Multi-Clip Video Editing from a Single Viewpoint

Vineet Gandhi, Rémi Ronfard, Michael Gleicher

To cite this version:

Vineet Gandhi, Rémi Ronfard, Michael Gleicher. Multi-Clip Video Editing from a Single Viewpoint. CVMP 2014 - European Conference on Visual Media Production, Nov 2014, London, United Kingdom. pp.Article No. 9 10.1145/2668904.2668936 . hal-01067093

HAL Id: hal-01067093
https://inria.hal.science/hal-01067093

Submitted on 23 Oct 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Multi-Clip Video Editing from a Single Viewpoint

Vineet Gandhi
INRIA/Laboratoire Jean Kuntzmann, France
vineet.gandhi@inria.fr

Remi Ronfard
INRIA/Laboratoire Jean Kuntzmann, France
remi.ronfard@inria.fr

Michael Gleicher
University of Wisconsin-Madison, USA
gleicher@cs.wisc.edu

ABSTRACT

We propose a framework for automatically generating multiple clips suitable for video editing by simulating pan-tilt-zoom camera movements within the frame of a single static camera. Assuming important actors and objects can be localized using computer vision techniques, 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 $L1$-norm 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. We demonstrate our approach on five video sequences of a live theatre performance by generating multiple synchronized subclips for each sequence.

Categories and Subject Descriptors

I.3.8 [**Computer Graphics**]: Applications; I.4.9 [**Image processing and Computer Vision**]: Applications

General Terms

Computer Vision, Computer Graphics

Keywords

Video Editing, Video Processing

1. INTRODUCTION

High quality video uses a variety of camera framings and movements edited together to effectively portray its content on screen. 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 paper, we introduce an approach 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.

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). Computer vision techniques are 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 linear program allowing for efficient and scalable solution. The key insight is that

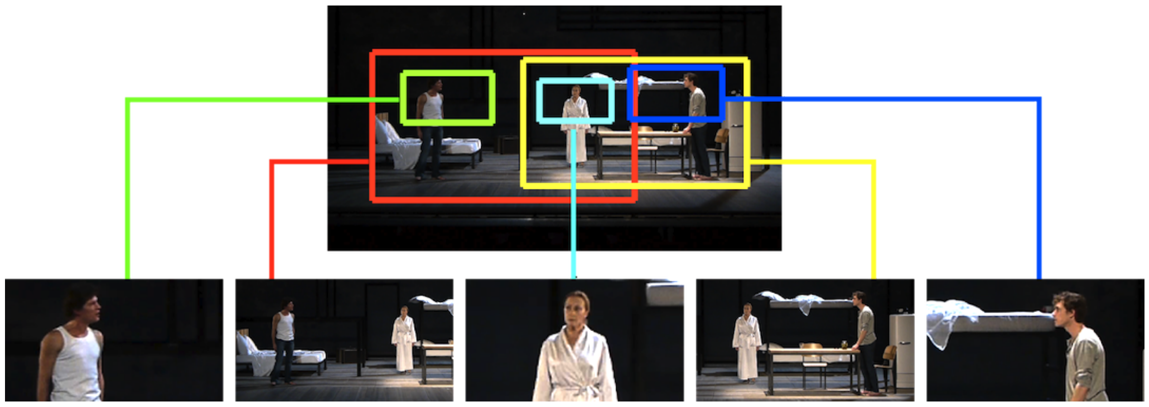

Figure 1: Our method takes as input a high resolution video from a single viewpoint and outputs a set of synchronized subclips by breaking the groups into a series of smaller shots.

many of the principles of good camera motion, including what camera shots are useful in editing, can be cast within this optimization framework.

2. RELATED WORK

The problem of creating a multi-view edited video from limited input camera views has been considered previously in very specialized scenarios. The Virtual Videography system [14] used virtual PTZ to simulate multiple cameras from a lecture video. The MSLRCS [23] and AutoAuditorium [5] systems use a small set of fixed views including input of presentation slides. These systems achieve full automation, albeit at the expense of allowing for creativity and human input in the editing process. They are also limited to working in the more constrained environment of usually a single presenter in front of a chalkboard or a slide screen. Extending to the richer environment of multiple actors in more complex stagings would be challenging, especially as it requires a richer set of shot types to be generated. In contrast, our system addresses only the rush generation aspect, but can generate a rich range of shot types from more complex scenarios including multiple actors.

