Selectively De-animating and Stabilizing Videos

Jiamin Bai

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-213
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-213.html

December 11, 2014

Acknowledgement

This work was supported in part by ONR PECASE grant N00014-09-1-0741,
NSF grants CCF-0643552, IIS-1016920, an A*STAR NSS PhD fellowship, gifts and software from Adobe, funding from Nokia, and the Intel Science and Technology Center for Visual Computing.

Selectively De-animating and Stabilizing Videos

by

Jiamin Bai

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ravi Ramamoorthi, Chair
Professor Maneesh Agarwala
Professor Marty Banks

Fall 2014

Selectively De-animating and Stabilizing Videos

Abstract

Selectively De-animating and Stabilizing Videos

by

Jiamin Bai

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Ravi Ramamoorthi, Chair

This thesis presents three systems for editing the motion of videos. First, selectively de-animating videos seeks to remove the large-scale motions of one or more objects so that other motions are easier to see. The user draws strokes to indicate the regions that should be immobilized, and our algorithm warps the video to remove large-scale motion in regions while leaving finer-scale, relative motions intact. We then use a graph-cut-based optimization to composite the warped video with still frames from the input video to remove unwanted background motion. Our technique enables applications such as clearer motion visualization, simpler creation of artistic cinemagraphs, and new ways to edit appearance and motion paths in video. Second, we design a fully automatic system to create portrait cinemagraphs by tracking facial features and de-animating the video with respect to the face and torso. We then generate compositing weights automatically to create the final cinemagraph portraits.

Third, we present a user-assisted video stabilization algorithm that is able to stabilize challenging videos when state-of-the-art automatic algorithms fail to generate a satisfactory result. Our system introduces two new modes of interaction that allow the user to improve an unsatisfactory automatically stabilized video. First, we cluster tracks and visualize them on the warped video. The user ensures that appropriate tracks are selected by clicking on track clusters to include or exclude them to guide the stabilization. Second, the user can directly specify how regions in the output video should look by drawing quadrilaterals to select and deform parts of the frame. Our algorithm then computes a stabilized video using the user-selected tracks, while respecting the user-modified regions.

To my Father and Mother.

Contents

List of Figures

List of Tables

Acknowledgments

I want to thank my Advisor Prof. Ravi Ramamoorthi for supporting my various explorations in different fields in computer graphics and vision. The freedom allowed me to find projects that I truly enjoyed. Prof Maneesh Agrawala has been a great source of encouragement and support when projects were progressing well. I want to thank Aseem Agarwala for being patient and providing insightful suggestions throughout the latter part of my program. I have been fortunate to work with Dr. Manmohan Chandraker and Dr. Dikpal Reddy as well as Dr. Milos Hasan and Dr. Huamin Wang who provided great advice on graduate school.

I also want to thank Prof. Brian Barsky for the opportunities he has provided me. Dr Tian Tsong Ng for providing guidance prior to my graduate school studies in I2R. I was like to thank Prof. Charlee Brodsky, Prof. Mark Perrott, Prof. Martin Prekop and Prof. Elizabeth Raymer-Griffin from CMU for nurturing the photographer in me; It is why I do research in Computational Photography. I would also like to thank Prof. Jean-Paul Bourdier and Prof. Anthony Dubovsky for welcoming me into their classes. I have learnt so much from them.

My graduate life has been wonderful due to the great friends and lab mates I have. In particular, Michael Tao has not only been a collaborator, but also a great friend. Dr. Fu-Chung Huang, Brandon Wang, Florian Hecht, Dr. Dikpal Reddy and Armin Samii helped to make working in lab enjoyable and fun. I want to thank James Do, Linda Yu and Max Zheng for the hours we spend outside of work. I would also like to thank Cecilia Chang for making my last year in graduate school exciting and showing me a different way to live life.

I would like to thank Sean Arietta for playing the guitar, Kathy Ngo, Sheryl Marie, Jessy Berry, Christina Russo, Siang Yun Ang and Armin Samii, Dikpal Reddy, Michael Tao, Brandon Wang, Soham Mehta, Jimmy Andrews, Rahul Narain who agreed to be models.

This work was supported in part by ONR PECASE grant N00014-09-1-0741, NSF grants CCF-0643552, IIS-1016920, an A*STAR NSS PhD fellowship, gifts and software from Adobe, funding from Nokia, and the Intel Science and Technology Center for Visual Computing.

Chapter 1

Introduction

1.1 Visual Media

Visual perception is one of the primary ways humans gain information and understanding from the world. Sight gives us the ability to observe and reason about objects or surroundings at a safe distance almost instantaneously. The ability to see was an important tool for survival in the early days when our ancestors had to avoid dangerous terrain, hunt animals and gather food. Today, sight allows us to drive modern scientific progress; we use sight to observe natural phenomena or complex data visualizations to make discoveries. For example, Joseph von Fraunhofer observed that there are gaps in the spectrum of visible light when magnified. This discovery eventually became the core of astronomical spectroscopy. It allowed scientists to make observations about the composition of stellar objects as different atoms would absorb different specific wavelengths of light [63].

Visual communication is also an integral part of how humans communicate with each other. In fact, our visual perception also has an effect on our auditory senses. The McGurk effect demonstrates an interaction between vision and hearing [80]. In particular, what we hear can be affected by what we see. For example, when the sound /ba-ba/ is played with the lip movements of /ga-ga/, the subject often perceives the sound /da-da/ instead. What we see also plays an important part during face-to-face communication between people, as it has been estimated that over 65% of communication is done non-verbally [85]. Body-langauge from facial expression, posture and body control often provide hints on how a person is feeling or what he is thinking.

Visual communication can also transcend time, as humans have developed ways to record and replay visual representation of ideas in the form of paintings, photographs and videos. The earliest examples of such recordings are cave paintings by our ancestors across various cultures. Rock paintings have been discovered over various cradles of civilization such as India, South Africa, Spain, France and Sweden. Figure 1.1 shows some examples of cave painting that are dated from 14000 BC to 550 BC. It seems that we have an innate need to document our experiences in some visual form to tell stories.

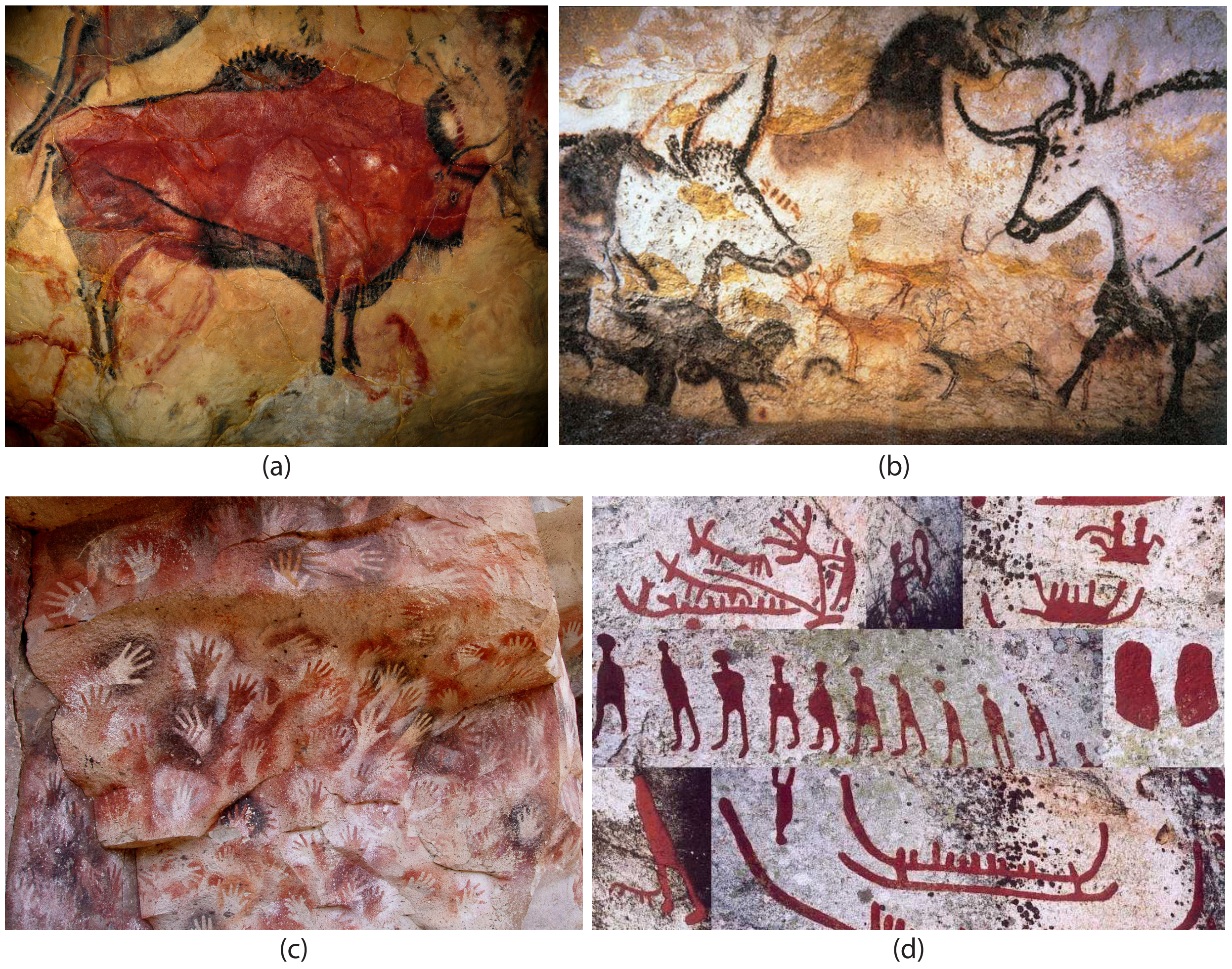


Figure 1.1: (a) Painting of a Bison at the Cave of Altamira, Spain. (b) Paintings of Aurochs, horses, and deer at Lascaux Caves, France. (c) Paintings of hands in Santa Cruz, Argentina. (d) Petroglyphs from Sweden. These caves paintings are found around the world dated between 14000 BC to 550 BC.

As civilization developed, paintings gained utility as propaganda for imperial and religious purposes. For example, Michelangelo was commissioned to paint the Sistine Chapel ceiling with frescoes depicting biblical stories which include the famous painting, 'The Creation of Adam'. Michelangelo also painted 'The Last Judgement' from 1535 to 1541 depicting historic events of the Holy Roman Empire as shown in Figure 1.2(a) on a wall in the Chapel. Eventually, paintings were adopted by the wealthy for portraits or as a form of art. These paintings help to shed light on the daily life of the wealthy in China and Rome as shown in Figure 1.2(c-d).

In contemporary art, visual pieces are used to evoke and elicit response from the viewer.

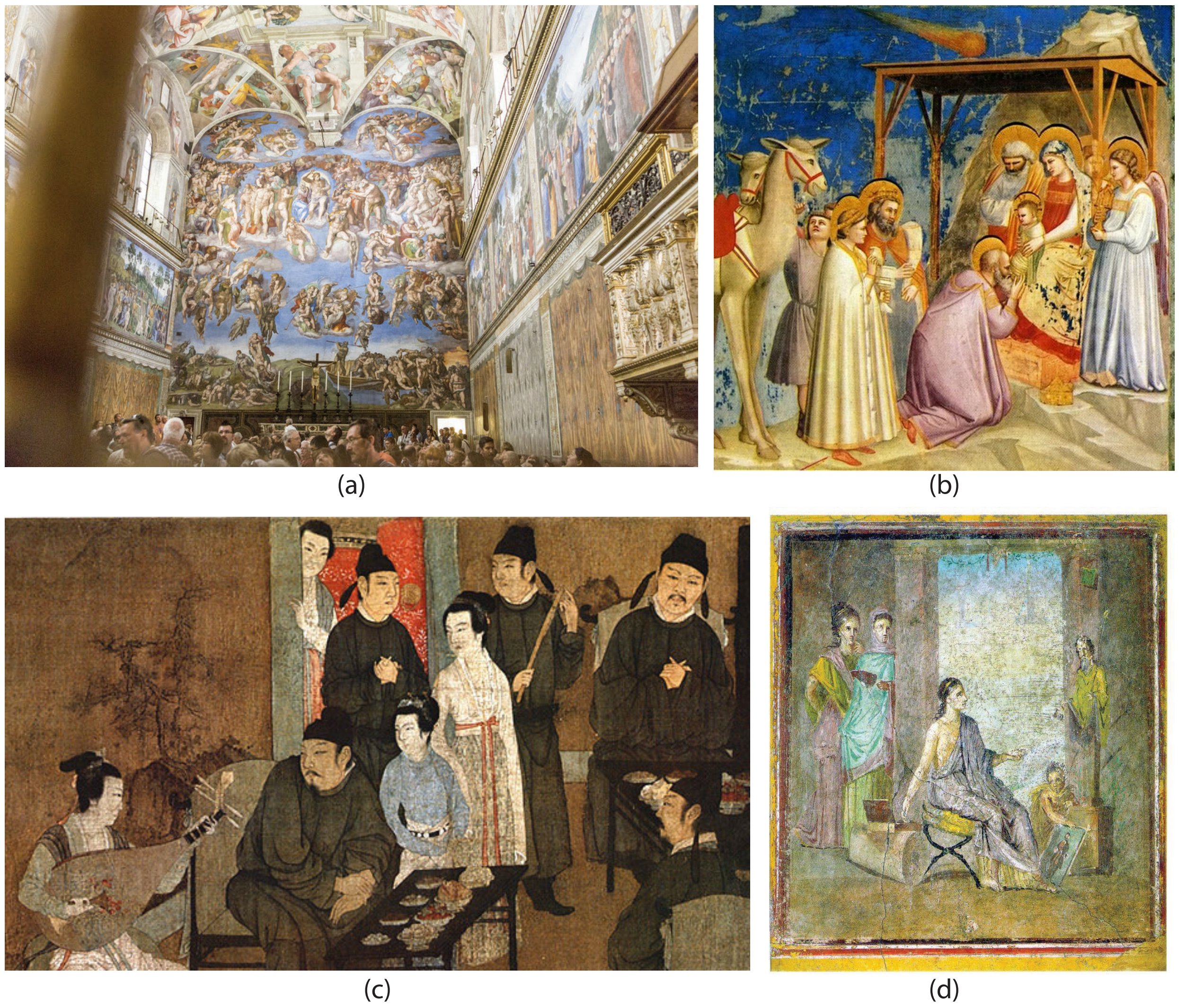


Figure 1.2: (a) Photograph taken inside the Sistine Chapel showing 'The Last Judgement' on the far wall and parts of the ceiling frescoes painted by Michelangelo. (b) 'Adoration of the Magi' by Giotto depicts the the Magi worshipping the newborn Jesus. (c) Painting of life in the Song Dynasty. (d) Roman art from Pompeii

Masters such as Picasso and Gleizes, produced works that are less narrative, but explore new visual paradigms such as cubism as shown in Figure 1.3(a-b). Modern pieces from Pollock and Albers further pushes the boundary of art as they explore new visual language such as lines, chaos and geometry without any connection to material objects in Figure 1.3(e-f).

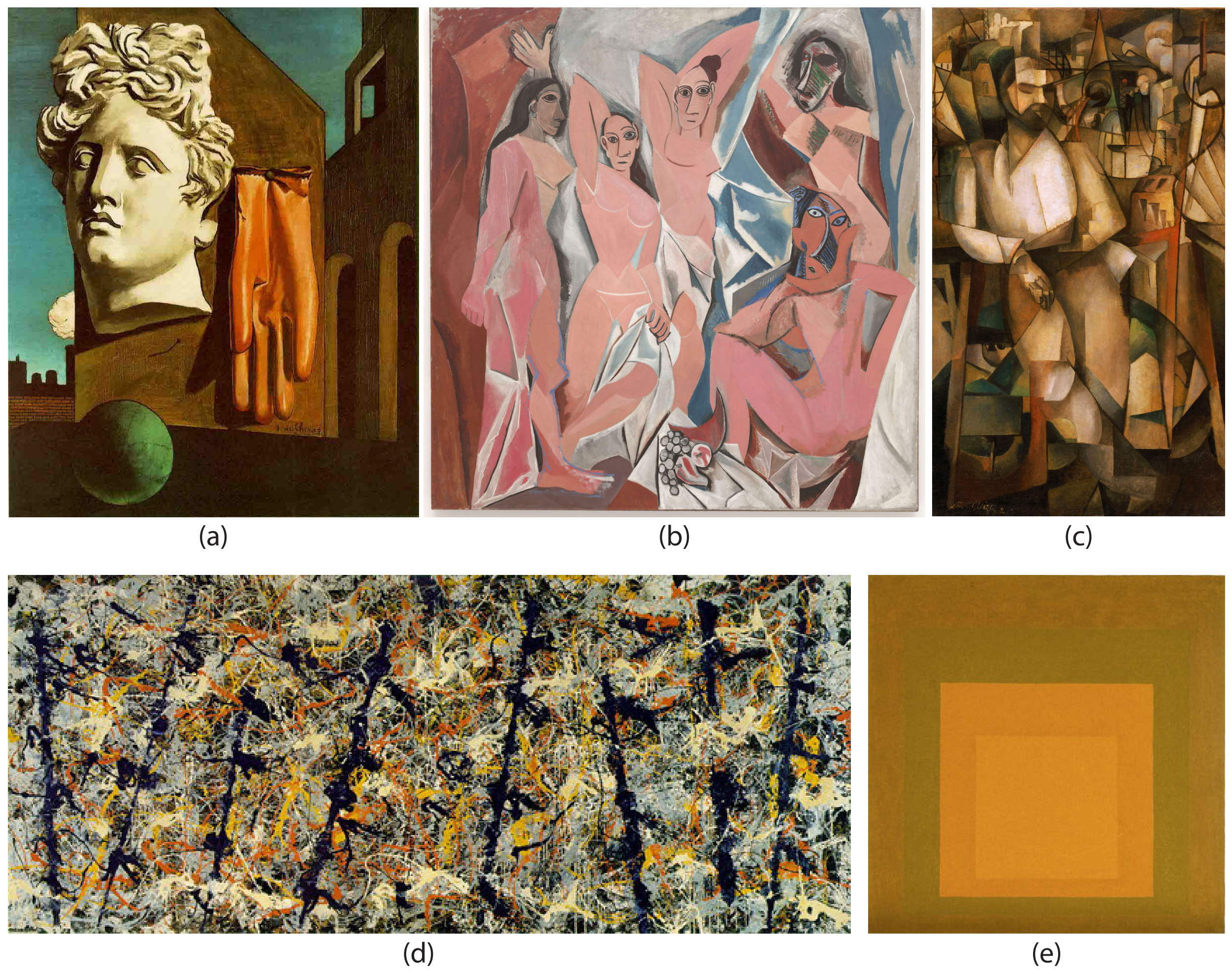
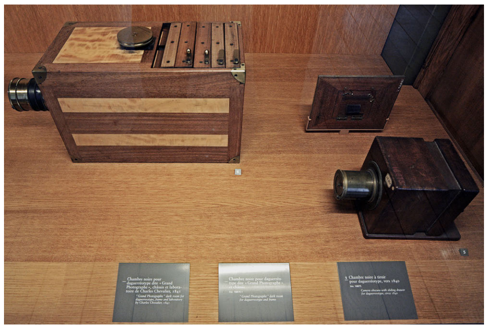


Figure 1.3: (a) 'De Chirico's Love Song' by Giorgio de Chirico. (b) 'Les Demoiselles d'Avignon' by Pablo Picasso. (c) 'L'Homme au Balcon' by Albert Gleizes. (d) 'Blue Poles' by Jackson Pollock. (e) Geometric abstraction by Josef Albers.

1.2 Photography

The introduction of modern photography was perhaps the greatest tool for telling visual stories. For the first time, an individual without any painting skills could create an accurate depiction of the world in a fraction of the time it takes to paint. The daguerreotype and the calotype were among the first few techniques which allowed users to record what the human eyes could see onto an imaging surface. Figure 1.4(a) shows a daguerreotype of the inventor, Louis Daguerre. The daguerreotype needed large view cameras, exposures which lasted seconds, and tedious chemical processes to produce an image as shown in Figure 1.4(b). In fact, each exposure of the daguerreotype produced a positive image that cannot be printed

(a)


(b)

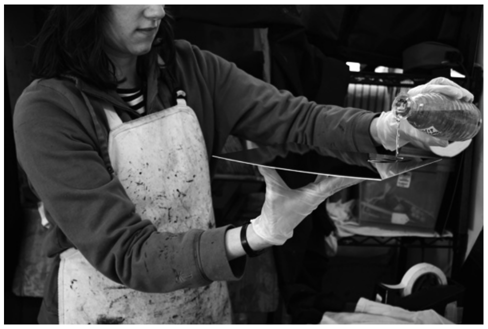

(c)

Figure 1.4: (a) A daguerreotype taken of Louis Daguerre. (b) Cameras used to make daguerrotypes. (c) A photographer coating a piece of glass with light sensitive elusion for the wet collodion process.

again. The wet collodion process was the first method to produce negative images. This new process replaced daguerreotype rapidly due to several advantages. First, it was cheaper as a glass plate was used instead of a polished silver plate. Second, exposure times were much shorter due to a more sensitive emulsion. Third, the development process was safer as mercury vapor was not needed unlike daguerreotypes. Finally, the wet collodion process allowed multiple positive photographs to be reproduced from a single capture. Figure 1.4(c) shows a photographer preparing a glass negative for the wet collodion process.

The development of the cellulose film and the Kodak Brownie in 1900s allowed common users to take photographs and have their images processed by Kodak. The Brownie popularized low-cost photography and allowed photographs to be taken by non-expert users, as all photographic processing was sent back to Kodak. The very first Brownie is shown in Figure 1.5(a). While cellulose film had eliminated the need for large glass plate negatives and photo development expertise, there were still limitations. First, the user had no idea how the photographs would look like until after it was developed and printed. Second, all photographs on the same roll have the same sensitivity and had to be processed in a batch with the same parameters.

Digital sensors slowly replaced cellulose film as the benefits of digital are immense. Figure 1.5(b-c) shows an example of 35mm negative film and a modern digital sensor. Digital sensors allowed instant preview of the image, near zero cost per image, the ability to switch ISO between shots, and amenability for miniaturization. As a result, almost all consumer electronic devices today from computers to phones have a digital imaging system. In fact, majority of the photos taken and shared by consumers are taken with mobile devices with cameras. With digital photography, new algorithms were needed in place of chemical processes to generate the final image.

