Automatic rush generation with application to theatre performances

Vineet Gandhi

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathématiques-Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Vineet Gandhi

Thèse dirigée par **Remi Ronfard**

préparée au sein **du Laboratoire Jean Kuntzmann (LJK)**
et de **l'école doctorale EDMSTII**

Automatic Rush Generation with Application to Theatre Performances

Thèse soutenue publiquement le **18 Dec 2014**,
devant le jury composé de :

James L. Crowley
Professeur, Grenoble I.N.P., Président
Patrick Perez
Distinguished Scientist, Technicolor, Rapporteur
Frederic Jurie
Professeur, Université de Caen, Rapporteur
Alexander Sorkine-Hornung
Senior Research Scientist, Disney Research Zurich, Examinateur
Michael Gleicher
Professeur, University of Wisconsin, Examinateur
Remi Ronfard
Chercheur, Grenoble Université/INRIA, Directeur de thèse

RÉSUMÉ

Les captations professionnelles de pièces de théâtre utilisent un grand nombre de caméras afin de montrer l'ensemble du spectacle sous tous ses angles. C'est un processus complexe et coûteux, qui fait appel aux compétences d'un grand nombre de techniciens qualifiés pour assurer le cadrage puis le montage de toutes les prises de vues. Dans cette thèse, nous explorons une approche différente, consistant à calculer automatiquement en post-production des cadrages dynamiques à partir d'un petit nombre de prises de vues obtenues en caméra fixe, sans opérateurs. Pour atteindre cet objectif, nous proposons de nouveaux algorithmes de vision par ordinateur qui nous permettent de formaliser et reproduire les régles du cadrage cinématographique.

Dans cette thèse, nous proposons de nouvelles méthodes d'analyse vidéo pour calculer automatiquement le cadrage le plus approprié aux mouvements des acteurs qui évoluent sur scène. Nous simulons pour cela les mouvements d'une caméra "pan-tilt-zoom" extraite du cadre d'une prise de vue en caméra fixe. Une contribution importante de la thèse consiste à formaliser le problème du cadrage cinématographique comme un problème d'optimisation convexe.

Dans une première partie de la thèse, nous proposons des méthodes nouvelles pour détecter et reconnaitre les acteurs à l'aide d'une modélisation explicite de leurs apparences, qui inclue leurs caractères physiques ainsi que leurs costumes et chevelures. Nous présentons une approche pour apprendre ces modèles d'apparence à partir d'un petit nombre d'exemples, en maximisant leur vraisemblance. Nous montrons l'efficacité de ces méthodes sur des exemples de films de théâtre et de cinéma.

Dans une seconde partie de la thèse, nous montrons comment ces méthodes peuvent être utilisées pour effectuer le suivi des acteurs d'une pièce de théâtre, y compris sur de longues séquences de plusieurs minutes, par l'utilisation de méthodes efficaces de programmation dynamique, qui permettent de prendre en compte les entrées et sorties de scène des acteurs, ainsi que leurs occultations mutuelles.

Dans une troisième partie de la thèse, nous décrivons une méthode générale pour calculer dynamiquement le cadrage d'une caméra virtuelle incluant un nombre quelconque d'acteurs, tout en excluant les autre acteurs dans la mesure du possible. Notre méthode prend en compte un grand nombre de considérations esthétiques que nous avons extraites des ouvrages techniques consacrés à la cinématographie et au montage. Notre approche présente l'avantage de formaliser les règles de la cinématographie et du montage sous la forme d'un problème d'optimisation convexe, pour lequel nous pouvons proposer une solution efficace.

Tout au long de la thèse, nous illustrons et validons les approches proposées sur des exemples réels et complexes, que nous avons filmés au Théâtre de Lyon - Célestins. Les méthodes que nous proposons s'appliquent généralement au spectacle vivant (théâtre, musique, opéra) et permettent d'envisager de nouvelles applications de la vision par ordinateur dans le domaine de la production audio-visuelle.

ABSTRACT

Professional quality videos of live staged performances are created by recording them from different appropriate viewpoints. These are then edited together to portray an eloquent story replete with the ability to draw out the intended emotion from the viewers. Creating such competent videos typically requires a team of skilled camera operators to capture the scene from multiple viewpoints. In this thesis, we explore an alternative approach where we automatically compute camera movements in post-production using specially designed computer vision methods.

A high resolution static camera replaces the plural camera crew and their efficient camera movements are then simulated by virtually panning - tilting - zooming within the original recordings. We show that multiple virtual cameras can be simulated by choosing different trajectories of cropping windows inside the original recording. One of the key novelties of this work is an optimization framework for computing the virtual camera trajectories using the information extracted from the original video based on computer vision techniques.

The actors present on stage are considered as the most important elements of the scene. For the task of localizing and naming actors, we introduce generative models for learning view independent person and costume specific detectors from a set of labeled examples. We explain how to learn the models from a small number of labeled keyframes or video tracks, and how to detect novel appearances of the actors in a maximum likelihood framework. We demonstrate that such actor specific models can accurately localize actors despite changes in view point and occlusions, and significantly improve the detection recall rates over generic detectors.

The thesis then proposes an offline algorithm for tracking objects and actors in long video sequences using these actor specific models. Detections are first performed to independently select candidate locations of the actor/object in each frame of the video. The candidate detections are then combined into smooth trajectories by minimizing a cost function accounting for false detections and occlusions.

Using the actor tracks, we then describe a method for automatically generating multiple clips suitable for video editing by simulating pan-tilt-zoom camera movements within the frame of a single static camera. Our method requires only minimal user input to define the subject matter of each sub-clip. The composition of each sub-clip is automatically computed in a novel convex optimization framework. Our approach encodes several common cinematographic practices into a single convex cost function minimization problem, resulting in aesthetically-pleasing sub-clips which can easily be edited together using off-the-shelf multi-clip video editing software.

The proposed methods have been tested and validated on a challenging corpus of theatre recordings. They open the way to novel applications of computer vision methods for cost-effective video production of live performances including, but not restricted to, theatre, music and opera.

ACKNOWLEDGMENTS

These three years have been a good learning experience for me both professionally and individually. There is a long list of people who have helped me through this process and I take this opportunity to acknowledge and express my gratitude towards them. First of all, I would like to thank my advisor Remi Ronfard for his guidance and support throughout this period. The thesis would not have been possible without his valuable ideas, advice and suggestions. I express my gratitude towards Marie-Paule Cani, as the team leader she has always been supportive and encouraging. I am grateful to Michael Gleicher, though his presence at Inria was for a short spell, his counsel and suggestions were extremely useful.

Special thanks to my thesis rapporteurs, Patrick Perez and Frederic Jurie, for giving their valuable time to go through the entire manuscript in a relatively short time. I would also like to thank Alexander Sorkine-Hornung and Michael Gleicher, my thesis examiners, for their interest in my work and James Crowley, the thesis committee president, for making his time available for me.

I am thankful to Region Rhone Alpes (project Scenoptique) and ERC advanced grant EXPRESSIVE for the financial support. I would like to thank Cyrille Migniot and Laurent Boiron for their contributions to the work presented in this thesis. Heartfelt thanks to all my friends and colleagues in the Imagine team for making Inria a pleasant and interesting place to work at. Special thanks to Catherine and Fatima, for helping me through the unending piles of French administrative documents.

My work for this thesis bids a mention of gratitude towards the directors Claudia Stavisky and Nathalie Fillion, also the entire cast and crew of *Death of a salesman*, *Lorenzaccio* and *A l'ouest* for letting us record their rehearsal at Celestins, Theatre de Lyon. I also thank Hélène Chambon and Auxane Dutronc for their help, advice and expertise.

I am exceptionally thankful to all my friends in Grenoble, who were nothing less than being a family to me. I refrain from typing a long list of names fearing skipping a couple or more and regretting it later.

I would like to end this by thanking my family, their happiness was at the core of my motivation, to say the least. My parents have supported me relentlessly and have made infinite sacrifices throughout my studies and I do not think I can ever thank them enough. I thank Zeel for her unconditional support for the last two years, sacrificing both her personal and professional life. She has been my constant source of happiness and motivation.

CONTENTS

CHAPTER

1

INTRODUCTION

ARCHIVING the recordings of theatre performances is important for preserving and presenting the cultural heritage. It is also important for professional, educational, or research reasons. Many organizations around the world now actively produce and archive video recordings of theatre performances. The Theatre on Film and Tape Archive (TOFT) provides research access to video recordings of New York and regional theatre productions since 1970 and the collection now amounts to 7469 titles with a total of 20,000 items. The National Video Archive of Performance (NVAP) at Victoria and Albert Museum London provides research access to video recordings of numerous renowned theatre performances produced across England since 1992. The collection of French National Institute of Audiovisual (INA) includes more than 600 full-length recordings of theatre plays. The INA archive includes several versions of the famous plays (for example it has recordings of seven different versions of the famous play '*Le Misanthrope*' by Moliere).

Impermanence in the Arts is often considered good, making the archives valuable. For example, Arthur Millers play '*Death of a Salesman*' first premiered at Broadway theatre, New York in 1949 and its success inspired several movies and theatre re-adaptions over the years. The broadway theatre recently reproduced the play with a similar staging and music, almost 63 years after the original production. The director in an interview said, "*When something is so completely achieved by the people who made it, you better know how they got there, not to include that in your work is to miss the play*". The snapshots from the original archive and several adaptations of the play are shown in Figure 1.1.

Not just for archiving, video production of theatre is also important for broadcast on television, cinema or the web. Theatre performances on television has been appreciated over the years, a successful example is BBC's '*Play of the Month*' series which ran for nearly eighteen years (1965-1983). Recent technologies have even allowed live streaming of theatre in Cinemas. The National Theater company based in London has seen great success with their recent venture NT Live which broadcasts the best of British theatre live from the London stage to

Figure 1.1: *Snapshots from the archive of the first premiere of the play death of the salesman at Broadway Theatre in 1949 (top left), a reproduction by Broadway Theatre using similar staging and music in 2012 (top middle) and several other movie adaptations of the play including the one featuring Dustin Hoffman in 1985 (bottom middle).*

cinemas across the UK and around the world. According to their website, their broadcasts have now been experienced by over 1.5 million people in 500 venues around the world.

The most relevant case study for this thesis is the successful ventures providing the recorded theatre performances on the web. Recently launched companies like Digital Theater, use their own equipment to record theatre productions, then sell them for far less than the cost of an actual theatre ticket to watch online, but from the best spot possible. The website[1] of French National Institute of Audiovisual (INA) also allows viewers to download recorded theatre plays at considerably low prices. You can watch the trailers of theatre performances just like movies on variety of devices and then download the entire play if it is of your interest. The web platforms have also allowed for free form experimentation with recorded sequences, which is not so convenient on actual TV or Cinema. For example, FranceTV channel in 2013 uploaded a webpage[2] showing 5 different versions of the same play '*Theatre sans animaux*', showing both how a theatre play develops in different stages and how it can be presented to the audience in different styles. The web platform has also allowed several theatres to share the extracts and trailers of upcoming performances for publicity purposes. Importantly, web based platforms allow even the small production companies or individuals to easily share the recorded content with the desired audience.

But it is a difficult task to create recorded performances to match up to the live experience of watching a play. When you watch a stage play in an actual public theatre you generally only get to observe the actions of the performers from one static viewpoint – your seat. But if one was to watch a recorded performance from a static viewpoint covering the entire action, it would be too monotonous to hold the interest of the viewer. Recordings from a single static viewpoint may also fail to highlight important details in the scene. Even with these limitations and inability to create a nice viewing experience, this approach is still commonly practiced in small scale productions due to its convenience, requiring just an single static camera covering the entire stage.

The approach which is used in more professional scenarios is to shoot the performance using multiple cameras situated at different viewpoints with each camera being handled by a skilled operator. The editing is then performed between the multiple recorded sequences to create an intimate viewing experience. Editing here refers to first synchronizing the recorded camera sequences in time and then choosing which viewpoint to show at a given time by cutting between the available sequences. Building such video recordings can be extremely costly, requiring large number of equipments with a highly skilled production crew and is reserved only for large budget productions.

Jill Evans, who is a producer at 'National Video Archive of Performance (NVAP)' says that, "*The aim is to do justice to the actors and the performance by using multiple camera positions and filming techniques appropriate to the nature of performance. We do not intend the recordings to be mere museum pieces, because pleasure is an intrinsic part of watching theatre, and we aim to reproduce the director's vision as faithfully as possible. It's not cheap: we buy up to 10 seats for every camera; at West End prices the cost of a three-camera shoot soon mounts up. The cost of shooting a production remains so high, we have to make tough*

[1] http://www.ina.fr

[2] http://nouvelles-ecritures.francetv.fr/theatre-sans-animaux/piece-enrichie/

choices about what merits archiving."

Apart from cost constraints, creating multi camera editing of theatre performances is also difficult due to several other reasons. First, the available camera locations are restricted. Second, the recordings should be made in a non-intrusive manner, not allowing any special equipment which will disturb the audience. Third, the production crew are merely present as guest members and have whatsoever no control over the start and stop of the scene. There will be no retakes and considerable amount of pre-planning among camera operators is required to shoot sequences which can be nicely edited together later. Shooting in such restricted scenarios is vulnerable to mistakes.

In this dissertation, we illustrate a novel approach, where we shoot the entire scene in high resolution from a single viewpoint and extract interesting parts in it by controlling a virtual pan-tilt-zoom camera in post-production. Showing only important parts of the image at a given time and switching between different parts can keep the viewer interested. But manually selecting a cropping window at each frame of lengthy recordings is an extremely tedious and difficult task. The key idea of the proposed thesis is a framework for automatically simulating a virtual pan-tilt-zoom cameras based on an intuitive user defined shot specification. Multiple virtual cameras can be simulated from a single video using different descriptions and the generated sub-clips can then be edited together in a standard video editing software to create the final edited video, just as in the case of multiple camera shoots.This work was done as part of the 'Scenoptique' project in partnership with theatre de Celestin in Lyon, France and proposes a cost effective semi-automatic system to efficiently archive their local theatre productions. The formal problem statement is described in proceeding section.

PROBLEM STATEMENT

In usual scenarios of a multi camera shooting in theatre, multiple cameramen sit at specific locations in the audience with their equipment. The filming is then done by these array of camera men with a preplanned strategy of each tracking a special case. With the proposed line of work, we intend to substitute the camera crew with just a set of static high resolution cameras, in the simplest case just one static camera covering the entire view point of the stage. Then simulate virtual pan tilt zoom cameras inside this high resolution video imitating what an actual camera crew would have done.

A virtual pan-tilt-zoom camera can be simulated by simply choosing a cropping window at each time frame of the original video. The challenge is to simulate the virtual camera crew: What are important elements of the scene to be focused on? How to computationally extract the important information from the scene? How many virtual cameras are to be simulated? What should be the composition of each virtual camera given the important elements to be included in it? When should the virtual camera be static? When should the virtual camera move? What should be the trajectory of the virtual camera movement? When to cut between the simulated virtual cameras?

All of the problems mentioned above are directly addressed in this dissertation. In summary we investigate the the problem of generating a fully edited movie from a single viewpoint. We break down this problem into four parts:

1. Content analysis: extracting important information from the scene which is important for editing. For example the location of the actors on stage; the events and actions happening on stage; identifying the speaker and the listener(s) etc.

2. Camera assignment: assigning virtual cameras based on the extracted information. For example a computer readable language which could be used by the user to define what he wants the virtual camera to look at.

3. Rush generation: generating multiple virtual cameras given the camera assignments and the extracted information

4. Rush selection: identifying which camera should be online (should be selected) in the final video at a given time from the generated rushes.

Our main contributions are in the first three parts and we offer perspectives on how our work can contribute to the fourth part.

CONTRIBUTIONS

The main contributions of this dissertation are following:

1. A database of theatre performances is presented. It consists of recordings of multiple rehearsals of three different plays produced at theatre de Celestin Lyon, including entire dress rehearsal of two of the plays. It can serve as a rich and fertile base for research in several application scenarios like actor detection and tracking, virtual camera simulation, script alignment, mixed illumination analysis etc.

2. We present "The prose storyboard language (PSL)", which is a formal language for describing movies shot by shot, where each shot is described with a unique sentence. The language uses a simple syntax and limited vocabulary borrowed from working practices in traditional movie-making, and is intended to be readable both by machines and humans. The language is designed to serve as a high-level user interface for intelligent cinematography and editing systems. We demonstrate how PSL can be used to describe challenging sequences from existing movies and show how a reduced version of PSL can be used for the application of automatic rush generation from theatre recordings.

3. We introduce view independent generative models for learning actor and costume specific detectors. The model consists of two parts, namely the head and the shoulder and each part is represented as a constellation of optional color regions. We present a novel clustering algorithm to learn these models from a small number of labelled keyframes (uniformly sampled across different viewpoints of the actor) using maximally stable color regions (MSCR) [For07]. We demonstrate the model on the task of localizing and naming actors in long video sequences. With results on an entire movie sequence we show that benefiting from view-independent models the proposed "color blob detector (CBD)" can perform accurate localization despite changes in viewpoint

and frequent actor to actor occlusions. We compare our approach with generic detectors [FGMR10, DT05] and show that using actor and costume specific information can significantly improve detection recall, specially in the scenarios like theatre or movies where the appearance of the actors are consistent for long periods. Also, using actor specific detectors give additional actor identity information which is useful in many application scenarios like ours.

4. We introduce a patch based rigid multi-part generative model and a complete framework for actor detection using the rigid model. The patch based model is also view-independent and is learnt from a small number of labeled keyframes. The rigid model is targeted to improve the performance in textured scenarios where MSCR features may not perform well. We compare the proposed patch based detector (PBD) with generic detectors [FGMR10, DT05] for the task of actor localization on sequences from live theatre and demonstrate improved recall in each case. We also compare the patch based detector (PBD) with the color blob detector (CBD). Although the rigid models are less flexible than MSCR based approach, we show that they improve recall over the CBD at lower precision rates.

5. We present a tracking algorithm using actor specific models. The algorithm works offline and assumes that a representative set of images for the target object are available prior to tracking. We learn the patch based appearance model (PBD) from the representative images and use it to find candidate detections in each frame of the video. We introduce an optimization process to combine observations from multiple frames into a coherent path, including modeling occlusions. The built appearance model in our approach is static and not updated after the learning step. We show that in many scenarios actor/object specific models can be built easily and can outperform tracking approaches trying to adapt the appearance model during the tracking process, which is a difficult task. We demonstrate that even using simple color features in the proposed multi-part object model provides significantly better results than the state of the art. We provide a thorough quantitative and qualitative analysis over long and challenging video sequences.

6. Using the precomputed actor tracks, we propose a framework for generating multiple synchronized sub-clips from a single viewpoint video, which can be directly edited together in a standard editing software. The subclips are generated by simulating the virtual pan-tilt-zoom cameras inside the original footage. Each subclip requires a simple user input defining the subject matter and size. We use a $L1$-norm optimization framework to automatically compute virtual camera trajectories such that the subject matter is nicely composed and the output subclip is nicely cuttable with other generated subclips. We show that various cinematographic principles can be modeled as penalties or constraints in a single convex minimization problem leading to efficient and scalable solutions. We show multi-clip generation results on a variety of sequences from live theatre performances.

OUTLINE OF THE THESIS

The outline of the remaining chapters is as follows. Chapter 2 describes the basic terminology, definitions and rules in video production to help the reader assimilate the proceeding chapter with ease. In chapter 3 we review the work related to the problem of automatic rush generation. Chapter 4 describes the theatre dataset. We discuss the challenges in recording live theatre and

detail the hardware which was used to meet these challenges. Chapter 4 also describes the prose story board language with its usage for describing existing movies and for vertical editing.

The generative models for actor specific detection are detailed in chapter 5. Both the blobs based approach and the patch based approach are described and compared with generic detectors. In Chapter 6 we describe the offline tracking algorithm using the actor specific detectors. Chapter 7 details our method for automatic rush generation. In chapter 8 we present some additional direct applications of our work and perspectives on future work. We conclude this dissertation document in Chapter 9.

CHAPTER

2

BACKGROUND IN VIDEO-PRODUCTION

VIDEO production or movie making is the art and service of creating content and delivering a finished video product. It involves a number of discrete stages including an initial story, script writing, casting, shooting, editing and screening the finished product before an audience. The scale of each production stage differs on the target audience, for example some of the Hollywood or television productions may involve large number of crew members and plenty of special equipment to conform the closely controlled technical standards. On the other hand, some other small scale video productions to be distributed online may only involve a single person using a hand held camera. With the high quality of today's consumer and prosumer equipment, the differences have become increasingly blurred. A lower budget production, does not mean that few people see them. A simple tour of YouTube will show that millions of people are watching variety of video productions every day.

Computational advances have been the building block of this success story, which has made it easier to both create and distribute the videos. For example, in early days editing was done manually and was considered similar to knitting, it was done by physically cutting and pasting film pieces using a razor blade, splicer and a threading machine. Today, most films are edited digitally (using softwares such as Avid Media Composer, Final Cut Pro, Adobe Premiere or Lightworks) and bypass the film positive workprint altogether. This is illustrated in Figure 2.1. Such an easy and non destructive workflow has made the way for much more free form experimentation and has provided a wider room for creativity.

Despite the computational advances, quality video production process still remains challenging in several scenarios. In this thesis we are proposing novel computational tools to further facilitate the process of video creation especially in the cases where a single camera can cover the entire field of action. The proposed work in this thesis is focused on one such scenario i.e. video production of staged performances (particularly theatre as it represents a generalized scenario) to be distributed either online or in the form of DVD's. We show that how novel videos can be generated from a single viewpoint video by simulating a virtual pan-tilt-zoom camera.

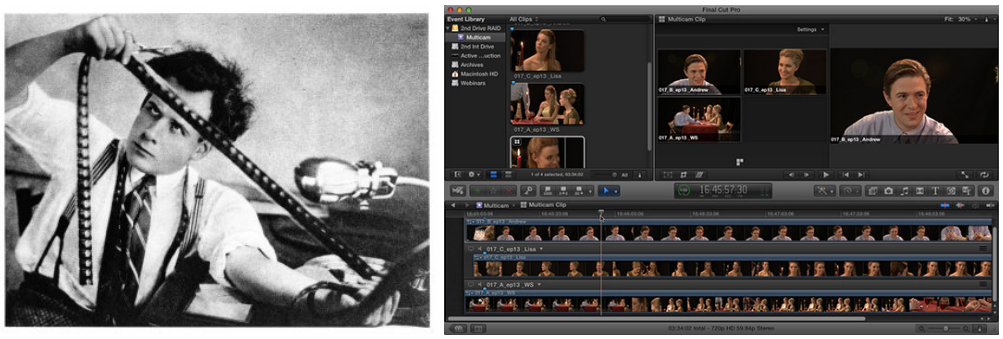


Figure 2.1: *Video editing in early days involved physical examination of negatives and physically cutting and pasting pieces of film together. The left image shows noted director and editor Einsenstein physically examining the films. Now, the recorded videos can be directly watched in front of a computer and can be combined together in standard softwares. A screenshot from the editing software Final Cut Pro is shown on the right.*

Before delving into the details of the proposed approach, we introduce the reader to the basic rules and principles involved in different stages of the video production process. Although it is a creative process and one may choose to perform them in the way they like, there are certain basic rules and guidelines that have gradually evolved and are commonly accepted in the entertainment and visual communication fields. In this chapter we introduce these basic rules, conventions, and practices of the global visual language of video/movie productions to help the reader better understand the proceeding chapters.

This thesis mainly focuses on video production of theatre plays and thus the two stages of 'shooting' and 'editing' are of particular interest to us (the other initial stages like script writing and casting are managed by the director of the theatre play). We first explain the cinematic principles/rules involved in the shooting process and then briefly review the editing process. The art of cinematography and video editing is extensive in itself, and thus we only cover the areas which are relevant in the context of the proposed work.

2.1 GRAMMAR OF CINEMATOGRAPHY

Cinematography is the art and science of capturing motion or live scenes on film/disk. The name comes from combining two Greek words: kinema, which means movement, and graphein, which means to record. Cinematography deals with all visual elements of a film like composition, lighting and camera motion. The person in charge of this is called the cinematographer. In the past, the cinematographer was also the camera operator but now a days he acts more as a supervisor to the actual camera operator. He is the one who decides every setting for the shoot, from color to depth-of-field – how much of the shot is in focus versus how much is blurry – from zoom to the positioning of people and objects within any given frame.

Similar to a language grammar which helps communicate verbally with others, over the years cinematography has developed its own set of conventions and rules that define how visual information should be displayed to a viewer. These set of rules, conventions and a unique vocabulary comprise the grammar of cinematography. In this section we discuss the commonly accepted vocabulary and the guidelines which govern the construction and presentation of visual elements. They are summarized from a set of standard cinematography and editing literature [TB09b, TB09a, Mas65, MJ08, Mer10, Mur01, Ari91].

2.1.1 Aspect ratio

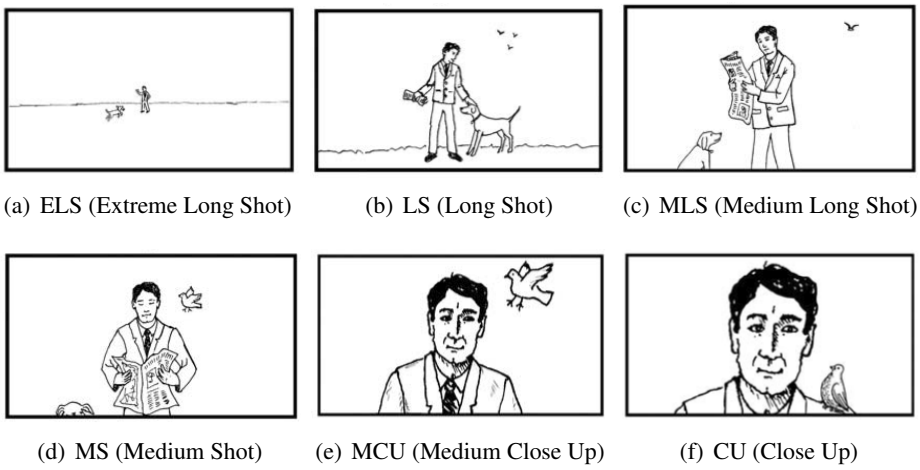
The aspect ratio of an image describes the proportional relationship between its width and its height (more specifically the ratio of width to the height). It is often expressed in the W:H format, where W is the width and H the height. For example, a 16:9 aspect ratio means that for a width of 16 units, the height must be 9 units. Most common aspect ratios used today in the presentation of films in cinema are 1.85:1 and 2.39:1. Two aspect ratios which are commonly used in television or online videos are 4:3 and 16:9. An example showing same scene recorded with different aspect ratios is illustrated in Figure 2.2

(a) 21:9 (b) 16:9 (c) 4:3

Figure 2.2: *Different aspect ratios (with same image height)*

2.1.2 Shot sizes

What really differentiates movies from watching actual plays is the way filmmakers manipulate the audience's field of view. While watching a play in theatre, the audience is in a wide shot (Extreme Long shot), always looking at entire stage and all the actors on it (although they may focus at different elements at different times). In movies, the filmmaker directs what the public sees and how. Different shot sizes allow the director to highlight different aspects of the scene as illustrated in Figure 2.3 and 2.4.



(a) ELS (Extreme Long Shot) (b) LS (Long Shot) (c) MLS (Medium Long Shot)

(d) MS (Medium Shot) (e) MCU (Medium Close Up) (f) CU (Close Up)

Figure 2.3: *Example of six different shot sizes. Figure reproduced from [TB09b]*

Besides the purpose of showing different elements of the mise-en-scène[1], shot sizes are also important for variety. If the audience were always looking at, say, a close-up shot, they could get bored of that unchanging frame. But because shot sizes are always different within a scene, spectators often have something new on the frame to study. Additionally different shot sizes also allow the director to highlight the important parts of the scene at different times.

Different shot sizes can be used for different purposes or to convey different emotions. Thomson in his book 'Grammar of the Shot' divides the shot sizes into three main categories:

[1]The arrangement of everything that appears in the framing – actors, lighting, décor, props, costume – is called mise-en-scène, a French term that means "placing on stage." The frame and camerawork also constitute the mise-en-scène of a movie.

(a) ELS (b) LS or FS (c) MS

Figure 2.4: *We can observe that for a filmmaker, several shot size choices are available at any given time in a scene. Example with three of different shot sizes from a theater scene are shown here. (a) Extreme Long Shot, covering the entire stage; (b) Full Shot (FS) or a Long Shot(LS), framing an actor entirely from head to toe; (c) Medium Shot (MS) framing an actor from head to around waist.*

the Long Shot (LS) or the Full Shot (FS), the Medium Shot (MS) and the Close Up (CU). He explains that these three basic categories can then be further extended to several other shot types. Figure 2.3 illustrates six different shot sizes with a single subject in a plain environment with the recording camera placed roughly at the same height as the subject's eyes. For simplicity the subject is centered and is looking straight in the lens.

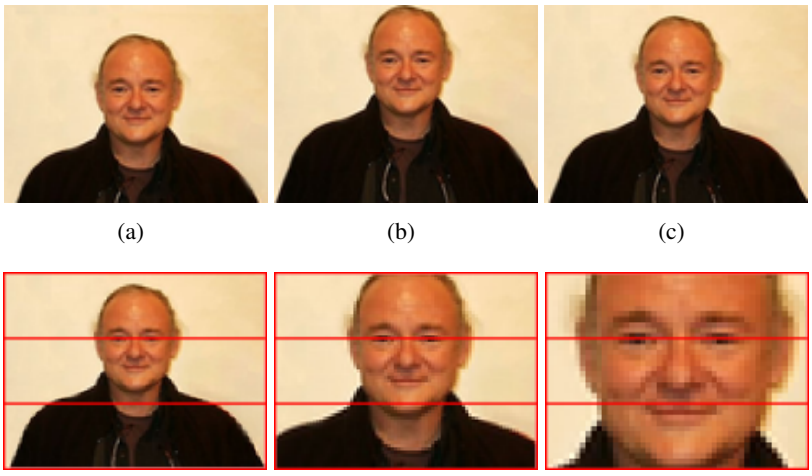
Extreme Long Shot (ELS) encompasses a large field of view and it is often used as an establishing shot in start of new sequence. Extreme long-shot range corresponds to approximately what would be the distance between the back row of the audience and the stage in live theatre. It is also common to refer ELS as "wide shot (WS)" because it often requires the use of a wide-angle lens. It typically shows all the details in the scene with all the actors and helps to build the relationship of the actors with the surroundings.

A long shot or full shot (FS) covers the full body with head and feet visible in the frame. A larger human figure takes attention away from the environment; however the immediate surroundings of the actor are still visible. A full shot makes it easier to observe details like gender, clothing and movement compared to ELS. The next commonly used shot is the medium shot (MS) which maintains a framing that cuts the human figure just around waist height. A medium shot nearly approximates how we, as humans, see the environment most immediately around us. Imagine that you are in a room with another person and the two of you are engaged in conversation. Typically there would be several feet of space between you (let us say five to ten feet) and, as a result, you would most likely be viewing each other in a medium shot. A viewer watching a medium shot should feel very comfortable with the image because it should feel like a normal observation.

In close-up (CU) shots, the subject occupies most of the frame, allowing very little observation on the environment. The close up shot usually covers the human subject from just above the hair to just above the shoulders. Close-ups are preferred when emphasizing someone's emotion. Two other common shots are Medium Long shot (MLS, frame cuts the human subject around knee height) and Medium Close Up (MCU, frame cuts the human subject around chest). In this dissertation we will be mostly using only the ELS, FS and MS.

(a) Too much headroom (b) Not enough headroom (c) Correct headroom

Figure 2.5: *Headroom with a medium shot.*



(a) (b) (c)

(d) (e) (f)

Figure 2.6: *Headroom with a medium close up (MCU). (a) Excessive amount of headroom, with the subject's nose centered in the frame. (b) A subtle lack of headroom with the subject's eyes located at about 28 percent of the way down from the top. (c) Appropriate head room, with the subject's eyes one-third of the distance down from the top of the frame. (d) The image vertically divided into three segments with eyes on the top one third line. (e) Close up shot with eyes at top one-third line. (f) Extreme close up with eyes at top one-third line. We can observe that while zooming in to tighter shots, the head is out of the frame, but the eyes are still kept at the one-third. Image reproduced from Wikipedia.*

2.1.3 Head room and Look room

The placement of the head within the frame is very important and is defined by the headroom. Headroom 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. The amount of headroom that is considered aesthetically pleasing is a dynamic quantity; it changes relative to the shot size (how much of the frame is filled by the subject).

Too much space between the subject's head and the top of the frame results in dead space. It leaves a strip at the top of the shot that has no significant elements in it. Aside from resulting in awkward framing, it also distract the audience from what is truly important in the shot. It is a common mistake and leads to wastage of premium screen space. On the other hand, too little head space may give a feeling that the subjects head is glued to the top of the frame or it may end up cropping the subject's head, which are both aesthetically unpleasant. An example

(a) (b) (c)

Figure 2.7: *Lookroom is the empty space in the direction the object is facing or moving. (a) Subject looking at left and positioned on the right side of the frame, leaving a good amount of look room. (b) Subject centered in the frame. (c) Object looking at left and also positioned at left side of the screen, leaving very little look room.*

Figure 2.8: *Rule of thirds*

of the right headroom for a medium shot is illustrated in Figure 2.5.

For shot closer shots like MCU and CU, a general rule which is often followed is to set the subject's eyes at nearly 2/3 up the framing, automatically giving the right head room. This is illustrated in Figure 2.6. This is based on the rule of thirds which states that if an image is divided into thirds using two equally spaced imaginary lines vertically and horizontally making nine equal parts, important compositional elements should be placed along these lines or their intersections. While framing human subjects eyes are the most important elements as human beings naturally tend to look each other in the eyes when communicating. The rule of third is obeyed by putting the eyes one-third of the way down from the top of the frame. The concept of headroom via the rule of thirds applies well for even tighter shots as illustrated in Figure 2.6.

Similarly, look room refers to the empty space in direction the subject is facing or moving. It is also referred as lead room or nose room. For human subjects it is the empty space provided within the frame, between the subject's eyes and the edge of the frame opposite the face. As illustrated in Figure 2.7(a), providing a good look-room creates a empty area that helps balance out the frame. For example in this case the weight of the empty space on the frame right balances out the weight of the subject on the frame left. Also a proper look-room provides space for the subject's gaze and his gestures so that the viewer can see them.

On the other hand with not enough look room the subject may appear congested or trapped as the subject's face appears looking at the "wall" of the frame (this is illustrated in Figure 2.7(c)). The weight of the empty space and the subject still exist but their placement

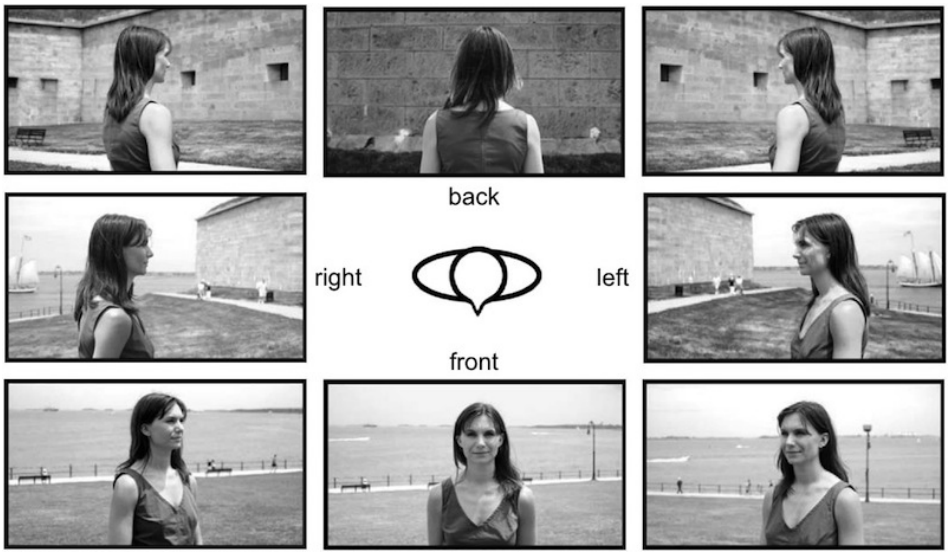


Figure 2.9: *Shots taken at different profile angles. Images reproduced from [TB09b]*

do not feel correct. There is nothing wrong with the subject always horizontally centered (as illustrated in Figure 2.7(b))in the frame but it is perhaps too uniform and boring.

Similar to headroom, the rule of third can also be applied horizontally as shown in Figure 2.8) by keeping the space in front of the subject around 2/3rd of the entire space and the remaining 1/3rd behind the subject. This general rule often applies to all types of shots, whether it is a medium shot, full shot or a close up.

2.1.4 Camera angle

Always shooting a subject with a camera located directly on the front of the face may yield a flat and uninteresting image. Such a flat frontal shot may be a good choice for shooting a news report but not for other creative purposes. Different angles are created either by asking the subject to angle their face/eyes away from the camera lens or by moving the camera around the subject. The relative horizontal orientation of the subject with respect to the camera lens is called the profile angle. The resulting images with different profile angles are illustrated in Figure 2.9.

(a) Neutral angle (b) High angle (c) Low angle

Figure 2.10: *Shots taken at different vertical angles. Images reproduced from [TB09b]*

Similarly different effect can be created by changing the vertical camera angles. A neutral angle is where the camera and its angle are looking at the subject from the same horizontal plane as the subjects eye. This angle is used as a general guideline since the audience can better relate the characters as equals. High and low camera angles can be used to create more dramatic effects. A high angle shot often creates an understanding that the subject on screen is smaller or weaker. A low angle shot generally used to make the character look significant and powerful. An example with neutral, high and low angle is illustrated in Figure 2.10.

2.1.5 Camera Movements

Until now we have discussed several important elements of visual grammar like shot sizes, headroom, lookroom and camera angles which when combined together form a composition. We have also discussed some guidelines for creating a nice composition. But we have only looked at the simple scenario of static compositions with a single subject, where both the subject and the camera are stationary. It is easy to understand that a composition can change either due to camera movement or subject movement or both. This movement is what differentiates motion pictures from still photographs. A still photograph may suggest motion, but it deals in space relationships only. It can , therefore , be well composed only within its singular frame of reference (frozen in time). A motion picture, on the other hand, is composed in both space and time.

The first form of dynamic composition is when the camera is stationary and the subjects are moving within the camera frame. It is also known as blocking talent [TB09b] (the name comes from a theater term blocking which refers to the precise movement and positioning of actors on a stage in order to facilitate the performance of a play, ballet, film or opera). Creating interesting blocking can engage the viewer's eye and keep them involved in the current image and therefore in the story. Lets now look at more interesting cases where the camera moves.

The moving camera can bring additional energy into the shots but setting the right kind and amount of movement is difficult - too little could be just boring and too much could be confusing. It is also difficult to find the right motivation for camera movement, as non motivated camera movements may look puzzling to the viewer. In order to figure this out, it would be helpful if we explored the many ways in which the camera can move.

(a) pan (b) tilt

Figure 2.11: *Pan and Tilt*

Pan and tilt are two of the most basic camera movements. In both cases the camera only makes pivotal movements keeping the actual mount stationary (illustrated in Figure 2.11). A pan (or panoramic shot) keeps the camera anchored to the center of an imaginary circle but rotates or swivels the camera lens horizontally. Panning a camera results in a motion similar to that of someone shaking his head from side to side. There are two common usages of the pan shots [Mer10]: "pan with" and "pan to". The "pan with" shots are often used to follow a subject as it moves across a location, and are said to be "motivated" camera moves because the movement of the subject motivates the movement of the camera. The "pan to" shots are used to shift the view from one subject to another, in this case the motivation is not the subject's movement but some aspect of the narrative; for instance, a character looking at something off-screen can motivate a pan that reveals what he is looking at. A tilt is moving the camera's lens up or down while keeping its horizontal axis (rotation in a vertical plane). Tilting a camera is similar to nodding the head up and down. Tilts are often employed to reveal vertical objects like a building or a person. There is also a combination shot that combines a pan with a tilt where the camera lens is simultaneously panned across the film space and tilted up or down while panning across. These movements are often achieved by using a pan and tilt tripod head.

A good pan/tilt movement should be smooth and steady,"leading" the movement of the subject [TB09b]. By this we mean that proper headroom, look room, and pictorial composition should be maintained throughout the life of the pan or tilt action. Thomson [TB09b], Marscelli [Mas65] and Arijon [Ari91] have all emphasized the point that a pan/tilt camera movement should be accompanied by well composed static camera shots at the start and the end. This is important for editing purposes later. We will discuss this with detail in the later sections but as a quick word: cutting on movement, either into a shot already in motion or out of a shot once in motion, is difficult and often looks bad. Several seconds of static camera at the start and the end of the shot will allow the editor to cut during the static segments.

Another common way to create dynamic shots is by zooming, which involves changing the focal length of the lens (hence changing the field of view). The technique allows to change from a large shot size to a small one without moving the camera (it only requires to move the lens), for example from a close-up to a wide shot (or vice versa). The zooming towards longer focal lengths (narrower field of view) in referred as 'zoom-in' and zooming towards shorter focal lengths (broader field of view) is known as 'zoom-out'. The zooming is also often used in combination with pan and tilt to create more interesting camera dynamics.

Other more complicated camera movements require moving the entire camera (moving the actual mount of the camera) using specialized equipment (like tripods attached with rails or tracks). The most commonly used movement is known as dolly, where the entire camera is moved forward or backward. The look of a shot that has the dolly pushing in and out may appear similar to a zoom, but it is much different. The actual movement of the camera creates a feeling that the viewing audience is physically traveling with the camera. Dolly can also be used sideways to follows a subject within the frame (it is often referred as 'dolly with' or a 'tracking shot'). Other specific camera movements are created using equipment's like cranes and trucks, for example moving camera from a higher viewpoint to a lower viewpoint. It is also common to use 'handheld' camera for shooting using special equipment called steadycam (which help to stabilize the camera movement). Handheld shooting allows operator to move freely around the set or location and is helpful if the action is moving too quickly or too unpre-

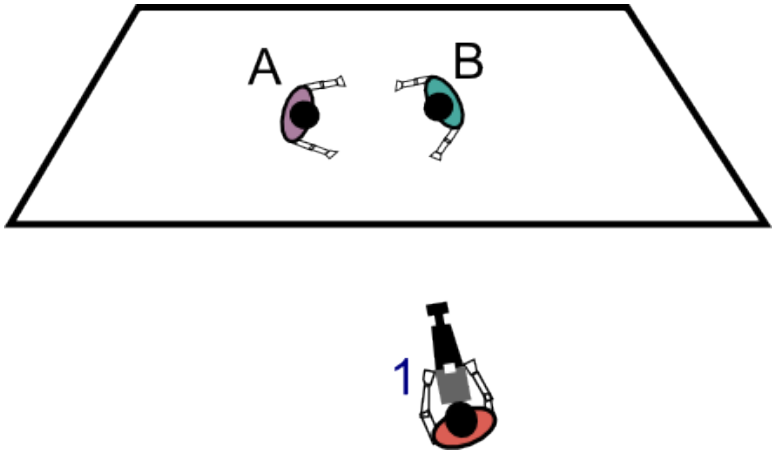


Figure 2.12: *Single camera setup*

dictably for the camera to be on a tripod.

Thomson [TB09a] classifies the shot types based on the camera movement into simple, complex and developing shots:

- Simple shots: created with a completely static camera (with no lens movement, no camera movement and no mount movement).
- Complex shots: involves lens movement and camera movement but no mount movement. Example includes, pan, tilt, zoom or any combination of the three.
- Developing shots: comprises of the actual mount movement combined with (optionally) the lens movement and camera movement. Examples includes the dolly shot or a crane shots.

In this thesis, we will be focusing on creating complex shots from simple shots.

2.2 CAMERA SETUP

2.2.1 Single camera setup

As the name suggests, a production using single-camera setup employs only one camera. The most simplest form of single camera setup is using a static camera which covers the entire action in the scene. This method is commonly used for archiving purposes for example in theatre recordings or other staged performances. The advantage of this method is that it is simple and economic, it does not require any camera operator and can be recorded automatically throughout the duration of the scene. Such a recording carries enough information to be used on its own but it is not the preferred options for professional video production (as watching the video from the same angle may not be engaging enough).

