attention from shot to shot, the concept of *180 degree line* or *line of interest* (LOI) was introduced. The LOI is an imaginary line, commonly established as the sight line of the subject in a one-character shot, or the line passing through both subjects for a two-character

shot. When editing two shots, one must ensure that the camera does not "cross the line" (*i.e.* ensure that the two shots are taken from the same side of this line). Indeed, crossing the *LOI* would reverse the established directions of left and right and lead to disorientation in the audience. Note that, when it is the intended effect, cinematographers can voluntarily introduce such a cut.

Continuity of motion If during a shot a subject walks out left of the frame, the subject's leftward motion dictates that in the new shot it should enter from the right of the frame. This enforces the direction of leftward motion within the film's space.

Jump Cut Because each shot or view of the action is supposed to show new information to the audience, it also makes sense that one should edit shots with a significant difference in their visual composition. This will avoid what is known as a *jump cut* – two shots with similar framing of the same subject, causing a visual jump in either space or time. This rule is commonly implemented by providing a minimum change in the view angle (usually 30 degrees) and/or a minimum change in the size of a subject.

1.3.2 Idioms

Given prototypical situations (*i.e.* simple actions performed by the actors, such as dialogue scenes, walking scenes or door passing), there are typical camera placements (called idioms) to convey them (see [Ari76] for a cookbook of idioms). For instance, in dialogue scenes, a classical idiom is to alternate between an OTS shot of one character and an OTS shot of the other character (also known as *shot / reverse shot*). Usually, each shot shows the speaker from front, and a cut is made when a change of speaker occurs. One can also introduce reaction shots (*i.e.* shots on the listener, to see his reaction to the speaker's words). For a given situation, however, the number of possible idioms and how they are constructed are only limited by the creativity of cinematographers.

1.3.3 Pacing

Pacing is the rhythm at which transitions (cuts or camera motions) occur. Pacing has a strong impact on the emotional response from the audience. For instance, in highly dramatic moments, the story's tension may be conveyed through fast-paced cuts; while a romantic scene would rely on slow-paced cuts.

1.4 From Real Cinematography to Virtual Cinematography

Through this section, we have shown that real cinematography supplies filmmakers with a range of practical tools and rules (on camera placement, camera motions and editing) which both furnish a framework for creating consistent movies and means to express their creativity. Virtual camera control can also benefit from such cinematic elements. This however requires to be capable of (i) modeling these elements and (ii) providing the user with means to re-use them to fully express his creativity in 3D environments. In the following sections, we review techniques that have been proposed

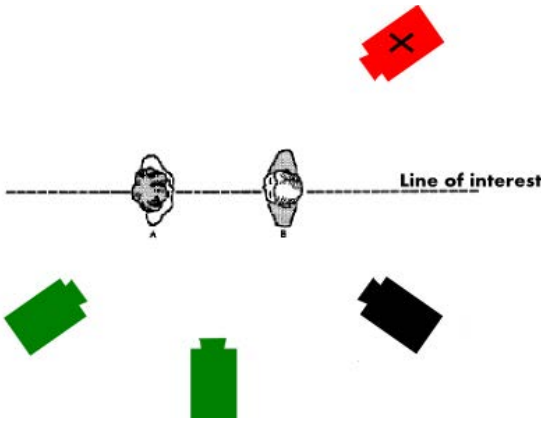


Figure 2.4 – 180° rule. To enforce space continuity, a cut between two camera viewpoints must never cross the line of interest, drawn between the two subjects. The current viewpoint is drawn in black. A cut to a green camera would enforce space continuity, while a cut to the red camera would break space continuity.

to control a virtual camera, and pay particular attention to how these elements have been accounted for.

2 Interactive Control

The interactive control of a camera is the process through which a user interactively modifies (directly or indirectly) camera parameters. Explored techniques can be categorized into three classes: direct control techniques, assisted control techniques and "through-the-lens" control techniques.

2.1 Direct and Assisted Control

Direct control of a virtual camera requires dexterity from the user. It is however problematic to deal simultaneously with all seven degrees of freedom (*dof*). Researchers investigated means to facilitate this control; they provided mappings of the *dof*s of an input device (commonly a mouse) into camera parameters. Such mappings are referred to as camera control metaphors, and have mainly been designed for applications such as proximal inspection of objects or exploration of 3D worlds.

As stated in [CON08a], Ware and Osborne [WO90] reviewed the possible mappings, and categorized a broad range of approaches:

- *eyeball in hand*: the camera is directly manipulated as if it were in the user's hand, encompassing rotational and translational movement. User input modifies the values of the position and orientation of the camera directly.
- *world in hand*: the camera is fixed and constrained to point at a fixed location in the world, while the world is rotated and translated by the user's input. User

input modifies the values of the position and orientation of the world (relative to the camera) directly.
- *flying vehicle*: the camera can be treated as an airplane. User input modifies the rotational and translational velocities directly.
- *walking metaphor*: the camera moves in the environment while maintaining a constant distance (height) from a ground plane [HW97][Bro86].

Drucker *et al.* [DGZ92] proposed CINEMA, a general system for camera movement. CINEMA was designed to address the problem of combining metaphors (*eyeball in hand*, *scene in hand* and *flying vehicle*) for direct interactive viewpoint control. CINEMA also provides a framework in which the user can develop new control metaphors through a procedural interface. Zeleznik in [ZF99] demonstrated the utility of this approach by proposing smooth transitions between multiple interaction modes with simple gestures on a single 2-dimensional input device. Chen *et al.* [CMS88] reviewed the possible mappings of 3D rotations using 2D input devices. Notably, Shoemake [Sho92] introduced the concept of the *arcball*, a virtual ball that contains the object to be manipulated. His solution relies on quaternions to stabilize the computation and avoid Euler singularities (*i.e.* gimbal lock) while rotating around an object. Shoemake's techniques only considers the position parameters. The camera is oriented toward the central key subject. Jung *et al.* [JPK98] suggested a *helicopter metaphor*, as a mapping function in which transitions are eased by using a set of planes to represent the camera's 6 *dof*s. Transitions between planes can then be specified with a 2-dimensional input device.

Presented metaphors does not account for physical properties of cameras. To get closer to a real camera's behavior, Turner *et al.* [TBG91] explored the application of a physical model to camera motion control. User inputs are treated as forces acting on a weighted mass (the camera). Friction and inertia are then incorporated to damp degrees of freedom. Turner *et al.* 's approach easily extend to manage any new set of forces and has inspired approaches that rely on vector fields to guide the camera parameters.

Direct camera control metaphors lack avoidance of occlusions of key elements and collisions with the scene elements. To overcome these issues, researchers proposed techniques to assist the user in interacting with the camera parameters. Khan *et al.* [KKS+05] proposed an assisted interaction technique for proximal object inspection that automatically avoids collisions with scene objects and local environments. The *hovercam* tries to maintain the camera at both a fixed distance around the object and (relatively) normal to the surface, following a hovercraft metaphor. Thus the camera easily turns around corners and pans along flat surfaces, while avoiding both collisions and occlusions. 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 designed to constrain the camera to yield interesting viewpoints of the object that will guarantee a certain level of quality in the user's exploration, and automated transitions are constructed between the edges of different surfaces in the scene.

Such camera control metaphors enable full control over the camera placement and motions. These tasks however remain tedious and are not intuitive *w.r.t.* cinematographic practice.

2.2 Physical Controllers

In the purpose of getting closer to the look, feel, and way of working of real movie cameras, physical or tangible human input controllers have been proposed. These physical controllers typically include a combination of a portable display, hand-held motion sensors, buttons, and joysticks. Physical input controllers enable filmmakers to intuitively operate virtual cameras in computer graphics applications. Filmmakers can then rapidly preview and record complex camera motions by simply moving, turning, and aiming the physical controller as if it were a real camera.

Though physical controllers enable fine and intuitive control of the camera placement and of camera motions *w.r.t.* cinematic tasks, the operator needs to manually set a scale factor between the physical controller motion and the motion mapped to the virtual camera; this enables handling motions at different scales, from fine camera motions to large fly-by movements. Setting the camera so as to match a given composition of scene elements moreover remains tedious, and involves the operator to physically explore the space of camera placements.

Figure 2.5 – Physical controller for virtual cameras (here the NaturalPoint Insight VCS). Real-time virtual camera tracking replicates the organic cinematographer's experience with fidelity.

2.3 Through-The-Lens Control

Gleicher and Witkin [GW92] introduced the paradigm of *"Through The Lens"* Camera Control. This paradigm relies on the handling of the camera parameters through direct manipulations on the visual composition of a shot. The mathematical relation between an object in the 3D scene and its projection on the 2D screen is however strongly non-linear. If we consider a Euler-based camera model (see Figure 2.6) for which the parameters are $q = [x_c, y_c, z_c, \varphi_c, \theta_c, \psi_c, \phi_c, a_c]^T$, then the projection is given by Equation 2.1. This relation is expressed as a 4×4 uniform matrix $M(q)$ representing the transformation from the world coordinates (x, y, z) of a 3D point to its on-screen coordinates (x', y'). This transformation is the combination of three consecutive operations: a translation $T(x_c, y_c, z_c)$, a rotation $R(\varphi_c, \theta_c, \psi_c)$ and a perspective projection

$P(\phi_c, a_c)$. This transformation can be put into equations as follows:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = P(\phi_c, a_c) \cdot R(\varphi_c, \theta_c, \psi_c) \cdot T(x_c, y_c, z_c) \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = M(q) \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2.1}$$

with $z' = -1$ (depth of the camera plane). The strong non-linearity of this relation makes it difficult to invert (*i.e.* to decide where to position the camera and how to orient it given both the world and screen positions of the object).

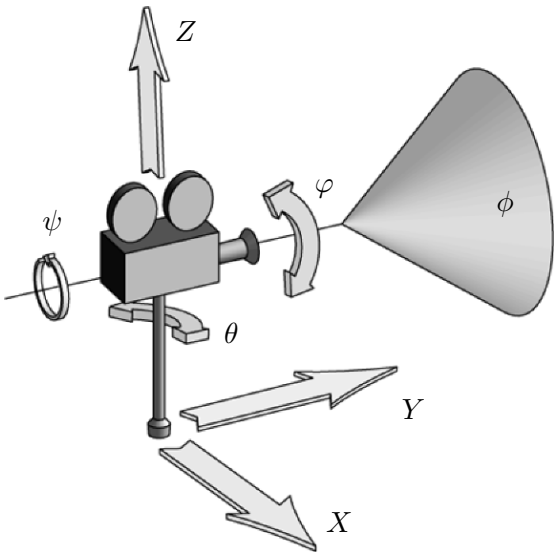


Figure 2.6 – A simple camera model based on a position (x, y, z), set in a global Cartesian coordinate system $(\overrightarrow{X}, \overrightarrow{Y}, \overrightarrow{Z})$, and three Euler angles: pan (θ), tilt (φ) and roll (ψ). The camera also has two intrinsic parameters: a field of view ϕ – which defines the zoom factor –, and an aspect ratio a (generally fixed) – which defines the ratio width/height of the screen.

Gleicher and Witkin proposed a technique that allows a user to control a camera through a direct manipulation of on-screen positions of elements (m points). New camera parameters are recomputed to match the user's desired positions. The difference between the actual and desired on-screen positions is then treated as a velocity. They expressed the relationship between the velocity $(\dot{h})$ of the m displaced points on the screen and the velocity $(\dot{q})$ of the camera parameters through a Jacobian matrix J representing the derived perspective transformation. As detailed in [CON08a], Gleicher and Witkin presented this as a non-linear optimization problem in which a quadratic energy function is minimized. This problem is then converted into a Lagrange equation, and solved. When the problem is over-constrained (*i.e.* the number of control points, m, is higher than the number of degrees of freedom), the complexity of the Lagrange process is $O(m^3)$. The formulation of this problem has then been improved and extended by

Kung *et al.* [KKH95], reducing the complexity to $O(m)$.

This technique enables an easy control of shot composition rules such as the rule of thirds, the head-room and the look-room. The handling of shot size and view angles is also possible, with some restrictions. For instance, the proposed method hardly handles long shots – because points are mixed up from a certain distance –, or back views – where one or more points are occluded. Controlling the camera motions through the on-screen motion of projection points is moreover a tedious task.

2.4 From Interaction to Automation

On one hand, the interactive control of cinematic aspects is a tedious, and time consuming task; especially as we think of dynamic scenes. Indeed, it is extremely difficult to manually handle shot composition, camera motions and editing at the same time, while minimizing occlusion of key subjects. On the other hand, cinematic knowledge is made of well established rules; these rules can be formalized. The research community thus focused on such formalizations and on the automation of camera positioning tasks, to tackle the automated composition of shots, planning of camera motions and editing. In the following sections we review the automated camera control techniques that have been proposed to tackle each of these three tasks.

3 Automated Camera Composition

Given the difficulty of manually arranging the composition of shots, researchers have proposed automated camera composition techniques. Given a number of constraints on the composition (*i.e.* visual properties to be satisfied), camera *dof*s are computed automatically. The main problem here is to find where to position and how to orient a camera so that such visual properties are satisfied. Camera composition techniques can be categorized into three classes of approach: direct algebra-based approaches, constraint-based approaches and optimization-based approaches.

3.1 Direct Algebra-based Approaches

The very first automated camera composition system has been proposed by Blinn [Bli88], who developed a framework for configuring a camera so that a probe and a planet appear on screen at given exact coordinates.

Blinn's expressed his spacecraft problem in terms of vector algebra for which both an iterative approximation and a closed form solution can be found. Blinn's approach requires the world coordinates of the objects f and a (see Figure 2.7), their desired position on the screen (x'_f, y'_f) and (x'_a, y'_a), the up-vector $\overrightarrow{U}$, as well as the camera's field of view. The closed form solution computes the parameters of the translation matrix T and rotation matrix R of the transformation from world space to view space (Equation 2.1). Blinn's numerical solution has the advantage of producing an approximate result even where the specified properties have no algebraic solution. Two main

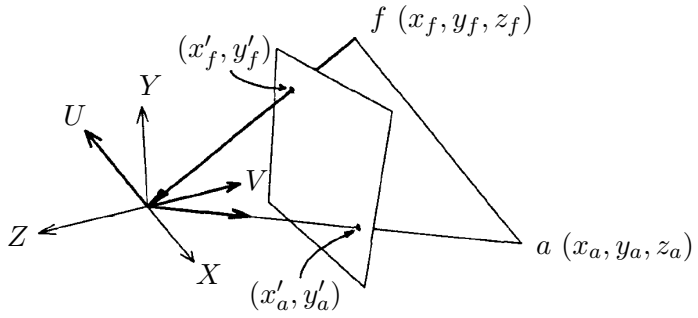


Figure 2.7 – Blinn's algebraic approach to camera control: two points on the screen, the field of view and an up-vector allow direct computation of the camera position [Bli88].

drawbacks can however be highlighted. As for most vector algebra approaches, the solution is prone to singularities. Blinn's technique is moreover restricted to this simple composition problem; it cannot extend to more evolved composition problems.

To implement cinematic knowledge related to visual composition, there is a need to resolve more complex problems involving a combination of both visual properties (*e.g.* on-screen position or projected size of subjects) and properties on the camera (*e.g.* distance or view angle *w.r.t.* subjects). With this in mind, researchers then proposed a range of declarative approaches. They take a set of properties as input, then rely on constraint-based and/or optimization-based techniques to compute the camera parameters that best satisfy those properties.

3.2 Constraint-based Approaches

Constraint satisfaction problem (CSP) frameworks enable modeling a large range of constraints in a declarative manner, and propose reliable techniques to solve them.

For instance, interval arithmetic-based solvers compute the whole set of solutions as interval boxes through an *evaluate-and-split* dynamic programming process, as presented in Figure 2.8. Each unknown in the problem is considered as an bounded interval representing the domain within which to conduct the search. All the computations are therefore casted into operations on intervals. For a better insight of the use of interval arithmetic in computer graphics, see [Sny92].

Bares *et al.* [BMBT00] designed a graphical interface to efficiently define sophisticated camera compositions by creating *storyboard frames*. The storyboard frames are then automatically encoded into an extensive set of camera constraints capturing their key visual composition properties. They account for visual composition properties such as the size and position of a subject appearing in a camera shot. A recursive heuristic constraint solver then analyzes the camera space to determine camera parameters which produce a shot close to the given storyboard frame.

However, applicable constraints are numerous, and imposing conflicting constraints prevents solving the entire problem. Such conflicts can appear between two constraints, or in the combination of three or more constraints for example. Consequently, there is a necessity of a trade-off in the resolution of constraints and/or of a relaxation value

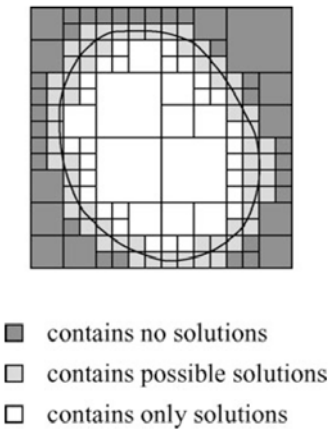


Figure 2.8 – Evaluate-and-split approach to solve $f(x) \leq 0$ with interval arithmetic. White boxes contain solutions configurations only, dark boxes contain configurations that do not satisfy the relation, and gray boxes contain both solution configurations and configurations that do not satisfy the relation.

on each constraint (*e.g.* an acceptable interval of values).

Hierarchical constraint approaches are able to relax some of the constraints to provide an approximate solution to the problem. Bares *et al.* [BZRL98] proposed ConstraintCam, a partial constraint satisfaction system for camera control. Inconsistencies are identified by manually constructing a pair graph of incompatible constraints. If the system fails to satisfy all the constraints of the original problem, it 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]. ConstraintCam uses a restricted set of cinematographic properties (viewing angle, viewing distance and occlusion avoidance) and the constraint satisfaction procedure is applied to relatively small problems (*i.e.* involving two objects). Halper *et al.* [HHS01] proposed an incremental solving process based on successive improvements of a current configuration (see Figure 2.9). Their system accounts for various properties, including relative elevation, size, visibility and screen position. The solving process then incrementally satisfies the set of screen constraints at each frame, relaxing subsets as required.

One of the main drawbacks of constraint-based approaches is their lack in detecting and effectively resolving inconsistencies in over-constrained problems. Such approaches are thus hardly usable when no optimal solution exist. This has lead researchers to take an active interest in optimization-based techniques.

3.3 Optimization-based Approaches

Pure optimization techniques express shot properties to verify as a set of objective functions $f_i(q)$ (one for each property) to maximize. Each objective function is expressed from both a description of the problem and one or more metrics evaluating the satisfaction of the property. A global fitness function F is then defined as the aggregation of objective functions $f_i(q)$. Finally, an optimization solver performs a search

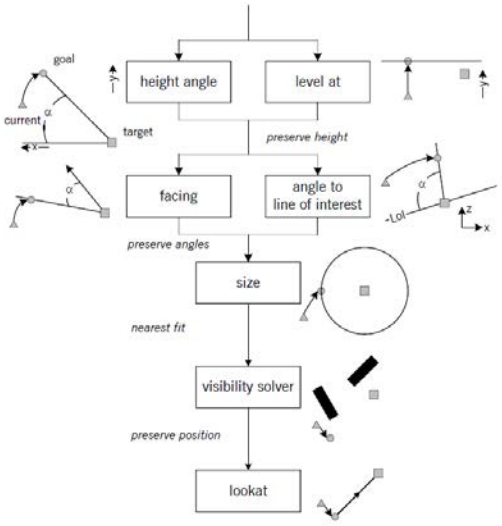


Figure 2.9 – Hierarchical application of constraints [HHS01].

in the $7D$ space of possible camera configurations Q. This optimization process can formulated as

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

where $f_i : Q \to \mathbb{R}$ is the objective function of property i to maximize, and F is generally a linear combination of scalar weighted functions:

$$F(f_1(q), f_2(q), \ldots, f_n(q)) = \sum_{i=1}^{n} w_i \cdot f_i(q)$$

where w_i is the weight of property i. Note that the metric provided for a given visual property is not necessarily accurate. Ranon *et al.* [RCU10] proposed a rendering-based technique that provides, through pixel counts, a rather accurate approximation of the actual satisfaction of commonly used visual properties or a combination of them.

Classical optimization techniques range from deterministic approaches (such as gradient-based) to non-deterministic approaches such as population-based algorithms (*e.g.* genetic algorithms), probabilistic methods (Monte Carlo) or stochastic local search methods (Artificial Potential Fields).

Discrete approaches tackle the exploration of this 7-*dof* continuous search space by considering a regular discretization on each *dof*. In [BTM00], Bares *et al.* proposed to use an exhaustive search by covering the search space by a $50 \times 50 \times 50$ grid of camera positions, every $15°$ angle for orientation and 10 possible values for the field of view. They further reduced the computational cost of their technique by (i) narrowing the discretization to feasible regions of the search space (built from the intersection of individual regions related to each composition property) and (ii) reducing the grid resolution at each step. The time complexity of such an exhaustive search is however

$O(n^7)$, which is the worst case possible. The grid structure will moreover never find the optimal solution if it is not located exactly on the grid.

To scan the configuration space in a more continuous way, researchers explored the use of population-based algorithms. Olivier *et al.* [OHPL99] proposed CamPlan, a camera planning subsystem for polygonal graphics. They used a large set of properties including explicit spatial relationships between objects, partial/total occlusion and size of objects. CamPlan uses a polygonal representation of the scene elements. The fitness function is a linear-weighted combination of the fulfillment of the shot properties. The seven parameters of the camera are encoded in the allele. A population of cameras is then randomly distributed in the search space. Each individual of this population is evaluated *w.r.t.* the set of objective functions. The top 90% of the population survives to the next generation, and the selection is made by binary tournament. The remaining 10% is re-generated by random crossover and/or mutation. This was extended for a dynamic camera by optimizing the control points of a quadratic spline (with a fixed look-at point and known start and end positions) [HO00]. The computation time CamPlan requires is however uneven and strongly dependends on the inital population of cameras.

More recently, Ranon *et al.* [DER08] proposed to use a Particle Swarm Optimization (PSO) technique, which exhibits better performances. PSO is a population-based method for global optimization, which is inspired by the social behavior that underlies the motions of insects or flocks of birds. They approximated subjects by a bounded sphere and accounted for constraints such as subject view angle, subject inclusion/exclusion, subject distance, subject projection size, subject occlusion, and subject on-screen position (as frames), as well as constraints on the camera position. Each particle is represented by a position, and a velocity. At the beginning of the search, the particles' position and orientation are chosen randomly. The search is then performed as an iterative process, which at step n modifies the velocity and position of each particle on the basis of the values at step $n-1$. During the search, the particle best visited positions are memorized. Abdullah [ACS+11] used a similar technique, but used low-resolution renderings for the evaluation of properties' satisfaction. This has the advantage of providing more precise results, at the cost of a higher computation time.

In some cases, multiple solutions with approximately the same value of fitness may exist. Preuss and Burelli [PBY12] proposed the application of niching and restart evolutionary algorithms to the problem of diversity of shot generation in virtual camera composition. Their technique simultaneously identifies multiple valid camera configurations. Though effective, their method does not evaluate accurately how different are the shots generated by two solutions. For such purposes, Vázquez *et al.* [VFSH03] explored the use of viewpoint entropy as a metric to maximize the quantity of information in a minimal set of views.

The main drawbacks of population-based approaches are their computational cost, and their non-deterministic behavior.

More generally, optimization-based approaches suffer from the difficulty of modeling objective functions for each property, and aggregating multiple properties into a single

fitness function. These two difficulties lead to a tedious tuning process.

3.4 Constrained-optimization Approaches

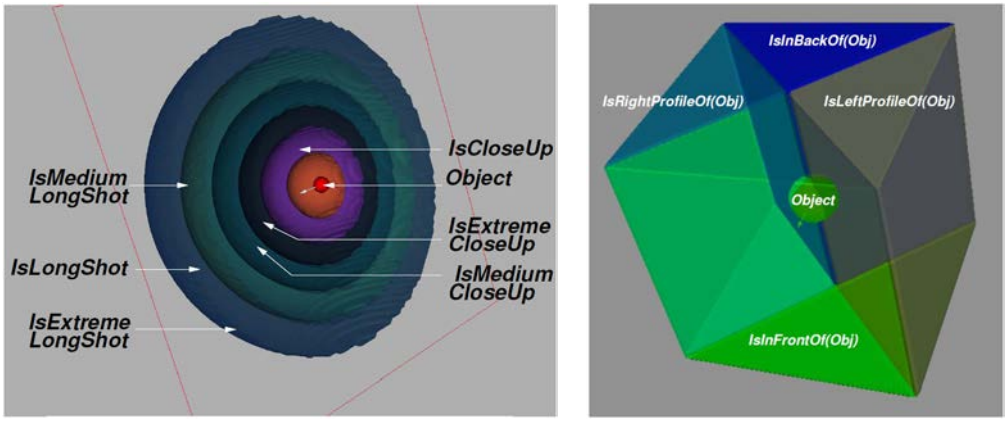
On one hand, pure optimization techniques enable computing of a possibly good solution, *i.e.* in which each property is satisfied to some degree. In the worst case, this partial satisfaction can however lead to unnatural solutions in which either one function dominates, or both are poorly satisfied. On the other hand, pure complete constraint-based techniques compute the whole set of solutions, but are often computationally expensive. Furthermore, pure constraint-based systems are not suitable in case of an over-constrained problems, *i.e.* they cannot provide an approximate solution for such problems. A compromise can be found in the use of constrained-optimization approaches. In such approaches, the camera control problem is expressed as a set of properties to enforce (constraints) and a set of properties to optimize (through a fitness function to maximize).

Drucker and Zeltzer [DZ95] proposed the specification of behaviors for virtual cameras as task-level goals (objective functions) and constraints on the camera parameters. The main drawback of their technique is that constraint functions and fitness functions must be continuously differentiable, thus limiting their approach to smooth functions subject to smooth constraints; this conditions are often difficult to guarantee.

3.5 Hybrid Approaches

Constraint-based and optimization-based approaches directly tackle the whole 7-*dof* problem, and consider the whole scene as search space. To find a good representative, the complexity of their solving process is then $O(n^7)$ with n large. A possible way to reduce this complexity is to consider the whole problem as made of sub-problems of lower complexity. Sub-problems can then be solved incrementally by using (for each) either a constraint-based or an optimization-based approach.

Researchers proposed to firstly reduce the search space, by creating geometric volumes (only based on camera positions), to build models of the feasible space [BMBT00, PO03]. Classical stochastic search (as in [PO03]), or heuristic search (as in [BMBT00]) can then be applied within the resulting volumes. Christie and Normand [CN05] proposed the notion of *Semantic Volumes*, which can be considered as an extension of *visual aspects* [KV79] and is closely related to *viewpoint space partitioning* [PD90]. Semantic Volumes are 3D volumes that categorize possible solutions *w.r.t.* their cinematic properties (see Figure 2.10). Their approach relies on (i) a space partitioning process derived from a study of possible camera locations *w.r.t.* the subjects in the scene, and (ii) on local search numerical techniques inside volumes to compute good representatives of each volume. [BDER08] used a similar process to compute feasible camera volumes from some of the properties. The search inside volumes is then carried out with PSO. The visual properties they adopted, as well as the solving process, are however not meant for dynamic scenes with temporary occlusions, which requires properties expressed over more than just the current point in time.



(a) **(b)**

Figure 2.10 – Semantic partitioning [CN05]: (a) encoding of camera distances, (b) encoding of relative camera angles.

In this section we have focused on the problem of automatically setting the camera so as to compose static shots (*i.e.* a given frame of a shot). One may now want to create dynamic shots, *i.e.* containing camera motions. In the next section we will review techniques proposed to automate the planning of camera motions.

4 Automated Camera Planning

A dynamic shot can be viewed as a discrete series of images (*e.g.* 25, 30, or 60 images per second). In terms of camera placement, it then corresponds to a discrete sequence of camera configurations; one configuration for each frame. These sequences have to fulfill a set of requirements, such as enforce visual composition, avoid collision with scene elements or mimic the behavior of a real camera.

4.1 Procedural Camera Movements

Early attempts to build cinematic-like camera motions focused on a direct derivation of cinematic conventions. Palamidese [Pal96] proposed a camera motion metaphor based on film grammar. She used a camera operator, that incorporates camera motion rules derived from accepted conventions in filming practice. Her camera operator can use a set of basic techniques and concepts such as framing, tracking, panning, zooming and shooting to build complex camera actions. Bares *et al.* [BZRL98] developed a *cinematic task modeling* framework for automated real-time task-sensitive camera control. Cinematic task models dynamically map the intentional structure of users' activities to visual structures that continuously portrays the most relevant actions and subjects in the environment. By exploiting cinematic task models, a cinematography interface

to 3D learning environments can dynamically plan camera positions, view directions, and camera movements that help users perform their tasks.

Though procedural techniques allow to mimic cinematic camera movements, they generally account for single subject situations. They moreover poorly account for issues related to complex scenes, such as occlusions or collisions. Indeed, they consider either a fully static environment or a single dynamic subject traveling in a static environment. Camera movements are modeled as splines; they does not account for collisions and occlusions along the camera path, or use a cut to overcome this problem. As a consequence, there is a need for techniques capable of automatically planning camera paths in complex dynamic environments, while avoiding collisions and occlusions.

Camera planning is the process of planning a path between a source and a target camera configuration, while enforcing properties along the path (*e.g.* maintaining composition properties, avoiding collisions or occlusions, avoiding jerkiness of the camera). This problem is a sub-class of motion planning problems. Motion planning techniques have mainly been inspired from robotics [Lat91] and have been re-explored in computer graphics [LaV06]. Approaches can be categorized into three main classes: purely reactive approaches, constraint-based approaches, and optimization-based approaches.

4.2 Reactive Approaches

Reactive approaches consists in re-computing a camera configuration to respond to changes in the image properties without resorting to any search process. Such approaches are generally based on techniques from the robotics community. For instance, visual servoing approaches [ECR92] involve the specification of a task (usually positioning or subject tracking) as the regulation of a set of visual features in an image. In [CM01], Courty and Marchand proposed a visual servoing approach integrating constraints on a camera trajectory to address a range of non-trivial computer animation problems. In a way similar to [GW92], a Jacobian matrix expresses the interaction between the motion of the points on the screen and the motion of the camera.

Visual servoing approaches are computationally efficient. Consequently, they are suitable for highly dynamic environments such as in computer games. It is however difficult to determine beforehand which camera *dof* will be instantiated by the main task. Furthermore, such approaches often lead to sudden modifications of the camera speed and direction; they thus require to use an additional process to smooth the camera path.

4.3 Constraint-based Approaches

Similarly to Section 3.2, constraint-based planning techniques are based on the declaration of constraints on the visual features and on the camera path, then rely on a numerical resolution process. Jardillier *et al.* [JL98] proposed a method in which the path of the camera is created by declaring a set of properties on the desired shot including the vantage angle, framing and size of objects. They used pure interval methods

to compute camera paths, which yields sequences of images that satisfy temporally indexed image properties. The path is further modeled by using a parameterized function (of degree 3) for each *dof* of the camera. Christie *et al.* [CLGL02] proposed a range of improvements, from the inclusion of more expressive camera path constraints to the integration of propagation techniques in the solving process. They model the camera path as a set of primitive camera movements (*hypertubes*), which are then sequentially linked together. These primitives include travelings, panoramics, arcings and any composition of these primitives. Their interval-based approach guarantees the satisfaction of properties all through the shot. Each primitive is then treated as a separate, but related, constraint satisfaction problem. The problem is processed sequentially, by constraining the end of hypertube i to join the beginning of hypertube $i+1$.

Constraint-based planning techniques allow implementing cinematic-like camera motions. They however do not account for issues related to the geometry of the environment, such as collisions with the scene or occlusions of key subjects. Moreover, as for the camera composition, pure constraint-based approaches fail to find an acceptable solution when not all constraints can be satisfied. To account for this problem, one may use optimization-based approaches, which allow providing sub-optimal solutions for the camera path.

4.4 Optimization-based Approaches

Optimization-based planning techniques are based on the construction of a representation of the environment, then on a search (global or local) in this structure. Representations of environments can be categorized into three classes: Cell Decompositions [TS91], Artificial Potential Fields [Kha86], and Sampling-based Roadmaps.

4.4.1 Artificial Potential Fields

Potential fields come from theoretical physics and the study of the interaction of charged particles in electrostatic fields. A path-planning problem can be modeled by considering both the obstacles and the camera as similarly charged particles. The solving process is based on a series of local motions following the steepest descent [Kha86]. A field function F is defined as the sum of attractive potentials (the targets) and repulsive potentials (the obstacles). For efficiency, repulsive potentials attached to obstacles are usually set to zero outside a given distance from the obstacle (see Figure 2.11). The gradient of the field function F at position pv then gives the direction of the next motion:

$$M(p) = -\nabla F(p)$$

For example, [HKB+97] and [CKL+98] presented a technique that avoids collisions for guided navigation in the human colon. The surfaces of the colon and the center line of the colon are resp. modeled with repulsive and attractive fields. Burelli and Jhala [BJ09] moreover proposed to control both the camera motion and the visual composition by casting frame constraints into force fields. They employed Artificial Potential Fields to move the camera and solve frame composition tasks. Obstacle avoidance is

modeled using repulsive forces and frame composition is obtained by translating frame constraints into forces affecting both the position and the look-at point of the camera.

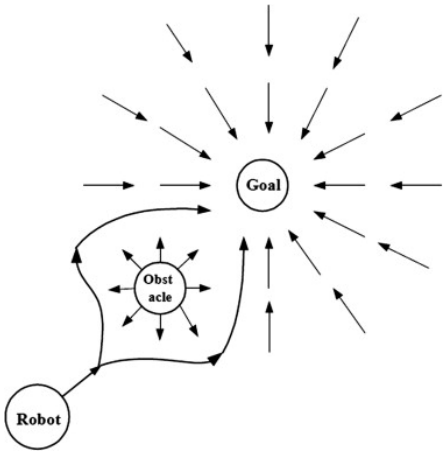


Figure 2.11 – Artificial potential field. A field function is defined as the sum of attractive potentials (goals) and repulsive potentials (obstacles). The gradient of the field function then gives the direction of the next move.

Dedicated to the specific task of creating overviews of human motions (*e.g.* from mocap data), Assa *et al.* [ACoYL08] casted the problem of camera control as an energy minimization process guided by the conjunction of external forces (describing the viewpoint quality at each location and time) and internal forces (enforcing smoothness on the generated path). Internal forces are composed of a number of criteria based on the motion analysis of a single character including widest aspect descriptor (an approximation of the largest projection of the character) and limb visibility (see Figure 2.12). Other criteria such as motion direction and facing the character are included. As a result, the system generates a sequence of camera paths (with edits) using an optimization process on a potential field defined by the aggregation of forces. Interestingly the authors provide means to detect the best moment to perform cuts between shots (as a threshold on path quality and shot duration). While purely based on character motion analysis and viewpoint quality heuristics, the process generates meaningful overviews of human motions without integrating semantic aspects and notions of continuity between shots (other than line of motion).

The low cost of implementation and evaluation of potential fields make them a candidate for applications in real-time contexts. The efficiency of the method is, however, overshadowed by its limitations with respect to the management of local minima. It also has difficulties incorporating highly dynamic environments. Some authors however proposed some extensions. Beckhaus [Bec02] relies on dynamic potential fields to manage dynamic environments by discretizing the search space, and therefore only locally re-computing the potentials. In [BY10], a process similar to [BJ09] is used to find the best camera configuration, and a stochastic population-based global search is employed to avoid premature convergences to local minima.

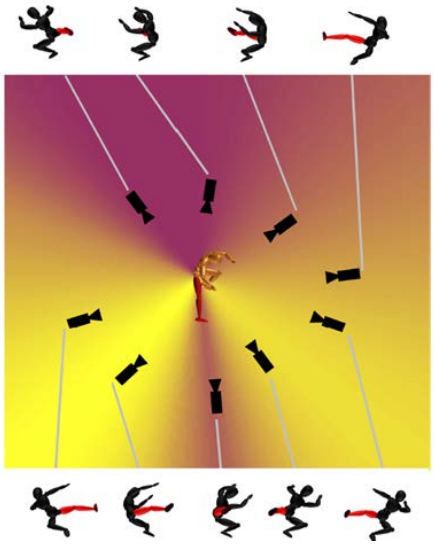


Figure 2.12 – Heat map illustrating an analysis of viewpoint quality *w.r.t.* limbs visibility (the right leg here) [ACoYL08]. Yellow areas represent viewpoint with best quality, *i.e.* for which the projected size of the right leg silhouette is maximal.

4.4.2 Cell Decomposition

Cell decomposition approaches split the environment into spatial regions (cells) and build a network (usually known as a *cell-and-portal graph*) that connects the regions. Navigation and exploration tasks then utilize this cell connectivity while enforcing other properties on the camera. In [AVF04], for instance, Andùjar *et al.* proposed a technique to ease the navigation process and achieve shots of key subjects and locations. Critical way-points for the path could be identified by using a cell-and-portal decomposition of the scene together with an entropy-based measure of the relevance of each cell. The search of the camera path is then operated in two steps. A global search is operated in the cell-and-portal graph. In each traversed cell, a path is then searched between the entry point and the exit point of the cell (see Figure 2.13).

4.4.3 Sampling-based Roadmaps

Roadmap planners operate in two phases: they first sample the space of possible configurations, then construct a connectivity graph (roadmap) linking possible consecutive configurations. There are three main classes of sampling-based methods: the Rapidly-exploring Random Trees [Ove92, KL00], the Probabilistic Roadmaps [KL94] and the Corridor Maps [GO07].

The Rapidly-exploring Random Trees (RRT) method is based on the online iterative construction of a search tree, whose root node is the source camera configuration. The construction process of the tree then attempts, at each step, to reach the target camera configuration. An illustration of this construction process is given in Figure 2.14. This method has been designed for single-query problems and is not well-suited for multiple-query problems. To our knowledge, no use of this method has been made in controlling

Figure 2.13 – Cell decomposition and path planning. A cell-and-portal model partitions the search space and the relevance of each cell is evaluated with an entropy-based measure. A path is built by ordering and connecting the most interesting cells [AVF04].

a virtual camera.

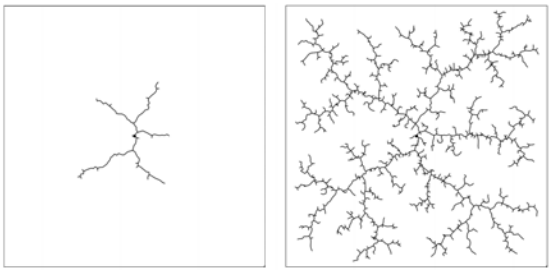
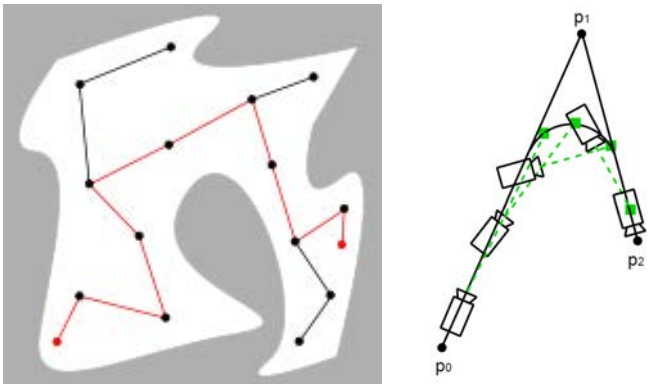


Figure 2.14 – The Rapidly-exploring Random Trees (RRT) [Ove92, KL00] are based on the inline iterative construction of a search tree. Its root node is the source camera configuration, and it tries, at each step, to reach the target configuration.

The Probabilistic Roadmap Method (PRM) is based on the offline construction of a (static) roadmap, by randomly sampling the space of camera configurations. Sampled camera configurations that are selected (*i.e.* outside any obstacle) are denoted as nodes in the roadmap. The construction process then connects each pair of possible consecutive configurations by a straight line segment (denoted as an edge in the roadmap). The criteria to create an edge in the roadmap is that the related segment is free from collision with the geometry of the environment. A search of a path can finally be operated, by adding the source and target configurations to the roadmap, then applying a search algorithm (an A* or IDA* for instance) to find an acceptable path in the roadmap (see Figure 2.15a). Probabilistic roadmap approaches have been used to compute collision-free paths to correct a user's input, as described in [LT00].

The main issue of PRMs is their lack to produce natural-looking paths (*i.e.* enforcing continuity in the camera path). Indeed, the planned paths are broken lines. They are C^0 continuous. Additional smoothing techniques are necessary on these paths. In



(a) **(b)**

Figure 2.15 – The Probabilistic Roadmap Method consists in building a connectivity graph between randomly sampled configurations then building a path between an initial and a final configuration. (a) The computed path from a source to a target configuration (in red). (b) The planned path can be smoothed by using circular blends [NO04].

[LT00], Li and Ting used a global roadmap. During the interactive navigation process, a set of possible paths are then selected according to the current configuration and the user's input, from which the shortest and smoothest one is chosen. Nieuwenhuisen and Overmars [NO04] also proposed a technique to improve the computed path. They used circular blends between the edges that meet at a node, and replaced parts of the path by a circle arc lying in the plane spanned by the two edges. By this process, the path is made C^1 continuous (see Figure 2.15b).

Another issue of PRMs is their lack of flexibility. The search is restricted to the sampled configurations and segments linking them only. They consequently can not account for dynamic obstacles. In their real-time camera planning system, Li and Cheng [LC08] used a roadmap update strategy similar to the one called Lazy PRM, proposed in [BK00]. The validity of a node or a link is checked only when the node or the link is under consideration during the search. They then used an "anytime" search (*i.e.* a search operated within a given time budget) to build camera paths.

To overcome these two lacks, Geraerts and Overmars [GO07] also proposed an other approach: the Corridor Map Method (CMM). The CMM is based on a two phase process. First, in an offline phase, a system of collision-free corridors is created for the static obstacles of the environment. Then, in a query phase, paths are planned inside the corridors while avoiding dynamic obstacles. In [Ger09], Geraerts used the CMM to plan camera motions. Their technique tackles, in real-time, the trade-off between good motions (trajectories that are collision-free and keep the target in clear view), and frame coherence (*i.e.* the view changes smoothly such that the viewer does not get disoriented).

In the last two sections, we have focused on automatically computing camera placements and sequences of camera placements to build shots. It is now necessary to take

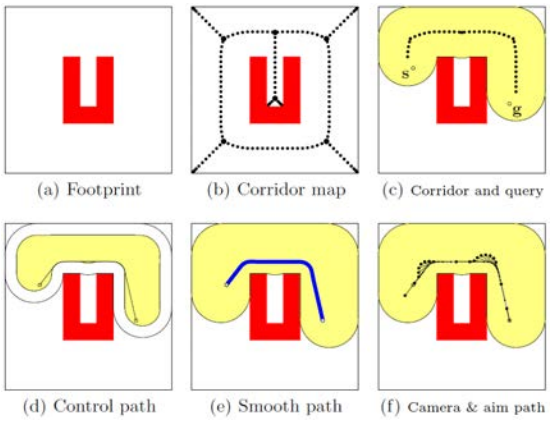


Figure 2.16 – The Corridor Map Method (CMM) [GO07] is based on a system of collision-free corridors. Corridors are created for static obstacles. Paths are the planned inside the corridors while avoiding dynamic obstacles.

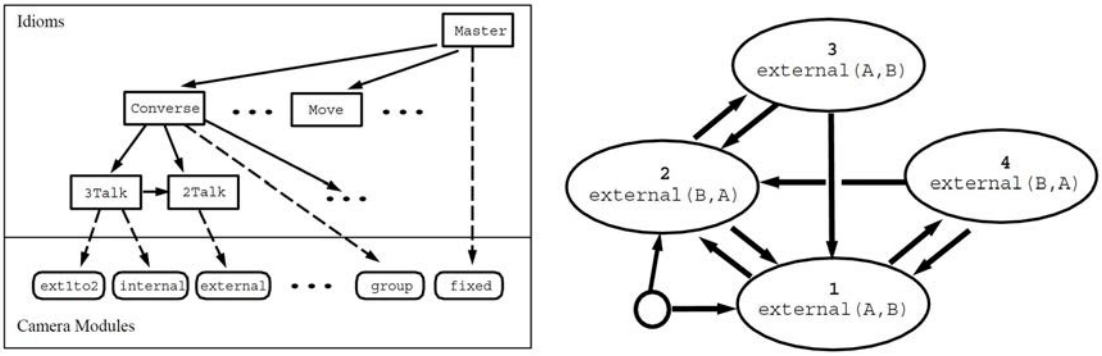
a look at editing shots together, *i.e.* constructing proper sequences of shots, so as to satisfy cinematographers' communicative goals. In the next section, we will review techniques proposed to automate the editing and storytelling process.

5 Automated Editing

The cinematic expertise *w.r.t.* editing is built upon a wide range of empirical rules, which are used differently by cinematographers (*e.g.* timing, preferred shots, preferred camera movements). Researchers have investigated means to encode this knowledge. We here review proposed editing approaches, ordered from purely procedural approaches to more expressive approaches.

He *et al.* [HCS96] proposed the first cinematography system, namely The Virtual Cinematographer. The authors encoded cinematographic expertise (film idioms) as small hierarchically-organized finite state machines (see Figure 2.17). Though their system runs in real-time, it is fully procedural and thus lacks expressiveness. Indeed, no variation is possible in the directorial style. To each prototypical situation corresponds a single idiom (*i.e.* a single finite state machine) which is hard coded in the system. The execution of such state machines is obviously deterministic.

To overcome this lack of expressiveness, researchers proposed high-level declarative languages for encoding film idioms, and structured a film being generated as a *film tree* – a AND/OR graph (see Figure 2.18). Christianson *et al.* [CAH+96] proposed a language to declare film idioms as procedural sequences of shots. Their system then operates in two steps. It first expands the film tree until it has compiled detailed solutions for each idiom. A heuristic evaluator then selects appropriate candidate frames for each scene. Amerson *et al.* [AKY05] abstractly defined film idioms as weighted constraints, and in case of an over-constrained problem, used a procedural model for the relaxation



(a) idiom choosing **(b)** idiom 2Talk

Figure 2.17 – Cinematic expertise (*i.e.* film idioms) is encoded as hierarchically organized finite state machines [HCS96].

of constraints. Friedman and Feldman [FF04] split film idioms into shots (default viewpoints, list of viewpoints, preferred viewpoints) and cut rules (minimum change in angle, do not cross the line of interest). They then used a non-monotonic reasoning process to output the sequence of shots. A good property of their approach is that it enables emergent behaviors to appear, *i.e.* in case of conflicts between rules, shots can be introduced without being explicitly specified.

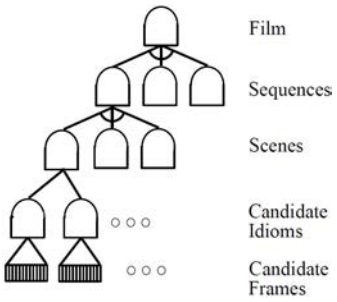


Figure 2.18 – Film tree [CAH+96]. a movie is organized into a hierarchical structure.

Although such idiom-based systems enable expressing cinematic knowledge for simple situations (*e.g.* discussions between 2 or 3 actors), they fail to be expressive enough to take the users' communicative goals into account, and necessitate the explicit definition of idioms for each possible situation. To reach a higher level of expressiveness, it is necessary to provide editing rules that are more generic, and to propose models of users' communicative goals.

The earlier attempt to account for user's communicative goals is UCam [BL97]. Users can describe their visualization preferences by using a Cinematographic Specification Language (CSL). UCam constructs a cinematographic user model consisting in probabilistic micro-level camera planning directives (*e.g.* shot type, camera orientation, minimum shot durations). A camera planner then interprets this model to compute

executable directives for shot type, viewing angle, viewing elevation, transitions (cut, tracking, panning), shot duration, and panning/tracking speeds in real-time. Their approach works extremely well in real-time, but still lacks expressiveness. Indeed, their user model is restricted to eight possible directorial styles, as a combination of three features: a viewpoint style (informational or dramatic), a camera pacing (slow or fast) and a transition style (gradual or jump).

More recently, significant advances have been made in improving the expressiveness of virtual cinematography systems. In their real-time navigation system, Li and Cheng [LC08] also integrated editing rules in their local PRM. They implemented two cut rules (LOI and minimum change in angle) in a simple way. The nodes of the local PRM are split into regions (see Figure 2.19). A cut is then possible between two regions lying on the same side of the LOI, and which are not adjacent. Their technique is however limited to tracking a single subject; the PRM being locally defined around a single subject. In what could be seen as an extension of [ACoYL08] to multiple characters [YLL12], Yeh *et al.* proposed to automatically extract *social events* from the joint analysis of multiple character motions (*e.g.* related to trajectories and similarities in distance and direction of character motions). Events are then processed, analyzed for spatio-temporal correlation and ranked so as to generate a motion clip segmentation. The motion clip segmentation is used as a basis to express an optimization problem on the camera parameters for each motion clip (actually long motion clips can be separated into segments solved individually). Following the lines of [ACoYL08], the optimization process is guided by internal, external and continuity forces shaping a potential field. Continuity forces integrate the 180-degree rule as well as the jump-cut rule. Automated editing of different viewpoints has further been addressed in an approach similar to [YLL12]. The objective is to properly convey motions of multiple humans in a real-time context [AWCO10] (while [YLL12] reported overall computation time between 1 and 3 minutes). The authors consider a set of virtual cameras animated in real-time that shoot multiple-character motions and offer means, not to control the cameras but to appropriately select viewpoints that maximize a measure of correlation between the motion of the characters in the scene, and the projected motion of the characters on the screen. The hypothesis is that a strong correlation is a good indicator of viewpoint quality. The approach uses a correlation analysis based on a optimized implementation of CCA (Canonical Correlation Analysis). The motion capture data is viewed as a sequence of joint rotation vectors (the animation stream) and the projected data is a rendering of the 3D scene (the view stream). At each time step, the correlation is computed for a number of camera viewpoints and camera paths (the computation of their trajectories is either performed randomly or uses basic heuristics such as three-quarter views or over the shoulder cameras), and the best one is selected. In order to perform cuts between viewpoint an interesting measurement of quality erosion is proposed. The measurement defines a maximum shot duration (3 seconds) coupled with a quality erosion based on the evolution of the quality of the current view in comparison with the best view of the sequence. Furthermore, continuity in edits are guaranteed by filtering the viewpoints which do not respect the jump-cut rule or the 180 degree rule, before correlations are computed. The necessity to compute and analyze the projected view for each camera limits the number of cameras that are simultaneously

considered. Their technique is further limited in the number of subjects considered when performing cuts; the jump cut and 180 degree rules being only applicable to either a single subject or a pair of subjects. Kardan and Casanova proposed a method for defining hierarchical lines of actions [KC08], and identified relevant first principles of cinematography for using these lines of actions. Their approach is more flexible and powerful than previous techniques because it naturally generalizes to any number of subjects. They also provided an explicit enumeration of heuristics for continuity editing and stylistic rules resulting in various possible edits [KC09]. Their approach is, however, only suitable for static scenes.

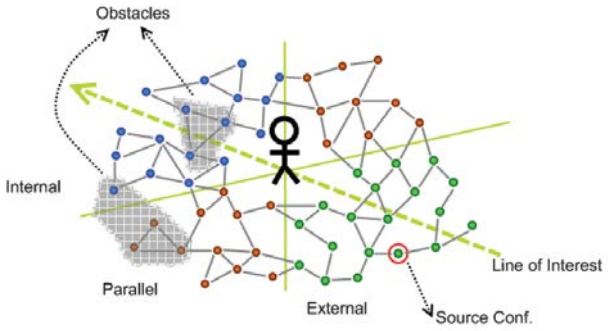


Figure 2.19 – Intercut PRM [LC08]. The local PRM is augmented with editing information. Nodes are categorized into regions of space. Cut rules (180°rule and minimum change in angle) are then implemented by cuts between two regions on the same side of the line of interest and which are not adjacent.