Color is registered in 3 different layers (Figure 1.5(d)) in film. However, most digital sensors have a single layer and rely on a bayer mosaic to capture color as shown in Fig-

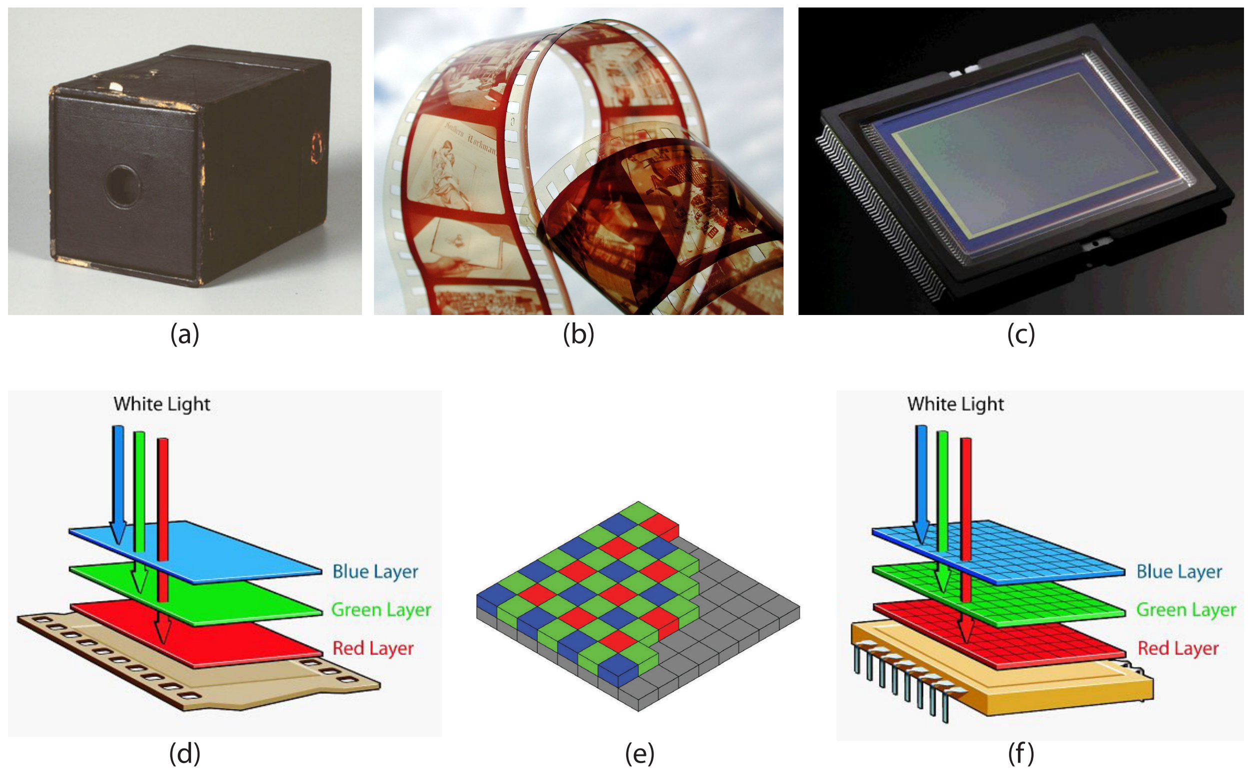


Figure 1.5: (a) The first Kodak Brownie camera allowed consumers to take photos without knowledge on how to develop and print photographs. (b) 35mm film allowed image negatives to be recorded and multiple positive images to be printed. (c) A modern digital sensor. (d) Film records color by using multiple layers of light sensitive pigments and filters. (e) A bayer filter is used on most digital sensors to record in color. The final color image is interpolated from the samples. (f) An alternative digital sensor where each pixel site senses in 3 colors by having layered sensors.

ure 1.5(e). However, since each pixel site in the bayer mosaic only captures one color, interpolation techniques are needed to reconstruct full color at all pixel locations. An alternative new sensor technology allows three layers of photosensors to be constructed at each pixel location such that each layer senses a different color (Figure 1.5(f)). This avoids the need for color interpolation which results in better color reproduction.

1.3 Video

The ability to accurately record what happens in the world allowed Eadweard Muybridge to create multiple photographs in quick succession to study the motion of a galloping horse. The goal of his experiment was to investigate if all four legs of a horse would leave the ground

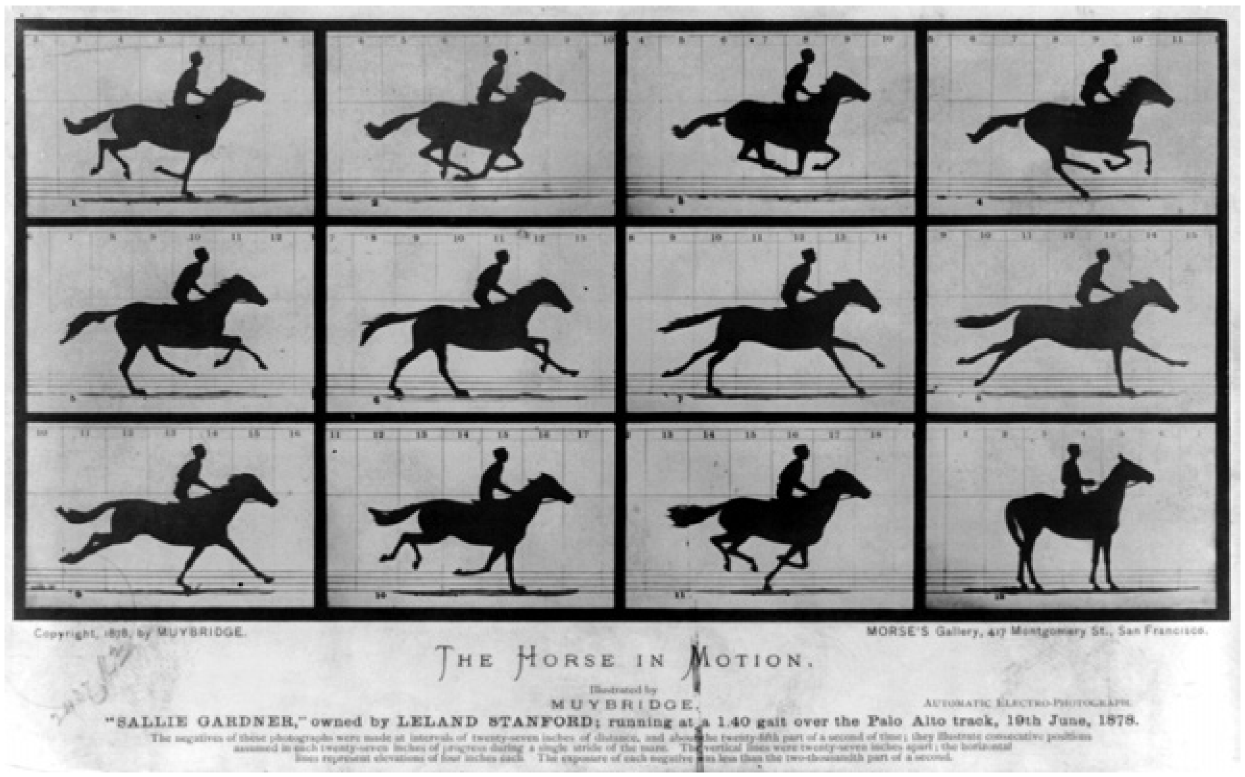

Copyright, 1878, by MUYBRIDGE.
MORSE'S Gallery, 417 Montgomery St., San Francisco.
THE HORSE IN MOTION.
Illustrated by
MUYBRIDGE.
AUTOMATIC ELECTRO-PHOTOGRAPH.
"SALLIE GARDNER," owned by LELAND STANFORD; running at a 1.40 gait over the Palo Alto track, 19th June, 1878.

Figure 1.6: Photographs from Eadweard's setup to capture sequences of a horse galloping. It is clear from the second top left inset that all four legs of the horse leave the ground while galloping.

while galloping. The results of his experiment is shown in Figure 1.6. It is clear that a horse will have all four legs off the ground while galloping. While Eadweard's experiment required multiple cameras, inventors like Louis Le Prince subsequently introduced film cameras which can record motion using a single lens camera.

Since then, digital video recording devices have become commonplace as cellphones, point and shoot cameras, DSLRs have the capability to shoot high quality videos. This along with video sharing platforms such as YouTube, Facebook and other social media sites have fueled the rate consumers are capturing and sharing videos. It has been reported that over 100 hours of video is was uploaded to YouTube every minute in 2013. In fact, online video now accounts for 50% of all mobile traffic and expects it to increase to 55% of all internet traffic by 2016. It is evident that video consumption is gaining momentum.

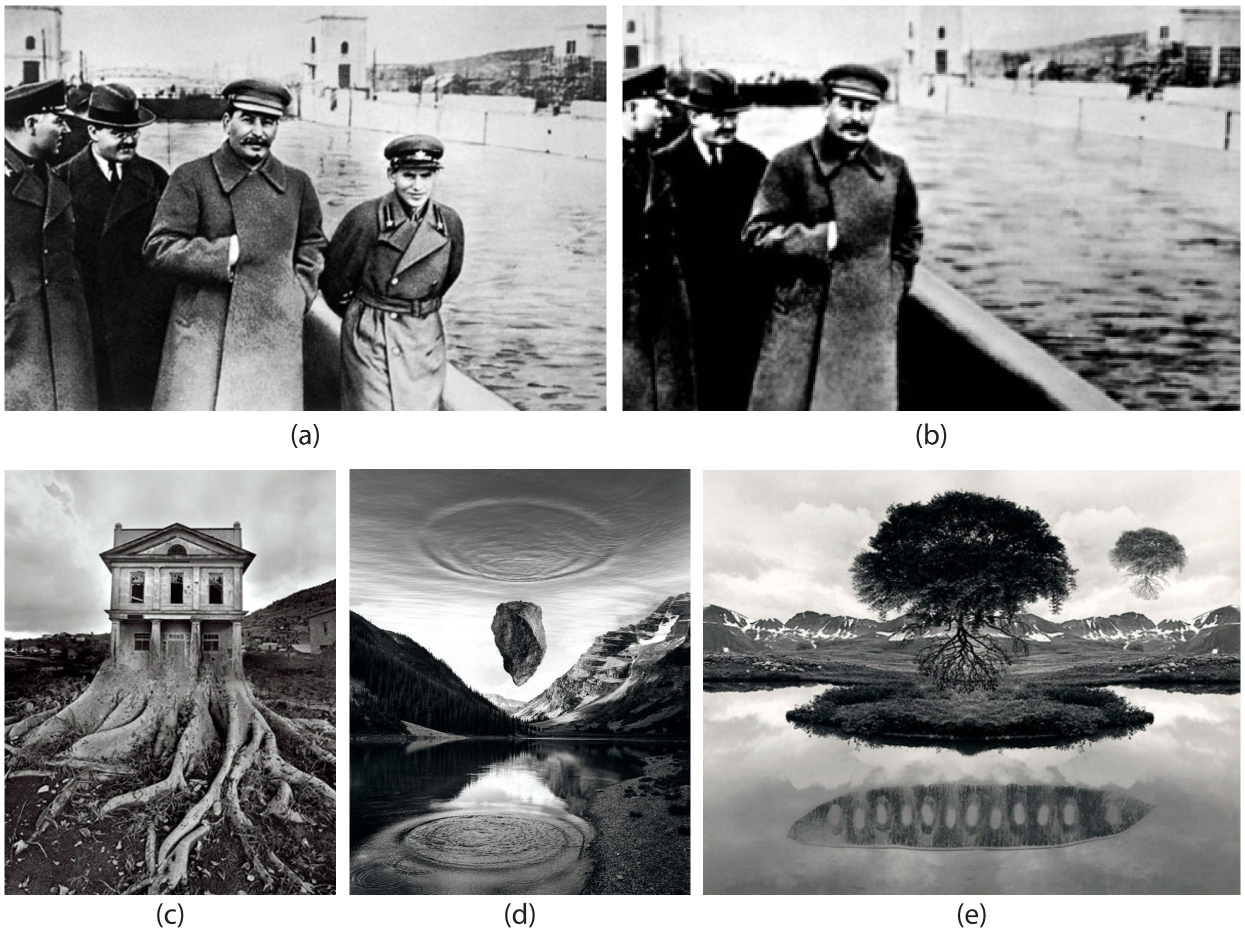


Figure 1.7: (a) Photograph of Stalin and Nikolai. (b) Notice that Nikolai has been removed from the picture after Stalin had purged him due to suspicions that he was disloyal. (c-e) Surreal photographs created by Jerry Uelsmann by printing multiple negatives onto the same paper without digital manipulation.

1.4 Image Editing

People have been interested in modifying images that they obtain from photography ever since the early 1900s even with film negatives. In the example shown in Figure 1.7, Stalin had removed Nikolai Yezhov from a previous photo after Nikolai fell out of his favor because he suspected Nikolai was being disloyal. Photography manipulation can also be used as a form of art. Jerry Uelsmann uses multiple negatives and composites them onto a single print to create surreal images as shown in Figure 1.7(c-e).

With digital photography, image editing and image manipulation have become easier as researchers develop new tools to synthesize new images from input images. Efros and Leung showed an alternative sampling technique by copying pixels can generate stochastic textures with greater effect compared to parametric methods [39]. This process can be accelerated by copying patches instead of pixels [38]. Areas with unwanted content could be removed and filled-in using a technique called in-painting [20, 35, 19, 34, 57] by copying appropriate image patches. PatchMatch is a random sampling algorithm to speed up approximate nearest neighbor search for patch similarity by an order of two [15]. With a large database of images, large regions of content can be replaced by searching for similar images and copying entire regions from it to generate a new novel image [53]. Image melding seeks to generate in-between images which bears properties from two different images by constructing new images with patches from both images [36]. An alpha matte allows the foreground object to be extract and composited onto a new background [29, 101, 66] for image compositing. Possion blending allows image composites to blend seamlessly as gradients are copied instead of pixel values [86, 105]. Instead of cropping or stretching an image to resize it, techniques such as seam carving, shift-map editing, and path transform allows images to be retargeted to a different size while maintaining scene content without distortion [9, 88, 28]. Multiple images are blended together to form a larger composite to create panoramas with up to 360 degrees field of view [104]. Multiple images can also be composited from different times with a moving object to convey the motion in a single photograph [102] Researchers have also used models as a proxy for image editing. By fitting 3D models of bodies, human body shape can be edited parametrically [58]. Human face can also be edited to change expression and attractiveness [70]. With a large collection of 3D objects, image editing becomes more powerful as novel views of objects, its lighting and motion can be generated [61].

1.5 Computational Photography

Computational photography has gained traction in recent years as researchers seek to leverage on increased computational power on devices to increase the capabilities of traditional photographic processes. Computation is added either during, and/or after image capture to augment photography.

For instance, by adding codes to the aperture, Levin et al. has shown that it is possible to recover depth with a natural statistics prior [68]. A Lattice-focal lens can also increase the depth-of-field of captured images by preserving high frequencies in the defocus image such that a sharp image can be recovered [67]. Coded masks can also be added to projectors to improve the depth-of-field of the projection [49]. Researchers have also explored coding in the temporal domain by using a shutter that flutters over time. This allows high frequency details to be captured in the blur image, therefore producing better deblurring results [91].

Multiple photographs captured at different times with varying parameters can be combined to produce better images that represent the scene. For example, High Dynamic Range imaging seeks to record the large variation of light intensities by combining multiple photographs captured at different exposures [37]. Images of a static scene can avoid high sensor noise and inaccurate color representation by combining information from a high ISO photograph and one that is taken with a flash [87]. Instead of using a visible flash, researchers have also used infra-red flash to reduce the visible burst of bright light that some subjects might find undesirable [64].

Researches have also explored ways to capture different modes of light transport. By projecting complimentary high frequency patterns onto a scene, researchers can separate direct illumination and global illumination effects [81]. In fact, once the light transport of the scene is known, each individual bounce of light transport can be separated for diffuse surfaces [10]. By using a very fast pulse of light, researchers can reconstruct and image the propagation of light through space. [110]. As a consequence, researchers demonstrate that they can image around occluders by considering the inter-reflections of light [111]. Subsequently, researchers have developed methods that uses inexpensive time-of-flight sensors or projector and camera systems to capture light moving through space as well as imaging around occluders [55, 83, 54].

Researchers have also modified low frame rate cameras to capture higher frame rates using compressive sensing and modulating each pixel independently [94]. High speed video can also be captured using low frame rate cameras by using an over-complete dictionary and per-pixel coded exposure [56].

Lightfield imaging is also an emerging field within computational photography. Instead of capturing the incident light rays falling on the sensor, light field imaging seeks to capture the incident angle of the rays as well. The additional angular information allows new applications such as image refocusing, perspective change, and depth reconstruction [69, 82, 106]. Computation is used to simulate a virtual focusing plane from the sampled light rays to produce a final image. Light field cameras can be constructed by using an large array of cameras [114], masks at the aperture [109], masks on the sensor [109], lenslets on the sensor [82], or masks external to the lens [93].

1.6 Contributions

The ideas which drive computational photography are the basis for the Moment Camera [32]. The new generation of cameras not only capture a single slice in time, but a moment which may span over a duration by using computation in addition to the imaging system. In general, computation is used to combine and/or select suitable samples of time and space to produce a media that best represents what the creator experienced during the moment.

The work in this thesis uses computation to remove large-scale motions in videos to augment the viewing experience. In particular, we present three complete systems that allow a user to remove unwanted motion from videos to enhance the viewing experience. The first, is Selectively De-animating Videos [12], where the user has fine control over where motions in the video should be retained and isolated. Second, we extend the system to operate automatically for a special class of videos; portraits [11]. Third, we present a video stabilization algorithm which leverages on user input to remove camera shakes from very challenging input videos [13].

In De-animation, the goal is to remove large-scale motions in the video so that finer-scale motions become more apparent. This allows the viewer to study finer-scale motions without the distraction of the large-scale motions. This technique can also be used to create Cinemagraphs. Cinemagraphs are a new type of visual media that is in-between a still photograph and a video. It is essentially a still image, with isolated regions with repeating motions. This helps to bring attention of the viewer to specific areas and it is surreal and more captivating than a traditional photograph, but not as distracting as a video. We propose a system that allows a user to removal large-scale motions in videos easily with only simple user input strokes.

Video stabilization also uses computation to remove large-scale motion. However, the goal in video stabilization is to compensate for unwanted camera motion in handheld videos. Handheld videos usually do not have a steady camera path which might result in a shaky video. Video stabilization seeks to remove motion in the video due to camera motion to mimic what a steady camera might record. However, traditional video stabilization algorithms do not leverage on any information other than tracking key points in the video. This might lead to undesirable results as objects in the result could be distorted. We propose allowing the user to provide information on how the result video should look like to further improve the results.

1.7 Overview

The rest of the thesis is organized as follows, after a brief overview of related work in Chapter 2, we will detail our project on Selectively De-animating Video in Chapter 3. The system requires user input and can be used on a wide variety of input videos. However, if we restrict the domain to portraits videos, we can design a fully automatic system which is presented in Chapter 4. In Chapter 5, we present a user assisted video stabilization algorithm which allows the user to improve video stabilization results by providing additional information.

Chapter 2

Related Work

2.1 Overview

The work in this thesis presents three video editing systems which are goal oriented and rely on user input or vision algorithms to guide the algorithm. In particular, these video editing systems allow a user to remove unwanted motion from videos to enhance the viewing experience. The core technologies that we leverage on are image warping and video stabilization to compensate for camera or object movement. We also use graph-cut compositing to find seamless transitions for spatial composites as well as temporal looping. Other key areas that we use include facial feature detection and motion clustering.

The first two systems, Selectively De-animating Video (Chapter 3) and Automatic Cinemagraph Portraits (Chapter 4), seeks to remove large scale motions to reveal smaller scale motion. We use similar approaches from Content-preserving warps [74] for Selectively De-animating Video and Automatic Cinemagraph Portraits to warp the frames to remove unwanted motion. The third system, User-assisted Video Stabilization (Chapter 5), seeks to incorporate content information and user goals into video stabilization. We use the state-of-the-art Bundled Paths [77] automatic video stabilization as a baseline and build upon their work to include the user in the loop.

We use graph-cut compositing to merge dynamic regions in Selectively De-animating Video and Automatic Cinemagraph Portraits with static stills. We use alpha-expansion [22] to solve for a cut in a Markov Random Field that minimizes our energy function. The energy function we use is inspired from Interactive Digital Photomontage [7] and seeks to find seams along regions that have a similar appearance or along object boundaries.

We are able to automatically generate cinemagraph portraits because we use a robust facial feature tracker to find key facial features points on the face in the video [96]. These points allow our algorithm to determine which parts of the face is moving and stabilize the video using areas of the face that are stationary throughout the video. We also then composite the moving parts of the face into a still image to generate the final result.

In User-assisted Video Stabilization, we modify existing track clustering techniques such that it is parallelizable and works for long sequences. We adopt a non-negative matrix factorization technique to find basis vectors over small windows to identify tracks with similar trajectories [26]. This reduces the work the user has to do when she refines the track selection for video stabilization. In the following subsections, we describe relevant approaches in these areas.

2.2 Video Editing

Researchers have developed many tools for specific video editing tasks. For example, rotoscoping tools such as Video SnapCut seeks to extract dynamic objects from a video sequence for compositing [14]. Editing interview footage can be challenging as a good audio cut might result in a unsatisfactory video cut as the subject might have awkward video jump cuts. Hidden transitions can be generated automatically to hide these transitions by finding and interpolating suitable frames [21]. As instructional videos are getting popular on the web, researchers have also developed systems to speed up the creation of such videos [27]. Tools for creating Video Digests for informational videos can greatly reduce the effort required to produce videos that allow browsing and skimming along with text summaries for video sections [84]. Our work introduces tools for motion editing in videos.

2.3 Video Stabilization

There are several video stabilization tools available today. For example, a user could stabilize the video in real-time using gyroscopic information from the camera [60]. She can also use tools such as Warp Stabilizer from Adobe during post-production [74, 75], or the video stabilization tool on YouTube after uploading the input video [50, 51].

Video stabilization works by first estimating scene and camera motion and then re-projecting each frame such that the unwanted camera motions are minimized. RANSAC [42] is used to automatically select tracks on background regions to stabilize the video. If the scene is largely planar, novel camera viewpoints can be generated using homographies [52, 46, 50]. For general scenes, a single homography is too restrictive and re-projections using a grid mesh have better success in stabilizing such videos.

Content-preserving warps [74] reconstruct the 3D scene and plan an optimal camera path. A grid mesh which preserves the structure of the scene content is used for re-projection. However, an alternative is to use 2D feature trajectories to guide the re-projection after applying a low-rank subspace constraint for camera smoothing [75]. Epipolar geometry can also be exploited to avoid reconstructing the 3D scene to stabilize the video [48]. Motion in-painting can also be included in video stabilization [79] and parallax can be handled explicitly [113]. Video stabilization techniques have also been developed for light fields [99] and for videos with depth information [100]. Bundles of homographies guiding cells in a grid mesh achieve state-of-the-art results by optimizing the path of each grid cell while enforcing spatial coherence [77].

2.4 Motion Visualization

One approach is to depict the motion using a static image [47, 25, 16, 33, 8, 6, 62, 102]. These methods use repetition, blur, spatial offsetting, montage and diagrammatic elements to represent motion, but because the result is a still, viewers must mentally reconstruct timing information. Another approach is to create a short video abstract that retains only the most salient motions. Truong and Venkatesh [108] survey a number of recent techniques for identifying such salient video segments. Pritch et al. [89] summarize surveillance video by non-chronologically rearranging spatio-temporal video segments representing salient activity. While these methods compress the length of a video, they do not emphasize important motions within the frame.

Motion magnification [73, 115, 112] highlights small spatial motions (e.g. breathing, deformations) by exaggerating them. Thus, its goal is the opposite of ours, as we focus on emphasizing the fine-scale motions of an object by removing larger-scale motions. In fact, motion magnification could be used in conjunction with our approach to further emphasize the most important motions. Bennett and McMillan [18] provide tools for creating time-lapse videos with stroboscopic motion trails to emphasize translational motions. Rubinstein et al. [95] remove transient, distracting motions from time-lapse sequences to increase the comprehensibility of longer-term movements. Our work is directly inspired by the vision of the Moment Camera [32] as we provide tools for isolating short motions that represent key moments in a scene.

Web-based instructions for creating simple cinemagraphs ask users to manually create a single alpha matte around the moving object and composite it with a still background frame.[1] Recently, Tompkin et al. [107] presented a more automated technique for creating such simple cinemagraphs. They focus on stabilizing camera motion in handheld video, masking moving objects and generating seamless loops. These approaches cannot de-animate large-scale motions of specific objects while leaving the finer-scale motions intact. Video textures [97, 65, 7, 77] are a well-known approach for seamlessly looping stochastic motions. Like cinemagraphs, a video texture lies between photography and video, but a video texture does not suppress distracting motions within the frame. Another approach to creating cinemagraphs is to start with a still image, manually segment it into objects, and then add motion using procedural motion templates [30]. In contrast, we focus on creating cinemagraphs from video by removing the motion of objects, instead of adding motion to an image.

There are now several publicly available tools for creating cinemagraphs that were developed concurrently with our work, such as Cliplets from Microsoft Research[2], and several

[1]http://fernandojbaez.com/cinemagraph-tutorial
[2]http://research.microsoft.com/en-us/um/redmond/projects/cliplets/

iphone apps[3].

