Zooming On All Actors: Automatic Focus+Context Split Screen Video Generation

Moneish Kumar, Vineet Gandhi, Rémi Ronfard, Michael Gleicher

Zooming On All Actors: Automatic Focus+Context Split Screen Video Generation

Moneish Kumar[1], Vineet Gandhi[1], Remi Ronfard[2] and Michael Gleicher[3]

[1]IIIT Hyderabad, [2]Univ. Grenoble Alpes/INRIA/LJK, [3]University of Wisconsin-Madison.

Abstract

Recordings of stage performances are easy to capture with a high-resolution camera, but are difficult to watch because the actors' faces are too small. We present an approach to automatically create a split screen video that transforms these recordings to show both the context of the scene as well as close-up details of the actors. Given a static recording of a stage performance and tracking information about the actors positions, our system generates videos showing a focus+context view based on computed close-up camera motions using crop-and zoom. The key to our approach is to compute these camera motions such that they are cinematically valid close-ups and to ensure that the set of views of the different actors are properly coordinated and presented. We pose the computation of camera motions as convex optimization that creates detailed views and smooth movements, subject to cinematic constraints such as not cutting faces with the edge of the frame. Additional constraints link the close up views of each actor, causing them to merge seamlessly when actors are close. Generated views are placed in a resulting layout that preserves the spatial relationships between actors. We demonstrate our results on a variety of staged theater and dance performances.

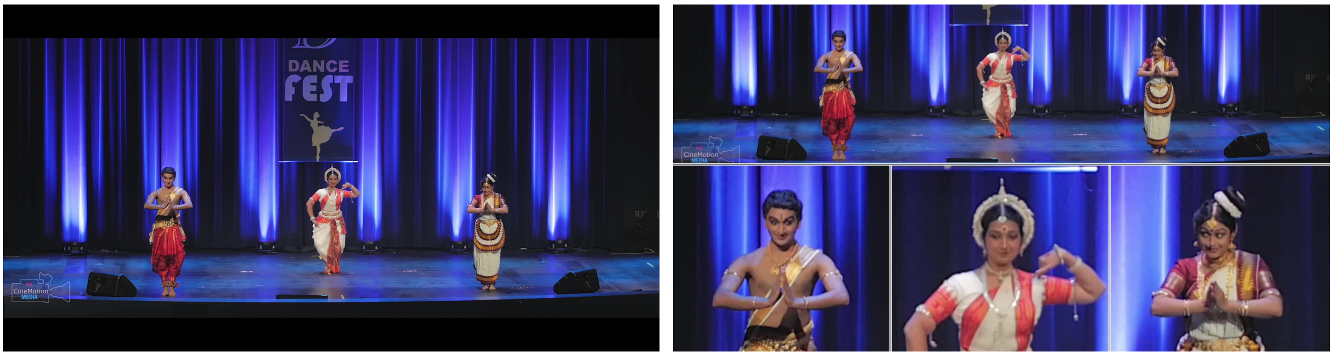
Categories and Subject Descriptors (according to ACM CCS): I.2.10 [Computing Methodologies]: Vision and Scene Understanding—Video Analysis I.3.3 [Computing Methodologies]: Picture/Image Generation—Viewing Algorithms

1. Introduction

A video presenting a staged event, such as theatre or dance, must choose between providing a wide field of view of the whole scene or close-up views that show details. Recordings of staged performances typically use multiple cameras, with multiple camera operators, to capture multiple views which are edited together to make a single video. Alternatively, these views may be composited together to create a *split-screen composition* (SSC). Split-screen compositions give the user the decision of what to attend to, reducing the need for editorial decisions that can be difficult to automate. Creating good split-screen compositions requires creating a set of views that are good individually and can be used together, as well as creating layouts that correctly convey the scene and its details.

In this paper, we present an approach for creating split-screen compositions of staged performances. We record a high-resolution, but wide field-of-view, video of the event with a static (unattended) camera. While this easy to create recording may capture detail, it does not necessarily provide a convenient way for a viewer to see both the whole scene and important details (like actors' facial expressions or gestures). Therefore, we provide an automatic system that transforms the video into a split-screen composition of both the wide view of the scene as well as detailed views of the actors' faces (Figure 1). The close-up views are created as virtual camera movements by applying panning, cropping and zooming to the source video. The key challenges are: (a) to compute appropriate virtual camera movements in a way that creates good close-ups which work together; and (b) to create proper layouts of these views that preserve the spatial relationships between actors.

The core of our approach is a novel method to determine the camera movements for close-up views of each actor given their position on stage (tracking information). Our method takes tracking information as input, and, therefore, is independent of the tracking algorithm. For example, our prototype implementation uses an interactive offline approach that provides acceptable actor position estimates with some manual annotations. Given this tracking information, our approach poses camera trajectory computations as a convex optimization, which aims at showing an actor as large and centred in the close-up as possible, given the constraints that: (a) the entire face must be visible; and (b) the frame should avoid cutting other actors' faces and torso. An $L(1)$ regularization term creates movements that are as smooth as possible but also follow the kinds of acceleration patterns preferred by cinematographers. When multiple actors come close together, the system combines them into a single close-up view to avoid cutting their faces (or torso) with the frame. By adding additional constraints between the camera movements, the different camera paths merge seamlessly as



(a) (b)

Figure 1: *Illustration of our approach showing a frame from the input video (a) and its corresponding split screen composition (b). The split screen view shows the emotions and expressions of the performers, which are hardly visible in the original view (master shot).*

actors approach. A single convex optimization is computed for all trajectories over an entire video segment.

Our system produces videos with a wide "master shot" and a set of close-ups of all identified actors. This simple layout avoids making editorial decisions about what is more or less important in the scene. Yet, it provides a focus+context view that shows both the overall action as well as the details of actors faces to allow for seeing emotion and dialogue. The system presents the close-ups in an arrangement that preserves the spatial relationships between the actors. As actors move, the close-ups seamlessly merge as they approach, split as they distance, and re-arrange to preserve ordering.

This paper presents our approach to automatically creating split-screen, focus+context videos from captured recordings of staged performances. We provide an overview of the visual goals of our system in Section 3, and the layout adjustment algorithm is described in Section 4. Section 5 describes how the virtual camera paths are computed as a convex optimization to be seamlessly composited together. Section 6 describes our prototype implementation and examples of results we have produced with it. This paper's contributions include:

1. An end-to-end framework to automatically obtain SSC's from a single static master shot and the screen position (tracks) of actors present in the scene.
2. A method to dynamically adjust the partitions of the split screen composition as the actors move around and interact with each other.
3. A method to obtain jerk free transitions from a layout to another by using top down constraints on the individual virtual camera simulations.
4. Results on a variety of video sequences from theatre and dance with multiple actors and complex movements to demonstrate the effectiveness of our approach. We also present a novel application of SSC's with partition aware subtitles, which can enhance the perception of staged theatre for hearing challenged viewers.

2. Related Work

The previous attempts for generating SSC's from a static camera (or a set of static cameras) has been limited to specific scenario like lecture videos, where a 'picture in picture' setting shows the inset view of speaker over the slides of the lecture. Bianchi's auto-auditorium [Bia04] was one of the first systems to demonstrate the automated production of lecture videos. The auto-auditorium system employed a static camera to look at the entire scene and analyzed it to control a robotic camera to follow the speaker to be shown in the inset view. The robotic camera was replaced by virtual camera simulations in [SFKM05] and [HWG07], where the speaker inset view was generated by moving a cropping window inside a high resolution camera frame. However, most of these methods have been confined to a simple restricted setting of single presenter in front of a chalkboard or slides. The SSC in these methods is also limited to a single simulated window of the presenter overlaid usually at the bottom right part of the display.

