Object-centric Video Navigation and Manipulation

Thesis submitted in partial fulfillment
of the requirements for the degree of

Masters of Science (by Research)
in
Computer Science

by

Rajvi Shah
200907017
rajvi.shah@research.iiit.ac.in

Center for Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
February 2012

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Object-centric Video Navigation and Manipulation" by Rajvi Shah, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Advisor: Prof. P J Narayanan

To My Parents

Acknowledgments

I am profoundly grateful to my advisor, Prof. P. J. Narayanan for his constant motivation, support and insightful guidance over past three years. I sincerely acknowledge him for patiently listening to my half-baked ideas and for always encouraging me to pursue them.

I am also thankful to Prof. Jayanthi Sivaswamy for introducing me to the area of image processing and for her constant interest and motivation. I also thank Prof. C. V. Jawahar and Prof. Anoop Namboodiri for teaching me the foundations of the ever-exciting field of computer vision and pattern recognition.

I would like to thank Dr. Vivek Kwatra, for offering me the summer internship and for mentoring me. Working with him and my fellow interns, Poonam, Pratabh, Varun and Venkata was a great learning opportunity.

I am indebted to all my friends and colleague at CVIT and IIIT for a wonderful journey together. I would specially like to thank Anu, Dinesh, Gomathi, Mansi, Nisarg, Nitin and Shilpi for always being there for me.

Above all, I am most thankful to my parents, and sisters Dhruvi and Pruthvi, for their love and support in all my endevours.

Abstract

The past decade has seen tremendous advancements in consumer electronics and web technology. The proliferation of digital cameras and popularity of content sharing sites have caused a rapid growth in creation and distribution of images and videos by home-users. Enhancement and manipulation of captured images is fairly popular among common users due to availability of numerous easy-to-use photo editing utilities like *Instagram, Picasa, Photo Gallery,* etc. In comparison, video manipulation is still less popular among common users due to the lack of easy-to-use yet powerful video editing platforms.

Basic video editing platforms for home-users are simple and intuitive, but these tools provide limited functionality such as split and merge videos, add captions or audio etc. Professional video editing platforms are rich in functionality, but these tools demand high technical expertise for use. A novice user usually gets discouraged by complex interactions and cumbersome processing. Moreover, the traditional video editing interfaces model and represent videos as a collection of frames against a timeline. In a user's perception, a video has more meaningful semantics such as objects, actions, events, interactions, etc. The gap between perception and representation makes object-centric manipulation of videos an unnatural and laborious task.

In this thesis, we attempt to bridge the gap between the power and usability of video manipulation interfaces by using computer vision techniques. We propose a representation based on three high-level video semantics, scene mosaic, object motion, and camera motion to enable simple and meaningful interaction for object-centric navigation and manipulation of long shot videos. We build an extended field of view mosaic of the video scene and represent object motion in this scene mosaic using $3D$ space-time trajectories.

We define novel object and camera manipulation operations using object trajectories as basic interaction elements. The use of object trajectories as basic video semantics replaces complex interface elements by interactive curve manipulation operations. The object operations allow the users to perform various temporal manipulations on the video objects by interactively manipulating the object trajectories. For example, users can delay or advance the video objects by dragging the trajectories along the timeline or replicate objects by creating multiple copies of the object trajectories. The camera operations model the camera as a movable and scalable aperture and allow the users to simulate camera pan, tilt, and zoom by creating new aperture trajectories. Object and camera operations, in combination allow users to perform a number of high-level video manipulations in a simple 'click and drag' fashion.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the advancements in camera technology, digital cameras are becoming better, smaller and cheaper day-by-day. Continual advances in embedded hardware and mobile operating systems like Android and iOS have brought the world into the smart phone era. Today's smart phone cameras are equipped with advanced optics and sensor technology. Even the low-end mobile phones offer cameras with good enough resolution and frame rates for capturing day-to-day life events. Once the gadgets of the hobbyists and the professionals, the camera devices have now become integrated into a common man's daily life. Such proliferation of digital cameras has caused a tremendous increase in consumer created images and videos.

Social networking sites like *Google+, Facebook, MySpace, Orkut* and content sharing sites like *YouTube, Vimeo, Picasa, Flickr,* etc. allow the users to showcase and share their content to millions of people worldwide. Thousands of new photos and videos are uploaded to these sites daily, having viewership of millions. Sites like *Vimeo* and *flickr* are specific to only user-made personal images and videos, making it popular among amature artists and hobbyists. Viewers on these sites can not only passively view the uploaded media content but can also like, rate, comment or share it on their personal blogs or with other people in their social networking circles. Videos with interesting content and appealing aesthetics are instantly liked by the viewers. Such continual growth in creation and consumption of digital media by common users has posed the need to address a common user's requirements for high-level content manipulation.

Today, most photo editing softwares for home-office users provide advanced manipulation utilities, like one touch beautification, artistic effects filters, photo retouching, photo fusion, creating wide angle panoramas, etc. However, a similar trend is not observed for video manipulation. High-level video manipulation is still uncommon among home-users. Conventional video editing softwares for home-office consumers provide only basic utilities such as crop or trim videos, combine multiple video clips, add or remove audio, add captions, synchronize various multimedia objects, etc.

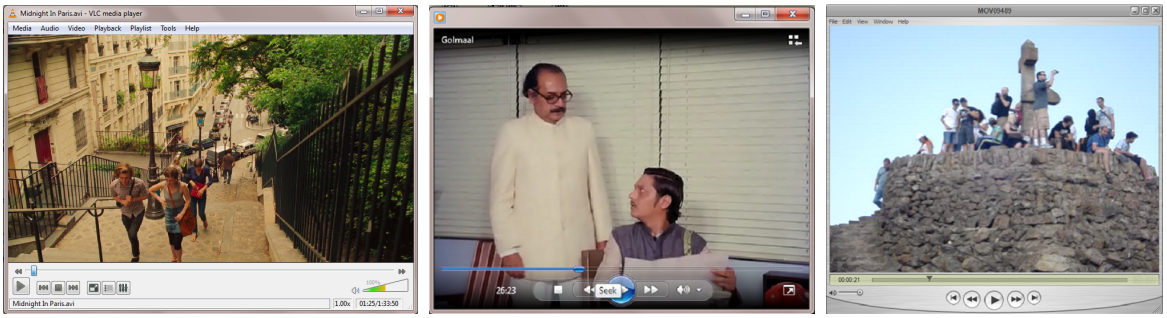
Though there has been a significant advancement in computer vision algorithms for video understanding and processing, utility of these techniques has been limited to only high-end video post-production softwares. Such professional softwares provide advanced editing functionalities but demand

high technical expertise for use. A naïve user usually gets discouraged by complex software controls and cumbersome processing.

The motivation of this thesis is to bridge the gap between the power and usability of video editing interfaces for common users. We propose an object-centric representation to enable intuitive and meaningful interaction for several video navigation and manipulation tasks.

1.1 Problem Overview

Most video browsing and manipulation interfaces adopt the frame-time semantics for video representation. This representation models a video as a collection of frames indexed by a timeline. Though the frame-time video representation video is apt for passive playback and media synchronization tasks, it is ill-suited for object-centric manipulation of videos.



(a) VLC Media Player (b) Windows Media Player (c) Quick Time Movie Player

Figure 1.1: Various video playback interfaces

Figure 1.1 shows snapshots of some popular video players. These players use a timeline slider as the basic control mechanism. The timeline slider allows a user to go to a specific position in a video by scrubbing the slider to the desired position. Consider a scenario, when a user is interested in finding a specific event in the video, e.g. when the car is parked, when two actors meet. A timeline slider based interface does not allow such object based browsing. The only way to browse a video is by navigating in time.

Figure 1.2 shows a snapshot of *Windows Movie Maker*, a video editing program for home users. This interface too adopts the frame-time representation. It represents the video shots using temporally ordered thumbnails. Users can add captions or music to a clip, split a video clip at a desired time, introduce fade and blend effects at cuts, etc. This interface is simple to grasp for even novice users, but it does not facilitate object-level video operations.

The frame-time video model is very restrictive for even simple object operations. For example, consider if the user wants to attach a text annotation to a moving object in a video. The required operation is simple in principle, transfer the object's motion to the text. But the frame-time representation requires

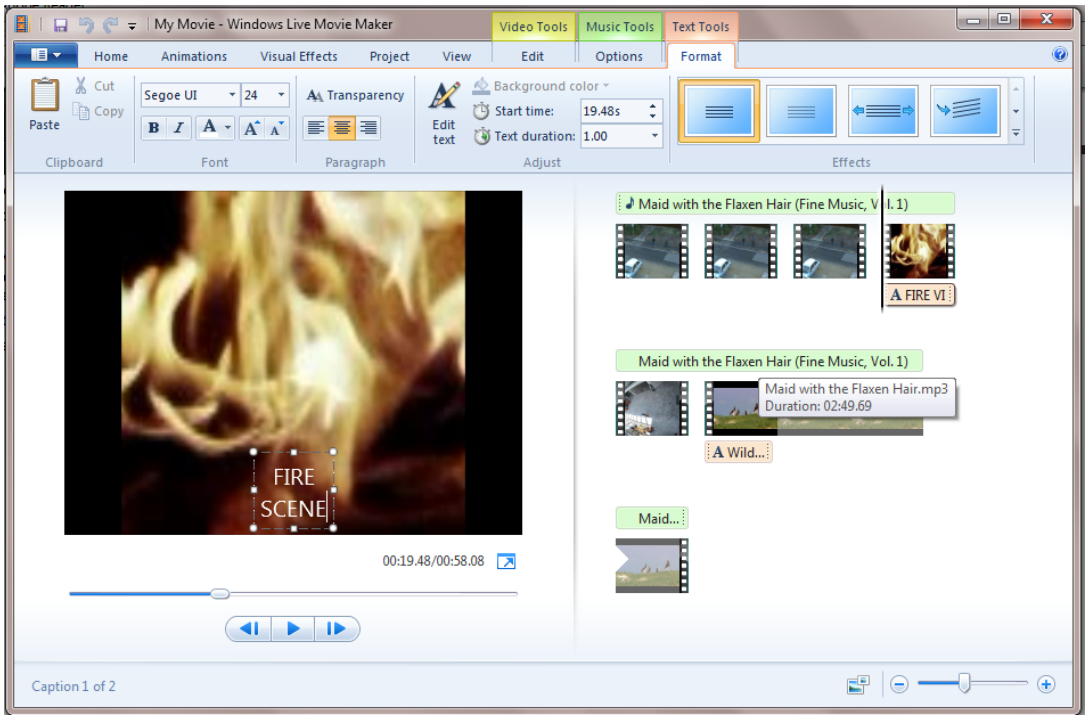


Figure 1.2: Snapshot of a typical session of *Windows Movie Maker*

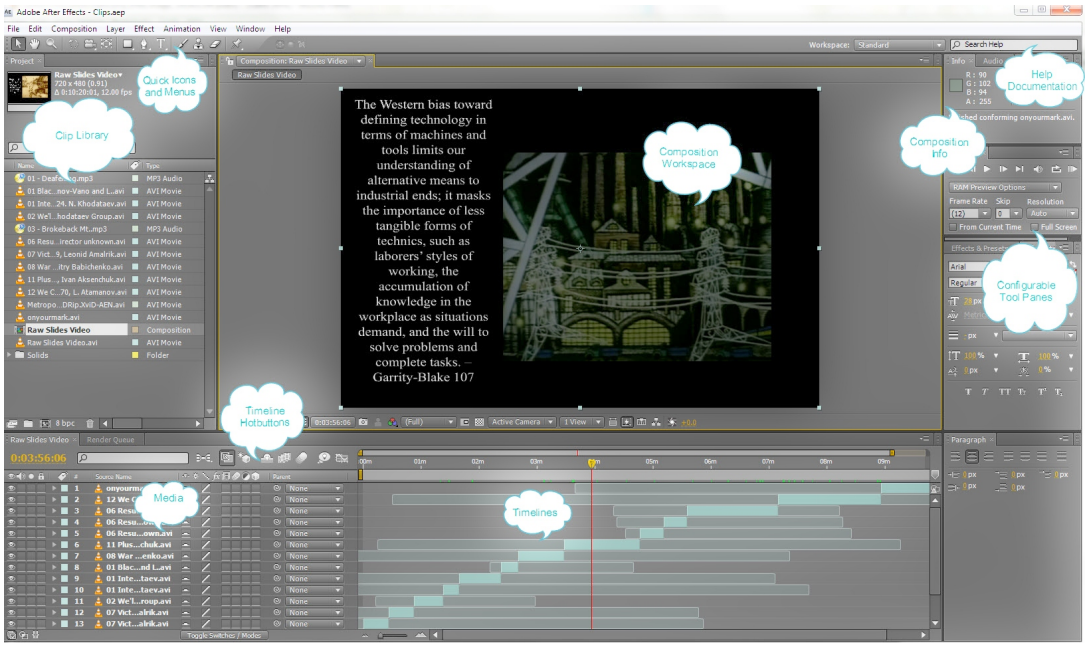


Figure 1.3: Snapshot of a typical session of *Adobe After Effects*

it to be added manually to the object's location in each frame or at certain intervals. Present day computer vision techniques for motion tracking have advanced significantly to make such tasks simpler to perform.

Professional video authoring tools use many advanced computer vision techniques to facilitate high-level video manipulation tasks. But using such softwares require sophisticated training. Figure 1.3 shows a snapshot of *Adobe After Effects*, an advanced video editing software used by film and video professionals. It can be seen that the controls are too complex and cumbersome for a common user to understand intuitively. Also the fundamental interface is still centered around the timelines.

Simplifying the video interactions require a high-level and meaningful video representation. This can be achieved by automatic understanding of higher-level video semantics such as shots, actors, objects, activities, etc. The state-of-the-art in computer vision has advanced significantly to automate mid-level video understanding. For example, robust and accurate algorithms exist for tracking points, matching patches, detecting objects, etc. These algorithms can be efficiently used for automatic extraction and understanding of mid-level video features such as shot boundaries, camera motion, object trajectories, etc. If extracted, such information can be used to provide an alternate meaningful video representation, enabling intuitive interactions for video navigation and manipulations.