Remarkably little work as been devoted specifically to the problem of optimizing the composition of a virtual PTZ camera in post-production. Recent work has focused more on providing interactive control of the virtual PTZ camera in real time [17, 8, 9]. A method for automatic editing of basketball games was recently proposed by Carr et al. [6] employing both robotic and virtual PTZ cameras. The robotic camera follows the centroid of the detected players and then a virtual camera is employed to crop subregions of the obtained images to compensate for motor errors. Because they are targeting online applications, their method makes decisions in near real time (with a small delay), which can lead to sub-optimal solutions. To the best of our knowledge, our technique provides the first general solution to the problem for *offline* computation of a virtual pan-tilt-zoom camera shot given a list of visual targets over the entire duration of a recorded performance.

The question of a computational model to describe "good" camera movements, particularly in terms of subwindows of video, has been considered by work in video stabilization. While most video stablization work simply aimed to remove jitter, Re-Cinematography [11] formalized cinematic principles in terms of movement in the frame for computation. Grundmann et al. [13] showed that this can be formulated in an optimization framework. We build on this computational formulation of camera movement, extending it to the problem of multiple rush generation.

Our work is also related to the general problem of automatic video editing. Our main contribution is this area is concerned with generating rushes that can easily be cut together. To the best of our knowledge, this has not been addressed in previous work. Berthouzoz et al. [4] have looked at the problem of *when* to place cuts and transitions during dialogue scenes, based on audio analysis, assuming that the input shots are correctly framed. We solve the complementary problem of generating correctly framed input shots. Wang et al. [22] proposed an approach for automatic editing of home videos. Their method selects important parts of the scene based on detections and motion saliency and summarizes video by removing parts of the video both temporally and spatially. Our method does not remove parts of the video temporally and generates multiple shots for the entire duration of the video with controlled camera dynamics. Arev et al. [2] 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. We are solving the opposite problem of using a single high-resolution viewpoint. Interestingly, multiple cameras focusing on the same attention point give a heuristic measure of the importance of that attention point, which can be used to automatically edit the video without deep understanding of the scene. In our case, we cannot rely on such heuristic and instead focus on the rush generation problem, leaving the "final cut" to the user.

A general framework for automatically cropping sub-clips from a panoramic video and editing them together into a movie in real-time has been proposed for the case of sports events [7, 20]. Operating in real-time limits the capability of that system to tracking of simple targets. Because we are focusing on post-production, rather than live broadcast, we are able to perform tracking and recognition of multiple actors even in complex cases, which makes our method more generally applicable to rich cultural heritage content. Furthermore, our method can be used to generate arbitrary shot compositions, which better addresses the challenges of post-production.

Using computer vision to control cameras and create full-edited movies is not a new idea. Pinhanez and Bobick [19] have investigated the use of low-level motion tracking to control TV cam-

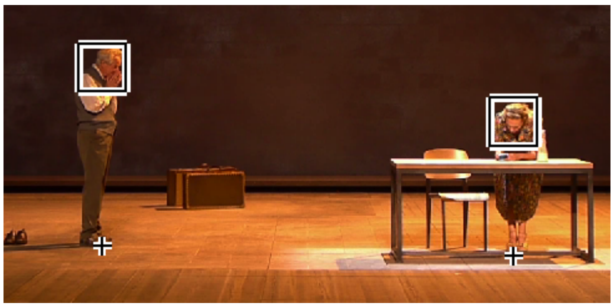

Figure 2: Input to the system are the upper body bounding boxes for each actor (shown with squares) and the points where the actors touch the stage floor (shown with cross markers).

eras but their method is limited to closed-world domains (exemplified by cooking shows) and requires extensive domain-specific knowledge. In contrast, our approach is domain-agnostic and relies on general principles of image composition and film editing. Our method assumes a pre-processing stage, where actors are tracked and the location of their heads and floor projections are recovered in each frame of the input video. In this paper, we implemented a simple offline tracker based on actor specific detections [10] but any suitable tracker can be used instead, including interactive trackers introduced recently by Bailer et al. [3] which are well suited for the task of video production.

3. VIRTUAL CAMERA SPECIFICATION

Given a single video covering the entire stage our method allows the user to create different reframed shots or rushes with a simple description for each shot. Each reframed video is generated by computing a virtual camera trajectory over the master shot i.e. choosing a cropped window over the master shot at each time. The core component of our approach is an optimization framework to compute the virtual camera trajectory with optimal composition, movement and “cuttability”. 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. Several established video production books emphasize the role of composition and cutting in cinematography [1, 16, 15, 18]. Mascelli [16] 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.

3.1 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 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 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.

3.2 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.

3.3 Shot naming conventions

Based on common practice in video production [21], shots are specified by their subject matter and size. The subject matter for a shot is typically the list of actors who should be onscreen. Optionally, objects or stage locations can also be subject matters for a virtual camera shot, although this is not used in this paper. 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. Those specifications are given for the entire duration of the master shot. For instance, the specification for a virtual camera can simply be a "full shot of actor A and actor B" or just "FS A,B". Figure 3 shows that a total of five virtual cameras can be specified using only two shot 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.