The virtual camera work has also been investigated in the field of sports. The work in FascinatE project [SFW*13] demonstrated real time interactive virtual camera editing in ultra high resolution video recording of soccer games. Carr et al. [CMM13] used multiple static cameras to obtain player tracks in Basketball games and used the centroid of the tracks to guide a robotic camera. They showed that smooth, human-like camera trajectories can be obtained as a post-processing step utilizing non-causal filtering.

A variety of approaches have been proposed to obtain smooth virtual camera trajectories. Chen and Vleeschouwer [CDV10] use markov random fields (MRF). Yokoi and Fujiyoshi [YF05] used bilateral filtering to reduce the noise from virtual camera trajectories obtained using a region of interest (ROI) computed at each frame. Sun et al. [SFKM05] used a rule based approach combined with a Kalman filter to smooth out noise from initial tracking estimates. Heck et al [HWG07] explicitly modelled the camera movement into three segments (parabolic, linear and then again parabolic). However, most of these methods are limited by ad-hoc rules or lack of generalization. Moreover, the conventional filtering based approaches only suppresses high frequency jitter and may not mimic the behaviour of a professional cameraman.

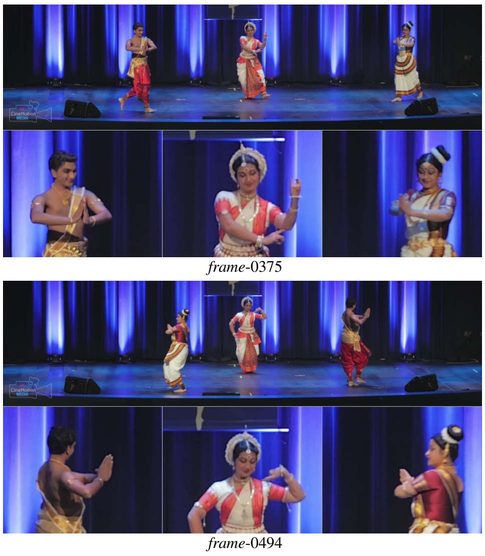


Figure 2: *A fixed-layout SSC does not preserve the left to right order of the actors on stage. Our approach addresses this issue.*

Our work is closely related to recent $L(1)$-norm based camera stabilization/simulation methods [GKE11, GRG14]. The work in [GL08] showed that the trajectories of professional camera work can be modeled as distinct static, linear and parabolic segments and used this idea to stabilize noisy recordings. Grundmann et al. [GKE11] showed this can be implemented using a linear optimization framework. Gandhi et al. [GRG14] extended this idea for virtual camera simulation and presented an application of multi-clip editing from single viewpoint. These algorithms are focused on creating smooth camera movements, considering one trajectory at a time. We build upon these works and allow interaction between multiple simulated trajectories in a nested optimization framework. We further demonstrate that how these ideas can be incorporated for an altogether novel application of creating SSC's.

Prior work [LG06, DDN08, JSSH15] explores re-editing cinematic videos by computing cropping windows. Re-editing employs both new camera movements (generated by moving the cropping window inside the original frames) and cuts (generated by switching between cropping windows). The work in [LG06] and [DDN08] perform a spatio-temporal optimization to retain the computed salient parts of the frame inside the cropped window. The work by Jain et al. [JSSH15] makes re-editing decisions by employing an eye tracker instead. For instance a quick shift in the viewers' attention may suggest a possibility of a cut and a steady shift may suggest for a pan movement in the re-edited video. Our method, on the other hand simulates and handles multiple virtual camera feeds simultaneously.

3. Overview

Our SSC design shows the wide-field of view (source) imagery as well as a set of generated close-up views (e.g., Figure 1). Each actor is shown in one close-up. Normally, there is a view per actor, however, when multiple actors are close, they are shown together to avoid having part of an actor appear at the edge of a view of another actor. The layout of the SSC partitions refers to the resulting screen area showing the wide view at top, and a horizontal row of the close-ups. The close-ups are presented so that the actors appear in the same left-to-right order they appear in the master shot.

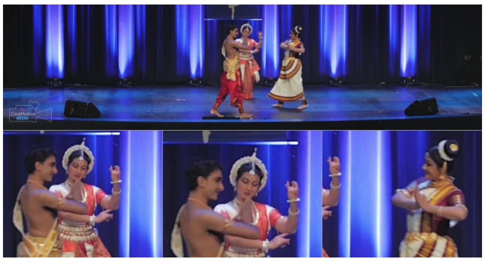


Figure 3: *A fixed layout causes problems when actors interact. Our approach addresses this issue.*

The inputs to our system are the recording from a static camera (master shot) and the position of the actors in each frame (actor tracks). A parameter controls the desired size of the focus view (the size the actors will appear). Our system automatically creates an SSC video from these inputs.

3.1. Challenges

Traditional SSC's are usually created by combining feeds from multiple cameras, where each camera is operated by a professional (focusing on particular subject). The layouts are fixed, where a partition of screen space is assigned to a particular subject. SSC's are most often used in settings where the movements of the subjects is minimal, and the subjects do not interact in ways that would cause the different views to merge or cross. Hence, SSC's have mostly been prevalent in scenario such as broadcast of debates or discussions of a panel etc., which are recorded in a controlled news studio like environment.

This standard SSC creation does not work well in our target scenario of theatre stage recordings, where the subjects move around and interact with each other. A fixed layout becomes problematic when actors cross as illustrated in Figure 2, where the close-up view of the male actor is still shown on the left, even after he crosses over to the right side of the stage. This may hinder the correct understanding of the scene. Hence, a goal of our SSC creation algorithm is to dynamically adjust the layout to preserve the actors configuration on stage. However, such adjustments must be handled gracefully so they are not jarring to the viewer.

A related problem is caused by the interaction of the actors. When actors approach (for example when they cross), the close-up of one may contain another. This may lead to redundancy or having an actor cut by the edge of the view, see Figure 3. Our approach avoids such redundancy, merging views as the actors come close. When actors split apart, the views follow their movements smoothly preserving their ordering.

As our method generates the SSC's from a single master shot (instead of manually operated multiple cameras), it must generate multiple camera views from this single feed. This is achieved

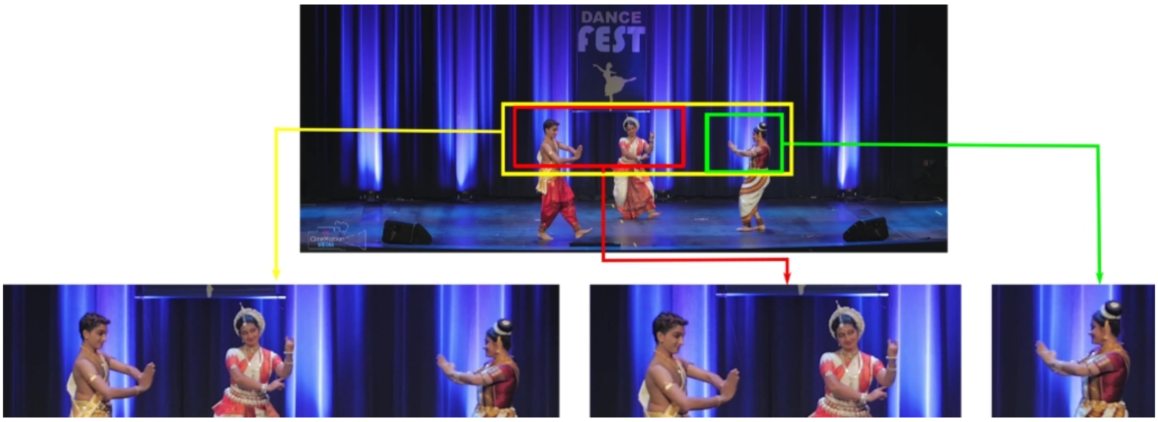


Figure 4: *A close-up view varies with the number of actors to be included in it. The figure shows the cropped framing corresponding to a close-up view of a single actor, two actors and three actors.*

by simulating virtual pan/tilt/zoom cameras by moving a cropping window inside the master shot. Because the actor tracks are often wobbly, simple following would lead to unsteady views. We overcome this problem using a $L(1)$ regularized optimization framework (explained with detail in Section 5). This framework includes constraints to make sure that the merging and splitting of views is seamless (Section 5.4).