1.2 Contributions

In this thesis, we propose an object-centric representation and novel interaction schemes for simple, intuitive and meaningful navigation and manipulation of long shot videos. We model a video using high-level video semantics, scene background, object motion, and camera motion. We build an extended field of view mosaic of the video scene and estimate object and camera motion using computer vision techniques. We represent the object motion in the scene mosaic space using $3D$ space-time trajectories. We use these $3D$ object trajectories as basic interaction elements and allow the users to perform a number of object and camera operations using simple and interactive curve manipulation tasks. Object operations allow users to independently retime, reorder, remove, revert or replicate video objects. Camera operations allow users to create new camera trajectories to alter camera path, tilt, and zoom. Object and camera operations, in combination allows a user to perform a number of high-level manipulations. Following are the key contributions of this thesis:

1. We propose a novel interface using object trajectories as basic interaction elements. To the best of our knowledge, ours is the first work which explicitly uses object trajectories as user input element for video manipulation tasks.

2. We identify appropriate application scenarios, formulate valid assumptions for the video capture process and propose an interactive algorithm for mosaic based video modeling.

3. We define various video manipulation tasks as visually meaningful curve manipulation operations. Such intuitive and easy to mentor interactions reduce seemingly complex video manipulation tasks to simple 'click and drag' operations.
4. We demonstrate various example compositions created using the proposed operations, validating the applicability of the proposed representation and interactions.

1.3 Thesis Organization

Chapter 2 gives an overview of the prior work on object-centric video navigation and manipulation, and also summarizes the underlying computer vision algorithms, fundamental to this line of work. Chapter 3 discusses the proposed video representation based on scene mosaic and object trajectories. It further explains the algorithms used for pre-processing in detail. Chapter 4 explains the proposed interactions and operations for various object and camera manipulations along with several demonstrations of example video compositions. Chapter 5 concludes this thesis with a discussion on limitations and future work.

Chapter 2

Background and Previous Work

This chapter gives an introduction to the prior work on object-centric video navigation and manipulation. Most work in this area, including the work in this thesis is built on top of robust computer vision algorithms for video understanding. The later part of this chapter provides an insight into the underlying computer vision techniques for motion tracking, object segmentation and image mosaicing.

2.1 Object Centric Video

Traditional interfaces for video navigation and manipulation adopt the frame-time semantics for video representation. In a viewer's perception, a video has much richer semantics, such as objects, events, actions and interactions. This makes the frame-time representation unnatural and restrictive for object or activity centric interactions with videos. Alternative video representations have been proposed earlier for a number of object centric tasks such as video object navigation, annotation, visualization, composition, synopsis, etc. This section gives an overview of the literature in this area.

2.1.1 Video Navigation

Typical video players have a single control mechanism for video playback, a timeline slider. Users can browse the video only in time, by seeking the slider to a specific time. Shortcomings of the timeline slider for video navigation were first identified in 1999 by Satou et al. [58]. They argued that the timeline sliders used for video playback do not capture dependency of space-time in the video data and introduced *CyberCoaster*, a polygonal line shaped spatio-temporal slider, represented in the visual data space for interactive video playback. These polygonal line sliders represented object motion trajectories. Users could click and scrub these trajectories to navigate the video. CyberCoaster was an interesting interaction design but it used manual annotation to specify object motion, making it unsuitable for practical application. Several years later, multiple research groups rediscovered the same idea and developed prototype players for interactive video browsing [22, 37, 39, 29]. Unlike *CyberCoaster* which used manual annotation for motion, these systems employed automatic motion analysis algorithms.

(a) DimP Interface[22] (b) DRAGON Interface[37] (c) Trailblazing Interface[39]

Figure 2.1: Snapshots of different interfaces for direct manipulation video navigation

The focus of these systems was to provide a direct manipulation interface. In HCI terminology, *Direct Manipulation* refers to an interaction style which involves continuous representation of objects of interest, and rapid, reversible, incremental actions and feedback. A direct manipulation interface for video browsing, allows a user to directly access and drag video objects along their predefined trajectories. This action does not alter the content of the video but displays the frame according to the object's spatial position, creating an illusion that the user's action is moving the object in the video. In direct-manipulation video players, the user directly manipulates the video content and indirectly manipulates the timeline slider, whereas in traditional players, the user directly manipulates the timeline slider and indirectly manipulates the video content. These interfaces, their underlying assumptions and limitations are discussed in detail here.

Dragicevic et al. [22] proposed a pixel-level direct manipulation interface- *DimP* (**Di**rect **m**anipulation **P**layer) for video browsing. This interface is shown in Figure 2.1(a). *DimP* is designed for general videos and employs sparse feature-flow for motion estimation. It uses SIFT (Scale Invariant Feature Transform [45]) as robust features. Per-pixel flow is deduced from the sparse feature-flow using nearest neighbour interpolation. In other words, it assumes that the displacement of a pixel is the displacement of the nearest feature point. The motion trajectory going through a given pixel of a given video frame is then built by adding up flows forward and backwards in time. The cumulative error due to rounding is negligible because the location of SIFT feature points has sub-pixel accuracy. Figure 2.2 demonstrates various stages of *DimP*'s motion analysis process.

If the camera is also moving, then perceived motion of the objects would be different from the actual motion. *DimP* employs background subtraction with linear camera motion assumption to enable relative dragging. The task of detecting the background motion is formulated as a clustering problem. It uses a simple, greedy screen-space binary partitioning scheme to find the most dense motion region in the space of feature motions. This algorithm yields the dominant feature displacement for a given pair of frames, which is identified as background translation and subtracted from the feature flow. Relative flow computation is visually shown in Figure 2.3.

(a) SIFT Interest Points (b) Matched Points (Red) (c) Feature Flow

(d) Optical Flow (Interpolated) (e) Point Trajectories (Cumulative) (f) Visualization of Interaction

Figure 2.2: Various stages of motion analysis and final interaction in *DimP*

(a) Feature Flow (b) Optical Flow (Before Dominant Flow Subtraction) (c) Relative Flow(After Dominant Flow Subtraction)

Figure 2.3: Relative flow computation with translational camera motion assumption in DimP

This interface works well on high-quality videos with large continuous motions. There are a few limitations of the motion analysis technique. Though SIFT feature extraction and matching is much faster as compared to optical-flow computation, non-accelerated implementations take considerable time on high-resolution videos (order of seconds per frame). To reduce the computation time, video frames are sub-sampled to 128×128 gray-scale images prior to motion analysis. As a result motions of small objects are not detected. Also, continuous trajectory for a feature point is built by merely adding up the pairwise feature-flow. Such memory-less implementation leads to discontinuous trajectories for briefly occluded objects. Moreover, translational camera motion for background subtraction is an oversimplified assumption, leading to imperfections in relative flow.

Karrer et al. [37] proposed a similar direct manipulation interface - *DRAGON* (**DRAG**able **O**bject **N**avigation) for in-scene video navigation, shown in Figure 2.1(b). Unlike *DimP*, *DRAGON* uses highly accurate dense optical-flow algorithm by [13] for pixel-wise motion estimation. In comparison to *DimP*, *DRAGON* interface has several advantages as well as limitations.

Since, *DRAGON* computes dense optical flow, it allows navigation of small objects as well. But, accurate optical-flow computation is much slower than feature-flow computation, taking around 15 seconds for per frame optical-flow computation on a $3.15GHz$, quad-core machine. *DRAGON* does not perform background subtraction to compute relative flow. Hence, perceived motion may differ from the absolute motion trajectories. Similar to *DimP*, *DRAGON* also suffers from discontinuous trajectories due to object occlusions. *DRAGON* introduces a concept of *Intertia* to its interaction, which allows a user to push an object along its trajectory. This feature allows *DRAGON* to resolve trajectory ambiguities as well as occlusions by navigating in the direction of the push until the ambiguity is resolved.

Major limitations of *DRAGON* are addressed by Wittenhagen [77] in an enhanced interface called *DragonEye*. *DragonEye* employs a combination of point tracking and color based tracking algorithms to estimate motion trajectories. For real-time performance, *DragonEye* uses a GPU implementation of SIFT feature extraction and KLT tracker [64] for point tracking. Additionally, it employs CAMShift (Continuous Adaptive Mean Shift) [10] on color histograms as a second tracking algorithm to augment the point tracker.

Kimber et al. [39] presented a video navigation interface specifically for surveillance videos called-*Trailblazing*, shown in Figure 2.1(c). Trailblazing allows object-level interaction as opposed to pixel-level interaction. It uses background subtraction for object detection and tracking (as explained in [65]) to extract object motion trajectories in a static camera environment. This system also demonstrates a camera-view to floor-plan mapping of object trajectories for a multi-camera surveillance environment. Users can scrub the object trajectories on video surface or on the floor plan to navigate in the footage.

2.1.2 Video Visualization, Annotation and Composition

In present day video browsing interfaces, video clips are represented using single-frame thumbnails, which doesn't convey what happens in the video. Viewing a video requires a time commitment. Irani and Anandan [33] proposed a scene-based representation for video visualization which conveys geometric and dynamic information in the video and allows direct and rapid access to the information of interest. They demonstrated applications of this representation for video summarization, non-linear video browsing and moving object annotation. This video representation is divided in three parts: (a) Panoramic Mosaic Image - which captures extended spatial view of the video scene, removing spatial redundancy, (b) Geometric Transformations - which captures frame-to-mosaic mapping to relate scene point to the video frames and otherwise, allowing spatial location based video browsing, and (c) Dynamic Information - the moving objects, allowing stroboscopic summarization, object annotation and object based video browsing.

Figure 2.4: Frames from an aerial video clip (left); Stroboscopic summary of the video, showing scene mosaic and objects along with their trajectories (right). (Irani and Anandan [33])

(a) Video Frames (Left), Schematic Storyboard (Top), Handdrawn Storyboard (Bottom)

(b) Video Frames (Left), Sketch-rendered Schematic Storyboard (Right)

Figure 2.5: Schematic storyboarding from videos (Goldman et al. [28])

Figure 2.4 demonstrates Irani and Anandan [33]'s video representation. Figure 2.4(a) shows four frames from an airplane take off sequence. Figure 2.4(b) shows stroboscopic summary of the actual video. A *stroboscopic summary* is a static representation of a dynamic event, in which the dynamic objects are represented by static instances sampled at a certain time interval.

Goldman et al. [28] 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 motion in the scene. The framework used manual annotation to track feature points for mosaicing and interactive foreground segmentation [56] for laying out objects on the mosaic. Arrow annotations are laid out on the composite to describe object and camera motions. Users can scrub the motion annotation arrows in the schematic as a non-linear video navigation mechanism. Goldman et al. [28] demonstrated applications of the storyboard representation for non-linear video browsing, surveillance summarization, assembly instructions, composition of graphic novels, and illus-

(a) Navigation by Scrubbing (b) Moving Annotations (c) Desired Still Composition

Figure 2.6: Examples of video object navigation, annotation and composition (Goldman et al. [29]): (a) Object-level navigation (top), Pixel-level interaction (bottom); (b) Color-coded particle groups (top), Moving annotations attached (bottom); (c) Boy in the left dragged to a desired position (top), Girl in the middle dragged to a desired position (bottom)

tration of camera technique for film studies. Figure 2.5 shows two example storyboards rendered by this framework.

Two years later, Goldman et al. [29] proposed an extended framework for video object navigation, annotation, and composition. Instead of a storyboard layout, this framework utilized direct-manipulation interaction style, allowing users to directly scrub the pixels for video navigation. This framework computes dense and long range particle flow [57] for extracting motion trajectories, allowing a detailed particle level interaction. Particles that move together are grouped using k-affine motion models. This allows a user to select a group of pixels and attach annotation that will move along with the moving particles. Users can also drag the objects appearing at different time intervals in a desired position to a common still frame to create a desired composite. Figure 2.6 shows some example applications of this framework.

2.1.3 Dynamic Video Synopsis

Rav-Acha et al. [55], Pritch et al. [54], Kang et al. [36] and Correa and Ma [20] proposed object activity based saliency models for producing video synopsis. A synopsis video is compact than the original video without the loss in activity. Rav-Acha et al. [55] and [54] pose 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 loss in activity as the objective function. Finally, iterative graph-cut techniques [41] are used to solve for this

Figure 2.7: Examples results of different video synopsis approaches. Dynamic video synopsis (left); Space-time video montage (right-top); Dynamic video narratives (right-bottom)

objective function. They also produce stroboscopic synopsis for moving camera videos using mosaic based representation. Correa and Ma [20] also proposes a technique to create stroboscopic narratives by video alignment and object segmentation. They further blend two mosaics corresponding to two video shots in a seamless manner [8] to produce composite narratives. Kang et al. [36] combine multiple video clips to a montage video that is compact in both space and time. They use background subtraction to model saliency of pixels and use first-fit and graph-cut algorithms to maximize overall saliency. Figure 2.7 shows examples frames from various synopsis videos.

2.2 Underlying Computer Vision

Computer vision is being exhaustively used to enable automatic understanding of video data for a multitude of tasks. Better the video is understood in terms of shots, actors, objects, activity, etc., better representations can be produced to enhance many of the video related tasks, such as search, summarization, visualization as well as interactions. For the purpose of object centric video interaction, a mid-level understanding of the video is required. Meaning, recognizing objects or actions is not necessary but an overall description of scene, objects and activity is needed. Many algorithms have been proposed in computer vision literature to robustly identify mid-level video features for such as scene boundaries, object motion, camera motion, etc. with reasonable accuracy. We partition the algorithms relevant to this thesis in three topics, (a) Tracking - algorithms for understanding motion in videos, (b) Segmentation - algorithms for obtaining object boundaries across the video frames and (c) Mosaicing - algorithms for aligning images to a common reference and building a complete field-of-view mosaic of the scene. Here, we discuss the relevant background and summarize the literature on these topics.

2.2.1 Motion Tracking in Videos

Motion tracking is a vast sub-field in the area of computer vision, useful for a variety of applications. Numerous tracking algorithms have been proposed differing based on the end application. For example, a surveillance scenario requires tracking only some specific targets, like a particular person or vehicle. This can be achieved by region tracking algorithms. These algorithms track target objects by modeling specific target properties like appearance, motion etc. On the contrary, applications based on video registration, requires tracking dense or sparse feature points across the video frames. We summarize the fundamental algorithms for both point tracking and region tracking here.

