Interactive Video Manipulation using Object Trajectories and Scene Backgrounds

Rajvi Shah and P. J. Narayanan

***Abstract*—Traditional video editing interfaces model and represent videos as a collection of frames against a timeline, which makes object-centric manipulation of videos a laborious task. We enable simple and meaningful interaction for object-centric navigation and manipulation of long shot videos, by introducing operators on three high-level video semantics: background mosaics, object motions, and camera motions. We estimate the scene background and represent the object motion using 3D space-time trajectories. We use the 3D object trajectories as basic interaction elements and define several object and camera operations as simple and intuitive curve manipulations. These allow users to perform various video object temporal manipulations by interactively manipulating the object trajectories. The camera operations model the camera as a movable and scalable aperture and allow the users to simulate pan, tilt, and zoom effects by creating new camera trajectories. With several example compositions we demonstrate that our representation and operations allow users to simply and interactively perform numerous seemingly complex, high-level video manipulation tasks.**

***Index Terms*—Object-based Video Interaction, Video Navigation, Video Manipulation, Video Mosaicing, Interactive Techniques.**

I. Introduction

With the ubiquity of video-capture devices and social media platforms, there is an exponential rise in popularity of user generated videos, creating an immediate need for simple, effective, and high-level video manipulation tools for amateur users. Enhancement or manipulation of captured images is popular among home users due to the availability of numerous easy-to-use photo editing utilities such as *Instagram, Picasa, and Windows Photo Gallery*. These tools provide advanced features such as one touch beautification, artistic effect filters, photo retouching, photo fusion, and creating wide angle panoramas. In comparison, video manipulation is less popular among common users due to the lack of easy-to-use yet powerful video editing software.

Basic video editing platforms for home-users are simple and intuitive, but these tools provide only limited functionality such as to split or merge videos, or to add captions and audio. Though there has been significant advancement in computer vision algorithms for video understanding and processing, the use of these techniques has been limited to only high-end video post-production software. Such professional video editing platforms are rich in functionality, but demand high technical expertise for use. A naïve user is easily discouraged by complex or cumbersome interactions.

Moreover, most video editing interfaces adopt frame-time semantics for video representation, even though a video is perceived by the viewer with more meaningful semantics such as objects, actions, events, and interactions. Though the frame-time model is best suited for passive playback and media synchronization tasks, the discrepancy between perception and representation makes object-centric manipulation of videos an unnatural and tedious experience. We wish to improve the usability of video manipulation interfaces by using computer vision techniques.

Previously, we proposed an interface for simple and intuitive navigation and manipulation of video objects based on object trajectories [1]. We demonstrated various applications of this interface, including object synchronization, saliency magnification, visual effects, and composite video creation. However, the background estimation pre-processing could only model fixed cameras in long shot videos. In this paper, we generalize the concepts to also handle in-plane camera motion in long shot videos.

In cinematographic terms, a long shot is defined as a continuous camera shot taken at some distance from the subjects so that they are seen in full, within their surrounding environment. Many amateur videos can be categorized as long shot videos, such as sports or art performance videos. However, most amateur long shot videos have significant camera motion, either intended or induced due to hand shake. Hence, generalizing our representation to model moving camera videos significantly improves the scope and applicability of the previous work. Moreover, in this paper, we introduce novel camera manipulation operations. These operations augment the previously proposed trajectory-based interactions, increasing the scope of interaction. We unify these contributions with the previous work and coherently describe the proposed object-centric representation and interactions.

Our proposed representation models long shot videos using three high-level video semantics: scene background, objects, and camera. Figure 1 shows an outline of our framework. We estimate the camera motion, construct a static scene background, and segment the moving objects using background subtraction. We represent the object trajectories against the scene background in a 3D space-time *Interaction Grid*, and define a set of interactive operations that allow users to perform a number of object and camera manipulation tasks in a simple and intuitive manner. Users can visualize the resulting spatial occupancy and object overlap in a separate 3D space-time *Visualization Grid*. Figure 2 shows the spatio-

The authors are with the Center for Visual Information Technology, International Institute of Information Technology, Hyderabad 500032, India (e-mail: rajvi.shah@research.iiit.ac.in; pjn@iiit.ac.in).

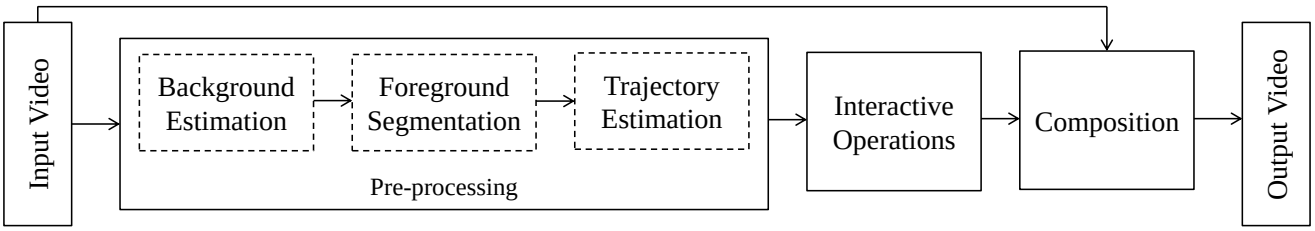


Fig. 1: Block diagram showing the main stages of our framework.

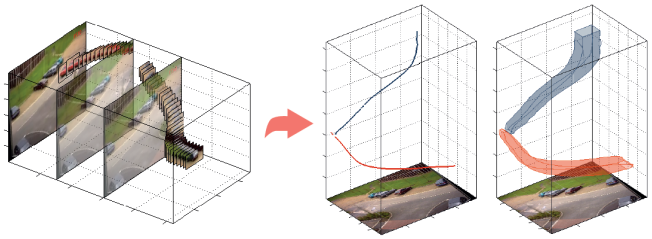


Fig. 2: *Left*: Volumetric representation of the video objects against the scene background. *Right:* Interaction and visualization grids, showing the object trajectories and object volumes in spatio-temporal 3D plots.

temporal video volume with the object tubes on the left and the Interaction and Visualization Grids on the right.

The proposed representation and operations replace complex user input elements, such as parameter specification dialogs and sliders, by interactive curve manipulation operations like select, break, join, move, resize, erase, copy, and paste. Most home-office users are already familiar with similar operations, making the interactions intuitive. Object operations allow the users to retime, reorder, remove, revert or replicate video objects independently. Camera operations allow users to create new camera trajectories to alter camera path, tilt, and zoom. In combination, object and camera operations allow users to perform high-level manipulations in a simple and interactive manner. Our key contributions are:

1) A novel interface using object trajectories as basic interaction elements, and the translation of common video manipulation tasks to visually meaningful curve manipulation operations. To the best of our knowledge, ours is the first work which explicitly uses object trajectories as user input elements for video manipulation tasks.
2) A semi-automatic framework for building the trajectory-based representation for long shot videos.
3) Example compositions created using the proposed operations on several amateur long shot videos. These validate the proposed representation and interactions.