These principles lead to five major goals for our approach to automatic SSC generation:

1. Each close-up view (insets) should be good individually, i.e. as large and well-framed as possible.
2. Virtual camera movements for each close-up view should follow cinematic norms, e.g., be smooth with damped acceleration profiles.
3. SSC layout should preserve arrangement: all actors are shown with their left to right order preserved.
4. Redundancy among different screen partitions (different inset views) should be avoided i.e. no actor appears more than once and no "cut faces."
5. The changes in layout(splitting or merging close-up view of different actors), if any, should be seamless.

4. Layout

The first phase of our method chooses the layout, selecting which views will be shown for each frame of the resulting video. It chooses how many close-up views, and which actors will be shown in each. A second phase, described in the subsequent sections, determines the content to be displayed in each view.

Our algorithm partitions a horizontal space for a set of close-up views. Because we enforce left-to-right ordering, there is a small set of possible configurations: each actor can have his own view, or adjacent actors may be combined into a single view. Combinations are possible until there is a single group combining all actors. For a scene with 3 actors, there are 4 possible partitions as illustrated in Figure 6. For a scene with 4 actors, there are 8 possibilities. The algorithm must select from this for each frame. We pose this selection as a constrained optimization over the entire video segment so that we can enforce continuity objectives.

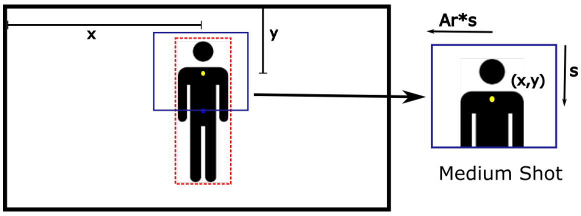


Figure 5: *Framing convention. The framing (blue rectangle) is obtained using the actor track (red dotted rectangle) and the desired size of the close-up view (medium shot i.e. head to waist in this case). It is defined using the center position (w.r.t to original frame), size and the aspect ratio.*

Each view is defined by its virtual cropping frame, the area of the source image from which it will be taken. We represent the frame by the position of its center (x,y), size (s) and aspect ratio A_r (as illustrated in Figure 5). We determine an initial cropping frame for each candidate view (i.e. view that may be used in the layout) in order to select the layout, and then we use the selected layout to compute a refined cropping frame for each view that is used. The frames are computed using the input actor tracks, which are given in the form of bounding boxes from head to toe. For a single actor, the frame is the top half of its tracking bounding box in desired aspect ratio while keeping some empty space above the head (called headroom) for proper framing. Similarly, the frame for two actors is the smallest frame that covers both as they are covered in the individual framing. Note, that in standard film terminology these are technically "medium shots" not close-ups. The aspect ratio is chosen to divide the total width by the number of actors, e.g. for three actors, one actor takes a third of the total width, two actors take two thirds.

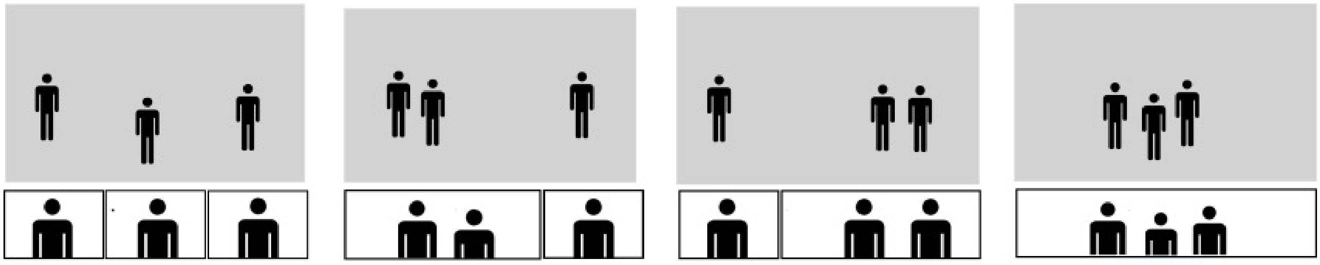


Figure 6: *We choose different split-screen layouts depending on the grouping of actors. This figure illustrates the four possible layouts for three actors.*

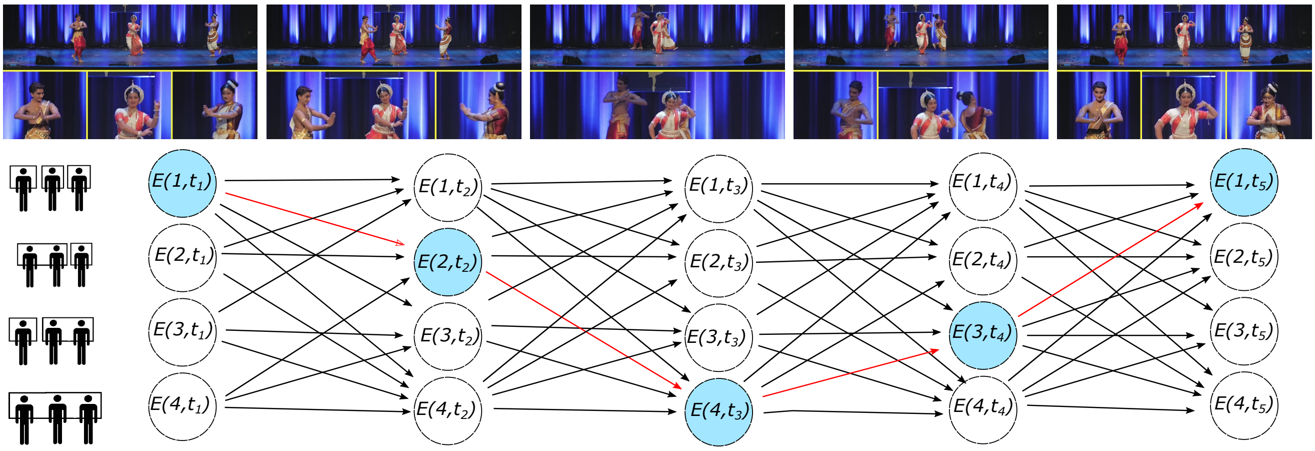


Figure 7: *Illustration of layout selection algorithm with a realistic three actor example. The algorithm selects a node for each time t from the given possibilities (four in this case). For each node in the graph, we compute the minimum cost $E(r_t,t)$ to reach it. Backtracking is then performed from the minimum cost node in the last column. The blue color shows the selected state, given the positions of the actors at the respective time (images above show the SSC's).*

4.1. Layout configuration selection

Our algorithm selects among the k possible configurations for each frame, producing a sequence $\varepsilon = \{r_t\}$ of states $r_t \in [1:k]$, for all frames $t = [1:N]$. The optimization minimizes two terms in a global cost function:

$$E(\varepsilon) = \sum_{t=1}^{N} E_o(r_t) + \lambda \sum_{t=2}^{N} E_s(r_{t-1}, r_t). \quad (1)$$

Where the first term (E_o) computes an overlap cost for each frame that attempts to avoid overlapping views and encourages views to be separated, and E_s is a smoothness term with weight λ.

