Automatic Editing of Footage from Multiple Social Cameras

Ido Arev[1,3], Hyun Soo Park[2], Yaser Sheikh[2,3], Jessica Hodgins[2,3], and Ariel Shamir[1,3]

[1]The Interdisciplinary Center Herzliya [2]Carnegie Mellon University [3]Disney Research Pittsburgh

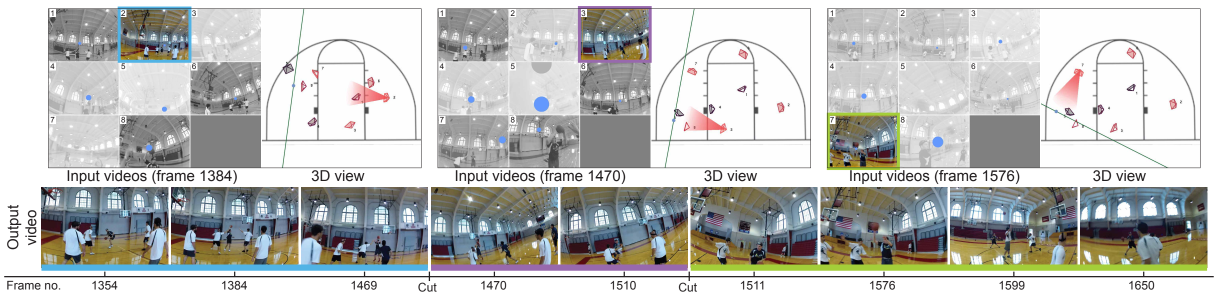


Figure 1: *A visualization of three shots from a coherent video cut of a social event created by our algorithm. In this case, eight social cameras record a basketball game. Their views are shown at three different times. The the 3D top-view shows the 3D positions of the cameras, the 3D joint attention estimate (blue dots), and the line-of-action. Using cinematographic guidelines, quality of the footage, and joint attention estimation, our algorithm chooses times to cut from one camera to another (from the blue camera to purple and then to green).*

Abstract

We present an approach that takes multiple videos captured by *social* cameras—cameras that are carried or worn by members of the group involved in an activity—and produces a coherent "cut" video of the activity. Footage from social cameras contains an intimate, personalized view that reflects the part of an event that was of importance to the camera operator (or wearer). We leverage the insight that social cameras share the focus of attention of the people carrying them. We use this insight to determine where the important "content" in a scene is taking place, and use it in conjunction with cinematographic guidelines to select which cameras to cut to and to determine the timing of those cuts. A trellis graph formulation is used to optimize an objective function that maximizes coverage of the important content in the scene, while respecting cinematographic guidelines such as the 180-degree rule and avoiding jump cuts. We demonstrate cuts of the videos in various styles and lengths for a number of scenarios, including sports games, street performance, family activities, and social get-togethers. We evaluate our results through an in-depth analysis of the cuts in the resulting videos and through comparison with videos produced by a professional editor and existing commercial solutions.

CR Categories: I.2.10 [Computing Methodologies]: Artificial Intelligence—Vision and Scene Understanding

Keywords: Multiple Cameras, Video Editing

1 Introduction

Cameras are now ubiquitous in our everyday lives—we are rarely without our smart phones and pocket-size camcorders. Recently, even wearable cameras have become quite common. These *social cameras* are used to record our daily activities: time with family and friends, participation in hobbies and games, and group activities such as concerts, sporting events, and parties. A given activity is often captured by multiple people from different viewpoints resulting in a sizeable collection of social camera footage even for just a single event. With millions of hours of video captured in this way each year, algorithms to effectively summarize and understand such content are urgently needed.

Social cameras create a new form of media as they are always available, generally handheld, and reflect the personal viewpoint of the camera operator. The footage from social cameras is quite different from what has traditionally been created by professional cameramen. Professionals use tripods or stabilized rigs and carefully compose and light their shots. In contrast, social camera footage contain an intimate view of the proceedings, often from a first person viewpoint. Large portions of social camera footage are rough and unstable. These differences present a challenge for editing such videos into a coherent "cut", and this challenge is amplified when large numbers of input video streams of a single event are available. In an experiment we conducted with a professional editor, the editing of just a few minutes of video from multiple social cameras required about twenty hours of effort.

This paper describes an algorithm that automatically creates a video "cut" of an activity (Figure 1) from multiple video feeds captured by a set of social cameras. To create this cut, we leverage and extend existing cinematographic rules that were designed to guide the editing of traditional footage into a narrative by human editors. Our key insight is to take the center of attention of the cameras (and therefore of the viewers) as a strong indicator of what was important at that moment in time. While this indicator does not describe a complex narrative flow as a human editor might (through editing tricks such as replays, close-ups, and flashbacks), it does provide the information needed to create a watchable cut of the video that condenses the video footage hundredfold or more while retaining the important action that occurred during the event.

The joint attention of a group is a powerful indicator because people engaged in a social activity naturally arrange themselves to ensure that everyone gets a good view of the activity. Individually, their orientation signals what they consider most interesting and worth attending to. When a member of such a social group records the activity via a camera, the camera inherits this social signal. When multiple social cameras observe the same activity, this signal can be consolidated to obtain what is called *gaze concurrence* or *3D joint attention*: a consensus measurement of the spatial and temporal location of the important "content" of an activity [Kim et al. 2010; Fathi et al. 2012; Park et al. 2012].

We use a graph-theoretic method to optimize an objective function guided by the traditional rules of cinematography and modified to deal with the unique properties of the new media created by social cameras. First, a graph is built, whose nodes represent the joint attention in the frames of each camera, and whose edges represent transitions between cameras. Then, an optimal path in this graph is found automatically using a modified dynamic programming algorithm. This path represents the final movie that is rendered by combining the footage from the cameras on the path. We show results on a number of activities, including playing sports, attending a party, and watching street performances, and produce coherent video cuts in various lengths and styles (e.g. including first person views or not).

Minimal human input to the system is optional – either just the desired length of the output or a threshold for "level-of-interest" for the creation of summarization video, or other parameters such as who is an "important characters" for the creation of a personalized video. We evaluate our results by dissecting the cuts chosen by our algorithm in various examples and through comparison with edits created by commercially available software, a random selection process, and a professional editor.

Contributions: We present a method for automatic editing of multiple social camera feeds. Although our automatic method does not have the superb story telling capabilities of professional editors, it creates a watchable, coherent video of a user-specified length from a large number of feeds that include intimate, first person views and a significant percentage of bad footage. Our contributions are threefold: (1) A demonstration that the center of attention of hand-held cameras can serve as an effective measure of what is important in a scene. (2) Establishing that the long-standing rules of cinematography can be adapted to footage produced from social cameras despite the significant differences between this new media and traditional video footage. (3) An optimization algorithm and objective function for finding a path in a trellis graph structure to produce watchable cuts of social camera footage in a variety of lengths and styles.

2 Related Work

Our approach is most closely related to work that selects from among multiple input videos based on content. A number of approaches attempt to recognize the activity of people in the scene and interpret these activities to determine the most appropriate camera view. Hata and colleagues [2000] presented a method to summarize multiple video feeds from surveillance cameras based on the activity of people in a each view. He and colleagues [1996] used finite state machines to decide when to select cameras in a virtual party scenario. This approach was applied to real data in a lecture setting [Rui et al. 2001]. Dale and collegues [2012] use frame similarity to align multiple videos and a similar dynamic programming algorithm to ours to choose parts and create a summarization. However, they do not handle social cameras nor use cinematographic guidelines in their summary. With his *Virtual Director* system for webcasts of lectures, Machniki [2002] used audio cues

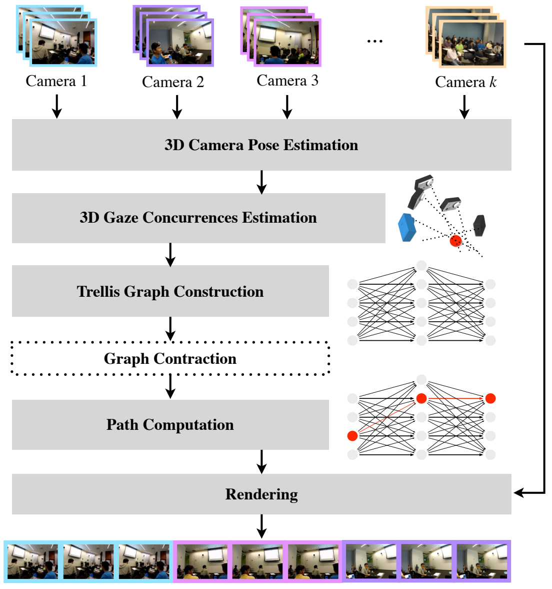

Camera 1
Camera 2
Camera 3
...
Camera *k*
3D Camera Pose Estimation
3D Gaze Concurrences Estimation
Trellis Graph Construction
Graph Contraction
Path Computation
Rendering

Figure 2: *Our method's pipeline: from multiple camera feeds to a single output "cut".*

and an automation specification language to allow a user to automate when to cut between audience and speaker cameras. Heck and colleagues [2007] presented a system to automatically produce an edited video of a classroom lecture from a single camera feed, selecting from different virtual cameras created by 2D zooms and pans in image space. The selection was done by analyzing the content on the board. In a series of papers, Takemae et al. [2004] used head orientation to determine camera selection in meetings. These systems work well but only in controlled settings such as classrooms or meeting rooms where the variety of content is limited.