2.5 Compositing

One way to composite video regions spatially and temporally is using graph-cuts [22, 65, 7]. The energy function when minimized provides a labeling for each pixel to copy pixel sources from to provide a visually seamless composite between sources. An alternative to graph-cut-based seamless compositing would be to extract a temporally-varying alpha matte for the user-selected foreground with video matting techniques [31, 14]. Poisson blending [101] can be used to further improve compositing seams by copying gradients and integrating. In our work, we composite both static (single-frame) and video regions; however, our energy function is significantly different, since we incorporate constraints from user-drawn strokes, and we can utilize additional information about the location of occluding boundaries.

2.6 Motion Clustering

Multi-frame motion clustering seeks to group tracks with similar behavior over frames. Techniques which are based on subspaces such as agglomerative lossy clustering [90] or sparse subspace clustering [40] typically require some tracks which span the length of the video sequence. Since hand held videos typically do not have long track trajectories due to occlusions or large motions, these techniques are not robust enough for motion clustering.

Another approach for motion clustering is affinity-based, which measures the pairwise relationship between track trajectories. Affinities based on spatial distance, and similarity of translational motion of track trajectories can be used as features for clustering [24]. Affine motion similarity can also be used as a basis for affinities for clustering [44]. Weights from non-negative matrix factorization of the trajectory speed and direction give reliable results for partial track data [26]. However, hand held videos have many short tracks and clustering can be computationally intensive. We instead use a moving window and non-negative matrix factorization of the track trajectory's speed and direction to build an affinity matrix for clustering, which greatly decreases the computation time.

[3]http://kinotopic.com, http://cinemagr.am, http://www.icinegraph.com

Chapter 3

Selectively De-animating Videos

3.1 Introduction

In this chapter, we present a semi-automated technique for selectively *de-animating* or removing the large-scale motions of one or more objects.

The large-scale motion of an object can sometimes make it difficult to see its finer-scale, internal motions, or those of nearby objects. Consider a video of a guitar teacher demonstrating the finger motions required to play a song. The fine motions of the fingers on the strings and frets are obscured or visually masked by the larger-scale, gross motions of the guitar body. The finger positions required to play notes and chords would be much easier to follow if the guitar were immobilized (Figure 3.1 and video on project webpage).

The user draws a small set of strokes indicating the regions of the objects that should be immobilized and our algorithm warps the video to remove the gross motion of these regions while leaving finer-scale, relative motions intact. The warp may, however, propagate unwanted motions to other regions that should remain still, such as the background. To eliminate such undesirable motion, we develop a method for compositing the warped video with still regions taken from frames of the input. The resulting videos highlight fine-scale, internal motions of one or more objects, as well as relative motions of nearby objects. In the guitar example (Figure 3.1) we have immobilized the guitar and composited still frames of the head and body to make it easier for viewers to focus on the finger positions.

Our algorithm facilitates the creation of *cinemagraphs* [1] which have recently become a popular form of motion visualization. They are designed to draw the viewer's attention to specific objects and motions in a scene. Cinemagraphs lie between the traditional media of video and photographs; the most important objects remain animated while the surrounding objects are held still. While some cinemagraphs simply composite a moving object onto a still frame background, the most effective also eliminate distracting, large-scale motions of the primary objects so that viewers can focus on the subtle, fine-scale motions. Cinemagraphs are also designed to loop endlessly like a video texture [97]. Therefore, users can optionally use our compositing algorithm to also compute a seamless video loop. Well-designed cin-

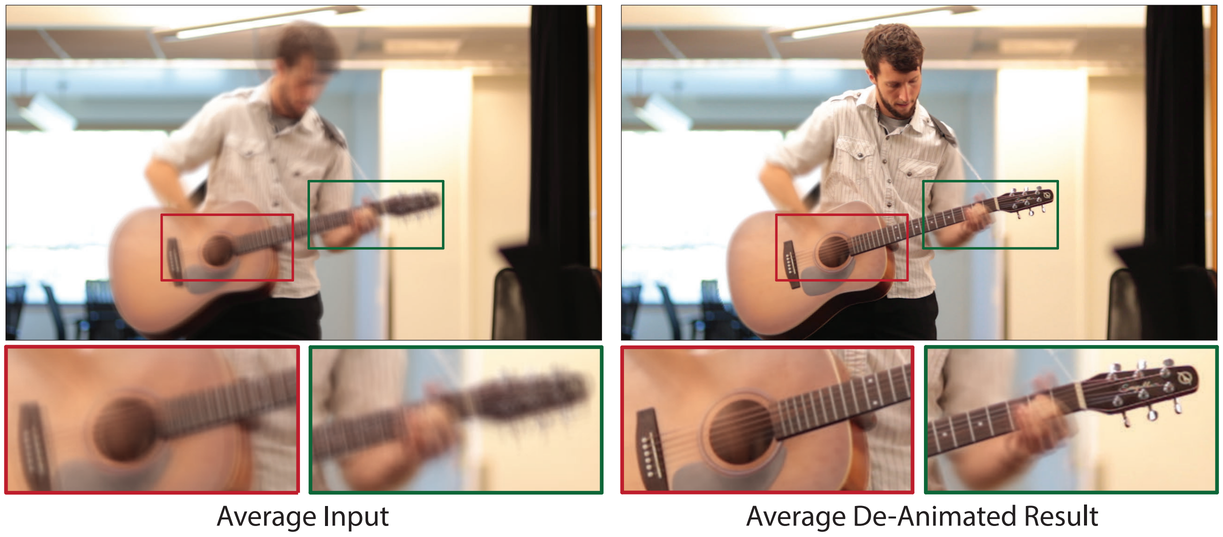


Figure 3.1: Large-scale motions of the guitar body can make it difficult to follow the finer-scale motions of the strings and fingers. We visualize the amount of movement by averaging the frames of the input video (left) and find that the body and fretboard of the guitar, as well as the strings and fingers are blurred because they move a lot. With our selective de-animation technique, we remove the large-scale motions of the guitar to make it easier to see the finer scale motions. Averaging the frames of our de-animated result (right) shows that the body and fretboard are sharp and therefore immobilized. Note that while the strings and fingers are sharper than in the visualization of the input video, they remain blurry because their fine-scale motions are retained in our de-animated result. We encourage the reader to view the video, to see this comparison in video form.

emagraphs can be surreal and are often unexpectedly engaging. They are now widely seen in major online newspapers, magazines, advertising and retail websites. Our cinemagraph examples can be seen in Figure 3.6.

Beyond motion visualization and cinemagraphs, we also demonstrate applications of our de-animation technique to video editing. First, we show how de-animation facilitates appearance editing of moving objects in video. The user edits an object after it is immobilized in a single frame of the de-animated video, and our system propagates these edits back to the original video (Figure 3.7). Second, we show that it is easier to animate a virtual object that moves within the frame-of-reference of a larger object. After the animator plans the motion of the virtual object in the de-animated video, our system adds in the original motions of the larger, frame-of-reference object (Figure 3.9).

Our main technical contribution is the first semi-automated method for selectively de-animating video to eliminate the gross motions of a specific object or objects. The two main steps of our algorithm follow the general approach of previous work, namely content-preserving warps for video stabilization [74], and graph-cut video compositing [65]. Novel to

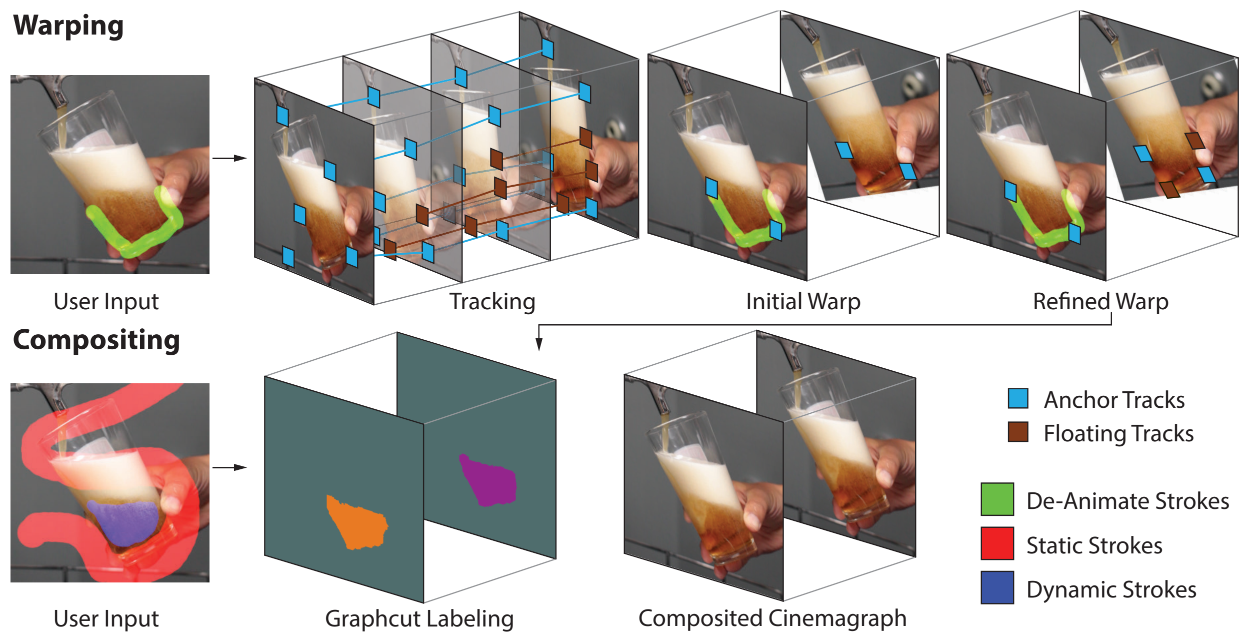


Figure 3.2: A diagram of our overall system. The algorithm proceeds in two stages, Warping (Section 3.4) and Compositing (Section 3.5), and utilizes three types of user-drawn strokes on a single reference frame (left).

our problem domain is the challenge of only stabilizing locally-defined regions which have been identified by the user *in a single frame*. Therefore, our energy function for warping is more complicated, and requires a multi-stage solution method. We also modify the energy functions used for compositing from those found in previous work to achieve good results for our problem domain.

3.2 User Input

We designed our de-animation method to give users creative control over the output while minimizing repetitive effort. Users must specify three types of information using a coarse set of strokes on a single frame of the input video; (1) which regions of the video should be de-animated so that their large-scale motions are eliminated (Figure 3.2, green strokes), (2) which regions of the video should be held static (red strokes) and (3) which should remain dynamic (blue strokes). Note that we use the terms *de-animated* and *static* to refer to two different type of output regions in the output video. De-animated regions have been warped to remove large-scale motions, and may be either static (e.g., completely still) or dynamic (e.g., moving) in the final output. For example, in Figure 3.1, the large-scale motions of the guitar are de-animated, *but* its fine-scale internal motions, such as the motions of its strings, remain dynamic. The teacher's arms should also remain fully dynamic and the large-scale motions are *not* removed. Other regions, like the head, are completely static in the final video.

To make the task easier for our algorithm, we ask the user to draw green de-animate strokes on only the most static regions of the object, avoiding any internal motions that should be preserved (Figures 3.2, 3.5 and 3.6). Blue strokes indicate output regions that should remain dynamic, while red strokes indicate those that should be completely static. Finally, the user can specify whether the final output should loop seamlessly, and a minimum duration for the loop. Such looping is commonly used to create cinemagraphs.

3.3 Overview

As shown in Figure 3.2, our algorithm has two main stages; warping (Section 3.4) and compositing (Section 3.5). The warping stage first tracks feature points through the input video. It uses a subset of the tracks to compute an initial warp and then refines the warp with additional tracks. The compositing stage combines the warped video with static frames of the unwarped video to eliminate undesirable motions in regions that the user specifies should remain still (red strokes) while preserving motions in regions that should remain dynamic (blue strokes). The result is a de-animated video, in which the large-scale motion of one or more objects have been removed, and parts of the video are completely static.

Our approach relies on a few assumptions about the input video. First, we avoid the issue of camera stabilization by assuming the input was captured with a tripod, or previously stabilized. Second,we assume the large-scale motions can be de-animated with 2D warps of the video frames and that complex 3D analysis to reverse self-occlusions is not necessary. Third, we assume the objects to be de-animated are shot in front of a defocused, uniform, or uniformly-textured background. As we will show, this assumption allows our algorithms to de-animate objects without building accurate alpha mattes for them. Despite these assumptions we have found that we can apply our technique to a large class of videos.

3.4 Warping

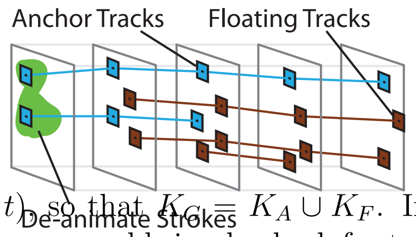
Our algorithm begins by tracking a set of points throughout the sequence. We use Kanade-Lucas-Tomasi (KLT) tracking [78, 98] to follow distinctive points in the input video $I(x, y, t)$. Tracks may begin or end at any frame, and most tracks do not last throughout the entire video. We define the set of tracks as a table of 2D coordinates $K(s, t)$, where s is a unique index for each track, and t indicates time (frame number); $K(s, t) = \emptyset$ if track s is not present on frame t. Since very short tracks are often unreliable, we remove tracks whose duration is less than 15% of the input video duration.

The user draws green strokes on reference frame t_a of the input video. The goal of our warping algorithm is to warp the video so that the content indicated by the green de-animate strokes remains spatially fixed to that location for all frames of the output video. We define

$K'(s,t)$ as the location of the tracks after warping, and $K_G(s,t)$ as the subset of the tracks that lie on the indicated content. Similarly, $K'_G(s,t)$ are the locations of these tracks after warping. The goal of our warp can be expressed mathematically as,

$$K'_G(s,t) = K_G(s,t_a). \tag{3.1}$$

How do we identify the tracks K_G? For tracks that exist on the reference frame t_a, $(K(s,t_a) \neq \emptyset)$, we simply check if they lie within the green strokes. We call these *anchor tracks*, and collect them into the subset $K_A(s,t)$ (see inset). The task is more complicated for tracks that do not exist on frame t_a, but still lie on the content marked in green. We call these *floating tracks*, or $K_F(s,t)$, so that $K_G = K_A \cup K_F$. If we knew how to warp the video to immobilize this content then we could simply check for tracks that fall in the same green strokes across the entire video to identify floating tracks.



Since we don't know the warp a priori, our approach is to compute two consecutive warps. The initial, approximate warp uses only the anchor tracks, i.e., the constraint $K'_A(s,t) = K_A(s,t_a)$. If enough anchor tracks exist throughout the entire input video, this initial warp can successfully immobilize the content indicated by green strokes. It is more common, however, for the anchor tracks to last a shorter duration. We therefore use the initial warp to identify additional floating tracks (e.g. the set of tracks that lie in the green strokes after the initial warp) and then compute a more accurate, refined warp using both sets of tracks.

Initial Warp

Given just the anchor tracks $K_A(s,t)$, our goal is to warp the video so that $K'_A(s,t) = K_A(s,t_a)$. One approach is to simply fit a full-frame warp like a homography to each frame that minimizes the squared sum of errors of this constraint. This approach can work well for some scenes, but others, which are less rigid and planar, require a more flexible representation (Figure 3.10). We therefore use a spatially-varying warp defined on a rectilinear mesh placed on each video frame, similar to the approach of Liu et al. [74]. However, in their case the main warping constraint is given by the output of a 3D reconstruction, while in our case, the goal is to warp each output frame so that the anchors align with the reference frame.

More specifically, we create a 64×32 rectilinear grid mesh on each input frame, with vertices $V(i,j,t)$ and use it to compute an output mesh $V'(i,j,t)$. We can express each track $K(s,t)$ as a bilinear combination of the four vertices of the quad enclosing the track on frame t. We define $\mathbf{V}(s,t)$ as the vector of these four vertices; $\mathbf{V}'(s,t)$ represents the same four vertices on the output grid. The vector $\mathbf{w}(s,t)$ contains the four bilinear interpolation weights that sum to 1, so that $K(s,t) = \mathbf{w}(s,t) \cdot \mathbf{V}(s,t)$ on the input mesh, and $K'(s,t) = \mathbf{w}(s,t) \cdot \mathbf{V}'(s,t)$ on the output mesh we are solving for. The main constraint (Eqn. 3.1) of the initial warp then becomes

$$E_a = \sum_{s \in K_A, t} l(s,t) |K_A(s,t_a) - \mathbf{w}(s,t) \cdot \mathbf{V}'(s,t)|^2, \tag{3.2}$$

where the unknowns are the output mesh vertices contained in $\mathbf{V}'$. We use the weighting function $l(s,t)$ of Liu et al. [74], to temporally fade-in and fade-out the strength of the constraints from each track to preserve temporal coherence. (The weights vary from 0 to 1 with a linear ramp window of 15 frames for fading-in and fading-out when the track points appear and disappear).

We include an additional term E_s on the warp to reduce visible distortion by encouraging each quad to undergo a similarity transform. We use the formulation given by Liu et al. [74]. We compute the shape term used in Eqn. 3.5 by first splitting the input quad mesh into a set of triangles. Since each quad in the input mesh is a square, each triangle is an isosceles right triangle. Considering a single triangle with vertices (V_1, V_2, V_3), if V_2 is opposite the hypotenuse, we can construct vertex V_1 from the other two vertices,

$$V_1 = V_2 + R_{90}(V_3 - V_2), \quad R_{90} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}. \tag{3.3}$$

If this triangle undergoes a similarity transform, its vertices after the transform will still follow this equation, and we can measure the deviation of a transformed triangle from a similarity transform by the error in this equation. We can therefore write

$$E_s = \frac{1}{8} \sum_{triangles} ||V_1' - (V_2' + R_{90}(V_3' - V_2'))||^2, \tag{3.4}$$

where the sum ranges over each of the eight triangles containing each vertex. Using all eight triangles is redundant, but avoids special handling at the mesh boundaries.

The final energy function is

$$E = E_a + \omega E_s. \tag{3.5}$$

We set the weight $\omega = 4$ through trial-and-error. Since the energy function is a sum of quadratic terms, we minimize it with respect to V' using least squares to solve for the output grid mesh. Finally, we render the output sequence by texture mapping the input frames onto the output grid mesh.

Refined Warp

To further improve the result we use floating tracks. We first eliminate floating tracks that fall outside the green strokes. We use the output of the initial warp, which approximately aligns each video frame with the reference frame t_a, and include a warped floating track $K_F'(s,t)$ only if it falls within the green strokes (since the track may vary with time, we require it to lie within the strokes for at least 75% of its duration).

We cannot use the constraint in Eqn. 3.2 for floating tracks, since we do not know their locations $K_F(s,t_a)$ in the reference frame. Instead, we constrain the floating tracks to remain fixed in space over time

$$E_f = \sum_{s \in K_F, t} l(s,t) |\mathbf{w}(s,t) \cdot \mathbf{V}'(s,t) - \mathbf{w}(s,t{+}1) \cdot \mathbf{V}'(s,t{+}1)|^2. \tag{3.6}$$

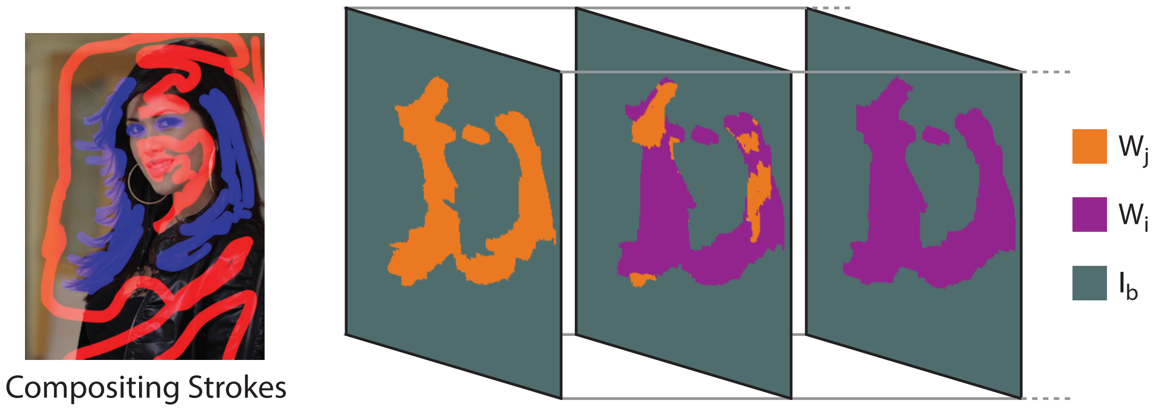


Figure 3.3: The computed labeling for three frames of a cinemagraph. In this example, graph cut chooses a single still frame label (gray) for the background and static regions of the model, and two dynamic candidates (purple and orange) for the hair and eyes.

Since this constraint includes unknown mesh vertices at different moments of time, we cannot solve for each frame's output mesh separately, as in Eqn. 3.5 and in Liu et al. [74]. We therefore solve for the entire output video mesh simultaneously using least squares. The final energy function we minimize to compute the refined warp is

$$E = E_a + E_f + \omega E_s. \tag{3.7}$$

3.5 Compositing

The final step is to composite the warped video with static regions taken from a few frames of the input video, in order to remove any extraneous motions outside the immobilized video regions.

We use graph cuts to perform Markov Random Field (MRF) optimization, to seamlessly composite dynamic regions from the warped video with static regions from input frames into a single output video. This stage of the algorithm takes as input the original video $I(x, y, t)$ and the warped video $W(x, y, t)$, and produces a final output video $O(x, y, t)$.

Each pixel p of the output video volume $O(x, y, t)$ is assigned a discrete label $\lambda(p)$ that respects a set of labeling constraints and minimizes an energy function. Each candidate label λ corresponds to a pixel source, which we refer to as a *candidate video volume*. An example of a candidate video volume is the warped video. The labeling constraints apply to pixels that are indicated by user-drawn red and blue strokes. The energy function measures the degree to which a transition between candidate videos is visually noticeable. The resulting labeling therefore produces an output video that respects the user-drawn strokes, and consists of regions of the candidate videos with seamless, spatio-temporal transitions between them (Figure 3.3).

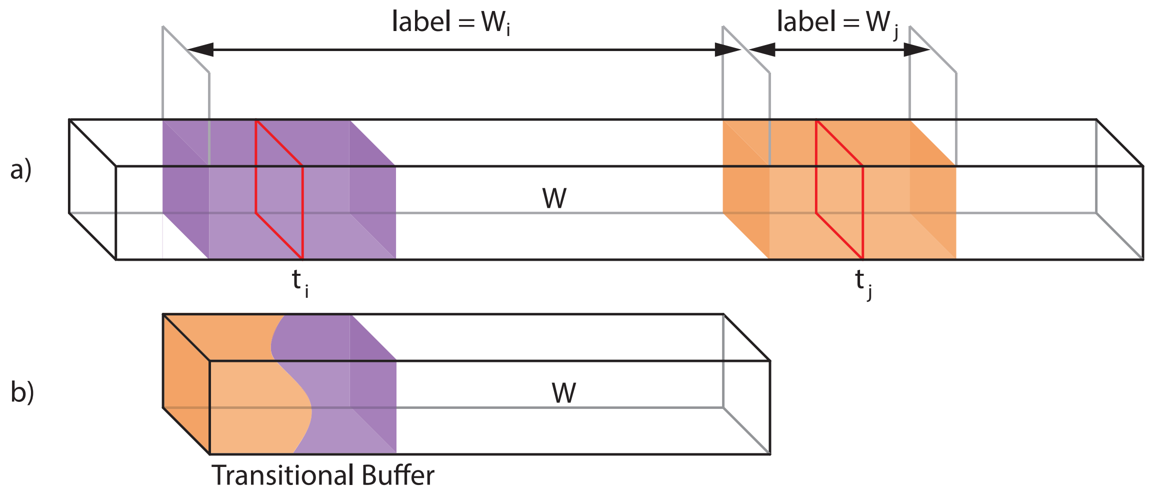


Figure 3.4: We select two candidate video volumes, W_i and W_j **(a)**, and overlap them in the output to form a seamless loop **(b)**. Transition regions (purple and orange) surround frames t_i and t_j (red), and W_j is time-shifted to overlap with W_i in the output, so the first and last frames of the loop come from consecutive input frames.