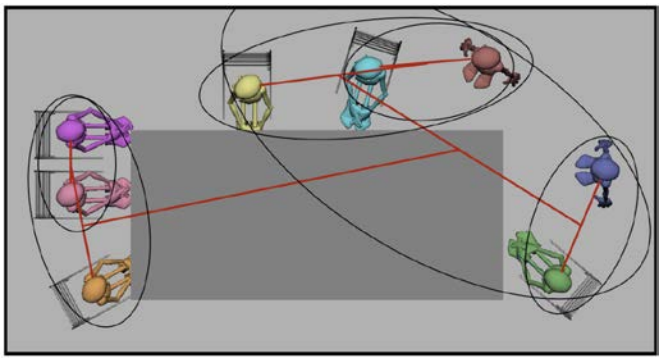


Figure 2.20 – Hierarchical lines of action are built by grouping actors [KC08]. The line of interest relative to a pair of subject is then the line of action built for their respective groups. This enables filming dialogues whith more than two or three subjects, while enforcing space continuity.

To get closer to how real movies are built, the additional control of the staging has also been explored. Elson and Riedl [ER07] proposed Cambot, a system to assist users in machinima production. They defined three different levels (or facets) of cinematic knowledge (see Figure 2.21): the *stage* (area of space that is assumed to

be free of occlusion and obstruction), the *blocking* (geometric placement of subjects *w.r.t.* the center point of a stage), and the *shot* (position, rotation, and focal length of a virtual camera *w.r.t.* the center point of a stage). These modular and reusable facets of cinematic knowledge enable the coordination between positions and movements of the camera and the actors. CAMBOT then relies on a dynamic programming algorithm to search for the best sequence of shots. CAMBOT also considers a few directorial style parameters, such as pacing (slow or fast), viewpoint preferences and intensity.

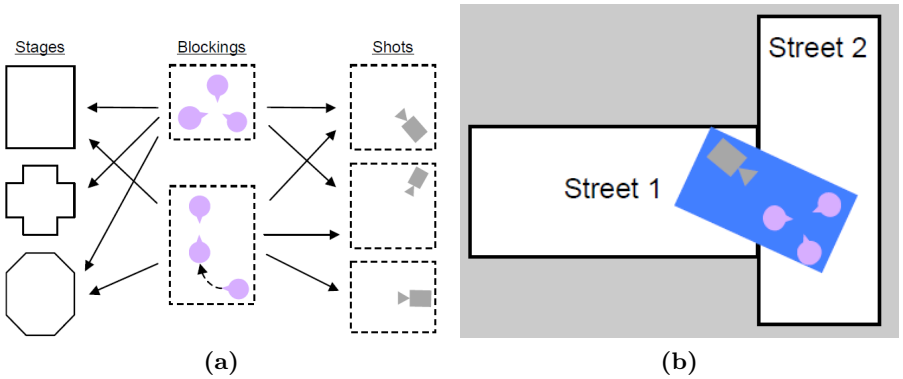


Figure 2.21 – Coordination of positions and movements of camera and actors [ER07]. (a) relationships between the different facets of cinematic knowledge; blockings and shots are defined *w.r.t.* to the center point of a stage; (b) a stage is associated with a blocking and shot.

As introduced in Section 1, every shot provides information (about narrative elements). Cuts may also provide information, *i.e.* links between narrative elements; these links can be intentional or unintentional. However presented approaches still lack considering the viewer's comprehension of the complete sequence of shots and cuts. Jhala and Young [JY10] proposed Darshak, that uses cognitive models of narrative comprehension to automatically construct cinematic discourse of a story in a 3D environment. Dramatic situation patterns are realized through abstract communicative operators; these operators represent operations on a viewer's beliefs about the story and its telling. Darshak then relies on a hierarchical partial-order causal link (POCL) planning algorithm to generate story events and directives for filming them.

On one hand, recent approaches have enabled modeling, with some flexibility, both classical cinematic knowledge and aspects of directorial style, and to compute acceptable sequences of shots. On the other hand, two issues remain: (1) it is still difficult to express a specific style (or genre), and (2) there are so many possible combinations of shots and transitions that it is difficult to properly evaluate them.

6 Visibility/Occlusion Handling

Visibility is a transverse issue in controlling a virtual camera. Visibility computation received much attention in computer graphics; for a complete survey on visibility, see [COCS+03]. Paradoxically, it has been under-addressed in the virtual camera control domain. In most existing approaches (*e.g.* virtual exploration, computer games or virtual storytelling), the visibility of subjects is computed by using raycast-based techniques on an abstraction of subjects (commonly bounding boxes). There are however better ways of tackling visibility issues related to virtual camera control. We here first review the main visibility concerns when controlling a virtual camera, then review the most significant visibility computation techniques proposed to tackle these issues.

In a camera control context, we identified four issues in handling visibility:

Static *vs.* Dynamic One question is whether the scene is purely static, or contains dynamic elements (occluders/occludees). For the static part of the environment, the visibility information can be pre-computed. For the dynamic part, the visibility information must be computed on-the-fly, with real-time constraints imposed by interactive applications.

Accurate *vs.* Approximate *vs.* Inaccurate The visibility computation techniques can be categorized into three main classes, *w.r.t.* the level of accuracy of the result: the techniques providing an accurate visibility information (*i.e.* a precise degree of visibility, as a real value between 0 and 1), the techniques providing an approximate visibility information (as an approximate degree of visibility, such as visible/partially visible/occluded), and the techniques providing an inaccurate visibility information (as a boolean result such as visible/occluded).

Adapted to path planning *vs.* Adapted to viewpoint planning The potential of a camera control approach is often dependent on the capabilities of the visibility computation technique it uses. In order to compute unoccluded viewpoints (*e.g.* for checking the visibility/occlusion from the current camera viewpoint, or for planning a cut to a new camera viewpoint), the visibility process must compute the visibility/occlusion of key subjects from discrete points of space. In the same way, to plan a path between two given camera viewpoints, the visibility process must compute the visibility/occlusion of key subjects along possible sequences of intermediate camera viewpoints.

Anticipation In a range of applications (*e.g.* reactive approaches), the visibility process must be capable of anticipating future occlusions of key subjects. In other words, it must provide a good prediction of their visibility over time. This may then serve for instance to predict a change in the camera position in reaction to past and/or current changes in the scene.

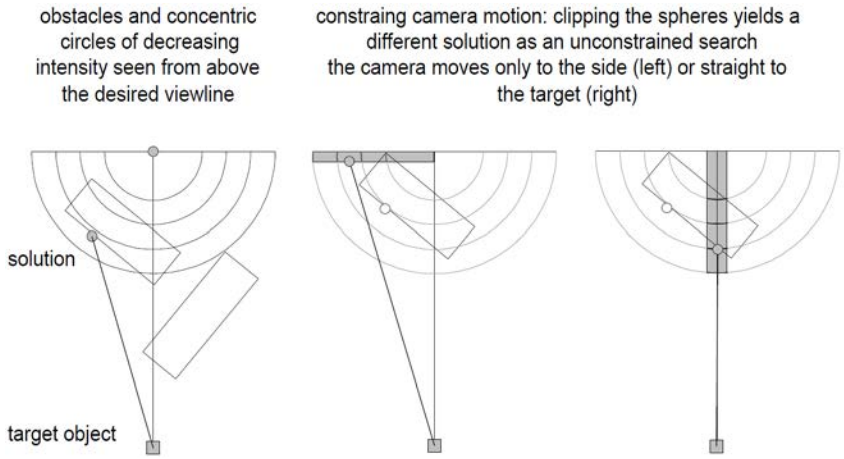
We will now focus on relevant contributions on visibility handling for camera control.

Halper *et al.* [HHS01] proposed handling occlusion avoidance by using *Potential Visibility Regions* (PVR). They imposed constraints on the camera movements as a set of geometric constraints using polygons (spheres). Polygons are shaded *w.r.t.* their

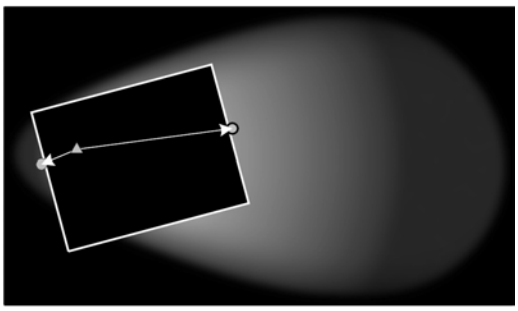
preference (a brighter color meaning a higher preference). To find the best position that satisfies visibility requirements, a visibility map is created. This visibility map only considers the depth information from potential occluders. The potential visibility geometry is then rendered to the buffer from most desirable to least desirable regions. The result is an image buffer, whereby the brightest colors that first pass the depth test are visible. The brightest color in the image buffer, therefore, denotes the most desirable position for the camera to move to (see Figure 2.22). They also tackled the problem of occlusion anticipation. Their method uses approximative calculations for future scene and object state based on past trajectory and acceleration information, and solve the camera for that predicted state. The current camera path is adapted so that the camera will be at this predicted position at the predicted time t. An estimated camera position is finally computed for the next frame along this path (see Figure 2.23). Halper *et al.* 's approach is however only suited for planning camera paths locally (*i.e.* step by step). If no solution exists in the rendering, their method will fail finding an unoccluded position (even if one exists), and planning a global camera path to this unoccluded position.

To enable computing collision-free and occlusion-free camera paths in arbitrary environments, Oskam *et al.* [OSTG09] proposed to augment a roadmap graph with visibility information, what they call a *visibility-aware roadmap* structure. Their method relies on the precomputation of a coarse representation of the scene (with a grid of bounding spheres), the creation of a roadmap by connecting overlapping spheres (through circular portals), and an estimate of pair-wise visibility between all spheres by using a Monte-Carlo ray tracing algorithm (see Figure 2.24). Their system executes a path planning algorithm using the precomputed roadmap values to find a coarse path. This path is then refined using a sequence of occlusion maps computed on-the-fly. Their global planning strategy on the visibility-aware roadmap enables both large-scale camera transitions and a local third-person camera module following a player and avoiding obstructed viewpoints. The data structure itself adapts at run-time to dynamic occluders of the environment. Dynamic occluders are handled by performing an on-the-fly update of the roadmap information. Moving objects are approximated by one or more bounding spheres. When searching a path in the roadmap, all the connections between two circular portals that touch the bounding spheres are marked as being occupied. This prevents the camera from colliding with the moving object. To anticipate future occlusions of the player, they also proposed a *risk prediction* method. They extended an algorithm inspired by the work of Bandyopadhyay *et al.* [BLAH06]. Their prediction algorithm uses the visibility-aware roadmap to find a path to the sphere closest to the player and which is not visible from the camera's current position. This path represents the highest risk of escape. The camera's current position is then rotated to a new vantage so that this escape route is in view. An illustration of their method is given in Figure 2.25. Their technique is however limited to single-point focus targets.

Christie *et al.* [CON08b] presented an approach to the real-time evaluation of the visibility of multiple target objects. The visibility of both targets is simultaneously computed for a large sample of points. This involves to perform a low resolution projection from points on pairs of target objects onto a plane parallel to the pair and behind the camera (see Figure 2.26). The combination of the depth buffers for these



(a)



(b)

Figure 2.22 – Potential Visibility Regions [HHS01]. (a) sample use of rendering camera solution regions; selection is based first on highest color, then on closest angle. (b) The camera solution region is warped according to motion characteristics of the camera. In this case, selected regions are favored in the direction of motion, and the algorithm returns the point to the right even though its angle of change is greater than that of the point to the left.

two projections then enables creating visibility volumes around the camera, with the joint visibility information on the pair of objects. They also provided an extension of this pair-wise computation for three or more target objects. To mitigate over-reactive camera behavior, visibility results are aggregated in a temporal window. They moreover addressed the problem of idealization of the target objects geometry. To this end, they used a stochastic approximation of the physical extent of target objects by selecting projection points randomly from the visible surface of the target object. A different projection point is used at each frame, so the aggregation of visibility information along the time provide a good approximation of the degree of visibility of the target object.

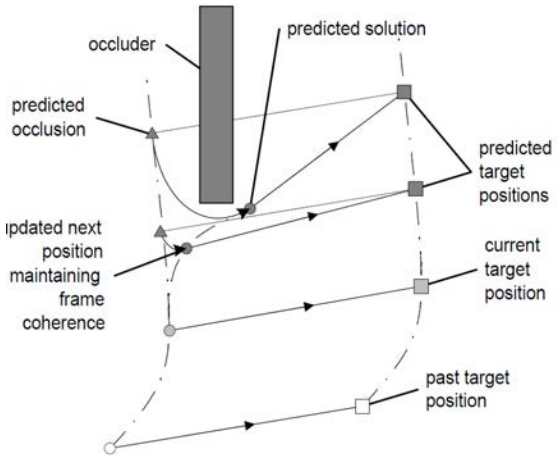


Figure 2.23 – Altering the camera path to adapt to a predicted occlusion [HHS01].

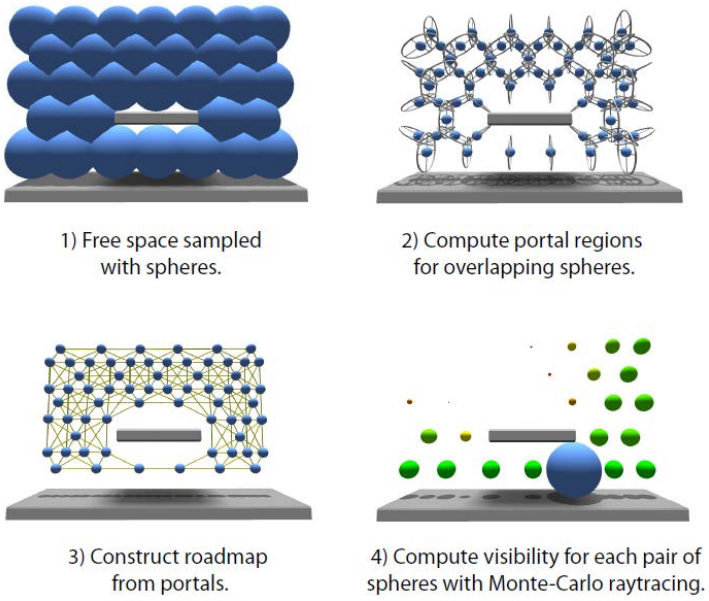


Figure 2.24 – Sphere-to-sphere visibility [OSTG09]. Based on an initial geometry of the environment, a spatial discretization is computed. A roadmap is then built from the overlap regions. Finally, for each pair of spheres, a visibility probability is computed with a Monte-Carlo raytracing algorithm.

7 Conclusion

From the literature study, we detailed a number of methods dedicated to controlling a virtual camera in 3D environments. We highlighted that the research community is moving towards models of cinematic knowledge that are more and more expressive. There are however some important issues to tackle.

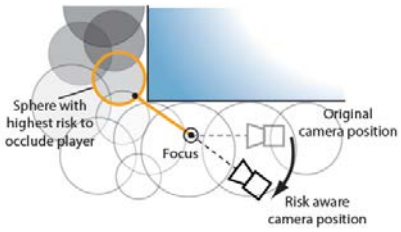


Figure 2.25 – Proactive camera movement [OSTG09]: the algorithm finds the closest sphere (indicated in orange) that has reduced visibility from the original camera position. After detecting a high escape risk, the camera's viewing direction is aligned toward this sphere. Spheres with reduced visibility from the original camera position are shown in darker shades of gray.

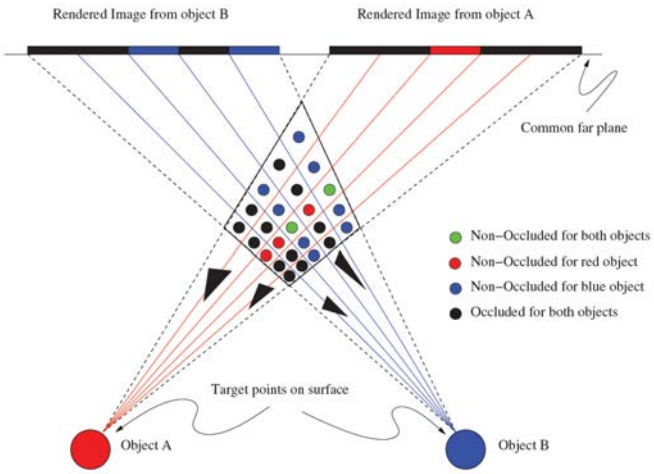


Figure 2.26 – Computation of the visibility inside a camera volume by using the depth buffer [CON08a]. The scene is rendered from each target back towards the actual position of the camera. The visibility information for all target objects is then composed as a visibility volume.

In this thesis, we identified three research axes:

1. Existing automated approaches generally focus on one or two cinematic aspects only: visual composition, tracking of targets (camera planning), cinematography (editing, with no account for the visibility of subjects). These techniques also lack expressiveness, *i.e.* they provide limited account of styles or genres. Furthermore, the handling of visibility appears as an essential aspect in controlling a camera, but has been under addressed in the field. In Chapter 3, we propose the first fully integrated cinematic engine, which handles both viewpoint computation, planning and editing, while accounting for visibility and proposing means to control some elements of style.
2. On one hand, fully automated cinematography approaches pose the problem of tuning a large set of high-level parameters to account for each single aspect (visual

properties, properties on the camera path, editing properties, visibility properties). Furthermore, not all elements of directorial style can be modeled. Though automated approaches provide good results, the result intended by a user is generally more subtle and cannot be easily expressed. Indeed, one may use metrics to approximate a given directorial style (*e.g.* statistics of shot preferences, or shot duration), but we will never be capable of reproducing exactly the creative process of a given editor (*e.g.* build a "Spielberg-like" movie). This therefore raises the necessity to let the user interact in the editing process. On the other hand, some interactive control techniques enable finely controlling cinematic aspects (visual composition and camera motions) within a single shot, but lack the handling of the aspects related to editing. In Chapter 4, we propose a hybrid approach that build upon an evaluation of the different cinematic aspects, and combines an automated control of modeled cinematic aspects with direct interaction of an editor in the editing process.

3. The central element in controlling a camera is the visual composition. A large range of computer graphics applications rely on the computation of viewpoints over 3D scenes that need to display a number of characteristic composition properties. Such problems are generally casted as non-linear optimization problems in a 7 degree-of-freedom search space. Given the size of the search space and the computational cost in the evaluation of composition properties (typically the visibility of subjects), this optimization process is a time-consuming task and hampers the use of evolved composition techniques in most applications. In Chapter 5, we introduce the *Toric Space*, a novel and efficient approach to virtual camera control. Based on a simple parametric model of the space of camera configurations, the *Toric Space* has the intrinsic advantage to reduce the search space, and most of classical visual properties can be easily expressed and combined in this space.

A Cinematic Engine for Interactive 3D Environments

3

Related publications: This chapter presents a unifying approach to a fully automated cinematic engine. As described in the state of the art, current contributions in the field generally focus on individual aspects of cinematography: (i) on-screen composition [BMBT00], (2) visibility and path-planning [OSTG09] or (iii) editing [CAH+96, ER07, AWCO10]. They moreover suffer from a limited integration of directorial style and visibility enforcement. In this chapter, we propose CineSys, a fully automated cinematic engine for interactive environments accounting for both viewpoint computation, viewpoint planning and editing.

The provision of a fully automated camera control system is complex as it raises four important challenges:

- First, such a system needs to be underpinned by a narrative model that both structures the way the actions are organized, and allows to reason on their relative importance and presentation. Following Chatman's bipartite model [Cha80], interactive narratives have two levels: (1) the story, *i.e.* the chronologic account of events over time; and (2) the discourse, *i.e.* the way in which events are organized and presented to the spectator.
- Second, the system should provide the user with well-composed viewpoints of key subjects in the scene following established conventions in the literature, while maintaining the visibility of these subjects in fully interactive 3D environments. For example, the simultaneous computation of the visibility of multiple subjects in a dynamic environment requires both an efficient solution and a good estimation of the visibility for a large range of viewpoints.
- Third, the system must address the innate complexity of planning path between viewpoints. For example, selecting appropriate tracking shots (sequences of viewpoints to follows an event) involves planning a camera's motion in a dynamic environment whilst simultaneously taking account of the visibility of scene elements from viewpoints along the spatio-temporal path.
- Last, a critical issue is related to the challenge of operationalizing editing rules and conventions in a sufficiently expressive model of automated editing. Such rules are often expressed as a set of idioms (*i.e.* a typical set of camera configurations that, when assembled, bring to the screen the viewer's understanding of the spatial and causal relations pertained to a specific action). While current implementations strongly rely on these pragmatic representations to model camera editing systems, there is a clear necessity and demand to perform a shift toward more expressive

representations that augment the range of possible variations in the montage. To this end, another essential task consists in identifying some of the parameters and devices (*i.e.* indicators) that underpin directorial style, such as pacing, dynamicity, preferred views and enforcement of specific cinematic conventions. As a consequence, the challenge consists in the design of a model expressive enough to control such indicators and to yield a range of variations in results.

1 Contributions

We propose a fully automated system that allows the construction of a movie from a specified sequence of low-level narrative events (*e.g.* subjects' actions and motions, or establishing subjects in the scene). In essence, our approach contributes to the domain on the following points:

Real-time integrated system Our model proposes an integrated approach that characterizes full and partial visibility of key subjects, performs camera planning between viewpoints, and enforces continuity editing rules in real-time.

Expressiveness Our cinematic engine introduces the notion of *Director Volumes* as a spatial characterization of viewpoint regions around key subjects. Director Volumes encode the empirical knowledge from practice and literature on camera staging (*e.g.* Internal, External, Parallel) as well as classical shot sizes (*e.g.* close, medium, long). Additionally, knowledge related to continuity editing and style is encoded by filtering operators over the Director volumes (line-of-interest, line-of-action, thirty-degree angle, preferred viewpoints). At last, our engine enforces on-the-screen composition that selects the best camera configuration from the resulting volumes using a local search optimization technique and following well-known composition rules.

Variation in directorial style CineSys moreover provides means to express notable variations in directorial style in a real-time context by controlling indicators such as pacing, camera dynamicity and composition. High-level narrative dimensions such as isolation, dominance and affinity between key subjects can furthermore be enforced.

This chapter is organized as follows. First, we present an overview of our approach (Section 2), before detailing the construction of Director Volumes (Section 3) and the mechanics of the reasoning process for editing (Section 4). Second, we present results on a 3D replica of the canteen scene in Michael Radford's 1984 movie (Section 7), in which we demonstrate variations in the application of directorial styles (pacing, dynamicity), together with variations in the visibility degree of subjects and in the narrative dimensions. Finally, we compare our model to the state of the art in virtual cinematography on a set of important features.

2 Overview

Our cinematic engine takes as input a flow of *narrative events* (which are not known beforehand). Our system computes appropriate viewpoints on these narrative events. A camera viewpoint is a camera placement defined at a symbolic level *w.r.t.* to a configuration of subjects, as the combination of a shot size (*e.g.* medium close-up, long) and a view angle (*e.g.* internal, external, parallel). Viewpoints are computed by selecting specific regions referred to as *Director Volumes* that capture information both on visibility and on stereotypical viewpoints. We then rely on filtering-based techniques to reason over Director Volumes and from a large set of possible camera configurations to select the appropriate candidate camera configuration (a camera configuration is a camera placement defined at a numerical level). In this filtering process, cinematographic rules are expressed as constraints (applied on with constraint-based filtering processes) and cinematographic style is expressed as preferences to choose where to position the camera, how to plan paths between viewpoints, and when to perform cuts. The output of our system is a movie which conveys these narrative events according to some cinematic rules and directorial style parameters. A movie is defined as a sequence of shots separated by cuts, where shots are continuous sequences of camera configurations. We furthermore propose the notion of an *action shot* as a continuous sequence of camera configurations that conveys a given narrative event. A shot is then described as a sequence of (one or more) action shots separated by continuous transitions (*i.e.* camera motions).

At runtime, our system first selects the most relevant event (or set of events) and chooses the best viewpoint to convey this event by filtering the Director Volumes according to (i) the visibility of key-subjects, (i) the coherency in the sequence of viewpoints, (iii) and a characterization of style. A transition planner then determines appropriate viewpoint transitions, either by using a cut or a path-planning process between viewpoints. specifically, our system follows a five-step process to compute viewpoints and transitions (see Figure 3.1):

1. **Selecting narrative events.** Whenever a new transition is enabled, the system selects at run-time an event among all those occurring during the same time step. We here consider that relevance of an event is computed by an external process (*e.g.* a narrative engine) that distinguishes the relative importance of events running in parallel. This step selects the most relevant event; and in case of failure in the following steps, re-iterate by selecting another event, considered by decreasing order of relevance.
2. **Computing Director Volumes.** This stage turns a selected event into areas which can potentially portray the event (we refer to them as Director Volumes), by composing areas of characteristic viewpoints with an analysis of the full and partial visibility of key subjects involved in the event.
3. **Reasoning over Director Volumes.** The selection of appropriate Director Volumes among all available is performed by applying two *filtering* processes, then a selection operator. *Continuity filters* typically enforce classical continuity rules

between shots when performing a cut. They prune inconsistent regions in the set of Director Volumes (for example those that would make the camera cross a line-of-action). *Style Filters* then annotate Director Volumes *w.r.t.* elements of directorial style (*e.g.* preferred viewpoints, enforcement of a narrative dimension). A *Selection Operator* finally selects one (or a set of) Director Volume(s) from all available, by using a ranking over remaining volumes.

4. **Computing frame composition.** A numerical optimization process is performed in each selected Director Volume to compute the best camera configuration in terms of on-screen composition (exact locations of key subjects on the screen).
5. **Computing transitions.** Whenever it appears necessary to switch between Director Volumes (*e.g.* end of event, new event, subject(s) occluded, pace in transitions), the system selects an appropriate transition (either a cut or a continuous camera motion). For camera motions, we first build a roadmap over the Director Volumes. We then rely on an incremental Dijkstra search process to plan an appropriate path between the current camera configuration and a set of target Director Volumes (that may evolve as the scene evolves). When the camera enters a target Director Volume, we compute an appropriate camera configuration inside the volume, as described above, and moves the camera toward this configuration.

In the sequel we detail the major steps of our computational process, namely computing Director Volumes, reasoning over Director Volumes and performing transitions.

3 Computing Director Volumes

At the core of our approach stands a spatial representation: the *Director Volumes*. A Director Volume aggregates camera configurations that give rise to a "cinematographically" equivalent viewpoint in terms of information conveyed to the user (*i.e.* its semantic *w.r.t.* a narrative event to convey), and in terms of visibility of key subjects. The Director Volumes are computed through a three-step process:

- A first characterization process captures the semantic of viewpoints through a spatial representation of regions around key subjects. These regions represent stereotypical viewpoints, and we refer to them as *Semantic Volumes* [CN05].
- A second characterization process captures visibility information on key subjects through a spatial representation of regions of full visibility, partial visibility and full occlusion of each key subject. We refer to these regions as *Visibility Volumes*.
- An intersection operator then augments the Semantic Volumes with visibility information. This intersection operator takes as input the previously computed spatial representations and outputs a new spatial representation of regions (the *Director Volumes*) that can potentially portray the narrative event.

To implement these spatial representations of regions (or volumes), we rely on BSP data-structures. We use annotated Binary Space Partitions (a-BSPs), which are classical BSPs augmented with information on the nodes and leaves. These structures are dynamically and procedurally processed *w.r.t.* the configuration of key subjects, for each frame, and allows to efficiently characterize and partition the environment into

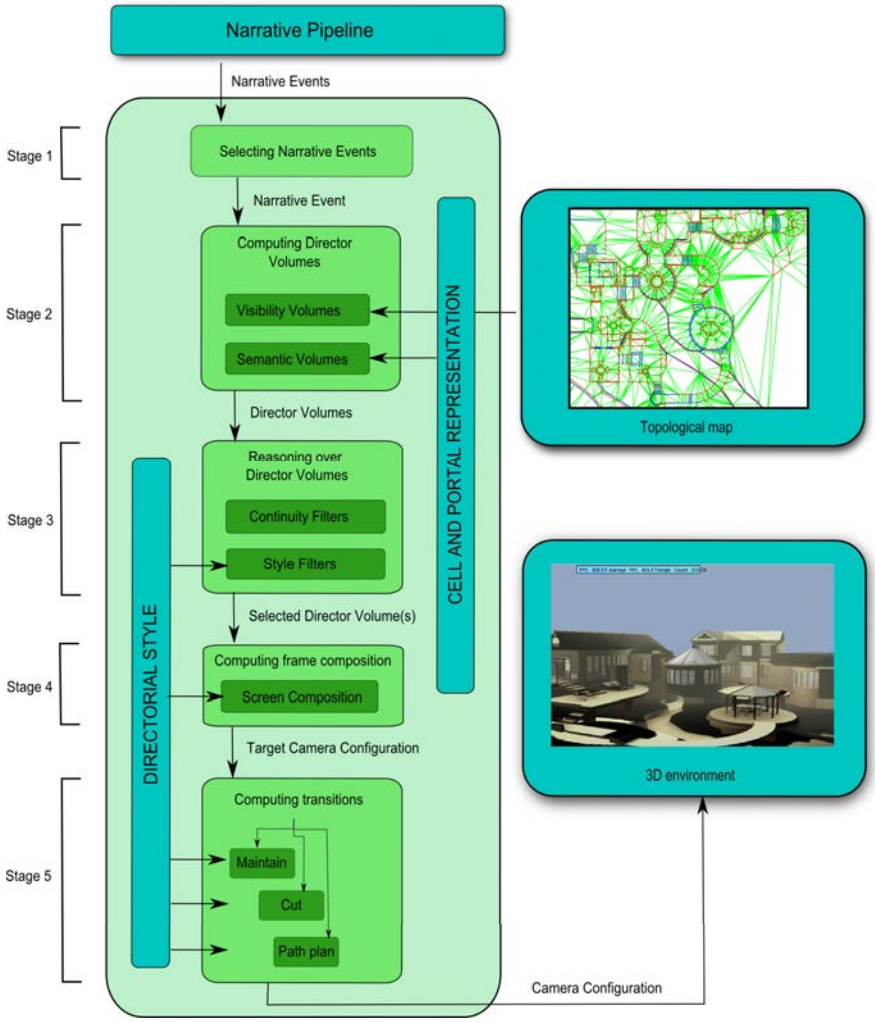
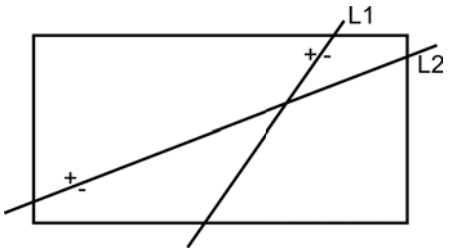


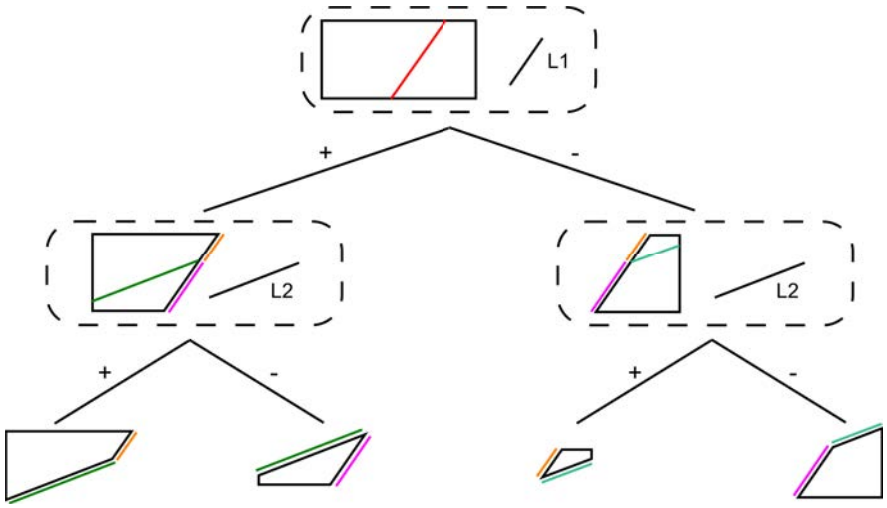
Figure 3.1 – Overview of our real-time cinematic engine CineSys. First, a relevant narrative event is selected from a flow of unfolding events. This narrative event is then turned into Director Volumes (all regions of space from which the narrative event can be portrayed). Editing is performed by reasoning over Director Volumes to select one or more target volume(s), given continuity rules, and style rules. A target camera configuration is then computed by using a local optimization process inside selected volumes, given composition rules. Finally a transition between the previous and new camera configuration is computed (as a cut or as a continuous transition).

such sets of viewpoints (see Figure 3.2). Each node is represented by an intermediate (convex) cell and a stabbing line. Nodes are further annotated with information on sub-cells' connectivity, which is updated as new intermediate cells are drawn. Each leaf then corresponds to a categorized cell (or volume), annotated with relevant information (*e.g.* viewpoint semantic, subjects' visibility). The intersection operator then operate by merging the a-BSP structures associated with both Semantic Volumes and Visibility Volumes, to create a new a-BSP structure corresponding to Director Volumes.

This a-BSP representation enables (i) an on-the-fly cell subdivision process, through the use of a description of intermediate cells in each node; and (ii) a reduction of the search complexity when characterizing a given camera configuration in the environment (binary tree search of complexity $O(\log n)$, for a BSP with n leaves).



(a) Cell splitting

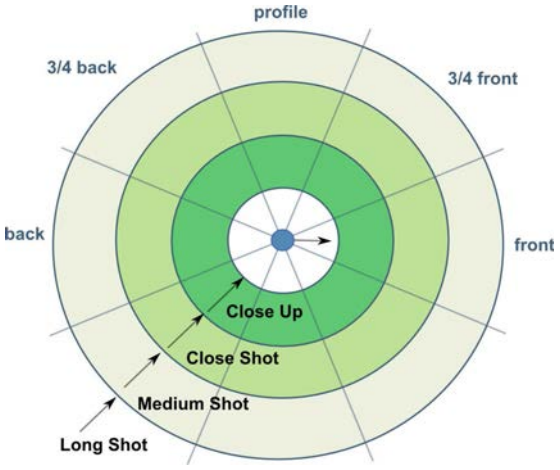


(b) a-BSP data structure over the cell

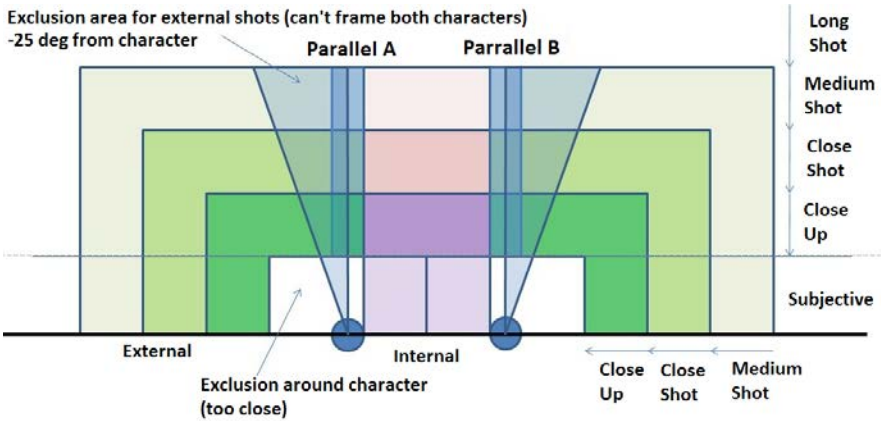
Figure 3.2 – Construction of an annotated Binary Space Partition (a-BSP) data structure. The a-BSP is procedurally processed through a top-down injection of stabbing lines. Each node (dotted box) is represented by an intermediate convex cell (in black) and a stabbing line. Nodes are also annotated with information on sub-cells' connectivity (colored segments separating two sub-cells), which is updated as new intermediate cells are drawn. Each leaf (at the bottom) corresponds to a categorized cell (or volume), annotated with relevant information (*e.g.* viewpoint semantic, subjects' visibility).

3.1 Semantic Volumes

The first step consists in translating the selected narrative event into Semantic Volumes. Semantic Volumes are a formalization of the cinematic knowledge in terms of camera staging. In our approach, each narrative event is associated with a collection of stereotypical viewpoints that convey the event according to established cinematographic conventions. A narrative event is modeled by a semantic tag (the performed action, *e.g.* "A speaks to B", where participating subjects are abstracted), a start and end time,



(a)



(b)

Figure 3.3 – Design of Semantic Volumes (a) for a single subject action, and (b) for a classical interaction between two key subjects (adapted from [Ari76]). Each blue circle represents a key subject.

as well as the participating subjects (abstracted as "A", "B", "C", etc. in the semantic tag). Typically an event such as "Symes speaks to Smith" can be portrayed through a large collection of viewpoints described in the literature (Internal, External, Apex, Subjective) together with a range of distances (Close-Up, Medium Shot, Long Shot, Extreme Long Shot) [Ari76]. Do note that for such an event, the utterance of a key subject may be portrayed either by framing the talking key subject, the listening key subject, or both (the choice is a matter of continuity and directorial style). By default each narrative event is portrayed by all possible viewpoints, though some events may require a specific subset of viewpoints (*e.g.* "Symes looks at Smith" obviously requires that the camera should not be located behind Syme).

In the task of precisely designing the Semantic volumes that encodes cinematographic knowledge, we followed an experimental process that consists in positioning

and evaluating camera configurations in a 3D modeler for key configurations of subjects (single subject, two subjects facing, two subjects not facing). This process was guided by empirical evidence in literature and a professional experienced modeler (Guy Schofield, Newcastle University). As a result we characterized the spatial extents of viewpoint regions for key configurations of subjects. Resulting Semantic Volumes are displayed in Figure 3.3. Additionally, we studied the evolution of these Semantic Volumes *w.r.t.* the distance between the key subjects and the orientations of the key subjects. In cases where the subjects are too close to each other, areas of Subjective Shots are removed. In a similar way, as the distance between key subjects increases, the range of typical shot sizes is extended for Internal Shots (Close-Up, Close Shot, Medium Shot, Long Shot).

To represent Semantic Volumes, we rely on a single *a*-BSP data-structure. This structure is dynamically and procedurally processed *w.r.t.* the configuration of the key subjects (*e.g.* single subject, two subjects). First, the root cell of this *a*-BSP is set to a bounding cell of all Semantic Volumes. Second, the regions of space corresponding to the different shot sizes and view angles are modeled with bounding cells, and stereotypical viewpoints (Semantic Volumes) are described with set operations on these regions (*e.g.* union, intersection, difference). Third, edges of these bounding cells are successively injected into the *a*-BSP structure. Last, final leaves are categorized *w.r.t.* their inclusion in the Semantic Volumes as described above.

Some sub-regions of space cannot however be selected as candidates since key subjects are fully or partially occluded. To efficiently compute the visibility information that we then store in the regions, a topological level of representation of the geometry is employed.

3.2 Visibility Volumes

The second step consists in computing and characterizing the visibility of key subjects. Real-time visibility computation is a prohibitively expensive process. We address this constraint by reducing the practical complexity through two assumptions on the geometric properties of the environment. Firstly, we estimate the visibility of a key subject (here, a character) by using a 2D convex hull of the projection of its geometry onto the floor. Secondly, we restrict the setting to static 3D indoor environments which we represent as a 2D cell-and-portal (C&P) structure [TS91]. Cells represent rooms and portals represent doorways and connections between cells. Further, hard visibility constraints are related to the portal extremities (*e.g.* walls or doorways); each boundary between a wall and a doorway supports a stabbing line representing a separation between visibility and occlusion. By assuming that the indoor environment resides on plane (*i.e.* distinct floors) we extract a 2.5D topological C&P representation. All geometric elements in the scene above a specified height (here 1.5m) are treated as occluders. The visibility process is therefore restricted to dynamic targets in static environments.

To characterize partial visibility, we extend classical cell-and-portal visibility propagation techniques [TS91] to handle from-region visibility where subjects are approxi-

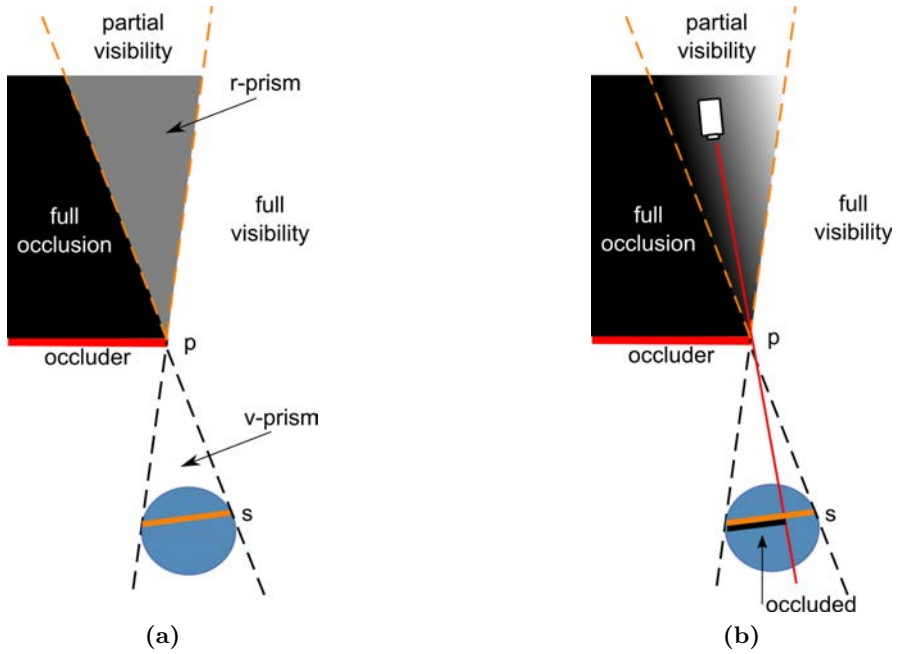


Figure 3.4 – From a portal vertex p, two stabbing lines are defined *w.r.t.* a key subject (blue circle). (a) By combining both stabbing lines, a categorization into three regions (full occlusion, partial visibility and full visibility) is obtained. 2D visibility of a key subject corresponds to a segment s (*visibility segment*) linking its extreme points. (b) From a viewpoint with partial visibility, we approximate the visibility degree of the key subject by first computing the length of the non-occluded portion of segment s, then calculating the relative proportion which remains visible.

mated as convex cells [COCS+03]. We base our visibility analysis on the use of stabbing lines to compute a dynamic categorization of the full and partial visibility of a key subject. First, we identify the current topological cell in which the key subject is located, and list all adjacent topological cells (*i.e.* which can be reached through a single portal). Second, for a portal vertex p, two stabbing lines are defined such that each line is tangential to, but on opposite sides of, the convex hull of the key subject (Figure 3.4). Each stabbing line therefore defines a point of contact (or extreme point) e on the convex hull of the key subject. The stabbing line tangential to such a point e separates two regions: the region from where e is visible, and the region from where e is occluded. As the key subject is fully included between stabbing lines, the visibility categorization of the whole key subject is then performed by combining the visibility categorization of each stabbing line. This categorization follows a two-step process:

1. A visibility prism (or v-prism) is defined as the triangle between the portal vertex p and the two extreme points of the key subject. The segment s joining both extreme points is then viewed as the abstraction of the 2D visible aspect of the key subject at point p.
2. A reversed prism (or r-prism) is computed as the point reflection of the v-prism

w.r.t. point p. This r-prism defines the partial visibility region generated by the vertex p. Three regions are then defined: a region from where the key subject is fully occluded, a region from where the key subject is fully visible, and an intermediate region from where the key subject is partially visible (r-prism). Further, for any camera viewpoint v inside the r-prism, we estimate the visibility degree of the key subject as follows. We first create a frustum F, defined as a triangle between v and s, which is used to detect the occluding scene elements. We then combine all portions segment s that are occluded by at least one scene element. We finally calculate the visible degree as the remaining proportion of s which remains visible.

This process is further repeated for all portals connected to the current topological cell in which the key subject is located (see Figure 3.5). Areas of full visibility, partial visibility and full occlusion are then propagated through the cell-and-portal representation in a way similar to [TS91].

An a-BSP representation is used for each topological cell to represent the regions of full visibility, partial visibility, and full occlusion (*i.e.* the Visibility Volumes) in this cell. The root cell of such an a-BSP is first set to the associated topological cell itself. The r-prisms are then propagated (and eventually reduced) incrementally to adjacent topological cells through portals (see Figure 3.5). The propagation of r-prisms in a topological cell is processed by using a three-step process. First, we inject stabbing lines associated with each r-prism into the a-BSP of the topological cell. Second, we categorize the leaves of the a-BSP included in at least one r-prism as "partially visible". Last, to categorize remaining leaves as either "fully visible" or "fully occluded", we rely on the information held by stabbing lines (*i.e. w.r.t.* visibility/occlusion of associated extreme points of the convex hull of the key subject).

3.3 Director Volumes

In the previous sections, we have computed volumes corresponding to semantic information on stereotypical viewpoints around key subjects, and volumes corresponding to visibility information on key subjects. The last step consists in combining this information to characterize sets of "cinematographically" equivalent viewpoints (the Director Volumes) in terms of both semantic and visibility of key subjects. The intersection operator uses the compatibility between both BSP structures to fusion them, and create an a-BSP representation of Director Volumes per topological cell. This intersection operator can be formalized as follows.

- First, we refer to the a-BSP characterizing Semantic Volumes around key subjects as $\mathcal{S}$.
- Second, we refer to the a-BSP characterizing visibility of key subject k in topological cell j as $\mathcal{V}_k^j$.
- Last, we refer to the a-BSP characterizing Director Volumes around key subjects in topological cell j as $\mathcal{D}^j$, procedurally computed as

$$\mathcal{D}^j = \left(\bigcap_k \mathcal{V}_k^j \right) \cap \mathcal{S}$$

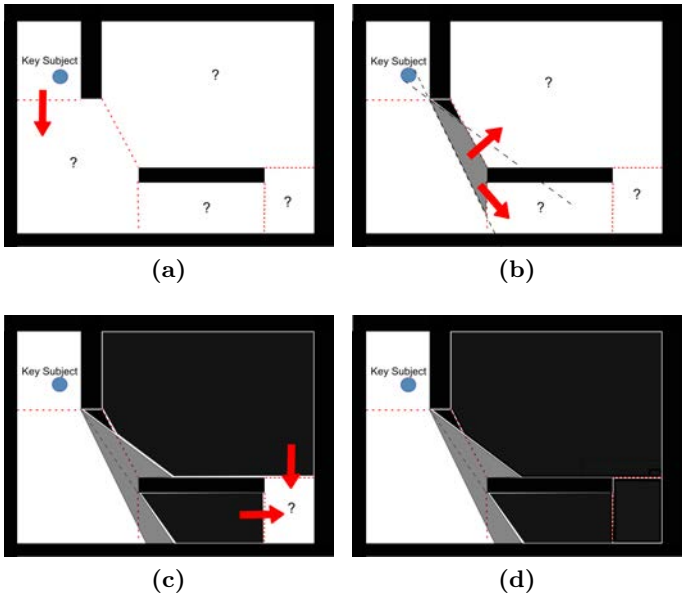


Figure 3.5 – Visibility propagation using the cell-and-portal representation. Each visibility prism issued from a vertex of a portal is propagated through the portals of the adjacent cells. The process is repeated inside each cell until all cells are explored.

The successive steps of this intersection process are illustrated in Figures 3.6 and 3.7. We finally categorize all leaves of $\mathcal{D}^j$ by collecting (i) the visibility information associated with each subject k by using a binary tree search in $\mathcal{V}_k^j$ and (ii) the semantic information on stereotypical viewpoints by using a binary tree search in $\mathcal{S}$.

This process provides a good basis to reason on camera Director Volumes to either perform a cut, move in the environment, or maintain the camera position. In the following we will present our editing method, providing the camera paths and cuts given as input to the dynamic resolution of non-occluded camera positions.

4 Reasoning over Director Volumes

Once the space of viewpoints has been characterized in terms of Director Volumes, the next step consists in selecting the best candidate volume(s) following continuity rules and directorial style.

To enforce elements of continuity and style in an expressive way, we propose to model rules as a two step generic process that operates over a set of Director Volumes. Our editing model considers a set of Director Volumes as input. We then sequentially apply a *filtering operator* over Director Volumes, then a *selection operator* over remaining volumes. The filtering operator is responsible for (i) pruning inappropriate viewpoints in Director Volumes and (ii) annotating remaining Director Volumes with information that will be used by the selection operator. The selection operator is then responsible for evaluating and ranking remaining Director Volumes (through a fit-

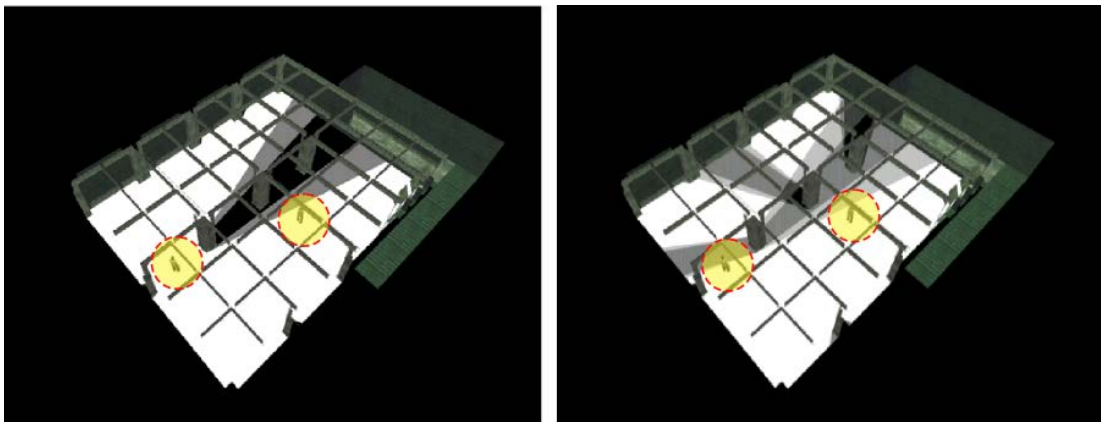


Figure 3.6 – Combination of visibility information for a two-subject configuration (subjects are displays with a yellow area). The left-most image displays the Visibility Volumes of the left-most subject. The right-most image displays the combination of both visibility information. This combination consists in building a new set of Visibility Volumes, informed with both visibility information, and is operated through an intersection of visibility a-BSPs of both subjects. In this image, black areas represents regions where both subjects are fully occluded while white areas represent regions where both subjects are fully visible.

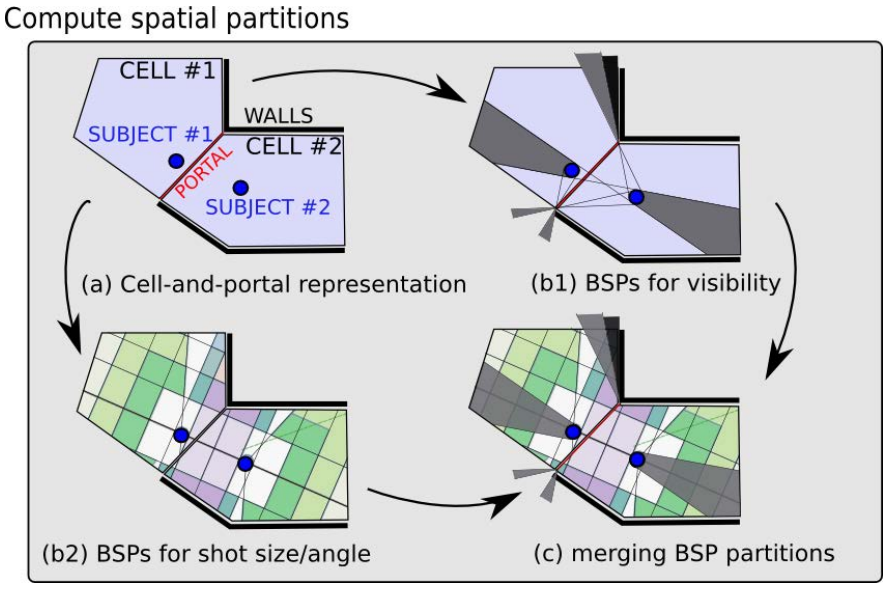


Figure 3.7 – Computation of Director Volumes for a two-subject configuration. From a topological analysis of the scene (top-left image), we compute two a-BSP representations around subjects: a representation of full/partial visibility regions (top-right image, see Figure 3.6), and a representation of semantic regions (bottom-left image, see Figure 3.3). We then combine both representations to build Director Volumes around subjects, by simply intersecting the visibility a-BSP with the semantic a-BSP.

ness function accounting for annotations), and returning a sub-set of Director Volumes (which maximize the fitness value). The process is detailed in Figure 3.8.