Other efforts focus only on assessing the quality of the available footage or leverag user input. [Bao and Roy Choudhury 2010] present a method to select a representative video based on footage attributes but do not use cinematographic rules. Sumec [2006] collated multiple video feeds of a meeting based on rules for the quality of each feed. Shrestha and colleagues [2010] create a "mashup" of multiple video feeds by optimizing a cost function composed of several low-level video measures and user-defined scores. Zsombori and colleagues [2011] presented "MyVideos", a semi-automatic system to align and select videos using Narrative Structure Language (NSL) for multi-media items. A number of commercial products exist today that combine footage from multiple cameras to produce a coherent video (Vyclone http://www.vyclone.com or Switchcam http://make.switchcam.com). Their solutions appear to be limited to choosing the most stable or best lit view and periodically switching between views. Because these algorithms do not know the spatial relationship of the cameras to the action, they cannot take into consideration higher level cinematographic guidelines such as jump cuts and the 180-degree rule.

Our approach is also related to video summarization as we condense multiple videos to produce an edited version. However summarization is usually performed on just a single video stream. Many techniques for summarization have been proposed [Ponto et al. 2012; Lee et al. 2012; Kumar et al. 2010; Money and Agius 2008; Truong and Venkatesh 2007; Taskiran and Delp 2005; Barbieri et al. 2003]. Most approaches are based on some level of video understanding such as human detection, visual saliency, or video quality analysis. Some methods handle specific types of video such as interview videos in [Berthouzoz et al. 2012] or egocentric footage in [Lu and Grauman 2013]. The first presents a nice tool that links text to video, allowing the video to be edited simply by editing the text transcript. The second presents a nice advance towards higher level story-driven summarization by segmenting the continuous video into sub-shots and building chains of sub-events that "lead to" each other.

The idea of using joint attention to identify salient locations has been investigated in a number of papers. Kim and colleagues [2010] present an approach to use the convergent motion of soccer players to identify areas of importance on the field. Fathi and colleagues [2012] use convergence of attention to identify and analyze social interactions in first person cameras. Jain and colleagues [2013] recognizes shared content in videos and joint attention by finding the number of the overlapping 3D static points. In this paper, we leverage the approach of Park and colleagues [2012], that present a method to find spatiotemporal points of saliency in a scene by analyzing the convergence in the fields of view of multiple first person cameras. We explore the use of such 3D joint attention points to select and time the transitions between multiple feeds.

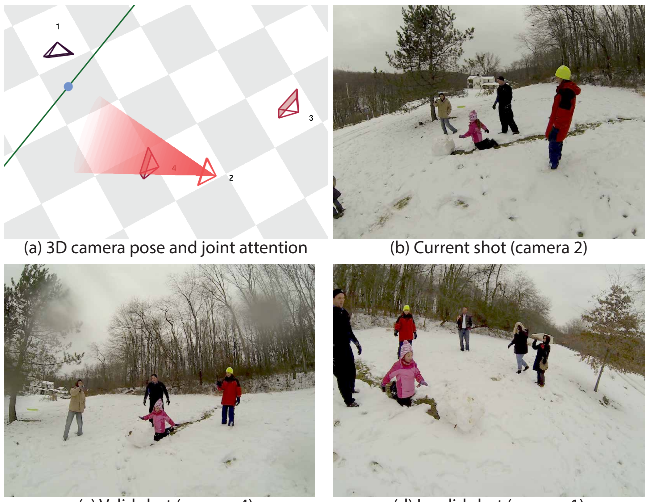
3 Overview

The input to our algorithm is k synchronized video feeds of the same event or scene, and the output is a single edited video that is created from the k videos and best represents the event or scene captured according to the objective function. The algorithm pipeline is illustrated in Figure 2. First, at each time instant, we reconstruct the 3D positions of the cameras and estimate the 3D joint attention of the group based on [Park et al. 2012] (Section 5). Over time, this provides the 3D camera trajectories and joint attention for all the footage. Next, we construct a trellis graph and calculate the weights for all nodes and edges based on cinematographic guidelines and style parameters (Section 6). If summarization or personalization style parameters are given, we apply contraction operations to the graph (i.e., remove some of its nodes and edges—-Section 9). Next, we use dynamic programming to find the best route in the trellis graph from the first slice to the last (Section 7). Such a path defines a single output video containing appropriate cuts between cameras.

4 Adapting Cinematographic Guidelines

Edward Dmytryk, a pioneer of Hollywood film directing and editing, wrote: *"If the film is well shot and well cut, the viewer will perceive it ... to flow in continuous, unbroken movement on a single strip of film"* [Dmytryk 1984]. Raw footage from multiple cameras rarely produces such continuous flow and must therefore be heavily edited. As we start from handheld or body-worn social cameras footage created by non-professionals, we must filter out much of it because of low quality, and we must choose and cut between cameras following cinematographic editing guidelines to achieve a single coherent and smooth viewing experience. Below we describe how these guidelines are applied and adapted to our settings.

Content, then form. Dmytryk's first rule is *content, then form*. This rule demands a semantic understanding of the video footage, however acquiring a high-level understanding of activities using automatic video processing remains a challenge. We leverage the fact that the collective ego-motion of multiple social cameras encodes the judgment of the social group about which areas in the scene are significant. We posit that areas that attract greater attention have greater salient content. Thus, we use the joint attention of social cameras, as estimated by concurrences in 3D camera gaze vectors, to determine significant content and guide our basic editing choices (Section 5). Furthermore, as described in Section 6, we use the



(a) 3D camera pose and joint attention (b) Current shot (camera 2) (c) Valid shot (camera 4) (d) Invalid shot (camera 1)

Figure 3: *The 180-degree rule is defined by an imaginary line in the scene. A cut from one camera to another should not cross this line as it will confuse the viewer by reversing directions. If we use the green line in (a) for the shot of camera 2 (seen in b), then camera 4 (seen in c) and camera 3 are valid to cut to, while cutting to camera 1 (seen in d) will violate the rule—note how the snow ball moves from left to right instead of right to left.*

graph node costs to represent the quality of the footage to direct the algorithm to choose higher quality frames.

Jump Cuts. A core guideline in video editing is to avoid *jump cuts*. These are transitions between two cameras that shoot the same scene from almost the same angle (e.g., below 30 degrees difference), or have a noticeable portion of overlap in their frames. When choosing to cut from one camera to another, our algorithm favors a significant difference in frame content and angle.

180-degree rule. Another fundamental guideline of editing is the *180-degree rule*. In its strict form, this guideline keeps the subject of the scene on one side of an imaginary line-of-action, as shown in Figure 3. Shooting the same subject from opposite sides creates an abrupt reversal of the action and can confuse the viewer. This guideline is important in our setting as social cameras can often be found on both sides of this line. For highly dynamic events, we also explore the use of a relaxed version of this guideline. We allow the camera to move from one side of the scene to the other in small steps across multiple cuts. Viewers are able to adjust to the gradually changing viewpoint without confusion. Note that applying the 180-degree rule and avoiding jump cuts both necessitate a 3D understanding of the configuration of the cameras.

Shot Selection. Selecting the correct camera view and the time to cut to it is critical in maintaining the perception of continuity. We determine cut timing and select cameras based on the following guidelines.