2.2.1.1 Point Tracking

Points in a video are tracked sequentially on a frame-by-frame basis. Tracking points between two consecutive video frames is a specialized case of finding corresponding points between two or more related images. This is a widely researched area in computer vision. We discuss the two basic classes of algorithms, for finding dense and sparse correspondences here.

Optical Flow: Optical flow indicates the motion of each pixel between a pair of images. Many algorithms have been proposed for optical flow estimation [32, 3, 5]. Horn and Schunck [32] define optical flow as a velocity field in the image which transforms one image into the next image in a sequence. This technique is based on the brightness constancy assumption. Brightness constancy assumption suggests that intensity of a pixel does not change as it is tracked from one frame to the next. Lucas and Kanade [46] combine the brightness constancy assumption with two more constraints, temporal persistence and spatial coherence to estimate optical flow. Though the algorithm by Lucas and Kanade [46] was originally proposed for dense optical flow, due to the local nature of the algorithm it has become more popular for sparse flow estimation. Most methods for optical flow estimation can be divided into Parametric Methods and Variational Methods. Parametric optical-flow algorithms are based on the assumption that the motion between the pair-of-frames can be modeled by a simple parametric motion model [34, 6, 7]. Variational methods estimates the optical-flow by modeling the motion fields using differentiable energy functions [75, 50, 14]. These energy functions impose data constancy and spatio-temporal smoothness using a combination of a data cost term and a regularization term. Robust and accurate optical flow estimation is a compute intensive process. Hence, optical flow based tracking is used only when accurate pixel level detail is required even at the cost of time, for example in movie production.

Feature Tracking: Unlike optical flow, feature based tracking algorithms detect and track only few distinctive key-points instead of each pixel. This gives a sparse representation of motion vectors between two frames. Optical flow is typically short-range where it is possible to repeatably detect and match distinctive feature points across many frames until occluded or until there is a drastic change

in its appearance. A vast literature exists on key-point selection. Typically, most algorithms consider corner points with a distinctive textured neighbourhood as good features to track [31, 63, 45]. Most tracking approaches track the detected key-points by iterative gradient-descent minimization on an error surface defined over a local neighbourhood similarly to the LK technique. Recently, descriptor based feature matching approaches like *SIFT, SURF, DAISY*, etc. have gained popularity [45, 4, 69, 48]. These methods represent the point neighbourhood using a descriptor, invariant to scale, viewpoint and illumination changes. Corresponding key-points are found by matching descriptors across the image pair. In the past decade, SIFT has been extensively used by many vision researchers for various problems and has become a milestone in feature detection and description literature. We also use SIFT keypoints for feature based background alignment in this thesis.

2.2.1.2 Object Tracking

Object tracking is a widely researched area with applications in multiple domains. Yilmaz et al. [80] gives a detailed survey of various classes of object tracking algorithms. The problem of tracking objects in a video can be posed in multiple different ways. Many approaches build object tracking simply as point tracking problem by defining the object as *"A group of points that move together"* [61, 62, 71]. These approaches imposes motion heuristics to handle the challenges associated with object tracking such as entry, exit, occlusions, deformations, etc. Another group of algorithms use statistical constraints to model such challenges. These methods model long range point tracking as a statistical state estimation problem. Kalman Filtering [76, 11] and Particle Filtering [40] are two popular estimation techniques used for object tracking. These methods statistically model the object properties such as position, velocity, etc. The Kalman Filter operates in three stages, measurement, prediction and correction. Measurement indicates the state (location) of the object in current frame, prediction estimates the state of the object in next frame using the statistical state model. Correction uses the next frame measurement to update the state model. The Kalman filter assumes that the state variable follows a gaussian distribution. Particle filtering overcomes this limitation by using weighted sampling of randomly selected points for state representation. Another class of algorithms perform region level appearance matching for object tracking. These algorithms differ in their appearance representation and search techniques [60, 26, 18, 10, 35]. In this thesis, we use a hybrid tracker for object tracking which combines Kalman Filter with an appearance based Mean-shift tracker. This tracker is discussed in detail in Chapter 3.

2.2.2 Video Object Segmentation

Segmenting objects in a video is a challenging task. Several classes of algorithms have been proposed for solving this problem. One class models the video as static background and moving foregrounds. This class of algorithms solve the segmentation problem by first detecting the moving objects, learning a static background and extracting foreground objects from each frame by background subtraction [1, 53].

Numerous algorithms, both parametric as well as non-parametric, have been proposed in literature for learning background models [70, 65, 38]. The application of background subtraction algorithms is limited to only static videos with non-cluttered background. If the camera is moving and the background consists of dynamic objects, such methods fail to produce accurate segmentations. Sawhney and Ayer [59] and Wang et al. [72] proposed motion fields based segmentation algorithms which segments object regions by clustering the motion field with EM algorithm [49]. Segmentation results produced by such algorithms are seldom pixel accurate.

For advanced video composition applications like rotoscoping, accurate object masks are required. Producing such accurate object masks is known as alpha matting in composition literature. Many interactive algorithms have been proposed previously for such accurate image and video matting [73]. Majority of these algorithms pose segmentation as a labeling problem and use graph-cut based energy minimization techniques [41] for producing segmentation results. These energy functions are a combination of a data cost term and a smoothness cost term. Data cost defines the cost of assigning a certain label to the given pixel, whereas the smoothness cost impose smoothness constraints between neighbouring pixels/voxels. Interactive segmentation algorithms allow the users to label some seed pixels as belonging to foreground or background. These labels are then used by the optimization to impose known data costs on labeled pixels and propagate these labels to the neighbouring pixels/voxels using the smoothness terms [44, 17, 74].

Recently, many algorithms have been proposed for automatic and long range object segmentation for dynamic videos scenes [30, 43]. Grundmann et al. [30] extends the graph based image segmentation approach of Felzenszwalb and Huttenlocher [25] to hierarchical video object segmentation. This approach first over segments the video frame into homogeneous regions using mean-shift filtering [19] and hierarchically merge coherently moving regions for object segmentation. Lezama et al. [43] extends this approach by incorporating long range, clustered point trajectories for merging regions.

Interactive matting algorithms as well as region based algorithms listed above are computationally expensive. In this thesis, we work with only long shot videos scenes. Such videos can be modeled in a relatively simple manner as compared to dynamic scenes. Hence we use much simpler techniques like background subtraction for object segmentation. The approach used for segmentation is explained in detail in Chapter 3.

2.2.3 Image Alignment and Stitching

As seen in the previous section, for many object centric video tasks such as synopsis, the video scene is represented using a mosaic of the scene background. Such a mosaic is created by taking a subset of frames from the video, aligning them to a common reference frame and blending them in a seamless manner such that the final mosaic presents the entire field-of-view captured by the video. We also model the video background by a scene mosaic in this thesis. The process of aligning a set of images and compositing them is commonly known as Mosaicing. The mosaic composition from aligned set

of images is also referred to as Image Stitching to represent a wider variety of blending and merging algorithms. We use both these terms interchangeably throughout this thesis.

Mosaicing is one of the oldest and most thoroughly researched problem in computer vision. Numerous algorithms have been proposed to solve this problem in various ways. An extensive survey of alignment and stitching techniques is published in [67]. Here, we briefly outline the basic motion models defining the warping transformation between related frames and algorithms for alignment and stitching relevant to this thesis.

2.2.3.1 Motion Models

The key to aligning a pair of images is finding a warping transformation that maps pixels from one image to the other. The projection of a $3D$ 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 changes. Under planar assumptions (scene is planar), 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.

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 as follows,

$$p' = H.p \tag{2.1}$$

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \tag{2.2}$$

Translation	Rotation	Similarity	Affine	Perspective
$\begin{bmatrix} 0 & 0 & t_x \\ 0 & 0 & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} cos\theta & -sin\theta & t_x \\ sin\theta & cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} a & -b & t_x \\ b & a & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}$

Table 2.1: Planar Transformations

Here, the transformation describes the the most general camera motion model corresponding to perspective projection with 8 degrees of freedom. Table 2.1 lists various motion models corresponding to basic set of projective transformations in homogeneous coordinates. These transformations are visually depicted in Figure 2.8. We discuss these transformations in detail here.

Translation: The pure translational motion between two frames can be described by a 2 parameter transform. A single point correspondence is sufficient to estimate the translation parameters.

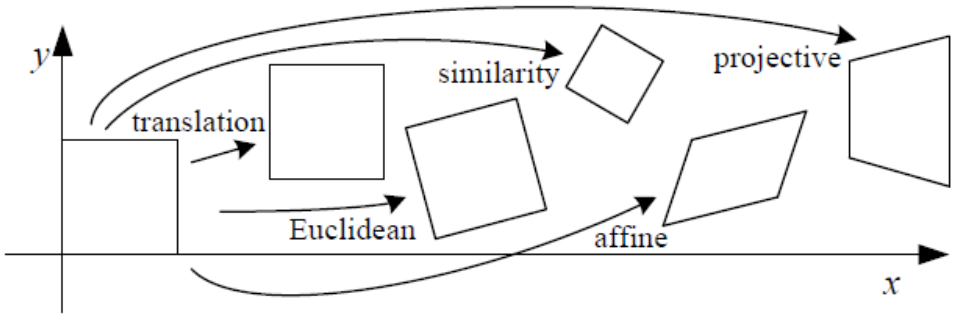


Figure 2.8: Basic set of 2D planar transformations. (Image Courtesy: Szeliski [67])

Rotation: This motion model can describe translation and rotation between two frames. It has 3 unknown parameters, translation along x and y and a rotation angle.

Similarity: This motion model describes uniform scaling along both axes in addition to rotation and translation. It has 4 unknown parameters, two translations, a rotation angle and a scale factor.

Affine: This motion model allows non-isotropic scaling and skewing, in addition to rotation and translation. It has 6 unknown parameters, translation in two directions, a rotation angle, two scale factors and a skew factor. Up to similarity transform, all motion models are shape preserving. Affine transformation does not preserve relative lengths or angles, but it preserves parallelism, i.e. parallel lines remain parallel even after the transformation.

Perspective: This is the most general form of projective transform which describes $3D$ motion of a plane as viewed from different positions. It is also popularly known as *Homography*. Unlike affine transform, perspective transform does not preserve parallelism. Parallel lines meet at a vanishing point (point at ∞) after a perspective projection. Perspective projection has 8 unknown parameters.

Homography or Perspective Transformation can model arbitrary camera motion as long as the scene being captured is planar. A truly planar scene is a too strict restriction for most real-world environments. However, in most practical scenarios when the camera is at a large distance from the scene being captured, a perspective projection can reasonably describe the image-to-image transformation. For the purpose of this thesis, we use perspective transform (homography) to estimate frame-to-frame motion and to align frames. We now discuss the commonly used techniques for image alignment.

2.2.3.2 Image Alignment

To align a given pair of images, we need to estimate the parameters of the motion model that relates the two images. One approach is to search through the entire search space of the parameters by warp-

ing the frame according to chosen parameters and measuring pixel-by-pixel dissimilarity. This class of methods are commonly referred to as *Direct Alignment* methods. Another approach is to identify corresponding points or regions in two images and solve for the warping parameters using Equation 2.2. This class of methods are known as *Feature-based Alignment* methods.

Direct Alignment: Direct Methods for image alignment are based upon the *brightness constancy constraint*, which assumes that brightness of a pixel does not change due to camera motion. If pixels p and p' in images I and I' are related by homography H then according to the *brightness constancy constraint*,

$$|I'(p') - I(p)| = 0 \tag{2.3}$$

$$|I'(H.p) - I(p)| = 0 \tag{2.4}$$

Direct alignment methods try to estimate the motion parameters (Homography H) by minimizing the pixel-by-pixel dissimilarity between the warped image $I'' = I'(H.p)$ and I.

$$H = arg\min_{H} \sum_{p} \rho(I'(H.p) - I(p))$$

Most methods differ in the *Error Metric* (ρ), used for measuring the dissimilarity or in the *Search Technique* used for finding parameters of H such that it minimizes the error. More detailed discussion on these methods can be found in [67].

Feature-based Alignment: We have already discussed point correspondence problem in subsubsection 2.2.1.1. Given corresponding points between two images, the parametric relationship between this pair of points can be described by Equation 2.2. Writing it using the inhomogeneous formulation gives,

$$x' = \frac{H_{11}.x + H_{12}.y + H_{13}}{H_{31}.x + H_{32}.y + H_{33}} \tag{2.5}$$

$$y' = \frac{H_{21}.x + H_{22}.y + H_{23}}{H_{31}.x + H_{32}.y + H_{33}} \tag{2.6}$$

It is clear that each point correspondence leads to 2 equations. Assuming the motion model to be perspective with 8 degrees of freedom, minimum 4 point correspondences are required to solve for H. Points which are consistent with the homography parameters are called *inliers*. Typically, the search is not limited to finding 4 correspondences as point matching is an error prone process and estimated matches may also be *outliers*.

One solution is to find as many correspondences as possible and use a *Least Squares* solution. However, least squares solution will only reduce the effect of outliers instead of ensuring an accurate alignment.

Most popular solution to this problem is known as RANdom SAmple Consensus, or RANSAC [27]. RANSAC computes an initial estimate of parameters using a randomly selected subset of k correspondences and computes inliers which are consistent with estimation. This experiment is repeated N times and the estimation corresponding to the maximum number of inliers is used for alignment. In this thesis, we use an improved version of RANSAC proposed by Farin [24] for homography estimation, described in Chapter 3.

2.2.3.3 Mosaic Composition

Once the frame-wise homographies are estimated, a final composite has to be created by warping all input images to a common reference. The composite image, also known as *Mosaic* offers an extended field of view of the scene captured by individual input images. Compositing a mosaic is a multi-stage process involving warping surface selection, pixel source selection, blending and exposure compensation. We briefly discuss each of these stages here.

Selecting Warping Surface: Warping surface is the surface onto which the aligned images are mapped. If the field of view covered by the input images is not too wide, one of the input images, typically the central image can be chosen as a *Reference Image* and all input images can be warped w.r.t the reference image. This mapping is equivalent to choosing a planar warping surface. The mapping remains a perspective projection preserving collinearity. However, as the viewing angle broadens, a planar warping surface results in severe image stretching near both the ends. It is a popular choice to use cylindrical [66] or spherical [68] warping surface for compositing wider mosaics. These warping surfaces do not retain the local appearance of the scene structure but results in lesser distortion globally.