E_o is the overlap cost, penalizing any overlap between the framings in the given layout. In case of three actors (a,b and c from left to right), this is defined as follows:

$$E_o(r_t,t) = \begin{cases} O_{a,b}(t) + O_{b,c}(t) & \text{if } r_t = 1\,, \\ -O_{a,b}(t) + O_{b,c}(t) & \text{if } r_t = 2\,, \\ O_{a,b}(t) - O_{b,c}(t) & \text{if } r_t = 3\,, \\ -O_{a,b}(t) - O_{b,c}(t) & \text{otherwise.} \end{cases} \quad (2)$$

Here, $O_{a,b}(t)$ denotes the overlap cost between the individual framing of left and middle actors a and b at time t. Given the individual framings (x_t^a, y_t^a, s_t^a) and (x_t^b, y_t^b, s_t^b) at time t for actors a and b respectively, it is defined as sum of width minus the distance between the center of the two framings:

$$O_{a,b}(t) = A_r(s_t^a + s_t^b) - |x_t^a - x_t^b| \quad (3)$$

The above term is negative if there is no overlap between the individual framing of actor a and actor b. It takes a positive value in case of overlapping framings. The term $O_{b,c}(r_t,t)$ is similarly defined as the overlap cost between the individual framing of right actor c and the middle actor b. We can observe in Equation 2, the first state will be the preferred state when all three actors are further apart, with both $O_{a,b}$ and $O_{b,c}$ taking negative values. State two will be preferred state when there is overlap between actor a and actor b, but actor c is further apart (with $O_{a,b}$ taking a positive value and $O_{b,c}$ taking a negative value). Similarly, state four will be preferred when all three actors are close to each other(both $O_{a,b}$ and $O_{b,c}$ taking positive values).

The smoothness term E_s penalizes frequent transitions (momentary transitions may distract the user), it is defined as follows:

$$E_s(r_t, r_{t-1}) = \begin{cases} 0 & \text{if } r_t = r_{t-1} \text{ ,} \\ 1 & \text{if } r_{t-1} = 1 \text{ and } r_t \in \{2,3\} \text{ or vice versa,} \\ 1 & \text{if } r_{t-1} \in \{2,3\} \text{ and } r_t = 3 \text{ or vice versa,} \\ 2 & \text{if } r_{t-1} = 3 \text{ and } r_t = 1 \text{ or vice versa.} \end{cases} \quad (4)$$

No penalty is incurred if the layout configuration does not change. A change from state one to either state two or three is penalized with an unit cost. A transition from state one to state four directly is penalized with twice the unit cost. The unit cost is defined by the parameter λ in Equation 1.

Here, have taken an example scenario of three actors for explanation purposes, but the system can be easily extended to handle split screen compositions with different number of actors. In general, the number of possible states is equivalent to finding number of subset of the $(n-1)$ separators between n actors, which equals 2^{n-1} possibilities (each separator is either active or not). For example in case of two actors there are two possible states and we only need to handle a single overlap term and single transition. Similarly, for four actors, there are eight possible states which need to be handled with three overlap terms and sixty four possible transitions.

Finally, we minimize Equation 1 using dynamic programming. The process is illustrated with a synthetic example in Figure 7.

5. Split screen optimization

After determining the layout for each video frame, our method then simulates the camera feeds for each view that appears. For example, in state one in Figure 6, three views are needed, whereas in state two, two views are needed. A simulated camera feed, or view, requires determining the cropping window (a rectangle within the wide shot). The layout specifies which views must be generated, which actors appear in each view, and the aspect ratio for each view. An optimization process determines the center position and size for each window (for each view in each frame). Following the approach of [GRG14], the optimization considers all frames together to insure smoothness, but it also adds constraints between views to ensure seamless transitions between layout changes. Hard constraints are used to make sure the view shows the actors and that views align at transition points, while objective terms impels the views to be proper close-ups and that camera motions are smooth.

The optimization algorithm takes as input the actor tracks, the aspect ratio and the number of actors to be included in the particular view (virtual camera feed). The actor track bounding box at time t for a particular actor a is defined by its center bx_t^a, by_t^a, half width bw_t^a and half height bh_t^a. The initial virtual camera cropping window is individually computed for each frame of the video using the actor tracks. This initial computed virtual camera frame f_t is denoted by its center and size (x_t, y_t, s_t) for each frame t (as illustrated in Figure 5). The output of the algorithm is the optimized set of framing coordinates $\xi = \{x_t^*, y_t^*, s_t^*\}$ following the cinematographic conventions for composition and movement of the camera.

5.1. Shot size penalty

Each view should contain a close-up of the specified actors. We define this as having the view approximately cover the actors from head to waist. The shot size penalty thus penalizes any deviation from the desired close-up view size. It is defined as follows:

$$D(\xi) = \sum_{t=1}^{N} ((x_t^* - x_t)^2 + (y_t^* - y_t)^2 + (s_t^* - s_t)^2). \quad (5)$$

The function increases the cost if the optimized virtual cropping frame deviates from the initial cropping window individually computed for each frame.

5.2. Inclusion constraints

We introduce two sets of hard constraints, first that the virtual cropping window should always lie within the master shot and second that the appropriate part of the actor track should be included inside the virtual cropping window. The first constraint ensures that we always get a feasible solution and second constraint ensures that the actors are not strangely chopped off by the virtual cropping window. For instance, the left most coordinate of the cropping window $x_t - A_r * s_t$ should be greater than zero and less than the left boundary of the actor bounding box $bx_t - bw_t$. Similarly, the rightmost coordinate of the cropping window $x_t + A_r * s_t$ should be greater than the right boundary of the actor bounding box $bx_t + bw_t$ and should be less than the width of the master frame W. Formally, we define the horizontal constraints as:

$$0 < x_t - A_r * s_t \le bx_t - bw_t \text{ and } bx_t + bw_t \ge x_t + A_r * s_t \le W \quad (6)$$

In case of view with multiple actors, the leftmost actor is considered for the left constraint and the right most actor in the shot is considered for the right constraint. The vertical constraints are similarly defined for the top and the lower virtual frame boundaries. The lower constraint is adjusted according to the view size.