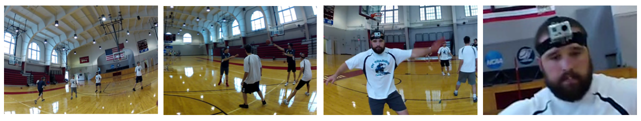

(a) (b) (c) (d)

Figure 4: *Shot sizes: (a) wide shot, (b) long shot, (c) medium shot, and (d) close-up shot.*

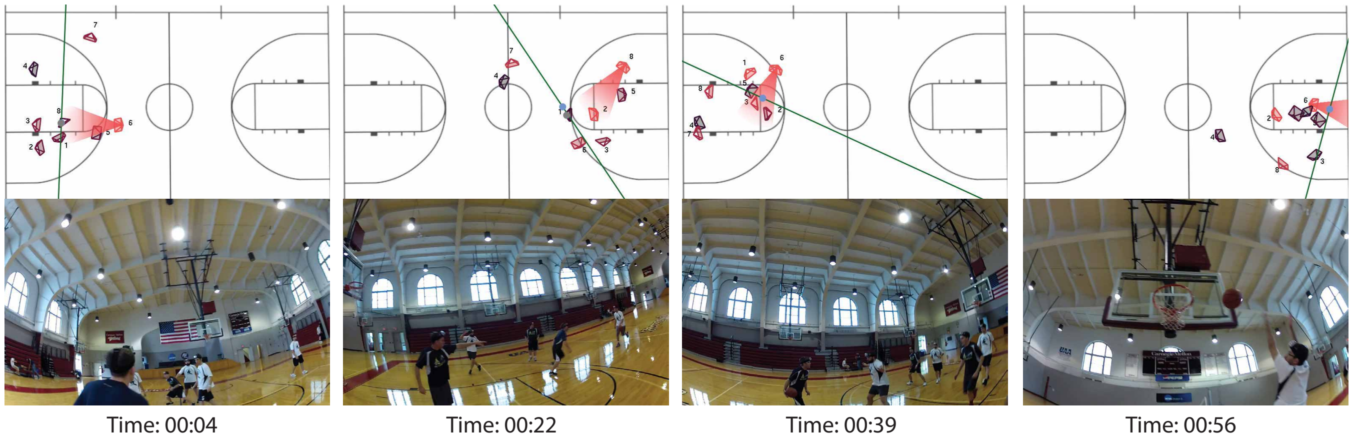


Figure 5: *3D reconstruction of the camera position and orientation allows our algorithm to track the joint attention through the basketball game. The blue dot in the top view of the reconstructions is the joint attention point. One can see how most players are watching the same position in the scene (usually the ball position), which is the focus of joint attention during the game. The green line is the 180-line according to the current camera (see also Figure 1).*

Shot Size Diversity: Controlling *shot size* transitions promotes both smoothness and interest in a video. Shot size is defined as the size of the frame with respect to the main subject—the subject can be seen from afar or closer up. There are five major types: wide shot, long shot, medium shot, close-up, and extreme close-up (Figure 4). We determine shot size by measuring the distance to the 3D joint attention. Using a single shot size throughout the movie is boring, while jumping too often or in steps that are too large is distracting. We create two shot size alternatives from each original frame by cropping, and promote choosing a diverse set of sizes during movie creation (Figure 11).

Shot Duration: Varying *shot duration* also affects smoothness and interest. Very short-duration shots can be disturbing, while long-duration shots can be boring. We set a minimum and maximum shot duration. Our algorithm then prevents cuts before the minimum duration has elapsed and promotes a cut when the maximum duration is reached (Section 7).

Shot Quality: Each individual camera feed should also be weighted for smoothness and aesthetics. In this work we concentrate on choosing stabilized shots and create better compositions using cropping (see Section 6.1). Other characteristics such as focus and lighting could be included in the node costs of each frame to allow the algorithm to pick better shots.

Cut-on-Action. Using characters' action/movement, editors match between two shots to help hide cuts. By cutting exactly at the height of action (when the door is shut or the ball hits the bat etc.), the action itself distracts the audience from the actual cut. Although our algorithm does not have an exact understanding of the action in the scene, we use an approximate measure to mimic these types of cuts. We track the movement of the 3D joint attention: whenever there is a large acceleration of this movement, we take this to mean that some action has happened that changed the center of attention abruptly, and the algorithm tries to cut to a different camera. This guideline is more appropriate for some types of footage than others and the user can enable/disable this guideline.

5 Content from 3D Joint Attention

To identify which views best capture the content of the activity, we need a semantic representation of the activity in the scene. However, algorithms capable of interpreting social activity from cameras remain an active area of research. As a proxy for a semantic understanding, we use the 3D location of joint attention of the cameras. A feature of social cameras is that they inherit the gaze behavior of their users: people tend to point their cameras at what they find interesting in a scene. We leverage this phenomenon by explicitly computing the 3D motion of the k cameras and analyzing their collective motion to identify where they focus their attention. We use a standard structure-from-motion algorithm [Snavely et al. 2006] to simultaneously reconstruct the 3D structure of the scene and the motion of the cameras using a subsampled set of the camera frames. Using the reconstructed 3D structure, we estimate the pose of each camera using a perspective-n-point algorithm [Lepetit et al. 2009]. However, because of imaging artifacts, such as motion blur, rolling shutter, and pixel saturation, there are a significant number of frames for which we cannot recover the camera pose directly.

To handle these missing frames, we employ data-driven camera motion interpolation. Between any two cameras that have been posed, which we refer to as anchor cameras, the interpolation is performed by estimating the frame-to-frame fundamental matrix at each consecutive frame, which gives us the relative orientation and translation up to scale. This relative information corresponds to differential constraints on the camera motion. Our approach then interpolates the camera translation and orientation using a Discrete Cosine Transform basis, with gradient constraints defined by the relative orientations and translations. This computation provides the 3D pose of each video camera through time (Figure 5). Once we have the 3D trajectory of each camera, we use the gaze clustering algorithm of Park et al. [2012] to extract 3D points of joint attention (JA-point) in the scene at each time instant. We calculate all gaze concurrences g in the scene through time, and rank their importance rank(g) by counting the number of camera gazes that intersect at that point. Thus, this process produces multiple 3D locations of joint interest and our algorithm uses them all during its reasoning about producing the cut.

6 Trellis Graph Construction

To represent the elements in our optimization, we use a trellis graph $G_T = \{V, E\}$, where the node set V contains T time-slices: $V = \bigcup_t \{S_t | 1 \leq t \leq T\}$. Every slice contains a set of nodes, where each node is defined by pair (C_i, ja^t_j), associating a camera and a 3D JA-point (e.g., a point of 3D joint attention as estimated via gaze

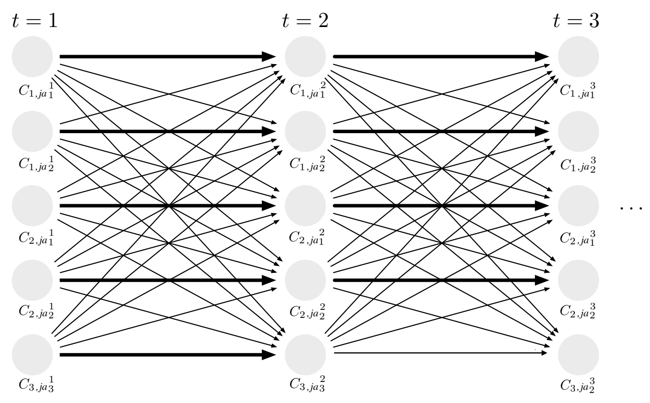


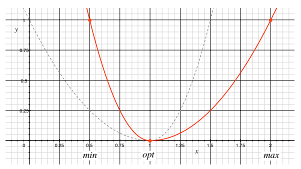
Figure 6: *The trellis structure used in our algorithm; the frames are nodes* $\left(C_j, ja_i^t\right)$ *for time t, camera* C_j *and 3D joint attention* ja_i^t *with two types of edges: continuous (bold) and transition edges. In this example, at each time t there are three synchronized camera sources* C_1*,* C_2*, and* C_3 *and there are three joint attention points* ja_1*,* ja_2*, and* ja_3*. Assume that* ja_1^t *and* ja_2^t *project onto* C_1 *and* C_2 *and* ja_3^t *on* C_3*, for* $t = 1$ *and 2. Further, assume that the third point of 3D joint attention* ja_3^3 *ceases to exist as a point of interest at* $t = 3$*, and camera* C_3 *views* ja_2^3 *instead. The graph contains a set of five nodes for every slice* S_t *as follows:* $S_t = \{(C_1, ja_1^t), (C_1, ja_2^t), (C_2, ja_1^t), (C_2, ja_2^t), (C_3, ja_3^t)\}$ *for the first two time instances, and* $S_3 = \{(C_1, ja_1^3), (C_1, ja_2^3), (C_2, ja_1^3), (C_2, ja_2^3), (C_3, ja_2^3)\}$ *for* $t = 3$*.*

concurrences). The edges E in the graph connect all nodes in slice S_t to nodes in slice S_{t+1} for all times $1 \le t \le T-1$. Figure 6 illustrates a graph with three synchronized camera sources and three JA-points. Each JA-point defines a different narrative allowing us to create various video "stories". To allow different zoom or cropping options for a cut we clone each node for each available size. To create a crop, we define the cropping center to be the node's point of interest (the 2D projection of the 3D joint attention point). To retain enough resolution, we use only two possible crops: the original size and a zoom-in (e.g., long shot cropped to medium or medium shot cropped to close-up). This process enables smoother and more interesting cuts [Heck et al. 2007], as the algorithm can choose the zoom as part of the main path calculation, avoiding large changes in zoom but adding interest by changing shot sizes.