If the user specifies that the output should loop seamlessly, we first follow the approach of Schodl et al. [97] and find a pair of starting and ending frames that are visually similar in the warped video. Specifically, we search for a pair of frames (t_i, t_j) in $W(x, y, t)$ that are at least as far apart as a minimum duration specified by the user, and that minimize the sum of RGB distances between their pixels. We only compare pixels within the blue strokes, since these regions will remain dynamic in the final composite. We then reduce the input video volume to this duration, and set the output duration to the same number of frames.

Candidate Video Volumes

We define a set of labels **L**, that represents two types of candidate video volumes: *dynamic candidates* that are copies of the warped video $W(x, y, t)$, and *static candidates* that are still-frames from the input video repeated to fill the duration of the output. The first type allows us to place dynamic, warped video into the parts of the output constrained by blue strokes, while the second type allows us to select static regions for parts of the output constrained by red strokes. We denote the set of dynamic candidates as **W** and the set of static candidates as **S**, so that $\mathbf{L} = \mathbf{W} \cup \mathbf{S}$.

We first add the warped video to the dynamic label set as W_i. If the output should loop seamlessly, we add another copy of the warped video, W_j, to the labels **W** as shown in Figure 3.4. We create a transition region of time (21 frames) surrounding frames t_i and t_j, and the two labels indicate videos beginning at the start of those transition regions. We time-shift W_j to overlap it with W_i in the output, so that the first and last frames of the output loop come from consecutive input frames. The spatio-temporal transition between W_i and W_j within the transitional buffer will be the best seam for looping from frame t_j to frame t_i.

We define the static candidate label and video volume $I_k(x, y, t)$ as a video where the k'th frame of the input video is repeated for the duration of the output. Using all the frames of the input video as candidates is cost-prohibitive, so we evenly sample to select five frames. If b is the time interval that evenly samples the input five times, we define the set of static candidates $\mathbf{S} = \{I_b, I_{2b}, \ldots, I_{5b}\}$. Finally, the user can optionally choose to include a "clean plate" still frame candidate containing just the background of the scene.

Labeling Constraints

We create a final composited result by choosing a label $\lambda(x, y, t)$ from the set $\mathbf{L}$ for each pixel. We place four constraints on this labeling. The first two are based on the user-drawn compositing strokes. We represent the strokes as $\upsilon(x, y) \in \{\text{red}, \text{blue}, \emptyset\}$. Blue-stroked pixels must come from dynamic candidates, while red-stroked pixels must come from static candidates.

1. If $\upsilon(x, y) = \text{blue}$, $\lambda(x, y, t) \in \mathbf{W}$
2. If $\upsilon(x, y) = \text{red}$, $\lambda(x, y, t) \in \mathbf{S}$

The second pair of constraints limit the durations of the videos W_i,W_j, and are only used if the output must seamlessly loop. In order to achieve the temporal transition shown in Figure 3.4b, the first and last frames of the overlapped transition regions must come from W_j and W_i, respectively, except for pixels that are static.

3. $\lambda(x, y, 0) \neq W_i$.
4. $\lambda(x, y, 20) \neq W_j$.

Energy Function

In order to generate a visually seamless composite video we minimize an energy function that measures the visual noticeability of transitions between candidate video volumes. Like Kwatra et al. [65], we prefer two kinds of seams; those for which the RGB values are similar across the seam and those that lie along strong edges. However, unique to our application is the knowledge that the dynamic candidates typically contain foreground objects, while the static candidates typically contain background. Transitions between foreground and background along foreground boundary edges are desirable, since they correspond to occlusion boundaries. Therefore, we only use the edge-strength in the dynamic video for seams between dynamic and static candidates.

We define (p_1, p_2) as two neighboring pixels in the output (we use a 6-neighbor system across space and time), and $C(p, \lambda(p))$ as the color of the pixel p in candidate video volume $\lambda(p)$. We use λ_1 and λ_2 as shorthand for $\lambda(p_1)$ and $\lambda(p_2)$, respectively. If the labels of the

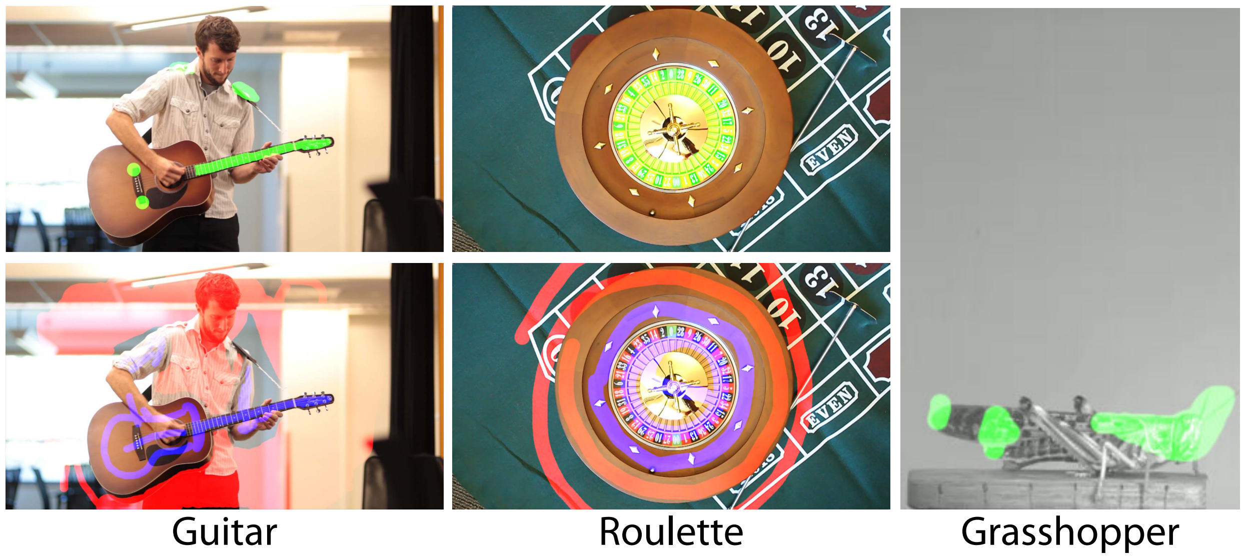


Figure 3.5: User provided strokes for our motion visualization examples. Green strokes are for de-animation, while red and blue strokes are for compositing. Note that minimal user input and coarse strokes suffice for our method.

neighboring pixels are equal ($\lambda_1 = \lambda_2$), the seam cost is zero since no seam exists. Otherwise, the seam cost Φ is

$$\Phi(p_1, p_2, \lambda_1, \lambda_2) = \frac{\gamma(p_1, p_2, \lambda_1, \lambda_2)}{Z(p_1, p_2, \lambda_1, \lambda_2)} \tag{3.8}$$

$$\begin{aligned}\gamma(p_1, p_2, \lambda_1, \lambda_2) = {} & |C(p_1, \lambda_1) - C(p_1, \lambda_2)|^2 \\ & + |C(p_2, \lambda_1) - C(p_2, \lambda_2)|^2\end{aligned} \tag{3.9}$$

As in Kwatra et al., the seam cost measures RGB differences across the seam ($\gamma(\cdot)$) divided by edge strength ($Z(\cdot)$) across a seam. For our application, we choose to average the edge strength between two static candidates and between two dynamic candidates, and only use edge strength in dynamic candidates for seams between dynamic and static candidates

$$\begin{aligned}& Z(p_1, p_2, \lambda_1, \lambda_2) = \\ & \begin{cases} \sigma(p_1, p_2, \lambda_1) & \lambda_1 \in \mathbf{W} \wedge \lambda_2 \in \mathbf{S} \\ \sigma(p_1, p_2, \lambda_2) & \lambda_1 \in \mathbf{S} \ \wedge \lambda_2 \in \mathbf{W} \\ \frac{1}{2}[\sigma(p_1, p_2, \lambda_1) + \sigma(p_1, p_2, \lambda_2)] & \text{Otherwise} \end{cases}\end{aligned} \tag{3.10}$$

where edge strength within a specific video candidate λ is represented by $\sigma(p_1, p_2, \lambda)$, and computed with a 3×3 Sobel filter averaged across RGB.

In total, we seek a labeling that minimizes

$$\sum_{p1,p2} \Phi(p_1, p_2, \lambda_1, \lambda_2) \tag{3.11}$$

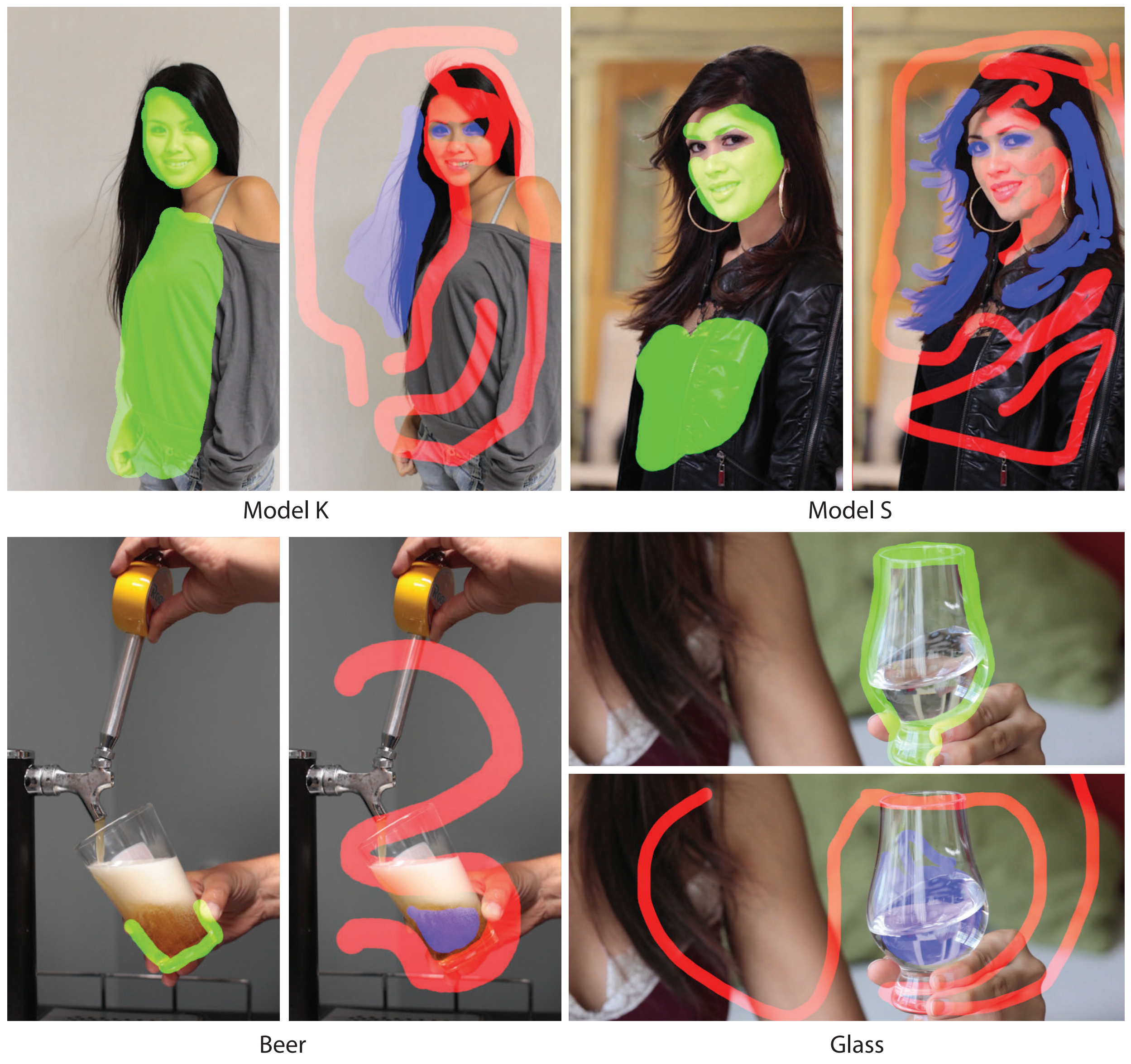


Figure 3.6: User provided strokes for our cinemagraph examples. Green strokes are for de-animation, while red and blue strokes are for compositing. Note that minimal user input and coarse strokes suffice for our method.

for each pair of neighboring pixels (p_1, p_2), subject to our stated constraints. We minimize this energy function using the alpha expansion algorithm [22]. Once a labeling is computed, we create a final video simply by copying pixels from the candidate videos. An example labeling is shown in Figure 3.3.

When the length and resolution of the output is too large to compute within reasonable time and memory, we downsample the input so that there are less than 5 million variables in the MRF optimization. If the downsampling factor is less than 8, the full-resolution labeling is created by bicubic interpolation of the low-resolution labeling, which creates feathered transitions between the regions. If the factor is larger, we use hierarchical graph-cuts [7] at a 4x downsampled resolution before performing a final upsampling with interpolation.

3.6 Results

We captured a variety of scenes to evaluate and demonstrate our algorithm. Input sequences range from 3 seconds to 12 seconds. We down-sample the frames if necessary such that the height of the video is 720 pixels. We show the input strokes used to create some of our examples in Figures 3.5 and 3.6; these strokes are the only user effort required by our system. All examples are shown in the accompanying video. For each, we also provide a video that compares our approach to alternative de-animation methods on our project webpage.

Motion Visualization

Our first four results reveal motion that is otherwise difficult to see (Figure 3.5).

Guitar: We de-animate both the guitar and the shoulder of the musician to clarify the finer-scale motions of his fingers. Our spatially varying warp is able to handle multiple opposing constraints for immobilizing and preserving the shape of both the guitar and shoulder (Figure 3.1). For this example, we include a clean plate image of the scene as a static candidate during compositing.

Roulette: In roulette, a ball is rolled clockwise along the outer rim while an inner wheel is rotating counter-clockwise. The complicated motions of the ball as it hits the inner wheel makes it difficult to see its path. We de-animate the rotating inner wheel to better reveal the path and physics of the ball and possibly allow the viewer to predict the outcome of the games.

Grasshopper: The finer-scale leg motions of a jumping grasshopper can be difficult to see because of the larger-scale motions of its body [103]. While entomologists sometimes glue insects to sticks in order to eliminate the large-scale motions and observe fine-scale motions of legs and wings, the glue and sticks can also impact the range of motions an insect can perform. For example, a full jumping motion is impossible while glued to a stick. Instead, we de-animate a video of an unencumbered grasshopper so that the motions of its legs are easier to see at lift-off. It is also easier to compare the length of the legs at full extension to the length of the body when the grasshopper is de-animated. We do not use our compositing technique for this example, and instead just crop the warped video.

Snorricam: A Snorricam[1] is a specially constructed camera device rigged to an actor's body who then moves through a scene. The result is that the actor is mostly static while the scene moves relative to her. We can simulate this effect without special hardware by capturing hand-held video, and then de-animating the actor's body. We show video of such

[1]http://en.wikipedia.org/wiki/SnorriCam

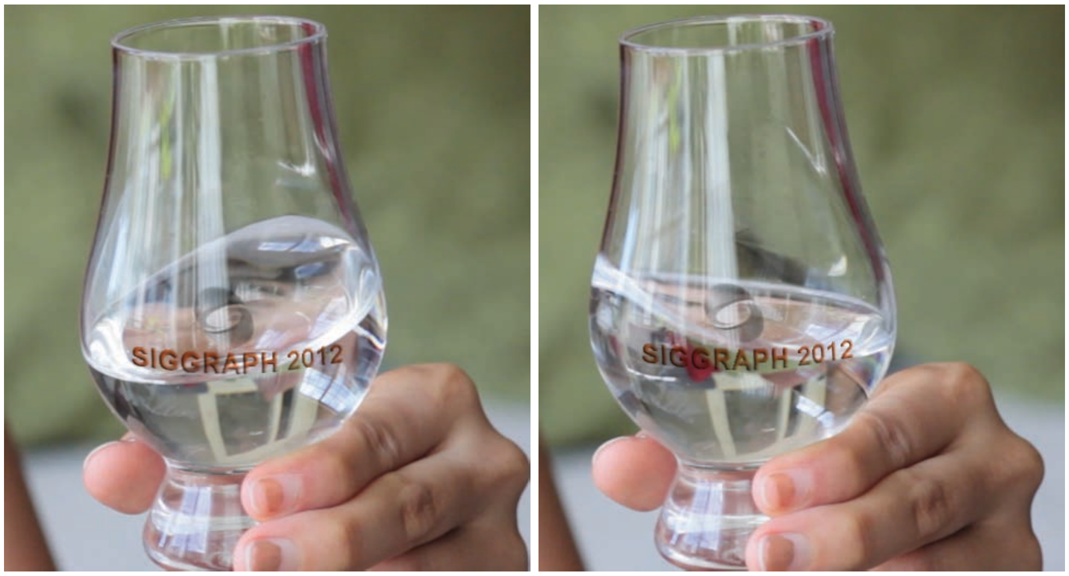


Figure 3.7: The image on the left shows the location of the logo the user places in a single frame. The image on the right shows the logo edit successfully affixed onto the glass 50 frames later. Note that the water has significantly changed.

H	Homographies with all tracks
AE	After Effects Warp stabilizer
AEM	After Effects Warp stabilizer with object mask
HA	Homographies with Anchor tracks
SVWA	Spatially Varying Warps with Anchor tracks
SVWF	Spatially Varying Warps with Floating tracks

Figure 3.8: Legend of method acronyms used in comparisons.

a result on the project web site, as well as average image visualizations of the input and de-animated output; the actor's body is much sharper in our output.

Cinemagraphs

Our next four examples are seamlessly looping cinemagraphs (Figure 3.6).

Beer and Glass: We have generated two cinemagraphs that immobilize glassware while retaining the dynamic motions of the liquid within. The first example shows beer pouring into a glass while the second shows water swirling in a glass. Because the glassware, hands and all surrounding objects are still, the motions of the liquid are unexpected and surreal. De-animating glassware is challenging because the glass is thin and transparent, and the dynamic liquid within can confuse trackers. Nevertheless our approach is able to properly immobilize the rims of the glassware. We also show a second result for the beer example

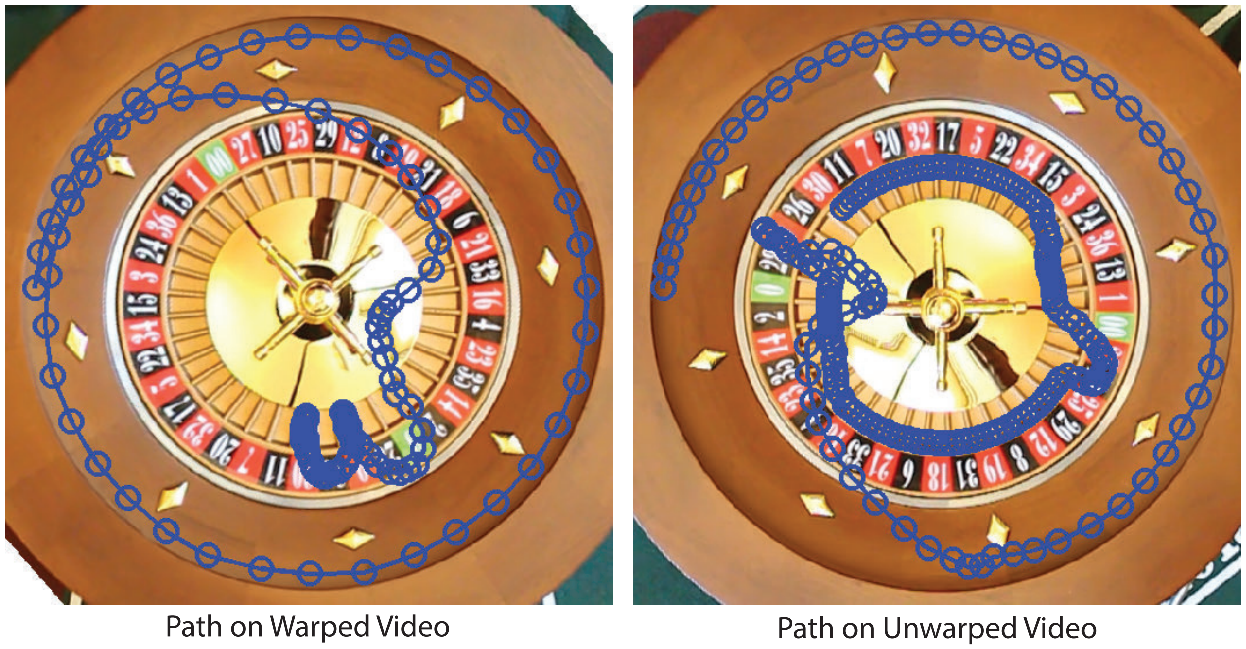


Figure 3.9: The image on the left shows the animated path the user drew to generate a winning number for roulette. The image on the right shows the same path of the ball in the unwarped video. Notice the complex path the user would have to plan if the inner wheel was not the frame of reference.

where we animate the beer flowing from the spout. Since the flow is already immobilized in the input, we use a special, yellow stroke to constrain those pixels to come from the input video during compositing.

Model K and Model S: It is particularly challenging to de-animate human faces and bodies because their natural motions involve non-rigid deformations. Our Model K and Model S cinemagraphs eliminate the motion of the faces and bodies while the motions of the eyes and hair remain intact. As in the glassware examples, the juxtaposition of static and dynamic regions is surreal because the resulting cinemagraphs are neither photographs nor videos.

Video Editing

Editing the appearance or motions of objects in video is a challenging problem. With our de-animation techniques users can modify a single reference frame in which the object is immobilized and our system propagates the edits back into the original video. While our approach is inspired by earlier work on editing video by modifying a reference frame [92], we offer a simpler method, but one that cannot handle as significant deformations.

Appearance Editing: In Figure 3.7, we add a logo onto the moving glass. For the edit to look realistic, the logo must appear to be attached to a fixed location on the surface of the glass. We de-animate the glass, then superimpose the logo at the desired location. When the edited video is un-warped, the logo appears properly attached to the glass. Even the liquid seen through the logo appears natural.

Motion Editing: In Figure 3.9, we edit the motion of a roulette ball. In the original video the motion of the ball becomes very complicated as it hits the inner rotating wheel. Designing a new path for the ball is extremely challenging because we must account for the rotational motion of the inner wheel. Instead, we first de-animate the inner wheel to provide a still frame of reference and then create a new motion path for the ball with respect to the immobilized wheel. With de-animation we no longer have to account for the relative motion of the inner wheel. Once we are satisfied with the new path of the ball we un-warp the video to obtain the final result. The accompanying video shows the full edit and animation.

3.7 Evaluation

Warping

Figure 3.11 quantitatively evaluates the success of the warp in immobilizing user-specified regions. (A table of method acronyms is in Figure 3.8.) The numbers in Figure 3.11 are mean RGB pixel variances, computed within the green strokes across all frames of the output video. While lower variance usually implies better de-animation, correlation with our perceptual assessment of the de-animated video is not exact; in particular, brief temporal artifacts can be very noticeable without significantly changing variance. We therefore strongly encourage the reader to view the video on our project page which show a detailed comparison for each sequence.

No existing method is designed specifically for de-animating objects. The closest related problem is video stabilization, which is designed to remove camera motion rather than object motion. We compare our approach to the Warp Stabilizer [76] in Adobe After Effects to show that existing stabilization techniques are not sufficient for our task, with the caveat that we are using the existing Warp Stabilizer for a purpose it was not designed for. We include two types of comparisons; we use Warp Stabilizer directly on either (1) the input video directly (AE) of (2) a version of the input video in which we manually masked out the background so that the stabilizer would eliminate the motion of the desired object (AEM). We compute AE and AEM with the stabilization set to ‘no-motion’ using ‘Subspace warp’ with ‘detailed analysis’. We also compare with simply computing the best homography, using all tracks (H). In all cases, our algorithm achieves lower numerical variance. From the videos, it is clear that our approach is also qualitatively better at immobilizing user-selected regions. Finally, to visualize our results, we compare the average image after de-animation for some examples in Figure 3.10. In the ideal case, the average image would be perfectly sharp, and our final method on the right should be compared to the input on the left. However, ghosting and blurring is clearly visible in results for H, AE and AEM.