Section II overviews prior work on object-centric approaches for video related tasks. Section III discusses our video representation and explains the required pre-processing. Section IV explains the proposed interactions and operations for object and camera manipulations. Section V gives several demonstrations of example video compositions. Section VI discusses the limitations of the current approach and future extensions. Finally, Section VII concludes this paper.

II. Previous Work

A. Video Navigation

Typical video browsing interfaces have a single control mechanism for video time manipulation: the timeline slider. Users can browse the video only in time, by moving the slider to a specific time. Previously, efforts were made to use motion information to improve interaction for direct video browsing. Satou et al. [2] identified the shortcomings of the timeline slider for video navigation, and introduced *CyberCoaster*, a polygonal line shaped spatio-temporal slider, to represent time in space for improved interactive video playback. These polygonal line sliders represent object motion trajectories and allow users to navigate the video by the objects within. *CyberCoaster* is an interesting interaction design, but it requires manual annotation to specify object motion which makes it unsuitable for practical applications.

Similar ideas were developed into prototype players for interactive video browsing [3, 4, 5, 6]. Unlike *CyberCoaster*, these systems employ automatic motion analysis algorithms. This allows users to navigate the video by dragging the object along its trajectory on the video surface, providing direct manipulation interaction. These interfaces do not alter the content of the videos, but instead display the corresponding frame based on the spatial position of the object. This creates the illusion that the dragging action is moving the object in the video.

Dragicevic et al. [3] developed a pixel-level direct manipulation interface *DimP* which employs sparse SIFT feature-flow [7] for motion estimation. They deduced per-pixel flow from the sparse feature-flow using nearest-neighbor interpolation. The motion trajectory through a pixel in a video frame could then be built by accumulating flows forward and backward in time. This interface works well on high-quality videos with large continuous motions.

Karrer et al. [4] developed a similar direct manipulation interface for in-scene video navigation – *DRAGON*. Unlike *DimP*, *DRAGON* employs the highly-accurate dense optical flow algorithm proposed by [8] for pixel-wise motion estimation. Hence, *DRAGON* could capture small motions as well. However, dense optical flow computation is much slower than feature flow computation. The speed limitation of *DRAGON* was addressed by Wittenhagen [9] in an enhanced interface called *DragonEye*. For real-time performance, *DragonEye* employs a GPU implementation of SIFT feature extraction, and uses KLT tracking [10] for point tracking. It also uses CAMShift (Continuous Adaptive Mean Shift) [11] on color

histograms as a secondary tracking algorithm. In recent work, Karrer et al. [12] analyzed the problem of temporal ambiguities in direct manipulation navigation interfaces caused by self-occluding object trajectories and proposed methods to solve it.

Kimber et al. [5] presented a video navigation interface specifically for surveillance videos called *Trailblazing*, which allows object-level interaction as opposed to pixel-level interaction. The system employs background subtraction for object detection and tracking as explained in [13] to extract object motion trajectories in a fixed camera environment. This system also demonstrated a camera-view to floor-plan mapping of object trajectories for a multi-camera surveillance environment. Users can scrub the object trajectories on the video surface or on the floor plan to navigate the footage.

The focus of all these systems is to provide a more natural browsing experience using direct manipulation interaction and not to change or manipulate the actual content of the video. Our work focusses on both the navigation and manipulation aspects of video interaction.

Goldman et al. [6] proposed an extended framework for video object navigation, annotation, and composition. They used motion trajectories to enable scrubbing for navigation and manipulation tasks, such as attaching annotations to moving objects and compositing a desired still frame from the video. Their system computes dense motion at the particle level using the tracking approach of Sand and Teller [14], which allows a detailed, per-pixel interaction. Though this system allows limited composition tasks, such as creating re-timed still frame compositions by dragging objects to the desired positions, it does not allow the creation of an arbitrary re-timed video. Our interactions are a superset of [6] as we also allow the combination of multiple temporal effects and interactive video object compositing. Also, particle video computation is an expensive operation. We avoid this and compute only object-level motions, which works well for object-level manipulation in long shot videos.

Recently, Walther-Franks et al. [15] proposed a direct manipulation interface for performance timing of keyframe animations which draws similarities to the video navigation systems discussed above and also to the video retiming operation proposed in our work.

B. Video Visualization and Summarization

Present day interfaces typically represent video using single-frame thumbnails, which only depicts a single time instance of the action in the video. Viewing or exploring a video is a time-consuming process. Many approaches have been proposed in literature for efficient video visualization and summarization. Though the literature in this area is vast, we discuss some of the object-based approaches, which use similar video representations to our framework.

Daniel and Chen [16] and Nguyen et al. [17] proposed a 3D-volume based representation for visualization and summarization of video data. These methods extract meaningful information from the videos by analyzing voxel activity and employ volume-rendering techniques for effective visualization and summarization of active video information in 3D.

Irani and Anandan [18] proposed a mosaic-based representation for video visualization which conveys dynamic information and allows for direct and rapid access to the information of interest. They demonstrated applications of this representation for stroboscopic video summarization, rapid object indexing, and object annotation.

Goldman et al. [19] proposed a video storyboarding framework for video visualization. Storyboard is an iconographic representation used in film production to describe a video shot. This framework represents a video shot by a mosaic-like static image along with iconic annotations (text and arrows) describing object and camera motions in the scene.

Many methods have been proposed for producing synopsis videos using object-activity-based saliency models [20, 21, 22, 23]. A synopsis video is more compact than the original video with minimal or no loss of presented activity. Rav-Acha et al. [20] and Pritch et al. [21] posed video synopsis as an energy-minimization problem to maximize spatio-temporal activity. This formulation first labels each pixel in the video as active or inactive using background subtraction. Background pixels are marked as inactive and moving foreground pixels are marked as active. The energy formulation models the loss in activity (foreground pixels). Finally, iterative graph-cut techniques [24] are used to minimize the energy to produce a synopsis video.

Correa and Ma [23] also proposed a technique to create stroboscopic narratives by video alignment and object segmentation. They further blended two mosaics corresponding to two video shots [25] to produce long tapestry-like composite video narratives. Kang et al. [22] proposed a formulation to combine multiple video clips into a single montage video which is compact in both space and time. They used background subtraction to model the saliency of pixels and used first-fit and graph-cut algorithms to maximize overall saliency.