We now describe the different node and edge weights we use in the graph. The weights are defined as cost functions in which the cost is higher as the given input parameter deviates from a predefined optimum value. To normalize them to the same range of values and to define the desired shape of a cost function, we use a piecewise quadratic function:

$$N(min, max, opt, x) = \begin{cases} \left(\frac{x - opt}{opt - min}\right)^2 & min \le x < opt \\ \left(\frac{opt - x}{max - opt}\right)^2 & opt \le x \le max \\ \infty & \text{otherwise.} \end{cases} \quad (1)$$

This function has a minimum at the optimum input value opt, and maps any $[min, max]$ range of values to $[0, 1]$. In the figure on the right, the red curve shows the normalization function and the grey dotted line shows the rest of the half-functions used to produce the normalizing functions. The three operating limits are highlighted by red dots, where in this example $min = 0.5$, $opt = 1$, and $max = 2$.



6.1 Node Costs

The node cost $W_n(n_i) : V \to [0, 1]$ combines various parameters of the node's n_i camera frame. It is defined as:

$$W_n(n_i) = \frac{\lambda_s \cdot W_s(n_i) + \lambda_r \cdot W_r(n_i) + \lambda_{ja} \cdot W_{ja}(n_i) + \lambda_v \cdot W_v(n_i)}{\lambda_s + \lambda_r + \lambda_{ja} + \lambda_v}, \quad (2)$$

where $\lambda_s, \lambda_r, \lambda_{ja}, \lambda_v$ are the weighting factors for the different terms which are based on **S**tabilizing, **R**oll, **J**oint **A**ttention, and global **V**ector constraints. We use $\lambda_s = 1$, $\lambda_r = 1$, $\lambda_{ja} = 3$, and $\lambda_v = 1$, if the global 180-degree rule vector is enabled, or $\lambda_v = 0$ otherwise.

Stabilization Cost. To limit shaky camera movement, we constrain the camera movement frequency in $W_s(n_i)$; lower frequency motion gives a lower cost. Unlike conventional methods to measure video frequency based on image processing (e.g., optical flow), we use the estimated 3D motion of the camera (Section 5) to ensure that scene motion does not disturb the stability measure. We use a measure of the rotation frequency of each camera coordinate system along with a measure of camera movements. For instance, if we have a head-mounted camera and the head is undulating at a high frequency, the camera's forward and up vector, as well as the camera's position will present a high frequency change, yielding a jerky video. We measure the frequency of these geometric values for each node n_i in a window of ten consecutive frames and refrain from cutting to a camera if its frequency is too high in this duration. The stabilization cost is then:

$$W_s(n_i) = N\left(0, max_f, 0, frequency(n_i, k)\right), \quad (3)$$

where $frequency(n_i, k)$ measures the average frequency of the forward, right, and up vector and camera movements of node n_i in a window of k consecutive frames.

Camera Roll Cost. Extreme viewing angles are difficult to interpret and therefore we constrain the camera roll using our normalization function:

$$W_t(n_i) = N(0, 90, 0, roll(n_i)), \quad (4)$$

where $roll(n_i)$ is the roll angle of node's n_i camera. When the camera is perfectly aligned to the horizon then $roll(n_i) = 0$.

Joint Attention Cost. We use three parameters to define the cost of a 3D JA-point: its ranking, the 2D location of its projection in the frame, and its 3D distance from the camera:

$$W_{ja}(n_i) = \text{rank}(ja)^{-2} \cdot \frac{1}{2}(W_2(n_i, ja) + W_3(n_i, ja)), \quad (5)$$

where ja is the 3D joint attention point of node n_i, rank(ja) is the ranking of ja according to how many members of the social group are participating in the JA-attention. $W_2(n_i, ja)$ is a "validity" check to ensure the 2D projection of the JA-point lies within the usable field of view of the camera of n_i. If this projection lies outside the 10% margin of the frame (or if there is no projection of this JA-point in node n_i), then $W_2(n_i, ja) = 1$, otherwise we set $W_2(n_i, ja) = 0$. We limit the position of the projection not to lie in the margins of the frame as we usually crop around this position: to center the frame around the main point of interest of the narrative, to stabilize shaky footage, to reduce distortion in wide FOV cameras, or to create more aesthetic compositions.

$W_3(n_i, ja)$ is the cost of the 3D distance between the JA-point and node's n_i camera center:

$$W_3(n_i, ja) = N\left(min_{jad}, max_{jad}, opt_{jad}, dist(n_i, ja)\right), \quad (6)$$

where $dist(n_i, ja)$ is the normalized distance between the location of the JA-point and the camera of n_i, and $min_{jad}, max_{jad}, opt_{jad}$ are the normalized minimum, maximum, and optimum distances in 3D respectively. We use $min_{jad} = 0$ feet, $max_{jad} = 100$ feet, which gives us wide shots, and $opt_{jad} = 20$ feet which is a medium shot size. These constraints eliminate cameras that are too far or too close. W_3 also controls the use of first-person view-point in the movie. For example, increasing min_{jad} will eliminate such viewpoints from the resulting movie altogether.

Global Vector Cost. This cost is defined by a global vector $\vec{v}$, favoring cameras from a certain direction. For instance, in a basketball game, if a global 180-degree rule is desired, the user can set $\vec{v}$ to a line passing through the center of the court. If a story from the point of view of the attackers in the game is desired, the user can set $\vec{v}$ to be a static vector from the center of the field to one of the baskets. Our algorithm can also handle a dynamically changing vector by checking the current direction of the camera or averaging the movement direction of the players. This cost is computed by measuring the angle $angle(n_i, \vec{v})$ between any node n_i camera's forward vector and the vector $\vec{v}$. We use our normalization function where the optimal angle is 0:

$$W_v(n_i) = N(-180, 180, 0, angle(n_i, \vec{v})). \quad (7)$$

6.2 Edge Weights

The edge weight $W_e(n_i, n_j) : E \to \mathbb{R}$ depicts the cost of transition from node n_i to node n_j. We distinguish between two types of edges: *continuous* edges and *transition* edges. A continuous edge is defined as an edge between two nodes that represent the same camera, point of interest, and zoom level (see bold edges in Figure 6), while all other edges are transition edges. Each type of edge carries a different weight definition.

Continuous Edge Weight. Because following a continuous edge does not produce a real cut, the cost of following this edge comes only from the camera movement: $W_e(n_i, n_j) = W_{ca}(n_i, n_j)$. High-rate camera movement is confusing and we want to avoid it. Hence, we constrain the absolute angle difference $angle(n_i, n_j)$ between the two front vectors of adjacent frames in n_i and n_j of the same video feed:

$$W_{ca}(n_i, n_j) = N\left(0, 180, 0, angle(n_i, n_j)\right). \quad (8)$$

Transition Edge Weight. Transition edges produce real cuts in the output video and hence should follow cinematographic guidelines. We use a weighted combination of three parameters to score the likelihood of a transition:

$$\begin{aligned} W_e(n_i, n_j) = \quad & \alpha \cdot W_{ta}(n_i, n_j) + \beta \cdot W_{td}(n_i, n_j) + \\ & (1 - \beta - \alpha) \cdot W_{tz}(n_i, n_j) - W_{ts}, \end{aligned} \quad (9)$$

where $0 \leq \alpha, \beta \leq 1$ are the weighting factors between angle weight W_{ta}, position weight W_{td}, and shot size weight W_{tz}. W_{ts} is a cut-on-action transition factor, explained below.