Varying viewpoint in single camera editing requires multiple takes i.e. actors play the scene multiple times with re-positioning of the camera. For example, in a scene with two actors A and B as illustrated in Figure 2.12, the director will first point the camera towards A and shoot

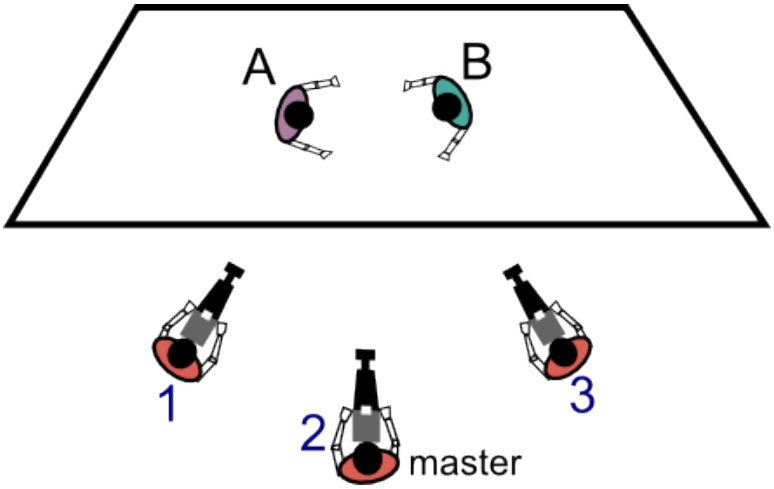


Figure 2.13: *Multi-camera setup*

shots number 1, 3, 5, 7, and so on. Then they will point the camera toward B and do shots number 2, 4, 6, 8, and so on. In the post-production editing process, the shots will be assembled sequentially to fit the script resulting in a video sequence which cuts back and forth between actor A and actor B. This is a commonly used approach in traditional film making.

The advantage of single camera setup is that it gives the director more control over each shot. It is easier for the director to only have to worry about the one camera. It is convenient to film specialized shots (with complicated camera movements, needing special equipment) with single camera techniques as the cameraman can move freely on stage and can get right in-between the actors without worrying about the field of view of other cameras. Also only a small crew is needed to operate a single camera.

A single camera theatrical filming with multiple takes, requires actors who are capable of repeating their performance a number of times. Not just the actors should exactly repeat their lines, they should also be able to replicate their movements for sequential matching of shots in the editing process. Shooting long or complex sequences often requires experienced actors, capable of sustained exact performances.

The disadvantage of single camera setup with multiple takes is that it is time consuming (as it requires change of settings and re-positioning of the camera with each shot). Continuity is also an issue, as the camera shots are not filmed in chronological order, it makes the editing process difficult and time consuming. Single camera setup is unemployable in non intrusive scenarios like live theatre or music concerts where multiple takes are not possible (unless using the single static camera approach).

2.2.2 Multiple camera setup

The multi camera setup is the use of more than one camera on one shoot arranged to show the same scene at different angles and sizes. Multiple shots are obtained in a single take without having to start and stop the action. For example, in a scene with two actors A and B as illustrated in Figure 2.13, camera-1 and camera-3 focus on the closer shots of actor A and B

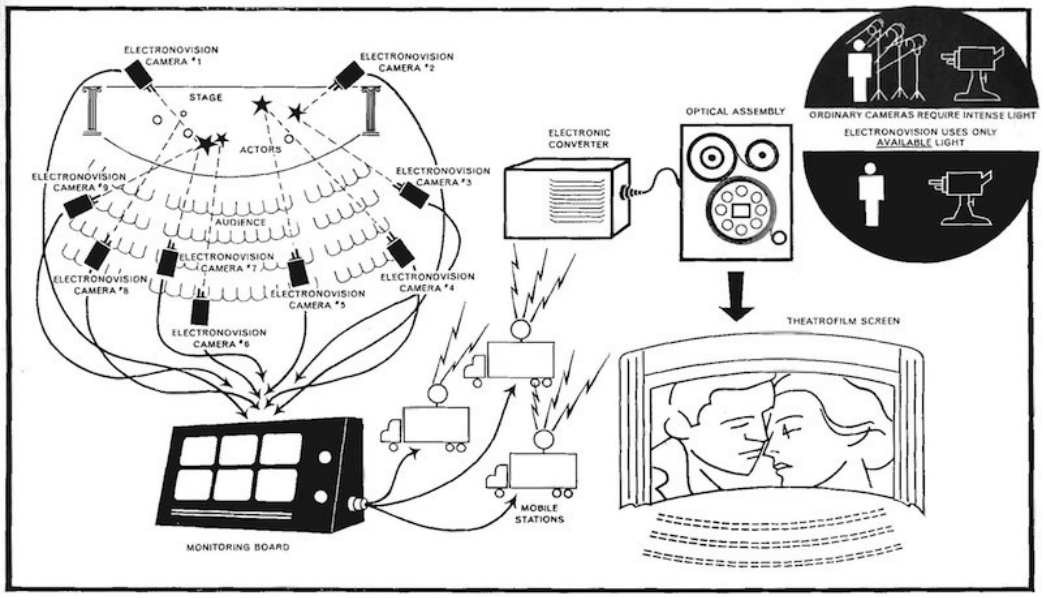


Figure 2.14: *Multi camera Electoronovision system employed for recording staged theatre. It was used for recording several famous theatrical plays including John Gielgud's Hamlet in the 1960s and early 1970s. Image reproduced from Wikipedia.*

respectively and central camera-2 shoots a wider shot to capture the overall action and establish the geography of the room.

It is a common practice to shoot the wider static shot in multi-camera setup covering the entire action in the scene. This is knows as a 'master shot'. It records the entire action, a complete run-through from that same camera position. Master shot plays an important role in multi-camera setup when it is not possible to repeat the scene. For example if a tighter shot is forgotten or mistaken during coverage, the director knows the editor can always cut back to the master shot.

The multiple camera setup gives the director less control over each shot. The restrictions in term of location (constrained by the field of view of other cameras) makes it harder to take shots with complex camera movements. It also requires a larger camera crew and more equipment which could add up the production costs. The direction is also of more technical nature and it is often a less personal process. But multi-camera setup saves a lot of recording time and the recorded sequences are much quicker and easier to edit as each of the shots are in order and the continuity is better.

Multi-camera setup is virtually the only possible choice while recording in certain scenarios for example in live sports broadcast. It would be really uninteresting to watch a football match from a single static viewpoint and it may be dangerous to broadcast it using a single moving camera (it would be very easy to make mistakes and quick camera moves might be very distracting to the viewer). The only choice available is to record the action from multiple cameras and to switch within them to highlight details and provide variety to keep the viewer interested.

Multi-camera setup is also a preferred choice in the video production of staged performances like theaters (which is the main focus of this dissertation). A coverage from multiple cameras with varying viewpoints, sometimes showing the blocking and sets, sometimes focusing on exciting details make up for the thrill of watching the live event. Benefiting from multi-camera shoots, the film audience in fact "sees" more views and details than the theatre audience at the actual event. One well known example is John Gielgud's *Hamlet* starring Richard Burton in 1964, which used 19 cameras (the Electronovision system which was used for the recording is illustrated in Figure 2.14).

It is also common to use multiple camera setup with multiple takes and is often employed in large budget productions.

2.3 VIDEO EDITING (MONTAGE)

The raw footage or the collection of recorded sequences from the shooting step are known as 'Rushes'. The process of rearrangement, selection, cutting shots from the rushes and combining them into sequences to create a finished work is known as 'video editing' or 'montage'. It is a crucial step in the post production stage which makes sure that the video flows in a way which achieves its goal of providing information or telling a story. Often editing is also used in context of applying color correction, filters and other enhancements but in this dissertation it is only referred in the sense of manipulation (selecting parts and cutting) and rearrangement on the timeline.

Even though the editor had no control over which shots were recorded on the film set or how they were composed, it is their job to review all of the material and choose the best viewpoints at each time and combine these various shots to show the audience the best visual presentation of the action in the story. Similar to the shooting step, there are several commonly accepted rules which guide the process of video editing. In this section, we will briefly discuss these guidelines of visual construction used which are to assemble the shots into a meaningful story.

2.3.1 The basic edit transitions

There are four basic ways one can transition from one shot to another:

- Cut – An abrupt change from one shot to another. The term stems from the days when motion pictures were shot and edited on very long strips of emulsion film. The editor used to physically cut the pieces of useful parts of the film and glue or tape them together. The cut and join represented the point of transition from one shot to another. Nowadays this process is done using special editing softwares but the term is retained. A cut represents a continuous transition in setting and time. Cuts are often used when there needs to be a change of impact.

- Dissolve – a transitional editing technique between two sequences, shots or scenes, in which the visible image of one shot or scene is gradually replaced, superimposed or blended with the image from another shot or scene. This is traditionally achieved via a superimposition of both shots with a simultaneous downward and upward ramping of opacity over a particular period of time. As the end of the first shot 'dissolves' away,

the beginning of the next shot 'resolves' onto the screen at the same time. Generally the use of a dissolve is held to suggest the passage of time (that a period of time has passed between the two scenes) or locale (transitioning to scene at different location).

- Wipe – It is the technique where one shot is replaced by another by the movement of an edge, or line, which replaces the previous shot by 'wiping' it. By revealing a new scene, environment or space the wipe offers a spatial or temporal transition to the director.
- Fade – A gradual change from a solid black screen into a fully visible image (fade from black or fade-in). A gradual change from a fully visible image into a solid black screen (fade to black or fade-out). Fade-in and fade-out are often used at the beginning and end of a motion picture or any video program. They are also used to represent at start and end of a chapter, scene, sequence or an act.

Out of the four transitions, cut is the most practiced form of transition in films and will be the focus of most of the discussion in the proceeding sections.

2.3.2 Guidelines for a good edit

Motivation

It is necessary to motivate a transition from a shot to another or it may confuse the viewer. A new shot should present new information to the user and there should be a reason to leave the previous shot [TB09b]. The motivation can be either visual or aural. It could be as small as a small movement of a face (perhaps a character in close-up only moves his eyes slightly to the left as if looking at something off-screen and would allow the editor to cut to his object of interest). Other example motivations could be to show someone entering the scene; to show reaction of an actor in a dialogue or to show some kind of movement of the subject in the scene. Remember, one of the many tasks set up for the editor is to engage the audience both emotionally (to make them laugh, cry, scream in fright, etc.) and mentally (to make them think, guess, anticipate, etc.). Motivated transitions at right timings play important role in achieving this goal.

Movement

Although camera movements are often implemented to add dynamism to shots, they can be distracting and even annoying when overused or used without a reason [Mas65, Ari91]. Small camera movements should be avoided, a static camera is preferred option to shoot simple actions and make up the bulk of dialogue driven motion picture content. The static shots are sharper, simpler, and clearer to the viewer. It is also important that the camera should be fully locked off in static shots (no lens movement, no camera movement). It is much better to be slightly off center in framing than to make small camera movements. For example, a static shot of a player moving into the frame, such as sitting into a close-up, should be filmed with a locked camera, not one that moves about nervously at the beginning of the shot until the cameraman is satisfied with the framing. The editor may experience difficulty in match-cutting such shots to a preceding scene filmed with a locked camera.

Marscelli [Mas65] asserts that a moving shot must be used in its entirety (or any continuous portion) because it is difficult to cut during camera movement. Cutting across two moving shots is difficult as it requires the editor to carefully match the movement. Also, it is very

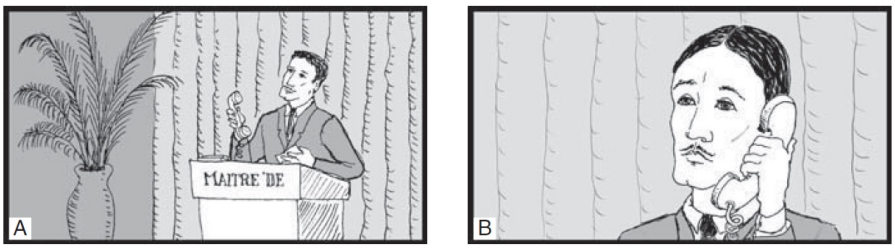


Figure 2.15: *Cutting from a medium shot of actor with phone in the right hand to to the medium close up of the same actor with the phone in the left hand will not maintain the action continuity. Image reproduced from [TB09a]*

jarring to cut from a static shot of a static subject to a moving shot that begins moving immediately; or to go from a continuously moving shot of a static subject to a static shot. Both Marscelli [Mas65] and Thomson [TB09b] emphasize that camera movements should always be filmed with a static camera at the beginning and end of the shot. This allows to place the cut across static frames, with the movement sandwiched in between.

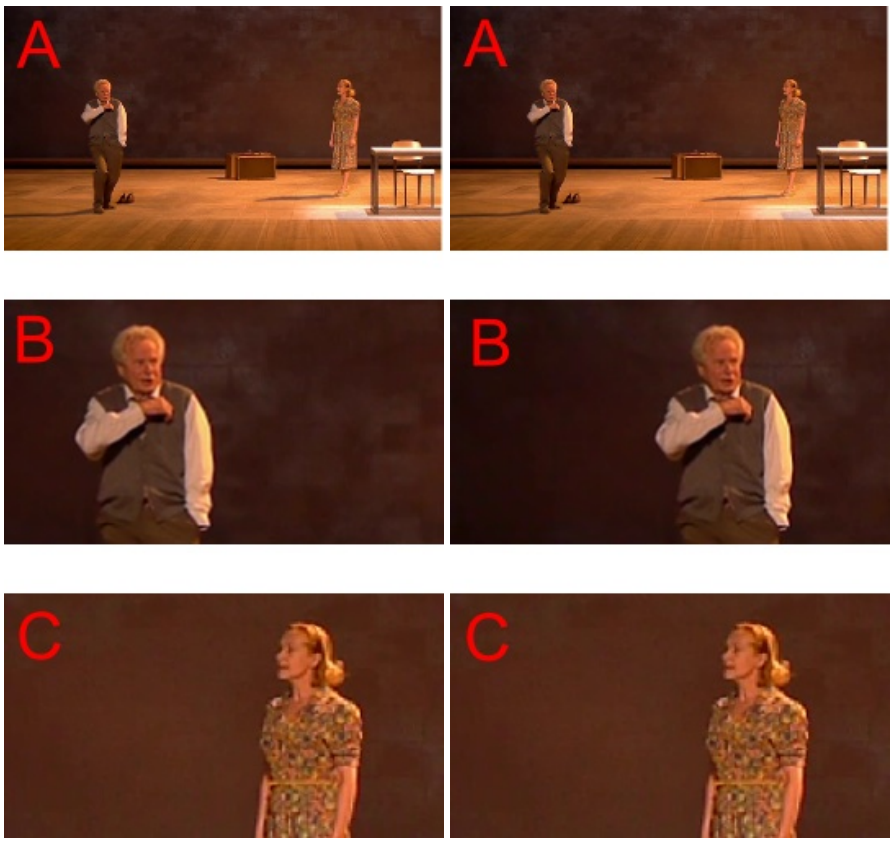
Cutting is a powerful tool of editing and cutting between static shots can be used to represent a fast flow of events. Marscelli [Mas65] in his book describes: "*Many cameramen and directors mistakenly believe that a moving shot contributes flow to the story-telling and speeds the screen action . In many instances, movement slows the screen story because it takes longer to come to the point! Unless the camera move is dramatically motivated , it is much better to shoot several static shots that may be straight cut, rather than a long moving shot which drags from one significant bit of action to another. Even when a moving shot is satisfactory from an editorial standpoint, it may be difficult to insert between static shots*".

Similarly Arijon [Ari91] says: "*A straight cut is faster than a moving shot because it establishes the new point immediately. If we pan or track to a new point of interest instead of cutting, we expand a lot of useless footage photographing irrelevant things, simply to travel there. Significant actions of objects must be photographed during the pan or track to justify its use or the sequence may drag*".

In summary, although camera movements are important part of cinematography, they should be carefully used. And importance of static camera shots should be kept in mind. If used correctly (with rightly motivated cuts), the static shots can be very powerful. Thomson [TB09a] states that in motion pictures, 'simple shots' often make up the bulk of the rushes.

Action continuity

Providing smooth, seamless continuity across transitions is a very important element to keeping the edits unnoticed by the viewer [TB09a, Ari91]. For instance, if in one shot a milk glass is empty, it should not be full in the next shot. Not just objects, the actions performed by the on-camera talent must match from one shot to the next. In single camera setup, where actors repeat the same set of actions with multiple takes it is not necessary that the actions match accurately. An example is illustrated in Figure 2.16, where cutting from frame-A to frame-B causes discontinuity in action because the actor on stage switches the hand holding the telephone. One solution which is commonly used to fix errors in recording is a 'cut away shot',



(a) (b)

Figure 2.16: *Spatial continuity (a) Cutting from frame-A to frame-B will maintain the spatial continuity, as the actor on the left is kept on the left side in tighter shot. (b) Cutting from frame-A to frame-B may confuse the viewer, as the actor is kept on the right side of the frame in the tighter shot.*

which may provide requisite break from the action (show something else for a while, for example some other actor or object and then cut back to frame B).

Cutting among a synchronized set of multiple camera shoots of the same scene always maintains the continuity of action. Live coverage of a sporting event would be an example of footage that is very continuous. Since the live operators are cutting from one live feed to another, the physical action of the shots matches very closely. Similarly, in our case we maintain action continuity simply by keeping entire duration of the performance.

Spatial continuity

It is important to maintain a sense of spatial positions in the film space based on the actual position of the subjects. As we have discussed earlier it is a common practice to use establishing shots to initially build the mise-en-scène (the entire arrangement of the scene). While cutting to tighter shots an editor should be careful that the sense of space is maintained at the point of transition (it is also the job of the cameraman to keep the editing in mind while shooting).

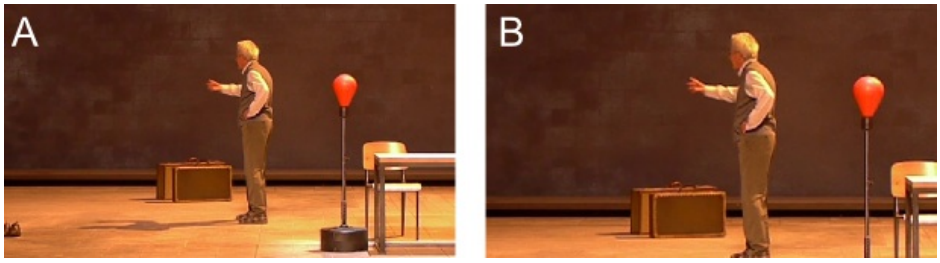


Figure 2.17: *Cutting from frame A to frame B will lead to a jump cut. As frame B is not different enough to be a new shot, the transition from frame A appears to be a mistake in editing and gives the effect of jumping forward in time.*

An example is shown in Figure 2.16, where keeping the actor of the left on the left side of the frame in the tighter shots maintains the spatial sense on the screen space. The other case might confuse the viewer.

It is also important for the editor to string together shots where that subject or object placement is maintained continuously. If an actor is shown on frame right in shot one, then he must be somewhere on frame right in any subsequent shots during that scene. Of course, if the actor physically moves, during the shot, to a different location within the film space then it is logical to show him on a new side of the frame. Cutting together two shots that cause the subject or object to jump from one side of the screen to the other will distract the viewer and the illusion of smooth editing will be broken [TB09a].

Change in composition

An editor should be careful not to cut between two shots of the same subject which are too similar in angle or size. This will result into a jump cut (this type of edit gives the effect of jumping forwards in time). Such cuts should be avoided unless used for special creative purposes. An example of jump cut is illustrated in Figure 2.17.

Colloquial use of the term jump cut can be used to describe any abrupt or noticeable edit in a film. For example, a jump in framing may occur due to discontinuity of action but such usages are not technically correct. In this thesis, a jump cut will only refer to the case where the jump occurs while cutting between shots taken from the camera positions that vary only slightly. There are several ways to avoid a jump cut like making sure that the change in framing is large enough (i.e., full shot to medium shot); inserting a cut away shot; zooming or moving the camera instead of cutting; or changing the angle of the new shot more than 30 degrees.

2.3.3 Multi camera editing vs single camera editing

Rushes from the multi-camera shoots are synchronized prior to editing. Recent video editing software provide tools to do this automatically based on audio or time codes. The editing then involves the choice of camera to show at a given moment and the points to make the transition from one camera to another. Since the recorded sequences are already synchronized (aligned in time-line) the editor need not worry about maintaining action continuity. But he still needs

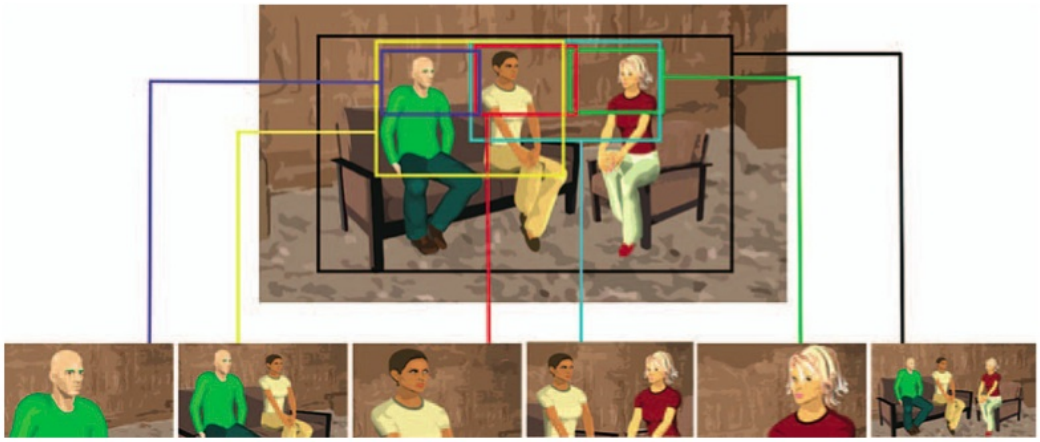


Figure 2.18: *Groups can be broken into series of smaller shots. Image reproduced from [MJ08].*

to take decision to maintain spatial continuity and to avoid jump cuts. In live broadcasts editing decisions need to be taken online and there is no place for making a mistake. In offline cases, the editor can take more time to experiment and adjust the transitions. Although, it should be understood, that an editor can only try to make the best out the available recordings and cannot always cover the mistakes of the cameraman. Hence, it is important that the camera setup and recording should be done keeping the editing in mind.

Editing in single camera setup is much more difficult and time consuming than the case of editing multiple synchronized sequences. It also involves the selection of shot to be shown at a given moment but maintaining action and spatial continuity is much more challenging. Hence, single camera setups often requires considerable amount of planning prior to the shooting. It is a common practice to pre plan all the shots based on storyboards (graphic organizers in the form of illustrations or images displayed in sequence for the purpose of pre-visualizing).

2.4 EDITING WITHIN THE FRAME

Consider the scene in Figure 2.18 with three actors and some of the possible shot choices available to frame the given set of actors. Shooting this scene with multiple cameras will require a different camera focusing on different shot (limiting the number of shots to the number of cameras). On the other hand, shooting the same scene with single camera would require the entire scene to be repeated as the camera is re-positioned (limiting the number of shots to the number of repetitions).

A different approach is to edit within the frame i.e. crop windows from the original recording (a master shot covering the entire scene) to generate different shots (in sub resolution). This approach is also known as '*vertical editing*', a term which was coined by the famous film editor Walter Murch.

> *Up until now [motion] picture editors have thought almost exclusively in the horizontal direction. The question to be answered was simply, What's next? that's*

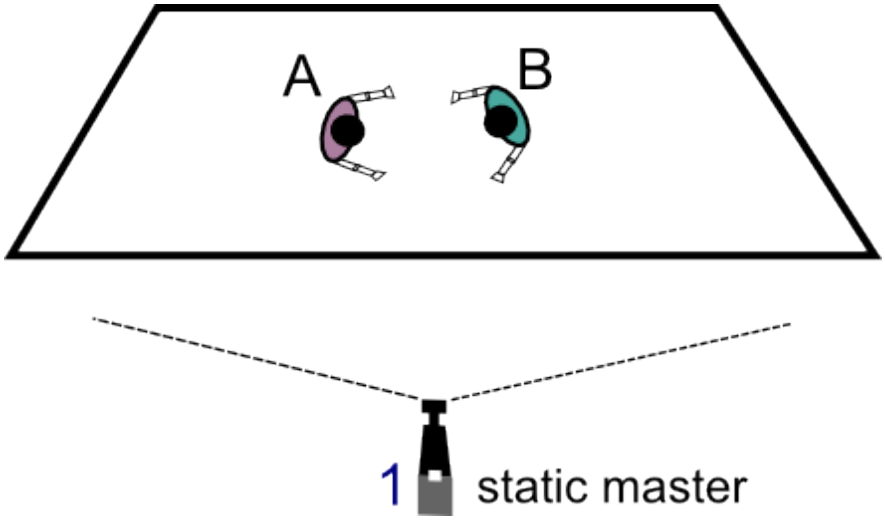


Figure 2.19: *A simple setup for automatic rush generation includes just a single static camera covering the entire action in the scene.*

> *complicated enough - there are a tremendous number of options in the construction of a film. In the future, that number is going to become even more cosmic because film editors will have to start thinking vertically as well, which is to say: What can I edit within the frame? [Mur01]*

In this dissertation, we directly address some of the challenges of vertical editing or "in the frame" editing. A virtual pan/tilt/zoom (PTZ) camera can be simulated using vertical editing by choosing a cropping window at each time inside the original recording. The pan/tilt movements are simulated by moving the cropping window vertically and horizontally inside the image boundaries of the original recording and the zoom is simulated by changing the size of the cropping window keeping a fixed output resolution. Please note that an environment may be real or virtual. Real camera (which captures light using an image sensor) can be used in real environments only, but virtual cameras (which renders image using geometric and photometric information) may be used in both real and virtual environments. For the sake of clarity we would like to mention that in this thesis the term 'virtual PTZ camera' will always refer to simulating virtual cameras in real scenes (more specifically simulating a cropping window inside an already recorded video) unless specified otherwise.

Multiple virtual cameras can be simulated from a single master shot by focusing on different actors/objects in the scene, we call it as the problem of '*rush generation*'. The computational model for "*automatic rush generation*" is the main focus of this dissertation. The computational model is analogous to a virtual camera crew which shoots nice compositions but also keeping the editing in mind. The generated rushes can then be edited together in standard softwares like usual multi camera scenario. We will not dive into the details at this moment but let us look at some of the advantages and disadvantages of virtual camera simulation within a master shot.

One major advantage is that the setup requires only a single non operated camera covering the entire action in the scene (as illustrated in Figure 2.19) and allows the editor to create a variety of smaller shots. Lets go back to the example in Figure 2.18, with multiple camera

setup generating 6 different shots will require 6 different cameras. With single camera setup, it will require 6 repetitions of the same scene. Using virtual camera setup, all 6 shots can be generated using a single master shot, saving both cost and time. With more actors present on stage, the number of possible shots may be even higher, and virtual camera generation allows the editor to experiment and generate a cosmic range of options.

Not just independently, virtual camera simulation is useful even in usual multi-camera shooting, where it is a common practice to use master shots. Using the same master shot the editor can create more choices using virtual camera simulation, without any additional cost. In many situations like recording a live theater (where it is not possible to shoot multiple retakes), virtual camera simulation allows to make up for the mistakes done by the cameramen. Apart from the usual solution of cutting to the master shot in case of mistakes, virtual camera simulation allows to create and use novel tighter shots during the editing process.

One of the obvious disadvantages of simulating virtual cameras within a master shot is that it does not allow to use the full resolution of the camera. Simulating tighter shots may lose a large part of the camera resolution. Hence, it is only applicable to target audience where loss in resolution is acceptable, for example in web streaming. But with $4K$ consumer cameras already in market the application scenario may broaden in the future. One other limitation of simulating shots within a single master shot is that it is not possible to change the camera angle (all the generated virtual camera shots will retain the angle of the original recording).

2.5 SUMMARY

This chapter introduces the reader to the basic rules and guidelines, also practices of the global visual language covering only the areas which are relevant to the current work. The chapter illustrates the rules, conventions and a unique vocabulary that comprises the grammar of cinematography viz. aspect ratio, shot sizes, head and look room, camera angles and various camera movements. The chapter then proceeds to describe in detail the camera set-ups. The cases being the usage of single or multiple cameras, their respective benefits and limitations are then highlighted.

Montage - editing, the most notable and crucial step in the post production process is comprehensively described. The assembly of various shots and the basic methods employed for transiting between them are stated, which maintain the video-story in a continuous flow. Drawing from respected technical textbooks, we have cited various guidelines used by professional cinematographers and editors for generating and editing high quality rushes. Motivation, movement, action continuity, spatial continuity and change in composition comprise the description of a good edit. The ease of editing of rushes from a multi-camera setup are compared with single camera setups. The chapter then defines ‘vertical editing’, an approach where editing is performed within the frame by cropping windows from the original recording.

CHAPTER

3

RELATED WORK

THIS dissertation is focused on the problem of automatic rush generation from a single viewpoint. The exact problem has not been addressed in previous work, but we review some related approaches in this chapter. We will discuss the media re-targeting methods which adjusts a given video for enhanced viewing in variable aspect ratios. The automated robotic camera systems and virtual camera systems proposed in the previous literature will also be discussed.

The early usage of vertical editing (editing within the frame) can be drawn from still imagery where zooming and panning across photographs were used to generate the feeling of motion to retain the interest of the viewer. This technique is still commonly used and is known as Ken Burns effect. Such an effect is mostly used in historical documentaries where film or video material is not available. Ken Burns effect gives action to still photographs by slowly zooming in on subjects of interest and panning from one subject to another. For example, in a photograph of a group of people, one might slowly pan across the faces of everyone and comes to a rest on the main character the whom the narrator is discussing.

Another common example of vertical editing is 'Pan and Scan', which is a technique used to adapt widescreen film or video footage into a conventional $4:3$ aspect ratio (used by old-style television sets). Pan and scan involves a process of selecting the most important part of the frame and discarding the other parts. In this way the image width is reduced to fit the $4:3$ screen dimensions (an example is illustrated in Figure 3.1). Although, recent television sets have a much wider aspect ratio the technique of pan and scan is still used for other screens (for example mobile screens or screens in an aeroplane).

Usually the "pan and scan" process is done manually, where an editor selects the parts of the original filmed composition that seem to be the focus of the shot and makes sure that these are copied (i.e. "scanned"). When the important action shifts to a new position in the frame, the operator moves the scanner to follow it, creating the effect of a "pan" shot. In a scene in which the focus does not gradually shift from one horizontal position to another—such as actors at each extreme engaging in rapid conversation with each other—the editor may choose to "cut" from one to the other rather than rapidly panning back and forth.

Figure 3.1: *Illustration of pan and scan. The red rectangle shows the pan and scan version of the original frame in wide-screen format. Images reproduced from* http://www.videohelp.com.

With a wide range of recording devices and displays available these days, the problem of resolution mismatch has become much more common and can occur for any form of media (photographs or personal videos or web streams). The problem of adapating one format of the media to another (with different resolutions or/and aspect ratios) is now commonly known as media re-targeting.

3.1 MEDIA RETARGETING

Computational methods for automatic or semi-automatic media re-targeting have been proposed over the past few years. These methods can be classified into three basic categories:

- Cropping based approaches [LG06, SAD*06, DDN08] find a sub-resolution cropping window inside the original media content minimizing the information loss (based on pre-calculated saliency or importance maps).
- Discrete approaches [AS07, RSA08, RSA09, PKVP09] remove and/or shift pixels judiciously to preserve the salient media content.
- Continuous approaches [GSCO06, WGCO07, ZHM08, KLHG09] optimize a mapping (warping) from the source media size to some target size using several types of constraints to protect media content. (a non homogeneous transformation, where less important regions are scaled more than important ones).

The first step in these algorithms mostly involves computing an importance map from the media. Saliency maps and gradient maps are used as a part of important information in most of the methods. Some of these algorithms also utilize face detections and manual annotations from the user, to apply various constraints in the optimization framework. For example, re-targeting method by Gleicher et al. [LG06] penalizes cropping the detected faces in the energy minimization framework. Another example is the method proposed by Krähenbühl et al. [KLHG09] which adds a constraint to preserve the shape and position of important user annotated scene objects. In videos, additional information like motion saliency or optical flow is also used.

The second step involves applying a resizing/cropping operator preserving the important media content based on energy minimization or a dynamic programming framework. Some of these methods [SAD*06, AS07, PKVP09, GSCO06] are limited to re-targeting still images while others extend to video re-targeting. Although re-targeting images is challenging in itself, an additional dimension of difficulty is added with video re-targeting. One important point which is missed in most of these methods is that the problem is not just limited to encompassing important composition at each frame and salient motion between frames, but the preservation of cinematic intent. Any unmotivated movements of the cropped window in the re-targeted video may appear strange to the viewer. To our knowledge, the earlier work by Gleicher et al. [LG06] is the only re-targeting paper which discusses this aspect.

A recent study [RGSS10] compares various image re-targeting approaches on a variety of images (focusing on different attributes namely lines/edges; faces; texture; foreground objects; geometric structures; and symmetry). Interestingly a manual crop was often favored by the viewers over other re-targeting methods. This suggests that loss of content is generally

preferred over deformation artifacts. Also it will be valid to say that such artifacts will only increase in videos with added difficulty of maintaining temporal coherence. Hence, the cropping based approaches may be a good direction to work in, for more practical and direct problems like automatic pan and scan. Although, other re-targeting methods still hold their own importance because they can achieve more advanced image/video manipulations like removing objects or impainting, which holds a great artistic value.

The problem of automatic rush generation relates to video re-targeting in the sense that they both extract parts from a higher resolution video to create a novel sub-resolution video. The major difference is that video re-targeting is applied on already edited sequences and it is difficult to maintain the editors original intentions. Not just that, the current automatic re-targeting systems are good enough for practical use in long sequences, even with manual pan and scan it is not always possible to retain the editors original vision and intentions (it can remove up to 45 percent of the original image while adapting to $4:3$ from an original aspect ratio of $2.39:1$). Due to this reason some film directors and film enthusiasts disapprove of pan and scan cropping.

On the other hand, automatic rush generation is the problem of simulating multiple cameras inside a master shot (higher resolution shot covering the entire action of the scene). It allows the editor to easily create shots with different aspect ratios for display at different devices. It is aimed to generate novel interesting shots from a static recording which is not interesting enough to be watched by itself.

3.2 AUTONOMOUS CAMERA SYSTEMS

The problem of automatic rush generation can be related to the work in the field of automated camera systems [CC14], which also focuses on automatic recording of the scene based on content analysis. Autonomous camera systems have been proposed for both real and virtual environments. Both involve automatic camera selection and movement based on information extracted from the scene. The high level semantic information of the scene is easily available for camera systems in virtual environments and systems in real environments face an additional problem of actually extracting such information from the scene. Although, advancements in computer vision have made it possible to extract large variety of information from real camera data (like actor positions, their actions) it still remains far from matching the amount of accurate information available in virtual scenes. Camera movement in virtual environments can be highly mobile and can support free form movement. On the other hand, autonomous camera systems in real environment are much more restricted by design.

Many similarities can also be drawn from both directions of work. By watching cinema and TV broadcasts for several years, viewers have developed an invisible sense of apprehension of the camera work and not following these lines may leave them confused. Hence, models simulating cameras in virtual environments are often inspired by the movement of real cameras in real environments. Our discussion will be mainly focused on autonomous camera systems in real environments and readers can refer to a survey by Christie et al. [CON08] for a thorough summary of camera systems in virtual environments.

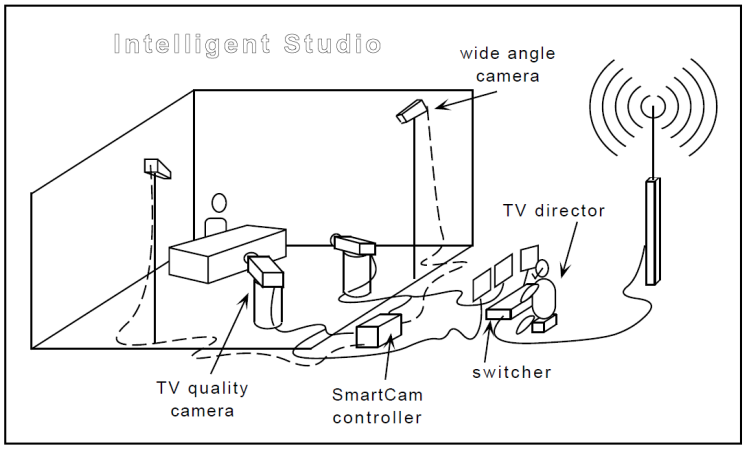


Figure 3.2: *The intelligent studio proposed by Pinhanez et al. [PB95]. The system includes wide-angle imagery to understand the events happening in the studio and robotic TV quality cameras to generate the image to be aired.*

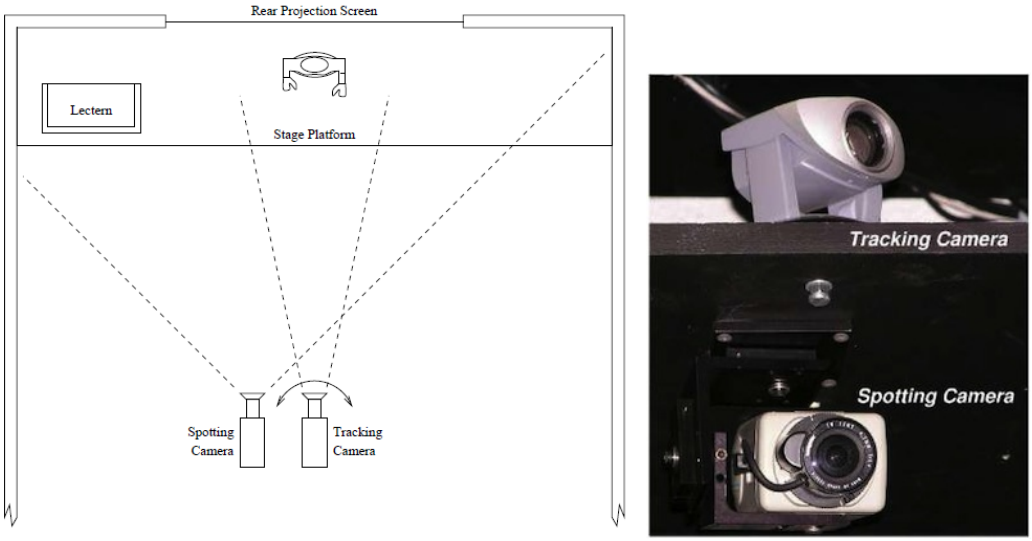
We further divide the autonomous camera systems in real environment into two different categories, first that controls a robotic camera based on information gathered from other sources (including extra cameras) and second which simulates a virtual pan/tilt/zoom (PTZ) cropping window inside the data captured from real cameras. We first discuss the robotic approaches before moving to a more detailed discussion on virtual cameras in real environments which are directly related to our work.

3.2.1 Autonomous robotic cameras

Robotic camera control requires feedback loops (examining where camera should ideally be and where it actually is). The task of keeping an object of interest within the field of view is referred as visual servoing (as the feedback in this case is extracted from a visual sensors). Discussion on visual servoing is out of the scope of this thesis but interested readers can refer to an excellent tutorial by Chaumette et al. [CH06, CH07].

One of the earliest models of autonomous robotic camera systems was proposed by Pinhanez et al.[PB95] and is illustrated in Figure 3.2. Their system was aimed to build intelligent robotic cameras, able to frame subjects and objects in TV studio upon verbal request from a TV director. To guide the robotic cameras they proposed to use information extracted using computer vision approaches on complementary wide angle cameras overlooking the scene and the script for other contextual information. Their system was limited to scripted cooking shows and used frame differencing to approximately locate the subject (the 'chef' in this case) given the initial position. They represented framing requirements by rules. But their system did not actually present results using robotic cameras and used an offline approximation instead.

The autoauditarium system by Bianchi et al. [Bia98] was one of the first to demonstrate how robotic cameras with additional static wide angle cameras can be used for fully automatic production of lecture videos (it is illustrated in Figure 3.3). Their system uses a stationary camera looking over the entire region of interest (they call it spotting camera), a robotic PTZ camera and other specialized cameras (for example one specifically looking at the lecture slides



(a) (b)

Figure 3.3: *(a) Overhead view of the AutoAuditorium [Bia04] system. (b) The actual tracking camera and spotting camera used in their system.*

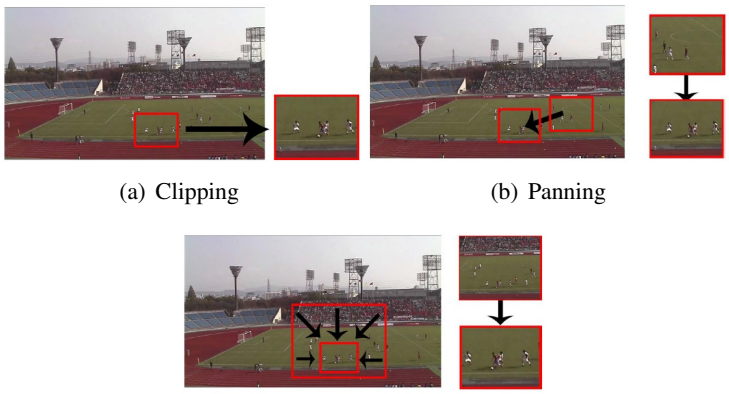
or a chalk-board). The images from the spotting camera were analyzed for the motion of the presenter on stage and the analysis was then used to pan, tilt and zoom the robotic tracking camera. Their later work [Bia04] also presents the interesting history behind development of autoauditarium system.

Other related robotic camera systems focusing on the specific case of recording lecture videos include Microsoft iCam system [RGGH04], Microsoft Research LecCasting System (MSRLCS), [ZRCH08], CARMEL system [KNM03], FlySpec system [LKF*02]. Some of these systems are quite extensive, for examples the CARAMEL systems uses more than 8 cameras and requires the presenter to wear specialized ultrasonic sensors for accurate localization. The flypack system also uses specialized omni directional cameras with the robotic PTZ cameras. But most of these systems are designed for constrained environments and the camera tracking is restricted to simple movements of a presenter (often limited to a single presenter).

These systems are difficult to use in more dynamic scenes because of the requirement of smooth and purposeful camera movement (otherwise it will disorient the viewer). To achieve such smooth motion it is necessary to use low control gains in the feedback loop and that limits the changes in acceleration of the robotic cameras. Moreover, these systems are designed to operate online (based on near real time feedback) and that restricts more planned and professional looking camera motions.

3.2.2 Autonomous virtual cameras

Virtual camera simulation in real environment involves re-sampling the data captured from real cameras. The virtual camera movements are generated by moving a cropping window inside the original recorded video (an example of virtual camera work is illustrated in Figure 3.4).



(a) Clipping (b) Panning

(c) Zooming

Figure 3.4: *Virtual camera work. Image reproduced from [AKK06]*

One major advantage which comes with virtual camera systems is that they can afford to work offline. It is true that applications like live broadcast of sports require real time online systems but many other applications do not pose such a restriction. Offline capabilities provide following benefits over online systems:

- More computationally expensive computer vision algorithms can be used in offline systems which are not possible to be used in real-time
- Looking ahead in time is beneficial to generate more gracious and skillful looking virtual camera movements. On the other hand, online systems are often limited to simple and cautious camera movements (as it requires prediction of the near future events)
- Looking at the entire (or parts) video simultaneously can help better help enforcement of temporal constraints, for example avoiding jittering and adhering to the cinematic rules
- Allows more enhanced user interaction and iterative improvements based on the user feedback

The work by Gleicher et al. [GM00] was one of the earliest paper to propose a conceptual framework for virtual camera simulation. Their method was specifically aimed for recording classroom lectures and they called it virtual videography. Several more integrated approaches involving virtual camera simulation have been followed over the years. These systems have mainly focused on two specific application scenarios: the first is lecture halls or video conferences [SFKM05, YF05, HWG07, MAP*10] and the second is sports [DO04, AKK06, CDV10]. Most of these systems employ a common framework: one or more high-resolution cameras capture the event and features such as location of the players or a presenter are extracted offline. A novel video (to be presented to the viewer) is then generated as a post process by determining the optimal subregion of the appropriate fixed camera at each time instant.

One of the main components which differentiates these methods is the way the optimal sub-region to be cropped is determined. The majority of virtual camera systems use human detection and tracking for this purpose (face or the entire body). In lectures and video conferences this is usually limited to a single presenter whose location is estimated using background subtraction [HWG07, MAP*10] or frame differencing [YF05] using a fixed static camera. This

is easy to compute given the simplicity of the lecture environment and the information is sufficient to frame a single presenter. More variety has been proposed in sports.