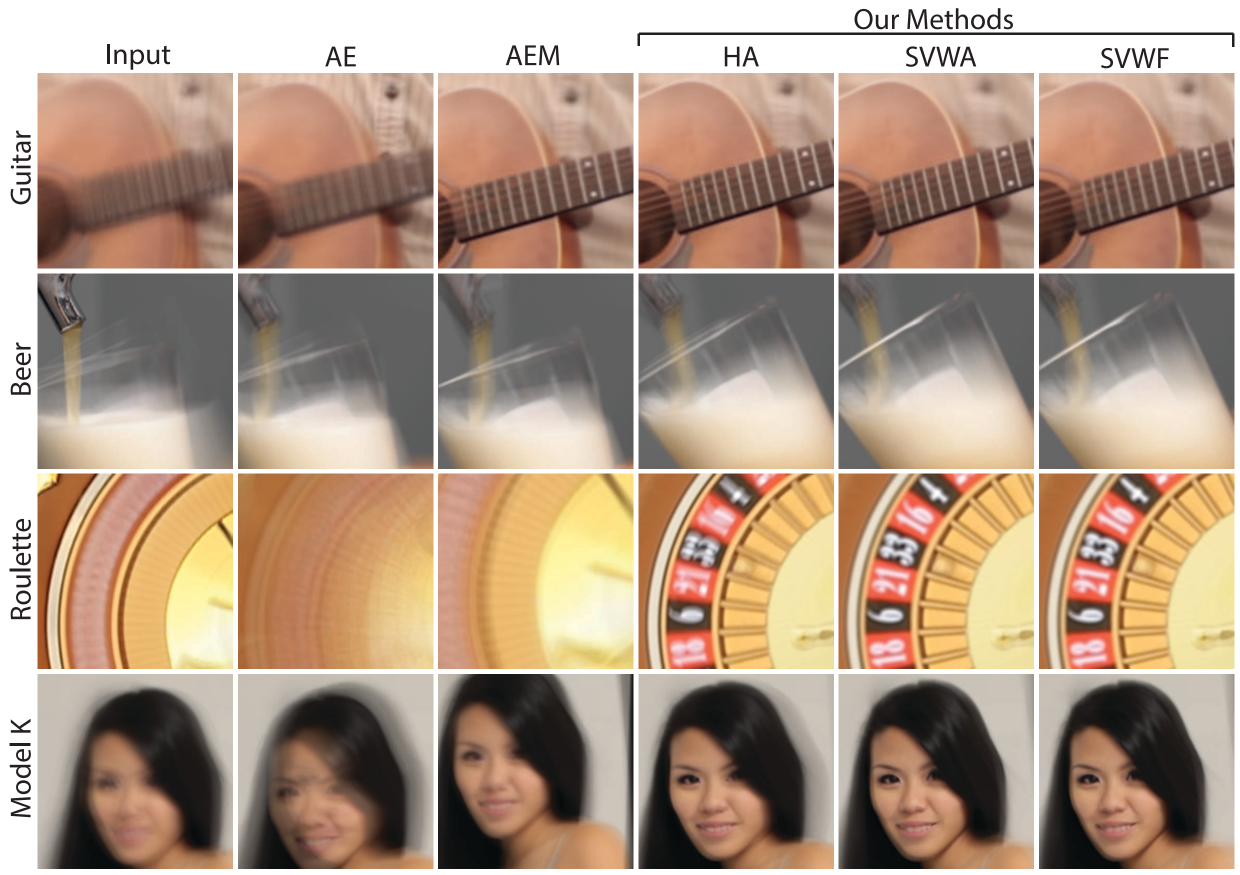


Figure 3.10: Average image of de-animated video. Notice the blurring of the guitar, rim artifacts on the beer, blurring of the roulette wheel, and ghosting in model K for the input and several comparison methods. Our result SVWF shown rightmost performs significantly better.

Figure 3.11 also evaluates the steps of our warping algorithm. First, we can use just the anchor tracks and apply a simple homography to warp each frame to the reference frame (HA). Our initial warp in Section 3.4 goes beyond a homography to use a spatially-varying warp (SVWA). Our final refined warp in Section 3.4 also uses the floating tracks (SVWF). In some cases, HA is adequate, but for more complex examples like Roulette or Grasshopper (see video), a spatially-varying warp is essential. Even for the Beer example, Figure 3.10 shows some jerking in the rim. A similar issue arises for Model K, where HA achieves the lowest numerical variance, but contains temporal jerks due to non-rigid deformations, resulting in the ghosting effects for HA in the bottom row of Figure 3.10. Using only the anchor tracks and SVWA is adequate in many cases, and close to the final numerical variance. However, there are difficult examples like Roulette and Guitar, where we obtain a more accurate result by also using floating tracks.

Compositing

Figure 3.12 shows the average image for the input, after warping the beer glass (Section 3.4), and compositing with the still background (Section 3.5). In the input, large-scale motions of the glass (blurred in the visualization) can mask the internal motion of the liquid. After de-animation, the glass is sharp, but distracting motions (again, blurred in this visualization) are transferred to the background. The final composite appears sharp for both the glass and background. The final video preserves internal motions of the liquid, while removing gross motions of the glass.

User Interaction and Timings

Tracking takes approximately 30 seconds for the longest video. The time for warping and compositing is listed in Figure 3.11, and currently takes a few minutes. User interaction to draw the coarse strokes can often be done in 1-2 minutes per video. Hence, the complete pipeline to generate a cinemagraph typically takes less than 15 minutes, with only a fraction of that time due to user input. Note that our algorithm is currently unoptimized, and we expect to be able to speed it up considerably in the future. Nevertheless, this workflow is significantly faster than the largely manual process artists must use today. Indeed, it is reported that skilled artists can take several hours, if not a day or two to create complex cinemagraphs[2].

3.8 Limitations

There are limits on the operating range of our method. It cannot handle or remove large 3D motions, such as a face rotating from a frontal to a profile view, or the full body of a walking person.

An example is shown in Figure 3.13, where we try to de-animate the subject's face, which undergoes large 3D rotations. The artifacts are easier to understand by watching the video on our project webpage. A good de-animation result should have the large-scale motions of the eyes, nose and mouth removed. However, due to the extreme motions of the face, our system fails to find a spatially varying warp that immobilizes the face. The nose and mouth still exhibit large scale motions as they do not remain in the same spatial location throughout the sequence.

Finally, since our method does not include video matting, we limit the types of background we use for our examples. For example, if our algorithm moves foreground objects relative to their background during warping, and the background is non-uniform, successful compositing might require alpha mattes to composite the de-animated foreground onto still images. Failing to composite only the foreground object from the warped video might result in animated background regions in the output sequence, which is distracting as shown in

[2]http://www.filmindustrynetwork.biz/nyc-photographer-jamie-beck-cinemagraph/12173

			Comparison Methods			Steps of our Algorithm			Timings for SVWF	
	# of frames	Input	H	AE	AEM	HA	SVWA	SVWF	Warping	Compositing
Beer	75	0.025	0.011	0.022	0.026	0.009	0.009	**0.009**	22 s	404 s
Glass	83	0.020	0.012	0.015	0.017	0.010	0.010	**0.010**	73 s	576 s
Grasshopper	44	0.031	0.013	0.042	0.033	0.016	0.007	**0.006**	14 s	-
Guitar	200	0.040	0.019	0.037	0.055	0.019	0.016	**0.017**	144 s	1287 s
Model K	270	0.026	0.008	0.024	0.031	0.008	0.009	**0.009**	42 s	300 s
Model S	117	0.010	0.002	0.004	0.006	0.002	0.001	**0.001**	260 s	650 s
Roulette	250	0.036	0.061	0.066	0.088	0.028	0.028	**0.028**	204 s	354 s
Snorri	380	0.127	0.063	0.273	0.275	0.063	0.026	**0.028**	959 s	-

Figure 3.11: Variances of de-animated results, with respect to the target frame. Values are computed in RGB space, varying from 0 to 1. Leftmost columns present the time required for the warping and compositing steps of the SVWF variant of our algorithm.

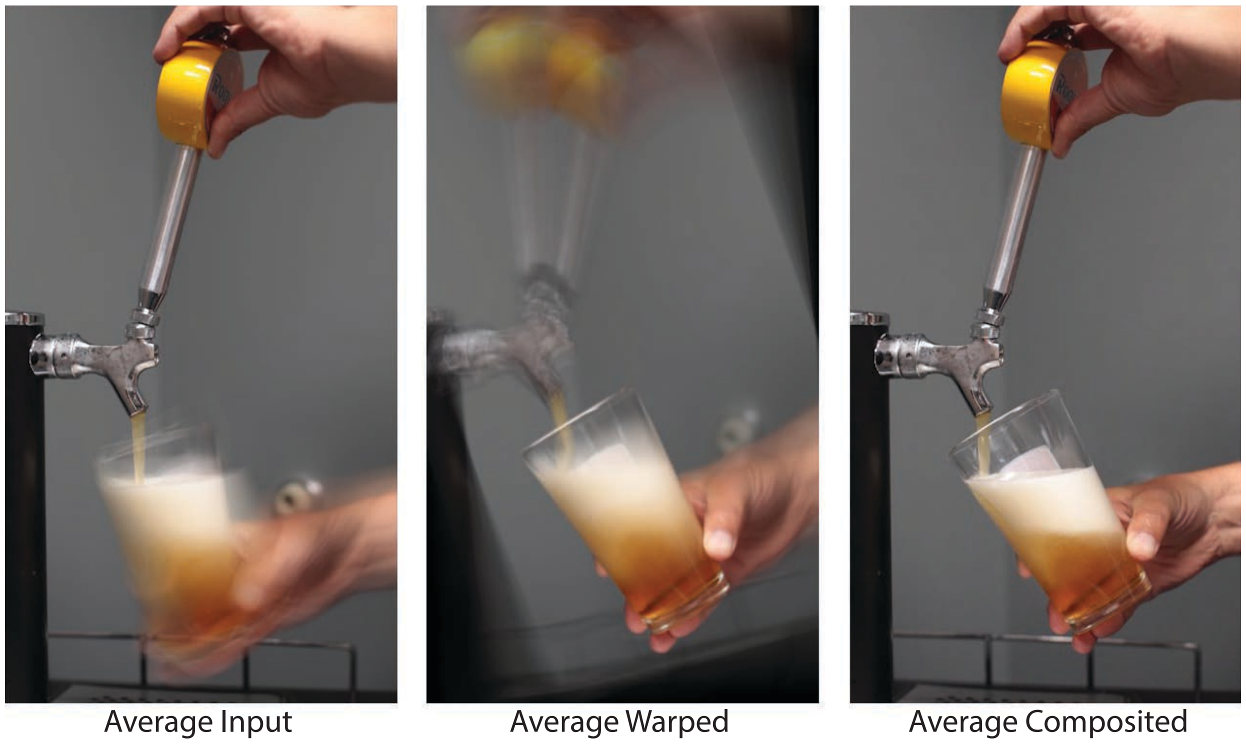


Figure 3.12: The average image of the input sequence, warped sequence and the composited sequence shows the effectiveness of combining the still background with the de-animated beer glass while retaining the internal motions of the liquid.

the Ketchup example in the bottom of Figure 3.14. In the video on our webpage, the out of focus grid 'swims' as the background region from the warped video is included in the output.

An obvious solution is to include matting in our algorithm, and we made a first attempt to model the foreground and background appearances with Gaussian Mixture Models (GMMs), using an energy term during compositing that prefers cuts along object boundaries. This simple approach is adequate in some cases; in Figure 3.14, the compositing seam does not cut through the background for Ketchup when we use GMMs.

However, GMMs fail for some cases when the foreground and background appearances are very similar. For example, the compositing seam cuts through the hand of the musician in the guitar example instead of its boundary as shown in the top row of Figure 3.14. This artifact appears because the color of the hand is similar to the color of the wall on the right and the ceiling. In this case, the energy minimization finds a composition that is seamless and obeys the color distribution of the foreground and background but fails to achieve the goal of matting. In summary, including matting in our algorithm remains a difficult challenge and topic of future work; while an approach based on GMMs or an alternative technique might succeed, we do not currently have a robust solution.

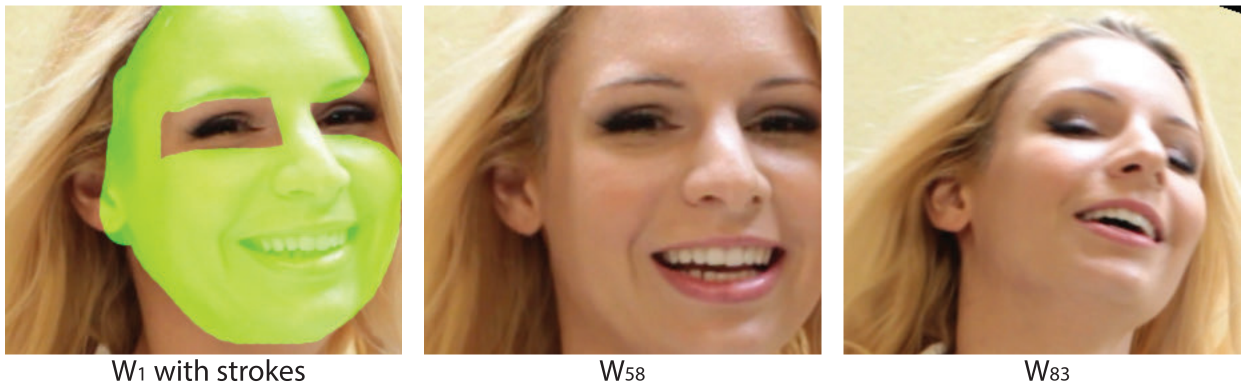


Figure 3.13: Our method fails to de-animate a face undergoing a large 3D rotation; the nose and mouth are not aligned throughout the output sequence. The image sequence shows the 1^{st},with de-animate strokes overlaid on it, 58^{th} and 83^{rd} warped frame.

3.9 Conclusion and Future Work

In this chapter, we have presented a simple and largely automated method for selectively de-animating objects in a video, removing gross motions to focus on finer-scale movements. The user only needs to specify coarse strokes on a single frame of the input. With the increasing use of video as a medium to enhance still imagery, we believe our technique will lead to much future work on analyzing motions to create more compelling and illustrative videos.

One limitation for this system is that although it can greatly reduces the time to generate high quality results from hours to minutes, it still requires some expert knowledge. Specifically, users will have to understand the nature of the algorithm to figure out where to draw green strokes to select tracks in order to successfully isolate motion she's interested in. While it is difficult to design a system that requires minimal or no user input for general classes of videos, we can design specific systems for restricted classes of videos.

In the next chapter we explore how portrait videos is a class where we can design a fully automated system that requires no user input to generate compelling cinemagraphs.

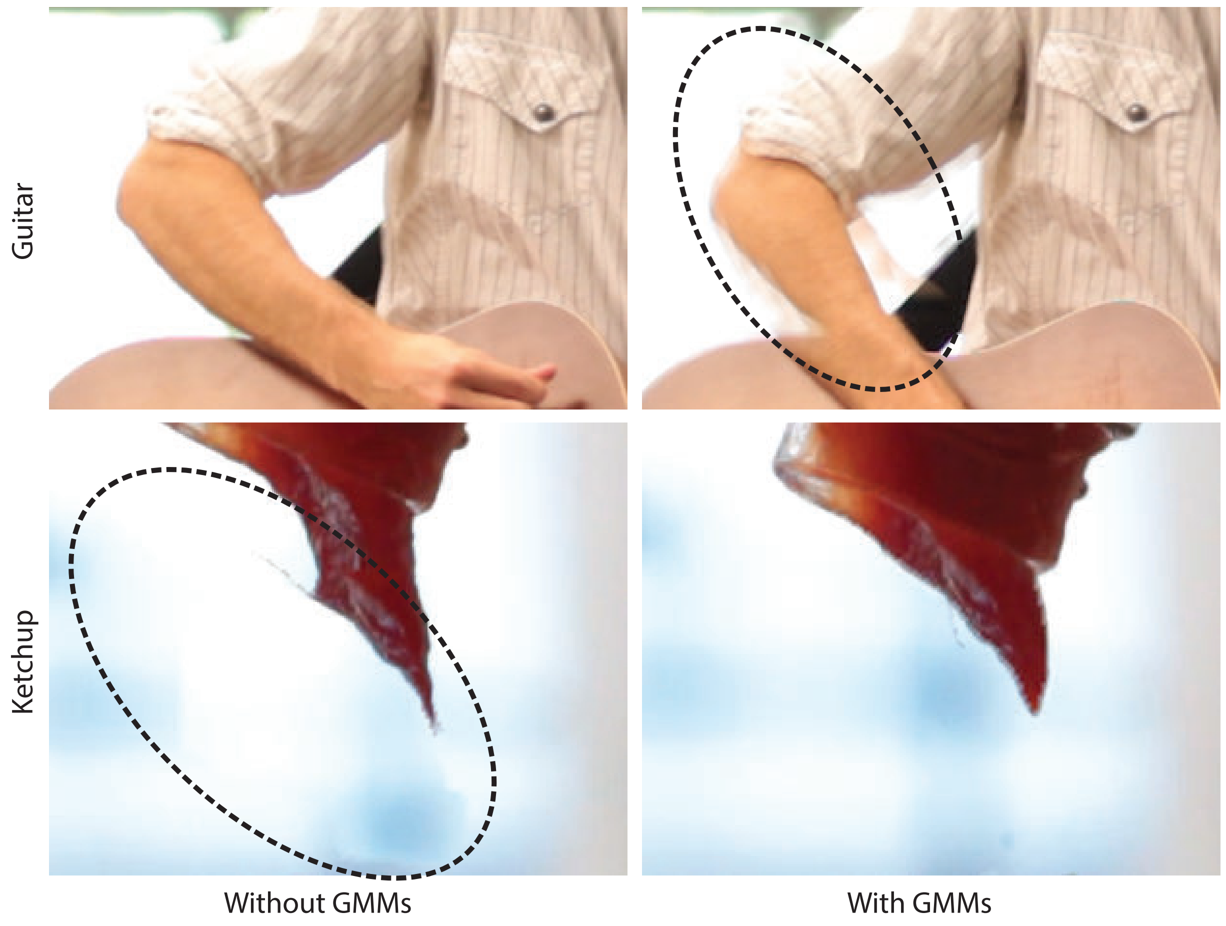


Figure 3.14: Our method might include background regions from the warped video in the final composite, causing distracting motions in the output for Ketchup. GMMs can be used to model the foreground and background appearance to achieve a better composition in this case. However, GMMs does not work for all cases as shown in Guitar, where it fails to find the boundary for compositing.

Chapter 4

Automatic Cinemagraphs

4.1 Introduction

In the previous chapter, we showed how our system can enable users to create new motion visualizations such as cinemagraphs. Cinemagraphs are a new medium that combines the benefits of static images and videos; most of the frame is static, but some parts animate in a seamless loop [1]. For example, a portrait cinemagraph might show dynamic facial expressions while maintaining a static overall head pose and background. Unfortunately, such portrait cinemagraphs are remarkably difficult to create, taking professional artists several hours if not a day [2]. Our system presented in chapter 3 allows users to create cinemagraphs within minutes by drawing appropriate strokes to select tracks to isolated desired motion and strokes to composite them onto the final output. However, these strokes might still be challenging for a novice user to draw as they might not fully understand the inner workings of the system. Therefore, we seek to design a fully automatic system that can generate high quality results by restricting the domain of input videos to portraits.

We all use portrait photographs to express our identities online. Portraits are often the first visuals seen by visitors to our professional webpage, social media profiles, and online dating sites. However, static portraits can fail to capture the personality we see on people's faces in real life, which includes their neutral expressions, smiles, laughs, and the transitions in-between. Short videos can capture these dynamics, but videos are awkward as online portraits because they are *too* dynamic; videos can contain camera and background motion, as well as large-scale motions of the head and body. They also have a timeline — a beginning, middle and end.

In this chapter, we completely automate the creation of cinemagraph portraits. Users simply capture a short hand-held video of a person, push a button, and our system outputs a portrait cinemagraph. There are a number of recent techniques for creating cinemagraphs, including mobile apps [3, 4, 5] and several research systems [107, 59, 116, 71]. However, these techniques are all challenged by portraits, because people are not good at keeping their head and body perfectly still. The existing techniques only stabilize overall camera motion, and a cinemagraph portrait requires removing the gross motion of the subject while preserving facial expression dynamics. Our system presented in chapter 3 removes the large-scale motions of objects, but it requires three different sets of strokes from the user, and the results can be very sensitive to stroke placement. Since compute time is also quite long, the system is impractical for casual users.

We contribute a fully automatic pipeline for portrait cinemagraphs. This method is adapted from that in Selectively De-animating Video (Chapter 3), but does not require user strokes. We use face tracking [96] as well as Kanade-Lucas-Tomasi [78] point tracks to automatically segment motion into two classes: large-scale head and torso motion that should be removed from the input video, and fine-scale facial expression motion that should be preserved in the cinemagraph. We use the large-scale motion information to guide a spatially-varying warp [74, 12] that removes the gross motion of the portrait subject. We then use a 2D graph-cut technique [22, 65, 6] to compute a mask that seamlessly combines the dynamic facial expressions with a still image for the rest of the subject and background. Our 2D approach is much faster than the 3D graph-cut in Selectively De-animating Video, which makes our approach amenable to implementation on portable devices such as phones and tablets. We also compare our method with a more naive approach that does not first segment motion into two types.

4.2 Overview

Our approach has three distinct stages as shown in Figure 4.1. In the first stage, we automatically select Kanade-Lucas-Tomasi (KLT) tracks which can immobilize the face and torso after a warp. Specifically, our method identifies and eliminates KLT tracks that follow expression motions in the face rather than the gross motion of the entire face. We then use RANSAC [43] on the remaining static KLT tracks to select a subset that can immobilize the face using a homography transform. In the second stage, we warp all frames to an automatically selected target frame using a spatially-varying warp [74, 12] to immobilize the face.

In the third stage, we composite dynamic facial regions from the stabilized face video into a still background image. We use graph-cuts optimization on a Markov Random Field (MRF) to find the best compositing seam between the stabilized face video and the selected background frame. We automatically generate energy penalties corresponding to the detected moving facial parts in the output to act as constraints for the energy minimization. We also design new seam costs to encourage compositing seams to occur at regions with minimal pixel variation over time.

Compared to Selectively De-animating Video, there are two key technical differences that enable fast, completely automatic cinemagraph generation. First, we automatically analyze facial motion (Section 4.3) to differentiate between overall head motion and internal motion of facial expressions. Before this analysis the user needed to carefully place strokes in regions that only undergo overall head motion, which was challenging. Second, we improve speed by

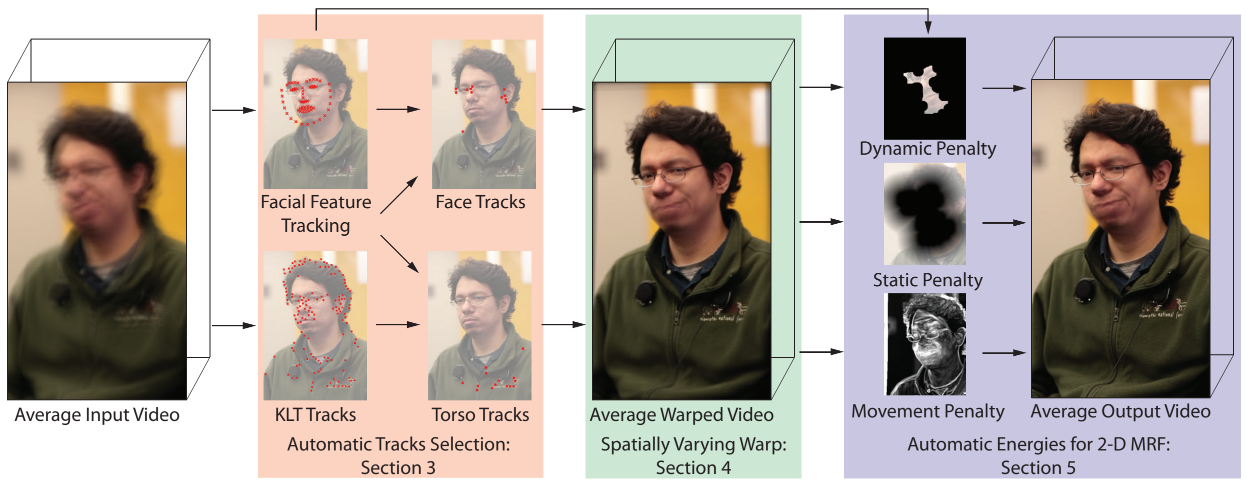


Figure 4.1: Our method automatically generates a portrait cinemagraph from an input video. The input, warp and output videos are visualized as averages across time. Notice that the face and torso are blurry due to motion in the input video. After warping, the face is sharp as it is stabilized, but the mouth and jaw are blurry as we preserve the internal motions of the face. The blurred border in the average warp video is due to the motion of the background. The average output video is sharp for static regions and blurry for facial parts which are animated. The key to our algorithm is a fully automatic technique to select KLT tracks which lie on static regions of the face, which allows us to immobilize the face after warping. We also compute automatic energy values for a graph-cut optimization which composites the warped video with a still image to create the final cinemagraph.

two orders of magnitude by using a 2D MRF with new cost functions based on facial motion analysis (Section 4.5), rather than a 3D MRF, along with several other optimizations.