To avoid jump cuts when shooting the same JA-point from two different cameras and to follow the 180-degree rule, we constrain both the transition angle $angle(n_i, n_j)$ and the distance $dist(n_i, n_j)$ between the two cameras. The first ensures small overlap between their frames and different background arrangements, and the second ensures small angle change between the transition frames. We use our normalization function to define the cost of $W_{ta}(n_i, n_j)$ and $W_{td}(n_i, n_j)$:

$$\begin{aligned} W_{ta}(n_i, n_j) &= N\left(0, max_{ta}, opt_{ta}, angle(n_i, n_j)\right), \\ W_{td}(n_i, n_j) &= N\left(0, max_{td}, opt_{td}, dist(n_i, n_j)\right), \end{aligned} \quad (10)$$

where we use transition angle parameters $max_{ta} = 180, opt_{ta} = 30$. We empirically found the best distance for social events is $opt_{td} = 20$ feet, while distance greater than $max_{td} = 100$ feet prevents the viewer from perceiving the social scene details. This threshold depends on the camera specifications to some extent.

For $W_{tz}(n_i, n_j)$ we identify the size of each shot as wide, long, medium, close-up or extreme close-up (Figure 4). This is done according to the distance from the JA-point. Transitions between shots whose sizes are more than two levels apart can be confusing for the viewer. Hence, we set $W_{tz}(n_i, n_j) = 1$ for transitions differing in more than two levels and $W_{tz}(n_i, n_j) = 0$ otherwise.

Lastly, we want to promote cut-on-action transitions. Because our center of action is the "joint attention" point of social cameras, we use the acceleration (rate of change of speed) of the 3D JA-point as a measure for action change. We normalize this value to be between 0 and 1, and track each JA-point through time. For each, we find the local maximum in regions where the speed is above a certain threshold (0.7). We set the value of W_{ts} in all transition edges between nodes of the respective JA-point at this specific time to the value of the normalized speed value. For all other edges W_{ts} is set to zero.

7 Path Computation

A path in the trellis graph G_T starting at the first slice and ending at the last defines an output movie whose length matches the length of the original footage (Figure 8). Following continuous edges in the path continues the same shot in the movie, while following transition edges creates a cut. The cost of a path is the sum of all edge weights and node costs in the path. Once our graph is constructed, we can find the "best" movie by choosing the lowest cost path. Although there are an exponential number of paths, Dijkstra's algorithm could provide a feasible solution. However, there are other constraints we would like to impose on the output movie and its individual shots. One of them is the duration of shots to avoid boring long-duration shots and jumpy short ones. Unfortunately, the duration of the shot cannot be mapped to edge or node weights. We solve this by using a dynamic programming algorithm that takes into account not only the path cost but also the shot length.

The algorithm proceeds by filling a cost table ϕ, where each cell $\phi(n_i^t) = \left\langle D_i^t, P_i^t \right\rangle$ of node $n_i^t \in S_t$ for $1 \leq t \leq T$ is a pair containing the current shot duration D_i^t and the accumulated cost of the path up to that cell P_i^t. The table is filled from left to right starting at the first column representing slice S_1 and ending at the last representing S_T. Then, we backtrack from the cell containing the smallest cost to recover the full path.

Assume that min_l and max_l are the minimum and maximum constrained shot length respectively. The first column ($t = 1$) just contains the node costs and a duration of 1:

$$\phi(n_i^1) = \left\langle 1, W_n(n_i^1) \right\rangle. \quad (11)$$

For every cell $\phi(n_i^t), 1 < t \leq T$ we first examine all cells in the previous column excluding row i, which have a duration larger than min_l and find the one with the smallest cost:

$$k = \arg\min_{j \neq i} \{P_j^{t-1} | D_j^{t-1} \geq min_l\}. \quad (12)$$

Now, if the previous cell in the same row has lower cost than P_k^{t-1}, and its duration is not larger than max_l (i.e., if $P_i^{t-1} < P_k^{t-1}$ and $D_i^{t-1} < max_l$), then we continue with the same shot as follows:

$$\begin{aligned} P_i^t &= P_i^{t-1} + W_e(n_i^{t-1}, n_i^t) + W_n(n_i^t) \\ D_i^t &= D_i^{t-1} + 1. \end{aligned} \tag{13}$$

If not, then we introduce a cut (from row k to row i) and a new shot is started:

$$\begin{aligned} P_i^t &= P_k^{t-1} + W_e(n_k^{t-1}, n_i^t) + W_n(n_i^t) \\ D_i^t &= 1. \end{aligned} \tag{14}$$

If the previous cell in the same row has a duration equal to max_l but no other cell in the previous column has a duration larger than min_l then we still continue with the same shot as in Equation 13. Finally, we fill this cell's values $\phi_i^t = \langle D_i^t, P_i^t \rangle$, and continue to the next cell in the same column or to the first cell in the next column.

Each cell in the table corresponds to one node in the graph, and for each cell, we examine all cells in the previous column. If $s = \max_t(|S^t|)$ is the maximum number of nodes in each slice, then the running time complexity of the algorithm is $O(|V| \cdot s)$. This computation is linear in the number of nodes since $s << |V|$. Note that because of the added constraint on shot length, the algorithm does not guarantee a global minimum on the path. However, in practice we found this approach provides satisfactory results.

8 Stylistic Control

Our algorithm allows significant stylistic control over the length and appearance of the edited video.

Length Control. To accommodate a different audience or to filter uninteresting or unwatchable parts, we define an importance measure that reflects how interesting or important the action in an event is. The user can control the length (in time) of the cut in two ways: either by choosing an importance threshold, resulting in the algorithm filtering out any part that falls below the threshold, or by designating the approximate output length desired.

There are many ways to define an importance measure. We utilize the JA-points ranking to define the measure of interest. For each slice S_t in our trellis graph, we define the importance by averaging the square of the JA-points ranking of its nodes. We use the square of the ranking to emphasize more important JA-points. Assume $S_t = \{(C_1^t, ja_1^t), \ldots, (C_k^t, ja_k^t)\}$ are the nodes in slice S_t where $\{ja_1, \ldots, ja_k\}$ are their JA-points, then we define: $A(S_t) = \frac{1}{k}\sum_{i=1}^{k} \text{rank}(ja_i)^2$. To create a smoother version of the importance function, we average the measure over time in a window of size $2w+1$ (we use $w = 7$):

$$\text{importance}(t) = \frac{1}{2w+1} \sum_{i=-w}^{w} A(S_{t+i}). \tag{15}$$

Next, we normalize the importance measure by dividing it by its maximum value for all slices to arrive at a measure between 0 and 1. To create shorter versions of a video, the user chooses an importance threshold $f, 0 \le f \le 1$, and the algorithm removes from the graph all consecutive slices of length $> m$, whose importance falls below the threshold f. The integer m reflects the minimum number of consecutive slices we allow to be removed. This minimum is required as removing very short segments from a video will often create jump cuts (we also allow trimming any number of slices at the beginning and end of the footage). Before running the path computation algorithm, the two slices on both sides of a removed portion are connected using only transition edges (there are no continuity edges between these slices). To create output of a given length, the algorithm continues to raise f and trim the graph until it reaches the desired length. Then, the path computation algorithm is executed as usual on the contracted graph. This operation will create a shorter output video, where only important parts of the scene are included.

Multiple Sub-Scenes Control. In cases when there is more than one center of attention in a scene, several clusters of JA-points that are far apart will be introduced. The user can decide if the resulting cut should include all, or some, of these clusters. To remove undesired sub-scenes from the movie, we simply remove their respective nodes from the graph reducing the graph in breadth but maintaining the same length. We can create a separate movie for each sub-scene by retaining only its relevant nodes in the graph. To merge several sub-scenes together into one movie, we add a minimum scene length constraint to the path computation algorithm that imposes a minimum number of seconds for continuous shots taken from one sub-scene. This guideline prevents jumping from one scene to another in the video, and allows the creation of a video that merges several scenes happening simultaneously.

First Person Viewpoint Control. As mentioned earlier, hand-held or wearable cameras often include first-person viewpoint shots. This type of viewpoint is very different from the ones used in classic narrative photography, but creates a unique style (see [Wardrip-Fruin and Harrigan 2004]). Therefore, we allow the user control over the use of such shots. We measure the distance of the cameras from the current JA-point. When first-person viewpoints are undesirable, we refrain from using cameras whose distance is too small, otherwise we allow the use of all cameras, including first-person viewpoint shots in the cut.

Algorithm Parameters Control. Additional style control can be achieved by setting different parameters of the algorithm. These include the use of the soft 180-degree rule. Using, or not, the cut-on-action mechanism to bias the results towards cutting when there is a rapid change in the joint attention, as well as using, or not, cropping of the footage to allow diversity in sizes of shots.