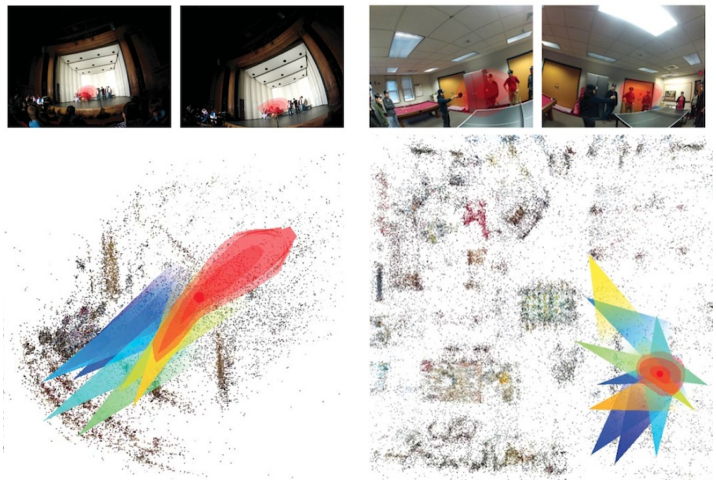
Chen et al. [CDV10] presented a method for personalized production of basketball videos. They use multiple static cameras to track players, referee and the ball. They also compute the identification information based on the jersey numbers [DDDV09]. The subregion within a camera is then computed based on a set of fixed rules (for example the ball is given the highest interest so that it is always included in the scene) and user preferences (a player specified by the audience is assigned a higher interest than a player not specified).

Ariki et al. [AKK06] proposed a method for digital zooming in soccer games by automatically recognizing the game situations or events such as penalty kick or a free kick based on player and ball tracking. The tracking in their case is obtained using background subtraction and the event recognition is performed based on simple rules (for example a penalty kick is detected when the ball is static for several minutes and is placed near the penalty spot). Their method then defines different clipping window sizes based on different events.

Daigo et al. [DO04] uses audience gaze direction to estimate the areas of interest in a basketball match (assuming that the audience are always looking at the important part of scene). They also use other static cameras to detect rough regions where the players exist (based on motion). The broadcasting video is then generated using the combination of these two sets of information by virtual panning in a panoramic video created using four static cameras. They demonstrate that the individual camera resolution can be preserved by virtual panning in a panoramic videos. The framework of automatically cropping sub-clips from a panoramic video for sports broadcasts has also been proposed in other works [SFWS13, GLL*13, GLS*14]. Particularly, the FascinatE [SFWS13, KWK12] project has showcased some interesting tools for real time interactive virtual camera editing in ultra high resolution videos. They have also demonstrated real time person tracking (HOG based detection with point feature tracks) to be used for virtual direction in such high resolution videos [KTK*11].

Carr et al. [CMM13] proposed a hybrid approach combining robotic and virtual camera movements for aesthetic recording of basketball games. Their method employs a robotic camera to follow the centroid of player positions estimated using a real time person detector. The images captures by the robotic camera are then resampled (cropped) to generate the video frame of a virtual camera after a short delay to obtain smoother camera trajectories (the delay allows to better plan the virtual camera moves). They show that the loss of image resolution can be minimized by using a hybrid system (compared to cropping in single a static camera). Similar approach has been also used by Zhang et al. [ZRCH08] for recording lecture, which first uses a robotic camera to keep the lecturer in the center of the image and then crops a subregions of this image to compensate for motor errors.

Arev et al. [APS*14] looked at the problem of selecting rushes and editing them together using a large number of viewpoints taken by "social cameras" (i.e. cameras operated by the audience) possibly with reduced resolution. Their method used the idea of social saliency fields [PJS12] i.e that the social cameras share the focus of attention of the people carrying them and this information can be exploited to determine the important or salient parts of the scene (illustrated in Figure 3.5).



(a) Musical scene (b) Party scene

Figure 3.5: *Park et al. [PJS12] showed that the salient regions in a scene can be identified using the convergence of field of view multiple social cameras looking at the scene. Two examples from their work are shown above, one in a musical scene and another in a party scene. The field of view of each camera is modeled as a cone. Arev et al. [APS*14] showed that this idea can be used for automatic video editing.*

Another important component which differentiates virtual camera systems is the way cropping window moves inside the higher resolution recording (we have already discussed different approaches to select the optimal sub-region but not how the selected sub-region is temporally adjusted). The simplest approach is to choose a cropping window in each frame of the video independently based on the computed region of interest. But such a straightforward approach leads to noisy and annoying camera movement and some form of temporal regularization is used in almost all applications. Variety of different methods for obtaining smooth virtual camera movement have been proposed in previous work:

Chen et al. [CDV10] use Gaussian Markov Random Fields (MRF) to obtain smooth virtual camera movements. Their method uses a likelihood term to keep the final solution closer to the optimal subregion at each individual frame and an inter-frame smoothness prior to obtain jerk free camera movement (to avoid dramatic switching from a frame to another). Yokoi et al. [YF05] used bilateral filtering to smooth the noisy virtual camera trajectory obtained using region of interest (ROI) computed at each frame. They further partitioned the smoothed tracks into segments where the camera is stationary and the camera is moving, based on a fixed threshold. In the moving segments of the camera they investigated the application of learned parameters of camera panning by Kato et al. [KYA*97] to regulate the position of the subregion.

Sun et al. [SFKM05] presented a method for smooth ROI extraction in panoramic video using simple tracking approaches. Their method first uses Kalman filter to smooth out noise from the initial tracking estimates (assuming the noise is gaussian). A set of additional processes are then applied on top of these smoothed tracks for a improved virtual camera control. To avoid small movements of the camera, their method keeps ROI unchanged if the Kalman filter predictions of new position are within a specified distance of the registered position and the new

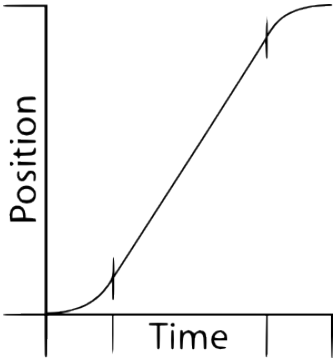


Figure 3.6: *The tracking shot proposed in [HWG07]. The function starts with a velocity of zero, accelerates over a set duration, holds a constant velocity, and then decelerates to a static state.*

estimated velocity is below a threshold. When the predictions are higher than the threshold, their method applies an additional low pass filter to further smooth the virtual camera motion (ROI tracks).

In the virtual videography systems proposed by Heck et al. [HWG07], the virtual camera switches between static shots at different sizes and a moving camera shot (a tracking shot) based on a combinatorial optimization function. They state that a good camera tracking shot should consist only of smooth camera motions in a single direction that accelerate and decelerate gradually to avoid jarring to the viewer. Given the bounding boxes at two different time instants, their method fits a tracking shot based on the function illustrated in Figure 3.6.

Virtual camera motion has also been studied concerning other applications. Liu et al. [GL08] proposed the re-cinematography approach that processes recorded video clips to improve the apparent camera movements. There method first uses the motion estimation techniques [Sze06] to compute the actual (real) camera motion. Then their method partitions and classifies the video into static (when the real camera motion is low) and moving segments. For the static segments they search for the best subregion in the video to be used for the entire duration (which minimizes the amount of uncovered area). For moving segments, their method creates a virtual camera path using greedy key-frame insertion (based on a penalty term) with linear interpolation in-between.

A more general approach for generating virtual camera paths was recently proposed by Grundmann et al. [GKE11] for video stabilization. They cast the problem into an unified constrained L1 optimization framework, minimizing the information loss and the first, second and third order derivatives. Similar to Liu et al. [GL08] their method also first estimates the state space trajectory of the real camera using interest point tracks and then optimizes it into a smoother version. A novel stabilized video is then created by moving a virtual cropping window over the optimized camera trajectory. Their method ensures that the cropping window is always fully inside the original video using hard constraints, avoiding any artifacts which may result due to inpainting required otherwise (as in [GL08]).

3.2.3 Automatic camera selection and editing

An important problem which arises in multiple camera systems is the selection of a camera which should be "on air" at a given time. The problem is also known as multiple camera editing. It involves two main aspects, first is to select the camera which best describes a dynamic scene at a given time and second to smoothly switch from a camera to another (avoiding jump cuts or jerks). In some cases it is also important to switch from a camera to another to create variety to keep the viewer interested (for example cutting to audience in a lecture recording for short intervals can enhance interest of the viewer). For applications like online sports broadcasts these decisions need to be made real-time, on the other hand in other applications like movies or interviews the editor can take more time to experiment. For professional videos this process is done manually by expert editors.

We have already discussed several guidelines for video editing with detail in the previous chapter (Section 2.3) with regard to motivation, movement, action continuity and spatial continuity. We have also discussed that editing in synchronized set of multiple camera setup often comes down to selection and careful switching among available choices. Now we will review some automatic camera selection methods which have been proposed in the previous work. We will discuss different selection algorithms irrespective of techniques used for content creation (covering both robotic and virtual camera systems). Our discussion will cover both online and offline systems.

He et al. [HCS96] used finite state machines(FSM) to decide when to select cameras in a virtual party scenario. Similar approach was proposed on real data by Liu et al. [LRGC01] for automated lecture recording. There method used a three state FSM to switch between a robotic speaker tracking camera, a robotic audience tracking camera and a static overview cameras looking at the entire scene. Transitioning from one state to another was governed by transition probabilities defined based on set of rules (developed by interviewing 7 professional video producers). The example rules were: (a) Establishing the shot first using overview camera; (b) Avoiding jump cuts; (c) Each shot should be longer than a minimum duration and shorter than a maximum duration etc. A similar approach was used by Heck et al. [HWG07] to switch between virtual camera shots of different types and sizes. The advantage being that by working offline they were able to model the problem of obtaining the best shot sequences of the entire video into a single constrained combinatorial optimization.

Doubek et al. [DGSG04] proposed a rule based system for selecting the best view possible in a camera network of static cameras. Their system also utilizes virtual zoom (for selecting only part of a view) and view-interpolation (to generate a novel views utilizing two different cameras). They employed cinematography inspired rules like using a long shot when the subject is moving and switching to a medium shot when it comes to rest. Fixed constraints and with user defined thresholds were used to regulate the camera switching.

Arev et al. [APS*14] used a trellis graph representation to optimize an objective function that maximizes coverage of the important content in the scene, while respecting cinematographic guidelines. Each node of the trellis graph was represented by framing of a particular camera at a given time and the cost of each node was computed using multiple factors like stabilization (preferring low frequency motion); camera roll (preferring camera angles aligned to the horizon); joint attention (preferring camera better covering the social salient regions) and

a global vector (preferring cameras in a certain direction). They further defined edge costs for continuing along the same camera or transitioning from a camera to another. These edge costs were used to regulate cinematic guidelines for example to avoid staying on the same camera for long duration or to penalize jump cuts.

Wang et al. [WXC*08] utilized actual recorded TV broadcasts to learn a HMM model for view selection between multiple cameras in a soccer match. They cast the problem as switching between a main camera (providing a panoramic view of the game) and the best available sub camera (providing more detailed tighter shots). Chen et al. [CDV10] also used HMM for selecting the right view to display among the multiple video streams captured by the investigated camera network in a basketball match. They jointly model the problem of first computing an optimal subregion in each static camera and then choosing the best option among the selected subregions in different cameras. Ronfard et al. [Ron09] described a method for selecting rushes of virtual cameras in virtual environments based on film editing principles by minimizing a semi-Markov cost function using dynamic programming. The method was shown to agree with subjective audience judgments [GRLC15].

Takemae et al. [TOM04] proposed a video editing system based on the participants gaze direction in indoor conversations. They showed results on debate scenarios consisting of 3 to 4 participants and where the gaze directions of each participant was manually labelled for the entire video. They applied a simple editing rule i.e. to cut to the close up of a person that most participants are gazing at. They performed a user study to compare this simple editing approach with three other ways of visual representations: first a wide shot (where all the participants were captures in a single shot), second a multiple view shot (where close ups of all participants were concatenated together) and third a speaker shot (always switching to the close shot of the speaker). The user study suggested that the gaze based approach was able to better convey the flow of conversation compared to other visual representations.

Recently two data driven approaches were employed by Chen et al. [CWH*13] for automatic view selection. In first approach they train two class SVM to learn a good/bad predictor of clips. They used a 25 dimensional feature covering variety of aspects like ball visibility, player distribution, temporal smoothing etc. For training the SVM, they determined aesthetic labels for different clips through a user study consisting of 35 participants. They were able to achieve 74 percent accuracy in prediction compared to user preferences. They further used the SVM scores to rank different clips for view selection. Interestingly, more than half the time their selection disagreed with the choices made by a broadcast director. In further experiments they directly learnt directorial styles using random forest classifiers based on a training set of previous broadcasts. They showed that this way of classification was able to reflect the director's preference very well (the director's choice in over ninety percent of the frames were in either the first or second choices suggested by the classifier).

3.3 COMPARISON WITH PROPOSED WORK

Previous literature of autonomous camera systems has mainly concentrated on the specific scenarios of lectures and sports. Many of these methods are fully automatic and do not allow for any user interaction or experimentation. Such closed form solutions have been successful in

the application of recording lectures but in other scenarios including sports they still remain in early stages and far from replacing the existing frameworks. Moreover, none of the previous approaches have looked at the general problem of automatic video production, for example how to edit a video with a group of actors talking to each other or three people dancing on a stage.

If we consider the two main stages of videos production, the rush generation and the editing, the cost and time spent on latter has significantly come down by the introduction of non linear editing systems. The editing softwares available these days allow quick and free form experimentation suited for this creative procedure. But the task of obtaining the rushes still remains an expensive and difficult process. The contributions of this dissertation are mainly focused to simplify the process of rush generation. Surprisingly no other method has ever looked at the general problem of automatic rush generation.

Our application domain (stage theatre) offers a much larger variety of situations than previous work with different set designs, different number of actors, diverse movement and dynamics (of both actors and objects), varying costumes and postures, artistic compositions/lighting, variety of gestures and actions of actors, expressive dialogue scenes etc. In fact it represents a close movie like scenario and theater performances are popularly recorded to be broadcasted on various medias (television, internet, dvds etc.). To our knowledge it is the first attempt to use such rich content in the field of automated camera systems.

The problem of editing a theatre sequence is much harder than that of editing a lecture video due to a number of additional complexities like lighting changes; unknown number of actors; fast movements; occlusions etc. It makes both the tracking (Chapter 5 and Chapter 6) and virtual camera simulation (Chapter 7) difficult. The required high aesthetic standards in case of theatre precludes ad hoc solutions for fully automatic editing used in the previous work. Thus, instead of proposing a fully automatic editing approach, we focus on a sub-problem of rush generation. Our work has developed based on the feedback from video producers who actually perform the professional video recording of live theatre and the proposed solution directly fits into the existing editing pipeline. Our work of automatic rush generation aims to empower editors instead of entirely replacing them, leaving the room for creativity.

Our work builds on the computational model for "good" camera movements (particularly in terms of subwindows of video) from the previous work [GKE11]. While the previous work uses the model for video stabilization and simply aims to remove jitter, we extend it to the problem of multiple rush generation. We will discuss this in detail in Chapter 7.

CHAPTER

4

THEATRE DATASET AND PROSE STORY BOARD LANGUAGE

In this chapter we present the first two contributions of this dissertation i.e. building the theatre dataset and the Prose Story Board Language (PSL). We will discuss the challenges in recording live theatre and will provide the details of the actual setup which was used to record the sequences. The reasons for choosing the particular setup will also be explained before giving the details about the recorded sequences. We describe the Prose Story Board Language (PSL) and its application for shot by shot annotation of existing movies and automatic rush generation.

4.1 THEATRE DATASET

The first obvious step towards cost effective editing of live theatre is to build an economical camera setup which can capture the theatre performances efficiently. The task of recording theatre sequences is extremely challenging. The possible camera positions are restricted and there is no check over the lighting of the scene. There is no control over the performance and there is no possibility for retakes if something goes wrong. Moreover the recording in our case is done for the purpose of virtual camera simulations and that imposes some additional requirements. Overall the camera setup should include the following requisites:

4.1.1 Required specification

- Economical and simple to use : The cameras used in professional film acquisition are extremely expensive and they are complicated to use. Such cameras are often accompanied by special set of equipment which are difficult to setup and adds up the shooting costs. Such an expensive setup is not affordable for low cost productions so the camera setup should be economical. Also, in our application the setup is supposed to be used by theatre personals so it should be easy to use (ideally just a start stop button).
- Non obtrusive : The camera setup should not disturb the audience present during the live theatre performances. The camera setup should be small and unnoticeable.
- High resolution : As the master shot will be used for simulating sub-resolution videos (using virtual pan/zoom), the camera should support high resolution recordings.
- Wide angle : It is required that the camera should cover the entire stage, considering the width (entire scene of action) this requires wide angle recording capabilities.
- A neutral angle : If only a single camera is used, the preference should be for neutral vertical and horizontal angle. As special camera angles may establish a special meaning to the shot (as we have discussed in Section 2.1.4 of the previous chapter)
- Should perform well in low light situations : Low light scenes are common in live theatre performances and to properly record such scenes requires employing faster lenses (a lens with a larger maximum aperture is called a "fast lens" because it delivers more light intensity to the focal plane, achieving the same exposure with a faster shutter speed).
- Large depth of field : The term depth of field (DOF) is the distance between the nearest and farthest objects in a scene that appear acceptably sharp in an image. Although a lens can precisely focus at only one distance at a time, the decrease in sharpness is gradual on each side of the focused distance, so that within the DOF, the unsharpness is imperceptible under normal viewing conditions. A theatre staging can be quite deep and it is important that the camera should record the entire stage sharply.

Figure 4.1: *A frame extracted from a sequence recorded in interlaced mode. Notice the artifacts in form of jagged edges near the boundaries. This is also known as 'combing' artifact.*

- Progressive mode : Progressive mode is a way of storing or transmitting moving images in which all the lines of each frame are drawn in sequence. Since our framework requires the processing of individual frames of a given sequence it is necessary to make the recordings in the progressive mode. The interlaced mode only stores odd or even lines of the frame alternatively and this might cause some artifacts while processing individual frames as illustrated in Figure 4.1.
- High dynamic range : The usage of directional lighting in theatre may result in a wide range of intensity variations (some parts of the stages dark and some brightly lit). As it is not possible to use additional lighting equipment the camera setup should support high dynamic range in recording.

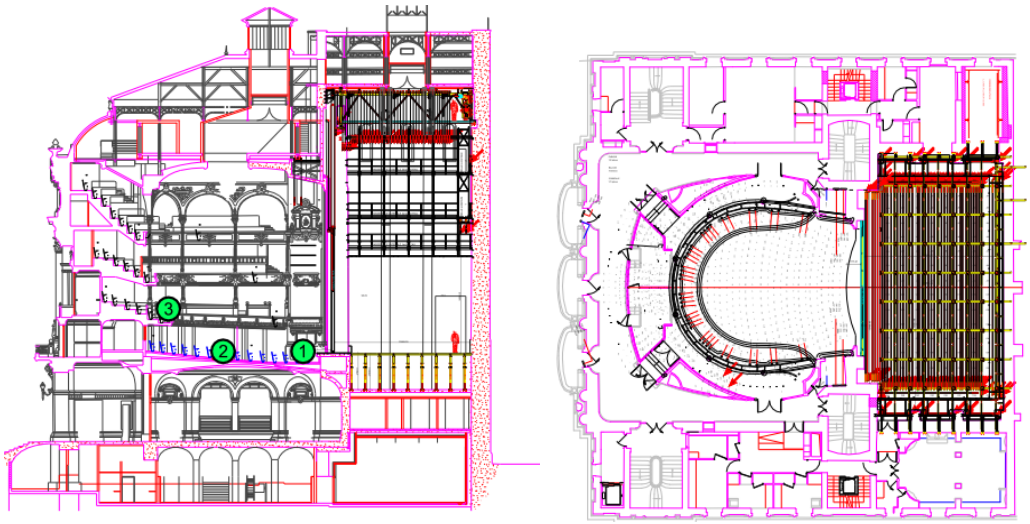
4.1.2 Employed Setup and Hardware

The architectural map of theatre de Celestin is illustrated in Figure 4.2. We tried three different feasible locations for the camera positioning and they are shown with green dots in Figure 4.2(a). The camera positions were following:

- First, just at the end of stage and before the first row of the audience.
- The second, behind the directors seat (behind 6th row of the audience).
- And third at the balcony, on the first floor.

The recordings were done during the rehearsals (both dress rehearsals and normal day rehearsals) and all the three locations were suitable to operate from, without causing any inconvenience to the audience or the performers.

The video frames recorded from these three different viewpoints are illustrated in Figure 4.3. All three positions offered different advantages, the setup in the first viewpoint is closest to the stage and captures more light as compared other two positions. The view point from the balcony is often preferred for aesthetic reasons (also known as the Prince's view). The second viewpoint gives a eye level recording where the camera viewpoint is parallel to the floor of the stage. It is called a neutral view and is used as a general guideline in film editing, as the captures done at particular angle from the stage floor might impose some special meaning to the scene. A neutral viewpoint may also be advantageous for applying computer vision algorithms. Considering these factors we decided to choose the second viewpoint for the final camera setup. Now we will bring our discussion to the actual camera setup which was used for



(a) Side view (b) Top view

Figure 4.2: *The architectural plan of theatre de Celestin. The green dots in the side view show the camera placements corresponding to three different camera view-points which were tested in our experiments.*

recording the videos.

It took us several visits to the theatre to estimate the approximate lighting conditions and the angle (field of view) required to cover the entire stage from the given viewpoints. After trying different possibilities we used a micro four third system consisting of the Lumix GH2 with a 20mm pan cake lens. The reasons for choosing this setup were the following:

1. The micro four third (MFT) assemblies are smaller and lighter, as the MFT system design specification does not provide space for a mirror box and a pentaprism. This allows smaller bodies to be designed with a shorter flange focal distance (the distance between the mounting ring to the film plane).

2. Lumix GH2 allows for recording in full HD (1920×1080) at 24P (progressive scanning with 24 frames per second). Cameras with 24P formats have become very famous in

(a) View point 1 (b) View point 2 (c) View point 3

Figure 4.3: *Frames of sequence from the play, À l'ouest recorded from three different view points. Recorded frame from: (a) A low camera setup ahead of the front row of the theatre room (just at the end of stage); (b) A camera setup at stage level behind the director's seat; (c) A camera setup ahead of the front row of the balcony (or the Prince's view).*

video acquisition, since they deliver film like motion characteristics and provide a much more economic alternative to film cameras.

3. The shorter flange focal distance allows for smaller, lighter and lower cost lenses (especially smaller wide angle lenses without any spherical distortions are available for micro four third assemblies). We tested two different wide angle pan cake lenses and the comparison is given in Table 4.1. The 20mm lens provided the larger maximum aperture which is an important factor for shooting in low light situation like theatre. Due to this reason we chose the 20mm lens in the final setup. Although 14mm lens provided the benefit of much wider field of view, the 20mm lens was sufficient to cover the entire stage from the desired location.

	G 20mm f/1.7 h	G 14mm f/2.5 h
Focal Length	f=20mm	f=14mm
Maximum Aperture	f/1.7	f/2.5
Minimum Aperture	f/16	f/22
Diagonal angle of view	57°	75°

Table 4.1: *Comparison of Lumix pancake lenses*

4. The small focal length lens and a smaller sensor size gives deeper depth-of-field. For obtaining deep depth of field, four parameters are important: aperture (lower the better); focal length (lower the better); distance of the object (farther the better); and the digital sensor size (smaller the better). In low light scenarios like theatre, it is not possible to reduce aperture. But the combination of a lens with smaller focal length, a digital sensor which is not so large and good enough recording distance provided a sufficiently deep depth of field in our case.

5. Although the dynamic range was not extremely impressive (which was expected given a moderate sensor size), the results were fairly acceptable except in some extreme situations.

6. The electronic viewfinder (EVF) in micro four third assemblies is quite beneficial. One of the major benefits is that EVF allows to zoom into the preview, which is something a mirror-based viewfinder cannot do. This zooming capability makes it easier to achieve quite precise manual focus (compared to setting the manual focus through a mirror). The EVF's also give the real time preview of the recordings. It displays how the sensor will see one's potential picture, rather than an optical view, which may differ.

In conclusion, the system met all of the desired requirements. We tried to operate the system both in manual and automatic focus modes. While operating at automatic focus mode, the camera was found re-searching for the focus quite regularly with low and fast changing illumination. So finally we operated the camera on the manual focus which was fixed at the start of the recording and kept unchanged throughout the day. The aperture was kept at the maximum and the shutter speed was fixed at 1/50th of a second (following the rule of thumb of keeping the denominator at least twice the frame rate).

A l'ouest
by: Nathalie Fillion

Lorenzaccio
by: Claudia Stavisky

Death of a Salesman
by: Claudia Stavisky

Figure 4.4: *Screenshots from three different plays recorded from the rehearsals at Celestin, Theatre de Lyon. The name of the play and the director are given in the captions below the screenshots.*

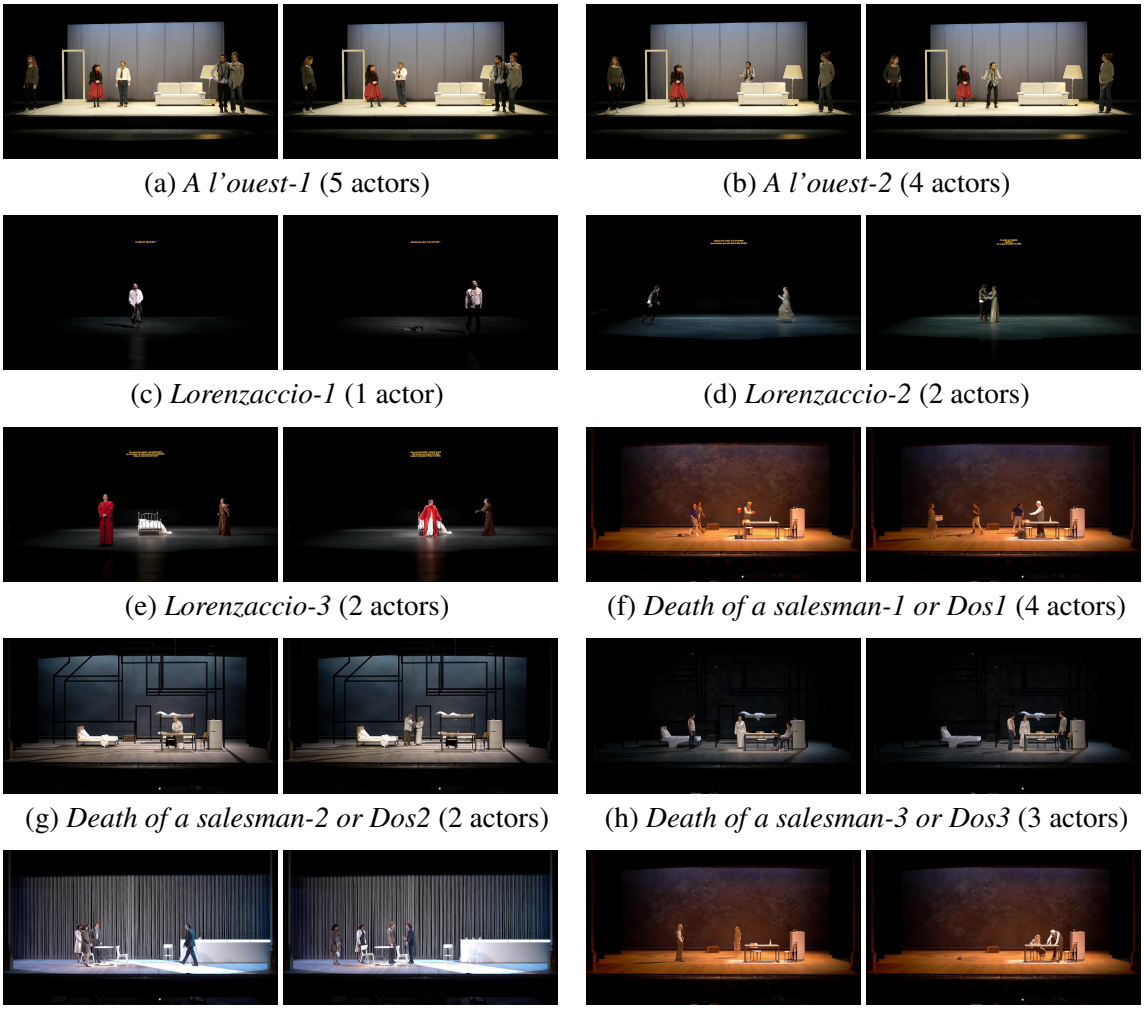
4.1.3 Recorded sequences

We recorded three different plays as part of the Scenoptique project in partnership with Celestin, Theatre de Lyon. The recordings were made both during the normal rehearsals and the dress rehearsals. The archived dataset currently consists of nearly a total of 19 hours recording, including two full dress rehearsals. The video sequences were recorded from rehearsals of three different plays: *A l'ouest* directed by Nathalie Fillion; *Lorenzaccio* directed by Claudia Stavisky; and *Mort d'un commis voyageur (Death of Salesman)* directed by Claudia Stavisky. The screenshots from three different recorded plays are shown in Figure 4.4. The database includes 140 minutes of rehearsals from the *A l'ouest*, 210 minutes of rehearsals from *Lorenzaccio* and 820 minutes of rehearsals from *Death of a salesman*. The recordings were obtained in AVCHD (Advanced Video Coding High Definition) format. The archived dataset provides a rich resource for research in virtual camera systems in real environments. It also provides an interesting scenario for studying other computer vision algorithms like actor detection and tracking.

For further analysis we selected several interesting short sequences from the recorded videos and hand annotated the actor positions and the identity information. A total of 10 sequences were annotated, the smallest sequence is of 1094 frames and the largest of 1788 frames. The set includes two sequences from *A l'ouest*, three from *Lorenzaccio* and five from *Death of a salesman*. The sequences cover examples with a single actor to up to 5 actors present on stage in varying illuminations. The selected sequences include multiple instances of interesting events like entry, exit, cross, sit, stand etc. They also include cases with fast and significant motion of the actors on stage. Example frames from the selected sequences are shown in Figure 4.5. The cropped sequences were encoded in the same H.264/MPEG-4 AVC codec as in original AVCHD recordings without quality loss (directly extracted from the container and were repackaged without any loss). We used the open source ffmpeg library for extracting the sequences. As the goal is to use these sequences for virtual camera simulation, it is important to monitor the codec and quality of the recorded videos.

4.2 Prose Story Board language

In movie production, directors often use a semi-formal idiom of natural language to convey the shots they want to their cinematographer. Similarly, film scholars use a semi-formal idiom of natural language to describe the visual composition of shots in produced movies to their read-



(a) *A l'ouest-1* (5 actors) (b) *A l'ouest-2* (4 actors)

(c) *Lorenzaccio-1* (1 actor) (d) *Lorenzaccio-2* (2 actors)

(e) *Lorenzaccio-3* (2 actors) (f) *Death of a salesman-1 or Dos1* (4 actors)

(g) *Death of a salesman-2 or Dos2* (2 actors) (h) *Death of a salesman-3 or Dos3* (3 actors)

(g) *Death of a salesman-4 or Dos4* (4 actors) (h) *Death of a salesman-5 or Dos5* (2 actors)

Figure 4.5: *Frames from selected theatre sequences*

ers. In order to build intelligent and expressive virtual cinematography and editing systems, we believe the same kind of high-level descriptions need to be agreed upon. We propose a formal language that can serve that role. Our primary goal in proposing this language is to build software cinematography agents that can take such formal descriptions as input, and produce fully realized shots as an output [CLR12, Ron12]. A secondary goal is to perform in-depth studies of film style by automatically analyzing real movies and their scripts in terms of the proposed language [RTT03, Ron04].

The prose storyboard language is a formal language for describing shots visually. We leave the description of the soundtrack for future work. The prose storyboard language separately describes the spatial structure of individual movie frames (compositions) and their temporal structure (shots). Our language defines an infinite set of sentences (shot categories), which can describe an infinite set of shots.

In film analysis, there is a frequent confusion between shots and compositions. A *medium shot* describes a composition, not a shot. If the actor moves towards the camera in the same

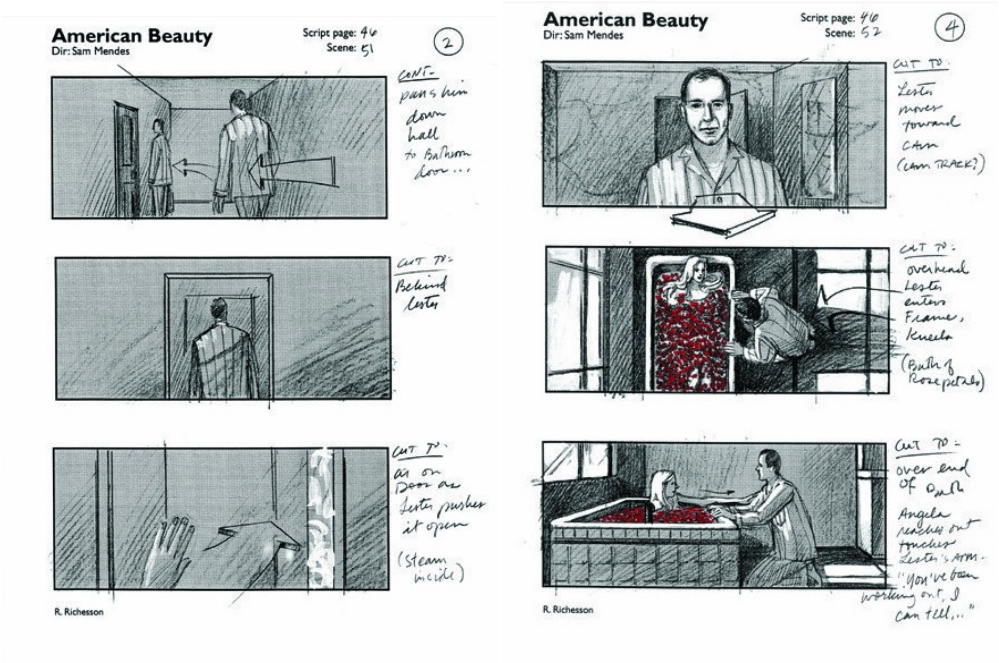


Figure 4.6: *An example graphic storyboard from the movie American Beauty. The camera movement is marked by arrows and accompanying text is written on the right.*

shot, the composition will change to a *close shot* and so on. Therefore, a general language for describing shots cannot be limited to describing compositions such as *medium shot* or *close shot* but should also describe screen events which change the composition during the shot.

The language can be used indifferently to describe shots in pre-production (when the movie only exists in the screen-writer and director's minds), during production (when the camera records a continuous "shot" between the times when the director calls "camera" and "cut"), in post-production (when shots are cut and assembled by the film editor) or to describe existing movies. The description of an entire movie is an ordered list of sentences, one per shot. Exceptionally, a movie with a single shot, such as Rope by Alfred Hitchcock, can be described with a single, long sentence. The scope of PSL is much broader than the main application of automatic rush generation pursued in this dissertation, but we will show how a simplified version of PSL can be used for generating rushes from a master shot.

We assume that all shot descriptions are manually created. We leave for future work the important issue of automatically generating prose storyboards from existing movies, where a number of existing techniques can be used [Bra97, VRB00, DMR05, GCSS06, GR13]. We also leave for the future works, the difficult problems of automatically generating movies (in 3D virtual world) from their prose storyboards, where existing techniques in virtual camera control can be used [HCS96, CAwH*96, JY06, CON08].

4.2.1 Related Work

Our language is loosely based on existing practices in movie-making which we have discussed in Chapter 2 and previous research in the history of film style [Bor98, Sal09]. Our language is

also related to the common practice of the graphic storyboard. In a graphic storyboard, each composition is illustrated with a single drawing. The blocking of the camera and actors can be depicted with a conventional system of arrows within each frame, or with a separate set of floor plan views, or with titles between frames. An example of a graphic storyboard is shown in Figure 4.6.

In our case, the transitions between compositions use a small vocabulary of screen events including camera actions (pan, dolly, crane, lock, continue) and actor actions (speak, react, move, cross, use, touch). Although the vocabulary could easily be extended, we voluntarily keep it small because our focus in this work is restricted to the blocking of actors and cameras, not the high-level semantics of the narrative.

We borrow the term prose storyboard from Proferes [Pro08] who used it as a technique of decomposing a film's script into a sequence of shots, expressed in natural language. The name catches the intuition that the language should directly translate to images. In contrast to Proferes, our prose storyboard language is a formal language, with a well defined syntax and semantics, suitable for future work in intelligent cinematography and editing.

Our proposal is complementary to the Movie Script Markup Language (MSML) [RKV*09], which encodes the structure of a movie script. In MSML, a script is decomposed into dialogue and action blocks, but does not describe how each block is translated into shots. Our prose storyboard language can be used to describe the blocking of the shots in a movie in relation to an MSML-encoded movie script.

Our proposal is also related to the Declarative Camera Control Language (DCCL) which describes film idioms, not in terms of cameras in world coordinates but in terms of shots in screen coordinates [CAwH*96]. The DCCL is compiled into a film tree, which contains all the possible editings of the input actions, where actions are represented as subject-verb-object triples. Our prose storyboard language can be used in coordination with such constructs to guide a more extensive set of shot categories, including complex and composite shots.

Our approach is also related to the work of Jhala and Young who used the movie Rope by Alfred Hitchcock to demonstrate how the story line and the director's goal should be represented to an automatic editing system [JY06]. They used Crossbow, a partial order causal link planner, to solve for the best editing, according to a variety of strategies, including maintaining tempo and depicting emotion. They demonstrated the capability of their solver to present the same sequence in different editing styles. But their approach does not attempt to describe the set of possible shots. Our prose storyboard language attempts to fill that gap.

Other previous work in virtual cinematography [SMAY03, JY05, FF06, ORN09, LCCR11, MJSB11] (both in real and virtual environments) has been limited to simple shots with either a static camera or a single uniform camera movement. Our prose storyboard language is complementary to such previous work and can be used to define higher-level cinematic strategies, including arbitrarily complex combinations of camera and actor movements, for most existing virtual cinematography systems.

4.2.2 Requirements

The prose storyboard language is designed to be expressive, i.e. it should describe arbitrarily complex shots, while at the same being compact and intuitive. Our approach has been to keep the description of simple shots as simple as possible, while at the same time allowing for more complex descriptions when needed. Thus, for example, we describe actors in a composition from left to right, which is an economical and intuitive way of specifying relative actor positions in most cases. As a result, our prose storyboard language is very close to natural language (see Figure 4.11).

It should be easy to parse the language into a non ambiguous semantic representation that can be matched to video content, either for the purpose of describing existing content, or for generating novel content that matches the description. It should therefore be possible (at least in theory) to translate any sentence in the language into a sketch storyboard, then to a fully animated sequence.

It should also be possible (at least in theory) to translate existing video content into a prose storyboard. This puts another requirement on the language, that it should be possible to describe existing shots just by watching them. There should be no need for contextual information, except for place and character names. As a result, the prose storyboard language can also be used as a tool for annotating complete movies and for logging shots before post-production. Since the annotator has no access to the details of the shooting plan, even during post-production [Mur01, Ond04], we must therefore make it possible to describe the shots in screen coordinates, without any reference to world coordinates.

4.2.3 Syntax and semantics

The prose storyboard language is a context-free language, whose terminals include generic and specific terms. Generic terminals are used to describe the main categories of screen events including camera actions (pan, dolly, cut, dissolve, etc.) and actor actions (enter, exit, cross, move, speak, react, etc.). Specific terminals are the names of characters, places and objects that compose the image and play a part in the story. Non-terminals of the language include important categories of shots (simple, complex, composite), sub-shots and image compositions. The complete grammar for the language is presented in Figure 4.12 in extended BNF notation.

4.2.4 Image Composition

Image composition is the way to organize visual elements in the motion picture frame in order to deliver a specific message to the audience. In our work, we propose a formal way to describe image composition in terms of the actors and objects present on the screen and the spatial and temporal relations between them.

Following Thomson and Bowen [TB09b], we define a composition as the relative position and orientation of visual elements called *Subject*, in screen coordinates. In the simple case of *flat staging*, all subjects are more or less in the same plane with the same size. Thus, we can describe this *flat composition* as follows:

```
<Size> on <Subject> [<Profile>][<Screen>]
{ and <Subject> [<Profile>][<Screen] }*
```

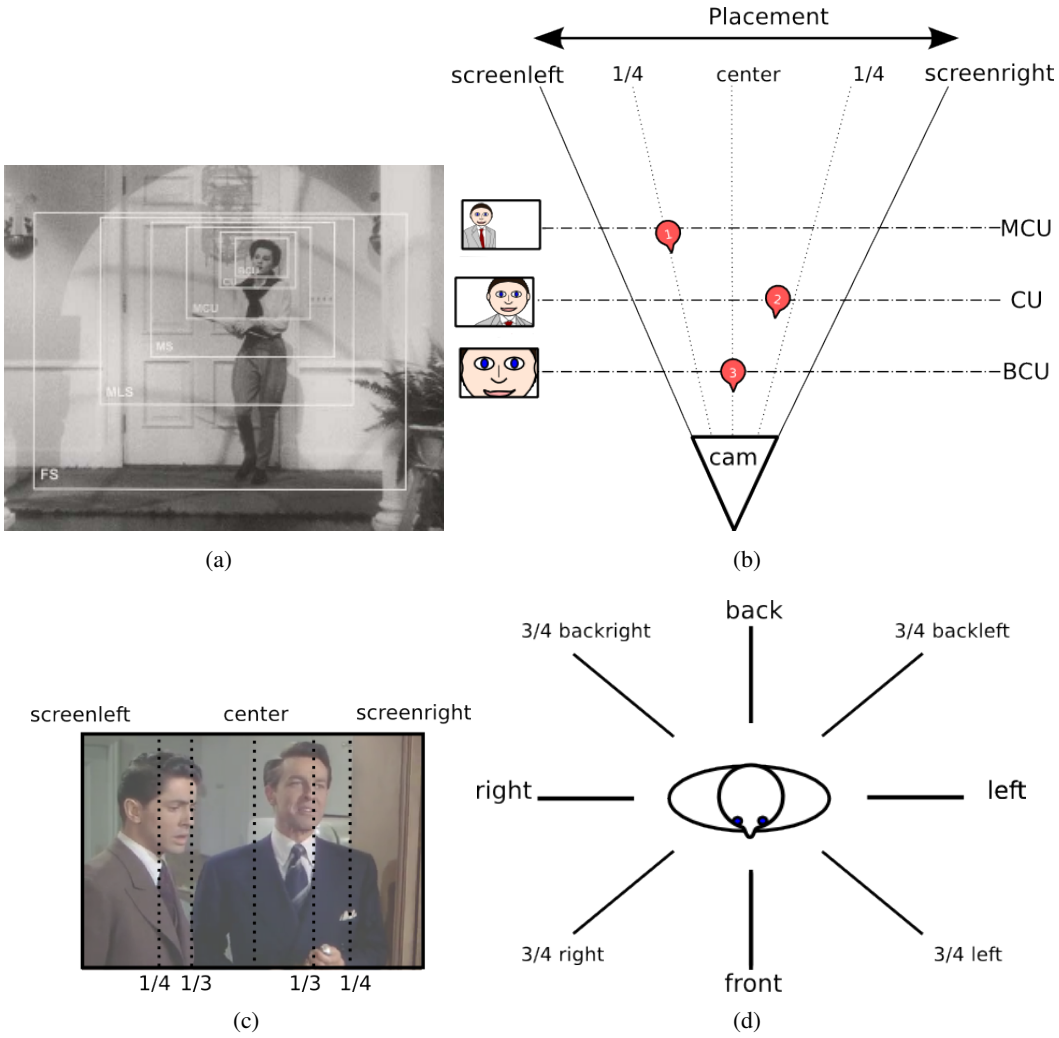
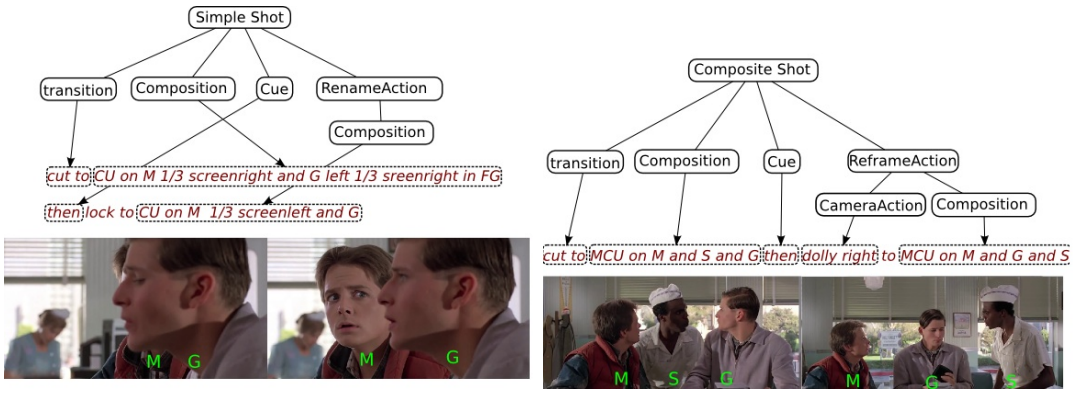


Figure 4.7: *A composition can be summarized using these four figures (summary in regard to prose story board language of what we have already discussed with detail in Chapter 2). (a) Shot sizes in the prose storyboard (reproduced from [Sal06]). (b) Shot size is a function of the distance between the camera and actors, as well as the camera focal length. (c) In prose storyboard language, the horizontal placement of actors in a composition is expressed in screen coordinates. (d) The profile angle of an actor defines his orientation relative to the camera. For example, an actor with a* left *profile angle is oriented with his left side facing the camera.*

where *Size* is one of the classical shot size illustrated in Figure 4.7(a) and *Subject* is generally an actor name. The *Profile* and *Screen* terms describe respectively the orientation and the position of the subject in screen space. See full grammar in Figure 4.12 for more details.
In the case of *deep staging*, different subjects are seen at different sizes, in different planes. We therefore describe such shot with a *stacking* of flat compositions as follows:

<FlatComposition> {, <FlatComposition>}*

As a convention, we assume that the subjects are described from left to right. This means that the left-to-right ordering of actors and objects is part of the composition. Indeed, because



(a) (b)

Figure 4.8: *Two shots from the movie Back to the future with the description in* Prose Storyboard Language *(red) and their associated parse trees. Top : Simple Shot with a fixed camera. The composition changes due to actor movements. Bottom: Composite shot. The composition changes due to camera motion. In both examples the actors are named by the corresponding letters in green.*

the left-to-right ordering of subjects is so important in cinematography and film editing, we introduce a special keyword *cross* for the screen event of one actor crossing over or under another actor.