4.1 Filtering operator

The first reasoning step consists in applying the filtering operator, which is built as a sequential application of filters over Director Volumes. A *filter* is defined in a generic manner, as an operator that takes as input a set of (annotated) Director Volumes, and returns a – possibly empty – new set of (annotated) Director Volumes. Each filter implements either a continuity rule or a style rule. Filters can therefore be defined recursively, *i.e.* a filter can be constructed as a sequential or parallel application of sub-filters.

Filters can be categorized into three classes in accordance to their reasoning level over Director Volumes: the geometric filters, the semantic filters and the annotation filters.

Geometric filters reason on the geometry of Director Volumes. They can split Director Volumes to prune inappropriate geometric areas from these Director Volumes (*e.g.* remove all viewpoints that subtend an angle lower than 30 degrees to a given subject from Director Volumes).

Semantic filters reason on the semantic aspects of Director Volumes. They either accept or prune entire Director Volumes according to constraints established at a semantic level (*e.g.* remove all Director Volumes which are Close Shots).

Annotation filters can reason on the specific properties of each Director Volume. They can add, remove or edit the annotations associated with each Director Volume. A typical example is the fitness property. Each Director Volume has a fitness (*e.g.* representing a degree of preference of this Director Volume) and rules can add values (annotations) to the fitness property.

Our filtering operator is defined as the combination of two filtering processes. First, continuity editing filters remove inappropriate viewpoints in Director Volumes that do not match editing continuity rules (*e.g.* crossing the line-of-interest or crossing the line-of-action). Second, directorial style filters prune inappropriate viewpoints *w.r.t.* the current directorial style, then annotate Director Volumes according to style preferences. Style preferences model for example the preferred set of viewpoints from which to shoot a narrative event or a scene.

In this approach, we pose as the hypothesis that continuity in editing prevails over style. Indeed, we believe that violation in continuity is far more noticeable for the viewer than violation in style. For this purpose, as filters are applied in sequence, we first process the Director Volumes with continuity filters before style filters.

4.2 Continuity editing filters

Continuity ensures compliance with cinematographic conventions and enforces the maintenance of the spatio-temporal context when the camera switches between two viewpoints. Conventions are well established in relation to the key subjects' motions,

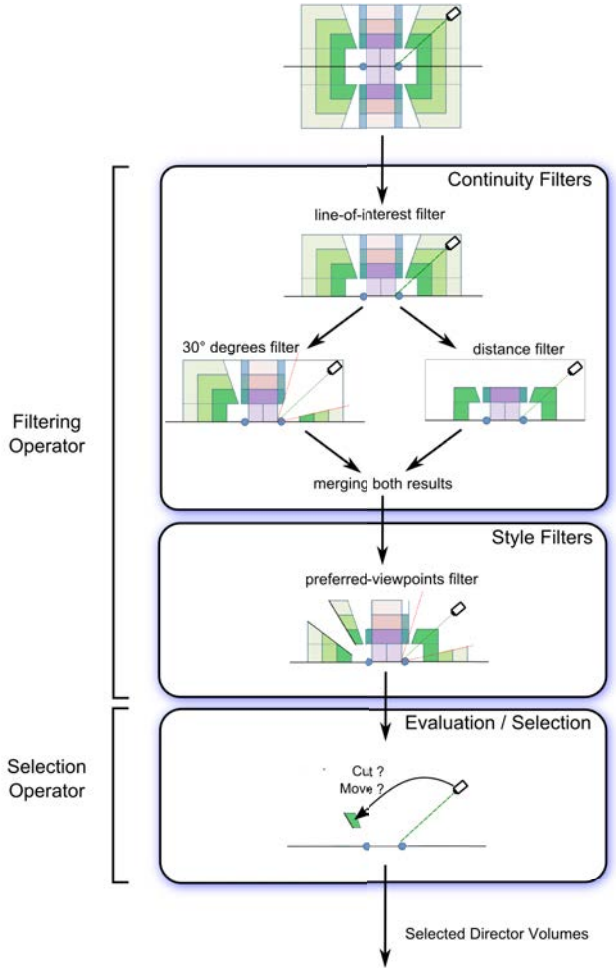


Figure 3.8 – Reasoning over Director Volumes. The selection of a sub-set of Director Volume is a two-step process. First, we apply a filtering operator on the set of Director Volumes. At this step, we sequentially apply (1) continuity editing filters, and (2) directorial style filters. We then apply a selection operator on remaining Director Volumes, to evaluate (*w.r.t.* a fitness function), rank, then select the sub-set of Director Volumes with best fitness value(s).

spatial locations, line-of-interest and actions. We describe some continuity rules and the way to model them as filters in our editing model:

Line-of-action continuity. Between two shots that portray the same key subject, coherency must be maintained in the apparent direction of motion of this key subject. A line-of-action (LOA) is modeled as a geometric filter. All the viewpoints located on the opposite side *w.r.t.* the direction of motion of the subject are removed from Director Volumes. For this purpose, we inject the line of action of the subject into the a-BSP representation (*i.e.* along the direction of motion of the subject) and we compute the intersection of the resulting half plane with the current set of Director Volumes.

Line-of-interest continuity. In narrative events that involve two or more key subjects, once the side of the line-of-interest (LOI) has been chosen, it should not be crossed, unless using an Extreme Long Shot (that re-establishes the key subjects in relation to the environment) or a continuous transition. The relative side *w.r.t.* the LOI is directly encoded in the Semantic Volumes structure, which is recomputed at each frame. The line-of-intersect is modeled as a semantic filter; all Director Volumes on the opposite side of the LOI, but Extreme Long Shots, are then discarded.

Matching between lines of interest. This LOI is, for a character, represented by his gaze direction. If a key subject (character) watches an item which does not appear in the frame, the corresponding viewpoints should convey the fact that his eyes are directed toward the place where user thinks the item is. Therefore, all the Director Volumes for which the direction of the characters gaze is not visible need to be removed. This is implemented by a geometric filter; in the a-BSP, we insert a new couple of half-lines representing a 60° wedge around the character's gaze. All viewpoints out of the wedge are then discarded.

Change in angle or size. Between two shots that portray the same key subject, there should be at least a thirty-degree difference in orientation, or a notable difference in size of portrayed subjects. This rule is implemented by applying to sub-filters in parallel. First, a geometric filter that, in each Director Volume, removes all viewpoints that subtend an angle lower than 30° to the subject; to this end, we inject two new half-lines in the a-BSP. Second, a semantic filter that prunes Director Volumes which are less than two units different in size (in a graduation that ranges in Close-Up, Close Shot, Medium Shot, Long Shot and Extreme Long Shot). Last, we build the union of the two resulting sets of Director Volumes.

4.3 Style filters

Style in Director Volumes is defined as a number of preferences as to which viewpoints should be considered first for the next shot. Style is encoded as a secondary filtering process that ranks the Director Volumes according to different style parameters. In our approach, we consider three style parameters, modeled as annotation filters:

Preferred viewpoints. Moviemakers generally shoot a scene with a restricted set of viewpoints (among all the possible ones). Obviously this sub-set is different from one scene to another and such a variation underlines the narrative discourse out through the whole movie. We offer the possibility to indicate the viewpoints that should be used in priority, together with the viewpoints that should be avoided. A scalar value ranging from -1 to +1 is attached to each Director Volume, indicating its degree of preference (+1 to give preference, and -1 to avoid). In case of no preference, all Director Volumes have the same value for this parameter. Note that we do not account for preferred transitions between viewpoints, but that such a preference would be easy to implement with our model. Indeed, to get closer from

cinematic idioms, another possibility would be to use transition matrices (*i.e.* a specification of the scalar value for each viewpoint, depending on the previous viewpoint).

Variation in the choice of successive viewpoints. Another style parameter to consider when selecting the next viewpoint, is how different it should be from the previous ones, *i.e.* what is the degree of variation in the choice of the successive viewpoints. Some styles impose a limited variation (*e.g.* in dialogue scenes of the Exotica movie, the camera switches back and forth between parallel shots only), while others provide an important variation (*e.g.* in dialogue scenes of the 1984 movie from Michael Radford). For this purpose, we offer the possibility of ranking the Director Volumes against their use in the previous shots. That is, when variation is required, the selection process will better rank the viewpoints which have not been employed recently. In case of no variation, all volumes are annotated with the same value and this parameter will not influence the final selection.

Narrative dimensions. One can augment the Director Volumes with some symbolic information representing narrative dimension (namely *dominance*, *affinity*, *isolation*). For instance, in case dominance of one character over an other needs to be conveyed in a dialogue, this filter selects a set of Director Volumes to be updated with the "Dominance" annotation. Typically, External Shots better enforce the impression of dominance (compared to Apex Shots). Consequently, in the selection process, whenever dominance style is required, the Director Volumes annotated with "Dominance" will be better ranked.

4.4 Selection operator

The second step is the use of a selection operator over remaining Director Volumes. From the set of Director Volumes which have been filtered for continuity in edits and style, this operator simply evaluate and rank volumes, then select the best N volumes ($N \geq 1$) of the set.

To evaluate a Director Volume v, we account for a range of features of the Director Volume: visibility, preferred viewpoint, variation in style, narrative dimension, and composition. To each feature i, we associate an objective function $f_i(v)$ to maximize. We then rank Director Volumes through a fitness function F expressed as a weighted sum over all objective functions:

$$F(v) = \sum_i w_i \cdot f_i(v)$$

where w_i is the (positive) weight associated with feature i.

First, objective functions related to preferred viewpoints, variations in style and narrative dimensions are built upon annotations added by style filters. Second, we define the objective function $f_{vis}(v)$ related to visibility (*w.r.t.* static scene elements) as the aggregation of the visibility degree of each key subject, which we can express as:

$$f_{vis}(v) = \frac{1}{k} \sum_k \sqrt{1 - \delta\left(vis_k^{actual}(v), vis_k^{desired}\right)}$$

In this formula, $vis_k^{actual}(v)$ represents the actual visibility degree of key subject k from volume v and $vis_k^{desired}$ represents its desired visibility degree (as an interval). The function δ represents a distance between them, returning a value ranging from 0 (the actual visibility degree belong to the desired interval) to 1 (the maximum distance value). The use of a square root in f_{vis} will therefore favor balancing the distance to the desired visibility degree for both key subjects. Third, the objective function related to composition is defined as follows. We here assume that we have computed the composition error ϵ_k of each key subject k for a single camera configuration in the volume v – the computation of this error is described later (see Section 5). This objective function is then expressed as

$$f_{comp}(v) = \frac{1}{k} \sum_k \sqrt{1 - \epsilon_k(v)}$$

where the error ϵ_k ranges from 0 (no error) to 1 (maximum error). Note that, similarly to visibility, the use of a square root will balance the error in composition for each key subject.

We finally select, among all available, the N Director Volumes of highest fitness value(s). Note that in case the number of available volumes is lower or equal to N, we select all those volumes without resorting to the fitness function.

4.5 Failures in available Director Volumes

Failures occur when filtering operators prune all possible Director Volumes (before applying Style filters). We handle such failures in two ways. First, in our design, Extreme Long Shots are not impacted by continuity rules; such viewpoints are generally used to (re-)establish the relation between a key subject and his environment, and filmmakers classically use this device. Second, continuous transitions are applied to violate continuity rules (*e.g.* crossing the line-of-interest) as long as the spatial continuity is maintained.

5 Enforcing Screen Composition

The next step consists in selecting an appropriate camera configuration inside a selected Director Volume and enforcing some composition rules (*e.g.* the rule of thirds and balance in the image). By definition each Semantic Volume encompasses some implicit elements of composition (number of key subjects and relative positions of the subjects on the screen). For example, an Apex Shot should portray two key subjects respectively on the left and on the right of the screen; and a Parallel Shot should portray a single key subject on one half of the frame, looking toward the other half of the frame. Our approach expresses the composition through the specification of the position of key subjects on the screen. When key subjects are characters, we actually abstract characters by using their eyes or head location (depending on the shot size).

To enforce composition, we first associate with each Semantic Volume some constraints on the composition for one, two or more key subjects on the screen. Since

the volumes already encompass a notion of size (Close Shot to Long Shot), we do not model the size of the key subjects on the screen as a constraint. We build an objective function to optimize as follows. For a given camera position inside the Director Volume, we compute the camera orientation algebraically. For each subject k, the composition error ϵ_k is defined as the Euclidean distance between the desired and actual on-screen positions of the subject (the eyes or head for a character). The objective function $f_c(p)$ to minimize is then defined as

$$f_c(p) = \frac{1}{k} \sum_k \epsilon_k(p)$$

where p represents a camera position. The on-screen positioning constraints can therefore be modified to enforce some specific narrative dimensions (*e.g.* dominance of one character over another can be expressed by constraining his eye-line to be higher on screen than the eye-line of the dominated character). The Results section details the implementation of narrative dimensions with our system and displays the impact on composition.

At this step, the handling of dynamic occluders could be included by adding an new component $vis_k^{dyn}(p)$, related to the visibility of each key subject for dynamic occluders, to the objective function. Simple techniques such as ray-casts or depth rendering can be integrated with little overhead.

We then rely on a local search optimization technique to compute the best camera configuration inside a Director Volume. We start with a uniformly distributed set of candidate positions p_i, $i \in \{1, n\}$ inside a Director Volume. For each position p_i, we then use the following algorithm

1. We randomly sample m candidate positions q_i^j around p_i.
2. We discard all samples which lie outside the volume.
3. Among others, the best neighbor, *i.e.* with lowest composition error $f_c(q_i^j)$, is referred to as q_i^{best}.
4. If $f_c(q_i^{best}) < f_c(p_i)$, then we use q_i^{best} as new candidate position and we reiterate the process.
5. If not, no improvement has been made and we stop the process.

We also consider two additional stopping conditions. First, we use a threshold on the number of iterations, ensuring the termination of the algorithm. Second, we use a threshold on the composition error. When $f_c(p_i)$ get under this threshold, the process is also stopped. Finally, among all candidate camera positions p computed in Director Volumes, we select the one with minimum value of $f_c(p)$.

See Figure 3.9 for an overview of the algorithm, and Figure 3.10 for an illustration of the on-screen positioning optimization. The specification of the framing for this example is further provided hereafter:

```
Viewpoint "OTS-B"
   configuration 2-subject
   type External
   size Medium Close-up
   side B
   Framing
      Subject -0.33 -0.33
      Subject +0.33 +0.33
```

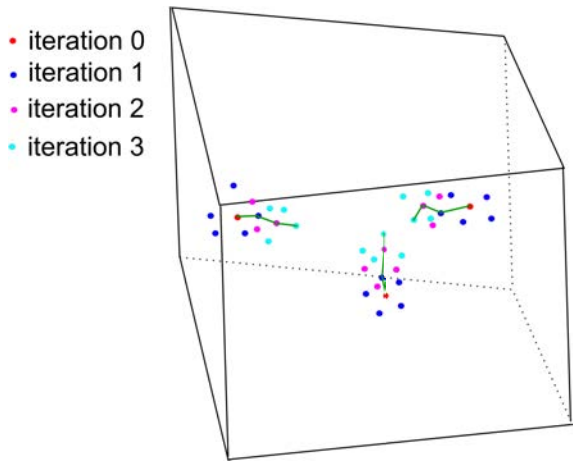


Figure 3.9 – A local search optimization is performed inside a Director Volume to search the viewpoint providing the best composition on the screen. The search process starts from a set of uniformly distributed viewpoints inside the 2.5D volume. Then, for each viewpoint, a number of neighbor candidates are generated and evaluated with regard to composition. The best neighbor is chosen as the current configuration and the process iterates on a new set of neighbors.

6 Performing cuts or continuous transitions

Our system constantly reasons as to whether to

- maintain the current position of the camera,
- perform a cut,
- make a continuous transition to another viewpoint,
- react to a new narrative event that has occurred.

These choices depend on: (i) the current and incoming narrative events (and their likely duration), (ii) the current viewpoint, (iii) the past viewpoint, (iv) continuity and style parameters.

(a) (b)

Figure 3.10 – The evaluation of a camera configuration *w.r.t.* frame composition is expressed as the Euclidian distances between on-screen actual positions (red) and desired positions (green) of characters eyes. The solving process seeks for minimizing these distances using a local search optimization process.

6.1 Controlling cuts with Pacing and Dynamicity

In our approach we introduce two indicators of directorial style that control transitions: *pacing* and *dynamicity*.

6.1.1 Pacing

As described in literature, cuts are motivated either by directorial style or by necessity. When a matter of necessity, transitions may be performed at any moment in the movie. Such transitions are motivated by the occlusion of a key subject, or by a key subject leaving the frame. When a matter of style, transitions are driven by the *pacing*, a specific indicator which represents the rate at which transitions are performed. Our pacing is modeled with three values $[d_{new} : [d_{min}; d_{max}]]$, where d_{new} is the minimum duration before making a transition to convey a (new) more relevant narrative event and $[d_{min}; d_{max}]$ is the interval of duration in which a transition is allowed to still convey the same narrative event. The probability of making a transition within these bounds is driven by a Gaussian (the closer to d_{min} the less probable, the closer to d_{max} the more probable). Transitions will occur either due to pacing, or due to the onset of a narrative event of higher relevance.

6.1.2 Dynamicity

Changes in viewpoints may be realized through continuous transitions (*i.e.* camera paths). Such changes are typically a matter of directorial style, which we model through the *dynamicity* indicator. In our implementation, dynamicity ranges from static shots to panoramics, travellings or free camera motions. Interestingly, continuous transitions

may allow the director to break continuity rules, for example by crossing the line-of-interest or a line-of-action. Thus in cases where no Director Volume is applicable after filtering on continuity, we allow the camera to continuously move to any visible Director Volume.

The possibility our system offers to control the camera dynamicity encompasses both the camera motions performed inside a Director Volume (none, panoramic, travelling) and the transitions between action shots (cuts or camera paths). Our cinematic engine offers four possible levels of dynamicity:

1. No Dynamicity (static shots): No camera motion is allowed inside an action shot, and action shots are always separated by cuts (*i.e.* each shot is made of a single action shot);
2. Panning: Camera motions are restricted to pans inside an action shot;
3. Travelling & Panning: Travelling camera motions are also allowed inside action shots;
4. Full Dynamicity (long take): All motions are allowed, and action shots are separated by continuous transitions instead of cuts (*i.e.* a single shot is built to convey story events).

Examples illustrating some variations in Pacing and Dynamicity are presented in the Results section.

6.2 Performing continuous transitions by path-planning

To enforce continuous transitions between viewpoints, it is necessary to first build a roadmap of the environment and second, from this roadmap, perform path planning requests between the camera current configuration and the desired location in the selected Director Volume.

6.2.1 Building a roadmap of the environment

Connectivity graphs of the scene can be defined at multiple levels:

Topological cell-and-portal graph By construction, the topological cell-and-portal decomposition offers a connectivity graph that avoids obstacles. At this level of connectivity, one could be used to plan coarse paths at the cell level as in [AVF04]. At this level, we could consider a cell-to-cell path length and potentially preferences in cells to go through, or to avoid.

Connectivity between Visibility Volumes Each topological cell contains an a-BSP decomposition into Visibility Volumes. On could then build a connectivity graph by interconnecting all adjacent cells inside a-BSPs and between a-BSPs in different cells. At this level of connectivity, one would be enabled to plan paths inside a topological cell; it would also adds the possibility to reason about the visibility (of one or more subjects) along the path.

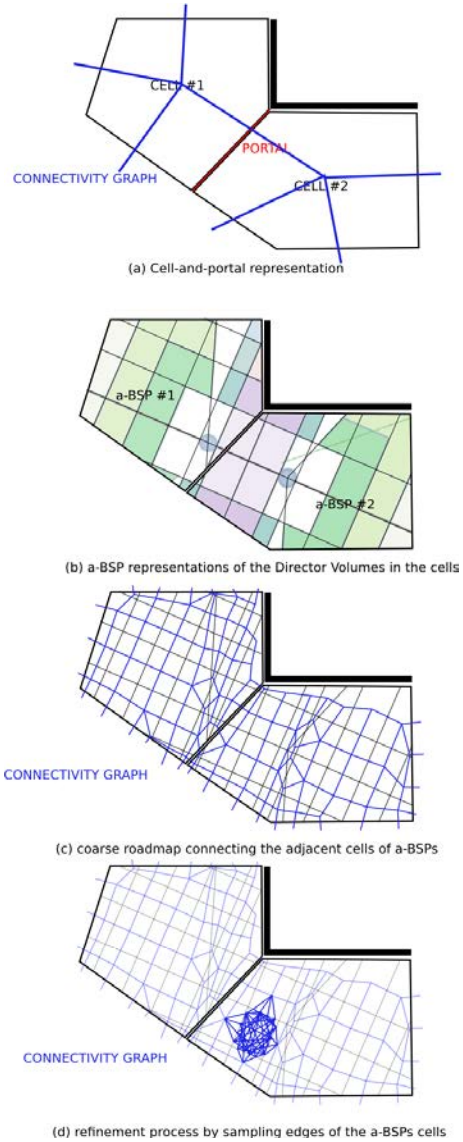
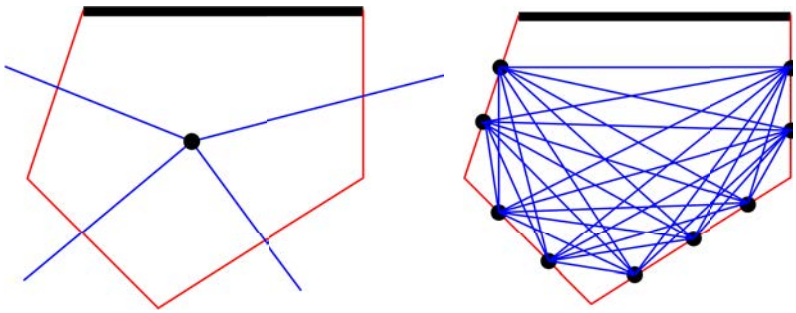


Figure 3.11 – The roadmap is built in multiple steps: (a) a first roadmap is built from the cell-and-portal topological representation, (b) topological cells are split by the a-BSP representation of Director Volumes which in turn (c) forms a coarse roadmap by interconnecting all adjacent partitions inside a-BSPs and between a-BSPs, (d) the roadmap is then refined by sampling the edges shared between the partitions to provide smoother paths in the planning process.

Connectivity between Director Volumes Each topological cell also contains an a-BSP decomposition into Director Volumes. As well as for Visibility Volumes, one could build a connectivity graph upon Director Volumes. At this level of connectivity, one could further reason about preferences in viewpoints to go through or to avoid.

Sampling-based roadmap We here propose a 4th level of connectivity, by construct-



(a) **(b)**

Figure 3.12 – Refinement of the connectivity inside a Director Volume. Walls are here represented as thick black edges. We refine connectivity through the volume by sampling way points (black dots) on accessible edges (in red) with a given density δ. We create a roadmap node to represent each way point. We then create a roadmap edge to represent a link (in blue) between each pair of way point located on two different edges. This sampling-based connectivity enables exploring more accurately the different possible camera paths inside a volume, thus enhancing the quality of the global camera path.

ing a finer roadmap on top of Director Volumes' connectivity. The connectivity graph built for Director Volumes is refined by using a two step-process. First, instead of considering a camera path as just a sequence of Director Volumes, we propose to further explore the different possible ways to go through a volume. We then uniformly sample way-points on each connection edge, with a sampling density δ; we further set the minimum number of samples on a connection edge to 1, so as to guarantee the connectivity between two adjacent volumes. Each sample way-point is then used as a node in the roadmap. Second, inside a given Director Volume, we create roadmap edges between all pairs of way-point nodes located on distinct connection edges. This sampling process has the advantage of providing a finer sampling of possible paths inside volumes, thus allowing to reason more accurately on the global path properties.

The different levels of connectivity are illustrated in Figure 3.11, and a focus on our sampling process inside volumes is illustrated in Figure 3.12. The sampling-based roadmap is therefore recomputed on demand each time the scene evolves (*i.e.* when a key subject moves or new narrative events occur which demand the re-computation of the Director Volumes).

In contrast to existing sampling-based roadmap methods, our roadmap has two main advantages. First, our roadmap is augmented with both semantic and visibility information, what enables reasoning about a range of properties along computed camera paths (*e.g.* minimize the path length, keep the camera at a certain distance from a key subject, enforce visibility along the path, avoid/prefer sub-sets of scene areas or viewpoints). Second, our roadmap is computed dynamically (the roadmap is updated as the Director Volumes evolve), what enables recomputing paths dynamically to account for changes in the key subjects' configuration (*e.g.* in case of subjects motions in the

scene).

6.2.2 Computing Camera Paths

For a matter of simplicity, practically, we plan paths over the complete sampling-based roadmap built on top of Director Volumes. Theoretically, however, it would be possible to design a more efficient path planning process taking advantage of the multi-level connectivity, and using an incremental planning algorithm from low resolution to high resolution connectivity graphs. This would particularly avoid building the complete roadmap; only the traversed cells/volumes would then need to be considered for sampling.

Our roadmap can be formalized as a vector $\langle N, E, A_N, A_E \rangle$, where N the set of nodes, E is the set of edges, A_N is the set of node annotations (*e.g.* position, distance to subjects), and A_E is the set of edge annotations (*e.g.* length, traversed Visibility Volume(s) and Semantic Volume). A camera path t can then be described in two ways:

- as a sequence of edges: $t = \{e_1, e_2, \ldots, e_m\}$, with $e_i \in E$;
- as a sequence of nodes: $t = \{n_0, n_1, n_2, \ldots, n_m\}$, where n_0 and n_m are resp. the initial and target camera configurations, and $n_i \in N, 0 < i < n$ is the node shared by edges e_i and e_{i+1}.

The task of planning a path between two Director Volumes is then expressed as a classical optimization process. The cost C (to minimize) of the path t is given by

$$C(t) = \sum_{e \in t} \sum_{i} w_i \cdot C_i(e) + \sum_{n \in t} \sum_{j} w_j \cdot C_j(n)$$

C_i and w_i are resp. the cost function (to minimize) and weight associated with edge feature i (*e.g.* path length, occlusions of key subjects); similarly, C_j and w_j are resp. the cost function (to minimize) and weight associated with node feature j.

In our implementation, we have considered two important features: the visibility of key subjects along the path and the length of the path. We have thus built our cost function C as a weighted sum of the path length and visibility of key subjects:

$$C(t) = \sum_{e \,\in\, t} \left[w^{vis} \cdot C^{vis}(e) + w^{len} \cdot C^{len}(e) \right]$$

In this formula, $C^{len}(e)$ is set to $|e| > 0$, the length of edge e. In the same way, $C^{vis}(e)$ is the cost associated with visibility along edge e. By default, we express C^{vis} as a weighted sum of the visibility of each key subject k

$$C^{vis}(s) = \sum_{k} w_k \cdot C_k^{vis}(e)$$

where w_k is the weight associated with the visibility of subject k, and with

$$C_k^{vis}(e) = \sqrt{\delta\left(vis_k^{actual}(e), vis_k^{desired}\right)}$$

Here the function δ is defined in the same way as in Section 4.4. The square root in $C_k^{vis}(e)$, together with the weights w_k allows both balancing the visibility degree of

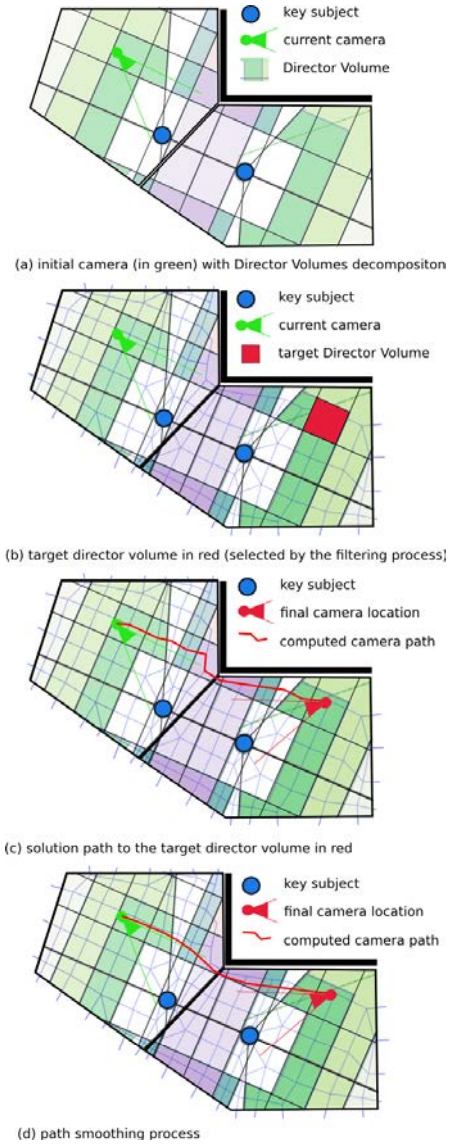


Figure 3.13 – We build upon the roadmap to compute paths between Director Volumes. Since the roadmap is built from the Director Volumes a-BSPs, each cell (and therefore each node of the roadmap holds the information of the Director Volumes (visibility, type of viewpoint). Therefore the search for a path in the roadmap may contain evolved search criteria such as finding paths with maximum visibility of targets, or finding paths which avoid certain shots (*e.g.* long shots).

subjects, and potentially favoring the maintenance of visibility on some subjects to the detriment of others. By tuning the two weights w^{vis} and w^{len}, such a representation allows favoring paths with (for each key subject) little, no or many occlusions on the way.

Our search is performed with a Dijkstra process that handles multiple target volumes and stops as soon as a solution path is found. We then use classical smoothing

techniques to improve the computed path.

7 Results

This section details the application of our virtual cinematography system to the canteen scene from Michael Radford's *1984* movie, by exploring different ways of filming the same set of narrative events occurring in the same environment. The degrees of variation in the directorial style that we explore are related to low-level style parameters such as camera dynamicity, visibility of characters, and pacing. We then demonstrate the editing possibilities of our system with high-level narrative dimensions (dominance, affinity and isolation of characters) that influence the main components of our system.

To generate results, we provided our system with an annotated screenplay. This screenplay specifies a set of narrative events (possibly overlapping in time) as they occur in a sequence of scenes by specifying, for each one, the starting and ending times, the involved subjects/objects (if any), a relevance value, and a textual description of the event. Events may be attached to a character or an object (*e.g.* "Parsons walks to the table"), or describe any general event (*e.g.* "Characters are entering the canteen"). In addition, the screenplay includes the names of particular 3D model parts for the subjects, which are required to perform appropriate screen composition in computing viewpoints (*e.g.* for a character, body part, head part and eyes parts). All 3D models, XML representations of narrative events and cinematic idioms, together with resulting videos are available at www.cameracontrol.org/1984-Canteen-scene.

In the context of the current application, our system however assumes that events are not known in advance; events are instead only considered in the system when they occur. An excerpt of the annotated screenplay for the canteen scene from Michael Radford's *1984* is provided below

```
Screenplay "1984"
  Scene "Canteen"
     Location "Canteen"
        Subject "Smith" Character
           body "SmithBody"
           head "SmithHead"
           leftEye "SmithLeftEye"
           rightEye "SmithRightEye"
        Subject "Syme" Character
        ...
     Event "A pours gin"
        relevance 5
        begin 0 end 4
        Character "Smith"
     Event "A drinks gin"
        relevance 5
        begin 4 end 12
```

```
        Character "Smith"
     Event "A turns"
        relevance 3
        begin 10 end 18
        Character "Smith"
     Event "A speaks to B"
        relevance 9
        begin 18 end 21
        Character "Smith"
        Character "Syme"
      ...
End Screenplay
```

7.1 Pacing

The pacing influences the duration of each shot, *i.e.* the interval of time in which the system is allowed to make a cut or perform a camera movement. We recall that the pacing is defined as an input to the system declared as three values $[d_0 : [d_1; d_2]]$, where d_0 is the minimum duration before making a cut to convey a new event, and $[d_1; d_2]$ is the interval of shot duration in which a cut is allowed to still convey the same event. Here, we display two simple examples of variations in editing by changing the pacing parameter in the 1984 canteen scene. For each of them, three snapshots of the system are taken at three different dates (88s, 90s and 93s).

Example #1: Fast pacing (pacing=$[2 : [4; 6]]$) In the first example, three cuts are performed (see Figure 3.14), respectively after 3 seconds, 2 seconds, and 3 seconds.

Example #2: Slow pacing (pacing=$[4 : [6; 8]]$) In this second example, only one cut is performed (see Figure 3.15). Snapshots were taken at the same time steps as in Example #1 (88s, 90s and 93s). All three snapshots occur in the same shot (no cut), whereas in the same case in Example #1, three cuts were performed.

7.2 Degree of Visibility

Another style parameter that we illustrate here is the degree of visibility that is desired for a key subject and how it influences the editing. Our system considers as input an interval of allowed visibility of key subjects in a continuous range between 0 and 1. Value 0 stands for no visibility, while value 1 stands for full visibility. This interval influences the editing process in (1) the choice of the viewpoints (the system needs to select viewpoints which ensure that the degree of visibility can be maintained), and (2) the critical moments to perform cuts (when the degree of visibility cannot be maintained any more). Here we display three results on the same narrative event "Syme turns around pillars", with three different ranges for the desired degree of visibility: full visibility (interval $[1; 1]$), full occlusion (interval $[0; 0]$) and partial visibility of Syme (interval $[0.25; 0.75]$).

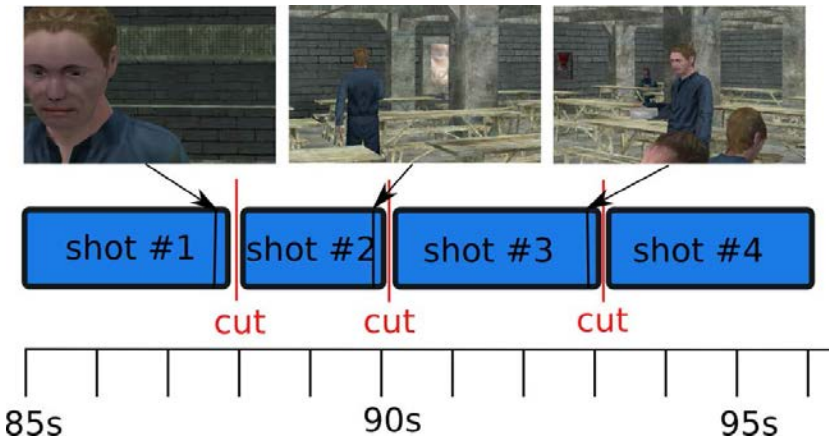


Figure 3.14 – Example #1: fast pacing. Pacing parameters are set to $[2 : [4; 6]]$. Therefore cuts to convey a new narrative event may be performed after 2 seconds of the same shot, and cuts to convey the same narrative event should be performed within the interval of 4 to 6 seconds. Snapshots 1, 2 and 3 were taken respectively at time steps 88, 90 and 93 seconds. Here three cuts of the same narrative event (Parson walking to the table) were performed.

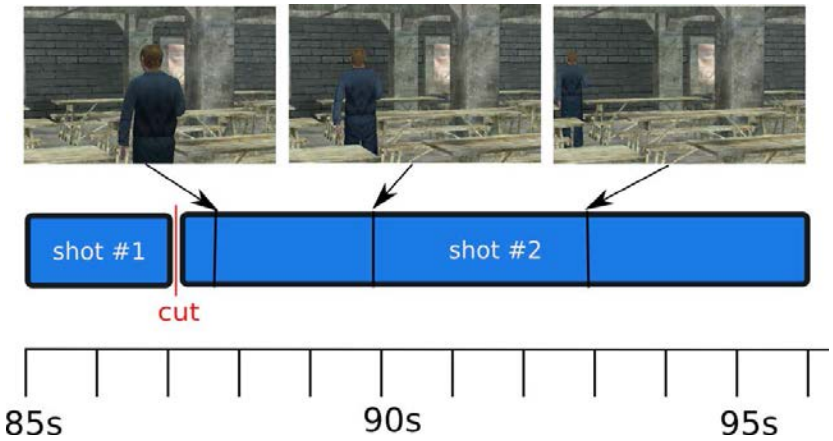


Figure 3.15 – Example #2: slow pacing. Pacing parameters are set to $[4 : [6; 8]]$. Therefore cuts to convey a new narrative event may be performed after 4 seconds of the same shot, and cuts to convey the same narrative event should be performed within the interval of 6 to 8 seconds. As a result, there are only two shots. Snapshots were taken respectively at time steps 88, 90 and 93 seconds (compare to Example #1, where three cuts were performed during this same interval).

Example #3: Full Visibility In this example, the degree of visibility is constrained to interval $[1; 1]$ (maintain full visibility of the subject). As displayed in Figure 3.16, as soon as the character is occluded by a pillar, the system performs a cut to an unoccluded view by selecting a Director Volume where visibility is equal to 1.

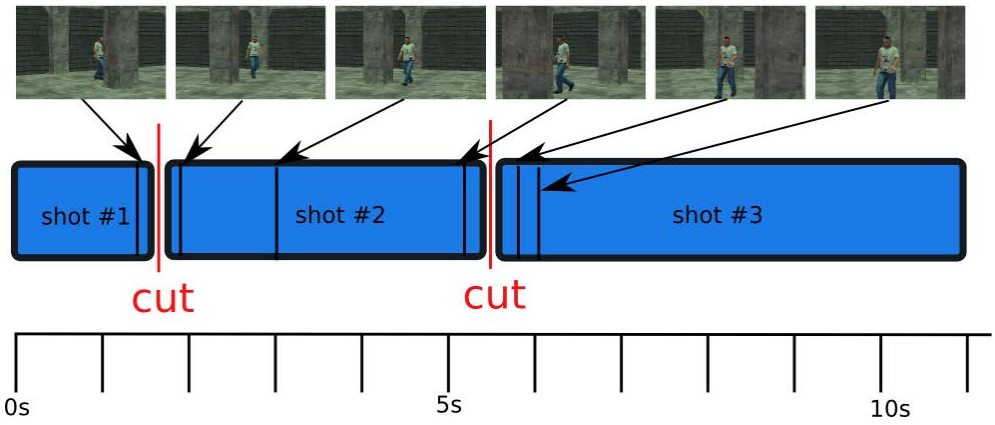


Figure 3.16 – Example #3: enforcing a full visibility. In our editing model, as soon as the visibility constraint is violated, the system performs a cut. In this example, the visibility is set to full visibility (interval $[1; 1]$). Therefore as soon as the character is occluded by a pillar, the system automatically cuts to a new viewpoint.

Example #4: Full Occlusion In this example, we constraint the visibility to interval $[0; 0]$ which stands for no visibility. As displayed in Figure 3.17, as soon as a part of the character starts to be visible, the system performs a cut to an occluded view by selecting a Director Volume where visibility is equal to zero.

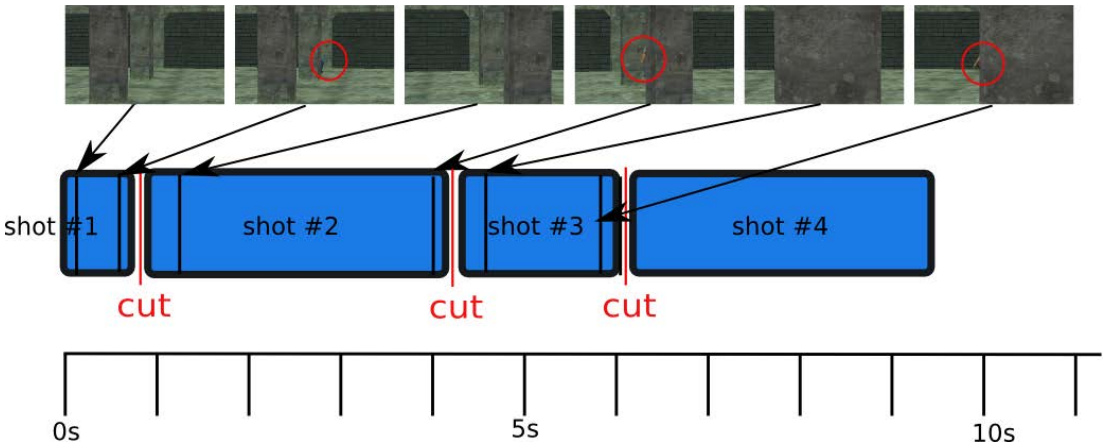


Figure 3.17 – Example #4: enforcing a full occlusion. In this example, the visibility is set to no visibility (interval $[0; 0]$). Therefore as soon as a part of the character is visible, the system automatically cuts to a new viewpoint where the character is occluded. On shots 2, 4 and 6 parts of the character which start to be visible are highlighted with a red circle, just before the system cuts to another viewpoint (time 0.9s, 4.3s and 6.1s).

Example #5: Partial Visibility In this example, the degree of visibility is constrained to interval $[0.25; 0.75]$, which stands for partial visibility of the key subject. As displayed in Figure 3.18, as soon as the character is either strongly visible or strongly

occluded, the system performs a cut to a more appropriate view by selecting adequate Director Volumes with visibility inside the specified range.

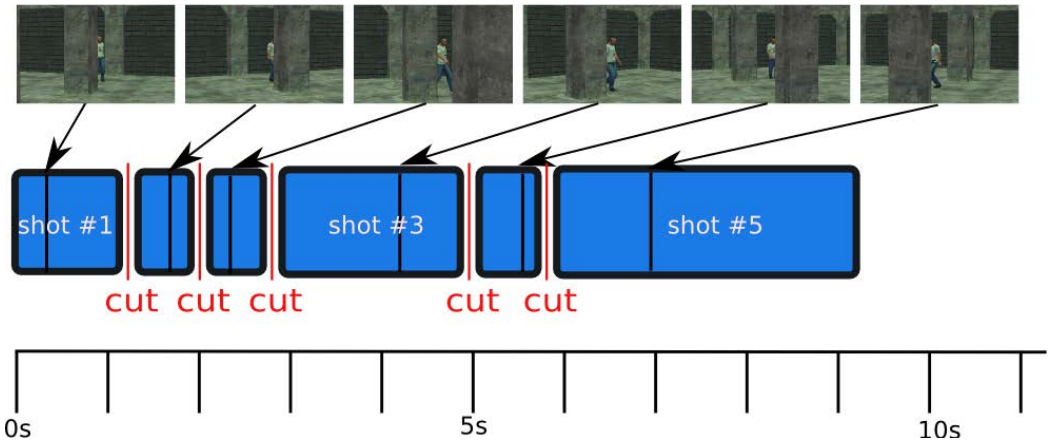


Figure 3.18 – Example #5: enforcing a partial visibility. In this example, the visibility is set to partial visibility (interval $[0.25; 0.75]$). Therefore as soon as the character is either strongly occluded (visibility < 0.25), or strongly visible (visibility > 0.75), the system automatically cuts to a new viewpoint where the partial visibility is satisfied. See how the different shots display the character as only partially visible.

7.3 Camera Dynamicity

In this result section, we describe how the camera dynamicity parameter impacts the editing and the generation of camera movements computed by the system. The camera dynamicity influences our system in (1) allowing specific camera movements (panning, travelling, etc.) and (2) the choice of the next viewpoint when performing a cut, due to changes in view angle and/or character size between the beginning and the end of a shot. Four level of dynamicity are proposed in our system:

- No dynamicity: only static camera positions are computed;
- Dynamicity on orientation: panning motions are allowed for the camera;
- Dynamicity on orientation and position: travelling and panning motions are allowed (together with compositions of travellings and pannings);
- Full dynamicity: all camera motions are allowed. Furthermore all the transitions between shots need to be continuous transitions instead of cuts (this generates a long-take shot).

In the system, camera movements on position (*i.e.* travellings) are enabled by using the roadmap to plan a linear motion of the camera. A continuous transition consists in planning a camera path from the starting Director Volume to the Director Volume chosen as candidate. A panning consists in re-orientating the camera to maintain the satisfaction of the frame composition.

Example #6: No Dynamicity This level of dynamicity disallows any camera movement. The editing is then only made of cuts between static viewpoints. Fig-

ure 3.19 illustrates some results.

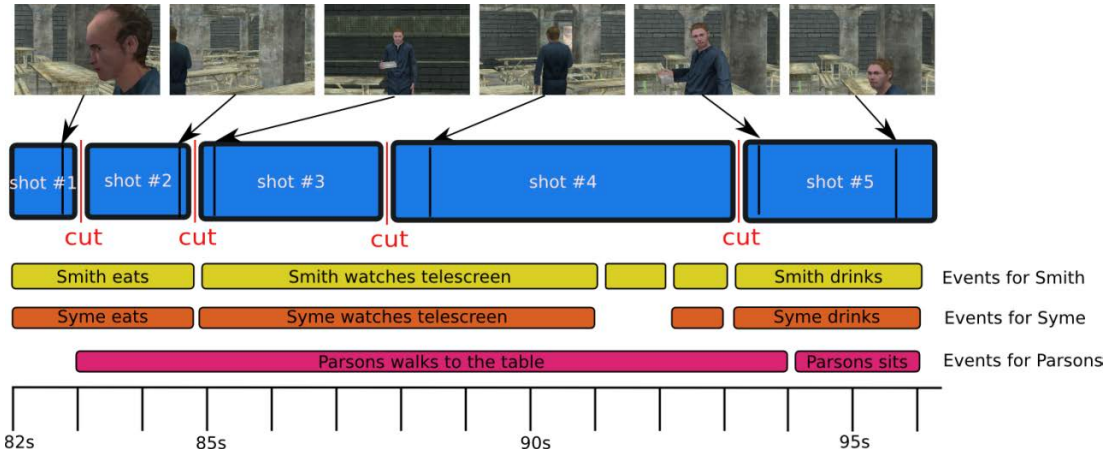


Figure 3.19 – Example #6: no dynamicity. In this example, the dynamicity of the camera is set to "no dynamicity", which means that only static shots are computed. The editing process therefore cuts according to the pacing constraints, the visibility of characters and the occurrence of new events. For example, the cut from shot #1 to shot #2 is due to the new event "Parsons walks to the table", the cut from shot #2 to shot #3 is due to the visibility of Parsons (he leaves the screen, see second snapshot), the cuts from shots #3 to #4 is due to visibility once again, while the last cut (#4 to #5) is due to the pacing parameter.

Example #7: Dynamicity on Orientation This level of dynamicity enables the camera to perform panning shots. The editing process then selects either static shots or panning shots depending on the nature of events occurring in the scene. In our system, events are tagged by the user with camera dynamicity features; *i.e.* for each event, the user specifies whether it may be shot with a static shot, or a dynamic shot (panning, travelling, full dynamic). This represents a matter of style, and different directorial styles will obviously tag the events in a different way. In our Example #7, the event "Symes eats" is tagged with "static shot", while the event "Parsons walks to the table" is tagged with "dynamic shot". Figure 3.20 details some results: shot #1 is a static shot, shot #2 is a panning shot and a cut is performed at time=90s due to pacing constraints. Shots #3 and #4 are panning shots too.

Example #8: Dynamicity on Orientation and Position This level of dynamicity enables the camera to perform both panning and travelling motions. In a way similar to Dynamicity on Position, our system uses the tags associated with the events to know whether a dynamic motion of the camera is possible. Travelling motions are computed by relying on the underlying roadmap (*i.e.* searching for close-to-linear paths). Snapshots have been taken at the same time steps than Example #7, which provides a nice basis for comparison: snapshots 1 and 2 are quite similar, while snapshots 3, 4, 5 and 6 display some variation since a travelling motion is used. Cuts 2 and 3 are performed due to pacing constraints only (see Figure 3.21).

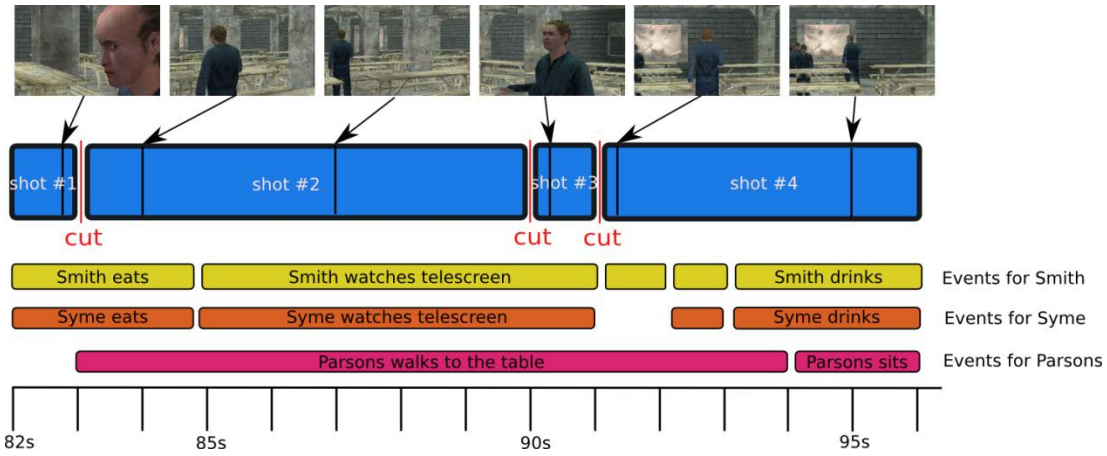


Figure 3.20 – Example #7: dynamicity on orientation. In this example, the dynamicity of the camera is set to "orientation", which means that the camera can perform static shots or panning shots. The editing process therefore cuts according to the pacing constraints, the visibility of characters and the occurrence of new events. Shot #1 is a static shot, while all others are panning shots. The second and third cuts occur due to pacing constraints.

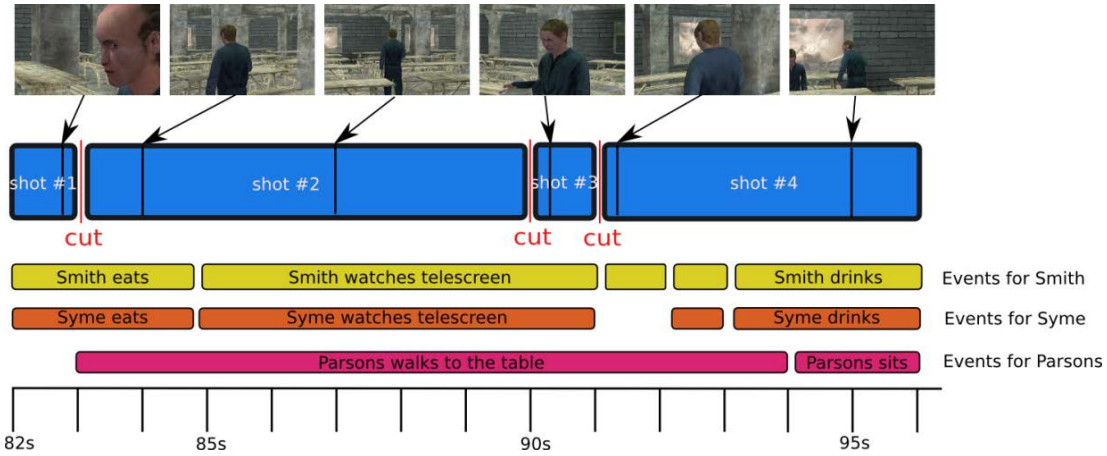


Figure 3.21 – Example #8: dynamicity on orientation and position. In this example, the dynamicity of the camera is set to "orientation and position", which means that the camera can perform static shots, panning shots, travelling shots or a mix between panning and travelling. The editing process therefore cuts according to the pacing constraints, the visibility of characters and the occurrence of new events. Shot #1 is a static shot, while all others are mixing panning and travelling shots. The second and third cuts occur due to pacing constraints. All snapshots are taken at the same time steps than Example #7, which provides a basis to compare the results.

Example #9: Full Dynamicity (Orientation, Position, Continuous Transition) This level of dynamicity enables full dynamicity; *i.e.* the camera moves freely in the environment, and no cuts are performed between the viewpoints. Such a parameter generates long-takes. All transitions between viewpoints are continuous transitions us-

ing the path planning process from the roadmap. When a new event occurs (*e.g.* here Parsons walking to the table), the camera performs a long movement and rotates to grab a view of Parsons, then the camera follows him continuously until he reaches the table (see Figure 3.22).