9 Results

We have tested our algorithm in various types of scenarios, including sports activities, artistic performances, and social and family gatherings. These experiments involved both hand-held and head-mounted cameras including GoPro HD Hero2, GoPro HD Hero3 (black), as well as cell phone cameras. In addition, we have applied our algorithm to three previously published datasets [Ballan et al. 2010; Park et al. 2012] arriving at a total of ten different scenes. The number of cameras used in the experiment varied between three to eighteen. To find the positions and directions of cameras, we use a computer vision algorithm that reconstructs the whole scene with computation time of several hours, after which we can apply our algorithm. Figure 7 gives the list of results, along with the number of cameras, graph statistics, and timing of the different parts of our editing algorithm. Results are included (and best seen) in the companion video and supplementary material.

We evaluate our results by comparing them to two extreme alternatives: a baseline method of cutting every three seconds to a randomly chosen camera, and movies created manually by a professional movie editor. In addition, we compared some of our results to the output of a commercial application (we chose Vyclone – http://www.vyclone.com — as the most popular social-video app).

To create a professionally edited version, we provided the raw in-

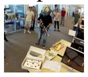
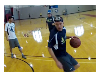
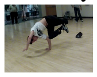
Scene	Surprise 	Fire eating	Basketball 	Snowman	Four scene	Juggling	Bboy 	Rothman	Park	Croquet
Cameras	12	6	8	4	11	6	16	3	6	7
Frames	1901	2150	2081	203	751	1501	281	2151	471	476
Output	64s	72s	70s	67s	150s	60s	93s	86s	47s	119s
Calc FPS	30	30	30	3	5	25	3	25	10	4
Graph Const.	14.1s	8s	11s	4s	6s	9s	2s	9s	5s	12s
Path Comp.	53.3s	17s	16s	0.5s	4s	2s	1s	5s	1s	1s
Rendering	7.2m	4.3m	8m	7.5m	17m	6m	8m	7.6m	5.2m	13m

Figure 7: *Statistics and timing results (s = seconds, m = minutes) on all the scenes used in our experiments. Note that these are timings of our algorithm execution, after we have the estimation of the 3D position and direction of the cameras.*

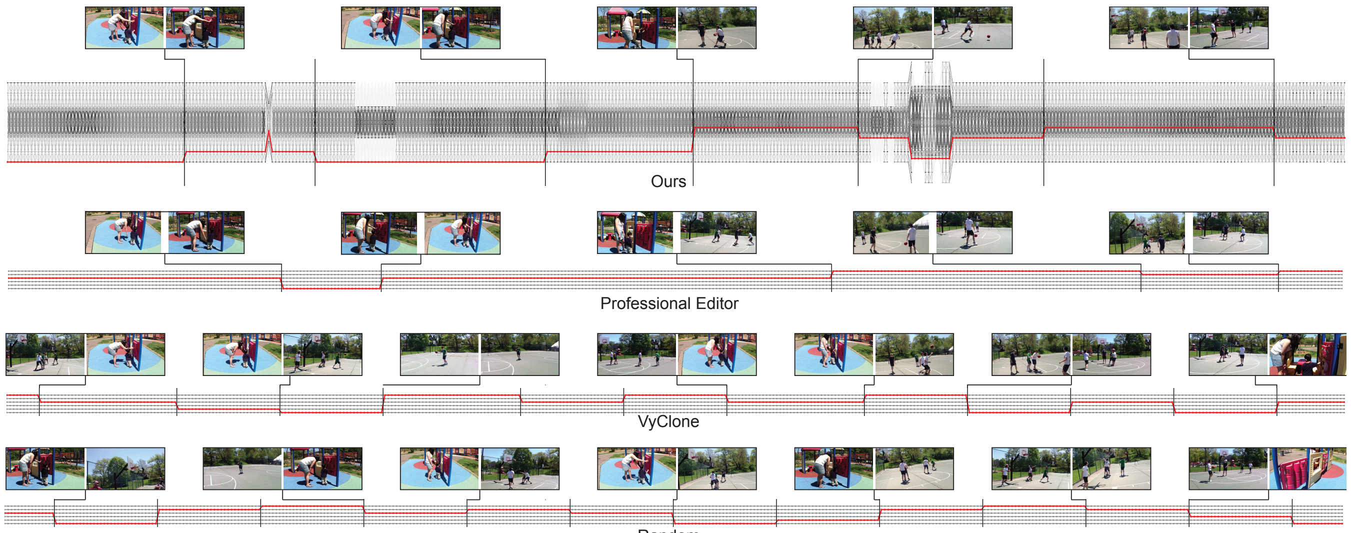


Figure 8: *Visualization of the movie cut created by different methods for the Park sequence where two distinct sub-scenes are taking place very close to each other in the same area: a basketball game and a kid playing in a playground. Our trellis graph is portrayed below the images and the path chosen is shown in red. The graph includes a node for each point of 3D joint attention and each zoom level and expands when more points of joint attention are found. For the other methods, the graph only includes nodes for each camera. Note that the edited videos produced by Vyclone and the Random method look similar in that the time between cuts is almost constant rather than changing to reflect the pace of the action. Neither approach handles this scene well because the edited videos cut three or four times from one sub-scene to the other, creating a confusing version of the event. In our edited video and the hand-edited one, the cuts are much more varied in length and there is one transition from the playground to the basketball game.*

put footage of cameras of five of the scenes (Basketball, Birthday, Croquet, Snowman, Park) to an editor and asked her to create a movie telling the story of the scene. The creation of a few minutes video took more than than twenty hours on average (27, 33, 20.5, 9, 14.5 hours respectively). A significant amount of time was spent on just sorting, filtering and reviewing the videos due to the quantity of footage and the low quality of some of it. The editor mentioned following classic cinematographic guidelines like the 180-degree rule, avoiding jump cuts and cutting-on-action. She also mentioned that she often avoided first-person viewpoints and chose the most stable cameras.

Global Comparison. Figure 8 shows a global comparison revealing the nature of the different methods. While random cuts and Vyclone output use fixed length shots and arbitrary cuts between different sub-scenes, ours and the professional version use varied shot lengths that reflect the pace of the action, and create a two-part video for a scene with two points of interest.

Following Content. To illustrate how our algorithm creates correct cuts that can follow the story in the scene we concentrate on a specific basketball move in Figure 9 (the full example can be seen in the supplemental video at 2 : 35). By using the 3D location of joint attention of the cameras, the algorithm can follow the movement of the ball. This is illustrated by the first shot from camera 3 that follows the player's break. However, before the long pass, the algorithm cuts to camera 7 to get a better view of the pass seen in the middle shot. When camera 7 player moves towards the basket the view of the ball is lost and the shot becomes unstable. At this point,

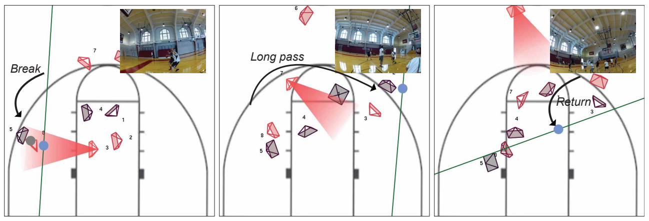


Figure 9: *Tracking motion*

the algorithm cuts to camera 6 which is on the same side as camera 7 but has a wider angle and a better view to follow the attackers return and the shot to the basket. Such high level editing decisions cannot be made simply by assessing shot quality. They cannot be made by simply tracking the joint attention. They are created by combining measures of shot quality, joint attention tracking, and cinematographic guidelines in our optimization algorithm.

Cut Decisions. We further illustrate the differences (and resemblance) between the different methods by explaining specific cut decisions. For example, while Vyclone and Random methods often violate the 180-degree rule, our results and the professionally edited ones rarely do so (see example in supplemental video at 3 : 10). Still, there are cases, such as in the Basketball sequence, where a fixed global 180-line is hard to follow due to the dynamic nature and relatively small number of cameras. In such cases we use our soft guideline, that is imposed locally on pairs of consecutive shots. As can be seen in the supplementary video examples of the basketball, these types of transitions are also found in the professionally edited cut of the game.