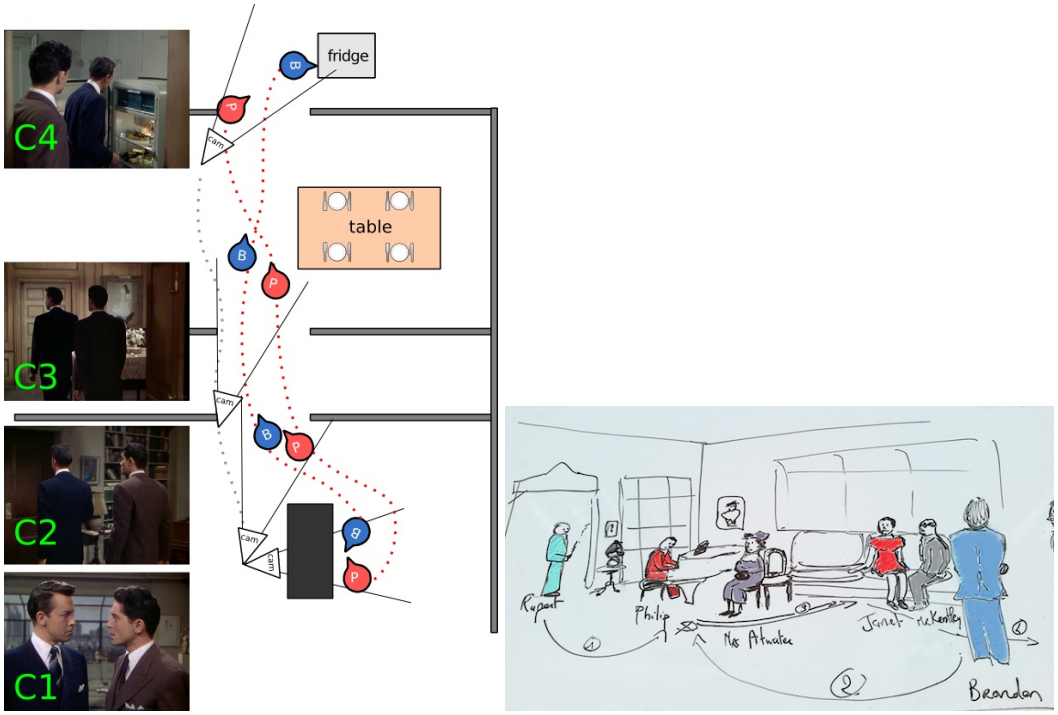
Shot sizes are used to describe the relative sizes of actors independently of the camera lens as illustrated in Figure 4.7(b). The $Profile$ term is used to describe the relative orientation of the subject with respect to the camera as illustrated in Figure 4.7(d).

The $Screen$ term describe the subject position in screen coordinate. It allows to slightly modify the generic framing by shifting the subject position to the left or right corner as shown in Figure 4.7(c). Thus, we can describe a more harmonious composition with respect to the head room and look room or the rules of thirds. We can also describe unconventional framing to create unbalanced artistic composition or to show other visual elements from the scene.

4.2.5 Shot Descriptions

Based on the taxonomy of shots proposed by Thomson and Bowen [TB09b], our prose storyboarding language distinguishes three main categories of shots :

- A simple shot is taken with a camera that does not move or turn. If the composition changes during a simple shot, it can only be the effect of the movement of the actors relative to the camera.
- A complex shot is taken with a camera that can pan, tilt and zoom from a fixed position. We introduce a single camera action (pan) to describe all such movements. Thus the camera can pan left and right, up and down (as in a tilt), and in and out (as in a zoom).
- A composite shot is taken with a moving camera. We introduce two camera actions (dolly and crane) to describe typical camera movements. Panning is of course allowed during dolly and crane movements.



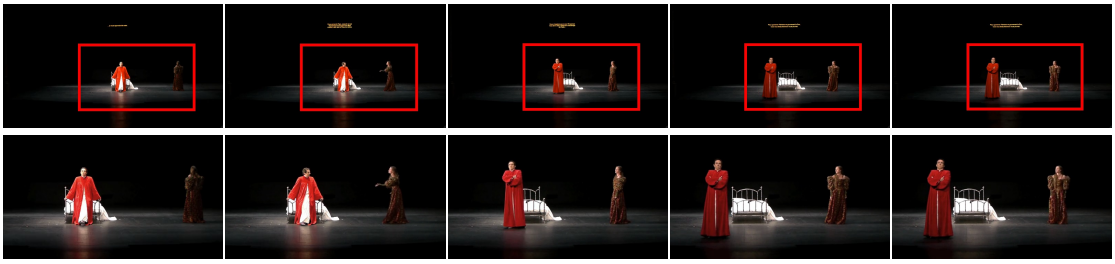
(a) (b)

Figure 4.9: *(a) Floor plan view of the first sequence in Figure 4.11. (b) Artist sketch of the second sequence in Figure 4.11*

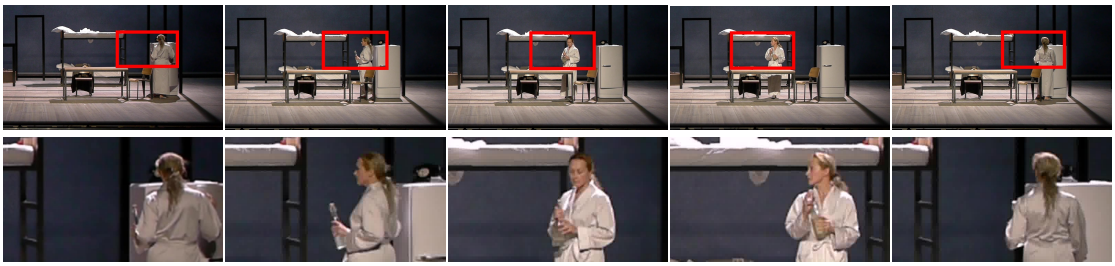
In each of those three cases, we propose a simplified model of a shot, consisting in a sequence of compositions and screen events. Screen events can be actions of the camera relative to the actors, or actions of the actors relative to the camera. A shot is therefore a sequence of compositions and screen events. Screen events come in two main categories - those which change the composition (*events-to-composition*) and those which maintain the composition (*events-with-composition*). In our model, we can be in only one of four different states:

1. Camera does not move and composition that does not change.
2. Camera does not move and composition changes due to actor movements.
3. Camera moves and composition does not change
4. Camera moves and composition changes.

In case (1), the shot can be described with a single composition. In case (2), we introduce the special verb *lock* to indicate that the camera remains static while the actors move, leading to a new composition. In case (3), we use the constructions *pan with*, *dolly with* and *crane with* to indicate how the camera moves to maintain the composition. In case (4), we use the constructions *pan to*, *dolly to* and *crane to* to indicate how the camera moves to change the composition, and we introduce a special verb *continue to* to indicate that the composition changes due to a combination of actor and camera movements. All three cases are illustrated in Figure 4.8, Figure 4.9 and Figure 4.11.



(a) Lock with FS on Cardinal and Lady



(b) Pan with MS on Linda 1/3 screenright.

(c) As Biff and Happy cross, pan to FS Willy and Biff and Happy.

Figure 4.10: *Use of PSL for vertical editing. Top rows in each case showing the original image from the sequence and the virtual camera window. Bottom rows in each case showing the cropped virtual camera frame. (a) Example of "lock with" shot where the virtual camera frame remains static while the actors move on stage. (b) Example of "pan with" shot, the virtual camera window moves to maintain the desired composition. (c) Example of "pan to" shot, the virtual camera window moves to change the composition.*

4.2.6 Experimental results

PSL for describing existing movies

As a validation of the proposed language, we have manually annotated scenes from existing movies covering different styles and periods. We illustrate the prose storyboard language on a particularly difficult example, a short sequence from the single-shot movie Rope by Alfred Hitchcock, where actor and camera movements interfere to produce a rich and dynamic visual composition. The prose storyboard for the example is presented in Figure 4.9 and Figure 4.11. Despite the complexity of the scene, the prose storyboard is quite readable and was relatively easy to generate. This example also illustrates how default values are used to describe simple cases. For instance, default values for a two-shot are supplied by assuming that the two actors are both facing the camera and placed at one-third and two-third of the screen. Such default

values are stored in a stylesheet, which can be used to accommodate different cinematographic styles, i.e. television vs. motion pictures, or musical vs. film noir.

PSL for vertical editing

A simplified version of PSL can be used for vertical editing (the simplified grammar is illustrated in Figure 4.13). We consider the specific case of simulating a virtual camera window within a static master shot. As the camera angle can not be adjusted in this scenario, describing a *composition* limits to size, subject and screen position. The shot descriptions also limit to three cases: *lock with* (where the cropping window remains static while actor movements may change the composition); *pan with* (the cropping window moves to maintain the composition); and *pan to* (the cropping window moves and composition changes). An example of each of these three cases is illustrated Figure 4.10. The procedure for obtaining such shots is described in Chapter 7 and Chapter 8

4.3 SUMMARY

The first part of the chapter elaborates the task of compiling the theatre dataset. It was recorded under certain constraints which motivated the employment of the setup conducive under it. The choices of camera position in the theatre along with the reasons for choosing them are described. Long recordings were performed during the rehearsals of three different plays and several shorter sequences were selected and hand annotated with the actor locations.

In the second part of this chapter, we have presented a language for describing the spatial and temporal structure of movies with arbitrarily complex shots. We have also shown how a simplified version of this language can be used for the purpose of vertical editing. The language can be extended in many ways, e.g. by taking into account lens choices, depth-of-field and lighting. Future work should be devoted to the dual problems of automatically generating movies in virtual 3D scenes from prose storyboards in machinima environments, and automatically describing shots in existing movies. It would also be interesting to extend this framework for the case of stereoscopic movies, where image composition needs to be extended to include the depth and disparity of subjects in the composition. We believe that the proposed language can be used to extend existing approaches in intelligent cinematography and editing towards more expressive strategies and idioms and bridge the gap between real and virtual movie-making.

Figure 4.11: *Prose storyboard for two short sequences from the single-shot movie Rope.*

```
<Scene> ::= <Shot> *

<Cue> ::= At <timeref> | As <Actor> <Action> | then

<Shot> ::= [<transition>] to [<Camera>] <Composition> {<Fragment>}* |
           <transition> <Camera>

<Fragment> ::= <Cue> (<RenameAction> | <ReframeAction>)

<RenameAction> ::= (lock|continue) to <Composition>

<ReframeAction> ::= <CameraAction> (to|with) <Composition>

<Composition>    ::= [<angle>] <FlatComposition>{, <FlatComposition>}*

<FlatComposition>::= <size>  on <Subject>[ <profile>][ <screen>]
                     { and <Subject>[ <profile>][ <screen>][in (back|fore)ground]}*

<CameraAction>::= [Speed]  pan [left|right|up|down]
                  | dolly[in|out|left|right]
                  | crane[up|down]

<Speed>   ::= slow|quick|following:(<Actor>|<Object>)

<Subject> ::= (<Actor>|<Object>) | (<Actor>|<Object>){, (<Actor>|<Object>)}*

<transition>  ::= cut|dissolve|fade in

<angle>       ::= (high|low)angle

<size>  ::= ECU|BCU|CU|MCU|MS|MLS|FS|LS|ELS

<profile>     ::= 34leftback|left|34left|front|34right|right|34leftback|back

<screen>      ::= center|[(13|14)]screen(left|right)

<Action>  ::= <Look>|<Move>|<Speak>|<Use>|<Cross>|<Touch>|<React>

<Look>    ::= looks at <Subject>

<Move>    ::= moves to <Screen>|<Place>|<Subject>

<Speak>   ::= (speaks | says <string>) [ to <Subject>]

<Use>     ::= uses <Object>

<Cross>   ::= crosses [over|under] <Subject>

<Touch>   ::= touches <Subject>

<React>   ::= reacts to <Subject>

<Place>   ::= from script

<Actor>   ::= from script

<Object>  ::= from script
```

Figure 4.12: *EBNF grammar of the prose storyboard language.*

```
<Scene> ::= <Shot> *

<Cue> ::= At <timeref> | As <Actor> <Action> | then

<Shot> ::= [<transition>] to [<Camera>] <Composition> {<Fragment>}* |
           <transition> <Camera>

<Fragment> ::= <Cue> (<RenameAction> | <ReframeAction>)

<RenameAction>  ::= (lock) with <Composition>

<ReframeAction> ::= <CameraAction> (to|with) <Composition>

<Composition>   ::= <FlatComposition>{, <FlatComposition>}*

<FlatComposition>::= <size>  on <Subject>[<screen>]{ and <Subject>[ <screen>]}*

<CameraAction>::= [Speed]  pan [left|right|up|down]

<Speed>   ::= slow|quick

<Subject> ::= (<Actor>|<Object>){, (<Actor>|<Object>)}*

<transition>  ::= cut|dissolve|fade in

<size>  ::= MCU|MS|MLS|FS|LS|ELS

<screen> ::= center|[(13|14)]screen(left|right)

<Action>   ::= <Move>|<Speak>|<Cross>|<Touch>

<Move>    ::= moves to <Screen>|<Place>|<Subject>

<Speak>   ::= (speaks | says <string>) [ to <Subject>]

<Cross>   ::= crosses [over|under] <Subject>

<Touch>   ::= touches <Subject>

<Place>   ::= from script

<Actor>   ::= from script

<Object>  ::= from script
```

Figure 4.13: *Simplified EBNF grammar of the prose storyboard language.*

CHAPTER

5

ACTOR DETECTION USING GENERATIVE APPEARANCE MODELS

DETECTING and naming actors is the first step towards the main goal of automatic rush generation pursued in this thesis. The task of detecting and naming actors is also important for variety of other applications, for instance it could be used for content-based indexing and retrieval of movie scenes or to support statistical analysis of film style. In this chapter we will discuss how actor and costume specific features can be learned from a small set of examples and can be used for detection in long movie or theatre scenes where the appearance of actors is consistent over time. We will also discuss the main challenges involved in the task of detecting and naming actors and compare different approaches which have been proposed over the past to tackle these challenges. We will demonstrate that using actor specific models can increase the detection recall compared to existing approaches on the basis of quantitative analysis on an entire movie and four sequences from live theatre.

5.1 PROBLEM STATEMENT

Given a video frame from a movie or a theatre recording, the goal is to localize all the actors present in the scene and to identify (name) each of them. The localization is to be done in the form of bounding boxes (rectangles) inside the image frame. (an example is shown in Figure 5.1). The localization can be done for the face, upper body or the full body. In this chapter we will be focusing on localizing upper body of the actors due to three main reasons:

- Upper body is stable with common pose changes like sitting or standing (a more common full body detector may not apply in cases like sitting postures)
- The visibility of upper body is higher (considering occlusions due to camera view; other objects; and actors). For example a full body may not be always fully visible in the frame (commonly observed in sitcoms and movies) and face may be too small to detect (as in theatre sequences).
- The costume or actor appearance is sufficiently visible in the upper body which is beneficial for detecting actors in varying viewpoints. For example detecting faces of actors looking away from the camera is difficult due to lack of features, on the other hand an upper body detector can benefit from additional information like actor clothing for localizing actors in such cases.

Challenges:

Detecting and naming actors is a extremely difficult task due to following challenges:

- **Occlusions:** The target actor may be partially of fully occluded by the background elements or other actors present on stage. The actor may disappear and re-appear from the camera view. A good detector should be able to handle partial occlusions (both due to background and other actors) and should mark the actor occluded if he is entirely occluded or if he leaves the camera view.
- **Viewpoint and pose changes:** The target actor may change viewpoint (back view, front view) and pose (sitting, standing). A good detector should be viewpoint and pose invariant.

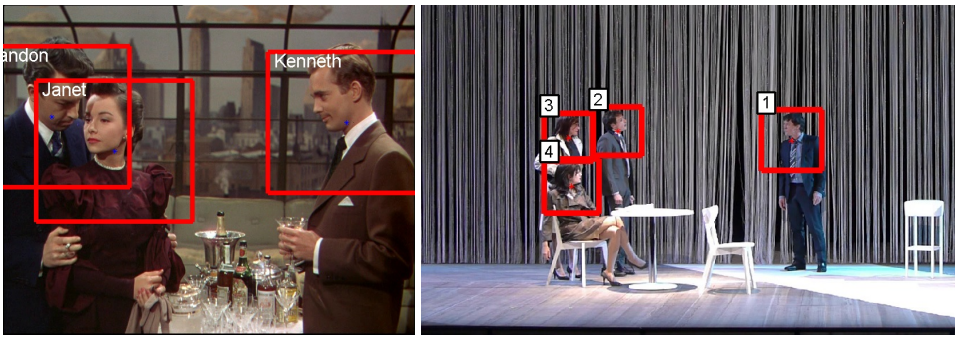


Figure 5.1: *Example of upper body detections using our method on a frame from a movie and a theatre recording. Our method provides both localization and identity information. Only a cropped part of the original theatre image is shown for visualization purposes.*

- **Scale changes:** Change of scale can occur due to camera or actor movement. For a good coverage, it is important that detection framework should support multi-scale detection. Although it may appear as an implementation detail, scale changes bring an additional degree of freedom making the detection process more vulnerable to noise.
- **Appearance changes:** The target actor may change appearance (for example clothing).
- **Illumination changes:** The actor appearance may vary with changes in illumination. A good detector should be able to deal with such illumination changes.
- **Motion blur:** Video frames may be corrupted by motion blur making the detection process difficult.
- **Distinguishing among actors:** Correctly identifying actors is another challenge when the appearances are subject to change (for example with varying illumination or poses) and are not discriminating enough.

Our approach:

We propose a complete framework to learn actor specific models using a small number of training images. We show that these models can be learned easily and can be efficiently used for the task of detection (allowing queries to detect only a specific actor). The main idea is that actor specific models can take advantage of the costume specific information to increase the detection recall (detection coverage) over generic detectors which by design discard such information. This is different from previous approaches which also use costume specific information but only to classify already obtained detections using generic detectors. Most of these methods only report recognition results on the cases where the actors are actually detected using generic detectors – not on the actual appearances of the characters (whether they are detected or not). Although generic detectors have progressed a lot in recent years the detection recall still remains lower in challenging scenarios. The proposed approach focuses on the cases where the appearance of the actors is consistent for long periods and demonstrates that using costume specific information during detection can significantly improve the detection coverage of generic detectors. Formally the contributions are following:

Contributions:

We propose a complete framework to learn view-independent actor models from a set of representative example using maximally stable color regions (MSCR) [For07] with a novel clustering algorithm. An actor's head and shoulders are represented as constellations of color blobs where the appearance of each blob is represented in a 9 dimensional space combining color, size, shape and position relative to the actor's coordinate system, together with a frequency term. Based on such a model we propose a detection framework with two stages. The first stage is a search space reduction using the k-nearest neighbours corresponding to the actor by just using the appearance of the blobs in the model. The second stage is a sliding window search for the best localization of the actor in position and scale. By repeating those two steps for all actors at all sizes, we obtain detection windows and actor names that maximize the posterior likelihood of each video frame. We compare the proposed approach for actor specific detection with generic detectors, based on an entire movie and four theatre sequences.

In the second part of the chapter we show that the view independent modeling can also be extended to rigid models based on simple color patches. We propose a complete framework for learning novel patch based, multi-layer and multi-part view independent generative models for actor specific detection. We compare the patch based rigid models with generic detectors on four theatre sequences. We also discuss the advantages and disadvantages of the two different approaches (the constellation model and the rigid model) based on the results on theatre sequences.

5.2 RELATED WORK

Much previous work on actor detection and recognition has been based on face detection. Everingham et al. [ESZ06] use scripts and subtitles to learn the association between character names and faces in television drama using a frontal face detector. Sivic et al. [SEZ09] demonstrate a much improved coverage (recall) by using profile views. In one of their examples, they report that 42% of actor appearances are frontal, 21% profile and 37% are actors facing away from the camera. Drawing on that observation, they suggest that future work on increasing coverage must go beyond the use of face detection as a first step. Indeed, generic methods for detecting upper-body or full-body actors have been proposed by Dalal et al. [DT05], Eichner and Ferrari [EF09] and Felzenswalb et al. [FGMR10], among others. While such methods can potentially increase the coverage of actor detectors by detecting actors in profile and back views, they also suffer from the higher variability of actor appearances in such views.

Extending the work in [SEZ09], Ramanan et al.[RBK07] demonstrate that color histogram of body appearance can be used as a strong cue to group detected faces into tracks. They show that given the unconstrained nature of the video, people can be tracked only about 50% of the time using such an approach. While the work in [SEZ09] and [RBK07] focuses on first obtaining tracks and later classifying or hand labeling them, our method can directly perform actor specific detections on individual frames. Closer to our approach, Sivic et al. represent people in a scene with a simple pictorial structures with 3 rectangular image regions (face, hair and torso) and use that model for finding the same people in repeated shots of the same scene [SZS06]. Starting from face detection regions, they use likely positions for the hair and torso regions and cluster them into actors. Instead, we use a training set containing front views, side views and even back views of actors, as shown in Figure 5.2 and Figure 5.3. Our actor model is

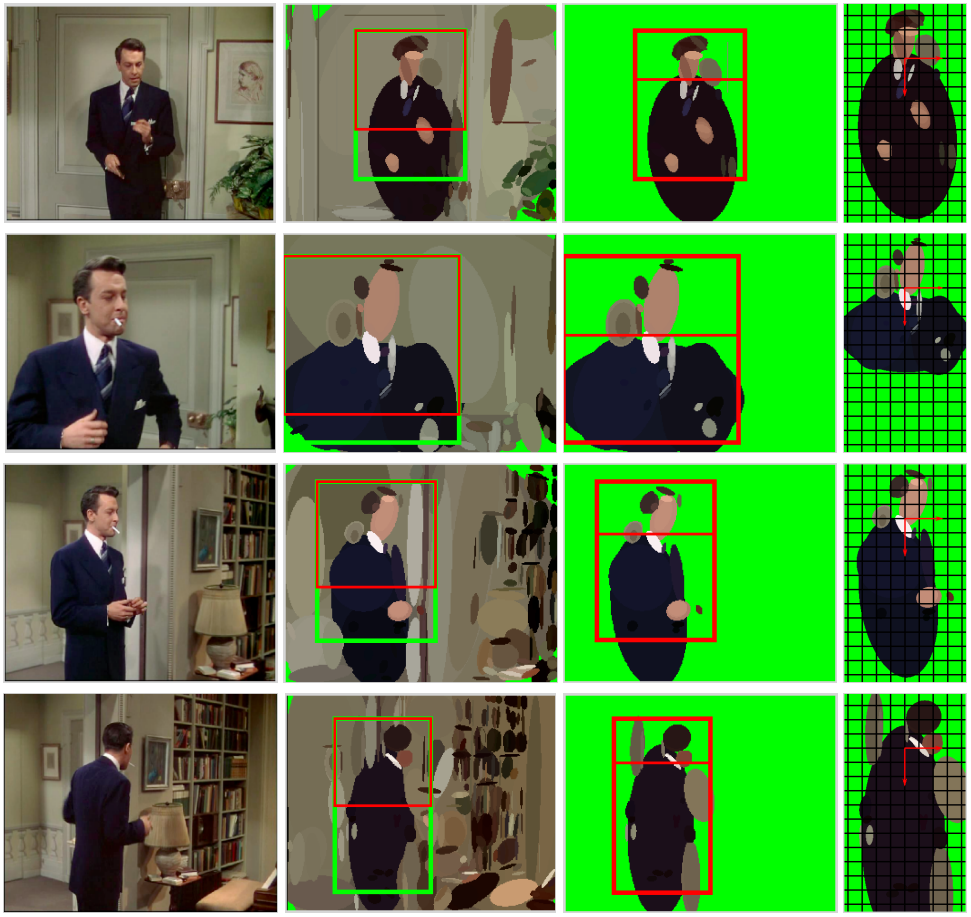


Figure 5.2: *Illustration of training process of an actor, for each row (a) Given training frame. (b) Detected upper body and the MSCR features. The red window shows the upper body detection and green window shows the extended window to include the torso, when available. (c) The blobs chosen for training shown with the head and the torso partition. (d) The color blobs scaled and shifted to the normalized actor coordinate system which is represented by the red axis.*

simpler since we only model two body parts (head and shoulders) but each part is an arbitrarily complex constellation model of color blobs.

The proposed generative model of actor's appearance is closely related to recent work in object detection and recognition[WEWP00, FPZ05]. In the classical constellation model, image features are generated in the vicinity of detected interest points and the features are clustered using k-means, which builds a visual vocabulary of "object part appearances". Appropriate feature detectors are then trained using these clusters, which can be used to obtain a set of candidate parts from images. It is difficult to apply such models to actor appearances because interest points and their image features are typically not very stable on actors, due to fast motion and complex clothing patterns, and the density of interest points and their features can be very low. We resolve this issue by representing parts of actors with color regions rather than

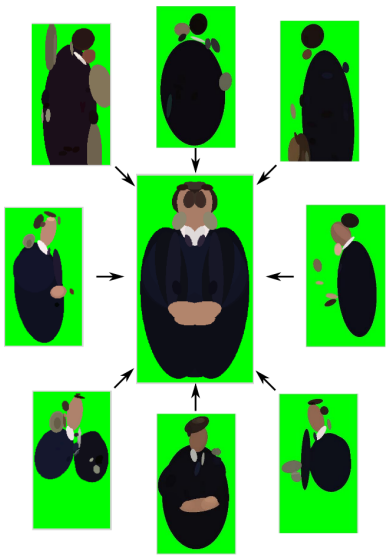


Figure 5.3: *Training images from different viewpoints are merged to obtain an actor model.*

local features. Color regions have been shown to give good results on the problem of person re-identification [FBP*10], where the goal is to identify a person given its detection window. We extend such previous work to the more difficult problem of "re-detecting" actors where the generic detectors fail due to variations in pose and viewpoints, partial occlusions, etc. This is similar to the recently-proposed Implicit Shape Model [LLS08, LLS04] which builds a star shaped structural model, where the position of each local part is only dependent on the object center. While the models in [LLS08] are view specific, we build a single view-independent model of each actor's appearance.

5.3 GENERATIVE MODEL

In this section, we introduce our generative model for the appearance of actors and describe a method for learning the model from a small number of individual keyframes or short video tracks. Our model is designed to incorporate the costume of the actor and to be robust to changes in viewpoint and pose. We make one important assumption that the actor is in an upright position and that both the head and the shoulders are visible. As a result, we model the actor with two image windows for the head and shoulders, in a normalized coordinate system with the origin at the actor's neck, and with unit size set to twice the height of the actor's eyes relative to the origin. The head region extends from $(-1, -1)$ to $(1, 0)$ and the shoulder region extends from $(-1, 0)$ to $(1, 3)$. Examples are shown in Figures 5.2 and 5.4.

More specifically, we associate with each actor a visual vocabulary of color blobs C_i described in terms of their normalized coordinates x_i, y_i, sizes s_i, colors c_i and shapes m_i, and their frequencies H_i. Color blobs above the actor's origin are labeled as "head" features and color blobs under the origin are labeled as "shoulder" features. Contrary to previous work [RST02, MjOB06, ARS09, FMJZ09], our head and shoulder models are not based on a fixed number of body parts but on an arbitrary number of salient color regions. This allows to ac-

commodate ample clothing not revealing the body parts, and to incorporate optional or even unusual costume elements such as glasses, hats and helmets, to name just a few.

Formally, our generative model for each actor consists in the following three steps:

1. Choose screen location and window size for the actor on the screen, using the detections in the previous frame as a prior.
2. Choose visible features C_i in the "head" and "shoulder" regions independently, each with a probability H_i
3. For all visible features C_i, generate color blob B_i from a Gaussian distribution with mean C_i and covariance Σ_i, then translate and scale to the chosen screen location and size

In effect our model is an AND/OR graph with one AND node and multiple OR nodes [ZCL*08]. Example of randomly generated blob images are shown in Figure 5.5. We learn the model parameters from image examples by computing maximal color regions in all examples, clustering those regions in x, y, s, c, m space and counting the frequency H_i of appearance for each cluster.We now describe each of those steps in more details.

5.3.1 Maximally stable color regions

The maximally stable color regions (MSCR) feature is a color extension of the maximally stable extremal region (MSER) feature [For07]. It is an affine covariant region detector which uses successive time steps of an agglomerative clustering of pixels to define a region. The raw moments up to order two are calculated for each detected region, which are then used to calculate the regions area, the centroid, the inertia matrix and the average color. Each region is thus represented by 9 parameters i.e. the centroid, average color and 4 raw moments representing the size and the shape of the region. An approximated ellipse is then defined over each detected region. These approximated ellipses are termed as color blobs in the later part of the text. Examples of detected MSCR features over an image are shown in Figure 5.2.

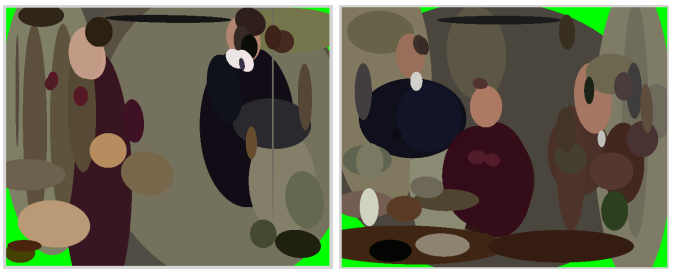
5.3.2 Clustering

We build a view-independent model of an actor's appearance by choosing a small number of front views, side views and back views for training (Figure 5.3). The actual choice of samples is not very important, as long as we cover the entire range of appearances of the actor (we try to keep the training set equally sampled across different views i.e. front, back and side views). Ideally a sequence of each actor performing a 360 degree turn would be sufficient to build such models. We manually draw the upper body bounding boxes for all training examples and label them with the actor's names.

We compute the MSCR features over all keyframes in the training set. We then collect the color blobs in all training windows, center and resize them, and assign them to actors. We cluster the blobs for all actors using a constrained agglomerative clustering. For every actor in n training images we get n set of blobs $(f_1, f_2, ..., f_n)$ with varying number of blobs in each set, where each blob is represented as a 9 dimensional vector in normalized actor coordinates. An example of this process is illustrated in Figure 5.2. Each blob in the first set f_1 is initialized as a singleton cluster. We then compute pairwise matching between those clusters and the blobs in

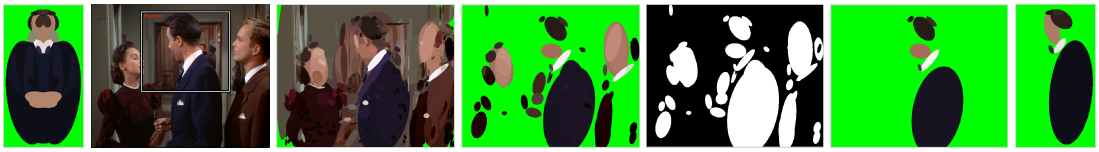
Figure 5.4: *Top two rows: Appearance models for all 8 actors in the movie "Rope" [Hit48]. Bottom two rows: Appearance models of 6 actors from two different theatre sequences. Two of the actors (Biff and Happy) are common in both sequences, the appearance model was rebuilt for the new appearances.*

the next set f_2. At each step, for each cluster, we assign at most one blob (its nearest neighbor) if it is closer than a threshold. Each cluster is represented by the mean value of its blobs. Blobs not assigned to an existing cluster are assigned to their own singleton cluster. The process is repeated for all frames and for all actors. Finally, we eliminate the remaining singleton clusters. The cardinality of a cluster is the number of its occurrences in the training set. We interpret this value as a frequency, i.e. the probability of the cluster being visible in the actor. Note that the number of clusters per actor is variable. As a result, actors with more complex appearances can be represented with a larger number of clusters. The appearance models for eight different actors are shown in Figure 5.4.



(a) (b)

Figure 5.5: *Example of two independent randomly generated blob images given the actor and background models. (a) Brandon and Janet. (b) Brandon, Janet and Kenneth*



(a) (b) (c) (d) (e) (f) (g)

Figure 5.6: *Illustration of the detection process. (a) Given Actor model. (b) Input frame and obtained detection for given Actor model with the proposed method. (c) MSCR features on the input frame. (d) Refined set of blobs for the given actor at given scale, based on appearance. (e) Binary map which can be used to reject some of the windows without processing, in practice we use separate binary maps for head and torso features. (f) The features in the image which were finally matched and considered for scoring for the final detection window as shown in (b). (g) The corresponding matched cluster centers in the given actor model.*

5.4 ACTOR DETECTION

Given the learned appearance models for a cast of actors, we now describe a framework for detecting them in any given video frame. Our framework searches for actors over a variety of scales, from foreground (larger scales) to background (smaller scales). For each actor we first perform a search space reduction using kNN-search. Then, we scan a sliding window and search for the most likely location for the actor. Then we perform a multi-actor non maxima suppression over all the detected actors. Finally, we report the best position and scale for each detected actor. The complete detection process is illustrated in Figure 5.6.

5.4.1 Search space reduction

Given an input frame first we calculate the maximally stable color regions over it as illustrated in Figure 5.6(b) and Figure 5.6(c). This gives us a initial set of blobs B over which we perform a refinement step using kNN search given the actor model and the particular scale. This pre-refinement is done only based on the color, size and shape parameters. The kNN-SEARCH(B,C^a,k), calculates k nearest neighbours in B for each cluster center in C^a by performing an exhaustive search over the Euclidean distances in the 7 dimensional space of appearance. It returns IDX and D_7, both $N \times k$ matrices, where N is the number of clusters in the given actor model. Each row in IDX contains the indices of the k closest neighbours in

Algorithm 1 Detection and localization algorithm

Given actor models(C^a, Σ^a, H^a) and the image features B.
for each actor a **do**
  for each scale s **do**
    Normalize image features w.r.t scale.
    $[IDX, D7]$ = kNN-SEARCH(B,C^a,k).
    Build inverted index i.e. for each unique blob B' in the kNN refined set, store corresponding clusters in C^a and respective distances using IDX and $D7$.
    for each position (x, y) **do**
      Find blob indices J_{head} and $J_{shoulders}$
      Compute m_{ij} using blob indices and inverted indices
      $score(x, y, s, a) = \prod_k (\sum_{j \in J_k} P(B_j, m_{ij}, a))$
    end for
  end for
end for
$[x^*(a), y^*(a), s^*(a)] = \text{argmax} \sum_a (score(x, y, s, a) - t_0)$

B corresponding to the k smallest distances in D_7.

This is further used to build inverted indices i.e. for each unique blob B' in the kNN refined set IDX, we store corresponding clusters in C^a and respective distances, ensuring that the distance is less then a threshold τ_1. This boosts the efficiency of the matching step in two ways. Firstly for each blob within the sliding window we only require to compare it with its corresponding entries in the inverted index table instead of doing an exhaustive search. Secondly, the distances in appearance are pre-calculated and distance in position D_2 can be directly added to calculate the full distance. An example of pre-refinement is illustrated in Figure 5.6(d). The figure shows the refinement for a particular scale but this is performed independently for each scale in sliding window search.

After the search space reduction, only sparse set of features remain for the matching step which leads to the speed up of the detection process in two ways, firstly many windows can be rejected without any processing using a importance term based on a binary map as shown in Figure 5.6(e) which can be efficiently implemented using integral images. Secondly, it reduces the number of matches to be performed by a large margin over a naive exhaustive matching. Notice that the number of required comparisons become independent of number of features in the image after the kNN refinement, which is done once in the beginning. Hence, the efficiency will go higher with increasing number of features. In addition to computational benefits, kNN refinement also makes the matching step much more robust by filtering out the background clutter, which is a major benefit while using low dimensional features like MSCR.

5.4.2 Sliding window search

We now proceed to define the detection scores for all actors at all positions and scales using a sliding window approach. Each actor detection score is based on the likelihood that the image in the sliding window was generated by the actor model using the previous frame detections as prior information. In practice we compute MSCR features in the best available scale and then shift and scale the blobs respectively while searching at different scales. Remember that

the model was normalized to a fixed scale during training. During recognition, we similarly normalize the size, shape and position of blobs relative to the sliding window. This ensures that all computations are performed in reduced actor coordinates. The detection procedure is illustrated in Algorithm 1.

We represent B as the set of all blobs detected in the image and C^a as the set of cluster centers in the model for a given actor a. Given a sliding window at position (x, y) and scale s, we find all blobs centered within the sliding window and assign the blobs indices J_{head} and $J_{shoulders}$. Using these indices and a matching function m_{ij}, we define the score as:

$$score(x, y, s, a) = \prod_k (\sum_{j \in J_k} P(B_j, m_{ij}, a)) \tag{5.1}$$

which is a product of parts, where each part is a sum of optional MSCR regions. Only two parts i.e. the head and the shoulder are considered in this case. The term $P(B_j, m_{ij}, a)$ is the similarity function between the model cluster C_i^a and the corresponding matched blob B_j in nine dimensional space (position, size, color and shape), which is defined as follows:

$$P(B_j, m_{ij}, a) = \sum_i H_i \cdot m_{ij} \cdot \exp\left\{-\frac{1}{2}\left(C_i^a - B_j\right)^T \Sigma_i^{a\,-1}\left(C_i^a - B_j\right)\right\} \tag{5.2}$$

where C_i^a is the center for cluster i in the actor model and Σ_i^a is its covariance matrix. A distinctive feature of our detection framework is that it requires us to find a partial assignment m_{ij} between blobs in the the sliding window and clusters in the model. More precisely, we compute m_{ij} such that each blob in the sliding window is assigned to at most one cluster, each cluster is assigned to at most one blob, and the assignment maximizes the total likelihood $\prod_k (\sum_{j \in J_k} P(B_j, m_{ij}, a))$ of the matched blobs. Although potentially more computationally intensive, we have found that this method produces significantly better results than computing the average score over all possible blob-to-cluster assignments, where the same blob may be assigned to multiple clusters, and the same cluster to multiple blobs, which is prone to detection errors. A key to our algorithm is therefore a fast approximate solution to the assignment/matching problem for evaluating the detection score benefiting from a pre-refinement step.

We further benefit from the previous frame detections D_{t-1} to modify the scores as follows:

$$score(x, y, s, a) = \begin{cases} l_{1,1} \cdot score(x, y, s, a) \cdot pr_{t,t-1} & \text{if } a \in D_{t-1} \\ l_{0,1} \cdot score(x, y, s, a) & otherwise. \end{cases} \tag{5.3}$$

where, $l_{1,1}$ and $l_{1,0}$ measure the probability that the same actor is observed in consecutive frames. When the actor is not present in the previous frame, all positions in next frame are equally probable. When the actor is present in both frames, we assume the new position to be close to the previous position, within some covariance term Σ_{pos}. Empirically, we have found that the terms $l_{1,1}$, $l_{1,0}$ and Σ_{pos} are in fact independent on the choice of actor or movie. The term $pr_{t,t-1}$ is defined using the covariance term as follows:

$$pr_{t,t-1} = \exp\left\{-(p_t - p_{t-1})' \Sigma_{pos}^{-1} (p_t - p_{t-1})\right\} \tag{5.4}$$

The above procedure is repeated for all actors individually and after non maximal suppression over $score(x, y, s, a)$ we get the potential detections for each actor. Benefiting from the fact that an actor can only appear once on a frame, we then search for the best possible positions $[x^*(a), y^*(a), s^*(a)]$ which maximizes the total score over all actors.

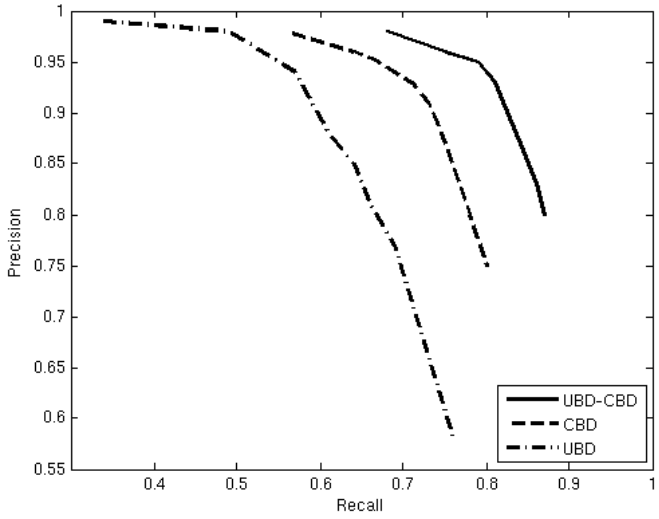


Figure 5.7: *Comparison of results on recall and precision for actor detection using Upper body detector(UBD), Color Blob detector(CBD) and Combined method (UBD-CBD)*

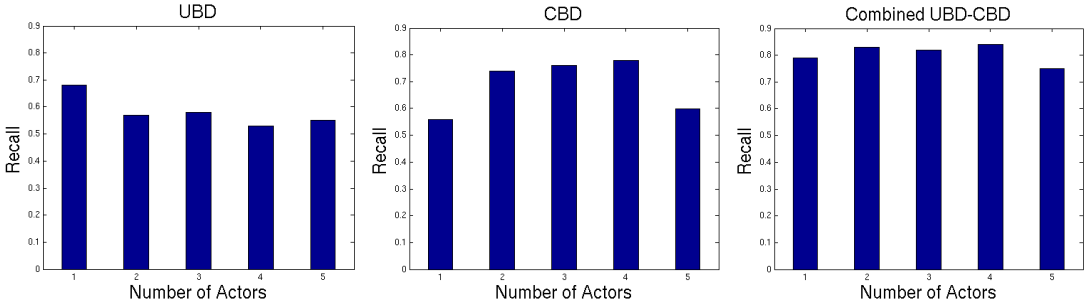


Figure 5.8: *Comparison of recall with increasing number of actors in UBD, CBD and combined cases*

$$[x^*(a), y^*(a), s^*(a)] = \text{argmax} \sum_a (score(x, y, s, a) - t_0) \tag{5.5}$$

A threshold t_0 is used to reject all the detections with score below a threshold value.

5.5 EXPERIMENTS

5.5.1 Dataset

As mentioned earlier the previous work in actor specific models have focused on obtaining tracks and then performing classification on the obtained tracks. And in datasets related to the re-identification problem, the bounding boxes are provided and the goal is only to identify the pedestrians. Our method on the other hand can perform direct detection using actor specific models, even on keyframes which is a much harder task than classifying tracks and we target the scenario where it is difficult to obtain face or upper body tracks.