Our system shares some fundamental components with these systems, such as mosaicing and background subtraction. However, the primary focus and contribution of this paper is not mosaicing or background subtraction itself, but our overall video representation and object interactions. The papers discussed above focused on producing compact video representations for visualization or for automatic synopsis applications. On the contrary, we create an object-trajectory-based representation and give creative freedom to the user. Our proposed interactive operations allow the users to perform various object and camera manipulations and combine them in any order to create composite videos, of which a synopsis video could be one such possible composite.

III. Video Representation

We represent a long shot video using a static scene background and object trajectories. This video representation requires estimating a clean background, segmenting the moving objects, and estimating the object trajectories. We explain the pre-processing required for these tasks in the following subsections.

A. Background Reconstruction

Reconstructing a clean background image in the presence of moving objects is a difficult task. It is particularly more

challenging when the camera is also moving. Many automatic algorithms have been proposed for robust modeling of scene backgrounds for fixed cameras [26]. However, these techniques typically fail to model moving camera videos.

Many long shot videos with interesting action, such as sports or dance, have significant camera motion as the camera operator tends to follow a moving target. Hence, it is not sufficient to model only fixed camera videos for these application scenarios. In the following subsections, we explain our approach for background estimation of both fixed and moving cameras.

1) Estimating background in fixed camera videos: For fixed cameras, we use the adaptive codebook-based background modeling algorithm proposed by Kim et al. [27]. This algorithm builds per-pixel codebooks for encoding intensity variations at each pixel in the video frame. A codebook is made up of codewords (boxes) which represent ranges of intensity values. These codewords grow to cover all the intensity values occuring at a given pixel. These values may correspond to background, foreground or noise. At this stage, the codebook is called a *fat codebook*.

The fat codebook is then refined in a temporal filtering step by separating the codewords contributed by the moving foreground objects from the true background codewords. A true background pixel value typically recurs within a bounded period. If a codeword does not recur within a bounded period (heuristically determined) during the training, then it is assumed that it was contributed by a moving foreground and is removed from the codebook in the temporal filtering step.

The filtered codebook corresponds to the background model. This codebook is used to classify pixels as foreground or background in all the training frames using a simple rule: if a pixel value is contained in the codebook then it is a background pixel; otherwise, it is a foreground pixel. Finally, to create a single static background image from all frames, we take the median of all background classifications at each pixel.

2) Estimating background in moving camera videos: Consider the video frames shown in Figure 3a. These frames are taken from a moving camera performance video. The per-pixel codebook-based algorithm cannot model such a video, since the pixels across the frames do not correspond directly. A frame at any point in time offers a limited field-of-view of the scene background. We use image mosaicing to combine these frames and reconstruct an extended field-of-view of the complete scene background. Given a complete representation of the background scene, it is possible to create an extended field-of-view video by mapping each of the original frames to the reconstructed background frame. This extended field-of-view video is free from any camera motion. This motion-compensated video can then be treated as a fixed-camera video for further processing and trajectory based interaction. In the remainder of this section we will discuss the underlying camera motion model, necessary assumptions, and required processing steps for mosaic-based background modeling in moving camera videos.

a) Motion Model: The projection of the 3D world onto a 2D image plane can be defined by a projective transformation. In a video, as the camera position changes with time, the parameters of underlying projective transformation also change. Under specific assumptions to be discussed later, it is possible to recover the 2D projective transformation that relates two frames captured from two different camera positions. This transformation also describes the underlying camera motion between two frames, and is commonly referred to as a *homographic transformation* or *homography*.

If $p = (x, y, w)$ and $p' = (x', y', w')$ are homogeneous coordinates of two corresponding points in two frames related by a 2D projective transformation then p and p' are related by the following equations, where H represents the homography.

$$p' = H.p \tag{1}$$

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \tag{2}$$

b) Assumptions: We make the following assumptions to describe the camera motion using homographies and to model the scene background using a static mosaic:

1) The captured scene is distant from the camera. This assumption holds true for many long shot videos.
2) The camera translation is negligible compared to the scene-to-camera distance. This assumption holds true for videos with a relatively stationary vantage point. This restriction on translation is relaxed for truly planar backgrounds.
3) The scene background is stationary and non-cluttered. This is required to be able to produce a clean static background mosaic.

Many casually-captured long shot videos such as stage performance videos or sports videos satisfy these assumptions. Hence, it is possible to represent the camera motion in these videos by a series of frame-to-frame homographies and reconstruct the scene background as a planar mosaic.

c) Background Mosaicing: Mosaicing is a well-explored problem in computer vision literature [28, 29]. Most approaches consider the general problem of constructing a seamless mosaic from a collection of images. These approaches can be extended to videos by providing every frame of the video as input to the mosaicing system; however, this is largely unnecessary due to high amount of temporal redundancy in a video. Additionally, moving objects in the video will create ghosting artifacts in the mosaic if all frames are used. We overcome these problems with an approximate frame overlap key-frame selection and an interactive object removal step. The complete procedure for key-frame selection, object removal, and mosaicing is explained here step-by-step:

i) Feature Extraction: We extract SIFT feature points and descriptors [7] from the video frames. While the background mosaic construction uses only a subset of the video frames, we extract features from all frames as we later use them to

Frame 2072 Frame 1262 Frame 782 (R) Frame 902 Frame 1412

(a) Key-frames selected from a dance performance video after spatial overlap-based ordering.

(b) Frames aligned to the reference frame (782).

(c) Final background mosaic.

Fig. 3: Key-frame selection and mosaicing.

align and warp the original video frames to the extended field-of-view mosaic.

ii) Approximate Frame Alignment: We select every n^{th} frame from the original video and compute feature-flow between adjacent frames using SIFT descriptor matching. The camera may pan across the scene background multiple times in any direction. We compute approximate translation between adjacent frames using the pairwise feature-flow and use this information for spatial ordering of the key-frames and estimating overlap after relative translation.

iii) Key-frame Selection: Once the initial set of key-frames are spatially ordered, a subset of these frames are selected based on estimated ordering and overlap. This step prunes most of the key-frames and retains a small number of frames for the final set.

iv) Accurate Frame Alignment: Accurate frame-to-frame homographies are computed for this small ordered set of keyframes using an improved RANSAC algorithm [30, 31] and Linear Least Squares. This algorithm adds an iterative refinement step which improves the homography estimation by applying RANSAC on inliers obtained in previous iterations. This refinement increases the number of inliers after each iteration and significantly reduces the probability of incorrectly estimating H. Farin [31] describes this algorithm in detail.

v) Foreground Removal: We present the final set of key-frames to the user as a filmstrip for marking foreground regions. Since the spatial ordering and overlap selection prunes most of the redundant frames from the initial set, the amount of interaction required for marking the foreground objects is reduced.