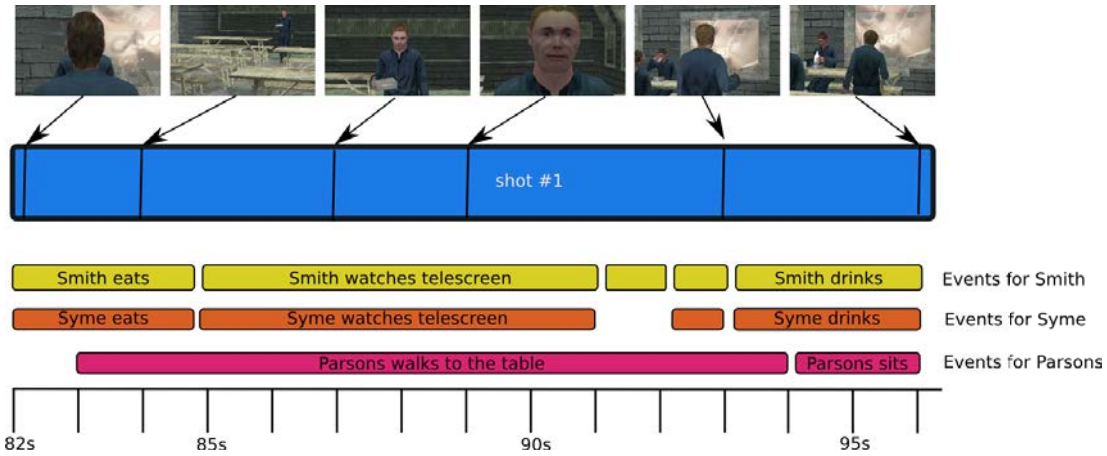


Figure 3.22 – Example #9: full dynamicity. In this example, the dynamicity of the camera is set to "full dynamicity", which means that the camera can perform any motion, and performs path planning between viewpoints conveying different events. Only one shot is produced with this system (a long take).

7.4 Narrative Dimension

In this section, we selected three narrative dimensions which could be manipulated to affect a viewer's perception of inter-character relationships in the scene: (1) isolation, (2) dominance of one character over another and (3) characters' affinity for each other. The manipulation of these dimensions substantially changes the dramatic outcomes of a section of the narrative, independently of dialogue and other characters' actions.

We here detail the mechanisms by which we enforce dominance, affinity and isolation within our cinematography system and present our results with the 1984 Canteen scene. The narrative dimension influences our system in (1) specifying preferences on the set of characters that appear on screen and in (2) applying a specific frame composition of this set of characters.

The enforcement of these dimensions produces fairly different results in terms of editing, viewpoint selection and composition. For each dimension, three snapshots of the system were taken at exactly the same time steps (19s, 41s and 50s), and clearly demonstrates the variation (see Figures 3.26 to 3.28).

In order to implement these narrative dimensions, we extended the virtual cinematic engine in four ways:

- narrative dimensions influence the selection of the next narrative events to convey. For example, by selecting a different protagonist in the story, only the narrative events related to this protagonist (and his interactions) will be considered.

- each Semantic Volume is annotated beforehand with the appropriate dimensions it can convey (*e.g.* Apex shots better convey affinity, external shots better convey affinity and dominance, parallel shots better convey isolation). Each Semantic Volume may be annotated by multiple dimensions.
- a specific filter (called DimensionFilter) is added in the style filtering queue. Its role is to remove from the list of Director Volumes, all Volumes (*i.e.* viewpoints) which do not enforce a specified dimension.
- constraints on the screen composition are different from one dimension to an other. For example, in an External shot, the dominant character will have his eyes constrained to a higher location on the screen than the dominated character. Similarly, for isolation, the characters body, head or eyes (depending on the size of the shot) will be constrained on the far left or far right side of the screen.

Figures 3.23 and 3.24 illustrate how these narrative dimensions influence the different components of our cinematography system on an example with multiple narrative events. As displayed in these figures, the narrative dimensions mainly control the style parameters which in turn control the choice of appropriate Director Volumes, editing and screen composition. The narrative dimensions first influence the selection of appropriate narrative events in the story (*e.g.* selecting events related to the main protagonist). To each narrative event is associated a range of viewpoints. In a second step, this range of viewpoints is narrowed down by selecting preferred viewpoints (Director Volumes) associated with each narrative dimension. For example, affinity is better conveyed using apex shots or over-the-shoulder shots, rather than using parallel shots. Third, narrative dimensions influence the selection of the next best Director Volume to cut to, given previous camera locations and preferred shots (Resulting shots in the Figure). Finally, such dimensions influence the composition of key subjects on the screen, inside each Director Volume.

Example #10: Default Narrative Dimension For this narrative dimension, there is no preference as to which characters have to be framed: the DimensionFilter thus eliminates no viewpoint. Moreover, the frame composition is balanced: *i.e.* with regard to the rule of thirds, the eyes of both characters are placed on the top third horizontal line (see Figure 3.25).

Example #11: Affinity between Syme and Smith To convey affinity between both characters, the system will prefer shots with a balanced frame composition on Syme and Smith, and with regard to the rule of thirds, the eyes of both characters are placed on the top horizontal line. Specifically, the system will favor apex shots or external shots through the DimensionFilter which will only keep Apex or External shots (see Figure 3.26).

Example #12: Dominance of Syme To illustrate the dominance of Syme over Smith, the system will optimize the viewpoints where the eyes of Syme can be placed higher than eyes of Smith on screen: with regards to the rule of thirds, the eyes of Syme are placed on the top horizontal line, and the eyes of Smith are placed on the

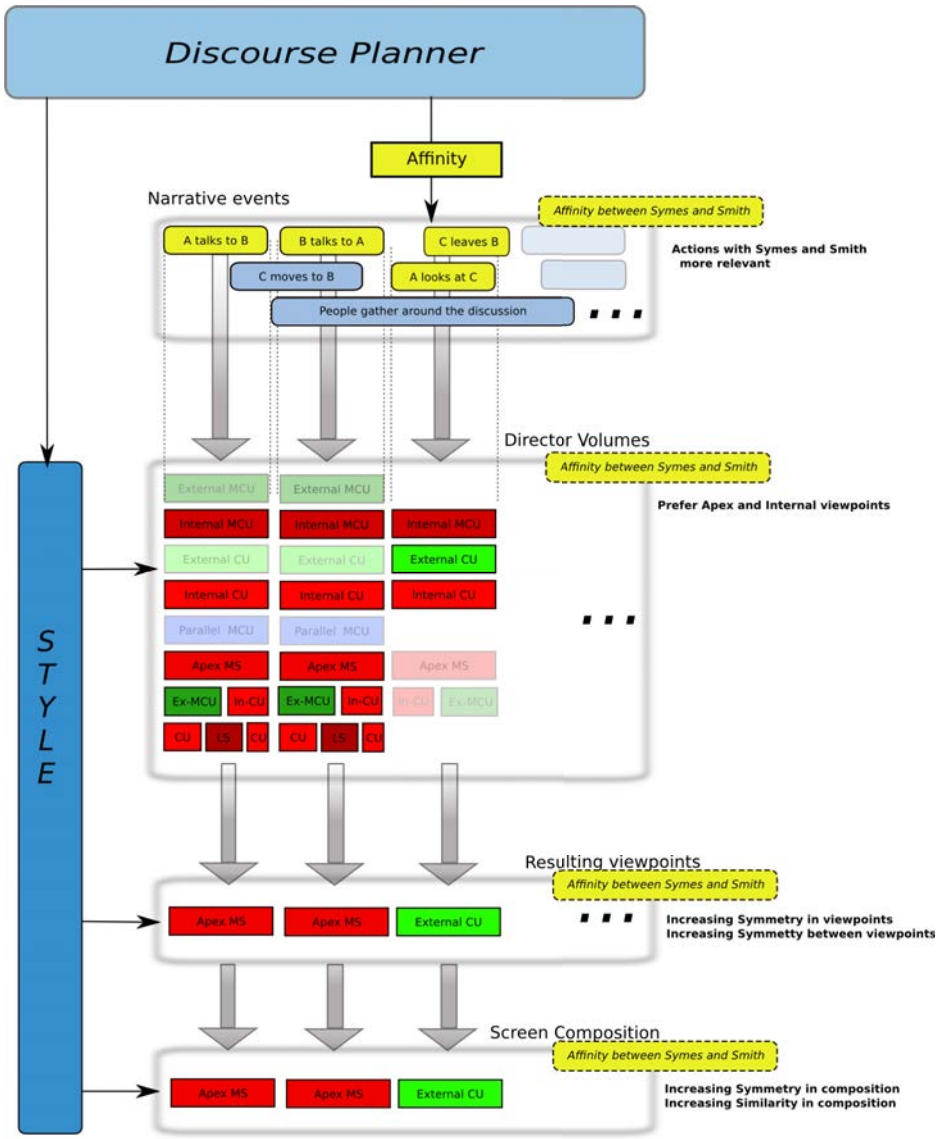


Figure 3.23 – Influence of the "affinity" narrative dimension on main components of CineSys. It influences both the selection of narrative events in the story (*e.g.* selecting events related to the main protagonist), the selection of preferred viewpoints (Director Volumes) and the composition of key subjects on the screen, inside each Director Volume. Affinity is better conveyed using apex shots or over-the-shoulder shots, rather than using parallel shots, and positioning subjects at the same height on the screen.

bottom horizontal line. In terms of shots, the system the system will prefer external shots: the DimensionFilter will only keep external shots (see Figure 3.27).

Example #13: Isolation of Syme To display the isolation of a character, our system favors shots where this character is alone. In this example, all along the dialog between Syme and Smith, the DimensionFilter will eliminate all viewpoints but internal

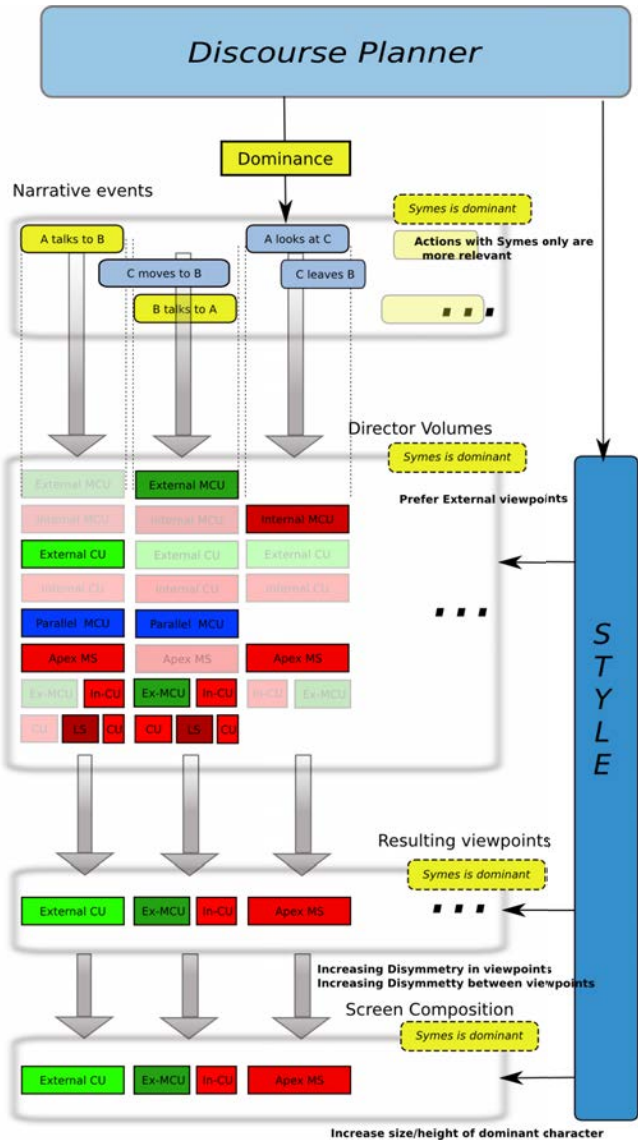


Figure 3.24 – Influence of the "dominance" narrative dimension on main components of CineSys. It influences both the selection of narrative events in the story (*e.g.* selecting events related to the main protagonist), the selection of preferred viewpoints (Director Volumes) and the composition of key subjects on the screen, inside each Director Volume. Dominance is better conveyed using over-the-shoulder shots, and positioning the eyes of the dominant subject higher than the ones of the dominated subject on the screen.

shots on Syme, parallel shots on Syme and subjective shots on Syme. Moreover, the frame composition shows Syme spatially isolated on screen (composition is expressed with the rule of the fifths, and character Syme is constrained to the rightmost vertical line (see Figure 3.28).

Figure 3.25 – Example #10: shots illustrating the "default" narrative dimension (no specific narrative dimension is enforced). All viewpoints are available to the process and compositions are balance according to the rule of the thirds. Shots were respectively taken at time steps 19s, 41s and 50s (see Figures 37, 38 and 40 for a comparison).

Figure 3.26 – Example #11: shots illustrating the "affinity" narrative dimension. Only external shots and apex shots will be selected by the DimensionFilter process. The composition is a balanced composition in both apex and external views. Shots were respectively taken at time steps 19s, 41s and 50s (see Figures 36, 38 and 39 for a comparison).

Figure 3.27 – Example #12: shots illustrating the "dominance" narrative dimension. Only external shots will be selected by the DimensionFilter process. The composition process computes shots where the character Syme (right character) is always higher on the screen than Smith (left character). Shots were respectively taken at time steps 19s, 41s and 50s (see Figures 36, 37 and 39 for a comparison).

7.5 Limitations

A limitation of our approach lies in that the visibility computation for static occluders is performed in 2D. Though the 2D cells are extruded to 2D$\frac{1}{2}$ (by considering the respective ceiling heights) and composition is computed in full 3D, the propagation of visibility between cells fundamentally relies on a 2D process. Therefore, in environments with multiple levels, inter-level visibility will not be computed although this could potentially be addressed by pre-computing inter-cell visibility for cells on different levels using ray casting techniques (in a way similar to [OSTG09]). Additionally visi-

Figure 3.28 – Example #13: Shots illustrating the "isolation" narrative dimension. Only shots with one character on the screen are kept by the DimensionFilter. The composition process computes shots where the character Syme is always on the right most part of the screen. Shots were respectively taken at time steps 19s, 41s and 50s (see Figures 3.25 to 3.27 for a comparison).

bility of key subjects *w.r.t.* dynamic occluders can be handled by including ray-casting or depth rendering techniques in the screen-composition process.

7.6 Discussion and Comparison

To compare our contribution to the state of the art, we selected a number of properties a virtual cinematography system should display in a real-time context.

- Interactive/Dynamic. First of all, such a system needs to be fully interactive, that is, enjoying the possibility to react to interactive events occurring in the environment. Events may be new actions, new characters entering the scene, new dialogues or any narrative events that influences the unfolding of the story. The dynamic aspect requires re-computing camera locations for characters motions which are not known in advance.
- Shot coherency. As empirical rules guide the way shots should be organized in a coherent way to convey a piece of a story, a virtual cinematography system needs to encode such knowledge. The key relies in the expressiveness of approaches to encode such coherency.
- Composition. The composition property represents the capacity of a system to handle screen composition and variation in screen composition. Composition is an essential component of cinematography, not only from the perspective of aesthetics (modeling aesthetics in images represents an impossible challenge), but more basically for understanding spatial relations between characters (a character on the left of the screen and looking right is implicitly interacting with a subject on his right) and causal relations. The range of composition properties and the expressiveness of the underlying computational process is a key property of any virtual cinematography system.
- Style. The style property represents the capacity of a cinematography system to provide a range of directorial styles, or some degree of variation in a style. While characterizing and enforcing style may represent a challenging task, a number of devices and parameters can be used to enforce variation.
- Path-planning. The path-planning property represents the ability of performing complex camera motions in environments. Nonetheless, path-planning is a key-

component to represent simple motions such as travellings and trackings, together with more complex motions such as those encountered with hand-held cameras.

- Cinematic discourse. This last property characterizes the ability of a camera control system to deliver a coherent construction of viewpoints over a set of events, thereby conveying a coherent narration of the events.

Table 3.1 provides a comparison of major scientific contributions in the domain of automated camera control. As displayed in the table, our approach satisfies most of the properties required in an interactive storytelling system and stands in stark contrast. The closest contribution to our work (Elson & Rield [ER07]) is not a reactive approach (the system runs offline) and therefore cannot handle interactivity, does not handle composition on the screen (all shots are pre-computed path sequences) and does not offer path-planning capacities for complex camera motions.

Paper	Technique	Interactive / Dynamic	Coherency	Composition	Style	Path Planning	Visibility	Staging	Cinematic Discourse
He *et al.* [HCS96]	Finite State Machine	Yes	Limited	No	No	No	No	No	No
Christianson *et al.* [CAH+96]	Tree Model	No	Limited	No	No	No	No	No	No
Jhala & Young [JY05]	Tree Model	No	Yes	Pre-computed	No	No	No	Yes	Yes
Elson & Riedl [ER07]	Pre-set Idioms	No	Yes	Pre-computed	Yes	Pre-computed	No	Yes	Limited
Li & Cheng [LC08]	Probabilistic Roadmaps	Yes	Limited	No	No	Yes	No	No	No
Assa *et al.* [AWCO10]	Correlation	Yes	No	No	No	No	Yes	No	No
Yeh *et al.* [YLL12]	Optimization	No	Yes	No	No	Yes	Yes	No	No
Our approach	Director Volumes	Yes	Yes	Yes	Yes	Yes	Yes	No	Limited

Table 3.1 – Comparing our cinematography system to main contributions in the domain. This table compares 7 properties of main cinematography systems proposed in the computer science literature, together with the techniques that we implemented. The "Staging" parameter characterizes the ability of the system to automate the staging of characters in the scene. The "Interactive/Dynamic" parameter represents the ability to react to interactive events occurring in the environment. "Coherency" stands for the capacity to enforce some degree of coherency in a sequence of shots. "Composition" is related to the possibilities the system have in changing the composition in shots. "Style" stands for the ability to perform some variations in the directorial style. Only Elson07 and our approach offer this possibility. "Path-planning" indicates whether the system is able to perform some camera motions inside the environment. Finally "cinematic discourse" represents the possibility of maintaining a cinematic discourse over a number of events. All in all, our approach fulfills most of the properties desired in a virtual cinematography system, and clearly represents a improvement in comparison to previous approaches.

8 Conclusion

In this chapter we have presented a unifying approach to the problem of interactive cinematography. The approach indeed handles viewpoint, editing and planning in a real-time context. We address the innate complexity of well understood problems such as visibility determination and path planning required in real-time camera control, while tackling higher-level issues related to continuity between successive shots in an expressive editing model. Our real-time cinematic engine encodes cinematographic idioms and continuity-editing rules to produce appropriate edits and camera paths from a set of narrative events and style indicators. The model relies on a spatial partitioning, the *Director Volumes*, providing a characterization into visibility regions (with full visibility, partial visibility or full occlusion) and characteristic viewpoints (the Semantic Volumes). We reason on these Director Volumes to identify how, when and where shot transitions should be performed. This semantic and geometric reasoning relies on a filtering-based encoding of cinematic conventions together with the possibility to implement different directorial styles. The expressiveness of our model stands in stark contrast to existing approaches that are either procedural in character, non-interactive or do not account for proper visibility of key subjects.

On one hand, as we stated earlier, not all elements of directorial style can be modeled. Further, the few elements of style we have considered are insufficient to express more complex directorial guidelines and more subtle compositions or edits. On the other hand, we have proposed a working automated model for efficiently tackling key components of film editing (particularly the composition of shots and editing rules, according to a given directorial style). There is a real interest in using this model as a tool to assist a cinematographer in his creative process. In the next chapter, we build upon CineSys to propose an approach of film editing which (i) accounts for the quality of the edit and (ii) provides directors with a novel and efficient workflow.

Integrating Director's Inputs into the Editing Process

4

This chapter presents two approaches to the problem of editing an animated movie that add a measure of the quality of the edit, and provide assistance to a cinematographer in his creative task. In existing approaches, the choice of shots and cuts is led either by (i) the application of a restricted number of deterministic and manually pre-encoded films idioms (one has to declare one idiom for each situation) or (ii) by relying on continuity editing rules. These techniques are restricted to computing cinematically "acceptable" sequences of shots and transitions. Further, it appears tedious for a user to generate a sequence s/he consider satisfying, due to a lack in providing a real control on the final edit. The user has to tune a set of parameters for which the impact they will have in terms of final edit are not obvious. Particularly, the user has no control on the moments of cuts, which is a key element in the art of building a good movie; and has no mean to keep some part(s) of the movie, which s/he finds satisfactory, and make changes on some other part(s) (one has to make changes on the system parameters, then regenerate the entire movie).

In this chapter, we propose the first computational model dedicated to the qualitative evaluation of an edit. We then propose to build upon CineSys, together with this evaluation model, to assist a director in his creative process of constructing a movie s/he finds satisfactory.

Note that Sections 2 and 3 stem from a joint work with Mathieu Chollet and Rémi Ronfard from INRIA Grenoble (France), and that Section 4 stems from a joint work with William Bares from Millsaps College (USA) and Roberto Ranon from the University of Udine (Italy).

1 Contributions

Quality measurement of film edits. We present the first computational model dedicated to evaluating the quality of an edit. In particular, we propose scores for shots, cuts and pacing (rhythm at which cuts are performed), independently of the film idioms used to generate them. The score for a shot is based on the Hitchcock principle of showing action from the best angle [ST85], the score for a cut between shots is based on the working practices of film and television [Tho93, Tho98] and the score for pacing relies on a well-founded model to account for shots duration.

Efficient automated construction of well-edited movies. We introduce an efficient search strategy for finding the best sequence of shots from a large number of candidates generated by traditional film idioms. By building upon our quality metrics, our approach enables separating correct from incorrect shot sequences. In contrast to related work, we account for a precise enforcement of pacing and study in detail the best moment to perform a cut, by precisely examining all possible edit points. We further provide the user with more control on the final edit, through a specification of *key shots* as constraints during the search (*i.e.* shots the user wants to maintain in the sequence).

Novel workflow for film preview. We propose a novel workflow based on the interactive collaboration of human creativity with automated intelligence that enables efficient exploration of a wide range of cinematic possibilities, and rapid production of a film preview. This workflow is the first to combine the creative intelligence and skill of filmmakers with the computational power of an automated cinematic engine.

Learning from examples. The approaches we propose learn editing preferences from examples of human experts. In particular, we introduce a novel modeling of film idioms, through transition matrices, and our work constitutes the first effort toward learning cinematic idioms from live human camera work, instead of trying to pre-encode a limited number of situations.

This chapter is organized as follows. We firstly present our quality measurement of film edits (Section 2). We secondly present the search algorithm we propose to automatically explore the possible sequences of shots and find the best film edit (Section 3). We thirdly present *The Director's Lens* (Section 4), our intelligent and interactive assistant for film preview, that builds upon CineSys. We finally discuss the limitations of our ranking-based approach and conclude on future research.

2 Film grammar rules

Evaluating the score of an entire sequence for a movie is built up from the scores of its shots and transitions. We make the assumption that a cost function can be computed for each fragment and cut in the sequence. We define a fragment as a piece of a shot of duration Δt, that we can express as a couple $\langle i, t \rangle$ where i denotes the shot index, and t the time. The cost per shot fragment is evaluated as a weighted sum of all violations of the rules of frame composition. And similarly, the cost of a cut is evaluated as a weighted sum of all violations of the rules of continuity editing. The cost associated with shot i at time t is then given by $C^S(i,t)$, and the cost associated with a transition from shot i to shot j at time t is given by $C^T(i,j,t)$, expressed as

$$C^S(i,t) = \sum_k w_k^S \cdot C_k^S(i,t) \qquad C^T(i,j,t) = \sum_l w_l^T \cdot C_l^T(i,j,t)$$

where w_k and w_l are weights associated with each rule.

We furthermore consider a third key component of a movie: the pace in transitions. We assume that the cost of an entire sequence of shots (*i.e.* a movie, which we denote as m) *w.r.t.* pace can be computed as a sum of a cost on the duration of each shot:

$$C^P(m) = \sum_{s=0}^{n} C^P_{Dur}(e(s) - b(s))$$

where $b(s)$ and $e(s)$ are resp. the beginning and end time of shot s.

We finally build the cost function C of an entire sequence of shots as the weighted sum of the costs taken on these three key components:

$$C(m) = W^S \cdot \left[\sum_{s=0}^{n} \sum_{t=b(s)}^{e(s)} C^S(s,t)\right] + W^T \cdot \left[\sum_{s=0}^{n-1} C^T(s, s+1, e(s))\right] + W^P \cdot C^P(m)$$

where W^S, W^T and W^P are the weights associated with resp. the scores of shots, the scores of transitions and the score of the pace in transitions.

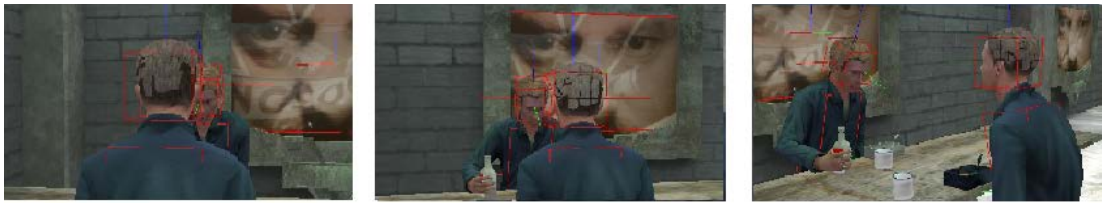
In the next sections, we detail the key features and rules we consider in this evaluation. These features and rules can be categorized into five classes: shot composition, continuity editing rules, relevance of a shot, relevance of a transition, and pace in transitions.

2.1 Shot composition

We evaluate the composition of a shot fragment according to (i) a score on visibility of subjects and (ii) a score on the composition of subjects on the screen.

Before detailing the components of $C^S(i,t)$, we first introduce some useful notations. Firstly, we will use K to represent the set of subjects that are within the frame in a fragment $f = \langle i, t \rangle$. Secondly, we denote the on-screen location of eyes (middle of both eyes) and the projected size of subject k in fragment f resp. as $e_k(f)$ and $s_k(f)$. Finally, we introduce two vectors computed in the 2D screen space of a fragment f: the gaze direction and the motion of a subject k (resp. $g_k(f)$ and $m_k(f)$).

2.1.1 Visibility



$C^S_{Vis} = 0.538$ $C^S_{Vis} = 0.340$ $C^S_{Vis} = 0$

Figure 4.1 – Visibility scores. Left: Poor. Middle: Better. Right: Best.

A good shot should maximize the visibility of the actions performed by the protagonists. To account for this feature, we express a visibility cost C^S_{Vis} which measures the occlusion degree of the projected bounding box of each protagonist

$$C^S_{Vis}(f) = \sum_a \left[\sum_{\substack{i \in prtg(a) \\ j \neq i}} occlusion(box_i, box_j) \right]$$

where $prtg(a)$ are protagonists of action a, box_i represents the projection of the bounding boxes of actor i on the screen in fragment f, and $occlusion(box_1, box_2)$ represent the quantity of box_1 that is occluded by box_2. This occlusion is measured by using the overlap of projections of both bounding boxes. Figure 4.1 displays three shots with increasing visibility scores in a scene with two subjects. Shots with poor visibility scores typically show subjects coming too close together, or hiding each other.

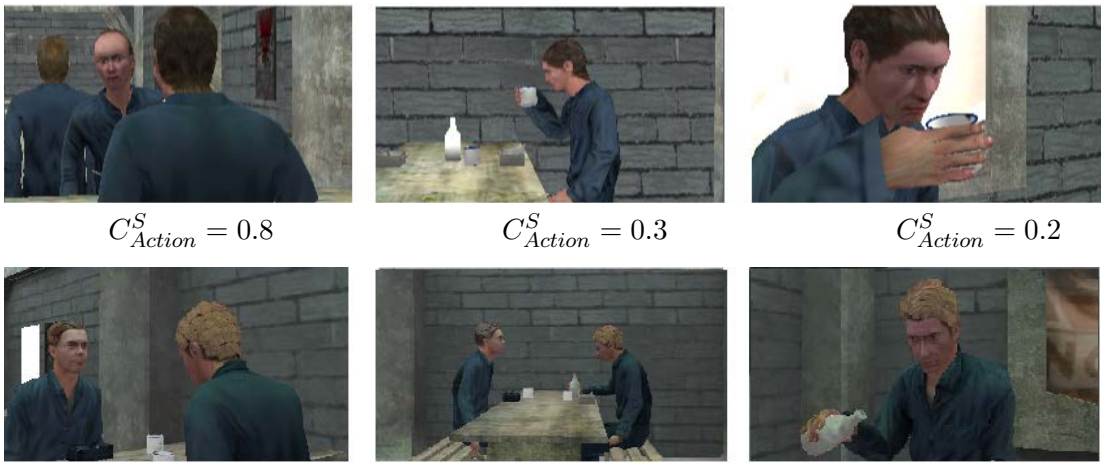
2.1.2 Look room

Following common practice, we favor shot composition where each key subject k is given enough screen space relative to the image frame, especially in the direction in which (s)he is looking. We compute this score as

$$C^S_{Look}(f) = \sum_{k \in K} [\ e_k(f) + \lambda\left(s_k(f)\right)\ \cdot\ g_k(f)\ ]$$

where λ is a scalar function of the subject's on-screen size s_k.

2.2 Relevance of a shot



$C^S_{Action} = 0.8$ $C^S_{Action} = 0.3$ $C^S_{Action} = 0.2$

$C^S_{Action} = 0.7$ $C^S_{Action} = 0.5$ $C^S_{Action} = 0.2$

Figure 4.2 – Action scores. Top: Action Drink. Bottom: Action Pour. Left: Poor. Middle: Better. Left: Best.

Relevance of a viewpoint is measured by exploring its capacity to enhance a viewer's comprehension of the unfolding narrative events. Each event has a relevance value that encodes its importance for the story (*e.g.* representing whether the event is a foreground action, an establishing action, or a background action). Viewpoints that depict more relevant events, from relevant viewpoints enforce the comprehension of the story and will have a higher quality. We choose to evaluate the relevance of a shot with an "Action" term, which measures the amount of the scene events which is missed in a shot fragment $f = \langle i, t \rangle$

$$C^S_{Rel}(f) = C^S_{Action}(f)$$

To this end, we define a cost table $A^S[a, s, p]$ which specifies the effectiveness of a shot size s and a profile angle p conveying an action type a. This effectiveness is expressed as a cost ranging from 0 (most effective) to 1 (least effective). For example, one can prefer framing a subject speaking using a medium close-up from a front view.

The computation of the Action term over fragment f is expressed with a weighted sum over all actions a occurring during this fragment. The weighting is expressed by the importance $imp(n)$ of an event at time t in the scene (the closer to 0, the more important). $a = type(n)$ represents the type of the event, while $size(n, f)$ and $pr(n, f)$ represents resp. the shot size and profile angle of the viewpoint in fragment f *w.r.t.* the main subject of narrative event n

$$C^S_{Action}(f) = \sum_n imp(n) \cdot A^S\left[type(n), size(n, f), pr(n, f)\right]$$

With a suitable choice of the $A^S[a, s, p]$ coefficients, the Action term can serve to enforce the Hitchcock principle, which states that the screen size of subjects should be proportional to their importance in the scene [ST85]. In our implementation, we tabulate the values in $A^S[a, s, p]$ with four action types: facial actions (speak, listen, watch, turn, nod), hand actions (lift, pour, drink, eat), feet actions (walk, sit down, stand up) and no action (idle). In the 3 cases, $A^S[a, s, p]$ stores preferences for shots showing the face, hands and feet with the largest possible screen size. For idle actions, $A^S[\text{'idle'}, s, p]$ stores preferences for shots showing the subjects with the least possible screen size. Because screen size is a limited resource, minimizing C^S_{Action} has the effect of allocating screen space according to the importance of actions. Figure 4.2 illustrates the preferences for shot sizes and profile angles for the special case of two hand actions: pour and drink.

2.3 Shot transitions

A transition between consecutive shots always causes discontinuity. The art of the editor is to choose minimally intrusive transitions by selecting appropriate shots and moments for cutting. A complete theory of what makes a cut intrusive is not currently available, although there has been work in film aesthetics [Ond02, BC11] and cognitive psychology [DV90, Smi05, ZM10] on the subject. For our purpose, we have found useful to compute the cost of a cut as the sum of terms measuring typical grammatical errors according to the classical style of continuity editing (continuities in the screen

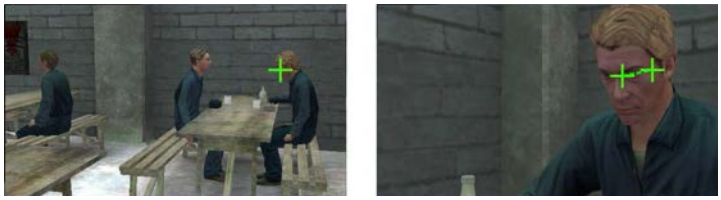
positions, gaze directions and motion directions of subjects). Continuity editing is the dominant style of editing in western cinematography and especially mainstream Hollywood movie-making. The primary goal of continuity editing is to hide the cuts as much as possible by avoiding all causes of screen discontinuities between shots. Such discontinuities result in ungrammatical shot transitions. Ungrammatical cuts can also be obtained when the screen positions, orientations and sizes of actors are so similar that the viewer may be tricked to believe that the subjects are jumping around the screen. Such jump cuts can be prevented by ensuring that the screen sizes and profile angles of subjects change significantly between successive shots.

Before detailing the components of $C^T(i, j, t)$, we introduce some additional notations. First, in the following, we evaluate a cut performed from a fragment $f_1 = \langle i, t \rangle$ to another fragment $f_2 = \langle j, t + \Delta t \rangle$. We then use K_i to represent the set of subjects that are within the frame in fragment f_i. Finally, we use the *Kroneker* symbol and refer to it as $\delta(x, y)$. For the record, the *Kroneker* symbol is defined as

$$\delta(x, y) = \begin{cases} 1, & \text{if } x \neq y \\ 0, & \text{if } x = y \end{cases}$$

2.3.1 Screen continuity

$C^T_{Screen} = 0.596$



$C^T_{Screen} = 0$

Figure 4.3 – Screen continuity scores. Left: cutting from left to right has the leftmost character jumping to the screen center resulting in a poor cut. Right: keeping the center character in the same screen location produces a smooth cut.

The score of screen continuity in transitions prevents subjects who appear in two successive shots to jump around the screen. Because the subject's eyes are the most important center of attention, we favor transitions which maintain the subject's eyes at the same screen location. We weight this term with the screen size of actors, so that continuity in the foreground receives a larger reward than in the background. As

a result, screen continuity is enforced by minimizing

$$C^T_{Screen}(i,j,t) = \sum_{k \in K_1 \cap K_2} s_k^{avg} \cdot \phi(\|e_k(f_1) - e_k(f_2)\|)$$

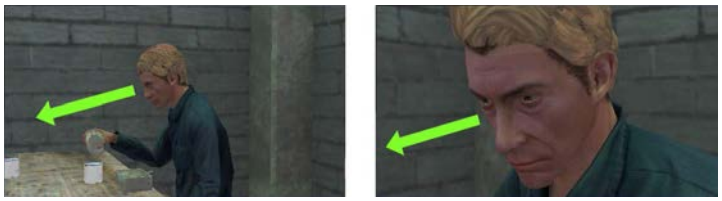
where ϕ is a non-linear function of the change in on-screen location (in terms of distance) such as a sigmoid or threshold function. Each term in the sum is further weighted with the average on-screen size s_k^{avg} of the subject in the two fragments

$$s_k^{avg} = \frac{s_k(f_1) + s_k(f_2)}{2}$$

Figure 4.3 displays two examples of screen continuity score.

2.3.2 Gaze continuity

$C^T_{Gaze} = 1$



$C^T_{Gaze} = 0$

Figure 4.4 – Gaze continuity scores. Left: the gaze direction of the main character changes between the left and right images, resulting in a poor cut. Right: keeping the gaze directions consistent results in a better cut.

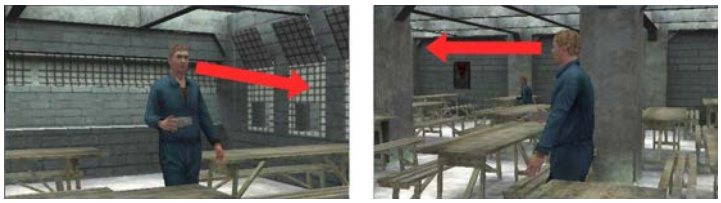
Another important focus of attention when watching a movie is the gaze direction of subjects. We propose a cost function that penalizes viewpoint changes that cause apparent reversals in the subjects' gaze directions. This cost is expressed as

$$C^T_{Gaze}(i,j,t) = \sum_{k \in K_1 \cap K_2} s_k^{avg} \cdot \delta\left(\ \mathrm{sign}\left[g_k^x(f_1)\right],\ \mathrm{sign}\left[g_k^x(f_2)\right]\ \right)$$

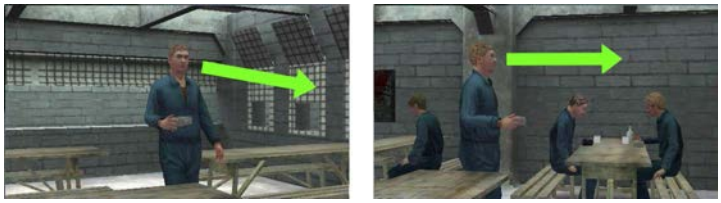
where $g_k^x(f)$ is the x component of $g_k(f)$. Figure 4.4 displays two examples of gaze continuity scores.

2.3.3 Motion continuity

Motion direction of subjects in two successive shots represents another focus of attention. The corresponding transition score penalizes viewpoint changes that cause



$C^T_{Motion} = 1$



$C^T_{Motion} = 0$

Figure 4.5 – Motion continuity scores. Left: the main character's head is oriented differently in left and right shots, resulting in a poor cut. Right: keeping the head orientations consistent results in a better cut.

apparent reversals in the subjects' motions. This score is expressed as

$$C^T_{Motion} = \sum_{k \in K_1 \cap K_2} s^{avg}_k \cdot \delta\left(\text{sign}\left[m^x_k\left(f_1\right)\right], \text{sign}\left[m^x_k\left(f_2\right)\right] \right)$$

where $m^x_k(f)$ is the x component of $m_k(f)$. Figure 4.5 displays two examples of motion continuity scores.

2.3.4 Left-to-right ordering

The left to right ordering of subjects is another important factor for ensuring screen continuity. Subjects whose relative screen location are reversed appear to be jumping around, which attracts attention to the cut. We penalize such situations with the following penalty cost

$$C^T_{Order} = \sum_{k,k' \in K_1 \cap K_2} s^{avg}_{k,k'} \cdot \delta\left(\text{sign}\left[e^x_k\left(f_1\right) - e^x_{k'}(f_1)\right], \text{sign}\left[e^x_k\left(f_2\right) - e^x_{k'}(f_2)\right] \right)$$

where $e^x_k(f)$ is the x component of $e_k(f)$. Unlike previous cases, each term is here weighted with $s^{avg}_{k,k'}$, the minimum value of the average screen size of subjects k and k'

$$s^{avg}_{k,k'} = \min\left(s^{avg}_k, s^{avg}_{k'}\right)$$

This gives more importance to changes in the relative screen locations of two subjects in the foreground, who are both the focus of attention. Figure 4.6 displays two examples of left-to-right ordering score.

$C^T_{Order} = 1$

$C^T_{Order} = 0$

Figure 4.6 – Left-to-right ordering scores. Left: the foremost character moves from the rightmost position to the leftmost position, resulting in a poor cut. Right: keeping the relative ordering of all characters results in a better cut.

$C^T_{Jump} = 1$

$C^T_{Jump} = 0$

Figure 4.7 – Jump-cut scores. Transitions with insufficient change in size or orientation *w.r.t.* actors do not make good cuts (left) while transitions with significant changes make good cuts (right).

2.3.5 Change-in-angle-or-size

An editor should always avoid putting together two shots which are too similar to each other. In other terms, a cut should be avoided between two viewpoints when there is not sufficient change in either the apparent size or the profile angle of at least one subject. Such a cut would be interpreted as a sudden change in the subject's pose, also

known as a *jump cut*. We penalize this type of transitions by the following cost

$$C^T_{Jump} = \sum_{k \in K_1 \cap K_2} s_k^{avg} \cdot \delta'\left(\left|s_k\left(f_1\right) - s_k\left(f_2\right)\right|, \Delta s_k\right) \cdot \delta'\left(\left|\theta_k\left(f_1\right) - \theta_k\left(f_2\right)\right|, \alpha\right)$$

where $\theta_k(f)$ is the profile angle of actor k in fragment f, α is the minimum accepted change in profile angle, and Δs_k is the minimum accepted change in the apparent size. Further, we here introduce δ', a modified version of the *Kronecker* symbol, which we define as

$$\delta'(x, y) = \begin{cases} 1, & \text{if } x < y \\ 0, & \text{if } x \geq y \end{cases}$$

We thus favor transitions where a significant change in appearance occurs for each subject on the screen; and we favor changes occurring on foreground subjects rather than background subjects. Transitions with no change for all subjects receive the highest penalty (see Figure 4.7).

$C^T_{Size} = 1$

$C^T_{Size} = 0$

Figure 4.8 – Size continuity scores. Left: cutting directly from medium close-up to long shot results in a poor cut (the change in shot range is too strong). Right: cutting to a medium shot results in a smooth cut.

We also penalize excessive changes in shot sizes, *i.e.* between a Long Shot and a Close-up, because they make it more difficult for the viewer to recognize correspondences between subjects in the two shots. This is enforced with the simple cost term

$$C^T_{Size} = \sum_{k \in K_1 \cap K_2} \phi\left(\left|s_k\left(f_1\right) - s_k\left(f_2\right)\right|\right)$$

where ϕ is a non-linear function of the size difference, such as a sigmoid or threshold function (see Figure 4.8).

2.3.6 Action line continuity

In addition to preserving the continuity of subjects between successive shots, it is equally important to preserve the direction of the *line of action* between subjects

$C^T_{LoA} = 0$

$C^T_{LoA} = 1$

Figure 4.9 – Line of action scores. In the left-most image, the gaze direction of the main subject defines the line of action. The right-most image shows the second subject. Top: the gaze direction of subjects are opposite, conveying that they are looking at each other. The action line is maintained. Bottom: the gaze direction is the same for both subject, breaking the action line. The camera has crossed the line, resulting in a poor cut.

(usually the world line between their eyes or heads). Thus, when cutting between two subjects A and B, the direction of the line (AB) should be preserved; even if the two subjects are not simultaneously visible on the screen. This typical case occurs when subjects are looking at each other: the cut is good when gaze directions cross each other in the successive shots, which we express as

$$C^T_{LoA} = \sum_{\substack{k \in K_1 - K_2 \\ k' \in K_2 - K_1}} s^{avg}_{k,k'} \cdot \delta \left(\text{ sign} \left[e^x_k \left(f_1 \right) - e^x_{k'} \left(f_1 \right) \right], \text{ sign} \left[e^x_k \left(f_2 \right) - e^x_{k'} \left(f_2 \right) \right] \right)$$

where $e^x_k(f)$ is x component of $e_k(f)$. Note that we also project the eyes of off-screen subjects, without clipping to the image frame. As a result, the lines between all subjects must retain their relative orientations when projected to the screen plane. Figure 4.9 displays two examples of score according to action line continuity.

2.4 Relevance of a transition

Relevance of a transition is measured by exploring its capacity to enhance a viewer's comprehension of successive narrative events and causal links between them. We choose to evaluate the relevance of a transition with two elements: an "Action" term, and a "View" term. The Action term measures the effectiveness in enforcing a continuity in the story, through the use of a cut between two narrative events n_1 and n_2. The View term measures the effectiveness in providing an acceptable link between these two events, through the use of a cut between a first viewpoint conveying n_1 and a second viewpoint conveying n_2.

To this end, we define two cost tables: $A^T(a_1, a_2)$ (where $a_i = type(n_i)$) and $V^T(s_1, p_1, s_2, p_2 \mid a_1, a_2)$. A^T specifies the effectiveness in cutting between two events n_1 and n_2 *w.r.t.* their (causal) links in the story. V^T specifies the effectiveness of a transition between two viewpoints (each characterized by a shot size s_i and a profile angle p_i), knowing the two narrative events and their existing links in the story. Each effectiveness is expressed as a cost ranging from 0 (most effective) to 1 (least effective). For example, within in a dialog, a cut to a viewpoint that still convey the dialog may be preferred to a cut to another event that has no link with the dialog. Similarly, when a change in speaker occurs in the dialog, a cut from a OTS shot on the first subject speaking and an OTS shot on the second subject speaking will probably be preferred, rather than cutting to a viewpoint where the new speaker is not framed.

We expressed this relevance cost as the product of two costs C^T_{Action} and C^T_{View}, intended to evaluate the effectiveness resp. on the Action term and on the View term

$$C^T_{Rel}(i,j,t) = C^T_{Action}(i,j,t) \cdot C^T_{View}(i,j,t)$$

with

$$C^T_{Action}(i,j,t) = [imp(n_1) + imp(n_2)] \cdot A^T[\, a_1, a_2 \,]$$
$$C^T_{View}(i,j,t) = V^T[\, size(n_1, f_1), pr(n_1, f_1), size(n_2, f_2), pr(n_2, f_2) \mid a_1, a_2 \,]$$

where $imp(n)$ denotes the relative importance of narrative event n in the story.

The values in tables P^T_{Action} and P^T_{View} can be left under the control of the user, or be inferred from example movie scenes.

2.5 Pace in transitions

In his book *History of Film Style and Technology* [Sal03], Barry Salt asserts that shot durations in movies generally follow log-normal distributions – similar to sentence lengths in natural language – and proposes to use the parameters of the log-normal distribution as a signature of film editing styles.

Remember the 2-parameter log-normal distribution for a variable X can be expressed as a function of its scale parameter m and shape parameter $\sum$. The scale parameter m is related to the mean μ of $\log X$ by the simple relation $\mu = \log m$. The shape parameter is simply the standard deviation σ of $\log X$. The resulting distribution is asymmetric and unimodal.

Because shot durations are such an important element of film editing style, we introduce a duration cost per shot, measuring the deviation from a log-normal law

$$C^P_{Dur}(d) = \frac{(\ln(d) - \mu)^2}{2\sigma^2}$$

where d is the duration of the shot to evaluate.

The values for μ and σ can be left under the control of the user, be inferred from the pacing of actions, or be inferred from example movie scenes.

In this section, we have introduced a computational model for evaluating an entire film edit *w.r.t.* film grammar. We have studied different key features and rules to

consider, and provided evaluation metrics for them. In the next section, we present an approach to film editing that take full advantage of our quality measurement and build upon CineSys to automate the task of editing a grammatically correct movie, while providing a filmmaker with direct control on the final edit.

3 An automated approach to constructing a well-edited movie

In this section, we propose a general computational framework which provides the user with an interactive process to generate a cinematic editing of a 3D animation. Based on the previously presented quality metrics for shots and transitions, we first cast film editing as selecting a path in time through a collection of takes (a take is a continuous sequence of images from a given camera) and precisely deciding when to cut in and out the takes. We then propose an algorithm suitable for on-line editing which uses an efficient best-first search technique. The algorithm relies on short-term anticipation to improve quality in cuts and produce movies consistent with the rules of cinematography and editing. The user can further specify "key shots" to constrain the search. In contrast to previous work, our method precisely examines all possible edit points and relies on a well-founded model to account for shot duration.

The section is organized as follows. After presenting an overview of our system, we detail how we generate takes, and report on the search process we propose for exploring the possible sequences of shots. We then present a collection of results before discussing the limitations of the system.

3.1 Overview

The generation process is decomposed into three major steps.

- In a first step, we procedurally compute a collection of takes from the events occurring in the 3D animation (the collection of takes is large enough to make the search problem strongly combinatorial by generating many different camera configurations – 80 in average per narrative event). All takes are cut into fragments of duration Δt and individually evaluated to establish their fragment quality.
- In a second step, we construct a graph (which we refer to as the *editing graph*), in which a node represents a fragment of a take i at time t, and an arc represents either a *cut-arc* or a *shot-arc*. A cut-arc represents a cut from a take i at time t to a take $j \neq i$ at time$t + \Delta t$. A shot-arc represents continuity (*i.e.* no cut) in a take i between time t and time $t + 2\Delta t$.
- In a last step, a search process is performed through this graph to compute a traversal of the animation sequence.

To integrate the user in the process of designing a sequence, constraints can be added, *e.g.* enforcing camera positions at specific times in the editing graph (we refer to these constrained shots as *key shots*). In this case, our system computes all intermediate shots that match the key shots ensuring both continuity rules and appropriate pacing. All variations in cinematic style are possible through the specification of numerous

parameters (pacing, preferred viewpoints for each type of event, preferred transitions, ...).

3.2 Computing takes

The first step consists in generating takes. We build upon our cinematic engine to procedurally compute, for each narrative event in the story, the set of possible viewpoints around subjects. A take is then associated with each computed viewpoint – we here assume that takes are "static" (*i.e.* no camera motion) –, and the take is made available for the duration of the event. We further extend this duration by using a look-ahead of $1s$ before the beginning and after the end of the event, to allow the director anticipating an event or making a short pause after conveying an event.

Sometimes, subjects involved in the event are moving so much that it is impossible to keep them within the frame during the entire duration of the take (typically, in feet actions). In such a case, we split the interval of time in which the event occurs into a set of sub-intervals of duration at least $1s$. We then apply the same generation process as previously, with a look-ahead on each sub-interval.

We have generated a set of takes conveying the events of the story. We can now use these takes to reason on possible sequences of shots and cuts.

3.3 Editing graph

Remember that a movie can be viewed as a sequence of shot fragments of duration Δt (time step). Our editing problem can be cast into the choice, at each time step, of the best take fragment to use *w.r.t.* the fragments that have been chosen at previous time steps.

The construction of our editing graph is operated in two steps. We first split the takes we have computed into fragments of duration Δt. We then construct an oriented graph (the editing graph) over all take fragments, which we will use as a way to explore the possible sequences of shots, cuts, and moments of cut. Our editing graph can be formalized as follows. Each node n of the graph corresponds to a take fragment ($n = \langle i, t \rangle$, where i is the take index, and t the beginning time of the take fragment). We then use two different types of arc in this graph: shot-arcs and cut-arcs. A shot-arc a_s corresponds to a continuity (*i.e.* no cut) in the take i from time t to time $t + 2\Delta t$, which we can formalized as the transition

$$a_s = \langle i, t \rangle \rightarrow \langle i, t + \Delta t \rangle$$

A cut-arc a_c corresponds to a cut from a take i to a take j ($i \neq j$) at time $t + \Delta t$, which we can formalized as the transition

$$a_c = \langle i, t \rangle \rightarrow \langle j, t + \Delta t \rangle$$

At each time step t, we create a graph node $\langle i, t \rangle$ for each available take. We then create a shot-arc between all pairs of node $\langle i, t \rangle$ and $\langle i, t + \Delta t \rangle$ when such nodes are available; and we create a cut-arc between all pairs of node $\langle i, t \rangle$ and $\langle j, t + \Delta t \rangle$ ($i \neq j$)

when such nodes are available. We further introduce two 'neutral' nodes (of cost 0): an initial node S (start) and and a final node E (end). Node S is linked to each available node at time 0 with a 'neutral' arc (of cost 0). Similarly, each available node at time t_{end} (end time of the story) is linked to node E with a cut-arc. Figure 4.10 illustrates the construction of the editing graph. The editing problem is then cast into a search of the best path from node S to node E, of which cost is computed as a sum of node costs and arc costs. The cost of a node corresponds to C^S, the cost of shot-arc is 0 and the cost of a cut-arc corresponds to C^T together with the cost of pace on the (current) shot duration C^P_{Dur}.