Due to this reason, for a detailed evaluation of our method we propose the ROPE dataset[1] which is set of keyframes at equal intervals, a frame every 10 seconds from the movie "Rope" [Hit48] by Alfred Hitchcock. This dataset presents significant scale and viewpoint variations for each actor with presence of motion and focus blur. There is significant camera movement and constant background variations from flat regions to cluttered patterns. The lighting changes considerably during the movie and the clothing appearance of all the actors remains consistent which makes it suitable for our experiments. The movie is composed of a single shot and choosing a frame every 10 seconds gives about 443 frames. There are 8 different actors in the entire movie (except the initial victim and Alfred Hitchcock himself). All of them were considered in our experiments. The number of appearances per actor vary between 38 and 275. All 443 frames were hand labeled with the names and screen locations for all actors to serve as ground truth.

We also perform evaluation on four theatre sequences, two sequences have two actors and other two have four actors present on stage. The sequences were captured from a static camera and present variations like fast motion, varying poses, varying viewpoint, frequent actor to actor occlusions, entry and exit of actors from stage and varying illumination (directional lighting leading to uneven distribution of light on stage). The ground truth with actor location and names was created by manual labeling similar to ROPE dataset. The detection results for theatre sequences were calculated without any temporal coherence and scores were calculated independently for each frame i.e. the step in Equation 5.3 was skipped and results from Equation 5.1 were directly used in Equation 5.5.

5.5.2 Results

This section describes the results on the proposed dataset. We ran our detection and recognition algorithm using the built actor models, on all 443 frames and compared the results with the ground truth. Figure 5.7 shows the results on recall and precision of actor detection using the proposed method (CBD) and it is compared with the state of the art Upper Body Detector (UBD)[2] and the combined case where we merged the detections obtained from both the methods individually. Results demonstrate an increase in recall from about 57 percent in UBD to 70 percent in proposed Color blob detector (CBD) to about 81 percent in combined approach for a similar precision. Thus, the coverage is significantly improved keeping the same false positive rate. In Figure 5.8 we plot recall rates for both UBD and CBD and combined case, with different number of actors present in the frame and it shows that the proposed method gives consistent results with varying number of actors.

Recognition results on the detected actors are presented in Table 5.1. As can be seen, our method not only increases the average recall rate for all actors, but also correctly names all actors with an average precision of 89 percent despite the large number of back views and partial occlusions.

We made similar evaluations on four theatre sequences. We also added the comparisons with full body detector (FBD) [FGMR10] in this case as the actors are entirely covered by the camera field of view in all the sequences (making the comparison relevant). The recall rates obtained using FBD, UBD and CBD are given in Table 5.2. The color blob detector (CBD)

[1] http://imagine.inrialpes.fr/people/vgandhi/CVPR_2013/
[2] http://www.vision.ee.ethz.ch/~calvin/calvin_upperbody_detector/

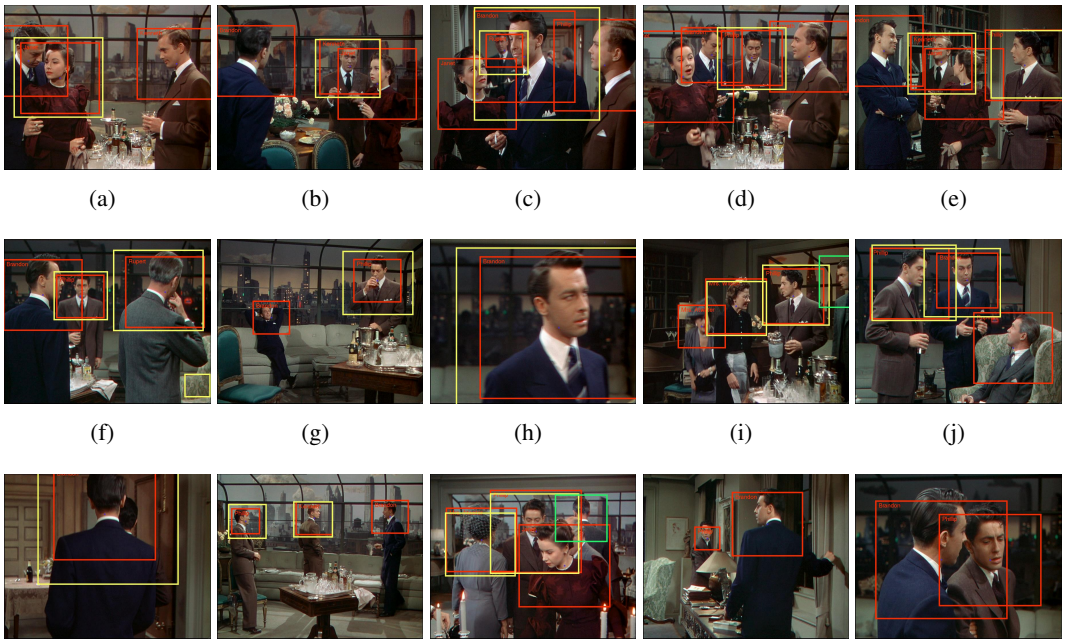


Figure 5.9: *Some detection results from Rope dataset using proposed method CBD (in red) with recognized actor names on top left, UBD (in yellow) and not detected by both (in green). Notice how our method is able to detect and identify the actors in the presence of multiple actors with partial occlusions [e.g. (a), (d), (e), (o)], varying viewpoint and posture [e.g. (f), (m), (g), (j)], varying illumination [e.g. (b), (l)], motion blur [e.g. (h)] etc.*

	A1	A2	A3	A4	A5	A6	A7	A8
A1	99	0	0	0	1	0	0	0
A2	0	78	0	16	0	2	3	1
A3	0	1	96	0	3	0	0	0
A4	0	25	0	74	0	0	1	0
A5	0	0	5	0	95	0	0	0
A6	0	4	0	0	0	93	0	3
A7	3	4	0	0	1	2	81	9
A8	0	0	0	0	0	6	22	72

Table 5.1: *Actor identification results for all 8 actors in Rope dataset (percentages).*

gave superior results in each of the sequences. Also, interestingly the UBD performed better than FBD on all the four sequences.

Some of the example detections results are shown in Figure 5.9 and Figure 5.10, they demonstrate how our method performs well even with severe cases of occlusions, viewpoint scale and pose variations in a multi actor scenario. Figure 5.11 shows some failure cases. Most failure cases are caused by insufficient illumination or severe occlusions. In 5.11(a) the blobs in the torso region of the undetected actor gets merged with the background, heavy blur causes mis-detection in 5.11(b), torso is largely occluded in 5.11(c). In fourth instance 5.11(d), the

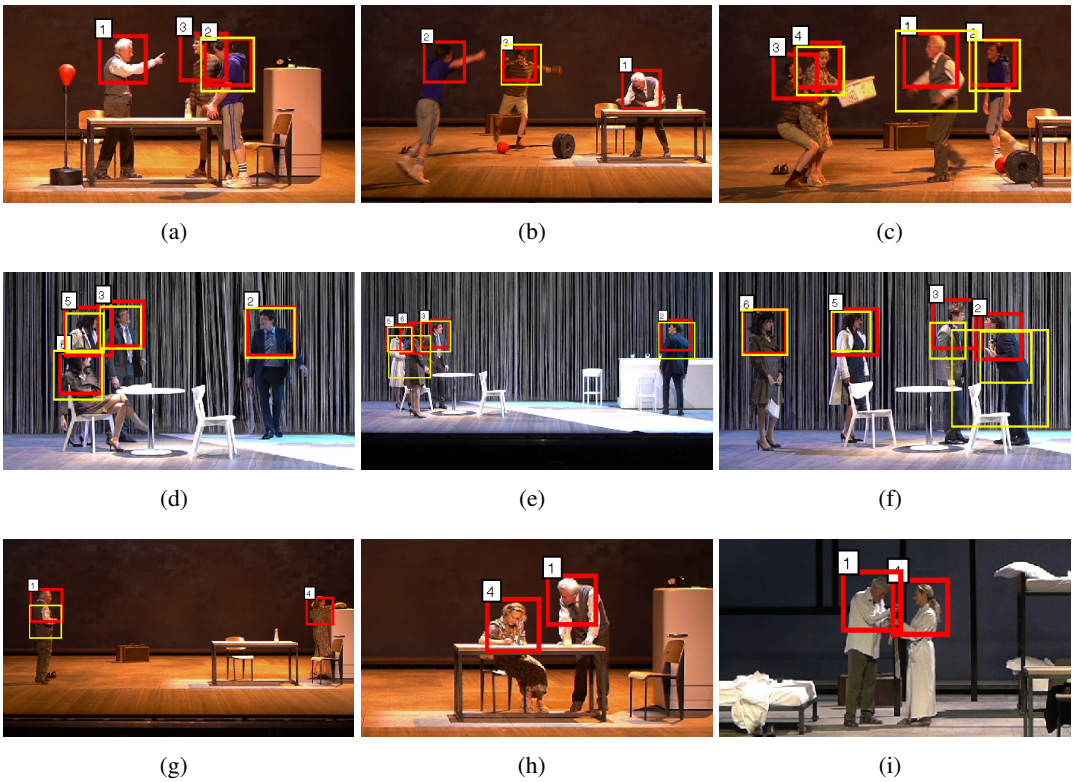


Figure 5.10: *Example results on theatre dataset using proposed method CBD (in red) with recognized actor identity on top left and UBD (in yellow). The images demonstrate that how the proposed method works extremely well even with large actor to actor occlusions [e.g. (a), (d), (e)] and varying poses [e.g. (b), (d), (h)].*

	FBD	UBD	CBD
Dos1 (4 actors, 1788 frames)	48	60	**72**
Dos2 (2 actors, 1490 frames)	76	80	**93**
Dos4 (4 actors, 1656 frames)	55	66	**84**
Dos5 (2 actors, 1094 frames)	43	54	**81**

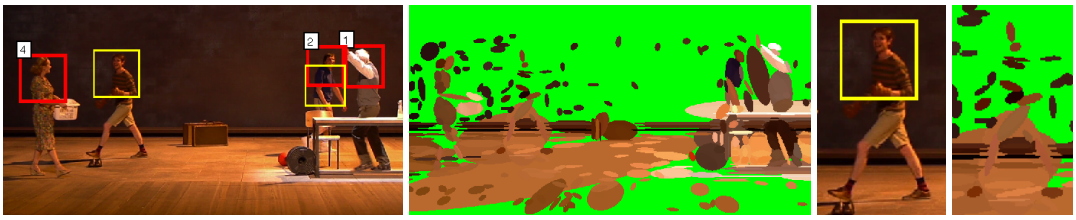
Table 5.2: *Detection recall (percentage) given a fixed precision rate (80 precent) on four theatre sequences. The number of actors and number of frames in each of the sequence is given in the first column.*

hand gets detected as the head and blobs below as torso, leading to a false detection. Note that there is very little temporal coherence between frames in our dataset. As a result, we cannot rely on context to resolve such problems. Detecting actors at a finer temporal scale (every second or every frame) may help alleviate the problem but that requires a stronger model of temporal coherence, taking into account the inter-frame motion of actors, the temporal coherence of the blobs, and the probability of visual events such as entrances and exits. This is left for future work.

(a) (b) (c) (d) (e)

Figure 5.11: *Few examples where our method wrongly detects or fails to detect the actors, the typical failure cases are due to shadows [e.g. (e)], merging of foreground blobs with the background due to low illumination [e.g. (a)] or heavy blur [e.g. (b)], confusing the hands as the head [e.g. (d)] and large occlusions [e.g. (c)]. This also demonstrates the difficulty of the proposed dataset.*

(a) (b) (c) (d)



(e) (f) (g) (h)

Figure 5.12: *A particular failure case where MSCR features are not consistent. First column shows the detection results (CBD in red and UBD in yellow). Second column shows the MSCR features (the large background blobs were filtered for visualization purposes). Third column shows the cropped image of the actor 'Happy' which was not detected by CBD. Fourth column shows the blobs in the image region of the undetected actor 'Happy', we can observe that t-shirt region is difficult to model using MSCR features (this is possibly due to repetitive stripes and the shadows.)*

In few cases the detection failure occurs due to lack of available MSCR features (this is illustrated with two examples in Figure 5.12). This happens in the particular cases with repetitive texture (as the striped t-shirt of actor in Figure 5.12(c)). Since the failure occurs due to unavailability of the features, the detector even fails to provide weak detection scores at the ground truth location in such cases. Hence, it may not be possible to resolve this problem even with a stronger model of temporal coherence. To deal with such cases we propose a simplified grid based appearance model which is described in the proceeding section.

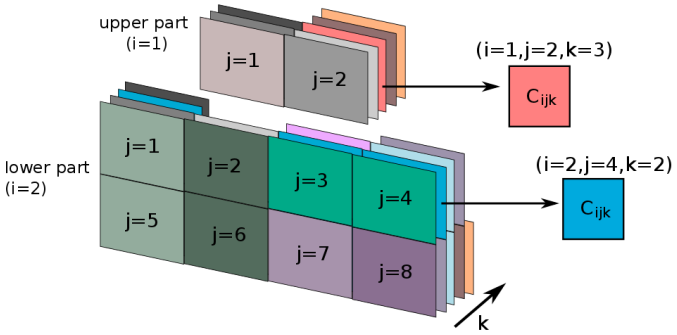


Figure 5.13: *Structure of our appearance model. The model has 2 parts and each part consists of a fixed grid. Each position on the grid stores a list of clusters. Each square block in this figure represents a cluster. Clusters (C) can be uniquely indexed by a part (i), position (j) and depth (k). The number of clusters (k) may vary with position (j).*

5.6 RIGID COLOR PATCH BASED APPEARANCE MODELS

We introduce modified appearance models well suited for both textured and non textured scenarios. Instead of MSCR features, the new appearance model is based on color patches on a fixed grid. Similar to previous approach the new generative appearance model (Section 5.6.1) combines information from a sampling of representative examples and are view independent. The model (illustrated in Figure 5.13) consists of a fixed set of parts, e.g. upper and lower. Each part consists of a fixed grid. At each position on the grid, the model stores a list of clusters of the corresponding patches from the training images. These clusters are represented by their mean color and a weight. Note that we still base our appearance model on color, rather than on texture, as many objects of interest (such as an actor's clothing) are stable in color, but not in textural details.

The probability of a candidate window containing the object is the product of probabilities of the occurrence of the parts, where the probability of each part is the sum over its positions. For each position, the probability of the corresponding observed image patch is based on the best matched cluster. The appearance model is constructed from the set of representative images, where each training image is first normalized to align with a fixed grid and then partitioned into fixed size patches. The patches at each position are then clustered as illustrated in Figure 5.14, yielding a single model combining all training examples (the model building procedure is detailed in Section 5.6.2). The initial constructed model is then updated based on the same training images to alleviate the background noise (as the training images are annotated using a simple rectangle and no segmentation information is used, significant background noise may appear in the actor models). The update procedure is explained in Section 5.6.3.

Once the model is updated, the detection is performed by sliding a window at multiple scales for each frame (described in Section 5.6.4). For each window, we estimate a probabilistic score of generating the object using the given appearance model. The detection is performed for each frame and each actor independently.

5.6.1 Generative Model

The appearance model described in this section is illustrated in Figure 5.13. The model is a set of parts (e.g. the upper and the lower parts). Each part is a grid of positions indexed by j (in the figure upper part has two while the lower one has eight). Each position has a list of clusters, represented by tuples (μ, γ) where μ is the mean color value in Lab color space and γ is a weight factor. The k^{th} cluster of part i in position j is indicated by $C_{ijk} = (\mu_{ijk}, \gamma_{ijk})$. A set of pixels in a patch could take any of the values from the underlying cluster and various combinations of the values give different appearances it could model. We wish to learn all the different color-weight tuples such that their mixture generates all the different viewpoints and poses of the object we want to model.

The probability that the observed grid of color patches B (aligned to model configuration and k=1) was generated from the given model C is a product over parts (i) :

$$P = \prod_i P_{part_i} \,, \tag{5.6}$$

where the probability of each part is the sum over all positions (j) in that part:

$$P_{part_i} = \sum_j P_{patch}(B_{ij}) \,. \tag{5.7}$$

The probability for each patch is defined by the best matched cluster at the given position among the k possible choices.

$$\begin{aligned} P_{patch}(B_{ij}) &= \gamma_{ijk^*} \mathcal{N}(C_{ijk^*}, B_{ij}) \,, \\ \text{where } k^* &= \underset{k}{\operatorname{argmax}} \mathcal{N}(C_{ijk}, B_{ij}) \,, \\ \text{and } \mathcal{N}(C_{ijk}, B_{ij}) &= \exp(-|\mu_{ijk} - B_{ij}|^2). \end{aligned} \tag{5.8}$$

For each position only the best matched cluster is considered for the scoring. B_{ij} is the mean color of the patch at a given part (i) and position (j). The index k^* represents the best matched cluster among the k possibilities at a given part (i) and position (j). The term $|\mu_{ijk} - B_{ij}|$ is the Euclidean distance in Lab color space. The proposed model like CBD is also related to AND/OR graphs [ZCL*08], taking products (AND) over mandatory parts and taking sums (OR) over optional positions per part. More specifically, it is a AND/OR graph with one AND node and multiple OR nodes.

5.6.2 Initial Model Construction

The appearance model is built from a set of representative images of the object. In this case we typically choose 8 examples uniformly sampled across all viewpoints. We choose 8 examples as standard practice because it broadly covers an entire rotation in front of the camera (front, 3/4frontright, right, 3/4backright, back, 3/4backleft, left, 3/4frontleft). Instead of rectangles we annotate each training image by drawing a single line. In the case of actors, the line is centered at the neck and extending to the top of the head (Figure 5.14(a)). We have found this form of

(a) (b) (c)

Figure 5.14: *(a) Manual annotation is done by drawing a line centered at the neck and extending to the top of the head. This line is then used to estimate the bounding box. (b) Each training image is normalized in size, segmented into blocks on a fixed size grid, and then merged into a single multi-view appearance model. (c) Our model consists of two different parts, i.e., the upper part and the lower part. Each part has a fixed grid of positions, each with a list of clusters of image blocks. Only the most frequent cluster at each position is shown, except for the position highlighted in green.*

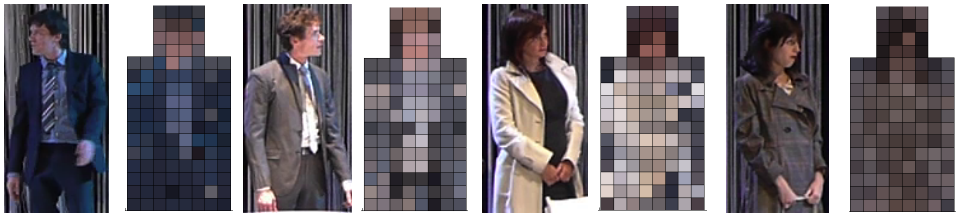
annotation to be fast and accurate in locating parts.

The initial construction procedure is illustrated in Figure 5.14(b). Given the set of training images we normalize each image to same scale and size. Each training image is individually partitioned into fixed size patches and each block is represented by the mean Lab color value $\mu = (L, a, b)$ of the pixels within it. The patches are merged by constrained agglomerative clustering, similar to the one used in CBD (Section 5.3.2), however the clustering is applied at each position independently in this case. The algorithm works in a stepwise manner where all patches in the first training image are initialized as singleton clusters. With each next incoming training image, a patch to patch comparison is made and if the feature distance (Euclidean Lab color difference) is less than a threshold (θ) the patches are merged and the weight is incremented otherwise a new singleton cluster is initialized with a weight of 1. This way each position on the grid is represented by multiple clusters with varying weights. In the end the weights are normalized by number of training images. Each cluster is represented by the mean color value (μ) and its weight (γ). The resulting model is a multi layer structure, where number of layers may differ for each position in the grid as shown in Figure 5.13.

5.6.3 Update

The initial appearance model has a list of clusters for each model position and their corresponding weights. Our method tunes these cluster weights by performing detection over the same training images. This step reinforces the clusters in under represented view-points and potentially reduces the background clutter in the model. For each detection only a set of clusters from the appearance model are considered for scoring. Let C_m be the matched set of clusters at the MAP location (maximum a posteriori i.e. the highest scoring location) in the given training image and C_g be the matched set of clusters at the ground truth location. If the ground truth

(h) Willy (i) Biff (j) Happy (k) Linda



(h) Biff (i) Happy (j) Miss Forsyth (k) Girl

Figure 5.15: *Patch based appearance models of 6 actors from two different theatre sequences. Two of the actors (Biff and Happy) are common in both sequences, the appearance model was rebuilt for the new appearances. Only the highest weighted patch of each position on the grid is shown for visualization purposes.*

location is the same as the MAP location, no update is performed. But if the two locations are different, we update the cluster weights as follows:

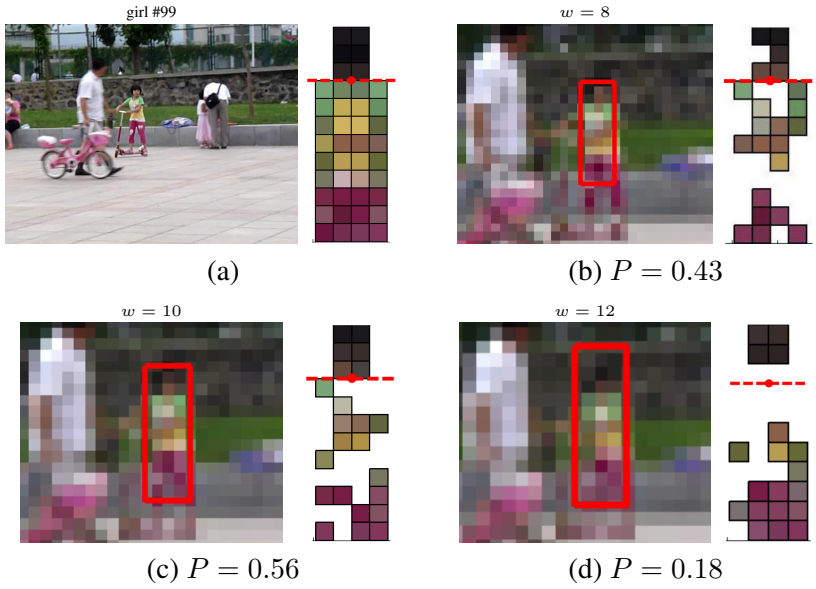
$$\gamma_{ijk} = \gamma_{ijk} + \begin{cases} -\lambda_0 & \text{if } [i,j,k] \text{ in } C_m \text{ and not in } C_g \ , \\ +\lambda_0 & \text{if } [i,j,k] \text{ in } C_g \text{ and not in } C_m \ . \end{cases}$$

The parameter λ_0 is a positive constant. The weights of the clusters are kept positive and the weight are only reduced if the current weight of the clusters is greater than λ_0. The basic idea is to penalize the clusters which were matched in the MAP location but not in the ground truth and to reinforce the clusters which were matched at the ground truth but not at the MAP location if the MAP location and ground truth locations are not the same. This update is iteratively performed for each training image until convergence *i.e.,* the MAP location in each training image is same as the ground truth location. The background clutter is significant in the patch based approach as we are not using any segmentation information while annotating the training images (this was not the case in CBD where the background blobs were easily filtered) and the update considerably improves the detection performance. Updated appearance models of six actors from two different sequences are illustrated in Figure 5.15.

5.6.4 Detection

After the model is updated the detection is performed for each object/actor independently in each individual frame. This process is illustrated in Figure 5.16.

Given an input image as in Figure 5.16(a), it is first partitioned into patches as shown in Figure 5.16(b). The appearance model is normalized to corresponding patch size and a sliding window operation is performed. Detection is performed at multiple scales by partitioning the input image into different size patches. For each given detection window with center (x, y) and scale (w), we calculate the detection score using Equation 5.6. For each patch in the



(a) (b) $P = 0.43$

(c) $P = 0.56$ (d) $P = 0.18$

Figure 5.16: *Detection results at different scales and corresponding detection scores. (a) Given as input are the test image and the appearance model. The figure for appearance model is showing only the most frequent cluster for each position. The red dotted line shows the part configuration. (b) The input is image partitioned into 8 by 8 blocks (image was cropped for visualization purposes). The red window displays the highest scoring detection at this resolution. On the right is the figure showing the clusters from the appearance model which matched at the shown detection window. (c,d) Detection results at different scales.*

given sliding window the best matching cluster is considered for the scoring function only if the feature distance is below the threshold (θ). Figure 5.16(b,c,d) shows the image being partitioned at different resolutions. At each scale, the highest scoring window is shown with a red rectangle. For each of these detections, the matched set of clusters from the appearance model are also shown. The detection score of top scoring windows at three different scales are also given below the images.

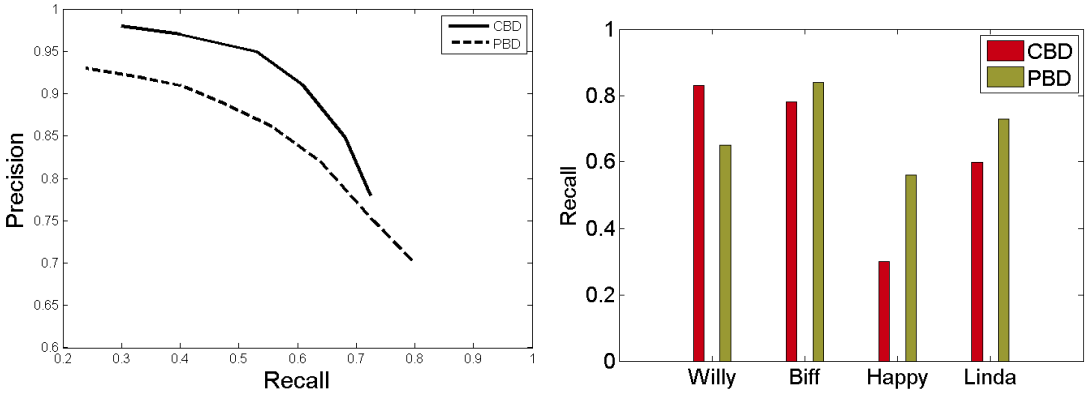
5.7 COMPARISON RESULTS WITH CBD

We compare the detection results using the rigid patch based detector (PBD) with the color blob detector (CBD) on four theatre sequences. For a fair comparison we use the same set of training images for learning the appearance model in each case. The results are illustrated in Table 5.3. The recall rates of upper body detector (UBD) and full body detector (FBD) are also mentioned in the table for reference.

The detection with CBD is more precise compared to PBD and gives fewer false positives. The benefit with PBD are mainly in two ways, first it generalizes well in both textured and non-textured cases and second it improves the overall recall rates (although at the cost of lower precision). The results on the sequence *Dos1* are shown in Figure 5.17. We can observe that the recall rate for 'Happy' significantly improves using PBD. We can also observe in Figure 5.17(a) that the PBD tends to improve the recall as the precision decreases while CBD tends to saturate quickly (this may be because some of the failures of CBD are due to lack of features and are

	FBD	UBD	CBD	PBD
Dos1 (4 actors, 1788 frames)	48	60	**72**	70
Dos2 (2 actors, 1490 frames)	76	80	**93**	86
Dos4 (4 actors, 1656 frames)	55	66	**84**	76
Dos5 (2 actors, 1094 frames)	43	54	81	**83**

Table 5.3: *Detection recall (percentage) given a fixed precision rate (80 precent) on four theatre sequences. The number of actors and number of frames in each of the sequence is given in the first column.*



(a) (b)

Figure 5.17: *Recall rates for Dos1 sequence. (a) Comparison of results for recall and precision of color blob detector (CBD) with the patch based detector (PBD). (b) Recall rates for each actor independently given the same precision rates.*

not recoverable). Detection results using PBD on some example theatre frames are shown in Figure 5.18. We can observe that the actor Happy is well detected in the frames 5.18(b) and 5.18(c) where it was missed by CBD.

5.8 SUMMARY

We have presented generative appearance models for detecting and naming actors in movies that can be learned from a small number of training examples. We have shown that low dimensional features like MSCR that were previously used for actor re-identification can also support actor detection, even in difficult multiple actors scenarios. Results show significant increase in coverage (recall) for actor detection maintaining high precision. To our knowledge, this is the first time that a generative appearance model is demonstrated on the task of detecting and recognizing actors from arbitrary viewpoints.

We have also presented a simpler patch based detector (PBD) for the scenarios where the MSCR features does not work well (for example, in cases with repetitive textures). The appearance models in PBD utilizes novel layer based rigid structure. Although, this form of modeling is less flexible than the CBD approach, it works well in practice and generalizes well in both textured and non textured scenarios. The PBD gives better results over the generic detectors

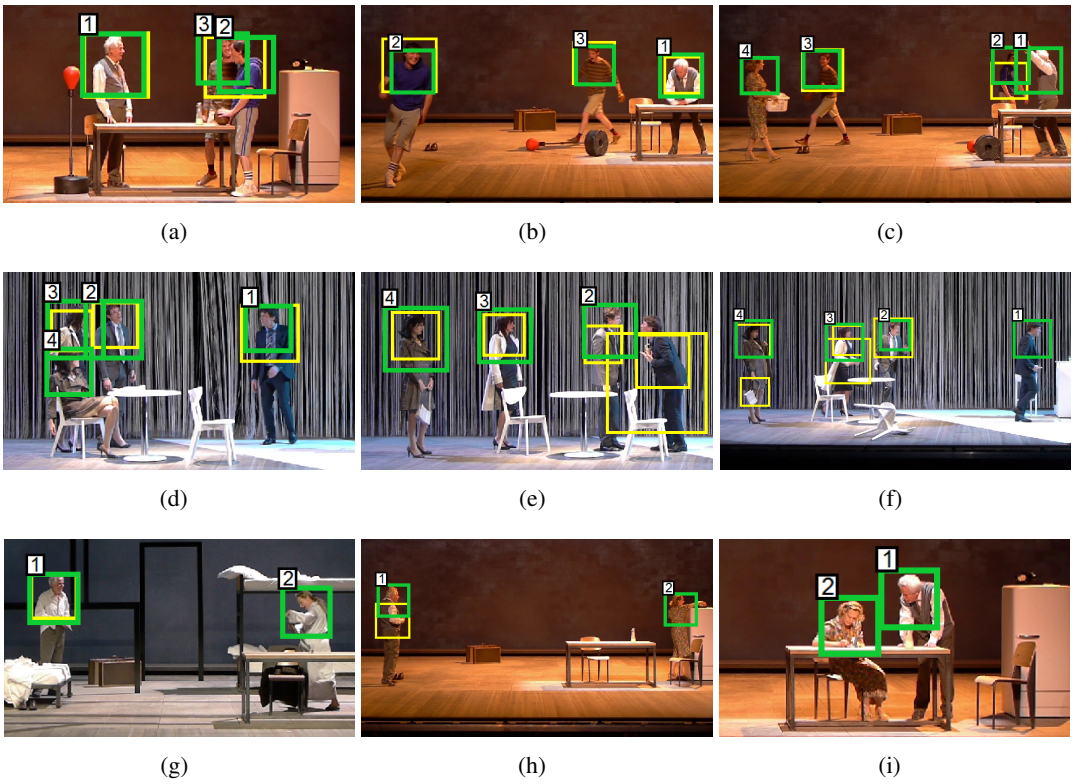


Figure 5.18: *Example results on theatre dataset using PBD (in green) and UBD (in yellow).*

but the results are less precise compared to CBD. Although it gives more false positives, it increases the overall recall over CBD.

The improved recall makes the proposed approach a good candidate for tracking actors/objects multiple actors constantly changing viewpoints and occluding each other in long video sequences such as "Rope" and recordings from live performances. In the next chapter we will show that the offline tracking using actor specific appearance models can significantly improve the tracking performance over the current state of the art.

One obvious limitation of our method is that it only handles cases where the appearance of actors does not change much over time. Future work should investigate extensions with mutually-exclusive appearances per actor, so that actors can change their appearances and costumes over time. Because each appearance requires so few training examples to be learned, we believe this extension is in fact possible. Another interesting avenue of research could be to perform weakly supervised training by extracting actor labels from temporally aligned movie scripts [RTT03, CJMT08].

CHAPTER

6

OFFLINE ACTOR TRACKING

ACTOR tracking is the problem of identifying and localizing a given target actor in a video stream over time. Tracking an actor/object over a long period of time is a challenging problem because of issues such as camera motion, full or partial occlusions, pose changes, and changes of object appearance. In this chapter, we present an approach to the long-term tracking problem that uses an offline approach that enables it to better address these challenges. The method is initialized with a small set of representative images that identify the target object, and performs tracking by considering the entire video sequence simultaneously. The proposed method is based on the learning and detection framework discussed in the previous chapter. In this chapter we demonstrate that individual candidate detections at each frame can be combined into smooth trajectories by minimizing a cost function accounting for false detections and long term occlusions. We compare our approach with nine state-of-the-art tracking methods on eleven long and challenging video sequences, demonstrating superior performance in all cases. Figure 1 shows a motivating example where our method succeeds in a challenging scenario.

6.1 PROBLEM STATEMENT

Consider a video stream depicting various actors/objects moving in and out of the camera field of view. Given a set of representative images of each actor/object present in the video, our goal is to automatically determine its bounding box in every frame of the video or to indicate that the it is not visible. Our approach targets the scenarios where the actor's appearance (for example clothing) is constant for long periods, like in theatre scenes. We refer to this task as long term actor/object specific tracking.

The tracking problem has been thoroughly investigated in the past. Most existing tracking methods work online, that is, they estimate the target state without considering future information. They are usually fast, return smooth trajectories, and require only simple initialization. However, existing methods fail to consistently track objects over long periods [YJS06]. Online algorithms usually drift away from the desired target either due to the accumulation of error during run time or due to full or partial occlusions and viewpoint changes. Post-failure behavior is not directly addressed in most of these methods, requiring an interactive setting for re-initialization upon failure. While online algorithms are required for real-time applications, such as surveillance and human computer interfaces, other applications (like rush generation) can benefit from an offline approach, if it provides better performance.

Figure 6.1: *Illustration of actor tracking in a long theatre sequence with significant actor motions, view-point variations, pose variations and occlusions. Actor specific models were used to independently compute trajectories of all actors. Such appearance models gives strong advantage over generic detectors in resolving multiple actor tracks in presence of frequent actor to actor occlusions.*

Our approach:

Our approach to long-term tracking is based on two key ideas. First, we model long term tracking in an offline manner. This gives a global perspective of the tracks over time, addressing issues such as drift and recovery from occlusions that are common in online algorithms. Second, our approach is initialized with a small set of representative training samples rather than a single initial position. This helps build a stronger appearance model which in turn significantly improves the recall and makes the tracker much more robust to view-point changes, pose variations and occlusions. The representative images may or may not be from the video to be tracked since we do not store the positional information. Ideally a 360 degree rotation of the object in front of the camera is sufficient for training. The use of multiple examples is similar in spirit to the work in [BJ98] which demonstrated that it is practical to represent an object using a small set of canonical views. Building on these ideas, we proposes a novel method for offline tracking making following contributions.

Contributions:

We propose a three stage algorithm for offline tracking specific to scenarios where a representative set of images for the target object are available prior to tracking. The three stages are namely learning, detection and optimization. The learning step combines the set of representative examples into a single multi-part view-independent appearance model. The built appearance model in our approach is static and not updated after the learning step. The detection step finds candidate detection in each frame of the video independently for each actor/object and each frame. The optimization step combines individual detections in each frame into smooth trajectories by minimizing a cost function over the entire video. For the first two stages we directly use the framework proposed in the previous chapter, specifically the patch based detector (PBD). The main novelty in this chapter is the optimization step.

We show that in many scenarios actor specific models can be built easily and can outperform tracking approaches trying to adapt the appearance model during the tracking process, which is a difficult task. We demonstrate that even using simple color features in the proposed multi-part object model (PBD) provide significantly better results than the state of the art. We provide a thorough quantitative and qualitative analysis over long and challenging video sequences.

6.2 RELATED WORK

The long term tracking problem can be approached from either a tracking or a detection perspective. Most recent methods with a tracking perspective [JFEM03, RLLY08, Avi05, BYB09, KL10, CLL05] have focused on designing better appearance models that are updated over time. These updates are necessary to handle significant changes in appearance and an efficient appearance model is of prime importance for the success of the tracker.

These tracking algorithms can be learnt either generatively [JFEM03, RLLY08, KL10] or discriminatively [Avi05, BYB09, CLL05, WLYY11]. Generative trackers usually model only the appearance of the object while the discriminative trackers also model the environment where the object moves. These methods perform adaptive model update using the close neighborhood of the current position as positive samples and distant surroundings as negative

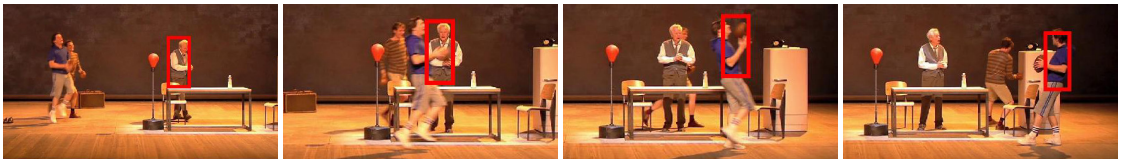


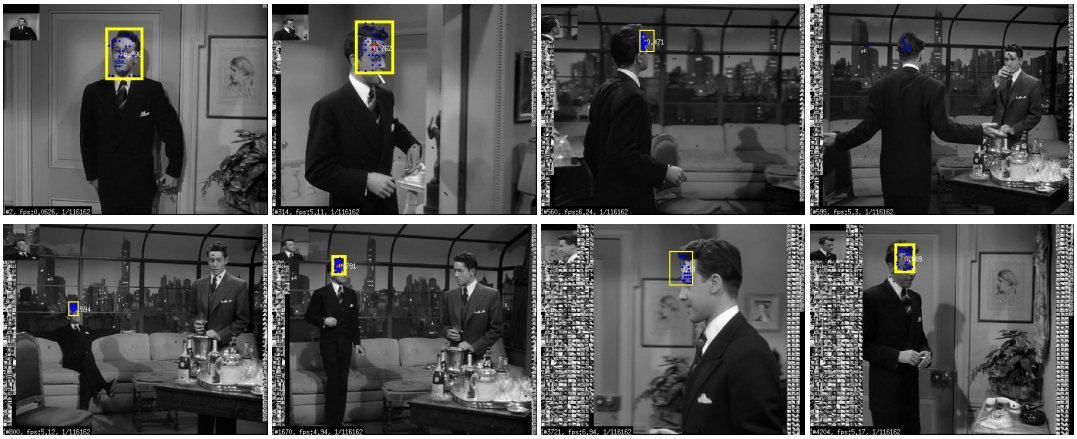
Figure 6.2: *Online adaptive trackers may drift away from the target. An example using the MILTrack[BYB09] tracker is shown in this figure. With frequent object to object occlusions, it becomes difficult to choose when to adapt and when not to adapt. Most online trackers like MILTrack[BYB09], IVT[RLLY08] or VTD [KL10] do not allow for automatic recovery once the target is lost and it is required to restart the tracker.*

samples. However, most adaptive tracking approaches suffer from the problem of drift (illustrated in Figure 6.2) and require manual re-initialization over failure.

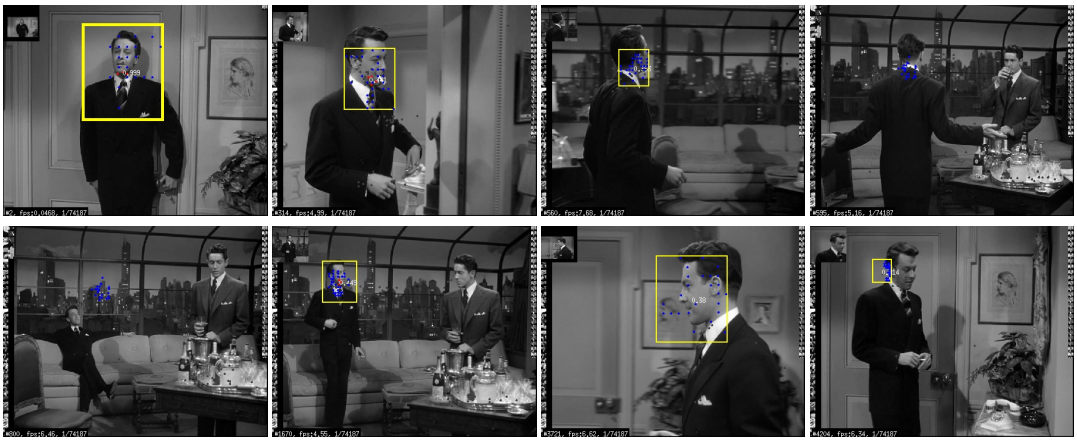
An alternative direction of work is tracking by detection that estimates the object location at each frame independently and then combines the available detections into object tracks using features across detections. This approach has been used in actor tracking [ESZ06]. Detectors by themselves do not have problems like drift and can recover if the object exits and re-enters the camera view. But most common generic detectors [FGMR10, DT05] are viewpoint dependent and do not take into account the object specific features which are often necessary to get sufficient recall for tracking as highlighted in some of the recent work [GR13, TBS12]. Training such detectors usually requires large number of training examples and cannot be quickly extended to unknown objects. Moreover, feature tracks used for combining these detections into a smooth track disappear or drift in presence of fast motion which leads into inaccurate localization at intermediate frames.

Tracking-by-detection approaches have also been studied for multi-target tracking [HWN08, BRL*09]. These approaches aim to automatically detect and track a variable number of targets in complex and crowded environments like pedestrians on a street. As many targets in such scenarios may have similar appearances, the main challenge addressed in these methods is to associate detections with multiple tracks simultaneously. The time span of individual tracks in these scenarios is generally short (typically 5-10 seconds). On the other hand, we focus on tracking in scenarios like theatre stage or movies where actors have consistent appearance for long intervals (several minutes). Benefiting from actor specific detections, our method optimizes tracks for each target individually which avoids the complexity of multi-target data association.

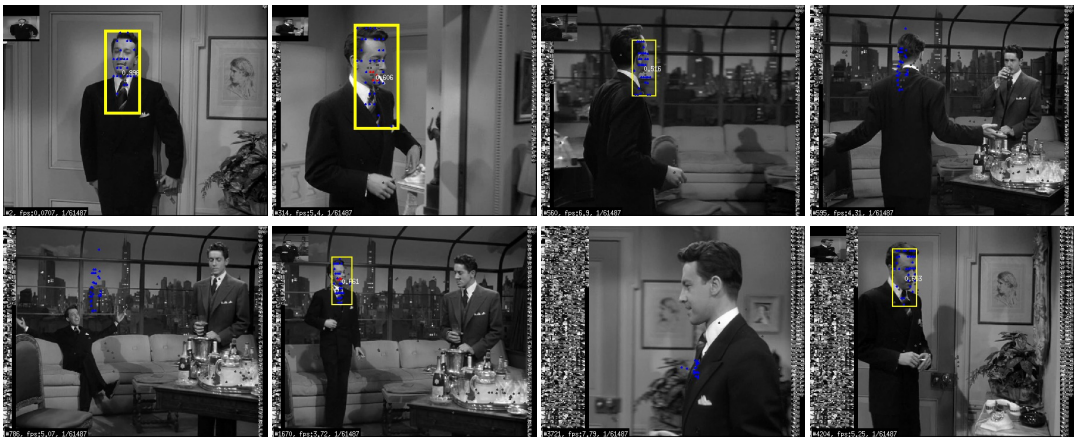
Recent work [KMM12] demonstrated that a detector can be learned by gathering the weakly labelled training data during tracking to address post failure behavior. But the algorithm is quite sensitive to initialization and fails prematurely in many cases possibly due to lack of distinguishing features. Results may significantly differ by varying the initialization window as illustrated in Figure 6.3. Also, such an approach is not robust in a multi-viewpoint scenario, because the detector may be biased towards a viewpoint from early tracking and may fail to continue tracking after fast viewpoint changes. It may re-detect the object if it returns to this viewpoint, but may lose a significant part of the object track. The results may significantly vary based on the viewpoint chosen for initialization (illustrated in Figure 6.4).