3.4 Composition

As emphasized by Mascelli [16], 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[16] 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 the bottom frame

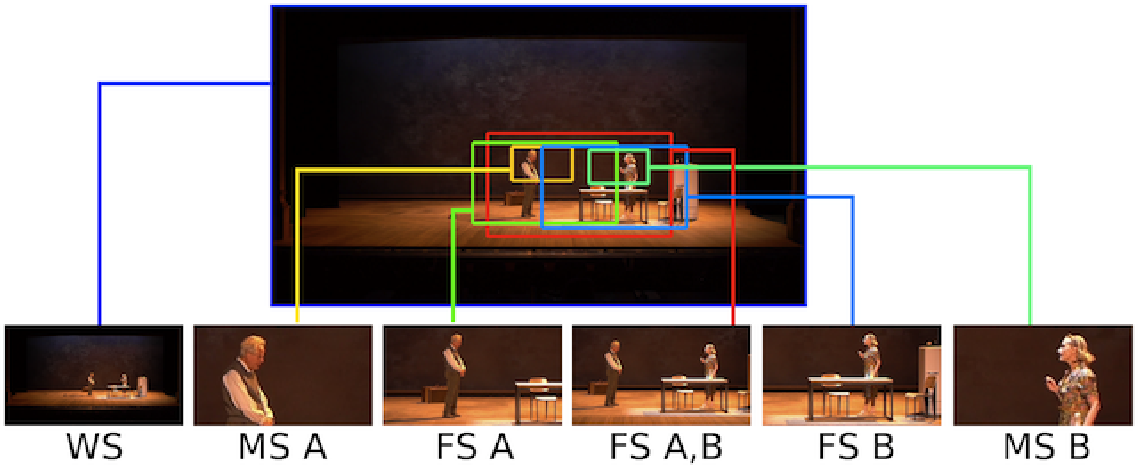

Figure 3: The figure shows the possible set of framing with two actors (A,B) and two shot sizes i.e. the medium shot (MS) and the full shot (FS). Even with a simple case of two actors a total of 6 camera choices are available including the original wide shot (WS).

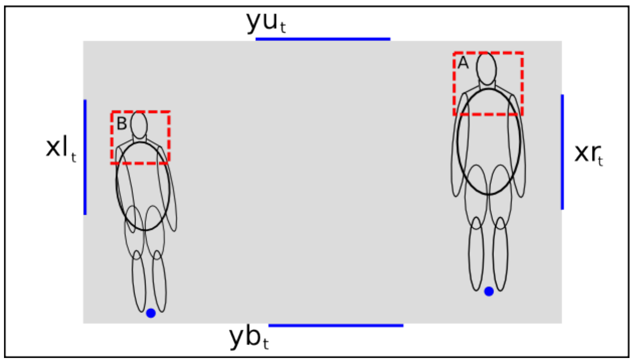

Figure 4: Inclusion region for shot specification "FS A,B" i.e. the full shot of Actor A and Actor B. A full shot of two actors is a tightest window which keeps both of them entirely inside the frame.

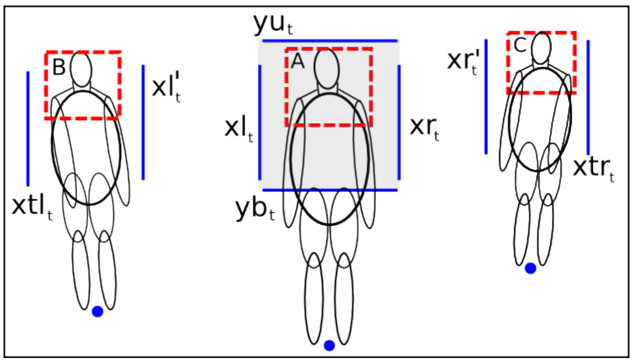

Figure 5: Inclusion region for shot specification "MS A" i.e. the medium shot of Actor A. The boundaries of nearest actors on the left and the right are also shown in this figure.

should *not* cut across subject's joints (knees, waist, elbows, ankles) but should instead cut *between joints*; (c) subjects 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 4 and Figure 5 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 upper body 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 its upper body bounding box. 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 the as 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 4.2.

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 4.6.

3.5 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. In this paper, we enforce "cuttability" of the shots by maintaining screen continuity of all actors and by creating only "sparse" virtual camera movements.

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 a preference for virtual shot compositions where the actors keep the screen positions from the master shot. An example is given in Figure 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".

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.

3.6 Camera movement

Importance of a steady camera has been highlighted by Thomson's '*Grammar of the shot*' [21]. 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 steady 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 [21] mentions that a good pan/tilt movement 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.