For a ballet video sequence of 2495 frames, choosing n = 10 leads to 250 frames in the initial key-frames set. Out of 250, only 5 key-frames were selected for mosacing after overlap pruning (see Figure 3). In most of our experiments, the interaction time required for marking moving foreground regions in the final key-frames was 30-40 seconds.

vi) Hole Filling: Once the foreground regions are marked, we warp the final key-frames to the center frame using the accurate homographies estimated in *Step iv*, and we remove the foreground regions in the warped frames. We compute a binary mask of the same dimensions as the final mosaic indicating if a pixel is a valid background pixel in any of the warped key-frames. If there are unfilled regions (holes) in the binary background mask due to the removal of the foreground objects, we add intermediate frames based on the spatial ordering and repeat the process of foreground removal and warping until there are no unfilled regions in the background mask. Typically, this process converges within 1-2 iterations.

vii) Final Mosaic Composition: Once the final key-frames are aligned and the foreground regions are removed, the final background mosaic is composited using Laplacian pyramid based blending [32]. We use the open-source utility Enblend [33] for compositing the final mosaic from the aligned key-frames.

Once a reliable background mosaic is composited, we estimate frame-to-mosaic homographies using the improved RANSAC algorithm (*Step iv*) and warp each input frame to the background mosaic to create an extended field-of-view video which is free from any camera motion.

B. Object Segmentation

Given a background image, we perform object segmentation using standard background subtraction techniques [34]. Per-pixel thresholding in the difference image leads to a noisy foreground mask with holes and clutter. To obtain a cleaner mask we use neighborhood distance thresholding as explained in [35]. This method is briefly summarized here:

- Let I_{bg} be the background image and I_c be the current video frame.
- Define the difference image as, $I_{dist} = Dist(I_c, I_{bg})$.
- The foreground image is obtained by thresholding the

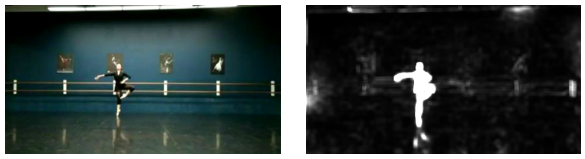
difference image I_{dist} using the following rule:

$$I_{fg}(i,j) = \begin{cases} 1 & if \sum_{(x,y)\in N(i,j)} I_{dist}(x,y) \geq T \\ 0 & otherwise \end{cases}$$

Here, $(N(i,j)$ represent the neighbourhood of the pixel (i,j). $Dist$ can be any valid distance metric in any colorspace. However, we found the Mahalanobis distance in *YUV* colorspace to give the best results [31].

We post-process this mask using morphological operations to fill holes and remove clutter. We retain only the k largest connected components as objects, where k is the number of objects in the video specified by the user.

Figure 4 shows foreground masks for an example frame at different processing stages. Since the proposed operations allows only temporal manipulations, we do not require a pixel-accurate foreground mask. However, an accurate mask can be computed using high-level matting techniques [36].



(a) Aligned frame. (b) Distance image.

(c) Initial object mask. (d) Object Mask after cleaning.

Fig. 4: Object Mask at different stages of processing.

C. Trajectory Estimation

We need to estimate the trajectories of objects from the binary segmented videos. In case of single object videos, we can simply compute the centroids of the foreground blobs. However, this can lead to erratic trajectories due to inaccurate segmentation. Also, in the presence of multiple overlapping objects, such a tracking method will not be able to resolve conflicts. Hence, we use a hybrid tracker as described by [37] for trajectory estimation. This tracker performs a connected-component tracking in binary segmented frames using Kalman filtering [38]. When the Kalman filter prediction suggests a possible overlap of objects in the next frame, a reliable mean-shift tracker [39] is used on the actual video frames.

We represent object trajectories against the static scene background as (x,y,t) line plots in the 3D interaction grid. We also show the object bounding boxes using volumetric plots in the 3D visualization grid. Here, (x,y) represent the pixel space of the scene background and t represents time. Figure 2 demonstrates this representation for a surveillance video sequence. We discuss the proposed interactions and operations based on this representation in the next section.

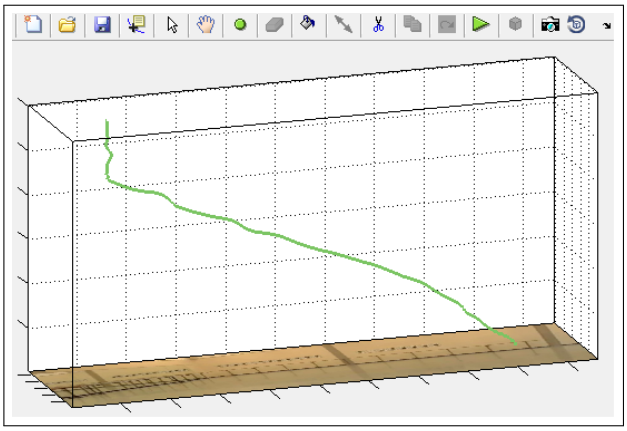


Fig. 5: A snapshot of the trajectory based interface

IV. Interactive Operations

Our prototype interaction interface is shown in Figure 5. We employ object trajectories as basic interaction elements and allow users to perform object and camera operations by manipulating object trajectories. In the following subsections, we discuss these operations in detail. The supplementary video demonstrating these operations and their effects can be seen here: **https://vimeo.com/42749995**.

A. Object Operations

We define various object operations as interactive curve manipulations on object trajectories. These interactions include *Scrub, Shift, Resize, Invert, Delete, Copy, and Break*. Users can navigate the video in different ways by scrubbing the object trajectories, or perform various temporal manipulations on video objects by interactively manipulating the object trajectories. Later on, we will demonstrate how these operations can be combined with camera operations to produce new camera motions. First, we will explain the object operations in detail.

1) Video Navigation: The user can control video navigation by scrubbing the object trajectories with the mouse. We provide two modes of navigation: Simple Video Navigation and Single Object Navigation. We also provide a *WYSIWYG* (what you see is what you get) mode in which users can create new videos in a similar way to how videos are browsed.

a) Simple Video Navigation: This mode of navigation simply replaces the timeline slider with object trajectories. Here, the user does not alter video frames; instead, video playback is controlled by the current mouse position over object trajectories. This browsing mechanism is similar to many direct manipulation interfaces discussed in Section II. However, there are two key advantages of using 3D trajectories as control elements over 2D trajectories: 1) Long range indoor motions can induce complex 2D trajectories containing self-occluding loops as shown in Figure 6 (left). A 3D representation frees trajectories from loops and self-occlusions. 2) 2D trajectories are not dependent on the action time. Hence, two objects moving along the same path at different velocities or at different time durations induce similar 2D trajectories (see Figure 6 (right)). Adding the temporal dimension resolves such conflicts.