(a) Initialization with face

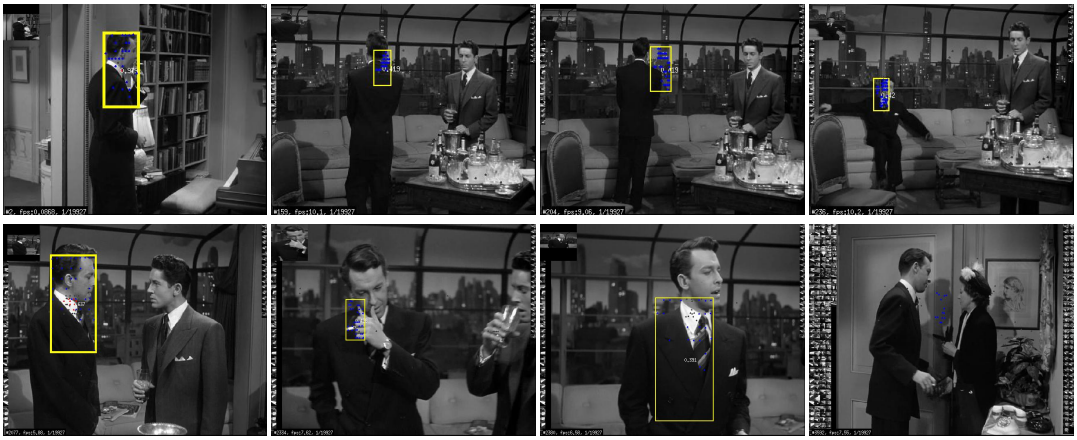


(b) Initialization with upper body

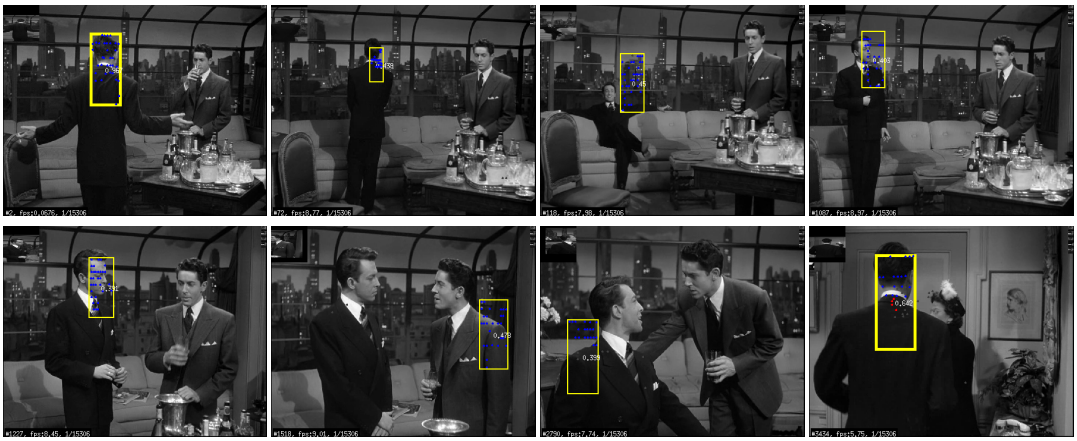


(c) Initialization with upper body (tighter)

Figure 6.3: *Tracking results with TLD[KMM12] tracker on Rope sequence with three different initialization windows. The TLD tracker is only initialized with a single window and has the ability to re-detect the target after occlusions or tracking failures. But the learnt detector is biased towards the initialized viewpoint and struggles with viewpoint variations. Also, the tracking results (both accuracy and coverage) significantly differ with different initiations.*



(a) Initialization with side view



(a) Initialization with back view

Figure 6.4: *Tracking results with TLD[KMM12] tracker initialized at different viewpoints. (a) Tracker initialized at a side view. We can observe that the tracker successfully re-detects the target when it returns to initialized view (second row, first frame) but tracking is inaccurate for large part of the video. (b) Tracker initialized with a back view. The tracker is inaccurate for large part of the video but interestingly it re-detects the target when it returns to initialized view (second row, last frame).*

Our approach extends previous work on offline tracking. For interactive offline tracking, Wei et al. [WTS07] show that tracking can be done by optimizing in the discrete space of candidate tracks. Our work builds on this to a fully-offline (non-interactive) scenario by introducing a stronger appearance model and being based purely on detections. Our work is also related to the self paced learning approach [SR13] that re-learns the discriminative model over multiple phases of offline tracking. In contrast, our approach uses view-independent generative models which are built prior to detection using a small set of training examples. Gu et al. [GZT11] also use a non-adaptive target and background model based on a set of user annotated images but their method works at individual pixel level and we use a structured patch-based model.

6.3 LEARNING AND DETECTION

Our approach uses the rigid patch based detector (PBD) presented in the previous chapter (Section 5.6). It takes as input a small set of representative images with the actor to be tracked annotated, as well as the video in which to track the actor. Combining information from multiple views into a single model instead of using a separate template for each view gives three main advantages: First, such a model gives preference to a feature which consistently appears at the same position across multiple views and still retains the benefit of choosing between optional features at a given position. Second, it is less susceptible to noise as compared to using multiple templates, where noise in one of the templates may have strong influence over the final detections. Third, other intermediate views (which are not included in the representative set) can be generated using the combination.

The training is performed using 8 examples for each target actor/object uniformly sampled across different views (front, side, back) and each training image is annotated using the line based approach explained in Section 5.6.2. The update and detection were performed in exactly the same manner as explained in Section 5.6.3 and Section 5.6.4. The update is performed only using the training images and the same model (updated) is used for detection throughout the video. Following the detection at multiple scales, a total of M top scoring windows per frame per actor are considered for the optimization step, after non maxima suppression.

6.4 OPTIMIZATION

Detections for each actor are individually optimized into a smooth track using the optimization step. Our method does not perform any motion propagation based trajectory growing to create more detection candidates or to reject existing ones like in other offline tracking methods [WTS07, ESZ06] and the optimization is purely based on individual detections at each frame. For a given actor, the optimization method takes as input all the detection windows over the entire video with the corresponding detections scores. It outputs a sequence $\xi = \{s_t\}$ of object states s_t for all frames. The object states are defined as:

$$s_t = \begin{cases} 0 & \text{if occluded and} \\ m & \text{if visible}, m \in [1:M] \,, \end{cases} \tag{6.1}$$

where M is the total number of detection windows considered in a frame. The location of a detection window is defined as $l(s_t, t) = \{x(s_t, t), y(s_t, t), w(s_t, t)\}$, where $x(.)$ and $y(.)$ are the center of the detection window and $w(.)$ is the width. By convention we take $l(0, t)$ as the last location where occluded object was visible. The corresponding detection scores are represented by $P(s_t, t)$. Given the input we minimize the following global cost function:

$$E(\xi) = \sum_{t=1}^{N} E_d(s_t) + \sum_{t=2}^{N} E_s(s_{t-1}, s_t). \tag{6.2}$$

The cost function consists of a data term E_d that measures the evidence of the object state given the detection score and a smoothness term E_s that measures the smoothness of the object motion. The data term and the smoothness term are defined as follows:

$$E_d(s_t) = \begin{cases} -\log(P(s_t, t)) & \text{if } s_t > 0\,, \\ \lambda_1 & \text{if } s_t = 0\,, \end{cases} \tag{6.3}$$

$$\text{and } E_s(s_{t-1}, s_t) = \begin{cases} D_m(l(s_{t-1}, t-1), l(s_t, t)) & \text{if } s_t > 0\,, \\ \lambda_2 & \text{if } s_t = 0\,. \end{cases} \tag{6.4}$$

D_m is the Mahalanobis distance and λ_1, λ_2 are fixed constants. Our method is in similar spirit to the dynamic programming approach used in [WTS07]. One main difference is that we define the occluded to visible transition based on the detection evidence in the current frame and smoothness term based on the last visible state, instead of a constant term used in [WTS07]. This smoothness term is inspired by the one used by Buchanan and Fitzgibbon [BF06]. We use the Mahalanobis distance in contrast to the Euclidean distance used in both these methods.

We prefer the Mahalanobis distance because we observed that each state element (x), (y) and (w) follows a normal distribution in the ground truth data. We also found that the variance of each state element was much higher than the inter-state variance, so we define the reduced distance term D_m as follows:

$$D_m(l_{t-1}, l_t) = \frac{(x_{t-1} - x_t)^2}{\sigma_x^2} + \frac{(y_{t-1} - y_t)^2}{\sigma_y^2} + \frac{(w_{t-1} - w_t)^2}{\sigma_w^2}. \tag{6.5}$$

Here l_t refers to $l(s_t, t)$ and σ_x, σ_y, σ_z are parameters. Finally, we use dynamic programming (DP) to solve the optimization problem presented in Equation 6.2.

The key idea is that optimization over time can be regarded as 'optimization in stages' and dynamic programming is an efficient way of reducing the search space for problems where one of a number of states must be chosen for each of a series of successive stages. In our case, the stages are the frames of the video sequence and the states are the possible detections (out of the top M detection candidates) or a hidden state in each frame. We implemented it by filling a table whose rows and columns represent the matches (states) and frames (stages) respectively, specifically an $(M+1) \times N$ table in our case. Exploiting the principle of optimality, the entries of the table are computed one column at a time. Each entry in the table holds the cost of the optimal track, ending on that state (through the whole sequence up to that frame) and the index of the entry in the last column leading to this state. For hidden states (entries in the last row), the last visible location (in image coordinates) is also stored.

An entry of a row(state) in a given column(frame) is computed by cycling through each entry of the previous column, and determining the total cost using the track up to that point (by adding the pairwise error of the two matches from Equation 6.2), the cost and index of the chosen match (with lowest cost) is then stored. In case of a visible to hidden transition, the location where the transition occurs is also stored to compute the cost of hidden to visible transition. For hidden to hidden transition, the last visible location is carried forward. The global optimum is the track whose error in the final column is lowest.

6.5 EXPERIMENTAL RESULTS

This section presents results of our method in a variety of application scenarios ranging from single target with static background to multiple targets with dynamic background. We first explain the experimental setup, which gives all implementation details required to replicate the proposed work. We then present comparative results with previous work, which demonstrate that our approach outperforms state-of-the-art tracking methods in all scenarios.

6.5.1 Experimental setup

For validation purposes, we chose a fixed set of parameter values for all the experiments. The color difference threshold (θ) was set to 10 throughout the clustering and the detection process. All singleton clusters were rejected as noise after the clustering step. We used $\lambda_0 = 0.1$, in the model update. We used a maximum of 6 different scales for detection.

The number of candidates per frame (M) was set to 5 times the number of scales used in detection. In the optimization step, the constant λ_1 was set to $-\log(P_m/4)$, where P_m was the maximum detection score for the given sequence and λ_2 was set to 0.22. The parameters σ_x, σ_y and σ_z were set to 15, 10 and 0.25 respectively.

6.5.2 Experimental data

We evaluated our algorithm on six challenging sequences chosen from prior work [SLS*10, KL10, ARS06, WLYY11] and two new sequences of actor performances in movie and theatre. Sequences used in prior work include the following: *liquor* and *lemming* from PROST [SLS*10], *girl* from SPT [WLYY11], *woman* from FragTrack [ARS06], *basketball* and *skating2* from VTD [KL10]. Our own sequences present actor performance in theatre (with a static camera) and film (with a moving camera). In the theatre sequence, we separately present tracking results for each of the four actors. The selected sequences include many of the challenging factors in visual tracking, including complex backgrounds, moving cameras, fast target movements, large variation in pose and scale, and many cases of partial and full target occlusions.

6.5.3 Comparative results

For each of the test sequences, we compared our tracker with the following state-of-the-art methods: MILTrack [BYB09], adaptive color-based particle filter (PF) [NKMG02], IVT[RLLY08], VTD [KL10], SPT [WLYY11], TLD [KMM12], SPLT[SR13] and OTLE[GZT11]. The methods SPLT and OTLE work offline and the rest are online trackers.

For fair comparison, we used the best reported results from prior work when available, or we carefully adjusted the parameters of all trackers with the code provided by the authors and used the best result from five runs. Some of these algorithms were very sensitive to noise and performed poorly if the initial bounding box included some background. For fair comparison we tried different initializations which better suited the nature of the given tracker and finally chose the one which gave best results. The method OTLE takes as input a set of labeled images and we use the same set of images as used in our method. One major difference is that OTLE stores the timing and position information and we only use the appearance information from the labeled images.

Sequence	PF	IVT	MIL	VTD	SPT	TLD	SPLT	OTLE	Ours
liquor (1741)	0.47	0.22	0.20	0.27	0.98	0.92	0.59	0.92	**0.99**
lemming (1336)	0.32	0.78	0.83	0.35	0.96	0.86	0.86	0.47	**0.97**
girl (1500)	0.74	0.07	0.37	0.54	0.79	0.27	0.85	0.35	**0.97**
woman (470)	0.06	0.10	0.08	0.05	0.66	0.90	**0.99**	0.68	0.98
basketball (725)	0.63	0.11	0.28	0.83	0.87	0.08	0.29	0.58	**0.97**
skating2 (707)	0.57	0.13	0.33	0.68	0.28	0.03	0.62	0.63	**0.84**

Table 6.1: *Tracking results. First column gives the name of the sequences and the total number of frames in it. Each row presents the recall of different algorithms over the given sequence (ratio of successfully tracked frames). The recall was based on evaluation metric of the PASCAL VOC object detection which is also used in other tracking algorithms [WLYY11, SLS*10]. Best results are shown in bold letters and the second best in blue letters.*

Quantitative results for all methods are summarized in Table 6.1 and commented in the next three paragraphs. Qualitative results are illustrated in Figure 6.5. Qualitative results on an additional longer sequence (5 min) from movie rope is illustrated in Figure 6.6. All the resulting videos are available online[1].

Fast motion and visual drift

The PF, IVT, MIL and Frag trackers shown drift with fast motion and often failed to recover. The VTD and SPT were comparatively stable to fast motion benefiting from a stronger motion model. The TLD tracker also struggled to accurately localize object in presence of fast motion due to lack of consistent feature tracks. The offline method OTLE gave inaccurate localization with fast motion because it is purely based on pixel level features which often get blurred and lost with fast motion. The method SPLT uses HOG features and did not have problems of drift but often shifted away to similar looking object in case of fast motion (as shown in *liquor* 380, *rope* 656 in Figure 6.5). The proposed method was robust to fast motion in all the sequences.

Viewpoint, pose and scale variations

The TLD tracker often gets biased towards the poses available during the early tracking. SPLT tracker also shows a strong affinity towards the object pose at initialization. Due to this reason, both the TLD and SPLT trackers often disappeared or gave unstable results with fast viewpoint and pose variations (as shown in *liquor* 1419, *lemming* 988 and *theatre-a1* 984 in Figure 6.5). Since our method uses a pre-built model from a set of representative images from different viewpoints, it did not show any such bias. The OTLE tracker although using multiple images for initialization was not as robust to viewpoint changes as the proposed approach (as shown in *liquor* 1419, *lemming* 1118 in Figure 6.5). It also has a strong limitation as it fails to handle scale changes.

Occlusions

We observed that occlusion was one of the main reason for failure of online methods. The trackers SPT and VTD worked fairly well in presence of object to background occlusions but failed with object to object occlusions (blue and yellow rectangles in *theatre-a4* sequence in

[1] http://imagine.inrialpes.fr/people/vgandhi/offline_tracking/

Table 6.2: *Tracking consistency results. The numbers denote the count of successfully tracked frames and recall (in bracket) for the Girl sequence on 5 different instances with different set of training images.*

user1	user2	user3	user4	user5
1461(.97)	1464(0.98)	1435(.96)	1482(0.99)	1461(.97)

Figure 6.5). The results on the long sequence *rope* show that all of the online trackers drifted away from the target eventually and were unable to recover from it. As expected, the offline methods and the TLD tracker recovered from occlusions in most cases. But our method gave accurate localization in presence of partial occlusions as compared to other methods (examples shown in *liquor* 1609 and *theatre-a1* 519 in Figure 6.5). Additionally, results on the *theatre* sequence show that how we can track multiple actors in presence of frequent actor to actor occlusions which may be difficult to attain using generic detectors [FGMR10, DT05]. This clearly demonstrates that using actor specific detections gives an added advantage in resolving multiple actor tracks.

Robustness over choice of initial training images

We performed experiments to measure the effect of the choice of examples views used in the learning step on the overall performance of our tracker. We asked 5 different users to make a quick selection of 8 images from the girl sequence in such a way that the set includes 2 near front views, 2 near back views and 4 side views. The video was shown to the user before the selection process. These set of images were then used to build the appearance model for tracking. Results in Table 6.2 show that our method is quite robust to the selection of initial training frames and gives consistent results in all cases.

6.5.4 Performance

In theory, the complexity of our method is dominated by the optimization step, which is $\mathcal{O}(KMN)$ in space and $\mathcal{O}(KM^2N)$ in time, where K is the number of targets, M the average number of candidates per target per frame and N is the number of frames in the sequence.

In practice, the model initialization and update steps are very quick, taking less than a second per sequence. Since we use only a small number of detection candidates at each frame, the optimization step is also quick and takes less than a second in all sequences. Detection is the most computationally expensive step and takes about 1/2 second per object per frame in our current implementation.

6.6 SUMMARY

It is interesting to note the limitations of the current state of the art tracking methods. Although many of the state of art trackers have shown impressive results on challenging short sequences using only an initial bounding box, the applicability to longer sequences is still limited. We have tested the existing tracking methods on variety of sequences but the cases of *Theatre* and *Rope* are of special interest to us. The theatre sequence (Figure 6.5) shows four actors interacting with each other, the camera is static and the illumination variation is also limited. In

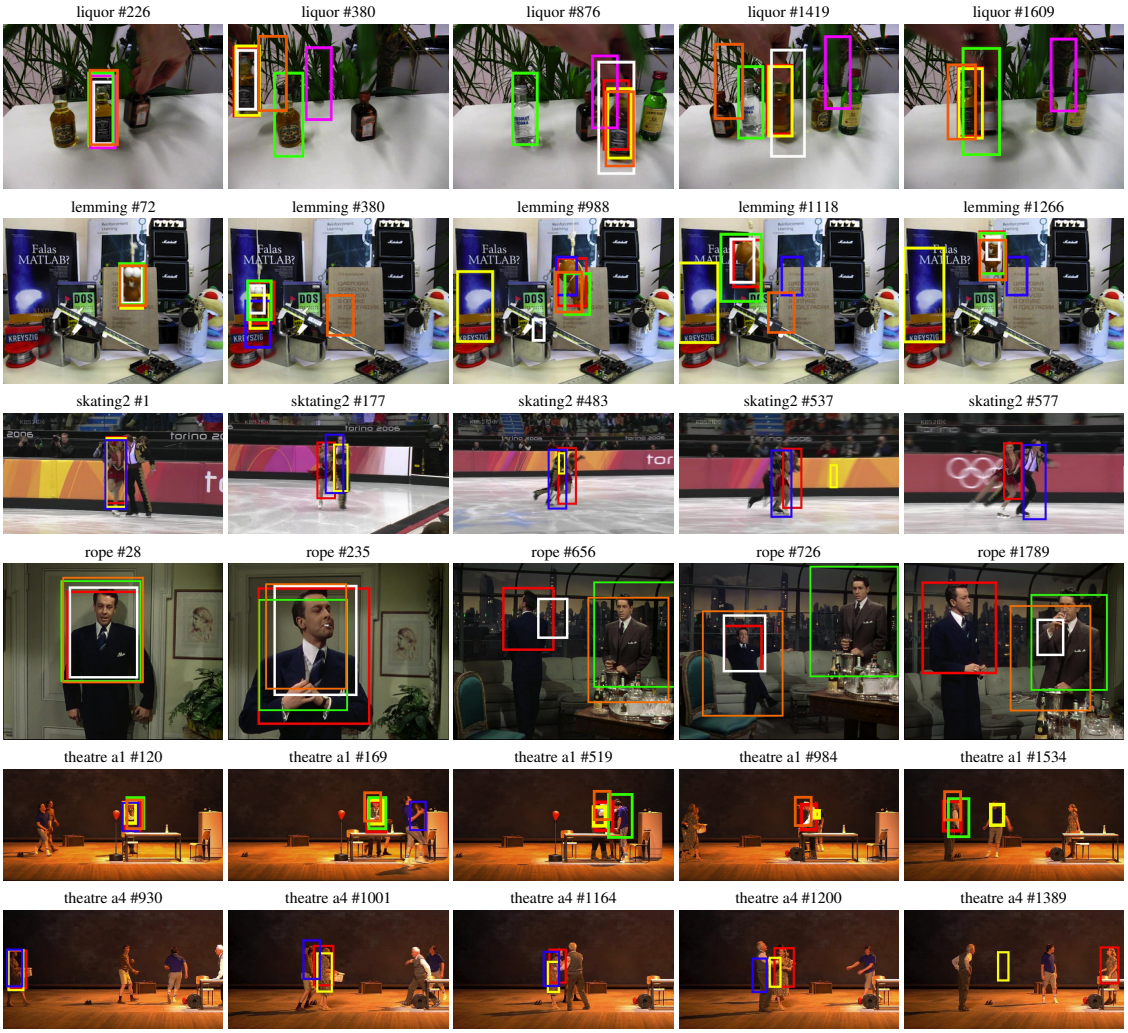


Figure 6.5: *Tracking results. The results by our tracker(in red), TLD(in white), SPT(in yellow), VTD(in blue), MIL(in magenta), SPLTT(in Green) and OTLE(in orange) are being shown in different colored rectangles. Only few competing algorithms are shown in each case for visualization purposes.*

the rope sequence (shown in Figure 6.5) a camera follows a moving actor who is changing viewpoint and scale, the illumination variation is also limited in this case. Both cases might appear simpler than shorter challenging sequences which have been used for experimentation in previous research. But most of the existing trackers fail to track consistently in both cases. In fact the tracking performance is far from something which can be used for practical applications, for example in an application like automatic rush generation proposed in this thesis. We believe that future research in tracking should focus on tracking on longer sequences, even if it requires to limit the applicability of the tracker.

Variations in target appearance are a major challenge in video tracking. Previous work has focused on learning those variations during tracking. This prediction is difficult and usually fails in long term sequences. We have demonstrated that how pre-learned view-independent models like PBD can be efficiently used to tackle this problem. We have demonstrated im-

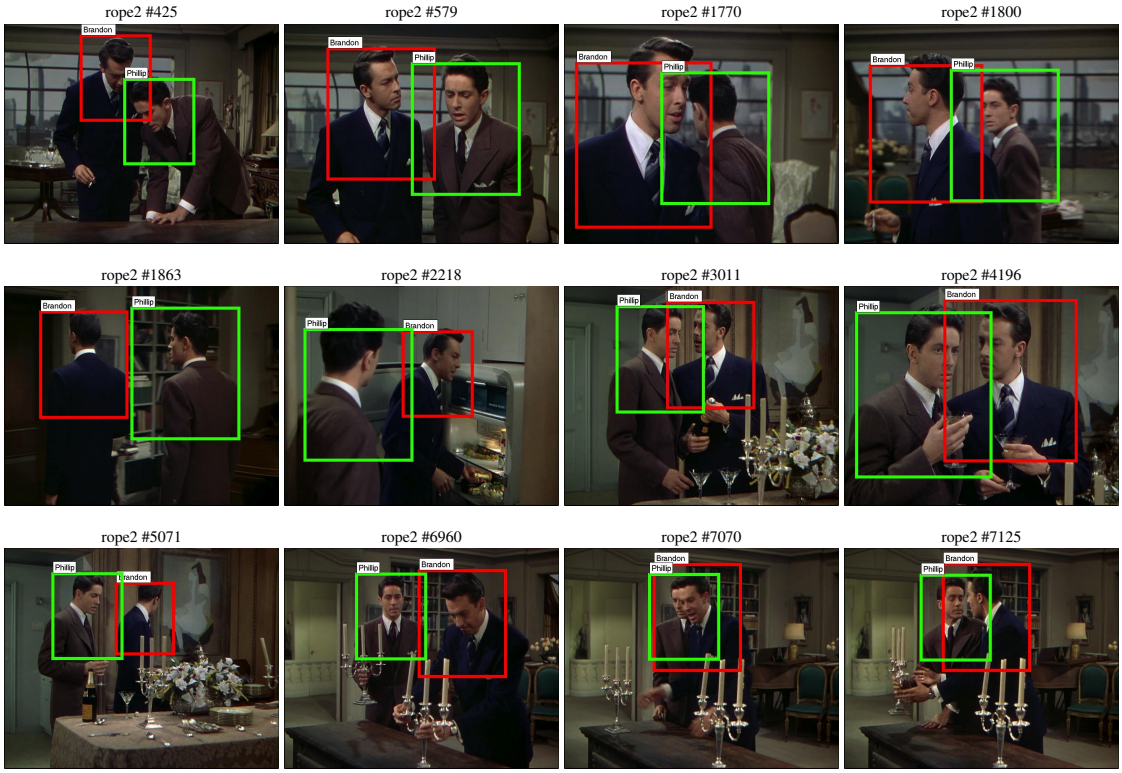


Figure 6.6: *Tracking results using our method on a five minute long sequence from rope with two actors. Results for Brandon are shown with red windows and for Phillip are shown with green rectangles. Our method successfully detects both actors in presence of frequent occlusions with significant view point and scale variations.*

provements in the recall rate of our tracker compared to previous work in a large number of scenarios, ranging from single-target, static background to multiple-target, dynamic backgrounds.

Our approach is designed for a specific scenario: where representative views of the object can be identified prior to tracking, and online performance is not necessary. Our methods exploit these restrictions to enable more robust performance. Future work should explore wider range of cases. For example, it might be an interesting idea to build a dataset of long video sequences where an 360 degree object rotation is available for training (prior to tracking). This would allow to test and compare different algorithms for long term tracking.

The proposed appearance model combines multiple viewpoints, but has a simple structure. Our use of a rigid part configuration limits it to near upright configurations and limits the amount of pose changes that may be handled. Future work should look at more flexible structures like CBD for the tracking purposes. Also, our current color models cannot handle large illumination changes or changes in object appearance (as when an actor changes clothes). Future work should explore multi-view tracking using a combination of both textual and color features (as they may well complement each other). Because our approach does not consider discrimination between targets, it may have problems when there are objects of similar appearance, such as when trying to track a particular player in a football game. It would be interesting

to add a discriminative aspect to our model.

Despite these limitations, our approach is successful in many challenging scenarios, especially in the cases of theatre performances which is the main focus of this dissertation. Although it is not a traditional approach in tracking to initialize using multiple views, we believe it can make a strong impact in future, especially in long term tracking.

CHAPTER

7

RUSH GENERATION

HIGH quality video uses a variety of camera framings and movements edited together to effectively portray its content on screen as we have discussed in Chapter 2. To produce such video from a live event, such as a theater performance or concert, requires source video from several cameras to capture multiple viewpoints. These individual camera videos, or *rushes,* are then edited together to create the final result. The requirement of a "multi-camera shoot", including having multiple synchronized cameras each with a skilled operator capable of creating good framings and camera movements, makes it expensive and intrusive, and therefore impractical for many scenarios.

In this chapter we introduce a new method to create multiple synchronized videos from a live staged event that are suited for editing. Our key idea is to use a single non-moving camera that captures the entire field of view of the event and *simulating* multiple cameras and their operators as a post-process, creating a synchronized set of rushes from the single source video. This strategy allows us to avoid the cost, complexity and intrusion of a multiple camera shoot. A pan-tilt-zoom (PTZ) camera can be simulated simply by cropping and zooming sub-windows of the source frame. The challenge we are addressing here is simulation of a competent *camera operator:* choosing the different virtual viewpoints such that their results are videos that are likely to be useful for editing. This means that each video must not only obey cinematic principles to be "good video" but also have properties that make it easier to edit with the other rushes.

7.1 PROBLEM STATEMENT

We study the problem of automatic rush generation from a single viewpoint i.e. automatically generating multiple clips suitable for video editing by simulating pan-tilt-zoom camera movements within the frame of a single camera. The input to the system is locations of the important actors and objects in the video and a minimal user description to define the subject matter of each sub-clip. The output is multiple synchronized sub-clips corresponding to different descriptions.

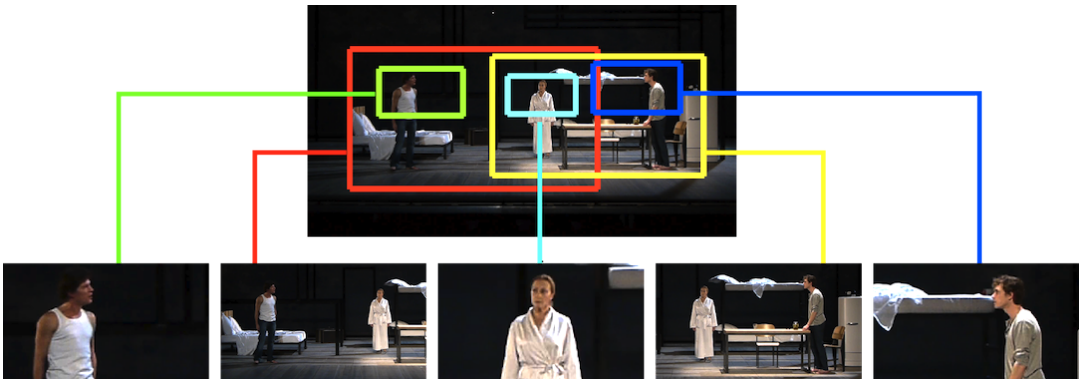


Figure 7.1: *Our method takes as input a high resolution video from a single viewpoint and outputs a set of synchronized subclips by breaking the groups into a series of smaller shots.*

Our approach:

Our solution, illustrated in Figure 1, takes as input a single "master shot" — a high resolution video taken from the viewpoint of a member of the audience. Because we consider staged events (such as theater performances or concerts), we can assume that this vantage point is sufficient to see all of the action (otherwise, the audience would miss it as well). The method described in Chapter 6 is used as a pre-process to identify and track the actors. Our method then creates videos by simulating each of the cameras individually. Each virtual camera takes a simple specification of which actors it should contain and how they should be framed (screen position and size). A novel optimization-based algorithm computes the dynamic framing for the camera i.e. the movement of a sub-window over the master shot. The optimization considers the specification, cinematic principles of good camera movements, and requirements that the videos can be cut together.

The output of our method is a set of synchronized videos (or *rushes*) that provide multiple viewpoints of the staged event. These rushes can then be edited together using traditional multi-track video editing software. This allows a human editor to use their creativity, expertise in editing, and understanding of the content, to create a properly edited video. Our approach automates the synthetic rush generation process, which is tedious to perform manually.

The central idea of our approach is that we can cast the problem of determining a virtual camera movement, that is the size and position of the subregion of the master clip over time, as an optimization problem. Specifically, we cast it as a convex program allowing for efficient and scalable solution. The key insight is that many of the principles of good camera motion, including what camera shots are useful in editing, can be cast within this optimization framework.

7.2 VIRTUAL CAMERA SPECIFICATION

In this section, we present and explain our shot specification scheme in details, and review some fundamental concepts in cinematography that motivate our optimization approach. As we have discussed in Chapter 2, several established video production books emphasize the role of composition and cutting in cinematography [Alt49, Mas65, LQ00, Mer10]. Mascelli [Mas65] identifies five main topics in cinematography: camera angles (where to place the camera relative to the subject); continuity (making sure that actions are fully covered); cutting (making sure that transitions between cameras are possible); close-ups (making sure that important parts of actions are emphasised); and composition.

Since we are working from a single camera angle (the camera position chosen for the input master shot) and the action is recorded in full continuity, camera angles and continuity are given as an input to our method. On the other hand, we need to take special care of composition and cutting when computing virtual camera movements to compensate for the constant camera angle.

7.2.1 Actor detection

The input to our method is a list of all actors present in the master shot and their bounding boxes. We assume they are given as (bx, by, bs, bh) where bx, by are the center coordinates of

Figure 7.2: *Input to the system are the upper body bounding boxes for each actor (shown with squares) and the points where the actors touch the stage floor (shown with cross markers).*

the upper body bounding box, bs is the actor's upper body size and bh is the actor's height on stage(all in pixels). An actors height on stage is the length from top of the upper body bounding box to the point it touches the stage floor. An example of the input bounding boxes and corresponding floor points are shown in Figure 7.2. For standing posture the stage height (bh) of an actor is approximately four times the size of the upper body bounding box (bs). But this ratio may not be true for other postures like sitting or bending and it becomes important to take into account the floor points to compute the stage height of the actor.

The actor tracks were obtained using the method proposed in Chapter 6. In some of the challenging sequences like the theater sequence in Table 6.1, part of the actor tracks were inaccurate or missed (for example the recall for theatre-a3 sequence in Table 6.1 is only 87 percent). The missing and inaccurate parts were manually corrected for experiments in this chapter. Given the actor tracks we compute the ground point projections using a calibration of the stage floor with respect to the coordinates of the detection windows. We model the stage floor by equation of a plane using three manually labelled 3D points with first two dimensions as image coordinates of the annotated point and the third dimension (depth) as inverse of size of the actor bounding box at the given point on stage. We then use the equation of the plane to compute the floor projection for any arbitrary actor bounding box (by inputting x-coordinate as bx and the depth as $1/bs$ to compute the y-coordinate).

7.2.2 Virtual camera geometry

The images of two virtual PTZ cameras with identical camera centers are related by a projective transformation (homography). In principle, a virtual PTZ camera image can therefore be specified exactly from the master camera image with four points (eight parameters).

In practice, we use a rectangular cropping window with a fixed aspect ratio as a simplified model of a virtual PTZ camera. Thus our camera model is specified with just three parameters : the virtual camera center fx, fy and the virtual camera height fs (all in pixels), where (fx, fy, fs) lie within the space of the master image. This has the benefit that the virtual camera does not create unwanted geometric deformations in the virtual image and it preserves the resolution of the master camera image. The virtual camera image can therefore be obtained by isotropic re-sampling of the cropped image from the master shot.

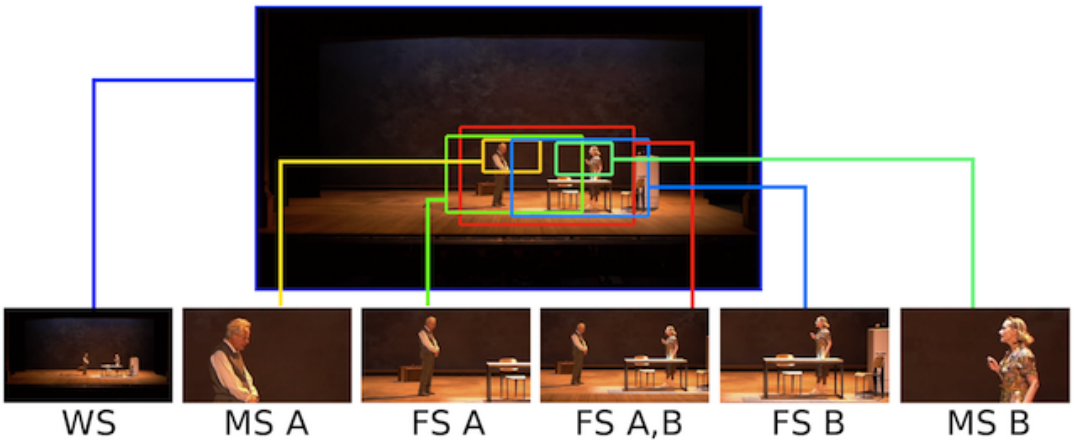


Figure 7.3: *The figure shows the possible set of framing with two actors (A,B) and two shot sizes i.e. the medium shot (MS) and the full shot (FS). Even with a simple case of two actors a total of 6 camera choices are available including the original wide shot (WS).*

7.2.3 Shot naming conventions

Based on common practice in video production [TB09b], shots are specified by their subject matter and size. The subject matter for a shot is typically the list of actors who should be on-screen. Optionally, objects or stage locations can also be subject matters for a virtual camera shot, although this is not used in this work. The size for a shot is typically defined by specifying the average screen size occupied by each actor. We use classical conventional shot size names including "long shot" (LS), "full shot" (FS) and "medium shot" (MS) as a specification.

Our system lets users choose the subject matters and shot sizes for each virtual camera independently. For sake of simplicity, in this chapter we keep the specifications constant for the entire duration of the master shot. (in Chapter 8 we will see some examples where the shot specification changes over time). With reference to the PSL described in Chapter 4, we limit to "*pan with*" shots and the screen position of the actors are automatically estimated. For instance, the specification for a virtual camera can simply be a "pan with full shot of actor A and actor B" or just "FS A,B" (in a reduced form). Figure 7.3 shows that a total of five shots can be specified using only two sizes and two actors.

Given the actor bounding boxes, the subject matters and shot sizes for all virtual cameras, the task is now to compute virtual camera trajectories resulting in good composition for each camera, and preserving possibilities for cutting between cameras.

7.2.4 Composition

As we have discussed in Chapter 2, good camera work begins with composition, which includes not just the composition of objects in a static frame, but also the composition of movements in a dynamic frame. Framing, or image placement, is the positioning of subject matter in the frame[Mas65] and is the most important aspect of shot composition in our context. For our purpose, the most important framing principles emphasized by Mascelli are that (a) subjects should *not* come into contact with the image frame; (b) the bottom frame should *not* cut across subject's joints (knees, waist, elbows, ankles) but should instead cut *between joints*; (c) subjects

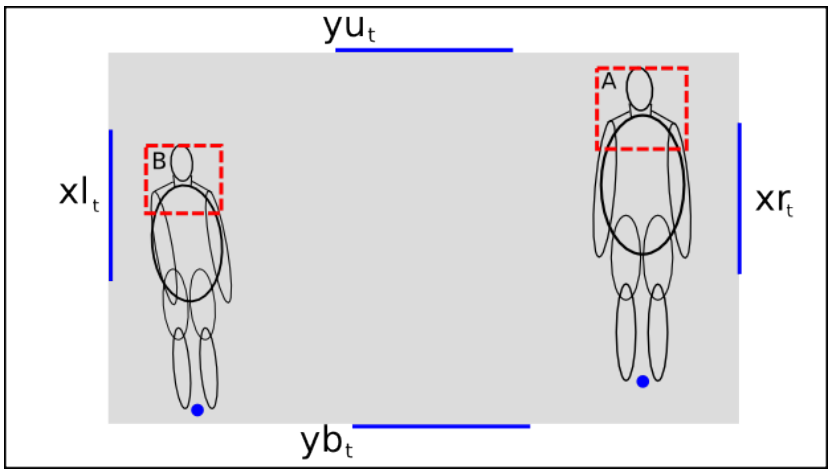


Figure 7.4: *Inclusion region for shot specification "FS A,B" i.e. the full shot of Actor A and Actor B. A full shot of two actors is a tightest window which keeps both of them entirely inside the frame. The inclusion region at time t is denoted by four coordinates (xl_t, xr_t, yu_t, yb_t), denoting the left, right, top and the bottom boundaries of the region respectively.*

must be given more space in the direction they travel and the direction they look.

The subjects in our case are the actors in the shot specification. To ensure that the subjects do not come into contact with the image frame, we define an inclusion region for the given shot specification. This inclusion region is then encoded as a hard constraint in our optimization framework to make sure that the subjects are always nicely kept inside the virtual camera frame. Examples of inclusion regions with two different shot specifications are shown in Figure 7.4 and Figure 7.6 with shaded rectangles. The inclusion region is defined using four coordinates (xl_t, xr_t, yu_t, yb_t). The values xl_t, xr_t and yu_t denote the leftmost, rightmost and the topmost coordinate of the bounding boxes of actors included in the shot. The coordinate yb_t is defined differently for the full shot and the medium shot. For a single actor in a medium shot yb_t is the topmost coordinate plus twice the size of his upper body. In the case of full shot of an actor yb_t is the point where the actor touches the stage floor. In case of multiple actors, the lower coordinate is computed for each actor individually and yb_t is taken as the maximum value among them. A shot size penalty in the optimization cost function tries to keep the virtual camera framing close to the inclusion region maintaining a nice composition. The penalty is explained with detail in Section 7.3.2.

The shot size penalty and hard constraints ensure that the actors specified in the shot are nicely framed inside the virtual camera. But other actors may still come in contact with virtual camera window. To avoid this we add another penalty term in the optimization framework which avoids chopping external actors and tries to keep them either fully outside or pulls them fully inside the virtual camera window. This is explained in detail in Section 7.3.6.

7.2.5 Cutting rules

An another important consideration in our work is to make sure that the virtual PTZ cameras produce shots that can easily be edited together. We enforce “cuttability” of the shots by maintaining screen continuity of all actors and by creating only "sparse" virtual camera move-

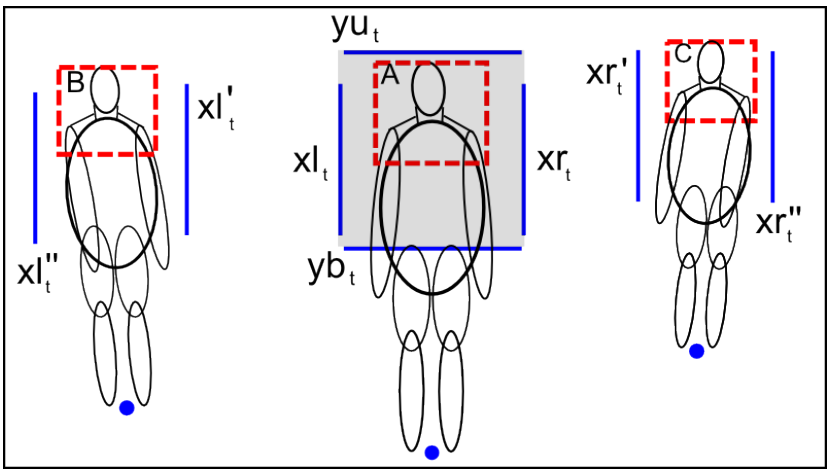


Figure 7.5: *Inclusion region for shot specification "MS A" i.e. the medium shot of Actor A and boundaries of the nearest actors outside the inclusion regions at time t. The boundaries of the nearest actor outside the inclusion region on the left are denoted by xl'_t and xl''_t. To avoid chopping this actor, left coordinate of the virtual camera frame should not lie within xl'_t and xl''_t. Similarly, the boundaries of nearest actor outside the inclusion region on the right are denoted by xr'_t and xr''_t.*

ments.When cutting between cameras showing the same actors with different compositions, it is important to keep them in similar screen positions. To enforce such screen continuity, we give preference for virtual shot compositions where the actors keep the screen positions from the master shot. An example is given in Figure 7.3 with two actors. The actor on the the left is kept on the left side in the virtual camera shot composition "MS A" and the actor on the right is kept on the right side of the virtual camera shot composition "MS B".

As we have discussed in Section 2.3.2 cutting during camera movement is difficult because the movements of the two cameras should be matched. As a result, film editors typically prefer to cut when none of the cameras are in motion.To maximize the number of opportunities for cutting, we therefore give a preference to virtual cameras with *sparse* pan, tilt and zoom movements. As will be explained in the next section, we enforce this preference by regularizing the first order derivative of the virtual camera coordinates in the $L1$-norm sense.

7.2.6 Camera movement

We have discussed the importance of fully static 'locked' camera in Section 2.3.2. It is of no use to prepare a well-composed shot only to have its image blurred or confused by an unstable camera. As discussed in earlier section, a static camera is also beneficial for cutting. Also the camera should not move without a sufficient motivation as it may appear puzzling to the viewer.

The goal of avoiding movement as much as possible still leaves the question of what kinds of movements to use when they are necessary. Thomson in his book [TB09b] mentions that a good pan/tilt shot should comprise of three components: a static period of the camera at the beginning, a smooth camera movement which "leads" the movement of the subject and a static period of the camera at the end.

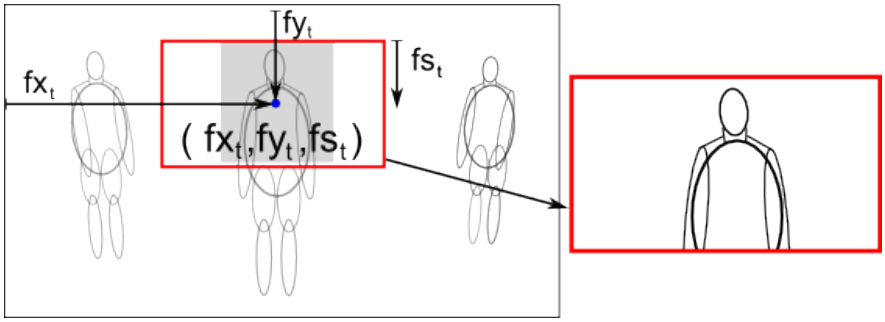


Figure 7.6: *The output of the optimization algorithm is a cropping window for each frame of the video. The cropping window at a frame t is denoted by three coordinates* (fx_t, fy_t, fs_t) *within the original master shot.*

As mentioned in the earlier section, we use L1-norm regularization over the first order derivative of virtual camera coordinates to get the static camera behavior. In order to obtain smooth transitions between the static segments we add L1-norm regularization term over the third order derivative of the virtual camera coordinates. This will tend to give segments of constant acceleration and deceleration, creating the ease-in and ease-out effect. This is explained with detail in Sections 7.3.3 and 7.3.4.