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 4.3 and 4.4.

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 which 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 4.5.

4. OPTIMIZATION

In this section we show that 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 $\{xl_t^{'}, xr_t^{'}, xtl_t, xtr_t\}$, 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.

We also define $x_t = \frac{1}{2}(xl_t + xr_t)$ as the midpoint of the left and the right coordinates of the inclusion region and $y_t = \frac{1}{2}(yu_t + yb_t)$ as the midpoint of vertical inclusion coordinates. We define $s_t = \frac{1}{2}(yb_t - yu_t)$ 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 $A_r fs_t$.

4.1 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 . Formally, we define the horizontal inclusion constraints as:

$$0 < fx_t - A_r fs_t \leq xl_t \text{ and } xr_t \leq fx_t + A_r fs_t \leq W. \qquad (1)$$

Similarly, we define the vertical inclusion constraints:

$$0 < fy_t - fs_t \leq yh_t \text{ and } yb_t \leq fy_t + fs_t \leq H, \qquad (2)$$

where H is the height of the master shot.

4.2 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 of left and right coordinates of the inclusion region. Similarly, we want fy_t to be close to the midpoint of the top and the bottom coordinates of the inclusion region. Also, we want the size to be closer to the length of vertical inclusion constraints (i.e. $yb_t - yh_t$). Any diversion from the desired size is penalized using a data term:

$$D(\xi) = \frac{1}{2}\sum_{t=1}^{N}((fx_t - x_t)^2 + (fy_t - y_t)^2 + (fs_t - s_t)^2). \qquad (3)$$

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 3.5, 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_t 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 $(fx_t - x_t)$ in Equation 3 to $(fx_t + 0.17Arfs_t h_t - x_t)$.

4.3 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

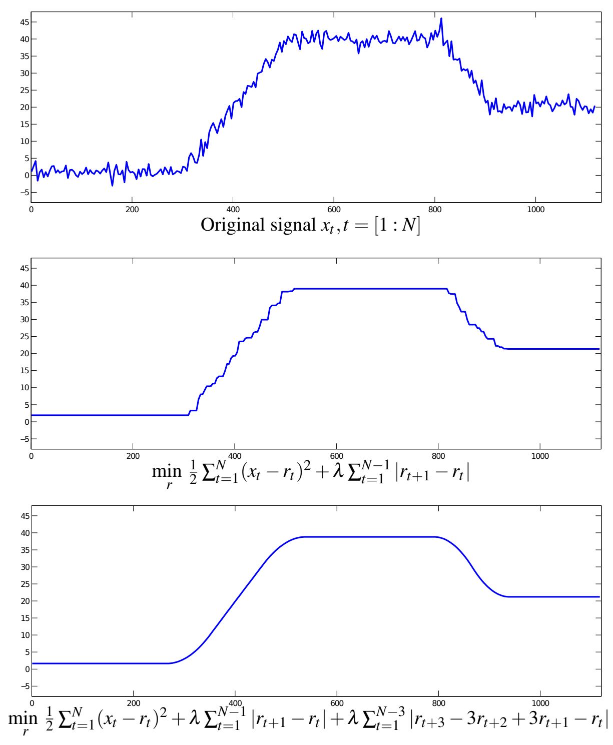

Figure 6: Top: Synthetic one dimensional data x. Middle: Optimized signal r, minimizing the sum of squares closeness term to original data with L1-norm regularization on velocity. Bottom: Optimized signal r, minimizing the sum of squares closeness term to original data with L1-norm regularization on both velocity and jerk.

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:

$$L_{11}(\xi) = \sum_{t=1}^{N-1} (|fx_{t+1} - fx_t| + |fy_{t+1} - fy_t| + |fs_{t+1} - fs_t|). \quad (4)$$

This is illustrated in Figure 6 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 6 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.

4.4 Third order L1-norm regularization

When the camera moves it should move smoothly. The camera movement should start with a segment of constant acceleration and should end with a segment of constant deceleration. Using only L1-norm on velocity will lead to sudden start and stop of the camera (sharp corners in middle plot of Figure 6). It also leads to a staircase artifact (slopes in middle plot of Figure 6). Previous work [13] 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:

$$\begin{aligned} L_{13}(\xi) = \sum_{t=1}^{N-3} (&|fx_{t+3} - 3fx_{t+2} + 3fx_{t+1} - fx_t| \\ &+ |fy_{t+3} - 3fy_{t+2} + 3fy_{t+1} - fy_t| \\ &+ |fs_{t+3} - 3fs_{t+2} + 3fs_{t+1} - fs_t|). \quad (5) \end{aligned}$$

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 6 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.