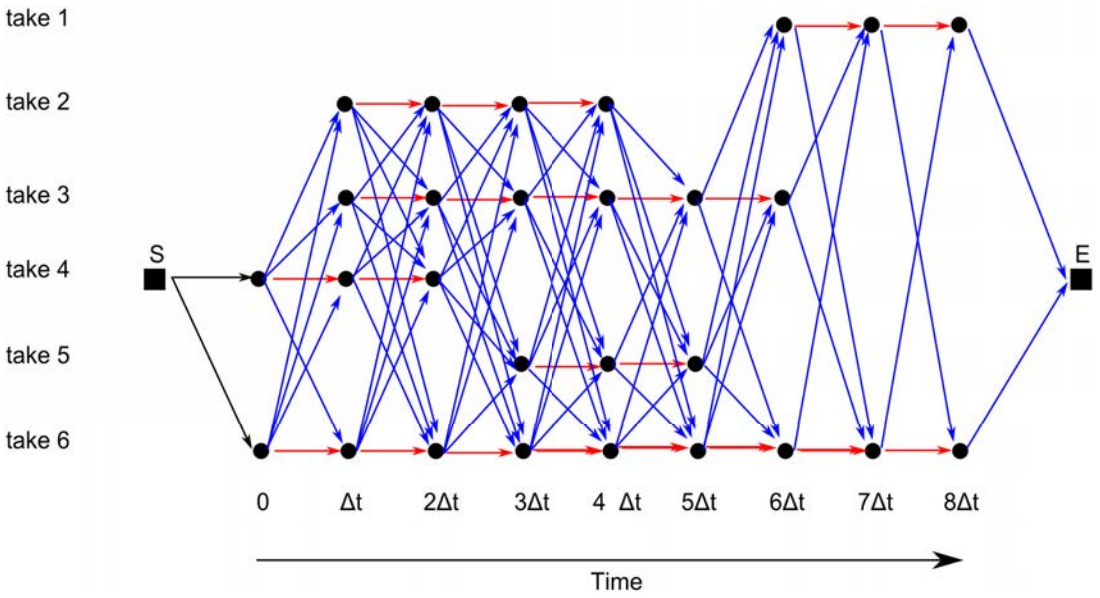


Figure 4.10 – Editing graph. Nodes (black dots) represent available take fragments. Shot-arcs (in red) represent continuity in a take (*i.e.* no cut). Cut-arcs (in blue) represent a cut from a take to another. The black squares (S and E) represent resp. the initial node (start) and final node (end). They are linked to the takes available resp. at the beginning and at the end of the story. The construction of a movie is then operated by searching the best path from node S to node E in the editing graph.

We discuss the way the user can constrain the search, and how constraints are taken into consideration in the editing graph, in Section 3.4.2.

3.4 A best-first search for film editing

The computation of an optimal sequence of shots consists in searching for the path of least cost in our editing graph. Algorithms to solve these classes of problems (dynamic programming, A*-based algorithms) remain computationally complex in practice. Additionally, the problem we address displays a very large collection of solutions which yield similar costs (*e.g.* consider the case of symmetric viewpoints around a subject), and for which an optimal computation seems unnecessary. We thus propose a sub-optimal best-first search with anticipation through the editing graph.

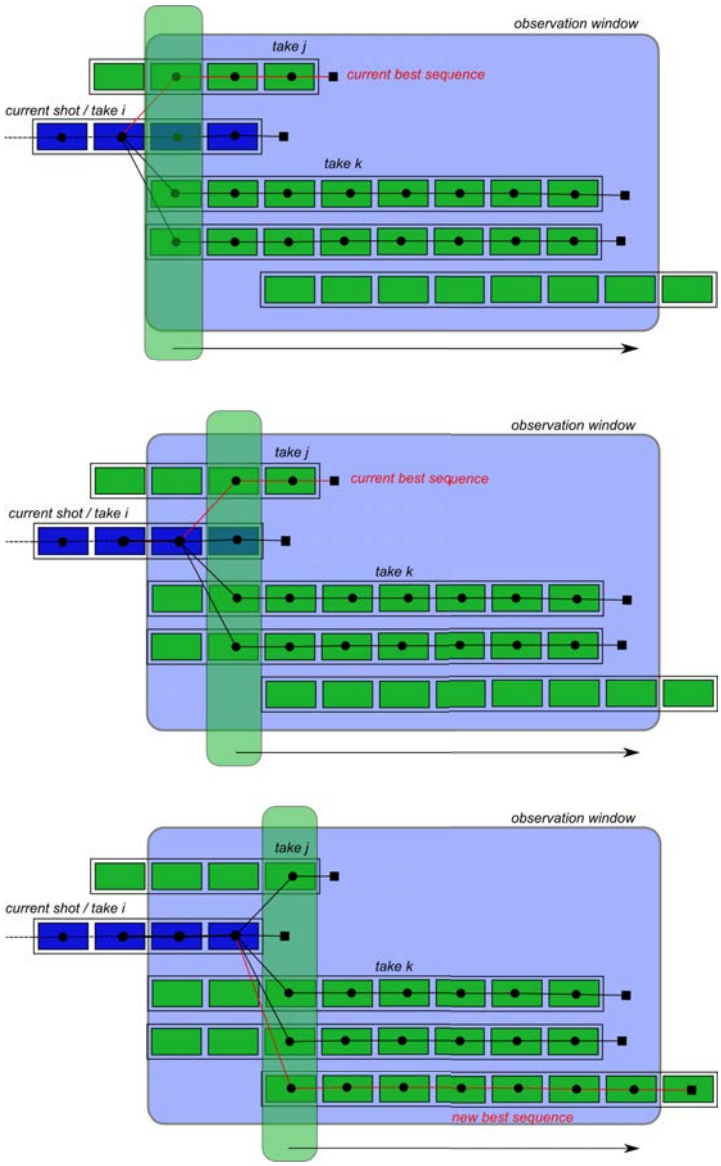


Figure 4.11 – Searching for the optimal transition in the observation window to decide whether to cut or stay in the shot. Each green/blue rectange represents a fragment of a shot. On the top image, the current optimal shot sequence is drawn in red and recommends to cut at time t. A scanning process over the observation window (window in light blue) is performed to seek for a possible better moment. On the bottom image, a sequence with a better cost is found (displayed in red). The observation window is then shifted ahead one fragment in time and the process starts over.

3.4.1 Best-first search with anticipation

Given that the quality of the overall sequence strongly depends on the output of this search process, we propose to enhance the search by proposing an *informed* best-first search. This informed search uses a sliding observation window over the next fragments

to locally reason on the best moment for a transition.

At a given depth in the search process (*i.e.* advancement in time over the fragments), a critical decision needs to be made whether to stay within the current take or to perform a transition to another take. To inform this decision, we rely on this observation window over the next w fragments. We study within this window the best transition to be performed, given the knowledge of the takes to come. If the best transition occurs immediately, the cut is performed. If the best transition occurs later in the observation window, we shift the window a fragment ahead and start the process over.

To compute the best moment for a transition inside the observation window we use an incremental scanning process. The process is illustrated in Figure 4.11. Given the current take is i, for a given time t in the observation window and for each take $j \neq i$, we compute the cost C^{Cut} of a possible transition from shot i to shot j, and we compare it to the cost C^{Stay} of staying in the current take for the next w fragments.

$$\begin{aligned} C^{Cut}(i,j,t_c) \; &= W^S \cdot \left[\sum_{t_w=0}^{t_c} C^S(i, t+t_w) + \sum_{t_w=t_c}^{w\Delta t} C^S(j, t+t_w) \right] \\ &\quad + W^T \cdot C^T(i,j,t+t_c) + W^P \cdot \left[C^P_{Dur}(d_w + t_c) + C^P_{Dur}(w\Delta t - t_c) \right] \end{aligned}$$

and

$$C^{Stay}(i) \; = W^S \cdot \left[\sum_{t_w=0}^{w\Delta t} C^S(i, t+t_w) \right] + W^P \cdot C^P_{Dur}(d_w + w\Delta t)$$

where t is the current time (*i.e.* the beginning time of the observation window), t_c is the time of the cut within the observation window and d_w is the duration of the current shot at time t (*i.e.* the duration already spend in that shot before the observation window). Further, for a shot for which (i) the end time is not yet known (*i.e.* occurs after the end of the observation window) and (ii) the duration d is still lower than the mode (*i.e.* $\ln(d) < \mu$), we set $C^P_{Dur}(d)$ to 0. This is to avoid penalizing such shots, as the cut will occur later (and we do not know at which time yet). Indeed, in such a case we cannot yet state that the shot duration will not be close to the mode, which would correspond to the lowest cost of pace (*i.e.* 0).

Theoretically, to decide whether to operate a cut within w time steps or not, we need to scan the entire observation window. This is the worst case, using a search algorithm with complexity $O(nw^2)$ (considering n takes and w time steps in the observation window). If the cost of staying in the current take i have the minimal cost, we have

$$C^{Stay}(i) \leq \min_{j,t_c} C^{Cut}(i,j,t_c)$$

In this case, we extend the duration of current shot by Δt and we shift the observation window a fragment ahead. Another case is when there exists a take j such that

$$\begin{cases} C^{Cut}(i,j,0) = \min\limits_{k} C^{Cut}(i,k,0) \\ C^{Cut}(i,j,0) < C^{Stay}(i) \end{cases}$$

In such a case we need to know whether to cut at the current time t to take j, or to wait for a better moment. To implement this, the process explores the successive

fragments at $t_c = \Delta t, t_c = 2\Delta t,$ .. $t_c = w\Delta t$ in the observation window until a cost lower than $C^{Cut}(i, j, 0)$ is found. On one hand, if no better cut is found later in the window, the current time t represents the best moment for a transition and a cut is performed toward shot j. On the other hand, if any better cut is found later in the window, we know that the best moment for cutting (no matter which) occurs later; the observation window can then be shifted a fragment ahead. And so it is for the initial case (continuing with the current take). We also know that the best moment for cutting (no matter which) occurs later as soon as

$$C^{Stay}(i) \leq \min_j C^{Cut}(i, j, 0)$$

This has two main advantages. First, we can evaluate cut moments with more anticipation on what occurs after the cut. Second, this provide a significant improvement to the search complexity. In most cases, our scanning process have a complexity close to $O(n)$; and the worst case only occurs when a cut is necessary.

We fixed the size of the observation window to the mod m of the log normal law ($m = \log \mu$): $w = m/\Delta t$. Indeed, the evaluation over the window only considers one possible transition between takes, which optimally occurs every m seconds. Any window size larger than the mod would provide an erroneous estimation (considering one cut when two may occur) and any size smaller would not fully consider the cost of the pacing over the next shot.

3.4.2 User Constraints: Key Shots

The process we describe enables a user to specify any number of key shots in the sequence. Each key shot is expressed as the specification of the take to use during a selected interval of time. One may also specify a given part of an edit, s/he finds satisfying, as a sequence of key shots. Key shots thus represent constrained shots which need to be integrated as mandatory nodes in the editing graph. These constraints are implemented by modifying the topology of the editing graph. We first create an intermediate 'neutral' node K_i (similar to nodes S and E) to represent each key shot i in the editing graph. Each node immediately before a key shot i is then only linked to K_i, by a cut-arc. In a symmetric way, only K_i is linked to each node immediately after the key shot, by a cut-arc. This enables recomputing some parts of the edit only, by searching the editing graph (in parallel) between each pair of neutral node (*i.e.* from node E to K_1, from a node K_i to a node K_{i+1} and from node K_n to node E).

3.4.3 Bidirectional search

The presented algorithm provides a good support for constructing correct movies. There however remain a significant issue on the last shot, which may be of poor quality due to pace. Indeed, our search algorithm may lead to a last shot with very short duration, as all nodes (take fragments) at the end of the movie are forced to cut to the final node E.

To improve the quality of the process, we propose to run a second best-first search process, in reverse order between all pairs of neutral nodes. We then build a new graph

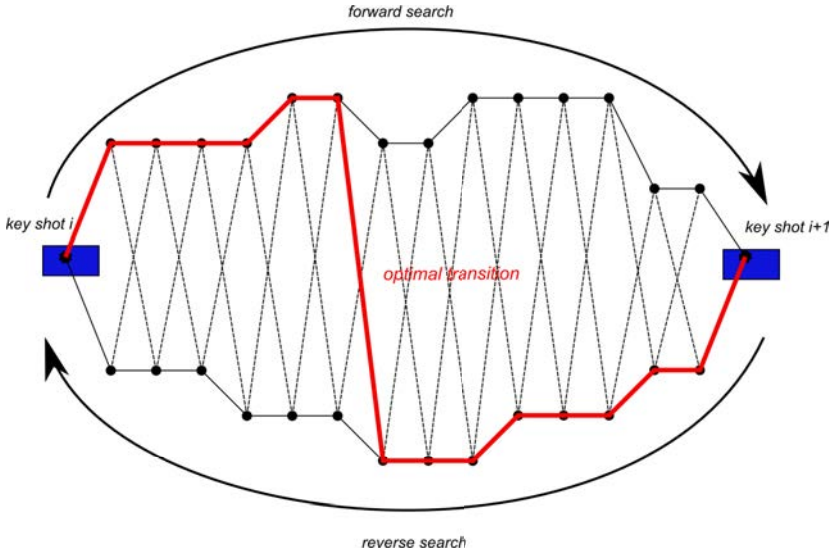


Figure 4.12 – Searching for the optimal transition between forward and reverse searches in the editing graph. Forward search may lead to an ending sequence of average quality due to constraint imposed by the user on the last shot. A reverse process is then performed from the last key shot to the first one. And in a final stage, we create a new editing graph with all possible transitions, and compute the optimal transition between both paths.

only composed of the forward and backward paths, together with all possible transitions between the take fragments composing the paths (see Figure 4.12). Finally the optimal transition between the two paths (the transition of minimal cost for the whole graph) is computed using a simple graph traversal algorithm (given the topology, the number of alternatives to evaluate is M the number of time steps).

This interactive process offers both the flexibility of fully automated computation and the ability to let the user integrate his own shots.

3.5 Feature weights selection

Putting everything together, we now build the cost function for an arbitrary sequence of key shots, as a weighted sum of all its features. We choose the weights with a linear discriminant analysis (LDA) of shots and cuts labeled as 'grammatical' and 'ungrammatical'. LDA finds linear combinations of the cost features that minimize the overlap of the transformed distributions [DHD00]. We separately set the weights with a first LDA for the shot terms using shot examples, then for the cut terms using cut examples. Finally, we arbitrarily set the relative weights W^S, W^T and W^P to 2, 1 and 1, resp. representing the weighting of C^S, C^T and C^P_{Dur}.

3.6 Experimental results

We have generated experimental results by using our system on the 1984 animation content and annotated scene. The duration of the animation is 3 minutes, with over 70 actions occurring with overlaps. Currently, the duration Δt of a fragment is set to $250ms$ (*i.e.* cuts between shots may occur every $250ms$). For each action occurring in the scene, we generate between 50 shots (single-subject actions) and 80 shots (two-subject actions), corresponding to classical cinematic viewpoints.

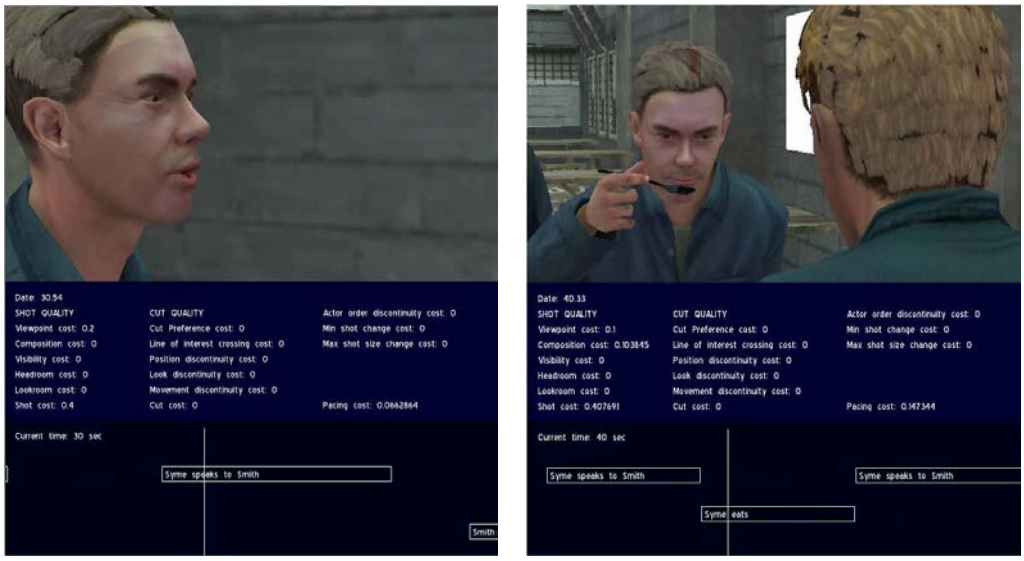


Figure 4.13 – Two images illustrating the on-going actions (bottom part), the costs (middle part) and selected shot (top part). The left shot clearly displays the action (the character is speaking), and respects head-rooom and look-room. The right shot displays the actor eating and guarantees a good visibility even if another character is in the frame.

Figure 4.13 displays two shots at different moments in a generated movie with details on the current action and values of all cost functions. We now allow the user to constrain the first shot of the movie by selecting a key shot. The system therefore computes a sequence which respects all continuity rules. Figure 4.14 compares the first shots with and without the key shot constraint. For the three-minute sequence in 1984, the fully automatic method takes 15 seconds; generating key shots manually from a gallery of candidate cameras and automatically recomputing the remaining shots makes it possible to edit the entire scene in less than five minutes.

In this section, we have introduced a novel framework for virtual cinematography and editing which, in contrast to previous approaches, considers an evaluation of the quality of the movie edit. Experimental results demonstrate that our approach is efficient in separating correct from incorrect shot sequences. Furthermore, we have introduced an efficient search strategy for finding the best sequence of shots from a large number of candidates generated by traditional film idioms.

Figure 4.14 – Two different edits computed without a key shot at the beginning (top), and with a key shot at the beginning (first shot from the bottom row is user-constrained).

One of the limitations of our system resides in the learning phase. The system requires a substantial amount of correct and incorrect examples. In this task, it is however possible to provide the system with real movies, in which frames would be annotated with information used in our evaluation metrics (*e.g.* subjects' eyes location, gaze direction or size) – this would for instance enable approximating a given directorial style –, even if such an annotation remains tedious. Further, even in case of manually created edits, the learning phase provides a significant improvement compared to previous systems. Indeed, our system can be parameterized through an explicit specification of what leads to a "correct" or "incorrect" result, whereas existing approaches necessitate the user to manually tune sets of parameters which are not directly related to the result quality. In our system, the contribution lies in the fact that the user interacts with a high-level parameterization of the system, and do not need to be concerned about the low-level parameters of the system (*i.e.* feature weights).

A second limitation is in our evaluation of pace. Inherently, the log-normal distribution of shot durations is the (involuntary) result of a filmmaker's creative process, which is further strongly linked to the flow of events. Using such an output as an input of an editing system, when applied to a completely different flow of events, may lead to an edit that seems unnatural. This raises an open question, about the best moments for cutting. Relevance of the transitions timing strongly depends both on the flow of narrative events, on their unfolding and on existing causal relations between them. In particular, the extraction of critical moments for entering or leaving takes is an interesting and open problem.

A third limitation of our system is that the user control on the final edit remains limited and appears tedious. Indeed, one has to explore takes on multiple features: the time, the number of subjects (*e.g.* single-subject events, two-subjects events, ...), as well as the shot type and shot size. In this task, the user is missing the provision of a clear visualization of the events timeline so as to decide when to cut, as well as a mean to compare a take with one another so as to decide where to cut to. Our approach is however the first automated editing technique that gives a user a real opportunity to interact with the final edit directly.

A last limitation of our technique is that it does not account for dynamic shots, *i.e.* camera paths inside takes. Indeed, it is difficult to explore every possibility of a moment to initiate and terminate a camera motion, as well as every possibility of an initial and final camera configuration. Similarly, evaluating the quality of each

camera configuration on each camera path would lead to an exponential computational complexity.

More generally, an important research issue is the discovery of film idioms and styles from examples. For instance learning pace parameters from a user created edit is an interesting problem; and more particularly in the case where shot durations of this edit do not follow a log-normal distribution.

In the next section, we present an intelligent assistant which provides the user with a more efficient workflow for creating a film edit and which enables learning cinematic idioms from user live camerawork.

4 The Director's Lens: an intelligent assistant for interactive preview

In in this section, we build upon the limitations of our previous technique, together with a more general analysis of automated camera control techniques. We here make two findings: it is difficult to (i) integrate all composition feature in the process of positioning a camera and (ii) create camera paths that mimic cinematic camera motions (particularly when considering complex scenes, with occlusions and dynamic scene elements). In this section, we propose to combine automated control techniques and manual control techniques. We here present the *Director's Lens*, an intelligent interactive assistant for crafting virtual cinematography using a motion-tracked handheld device that can be aimed like a real camera. Our Director's Lens system builds upon CineSys to compute, at the request of the filmmaker, a set of suitable camera placements (or *suggestions*) for starting a new shot. These suggestions represent semantically and cinematically distinct choices for visualizing the current narrative. In computing suggestions, the system considers established cinema conventions of continuity and composition along with the filmmaker's previous selected suggestions. His or her manually crafted camera compositions are also considered, by a machine learning component that adapts shot editing preferences from user-created camera edits.

4.1 Overview

The core of our Director's Lens system relies on the automated generation of a large range of suggested cameras and on the ranking of these suggestions according to the last frame of the current shot.

As the user requests a list of suggestions (at time in the movie we denote t_s), our system performs a three-step computation:

- from the screenplay the system selects the list of all narrative events overlapping time t_s or occurring within a second after time t_s (this enables to suggest a shot which anticipates an event);
- for each selected event, the set of key subjects (characters, objects, buildings) is extracted and our cinematic engine generates a wide collection of viewpoints ensuring both a complete coverage of the key subjects and events, as well as

significant difference between viewpoints (see detailed description in next Section). Each viewpoint represents a suggested starting point for a shot;

- each suggestion is then ranked by considering the enforcement of cinematic continuity rules between the current viewpoint and the suggested viewpoint, the quality of the composition in the shot (how elements are spatially organized on the screen), the relevance of the suggestion with relation to the action type and the quality of the transition between the current shot and the suggestion.

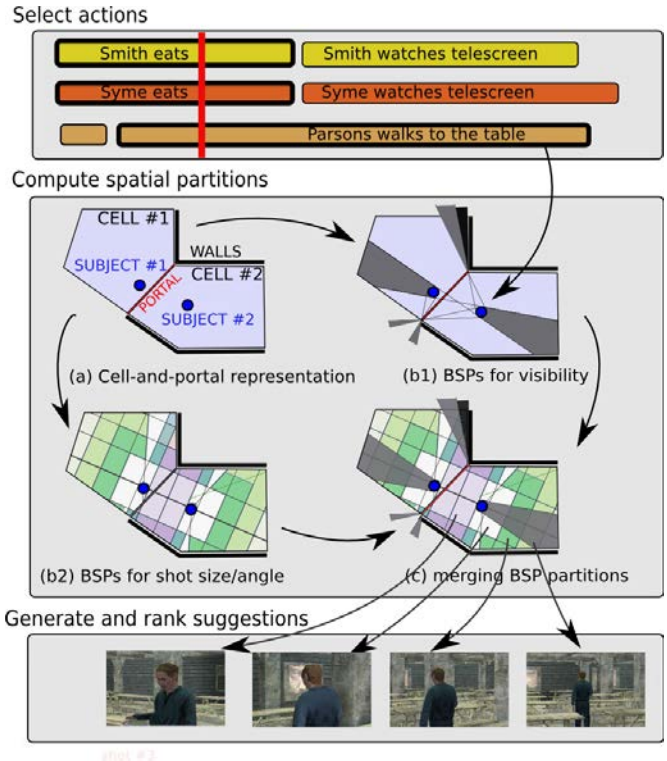


Figure 4.15 – The automated computation of suggestions. Each selected action in the screenplay leads to the computation of a set of spatial partitions (each partitioned area represents a typical shot of the action). Suggestions are computed and ranked for each area.

4.2 Computing suggestions

The first step consists in generating, for a given set of events occurring at time t_s, a range of viewpoints around participating subjects of these events. To that purpose, we build upon our space partitioning into Director Volumes. For each volume, our process generates three suggestions: a high-angle shot, a low-angle shot, and a medium angle shot; and for each of these suggestions, a default screen composition is computed by enforcing the rule of the thirds (key subjects are spatially arranged on the screen to appear at the intersection of power lines [Ari76]).

4.3 Ranking suggestions

Once a set of suggestions is computed (for a two-subject configuration, over 240 shots are generated and for a one-subject configuration, 120 shots are generated), our system performs a quality ranking process, whose result is used when displaying the suggestions to the user (see Section 4.5).

The quality $Q(s) > 0$ of a suggestion s is defined as a weighted sum of qualities assessing specific features of s:

$$Q(s) = w^S_{Comp} \cdot Q^S_{Comp}(s) + w^T_{Cont} \cdot Q^T_{Cont}(s) + w^S_{Rel} \cdot Q^S_{Rel}(s) + w^T_{Rel} \cdot Q^T_{Rel}(s)$$

with

$$Q^T_{Cont}(s) = w^T_{LOI} \cdot Q^T_{LOI}(s) + w^T_{Change} \cdot Q^T_{Change}(s)$$

where w_i are weights associated with each feature. In these formulas, Q^T_{Cont} measures the enforcement of *continuity* rules in the cut from current shot to suggestion s. Q^T_{LOI} then measures compliance with *line-of-interest* rule and Q^T_{Change} the compliance with the change-in-angle-or-size rule. Q^S_{Comp} represents the satisfaction of composition rules, Q^S_{Rel} represents the relevance of the suggestion s with relation to the relevance of the current event, and Q^T_{Rel} represents the relevance of the transition between the current viewpoint and the suggestion s. All the $Q^S_i(s)$ and $Q^T_j(s)$ functions return positive real values. We have implemented some of these quality functions by relying on either the capacities of filters supplied by our cinematic engine (Q^T_{Cont}), or the evaluation function we have presented in Section 2 ($Q^S_{Comp}(s)$).

We hereafter detail how we build each of these quality functions.

4.3.1 Respecting continuity in the cut

First, filters associated with continuity editing rules (the line-of-interest filter and the change-in-size/change-in-angle filters) are switched to "annotation" mode. If a volume violate a rule associated with a given filter, this filter will tag it with a specific annotation. Then, the resulting suggestions are then evaluated *w.r.t.* these annotations:

$$Q^T_X(s) = \begin{cases} 1, & \text{if } s \text{ respects the rule } X \\ 0, & \text{if } s \text{ violates the rule } X \end{cases}$$

4.3.2 Respecting classical composition rules

We first use the visibility filter supplied by our cinematic engine to prune suggestions where a key subject is occluded by a static element of the scene. We then use the two visual composition features (visibility and look-room) defined in Section 2, to rank the remaining suggestions as follows

$$Q^S_{Comp}(s) = \phi\left(C^S(s)\right)$$

with

$$C^S(s) = w^S_{Vis} \cdot C^S_{Vis}(s) + w^S_{Look} \cdot C^S_{Look}(s)$$

where the feature costs C_i^S are computed similarly as in Section 2.1, and w_i denotes the weight associated with feature i. In the top formula, ϕ represents a non-linear quality function, such as a sigmoid or threshold function; $\phi(s)$ returns a value ranging from 0 (highest value of $C^S(s)$) to 1 (lowest value of $C^S(s)$).

4.3.3 Relevance of a suggestion *w.r.t.* current event

Relevance of a suggestion is measured by exploring the capacity of the viewpoint to enhance a viewer's comprehension of the unfolding actions. Each action has a relevance value that encodes its importance for the story (*e.g.* representing whether the action is a foreground action, an establishing action, or a background action). Viewpoints that depict more relevant actions, from relevant viewpoints enforce the comprehension of the story and will have a higher quality. To evaluate this feature, we use the "action" term defined in Section 2, to rank suggestions as follows

$$Q_{Rel}^S = \phi\left(C_{Action}^S(s)\right)$$

where the feature cost C_{Action}^S is computed similarly as in Section 2.2. As previously, ϕ represents a non-linear quality function, such as a sigmoid or threshold function; $\phi(s)$ returns a value ranging from 0 (highest value of $C_{Action}^S(s)$) to 1 (lowest value of $C_{action}^S(s)$).

4.3.4 Relevance of transitions between viewpoints

We here measure the quality in transitions (*i.e.* cuts) by using transition matrices accounting for both the "Action" and "View" terms described in Section 2.4. We model a transition matrix as a stochastic matrix describing the transitions of a Markov chain. We encode our transitions with a right stochastic matrix, *i.e.* a square matrix where each row consists of nonnegative real numbers summing to 1. Each value t_{ij} of the matrix T (i being the row index, and j the column index) corresponds to the probability of performing a cut from viewpoint (*i.e.* Semantic Volume) i to viewpoint j. We use three different matrices depending on whether the transition is performed inside the same action (T_S), between related actions (T_R), or between unrelated actions (T_U). The quality of the transition is given by an affine function $y = ax + b$, where $a > 0$, $b > 0$ and x is equal to the value t_{ij}^k related to the transition in the corresponding matrix T_k. The values in the matrices therefore represent user preferences in performing cuts. In our approach, the values in these matrices are updated by a learning process described in Section 4.4.

4.4 Learning from the user inputs

We rely on a simple reinforcement learning technique to update the probabilities in the transition matrices, using the cuts already performed by the user. Three distinct transition matrices are encoded depending on whether the transition is performed:

1. **during the same event (matrix T_S)**: the two consecutive shots are conveying the same action;

2. **between related events (matrix T_R)**: the two consecutive shots are conveying two different actions, but the actions have a causal link (*e.g.* in case of dialog, the first action being "Syme speaks Smith" and the second one "Smith answers to Syme" for instance). A causal link is established when the two actions share the same key subjects;
3. **between unrelated events (matrix T_U)**: the two consecutive shots are conveying two different actions, and there is no causal link between them;

These transition matrices actually define preferences in using some transitions between viewpoints over others. A viewpoint is identified by its type (orientation to subject) and its size (Extreme Long to Extreme Close-up). Our system learns the transition matrices from the user inputs, by analyzing the successive choices in viewpoints performed by the user. The learning process operates as follows. Each time the user selects a suggestion as the new camera viewpoint, we first determine the transition matrix T_k to consider by analyzing whether (1) the successive conveyed events are the same, (2) the events are different but causally linked or (3) the events are different and have no causal link. Then all the values t_{ij}^k of the row i of the corresponding matrix T_k (where i is the viewpoint used in the last frame of the current shot), are updated in the following way. We first introduce a matrix P_k, representing the number of use of each transition, in which all values p_{ij}^k are initially set to 1. We then assume that the current viewpoint at time t_s is i. When a viewpoint of type n is selected as the start of a new shot, the value p_{in}^k is increased by 1. The values t_{ij}^k are then re-computed as the number of use of such a transition compared to the number of transitions already operated from viewpoint i

$$t_{ij}^k = \frac{p_{ij}^k}{cuts_i^k}, \text{ with } cuts_i^k = \sum_x p_{ix}^k$$

The probabilities in the matrices influence the quality of a suggestion by ranking preferred cuts higher (the quality of transition $Q_{Rel}^T(s)$ is expressed as a function of t_{ij}^k).

4.5 The Director's Lens system

We now describe the interface and user interactions with the Director's Lens system. To demonstrate the system, we use as example a reconstruction from a sequence of Michael Radford's *1984* movie, together with a set of 70 actions which describe in details the events occurring over the scene. The scene is composed of 5 key subjects (Syme, Parsons, Julia, Smith and an anonymous member of the party).

In the following, we detail user inputs devices we use and the different possible interactions enabled between a user and our system.

4.5.1 User input devices

The Director's Lens system features a 7-inch HD LCD touch-screen mounted to a custom-built dual handgrip rig (see Figure 4.16) that can be paired with either a $3dof$ or $6dof$ motion sensor.

- When used with a 3*dof* sensor (rotation controls camera aim direction), the two thumb sticks control camera position, and buttons adjust lens zoom, movement/zoom speed, and record/playback functions.
- When paired with an optical 6*dof* tracking system, a rigid cluster of reflective markers is fixed to the device and one of the thumb sticks is configured to increase or decrease lens zoom angle.

The spatial configuration (position/orientation) read by the tracking sensor is synchronized with the spatial configuration of the virtual camera in the system, so that (besides a distance scaling factor that can be adjusted) movements of the motion-tracked device will correspond to analogous movements of the virtual camera in the 3D scene.

The system can also be used with just a desktop or notebook computer and a mouse. In this mode, the preferred workflow is primarily one of clicking to select a suggested camera placement for each shot to very rapidly compose a virtual 3D movie made with shots featuring static cameras.

Figure 4.16 – Our hand-held virtual camera device with custom-built dual handgrip rig and button controls, a 7-inch LCD touch-screen.

4.5.2 User interfaces and interaction

The user's interaction with Director's Lens is divided into two interface modes: (i) *explore suggestions* (Figure 4.17) and (ii) *manual control/movie playing* (Figure 4.19).

Interface #1: explore suggestions The *explore suggestions* interface (Figure 4.17) displays the suggestions for beginning a new shot from any instant in the recorded movie. The suggestions are presented as small movie frames, arranged in a grid whose rows and columns correspond to visual composition properties of the suggested cameras. More specifically, the horizontal axis from left-to-right varies by decreasing shot size (or distance) in order of Extreme Long, Long, Medium, Close-up, and Extreme Close-up. The vertical axis from top-to-bottom presents suggestions from a variety of camera heights in order of High angle, Medium angle, and Low angle. For example, Figure 4.17 shows the suggestions that are proposed to transition from a shot depicting the event "Syme eats" to a new shot where two parallel events occur: "Syme eats" (which continues from the previous shot) and "Smith speaks to Syme" (the last frame of the previous shot is shown in the reel at the bottom left). Notice, for example,

that all suggestions in the top row are viewed by cameras positioned above the virtual actors.

When more than one suggestion is generated for each shot size and camera angle, the system displays a stack of frames in that grid position, where the top one is the most highly ranked (of all the suggestions in the stack). If the user wishes to browse a stack (s)he can expand the actually displayed suggestions for a grid column or row by clicking on the corresponding column/row heading. For example, Figure 4.18 shows the suggestions grid expanded to display all suggestions for Medium size / Medium angle shots.

When the suggestions do not refer to the first instant in the movie (*i.e.* there are previous shots), the system allows the user to visually filter the presented suggestions on the basis of respect of cinematography rules, namely the line-of-interest, and minimum change-in-angle-or-size with respect to the previous shot's last frame. For example, in Figure 4.17 we have selected to display only suggestions that respect the line-of-interest rule (*i.e.* the camera is on the same side of the line-of-interest as in the last frame of the previous shot), while Figure 4.18 displays only suggestions where the camera has at least a 30° difference in orientation *w.r.t.* the key subject (with respect to the last frame of the previous shot).

The user selects a suggestion by touching its icon in the grid. The larger image framed in the reel at the bottom left represents the previous shot to assist the user in choosing or modifying a suggestion to best follow its predecessor, and shows the currently selected suggestion to its right. The user finally confirms his or her choice and goes to the *camera control/movie playing* interface.

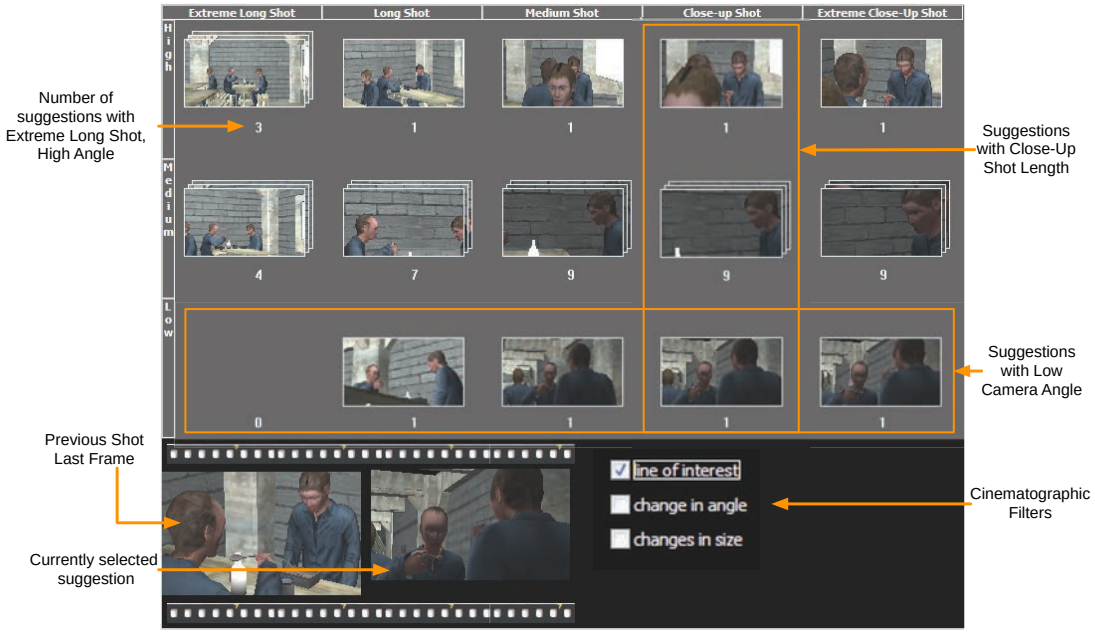


Figure 4.17 – Screenshot of the interface in **Explore Suggestions** mode. In this example, we filter suggestions *w.r.t.* the line-of-intersect rule (only suggestions that respect this rule are displayed).

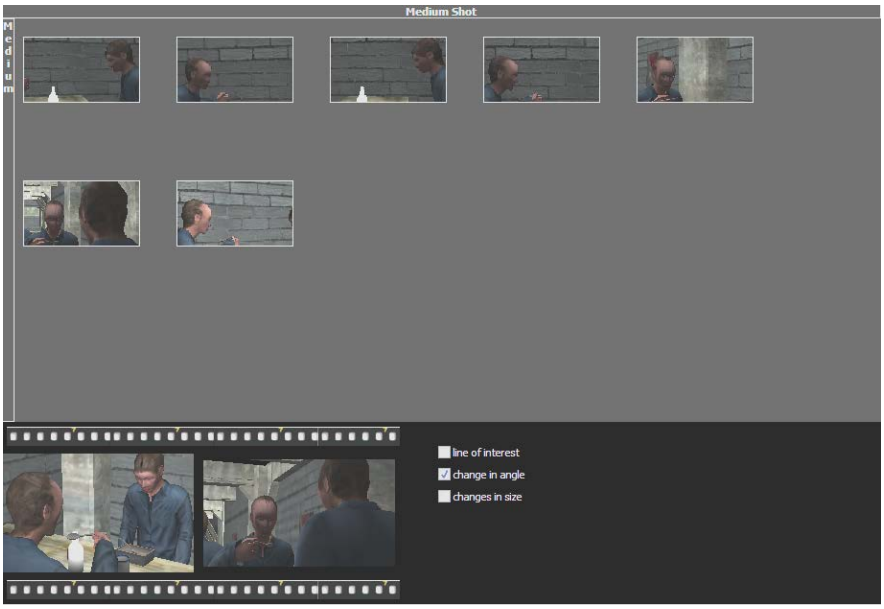


Figure 4.18 – Screenshot of the interface in **Explore Suggestions** mode with expansion of medium shots and medium camera angles that were visualized in Figure 4.17. In this example, we also filter suggestions *w.r.t.* the change-in-angle rule (only suggestions that provide at least a 30° difference *w.r.t.* previous frame are displayed).

Interface #2: manual control / movie playing The *manual control/movie playing* screen (see Figure 4.19) allows the filmmaker to view the recorded movie or manually take control of the camera to record a shot. The interface features a single large viewport to display the 3D scene as viewed from a virtual camera corresponding to either (i) the filmmaker's choice for the current instant in the movie or (ii) the position/orientation of the motion-tracked device (when the tracking of the device is enabled).

When camera tracking is not enabled, the *Play* button allows one to play-pause the animation, and visualize it from the currently recorded shots; while *Previous Shot* and *Next Shot* buttons enables to move in the recorded movie shot-by-shot. With these functionalities, the user can review the recorded movie, and decide where to introduce a cut - this is implicitly introduced when one asks the system to explore suggestions.

When camera tracking is enabled, the virtual camera is driven by the motion tracking and lens zoom controls of the hand-held device. The tracking starts from the currently used camera, so that, if the user is coming from the *explore suggestions* interface (after having selected a suggestion), tracking starts from the virtual camera in the configuration associated with that suggestion. By using the *Turn on Recording* button, the user starts/Stops recording of the stream of virtual camera position, orientation, and lens angle input data. Starting recording also starts playback of the pre-produced animated character event and digitized audio dialog.

A graphic overlay can appear over the camera image to display an animated timeline of the unfolding subject events and dialog. Horizontal bars along each row represent one

Figure 4.19 – Screenshot of the interface in **Manual Control / Movie Playing** mode. The overlay horizontal bars show an animated timeline of screenplay events and their duration, together with an indication of the camera angle and framing that is currently being used in the movie.

subject's event or dialog. The length of each bar represents the duration of that event. Bars scroll by as the animation plays to indicate the passage of time. This allows the filmmaker to move the camera at the precise time in anticipation of an upcoming event or spoken dialog in the screenplay. The overlay also displays the framing properties (*e.g.* Medium, Low, etc.) of the current camera view.

Figure 4.20 demonstrates an example in which the system suggests shots which match the composition of the user's previous shot. In this scene, Smith pours gin while seated facing Syme. The user has composed a shot (shown in the right image in the figure) in which Smith fills the rightmost third of the frame, gazing to the left and the camera is located to his left-hand side. Consequently, when we click to enable the line-of-interest filter, the system generates suggestions in which no viewpoints violate the line-of-interest by placing the camera on Smith's right-hand side. Furthermore, all viewpoints display Smith on the right-most third of the frame. Also note suggested compositions leave more room ahead of Smith's face, as did the user's composition.

4.6 Results

Our system is coded in C++ and uses the OGRE real-time rendering engine, and a custom-written abstraction layer to interface to a variety of button and joystick input devices and motion trackers. In our tests we used 6*dof* rigid body reflective markers with an A.R.T. optical tracker with 16 cameras covering a workspace of 4×8 meters. The application runs on a computer whose video output is routed over an HDMI cable to the small LCD touch screen. The largest computational cost of the cinematic engine is in finding visibility and Director Volumes, which depend on the geometric complexity of the scene and the variations in position of the key subjects. In the pictured example, the computation of suggestions takes around a second on a Intel Core i7 2.13 GHz

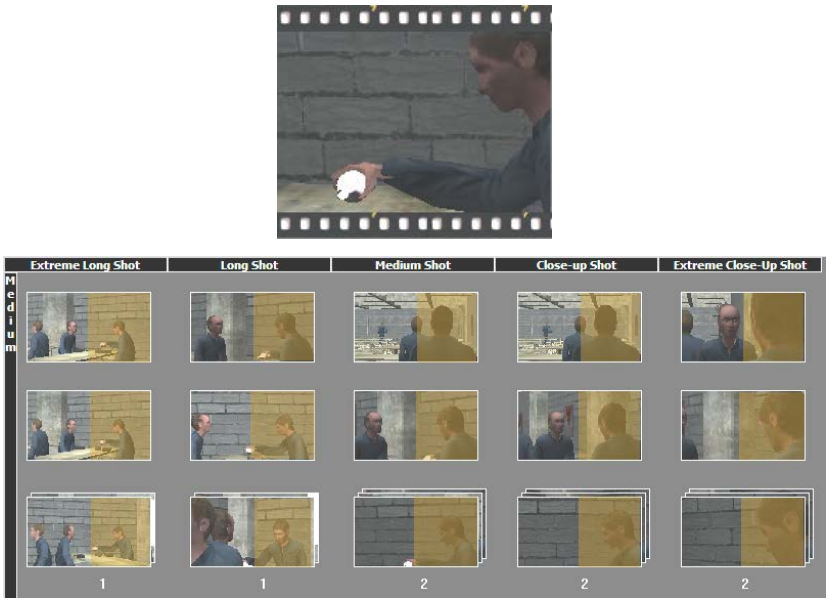


Figure 4.20 – Given user's composition for the current shot (shown in the left image), the system suggests shots (right image) which satisfy continuity rules of editing.

notebook, which is acceptable since this computation is not required to happen in real-time.

4.6.1 Applicability to Other Scenarios

The examples shown here feature mostly dialog events involving two or three characters, and also generic events (*e.g.* "Parsons walks to the table") for which proper suggestions are computed. In general, the system can model any event where one character or object, or group of objects (e.g. , cars driving, architectural elements) is involved, and propose suggestions to properly frame it using different angles and shot lengths. Note that for many situations (*e.g.* shooting of architecture, or crowds) there are not many cinematographic conventions other than using establishing shots first, and then closer shots. However, in general, the cinematography knowledge the system uses can be expanded by designing additional Director Volumes for specific cases or using the automatically learned tables of transitions. For situations that involve, in the same instant, more than three characters, the system could be extended by employing the method of hierarchical lines of action [KC08].

Scenes involving very fast action can be a bit problematic as the system might generate suggestions that are good for the considered instant, and decrease in quality after a short delay, for example because the framed character moves away or is covered by some moving objects. This kind of situation would require the system either to consider animations occurring in an interval of time, instead of an instant, or to suggest camera movements, instead of just starting frames. However, this would be of much greater computational cost than deriving suggestions for single instants. In these cases, however, the filmmaker can at worst ignore the suggestions and deal with the situation

by taking manual control.

Finally, the system computes all suggestions using a single fixed lens field of view. However, the user can quickly adjust the field of view by manual control. The system's search algorithm can be extended by taking into account variations on lens angle. In practice, the set of lens angles to consider would be limited to model the cinematography practice of using a small number of user-selected "prime" or standard lenses.

4.6.2 User Feedback

While we do not currently have results from formal and extensive user evaluations, a few videographers have tried the system, reporting great satisfaction in using it.

In general, users found the system workflow refreshing compared to mouse-based solutions, were very willing to try it, and could rapidly use it to produce results. In our tests, after a very brief introduction to the system, users familiar with camerawork could shoot a 3 minutes video of the 1984 scene in around 10-15 minutes, including taking decisions on camera angles and cuts. For comparison, producing a video for the same animation sequence took an expert user a few hours of work using 3DS Max. The reason for the huge difference in time comes from the fact that with typical animation software tools one is forced to a workflow where cameras have to be manually positioned and animated, and then the shot can be watched, and often one has to go back and forth between cameras modifications and testing the result. Furthermore, there's the complexity of manually keeping track of timings and motion curves, making sure that edits occur in the correct place and time.

While the use of a motion-tracked virtual camera is surely pivotal in reducing the amount of time needed, we can hypothesize that suggestions also played a role, as in general users needed to make very small movements to refine the suggestions and find their ideal starting camera positions. Moreover, from our feedback with professionals in the development of motion-tracked virtual cameras, we know that having to walk around in the tracked space to explore angles and distances is perceived as a real issue. Finally, users more experienced in real camerawork appreciated on-screen information overlays of character actions and their timing while shooting.

The interface for browsing suggestions by shot size and height was considered easy to learn and effective for a videographer, and the possibility of quickly visualizing many alternatives was very appreciated, though not all users agreed with each suggestion. This is expected since the perceived quality of a suggestion is a subjective factor, which changes from one videographer to another. However, users' attitude towards the cinematography engine work was positive, mainly because the system does not force one to any shooting or editing style, and suggestions can be ignored. However, many times, they caused the videographer to consider solutions he had not thought about, thus enhancing creative thinking and exploration.

While a few experiments have been done in combining human expertise with automatic camera planners, we believe this work is the first attempt at innovating the shooting process of CG animations, by providing intelligent and adaptive support in creating shots. In the shooting of CG movies, video game cinematic scenes and ma-

chinima, the system has the potential of making the process much easier and quicker than current methods. From our early tests, unlike existing automated systems which rely on pre-coded cinematic knowledge to cover all anticipated scenarios, our interactive approach proves effective even if no suggested viewpoint is "perfect" in the user's creative judgment.

5 Discussion and Conclusion

In this chapter, we have introduced a novel framework for virtual cinematography and editing which adds a cost-driven evaluation of the key editing components. We have further proposed two ranking-based approaches to editing a movie, that build upon the proposed evaluation metrics to assist a filmmaker in his creative process. We have introduced an efficient search strategy for finding the best sequence of shots from a large number of candidates generated from traditional film idioms, while providing the user with some control on the final edit. We further enable enforcing the pace in cuts by relying on a well-founded model of shot durations. We have then presented an interactive assistant whose result is a novel workflow based on interactive collaboration of human creativity with automated intelligence. This workflow enables efficient exploration of a wide range of cinematographic possibilities, and rapid production of computer-generated animated movies.

In the last chapters, we have relied on the concept of *Director Volumes* that provide a good support in representing and solving viewpoints corresponding to a given sub-set of visual composition elements. This space representation is however prone to some issues: (1) volumes remain expensive to compute, (2) they do not allow modeling all composition features, thus necessitating to re-evaluate computed viewpoints *w.r.t.* other features and (3) enforcing composition features while moving the camera is, as in most existing approaches, difficult to implement with this model. More generally, due to computational cost, all existing automated camera control techniques reduce the search of a camera configuration by first searching the camera position, then computing algebraically an orientation corresponding to this camera position. Because of this reduction, it is really difficult to find a camera configuration satisfying all composition features; and particularly the exact position of subjects on the screen, as this feature is strongly dependent of the camera orientation.

In the next chapter, we propose and investigate a novel and robust space representation which, in contrast to existing approaches, allows taking the whole camera configuration (position and orientation) into consideration when searching for a solution camera configuration.

An Efficient Approach to Virtual Camera Control: The Toric Space

5

As presented in our state of the art, a large range of computer graphics applications rely on the computation of viewpoints over 3D scenes that need to display a number of characteristic composition properties (*e.g.* the on-screen position, size, vantage angle or visibility of key subjects). Simplified versions have been tackled with straightforward vector algebra approaches (*e.g.* within an iterative technique [Bli88]) that compute an approximate solution. More expressive versions are generally casted as non-linear optimization problems in a 7 degree-of-freedom search space, *i.e.* computing the camera position, orientation and field of view, given a number of composition properties expressed and aggregated in a single viewpoint quality function which is then maximized. Given the size of the search space and the computational cost in the evaluation of composition properties (typically the visibility of subjects), this optimization process is a time-consuming task and hampers the use of evolved composition techniques in most applications.

An analysis of the problem and current solutions reveals a central issue. Aggregating quality functions for all the properties reduces the capacities to guide the solving process through the search space, therefore leading to techniques which explore large areas of the search space without solutions. The specific problem of viewpoint computation is transformed into a general search process for which it is difficult to propose efficient and general heuristics. In the context of a dynamic scene, one generally re-executes this search process at each time step. The use of optimization-based approaches therefore lead to variations in the output camera viewpoint, which causes camera stability issues.

In this chapter, we introduce a novel and efficient approach to virtual camera control. Based on a simple parametric model of the camera configurations space, we reduce the search space and consider both the camera position and orientation in the search. We first address an important visual property in virtual camera composition: the specification of exact on-screen coordinates at which key subjects should project. We express the solution space for each couple of subjects as a 2D manifold surface. We demonstrate how to use this manifold surface to solve Blinn's spacecraft problem [Bli88] with a straightforward algebraic approach, and extend the solution to three subjects. We secondly extend this parametric model as a novel 3D search space, in which most of classical visual properties can be easily expressed. The result is a robust and efficient approach which finds a wide range of applications in virtual camera control and more generally in computer graphics.

1 Contributions

The key contributions of this chapter are the following:

The Toric Manifold We provide a novel way to express, in a 2D manifold representation, the solution space of the exact on-screen positioning of two subjects. We then provide an algebraic formulation using this manifold which is the first to solve Blinn's spacecraft problem [Bli88] without resorting to an iterative technique. We further provide an expression of the on-screen positioning of three or more targets as a search in a 2D space rather than in a 6D space (as performed in most of the literature).

The Toric Space We provide of a novel 3D search space (that we refer to as *Toric Space*), as an extension of the Toric Manifold concept, in which most of classical visual properties can be expressed and combined. We show how to compute algebraic solutions, or a good approximation, to expressive composition problems (*e.g.* on-screen position, vantage angle, distance or projected size of subjects) in the Toric Space. Furthermore, an inherent properties of our Toric Space is the provision of a mean to dynamically enforce a set of visual properties on moving subjects and to perform live editing. Particularly, our work provides the first model that enables expressing, in a simple way, continuity editing rules in 3D.