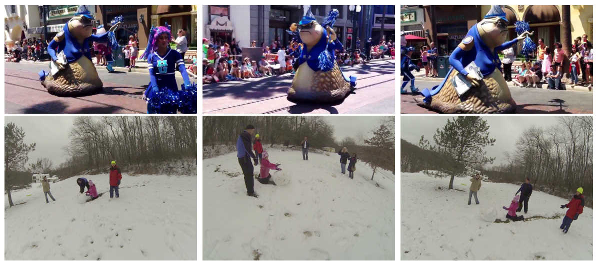


Figure 10: *Examples of 180-line rule violation. Top (Vyclon): in three consecutive shots the monster is moving from left to right, then right to left, then left to right. Bottom (Random): in three consecutive shots the snowball is rolled from right to left, then left to right, then right to lefts.*

An additional virtue of our method is the support for shot selection diversity. We have demonstrated (Figure 8) that our algorithm creates shot *duration diversity* similar to professional editors and unlike Random or Vyclone methods. Similarly, Vyclone and Random methods cannot support *size diversity* as they cannot crop any part of the frame. In contrast, because our method follows the points of 3D joint attention on each frame, it can use a variety shot sizes by cropping around the point of 3D joint attention. Figure 11 shows two frames from closer shots that were selected by the algorithm in our Croquet and Snowman results (see example in supplemental video at 3 : 25). These show better composition than the original frame, allow a closer view of the action and provide diversity and interest in the movie.

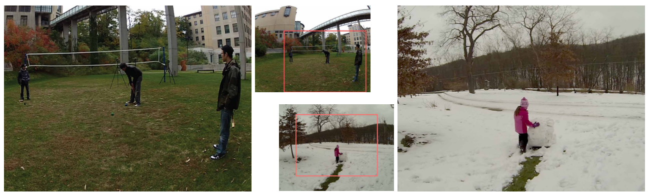


Figure 11: *Shot-size diversity is achieved by cropping around the point of 3D joint attention. Examples from the Croquet sequence and the Snowman sequence are shown with the original frame for comparison.*

Random and Vyclone methods cannot create cut-on-action style transitions between shots either. For sports activities these transitions were sought by the professional editor as they follow the ball movement in the game. We demonstrate the cut-on-action style transition that was generated by our algorithm in Figure 12 (see example in supplemental video at 3 : 17).

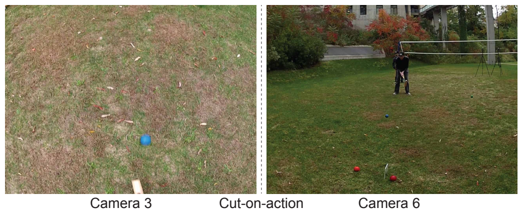


Figure 12: *Example of changing the camera according to cut-on-action: the player is hitting the ball (here shown in first-person point of view) and the camera cuts to see the ball rolling.*

Stylistic Control. To demonstrate the stylistic control of the user over the output duration we created shorter cuts of the Basketball sequence (30 and 50 seconds). These can be seen and compared in the supplemental materials. The full version contains four different attacks, two for each team. During attacks all players tend to look directly at the ball creating a single high rank point of joint attention, while during the formation of attacks players run from one side of the court to the other and there is no clear point of 3D joint attention. Using Equation 15, we are able to distinguish important from unimportant parts of the play. Removing the less important parts creates a jump in time exactly between the start of an attack and the height of the attack (Figure 13). This section is, indeed, the least interesting part of the game, and removed in the fifty second version. Reducing the length further removes two of the attacks entirely, leaving only a highlight video with one attack per team (see supplemental video at 2 : 50).

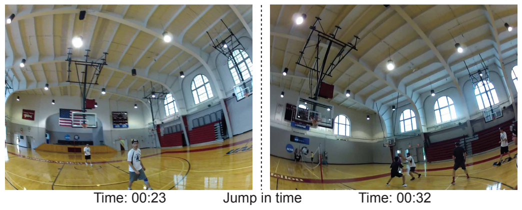


Figure 13: *Jumps in time are created when the length of video is reduced. In this case, the deleted time occurs exactly when the attack starts forming (left frame) and continues until the ball is near the basket (right frame). This deletion removes the least interesting part of the game.*

Another aspect of stylistic control is the use of first person point of view. In the croquet sequence seen in Figure 12 we create a movie allowing first-person shots creating some unique shots such as when the player looks at the ball from above and tries to aim it in the right direction before hitting the ball. The full results can be viewed in the supplemental materials, and an example in supplemental video at 3 : 24.

Results Diversity. In the supplemental materials of the paper, we provide an extensive and diverse set of ten movie cuts created from ten social camera scenes. Some scenes, like the Basketball scene, are highly challenging due to rapid motions and movements. Other scenes such as the various performance sequences are more stable and taken from one side of the action: six people filming a fire-eating act at night, and three and six shooting juggling acts from Ballan et al. [2010]. Some scenes are captured using a small number of cameras (the minimum we used was three), while others have many (eighteen cameras in a 360-degree arrangement in the B-boys dancing sequence). Two examples also include multiple sub-scenes: the Park scene and four-Scene party. The graph of this last scene and some example cuts from our results video can be seen in Figure 14, illustrating the complexity of the problem of finding high quality shots and cuts.

Various social settings are also demonstrated including a surprise party where a birthday boy is first surprised by a nerf-gun attack but then presented with a birthday cake and winter scene where a girl and her parents build a snowman. In these two examples, our automatically edited cut is similar in appearance to the professional cut as can be seen in the supplemental videos at 4 : 00.

10 Discussion

In this paper, we present an algorithm that takes, as input, multiple videos captured by participants in a social activity and produces, as output, a single video of user-specified length that cuts between the multiple feeds automatically. A graph-theoretic formulation is used to select the appropriate timing of cuts between cameras by optimizing an objective function created from cinematographic guidelines and shot quality. We demonstrate different types of cuts on a variety of events, such as social gatherings, sports events, and street performances. Comparing our results to those produced by a professional editor we note that they are similar in spirit, although understandably, not identical.

Social cameras present significant new challenges as the capture is not coordinated by a director and the videos are often poorly composed and highly redundant. The classic cinematographic guidelines were not developed for such types of camera feeds; in order to apply them we have had to modify some (the 180-degree rule), or create new interpretations of others (the cut-on-action), to make them better suited to this new type of media. Our adaptation of the cut-on-action was quite simple and a more complex implementation that took into account the direction of the action (perhaps through a measure of optical flow on the camera feeds) might provide stronger cuts. There is also the potential to apply more advanced edits [Gleicher and Liu 2007], including split-screen when two or more simultaneous points of 3D joint attention occur, and replays or slow motion for a fast-action scene with strong 3D joint attention.

A limitation of our approach is that it requires computing the 3D position and orientation of the cameras. However, this 3D understanding allows us to reason in 3D about cinematographic guidelines such as the 180-degree rule, avoiding jump cuts, and the cut-on-action as well as distinguish shot sizes. In our implementation, we rely on computer vision algorithms that take a significant amount of time. Accelerated techniques are available ([Agarwal et al. 2011]) but they can still fail if not enough cameras are present, if the scene is poorly textured, or if there are significant environmental artifacts (e.g., low-light, significant motion blur, or heavy rain/snow). Our algorithm does not actually require a full 3D scene reconstruction but requires only the position and orientation of cameras. This information could be extracted, for instance, using a sensor-based analysis (see e.g. [Cricri et al. 2012]), albeit at substantially worse precision with current commodity hardware. We also assume the camera feeds are synchronized, and we established synchronization manually using the audio channel. However, more advanced synchronization algorithms are available [Pundik and Moses 2010; Shrestha et al. 2010], and videos from cameras with GPS (e.g. cellphones) are already time-stamped.

Our edits of ten different events provide significant evidence that 3D joint attention is a powerful indication of what is important at a given moment in a scene. However, joint attention only provides one or two points of interest and those points are approximate. There may be cases where the main action will not be captured fully by focusing on the point of joint attention and there may be cases where the semantics of the scene is too complex to be so simply captured. Audio can also be a significant cue in determining the location of content in the scene. As we know the time-varying 3D location of each camera, it would be interesting to see if beamforming approaches can be used to reconstruct the motion profiles of the 3D audio sources and if that information can be fused with the 3D joint attention to provide a more powerful indication of what is important in the scene.

Lastly, we note that our algorithm can also be used to assist professional editors in their task of editing large amounts of footage by providing several possible different movies to choose from. It would be interesting to build an interface for such a semi-automatic editing tool.

References

AGARWAL, S., FURUKAWA, Y., SNAVELY, N., SIMON, I., CURLESS, B., SEITZ, S. M., AND SZELISKI, R. 2011. Building rome in a day. *Commun. ACM 54*, 10, 105–112.

BALLAN, L., BROSTOW, G. J., PUWEIN, J., AND POLLEFEYS, M. 2010. Unstructured video-based rendering: interactive exploration of casually captured videos. *ACM Trans. Graph. 29* (July), 87:1–87:11.