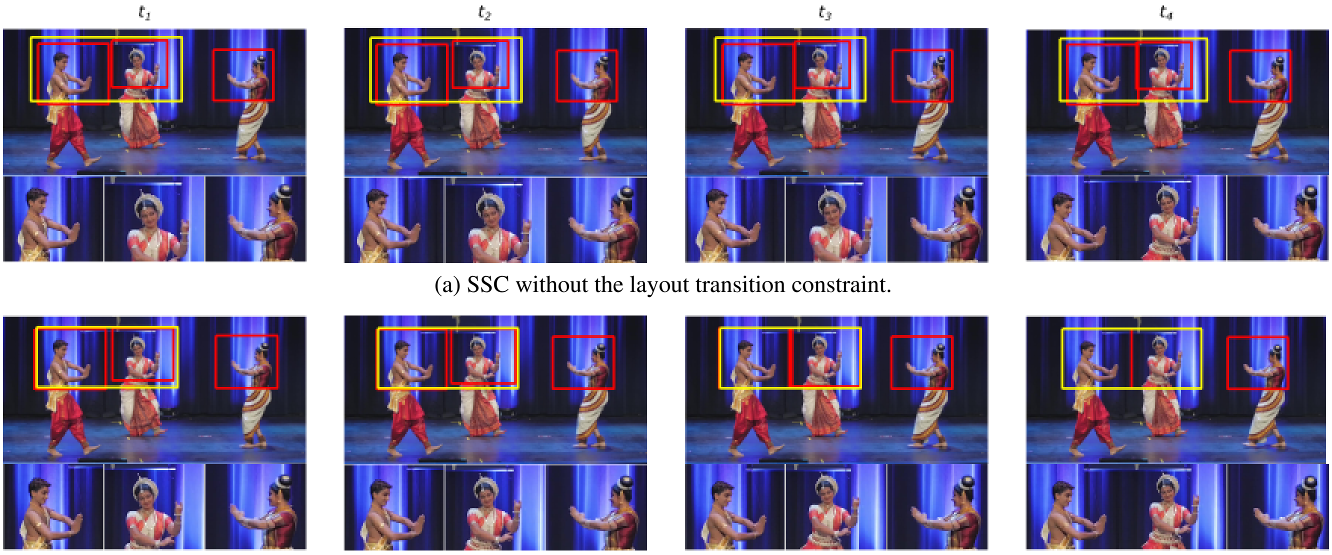
5.3. Movement regularization

Simply computing the framing for each individual frame may lead to a noisy virtual camera motion. According to professional cinematographic practices [BT13], a steady camera behaviour is necessary for pleasant viewing experience. A camera movement without enough motivation may appear irritating to the viewer, hence the camera should remain static in case of small and unmotivated movements. When the camera moves, it should start with a segment of constant acceleration followed by a segment of constant velocity and should come to a static state with a segment of constant deceleration. Gleicher and Liu translated this into heuristics [GL08], and Grundmann et. al showed that such motions could be computed as the minima of an $L(1)$ optimization [GKE11].

We introduce three different penalty terms to obtain the desired camera behaviour. The first term tends to keep the camera static by penalizing the $L(1)$ norm over the first order derivative:

$$M_1(\xi) = \sum_{t=1}^{N-1} (|x_{t+1}^* - x_t^*| + |y_{t+1}^* - y_t^*| + |s_{t+1}^* - s_t^*|). \quad (7)$$

The second term creates constant velocity segments while the



(a) SSC without the layout transition constraint.

(b) SSC with layout transition constraint.

Figure 8: *An example of layout transition. The two partitions on the left merge into a single one as the left dancer moves towards the screen right. The red rectangles shows the individual medium shots for each dancer and the yellow rectangle shows the combined medium shot of two actors on the left.*

camera moves by minimizng accelerations:

$$M_2(\xi) = \sum_{t=1}^{N-2} (|x^*_{t+2} - 2x^*_{t+1} + x^*_t| + |y^*_{t+2} - 2y^*_{t+1} + y^*_t| + |s^*_{t+2} - 2s^*_{t+1} + s^*_t|). \quad (8)$$

The third term minimizes jerk, leading to segments of constant acceleration:

$$M_3(\xi) = \sum_{t=1}^{N-3} (|x^*_{t+3} - 3x^*_{t+2} + 3x^*_{t+1} - 3x^*_t| + |y^*_{t+3} - 3y^*_{t+2} + 3y^*_{t+1} - 3y^*_t| + |s^*_{t+3} - 3s^*_{t+2} + 3s^*_{t+1} - 3s^*_t|). \quad (9)$$

Combining these three penalties yields camera movements consisting of distinct static, linear and parabolic segments.

5.4. Split and merge constraints

The split screen composition is obtained by simulating multiple virtual camera feeds and displaying them at the respective positions. For example in state one in Figure 7, three different camera feeds corresponding to each individual actor is concatenated to form the lower part of the SSC. However, the layout may differ as the actors change their positions on stage as described in Section 4.1. For instance state two in Figure 7, will require concatenation of only two virtual camera feeds i.e. a close-up view of two leftmost actors and a close-up view of the right actor. Overall, we need to simulate all possible close-up views (virtual camera feeds) for the entire video, which may be required for rendering the SSC. With example case of three actors, we will need to generate seven different virtual camera feeds (3 single actor feeds, 3 two actor feeds and 1 three actor feed).

However, individually generating each of these seven virtual camera feeds and then concatenating them will cause jerks in event of transition from one layout to another. This is illustrated in top row of Figure 8. At time t_1, all three actors are far apart and split screen composition shows their individual camera feeds side by side. The layout transition happens at time t_4, as the actor on the left moves closer to the middle actor. The computed virtual camera framings for the close-up view of all three actors (red rectangles) and the close-up view of the left two actors are also shown (yellow rectangle). We can observe that switching from the two individual close-up views to the combined close-up view of left two actors (at transition point t_4), will cause jerk (due to the discontinuity).

We address this problem, using a top down approach for virtual camera simulation i.e. we first optimize the highest order camera feed (including all actors) in the scene and use it to apply additional constraints on the lower order camera feeds (with lesser number of actors) at the point of transitions. We just need to make sure that at the point of transition, the higher order close-up view should exactly be equal to the union of its subset close-up views. We introduce this as a hard constraint to obtain fluid transitions from one layout configuration to another. The example in lower row of Figure 8 illustrates the output with top-down constraints, which makes sure that the individual framings of left two actors (red bounding boxes) exactly match the yellow rectangle at the point of transition. This top down approach builds upon previous work [GRG14], which optimize for each camera trajectory individually. It uses simpler constraints (reducing the complexity of the framework) and can yet allow for resolving limitations like jump cuts for the application of multi clip editing observed in [GRG14].