Source Pixel Selection: Once the input images are projected onto the warping surface, pixels from multiple images can map to the same pixel location in the final composite. The simplest idea is to fill the pixel location by selecting any one pixel value or by taking an average of all pixel values. However, such a simple measure can lead to severe artifacts in the final mosaic due to the errors in alignment, exposure, presence of objects in some images etc. Numerous techniques have been proposed for choosing the pixel values, based on median filtering [33], center-weighting (*feathering*) [67], minimum-likelihood [2], seam-selection [51, 78, 21, 23, 42, 2] etc. which reduce the effect of such artifacts. Szeliski [67] presents a more detailed discussion on these techniques.

Blending and Exposure Compensation: Even after the source pixels are carefully selected to remove unwanted objects and avoid visible seams, the final composite still can have visible artifects along the seam boundaries due to alignment errors or exposure differences. Simple blending techniques such as *feathering* mask out the sharp differences along the boundary by distributing the pixel values in a specific transition width. Exposure difference still remains visible but becomes less noticeable due to a gradual transition. Burt and Adelson [15] proposed a multi-band blending solution to this problem.

Instead of using a single transition width, a frequency-adaptive width is used by creating a laplacian pyramid and making the transition widths a function of the pyramid level. Gradient-domain blending is also a popular solution which can effectively deal with significantly varying exposures. Pérez et al. [52] proposed a gradient domain fusion technique for seamlessly compositing images with significantly varying exposures. In this technique, instead of copying the actual pixel values, gradients of the source image are copied to the canvas image. The actual pixel values are reconstructed using a guided interpolation, mathematically formulated by Poisson partial differential equation with Dirichlet boundary conditions. A guided interpolation locally matches the guidance field (gradients) while strictly obeying the boundary conditions (exact matching at the seam boundary). Many variants of these blending approaches have been proposed in composition literature. We use an open-source utility [79] based on Laplacian blending for mosaic composition in this thesis.

2.3 Summary

In this chapter, we presented an overview of the previous work in the area of object centric video and the underlying computer vision techniques. We discussed some of the significant and relevant papers on object centric navigation, visualization and summarization of videos. The work in this thesis is strongly inspired by these efforts. We also discussed in detail the literature on computer vision algorithms for motion tracking, object segmentation and image alignment and stitching. These algorithms are fundamental components for automatic video understanding and composition. The scope of this discussion is not limited to this thesis. As the fundamental computer vision and video processing techniques improves, the work in this thesis can be extended and enriched in many possible ways.

Chapter 3

Scene Mosaic and Object Trajectories based Representation

As discussed in previous chapters, most video navigation and editing platforms model and represent videos as a collection of frames against a timeline which makes object centric manipulation and browsing an unnatural and laborious experience. We propose an alternative video representation that goes beyond the traditional frame-time video representation and enables natural and intuitive video interaction. This chapter explains in detail our video representation and required pre-processing.

3.1 Video Representation

We use a scene background and object trajectories based representation to enable object-centric temporal navigation and manipulation of 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 consumer captured videos such as sports or art performance videos taken by the audience fall under this category. Figure 3.1 shows some examples of long shot video frames. Such videos can be modeled well by the object-background model under reasonable assumptions.

The proposed representation models a video using three high-level video semantics, scene background, object motion, and camera motion. We build an extended field of view mosaic of the scene background, compensating the camera motion if present and extract spatiotemporal object volumes using this static background image. We represent the extracted object trajectories against the static

Figure 3.1: Examples of long shot video frames

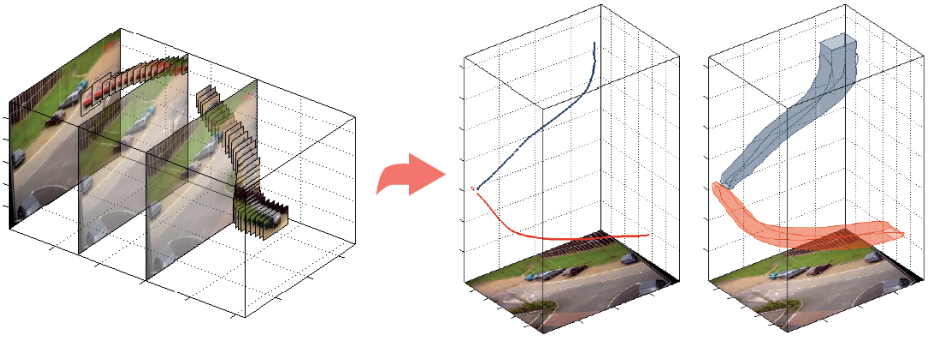


Figure 3.2: Object-tube video model (left); Interaction and Visualization Grids (right)

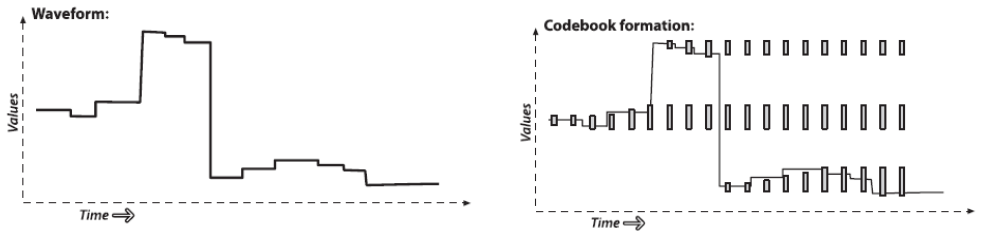
background in a $3D$ space-time - *Interaction Grid*, and define a set of interactive operations that allows users to perform a number of object and camera manipulation tasks in a simple and intuitive manner. Users can visualize the resulting spatial occupancy and object overlap in a separate $3D$ space-time - *Visualization Grid*. Figure 3.2 shows the object-tube model of a video on the left and interaction and visualization grids on the right.

This representation replaces complex input elements like parameter specification dialogs and sliders by interactive curve manipulation operations like select, break, join, move, resize, erase, copy, paste, etc. Most home-office users are already familiar with such operations. Visual nature of such operations makes the interactions intuitive and easy to mentor, reducing seemingly complex video manipulation tasks to simple 'click and drag' operations.

To create an object-background based video model, it is required to estimate scene background as well as object volumes. Reconstructing a clean background image in presence of moving objects is a difficult task. It is all the more challenging when the camera is also moving. In the following sections, we explain in detail the pre-processing required for scene background reconstruction and object segmentation for both static camera and moving camera scenarios.

3.2 Modeling Fixed Camera Videos

In fixed camera environments, background subtraction is the most popular technique for moving object segmentation. In order to perform background subtraction, first a model of the background has to be learned. Once learned, this background model is compared against the current image and the known background parts are subtracted away. The image parts remaining after the subtraction are presumably the moving foreground objects. Segmented objects need to be tracked across frames to resolve conflicts among multiple objects. We discuss segmentation, background reconstruction and tracking in detail in the following subsections.



(a) Time-varying waveform (b) Adaptive codebook

Figure 3.3: Adaptive codebook formation (Image Courtesy: Learning Opencv[9])

3.2.1 Object Segmentation

We use a modified version of the adaptive codebook based background subtraction algorithm proposed by Kim et al. [38] for segmenting moving objects. This is a non-parametric algorithm, it samples the pixel values over long times for background modeling. It computes an adaptive and compact background model that can capture structural background motion over a long period of time under limited memory. It can robustly learn background in presence of moving foreground objects, illumination variations and background noise.

This algorithm requires training for modeling background prior to actual segmentation. In the training phase, a codebook is built on per-pixel basis for encoding variations at each pixel. A codebook is made up of codewords (boxes) that grow to cover the common values seen over time. Consider the time-varying waveform as shown in Figure 3.3(a). It can represent intensity variations at a given pixel over time. Figure 3.3(b) shows how the codewords (shown by boxes) grow and how new codewords are added to the codebook when the change is too abrupt to cover by an existing codeword.

In the codebook method of learning a background model, each codeword is defined by two thresholds ($high$ and low) over each of the 3 color axes. Codeword thresholds expand ($high$ getting larger, low getting smaller) if a new background sample falls within a learning threshold ($cbBound$) above $high$ or below low. If new background samples fall outside the range of codeword and its learning thresholds, then a new codeword is added to the codebook to accommodate this sample.

It is apparent that, the codewords in this codebook will encode all the values occurring at any given pixel. It may be contributed by background, foreground or noise. This codebook is called a *fat codebook*. This codebook is then refined in a temporal filtering step by separating the codewords contributed by the moving foreground objects from the true background codewords. The true background, which includes both static pixels and moving background pixels, usually is quasi-periodic (values recur in a bounded period). This motivates the temporal criterion of *Maximum Negative Runlength* λ, which is defined as the maximum interval of time that the codeword has not recurred during the training period. Codewords with unusually large value of λ are removed from the codebook in the temporal filtering step.

Object detection is carried out by testing the color differences of the current image from the background model. If an incoming pixel value is covered by one of the codewords of the codebook then it is labeled as a background pixel. Otherwise, it is labeled as foreground. The binary labeled frames

are further cleaned by morphological operations and connected-components analysis to remove noisy labels. The complete outline for codebook generation and segmentation is given in Algorithm 1.

Let $\mathcal{X}$ be the training sequence for a single pixel consisting of T YUV vectors, $\mathcal{X} = \{x_1, x_2, ..., x_T\}$. Let $\mathcal{C}$ represent the codebook for this pixel consisting of L codewords, $\mathcal{C} = \{c_1, c_2, ..., c_L\}$. Each pixel has variable number of codewords in its codebook based on the YUV variations. Each codeword c_i, consists of a 6-tuple $aux_i = \{I_i^l, I_i^h, f_i, \lambda_i, p_i, q_i\}$, where I_i^l and I_i^h defines lower and upper bounds for YUV values for the codeword c_i; f_i denotes the frequency of codeword c_i; λ_i denotes the maximum negative run-length, the longest interval during the training for which c_i has not occurred; p_i and q_i denote respectively the first and last access times to the codeword c_i.

Algorithm 1 Adaptive codebook based algorithm for background subtraction

Construction of Fat Codebook

I $L \leftarrow 0, \mathcal{C} \leftarrow \emptyset$.

II for $i = 1$ to T

- i Find the codeword c_m in $\mathcal{C} = \{c_i | 1 \leq i \leq L\}$ such that,
  - a $I_m^l - matchBound \leq x_t \leq I_m^h + matchBound$
- ii If $\mathcal{C} = \emptyset$ or no matching codeword found then, $L \leftarrow L + 1$. Create new codeword c_L by setting,
  - a $I_L^l \leftarrow (Y, U, V) - cwBounds, I_L^h \leftarrow (Y, U, V) + cwBounds$
  - b $f_L \leftarrow 1, \lambda_L \leftarrow t - 1, p_L \leftarrow t, q_L \leftarrow t$
- iii Otherwise, update the matched codeword c_m by setting,
  - a $I_m^l \leftarrow min\{I_m^l, (Y, U, V)\}, I_m^h \leftarrow max\{I_m^h, (Y, U, V)\}$
  - b $f_m \leftarrow f_m + 1, \lambda_m \leftarrow max\lambda_m, t - q, p \leftarrow p_m, q \leftarrow t$

III For each codeword $c_i, i = 1, ..., L$, wrap around λ_i by setting $\lambda_i \leftarrow max\{\lambda_i, T - q_i + p_i + 1\}$.

Temporal Filtering of Fat Codebook

I Let $\mathcal{T}_{\mathcal{F}}$ be the threshold on maximum negative runlength λ, then filtered codebook $\mathcal{F}$ is defined as,

- i $\mathcal{F} = \{c_i | c_i \in \mathcal{C} \cap \lambda_i \leq \mathcal{T}_{\mathcal{F}}\}\}$

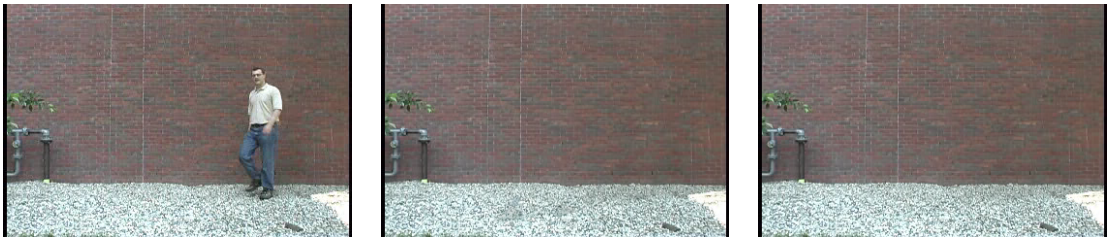
Background Subtraction

I Given the pixel to classify x_t and filtered codebook $\mathcal{F}$,

II Find the codeword c_m in $\mathcal{C} = \{c_i | 1 \leq i \leq L\}$ matching to pixel x_t

III

$$FG(x) = \begin{cases} background & \text{if matching codeword is found} \\ foreground & \text{otherwise} \end{cases}$$



(a) Example Frame (b) Mean Background (c) Median Background

Figure 3.4: Background reconstruction

3.2.2 Background Reconstruction

Background subtraction produces a binary labeled video indicating foreground and background regions. We need to construct a single static background image from multiple labeled frames. One can construct a static background by taking average or median of all observations of a pixel classified as background. Computing mean background is computationally more efficient but can lead to artifacts due to shadows, flicker, background movements, etc. Median gives a sharp background image but it can be non-coherent in some places in presence of large background movements. Figure 3.4 shows mean and median background images. Observe the unwanted shadow in the mean background image.

3.2.3 Trajectory Estimation

We need to estimate object trajectories from the binary segmented videos. In case of single object videos, one can simply compute the centroids of the foreground blobs. This can lead to erratic trajectories due to inaccurate segmentation. Also, in 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 [16] for trajectory estimation. This tracker performs a connected-component tracking in binary segmented frames using Kalman filtering [76]. When Kalman filter's prediction suggests a possible overlap of objects in next frame, a reliable Mean-shift tracker [18] is used on the actual video frames. The functioning of the hybrid tracker is depicted in Figure 3.5.