BAO, X., AND ROY CHOUDHURY, R. 2010. Movi: Mobile phone based video highlights via collaborative sensing. In *Proc. MobiSys '10*, 357–370.

BARBIERI, M., AGNIHOTRI, L., AND DIMITROVA, N. 2003. Video summarization: methods and landscape. In *Proceedings of SPIE*, vol. 5242, 1–13.

BERTHOUZOZ, F., LI, W., AND AGRAWALA, M. 2012. Tools for placing cuts and transitions in interview video. *ACM Trans. Graph. 31*, 4, 67:1–67:8.

CRICRI, F., CURCIO, I. D. D., MATE, S., DABOV, K., AND GABBOUJ, M. 2012. Sensor-based analysis of user generated video for multi-camera video remixing. In *Advances in Multimedia Modeling*, vol. 7131 of *Lecture Notes in Computer Science*. Springer, 255–265.

DALE, K., SHECHTMAN, E., AVIDAN, S., AND PFISTER, H. 2012. Multi-video browsing and summarization. In *Proc. CVPR Workshop on Large-Scale Video Search and Mining (LSVSM'12)*, 1–8.

DMYTRYK, E. 1984. *On Film Editing: An Introduction to the Art of Film Construction*. Focal Press.

FATHI, A., HODGINS, J., AND REHG, J. 2012. Social interactions: A first-person perspective. In *Proc. IEEE CVPR'12*, 1226–1233.

GLEICHER, M. L., AND LIU, F. 2007. Re-cinematography: Improving the camera dynamics of casual video. In *Proc. ACM International Conference on Multimedia*, 27–36.

HATA, T., HIROSE, T., AND TANAKA, K. 2000. Skimming multiple perspective video using tempo-spatial importance measures. In *Proc. of Visual Database Systems*, 219–238.

HE, L.-W., COHEN, M. F., AND SALESIN, D. H. 1996. The virtual cinematographer: A paradigm for automatic real-time camera control and directing. In *Proc. ACM SIGGRAPH '96*, ACM, 217–224.

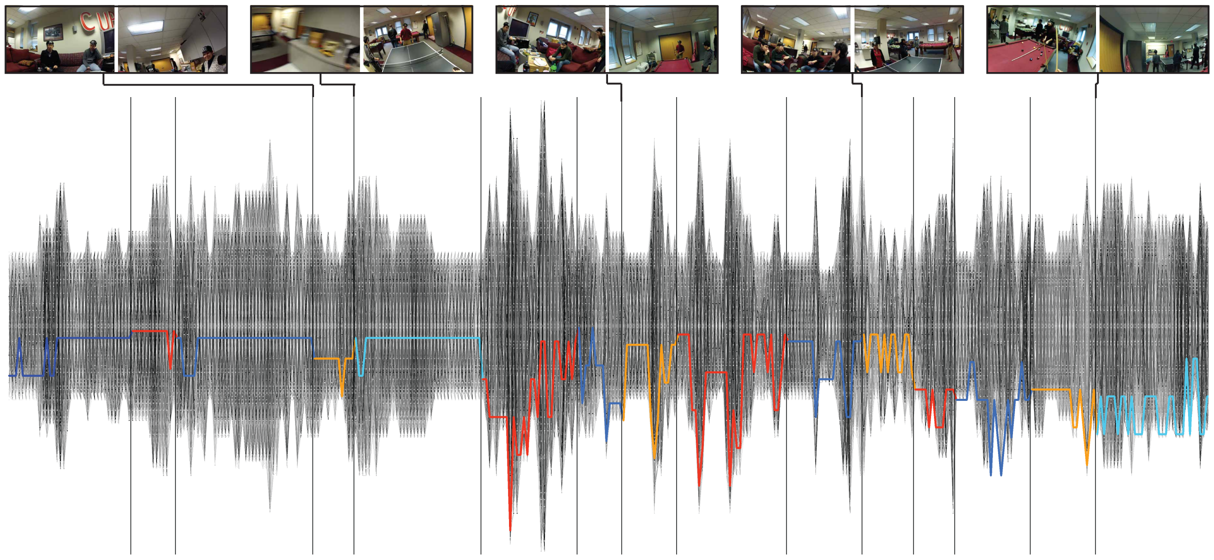


Figure 14: *The four-scene party footage graph expands and contracts because of the dynamic nature of the party scene – people are moving around and the point of 3D joint attention changes. Our algorithm can detect four sub-scenes by clustering the points of 3D joint attention spatially (each color in the path denotes a sub-scene), and creates a coherent story merging the four without jumping too much from one to another. At the top we illustrate some cuts from one sub-scene to another.*

HECK, R., WALLICK, M., AND GLEICHER, M. 2007. Virtual videography. *ACM Trans. Multimedia Comput. Commun. Appl. 3*, 1.

JAIN, P., MANWEILER, J., ACHARYA, A., AND BEATY, K. 2013. Focus: Clustering crowdsourced videos by line-of-sight. In *Proc. Embedded Networked Sensor Systems*, SenSys '13, 8:1–8:14.

KIM, K., GRUNDMANN, M., SHAMIR, A., MATTHEWS, I., HODGINS, J., AND ESSA, I. 2010. Motion field to predict play evolution in dynamic sport scenes. In *Proc. IEEE CVPR'10*, 840 – 847.

KUMAR, K., PRASAD, S., BANWRAL, S., AND SEMWA, V. 2010. Sports video summarization using priority curve algorithm. *International Journal on Computer Science and Engineering 2*.

LEE, Y. J., GHOSH, J., AND GRAUMAN, K. 2012. Discovering important people and objects for egocentric video summarization. In *Proc. IEEE CVPR'12*, 1346 – 1353.

LEPETIT, V., MORENO-NOGUER, F., AND FUA, P. 2009. Epnp: An accurate o(n) solution to the pnp problem. *Int. J. Comput. Vision 81*, 2, 155–166.

LU, Z., AND GRAUMAN, K. 2013. Story-driven summarization for egocentric video. In *Proc. IEEE CVPR'13*, 2714–2721.

MACHNICKI, E. 2002. Virtual director: Automating a webcast. In *Proc. SPIE Multimedia Computing and Networking*, 208–225.

MONEY, A., AND AGIUS, H. 2008. Video summarisation: A conceptual framework and survey of the state of the art. *Journal of Vis. Comm. and Image Rep. 19*, 2, 121–143.

PARK, H. S., JAIN, E., AND SHEIKH, Y. 2012. 3D social saliency from head-mounted cameras. *Advances in Neural Information Processing Systems* (December).

PONTO, K., KOHLMANN, J., AND GLEICHER, M. 2012. Effective replays and summarization of virtual experiences. *IEEE Transactions on Visualization and Computer Graphics 18*, 4, 607–616.

PUNDIK, D., AND MOSES, Y. 2010. Video synchronization using temporal signals from epipolar lines. In *Proc. ECCV'10*, 15–28.

RUI, Y., HE, L., GUPTA, A., AND LIU, Q. 2001. Building an intelligent camera management system. In *Proc. ACM International Conference on Multimedia*, 2–11.

SHRESTHA, P., DE WITH, P. H., WEDA, H., BARBIERI, M., AND AARTS, E. H. 2010. Automatic mashup generation from multiple-camera concert recordings. In *Proc. ACM International Conference on Multimedia*, 541–550.

SNAVELY, N., SEITZ, S. M., AND SZELISKI, R. 2006. Photo tourism: exploring photo collections in 3d. *ACM Trans. Graph. 25*, 3, 835–846.

SUMEC, S. 2006. Multi camera automatic video editing. *Computer Vision and Graphics 32*, 935–945.

TAKEMAE, Y., OTSUKA, K., AND MUKAWA, N. 2004. Impact of video editing based on participants' gaze in multiparty conversation. In *ACM CHI '04 Extended Abstracts on Human Factors in Computing Systems*, 1333–1336.

TASKIRAN, C., AND DELP, E. 2005. Video summarization. *Digital Image Sequence Processing, Compression, and Analysis*, 215–231.

TRUONG, B., AND VENKATESH, S. 2007. Video abstraction: A systematic review and classification. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP) 3*, 1, 3.

WARDRIP-FRUIN, N., AND HARRIGAN, P. 2004. *First person: New media as story, performance, and game*. MIT Press.

ZSOMBORI, V., FRANTZIS, M., GUIMARAES, R. L., URSU, M. F., CESAR, P., KEGEL, I., CRAIGIE, R., AND BULTERMAN, D. C. 2011. Automatic generation of video narratives from shared UGC. In *Proc. ACM Conference on Hypertext and Hypermedia*, 325–334.