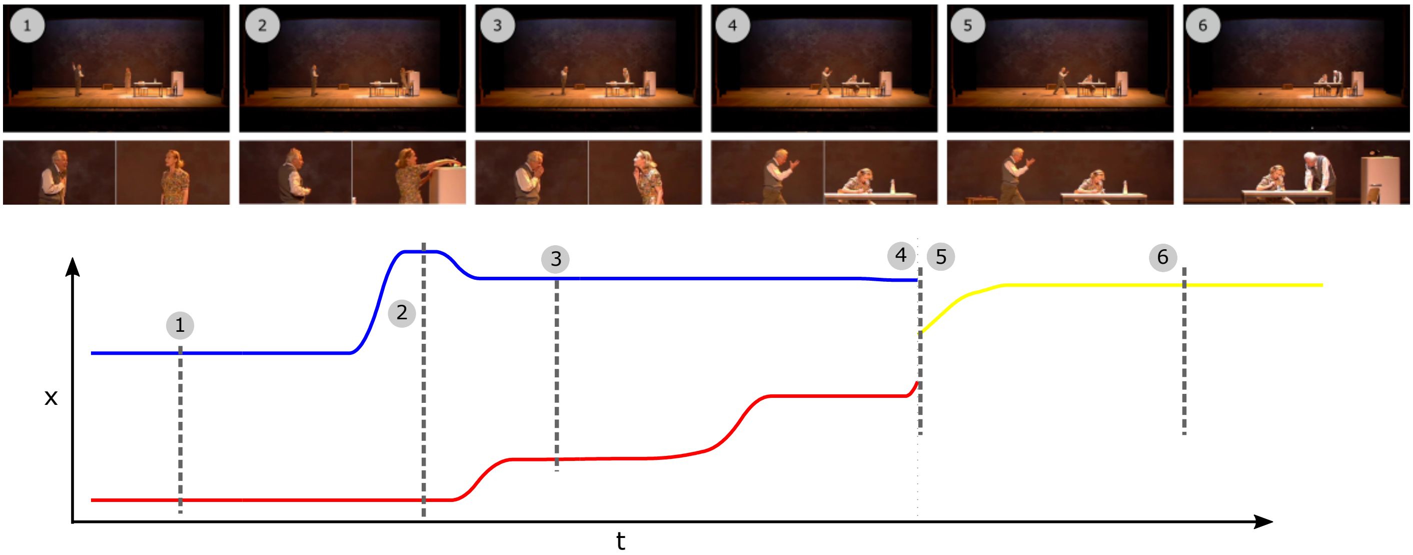


Figure 9: *Trajectories of the centre x-coordinate of the computed virtual framings over a theatre sequence with two actors (sequence1). Some selected images and the corresponding SSC's are also shown. The trajectory for individual framing of male actor 'Willy' and female actor 'Linda' are shown in blue and red color respectively. The yellow color presents the trajectory of the both actors together. The dotted black lines show the time instances where images were taken. The layout transition happens between frames 4 and 5.*

More formally, assuming that a higher order close-up view f_t^H gets partitioned into a set of two lower order close-up views $\{f_t^{s_l}, f_t^{s_r}\}$ at point of layout transition t (or the lower order views get merged into higher order views). For instance, a close-up view of three actors together gets partitioned into a view of a single actor and a view of two actors together. The following is the layout transition constraint:

$$f_t^{s_l} \bigcup f_t^{s_r} = f_t^H \text{ and } f_t^{s_l} \bigcap f_t^{s_r} = \phi, \tag{10}$$

where ϕ is the null set. This constraint is added at all points of layout transitions τ, which are pre-computed and are known prior to the virtual camera optimization. The above equation implies that there should be no overlap between the lower order subset views and their union should exactly comprise the higher order view. Let $f_t^{s_l}$ denotes the left side partition and $f_t^{s_r}$ denote the right side partition, then the Equation 10 can be defined as a set of *Split And Merge* linear constraints $SAM(f^{s_l}, f^{s_r}, f^H, \tau)$:

$$\begin{aligned}
&x_t^{s_l} + Ar^{s_l} h_t^{s_l} = x_t^H,\\
&x_t^{s_r} - Ar^{s_r} h_t^{s_r} = x_t^H,\\
&y_t^{s_l} = y_t^{s_r} = y_t^H,\\
&h_t^{s_l} = h_t^{s_r} = h_t^H,\ t \in \tau.
\end{aligned} \tag{11}$$

These conditions ensure that the (x,y)-coordinates and height of the lower order close-up views are correctly aligned with the higher order views, at the point of of transitions to avoid any jerk. The constraints can be easily extended to higher order splits (such as a three actor view splitting into three individual views), as a higher order split can be defined as multiple binary splits.

5.5. Energy minimization

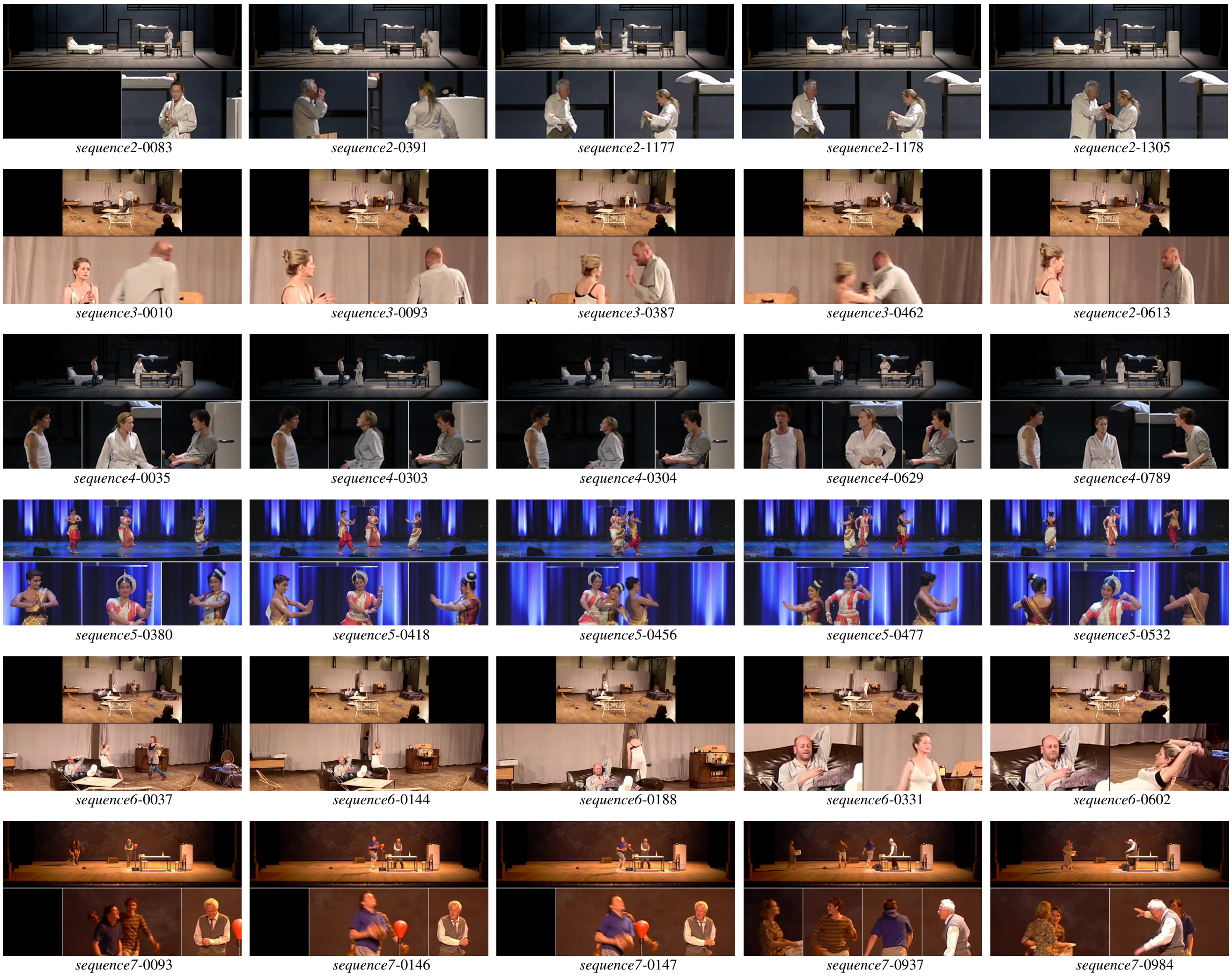
Finally, the problem of finding the virtual camera trajectory, can be simply be stated as a problem of minimizing a convex cost function with linear constraints. The overall optimization function is defined as follows:

$$\begin{aligned}
&\underset{x,y,s}{\text{minimize}}\ (D(\xi) + \lambda_1 M_1(\xi) + \lambda_2 M_2(\xi) + \lambda_3 M_3(\xi))\\
&\text{subject to}\\
&\qquad 0 < x_t^* - A_r * s_t^* \leq bx_t - bw_t,\\
&\qquad bx_t + bw_t \leq x_t^* + A_r * s_t^* \leq W,\\
&\qquad 0 < y_t^* - s_t^* \leq by_t - bh_t,\\
&\qquad by_t^\prime \leq y_t^* - s_t^* \leq H,\\
&\qquad SAM(f^{s_l}, f^{s_r}, f^H, \tau),\ t = 1, \ldots, N.
\end{aligned} \tag{12}$$

The variables λ_1, λ_2 and λ_3 are parameters. The optimization is done for each view separately from higher order to lower (views with more actor are optimized first). The SAM function, derives the hard layout transition constraints from the higher order views depending on the configuration (no layout constraints are applied on the view with all actors). The overall optimization function can be optimized using any off the shelf convex optimization toolbox, we use cvx [GB14].