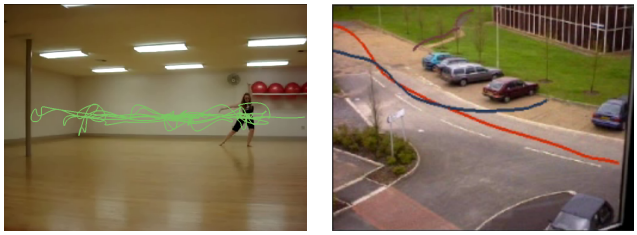


Fig. 6: Object trajectories laid out in (x, y)

(a) Simple video navigation. (b) Single object navigation.

Fig. 7: Modes of object-centric video navigation.

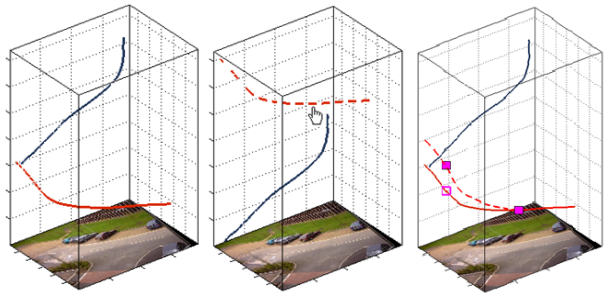
b) Single Object Navigation: In this mode, only the object corresponding to the active trajectory (the trajectory being scrubbed) is laid out on the background. Hence, scrubbing action results in motion of only the single currently active object, and other moving objects are replaced by constant background.

c) Composition by Navigation: This WYSIWYG mode allows users to composite videos as they are seen. In this mode, navigation actions are recorded and used to create a new video. This mode allows users to create various retiming effects in video by scrubbing object trajectories at the desired speed and in the desired order.

Figure 7 shows video frames for simple video navigation, single object navigation and composition by navigation. The frame shown in Figure 7(a) is the actual video frame and the frame shown in Figure 7(b) is generated with single object navigation by compositing active object segments onto the precomputed background.

2) Reordering: This operation allows a user to independently delay or advance events in the video. Users can drag and move object trajectories along the timeline to achieve desired timeshift. Shifting the trajectory effectively shifts the lifetime of the selected object. This operation can be useful to synchronize two non-overlapping events, change the relative order of two events, and so on.

3) Retiming: This operation allows users to change the pace (velocity) of different objects/events independently of the video playback rate. To achieve this, the user selects a trajectory segment and drags any endpoint of the segment to stretch or shrink the trajectory. Users can also select and extend a single point along the timeline to pause the selected object. Shrinking a trajectory along a timeline produces speed up (temporal downsampling) and stretching a trajectory results in slow down (temporal upsampling) of the selected object tube.



(a) Original State (b) Reordering (c) Reordering

Fig. 8: Reordering and Retiming on object trajectories. (a) Original state of the object trajectories. (b) User moves the trajectories to reorder the objects/ activities temporally. (c) User selects a trajectory segment and stretches it along the timeline to slow down the object.

Temporal downsampling can be achieved by skipping intermediate samples (object frames). Temporal upsampling requires interpolation between frames. Due to the motion of the objects, pixel-based frame blending introduces ghosting artifacts. Ghosting becomes severe as the upsampling rate increases. At higher upsampling rates, optical-flow based motion interpolation techniques like [40] should be used to produce better results.

4) Cloning: This operation allows users to create a clone of an object by simply selecting and copying the object trajectory. The copied trajectory needs to be time-shifted to create multiple visible instances.

5) Removal: This operation allows users to interactively remove objects from videos. Users can erase the trajectory or a segment of it by using the eraser tool or by selecting the trajectory or its segment and pressing *delete* to remove the object from the desired video segment.

6) Reversal: This operation allows the users to reverse the activity by inverting (mirroring) the trajectory along time.

Figure 8 shows reordering and retiming operations on object trajectories. Internally, these operations are performed by storing the frame-to-frame mappings between the original state and the modified state of the trajectories. Modified trajectories indicate the presence of objects in corresponding temporal segments in the modified video. Each frame in the modified video is rendered by copying the object bounding box from the corresponding source frame and compositing it on the static background image. Due to illumination differences between the source frame and the static background, visible seams may exist at object boundaries. If the difference is not severe, simple alpha blending or feathering is sufficient to suppress the sharp boundaries. However, if illumination differences are severe, gradient domain blending techniques [41] can be used instead.

B. Camera Operations

Mosaic-based representations destroy the camera-object association, and this association is an important aspect of storytelling. Moreover, for applications other than video synopsis,

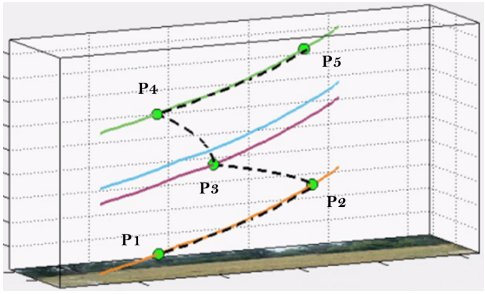


Fig. 9: Specifying Aperture Path: From P_1 to P_2, aperture follows the trajectory; From P_2 to P_3 aperture path is interpolated.

extended field-of-view backgrounds with no events of interest occupy a large display space. We present intuitive operations to produce novel moving camera videos from extended field-of-view video with desired focus of attention. These operations allow the user to perform simple cinematographic experiments in the mosaic space without having to re-shoot the video.

To achieve this, we mimic the camera with a movable and scalable view window aperture in the mosaic space. Movement of this aperture is restricted to be planar to avoid view-interpolation problems. The location, orientation, and scale of this aperture along the timeline decides the camera pan, tilt, and zoom in the video. Hence, users can simulate camera motion (pan, tilt and zoom) by manipulating the aperture parameters – $(x_t, y_t, \theta_t, S_t)$, representing the location, orientation and scale along time.

However, asking users to explicitly specify these parameters would be complex and tedious. When a user shoots a video, their objective is often to track objects and events, such as following a car until it crosses the bridge, or zooming to focus on a dancer. The user should be able to control the aperture onto the mosaic space similarly. We use the visually meaningful object trajectories to allow users to specify camera parameters in terms of objects and events of interest.

We propose simple operations to interactively create aperture trajectories to produce videos with the desired focus of attention. We explain these operations in three steps: specifying the aperture path, orientation, and scale.

1) Aperture Path: The path is a location map of the desired focus of attention at any time. Users can find object anchor points in videos by scrubbing the trajectories and marking an object or activity of interest. Consider the object trajectories as shown in Figure 9. The user has selected five anchor points, P_1 to P_5, represented by the green markers. A smooth aperture path is obtained from these anchor points using the following rule:

If P_i **and** P_{i+1} **are on the same object trajectory, then:**
Aperture follows the object trajectory.
Otherwise: Interpolate the transition from P_i to P_{i+1}.