4.5 Apparent motion penalty

To preserve the sense of activity on stage, the actual motion of the actors should be same as the apparent motion seen in the cropped window on the virtual camera. 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.

$$\begin{aligned} M_1(\xi) = \sum_m \sum_{t=1}^{N-1} (&cx_t^m |fx_{t+1} - fx_t| + cy_t^m |fy_{t+1} - fy_t| \\ &+ cs_t^m |fs_{t+1} - fs_t|). \quad (6) \end{aligned}$$

Here cx_t^m, cy_t^m and cs_t^m are pre-stored binary vectors 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 to the cost function, else this term is zero.

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

$$\begin{aligned} M_2(\xi) = \sum_m \sum_{t=1}^{N-1} (&max(0, -(\dot{bx}_t^m - \dot{fx}_t)\dot{bx}_t^m) \\ &+ max(0, -(\dot{by}_t^m - \dot{fy}_t)\dot{by}_t^m) \\ &+ max(0, -(\dot{bs}_t^m - \dot{fs}_t)\dot{bs}_t^m)). \quad (7) \end{aligned}$$

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.

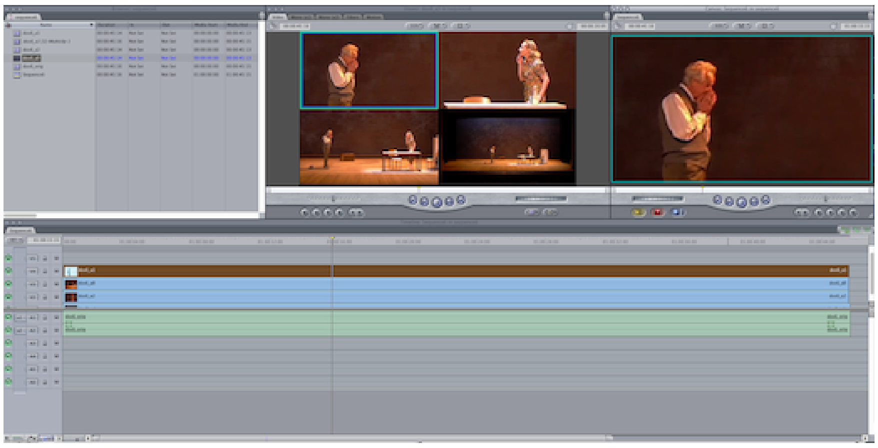
Figure 7: Screenshot of a multiclip sequence generated using a set of four sequences in Final Cut Pro. In the middle we see the four sequences including the original master shot and three reframed sequences (MS A, MS B, FS All) which were generated using the proposed method. On the right, we see the edited sequence.

4.6 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 add a penalty if the rightmost coordinate of the cropping window $fx_t + A_r fs_t$ lies within the right external actor boundaries $xr_t^{'}$ and xtr_t (please refer to Figure 5). Similarly, we would like to add a penalty if the leftmost coordinate of the cropping window $fx_t - A_r fs_t$ lies within the left external actor boundaries $xl_t^{'}$ and xtl_t (please refer to Figure 5). But such a conjunction is not convex.

To approximate this within the convex framework, we use a heuristic that pre-computes binary 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_{out} and E_{in}. The E_{out} penalty is only applied when no touch event is occurring on the left or the right inclusion regions. It is defined as follows:

$$E_{out}(\xi) = \sum_{t=1}^{N}((\sim tl_t) max(0, xl_t^{'} - fx_t + A_r fs_t) + (\sim tr_t) max(0, fx_t + A_r fs_t - xr_t^{'})). \quad (8)$$

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. Similarly, the penalty is also defined for left edge. The no touch event in Equation 8 is defined as the logical not ($\sim$) of the left and the right touch vectors tl_t and tr_t.

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

$$E_{in}(\xi) = \sum_{t=1}^{N}(tl_t\ max(0, fx_t - A_r fs_t - xtl_t) + tr_t\ max(0, xtr_t - fx_t - A_r fs_t)). \quad (9)$$

Here, xtr and xtl 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 xtr_t, otherwise a penalty of $(xtr_t - fx_t - A_r fs_t)$ is added to the cost function.

4.7 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:

$$\begin{aligned} \underset{fx,fy,fs}{\text{minimize}}\ & (D(\xi) + \lambda_1 L_{11}(\xi) + \lambda_2 L_{13}(\xi) + \lambda_3 E_{out}(\xi) \\ & + \lambda_4 E_{in}(\xi) + \lambda_5 M_1(\xi) + \lambda_6 M_2(\xi)) \\ \text{subject to}\ & \\ & 0 \le fx_t - A_r fs_t \le xl_t, \\ & xr_t \le fx_t + A_r fs_t \le W, \\ & 0 \le fy_t - fs_t \le yh_t, \\ & yb_t \le fy_t + fs_t \le H, t = 1, \ldots, N. \end{aligned} \quad (10)$$

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 10. In our case we use cvx [12].