6. Experimental results

We demonstrate results on seven different sequences from Arthur Miller's play 'Death of a salesman'; two sequences from Tennessee Williams' play 'Cat on a hot tin roof' and an Indian classical dance sequence. The play sequences were recorded from same viewpoint in Full HD (1920 × 1080) and the dance sequence was recorded



sequence2-0083 *sequence2*-0391 *sequence2*-1177 *sequence2*-1178 *sequence2*-1305

sequence3-0010 *sequence3*-0093 *sequence3*-0387 *sequence3*-0462 *sequence2*-0613

sequence4-0035 *sequence4*-0303 *sequence4*-0304 *sequence4*-0629 *sequence4*-0789

sequence5-0380 *sequence5*-0418 *sequence5*-0456 *sequence5*-0477 *sequence5*-0532

sequence6-0037 *sequence6*-0144 *sequence6*-0188 *sequence6*-0331 *sequence6*-0602

sequence7-0093 *sequence7*-0146 *sequence7*-0147 *sequence7*-0937 *sequence7*-0984

Figure 10: *Examples frames from SSC over different video sequences (sequence2, sequence3, sequence4, sequence6, sequence7 are from theatre and sequence5 is of a dance performance). This illustration includes cases with two (sequence2, sequence3), three (sequence4, sequence5, sequence6) and four (sequence7) actors.*

in 4K (3840 × 2160) resolution. To show the versatility of our approach, we have chosen sequences with two, three and four actors. For each of the given master sequence, we generate split screen compositions by rendering multiple virtual camera feeds. The computation of the SSC takes about 8 seconds for a minute long sequence with three actors on a laptop with i5 processor and 8GB RAM (excluding the rendering time). All the rendered videos are provided in the supplementary material. Comparison between a naive approach (of simply computing the framing for each actor independently at each time) and the proposed approach has also been provided in the supplementary material. Additionally, example frames from the SSC's of different sequences are shown in Figure 9, Figure 10 and Figure 11.

Layout transitions: Results on a two actor theatre sequence are shown in Figure 9. The Figure illustrates the selected images from the original video and their corresponding split screen compositions with the x-coordinate of the computed virtual framings over the entire sequence. We can observe the seamless transition of layout from two partitions to a single partition at frame-5 (the x-coordinate of the yellow trajectory corresponding to a combined medium shot is exactly at the mid-point of the two individual framings, at the point of transition). The fluidness of the transition can also be observed in the rendered images of the SSC where the union of two partitions at frame-4, exactly constitutes the two shot framing in the next frame. Similar transitions can also be observed at *sequence2*-1177/1178, *sequence4*-0303/0304 and *sequence7*-0146/0147 in Figure 10. Moreover, the example SSC's in Figure 10 clearly demonstrate the versatility of our approach, covering results comprising of different layouts with different aspect ratios; with different number of actors undergoing significant movements/interactions. The layout transitions avoid redundancy and preserves the actual left to right order of the actors.

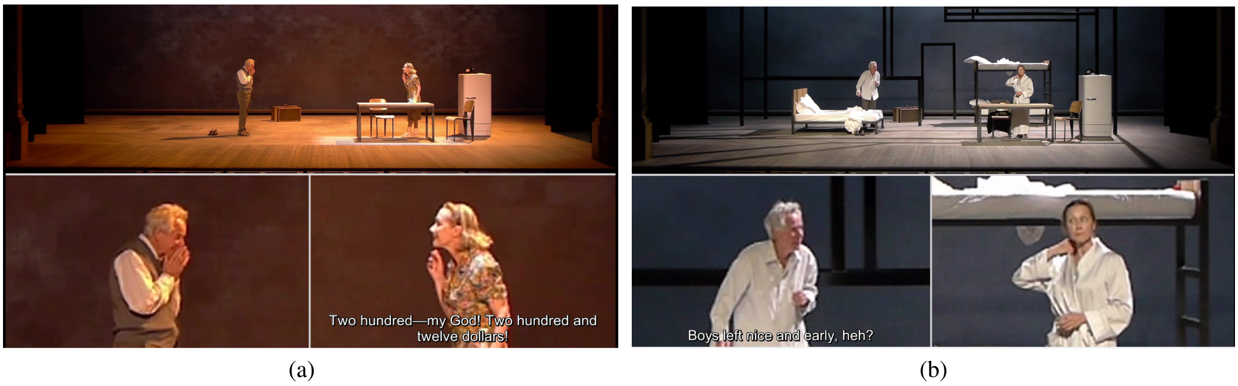

Two hundred—my God! Two hundred and twelve dollars!
Boys left nice and early, heh?
(a)
(b)

Figure 11: *Split screen compositions with subtitles placed selectively in the partition of the actor who is speaking. Such position aware subtitles can further enhance the perception of the staged events like theatre (usually recorded from a static wide angle camera), especially for the hearing challenged viewers.*

Camera movement: The centre x-coordinate trajectories of virtual framings in Figure 9 demonstrate the desired camera behavior comprising of smooth transitions between long static segments (when the camera moves, it moves smoothly or remains fully static). For example, observe how the virtual camera frame of Linda (blue trajectory) smoothly accelerates and then decelerates as she moves towards the fridge and comes to rest again. The optimization avoids jitter in the trajectories (due to small movements).

Composition: Examples *sequence3*, *sequence5* and *sequence7* in Figure 10 show that desired compositions are maintained even with large movements, interactions, and crossings of actors. The optimization algorithm avoids cutting the actors faces and maintains headroom, even with changes in aspect ratio; shot size or number of actors in a view. The composition is also maintained with significant changes in postures and poses (e.g., Figure 9 and *sequence6* in Figure 10). Integrating more information into the tracker like hand positions or motion saliency etc. can further help to improve the compositions for better perception of the gestures.

Viewing experience: It is evident in almost all examples, that it is difficult to perceive emotions and facial expressions in a wide shot of the entire stage. In fact, in some cases, it can even be difficult to recognize and locate speakers. As we observe in Figure 10, SSC's could be one way to resolve this issue as they clearly bring out the emotions and expressions of the actors. This makes our method attractive for digital heritage projects of the performing arts in general. Our method is also generally applicable to other domains such as sports and social events. One extremely interesting application of SSC is to enhance the viewing perception for hearing challenged people, with the help of partitioned subtitles (the current systems takes the subtitles file as input from the user). An example is illustrated in Figure 11. We can notice that, not only the SSC helps tracking the lip movements using zoomed-in views, putting the subtitles in correct partition can further help to enhance the understanding of the scene for hearing challenged viewers.