Combination of expressive visual properties We propose an efficient algorithm to combine a set of expressive visual properties and compute a representative set of solution viewpoints in the Toric Space. As an extension, our algorithm provides a mean to detect potential issues in the application of visual constraints (*e.g.* a conflict between two or more constraints).

The chapter is organized as follows. Firstly, we show how the exact on-screen positioning of two subjects can be expressed in a 2D manifold. We show how Blinn's spacecraft problem [Bli88] can be solved by using this manifold, and how to extend to three (or more) subjects. Secondly, we show how this model can be extended to a 3D search space so as to tackle more expressive composition problems. We provide the expression of a range of classical visual properties in this space then show how to tackle the combination of constraints for a range of applications.

2 Reducing search space in virtual camera composition

By focusing on the problem of composing two subjects on the screen, our objective is to show that large regions of the search space can be discarded. For this purpose, we illustrate the quality of the on-screen positioning of two subjects as a 2D heat map. Figure 5.1 presents a 2D topview of a scene comprizing two subjects. Regions around the subjects are colored *w.r.t.* viewpoint quality (expressed as the distance between the desired and actual on-screen positions of subjects), from dark red (highest quality) to dark blue (lowest quality) through yellow and green (intermediate quality). Note how the region with highest quality is restricted to a small continuous portion of the whole

search space. In the following section we show that the entire solution set for the exact on-screen positioning problem can be determined with simple mathematical concepts, and that this model serves as a cornerstone for the resolution of more complex camera composition problems.

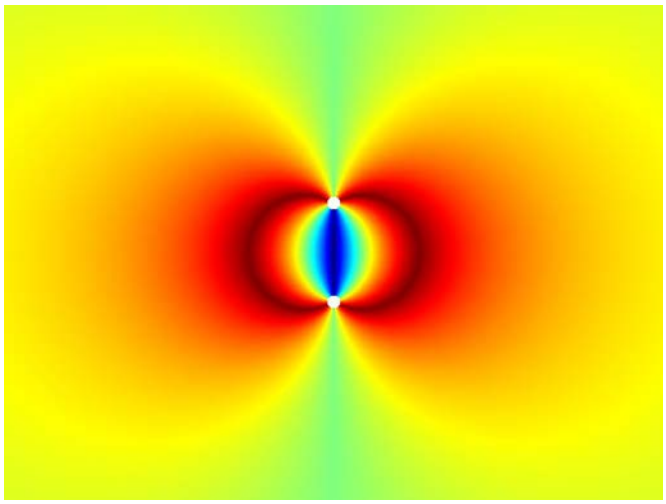


Figure 5.1 – Heat map representing the quality of on-screen composition for two subjects (white points) for a region of camera configurations (topview of a the 3D scene). Each colored point represents a camera configuration. Dark red areas are regions of good quality viewpoints (i.e. good compositions) while dark blue areas are regions of bad quality viewpoints. Green regions represent average quality viewpoints. Note that the best viewpoints are very local to a given curve around the subjects.

Definition. *Let two vectors* $\overrightarrow{u}$ *and* $\overrightarrow{v}$*. We denote the oriented angle between* $\overrightarrow{u}$ *and* $\overrightarrow{v}$ *as* $(\widehat{\overrightarrow{u}, \overrightarrow{v}})$ *and the non-oriented angle between* $\overrightarrow{u}$ *and* $\overrightarrow{v}$ *as* $\left|(\widehat{\overrightarrow{u}, \overrightarrow{v}})\right|$.

3 Tackling exact on-screen positioning of two subjects: The Toric Manifold

In this section, we show that the range of solutions for the exact composition of two subjects on the screen is a 2D manifold shaped as a spindle torus (we will refer to it as *Toric Manifold*) and that the manifold can be explored using two meaningful parameters θ and φ (see Figure 5.3). We then use this formulation to easily solve Blinn's spacecraft problem [Bli88], and show how to generalize the approach to three and more subjects.

3.1 Solution in 2D

We first provide an intuition of our model by studying the 2-subject on-screen positioning problem in 2D (*i.e.* in the plane). In this case the screen dimension is 1D.

The goal is to find a 2D camera position P such that two subjects of world positions A and B ($A \neq B$) project respectively at normalized screen positions $p_A, p_B \in [-1; +1]$, knowing the camera field of view ϕ. We here assume that A is the left-most subject (*i.e.* $p_A < p_B$).

Theorem. (Inscribed Angle Theorem) *The set of points P satisfying $(\overrightarrow{PB}, \overrightarrow{PA}) = \alpha$ is equal to the arc $\overset{\frown}{AB}$ (excluding A and B) of a circle $\mathcal{C}$ of center O passing through A and B, such that $(\overrightarrow{OB}, \overrightarrow{OA}) = 2\alpha$ (see Figure 5.2). The arc $\overset{\frown}{AB}$ then goes from A to B in the counterclockwise direction, and the radius r of $\mathcal{C}$ is equal to $\dfrac{AB}{2\sin\alpha}$.*

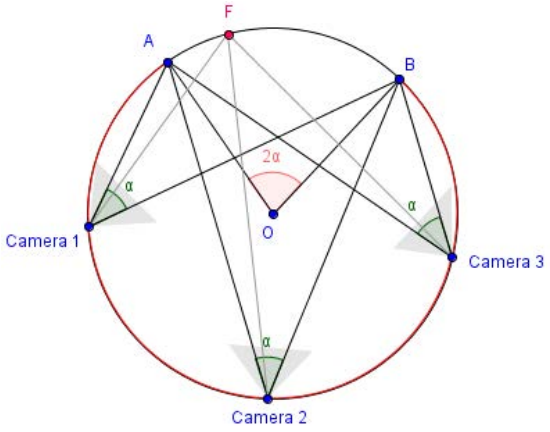


Figure 5.2 – The range of solutions for the exact composition of two subjects A and B on the screen is displayed in red. The set of points satisfying a given angle α is exactly the arc $\overset{\frown}{AB}$ of the inscribed circle centered on O. Moreover $(\overrightarrow{OB}, \overrightarrow{OA}) = 2\alpha$.

In our case, α is easily expressed from the on-screen distance between A and B and the fixed field of view ϕ. We introduce two points in the camera coordinate system, projecting respectively at p_A and p_B on the screen:

$$p_A^{cam}\left(\frac{p_A}{S}; 1\right) \quad \text{and} \quad p_B^{cam}\left(\frac{p_B}{S}; 1\right), \text{ with } S = \tan(\phi)$$

The solution of our 2D problem is then the arc $\overset{\frown}{AB}$ corresponding to $\alpha = (\overrightarrow{p_B^{cam}}, \overrightarrow{p_A^{cam}})$

To compute the direction vector $\overrightarrow{f}$ of our 2D camera, we also know to have

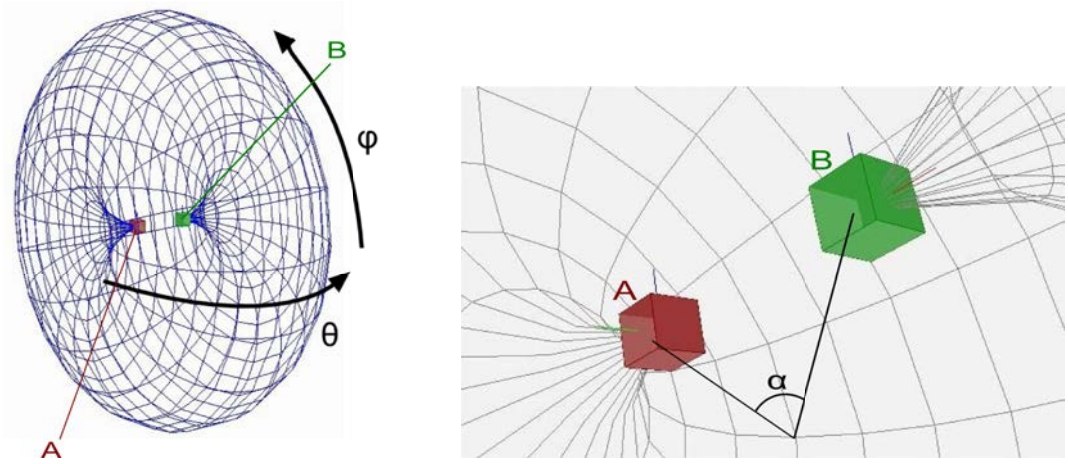
$$(\overrightarrow{PA}, \overrightarrow{f}) = \frac{\phi}{2} \cdot p_A \text{ and } (\overrightarrow{PB}; \overrightarrow{f}) = \frac{\phi}{2} \cdot p_B$$

By using the same theorem we know that the line of direction $\overrightarrow{f}$ and passing through P intersects the circle $\mathcal{C}$ at a point F such that $(\overrightarrow{OB}, \overrightarrow{OF}) = \phi \cdot p_B$, and is tangent to $\mathcal{C}$ when points P and F coincide. The camera orientation is thus algebraically determined from the camera position (see Figure 5.2).

As a conclusion, in 2D, the range of solutions to an exact on-screen positioning of two subjects is described with a single angular parameter $\theta \in]0\ ;\ 2(\pi - \alpha)[$. θ represents the angle $\widehat{(\overrightarrow{OP}, \overrightarrow{OB})}$(P being the position of the camera.

3.2 Solution in 3D

We now consider the same composition problem in 3D. The screen dimension is now 2D, and the goal is to find a 3D camera position P such that the two subjects A and B ($A \neq B$) project respectively at normalized screen positions $p_A\,(x_A; y_A)\,, p_B\,(x_B; y_B) \in [-1; +1]^2$, knowing the camera field of view ϕ and its aspect ratio a. Note that ϕ here represents the field of view on the x axis (representing the left-to-right direction on the screen). We also assume that A is the left-most key subject (*i.e.* either $x_A < x_B$, or $x_A = x_B$ and $y_A \neq y_B$).



(a) **(b)**

Figure 5.3 – Example of 3D solution set for 2 subjects. Subjects A and B are displayed resp. in red and green. (a) View of the solution set (a two-dimension parametric space (θ, φ)), for the on-screen projections of A and B, resp. $p_A\,(-0.33; -0.33)$ and $p_B\,(+0.33; +0.33)$. (b) The result obtained from the viewpoint parameterized with $\theta = \dfrac{\pi - \alpha}{2}$ and $\varphi = \dfrac{\pi}{6}$.

As previously, we introduce two points described in the camera coordinate system, projecting respectively at p_A and p_B on the screen:

$$p_A^{cam}\left(\frac{x_A}{S_x}; \frac{y_A}{S_y}; 1\right) \qquad p_B^{cam}\left(\frac{x_B}{S_x}; \frac{y_B}{S_y}; 1\right)$$

with $S_x = \tan(\phi)$ and $S_y = \dfrac{1}{a} \cdot \tan(\phi)$.

Remember the previous section. By considering a plane $\mathcal{P}$ defined by the two points A and B, together with a third point C (not lying on the line (AB)), the solution of our 3D problem on plane $\mathcal{P}$ is a 2D arc $\overset{\frown}{AB}$ corresponding to $\alpha = (\widehat{\overrightarrow{p_B^{cam}}, \overrightarrow{p_A^{cam}}}$(

We therefore have the solution of our problem for an arbitrary plane. Let's now consider that point C is the center of the inscribed circle in $\mathcal{P}$, that we previously called O. And let point I be the middle of segment AB. Note that, from the definition of the solution in 2D, we know to have $OI = \frac{AB}{2\tan\alpha}$. We build a plane $\mathcal{Q}$, defined by the point I and the normal vector $\overrightarrow{n} = \overrightarrow{AB}$. The set of possible positions for O is then the circle $\mathcal{C}_O$ of center I and radius $\dfrac{AB}{2\tan\alpha}$ defined in the plane $\mathcal{Q}$. This, together with the solution of our problem in an arbitrary plane containing A and B, defines a unique solution set for the camera position. This solution set is a 2-parameter manifold $\mathcal{M}$ shaped like a spindle torus, that we refer to as *Toric Manifold* (see Figure 5.3).

We now show how to compute the camera orientation that satisfies the composition, assuming a camera position $P \in \mathcal{M}$. We first construct an initial camera orientation (quaternion) q_i from an orthonormal frame made of three unit vectors: a forward (look-at) vector $\overrightarrow{f_i}$, a right vector $\overrightarrow{r_i}$, and an up vector $\overrightarrow{u_i}$. They are given by

$$\begin{aligned}
\overrightarrow{u_i} &= \overrightarrow{PB} \times \overrightarrow{PA} \quad \text{scaled to unit length} \\
\overrightarrow{f_i} &= \left(\frac{\overrightarrow{PB}}{\left\|\overrightarrow{PB}\right\|} + \frac{\overrightarrow{PA}}{\left\|\overrightarrow{PA}\right\|} \right) \quad \text{scaled to unit length} \\
\overrightarrow{r_i} &= \overrightarrow{f_i} \times \overrightarrow{u_i}
\end{aligned}$$

Note that $\overrightarrow{u_i}$ is normal to the supporting plane $\mathcal{P}$, and that $\overrightarrow{f_i}$ and $\overrightarrow{r_i}$ belong to $\mathcal{P}$.

The quaternion q_i here represents a "default" composition of A and B (*i.e.* $y_A = y_B = 0$ and $x_A = -x_B$). We then compute the rotation (quaternion) q_c such that, when applied to q_i, points A and B project at the appropriate locations on the screen (i.e. p_A^{cam} and p_B^{cam}). q_c is computed in a way similar to q_i, by replacing $\overrightarrow{PA}$ by $\overrightarrow{p_A^{cam}}$ and $\overrightarrow{PB}$ by $\overrightarrow{p_B^{cam}}$ in the formulas, then computing an orthonormal frame made of unit vectors f_c, r_c and u_c. The solution camera orientation q is finally given by

$$q = q_i \cdot (q_c)^{-1} \tag{5.1}$$

Note that q_i and q_c are algebraically determined from respectively the camera position, and the desired on-screen composition.

As a conclusion, in 3D, the range of solutions (6D camera configurations) to the exact on-screen positioning of two subjects is described with two angular parameters $\theta \in]0\ ;\ 2(\pi - \alpha)[$ and $\varphi \in]-\pi\ ;\ +\pi]$. φ represents the angle (in the circle $\mathcal{C}_O$) defining the point O; thus defining the supporting plane $\mathcal{P}$. As previously, θ then represents the angle $\widehat{(\overrightarrow{OP}, \overrightarrow{OB})}$ in this plane.

In other words, to each 2D point on our Toric Manifold corresponds a unique 6D camera configuration (position and orientation) satisfying this exact 2-subject on-screen positioning. The world (Cartesian) coordinates of a given viewpoint $P(\theta, \varphi)$ of our Toric manifold can then be computed in the following way. First, we define the reference position O_0, that we consider as the center of the inscribed circle for $\varphi = 0$, such that

$\overrightarrow{IO_0}$ has a z component equal to 0. Second, we compute the vector $\overrightarrow{n_0}$ normal to the corresponding supporting plane $\mathcal{P}_0$

$$\overrightarrow{n_0} = \overrightarrow{O_0A} \times \overrightarrow{O_0B} \text{ scaled to unit length}$$

Third, we build two quaternions q_θ and q_φ as the rotations resp. of θ radians around the axis $\overrightarrow{n_0}$ and of φ radians around the axis $\overrightarrow{AB}$ scaled to unit length. The position of $P(\theta, \varphi)$ on the manifold is then given by

$$\overrightarrow{P} = \overrightarrow{I} + q_\varphi \cdot \overrightarrow{IP_0} \tag{5.2}$$

with $P_0(\theta, 0)$ (belonging to $\mathcal{P}_0$) such that

$$\overrightarrow{IP_0} = \overrightarrow{IO_0} + q_\theta \cdot \overrightarrow{O_0B}$$

In the following sections, we show how our model is applied to the resolution of exact composition problems.

3.3 Application #1: Solution of Blinn's spacecraft problem

The problem presented by Blinn [Bli88] is similar to the following. Where to put the camera and how to set its orientation knowing (i) the result on-screen positions of two key subjects A and B, (ii) the distance d between the camera and the subject B and (iii) a given vector (a non-null vector $\overrightarrow{v}$ starting from B) that must appear vertical and pointing "up" on the screen? An illustration of this problem is given in Figure 5.6. To our knowledge, we here propose the first general technique to compute the exact solution (or range of solutions) to this on-screen composition problem.

On-screen positioning constraint We can already state that the solution of Blinn's problem belongs to a 2D manifold as defined in Section 3.2.

Distance constraint We here make the assumption that α is lower than $\pi/2$; we do not consider wide angle views, but the solution(s) for $\alpha \geq \pi/2$ can be computed with a similar resolution. By using the analytic solution of the on-screen positioning problem in 2D (Section 3.1), the distance constraint is quite simple to resolve. Indeed, let's define a point B' as the symmetric point to B *w.r.t.* the center point O of the inscribed circle. We will use the properties between angles and edges in the triangle BPB' which is rectangle in P. Let's then define the angle β as

$$\beta = \widehat{(\overrightarrow{BP}, \overrightarrow{BB'})} = \frac{1}{2}\widehat{(\overrightarrow{OP}, \overrightarrow{OB'})} = \cos^{-1}\left(\frac{d}{2r}\right)$$

Then we can state that there are 0, 1 or 2 solutions to this distance problem:

- no solution if $d = 0$ or $d > 2r$
- a single solution if $d \in]0\ ;\ AB]$: $\theta = \pi - 2\beta$

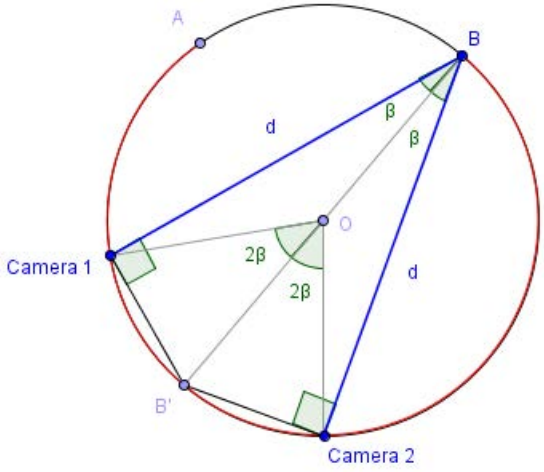


Figure 5.4 – Illustration of the method used to solve a distance-to-B constraint. The distance BP (P being the camera position) is d meters. The segment BB' is a diameter of the inscribed circle of radius r. The angle $\beta = \cos^{-1}\left(\frac{d}{2r}\right)$ corresponds to $\widehat{(\overrightarrow{BP}, \overrightarrow{BB'})} = \frac{1}{2}\widehat{(\overrightarrow{OP}, \overrightarrow{OB'})}$. The solution for $\widehat{(\overrightarrow{OP}, \overrightarrow{OB})}$ is then $\theta = \pi \pm 2\beta$, with the restriction that P must lie on the arc $\widehat{AB}$ (in red).

- a single solution if $d = 2r$: $\theta = \pi$
- two solutions if $d \in]AB\ ;\ 2r[$: $\theta = \pi \pm 2\beta$

The resolution method is illustrated in Figure 5.4. Each solution θ_n in 2D works in 3D since they are not dependent of the value of the parameter φ (vertical angle). Such a solution to the distance constraint can therefore be viewed as a circle $\mathcal{C}_n$ lying on the Toric Manifold; this circle is such that every point of the circle is at a distance d to B, *i.e.* this is the set of points $P(\theta, \varphi)$ such that $\theta = \theta_n$ on the manifold.

"Up" vector constraint In the following, we will assume that the position of the camera satisfying all the composition constraints is P. As stated above, P belongs to a circle $\mathcal{C}_n$ defined by the parameter $\theta = \theta_n$ (solution to the distance constraint). We then search for the appropriate value of φ (if one value exists) such that $P(\theta_n, \varphi)$ is solution to this last constraint.

Let us consider $\overrightarrow{u}$ the up vector of the camera (computed from the quaternion q described in equation Equation 5.1) at position P, and $\overrightarrow{w}$ the vector representing $\overrightarrow{PB}$. Solving Blinn's problem corresponds to finding a position P on the circle $\mathcal{C}_n$ such that the two following constraints are satisfied.

1. the vector $\overrightarrow{v}$ must appear vertical on the screen. This involves that $\overrightarrow{u}$, $\overrightarrow{w}$ and $\overrightarrow{v}$ are coplanar.
2. $\overrightarrow{v}$ must point "up" on the screen. This implies $\overrightarrow{u} \cdot \overrightarrow{v} > 0$.

From Equation 5.2, we know that the camera configuration (*i.e.* its position and orientation) is defined as a rotation of a reference configuration, in which the camera

position is $P_{n,0}(\theta_n, 0)$ belonging to the circle $\mathcal{C}_n$, by an angle of φ radians around the axis $\overrightarrow{AB}$. Our problem is then similar to finding the rotation of φ radians around the axis $\overrightarrow{BA} = -\overrightarrow{AB}$ (scaled to unit length) to apply to the vector $\overrightarrow{v}$ such that it appears vertical and points "up" on the screen when the camera position is set to $P_{n,0}$. Instead of searching the camera position P on the circle $\mathcal{C}_n$, we build a circle $\mathcal{C}$ representing the range of possible positions for the extremity of the vector $\overrightarrow{v}$, and look for a position on $\mathcal{C}$ such that the previous property is satisfied.

Before solving this dual problem, let us define three new vectors: $\overrightarrow{u_{n,0}}$ the up vector of the camera when at $P_{n,0}$, $\overrightarrow{w_{n,0}}$ the vector $\overrightarrow{P_{n,0}B}$, and $\overrightarrow{v_0}$ the vector $\overrightarrow{v}$ rotated by an angle of φ radians around $\overrightarrow{BA}$ (*i.e.* the solution vector). We then solve the problem as follows.

We first define a plane $\mathcal{P}$ by the point $P_{0,n}$ and a normal vector $\overrightarrow{n} = \overrightarrow{u_{0,n}} \times \overrightarrow{w_{0,n}}$ (scaled to unit length). By definition, B belongs to $\mathcal{P}$. We then define a half-plane $\mathcal{P}_\mathcal{H} \subset \mathcal{P}$ delimited by the line $\mathcal{L}$ of direction $\overrightarrow{w_{n,0}}$ passing through $P_{n,0}$. The points belonging to $\mathcal{P}_\mathcal{H}$ are located in the direction of $\overrightarrow{u_{n,0}}$. This half-plane defines the set of vectors starting from B that appear vertical and pointing "up" on the screen when the camera is positioned at $P_{n,0}$.

We then define an other plane $\mathcal{Q}$ by a normal vector $\overrightarrow{t} = \overrightarrow{AB}$ (scaled to unit length) and a point F given by $\overrightarrow{F} = \overrightarrow{B} + \overrightarrow{t} \cdot |\overrightarrow{v}| \cdot \cos(\pi - \gamma)$, where $\gamma = \widehat{(\overrightarrow{t}, \overrightarrow{v})}$ (*NB:* F belongs to the line (AB)). We then define the circle $\mathcal{C}$ as the circle of center F and radius $r' = |\overrightarrow{v}| \cdot \sin(\pi - \gamma)$ in the plane $\mathcal{Q}$.

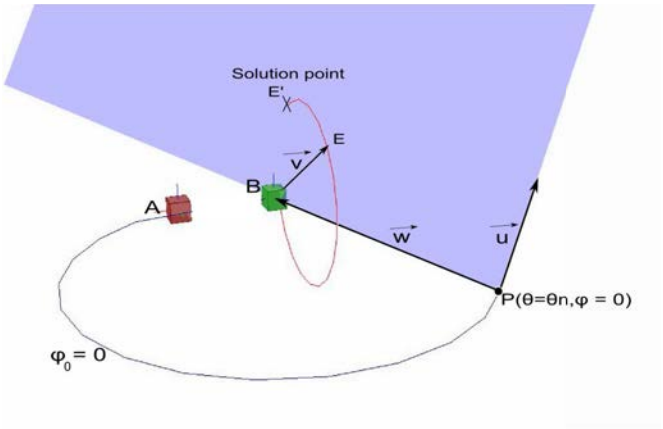


Figure 5.5 – Resolution of Blinn's spacecraft problem. The vector (in black) starting from B must appear vertical and pointing "up" on screen. Subjects A and B are drawn resp. in red and green. The camera position P is a solution to the distance constraint (see Figure 5.4), corresponding to a given value $\theta = \theta_n$, and located at a given vertical angle ($\varphi_0 = 0$) of the manifold. The red circle represents the appearance of $\overrightarrow{v}$ depending on the vertical angle φ applied to the viewpoint P. The half-plane (in blue) represents the set of vectors starting from B that appear vertical and pointing "up" on screen ($\overrightarrow{u}$ is the up vector of the camera when at P, and $\overrightarrow{w}$ represents $\overrightarrow{PB}$). The solution of Blinn's problem is given by the intersection (E') of the half-plane and the circle, and provides a mean to compute the rotation φ to apply.

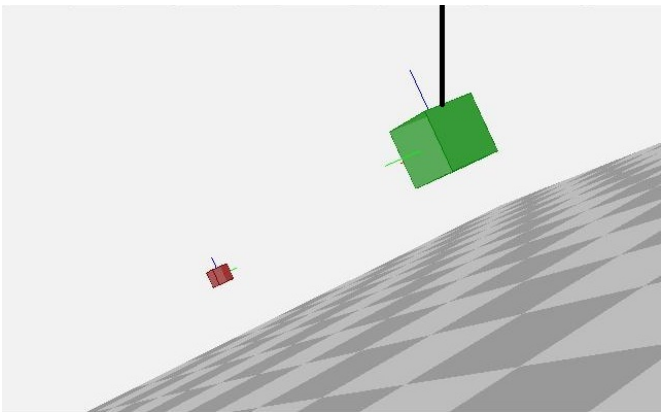


Figure 5.6 – Example solution of Blinn's spacecraft problem. Subjects A and B are drawn respectively in red and green. The camera must satisfy desired exact on-screen positions of A and B, and be d meters away from subject B. An additional constraint is that the vector (in black) starting from B must appear vertical and pointing "up" on screen.

The solution value of φ is finally given by using the intersection(s) of the half-plane $\mathcal{P}_{\mathcal{H}}$ and the circle $\mathcal{C}$. In other words, we compute the solution in a 3-step process:

1. we compute the intersection $\mathcal{I}$ of $\mathcal{P}_{\mathcal{H}}$ with $\mathcal{Q}$ (none, a half-line or the half-plane $\mathcal{P}_{\mathcal{H}}$ itself);
2. we compute the intersection(s) E' of $\mathcal{I}$ with the circle $\mathcal{C}$ in the plane $\mathcal{Q}$;
3. the vertical angle φ that is solution to our problem is then given by $\varphi = \overrightarrow{(\overrightarrow{FE'}, \overrightarrow{FE})}$(

Figure 5.5 provides an illustration of this geometric resolution, and Figure 5.6 provides an example of solution computed with this method.

By using our method, the overall Blinn's spacecraft problem (on-screen positioning, distance and "up" vector constraints) is solved algebraically in about 0.01ms per frame.

3.4 Application #2: Solution for three or more subjects

We now take an interest in the problem of finding a camera configuration satisfying the exact on-screen positioning of three or more subjects (abstracted as points).

This could be viewed as a PnP (perspective n-points) problem. PnP problems estimate the position (and sometimes the orientation) of an image sensor given the obtained on-screen projection of a well known pattern composed by n points. For instance, some techniques have been proposed that use analytic resolutions for problems involving a given number of points (*e.g.* P3P or P4P problems) [FB81, Hor87, HCLL89, DRLR89] or an iterative resolution for the more general PnP problem [DD95]. However, in PnP problems, one make the strong assumption that at least one solution camera configuration exist. Moreover, such resolutions generally extend the formulation of Fischler and Bolles [FB81] which involves the resolution of a non-linear system. This

requires a lot of computations and thus prevents such resolution techniques from being usable in interactive applications.

In opposition to PnP problems, we here start from a desired (but not necessarily feasible) on-screen positioning of independent 3d world points (*i.e.* subjects). As a consequence, there is no guarantee (at least for more than three subjects) that an exact solution camera configuration exist. Moreover, in our case we do not need the overall on-screen positioning to be exact. Actually, in a cinematic context, there are commonly one or two main subjects which the director wants to position precisely on the screen and a set of secondary subjects that can be positioned in an more flexible way. In order to account for these two preconditions, in the following, we propose a novel and efficient method based on our manifold concept to estimate a camera viewpoint for the "cinematic" on-screen positioning of three or more subjects.

Notice that the on-screen positioning approach we have presented for two subjects is also extensible to three or more subjects. Indeed, let consider the case of three subjects with world positions A, B, C (different from each other). As stated earlier, the solution set for each pair of subjects is a 2D manifold. Consequently, the camera configuration P satisfying this 3-subject composition must verify two conditions:

- the camera position P must be at the intersection of three manifolds $\mathcal{M}$, $\mathcal{M}'$ and $\mathcal{M}''$ respectively defined by couples (A, B), (B, C) and (A, C);
- the camera orientation corresponding to P must match for this three manifolds.

This defines three angles α, α' and α'' corresponding to resp. $\mathcal{M}$, $\mathcal{M}'$ and $\mathcal{M}''$. Similarly, we define three quaternions q, q' and q'' as the orientations (defined as unit quaternions) at a given 3D point P, as defined respectively on $\mathcal{M}$, $\mathcal{M}'$ and $\mathcal{M}''$. If for instance P belongs to $\mathcal{M}$, then it verifies the following manifold equation:

$$\widehat{(\overrightarrow{PB}, \overrightarrow{PA})} - \alpha = 0 \tag{5.3}$$

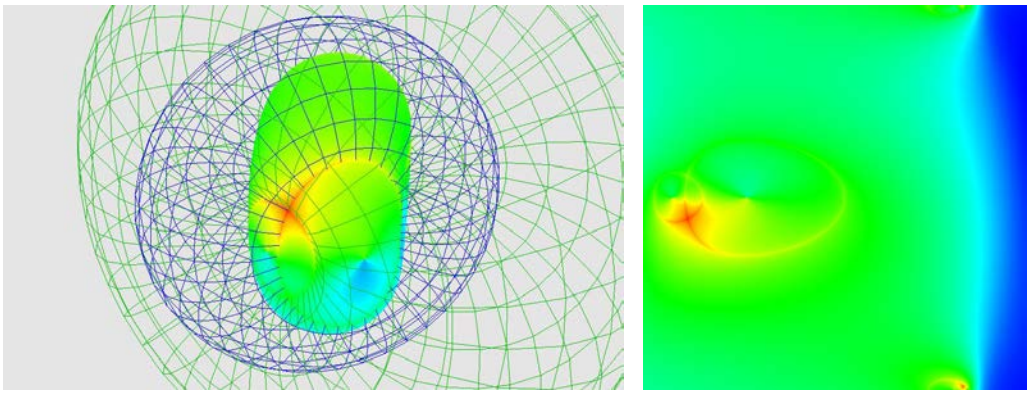
We here propose an algorithm which searches, on the surface of one manifold $\mathcal{M}$, the configuration which maximizes the viewpoint quality (defined as a distance to the other manifolds). Typically the on-screen position of two subjects is fixed, and we optimize the on-screen position of the third subject. The cost function to minimize is then

$$d \min_{\theta,\varphi} \left(\left| \widehat{(\overrightarrow{PC}, \overrightarrow{PB})} - \alpha' \right| + \left| \widehat{(\overrightarrow{PC}, \overrightarrow{PA})} - \alpha'' \right| + \left| 1 - \langle q', q'' \rangle^2 \right| \right)$$

where $\langle q', q'' \rangle$ denotes an inner product of q' and q'', providing a measure of the angle between them ranging from -1 (opposite orientations) to $+1$ (same orientation). This cost function is illustrated with a heat map in Figure 5.7. The solutions of an example 3-subject composition problem are displayed in Figure 5.8.

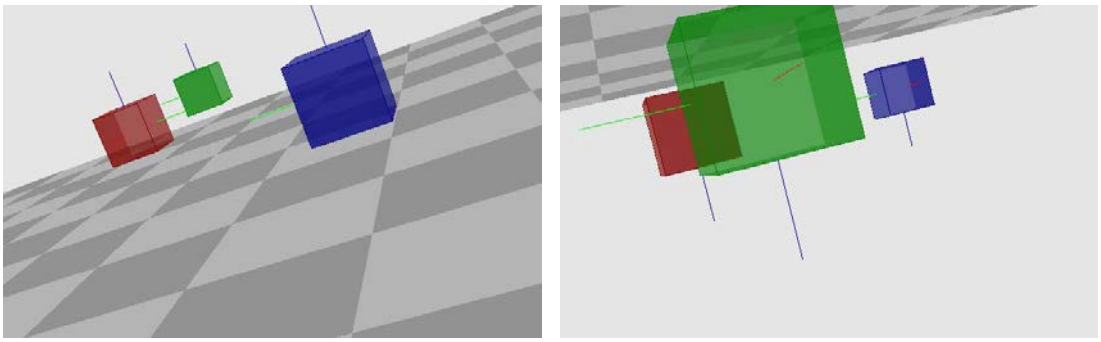
The main advantage of this technique is that it can easily extend to more than three subjects. Though the composition may not be strictly satisfied, it enables fixing the on-screen position of two (main) subjects, then optimizing the position of other subjects.

We used this method on both a 3-subject and a 4-subject on-screen positioning problem. In our test, we approximated the 3-subject exact composition in about 0.10ms per frame and we minimized the 4-subject composition error in about 0.42ms per frame.



(a) **(b)**

Figure 5.7 – Heat map drawn on one of the manifolds. The cost function represents the sum of the distance to the two other manifolds. Red: points close to both other manifolds (drawn in blue and green), thus verifying the on-screen positioning of the 3 subjects. Blue: points farest to both other manifolds. Green: intermediate-distance points. (a) 3D view of the manifolds and of the heat map; (b) view of the heat map when the manifold is unfolded.



(a) **(b)**

Figure 5.8 – Two solutions of the exact on-screen positioning of three subjects A (red), B (green) and C (blue). Their respective on-screen positions are $p_A(-0.50; +0.25)$, $p_B(-0.25; +0.50)$ and $p_C(+0.25; +0.50)$. These solutions correspond to the intersection of three solution set (described in Figure 5.7), each solving the exact on-screen positioning of two of the subjects.

In this section, we investigated the algebraic solution to the exact two-subject on-screen positioning problem. We showed that it corresponds to a 2D manifold surface. We showed how, by using this manifold, to solve Blinn's spacecraft problem without

resorting to an iterative algorithm; and how the on-screen positioning problem for three or more subject can be cast into a search in 2D. In the next section, we investigate more expressive composition problems in which a degree of relaxation on constraints is necessary so as not to restrict to much the possibility of finding an overall solution to the problem (by avoiding conflicts between constraints as much as possible).

4 Tackling more evolved on-screen composition problems: The Toric Space

In this section, we extend the concept of *Toric Manifold* to a 3-dimensional search space $(\alpha, \theta, \varphi)$ that we will refer to as *Toric Space*. This space represents the set of Toric Manifolds that may be generated around a pair of subjects (A, B). The parameter α is the manifold generator (which depends on both the field of view and the aspect ratio of the camera), and the pair (θ, φ) represents a unique position on a given manifold. In the following we will show that a number of classical constraints related to virtual camera composition are easier to manipulate in our Toric Space (3 angular parameters) than in the classical 6D camera space (3 Cartesian coordinates and 3 Euler angles). Illustration of Toric Manifold and Toric Space are given resp. in Figure 5.9 and Figure 5.10.

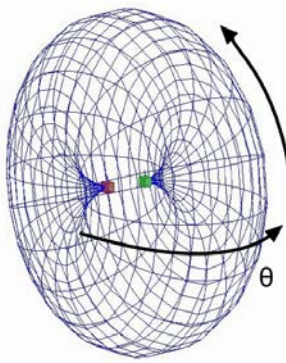


Figure 5.9 – Toric Manifold. To a given on-screen positioning of two subjects (red an green boxes) corresponds a unique angle α that generates such a manifold. Any point of this 2D manifold then represents a 6D camera configuration (position and orientation) satisfying this exact on-screen positioning.

Note that imposing conflicting constraints can prevent from finding an appropriate camera configuration, satisfying a set of exact visual features. Consequently, we hereafter consider composition problems in which there is a degree of relaxation in the application of constraints. In our approach the relaxation is represented as a range of accepted values for each feature.

In the next section, we compute solutions to constraints that are independent of the geometry of both the surrounding environment and of subjects (*e.g.* vantage angle, distance, on-screen positioning), together with good approximations to constraints such as visibility or projected size of subjects. We will then show how, by using interval analysis techniques, such computations in the Toric Space greatly improve the

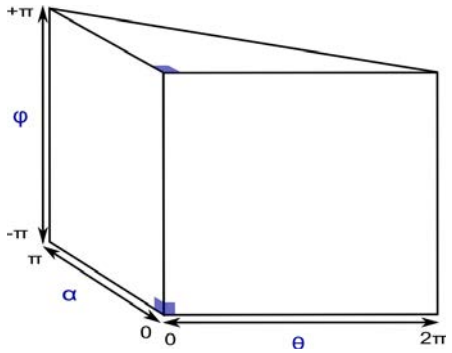


Figure 5.10 – Representation of our Toric Space (3D extension of the Toric Manifold concept) in an orthogonal coordinate system. Here, there are however two singularities: (1) the two planes $\varphi = \pm\pi$ (top/bottom) are actually indistinguishable, and (2) the three planes $\theta = 0$, $\alpha = 0$, and $\alpha = \pi - \frac{\theta}{2}$ (sides) represent the set of forbidden configurations.

placement of a virtual camera in a range of computer graphics applications.

5 Expressing classical visual constraints in the Toric Space

We here show how we express visual constraints in the Toric Space as 2D solution sets. In our process, two planes of the Toric Space appear useful: the plane (θ, α) and the plane (θ, φ). Note that a 2d solution set in the plane (θ, α) do not depend on the value of φ. Conversely, a solution set in the plane (θ, φ) will strongly dependent of the value of α. Indeed, there is a straightforward relationship between the parameter α and the maximum value the parameter θ can take:

$$0 < \theta < 2(\pi - \alpha)$$

We here introduce two additional angular parameters: $\beta, \beta' \in [0, \pi]$ (see Figure 5.11). β represents the angle between the vector $\overrightarrow{AB}$ and a vector $\overrightarrow{a}$ whose origin is A and destination is P. In the same way, β' represents the angle between the vector $\overrightarrow{BA}$ and a vector $\overrightarrow{b}$ whose origin is B and destination is P. As illustrated in Figure 5.11, in 2D, iff $(\beta + \beta') \in]0, \pi[$ then the line passing through A and of direction $\overrightarrow{a}$ and the line passing through B and of direction $\overrightarrow{b}$ intersect on the manifold generated by the angle $\alpha = \pi - (\beta + \beta')$. Moreover the coordinate θ corresponding to the intersection point on this manifold is given by (inscribed angle theorem)

$$\theta = 2\beta = 2(\pi - \alpha - \beta') \tag{5.4}$$

For practical reasons, we will then first solve a number of constraints related to respectively A or B in a virtual plane (β^*, φ), where β^* represents resp. β or β'. Second, we will extend the resulting solution into the plane (θ, φ) by using the Equation 5.4, then combine the computed solutions for both A and B (an example of such combination is illustrated in Figure 5.18).

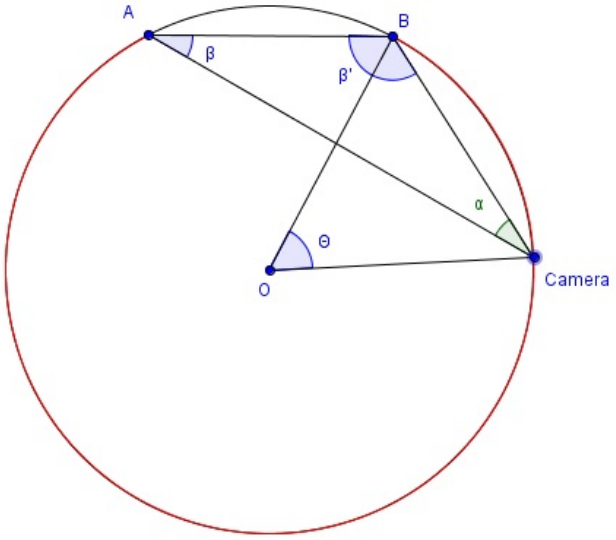


Figure 5.11 – There is a straightforward relationship between the parameter α and the parameter θ ($0 < \theta < 2(\pi - \alpha)$). In the same way, if we consider one line passing through A and of direction a vector $\overrightarrow{a}$ (making an angle β with $\overrightarrow{AB}$), and one line passing through B and of direction a vector $\overrightarrow{b}$ (making an angle β' with $\overrightarrow{BA}$), then iff $(\beta + \beta') \in]0, \pi[$ then these two lines intersect on the manifold generated by the angle $\alpha = \pi - (\beta + \beta')$. Furthermore, the coordinate θ of this intersection is given by $\theta = 2\beta = 2(\pi - \alpha - \beta')$.

5.1 On-screen positioning

We here consider a more flexible positioning constraint for both subjects. We want each subject (A and B) to project in a known convex shape expressed in the screen-space, that we will refer to them resp. as s_A and s_B (see Figure 5.13c). We then look for a set of camera configurations (position and orientation) that satisfies this constraint. For practical reasons, in the following, we will handle the camera position and orientation separately.

Camera position At this point, we first use a scanning algorithm on both shapes. We sample points on both shapes, then compute the angle α (with our previous method, presented in Section 3.2) for every couple of a point on s_A and a point on s_B. Since s_A and s_B are convex shapes, this allows computing a range $r_\alpha = [\alpha_{\min}, \alpha_{\max}]$ of possible angles α satisfying the constraint of having one subject projecting inside s_A and the other subject projecting inside s_B. The solution of the relaxed 2-subject on-screen positioning problem therefore corresponds to a horizontal strip in the plane (θ, α), as shown in Figure 5.12.

Camera orientation We here consider the camera position is set to $P_T(\alpha, \theta, \varphi)$, computed so as to satisfy the previous condition ($\alpha \in r_\alpha$). For a given angle $\alpha \in r_\alpha$,

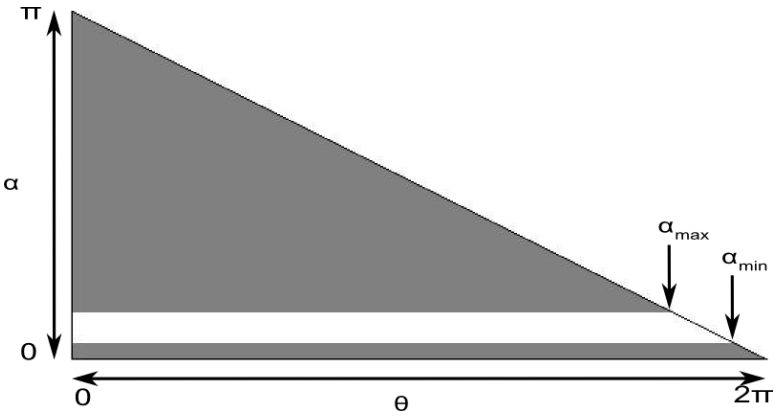


Figure 5.12 – The on-screen positioning of two subjects inside convex shapes corresponds to a horizontal strip $\alpha \in [\alpha_{\min}, \alpha_{\max}]$ in the plane (θ, α).

there are however an infinite number of orientations, satisfying the on-screen positioning of A and B or not. We here propose a 3-step method to compute an appropriate camera orientation.

Step 1: we compute a first orientation q_1 that guarantees the satisfaction of the on-screen positioning constraint. We first consider two pairs of points $P^{\min}(p_A^{\min}, p_B^{\min})$ and $P^{\max}(p_A^{\max}, p_B^{\max})$. They correspond to the points of s_A and s_B that respectively generate the angles $\alpha_{\min}$ and $\alpha_{\max}$. We then define two points, p_A^C and p_B^C, as the barycenter of resp. s_A and s_B. We finally use a dichotomy algorithm to search for appropriate on-screen positions p_A and p_B in resp. s_A and s_B. We consider $P^{\min}$ as minimum value, $P^{\max}$ as maximum value and $P_i(p_A^C, P_B^C)$ as initial value. The dichotomous process is stopped when the angle get close enough to the subject value α. An example of computed orientation is given in Figure 5.13a.

Step 2: we compute the orientation q that minimizes the roll angle of the camera. To this end, we first construct a "default" orientation q_d (*i.e.* such that $p_A = -p_B$) from an orthonormal frame, made of three unit vectors $\overrightarrow{r}$ (right vector), $\overrightarrow{f}$ (forward vector) and $\overrightarrow{u}$ (up vector). $\overrightarrow{f}(x_f, y_f, z_f)$ is computed as follows:

$$\overrightarrow{f} = \frac{\overrightarrow{P_W A}}{\left\|\overrightarrow{P_W A}\right\|} + \frac{\overrightarrow{P_W B}}{\left\|\overrightarrow{P_W B}\right\|} \text{ scaled to unit length}$$

where P_W is the world position corresponding to P_T. We then define a vector $\overrightarrow{f'}(x_f, y_f, 0)$ as the projection of $\overrightarrow{f}$ along the z direction. The right vector is then defined as $\overrightarrow{r} = q_r \cdot \overrightarrow{f'}$, where q_r is the rotation of $-\frac{pi}{2}$ radians around the axis $\overrightarrow{z}(0, 0, 1)$; and the up vector is defined as $\overrightarrow{u} = \overrightarrow{r} \times \overrightarrow{f}$. We secondly compute the rotation to apply so as to have the camera viewpoint centered on the barycenter of shapes s_A and s_B. We define two points, $P_O(0, 0)$ the origin of the screen and $p_M(x_M, y_M)$ the middle

point between p_A^C and p_B^C. We construct two 3d points in the camera coordinate system: p_O^3 and p_M^3, representing points that project resp. at p_O and p_M on the screen.

$$p_O^3(0,0,1) \text{ and } p_M^3\left(\frac{x_M}{S_x}, \frac{y_M}{S_y}, 1\right) \text{ scaled to unit length}$$

We then define q_m as the rotation of angle $\left|(\overrightarrow{p_O^3}, \overrightarrow{p_M^3})\right|$ around the axis $\overrightarrow{m} = \overrightarrow{p_M^3} \times \overrightarrow{p_O^3}$. The orientation q_2 of the camera is then built as $q_2 = q_d \cdot q_m$. An example of computed orientation is given in Figure 5.13b. Note that this orientation does not necessarily guarantee that A and B project resp. in s_A and s_B.

Step 3: we combine the two previously computed orientation, so as to get a camera orientation q that guarantees A and B to project in s_A and s_B, while minimizing the roll angle of the camera. To do so, we use a dichotomous process between q_1 and q_2 (through the use of the "Slerp" function to compute an interpolated orientation). The dichotomous process is stopped when the current positions p_A and p_B belong to resp. s_A and s_B. An example of computed orientation is given in Figure 5.13c.

The advantages of our method are fourfold: (i) it provides a fast orientation computation, (ii) a unique solution orientation can be computed for any position in the solution set – in particular, our computation algorithm prevents camera jerkiness –, (iii) the computed orientation minimizes the camera roll angle and (iv) it preserves continuity of the camera orientation in the space of possible camera positions.

5.2 Projected Size

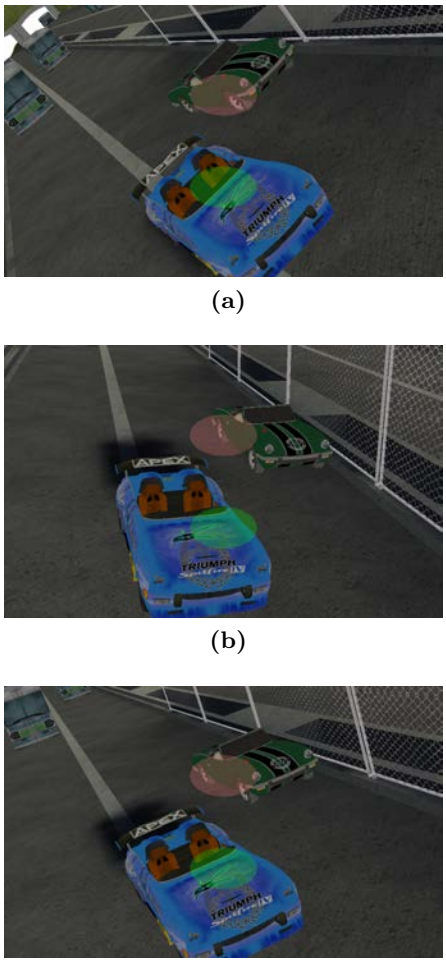
We here consider a constraint on the projected size (size in terms of ratio of the screen area) of a subject, as an interval of accepted sizes $[s_{min}, s_{max}]$. We then search the set of camera configurations for which the projected size of the subject belongs to this interval. Note that the solution set of a projected size is one of the most complicated to solve. Indeed, such a solution set is strongly dependent of the shape of the subject. Consequently, to our knowledge, previous work has mainly focused on sampling-based methods in a 6D search space. We here propose to compute an approximated solution set, through a 2-step computation. To do so, we first make a strong assumption: we consider that the subject is abstracted as a bounding sphere $\mathcal{S}$ of a given radius r.

5.2.1 Naive resolution

We here assume the camera is set at a given position and is oriented so that the subject is centered at the origin of the screen. The projection of the sphere $\mathcal{S}$ on the screen corresponds to an ellipse of parameters a and b, computed as follows:

$$a = \frac{r \cdot S_x}{d} \text{ and } b = \frac{r \cdot S_y}{d}$$

where d represents the distance between the camera and the subject. The projected size of the subject (*i.e.* the area of the ellipse in terms of frame ratio, the frame being



(a) **(b)** **(c)**

Figure 5.13 – Computation of the camera orientation for a given camera position. The on-screen positioning of the two subjects is here expressed as two convex areas (red and green ellipses) of the screen. (a) step 1: we compute a first orientation that guarantees that the center of each subject projects in the proper area; (b) step 2: we compute algebraically the orientation that minimizes the roll angle of the camera ($\psi = 0$, see Figure 2.6); (c) step 3: by using a dichotomy between these two orientations, we compute the final camera orientation that satisfies the on-screen positioning constraint while minimizing the roll angle of the camera.

of area 4) is then given by

$$4s = \frac{\pi \cdot r^2 \cdot S_x \cdot S_y}{d^2}$$

Starting from this formula, we then compute the reverse solution, which provides the distance d at which to position the camera depending on the projected size of the abstracted subject

$$d = r \sqrt{\frac{\pi \cdot S_x \cdot S_y}{4s}}$$

To consider the range of projected size $[s_{min}, s_{max}]$, one must then consider the corresponding approximate range of distance $[d_{min}, d_{max}]$, where the distances d_{min} and d_{max} correspond resp. to projected sizes s_{max} and s_{min}. This calculation provides a naive resolution of the interval $[d_{min}, d_{max}]$. With this interval, we can reduce the range of possible values for the parameter θ; this can further be applied for both subjects. The projected size is however dependent of on-screen position of the subject.

5.2.2 A more accurate approximation

We here consider that the subject projects at position $p(x, y)$ on the screen (for instance a position belonging to s_A or s_B). The projection of the sphere on the camera plane (*i.e.* before applying the projection component) corresponds to an ellipse of parameters a and b, computed as follows:

$$a = \frac{\sin\gamma\cos\gamma}{|\cos^2\lambda - \sin^2\gamma|} \text{ and } b = \frac{\sin\gamma}{\sqrt{|\cos^2\lambda - \sin^2\gamma|}}$$
$$\text{with } \cos\lambda = \left(\sqrt{\frac{x^2}{S_x^2} + \frac{y^2}{S_y^2} + 1}\right)^{-1} \text{ and } \gamma = \text{ atan}\left(\frac{r}{d}\right)$$

The projected size of the subject is then given by $s = (\pi \cdot a \cdot b) \div \frac{4}{S_x \cdot S_y}$ where the left-most part represents the area of the ellipse, and the right-most part represents the area of the rectangle representing the frame limits on the camera plane.

The distance d at which to position the camera can then be solved by using a dichotomous search. We start the search from a close neighborhood of the value computed with the naive resolution process, and we stop the search when the error on the size gets under a given threshold ϵ.

We have proposed a method to convert a range $[s_{min}, s_{max}]$ of projected size into an approximate range $[d_{min}, d_{max}]$ of distance at which to place the camera. In the following section, we investigate the resolution of a distance constraint, which we could then use for computing an appropriate camera configuration.