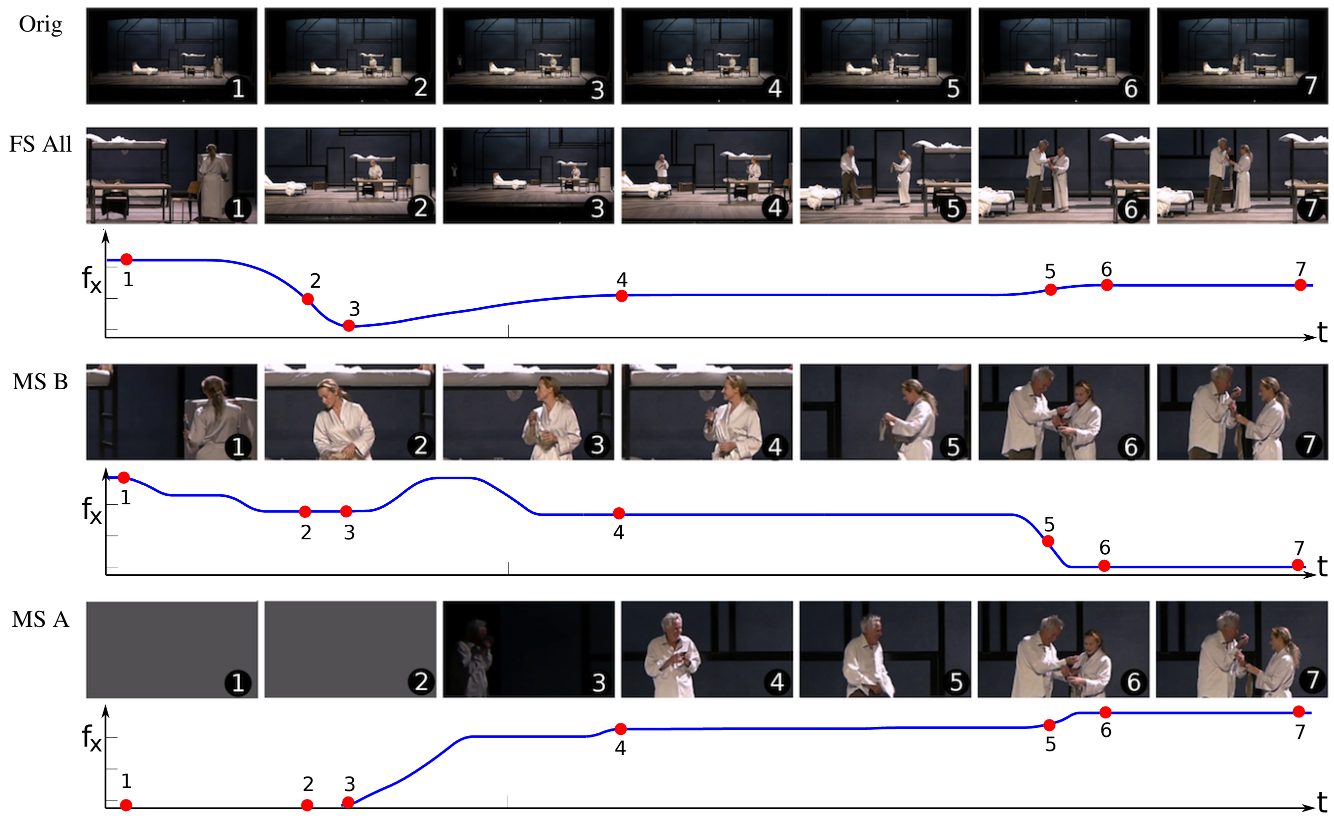

Figure 8: Reframing results on a sequence with two actors (A,B). The top row shows a set of selected keyframes from the original video. The corresponding keyframes from the three different virtual camera sequences are shown below. The three reframed sequences include the medium shot of each actor (MS 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.

5. RESULTS

We present results on five different sequences from Arthur Miller's play '*Death of a Salesman*'. The sequences were recorded during rehearsals at Célestins, Théâtre de Lyon. Each of these sequences were recorded from the same viewpoint in Full HD (1920×1080). 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 shows example of a multi-clip sequence consisting of the original sequence (master shot) and the three reframed sequences generated using our method. All the original videos and generated rushes are available online[1].