For smooth transitions, the ratio of the distance between two anchor points and the time duration between two anchor points (*Aperture Velocity*) must be above a threshold. To prevent sudden transitions, we do not allow the user to create an anchor point which fails this criteria.

Fig. 10: (Top) Frames with Fixed Aperture Scale, (Bottom) Frames with Adaptive Aperture Scale

2) Aperture Scale: Aperture scale can either be fixed or adapted to the change in object scale. For an adapted aperture scale, the aperture grows or shrinks according to the size of the object. This creates a tracked zoom effect in the video to focus on the object of interest. However, due to the inaccuracies in object segmentation, the per-frame scale change observations are also not accurate. Hence, adapting the aperture scale to these observations produces unwanted and distracting zooming effects. To avoid this, we fit a quadratic model to the per-frame object scale factors. Also, we limit the aperture scale to a magnification factor of 2 to prevent excessive blurring. Figure 10 shows the effect of adaptive aperture scale on the *lion sequence* (please see our demo video).

3) Aperture Orientation: Finally, like scale, the aperture orientation can also be fixed or adapted to the object orientation. Alternatively, a user can interactively specify the aperture orientation at anchor points.

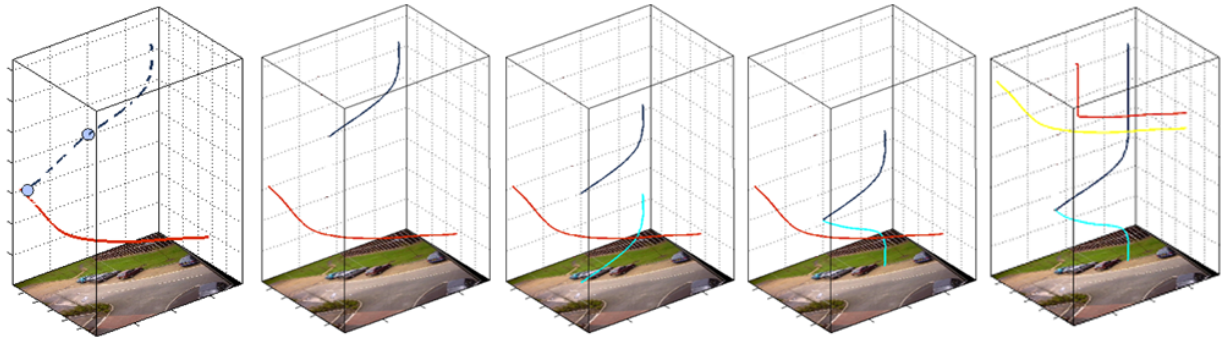
V. Results

We demonstrate the applicability of our representation by manipulating several long shot videos. Similar results can be produced by other video editing tools or techniques. However, our representation makes such manipulations significantly simple and intuitive to perform. Hence, it is not meaningful to visually compare these example results with the results of other techniques. Instead, we demonstrate several example video manipulations along with the required interactive operations in a supplementary video. This video can be found here: **https://vimeo.com/42749995**. In this section, we discuss these example results.

Figure 11 demonstrates an example of multiple object operations performed on a surveillance video (*PETS2000*[1]). Figure 11(a) shows the frames from the original video. In the original video, first a red car enters and leaves the scene, then a blue car enters from the opposite direction and moves towards the parking area. Figure 11(b) shows a series of operations performed on this video sequence. First, a segment of the blue car trajectory is deleted. This deleted segment corresponds to the blue car entering the scene and approaching the parking lot. The remaining segment shows the blue car being parked

[1]Source: PETS 2000, ftp://ftp.pets.rdg.ac.uk/pub/PETS2000

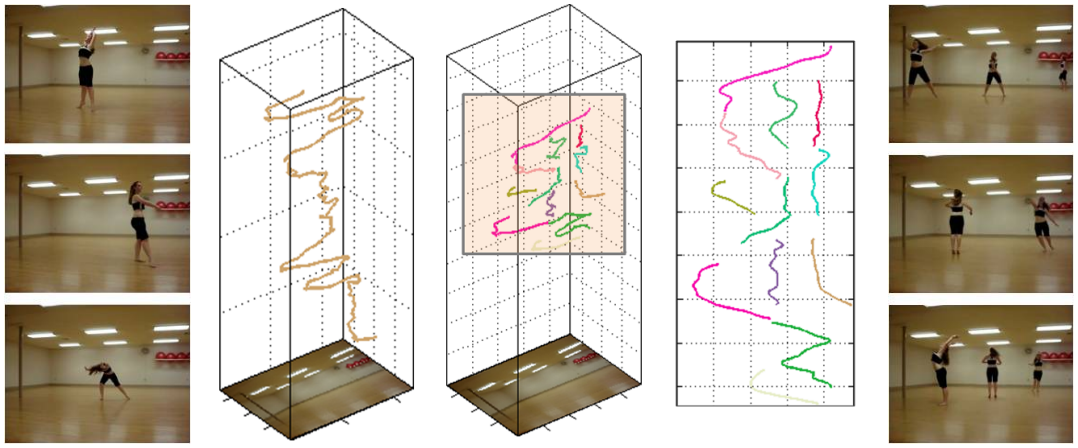
(a) Frames from the original video.



(b) Multiple object operations.

(c) Frames from the modified video.

Fig. 11: Multiple object operations on *PETS2000* video sequence. Object operations as shown in (a) from left to right: (i) A segment of blue car's trajectory is selected. (ii) Selected segment is erased (iii) The blue car's trajectory is copied and shifted in time. (iv) Blue car's shifted trajectory is inverted (v) The red car's trajectory is copied and shifted in time, tailing part is erased and the new endpoint is extended. Frames from the modified video are shown in (b).



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

Fig. 12: Composition of a dance video montage. (a) Frames from the original video, (b) Dancer's original trajectory, (c) User's arrangement of the trajectory segments, (d) Magnified XY view of the interaction grid, (e) Frames from the montage video.

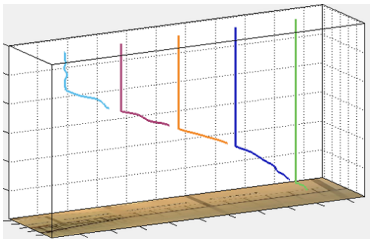
(a) Keyframes from the *Running Lion* video sequence.

(b) Example frame from the output video showing several lions.

(c) Effect of setting aperture path on cloned lions sequence. Compare the background features with original frames to observe the simulated camera motion.

Fig. 13: Object and camera operations performed on *Running Lion* video sequence.

(a) Keyframes from the *Bluebird Ballet* video sequence.



(c) Manipulated trajectory.

(d) Last frame of the modified video.