5.3 Distance

We here consider a constraint on the distance between the camera and a subject, as an interval of accepted distance. We then search for the set of camera positions, expressed in the Toric Space, that are at an acceptable distance from the subject. We solve such a constraint in the plane (θ, α). We first show how to solve a single strict distance constraint (*i.e.* to a single subject). We then show how to compute the solution set corresponding to a range of distance between the camera and a single subject, then to a (possibly different) range of distance for both subjects.

5.3.1 Single distance

Distance to A. Let first consider a strict distance constraint d_A to A. We have, $\forall\theta \in]0, 2\pi[$,

$$\alpha = \text{ acos}\left(\frac{d_A - AB \cdot \cos(\theta/2)}{\sqrt{d_A^2 + AB^2 - 2 \cdot AB \cdot d_A \cdot \cos(\theta/2)}}\right) \tag{5.5}$$

See Figure 5.14a for illustration and Figure 5.15a for an example (red curve).

Distance to B. Let now consider the strict distance constraint d_B to B. There are two cases to consider here: $d_B \leq AB$ and $d_B > AB$.

Case $d_B \leq AB$: we have, $\forall\theta \in \left]0, 2 \text{ asin}\left(\frac{d_B}{AB}\right)\right]$,

$$\alpha = \frac{\pi}{2} \pm \text{ acos}\left[\frac{AB}{d_B} \cdot \sin\left(\frac{\theta}{2}\right)\right] \tag{5.6}$$

Note that in the particular case $d_B = AB$, the upper bound of θ (*i.e.* π) will be excluded.

Case $d_B > AB$: we have, $\forall\theta \in]0, 2\pi[$,

$$\alpha = \frac{\pi}{2} - \text{ acos}\left[\frac{AB}{d_B} \cdot \sin\left(\frac{\theta}{2}\right)\right] \tag{5.7}$$

See Figure 5.14b for illustration and Figure 5.15b for an example (green curve).

5.3.2 Range of distance

Range of distance to A. Let consider a range of distance $r_A = [d^A_{\min}, d^A_{\max}]$. The solution set is the interval

$$\mathcal{I}^A_\alpha(\theta) = [\alpha^A_{\max}(\theta), \alpha^A_{\min}(\theta)]\backslash\{0, \pi\}$$

where $\alpha^A_{\min}(\theta)$ (resp. $\alpha^A_{\max}(\theta)$) is determined by injecting $d^A_{\min}$ (resp. $d^A_{\max}$) in Equation 5.5.

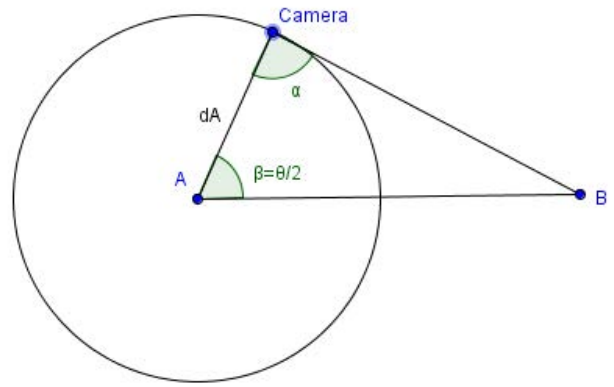
Range of distance to B. Let consider a range of distance $r_B = [d^B_{\min}, d^B_{\max}]$. The solution set is the interval

$$\mathcal{I}^B_\alpha(\theta) = \mathcal{I}^B_{\alpha,\max}(\theta) - \mathcal{I}^B_{\alpha,\min}(\theta)$$

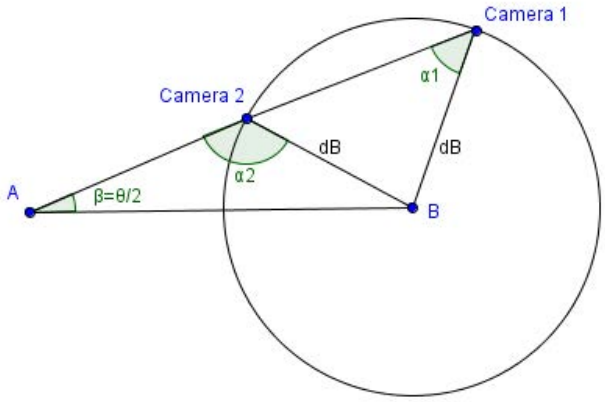
where $\mathcal{I}^B_{\alpha,\min}(\theta)$ (resp. $\mathcal{I}^B_{\alpha,\max}(\theta)$) is determined by injecting $d^B_{\min}$ (resp. $d^B_{\max}$) in the Equation 5.6 or 5.7.

As a conclusion, when constraining the camera by the two ranges of distance r_A and r_B, the solution is then the interval

$$\mathcal{I}_\alpha(\theta) = \mathcal{I}^A_\alpha(\theta) \cap \mathcal{I}^B_\alpha(\theta)$$



(a)

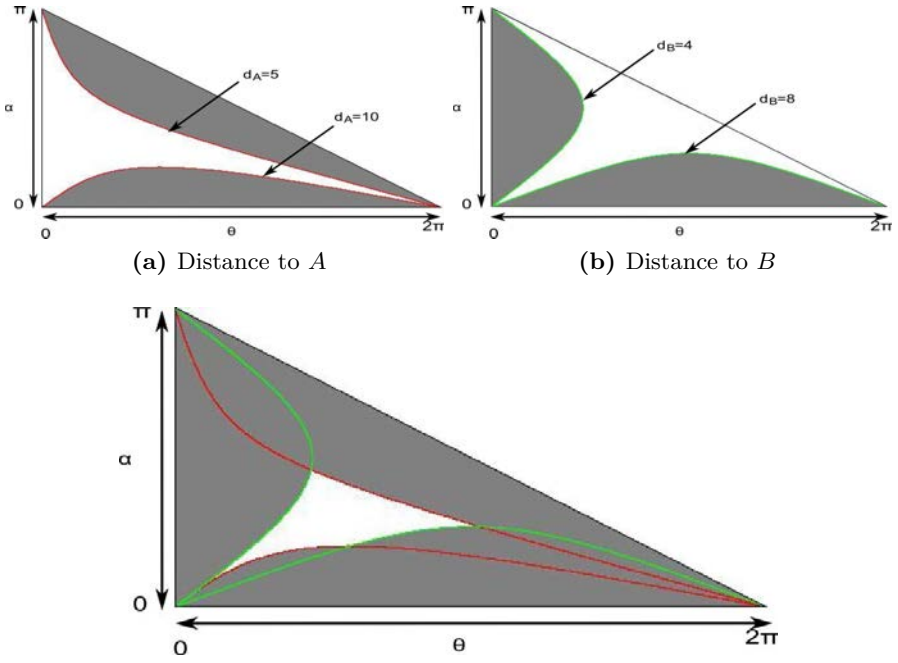


(b)

Figure 5.14 – Illustration of the resolution of a strict distance constraint in the plane (θ, α). We express the solution value(s) of α for a given value of the angle θ. (a) Resolution of a strict distance d_A to subject A. (b) Resolution of a strict distance d_B to subject B.

5.4 Vantage angle

The vantage angle constraint corresponds to a relative angle *w.r.t.* to a given subject. For instance, if we want to see a car from front with a high angle, this constraint can be expressed as a direction vector from the car. More generally, a vantage angle can be represented as a half-line of origin the subject (A or B) and direction vector $\overrightarrow{v}$. To relax this constraint, we here further consider a range of accepted directions in the form of a maximum deviation to this reference direction; we represent such a constraint as a vantage cone, of directrix $\overrightarrow{v}$ and half-angle γ (the maximum accepted deviation). The solutions then belong to this cone. We hereafter show how to express, in the Toric Space, the set of camera positions that satisfy a single vantage angle constraint



(a) Distance to A **(b)** Distance to B

(c) Solution to both distance constraints (A and B)

Figure 5.15 – Solution set corresponding to a camera in a range of distance to A and/or B. (a) Solution camera positions for a distance to A in $[5, 10]$ (white area); red curves correspond to the bounding values of the interval of distance. (b) Solution camera positions for a distance to B in $[4, 8]$ (white area); green curves correspond to the bounding values of the interval of distance. (c) Solution camera positions for the satisfaction of both ranges of distance to resp. A and B.

(*w.r.t.* either A or B). We then show how to solve the set of camera positions satisfying multiple vantage constraints (*w.r.t.* to both A and B) in the Toric Space.

5.4.1 Resolution for one subject

We here focus on a single vantage angle constraint. For generalization purposes, we will refer to the concerned key subject (either A or B) as K, and to the other key subject as K'; the set of camera positions satisfying the vantage angle, with maximum deviation γ, is the then set of points inside a *vantage cone* of apex K, direction $\overrightarrow{v}$ and half-angle γ. Note that the directrix $\overrightarrow{v}$ can be parameterized as a couple (λ, φ_v), where λ is the angle between vectors $\overrightarrow{v}$ and $\overrightarrow{KK'}$ ($\lambda \in [0, \pi]$) and φ_v represents the plane supporting $\overrightarrow{v}$.

In this section, we show how to solve the set of camera positions satisfying such a vantage angle constraint algebraically in the Toric Space; and more particularly in the plane (β^*, φ) – β^* represents either β or β' (see Figure 5.11).

Let first introduce three planes whose normal vector $\overrightarrow{n} = \dfrac{\overrightarrow{KK'}}{KK'}$:

$$\mathcal{P}^{\beta^*<\pi/2},\ \mathcal{P}^{\beta^*=\pi/2} \text{ and } \mathcal{P}^{\beta^*>\pi/2}$$

They are built such that $\mathcal{P}^{\beta^*<\pi/2}$ contains the point $K + \overrightarrow{n}$, $\mathcal{P}^{\beta^*=\pi/2}$ contains the point K itself, and $\mathcal{P}^{\beta^*>\pi/2}$ contains the point $K - \overrightarrow{n}$. That is, a vector $\overrightarrow{a}$ (resp. $\overrightarrow{b}$) intersect with them iff β^* is respectively lower than, equal to or greater than $\frac{\pi}{2}$.

Remember that the intersection of a cone and a plane is a conic; it can be either an ellipse (a particular case is the circle), a parabola, or a hyperbola. As a consequence, we here propose to use equations of conics to compute either the range of possible values for φ in function of the value β^*, or the range of possible values for β^* in function of the value φ.

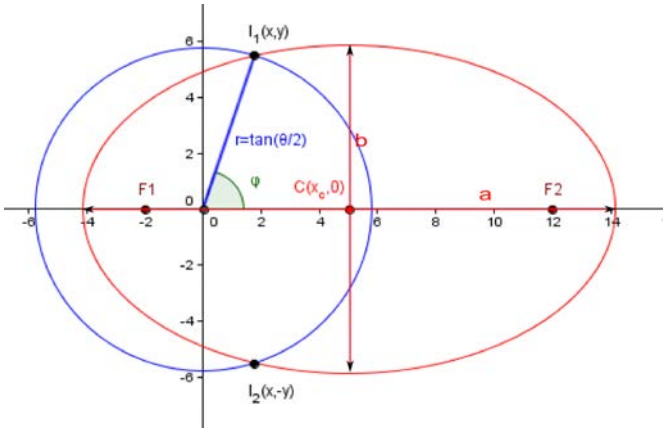


Figure 5.16 – Computation of the vantage function in the space (β, φ) in the case of an ellipse. We intersect the ellipse with a circle of radius $r = \tan(\beta) = \tan(\theta/2)$, then calculate the corresponding values of φ as $\varphi = \pm\ \text{atan}(y/x)$. This process is basically the same in case of a parabola or hyperbola.

In a first step, we will assume $\gamma \in \left[0, \dfrac{\pi}{2}\right]$. We will also assume that $\overrightarrow{v}$ belongs to the half-plane $\varphi_v = 0$. We will consider the general case afterwards. Here is the detail of resulting conics in function of λ and γ:

		$\mathcal{P}^{\beta^*<\pi/2}$	$\mathcal{P}^{\beta^*=\pi/2}$	$\mathcal{P}^{\beta^*>\pi/2}$
$\lambda = 0$		circle	$\emptyset$	$\emptyset$
$0 < \lambda \leq \pi/2$	$\lambda + \gamma < \pi/2$	ellipse	$\emptyset$	$\emptyset$
	$\lambda + \gamma = \pi/2$	parabola	single line	$\emptyset$
$0 < \lambda \leq \pi/2$	$\lambda + \gamma > \pi/2$	hyperbola	two lines	hyperbola
$\pi/2 \leq \lambda < \pi$	$\lambda - \gamma < \pi/2$			
$\pi/2 \leq \lambda < \pi$	$\lambda - \gamma = \pi/2$	$\emptyset$	single line	parabola
	$\lambda - \gamma > \pi/2$	$\emptyset$	$\emptyset$	ellipse
$\lambda = \pi$		$\emptyset$	$\emptyset$	circle

Note that the second half of this table is obtained by symmetry with the first half, by replacing both λ by $\pi - \lambda$ and β^* by $\pi - \beta^*$ in the different cases. In the following we will only consider the first half of the table and solve each case separately. We hereafter detail the two possible resolutions of the vantage angle. In any case, one will only use one of them, depending on the need.

Resolution 1: φ in function of β^* To express the frontier of the vantage cone in the Toric Space, as a function $\mathcal{I}_\varphi(\beta^*)$ – the interval of possible values of φ such that any point (β^*, φ) belong to the cone – we will compute the intersection(s) $I(x, y)$ of the given conic section $\mathcal{C}$ with a circle $\mathcal{C}'$ of center $(0, 0)$ and radius r, with $r = \tan\beta^*$. The equation of $\mathcal{C}'$ is given by $x^2 + y^2 = r^2$

We will then resolve the maximum angle φ as

$$\varphi \in \left[-\operatorname{atan}\left(\frac{y}{x}\right), +\operatorname{atan}\left(\frac{y}{x}\right) \right]$$

In the following cases, the range of values of β^* that need a resolution is given by $\beta^*_{inf} = max(\lambda - \gamma\ ,\ \gamma - \lambda)$ and $\beta^*_{sup} = min(\lambda + \gamma\ ,\ 2\pi - (\lambda + \gamma))$.

Moreover, here are the solution ranges $\mathcal{I}_\varphi$ when $\beta^* \notin [\beta^*_{inf}, \beta^*_{sup}]$:

	$\beta^* < \beta^*_{inf}$	$\beta^* > \beta^*_{sup}$
$\lambda - \gamma < 0$	$[-\pi; +\pi]$	
$\lambda - \gamma \geq 0$	$\emptyset$	
$\lambda + \gamma \leq \pi$		$\emptyset$
$\lambda + \gamma > \pi$		$[-\pi; +\pi]$

Resolution 2: β^* in function of φ To express the frontier of the vantage cone in the Toric Space, as a function $\mathcal{I}_{\beta^*}(\varphi)$ – the interval of possible values of β^* such that any point (β^*, φ) belong to the cone – we will compute the intersection(s) $I(x, y)$ of the given conic section $\mathcal{C}$ with a half-line $\mathcal{L}$ of origin $(0, 0)$ and direction $\overrightarrow{d}(\cos\varphi, \sin\varphi)$.

When $\varphi \neq \pm\frac{\pi}{2}$, the equation of $\mathcal{L}$ is

$$y = \tan\varphi \cdot x, \text{ with } sign(x) = sign(\cos\varphi)$$

When $\varphi = \pm\frac{\pi}{2}$, $\mathcal{L}$ is defined as

$$\left\{ (x, y) \text{ s.t. } x = 0 \text{ and } \begin{cases} y \in \mathbb{R}^+ \text{ if } \sin\varphi > 0 \\ y \in \mathbb{R}^- \text{ if } \sin\varphi < 0 \end{cases} \right\}$$

We then compute the bounds of the corresponding range of β^* as

$$\beta^* = \operatorname{atan}\left(\sqrt{x^2 + y^2}\right)$$

An illustration of these two resolutions is given, for the ellipse, in Figure 5.16. We will now focus on each case of conic section separately and proper solutions for them.

5.4.2 Ellipse

The conic section (ellipse) $\mathcal{C}$ is defined by its center $C(x_c, 0)$, its major radius a and its minor radius b, defined as follows

$$a = \frac{\sin\gamma \cdot \cos\gamma}{|\cos^2\lambda - \sin^2\gamma|}$$
$$b = \frac{\sin\gamma}{\sqrt{|\cos^2\lambda - \sin^2\gamma|}}$$
$$x_c = \frac{\sin\lambda \cdot \cos\lambda}{\cos^2\lambda - \sin^2\gamma}$$

The equation of $\mathcal{C}$ is here $\frac{(x-x_c)^2}{a^2} + \frac{y^2}{b^2} = 1$

Resolution 1: $\mathcal{I}_\varphi(\beta^*)$ The intersections $I(x, y)$ are computed by solving the following system

$$\begin{cases} Ax^2 + Bx + C = 0 \\ y = \sqrt{r^2 - x^2} \end{cases}$$

with $A = b^2 - a^2$, $B = -2b^2x_c$, $C = b^2x_c^2 + a^2(r^2 - b^2)$.

Resolution 2: $\mathcal{I}_{\beta^*}(\varphi)$

Case $\varphi \neq \pm\frac{\pi}{2}$. The intersections $I(x, y)$ are computed by solving the following system

$$\begin{cases} Ax^2 + Bx + C = 0 \\ y = \tan\varphi \cdot x \end{cases}$$

with $A = b^2 + a^2\tan^2\varphi$, $B = -2b^2x_c$, $C = b^2(x_c^2 - a^2)$.

Case $\varphi = \pm\frac{\pi}{2}$. The solution intersection $I(x, y)$ is given by

$$\begin{cases} x = 0 \\ y = \sin\varphi \cdot \frac{b}{a}\sqrt{a^2 - x_c^2} \end{cases}$$

Moreover, when $\gamma \leq \lambda$, the extreme values of φ are reached when $\Delta = B^2 - 4AC = 0$, *i.e.*

$$\mathcal{I}_\varphi \subseteq \left[-\operatorname{atan}\left(\sqrt{\frac{b^2}{x_c^2 - a^2}}\right) ; +\operatorname{atan}\left(\sqrt{\frac{b^2}{x_c^2 - a^2}}\right) \right]$$

Otherwise ($\gamma > \lambda$), $\mathcal{I}_\varphi \subseteq [-\pi; +\pi]$.

5.4.3 Circle

When $\lambda = 0$, the conic section (circle) $\mathcal{C}$ is here a special case of ellipse. The radius of $\mathcal{C}$ is then $r' = a = b = \tan\gamma$, and its center is $C(0,0)$. In this case, the only possible intersection between $\mathcal{C}$ and $\mathcal{C}'$ is when $r = r'$. In the same way, the intersection between $\mathcal{C}$ and $\mathcal{L}$ is always $I(\tan\gamma \cdot \cos\varphi, \tan\gamma \cdot \sin\varphi)$. Thus, the solution to both resolutions is trivial: it corresponds to the line of equation $\beta^* = \gamma$, for all $\varphi \in [-\pi; +\pi]$.

5.4.4 Parabola

The conic section (parabola) $\mathcal{C}$ is defined by its vertex $V(h,0)$, and its focal length f defined as follows

$$h = \frac{\tan\lambda - \cot\lambda}{2}$$
$$f = \frac{\cot\lambda}{2}$$

The equation of $\mathcal{C}$ is here $y^2 = 4f(x-h)$

Resolution 1: $\mathcal{I}_\varphi(\beta^*)$ The intersections $I(x,y)$ are computed by solving the following system

$$\begin{cases} Ax^2 + Bx + C = 0 \\ y = \sqrt{r^2 - x^2} \end{cases}$$

with $A = -1$, $B = -4f$, $C = 4fh + r^2$.

The intersection with the intermediate plane will moreover be a single line. It corresponds to the angle $\beta^* = \frac{\pi}{2}$, and the associated range $\mathcal{I}_\varphi$ is $\{0\}$.

Resolution 2: $\mathcal{I}_{\beta^*}(\varphi)$

Case $\varphi \neq \pm\frac{\pi}{2}$. The intersections $I(x,y)$ are computed by solving the following system

$$\begin{cases} Ax^2 + Bx + C = 0 \\ y = \tan\varphi \cdot x \end{cases}$$

with $A = \tan^2\varphi$, $B = -4f$, $C = 4fh$.

Case $\varphi = \pm\frac{\pi}{2}$. The solution intersection $I(x,y)$ is given by

$$\begin{cases} x = 0 \\ y = \sin\varphi \cdot 2\sqrt{-fh} \end{cases}$$

Moreover, when $\gamma \leq \lambda$, the extreme values of φ are reached when $\Delta = B^2 - 4AC = 0$, *i.e.*

$$\mathcal{I}_\varphi \subseteq \left[-\operatorname{atan}\left(\sqrt{\frac{f}{h}}\right); +\operatorname{atan}\left(\sqrt{\frac{f}{h}}\right) \right]$$

Otherwise ($\gamma > \lambda$), $\mathcal{I}_\varphi \subseteq [-\pi; +\pi]$.

5.4.5 Hyperbola

The conic section (hyperbola) $\mathcal{C}$ is defined by its vertex $V(x_v, 0)$, its major radius a, and its minor radius b, defined as follows

$$a = \frac{\sin\gamma \cdot \cos\gamma}{|\cos^2\lambda - \sin^2\gamma|}$$
$$b = \frac{\sin\gamma}{\sqrt{|\cos^2\lambda - \sin^2\gamma|}}$$
$$x_v = \frac{\sin\lambda \cdot \cos\lambda}{\cos^2\lambda - \sin^2\gamma}$$

The equation of $\mathcal{C}$ is here $\frac{(x-x_v)^2}{a^2} - \frac{y^2}{b^2} = 1$

Resolution 1: $\mathcal{I}_\varphi(\beta^*)$ The intersections $I(x, y)$ are computed by solving the following system

$$\begin{cases} Ax^2 + Bx + C = 0 \\ y = \sqrt{r^2 - x^2} \end{cases}$$

with $A = b^2 + a^2$, $B = -2b^2x_v$, $C = b^2x_v^2 - a^2(r^2 + b^2)$.

Resolution 2: $\mathcal{I}_{\beta^*}(\varphi)$

Case $\varphi \neq \pm\frac{\pi}{2}$. The intersections $I(x, y)$ are computed by solving the following system

$$\begin{cases} Ax^2 + Bx + C = 0 \\ y = \tan\varphi \cdot x \end{cases}$$

with $A = b^2 - a^2\tan^2\varphi$, $B = -2b^2x_v$, $C = b^2(x_v^2 - a^2)$.

Case $\varphi = \pm\frac{\pi}{2}$. The solution intersection $I(x, y)$ is given by

$$\begin{cases} x = 0 \\ y = \sin\varphi \cdot \frac{b}{a}\sqrt{x_v^2 - a^2} \end{cases}$$

Moreover, when $\gamma \leq \lambda$, the extreme values of φ are reached when $\Delta = B^2 - 4AC = 0$, *i.e.*

$$\mathcal{I}_\varphi \subseteq \left[-\operatorname{atan}\left(\sqrt{\frac{b^2}{a^2 - x_v^2}}\right); +\operatorname{atan}\left(\sqrt{\frac{b^2}{a^2 - x_v^2}}\right)\right]$$

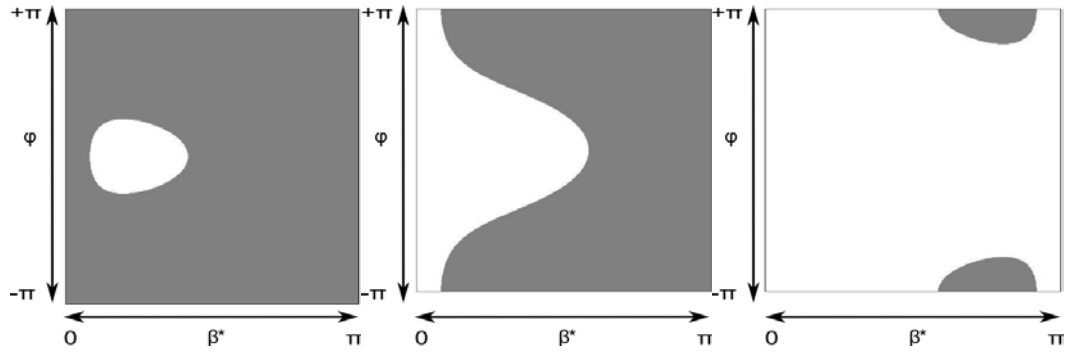
Otherwise ($\gamma > \lambda$), $\mathcal{I}_\varphi \subseteq [-\pi; +\pi]$.

5.4.6 Generalization

In the general case (*i.e.* $\varphi_v \neq 0$), these resolutions will serve as a basis to compute the actual solution. Indeed, the remaining operation is to shift the resulting range $\mathcal{I}_\varphi$ from φ_v upward. The actual solution with this offset is then obtained by replacing φ by $\varphi - \varphi_v$ in the resolutions above.

Note that, in the particular case of $\gamma = \frac{\pi}{2}$, the cone bounds will represent a plane and the solution set will then be a half-space. We furthermore study the case $\gamma > \frac{\pi}{2}$. In this case, the solution will then be given by considering the exclusion of the cone whose axis is opposite to the vantage vector and such that its half-angle γ' and γ are complementary angles (they sum to π). This cone is then parameterized as $\lambda' = \pi - \lambda$, $\gamma' = \pi - \gamma$ and $\varphi_{v'} = \pi - \varphi_v$.

Examples of solutions for various vantage angles are given in Figure 5.17.



(a) $\gamma \leq \pi/6$ **(b)** $\gamma \leq \pi/3$ **(c)** $\gamma \leq 5\pi/6$

Figure 5.17 – Solution range of a vantage angle, for a given direction vector and different values of the half-angle γ of the vantage cone. In the examples, the direction vector of the vantage cone is parameterized by $\lambda = \frac{\pi}{4}$ and $\varphi_v = 0$. In each case, the white area represent the set of couples (β^*, φ) satisfying the vantage angle.

5.4.7 Intersection for both A and B

We here consider the combination of two vantage angle constraints: one relative to A and one relative to B. Its solution is located at the intersection of both vantage cones. In order to find the set of camera positions satisfying both constraints, lets take a look at a section of the Toric Space at a given height $\varphi \in \mathcal{I}_\varphi^A \cap \mathcal{I}_\varphi^B$. We compute the ranges $\mathcal{I}_\beta(\varphi)$ *w.r.t.* A and $\mathcal{I}_{\beta'}(\varphi)$ *w.r.t.* B. We then build the functions that project the ranges of solution for A and B in the plane (θ, α), *i.e.* depending on the value of α. To do so, we use the Equation 5.4. Resulting functions are of the form

$$\theta_i^A(\alpha) = 2\beta_i$$
$$\theta_j^B(\alpha) = 2(\pi - \alpha - \beta'_j)$$

where $\theta_i^A(\alpha)$ and $\theta_j^B(\alpha)$ are relative to respectively the range of vantage angle of A and B. The intersection of two such functions is then the point at which these functions

are equal, *i.e.*

$$\theta_{i,j} = \theta_i, \text{ and } \alpha_{i,j} = \pi - \beta_i - \beta'_j$$

The solution of such a combination of vantage constraints is then, for a given value φ, a set of parallelograms in the plane (θ, α), as illustrated in Figure 5.18; and the entire solution is given by the integration of such intersections for all possible values of φ.

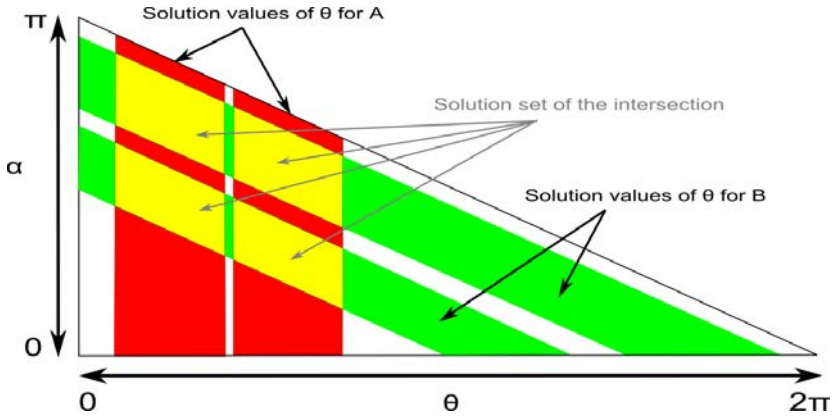


Figure 5.18 – Intersection, for a section at a given height φ, of the solution sets for both a vantage angle *w.r.t.* A and a vantage angle *w.r.t.* B. The set of solution couples (θ, α) for each section corresponds to a set of parallelograms (in yellow).

6 Combining Constraints

We hereafter detail the general process we propose to solve the entire set of constraints. Our process is made of three consecutive steps, and enables computing the range of overall solutions in terms of positions $(\alpha, \theta, \varphi)$.

Step 1 We compute the interval $\mathcal{I}_\varphi = \mathcal{I}_\varphi^A \cap \mathcal{I}_\varphi^B$ range of φ, for which there is potentially an intersection of both vantage angles. By default $\mathcal{I}_\varphi^X$ is set to $[-\pi; +\pi]$.

Step 2 For all $\varphi \in \mathcal{I}_\varphi$, we compute the ranges (*i.e.* one or more intervals) $\mathcal{I}_\theta^A(\varphi)$ and $\mathcal{I}_\theta^B(\varphi)$ corresponding to the ranges of values of θ such that (θ, φ) is solution to the vantage angle of resp. A and B. We then define the range $\mathcal{I}_\theta(\varphi)$, for which there is potentially a solution to the intersection, as $\mathcal{I}_\theta(\varphi) = \mathcal{I}_\theta^A(\varphi)$ (see Figure 5.18).

Step 3 For all $\theta \in \mathcal{I}_\theta(\varphi)$, we compute:

- $\mathcal{I}_\alpha^{VANT}(\theta, \varphi)$, the range of values of α corresponding to the satisfaction of vantage angles for both subjects.
- $\mathcal{I}_\alpha^{DIST}(\theta)$, the interval of values of α corresponding to the satisfaction of distance constraints for both subjects.
- $\mathcal{I}_\alpha^{OSP}$, the interval of values of α corresponding to the satisfaction of the on-screen positioning of both subjects.

We then compute the interval $\mathcal{I}_\alpha(\theta, \varphi)$ corresponding to the satisfaction of all constraints as

$$\mathcal{I}_\alpha(\theta, \varphi) = \mathcal{I}_\alpha^{VANT}(\theta, \varphi) \cap \mathcal{I}_\alpha^{DIST}(\theta) \cap \mathcal{I}_\alpha^{OSP}$$

This provides a general mean to solve the range of solution coordinates $(\alpha, \theta, \varphi)$. In addition, it provides a mean to check, at each step, issues in the application of constraints (*e.g.* unsolvable constraints, or conflicts between two or more constraints) by simply intersecting their solution intervals *w.r.t.* α, θ or φ.

An obvious way to compute the range of solutions would be to rely on interval analysis techniques and interval-based constraint solving. Such techniques provide an approximation of the inner and outer limits of the solution as boxes (in the solution, out of the solution and on the frontier of the solution). However, the computational cost of these techniques, coupled with the necessity to carefully control each step of the solving process led us to favor a progressive sampling technique. This efficient sampling-based technique computes a representative set of solution viewpoints. To do so, at each step, we use a sampling density (resp. d_φ, d_θ and d_α) and perform a regular sampling of the solution intervals $\mathcal{I}_\varphi$, $\mathcal{I}_\theta(\varphi)$ and $\mathcal{I}_\alpha(\theta, \varphi)$. Therefore, we sample φ at step 1 which provides a range of intervals on θ (step 2), which we sample to compute a range of intervals on α (step 3), which we then sample to evaluate visibility. Each step filters inconsistent intervals on the parameters. We then introduce a fourth step, needed to handle the visibility constraint on subjects.

Step 4 For all $\alpha \in \mathcal{I}_\alpha(\theta, \varphi)$ we compute the camera viewpoint (position and orientation) in world coordinates, then evaluate the visibility of each subject by using a ray casting technique over an object-oriented bounding box representing the subject (we cast rays to the 8 corners and the center of the bounding box). We finally either accept or discard the camera viewpoint depending on its satisfaction of the visibility constraint for each of the subjects.

7 Satisfaction of constraints

We here consider the case where there is no overall solution to a given composition problem. In such a case, one could then imagine to use an optimization-based technique guided by the algebraic solution of each single property in the Toric Space. As for the on-screen positioning constraint, the satisfaction can be trivially measured by using the solution interval r_α. We therefore study the two other properties we have previously introduced (the distance and vantage angle constraints), and show how to measure their satisfaction in the Toric Space.

7.1 Measuring the satisfaction of a distance constraint

Lets take a look at how we can evaluate, for a given camera position $P(\alpha, \theta, \varphi)$, the satisfaction of a distance constraint. The actual distances to A (d_A) and to B (d_B) are

$$d_A = \frac{AB}{\sin(\alpha)} \cdot \sin\left(\alpha + \frac{\theta}{2}\right), \text{ and } d_B = \frac{AB}{\sin(\alpha)} \cdot \sin\left(\frac{\theta}{2}\right)$$

We can then evaluate the satisfaction of the different distance constraints by a simple comparison between the actual distance(s) and the desired range(s) of distance. An illustration of this satisfaction measurement is given in the form of a heat map in Figure 5.19.

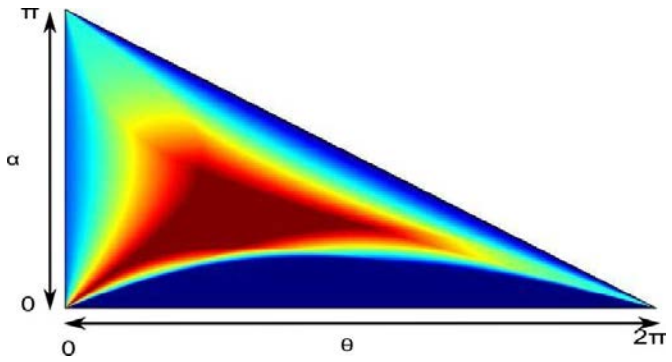


Figure 5.19 – Heat map representing the satisfaction of camera positions *w.r.t.* a range of distance to A and a range of distance to B. Dark red: highest quality (solution positions); dark Blue: lowest quality.

7.2 Measuring the satisfaction of a vantage constraint

Lets now take a look at how we can evaluate, for a given camera position $P(\alpha, \theta, \varphi)$ defined in the Toric Space, the satisfaction of a vantage constraint.

Firstly, we define two vectors $\overrightarrow{c}$, representing the axis of the cone; and $\overrightarrow{v}$, representing the vector $\overrightarrow{TV}$. Moreover, by using the equation Equation 5.4, the coordinates of $\overrightarrow{c}$ and $\overrightarrow{v}$ in the plane (β^*, φ) can be computed as respectively $(\beta^*_{\mathcal{C}}, \varphi_{\mathcal{C}})$ and (β^*_V, φ_V).

Secondly, to compute the angle between $\overrightarrow{v}$ and the axis $\overrightarrow{c}$, we use the three projection planes $\mathcal{P}^{\beta^*<\pi/2}$, $\mathcal{P}^{\beta^*=\pi/2}$ and $\mathcal{P}^{\beta^*>\pi/2}$. They serve as basis to express, in an orthonormal frame local to T, the coordinates of the intersection $I(x, y, z)$ of one vector ($\overrightarrow{c}$ or $\overrightarrow{v}$) with one of these planes. Here is the way we compute the coordinates of this intersection point, accordingly to the value of β^*:

	$\beta^* < \pi/2$	$\beta^* = \pi/2$	$\beta^* > \pi/2$
x	$\tan(\beta^*) \cdot \cos(\varphi)$	$\cos(\varphi)$	$\tan(\pi - \beta^*) \cdot \cos(\varphi)$
y	$\tan(\beta^*) \cdot \sin(\varphi)$	$\sin(\varphi)$	$\tan(\pi - \beta^*) \cdot \sin(\varphi)$
z	-1	0	$+1$

Finally, we compute the angle γ' as the non-oriented angle between $\overrightarrow{c}$ and $\overrightarrow{v}$. The satisfaction of a vantage angle constraint can then be computed by a simple comparison between γ' and the desired range of vantage angle. An illustration of this satisfaction measurement is given in the form of a heat map in Figure 5.20.

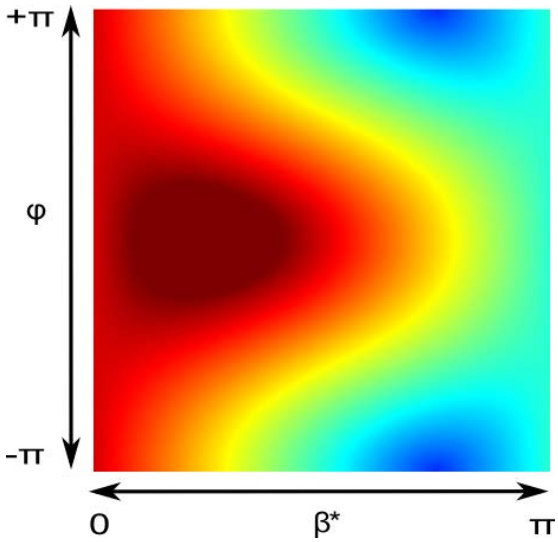


Figure 5.20 – Heat map representing the distance to the vantage cone for each point of the plane (β^*, φ). Here, the direction vector of the vantage cone is parameterized by $\lambda = \frac{\pi}{4}$ and $\varphi = 0$ and the half-angle γ is equal to $\pi/6$. Dark red: highest quality ; dark blue: lowest quality.

8 Results

We illustrate the key features of our model by exploring three different specifications: (i) applying, in a static scene, a set of visual constraints on a two-subject configuration, (ii) enforcing constraints on two moving subjects and (iii) using our model for cinematic purposes, *i.e.* editing a sequence of shots while enforcing a number of preferred framings on the two-subject configuration. For the materials, we rely on a fully animated 3D environment of a city, in which two cars are racing together. The city is a static geometry (*e.g.* buildings, bridges), and contains dynamic occluders (*e.g.* buses). Another particularity is that the scene also contains some occluders in which the cars get into (*e.g.* tunnels, trailers) during the race; this constitutes a challenge *w.r.t.* the search of a non-occluded viewpoint.

We cast Semantic Volumes specification into a set of constraints that our model solves: vantage angles, distances, on-screen positions of key subjects. In our tests, we have used two different framings:

Composition #1: Over-the-shoulder of A; A must be at a distance between 4 and 8, and B must be at a distance greater than 5; the visibility of both subject must be within $[0.25, 1.00]$; A is positioned at $(0.00, -0.33)$ on the screen (within a radius of 0.15), and B is positioned at $(0.00, +0.33)$ on the screen (within a radius of 0.15).

Composition #2: Over-the-shoulder of B; A must be at a distance greater than 5, and B must be at a distance between 4 and 8; the visibility of both subject must be within $[0.25, 1.00]$; A is positioned at $(0.00, +0.33)$ on the screen (within

a radius of 0.15), and B is positioned at $(0.00, -0.33)$ on the screen (within a radius of 0.15).

All tests have been run at a frame rate of 30fps on the same sequence of the animation, of duration 93s.

8.1 Specification #1: static application of constraints

In this example, we consider all frames of the animation separately. At each time step, we apply the composition to the two-subject configuration, without considering the previous/next time step. We then perform the two phases of the solution computation: (i) the sampling phase of candidate configurations, and (ii) the ray cast phase (to check the visibility feature) on these candidates. We here make the assumption that the evaluation of visibility is the bottleneck of the pipeline. We therefore propose a technique to reduce, as much as possible, the computation time necessary in this phase to select the best candidate viewpoint.

For the purpose of comparison, we launched a random ray cast in our scene, using 10^8 rays for which origin and target points are taken randomly in a bounding box of the whole environment. In our implementation, the ray casts are performed by using the C++ library *bullet*. In our scene, the mean cost per ray cast ranges from around 7. 10^{-4} ms in regions with weak visibility constraints to 1. 10^{-2} ms in regions with strong visibility constraints. In our tests, we sampled 9 points on the surface of each subject, corresponding to the 8 corners and the center of its bounding box; we then used 18 ray casts per candidate viewpoint. In the worst case, this computation could therefore cost up to 0.18 ms per candidate. To reduce cost in the worst case, we instead use a "smart" selection process of the best candidate. We first rank sample candidates with their satisfaction of constraints (or quality, by using a distance to the optimal value of each constraint). We sort samples *w.r.t.* their ranking Q^{Others} *w.r.t.* visual constraints (but visibility), then consider candidates in order of decreasing quality. We finally introduce a maximal value Q^{Vis}_{max} corresponding to the evaluation of subject's visibility (*i.e.* $Q^{Vis}(i) \leq Q^{Vis}_{max}$). We then stop our iterative selection process as soon as the current candidate verifies

$$Q_{best} \geq Q^{Others}(i) + Q^{Vis}_{max}$$

where i is the index of the current candidate, and Q_{best} is the best overall ranking (integrating the visibility quality) after iterating on the $i-1$ best ranked candidates.

We launched our viewpoint computation algorithm on both composition #1 and #2, and used four different sampling densities on the three parameters $d\varphi$, $d\theta$ and $d\alpha$ (5×5×5, 10×10×10, 15×15×15 and 20×20×20). Figure 5.21 illustrates an example of the evolution of parameters α, θ and φ, as well as the satisfaction of constraints over time. Table 5.1 further aggregates the performances of our computation algorithm according to the sampling density used. We compute and evaluate an average of 3208 viewpoints corresponding to a given composition in about 122ms (with a sampling density of 20×20×20). Four observations can further be made. First, one can observe that the number of cameras sampled increases proportionally to the sampling density,

and that it goes with an improvement in the number of successful frames (*i.e.* frames for which a solution to the overall composition problem has been found). Second, the mean computation time per camera is further improved through the use of a higher sampling resolution; in our tests this improvement however reaches a threshold for densities higher than 15×15×15. This result might be explained by a balance between the cost of computing intervals $\mathcal{I}_\varphi$, $\mathcal{I}_\theta$ (which remains expensive) and $\mathcal{I}_\alpha$ and the cost of computing more samples in these intervals (which in comparison is quite negligible, but necessitate the computation of more intervals in the next step). Third, as for the visibility computations, we can observe that the number of cameras tested for visibility is rather low (above 15%). An improvement in the ratio of cameras tested is further observed when increasing the sampling density. These two results demonstrates that our regular interval-based sampling technique effectively computes a set of camera viewpoints that properly represents the solution set of the overall composition problem, whether it be for low or high sampling resolutions. Indeed, if sampled cameras were too close to each other, they would have a similar ranking and one could not discard so many cameras. Last, interactive computation performances can reasonably be reached by using a budget between 800 and 950 cameras. In our tests, this could be implemented through the use of a sampling density between 12 and 13 samples on each interval (*i.e.* using the resolution 12×12×12 or 13×13×13).

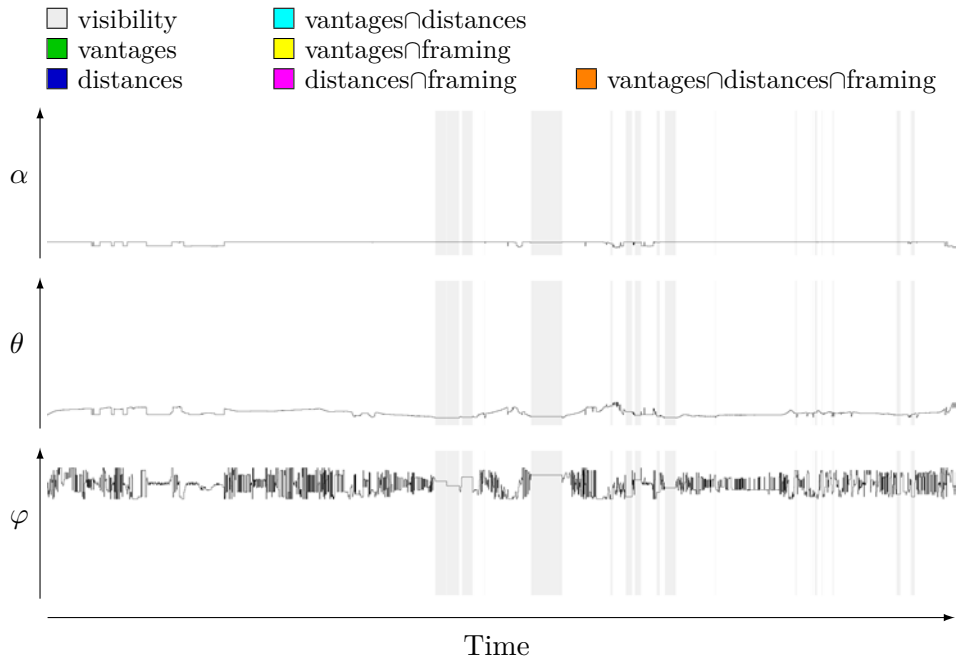


Figure 5.21 – Illustration of the evolution of the parameters α, θ and φ (black curves), as well as the satisfaction of constraints over time, for an example of static application of constraints. Colored areas corresponds to failures. The color legend explains failures in the application or combination of constraints. In this example, only the visibility caused failures.

	5×5×5	**10×10×10**	**15×15×15**	**20×20×20**
# frames	2792	2792	2792	2792
# successful frames	2369.5 (84.87%)	2498 (89.47%)	2519 (90.22%)	2539.5 (90.96%)
# cameras sampled	54	414	1301	3208
Sampling (ms)	0.15	0.63	1.13	2.36
Mean time per sampled camera (ms)	0.0030	0.0017	0.0009	0.0008
# cameras tested	7.5 (13.9%)	48.5 (11.7%)	143 (10.9%)	342 (10.6%)
Ray cast (ms)	2.52	17.36	46.71	119.58
Standard deviation (ms)	5.97	50.12	141.47	373.46
Mean time per sampled camera (ms)	0.0428	0.0399	0.0340	0.0348
Total (ms)	2.67	17.99	47.84	121.94
Mean time per sampled camera (ms)	0.0458	0.0416	0.0349	0.0356

Table 5.1 – Performances of the two consecutive phases of the solutions computation: (i) sampling candidate camera configurations and (ii) evaluating their visibility over subjects (through a ray casting technique). We here use four different sampling densities on the three parameters $d\varphi$, $d\theta$ and $d\alpha$ (5×5×5, 10×10×10, 15×15×15 and 20×20×20). The number of successful frames here corresponds to frames for which a solution to the overall composition problem has been found.

8.2 Specification #2: visual composition enforcement

Since the Toric Space is defined relative to the subjects, the camera will move along the subject. Therefore, a coordinate $(\alpha, \theta, \varphi)$ computed at time t may preserve the satisfaction of all or a part of constraints at time $t + \Delta t$, even if the subjects have moved in the world. The camera position computed in the Toric Space therefore serves as a mean to automatically re-compute an appropriate world camera position and orientation from the subjects new world positions.

We hereafter compare two possible ways of enforcing the constraints: using either a static camera or a dynamic camera in the Toric Space.

8.2.1 Enforcement of constraints through a static camera in the Toric Space

In this example, we compute a unique camera position $(\alpha, \theta, \varphi)$ satisfying a given composition in the Toric Space at time $t = 0$. This position is then used to enforce

the constraints satisfaction, during the whole sequence, while the two cars are moving. Only the camera orientation is recomputed at each time step. Figure 5.22 illustrates an example of evolution of the parameters α, θ and φ, as well as the satisfaction of constraints over time.

8.2.2 Enforcement of constraints through a dynamic camera in the Toric Space

In addition to previous enforcement, in this example, we maintain the camera position $(\alpha, \theta, \varphi)$ as much as possible and compute a new camera position $(\alpha', \theta', \varphi')$, in the neighborhood of the current camera configuration, when necessary. This new position is then used to enforce the constraints satisaction on the moving cars. One advantage of our Toric Space is that, by simply using a linear interpolation between these two configurations in the Toric Space, the camera can be moved in a natural way. Figure 5.23 illustrates an example of the evolution of parameters α, θ and φ, as well as the satisfaction of the constraints over time.

8.3 Specification #3: editing

The Toric Space also offers the possibility to implement classical continuity editing rules [Tho93] when constraints are not satisfied. We here detail how we express such rules by using the constraints we have introduced beforehand. We then compare the same constraints enforcement methods as previously, with the possibility to cut to another viewpoint.

8.3.1 Continuity rules

We here express the continuity rules through previously presented constraints. This enables defining rules in 3D, whereas previous techniques only focused on applying editing rules in 2D [HCS96, CAH+96, JY05, ER07, LC08]. We then combine these new constraints with others, by simply intersecting their solution ranges with $\mathcal{I}_\varphi$, $\mathcal{I}_\theta(\varphi)$ and $\mathcal{I}_\alpha(\theta, \varphi)$ in the appropriate step (as described in Section 6). In the following, we assume that the current camera configuration is $C(\alpha_c, \theta_c, \varphi_c)$.

Line of interest continuity: 180° rule. We have built our toric coordinate system so that the 180° rule is quite simple to implement.

$$\mathcal{I}_\varphi^{180} = \begin{cases} [-\frac{\pi}{2}, +\frac{\pi}{2}] & \text{if } \varphi_c \in [-\frac{\pi}{2}, +\frac{\pi}{2}] \\ [-\pi, -\frac{\pi}{2}] \ \cup \ [+\frac{\pi}{2}, +\pi] & \text{if } \varphi_c \notin [-\frac{\pi}{2}, +\frac{\pi}{2}] \end{cases}$$

Line of action continuity. To enforce the line of action of a subject, we simply remove the half-space located either on his/her left side or on his/her right side depending on the case. To do so, we proceed as follows. Let $\overrightarrow{r}$ be the right vector of a subject T (can be either A or B). If the non-oriented angle between $\overrightarrow{v}$ and $\overrightarrow{TC}$ is lower than $\frac{\pi}{2}$, then we apply a new vantage angle on T parameterized by the direction

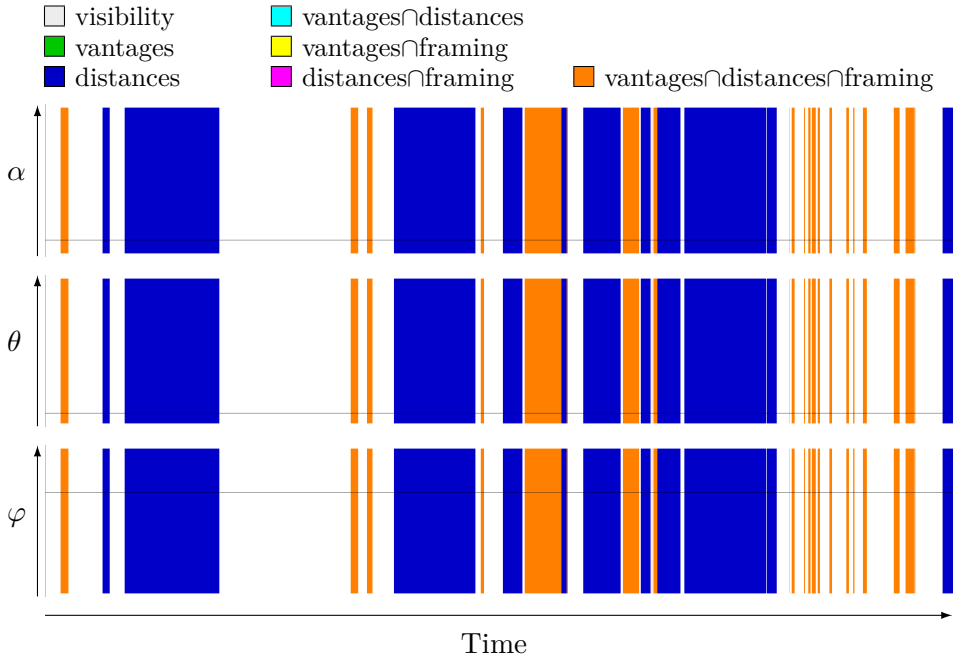


Figure 5.22 – Illustration of the evolution of parameters α, θ and φ (black curves), as well as the satisfaction of constraints over time, for an example of enforcement of constraints through a static camera in the Toric Space. The red bar represent the initial cut, and colored areas corresponds to failures. The color legend explains failures in the application or combination of constraints.