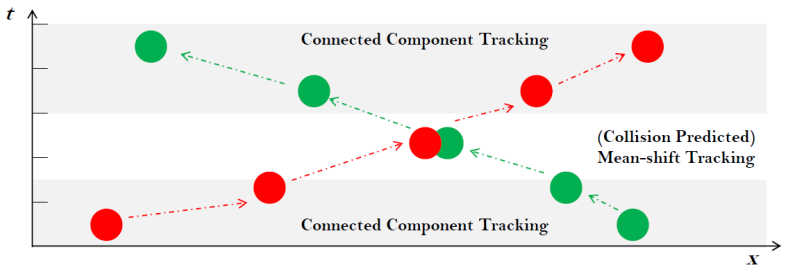


Figure 3.5: Hybrid tracking for object trajectory estimation

3.3 Modeling Moving Camera Videos

Most home-user created videos are captured by hand-held devices and hence they suffer from camera shake. Moreover, most videos with interesting action, like sports, dance, etc. have significant camera motion, as a cameraman tends to focus the camera on the moving target. Static background techniques as explained in previous section fail to model such videos. We use image mosaicing to model the background of such moving camera videos.

Consider the video frames shown in Figure 3.6(a). These frames are taken from a casually captured moving camera performance video. A frame at any point in time offers a limited field of view of the surroundings. If the span of camera motion is limited, it is possible reconstruct an extended field of view of the complete surroundings using mosacing. Given a complete representation of the surrounding scene, it is possible to create an extended field of view video by mapping each of the original frame to reconstructed background frame. This extended field of view video is free from any camera motion and can be modeled as a static camera video. In the following subsections we discuss the rationale, assumptions and required processing for mosaicing based modeling of moving camera videos.

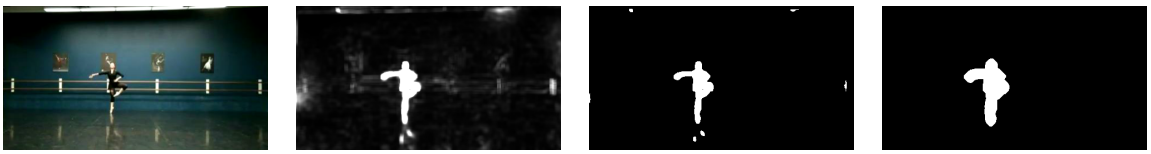
Frame 2072 Frame 1262 Frame 782 (R) Frame 902 Frame 1412

(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

Figure 3.6: Key-frame selection and mosaicing



(a) Aligned frame (b) Distance image (c) Initial Object mask (d) Mask after cleaning

Figure 3.7: Object mask after different stages of processing

3.3.1 Assumptions

We make the following assumptions to model the video based on background scene mosaicing.

1. Captured scene is sufficiently distant from the camera. This assumption holds true for nearly all long shot videos.
2. Camera translation is negligible compared to the scene-to-camera distance. This assumption holds true for videos with relatively stationary vantage point. The restriction on translation can be relaxed for planar backgrounds.
3. Scene background is stationary and non-cluttered.

Many casually captured videos such as stage performance videos or sports videos satisfy these assumptions. These assumptions allow us to reconstruct the scene background as a planar mosaic with reasonable accuracy by computing a series of frame-to-frame homography transformations.

3.3.2 Modeling

To create an object and scene background based video model, it is necessary to estimate scene background as well as object volumes. Reconstructing a clean background mosaic in presence of moving objects is a challenging task. Similarly, segmenting video objects in presence of camera motion is also challenging. Given solution to one, the other problem can be solved in a fairly simple way. Many interactive video object segmentation systems have been proposed previously [73]. These systems can generate pixel-accurate object masks but they are computationally expensive and requires significant user input. On the contrary, reliable and accurate background mosaics can efficiently be created, even in presence of moving objects, with small amount of user interaction. Once a static scene background is available, approximate object segmentation can be done using simple background subtraction techniques and morphological processing. Since, our interface allows only temporal manipulations, pixel-accurate masks are not required, only the outer bounding-boxes of the objects are needed. Hence, we use mosaicing based segmentation approach as explained in the following subsections.

3.3.2.1 Background Mosaicing

Mosaicing is a well explored problem in computer vision literature [67, 12]. Most approaches consider the general problem of constructing a seamless mosaic from a collection of images. These approaches can be simply extended to videos by giving every frame of the video as input to the mosaicing algorithm. This is an overkill due to high amount of temporal redundancy in a video. Also, moving objects in the video create ghosting artifacts in the mosaic. We employ an approximate frame overlap based scheme for key-frame selection and interactive object removal for efficient mosaicing. The complete procedure for key-frame selection and mosaicing is explained here step-by-step.

Feature Extraction: We use the feature based image alignment for mosaicing, as explained in Chapter 2. We extract SIFT feature points and corresponding invariant descriptors [45] for this purpose. Though, the background mosaic construction utilize only a subset of the video frames, features are extracted from all the frames as they are later used for warping the original video frames to the extended field-of-view background mosaic.

Approximate Frame Alignment We select every k^{th} frame from the original video and compute feature-flow between adjacent frames using SIFT descriptor matching. 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.

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 away most of the key-frames, retaining a small number of frames in the final set. For a ballet video sequence of 2495 frames, choosing $k = 10$ leads to 250 frames in the initial key-frames set. Out of 250, only 5 key-frames are selected for mosacing after overlap based pruning (See Figure 3.6)

Accurate Frame Alignment: Accurate frame-to-frame homographies are computed for this small ordered set of key-frames using an improved RANSAC algorithm [24] as outlined here.

Let C denote the set of correspondences $C = \{P \leftrightarrow P'\}$ and $d(P_i, P'_j)$ denote the Euclidean distance between two points P_i and P_j. The improved RANSAC based algorithm operates as follows,

1. Randomly choose a subset S of 4 point correspondences from C.
2. Use this subset to estimate the 8 parameter perspective transformation H by solving Equation 2.2.
3. Determine the set of inliers $I = \{P_i \leftrightarrow P'_i \in C \quad | \quad (P'_i, H.P_i) < \epsilon\}$ which is the set of correspondences consistent with the estimated H parameters.
4. Repeat Steps $1 - 3$ for N times and choose the set of inliers for which $|I|$, number of inliers is the largest.
5. Estimate accurate H parameters using the set of inliers using linear least-squares or similar technique. This solution is the result of the *normal* RANSAC algorithm.
6. **Refinement Step:** Use the estimated H parameters obtained in previous step to obtain a new set of inliers.
7. Repeat steps $5 - 6$, several times.

The idea here is to grow the number of inliers after each refinement step. As the number of inliers increase the probability of failure (estimating erroneous H) decreases significantly. Farin [24] shows a detailed analysis of the relationship between various RANSAC parameters and probability of success.

Foreground Removal: We present the final set of key-frames to the user as a filmstrip for marking the foreground regions. Since, the spatial ordering and overlap based selection prunes away most of the redundant frames from the initial set, the amount of interaction required for marking the foreground objects in a few key-frames is significantly low. In most of our experiments, the interaction time required for marking moving foreground regions in the final key-frames was observed to be less than 30-40 seconds.

Hole Filling: Once the foreground regions are marked, we warp the final key-frames to the middle frame using accurate homographies and mask out the foreground regions in warped frames. We compute a binary background mask of the size of 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 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.

Final Mosaic Composition: Once all the final key-frames are aligned and masked for the foreground regions, the final background mosaic is composited using a laplacian pyramid based blending technique [15]. We use an open-source utility *Enblend* based on [79, 15] for compositing final mosaic from the aligned key-frames.

The key-frame selection and mosaicing process explained above is summarized here.

Algorithm 2 Key-frame Selection and Mosaicing

1. Extract point features and descriptors from the video frames.
2. Select every k^{th} frame as key-frame and compute pairwise feature-flow.
3. Estimate approximate camera motion using pairwise feature-flow.
4. Order key-frames spatially based on camera motion and estimate overlap.
5. Select a subset of key-frames based on spatial ordering and overlap.
6. Compute accurate frame-to-frame homographies for the pruned set.
7. Interactively remove foreground objects from the final key-frames.
8. Warp final key-frames to the reference frame.
9. Compute the overlap mask indicating unfilled pixels due to foregrounds.
10. Iteratively add new frames until there are no unfilled pixels.
11. Blend the warped images to create a seamless background mosaic.

3.3.2.2 Object Segmentation and Trajectory Estimation in Extended Field-of-view Video

Once a reliable background mosaic is composited, the next step is to create a static background, extended field-of-view video which is free from any camera motion. We use the improved RANSAC algorithm explained previously for robust alignment of each input video frame to the background mosaic. Once the frame is warped to the mosaic, object segmentation can be done using standard background subtraction techniques. Per-pixel thresholding in difference image leads to a noisy foreground mask with holes and clutter. To obtain a cleaner mask we use a neighborhood based distance thresholding as explained in [1]. This method is briefly summarized here.

- Let I_B be the mosaic image and I_w be the projection of current image on plane of I_B.
- Define a difference image as, $I_{dist} = Dist(I_w, I_B)$. Among various distance metrics, Mahalanobis distance in YUV colorspace gives the best results.
- Foreground image is obtained by thresholding the difference image I_{dist} using the following rule.

$$FG(i,j) = \begin{cases} 1 & if \sum\limits_{x,y \in N_w} I_{dist}(x,y) \geq T \\ 0 & otherwise \end{cases}$$

We post-process this mask using morphological operations for filling holes and removing clutter. We retain only the largest top k connected components as objects, where k is the number of objects in the video specified by the user. Foreground masks for an example frame at different stages of processing are shown in Figure 3.7.

Once we have clean binary segmented warped frames, we merge warped frame I_w and mosaic frame I_B using feathering along the object boundary. These frames are used to create final composite video with complete field of view of the surroundings. This video is free from any camera motion and object trajectories can be estimated in this video in the same manner as static camera videos as explained in subsection 3.2.3.

3.4 Summary

In this chapter, we introduced our representation for modeling long shot videos and discussed the processing required to build this representation. We model the long shot videos using a static background image and segmented object volumes. If the camera is static, we estimate the background automatically using an adaptive codebook based algorithm. In presence of camera motion, we use an interactive algorithm to create an extended field-of-view background mosaic. We warp the original video frames to the mosaic space and create an extended-field-of-view, motion compensated video. Once the static background image is estimated, we perform background subtraction in each frame to extract the moving foreground objects. The segmented objects are marked by an approximate outer bounding box. After

the foreground objects are segmented, we use a hybrid tracking approach to estimate the space-time object trajectories and assign a unique identity to each object.

The programmatic representation used for storing the video model is built around three components, (a) background image, (b) structures storing the object state, and (c) structure storing the camera state. These structures are minimally defined in Listing 3.1.

```
struct VideoModel {        /* Structure to store video model       */
  Image      background;   /* Stores estimated scene background    */
  ObjectState*  object;    /* Stores state of each object          */
  ApertureState* camera;   /* Stores state of the camera aperture  */
}

struct ObjectState {       /* Structure to store object state             */
  int    objectID;         /* Unique object identity                      */
  Rect* boundingBox;       /* Object location in each frame               */
  int    presentIn;        /* Frames of modified video having the object  */
  int    whereFrom;        /* Source frame for frames in presentIn        */
}

struct ApertureState  {    /* Structure to store camera aperture state    */
  Rect*    frameBox;       /* Aperture window coordinates                 */
  float*  tiltAngle        /* Aperture orientation/tilt in each frame    */
  float*  scaleFactor;     /* Aperture scale/zoom in each frame          */
}
```

Listing 3.1: Structures used for internal representation of the video

The $3D$ representation displayed to the user is shown in Figure 3.2 (See Interaction Grid). The ground plane (plane at $(x, y, 0)$) is the background image stored in the `background` element of the `VideoModel` structure. The object trajectories are represented by (x, y, t), $3D$ line plots. The (x, y) coordinates of the trajectories are obtained by finding the midpoints of the `boundingBox` coordinates stored in the `ObjectState` structure corresponding to each object. The temporal coordinates (t) for the trajectories, are stored in the `presentIn` element of the `ObjectState` structure. The trajectories are colour coded based on the unique object identity stored in the `objectID` element of the `ObjectState` structure.

The `ApertureState` structure stores the position of the camera aperture in the background space in each video frame. In case of a static camera video, the `frameBox` element is initialized to the original frame dimensions otherwise it is initialized to the warped frame dimensions.

In the next chapter, we discuss various interactive operations based on this representation which allow the users to perform various object and camera manipulations. The structures `ObjectState` and `ApertureState` are modified after each operation. These modified structures are then used to composite new videos.

Chapter 4

Interactive Operations

In the previous chapter, we discussed the scene mosaic and object trajectories based representation for modeling long shot videos. In this chapter, we demonstrate how this representation is used to enable object centric video interaction. We built a prototype interface using the scene mosaic and object trajectories based video representation, shown in Figure 4.1. This interface represents the object trajectories using $3D$ line plots in a space-time grid named interaction grid. Users can perform various navigation and manipulation operations in this interaction grid and visualize the resulting occupancy and overlap of objects in a separate $3D$ grid called visualization grid.