Our approach relies on several assumptions about the input video. First, there is only one person with one facial pose in the video. We do not handle pose changes at the moment, although pose selection can be performed as a pre-processing step. Second, the face of the person should occupy a significant portion of the image; a good rule of thumb is about 10% of the image area. Third, the face can be aligned using a 2D warp, which means that there should not be large 3D rotations. However, in practice, reasonable results are obtained even when these assumptions are violated.

4.3 Track Selection

In the track selection stage, our goal is to select KLT tracks to guide a single spatially-varying warp to immobilize the face and torso. Our intuition is that KLT tracks that are useful for stabilizing the large-scale head motion should be located on the face, but not fall on any

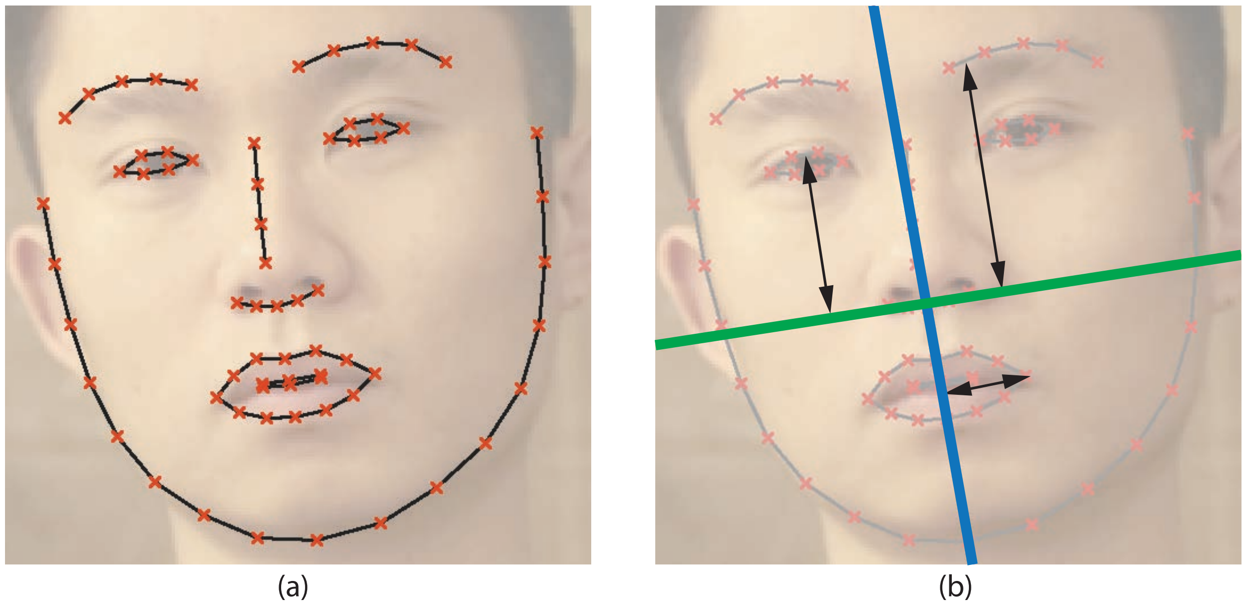


Figure 4.2: Facial features detected in a single video frame (a). We compute 2 axes per frame, a vertical line bisecting the face (blue) and a horizontal line across the base of the nose (green) (b). We measure facial regions by computing the perpendicular distance of their corresponding feature points to the axis perpendicular to their movement.

moving facial regions that are part of expression changes. They should also be consistent with a per-frame homography transformation, which is usually a good approximation for the overall motion of a face that is not significantly changing pose.

We first compute KLT tracks for the input video. KLT tracks which last for the entire video are denoted $\hat{K}_A$ and tracks which do not are denoted $\hat{K}_F$. This definition is similar to the anchor and floating tracks in Selectively De-animating Video (Chapter 3).

In order to find KLT tracks that are useful for stabilization, we use a face tracker [96] to mark facial feature points throughout the video and analyze their motion to identify which facial regions are moving, as shown in Figure 4.1 and discussed in the latter section. Removing KLT tracks in moving facial regions which are not useful for stabilization significantly decreases the number of iterations for track selection.

Facial Movement Detection

The goal of tracking facial features is to allow our algorithm to not only identify the location of the face, but also to determine the movement of the major facial regions in the face. For our application, we only consider four major facial regions; the eyes, eyebrows, mouth and lower jaw. The mouth is further sub-divided into three sub regions; the left and right-most region of the lip and the bottom lip. We use Saragih et al.'s [96] face tracker to track the location of 66 facial feature points across the face as shown in Figure 4.2a.

Facial Region	Feature #	Axis	Motion Threshold
left eye	38, 39	Horizontal	0.001
right eye	44, 45	Horizontal	0.001
left brows	20, 21, 22	Horizontal	0.003
right brows	23, 24, 25	Horizontal	0.003
left tip of lip	49	Vertical	0.0015
right tip of lip	55	Vertical	0.0015
bottom lip	57, 58, 59	Horizontal	0.0015
bottom jaw	8, 9, 10	Horizontal	0.0015

Figure 4.3: Each facial region is represented by the mean position of the corresponding feature number in the second column. The distance values for each region are measured from the assigned axis in the third column. The fourth column is the threshold value for motion detection after convolution with a LoG kernel and normalization by nose length l.

The location of each facial region is the center of its corresponding facial feature points which are listed in Figure 4.3. We monitor the position of each facial region across time by using a coordinate transform. We measure its perpendicular distance from one of two local face orientation axes to detect finer-scale motions, as shown in Figure 4.2b. One axis bisects the face vertically passing through the bridge of the nose, and the other bisects the face horizontally across through the base of the nose. The vertical axis (blue) is the best fit line which passes through the facial feature points through the bridge of the nose and chin, and the horizontal axis (green) is the best fit line which passes through the facial feature points through the base of the nose. We recompute the face orientation axes for every frame. Note that the two axes are not necessarily perpendicular. We have found that this fit is stable for most of our examples as the facial feature tracker is stable for small facial motions.

We measure the perpendicular distance of each facial region from the axis where the motion change is dominant. This gives us fine-grained movement discrimination, so we can differentiate a slight smile where lips are closed (horizontal displacement of edges) versus a laugh (vertical displacement of lower lip). For example, the position of the eyebrow is the perpendicular distance measured from the horizontal axis (green). Figure 4.3 shows which axis each facial region is measured from.

We also compute the radius r and the nose length l of the face. Radius r is computed by averaging the three pixel distances from the center of the nose to the lower jaw, left ear and right ear (features # 9,1,17). Length l is simply the average length of the nose in pixels computed from the start and end feature points of the nose over all frames (features # 28, 31). We normalize all distance computations so that our algorithm is independent of face size by dividing all distances with l.

We use a median filter of kernel size 5 to smooth temporal noise after we compute the

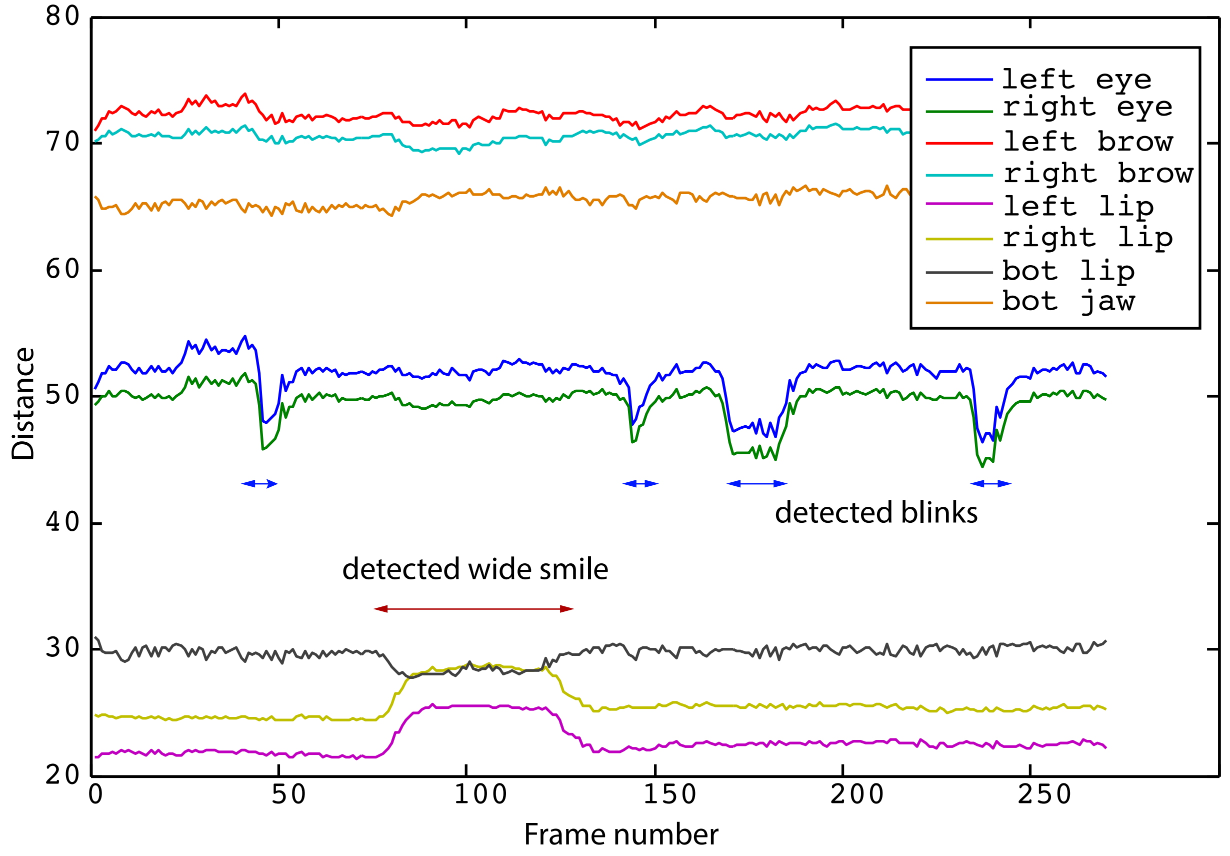


Figure 4.4: Typical raw distance values for facial regions in an input video. Notice the eye blinks give rise to distance drops for the eyes. These time instances are marked with a blue horizontal arrow. A smile causes the distance of the lips to change and this time instance is marked with an orange horizontal arrow.

distances for each facial region per frame. Figure 4.4 shows a typical plot of the raw facial region responses for an input video and the detected movement. Note how the distance measurements change significantly for the facial regions when they move. We convolve the measurements with a Laplacian of a Gaussian (LoG) kernel of size 35 with sigma 4 to detect sudden change within a 1 second timeframe. The peaks of the convolved response indicate potential movement for their corresponding regions. For a facial region to be considered dynamic with respect to the face, the peaks in the convolved response have to be larger than a specified value. Figure 4.3 shows the threshold for movement detection we use, normalized by the nose length l. These thresholds were determined empirically from our dataset.

Inlier ratio	0.6	0.64	0.69	0.75	0.81	0.9
# of iterations	3159	1378	556	204	65	17

Figure 4.5: Here we show the number of iterations needed to guarantee a probability of 0.99 to find a good homography fit for varying inlier ratios (total number of tracks = 30). As the inlier ratio improves, we need significantly fewer iterations to find a good homography.

Track Pruning

KLT tracks in $\hat{K}_A$ that do not track large scale motion of the head should not be used for stabilization. We first select only the tracks in the facial region by removing tracks in $\hat{K}_A$ that are outside of a radius of $1.5r$ pixels from the center of the face. Second, we further prune the tracks that lie on the dynamic facial regions that form expressions by removing tracks that are inside or within a $0.2l$ distance from a moving facial region.

Once these tracks have been removed, we use RANSAC to fit a per-frame homography to the face motion; RANSAC also removes additional outliers that do not follow the overall face motion. Note that the initial removal of tracks significantly improves the ratio of inliers to outliers for homography fitting. A slight improvement in inlier ratio can significantly decrease the number of iterations needed for RANSAC to have a good probability of finding a fit which in turn decreases the computation required. Figure 4.5 shows the number of iterations needed for a 99% confidence of finding a good fit with varying inlier ratios. In practice, we have found that using this track removal technique decreases the number of RANSAC iterations needed by up to 2 orders of magnitude.

At each RANSAC iteration, we attempt to remove the motion of the face over the entire video by fitting homographies per frame to the remaining tracks using linear least squares. The set of tracks which gives the lowest RMS error for inliers after fitting homographies are the final set of selected tracks. We run RANSAC for 50 iterations with a 3.5 pixel inlier threshold for all of our examples. We label the final selected tracks from $\hat{K}_A$ as K_A. Figure 4.6a shows K_A selected for the face.

$\hat{K}_F$ tracks do not last the entire duration of the video, but can still help to guide the warp if we can be sure they are within regions which do not contain finer-scale motions. They provide consecutive frames with constraints during warping. We find useful K_F tracks by warping $\hat{K}_F$ tracks using the homographies estimated to align the frames. If a $\hat{K}_F$ track deviates no more than 3 pixels for each frame from its mean position, we consider it to be an inlier. The inlier K_F tracks are then used as input for warping along with K_A tracks.

We apply the same technique to stabilize the torso, but set the center of the torso as $3r$ below the face and only consider tracks within a radius of $4r$ of that center. No torso-specific track selection is performed, but any tracks removed during facial track selection are also not used for the torso even if they fall within the torso radius. Note that the homographies fit

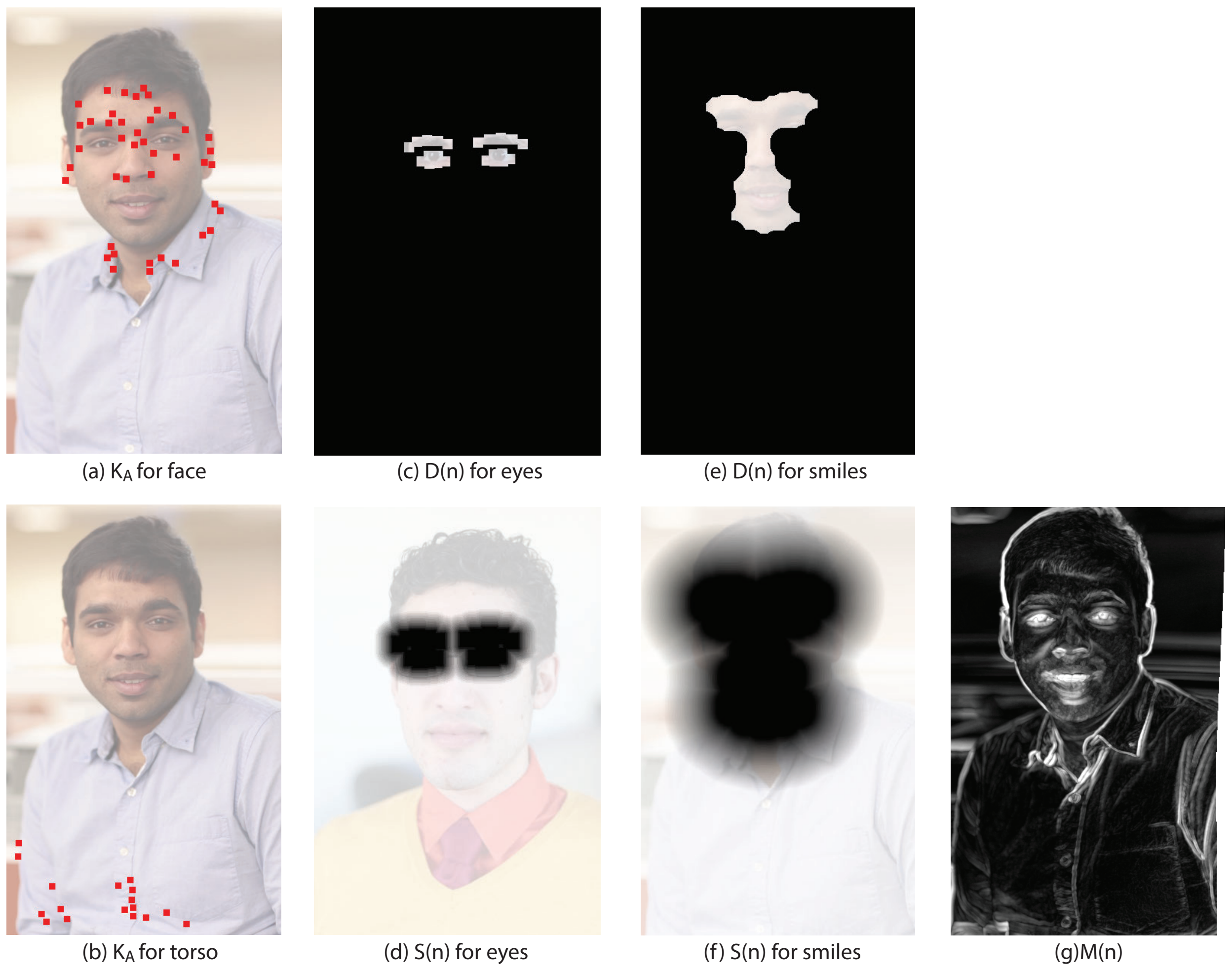


Figure 4.6: From left to right: (a) K_A for face, (b) K_A for torso, (c) example $D(n)$ for eyes, (d) corresponding $S(n)$ for eyes, (e) example $D(n)$ for smiles, (f) corresponding $S(n)$ for smiles, (g) example $M(n)$. The energy values are visualized overlaid with the portrait to provide spatial reference. High energy values of 1 have high luminance values, while energy values of 0 are black.

to torso movement are separate from the face homographies. Figure 4.6b shows K_A selected for the torso.

4.4 Warping

The goal of warping is to immobilize the face and torso in the video. Our approach finds a target frame to warp to instead of picking the first frame or forcing the user to select one. The automatically selected target frame should generate the least amount of warping after registering all other frames to it. Therefore, we find the target frame T by selecting the input frame t_1 that minimizes the L_2 distance between the locations of all K_A tracks in all other frames t_2.

$$T = \arg\min_{t_1} \sum_{\text{all tracks},t_2} |K_A(t_1) - K_A(t_2)|^2 \tag{4.1}$$

Frame T is also the target frame used in the next compositing stage.

We use the same warping algorithm as described in Selectively De-animating Video (Chapter 3.4) with our automatically selected K_A and K_F as input. The tracks act as constraints over a grid mesh defined over each frame. Since the tracks specify locations on static regions, they should not move after the warp. Therefore, we solve for a set of meshes where the tracks K_A and K_F are stationary over time using least squares. Since we have selected the tracks K_F, we only have to use the second, refined warp described in Selectively De-animating Video (Chapter 3.4), thus reducing computation costs by up to 50%. The warped output is generated by texture mapping the frames onto the grid mesh.

4.5 Compositing

At the final stage, we use graph-cuts over a 2D Markov Random Field (MRF) to composite the warped video with the target still frame. Each node n corresponds to a spatial pixel location and is assigned one of two labels: $\lambda = \{still, warp\}$ after a cut is computed. Since our MRF is 2D, the assigned label at the node determines the pixel source for that spatial location across time; either the target frame T or the warped video. Our compositing algorithm is based on Selectively De-animating Video (Chapter 3.5) , but with three key differences. One, we use a 2D rather than 3D MRF. Two, we use a single still target frame, rather than a set. Three, since we no longer have user-provided strokes to guide the choice between still and dynamic regions, we must instead design new cost functions based on our analysis of facial motion that select only tracks indicating overall head motion.

There are two types of energies for our MRF: node potentials and seam costs. Node potentials are determined by the label assigned at the node n and seam costs are defined over two neighboring nodes, n_1 and n_2. We design our energy such that when it is minimized over the MRF using graph cuts, the labeling results in a composite cinemagraph that is natural with desired animated and static regions that transition seamlessly. We describe our cost functions (automatic energy functions) in Section 5.1 and our overall energy functions in Sec 5.2.

We loop the output video by finding a pair of starting and ending frames that are visually similar. We minimize the total L_2 RGB distance in the warped video at the moving facial regions, using the same technique as video textures [97] to find the best loop. We then trim the input video and warped video to this duration. We also create a seamless cinemagraph by interpolating between the start and end frames to minimize visual artifacts using optical flow [72]. We advect pixels from both directions based on their computed corresponding flows to generate an additional 10 intermediate frames. We cross-fade between the two constructed interpolations to create a temporal seamless loop.

Automatic Energy Functions

We compute three sets of energy functions (D(n), S(n) and M(n)) that range from 0 to 1 and are shown in Figure 4.6c-g. The energy terms are computed per node and thus correspond to each pixel. The computation of our energy terms is dependent on the presence of lip or jaw movements detected in the facial expressions; we use the movement detection approach described in Section 4.3 and Figure 4.3. These automatic terms are designed to provide constraints to guide the MRF optimization. Specifically, D(n) is designed to encourage moving facial regions from the warped video to be composited in the final result. S(n) is designed to encourage relatively static regions to become static in the final result by compositing a still image. M(n) is designed to encourage seam transitions to happen at regions with minimal pixel variation over time.

Dynamic Region, D(n)

This energy is computed for nodes with label $\lambda = still$. We wish to assign a high energy for these nodes if they are within dynamic facial expressions, since they should be animated.

If no lip or jaw movements are detected, each feature point at a moving eye or eyebrow is assigned an energy value of 1 at its location at frame T with a radius of $0.1l$, as shown in Figure 4.6c. Pixels without any contributing features are assigned a 0 energy. The energy region is processed using morphological closing (image dilation followed by image erosion) with a radius of $0.25l$ to fill in holes in the energy region.

We generate this penalty differently if lip and jaw movements are present as the entire face is usually animated due to muscle structure. Each feature point at the lower jaw, eyes, eyebrows, nose and lips is assigned an energy value of 1 at its location at frame T with a radius of $0.1l$. The energy region is processed using morphological closing with a radius of $0.5l$ as shown in Figure 4.6e to fill in holes in the energy region.

Static Region, S(n)

This energy is associated with nodes with label $\lambda = warp$. We wish to assign a high energy for these nodes if they are not within dynamic facial expressions, since they should not be animated. We also want a ribbon of pixels between $S(n)$ and $D(n)$ where there are minimal constraints so that the algorithm is free to find seamless boundaries for compositing.

If no lip or jaw movement is detected, we generate this penalty by an image erosion of the complement of $D(n)$ by a size of $0.5l$. We then compute a linear penalty fall off with gradient 20 to encourage transitions to happen near the face, as shown in Figure 4.6d.

If lip and jaw movements are detected, we generate this penalty by an image erosion of the complement of $D(n)$ by a size of $3l$. We then compute a linear penalty fall off with gradient 5 to encourage transitions to happen near the face, as shown in Figure 4.6f. We use a larger kernel and a smoother gradient for image erosion because lips and jaw movements require a larger region for animation. The linear fall off is computed by using a Euclidean distance transform [41] multiplied by its corresponding gradient.