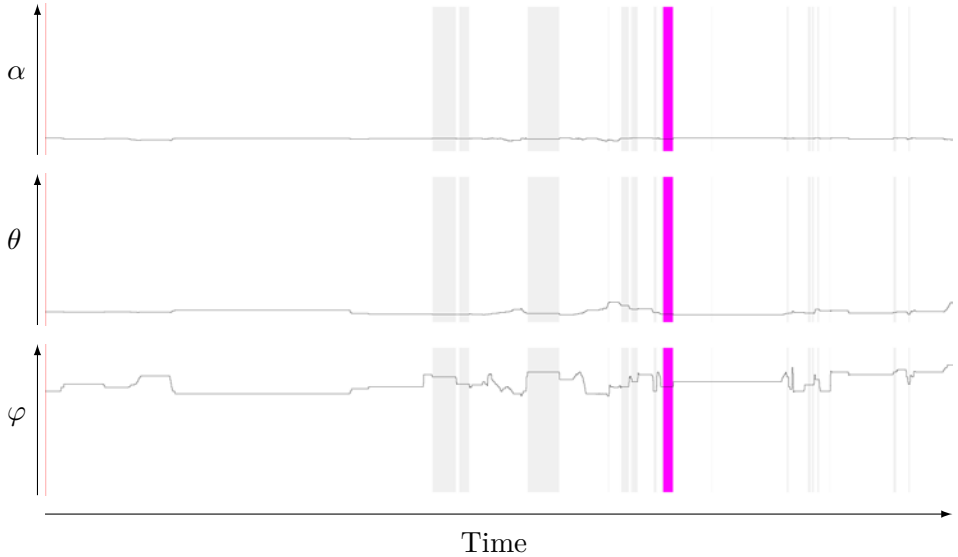


Figure 5.23 – Illustration of the evolution of the parameters α, θ and φ (black curves), as well as the satisfaction of constraints over time, for an example of enforcement of constraints through a dynamic camera in the Toric Space. The red bar represent the initial cut, and colored areas corresponds to failures. The color legend explains failures in the application or combination of constraints.

vector $\overrightarrow{v}$ and of half-angle $\gamma = \frac{\pi}{2}$. Otherwise, we apply a new vantage angle on T parameterized by the opposite direction vector $(-\overrightarrow{v})$ and of half-angle $\gamma = \frac{\pi}{2}$.

Minimum change in angle: 30° rule. We construct:

- a new vantage angle *w.r.t.* A parameterized by $\lambda = \frac{\theta_c}{2}$, $\varphi = \varphi_c$ and $\gamma = \frac{\pi}{6}$.
- a new vantage angle *w.r.t.* B parameterized by $\lambda = \pi - \alpha - \frac{\theta_c}{2}$, $\varphi = \varphi_c$ and $\gamma = \frac{\pi}{6}$.

The intersection of these two vantage angles then describes the region of space in which a camera would not satisfy the minimum change in angle.

Minimum change in size. We first compute the projected size of both subjects then construct a constraint on their projected size so that there is at most a 20% size difference between two consecutive shots for both subjects. This intersection of distance constraint on both subjects then describes the region of space in which a camera would not satisfy the minimum change in size.

The intersection of the two described regions (minimum change in angle and distance) then describes the region of space in which a camera would not satisfy the minimum change in angle; this region is removed from the search.

8.3.2 Enforcement of constraints through a static camera + editing in the Toric Space

In this example, we compute a unique camera position $(\alpha, \theta, \varphi)$ for each shot. This position is used to enforce the constraints satisfaction during the whole shot and a cut is performed when necessary (*e.g.* unsatisfied constraint, visibility issue). Figure 5.24 illustrates an example of evolution of the parameters α, θ and φ, as well as the satisfaction of constraints over time.

8.3.3 Enforcement of constraints through a dynamic camera + editing in the Toric Space

In this example, we maintain the camera position $(\alpha, \theta, \varphi)$ as much as possible and compute a new camera position $(\alpha', \theta', \varphi')$, in the neighborhood of the current camera configuration, when necessary. As previously, we use a linear interpolation in Toric Space to describe the camera motions. We then perform a cut when no solution configuration can be found locally. Figure 5.25 illustrates an example of evolution of the parameters α, θ and φ, as well as the satisfaction of constraints over time.

8.4 Performances

We here compare the performances of the previous constraints enforcement methods, with four different sampling densities on the three parameters $d\varphi$, $d\theta$ and $d\alpha$ (5×5×5, 10×10×10, 15×15×15 and 20×20×20). The results of our tests are shown in Tables 5.2 to 5.5. We can interpret these results as follows.

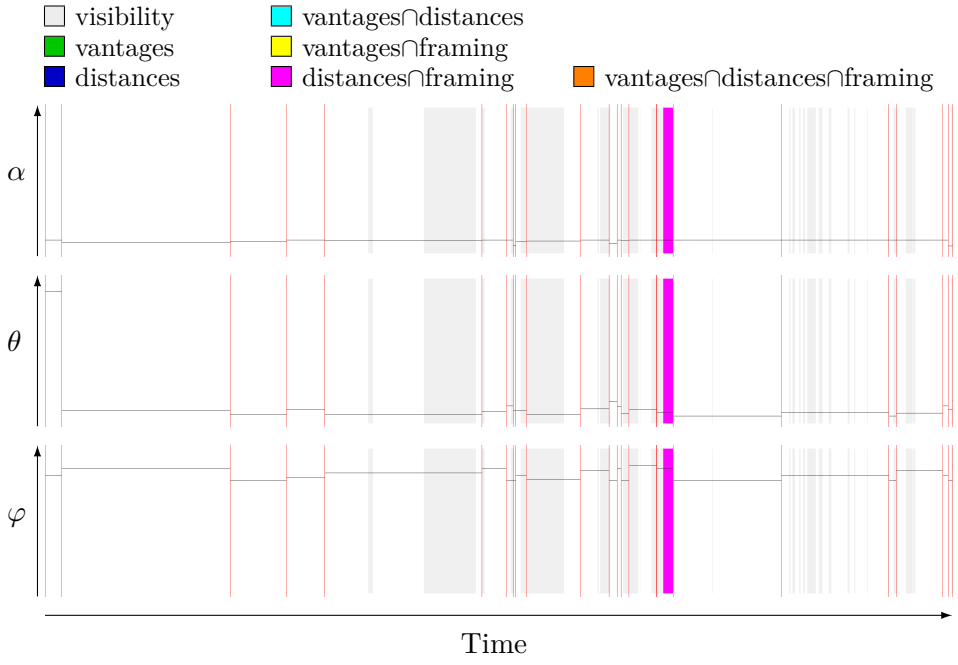


Figure 5.24 – Illustration of the evolution of the parameters α, θ and φ (black curves), as well as the satisfaction of constraints over time, for an example of enforcement of constraints through a static camera + editing in the Toric Space. The red bars represent cuts, and colored areas corresponds to failures. The color legend explains failures in the application or combination of constraints.

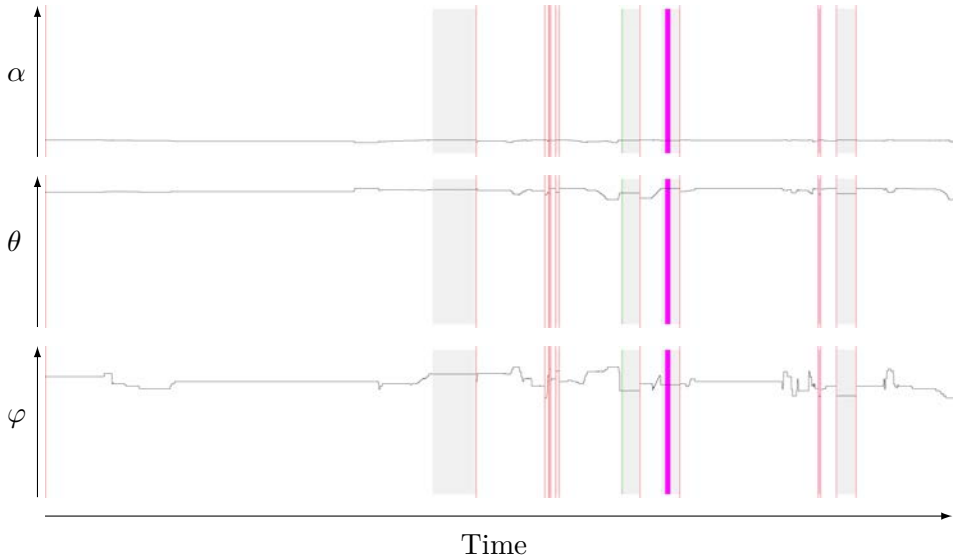


Figure 5.25 – Illustration of the evolution of the parameters α, θ and φ (black curves), as well as the satisfaction of constraints over time, for an example of enforcement of constraints through a dynamic camera + editing in the Toric Space. The red bars represent cuts, and colored areas corresponds to failures. The color legend explains failures in the application or combination of constraints.

The use of a static camera in the Toric Space all along the sequence already enables to have near 50% of the frames in which all constrainst are satisfied, at a very low cost; its cost is near to 0 (the 1ms computation time per frame shown integrates the test of constraints satisfaction). The use of a dynamic camera in the Toric Space significantly improves the quality of the sequence of viewpoints, but at cost of computation time due to re-computation of a solution viewpoint occurring more often (*i.e.* as soon as the current viewpoint does not satisfy all the constraints). Furthermore, increasing the sampling density provides a real improvement in the number of successful frames. It further leads to less occlusions of subjects, but above all leads to significant reduction of the duration of occlusions (up to a 17% redution). Editing seems a better choice to reach both a lower computation time and an improvement in the quality of the sequence of viewpoints. Using cuts between static cameras in the Toric Space provides a real improvement in the number of successful frames *w.r.t.* just using a single static camera for the full sequence (up to a 60% improvement). In our tests, it further enables reaching interactive computation performances even at highest resolution (20×20×20). Though, in some cases, editing may lead to more occlusions of subjects, the possibility to perform cuts when necessary enables reducing the mean duration of these occlusions. In a logical manner, increasing the sampling density also enables reducing the number of occlusions. When the sampling density becomes too important, one can however observe a rise in both the number of occlusions and their mean duration. In our test, the best results for cuts between static cameras are obtained by using an intermediate sampling density (10×10×10). Using dynamic cameras, instead of a static camera, in the Toric Space for each shot is a mean to reduce the number of cuts necessary (up to 60%), and to reduce both the number and mean duration of occlusions. This appears more clearly at high resolutions (15×15×15 and 20×20×20). The computation time is however close to three times higher. As a conclusion, it clearly appears that both editing, camera motions in Toric space and the use of a higher sampling resolution yield improvements in the quality of the viewpoints, but each at the cost of a higher computation time. Results of combinations of these three features are mitigated. Oner needs to consider a trade-off between the use of camera motions in Toric Space, the use of cuts and the sampling resolution associated with each. One good compromise seems to be the use of a dynamic camera with cuts when necessary, while using an intermediate sampling density, for instance 15×15×15. In our tests, these settings provide the best results in terms of number and mean duration of occlusions, while providing interactive computation performances (about 27fps in average).

9 Discussion

Though our approach is efficient, some aspects can be improved.

Projected size We here used a double approximation during the resolution: (i) a subject is assimilated to a sphere and (ii) a given subject size is approximated to a unique distance to the sphere, whatever the view angle. Theoretically, the approximation could benefit from the use of an aspect graph of the subject geometry, that would

5×5×5

Type		**SC**	**DC**	**SC + Editing**	**DC + Editing**
# frames		2792	2792	2792	2792
# cuts		1	1	25.5	12.5
# successful frames		1388.5 (49.7%)	2398 (86.9%)	1940 (69.5%)	1931 (69.2%)
# occlusions of A		29	24.5	35	25
duration (s)	**mean**	0.44	0.38	0.34	0.26
	(std dev.)	(0.83)	(0.83)	(0.58)	(0.31)
# occlusions of B		28.5	29.5	38	33
duration (s)	**mean**	0.44	0.33	0.37	0.30
	(std dev.)	(0.84)	(0.67)	(0.57)	(0.32)
# occlusions of both		23.5	22.5	30.5	22
duration (s)	**mean**	0.46	0.35	0.37	0.27
	(std dev.)	(0.90)	(0.75)	(0.62)	(0.33)
Time per frame (ms)		1.021	4.922	2.053	2.360

Table 5.2 – Comparison of performances of constraints enforcement techniques, for a sampling density 5×5×5 ($d\varphi$, $d\theta$ and $d\alpha$). SC: single camera; DC: dynamic camera.

10×10×10

Type		**SC**	**DC**	**SC + Editing**	**DC + Editing**
# frames		2792	2792	2792	2792
# cuts		1	1	37	15.5
# successful frames		1322.5 (47.4%)	2529.5 (91.6%)	2320 (83.1%)	2265 (81.1%)
# occlusions of A		32	20	24.5	25
duration (s)	**mean**	0.43	0.36	0.29	0.35
	(std dev.)	(0.85)	(0.66)	(0.39)	(0.64)
# occlusions of B		34.5	23.5	36.5	30
duration (s)	**mean**	0.40	0.31	0.30	0.31
	(std dev.)	(0.83)	(0.61)	(0.39)	(0.59)
# occlusions of both		28.5	16.5	24	23
duration (s)	**mean**	0.41	0.33	0.30	0.31
	(std dev.)	(0.90)	(0.71)	(0.40)	(0.66)
Time per frame (ms)		0.984	20.416	3.906	14.320

Table 5.3 – Comparison of performances of constraints enforcement techniques, for a sampling density 10×10×10 ($d\varphi$, $d\theta$ and $d\alpha$). SC: single camera; DC: dynamic camera.

15×15×15

Type		SC	DC	SC + Editing	DC + Editing
# frames		2792	2792	2792	2792
# cuts		1	1	32	14
# successful frames		1322 (47.3%)	2551.5 (91.4%)	2244.5 (80.4%)	2459 (88.1%)
# occlusions of A		30.5	19.5	28.5	19
duration (s)	mean	0.45	0.34	0.41	0.19
	(std dev.)	(0.84)	(0.63)	(0.80)	(0.23)
# occlusions of B		32	24	32	29
duration (s)	mean	0.42	0.29	0.38	0.25
	(std dev.)	(0.83)	(0.57)	(0.76)	(0.27)
# occlusions of both		26	17	24	18
duration (s)	mean	0.45	0.29	0.43	0.19
	(std dev.)	(0.90)	(0.66)	(0.86)	(0.24)
Time per frame (ms)		0.952	42.732	13.157	36.925

Table 5.4 – Comparison of performances of constraints enforcement techniques, for a sampling density 15×15×15 ($d\varphi$, $d\theta$ and $d\alpha$). SC: single camera; DC: dynamic camera.

20×20×20

Type		SC	DC	SC + Editing	DC + Editing
# frames		2792	2792	2792	2792
# cuts		1	1	40	15
# successful frames		1331.5 (47.7%)	2575 (92.2%)	2269 (81.3%)	2334.5 (83.6%)
# occlusions of A		31.5	23	28.5	24.5
duration (s)	mean	0.43	0.28	0.39	0.30
	(std dev.)	(0.88)	(0.51)	(0.75)	(0.56)
# occlusions of B		33	25	33.5	29
duration (s)	mean	0.42	0.26	0.35	0.27
	(std dev.)	(0.87)	(0.49)	(0.70)	(0.51)
# occlusions of both		28	18.5	24	21
duration (s)	mean	0.42	0.24	0.39	0.26
	(std dev.)	(0.93)	(0.54)	(0.80)	(0.57)
Time per frame (ms)		1.083	97.044	31.419	87.722

Table 5.5 – Comparison of performances of constraints enforcement techniques, for a sampling density 20×20×20 ($d\varphi$, $d\theta$ and $d\alpha$). SC: single camera; DC: dynamic camera.

(potentially) lead to a more precise computation. This could however be at the cost of a less generic technique. Another problematic of such a resolution would be the time necessary to compute the solution set, and to combine this solution set to the solution sets of other constraints.

Visibility Ray cast techniques provide very local visibility information (point to point check), while computationally expensive. A possibility offered by the Toric Space would be to express the subject's visibility as a spheric depth map around the subject, then cast this depth map onto the plane (β^*, φ). The remaining problem with such a computation is related to (i) how to compute the spheric depth map, (ii) how to map the depth map information onto a 2D manifold surface (iii) how to explore this information and (iv) how to overall reach a low computational cost for each step.

Anticipation To further improve results, we could use the inherent properties of the Toric Space to predict potential occlusions of subjects or that a given visual constraint will no longer be satisfied from the current viewpoint in the Toric Space at time $t + \Delta t$. From a predicted state of the world (positions and orientation of each relevant scene element, and particularly key subjects) at time $t + \Delta t$, one could compute the camera viewpoint (world position and orientation) corresponding to a given position in the Toric Space (current position or target position in case of camera motions or cuts). This camera viewpoint, together with the predicted state of the scene at time $t + \Delta t$, would provide a good mean to test the constraints satisfaction at time $t + \Delta t$. It could therefore assist our system in the decision of initiating a new camera motion or a new cut to prevent future issues, then in the computation of a target camera viewpoint (potentially statisfying all constraints at time $t + \delta t$) to which move or cut to.

Over-constrained problem In case of an over-constrained problem (*i.e.* if there is at least one conflict in the application of constraints), two trails could be investigated.

Trail #1 The algebraic solutions we have provided can serve as a good starting point to the use of an optimization-based method. Initial samples could then be taken from solution sets of each constraints, and a cost-minimization process (using the satisfaction measures we provided could finally be operated so as to quickly converge toward an approximate solution.

Trail #2 A set of partial solution S' (*i.e.* satisfying a solvable sub-set of constraints) could be computed. This solution set could then used as the starting point to the application of an optimization-based technique, by sampling S' then operating a cost-minimization on these samples through the use of the satisfaction functions we provided for each constraint. One problematic would here be the provision of techniques for constructing optimal solvable sub-sets of constraints.

More generally, these two trails lead to a central problematic closely related to proposing good sampling heuristics and optimization techniques that adapt to the shape of the solution sets.

10 Conclusion

In this chapter, we have introduced a parametric model (the Toric Manifold) to solve a range of problems that occur in the task of positioning a virtual camera given exact on-screen specifications. Our model solves Blinn's spacecraft problem [Bli88] by using an algebraic formulation rather than an iterative process. It casts simple camera optimization problems mostly conducted in 6D into searches inside a 2D space on a manifold surface. Though the solution for two subjects appears easy to formulate with vector algebra, it has not been reported before and the model serves as an expressive way on which to build more evolved techniques.

We extended our model as a 3D space (the Toric Space) in which we integrated most of the classical visual properties employed in the literature [RCU10]. The size of key subjects (or distance to camera), vantage angle and a softer on-screen positioning of subjects are expressed as 2D solution sets in the Toric Space. We have detailed the theoretical resolution of the combination of any number or such solution sets (note that this combination is difficult, if not impossible, to perform in the classical 6D camera space). We have presented an efficient technique to perform this combination to compute a representative set of solutions viewpoints, and a "smart" visibility estimation to compute the best candidate camera. Because of the reduction in the search space inherent to our Toric Space, the benefits in terms of computational cost greatly favors our approach.

The techniques presented in this chapter have the potential to replace a number of previous formulations related to camera control with a simpler and more efficient approach, and opens great possibilities to include more evolved on-screen composition techniques in a large range of applications in computer graphics.

One interesting problem to investigate would be the planning of cinematic camera motions in Toric Space, in static scenes as well as in dynamic scenes. This however give rise to problematics related to planning a global path in this space, while enforcing a range of visual constraints. In the same way, another interesting problem is about the planning of a camera motion from a two-subject configuration to a single-subject configuration, or to another two-subject configuration.

Our Toric Space could also be used to position lights, as light placement and camera placement have quite similar constraints. There however raises the problem related to expressing the set of constraints pertaining to light placement (expected visual effect) into either constraints similar to the ones we introduced earlier (*e.g.* distance, vantage angle), or as constraints that can be integrated in the Toric Space. To go further, in a cinematic context, the Toric Space could then serve as a good basis for controlling the staging (character placement), the camera and the lights simultaneously, as all three steps are closely linked to each other.

Conclusion

6

This thesis has focused on virtual camera control in interactive and non-interactive 3D environments. After identifying key issues, we have presented three contributions that are now summarized. We then conclude this manuscript by proposing perspectives for future research in the field.

In this thesis, we introduced the motivations and issues of controlling a virtual camera in 3D environments. We presented a state of the art of existing camera control techniques, ranging from purely manual to fully automated control of camera parameters. We highlighted that the research community is oriented toward a control that is more and more automated and models of cinematic knowledge that are more and more expressive. We identified three essential aspects in controlling a virtual camera: the viewpoint computation, the planning of camera motions and the editing. We identified another essential element, the handling of key subjects' visibility, which has been under-addressed in the field. We then identified three limitations in existing automated techniques. First, they generally focus on one or two of the aspects: visual composition (viewpoint computation), tracking of targets (viewpoint computation and planning), cinematography (editing, with no account for the visibility of subjects). The four elements (viewpoint, planning, editing and visibility) considered simultaneously are however an essential basis for constructing more evolved camera control and virtual cinematography techniques. Existing techniques further lack expressiveness, *i.e.* they do not provide means to address the modeling of directorial style and genre. Second, there is an interest in supporting the user's creativity when building a cinematographic sequence, by weaving automated computation steps and user interactivity. This aspect has only very partially been addressed in the literature. There exist interactive techniques enabling a finer control of some cinematic aspects (visual layout of elements on the screen, "cinematic-like" camera motions) but suffer from a tedious handling of the editing aspect. Third, the central element in controlling a camera is the visual composition (*i.e.* on-screen layout of key elements). Composition problems are generally casted into non-linear optimization problems in a 7 degree-of-freedom search space. Given the size of the search space and the computational cost in the evaluation of composition properties (typically the visibility of subjects), this optimization process is a time-consuming task and hampers the use of evolved composition techniques in most applications.

From these considerations, we identified three research axes:

- towards a fully integrated cinematography system, *i.e.* which handles viewpoint computation, planning, editing and the visibility of subjects interactively, while accounting for elements of cinematographic style;

- towards an interactive approach which assists the user in the task of constructing a movie, while having a degree of control on the final edit, through a hybrid approach which combines automated computation with direct user interaction;
- towards an efficient approach to the virtual camera composition problem.

Summary of our contributions

A cinematic engine for interactive 3D environments

We have proposed a unifying approach to the problem of interactive cinematography, as a fully integrated cinematic engine (CineSys) which handles viewpoint, editing and planning in a real-time context. We have introduced an expressive editing model, which addresses the innate complexity of problems such as visibility determination and path planning required in real-time camera control, and tackles higher-level issues related to continuity between successive shots. Our real-time cinematic engine encodes cinematographic idioms and continuity-editing rules to produce appropriate edits and camera paths from a set of narrative events and style indicators. We have introduced a novel spatial partitioning, the *Director Volumes*, on which our model relies. Director Volumes provide a characterization into visibility regions (with full visibility, partial visibility or full occlusion) and stereotypical viewpoints (the Semantic Volumes). Our cinematic engine then reason on these Director Volumes to identify how, when and where shot transitions should be performed. This semantic and geometric reasoning relies on a filtering-based encoding of cinematic conventions. Our reasoning process is further expressive enough to offer the possibility to implement different directorial styles. We have demonstrated the expressiveness of our model on a range of style indicators (visibility of subjects, pacing, camera dynamicity, narrative dimension). The expressiveness of our model stands in stark contrast to existing approaches that are either procedural in character, non-interactive or do not account for proper visibility of key subjects.

Integrating director's input into the editing process

We have introduced a novel framework for virtual cinematography and editing which adds an evaluation of the quality of shots, cuts and pacing. We have further proposed two ranking-based approaches to editing a movie, that build upon CineSys and proposed evaluation metrics to assist a filmmaker in his creative process. We have introduced an efficient search strategy for finding the best sequence of shots from a large number of candidates generated from traditional film idioms, while providing the user with some control on the final edit. We have further enabled enforcing the pace in cuts by relying on a well-founded model of shot durations. We have then presented an interactive assistant whose result is a novel workflow based on interactive collaboration of human creativity with automated intelligence. This workflow enables efficient exploration of a wide range of cinematographic possibilities, and rapid production of computer-generated animated movies.

An efficient approach to Virtual Camera Control: the Toric Space

We have introduced a novel and efficient approach to virtual camera control which has the potential to replace a number of previous formulations related to camera control and opens great possibilities to include more evolved on-screen composition techniques in a large range of applications in computer graphics.

First, we have introduced a simple parametric model (the *Toric Manifold*) which solves a two-subject exact on-screen positioning problem. We then used this concept to tackle a range of problems that occur in the task of positioning a virtual camera given exact on-screen specifications. Noticeably, we have proposed the first method to solve Blinn's spacecraft problem [Bli88] by using an algebraic formulation rather than an iterative process. As well, we have shown how to cast simple camera optimization problems mostly conducted in 6D into searches inside a 2D space on a manifold surface.

Second, we have extended our model as a 3D space (the *Toric Space*) in which we have integrated most of the classical visual properties expressed in the literature [RCU10]. We have expressed the size of key subjects (or distance to camera), vantage angle and a softer on-screen positioning of subjects as 2D solution sets in the Toric Space. We have detailed the theoretical resolution of the combination of any number or such solution sets, where such combination is really difficult (if not impossible) to perform in the classical 6D camera space. We have further shown how continuity editing rules can be easily expressed as a set of constraints (*e.g.* distance, vantage) and combined with other constraints in Toric Space. We have finally presented an efficient technique to compute a representative set of solutions viewpoints, and a "smart" visibility estimation to compute the best candidate viewpoint. Because of the reduction in the search space inherent in our Toric Space, the benefits in terms of computational cost greatly favors our approach.

Perspectives

In this thesis, we attempted to overcome limitations of existing camera control approaches in the placement of a virtual camera in 3D environments. However, virtual camera control is a very complex problem which is not limited to the aspects we addressed in the thesis. We here enumerate four points of interest which shape the future of our research in the field.

Learning from real input data

From a user-created edit one can extract indicators on the user style and preferences. Extracting indicators could then serve to better approximate a given directorial style. In this thesis, we already extracted a simple indicator from user input: the preference in transitions from one shot to the following shot, as transition matrices. Learning a directorial style is however more complicated, as it raises questions such as what indicator(s) do we learn, and from which input data? For instance, in case of shot

preference, one further needs to differentiate the various causes of a given preference for a shot over another: which composition criterias causes this choice? which link has this choice with other composition criteria? which link has this choice with previous/next shot(s)? Learning such indicators also raises the question about the influence one indicator may have on another, or about the influence of the set of narrative events on the indicator values (*e.g.* the pacing is strongly dependent of the scheduling of events). This learning process finally raises the question about how to re-apply such indicators as an input to an automated system. All these issues must be addressed if we want to properly learn elements of directorial style, and be capable of successfully re-applying this style to a different set of narrative events.

Visual composition

The aesthetic of a visual composition is more complex than just the geometric visual constraints which are commonly studied in the field (*e.g.* on-screen position, distance, vantage angle). For instance, the consideration of more subtle features, such as visual balance, weight or convergence lines represents a step in the provision of more expressive composition techniques, which could further greatly improve the quality of computed viewpoints.

Study of camera parameters linked to human vision

To go further in computing more qualitative viewpoints, one should also take an interest in additional camera parameters, such as stereoscopy and depth of field, which cause a modification in the visual perception of depth in images. Stereoscopy provides a depth information through the use of two separate images (*i.e.* two slightly different camera viewpoints, one for the left eye and one for the right eye) reproducing the parallax effect. The depth of field is basically an intrinsic consequence of camera lenses, which results in a sharp region (around the focus point) and a blurred region in the image. Note that for the depth of field parameter, one camera viewpoint can result in different images (depending on the focus point), and could then potentially serve as a new mean to perform a transition, by changing the focal point instead of changing the camera viewpoint. The study of these two parameters could further, through experiments, provide interesting information on the impact such depth perceptions may have on other visual features.

Narrative discourse

The narrative discourse addresses the scheduling of narrative events to convey, as well as the camera shots and cuts, so as to fulfill communicative goals (*e.g.* convey that Bill has been shot and that John is the murderer, but without showing the gun). This is a challenging problem which raises issues such as determining the potential cognitive state of a viewer, at each time step, depending on the sequence of shots and on the information contained in each shot; and finding a scheduling of narrative events (with or without time ellipses) and associated camera placements that make the viewer

follow a given cognitive path (as a sequence of changes in cognitive states). Note that each cognitive path may represent either a different way to tell the same story or to tell a completely different story (*e.g.* Bill survived, the murderer has not been found or we did not see who has been shot). In that sense, the integration of narrative discourse with evolved cinematographic techniques represents a key step in reaching more expressiveness in cinematography applications.

Bibliography

[ACoYL08] J. Assa, D. Cohen-or, I.-C. Yeh, and T. Lee. Motion Overview of Human Actions. *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH Asia 2008*, 27(5), December 2008. ix, 25, 27, 32

[ACS+11] R. Abdullah, M. Christie, G. Schofield, C. Lino, and P. Olivier. Advanced Composition in Virtual Camera Control. In *Smart Graphics*, volume 6815 of *Lecture Notes in Computer Science*, pages 13–24. Springer Berlin Heidelberg, 2011. 21, 19

[AKY05] D. Amerson, S. Kime, and R. M. Young. Real-time Cinematic Camera Control for Interactive Narratives. In *2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, pages 369–369, Stanford, CA, 2005. 30, 28

[Ari76] D. Arijon. *Grammar of the Film Language*. Hastings House Publishers, 1976. 7, 12, 47, 105, 5, 10, 45, 103

[AVF04] C. Andújar, P. Vázquez, and M. Fairén. Way-Finder: Guided Tours Through Complex Walkthrough Models. *Computer Graphics Forum*, 23(3):499–508, Septembre 2004. 26, 27, 61, 25, 59

[AWCO10] J. Assa, L. Wolf, and D. Cohen-Or. The Virtual Director: a Correlation-Based Online Viewing of Human Motion. *Computer Graphics Forum*, 29(2):595–604, January 2010. 32, 41, 80, 39

[BC11] T. Berliner and D. J. Cohen. The Illusion of Continuity: Active Perception and the Classical Editing System. *Journal of Film and Video*, 63(1):44–63, Spring 2011 2011. 87, 85

[BDER08] P. Burelli, L. Di Gaspero, A. Ermetici, and R. Ranon. Virtual Camera Composition with Particle Swarm Optimization. In *Smart Graphics*, volume 5166 of *Lecture Notes In Computer Science*, pages 130–141. Springer Berlin Heidelberg, 2008. 22, 21

[Bec02] S. Beckhaus. *Dynamic Potential Fields for Guided Exploration in Virtual Environments*. PhD thesis, Fakultät für Informatik, University of Magdeburg, September 2002. 26, 24

[BJ09] P. Burelli and A. Jhala. Dynamic Artificial Potential Fields for Autonomous Camera Control. In *5th Conference on Artificial Intelligence and Interactive Digital Entertainment*, Palo Alto, California, USA, 2009. 25, 26, 24

[BK00] R. Bohlin and L. E. Kavraki. Path Planning using Lazy PRM. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 521–528, San Francisco, CA, 2000. 29

[BKF+02] N. Burtnyk, A. Khan, G. Fitzmaurice, R. Balakrishnan, and G. Kurtenbach. StyleCam: Interactive Stylized 3D Navigation using Integrated Spatial and Temporal Controls. In *15th annual ACM Symposium on User Interface Software and Technology*, pages 101–110, Toronto, Canada, 2002. 14, 12

[BL97] W. H. Bares and J. C. Lester. Cinematographic User Models for Automated Realtime Camera Control in Dynamic 3D Environments. In *6th International Conference on User Modeling*, pages 215–226, 1997. 31, 29

[BL99] W. H. Bares and J. C. Lester. Intelligent Multi-shot Visualization Interfaces for Dynamic 3D Worlds. In *International Conference on Intelligent User Interfaces*, pages 119–126, Los Angeles, CA, USA, 1999. 19, 17

[BLAH06] T. Bandyopadhyay, Y. Li, M. H.. Jr Ang, and D. Hsu. A Greedy Strategy for Tracking a Locally Predictable Target among Obstacles. In *IEEE International Conference on Robotics and Automation*, pages 2342–2347, Orlando, FL, USA, 2006. 37, 34

[Bli88] J. Blinn. Where Am I? What Am I Looking At? *IEEE Computer Graphics and Applications*, 8(4):76–81, July 1988. iv, 3, 17, 117, 118, 119, 123, 160, 163, 15, 16, 115, 116, 121, 157, 161

[BMBT00] W. H. Bares, S. McDermott, C. Boudreaux, and S. Thainimit. Virtual 3D Camera Composition from Frame Constraints. In *Eighth ACM International Conference on Multimedia*, pages 177–186, 2000. 18, 22, 41, 16, 21, 39

[Bro86] F. P. Brooks. Walkthrough—A Dynamic Graphics System for Simulating Virtual Buildings. In *Workshop on Interactive 3D graphics*, pages 9–21, 1986. 14, 12

[BTM00] W. H Bares, S. Thainimit, and S. Mcdermott. A Model for Constraint-Based Camera Planning. In *AAAI Spring Symposium on Smart Graphics*, pages 84–91, 2000. 20, 19

[BY10] P. Burelli and G. N. Yannakakis. Global Search for Occlusion Minimisation in Virtual Camera Control. In *IEEE Congress on Evolutionary Computation*, pages 1–8, Barcelona, Spain, 2010. 26, 24

[BZRL98] W. H. Bares, L. S. Zettlemoyer, D. W. Rodriguez, and J. C. Lester. Task-Sensitive Cinematography Interfaces for Interactive 3D Learning Environments. In *International Conference on Intelligent User Interfaces*, pages 81–88, San Francisco, CA, USA, 1998. 19, 23, 17, 22

[CAH+96] D. B. Christianson, S. E. Anderson, L.-W. He, D. H. Salesin, D. S. Weld, and M. F. Cohen. Declarative camera Control for Automatic Cinematography. In *AAAI'96 Proceedings of the Thirteenth National Conference on Artificial Intelligencec*, volume 1, pages 148–155, 1996. ix, 30, 31, 41, 80, 152, 28, 29, 39, 78, 151

[Cha80] S. B. Chatman. *Story and Discourse: Narrative Structure in Fiction and Film*. Cornell University Press, 1980. 41

[CKL+98] R. C. H. Chiou, A. E. Kaufman, Z. Liang, L. Hong, and M. Achniotou. Interactive Path Planning for Virtual Endoscopy. In *IEEE Nuclear Science Symposium*, volume 3, pages 2069 – 2072. Toronto, Canada, 1998. 25, 24

[CLGL02] M. Christie, E. Languénou, L. Granvilliers, and E. Languénou. Modeling Camera Control with Constrained Hypertubes. In *Principles and Practice of Constraint Programming*, volume 2470 of *Lecture Notes in Computer Science*, pages 618–632. Springer Berlin Heidelberg, 2002. 24, 23

[CM01] N. Courty and E. Marchand. Computer Animation: A new Application for Image-based Visual Servoing. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 223–228, 2001. 24, 22

[CMS88] M. Chen, S. J. Mountford, and A. Sellen. A Study in Interactive 3-D Rotation using 2-D Control Devices. *ACM SIGGRAPH Computer Graphics*, 22(4):121–129, August 1988. 14, 12

[CN05] M. Christie and J.-M. Normand. A semantic Space Partitioning Approach to Virtual Camera Composition. *Computer Graphics Forum*, 24(3):247–256, September 2005. ix, 22, 23, 44, 21, 42

[COCS+03] D. Cohen-Or, Y. L. Chrysanthou, C. T. Silva, F. Durand, and Y. Chrysanthou. A Survey of Visibility for Walkthrough Applications. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):412–431, July 2003. 34, 48, 31, 46

[CON08a] M. Christie, P. Olivier, and J.-M. Normand. Camera Control in Computer Graphics. *Computer Graphics Forum*, 27(8):2197–2218, December 2008. ix, 7, 13, 16, 39, 5, 34, 36

[CON08b] M. Christie, P. Olivier, and J.-M. Normand. Occlusion-free Camera Control. Technical Report RR-6640, INRIA, 2008. 37

[DD95] D. Dementhon and L. Davis. Model-Based Object Pose in 25 Lines of Codes. *International Journal of Computer Vision*, 15(1-2):123–141, 1995. 126

[DER08] L. Di Gaspero, A. Ermetici, and R. Ranon. Swarming in a Virtual World: A PSO Approach to Virtual Camera Composition. In *Ant Colony Optimization and Swarm Intelligence*, volume 5217 of *Lecture Notes in Computer Science*, pages 155–166. Springer Berlin Heidelberg, 2008. 21, 19

[DGZ92] S. M. Drucker, T. A.. Galyean, and D. Zeltzer. CINEMA: A System for Procedural Camera Movements. In *1992 Symposium on Interactive 3D Graphics*, pages 67–70, 1992. 14, 12

[DHD00] R.O. Duda, P.E. Hart, and Stork D.G. *Pattern Classification (2nd ed.)*. Wiley Interscience, 2000. 101, 99

[DRLR89] M. Dhome, M. Richetin, J.-T. Lapresté, and G. Rives. Determination of the Attitude of 3D Objects from a Single Perspective View. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1265–1278, December 1989. 126

[DV90] G. D'Ydewalle and M. Vanderbeeken. *Perceptual and Cognitive Processing of Editing Rules in Film*. 1990. 87, 85

[DZ95] S. M. Drucker and D. Zeltzer. CamDroid: A System for Implementing Intelligent Camera Control. In *1995 Symposium on Interactive 3D Graphics*, pages 139–144, 1995. 22, 20

[ECR92] B. Espiau, F. Chaumette, and P. Rives. A New Approach to Visual Servoing in Roboticsh. *IEEE Transactions on Robotics and Automation*, 8(3):313–326, June 1992. 24, 22

[ER07] D. K. Elson and M. O. Riedl. A Lightweight Intelligent Virtual Cinematography System for Machinima Production. In *3rd Annual Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 8–13, 2007. ix, 33, 34, 41, 79, 80, 152, 29, 31, 39, 77, 78, 151

[FB81] N. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Application to Image Analysis and Automated Cartography. *Communication of the ACM*, 24(6):381–395, June 1981. 126

[FF04] D. A. Friedman and Y. A. Feldman. Knowledge-based Cinematography and its Applications. In *16th European Conference on Artificial Intelligence*, pages 256–260, Amstredam, The Netherlands, 2004. 30, 28

[Ger09] R. Geraerts. Camera Planning in Virtual Environments using the Corridor Map Method. In *Motion in Games*, volume 5884 of *Lecture Notes in Computer Science*, pages 194 – 206. Springer Berlin Heidelberg, 2009. 29, 27

[GO07] R. Geraerts and M. H. Overmars. The Corridor Map Method: A General Framework for Real-time High-quality Path Planning. *Computer Animation and Virtual Worlds*, 18(2):107–119, May 2007. 28, 29, 30, 25, 27

[GW92] M. Gleicher and A. Witkin. Through-the-lens Camera Control. *ACM SIGGRAPH Computer Graphics*, 26(2):331–340, July 1992. 15, 24, 13

[HCLL89] R. Horaud, B. Conio, O. Leboulleux, and B. Lacolle. An Analytic Solution for the Perspective 4-points Problem. *Computer Vision, Graphics and Image Processing*, 47(1):33–44, July 1989. 126

[HCS96] L. W. He, M. F. Cohen, and D. H. Salesin. The Virtual Cinematographer: A Paradigm for Automatic Real-time Camera Control and Directingh. In *SIGGRAPH '96 Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 217–224, 1996. ix, 30, 31, 80, 152, 28, 78, 151

[HHS01] N. Halper, R. Helbing, and T. Strothotte. A Camera Engine for Computer Games: Managing the Trade-Off Between Constraint Satisfaction and Frame Coherence. In *Computer Graphics Forum*, volume 20, pages 174–183, September 2001. ix, 19, 35, 36, 37, 17, 18, 32, 33, 34

[HKB+97] L. Hong, A. Kaufman, D. Bartz, S. Muraki, and T. He. Virtual Voyage: Interactive Navigation in the Human Colon. In *SIGGRAPH '97 Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 27–34, 1997. 25, 24

[HO00] N. Halper and P. Olivier. CamPlan: A Camera Planning Agent. In *Smart Graphics AAAI Spring Symposium*, pages 92–100, 2000. 21, 19

[Hor87] R. Horaud. New Methods for Matching 3D Objects with Single Perspective Views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(3):401–412, May 1987. 126

[HW97] A. J. Hanson and E. A. Wernert. Constrained 3D Navigation with 2D Controllersk. In *8th Conference on Visualization*, pages 175–182,, 1997. 14, 12

[JL98] F. Jardillier and E. Languénou. Screen-Space Constraints for Camera Movements: The Virtual Cameraman. *Computer Graphics Forum*, 17(3):175–186, September 1998. 24, 23

[JPK98] M.R. Jung, D. Paik, and D. Kim. A Camera Control Interface Based on the Visualization of Subspaces of the 6D Motion Space of the Camera. In *6th Pacific Conference on Computer Graphics and Applications*, pages 198–206, 1998. 14, 12

[JY05] A. Jhala and R. M. Young. A Discourse Planning Approach to Cinematic Camera Control for Narratives in Virtual Environments. In *AAAI'05 Proceedings of the 20th National Conference on Artificial Intelligence*, volume 1, pages 307–312, 2005. 80, 152, 78, 151

[JY10] A. Jhala and M. Young. Cinematic Visual Discourse: Representation, Generation, and Evaluation. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(2):69–81, June 2010. 34, 30

[Kat91] S. D. Katz. *Film Directing Shot by Shot: Visualizing from Concept to Screen*. Michael Wiese Productions, 1991. 7, 5

[KC08] K. Kardan and H. Casanova. Virtual Cinematography of Group Scenes using Hierarchical Lines of Actions. In *2008 ACM SIGGRAPH Symposium on Video Games*, pages 171–178, 2008. ix, 32, 33, 113, 29, 30, 111

[KC09] K. Kardan and H. Casanova. Heuristics for Continuity Editing of Cinematic Computer Graphics Scenes. In *2009 ACM SIGGRAPH Symposium on Video Games*, pages 63–69, 2009. 33, 29

[Kha86] O. Khatib. Real-time Obstacle Avoidance for Manipulators and Mobile Robots. *IEEE International Journal of Robotics and Automation*, 2:500 – 505, 1986. 25, 23, 24

[KKH95] M. H. Kyung, M. S. Kim, and S. Hong. Through-the-lens Camera Control with a Simple Jacobian Matrix. In *Graphics Interface*, pages 117–178, 1995. 16, 15

[KKS+05] A. Khan, B. Komalo, J. Stam, G. Fitzmaurice, and G. Kurtenbach. HoverCam : Interactive 3D Navigation for Proximal Object Inspection. In *2005 Symposium on Interactive 3D Graphics and Games*, pages 73–80, 2005. 14, 12

[KL94] L. Kavraki and J. C. Latombe. Randomized Preprocessing of Configuration for Fast Path Planning. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2138–2145, San Diego, CA, 1994. 28, 25

[KL00] J. J. Kuffner Jr. and S. M. LaValle. RRT-Connect: An Efficient Approach to Single-Query Path Planning. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 995–1001, San Francisco, CA, 2000. 28, 25, 26

[KV79] J. J. Koenderink and A. J. Van Doorn. The Internal Representation of Solid Shape with Respect to Vision. *Biological Cybernetics*, 32(4):211–216, January 1979. 22, 21

[Lat91] J.-C. Latombe. *Robot Motion Planning*. Kluwer Editions, 1991. 24, 22

[LaV06] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, 2006. 24, 22

[LC08] T.-Y. Li and C. C. Cheng. Real-Time Camera Planning for Navigation in Virtual Environments. In *Smart Graphics*, volume 5166 of *Lecture Notes in Computer Science*, pages 118–129. Springer Berlin Heidelberg, 2008. ix, 29, 32, 33, 80, 152, 27, 30, 78, 151

[LT00] T.-Y. Li and H.-K. Ting. An Intelligent User Interface with Motion Planning for 3D Navigation. In *IEEE Virtual Reality*, pages 177–184, 2000. 28, 26

[Mas65] J. Mascelli. *The Five C's of Cinematography: Motion Picture Filming Techniques*. Cine/Grafic Publications, Hollywood, 1965. 7, 5

[NO04] D. Nieuwenhuisen and M. H. Overmars. Motion Planning for Camera Movements. *IEEE International Conference on Robotics and Automation*, 4:3870 – 3876, 2004. 28, 29, 26, 27

[OHPL99] P. Olivier, N. Halper, J. H. Pickering, and P. Luna. Visual Composition as Optimisation. In *AISB Symposium on on AI and Creativity in Entertainment and Virsual Art*, pages 22–30, 1999. 20, 19

[Ond02] M. Ondaatje. *The Conversations: Walter Murch and the Art of Editing Film*. Knopf, 2002. 87, 85

[OSTG09] T. Oskam, R. W. Sumner, N. Thuerey, and M. Gross. Visibility Transition Planning for Dynamic Camera Control. In *2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 55–65, 2009. ix, 36, 38, 41, 77, 33, 35, 39, 76

[Ove92] M. H. Overmars. A Random Approach to Motion Planning. Technical report RUU-CS-92-32, Dept. Comput. Sci., Utrecht University, 1992. 28, 25, 26

[Pal96] P. Palamidese. A Camera Motion Metaphor Based on Film Grammar. *The Journal of Visualization and Computer Animation*, 7(2):61–78, April 1996. 23, 22

[PBY12] M. Preuss, P. Burelli, and G. Yannakakis. Diversified Virtual Camera Composition. In *Applications of Evolutionary Computation*, volume 7248 of *Lecture Notes in Computer Science*, pages 265–274. Springer Berlin Heidelberg, 2012. 21, 19

[PD90] H. Plantinga and C. R. Dyer. Visibility, Occlusion, and the Aspect Graph. *International Journal of Computer Vision*, 5(2):137–160, November 1990. 22, 21

[PO03] J. Pickering and P. Olivier. Declarative Camera Planning Roles and Requirements. In *Smart Graphics*, volume 2733 of *Lecture Notes in Computer Science*, pages 182–191. Springer Berlin Heidelberg, 2003. 22, 21

[RCU10] R. Ranon, M. Christie, and T. Urli. Accurately Measuring the Satisfaction of Visual Properties in Virtual Camera Control. In *Smart Graphics*, volume 6133 of *Lecture Notes in Computer Science*, pages 91–102. Springer Berlin Heidelberg, 2010. iv, 3, 20, 160, 163, 18, 157, 161

[Sal03] B. Salt. *Film Style and Technology: History and Analysis (2nd edition)*. Starword, 2003. 94, 92

[Sho92] K. Shoemake. Arcball: a User Interface for Specifying Three-dimensional Orientation using a Mouse. In *Graphics Interface*, pages 151 – 156, 1992. 14, 12

[Smi05] T. J. Smith. *An Attentional Theory of Continuity Editing*. PhD thesis, University of Edinburgh, 2005. 87, 85

[Sny92] J. M. Snyder. Interval Analysis for Computer Graphics. *ACM SIGGRAPH Computer Graphics*, 26(2):121–130, July 1992. 18, 16

[ST85] H. G. Scott and F. Truffaut. *Hitchcock-Truffaut (Revised Edition)*. Simon and Schuster, 1985. 83, 87, 81, 85

[TBG91] R. Turner, F. Balaguer, and E. Gobbetti. *Physically-based Interactive Camera Motion Control using 3D Input Devices*, chapter 2. Springer Japan, 1991. 14, 12

[Tho93] R. Thompson. *Grammar of the Edit*. Focal Press, 1993. 83, 152, 81, 149

[Tho98] R. Thompson. *Grammar of the Shot*. Focal Press, 1998. 8, 83, 6, 81

[TS91] S. J. Teller and C. H. Séquin. Visibility Preprocessing for Interactive Walkthroughs. *ACM SIGGRAPH Computer Graphics*, 25(4):61–70, July 1991. 25, 48, 50, 23, 46

[VFSH03] P.-P. Vàzquez, M. Feixas, M. Sbert, and W. Heidrich. Automatic View Selection Using Viewpoint Entropy and its Application to Image-Based Modelling. *Computer Graphics Forum*, 22(4):689–700, December 2003. 21, 20

[WO90] C. Ware and S. Osborne. Exploration and Virtual Camera Control in Virtual Three Dimensional Environments. *ACM SIGGRAPH Computer Graphics*, 24(2):175–183, March 1990. 13, 11

[YLL12] I.-C. Yeh, W. C. Lin, and T. Y. Lee. Social-Event-Driven Camera Control for Multicharacter Animations. *IEEE Transactions on Visualization and Computer Graphics*, 18(9):1496–1510, September 2012. 32, 80

[ZF99] R. Zeleznik and A. Forsberg. UniCam—2D Gestural Camera Controls for 3D Environments. In *1999 Symposium on Interactive 3D Graphics*, pages 169–173, 1999. 14, 12

[ZM10] J. M. Zacks and J. P. Magliano. *Art and the Senses*, chapter Film, Narrative, and Cognitive Neuroscience. Oxford University Press., 2010. 87, 85

Résumé

Le contrôle de caméra virtuelle est aujourd'hui un composant essentiel dans beaucoup d'applications d'infographie. Malgré cette importance, les approches actuelles restent limitées en terme d'expressivité, d'interactivité et de performances. Typiquement, les éléments de style ou de genre cinématographique sont difficiles à modéliser et à simuler dû à l'incapacité des systèmes actuels de calculer simultanément des points de vues, des trajectoires et d'effectuer le montage. Deuxièmement, elles n'explorent pas assez le potentiel créatif offert par le couplage potentiel d'un humain et d'un système intelligent pour assister les utilisateurs dans une tâche complexe de construction de séquences cinématographiques. Enfin, la plupart des approches existantes se basent sur des techniques d'optimisation dans un espace de recherche 6D, qui s'avèrent coûteuses et donc inadaptées à un contexte interactif. Dans cette thèse, nous proposons tout d'abord un cadre unique intégrant les quatre aspects clés de la cinématographie (le calcul de point de vue, la planification de trajectoires, le montage et la visibilité). Ce cadre expressif permet de simuler certaines dimensions de style cinématographique. Nous proposons ensuite une méthodologie permettant de combiner les capacités d'un système automatique avec une interaction utilisateur. Enfin, nous présentons un modèle de contrôle de caméra efficace qui réduit l'espace de recherche de 6D à 3D. Ce modèle a le potentiel pour remplacer un certain nombre de formulations existantes.

Mots-clés : contrôle de caméra, cinématographie virtuelle, interaction.

Abstract

Virtual camera control is nowadays an essential component in many computer graphics applications. Despite its importance, current approaches remain limited in their expressiveness, interactive nature and performances. Typically, elements of directorial style and genre cannot be easily modeled nor simulated due to the lack of simultaneous control in viewpoint computation, camera path planning and editing. Second, there is a lack in exploring the creative potential behind the coupling of a human with an intelligent system to assist users in the complex task of designing cinematographic sequences. Finally, most techniques are based on computationally expensive optimization techniques performed in a 6D search space, which prevents their application to real-time contexts. In this thesis, we first propose a unifying approach which handles four key aspects of cinematography (viewpoint computation, camera path planning, editing and visibility computation) in an expressive model which accounts for some elements of directorial style. We then propose a workflow allowing to combine automated intelligence with user interaction. We finally present a novel and efficient approach to virtual camera control which reduces the search space from 6D to 3D and has the potential to replace a number of existing formulations.

Keywords: camera control, virtual cinematography, interaction.

ACM Classification

Categories and Subject Descriptors (according to ACM CCS):
I.3.6 [**Computer Graphics**]: Methodology and Techniques–*Interaction Techniques*;
I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism–*Animation*;
H.5.1 [**Information Systems**]: Multimedia Information Systems–*Animations, Video*

General Terms: Algorithms, Human Factors

VU :

Le Directeur de Thèse
(Nom et Prénom)

VU :

Le Responsable de l'École Doctorale

VU pour autorisation de soutenance

Rennes, le

Le Président de l'Université de Rennes 1

Guy CATHELINEAU

VU après soutenance pour autorisation de publication :

Le Président de Jury,
(Nom et Prénom)