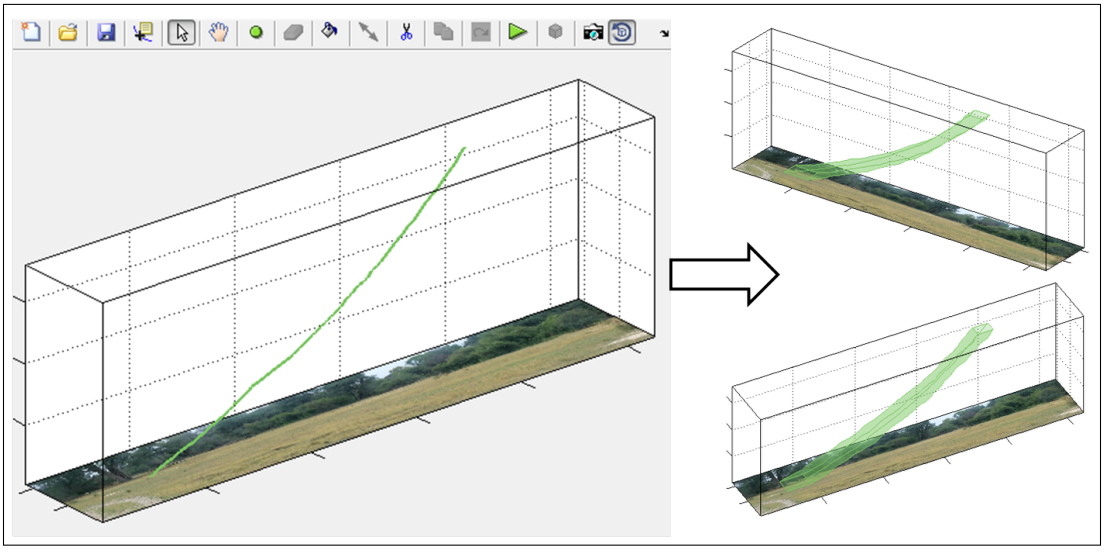


Figure 4.1: A snapshot of the trajectory based interface

We use object trajectories as basic interaction elements and propose various object and camera operations as simple interactions with the object trajectories. Object operations allow users to navigate and manipulate video objects temporally by scrubbing or modifying their trajectories, allowing them to produce various object centric effects in the video. Camera operations allow users to alter path, tilt and zoom of the camera aperture based on the object positions, allowing users to perform simple cin-

ematographic experiments in an intuitive way. Object and camera operations, in combination allow a user to perform a number of high-level video manipulations in a simple and interactive manner. In the following sections, we discuss these operations and some example compositions in detail.

4.1 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 simply scrubbing the object trajectories or create various temporal manipulations by interactively manipulating the object trajectories. We discuss both navigation and manipulation operations in detail in the following subsections.

4.1.1 Object Centric Video Navigation

The user can control video navigation by scrubbing object trajectories with 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 of creating videos in which users can create new videos, the way they browse it.

Simple Video Navigation: This mode of navigation simply replaces the timeline slider by object trajectories. In this mode, the user does not alter the video frames but the video playback is controlled by the current mouse position on the object trajectories. This browsing mechanism is similar to many direct manipulation interfaces discussed in Chapter 2. However, there are two key advantages of using $3D$ trajectories as control elements over $2D$ laid out trajectories. Long range indoor motions can induce complex $2D$ trajectories containing self-occluding loops as shown in Figure 4.2. A $3D$ representation simplifies the trajectories, free from loops and self-occlusions. Also, $2D$ 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 exactly the same $2D$ trajectories. Adding the temporal dimension resolves such conflicts.

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, the user's scrubbing action results in motion of only a single, currently active object, replacing the other moving objects by constant background.

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

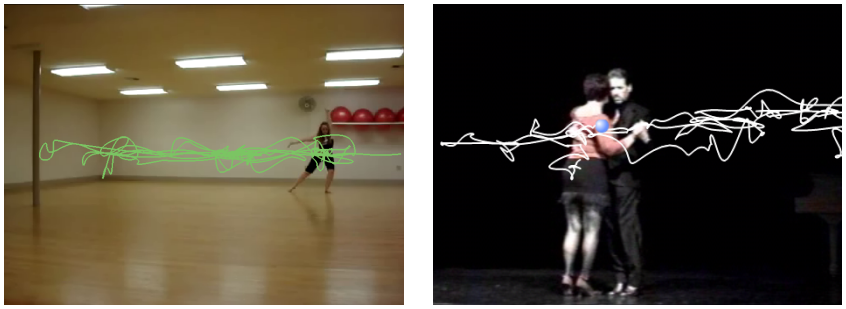


Figure 4.2: Example of a complex motion trajectories laid out in (x, y)

(a) Simple video navigation (b) Single object navigation (c) Composition by navigation

Figure 4.3: Modes of object centric video navigation

Figure 4.3 shows video frames for simple video navigation, single object navigation and composition by navigation. Frame shown in Figure 4.3(a) is the actual video frame, frame shown in Figure 4.3(b) is generated by superimposing active object segment on pre-computed background for object centric navigation and frame shown in Figure 4.3(c) is generated by compositing the objects according to the last scrubbed position for the dancer in black and the current position for the dancer in blue.

4.1.2 Object Centric Video Manipulation

We propose a number of interactive operations on object trajectories which allow users to directly perform various temporal manipulations on video objects. These curve manipulation operations have strong visual meaning associated with it, for example, erasing a trajectory removes the object from the video or copying the trajectory creates a clone of the object in the video etc. Such visually meaningful interactions make the operations intuitive and easy to grasp.

Consider the surveillance video of 24 seconds depicted by 10 keyframes in Figure 4.4. The video has three main objects/events. First a red car enters the scene at 3 seconds from the right, moving towards left and leaves the field of view at 10 seconds. At 11 seconds a blue car enters the scene from the left, moving towards the parking area and parks at 22 seconds. During the entire video, a person wearing black can be seen walking from a long distance, towards the road. The manipulation operations in this section would be illustratively discussed in reference to this video sequence.

Reordering: This operation allows a user to delay or advance the events in the video independently. Users can drag and move the 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 relative order of two events, etc. Figure 4.5 illustrates this operation on *PETS2000* video sequence. .

Retiming: This operation allows the users to change the pace (velocity) of the different objects/events independently of the video playback rate. To achieve this, the user can select a trajectory segment and drag any of the two endpoints of the segment to stretch or shrink the trajectory. User 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. Figure 4.6 illustrates this operation on *PETS2000* sequence.

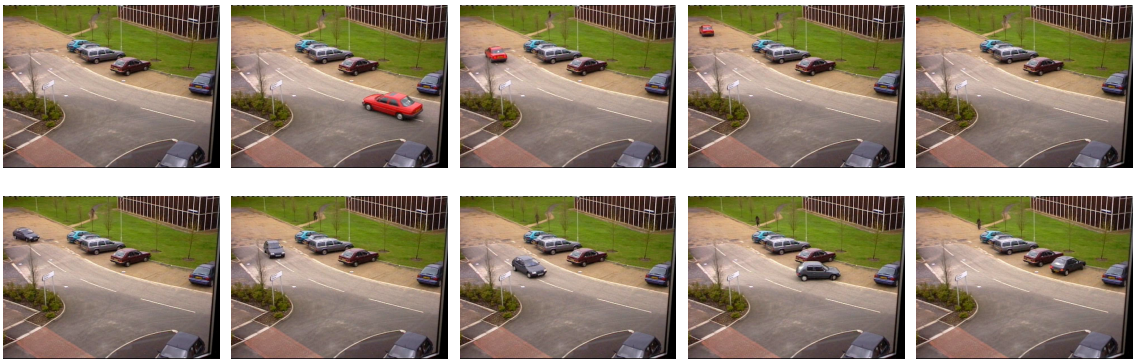


Figure 4.4: Keyframes from a 24 seconds surveillance sequence *PETS2000*[1]

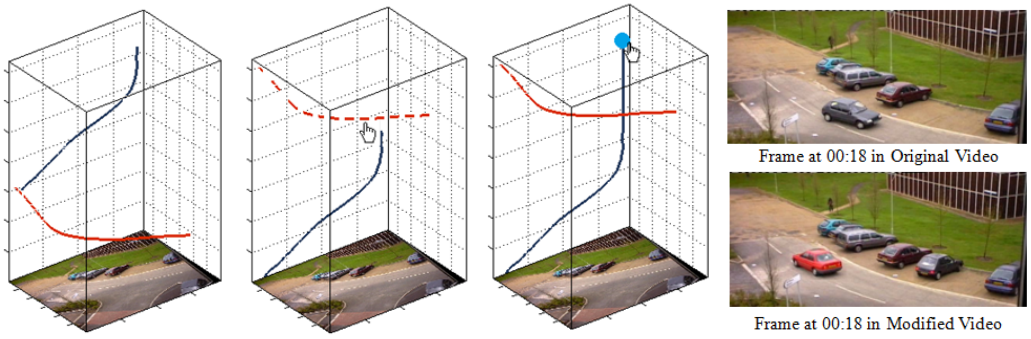


Figure 4.5: (a) Original state of red and blue cars' trajectories. (b) User drags the trajectories to reorder events. (c) User extends the blue car's trajectory to prevent it from disappearing (d) Frames taken at 18^th second from original video (top) and modified video (bottom), observe the change in order

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

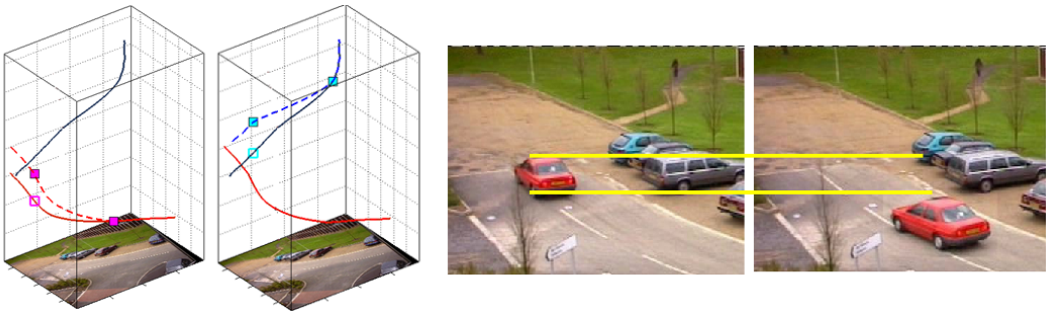


Figure 4.6: (a) User stretches the red car's trajectory to slow it down (b) User shrinks the blue car's trajectory to speed it up (c) Frames showing the result of stretching the red car's trajectory. Observe the difference in displacement at the same time.

Temporal downsampling can be achieved by skipping intermediate samples (object frames). Temporal upsampling requires interpolation (known as frame blending) between samples. Due to the motion of the objects interpolation introduces blurring artifacts. Blurring becomes severe as the upsampling rate increases. At higher upsampling rates, optical-flow based blending techniques [47] should be used to produce better results. Apart from retiming, it is also possible to pause an object by selecting and stretching a single point on the trajectory. In Figure 4.5(c), blue car's trajectory is extended in this manner to pause it after the trajectory ends.

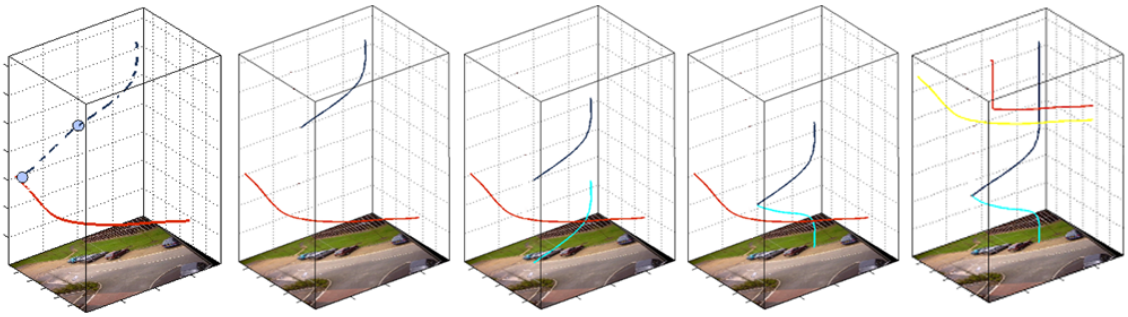
Cloning: This operation allow the users to create a clone of an object by simply selecting and copying the object's trajectory. The copied trajectory needs to be time-shifted to create multiple visible instances.

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

Reversal: This operation allow the users to reverse the activity by inverting the trajectory.

Annotation: This operation allow the users to tag specific events or objects in a motion synchronous manner. To attach a moving annotation to the object of interest, user selects the trajectory segment and adds an annotation to it.

Figure 4.7(a) demonstrates an example of multiple modifications being performed. First, a segment of the blue car's 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 from the road. This remaining segment is copied and time-shifted to the beginning of the video. This time-shifted trajectory segment is then inverted. Red car's trajectory is also copied and time-shifted. New trajectory is erased at the trailing end and extended till the end of the video. Effect of these operations is illustrated in Figure 4.7(b). Compare these frames with the original frames (Figure 4.4) and the modified



(a) Multiple object operations

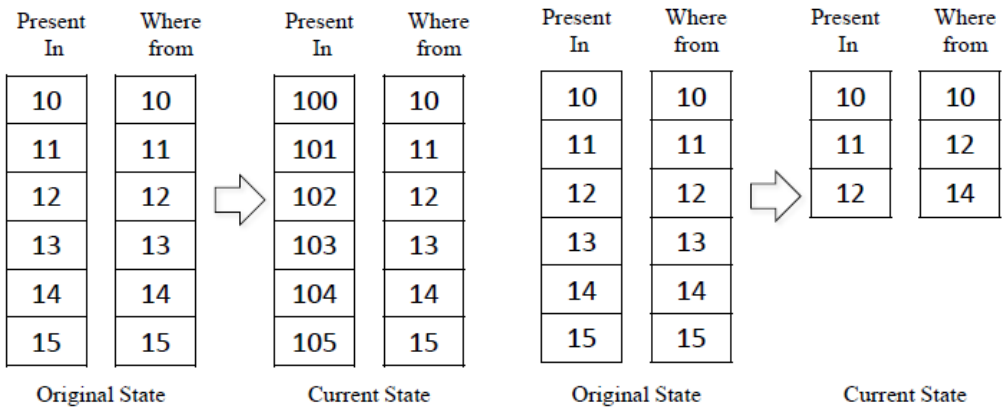
(b) Frames from the modified video after performing the operations shown in (a)

Figure 4.7: 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 the new endpoint is extended. Frames from the modified video are shown in (b).

trajectories (Figure 4.7(a)). In the modified video, blue car is seen already parked in the parking area. Then it is seen moving in the reverse direction from the parking area, pausing on the road for a moment and getting 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.

As discussed in section 3.4, the state of the object trajectories is internally stored in the `objectState` structure, defined in Listing 3.1. The elements `presentIn` and `whereFrom` are arrays of indices storing the frame mappings between the original trajectory and the manipulated trajectory. When the manipulated video is created, each output frame is rendered by pasting each object independently on the static background based on this mapping. The elements of the array `presentIn` indicates the frames the object should be pasted in. The corresponding elements of the array `whereFrom` indicate which original frames the object should be copied from. Figure 4.8 shows some examples of these mappings for retime and reorder operations.