Qualitative evaluation. The results on two different sequences are shown in Figure 8 and Figure 9. Each figure shows a few selected keyframes from the original video and the corresponding frames from the 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 8 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 9, 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 8 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 9, which maintains a tight full shot of actors B and C but pulls in actor A

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

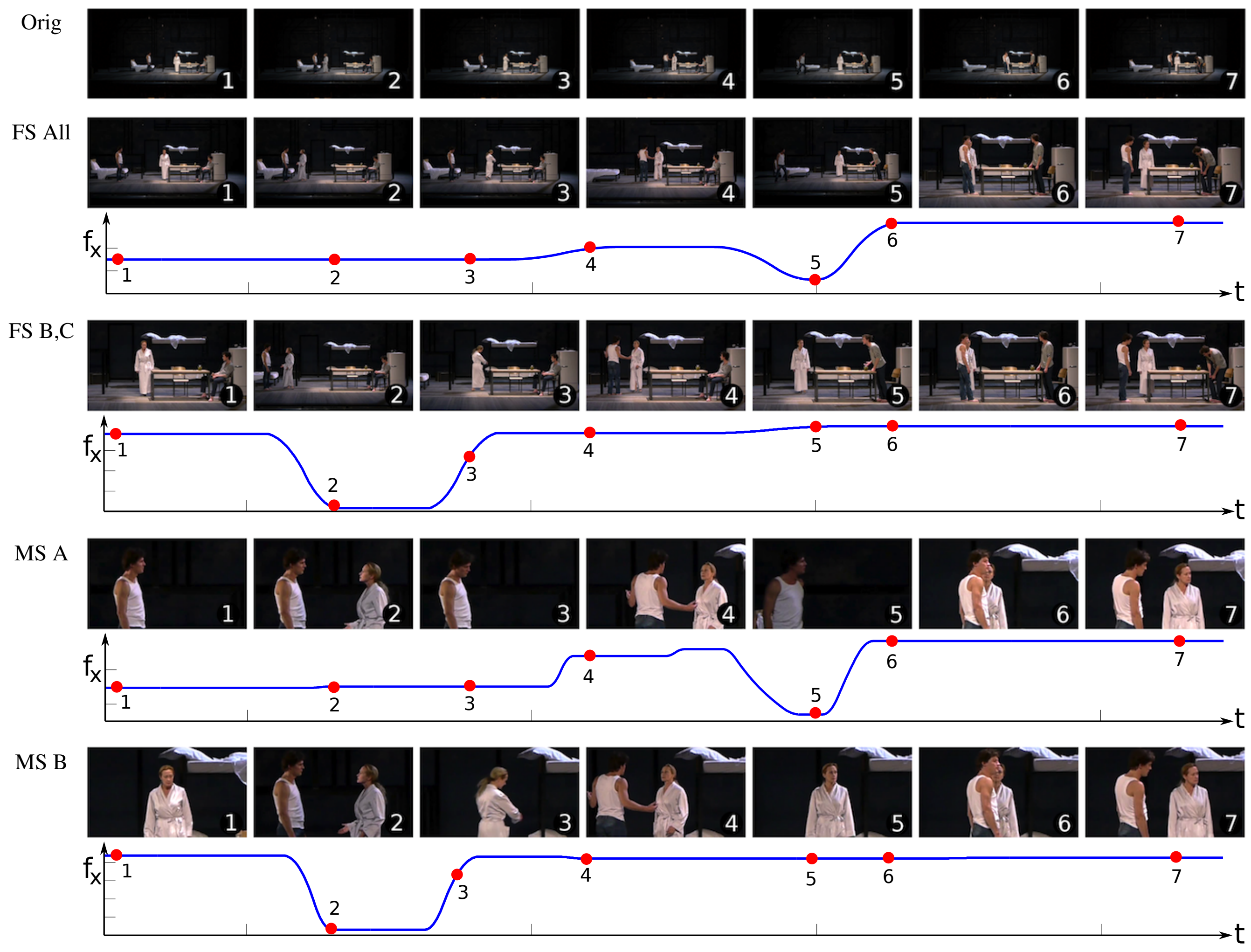

Figure 9: Reframing results on a sequence with three actors (A,B,C). Selected keyframes from the original sequence and 4 virtual camera sequences are shown in this figure. The four virtual camera sequences include the full shot of all three actors (FS All), full shot of actors two actor (FS B,C) and 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.

when it comes close to the camera frame.

Camera motion. The plots of f_x in Figure 8 and Figure 9 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 8 as the actors do not move significantly. When the camera moves, it moves smoothly preserving the apparent motion of the actors on stage. For example, observe how the camera "MS A" in Figure 8 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 provides 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 8. Similarly, the editor can switch among all five options at keyframe 1 in Figure 9. In a few scenarios the generated virtual cameras may not be cuttable, for example cutting between camera "MS A" and "MS B" at keyframe 6 in Figure 8 in not possible because it 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 8.