An instant problem which may arise while moving a cropping window inside the master shot is that the actual motion of the actor on stage may not be preserved inside the virtual camera. For example, an actor who is static on stage may appear moving/sliding inside the virtual camera frame or an actor moving on the left on stage may appear moving on the right in the virtual camera frame. We introduce another penalty term in the optimization framework to preserve the apparent motion of the actors. This is explained with detail in Section 7.3.5.

7.3 OPTIMIZATION

In this section we show how different cinematographic principles explained in the previous section are defined as different penalties or constraints and are combined in a single convex cost function which can be efficiently minimized to obtain the virtual camera trajectory for a given shot specification. We first summarize the notation and then explain each term of the cost function in detail.

Notation: The algorithm takes as input the bounding boxes $(bx_t^m,\ by_t^m,\ bs_t^m,\ bh_t^m)$ for each actor $(m = [1 : M])$ and time t. The algorithm also takes as input the inclusion region $\{xl_t, xr_t, yu_t, yb_t\}$ and the external actor boundaries $\{xl'_t, xr'_t, xl''_t, xr''_t\}$, which are derived using the actor tracks and the shot specification.

The algorithm outputs a cropping window $\xi = \{fx_t, fy_t, fs_t\}$ for each frame $(t = [1 : N])$, where (fx_t, fy_t) are the coordinates of the center and (fs_t) is the size i.e. half of the height of the cropping window (illustrated in Figure 7.6). We also define $gx_t = \frac{1}{2}(xl_t + xr_t)$ as the midpoint of the left and the right coordinates of the inclusion region and $gy_t = \frac{1}{2}(yu_t + yb_t)$ as the midpoint of vertical inclusion coordinates. We define $gs_t = \frac{1}{2}(yb_t - yu_t)$ as the desired size of the cropping window and A_r as the required aspect ratio. The variable fs_t denotes the vertical half length of the cropped window and the horizontal half length is given by $A_r fs_t$.

Symbol	Meaning
fx	horizontal center of the output virtual camera window
fy	vertical center of the output virtual camera window
fs	size of the output virtual camera window
$\dot{fx}$	horizontal velocity of the virtual camera window
$\dot{fy}$	vertical velocity of the virtual camera window
$\dot{fs}$	change of size of the virtual camera window between consecutive frames
bx^m	horizontal center of the upper body bounding box of actor m
by^m	vertical center of the upper body bounding box of actor m
bs^m	size of the upper body bounding box of actor m
bh^m	height of the actor m on stage
$\dot{bx}^m$	horizontal velocity of actor m on stage
$\dot{by}^m$	vertical velocity of actor m on stage
$\dot{bs}^m$	change of size of actor m on stage between consecutive frames
xl	left boundary of the inclusion region
xr	right boundary of the inclusion region
yu	top boundary of the inclusion region
yb	bottom boundary of the inclusion region
gx	horizontal center of the inclusion region
gy	vertical center of the inclusion region
gs	size of the inclusion region
(xl', xl'')	boundaries of nearest actor on left outside the inclusion region
(xr', xr'')	boundaries of nearest actor on right outside the inclusion region
W	width of the master shot
H	height of the master shot
h	screen position of the actor in shot specification
cx^m	Boolean variable which is 1 if $\dot{bx}^m$ is less than a threshold
cy^m	Boolean variable which is 1 if $\dot{by}^m$ is less than a threshold
cs^m	Boolean variable which is 1 if $\dot{bs}^m$ is less than a threshold
tl	Boolean variable which takes a value 1 if touch event occurs from the left
tr	Boolean variable which takes a value 1 if touch event occurs from the right

Table 7.1: *List of notation*

All the notations are listed in Table 7.1. Now we explain each of the constraints and penalties of the optimization cost function:

7.3.1 Inclusion constraints

We introduce two sets of hard constraints, first that the cropping window should always lie within the master shot and second that the inclusion region should be enclosed within the

cropping window. Hence, the left most coordinate of cropping window $fx_t - A_r fs_t$ should be less than xl_t and should be greater than zero. Similarly, the right most coordinate of cropping window $fx_t + A_r fs_t$ should be greater than xr_t and less than the width (W) of the master shot. As a result, we define the horizontal inclusion constraints as:

$$0 < fx_t - A_r fs_t \leq xl_t \text{ and } xr_t \leq fx_t + A_r fs_t \leq W. \tag{7.1}$$

Similarly, we define the vertical inclusion constraints:

$$0 < fy_t - fs_t \leq yu_t \text{ and } yb_t \leq fy_t + fs_t \leq H, \tag{7.2}$$

where H is the height of the master shot.

7.3.2 Shot size penalty

As explained earlier, to maintain the desired composition the virtual camera cropping window should remain close to the inclusion region. So, we want fx_t to be close to the midpoint (gx_t) of left and right coordinates of the inclusion region . Similarly, we want fy_t to be close to the midpoint (gy_t) of the top and the bottom coordinates of the inclusion region. Also, we want the size to be closer to the height (gs_t) of the inclusion region. Any diversion from the desired size is penalized using a data term:

$$D(\xi) = \frac{1}{2}\sum_{t=1}^{N}((fx_t - gx_t)^2 + (fy_t - gy_t)^2 + (fs_t - gs_t)^2). \tag{7.3}$$

This term by default always centers the given set of actors. This may not be always good for editing later, where an appropriate look-space is preferred. As discussed in Section 7.2.5, this problem can be resolved by maintaining the screen positions of the actor in the master shot. To do this, we pre-compute a vector h which is 1 at time t if the actor is rightmost on stage; -1 if the actor is leftmost on stage; and 0 if the actor is between other actors. Now appropriate look-space can be created by modifying the term $(fx_t - gx_t)$ in Equation 8.2 to $(fx_t + 0.17Arfs_t h_t - gx_t)$.

7.3.3 First order L1-norm regularization

Simply computing compositions independently at each time step, may lead to a noisy virtual camera motion. As discussed in previous section, a steady camera behavior is necessary for a pleasant viewing experience. Also, long static camera segments are favorable for the purpose of cutting. To obtain the desired static camera behavior we introduce an L1-norm regularization term over the first order derivative. When L1-norm term is added to the objective to be minimized, or constrained, the solution typically has the argument of the L1-norm term sparse (i.e., with many exactly zero elements). Hence, adding L1-norm term to the velocity will tend to give piecewise constant segments combined with fast transitions. This will filter out the noisy camera motion.

The term is defined as follows:

$$L_{11}(\xi) = \sum_{t=1}^{N-1}(|fx_{t+1} - fx_t| + |fy_{t+1} - fy_t| + |fs_{t+1} - fs_t|). \tag{7.4}$$

This is illustrated in Figure 7.7 with a synthetic one dimensional signal. This signal can be interpreted as the x coordinate of cropping window computed based on the inclusion region derived from noisy actor tracks. The middle plot in Figure 7.7 shows the optimized signal minimizing the closeness term to original signal (shot size penalty in one dimension) with L1-norm regularization on velocity term. We can observe that adding the L1-norm on velocity tends to give piecewise constant segments(with exactly zero motion). Using a more common L2 norm tends to spread the movement over many frames, leading to continual drifting motions, rather than distinct periods of zero movement.

7.3.4 Third order L1-norm regularization

When the camera moves it should move smoothly. The camera movement should start with a segment of constant acceleration (ease-in) and should end with a segment of constant deceleration (ease-out). Using only L1-norm on velocity will lead to sudden start and stop of the camera (sharp corners in middle plot of Figure 7.7). It also leads to a staircase artifact (slopes in middle plot of Figure 7.7). Previous work [GKE11] on camera stabilization has shown that a combination of first order L1-norm regularization with higher order L1-norm regularization on camera coordinates can be used in an optimization framework to obtain smooth camera trajectories with jerk free transitions between static segments. In the same spirit we introduce a third order L1-regularization term which is defined as follows:

$$\begin{aligned} L_{13}(\xi) = \sum_{t=1}^{N-3} (& |fx_{t+3} - 3fx_{t+2} + 3fx_{t+1} - fx_t| \\ & + |fy_{t+3} - 3fy_{t+2} + 3fy_{t+1} - fy_t| \\ & + |fs_{t+3} - 3fs_{t+2} + 3fs_{t+1} - fs_t|). \end{aligned} \quad (7.5)$$

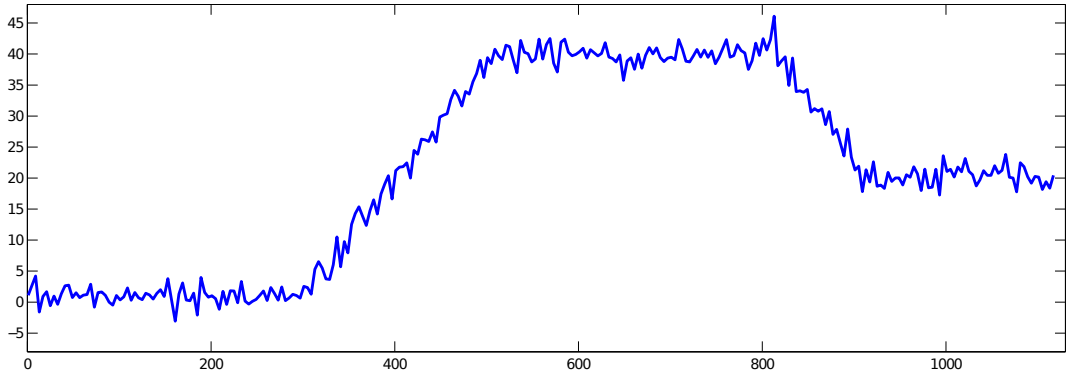
Introducing L1-norm on third order derivative will give jerk free transitions at the start and stop of the camera movement, with segments of constant acceleration and deceleration. We can observe this in bottom plot of Figure 7.7 that using a combination of L1-norm on both velocity and jerk gives the desired camera behavior showing smooth transitions between long piecewise constant segments.

7.3.5 Apparent motion penalty

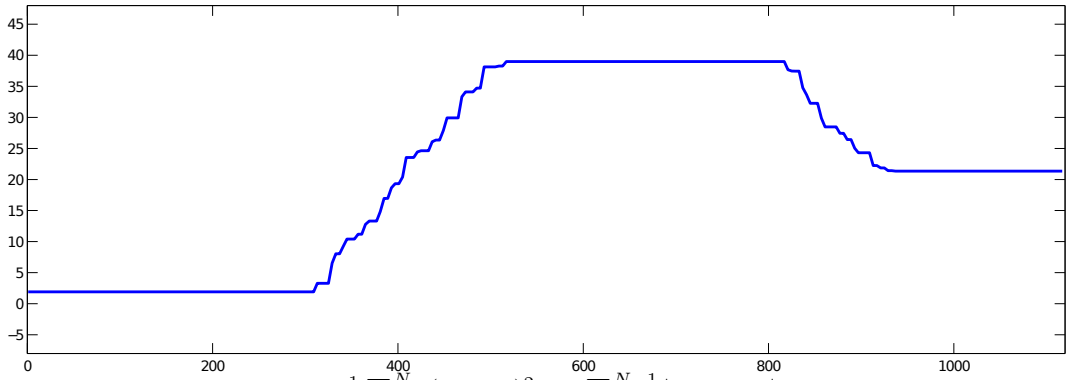
To preserve the sense of activity on stage, the apparent motion of the actors (seen in the cropped window on the virtual camera) should remain consistent with their actual motion. We introduce two different penalty terms to include this in the optimization cost function. The first term penalizes any cropping window motion if the actor included in shot specification is static.

$$M_1(\xi) = \sum_m \sum_{t=1}^{N-1} (cx_t^m \, |fx_{t+1} - fx_t| + cy_t^m \, |fy_{t+1} - fy_t| + cs_t^m \, |fs_{t+1} - fs_t|). \quad (7.6)$$

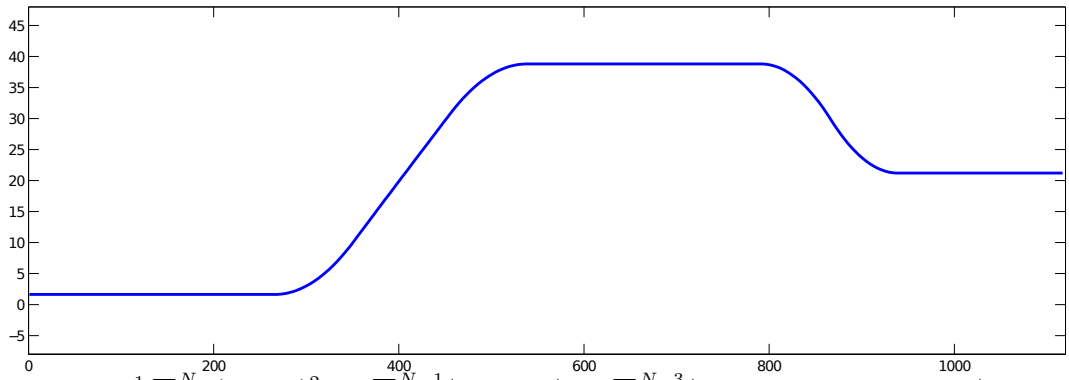
Here cx_t^m, cy_t^m and cs_t^m are pre-stored Boolean values which take a value of 1 if the actor is static in the position and size co-ordinates respectively. For example, cx_t^m is 1 if $(bx_{t+1}^m - bx_t^m)$ is less than a threshold else it is 0, where bx_t^m is x-coordinate of the center of the bounding box of the given actor (m) at time (t). This penalty is added for each actor specified in the shot description. If the actor is static, a penalty equivalent to the cropping window motion is added



Original signal $x_t, t = [1 : N]$



$\min_r \frac{1}{2}\sum_{t=1}^{N}(x_t - r_t)^2 + \lambda \sum_{t=1}^{N-1} |r_{t+1} - r_t|$



$\min_r \frac{1}{2}\sum_{t=1}^{N}(x_t - r_t)^2 + \lambda \sum_{t=1}^{N-1} |r_{t+1} - r_t| + \lambda \sum_{t=1}^{N-3} |r_{t+3} - 3r_{t+2} + 3r_{t+1} - r_t|$

Figure 7.7: *Top: Synthetic one dimensional data x. Middle: Optimized signal r, minimizing the sum of squares closeness term to original data with L1-norm regularization on velocity. Bottom: Optimized signal r, minimizing the sum of squares closeness term to original data with L1-norm regularization on both velocity and jerk.*

to the cost function, else this term is zero.

The second term adds a penalty if the direction of apparent motion is not preserved:

$$M_2(\xi) = \sum_m \sum_{t=1}^{N-1} (max(0, -(\dot{bx}_t^m - \dot{fx}_t)\dot{bx}_t^m) + max(0, -(\dot{by}_t^m - \dot{fy}_t)\dot{by}_t^m) + max(0, -(\dot{bs}_t^m - \dot{fs}_t)\dot{bs}_t^m)). \quad (7.7)$$

Here, $\dot{bx}_t^m = (bx_{t+1}^m - bx_t^m)$ gives the the actual horizontal motion of the actor on stage, $\dot{fx}_t^m = (fx_{t+1}^m - fx_t^m)$ is the horizontal motion of the virtual camera cropping window and $(\dot{bx}_t^m - \dot{fx}_t)$ is the apparent motion of the actor inside the virtual camera cropping window between consecutive time instants. The term $(\dot{bx}_t^m - \dot{fx}_t)\dot{bx}_t^m$ is positive if the apparent direction of motion is same as the actual direction of motion on the stage. A penalty is added if the term is negative, otherwise the penalty is zero. This is summed over the set of actors included in the shot description.

7.3.6 Pull-in or keep-out penalty

To avoid being cut by the frame, each actor must either be in or out of the virtual camera window. For actors included in the shot description, this is ensured by a hard constraint on the inclusion region. But other actors may still come in contact with the virtual camera frame (if they come in close vicinity of the inclusion region or cross across it). So we would like to penalize any instance where the rightmost coordinate of the cropping window $fx_t + A_r fs_t$ comes within the right external actor boundaries xr'_t and xr''_t (please refer to Figure 7.6). Similarly, we would like to add a penalty if the leftmost coordinate of the cropping window $fx_t - A_r fs_t$ comes within the left external actor boundaries xl'_t and xl''_t (please refer to Figure 7.6). But such a conjunction is not convex (each of the statement above needs to be defined as multiplication of two independent convex statements, which in turn is not convex).

To approximate this within the convex framework, we use a heuristic that pre-computes Boolean vectors tl and tr which take a value of 1 if a touch event occurs from the left or the right respectively or they take a value of zero. A touch event occurs if an outside actor comes in close vicinity of the inclusion region for a given shot specification. Using these two vectors, we define two separate penalty terms $E_{keepout}$ and E_{pullin}. The $E_{keepout}$ penalty is only applied when no touch event is occurring on the left or the right inclusion regions. It is defined as follows:

$$E_{keepout}(\xi) = \sum_{t=1}^{N} ((\sim tl_t)max(0, xl'_t - fx_t + A_r fs_t) + (\sim tr_t)max(0, fx_t + A_r fs_t - xr'_t)). \quad (7.8)$$

When no touch event occurs any instance of the cropping window frame touching the closest external actor on the left or the right is penalized. For example, if the right edge of the cropping window $fx_t + A_r fs_t$ is greater than xr'_t, an penalty of $fx_t + A_r fs_t - xr'_t$ is added, otherwise the penalty is zero (an example is illustrated in Figure 7.8). Similarly, the penalty is also defined for left edge. The no touch event in Equation 7.8 is defined as the logical not ($\sim$) of the left and the right touch vectors tl_t and tr_t.

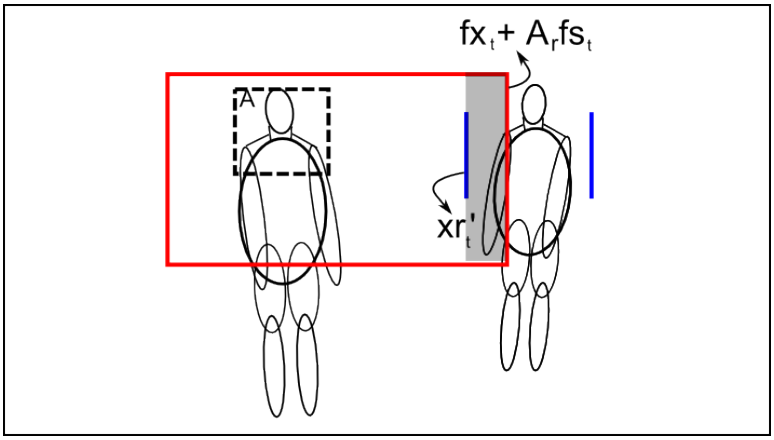


Figure 7.8: *Illustration of **keep out** penalty for shot specification "MS A". It penalizes any instance when an actor not included in the shot specification is chopped by the virtual camera frame (and it is not close enough to be included in the virtual camera frame). This figure illustrates an example where an external actor is chopped on the right side of the virtual camera frame. A cost equal to the extent of overlap (shaded region) is added to the objective function, which is equal to* $(fx_t + A_r fs_t - xr'_t)$ *in this case. By minimizing the overlap, the cost function tries to **keep out** the external actor.*

When a touch event occurs, the penalty term $E_{keepout}$ switches to E_{pullin}, which is defined as follows:

$$E_{pullin}(\xi) = \sum_{t=1}^{N} (tl_t \; max(0, fx_t - A_r fs_t - xl''_t) + tr_t \; max(0, xr''_t - fx_t - A_r fs_t)). \quad (7.9)$$

Here, xr'' and xl'' denote the leftmost and rightmost coordinate of the upper body bounding box of an outside actor (not included in shot specification) touching from the left or the right side respectively. For example, if an outside actor is touching from the right, the rightmost coordinate of the cropping window $fx_t + A_r fs_t$ should be greater than the rightmost coordinate of the tracking window of the touching actor xr''_t, otherwise a penalty of $(xr''_t - fx_t - A_r fs_t)$ is added to the cost function (an example is illustrated in Figure 7.9).

7.3.7 Energy minimization

Overall the problem of finding the virtual camera trajectory given the actor bounding boxes and the shot specification, can simply be summarized as a problem of minimizing a convex cost function with linear constraints. Which is defined as follows:

$$\begin{aligned} \underset{fx,fy,fs}{\text{minimize}} \; & (D(\xi) + \lambda_1 L_{11}(\xi) + \lambda_2 L_{13}(\xi) + \lambda_3 E_{keepout}(\xi) \\ & + \lambda_4 E_{pullin}(\xi) + \lambda_5 M_1(\xi) + \lambda_6 M_2(\xi)) \\ \text{subject to} \quad & \\ & 0 \leq fx_t - A_r fs_t \leq xl_t, \\ & xr_t \leq fx_t + A_r fs_t \leq W, \\ & 0 \leq fy_t - fs_t \leq yu_t, \\ & yb_t \leq fy_t + fs_t \leq H, t = 1, \ldots, N. \end{aligned} \quad (7.10)$$

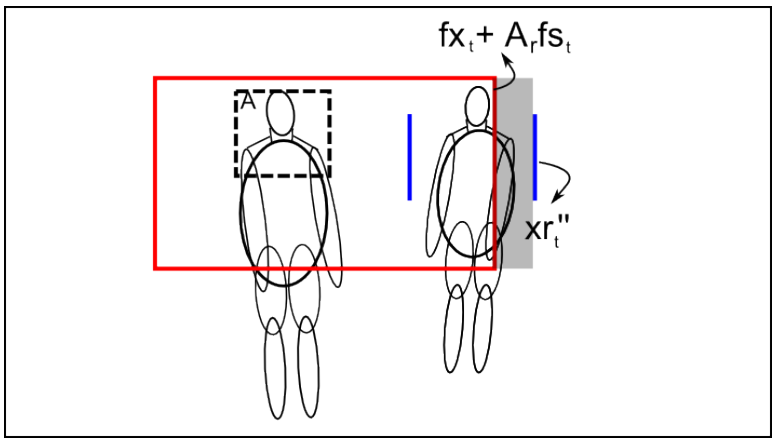


Figure 7.9: *Illustration of **pull in** penalty for shot specification "MS A". It penalizes any instance when an actor not included in the shot specification is chopped by the virtual camera frame (and it is close enough to be included in the virtual camera frame). This figure illustrates an example where an external actor is chopped on the right side of the virtual camera frame. A cost equal to the part not included (shaded region) in the virtual camera frame is added to the objective function, which is equal to* $(xr''_t - fx_t - A_r fs_t)$ *in this case. By minimizing the missing part in the virtual camera frame, the cost function tries to **pull in** the external actor.*

Here, λ_1, λ_2, λ_3, λ_4, λ_5 and λ_6 are parameters. They can be adjusted to control the amount of regularization and the weight of each penalty term. In this paper, we use only two parameters with $(\lambda_1 = \lambda_2)$ and $(\lambda_3 = \lambda_4 = \lambda_5 = \lambda_6)$, giving a similar preference to each penalty term. But this can be adjusted in special cases where higher preference may be required for a specific penalty term. One major advantage of our method is that any standard off the shelf convex optimization toolbox can be used to solve Equation 8.1. In our case we use cvx [GB14].

7.4 RESULTS

We present results on five different sequences (each in Full HD 1920×1080) from the play '*Death of a Salesman*'. Those sequences were chosen from scenes with two, three and four actors to demonstrate the versatility of our approach.

For each of these master shots, we generate a variety of reframed sequences with different shot specifications. The reframed sequences are generated with a resolution of (640×360), maintaining the original $16 : 9$ aspect ratio. These generated sequences can be directly imported and edited in a standard video editing software as a multi-clip. Figure 7.10 shows example of a multi-clip sequence consisting of the original sequence (master shot) and the three reframed sequences generated using our method. On average, the optimization took around 3-4 seconds per rush per minute. All the original videos and generated rushes are available online[1].

Qualitative evaluation

The results on two different sequences are shown in Figure 7.11 and Figure 7.12. Each figure shows a few selected keyframes from the original video and the corresponding frames from the

[1] https://team.inria.fr/imagine/vgandhi/cvmp_2014/

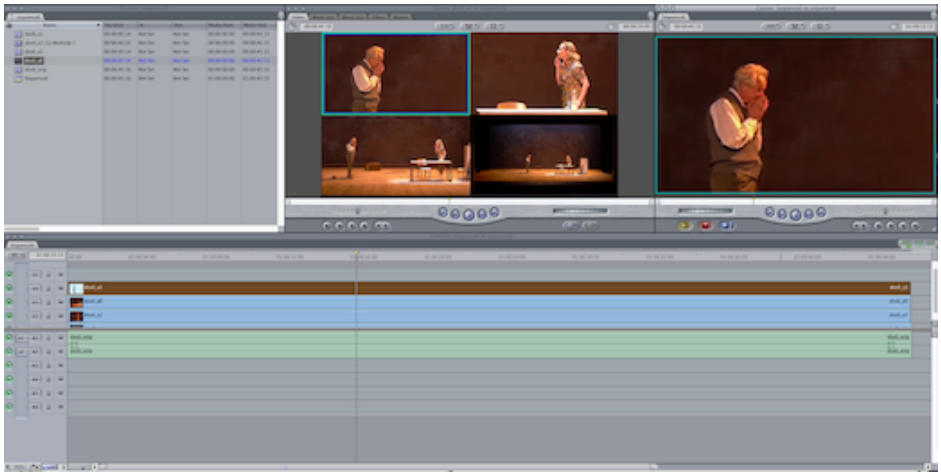


Figure 7.10: *Screenshot of a multiclip editing session in Final Cut Pro. The multi-clip was created using three reframed sequences (MS A, MS B, FS All) and the original master shot. On the right, we see the edited sequence.*

virtual camera sequences generated using our method. A plot of the horizontal position f_x of the virtual camera trajectory against time is shown for each of the generated sequences. The generated sequences allow the editor to highlight details which may be not be so easy to notice in the original sequence. Also, it provides much more variety to keep the viewer interested. Now we discuss the generated sequences on three important aspects of cinematography:

Composition

We can observe that the virtual cameras maintain a nice composition based on the shot specification. For example, the virtual cameras "MS A" and "MS B" in Figure 7.11 keep a stable medium shot of both actors avoiding the actors to come in contact with the image frame. The generated shot also preserves the screen continuity, for example the camera "MS B" keeps the actor B at 1/3 right as she is positioned on the right side of the stage. Similarly, the camera "MS A" keeps actor A on 1/3rd left as he enters from the left. Another example can be seen with camera "MS B" in Figure 7.12, where the camera keeps the actor in the center as it stays between two other actors on stage.

The virtual cameras also avoid cropping the actors not mentioned in shot specification. For example, the camera "MS B" in Figure 7.11 pulls in actor A when it comes close to actor B at keyframe 6. Similar example can be seen with camera "FS B,C" in Figure 7.12, which maintains a tight full shot of actors B and C but pulls in actor A when it comes close to the camera frame.

Camera motion

The plots of f_x in Figure 7.11 and Figure 7.12 show that the virtual camera path smoothly transitions between long static segments. Observe how the virtual camera remains static for long period between keyframes 4 to 5 and keyframes 6 to 7 in Figure 7.11 as the actors do not move significantly. When the camera moves, it moves smoothly preserving the apparent

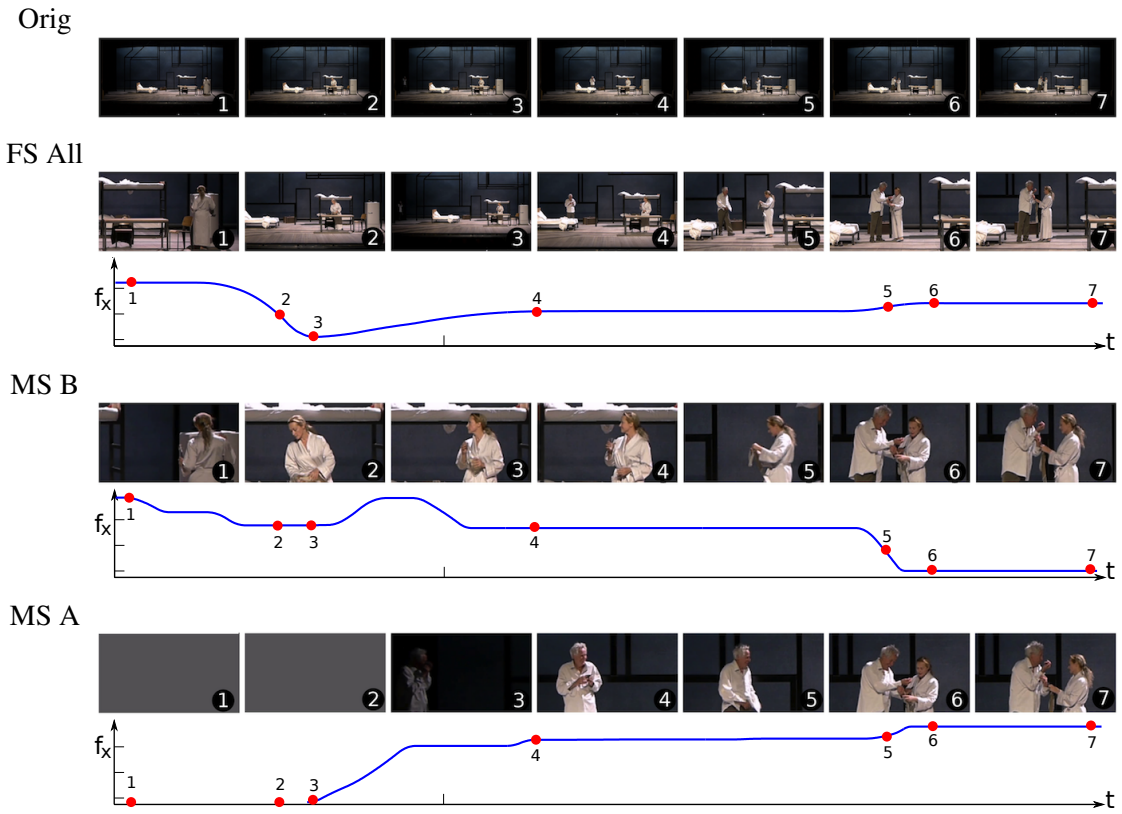


Figure 7.11: *Reframing results on a sequence with two actors (A,B). The top row shows a set of selected keyframes from the original video. The corresponding keyframes from the three different virtual camera sequences are shown below. The three reframed sequences include the medium shot of each actor (MS A, MS B) and a full shot of both the actors (FS All). A plot of the horizontal position f_x of the virtual camera trajectory against time is shown for each of the three reframed sequences. The position of the keyframes on the plot is marked with red dots.*

motion of the actors on stage. For example, observe how the camera "MS A" in Figure 7.11 moves to the right as the actor A enters the stage between keyframes 3 and 4.

Cuttability

Good composition, screen continuity and long static cameras in the generated virtual camera sequence provide the editor plenty of choices to cut. For example the editor can switch among all four possibilities (including the original) at keyframe 4 and 5 in Figure 7.11. Similarly, the editor can switch among all five options at keyframe 1 in Figure 7.12. In some cases the generated virtual cameras may not be cuttable, for example cutting between camera "MS A" and "MS B" at keyframe 6 in Figure 7.11 would create a jump cut. This happens because, due to the pull in event both cameras end up framing the same actors with slightly different compositions. In some cases, the virtual camera framing comes too close to the framing of the original master shot and cutting between them may lead to a jump cut. An example of this can be seen in keyframe 3 of camera "FS All" in Figure 7.11.

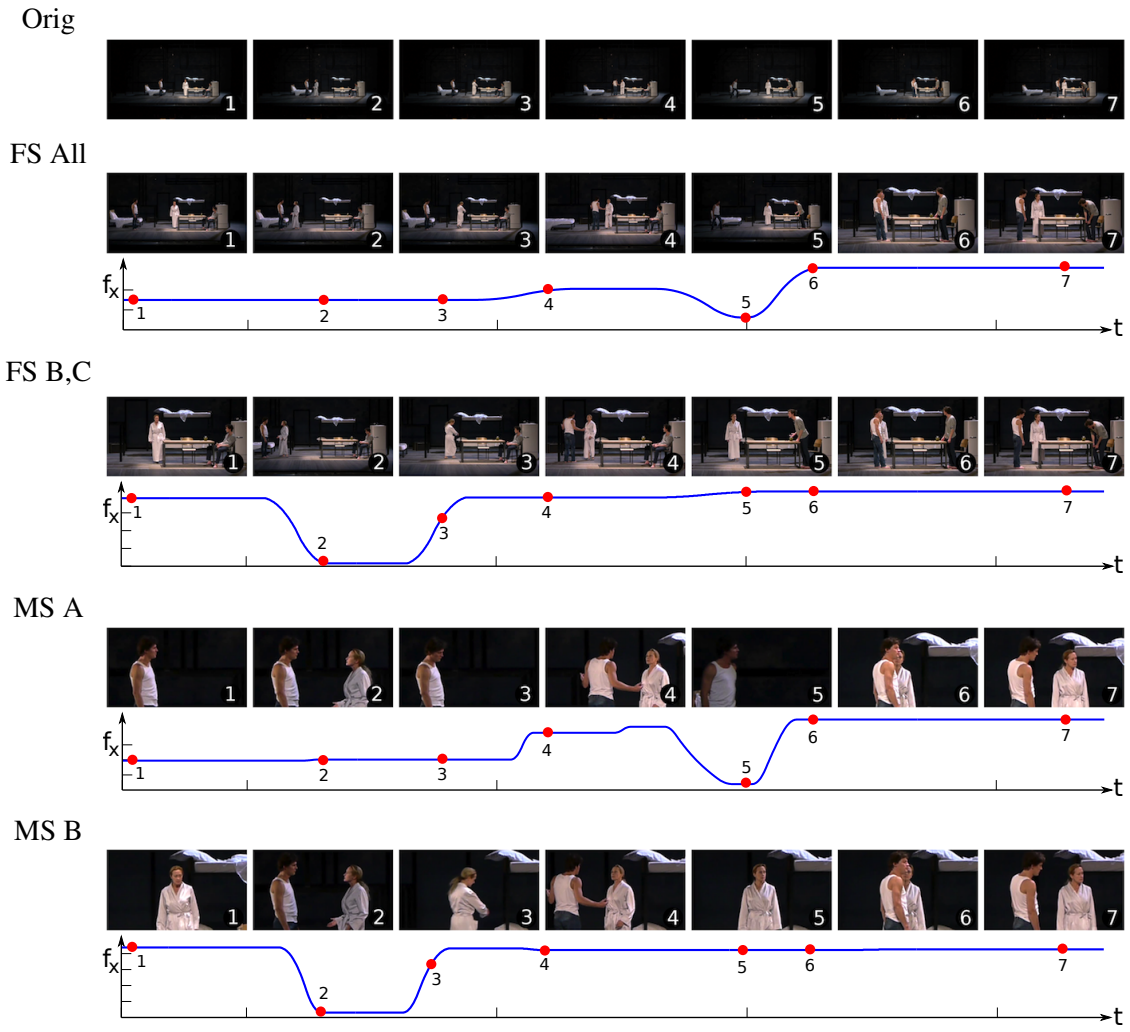


Figure 7.12: *Reframing results on a sequence with three actors (A,B,C). Selected keyframes from the original sequence and 4 virtual camera sequences are shown in this figure. The four virtual camera sequences include the full shot of all three actors (FS All), full shot of actors two actor (FS B,C) and medium shots of two of the actors (MS A, MS B). A plot of the horizontal position f_x of the virtual camera trajectory against time is shown for each of the four reframed sequences. The position of the selected keyframes on the plot is marked with red dots.*

7.5 SUMMARY

We have presented a system which can generate multiple reframed sequences from a single viewpoint taking into consideration the composition, camera movement and cutting aspects of cinematography. We have cast the problem of rush generation as a convex minimization problem and demonstrated qualitatively correct results in a variety of situations. To our knowledge this is the first time that the problem of rush generation has been addressed and validated experimentally. In effect, our method provides a cost-effective solution for multi-clip video editing from a single viewpoint.

Currently in our system, optimization is performed separately for each given shot specification. This may lead to jump cuts in few cases as discussed in previous section. Future work should investigate a joint optimization for the set of given shot specifications. The proposed work focuses on framing actors present on stage but does not allow to include objects in the shot specification. In future work, it would be interesting to integrate some simple objects in the shot naming conventions using standard objects detectors. It would also be interesting to perform further evaluation/validation based on a quantitative metric or user questionnaires.

The Full HD master shots used in the experiments did not provide us enough resolution to go closer than medium shots. But the method can be easily applied to master shots with higher resolutions ($4K$ or $6K$), which will allow to extend the range of shots to medium close-ups (MCU) and close-ups (CU). The reframed rushes obtained with our method are automatically annotated with actor and camera movements which makes them suitable for automatic editing. One of the promising direction of future research is to investigate the problem of automatic camera selection given the rushes.

The algorithm described in this chapter could become a plugin for Final Cut Pro (or Premiere or After Effect or Nuke or Open source Natron). A patent is being filed by INRIA (with Gandhi, Ronfard and Gleicher as co-inventors). The algorithm can also be used for role based reframing in virtual cinematography systems [GRCS14].

CHAPTER

8

APPLICATIONS AND PERSPECTIVES

In this chapter we present some additional direct applications of the framework proposed in the previous chapters. We will introduce a method for floor plan view reconstruction of the theatre stage. We will also describe how split screen compositions can be automatically generated using the actor tracks. We describe a PSL based approach for rush generation. We introduce the idea of shot graph and give perspectives on how it can be used for automatic editing. In the end we list some of the open issues and conclude this dissertation.

8.1 Floor plan view reconstruction

One of the important aspects in theatre direction is 'blocking', which refers to the precise movement and positioning of actors on a stage in order to facilitate the performance of a play. In contemporary theatre, the director usually determines blocking during rehearsal, telling actors where they should move for the proper dramatic effect, ensure sight lines for the audience and work with the lighting design of the scene. The term comes from 19th century theatre, where directors worked out the staging of a scene on a miniature stage using blocks to represent each of the actors.

During the rehearsals, usually the assistant director or the stage manager (or both) take notes about where actors are positioned and their movement patterns on stage. These notes are often made in form of markings (blocking notations) on the stage floor plan view. This is important to ensure that actors follow the assigned blocking from night to night [Spo85, Sch97]. To replace this tedious task of manual markings for numerous rehearsals, we present a system

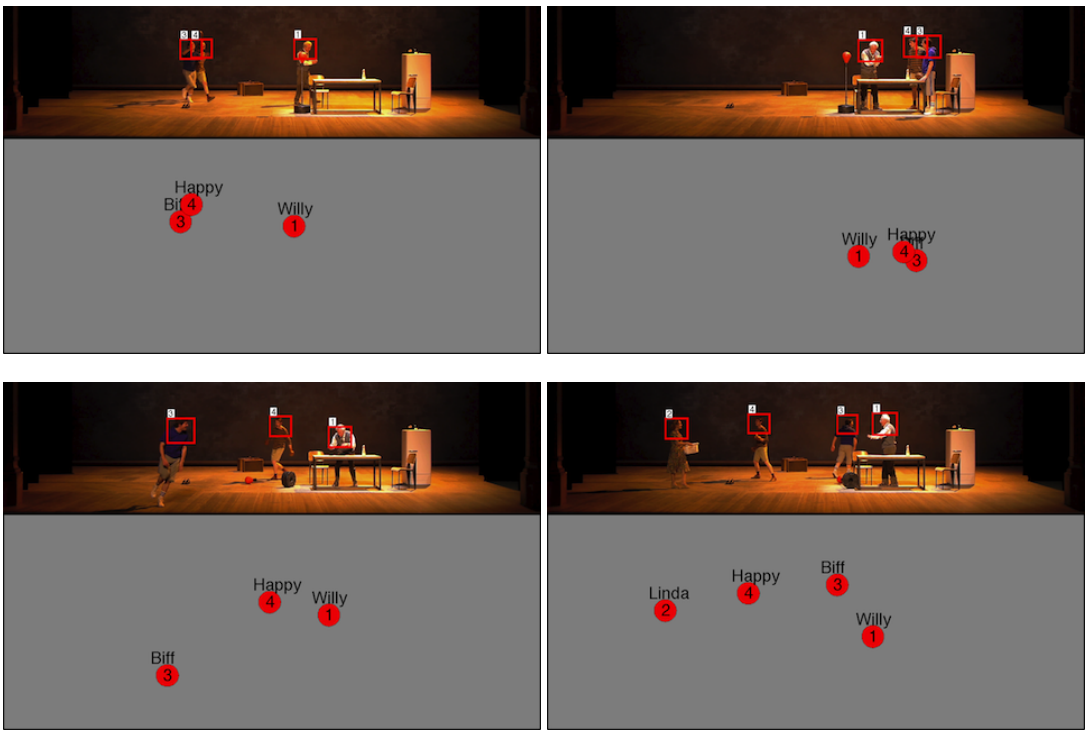


Figure 8.1: *Illustration of floor plan view reconstruction for four different frames. The gray part in each image shows the top view of the stage floor and the corresponding actor positions.*

(a) (b) (c)

Figure 8.2: *Usage of split screen compositions. (a) An example from TV debate. (b) An example from BBC sitcom Coupling. (c) An example from live theatre*[2].

for automatic floor plan view reconstruction which can be directly used to derive blocking notations from video recordings.

The application is illustrated in Figure 8.1. We use the tracking algorithm presented in Chapter 6 for estimating the floor point in two dimensions. The floor projections are calculated using the approach described in Section 7.2.1. We then use a vanishing points based approach [GMMB00] for computing a mapping of the stage floor in image coordinates to a plane in real world coordinates. The benefit of using the approach by Guillou et al. [GMMB00] is that it allows to recover the real world geometry of the stage floor from only a single image with a minimal user input i.e. two pairs of manually labeled parallel lines. The parallel lines are then used to automatically compute the vanishing points.

Floor plan view reconstruction on an example sequence using our approach is illustrated online[1]. Quantitative validation is left for the future work.

8.2 SPLIT SCREEN COMPOSITIONS

Split screen is a visible division of the screen into multiple equal/unequal parts. It has been commonly used for debates, sitcoms or movies. Some of the example cases are illustrated in Figure 8.2. The split screen is useful to create interesting effects like conveying emotions of two or more actors simultaneously using closer shots instead of switching between them using cuts. It has also found usages in production of live theatre. A recent example from *Theatre sans animaux* (illustrated in Figure 8.2(c)) shows that it can be used to create humorous effects. The split screen in the case of *Theatre sans animaux* was created manually, requiring a lot of effort and time of the editor.

We present an approach to automatically create split screen compositions using the actor detections. Example images from the split screen video of *Dos1* sequence with four actors are shown in Figure 8.3, Figure 8.4 and Figure 8.5 for medium close up (MS), medium shot (MCU) and the full shot (FS) respectively. Each of these figures illustrates the cropped horizontal strip of the original video combined with the split screen video generated using our approach. Our method assigns a separate partition for each actor and the partitions are merged together if the actors are not sufficiently far apart to be framed separately. A blank partition is left for the

[1] https://team.inria.fr/imagine/vgandhi/applications/

[2] http://nouvelles-ecritures.francetv.fr/theatre-sans-animaux/piece-enrichie/

Figure 8.3: *A full shot (FS) split screen composition on Dos1 sequence with four actors. The top partition of the each image shows the original video and the bottom part shows the generated partitions for each actor. The actors offstage are assigned an empty gray partition. Overlapping partitions are merged together to form a larger partition. The order of partitions is the same as the order of the actors on stage (for example rightmost actor is assigned the rightmost partition).*

Figure 8.4: *Similar results as Figure 8.3 but with medium shot (MS).*

Figure 8.5: *Similar results as Figure 8.3 but with medium close up (MCU).*

offstage actors. All the three splitscreen videos (corresponding to Figure 8.3, Figure 8.4 and Figure 8.5) are available online[3].