Since new frames are rendered by compositing objects from original frames on the static background, visible seams may exist at the object boundaries. Though the spatial location of the objects is not altered in this process, these seams are created due to small variations in illumination across frames. We use alpha blending to reduce the visible seams by suppressing the sharp boundaries. If illumination differences are severe, gradient domain blending techniques like [52] should be used instead.



(a) Mapping for Reordering Operation

(b) Mapping for Retiming Operation

Figure 4.8: Look up tables showing the mapping of original state and current state for: (a) reordering operation (the trajectory is shifted to achieve delay of 100 frames) (b) retiming operation (the trajectory is shrunk to achieve a speed up of 2)

4.2 Camera Operations

As explained in Chapter 3, our representation models a moving camera video as an extended field of view, static video by creating a static scene mosaic and warping the original video frames to the mosaic. This completely destroys the camera-object association in the video which is an important aspect of storytelling. Moreover, for applications other than video synopsis, panoramic backgrounds without many events of interest unnecessarily occupies major display space. We introduce some intuitive camera operations to produce novel moving camera videos from the 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 by a movable and scalable visual aperture (a view window) in the mosaic space. Movement of this aperture is restricted to be planar to avoid view-interpolation problems. Location, orientation and scale of this aperture along timeline decides camera path, tilt and zoom in the video.

Let $C_t = (x, y, \theta, S)$ represent the aperture parameters, location (x, y), orientation (θ) and scale S at any time t. Asking a user to explicitly specify these parameters is complex and tedious. When a user shoots a video, the objective is always simple and in terms of objects and events, like follow the car till it crosses the bridge, zoom in to focus on the lady standing by, etc. The user should be able to control the aperture in the mosaic space in a similar fashion. We utilize the visually meaningful object trajectories to allow the user to specify camera parameters in terms of objects and events of interest.

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

Aperture Path: Aperture path is a location map of user's desired focus of attention at anytime. User can browse the video by scrubbing the object trajectories and create an anchor point on the object trajectory to mark object or activity of interest. Consider the object trajectories as shown in Figure 4.9. 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: Transition from P_i to P_{i+1} is interpolated. To prevent unusually sharp transitions, selection of anchor points is restricted by distance to time ratio - Aperture Velocity.

Figure 4.10 shows effect of setting the aperture path on the *PETS2000* video sequence. The interaction grid on the left shows the user selected anchor points by green markers and resulting camera trajectory by the black dotted line plot. The frames on the right are taken from the modified video. Compare these frames with the original frames shown in Figure 4.4 to observe the effect of camera aperture.

Aperture Scale: Aperture scale can either be fixed or adapted to the change in object scale. If aperture scale is adapted, video will have a tracked zoom to focus on the object of interest. Adapting the aperture scale on per-frame basis produces distracting zooming effects. To avoid this, we fit a quadratic model to per-frame observations. Also, we limit the scaling to a magnification factor of 2 to prevent excessive blurring due to interpolation. Figure 4.11 shows the effect of adaptive aperture scale on two video sequences.

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

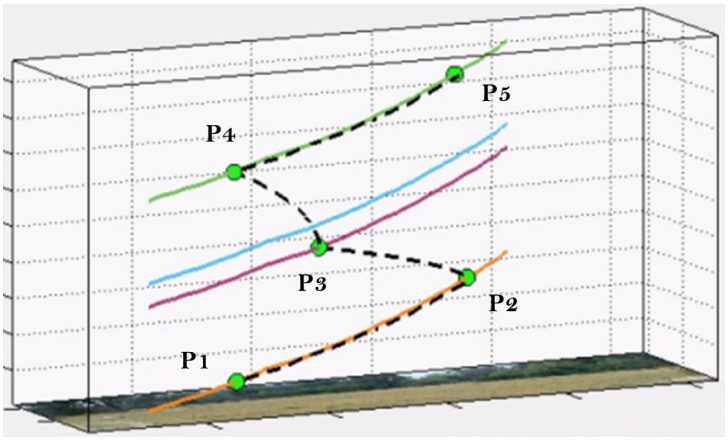


Figure 4.9: Specifying camera aperture path: From P_1 to P_2, aperture follows the trajectory; From P_2 to P_3 aperture path is interpolated

(a) User specified path

(b) Frames from output video

Figure 4.10: Effect of specifying aperture path on *PETS2000* video sequence

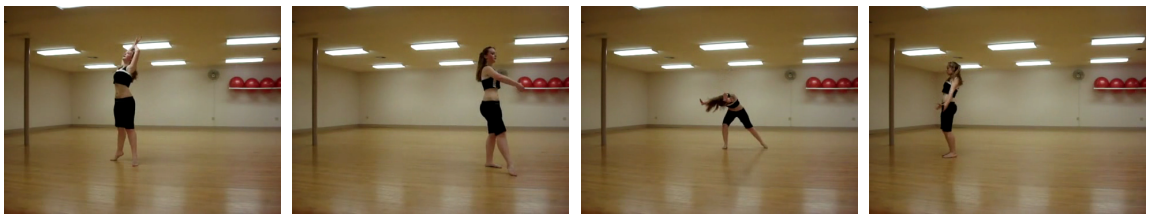
Figure 4.11: Effect of specifying aperture scale on two video sequences

4.3 Example Compositions

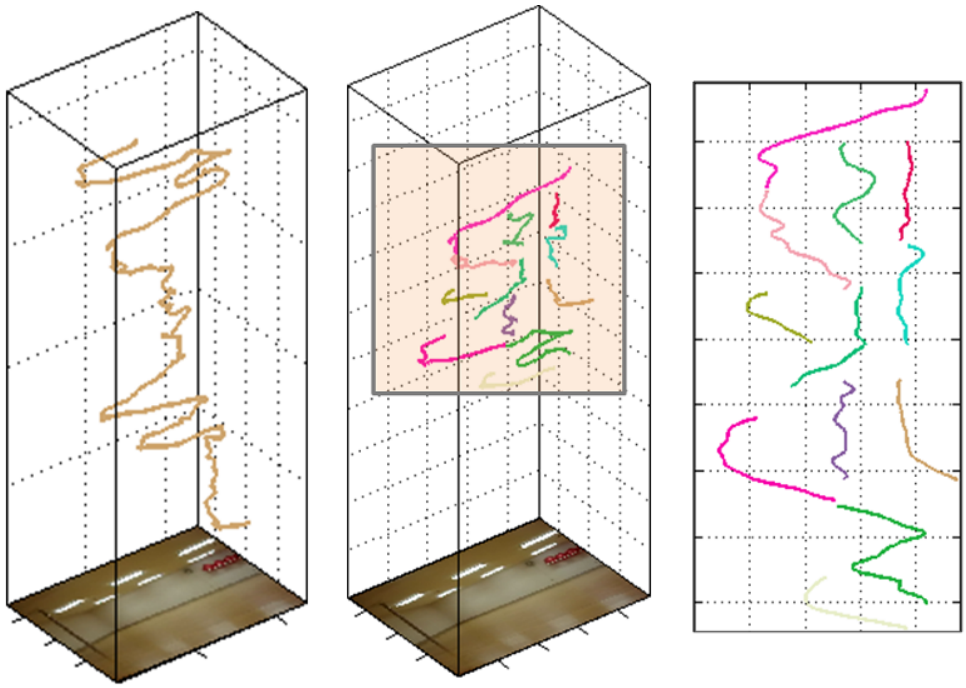
We demonstrated effects of several object and camera manipulations on various video sequences. This section demonstrates interactions and manipulations performed on some more long shot videos.

Figure 4.12 demonstrates composition of a dance video montage. Figure 4.12(a) shows keyframes from the original video sequence (*Only Hope Lyrical*). User breaks the dancer's trajectory into multiple segments and arranges them in a non-overlapping fashion. Figure 4.12(b) shows the dancer's original trajectory. User's arrangement of the trajectory segments is shown in Figure 4.12(c). Figure 4.12(d) shows magnified XY view of the interaction grid. Figure 4.12(e) shows keyframes from the montage video.

Figure 4.13 demonstrates keyframes from the *Running Lion Sequence*. Figure 4.14 demonstrates the scene mosaic constructed from the keyframes. Figure 4.15 shows the original state of the trajectories and manipulated trajectories.



(a) Keyframes from the *Only Hope Lyrical* video sequence



(b) (c) (d)

(b) Original trajectory of the dancer. (c) User's arrangement of the dancer's trajectory segments. (d) Magnified XY view of the interaction grid

(e) Keyframes from the user created montage video

Figure 4.12: Composition of a dance video montage [2]

[2] Video Source: `http://www.youtube.com/watch?v=Hlf5WU5ICUQ`, Uploaded by user: d100lt

Figure 4.13: Keyframes from the *Running Lion* video sequence [3]

Figure 4.14: Scene mosaic for *Running Lion* video sequence

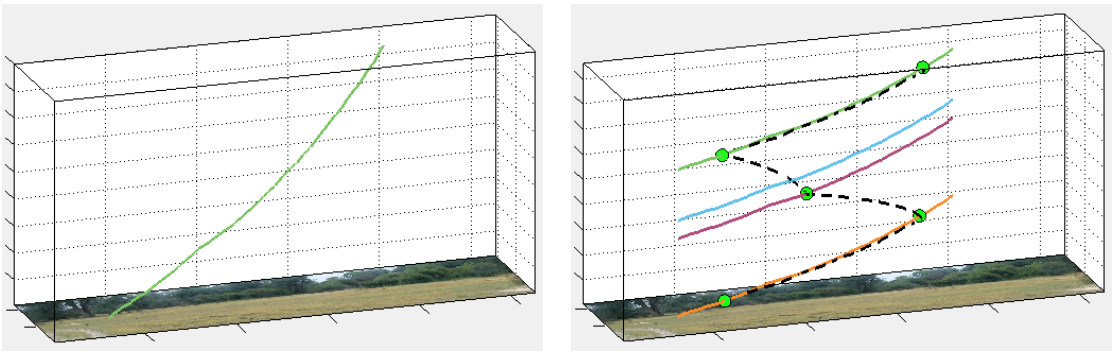


Figure 4.15: Interaction grid for the *Running Lion* video sequence before and after the manipulations

Figure 4.16: Example frame from the output video showing several lions

[3] Video Source: `http://www.youtube.com/watch?v=cD7dHTDudHM`, Uploaded by user: blazinggecko

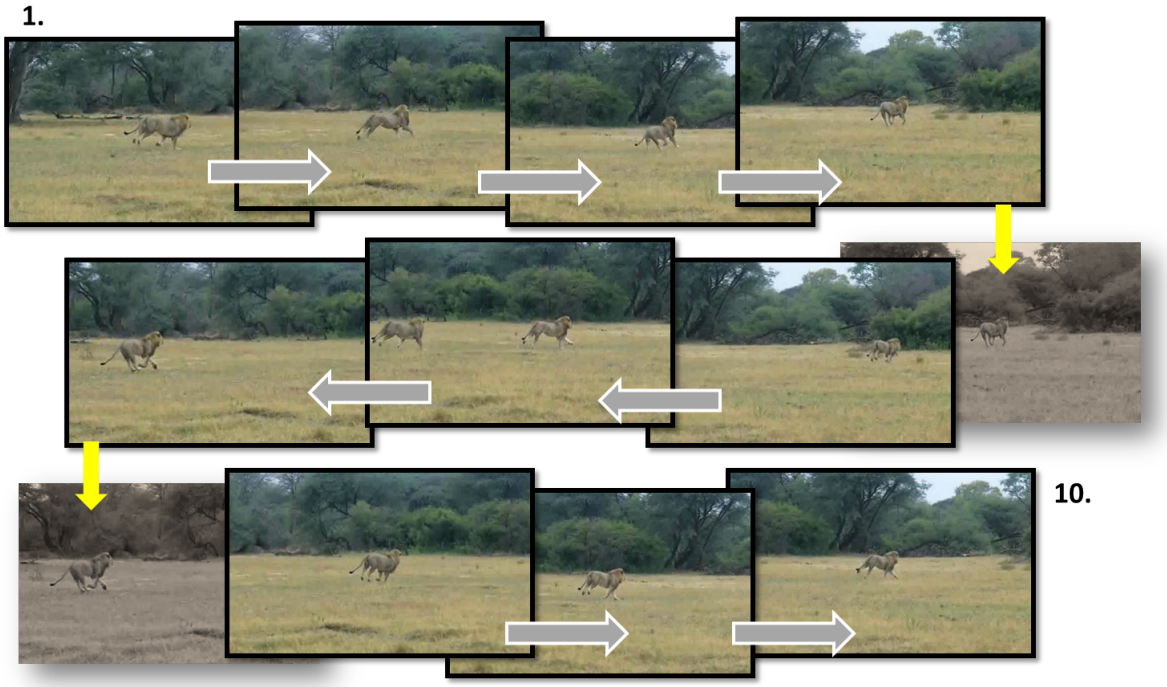


Figure 4.17: Effect of setting aperture path on cloned lions sequence. Gray arrows indicate direction of the camera motion. See the tree branches in the background to observe the simulated camera motion.

In this composition, first the running lion's trajectory is copied several times, creating multiple clones of the lion. The clone lions' trajectories are time shifted to produce desired temporal gap creating an 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 4.16 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 camera aperture path is shown in Figure 4.17. The camera moves from left to right, following the first lion, then sweeps towards left focuses on the second lion almost when it has reached at the middle of the scene, sweeps further left to focus on the third and fourth lions and then follows the last lion to the right end of the scene.

Figure 4.18 shows keyframes of the *Bluebird Ballet Sequence*. Figure 4.19 demonstrates the scene mosaic constructed from the keyframes. Figure 4.20 shows the state of the interaction grid before and after manipulation.

In this composition, the dancer's trajectory has been cut at iconic movements and the endpoint at each cut is extended till the end of the video. The modified video shows the dancer leaving a copy in its iconic movement as each breakpoint is crossed. Figure 4.21 shows the last frame from the modified video showing the iconic positions of the dancer at all breakpoints.