Performance with noisy actor tracks: In order to test the robustness of the system, we add synthetic noise to the input actor tracks to simulate the effects of inaccurate tracking. The noise is added in the form of uniformly distributed pseudorandom integers with sample interval of [-40, 40] pixels. We also add larger errors at random positions (shift up to 200 pixels) to simulate momentary tracking failures. We remove the hard constraint on actor bounding box in Equation 6 to allow convergence of the optimization function. The per frame close-up view estimation from noisy tracks are compared with the optimized ones in Figure 12. The figure shows that the optimized result (blue rectangle) is well composed even with large errors in tracking information where the per frame estimations (red rectangle) are completely off the target (for instance in frame2 and frame5). The video results for this experiment are provided in the supplementary material.

7. Limitations and future work

Our approach avoids making editorial decisions: it shows all actors, and considers the actors' faces important. Future extensions may select a subset of the actors and/or choose framings that consider important body parts (e.g., hands when an actor is gesturing). Similarly, our approach assumes that all actors are important. Future extensions may emphasize actors who are more important, for instance to highlight someone who is speaking. To date, we have focused on the application of staged performances which allow for some simplifying assumptions, such as a horizontal layout. Extending our approach more broadly may require relaxing these assumptions. Our method was tested and evaluated with qualitative, subjective assessments. We would like to better understand our systems performance with quantitative studies, as well as to understand the impact of SSCs on viewers.

Our method is currently limited to offline processing as the optimization must consider the entire video to determine a spatio-temporal optimum. An online variant would make the approach attractive for real-time applications such as live broadcast. At present, our implementation relies on an offline tracking [WSTS07] method based on a generative model of actor's appearances [GR13] which correctly tracks the actor's upper bodies but fails to detect their hands, which can cause artefacts. Tracking the actors pose would bring substantial improvements to our method.

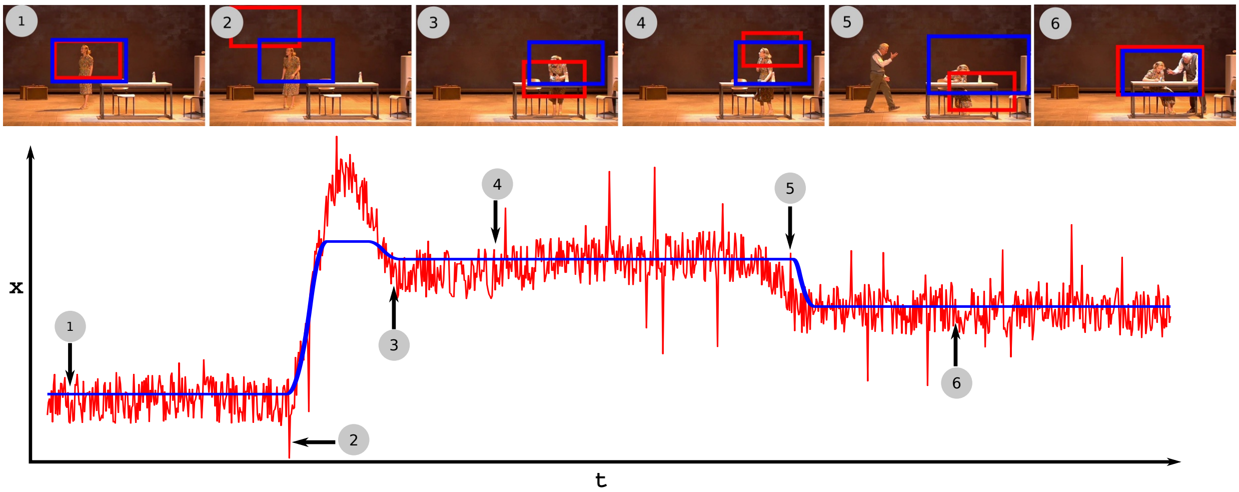


Figure 12: *The figure illustrates the performance of our system when the input actor position estimates are noisy. The x-coordinate of independent per frame close-up view estimation (red) for Linda is compared with the optimized version (blue) for sequence1. Some selected images with estimated close-up view bounding boxes are also shown.*

8. Conclusion

In this paper, we have presented an approach for automatically computing split-screen or multi-screen compositions from a single master shot. Our experiments show that this is a difficult task in general, requiring many adjustments for correctly matching the positions and movements of actors in the different subframes at all times, and controlling transitions from one screen layout to another, depending on the actors groupings. By carefully analyzing the constraints that must be observed, we have cast this problem of split-screen composition as a series of nested convex optimization problems, which have a unique solution and can be solved efficiently offline. Experimental results demonstrate the quality of the generated split-screen compositions in a variety of complex live action scenes and screen formats, making our method suitable for viewing ultra high definition video with improved comfort

Acknowledgements

This work was supported in part by an IIIT-H seed grant award, NSF award 1162037, a Mellon Foundation Award, ANR grant Spectacle-en-ligne(s), SCENOPTIQUE grant by Région Rhone Alpes and ERC advanced grant EXPRESSIVE. Special thanks to Claudia Stavisky, Auxane Dutronc and the cast and crew of 'Death of a Salesman' and 'Cat on a hot tin roof'.

References

[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).

[BT13] Bowen C. J., Thompson R.: *Grammar of the Shot*. Taylor & Francis, 2013.

[CDV10] Chen F., De Vleeschouwer C.: Personalized production of basketball videos from multi-sensored data under limited display resolution. *Computer Vision and Image Understanding 114*, 6 (2010), 667–680.

[CMM13] Carr P., Mistry M., Matthews I.: Hybrid robotic/virtual pan-tilt-zom cameras for autonomous event recording. In *Proceedings of the ACM international conference on Multimedia* (2013).

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

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

[GKE11] Grundmann M., Kwatra V., Essa I.: Auto-directed video stabilization with robust L1 optimal camera paths. In *CVPR* (2011).

[GL08] Gleicher M. L., Liu F.: Re-cinematography: Improving the camerawork of casual video. *ACM Trans. Multimedia Comput. Commun. Appl. 5*, 1 (2008), 1–28.

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

[GRG14] Gandhi V., Ronfard R., Gleicher M.: Multi-clip video editing from a single viewpoint. In *Proceedings of the 11th European Conference on Visual Media Production* (2014).

[HWG07] Heck R., Wallick M., Gleicher M.: Virtual videography. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 3*, 1 (2007), 4.

[JSSH15] Jain E., Sheikh Y., Shamir A., Hodgins J.: Gaze-driven video re-editing. *ACM Transactions on Graphics (TOG) 34*, 2 (2015), 21.

[LG06] Liu F., Gleicher M.: Video retargeting: Automating pan and scan. In *Proceedings of the ACM international conference on Multimedia* (2006).

[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 (2005), 981–990.

[SFW*13] Schreer O., Feldmann I., Weissig C., Kauff P., Schäfer R.: Ultrahigh-resolution panoramic imaging for format-agnostic video production. *Proceedings of the IEEE 101*, 1 (2013), 99–114.

[WSTS07] Wei Y., Sun J., Tang X., Shum H.-Y.: Interactive offline tracking for color objects. In *ICCV* (2007).

[YF05] Yokoi T., Fujiyoshi H.: Virtual camerawork for generating lecture video from high resolution images. In *Multimedia and Expo (ICME). IEEE International Conference on* (2005), pp. 4–pp.