Fig. 14: Object operations performed on *Bluebird Ballet* video sequence.

from the road. This remaining segment is copied and time-shifted to the beginning of the video, and inverted. The red car trajectory is also copied and time-shifted, and a portion is erased at the trailing end and extended until the end of the video. Figure 11(c) shows the effects of these operations in the modified video. In the modified video, the blue car is already parked in the parking area. Then it is seen moving in reverse away from the parking area, pausing on the road for a moment and being parked again. Later, two red cars are seen one after another entering the scene from the right, moving towards the left. The second red car stops on the road while the first car is seen leaving the scene.

Figure 12 demonstrates composition of a dance video montage (*Only Hope Lyrical*[2]). Figure 12(a) shows the original trajectory of the dancer. The user breaks the trajectory into multiple segments and arranges them to be non-overlapping. This arrangement produces a composite video showing multiple non-overlapping dance segments in the same image space. The arrangement of the trajectory segments is shown in Figure 12(b). Figure 12(c) shows a magnified XY view of the interaction grid. Figure 12(d) shows keyframes from the montage video.

Figure 13(a) demonstrates keyframes from the *Running Lion Sequence*[3]. Figure 13(b) demonstrates the scene mosaic constructed from the keyframes. Figure 9 shows the modified state of the trajectories. In this composition, first the trajectory of the lion is copied several times, creating multiple clones of the lion. The clone lion trajectories are time shifted to produce a temporal gap, creating the effect of multiple lions running one after another. The camera aperture path is set to follow the lion clones according to the selected anchor points to simulate camera motion. Figure 13(c) demonstrates a keyframe in the output video without the camera operations. Several lions can be seen running in this cloned sequence. The effect of specifying the camera aperture path is shown in Figure 13(d). The camera moves from left to right, following the first lion, then sweeps towards the left, focusing on the second lion almost when it has reached the middle of the scene. Then, the camera sweeps further left to focus on the third and fourth lions and finally follows the last lion moving towards the right until the end of the video.

Figure 14(a) shows keyframes from the *Bluebird Ballet Sequence*[4]. Figure 14(b) shows the modified trajectory. In this composition, the trajectory of the dancer has been cut at iconic movements and the endpoint at each cut have been extended until the end of the video. The modified video shows the dancer stationary in her previous iconic movements as she crosses the floor. Figure 14(c) shows the last frame from the modified video showing all the iconic positions of the dancer.

VI. Limitations and Future Work

We allow only temporal manipulation operations on trajectories. Since these operations do not alter the spatial configuration of objects with respect to the background, composition does not introduce severe artifacts. However, if there is an overlap of objects in the original video then the object segmentation in these frames needs to be refined using robust interactive segmentation techniques [42, 36]. Also, more accurate segmentation would be required to avoid shadow artifacts in the composited frames.

The present approach for background mosaicing and segmentation is not fully automatic. We require small amount of user interaction to specify the number of foreground objects in the video and also to mark the objects in key-frames.

The current interface uses a simple object tube representation for visualization. Further work can be done to improve the visualization of various operations to enhance the overall user experience.

Since we use a single homography transformation to align a pair of images, our background estimation is limited to videos with in-plane camera motion. Also, the feature based approach for mosaic construction may fail to estimate correct homographies under extreme conditions, like lack of texture in the background or large illumination variations. More user intervention in the mosaicing process can help us overcome this limitation under difficult scenarios.

A future extension of this work is to align multiple videos captured at the same location but at slightly varying viewpoints. We can build background mosaics for each of these videos independently and then align them to a common reference. This will allow us to combine events from videos shot at the same locations at different times and by different people in a background consistent manner. Another useful extension of this work is to estimate the complexity of object motion and represent it visually to aid a user to focus on important video segments.

VII. Conclusions

We proposed a scene mosaic and object trajectory based video representation to enable simple interaction for navigation and temporal manipulation of long shot videos. We model a video as a static background and a collection of moving objects in 3D space-time. We proposed a novel interaction scheme which uses object motion trajectories as basic interaction elements and defines simple and meaningful operations for navigation and temporal manipulation of video objects. We also use the static background representation to propose simple camera operations to alter the focus of attention in videos. These operations model the camera as a movable and scalable aperture and allow users to specify the aperture pan, tilt, and zoom to simulate desired camera motion.

Using combinations of object and camera operations, users can simply and intuitively produce seemingly complex video effects. We demonstrated the usability of our proposed representation and interactions by showing interesting compositions from several consumer captured long shot videos. Though the proposed representation is not generic enough to model any dynamic video, we showed that it is able to manipulate long shot videos such as surveillance, stage performance, and sports videos.

Overall, we believe that augmenting video context and motion cues with user interface can significantly improve the

[2]Source: Youtube User d100lt, http://youtu.be/Hlf5WU5ICUQ
[3]Source: Youtube User blazinggecko, http://youtu.be/cD7dHTDudHM,
[4]Source: Youtube User klara houdethttp://youtu.be/w6IagNw9SgQ,

usability of video manipulation tools. The work discussed in this paper is a step towards that goal. We believe that the fidelity and popularity of such interfaces will significantly increase with the progress in computer vision and video processing techniques.

ACKNOWLEDGMENT

We would like to thank James Tompkin for careful proof-reading of the manuscript and the anonymous reviewers for their insightful comments that helped to improve this paper.

REFERENCES

[1] R. Shah and P. J. Narayanan, "Trajectory based video object manipulation," in *Proceedings of the IEEE International Conference on Multimedia and Expo, ICME*, 2011, pp. 1–4.

[2] T. Satou, H. Kojima, A. Akutsu, and Y. Tonomura, "Cybercoaster: Polygonal line shaped slider interface to spatio-temporal media," in *ACM Multimedia*, 1999, pp. 202–.

[3] P. Dragicevic, G. Ramos, J. Bibliowitcz, D. Nowrouzezahrai, R. Balakrishnan, and K. Singh, "Video browsing by direct manipulation," in *ACM SIGCHI*, 2008, pp. 237–246.

[4] T. Karrer, M. Weiss, E. Lee, and J. Borchers, "Dragon: A direct manipulation interface for frame-accurate in-scene video navigation," in *ACM SIGCHI*, 2008, pp. 247–250.

[5] D. Kimber, T. Dunnigan, A. Girgensohn, F. M. S. III, T. Turner, and T. Yang, "Trailblazing: Video playback control by direct object manipulation," in *IEEE International Conference on Multimedia and Expo*, 2007, pp. 1015–1018.

[6] D. B. Goldman, C. Gonterman, B. Curless, D. Salesin, and S. M. Seitz, "Video object annotation, navigation, and composition," in *ACM symposium on User interface software and technology*, 2008, pp. 3–12.

[7] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 59–73, 2004.