The split screen composition in our current implementation is computed in four steps:

- Divide the width of the video by the maximum number of actors in the scene to compute the width of each partition in the split screen composition. Figure 8.3 shows an example with four partitions on a full HD video (1920×1080), where the width of each partition is set to 480 pixels.
- The height of each partition is then computed using the given aspect ratio. In the current implementation the height is computed in such a way that the height of the video equals 1080 pixels when combined with the cropped strip of the original video.
- Given the desired aspect ratio, shot size and the actor bounding boxes (bx_t^m, by_t^m, bs_t^m, bh_t^m), our method then computes the cropping window for each actor independently for each frame. The cropping window is horizontally centered at bx_t^m and the length is computed using the shot size.
- An iterative merging algorithm is then applied on each frame individually, if the cropping windows overlap. The cropping windows with higher overlap are merged first and the process continues until there is no overlap. The merging step computes the new cropping window considering the union of actor bounding boxes as a single box (with adjusted partition width).

[3] https://team.inria.fr/imagine/vgandhi/applications/

Since the partitions are computed at each frame independently, the resulting split screen compositions may not be smooth. To avoid jitter we regularize the actor bounding boxes prior to computing the split screen compositions.

8.3 RUSH GENERATION USING PSL

In Chapter 7 we have presented a method for generating rushes from a single viewpoint given the actor tracks and a shot specification. The results presented in Chapter 7 are limited to the "*pan with*" shots, where the shot specification remains the same for the entire video (the cropping window is optimized to maintain the desired composition for the entire video). We now describe two other shot types which can be generated by our method with suitable extensions to the optimization framework proposed in the previous chapter.

Lock with

The goal here is to find the best static frame containing a given set of actors in a given type of composition for the given time interval. The optimization problem proposed in Section 7.3.7 reduces to following:

$$\begin{aligned}
&\underset{fx,fy,fs}{\text{minimize}}\ (D(fx,fy,fs)+\lambda_3 E_{keepout}(fx,fy,fs)+\lambda_4 E_{pullin}(fx,fy,fs))\\
&\text{subject to}\\
&\qquad 0 \le fx - A_r fs \le xl_t,\\
&\qquad xr_t \le fx + A_r fs \le W,\\
&\qquad 0 \le fy - fs \le yu_t,\\
&\qquad yb_t \le fy + fs \le H, t = 1,\ldots,N.
\end{aligned} \tag{8.1}$$

The shot size penalty $D(\xi)$ also simplifies to the following:

$$D(fx,fy,fs) = \frac{1}{2}\sum_{t=1}^{N}((fx-gx_t)^2+(fy-gy_t)^2+(fs-gs_t)^2). \tag{8.2}$$

The difference with the original term described in 7.3.2 is that now the optimization searches for values (fx, fy, fs) which are constant for the entire sequence. Similar modifications are made in the $E_{keepout}$ and E_{pullin} penalty. Overall, the modified optimization function now searches for a constant framing close to the inclusion constraints, avoiding chopping actors which are not included in the shot specification.

Pan to

The aim here is to smoothly transition from one composition to another composition using virtual pan and zoom. The algorithm for computing "*pan to*" shots takes as input the "shot size" and "set of actors" both at the beginning and at the end of the shot. Additionally, the duration of the shot is also specified by the user. The "*pan to*" shots are computed using exactly the same optimization framework proposed in Section 7.3.7, only the constraints and boundaries are computed in a different manner. In our current implementation the "*pan to*" shots are computed in three steps:

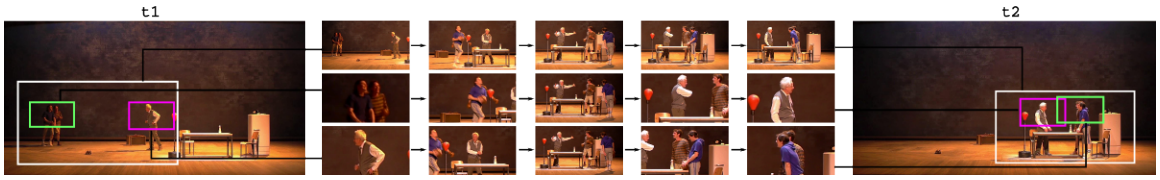


Figure 8.6: *Three of the many possible shots which can be generated by interpolation between compositions at two different time intervals. The first is a "pan with" shot with a full shot of all three actors. The second is a combination of two different "pan to" shots, where the camera first pans from a medium shot of Happy and Biff to a full shot of three actors and then pans to a medium shot of Willy. The third is also a combination of two different "pan to" shots, where the camera first pans from a medium shot of Willy to a full shot of three actors and then pans to a medium shot of Biff and Happy.*

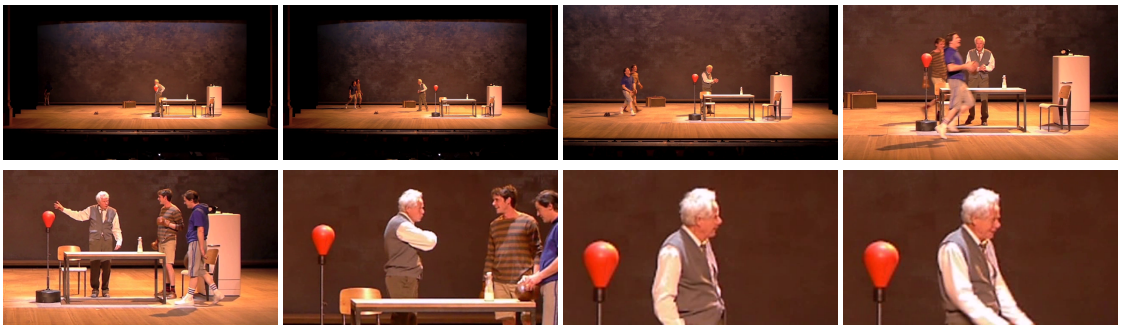
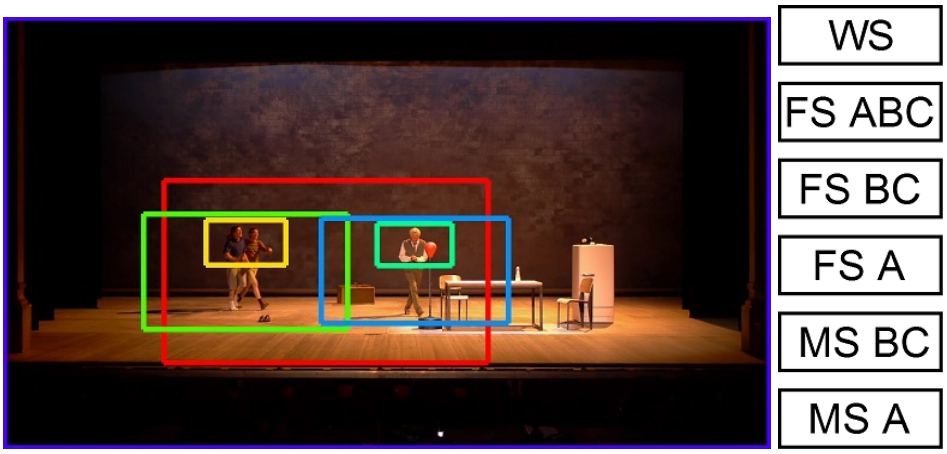


Figure 8.7: *Rush generation using PSL. The shot begins with a wide master shot (establishing shot), then pans to full shot on all three actors and then pans to medium shot of Willy and then maintains the medium shot on Willy. The PSL statement used in this case:* ***At t1 lock with WS then at t2 pan to FS on Willy and Biff and Happy then at t3 pan to MS on Willy then at t4 pan with MS on Willy****.*

- Divide the transition time into three equal segments. Assign the first segment to the starting composition, second segment to transition period and third segment to ending composition. For example, for a transition from "MS Willy" to "FS Willy, Biff, Happy" in six seconds, we assign two seconds to "MS Willy", two seconds for transition period and two seconds to "FS Willy, Biff, Happy".
- Calculate the inclusion region and the boundaries of the external actors independently for the first and the third segments.
- Optimize for the shot using Equation 8.1, where the terms $D(\xi)$, $E_{keepout}(\xi)$, $E_{pullin}(\xi)$, $M_1(\xi)$ and $M_2(\xi)$ and the constraints are only applied for the first and the third segments. Only the regularization terms $L_{11}(\xi)$ and $L_{13}(\xi)$ are applied in the transition segment.

Now rushes can be generated using multiple PSL statements combining "lock with", "pan with" and "pan to" shots. Figure 8.6 illustrates how different shots can be generated by interpolating between compositions at two different time intervals. Customized rushes for a long video can be then generated using multiple consecutive PSL statements as illustrated in Figure 8.7.



(a) (b)

Figure 8.8: *(a) Given any video frame and the actor bounding boxes our method automatically generates all possible shots in it. An example frame from Dos1 sequence with three actors is shown in this figure. A total of six shot specifications are possible in this frame including the original wide shot (limiting to two shots sizes, the medium shot and the full shot). (b) The list of possible shots. We can observe that the shots which are not feasible are automatically rejected by our method (for example, it rejects individual medium shots for actor B and C and proposes a combined medium shot).*

8.4 CONTENT AWARE INTERPOLATION

Given an image and actor bounding boxes, our method automatically computes all possible compositions in it with different PSL names. It is illustrated in Figure 8.8. The possible compositions over time are then connected to form the "shot graph" (illustrated in Figure 8.9). Each node in this graph represents a particular composition at a given time and the edges represent the possible transitions at the next time step.

Transitions between compositions can be caused by lens movements. For example, zooming-in can push an actor out of the frame. Zooming out can pull an actor into the frame. Transitions can also be caused by pan-and-tilt movements. For example, a left-to-right pan shot will reveal actors entering the frame from the right and hide actors exiting the frame on the left. We combine the effects of the lens and the pan-tilt head into camera edges. Finally, transitions between compositions can be caused by actor movement even with a fixed camera. Typical events are actors entering and exiting the frame, changing positions within the frame, hiding each other or showing up from hiding.

At any given time, a particular composition in the shot graph can have edges corresponding to the following events:

- The set of actors and composition remains the same (for example an edge from FS ABC to FS ABC between consecutive time intervals)

- The shot size decreases keeping the same set of actors (for example an edge from FS A to MS A in Figure 8.8 between t2 and t3)

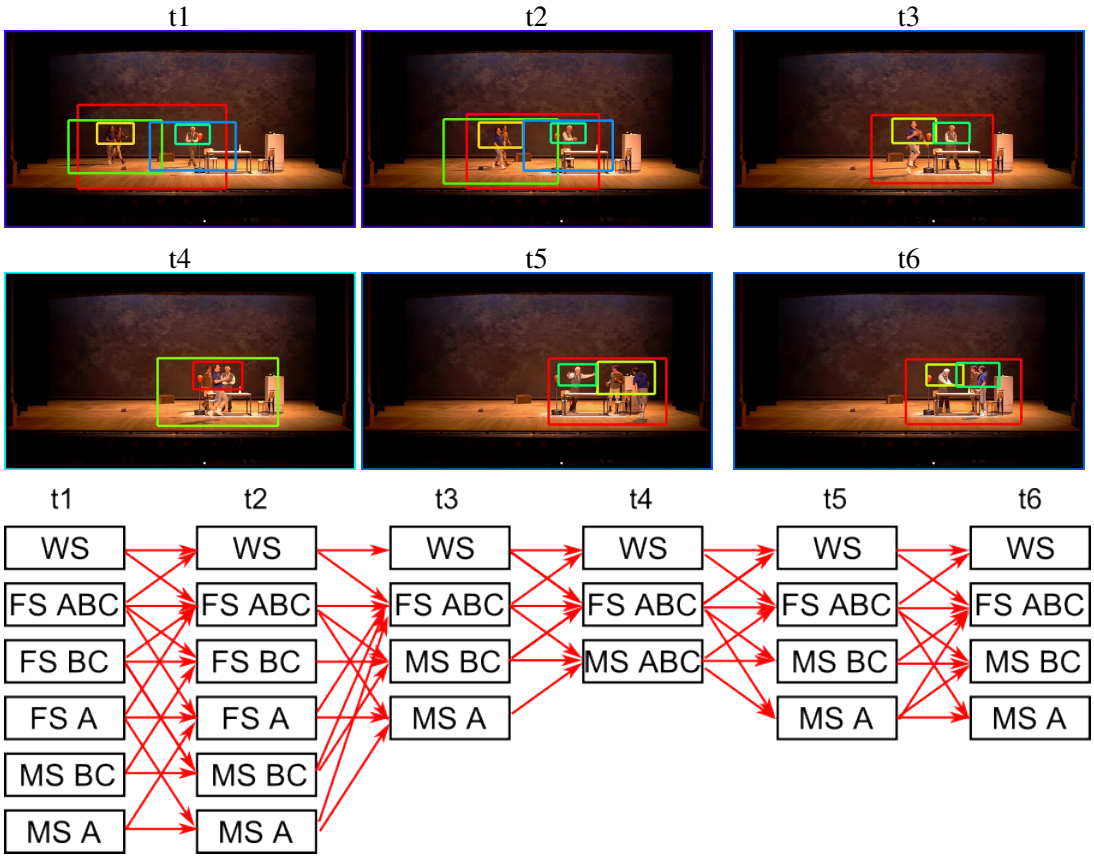


Figure 8.9: *Shot graph. Top two rows: images from Dos1 sequence and possible framings in each case. Last row: different framings over time are connected to form the shot graph. Only selected images from the sequence are shown for the visualization purposes.*

- The shot size increases keeping the same set of actors (for example an edge from MS A to FS A in Figure 8.8 between t1 and t2)
- The actor/actors from the left move out of the frame (for example an edge from FS ABC to FS A in Figure 8.8 between t1 and t2)
- The actor/actors from the right move out of the frame (for example an edge from FS ABC to FS BC in Figure 8.8 between t1 and t2)
- The actor/actors from the left moves into the frame (for example an edge from FS A to FS ABC in Figure 8.8 between t1 and t2)
- The actor/actors from the right moves into the frame (for example an edge from FS BC to FS ABC in Figure 8.8 between t1 and t2)

A composition can directly transition only to the next smallest composition, for example in Figure 8.9 there is no direct edge between FS ABC at t1 and MS A at t2 and the transition has to be made through FS A. While there is an edge between FS ABC at t5 to MS A at t6 because FS A is not feasible at t5. Similarly, a framing can directly transition only to next largest composition. Each edge or the transition between different compositions is caused either due to virtual camera movement or the actor movement (abrupt transitions like cuts are not

Figure 8.10: *Shot graph based interpolation vs linear interpolation. The figure illustrates the two different approaches to interpolate between the "FS Madeline, Adel" to "MS Anca". First row shows the starting and ending keyframes with the given bounding boxes. Second row shows the intermediate frames for a linear interpolation, we can observe how this leads to blank virtual camera frames. Third row illustrates the intermediate frames while interpolating using the shot graph, we can observe that the virtual camera first transitions to "FS Adel, Anca" and then transitions to "MS Anca", avoiding any blank spaces.*

considered in the shot graph).

Moving through the shot graph restricts the virtual camera to keeping "good" compositions or to transitions between these compositions. This can be used for content aware interpolation between two compositions at different time intervals. An example is illustrated in Figure 8.10, where we compare a shot graph based interpolation with a linear interpolation. We can observe how linear interpolation can lead to blank spaces (showing just background), while the shot graph based approach performs interpolation in a more intelligent manner.

Our current implementation performs interpolation by minimizing the number of transitions in the shot graph. In case of multiple possibilities with similar number of transitions, one is manually chosen. The points of transition are either manually chosen or are computed by equally dividing the total time between the starting and the ending frame by the number of transitions. Future work should explore how the points of transitions can be computed in a better way. It would also be interesting to study how shot graphs can be used for automatic editing.

8.5 Open issues

In this thesis we have resolved the problem of rush generation, leaving the problem of rush selection for the future work. As the generated rushes are already optimized for composition and

movements and are annotated with actor names and positions, it would be natural to use them for the problem of automatic rush selection. Solving the problem of automatic rush selection solely based on visual cues is extremely difficult but additional cues from a joint analysis of the audio and the script can make the task easier. For example, the audio alignments can be used to identify who is speaking to whom, which can serve as an important information for automatic editing. Other important factors which may be taken into consideration are the actions happening on the stage (which can be obtained in some cases from the play-script or by action recognition [WRB11, Pop10]); the desired average length of each shot (too long shots do not promote visual interest and too short may be jarring); jump cuts (should avoid cutting to too similar shots) etc. One direction to alternate between shots automatically could be to use finite state machines, where the problem could be cast as a sequential [HCS96] or global [HWG07] optimization.

The gaze of the actors can also serve as an important cue for automatic editing as shown in previous work [TOM04]. It is a common practice in computer vision to estimate gaze in single viewpoint video using head pose estimation [MCT09]. Previous work [MJZF11, PPMZR10] has shown that head pose estimation can be used for recognizing human interactions in TV show. Future work should explore how head pose estimation can be used for the task of automatically editing theatre sequences.

Rush generation using multiple static cameras is another interesting direction to extend this work. Not just that recording with multiple angles will provide variety for editing, it will also be useful to improve the actor tracking, as multi camera tracking [FBLF08, KD06] has been shown to be more successful and reliable compared to a single camera approach. In this dissertation, we have introduced generative models for actor detection and tracking. The proposed models are based on color features and hence cannot handle large illumination changes. Future work should explore techniques for color transfer [RAGS01, PKD05] to dynamically adjust the model according to the change in illumination.

Online extensions of both the tracking and the virtual camera simulation is another promising direction of work. Future work should explore how pre-built multi view appearance models can be used for the task of online tracking. This can open up several interesting applications like optimizing sound and lighting on stage automatically based on the actor positions. Current systems like StagetrackerFx[4] for automatic audio control based on positions of the actors on stage, requires the performers to carry special RFID tags (wireless radio tags) to localize their positions. A purely vision based system can eliminate such complexities.

In this dissertation we have only considered virtual pan and tilt camera movements. A future research problem could be to simulate virtual dolly or crane movements, based on existing approaches for image-based three-dimensional video editing [CPW*11, ZDJ*09]. This will require recording the performances from multiple camera rigs to estimate the depth information (as the camera is static, techniques like structure for motion are not applicable in our case). Another interesting problem could be to simulate virtual pan and scan movements in stereoscopic camera.

[4] http://www.tta-sound.com/

8.6 SUMMARY

We have proposed several extensions to the framework of Chapter 7, with possible applications for the preservation and online presentation of cultural heritage. A detailed description of applications viz. floor plan view construction, split screen composition, rush generation using PSL and content aware interpolation has been presented. We have shown some limitations of the proposed work and have offered some possible directions for future work.

CHAPTER

9

CONCLUSION

In this dissertation, we have studied the problem of automatic rush generation from a single viewpoint in theatre performances. We have addressed all major aspects of the problem, right from recording of the sequences to editing of the generated rushes, and have made significant contributions throughout the framework. We have discussed the challenges in capturing live theatre and have introduced a camera setup to efficiently record the theatre performances. A database of theatre performances has been presented, which consists of multiple rehearsals from three different plays produced at Theatre de Celestin, Lyon.

We have described a formal language (PSL) to serve as a high level interface for intelligent cinematography and editing systems. We have demonstrated its usage both for vertical editing and to describe existing movies. Two different generative models for actor and costume specific detections have been presented. We have shown that both color blob detector (CBD) and patch based detector (PBD) significantly improve the detection recall over the generic detectors. An offline tracking algorithm based on the actor and costume specific detectors has been presented with a thorough quantitative and qualitative analysis over long and challenging sequences. We have shown large improvement in tracking performance over the state of the art.

We have presented a framework for generating multiple synchronized sub-clips from a single viewpoint video. We have shown that various cinematographic principles can be modeled as penalties or constraints and the problem of optimizing virtual camera trajectories can be cast as a convex minimization problem. We have presented rush generation results on variety of sequences and have shown that they can be easily imported and edited as multi-clip sessions in standard video editing softwares. In the end, we have presented some additional direct applications of our work.

Parts of this manuscript have been published in peer reviewed conferences. Following is the list of publications:

1. GANDHI V., RONFARD R., GLEICHER M.: Multi-Clip Video Editing from a Single Viewpoint. In *CVMP 2014 - European Conference on Visual Media Production* (2014)

2. GANDHI V., RONFARD R.: Detecting and Naming Actors in Movies using Generative Appearance Models. In *CVPR* (2013)

3. RONFARD R., GANDHI V., BOIRON L.: The prose storyboard language: A tool for annotating and directing movies. In *2nd Workshop on Intelligent Cinematography and Editing part of Foundations of Digital Games-FDG 2013* (2013)

BIBLIOGRAPHY

[AKK06] ARIKI Y., KUBOTA S., KUMANO M.: Automatic production system of soccer sports video by digital camera work based on situation recognition. In *Multimedia, 2006. ISM'06. Eighth IEEE International Symposium on* (2006).

[Alt49] ALTON J.: *Painting With Light*. University of California Press, 1949.

[APS*14] AREV I., PARK H. S., SHEIKH Y., HODGINS J. K., SHAMIR A.: Automatic editing of footage from multiple social cameras. In *ACM Transactions on Graphics (SIGGRAPH)* (2014).

[Ari91] ARIJON D.: *Grammar of the Film Language*. Silman-James Press, 1991.

[ARS06] ADAM A., RIVLIN E., SHIMSHONI I.: Robust fragments-based tracking using the integral histogram. In *CVPR* (2006).

[ARS09] ANDRILUKA M., ROTH S., SCHIELE B.: Pictorial structures revisited: People detection and articulated pose estimation. In *CVPR* (2009).

[AS07] AVIDAN S., SHAMIR A.: Seam carving for content-aware image resizing. *ACM Trans. Graph. 26*, 3 (2007), 10.

[Avi05] AVIDAN S.: Ensemble tracking. In *CVPR* (2005).

[BF06] BUCHANAN A. M., FITZGIBBON A. W.: Interactive feature tracking using K-D trees and dynamic programming. In *CVPR* (2006).

[Bia98] BIANCHI M. H.: Autoauditorium: A fully automatic, multicamera system to televise auditorium presentations. In *Proc. Joint DARPA/NIST workshop on smart spaces technology* (1998).

[Bia04] BIANCHI M.: Automatic video production of lectures using an intelligent and aware environment. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia* (2004).

[BJ98] BLACK M. J., JEPSON A. D.: Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision 26*, 1 (1998), 63–84.

[Bor98] BORDWELL D.: *On the History of Film Style*. Harvard University Press, 1998.

[Bra97] BRAND M.: The "inverse hollywood problem": From video to scripts and storyboards via causal analysis. In *AAAI/IAAI* (1997), pp. 132–137.

[BRL*09] BREITENSTEIN M. D., REICHLIN F., LEIBE B., KOLLER-MEIER E., VAN GOOL L.: Robust tracking-by-detection using a detector confidence particle filter. In *ICCV* (2009).

[BYB09] BABENKO B., YANG M.-H., BELONGIE S.: Visual tracking with online multiple instance learning. In *CVPR* (2009).

[CAwH*96] CHRISTIANSON D. B., ANDERSON S. E., WEI HE L., SALESIN D. H., WELD D. S., COHEN M. F.: Declarative camera control for automatic cinematography. In *AAAI* (1996).

[CC14] CHEN J., CARR P.: Autonomous camera systems: A survey. In *AAAI-14 Workshop on Intelligent Cinematography and Editing* (2014).

[CDV10] CHEN F., DE VLEESCHOUWER C.: Personalized production of basketball videos from multi-sensored data under limited display resolution. *Comput. Vis. Image Underst. 114*, 6 (2010), 667–680.

[CH06] CHAUMETTE F., HUTCHINSON S.: Visual servo control, part i: Basic approaches. *IEEE Robotics and Automation Magazine 13*, 4 (December 2006), 82–90.

[CH07] CHAUMETTE F., HUTCHINSON S.: Visual servo control, part ii: Advanced approaches. *IEEE Robotics and Automation Magazine 14*, 1 (March 2007), 109–118.

[CJMT08] COUR T., JORDAN C., MILTSAKAKI E., TASKAR B.: Movie/script: Alignment and parsing of video and text transcription. In *ECCV* (2008).

[CLL05] COLLINS R. T., LIU Y., LEORDEANU M.: Online selection of discriminative tracking features. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 10 (2005), 1631–1643.

[CLR12] CHRISTIE M., LINO C., RONFARD R.: Film Editing for Third Person Games and Machinima. In *Workshop on Intelligent Cinematography and Editing* (2012).

[CMM13] CARR P., MISTRY M., MATTHEWS I.: Hybrid robotic/virtual pan-tilt-zom cameras for autonomous event recording. In *Proceedings of the 21st ACM International Conference on Multimedia* (2013).

[CON08] CHRISTIE M., OLIVIER P., NORMAND J.-M.: Camera control in computer graphics. *Computer Graphics Forum* (2008).

[CPW*11] CHEN J., PARIS S., WANG J., MATUSIK W., COHEN M., DURAND F.: The video mesh: A data structure for image-based three-dimensional video editing. In *Computational Photography (ICCP), 2011 IEEE International Conference on* (2011), pp. 1–8.

[CWH*13] CHEN C., WANG O., HEINZLE S., CARR P., SMOLIC A., GROSS M.: Computational sports broadcasting: Automated director assistance for live sports. In *Multimedia and Expo (ICME), 2013 IEEE International Conference on* (July 2013), pp. 1–6.

[DDDV09] DELANNAY D., DANHIER N., DE VLEESCHOUWER C.: Detection and recognition of sports(wo)men from multiple views. In *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on* (2009), pp. 1–7.

[DDN08] DESELAERS T., DREUW P., NEY H.: Pan, zoom, scan - time-coherent, trained automatic video cropping. In *CVPR* (2008).

[DGSG04] DOUBEK P., GEYS I., SVOBODA T., GOOL L. V.: Cinematographic rules ap[lied to a camera network. In *Omnivis2004, The fifth Workshop on Omnidirectional Vision, Camera Networks and Non-Classical Cameras* (May 2004), pp. 17–29.

[DMR05] DONY R., MATEER J., ROBINSON J.: Techniques for automated reverse storyboarding. *IEE Journal of Vision, Image and Signal Processing 152*, 4 (2005), 425–436.

[DO04] DAIGO S., OZAWA S.: Automatic pan control system for broadcasting ball games based on audience's face direction. In *Proceedings of the 12th Annual ACM International Conference on Multimedia* (2004), MULTIMEDIA '04, pp. 444–447.

[DT05] DALAL N., TRIGGS B.: Histograms of oriented gradients for human detection. In *CVPR* (2005).

[EF09] EICHNER M., FERRARI V.: Better appearance models for pictorial structures. In *BMVC* (2009).

[ESZ06] EVERINGHAM M., SIVIC J., ZISSERMAN A.: "Hello! My name is... Buffy" – automatic naming of characters in TV video. In *BMVC* (2006).

[FBLF08] FLEURET F., BERCLAZ J., LENGAGNE R., FUA P.: Multicamera people tracking with a probabilistic occupancy map. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 30*, 2 (2008), 267–282.

[FBP*10] FARENZENA M., BAZZANI L., PERINA A., MURINO V., CRISTANI M.: Person re-identification by symmetry-driven accumulation of local features. In *CVPR* (2010).

[FF06] FRIEDMAN D., FELDMAN Y. A.: Automated cinematic reasoning about camera behavior. *Expert Syst. Appl. 30*, 4 (May 2006), 694–704.

[FGMR10] FELZENSZWALB P., GIRSHICK R., MCALLESTER D., RAMANAN D.: Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 32*, 9 (sept. 2010), 1627 –1645.

[FMJZ09] FERRARI V., MARIN-JIMENEZ M., ZISSERMAN A.: Pose search: Retrieving people using their pose. In *CVPR* (2009).

[For07] FORSSEN P.-E.: Maximally stable colour regions for recognition and matching. In *CVPR* (2007).

[FPZ05] FERGUS R., PERONA P., ZISSERMAN A.: A sparse object category model for efficient learning and exhaustive recognition. In *CVPR* (2005).

[GB14] GRANT M., BOYD S.: CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, Mar. 2014.

[GCSS06] GOLDMAN D. B., CURLESS B., SALESIN D., SEITZ S. M.: Schematic storyboarding for video visualization and editing. *ACM Trans. Graph. 25*, 3 (2006), 862–871.

[GKE11] GRUNDMANN M., KWATRA V., ESSA I.: Auto-directed video stabilization with robust l1 optimal camera paths. In *CVPR* (2011).

[GL08] GLEICHER M., LIU F.: Re-cinematography: Improving the camerawork of casual video. *ACM Transactions on Multimedia Computing Communications and Applications (TOMCCAP) 5*, 1 (2008), 1–28.

[GLL*13] GADDAM V. R., LANGSETH R., LJØDAL S., GURDJOS P., CHARVILLAT V., GRIWODZ C., HALVORSEN P.: Interactive zoom and panning from live panoramic video. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop* (2013), NOSSDAV '14.

[GLS*14] GADDAM V. R., LANGSETH R., STENSLAND H. K., GURDJOS P., CHARVILLAT V., GRIWODZ C., JOHANSEN D., HALVORSEN P.: Be your own cameraman: Real-time support for zooming and panning into stored and live panoramic video. In *Proceedings of the 5th ACM Multimedia Systems Conference* (2014), MMSys '14.

[GM00] GLEICHER M., MASANZ J.: Towards virtual videography (poster session). In *ACM Multimedia* (2000).

[GMMB00] GUILLOU E., MENEVEAUX D., MAISEL E., BOUATOUCH K.: Using vanishing points for camera calibration and coarse 3d reconstruction from a single image. *The Visual Computer 16*, 7 (2000), 396–410.

[GR13] GANDHI V., RONFARD R.: Detecting and Naming Actors in Movies using Generative Appearance Models. In *CVPR* (2013).

[GRCS14] GALVANE Q., RONFARD R., CHRISTIE M., SZILAS N.: Narrative-Driven Camera Control for Cinematic Replay of Computer Games. In *Motion In Games* (2014).

[GRG14] GANDHI V., RONFARD R., GLEICHER M.: Multi-Clip Video Editing from a Single Viewpoint. In *CVMP 2014 - European Conference on Visual Media Production* (2014).

[GRLC15] GALVANE Q., RONFARD R., LINO C., CHRISTIE M.: Continuity editing for 3d animation. In *AAAI (under review)* (2015).

[GSCO06] GAL R., SORKINE O., COHEN-OR D.: Feature-aware texturing. In *Proceedings of EUROGRAPHICS Symposium on Rendering* (2006), pp. 297–303.

[GZT11] GU S., ZHENG Y., TOMASI C.: Linear time offline tracking and lower envelope algorithms. In *ICCV* (2011).

[HCS96] HE L.-W., COHEN M. F., SALESIN D. H.: The virtual cinematographer: a paradigm for automatic real-time camera control and directing. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), SIGGRAPH '96, pp. 217–224.

[Hit48] HITCHOCK A.: The rope, 1948.

[HWG07] HECK R., WALLICK M., GLEICHER M.: Virtual videography. *ACM Trans. Multimedia Comput. Commun. Appl. 3*, 1 (2007).

[HWN08] HUANG C., WU B., NEVATIA R.: Robust object tracking by hierarchical association of detection responses. In *ECCV* (2008).

[JFEM03] JEPSON A. D., FLEET D. J., EL-MARAGHI T. F.: Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 10 (2003), 1296–1311.

[JY05] JHALA A., YOUNG R. M.: A discourse planning approach to cinematic camera control for narratives in virtual environments. In *AAAI* (2005).

[JY06] JHALA A., YOUNG R. M.: Representational requirements for a plan based approach to automated camera control. In *AIIDE'06* (2006), pp. 36–41.

[KD06] KIM K., DAVIS L. S.: Multi-camera tracking and segmentation of occluded people on ground plane using search-guided particle filtering. In *ECCV* (2006).

[KL10] KWON J., LEE K. M.: Visual tracking decomposition. In *CVPR* (2010).

[KLHG09] KRÄHENBÜHL P., LANG M., HORNUNG A., GROSS M. H.: A system for retargeting of streaming video. *ACM Trans. Graph. 28*, 5 (2009).

[KMM12] KALAL Z., MIKOLAJCZYK K., MATAS J.: Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 34*, 7 (2012).

[KNM03] KAMEDA Y., NISHIGUCHI S., MINOH M.: Carmul: concurrent automatic recording for multimedia lecture. In *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on* (July 2003), vol. 2, pp. II–677–80.

[KTK*11] KAISER R., THALER M., KRIECHBAUM A., FASSOLD H., BAILER W., ROSNER J.: Real-time person tracking in high-resolution panoramic video for automated broadcast production. In *Visual Media Production (CVMP), 2011 Conference for* (2011), pp. 21–29.

[KWK12] KAISER R., WEISS W., KIENAST G.: The fascinate production scripting engine. In *Proceedings of the 18th international conference on Advances in Multimedia Modeling* (2012), MMM'12, pp. 682–692.

[KYA*97] KATO D., YAMADA M., ABE K., ISHIKAWA A., ISHIYAMA K., OBATA M.: Analysis of the camerawork of broadcasting cameramen. *SMPTE journal 106*, 2 (1997), 108–116.

[LCCR11] LINO C., CHOLLET M., CHRISTIE M., RONFARD R.: Computational model of film editing for interactive storytelling. In *ICIDS* (2011), pp. 305–308.

[LG06] LIU F., GLEICHER M.: Video retargeting: Automating pan and scan. In *Media '06 Proceedings of the 14th annual ACM international conference on Multimedia* (2006), pp. 241–250.

[LKF*02] LIU Q., KIMBER D., FOOTE J., WILCOX L., BORECZKY J.: Flyspec: A multi-user video camera system with hybrid human and automatic control. In *Proceedings of the Tenth ACM International Conference on Multimedia* (2002), MULTIMEDIA '02, pp. 484–492.

[LLS04] LEIBE B., LEONARDIS A., SCHIELE B.: Combined object categorization and segmentation with an implicit shape model. *Workshop on Statistical Learning in Computer Vision, ECCV* (2004), 17–32.

[LLS08] LEIBE B., LEONARDIS A., SCHIELE B.: Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision 77*, 1-3 (2008), 259–289.

[LQ00] LASZLO A., QUICKE A.: *Every Frame a Rembrandt: Art and Practice of Cinematography*. Focal Press, 2000.

[LRGC01] LIU Q., RUI Y., GUPTA A., CADIZ J. J.: Automating camera management for lecture room environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2001), CHI '01, pp. 442–449.

[MAP*10] MAVLANKAR A., AGRAWAL P., PANG D., HALAWA S., CHEUNG N.-M., GIROD B.: An interactive region-of-interest video streaming system for online lecture viewing. In *Packet Video Workshop (PV), 2010 18th International* (2010), pp. 64–71.

[Mas65] MASCELLI J.: *The Five C's of Cinematography: Motion Picture Filming Techniques*. Silman-James Press, 1965.

[MCT09] MURPHY-CHUTORIAN E., TRIVEDI M. M.: Head pose estimation in computer vision: A survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 31*, 4 (2009), 607–626.

[Mer10] MERCADO G.: *The Filmmaker's Eye: Learning (and Breaking) the Rules of Cinematic Composition*. Focal Press, 2010.

[MJ08] MILLERSON G., JIM O.: *Video Production Handbook*. Focal Press, 2008.

[MjOB06] MICILOTTA A. S., JON ONG E., BOWDEN R.: Real-time Upper Body Detection and 3D Pose Estimation in Monoscopic Images. In *ECCV* (2006).

[MJSB11] MARKOWITZ D., JR. J. T. K., SHOULSON A., BADLER N. I.: Intelligent camera control using behavior trees. In *MIG* (2011), pp. 156–167.

[MJZF11] MARIN-JIMENEZ M., ZISSERMAN A., FERRARI V.: Here's looking at you, kid. detecting people looking at each other in videos. In *BMVC* (2011).

[Mur01] MURCH W.: *In the blink of an eye*. Silman-James, 2001.

[NKMG02] NUMMIARO K., KOLLER-MEIER E., GOOL L. V.: An adaptive color-based particle filter. *Image and Vision Computing 21*, 1 (2002), 99–110.

[Ond04] ONDAATJE M.: *The Conversations: Walter Murch and the Art of Film Editing*. Random House, 2004.

[ORN09] O'NEILL B., RIEDL M. O., NITSCHE M.: Towards intelligent authoring tools for machinima creation. In *CHI Extended Abstracts* (2009), pp. 4639–4644.

[PB95] PINHANEZ C., BOBICK A.: Intelligent studios: Using computer vision to control tv cameras. In *Proc. of IJCAI'95 Workshop on Entertainment and AI/Alife* (August 1995), pp. 69–76.

[PJS12] PARK H. S., JAIN E., SHEIKH Y.: 3d social saliency from head-mounted cameras. In *NIPS* (2012), pp. 431–439.

[PKD05] PITIE F., KOKARAM A. C., DAHYOT R.: N-dimensional probability density function transfer and its application to color transfer. In *ICCV* (2005), vol. 2, IEEE, pp. 1434–1439.

[PKVP09] PRITCH Y., KAV-VENAKI E., PELEG S.: Shift-map image editing. In *ICCV'09* (Kyoto, Sept 2009), pp. 151–158.

[Pop10] POPPE R.: A survey on vision-based human action recognition. *Image and vision computing 28*, 6 (2010), 976–990.

[PPMZR10] PATRON-PEREZ A., MARSZAŁEK M., ZISSERMAN A., REID I. D.: High five: Recognising human interactions in TV shows. In *BMVC* (2010).

[Pro08] PROFERES N.: *Film Directing Fundamentals - See your film before shooting it*. Focal Press, 2008.

[RAGS01] REINHARD E., ASHIKHMIN M., GOOCH B., SHIRLEY P.: Color transfer between images. *IEEE Computer graphics and applications 21*, 5 (2001), 34–41.

[RBK07] RAMANAN D., BAKER S., KAKADE S.: Leveraging archival video for building face datasets. In *ICCV* (2007).

[RGB13] RONFARD R., GANDHI V., BOIRON L.: The prose storyboard language: A tool for annotating and directing movies. In *2nd Workshop on Intelligent Cinematography and Editing part of Foundations of Digital Games-FDG 2013* (2013).

[RGGH04] RUI Y., GUPTA A., GRUDIN J., HE L.: Automating lecture capture and broadcast: technology and videography. *ACM Multimedia Systems Journal 10* (2004), 3–15.

[RGSS10] RUBINSTEIN M., GUTIERREZ D., SORKINE O., SHAMIR A.: A comparative study of image retargeting. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 29*, 5 (2010), 160:1–160:10.

[RKV*09] RIJSSELBERGEN D. V., KEER B. V. D., VERWAEST M., MANNENS E., DE WALLE R. V.: Movie script markup language. In *ACM Symposium on Document Engineering* (2009), pp. 161–170.

[RLLY08] ROSS D. A., LIM J., LIN R.-S., YANG M.-H.: Incremental learning for robust visual tracking. *International Journal of Computer Vision 77*, 1-3 (2008), 125–141.

[Ron04] RONFARD R.: Reading movies: an integrated dvd player for browsing movies and their scripts. In *ACM Multimedia* (2004).

[Ron09] RONFARD R.: Automated cinematographic editing tool, may 2009. WO Patent 2,009,055,929.

[Ron12] RONFARD R.: A Review of Film Editing Techniques for Digital Games. In *Workshop on Intelligent Cinematography and Editing* (Raleigh, United States, May 2012), Arnav Jhala R. M. Y., (Ed.), ACM.

[RSA08] RUBINSTEIN M., SHAMIR A., AVIDAN S.: Improved seam carving for video retargeting. *ACM Trans. Graph. 27*, 3 (2008).

[RSA09] RUBINSTEIN M., SHAMIR A., AVIDAN S.: Multi-operator media retargeting. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2009) 28*, 3 (2009), 1–11.

[RST02] RONFARD R., SCHMID C., TRIGGS W.: Learning to parse pictures of people. In *ECCV* (2002).

[RTT03] RONFARD R., TRAN-THUONG T.: A framework for aligning and indexing movies with their script. In *ICME* (2003).

[SAD*06] SANTELLA A., AGRAWALA M., DECARLO D., SALESIN D., COHEN M.: Gaze-based interaction for semi-automatic photo cropping. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2006), CHI '06, pp. 771–780.

[Sal06] SALT B.: *Moving into pictures*. Starword, London, 2006.

[Sal09] SALT B.: *Film Style and Technology: History and Analysis (3 ed.)*. Starword, 2009.

[Sch97] SCHNEIDER D.: *The art and craft of Stage Management*. Harcourt Brace College Publishers, 1997.

[SEZ09] SIVIC J., EVERINGHAM M., ZISSERMAN A.: "Who are you?" – learning person specific classifiers from video. In *CVPR* (2009).

[SFKM05] SUN X., FOOTE J., KIMBER D., MANJUNATH B.: Region of interest extraction and virtual camera control based on panoramic video capturing. *Multimedia, IEEE Transactions on 7*, 5 (Oct 2005), 981–990.

[SFWS13] SCHREER O., FELDMANN I., WEISSIG C.AND KAUFF P., SCHAFER R.: Ultrahigh-resolution panoramic imaging for format-agnostic video production. *Proceedings of the IEEE 101*, 1 (January 2013), 99–114.

[SLS*10] SANTNER J., LEISTNER C., SAFFARI A., POCK T., BISCHOF H.: Prost: Parallel robust online simple tracking. In *CVPR* (2010).

[SMAY03] SHEN J., MIYAZAKI S., AOKI T., YASUDA H.: Intelligent digital filmmaker dmp. In *ICCIMA* (2003).

[Spo85] SPOLIN V.: *Theater games for rehearsal: A director's handbook*. Northwestern University Press, 1985.

[SR13] SUPANCIC J. S., RAMANAN D.: Self-paced learning for long-term tracking. In *CVPR* (2013).

[Sze06] SZELISKI R.: Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision 2*, 1 (2006), 1–104.

[SZS06] SIVIC J., ZITNICK C. L., SZELISKI R.: Finding people in repeated shots of the same scene. In *BMVC* (2006).

[TB09a] THOMPSON R., BOWEN C.: *Grammar of the Edit*. Focal Press, 2009.

[TB09b] THOMSON R., BOWEN C. J.: *Grammar of the shot*. Focal Press, 2009.

[TBS12] TAPASWI M., BÄUML M., STIEFELHAGEN R.: "knock! knock! who is it?" probabilistic person identification in tv-series. In *CVPR* (2012).

[TOM04] TAKEMAE Y., OTSUKA K., MUKAWA N.: Impact of video editing based on participants' gaze in multiparty conversation. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems* (2004), pp. 1333–1336.

[VRB00] VENEAU E., RONFARD R., BOUTHEMY P.: From video shot clustering to sequence segmentation. In *ICPR* (2000), pp. 4254–4257.

[WEWP00] WEBER M., EINHÄUSER W., WELLING M., PERONA P.: Viewpoint-invariant learning and detection of human heads. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition* (2000).

[WGCO07] WOLF L., GUTTMANN M., COHEN-OR D.: Non-homogeneous content-driven video-retargeting. In *ICCV* (2007).

[WLYY11] WANG S., LU H., YANG F., YANG M.-H.: Superpixel tracking. In *ICCV* (2011).

[WRB11] WEINLAND D., RONFARD R., BOYER E.: A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding 115*, 2 (2011), 224–241.

[WTS07] WEI YICHEN J. S., TANG X., SHUM H.-Y.: Interactive offline tracking for color objects. In *ICCV* (2007).

[WXC*08] WANG J., XU C., CHNG E., LU H., TIAN Q.: Automatic composition of broadcast sports video. *Multimedia Systems 14*, 4 (2008), 179–193.

[YF05] YOKOI T., FUJIYOSHI H.: Virtual camerawork for generating lecture video from high resolution images. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on* (2005).

[YJS06] YILMAZ A., JAVED O., SHAH M.: Object tracking: A survey. *ACM computing surveys 38*, 4 (2006).

[ZCL*08] ZHU L., CHEN Y., LU Y., LIN C., YUILLE A.: Max margin and/or graph learning for parsing the human body. In *CVPR* (2008).

[ZDJ*09] ZHANG G., DONG Z., JIA J., WAN L., WONG T.-T., BAO H.: Refilming with depth-inferred videos. *Visualization and Computer Graphics, IEEE Transactions on 15*, 5 (2009), 828–840.

[ZHM08] ZHANG Y.-F., HU S.-M., MARTIN R. R.: Shrinkability maps for content-aware video resizing. *Comput. Graph. Forum*, 7 (2008), 1797–1804.

[ZRCH08] ZHANG C., RUI Y., CRAWFORD J., HE L.-W.: An automated end-to-end lecture capture and broadcasting system. *ACM Trans. Multimedia Comput. Commun. Appl.* (2008).