Pixel Movement, M(n)

This energy measures a component of the cost of seams between neighboring pixels. We want seam transitions between warp and still regions to happen where there are minimal pixel fluctuations across time. Hence, we take the maximum RGB L_1 distance of each pixel from the mean image of the warped video $\hat{W}(x, y)$ across all frames, as shown in Figure 4.6g.

$$M(x, y) = max_t(|\hat{W}(x, y) - W(x, y, t)|); \tag{4.2}$$

Energy Function

There are two types of energies; node potentials Ψ and seam costs Φ. Each node n in the MRF has an energy associated with it depending on its label. A is a constant set to 1000000 to discourage any dynamic regions from being assigned a static label. Then the node potential Ψ is defined as:

$$\Psi(n, \lambda) = \begin{cases} A \times D(n), & \text{if } \lambda = \textit{still} \\ S(n), & \text{if } \lambda = \textit{warp} \end{cases} \tag{4.3}$$

Seam costs are defined over 2 neighboring nodes n_1 and n_2 depending on their assigned labels. Let $\lambda(n)$ be the assigned label at node n and $C(n, \lambda(n), t)$ be the color of the pixel p at the node location from its respective label source at frame t. If the labels of the neighboring nodes are equal ($\lambda_1 = \lambda_2$), the seam cost is zero. Otherwise, the seam cost $\hat{\Phi}$ at time t is

$$\hat{\Phi}(n_1, n_2, \lambda_1, \lambda_2, t) = \frac{\gamma(n_1, n_2, \lambda_1, \lambda_2, t)}{Z(n_1, n_2, \lambda_1, \lambda_2, t)} \tag{4.4}$$

$$\begin{aligned} \gamma(n_1, n_2, \lambda_1, \lambda_2, t) = {} & |C(n_1, \lambda_1, t) - C(n_1, \lambda_2, t)|^2 \\ & + |C(n_2, \lambda_1, t) - C(n_2, \lambda_2, t)|^2 \end{aligned} \tag{4.5}$$

This seam cost is based on the work of Agarwala et al. [6]. Function γ measures the color similarity of the source pixels at the two nodes along the seams, while Z measures the edge strength of the warped video. Therefore, $\hat{\Phi}$ encourages seams to have similar colors or lie at edges in the warped video.

$$Z(n_1, n_2, \lambda_1, \lambda_2, t) = \tag{4.6}$$
$$\begin{cases} \sigma(n_1, n_2, \lambda_1, t) & \lambda_1 \in \text{warp} \wedge \lambda_2 \in \text{still} \\ \sigma(n_1, n_2, \lambda_2, t) & \lambda_1 \in \text{still} \wedge \lambda_2 \in \text{warp} \\ \frac{1}{2}[\sigma(n_1, n_2, \lambda_1, t) & \\ +\sigma(n_1, n_2, \lambda_2, t)] & \text{Otherwise} \end{cases}$$

where $\sigma(n_1, n_2, \lambda, t)$ is the edge strength within a pixel source λ and is computed with a 3×3 Sobel filter averaged across RGB.

Since we are using a 2D MRF instead of a 3D MRF, we collapse the seam cost across time to compute the best overall static compositing matte. We also include M(n) in the seam cost to encourage seams to occur in regions that have minimal pixel variation over time.

Therefore, the total seam cost Φ, across all time is

$$\Phi(n_1, n_2, \lambda_1, \lambda_2) = \sum_t \hat{\Phi}(n_1, n_2, \lambda_1, \lambda_2, t) + \alpha(M(n_1) + M(n_2)) \tag{4.7}$$

where $\alpha = 200$. In total, our energy function which we seek to minimize is:

$$\frac{N}{4} \sum_n \Psi(n, \lambda) + \sum_{n1,n2} \Phi(n_1, n_2, \lambda_1, \lambda_2) \tag{4.8}$$

where N is the total number of frames in the output video. Ψ provides constraints on which general region should be static or dynamic while Φ encourages seams to occur between pixels with similar colors and a stable appearance over time, or lie at an edge. We minimize this energy function using the alpha-expansion algorithm [22]. Once a labeling is computed, we create a final video simply by copying pixels from their respective sources.

4.6 Results

We captured 15 portrait videos and produced cinemagraphs to demonstrate the wide range of results our algorithm can create. Input sequences range from 5 seconds to 10 seconds. We resize the frame to 405 by 720 pixels. The videos are captured using DSLRs as well as mobile phones. Figures 4.10, 4.11, 4.12, and 4.13 detail the examples that we used. The figures show the average image for both the input and output videos, tracks automatically selected by our algorithm and the automatically generated penalties. Notice that the average input images are blurry as there is camera shake, while the average output image is only blurry at facial features with movements. Please refer to our main video and portrait cinemagraphs for each model for the final results, as well as the supplementary material for comparisons.

Comparisons

To evaluate the quality of our result, we show some comparisons of our automated cinemagraphs against our user driven method described in Chapter 3 in Figure 4.7. Notice that the stabilization in the result video is equally good even though our method selects fewer (but the most critical) tracks. Our method is capable of selecting tracks within the face that might be too tedious for the user to select with fine strokes, such as tracks that are between the eyes (Figure 4.7b,d,f). The compositing seams are also comparable even though our method uses only a 2D matte (see videos). In some cases, our method includes a small portion of the background in the final output, which can be distracting as the background will be animated. This could be due to the use of a static 2D matte or the automatic approach failing to find a compositing seam within the face.

In some examples, the motion of the camera and face is simple and a regular video stabilization algorithm will be able to stabilize the input video well enough for creating cinemagraphs. However, input videos with moving backgrounds or erroneous tracks in the background region can cause regular video stabilization to fail. Our method does not suffer from this issue because we only use tracks which lie on stationary regions on the face and torso. Please refer to the supplementary material for comparisons.

Another comparison we perform is to a naive algorithm that does not first prune tracks on dynamic facial expressions. Given the inlier/outlier ratios for each of our 15 videos, we simulate the number of RANSAC iterations needed to give a 99% probability of finding the correct homography. We find an average reduction in the number of iterations to be a factor of 27, with the largest reduction a factor of 185 across our examples.

Timings

The timings for each stage of our pipeline are shown in Figure 4.8. Our current implementation is single threaded and timings are measured on a 2 Ghz Intel i7 laptop. Our method is faster than previous work in all stages of the pipeline. For track selection, our method averages 0.70 seconds while a user will take 90 seconds (Chapter 3.6), on average. Since our warp is simpler, our timings are also much faster. Our 2D graph cut is significantly faster than previous work due to the smaller number of nodes as well as the smaller set of labels. Our average compositing times are 14.41 seconds while the average compositing time in previous work is about 600 seconds for shorter videos (Chapter 3.6). Also, previous work often required iterations from the user to improve the input strokes.

Eyes and Eyebrow

All the input videos have eye movement from blinking. Our algorithm successfully composites the blinks as well as any eyebrow movements into the final cinemagraph. The large movements of the eyebrow in Model A are successfully retained and animated in the output cinemagraph. However, for some examples such as Model R, slight movements beyond the eye are also included in the output cinemagraph as seen in the video.

Lips and Jaw

Most of our examples have mouth movements such as smiles that are successfully animated in their output cinemagraphs. Jaw movements are the hardest to composite as the face usually changes its appearance significantly. Track selection becomes harder as there are fewer static tracks. The margin of error for compositing is also greatly reduced as most of the face is animated. Therefore, no previous work demonstrates animated smiles. None the less, our algorithm successfully composites large jaw movements and smiles such as Model

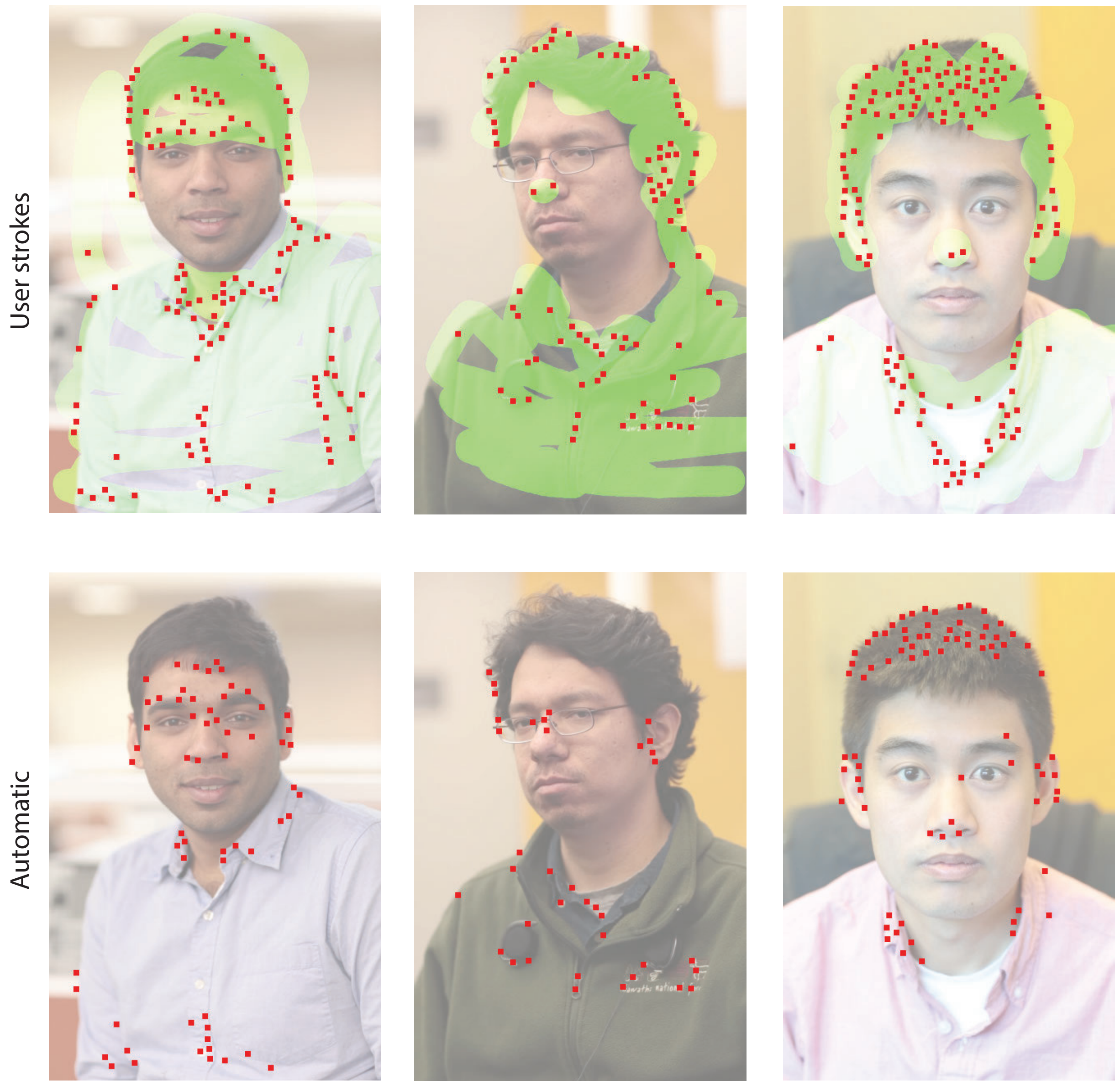


Figure 4.7: Comparison of the tracks selected using user-provided green strokes (Chapter 3), versus our automatic method. Notice that our method selects fewer but more crucial tracks for stabilization. Our method has the advantage of selecting tracks that are very near moving regions where it is hard for the user to draw appropriate strokes.

B and Model R, as shown in our video. Notice that while there are large motions for the entire face for Model R, we are still able to generate a good portrait cinemagraph.

Eg.	# frames	Track (Sec. 3)	Warp (Sec. 4)	Comp (Sec. 5)	Total
A	131	0.43 s	3.86 s	5.53 s	9.82 s
B	235	0.69 s	6.78 s	12.97 s	20.44 s
C	383	0.69 s	11.19 s	43.62 s	55.5 s
D	243	0.93 s	7.38 s	14.51 s	22.82 s
E	157	0.76 s	5.10 s	7.77 s	13.63 s
F	157	0.44 s	4.49 s	8.33 s	13.26 s
G	157	0.56 s	5.39 s	9.28 s	15.23 s
J	279	0.60 s	8.53 s	19.91 s	29.04 s
M	181	0.76 s	6.33 s	12.12 s	19.21 s
N	281	1.70 s	9.64 s	21.88 s	33.22 s
R	292	0.83 s	9.02 s	22.52 s	32.37 s
S	153	0.57 s	4.02 s	6.57 s	11.16 s
T	180	0.43 s	5.48 s	10.18 s	16.09 s
U	180	0.53 s	5.70 s	9.61 s	15.84 s
Y	180	0.72 s	5.93 s	11.37 s	18.02 s

Figure 4.8: The timings for track selection, warping, compositing and total time taken of our automatic algorithm for all our examples. Please look at our accompanying video for the results. Models E,F,T are shot with a mobile phone.

Failures

Our method fails at times when the facial deformation is too large and too fast. Our seam cost is an average of the seam costs across time. Therefore, if a large deformation lasts for only a short period of time, the aggregate seam cost will not change significantly and the composite will be poor as shown in Figure 4.9a as well as our video result.

User Study

The aesthetics of portrait cinemagraphs can be rather subjective as some people find the motion of only parts of the face surreal or uncanny. Therefore, we conducted a user study on Amazon Mechanical Turk to better understand how the general population would respond to portrait cinemagraphs. We showed 30 users 7 sets of results (30 HITs, each HIT has 7 sets). For each set, we showed the still image, the input video and our cinemagraph result side-by-side and asked them to pick their favorite medium to represent each person. We recorded 218 total votes and 80.3% of them were in favor of dynamic portraits (either input video or cinemagraph). 53.0% of the total votes were for cinemagraphs and only 19.7% were for a still image. Our collected votes yield $\chi^2 = 40.5229$ when computed against an expected uniform

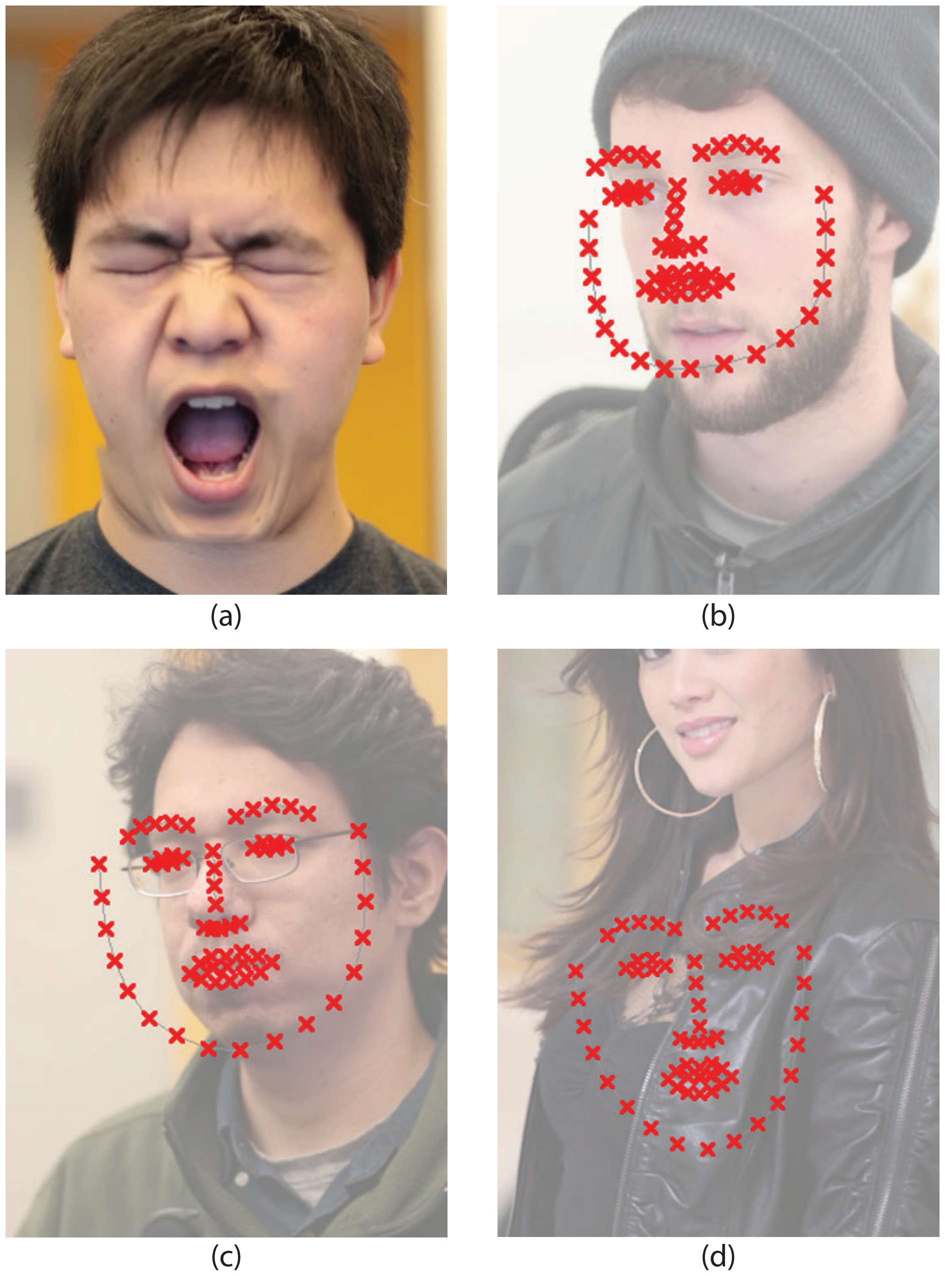


Figure 4.9: The example in (a) has a poor compositing result at the mouth due to the sudden large movement. The average seam cost is dominated by the rest of the video. The facial features detection is bad in examples (b),(c) and (d). While example (c) still produces a good result due to the correct estimates for the mouth and jaw, examples (b) and (d) are failures.

distribution and exceeds the χ^2 value of 13.82 for $p = 0.001$ with 2 degrees of freedom. Therefore, the survey supports our hypothesis that people do in general prefer cinemagraphs for portraits. We also asked the users if they would use an automated technique to generate their own cinemagraphs and 73.3% of them indicated that they are willing to try such a technique.

Limitations

Since our method is dependent on accurate facial feature tracking, we will not always produce good results if the facial feature tracker fails. In the example shown in Figure 4.9b, the jaw is moving but the face tracker fails to accurately detect its location. As a result, the video is stabilized with respect to the jaw causing the head to move up and down in the result video. Our method works in some cases when the facial feature tracker is partially correct as shown in Figure 4.9c but will certainly not work when it misses the face entirely as in Figure 4.9d.

Since we are using a spatially-varying warp to align faces, we cannot handle large rotations of the face, where occlusions as well as depth discontinuities become prominent.

4.7 Conclusions and Future Work

We have presented a fully automated technique for creating portrait cinemagraphs that requires no user input; the user simply captures a hand-held video and receives a portrait cinemagraph. Our technique also runs faster than the system presented in Chapter 3 due to a more efficient MRF optimization and a simpler one-stage warp. The key contribution is a technique for segmenting facial feature tracks into those that are part of facial expressions, and those that track the overall face pose. Also, by defining energy functions unique to our problem, we can automatically composite the warped video to create a portrait cinemagraph.

One avenue for future work would be to automatically detect and composite additional moving regions such as cloth or hair onto the final cinemagraph. These regions are significantly more difficult to handle, especially when the person is moving in the video, because of motion ambiguity between the background, cloth, hair, and rest of the person.

In summary, we believe our automatic technique will empower novice users with the ability to create their own cinemagraph portraits without effort and expertise. As cinemagraph portraits are more expressive than static photographs, we think that they will eventually become a popular alternative to conventional portraits in webpages and social media.

In chapter 3, we presented a system that allows users to de-animate objects and retain motion in desired areas to create novel visualizations and cinemagraphs. However, since the user input required for de-animation can be challenging for novice users, we restricted the class of input videos to only portrait videos. This restriction allows us to design a fully automatic system to generate portrait cinemagraphs with similar quality and up to 60 times faster in computation time compared to the user driven approach, detailed in this chapter.

In the next chapter, we explore how user input can be used to improve video stabilization results.

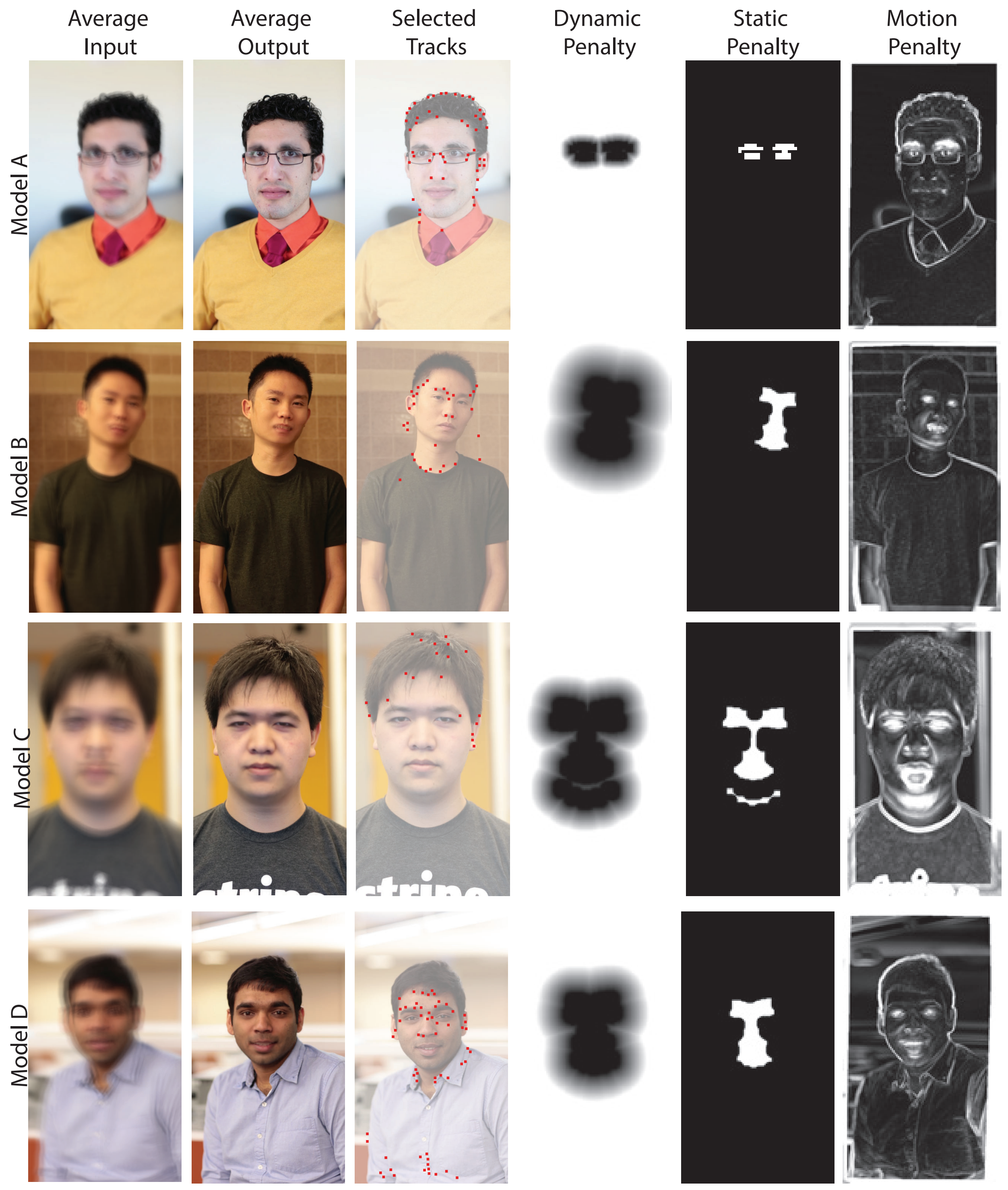


Figure 4.10: Notice that the average input images are blurry as there is camera shake, while the average output image is only blurry at facial features with movements. The automatically selected tracks lie on areas with no motion. The automatically generated penalties help to guide the composite to have motion only at moving facial parts and compositing seams should lie at regions with minimal appearance changes.