[8] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *ECCV (4)*, 2004, pp. 25–36.

[9] M. Wittenhagen, "Dragoneye - fast object tracking and camera motion estimation," Master's Thesis, RWTH Aachen University, Aachen, Germany, 2008.

[10] S. N. Sinha, J. michael Frahm, M. Pollefeys, and Y. Genc, "GPU-based video feature tracking and matching," In Workshop on Edge Computing Using New Commodity Architectures, Tech. Rep., 2006.

[11] G. R. Bradski, "Computer vision face tracking for use in a perceptual user interface," *Intel Technology Journal*, no. Q2, p. 15, 1998.

[12] T. Karrer, M. Wittenhagen, and J. Borchers, "Draglocks: handling temporal ambiguities in direct manipulation video navigation," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '12. New York, NY, USA: ACM, 2012, pp. 623–626.

[13] C. Stauffer and W. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 747 –757, 2000.

[14] P. Sand and S. J. Teller, "Particle video: Long-range motion estimation using point trajectories," *International Journal of Computer Vision*, vol. 80, no. 1, pp. 72–91, 2008.

[15] B. Walther-Franks, M. Herrlich, T. Karrer, M. Wittenhagen, R. Schröder-Kroll, R. Malaka, and J. Borchers, "Dragimation: direct manipulation keyframe timing for performance-based animation," in *Proceedings of the 2012 Graphics Interace Conference*, ser. GI '12. Toronto, Ont., Canada, Canada: Canadian Information Processing Society, 2012, pp. 101–108.

[16] G. Daniel and M. Chen, "Video visualization," in *IEEE Visualization*, 2003, pp. 409–416.

[17] C. Nguyen, Y. Niu, and F. Liu, "Video summagator: an interface for video summarization and navigation," in *ACM SIGCHI*, 2012, pp. 647–650.

[18] M. Irani and P. Anandan, "Video indexing based on mosaic representations," *Proceedings of the IEEE*, vol. 86, pp. 905–921, 1998.

[19] D. B. Goldman, B. Curless, S. M. Seitz, and D. Salesin, "Schematic storyboarding for video visualization and editing," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 862–871, 2006.

[20] A. Rav-Acha, Y. Pritch, and S. Peleg, "Making a long video short: Dynamic video synopsis," in *Proceedings IEEE CVPR*, 2006.

[21] Y. Pritch, A. Rav-Acha, A. Gutman, and S. Peleg, "Webcam synopsis: Peeking around the world," in *IEEE International Conference on Computer Vision*, 2007, pp. 1–7.

[22] H.-W. Kang, X.-Q. Chen, Y. Matsushita, and X. Tang, "Space-time video montage," in *IEEE CVPR*, 2006, pp. 1331–1337.

[23] C. D. Correa and K.-L. Ma, "Dynamic video narratives," *ACM Transactions on Graphics*, vol. 29, pp. 88:1–88:9, 2010.

[24] V. Kolmogorov and R. Zabin, "What energy functions can be minimized via graph cuts?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147 –159, 2004.

[25] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[26] S.-c. S. Cheung and C. Kamath, "Robust techniques for background subtraction in urban traffic video," *Electronic Imaging*, pp. 881–892, 2004.

[27] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground-background segmentation using codebook model," *Real-Time Imaging*, vol. 11, pp. 172–185, 2005.

[28] R. Szeliski, "Image alignment and stitching: a tutorial," *ACM Found. Trends. Comput. Graph. Vis.*, vol. 2, 2006.
[29] M. Brown and D. Lowe, "Automatic panoramic image stitching using invariant features," *International Journal of Computer Vision*, vol. 74, pp. 59–73, 2007.
[30] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[31] D. Farin, "Automatic video segmentation employing object/camera modeling techniques," Ph.D. dissertation, Eindhoven University of Technology, 2005.
[32] P. J. Burt and E. H. Adelson, "A multiresolution spline with application to image mosaics," *ACM Transactions on Graphics*, vol. 2, no. 4, pp. 217–236, 1983.
[33] Y. Xiong and K. Turkowski, "Registration, calibration and blending in creating high quality panoramas," in *IEEE Workshop on Applications of Computer Vision*, 1998, pp. 69–74.
[34] M. Piccardi, "Background subtraction techniques: a review," in *IEEE International Conference on Systems, Man and Cybernetics,*, vol. 4, 2004, pp. 3099–3104.
[35] T. Aach, A. Kaup, and R. Mester, "Statistical model-based change detection in moving video," *Signal Process.*, vol. 31, no. 2, pp. 165–180, 1993.
[36] J. Wang and M. F. Cohen, "Image and video matting: a survey," *ACM Found. Trends. Comput. Graph. Vis.*, vol. 3, pp. 97–175, 2007.
[37] T. P. Chen, H. Haussecker, A. Bovyrin, R. Belenov, K. Rodyushkin, A. Kuranov, and V. Eruhimov, "Computer vision workload analysis: Case study of video surveillance systems," *Intel Technology Journal*, vol. 9, pp. 109–118, 2005.
[38] G. Welch and G. Bishop, "An introduction to the kalman filter," Chapel Hill, NC, USA, Tech. Rep., 1995.
[39] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligenc*, vol. 25, no. 5, pp. 564–577, 2003.
[40] D. Mahajan, F.-C. Huang, W. Matusik, R. Ramamoorthi, and P. Belhumeur, "Moving gradients: a path-based method for plausible image interpolation," *ACM Transactions on Graphics*, vol. 28, 2009.
[41] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 313–318, 2003.
[42] Y.-Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski, "Video matting of complex scenes," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 243–248, 2002.

Rajvi Shah Rajvi Shah received her B.Tech degree in Electronics and Communication Engineering from Nirma University, Ahmedabad, India in 2009 and Master's by Research degree in Computer Science from Center for Visual Information Technology, IIIT, Hyderabad, India in 2012. After her masters, she was a research assistant at Graphics, Vision and Video Group at Max Planck Institute for Informatics, Saarbrücken, Germany for six months. Her research interests are Image and Video Processing, Computer Vision and Machine Learning.

P J Narayanan P. J. Narayanan is a Professor and Dean of Research at the IIIT, Hyderabad. He got his bachelors from IIT, Kharagpur and his PhD from the University of Maryland. He was a research faculty member at the Robotics Institute of Carnegie Mellon University from 1992 to 1996 and a scientist at the Centre for Artificial Intelligence and Robotics, Bangalore till 2000. His research interests include Computer Vision, Computer Graphics, and GPU Computing. He was made a CUDA Fellow in 2008. He was the General Chair of ICVGIP 2000 and the Program Co-Chair of ACCV 2006 and ICVGIP 2010. He was the Area Chair of ICCV 2007 and 2011. He is currently the President of the ACM India Council.