6. LIMITATIONS AND FUTURE WORK

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. In future work, we plan to perform 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, we plan to integrate some simple objects in the shot naming conventions using standard objects detectors. The Full HD master shots used in the experiments in this paper 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 from our method are automatically annotated with actor and camera movements which makes them suitable for automatic editing. In future work we plan to investigate the problem of automatic camera selection given the rushes.

7. CONCLUSION

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.

8. ACKNOWLEDGMENT

This work was supported by Region Rhone Alpes (project Scenoptique) and ERC advanced grant EXPRESSIVE. All images and videos are reproduced courtesy of Célestins, Théâtre de Lyon. Special thanks to Claudia Stavisky, Auxane Dutronc and the cast and crew of '*Death of a Salesman*'.

9. REFERENCES

[1] John Alton. *Painting With Light*. University of California Press, 1949.

[2] Ido Arev, Hyun Soo Park, Yaser Sheikh, Jessica K. Hodgins, and Ariel Shamir. Automatic editing of footage from multiple social cameras. In *ACM Transactions on Graphics (SIGGRAPH)*, 2014.

[3] Christian Bailer, Alain Pagani, and Didier Stricker. A user supported tracking framework for interactive video production. In *Proceedings of the 10th European Conference on Visual Media Production*, 2013.

[4] Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. Tools for placing cuts and transitions in interview video. *ACM Transactions on Graphics (SIGGRAPH)*, 2012.

[5] Michael Bianchi. Automatic video production of lectures using an intelligent and aware environment. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, 2004.

[6] Peter Carr, Michael Mistry, and Iain Matthews. Hybrid robotic/virtual pan-tilt-zom cameras for autonomous event recording. In *Proceedings of the 21st ACM International Conference on Multimedia*, 2013.

[7] F. Daniyal and A. Cavallaro. Multi-camera scheduling for video production. In *Visual Media Production (CVMP), 2011 Conference for*, 2011.

[8] Vamsidhar Reddy Gaddam, Ragnar Langseth, Sigurd Ljødal, Pierre Gurdjos, Vincent Charvillat, Carsten Griwodz, and Pål Halvorsen. Interactive zoom and panning from live panoramic video. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, NOSSDAV '14, 2013.

[9] Vamsidhar Reddy Gaddam, Ragnar Langseth, Håkon Kvale Stensland, Pierre Gurdjos, Vincent Charvillat, Carsten Griwodz, Dag Johansen, and Pål Halvorsen. 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*, MMSys '14, 2014.

[10] Vineet Gandhi and Remi Ronfard. Detecting and Naming Actors in Movies using Generative Appearance Models. In *Computer Vision and Pattern Recognition*, 2013.

[11] Michael Gleicher and Feng Liu. Re-cinematography: Improving the camerawork of casual video. *ACM Transactions on Multimedia Computing Communications and Applications (TOMCCAP)*, 5(1):1–28, 2008.

[12] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, March 2014.

[13] M. Grundmann, V. Kwatra, and I. Essa. Auto-directed video stabilization with robust l1 optimal camera paths. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[14] Rachel Heck, Michael Wallick, and Michael Gleicher. Virtual videography. *ACM Trans. Multimedia Comput. Commun. Appl.*, 3(1), February 2007.

[15] Andrew Laszlo and Andrew Quicke. *Every Frame a Rembrandt: Art and Practice of Cinematography*. Focal Press, 2000.

[16] Joseph Mascelli. *The Five C's of Cinematography: Motion Picture Filming Techniques*. Silman-James Press, 1965.

[17] Aditya Mavlankar and Bernd Girod. Video streaming with interactive pan/tilt/zoom. In *High-Quality Visual Experience*, Signals and Communication Technology, pages 431–455, 2010.

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

[19] Claudio S. Pinhanez and Aaron F. Bobick. Intelligent studios modeling space and action to control tv cameras. *Applied Artificial Intelligence*, 11(4):285–305, 1997.

[20] O. Schreer, I. Feldmann, P. Weissig, C.and Kauff, and R. Schafer. Ultrahigh-resolution panoramic imaging for format-agnostic video production. *Proceedings of the IEEE*, 101(1):99–114, January 2013.

[21] Roy Thomson and Christopher J. Bowen. *Grammar of the shot*. Focal Press, 2009.

[22] Tinghuai Wang, A Mansfield, Rui Hu, and J.P. Collomosse. An evolutionary approach to automatic video editing. In *Visual Media Production, 2009. CVMP '09. Conference for*, pages 127–134, Nov 2009.

[23] Cha Zhang, Yong Rui, Jim Crawford, and Li-Wei He. An automated end-to-end lecture capture and broadcasting system. *ACM Trans. Multimedia Comput. Commun. Appl.*, 2008.