Figure 4.18: Keyframes from the *Bluebird Ballet* video sequence [4]

Figure 4.19: Scene mosaic for the *Bluebird Ballet* video Sequence

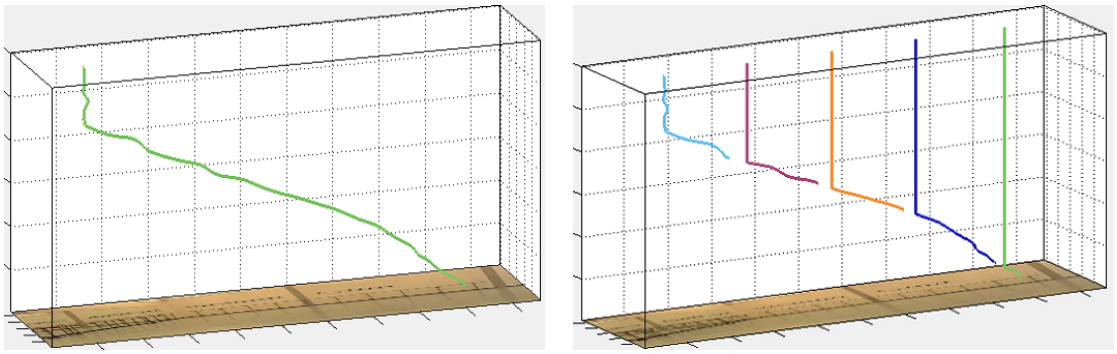


Figure 4.20: Interaction grids for the *Bluebird Ballet* video sequence before and after the manipulations

Figure 4.21: Last frame from the modified video showing iconic positions of the dancer

[4] Video Source: `http://www.youtube.com/watch?v=w6IagNw9SgQ`, Uploaded by user: klara houdet

Chapter 5

Conclusions

The exponential growth in consumer created video data has posed the need to cater the requirements of a home-user to be able to interactively organize, summarize, and manipulate this data. In this thesis, we focus on simplifying the video interactions for object-centric video navigation and manipulation tasks. We discussed the limitations of the present day interaction techniques based on frame-time video representation for video navigation and manipulation, justifying the need for object-centric video representation and interaction.

We proposed a scene mosaic and object trajectories based video representation to enable simple interaction for navigation and temporal manipulation of long shot videos. We model a video using a static background and a collection of moving objects represented by $3D$ space-time trajectories. We model the background using an adaptive codebook when the camera is static. In presence of camera motion, we model the video background by constructing a static background mosaic. We use the static background to segment the moving objects and estimate object trajectories.

We proposed a novel interaction scheme which utilizes object motion trajectories as basic interaction elements and defined simple and meaningful operations for navigation and temporal manipulation of video objects. We represent these object trajectories against the background mosaic in a $3D$ space-time interaction grid and allow the users to perform various temporal curve manipulation operations on these trajectories. These operations enables a user to perform various interesting operations like *retiming, reordering, removal, cloning,* etc. by performing simple curve manipulation operations like *move, resize, erase, copy,* etc in a 'click and drag' fashion.

We also utilized the static background representation to propose simple camera operations to alter focus of attention in the videos. These operations model the camera as a movable and scalable aperture and allow the users to specify the aperture path, tilt and zoom w.r.t the object trajectories to simulate desired camera motion. Using combination of object and camera operations, users can produce seemingly complex video effects in a simple and intuitive manner.

Though our representation is not generic enough to model any dynamic video, it is a very natural representation to manipulate long shot videos like surveillance, stage performance, sports etc. We

demonstrated the applicability of these interactions by creating various interesting compositions of several consumer captured long shot videos.

Though our representation can accurately model long shot videos, the feature based mosaic construction may fail to estimate correct homographies under extreme conditions, like lack of texture in 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 focus on probably more important video segments.

Overall, we believe that augmenting video context and motion cues with user interface can significantly improve the usability of video manipulation tools. The work in this thesis is one such step to achieve that overall goal. We believe that the fidelity and popularity of such interfaces will significantly increase with the progress in computer vision and video processing techniques.

Related Publications

1. Rajvi Shah and P. J. Narayanan, “Trajectory based Video Object Manipulation”, IEEE International Conference on Multimedia and Expo (ICME 2011), Barcelona, Spain, July 2011.

Bibliography

[1] Til Aach, André Kaup, and Rudolf Mester. Statistical model-based change detection in moving video. *Signal Process.*, 31(2):165–180, March 1993.

[2] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. Interactive digital photomontage. *ACM Transactions on Graphics*, 23(3):294–302, August 2004.

[3] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12:43–77, 1994.

[4] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110:346–359, June 2008.

[5] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Comput. Surv.*, 27(3): 433–466, September 1995.

[6] Michael J. Black. Robust dynamic motion estimation over time. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 296–302, 1991.

[7] Michael J. Black and P. Anandan. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63:75–104, January 1996.

[8] 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*, 26(9):1124 –1137, sept. 2004.

[9] Dr. Gary Rost Bradski and Adrian Kaehler. *Learning opencv, 1st edition*. O'Reilly Media, Inc., first edition, 2008. ISBN 9780596516130.

[10] Gary R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, (Q2):15, 1998.

[11] T J Broida and R Chellappa. Estimation of object motion parameters from noisy images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:90–99, January 1986.

[12] Matthew Brown and David Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74, 2007.

[13] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV (4)*, pages 25–36, 2004.

[14] Andrés Bruhn, Joachim Weickert, Christian Feddern, Timo Kohlberger, and Christoph Schnrr. Real-time optic flow computation with variational methods. In *In CAIP 2003*, pages 222–229, 2003.

[15] Peter J. Burt and Edward H. Adelson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, 2(4):217–236, October 1983.

[16] Trista P. Chen, Horst Haussecker, Alexander Bovyrin, Roman Belenov, Konstantin Rodyushkin, Alexander Kuranov, and Victor Eruhimov. Computer vision workload analysis: Case study of video surveillance systems. *Intel Technology Journal*, 9, 2005.

[17] Yung-Yu Chuang, Aseem Agarwala, Brian Curless, David H. Salesin, and Richard Szeliski. Video matting of complex scenes. *ACM Transactions on Graphics*, 21(3):243–248, July 2002.

[18] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligenc*, 25(5):564 – 577, may 2003.

[19] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619, May 2002.

[20] Carlos D. Correa and Kwan-Liu Ma. Dynamic video narratives. *ACM Transactions on Graphics*, 29:88:1–88:9, July 2010.

[21] J. Davis. Mosaics of scenes with moving objects. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 354 –360, jun 1998. doi: 10.1109/CVPR.1998.698630.

[22] Pierre Dragicevic, Gonzalo Ramos, Jacobo Bibliowitcz, Derek Nowrouzezahrai, Ravin Balakrishnan, and Karan Singh. Video browsing by direct manipulation. In *ACM SIGCHI*, 2008.

[23] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 341–346, New York, NY, USA, 2001. ACM. ISBN 1-58113-374-X.

[24] Dirk Farin. *Automatic Video Segmentation Employing Object/Camera Modeling Techniques*. PhD thesis, Eindhoven University of Technology, Dec 2005.

[25] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:167–181, September 2004.

[26] Paul Fieguth and Demetri Terzopoulos. Color-based tracking of heads and other mobile objects at video frame rates. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 0:21, 1997.

[27] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24 (6):381–395, June 1981.

[28] Dan B Goldman, Brian Curless, Steven M. Seitz, and David Salesin. Schematic storyboarding for video visualization and editing. *ACM Transactions on Graphics*, 25(3), 2006.

[29] Dan B. Goldman, Chris Gonterman, Brian Curless, David Salesin, and Steven M. Seitz. Video object annotation, navigation, and composition. In *ACM symposium on User interface software and technology*, pages 3–12, 2008.

[30] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph based video segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[31] C. Harris and M. Stephens. A Combined Corner and Edge Detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[32] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17: 185–203, 1981.

[33] M. Irani and P. Anandan. Video indexing based on mosaic representations. *Proceedings of the IEEE*, 1998.

[34] A. Jepson and M.J. Black. Mixture models for optical flow computation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 760 –761, jun 1993.

[35] A.D. Jepson, D.J. Fleet, and T.F. El-Maraghi. Robust online appearance models for visual tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(10):1296 – 1311, oct. 2003. ISSN 0162-8828. doi: 10.1109/TPAMI.2003.1233903.

[36] Hong-Wen Kang, Xue-Quan Chen, Yasuyuki Matsushita, and Xiaoou Tang. Space-time video montage. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1331–1338, 2006.

[37] Thorsten Karrer, Malte Weiss, Eric Lee, and Jan Borchers. Dragon: A direct manipulation interface for frame-accurate in-scene video navigation. In *ACM SIGCHI*, 2008.

[38] Kyungnam Kim, Thanarat H. Chalidabhongse, David Harwood, and Larry Davis. Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*, 11:172–185, June 2005.

[39] Don Kimber, Tony Dunnigan, Andreas Girgensohn, Frank M. Shipman III, Thea Turner, and Tao Yang. Trailblazing: Video playback control by direct object manipulation. In *IEEE International Conference on Multimedia and Expo*, pages 1015 –1018, july 2007.

[40] Genshiro Kitagawa. Non-Gaussian State-Space Modeling of Nonstationary Time Series. *Journal of the American Statistical Association*, 82(400):1032–1041, 1987.

[41] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147 –159, February 2004.

[42] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics*, 22(3):277–286, 2003.

[43] J. Lezama, K. Alahari, J. Sivic, and I. Laptev. Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[44] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. *ACM Transactions on Graphics*, 23:303–308, August 2004.

[45] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60, 2004.

[46] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[47] Dhruv Mahajan, Fu-Chung Huang, Wojciech Matusik, Ravi Ramamoorthi, and Peter Belhumeur. Moving gradients: a path-based method for plausible image interpolation. *ACM Transactions on Graphics*, 28, 2009.

[48] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1615–1630, October 2005.

[49] T.K. Moon. The expectation-maximization algorithm. *Signal Processing Magazine, IEEE*, 13(6): 47 –60, nov 1996. ISSN 1053-5888. doi: 10.1109/79.543975.

[50] Nils Papenberg, Andrés Bruhn, Thomas Brox, Stephan Didas, and Joachim Weickert. Highly accurate optic flow computation with theoretically justified warping. *International Journal of Computer Vision*, 67:141–158, April 2006.

[51] Shmuel Peleg, Benny Rousso, Alex Rav-Acha, and Assaf Zomet. Mosaicing on adaptive manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1144–1154, October 2000.

[52] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transactions on Graphics*, 22(3):313–318, July 2003.

[53] M. Piccardi. Background subtraction techniques: a review. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 4, pages 3099 – 3104, oct. 2004.

[54] Y. Pritch, A. Rav-Acha, A. Gutman, and S. Peleg. Webcam synopsis: Peeking around the world. In *IEEE International Conference on Computer Vision*, pages 1 –8, oct. 2007.

[55] Alex Rav-Acha, Yael Pritch, and Shmuel Peleg. Making a long video short: Dynamic video synopsis. In *Proceedings IEEE CVPR*, 2006.

[56] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23:309–314, August 2004.

[57] Peter Sand and Seth J. Teller. Particle video: Long-range motion estimation using point trajectories. *International Journal of Computer Vision*, 80(1):72–91, October 2008.

[58] Takashi Satou, Haruhiko Kojima, Akihito Akutsu, and Yoshinobu Tonomura. Cybercoaster: Polygonal line shaped slider interface to spatio-temporal media. In *ACM international conference on Multimedia*, pages 202–206, New York, NY, USA, 1999. ACM.

[59] Harpreet S. Sawhney and Serge Ayer. Compact representations of videos through dominant and multiple motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18: 814–830, August 1996.

[60] Haim Schweitzer, J. W. Bell, and F. Wu. Very fast template matching. In *Proceedings of the 7th European Conference on Computer Vision-Part IV*, ECCV '02, pages 358–372, 2002.

[61] I. K. Sethi and Ramesh Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:56–73, 1987.

[62] Mubarak Shah, Krishnan Rangarajan, and Ping sing Tsai. Motion trajectories. *Pattern Recognition*, 23:1138–1149, 1992.

[63] Jianbo Shi and C. Tomasi. Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 593 –600, jun 1994.

[64] Sudipta N. Sinha, Jan michael Frahm, Marc Pollefeys, and Yakup Genc. GPU-based video feature tracking and matching. Technical report, In Workshop on Edge Computing Using New Commodity Architectures, 2006.

[65] C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747 –757, aug 2000.

[66] R. Szeliski. Image mosaicing for tele-reality applications. In *IEEE Workshop on Applications of Computer Vision*, pages 44 –53, dec 1994.

[67] Richard Szeliski. Image alignment and stitching: a tutorial. *ACM Found. Trends. Comput. Graph. Vis.*, 2, 2006.

[68] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 251–258, 1997.

[69] E. Tola, V. Lepetit, and P. Fua. Daisy: an Efficient Dense Descriptor Applied to Wide Baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):815–830, May 2010.

[70] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: principles and practice of background maintenance. In *IEEE International Conference on Computer Vision*, volume 1, pages 255 –261, 1999.

[71] C. J. Veenman, M. J. T. Reinders, and E. Backer. Resolving motion correspondence for densely moving points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:54–72, 2001.

[72] John Wang, John Y. A. Wang, Edward, and H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3:625–638, 1994.

[73] Jue Wang and Michael F. Cohen. Image and video matting: a survey. *ACM Found. Trends. Comput. Graph. Vis.*, 3, 2007.

[74] Jue Wang, Maneesh Agrawala, and Michael F. Cohen. Soft scissors: an interactive tool for realtime high quality matting. *ACM Transactions on Graphics*, 26, July 2007.

[75] Joachim Weickert, Andrés Bruhn, Nils Papenberg, and Thomas Brox. Variational optic flow computation: From continuous models to algorithms. In *International Workshop on Computer Vision and Image Analysis*, 2003.

[76] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[77] Moritz Wittenhagen. Dragoneye - fast object tracking and camera motion estimation. Master's thesis, RWTH Aachen University, Aachen, Germany, October 2008.

[78] Daniel N. Wood, Adam Finkelstein, John F. Hughes, Craig E. Thayer, and David H. Salesin. Multiperspective panoramas for cel animation. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 243–250, 1997.

[79] Yalin Xiong and K. Turkowski. Registration, calibration and blending in creating high quality panoramas. In *IEEE Workshop on Applications of Computer Vision*, 1998.

[80] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Computing Surveys*, 38, 2006.