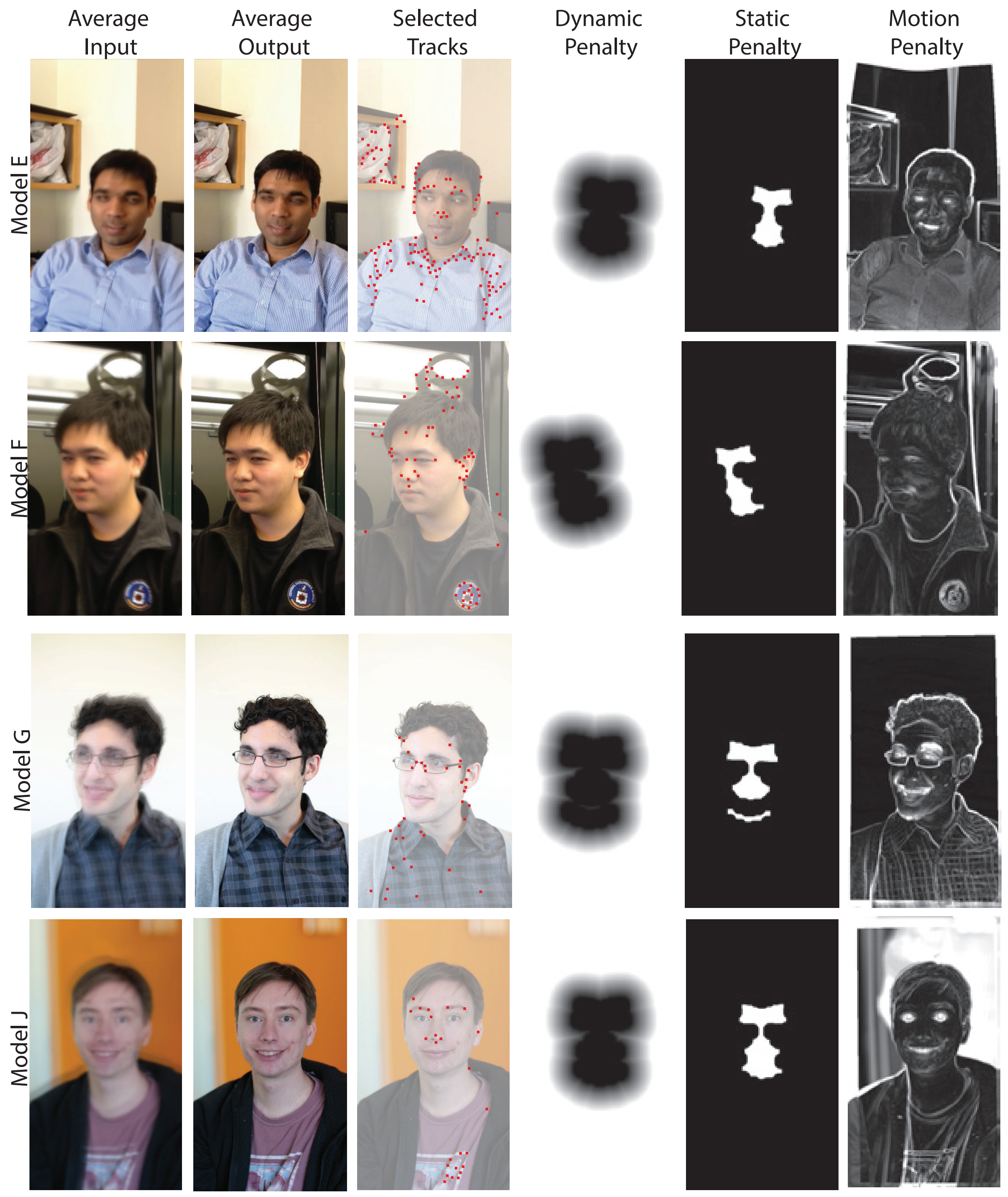


Figure 4.11: Notice that the average input images are blurry as there is camera shake, while the average output image is only blurry at facial features with movements. The automatically selected tracks lie on areas with no motion. The automatically generated penalties help to guide the composite to have motion only at moving facial parts and compositing seams should lie at regions with minimal appearance changes.

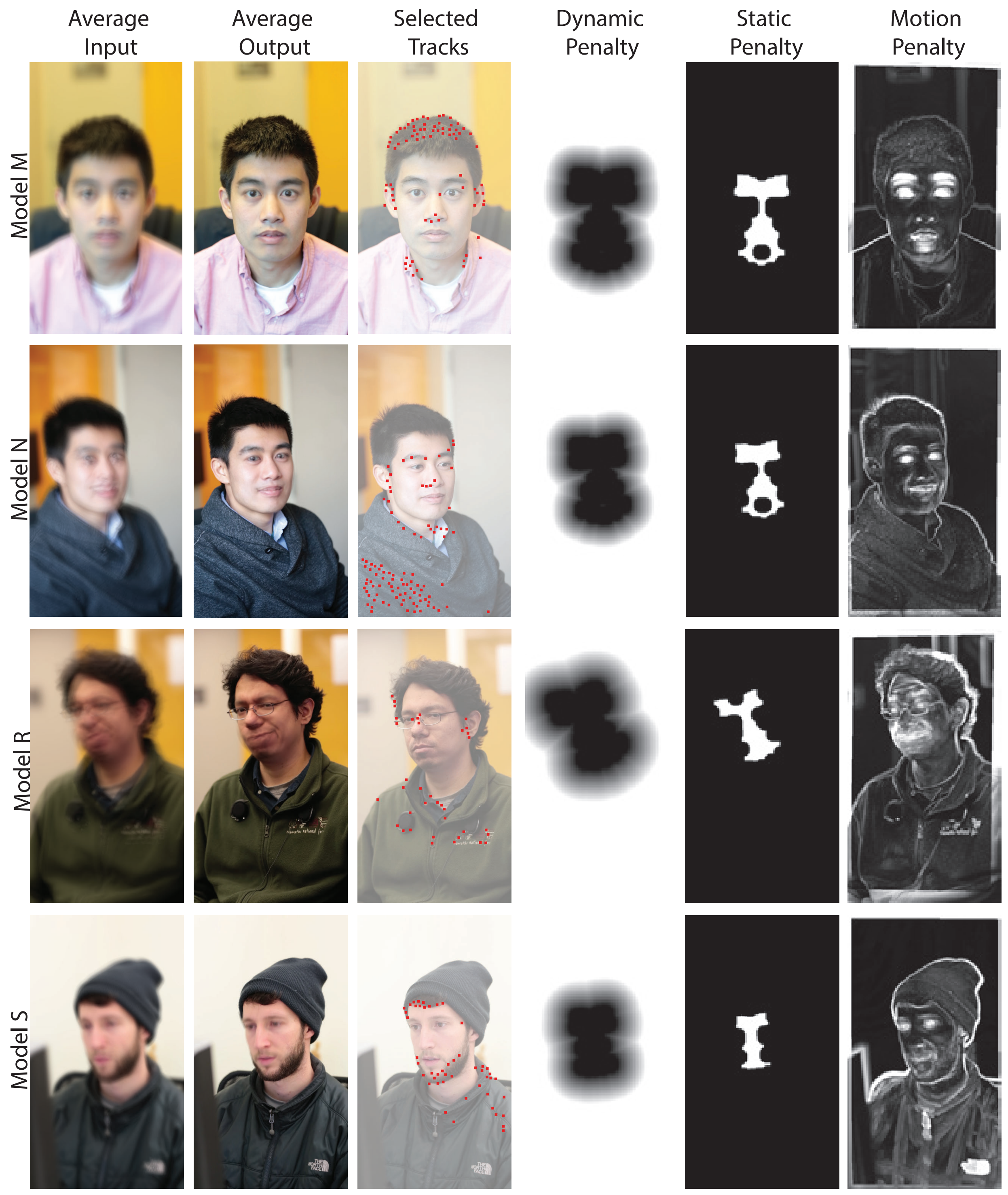


Figure 4.12: Notice that the average input images are blurry as there is camera shake, while the average output image is only blurry at facial features with movements. The automatically selected tracks lie on areas with no motion. The automatically generated penalties help to guide the composite to have motion only at moving facial parts and compositing seams should lie at regions with minimal appearance changes.

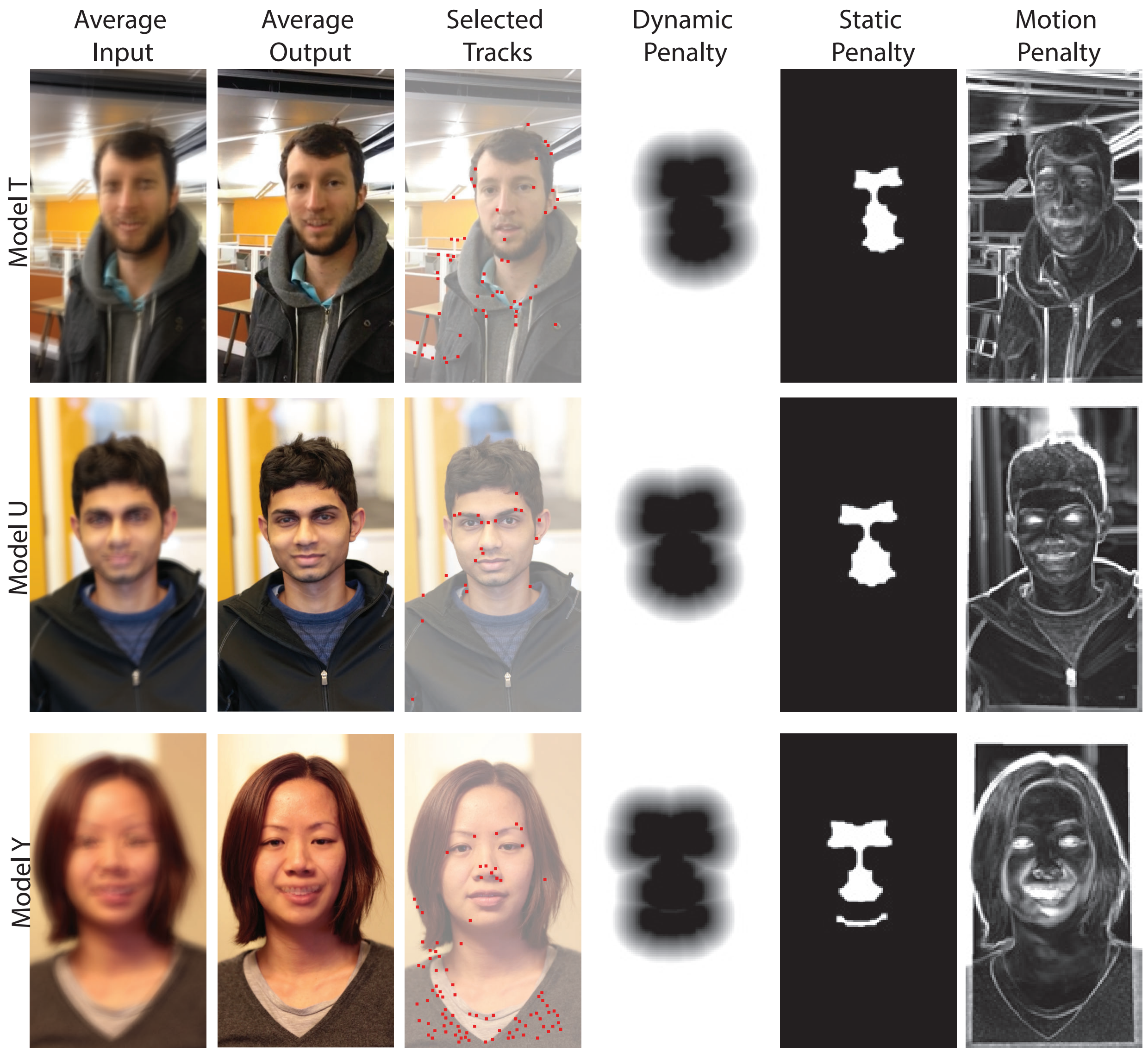


Figure 4.13: Notice that the average input images are blurry as there is camera shake, while the average output image is only blurry at facial features with movements. The automatically selected tracks lie on areas with no motion. The automatically generated penalties help to guide the composite to have motion only at moving facial parts and compositing seams should lie at regions with minimal appearance changes.

Chapter 5

User-Assisted Video Stabilization

5.1 Introduction

In chapter 3 and 4, we introduce new video editing algorithms that isolate finer-scale motions on objects and composites them onto still regions to form compelling visualizations or cinemagraphs. Our algorithm is able to generate these high quality results because we design them to use information regarding the input video's content either obtained from user input or computed using computer vision. In this chapter, apply the same insight to the problem of video stabilization. The goal is to improve automatic video stabilization results with user provided information about the video's content and to enable user control of the stabilized result.

Video stabilization algorithms have improved dramatically in recent years, and can simulate remarkably smooth camera paths. Recent algorithms are 3D-aware and go beyond single-homography motion models to produce much higher quality results. However, these algorithms all have a significant problem; they are one-button press, and offer no user control other than manipulating global parameters. If the stabilization result is flawed, or the user simply wants something else, there is no recourse. We introduce two techniques for interactive control of video stabilization.

The first step in current video stabilization methods is to track feature points that estimate scene and camera motion. The video is then stabilized by warping the frames such that the key feature points follow a smooth trajectory. However, challenging scenes such as those containing a large range of depths and dynamic objects are difficult to stabilize. One reason for failure is that selecting tracks that belong to the background becomes difficult for scenes with dynamic content. If the algorithm uses tracks on dynamic objects (Figure 5.1(c)) to estimate camera motion and warp the frames, the output video may be unstable. Another reason for failure is that the stabilized video is computed without any knowledge of the scene's semantic content. As a result, regular video stabilization might not yield a desirable visual appearance. For example, vertical structures in the stabilized scene shown in Figure 5.1(a) are tilted and a viewer might prefer these structures to remain vertical, as shown in our corrected version in Figure 5.1(b). Our work directly addresses these two issues by providing two tools for the user to customize the stabilization result. Our algorithm starts with an initial baseline automatic video stabilization result [77] which the user can further improve using our tools. First, the user can improve track selection to ensure that only background tracks are selected. We cluster tracks spatio-temporally across the video so that the user can interact with tracks at the granularity of whole objects and motions, rather than individual tracks. We then plot clustered track trajectories on the warped video to allow the user to visually inspect and correct the track selection if necessary as demonstrated in Figure 5.1(c-d). Our clustering improves upon the previous clustering techniques by using a moving window factorization which speeds up computation by 12 times while maintaining similar clustering properties.

Second, we allow the user to provide feedback to the algorithm describing how parts of the stabilized video should look. The user corrects regions at unsatisfactory frames by drawing quadrilaterals on the frame and providing her preferred location for the quadrilateral in the final stabilized video as shown in Figure 5.1(e-f).

After the user has provided edits to improve the stabilized video to her liking, we rewarp the video only using tracks that she has selected and constrain the output video to incorporate the region edits from the quadrilaterals that she has specified. The resulting video exhibits smooth motion for the selected tracks while also respecting the user-specified region constraints.

Our work is the first *user-assisted* video stabilization pipeline which allows the user to influence the video stabilization algorithm. In some cases, fixing user-specified artifacts may lead to new artifacts; however, for most examples we find there to be a net improvement in stabilization quality.

5.2 Background

Bundled paths [77] is a state-of-the-art video stabilization algorithm which performs well with challenging videos. The algorithm divides each frame into a 16 by 16 grid and estimates camera paths throughout time for each cell. The stabilized output video is obtained by smoothing the individual paths both temporally and spatially. We use bundled paths as the baseline in our system; the user will provide inputs (Sections 5.3 and 5.4) to further improve the stabilized video. This section describes the bundled paths algorithm.

Content-preserving warps [74] guided by SURF tracks [17] are used to align neighboring frames t and $t{+}1$. The homography which transforms grid cell i from frame t to $t{+}1$ is $F_i(t)$ as illustrated in Figure 5.2(a). The paths are parameterized by the transformations of the grid cells over time. Therefore, the camera path $C_i(t)$ for each grid cell is defined as the sequence of homographies the cell goes through from the start of the video to frame t. Mathematically, the camera path $C_i(t)$ for a grid cell i at frame t is $C_i(t) = F_i(t-1)\cdots F_i(1)F_i(0)$.

The stabilized camera path $P_i(t)$ is derived from the camera path $C_i(t)$ (Figure 5.2(b)). The algorithm computes $P_i(t)$ by optimizing an energy function $O(P(t))$such that camera

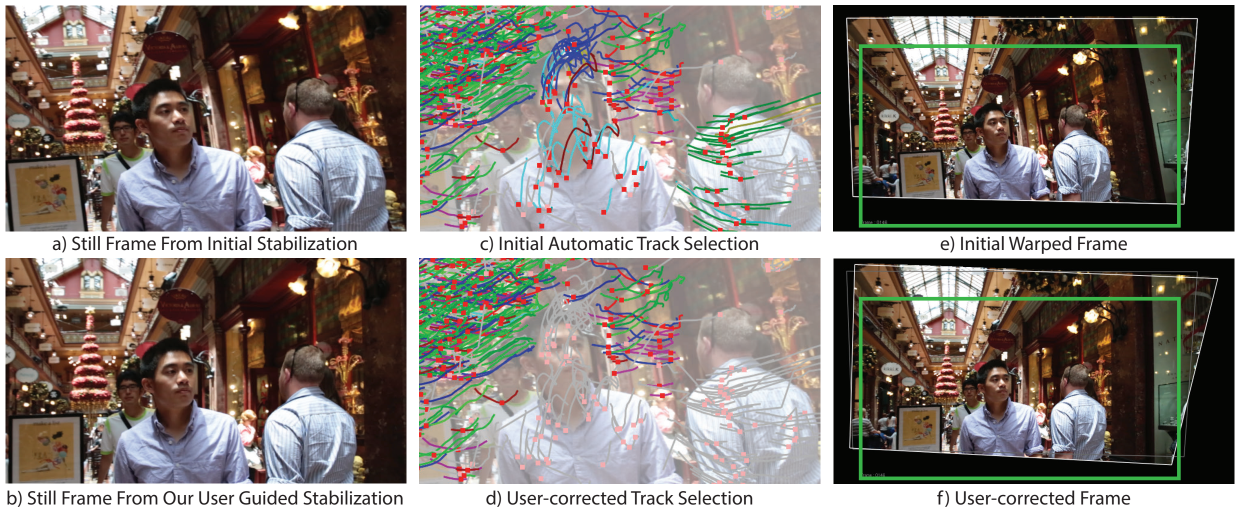

a) Still Frame From Initial Stabilization
c) Initial Automatic Track Selection
e) Initial Warped Frame
b) Still Frame From Our User Guided Stabilization
d) User-corrected Track Selection
f) User-corrected Frame

Figure 5.1: Automatic video stabilization using the state-of-the-art is unsatisfactory as shown in a) as the background and subjects are heavily skewed. We visualize clusters of tracks used for stabilization c) and the user removes tracks on dynamic objects d) using mouse clicks. Tracks that are not used for the final rewarp are drawn in grey. The green outline in e) and f) shows the original frame boundaries. The distortion of the frame in e) is removed by having the user draw a quadrilateral (white lines) and its desired transformation shown in f). The new track selection and user-drawn transformations are used to re-stabilize the video to obtain the final result as shown in b). Notice that the background is rectified and that the subjects are no longer distorted.

path for each grid cell i is smooth temporally and is similar to the path of its neighboring grid cells,

$$\begin{aligned} O(P(t)) &= \sum_t (||P(t) - C(t)||^2) \\ &+ \sum_t (\lambda_t \cdot \sum_{r \in \Omega_t} \omega_{t,r}(C) \cdot ||P(t) - P(r)||^2) \\ &+ \sum_t \sum_{j \in N(i)} ||P_i(t) - P_j(t)||^2 \end{aligned} \tag{5.1}$$

The first term enforces that the smooth path is similar to the original path to minimize cropping and distortion, the second term enforces smooth temporal changes, and the third term enforces spatial smoothness among the grid cells. The temporal window for smoothing, Ω_t, is set to 60 frames and $N(i)$ includes the 8 neighboring grid cells. The strength of the smoothing is controlled by λ_t. The smoothing kernel $\omega_{t,r}$ gives higher weight to paths with temporal proximity and similar transformations. It is set to the product of two Gaussians; the first Gaussian is a function of the frame distance and the second is a function of the difference in translation coefficients of the camera path $C_i(t)$. A Jacobi-based iterative solver is used to solve for the optimal camera path $P(t)$. For more details of the algorithm, please refer to the original paper [77]. We used our own implementation of the algorithm for our

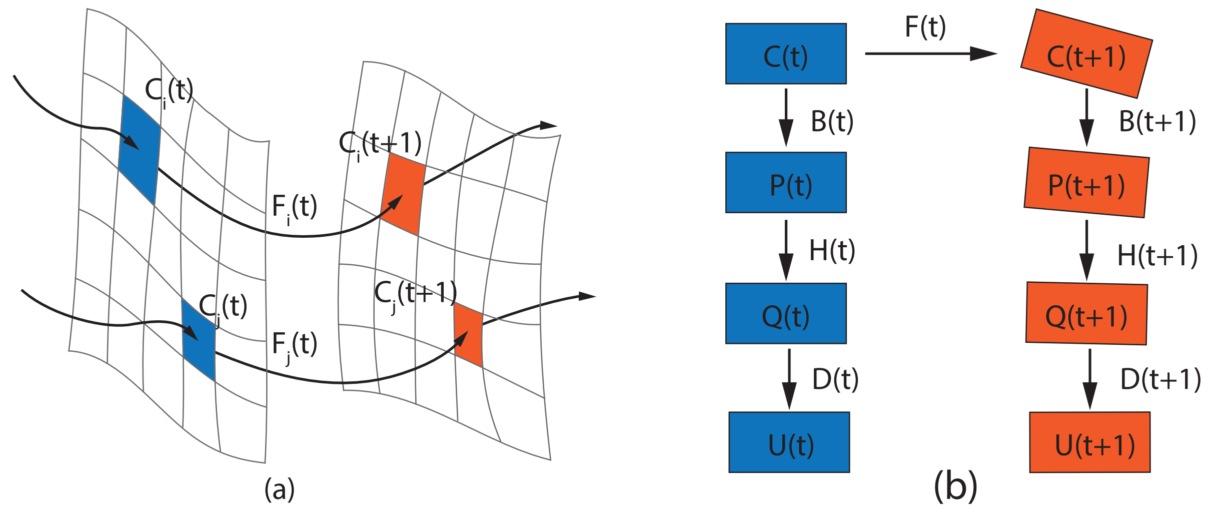


Figure 5.2: Homographies $F_i(t)$ computed between grid cells $C_i(t)$ and $C_i(t+1)$ illustrated in (a). Relationships between the computed camera path $C(t)$, the smoothed camera path $P(t)$, the user-specified path $Q(t)$ and the final user-assisted stabilized video $U(t)$ are shown in (b).

work.

The output video is obtained by applying the transformation $B(t)$ to the input video, which is computed as $B(t) = P(t)C^{-1}(t)$.

Our Work

We use the smooth camera path $P(t)$ as the baseline for our method, but do not assume that it is satisfactory. The user provides a set of constraints on which tracks are selected (section 5.3) and how regions in the video should look after stabilization (section 5.4). Region constraints are represented by $H_k(t)$, where some region in frame t is transformed with the homography $H_k(t)$ to obtain the user-specified path $Q(t)$. If no constraints are specified, then $H_k(t)$ is the identify transformation. Since the user does not specify changes to every frame, $Q(t)$ needs to be smoothed. We compute the final user-assisted stabilized video $U(t)$ by minimizing an energy function described in equation 5.5 in section 5.4. The transformation $D(t)$ is used to correct $Q(t)$ to get the $U(t)$ and it is computed as $D(t) = U(t)Q^{-1}(t)$. Figure 5.2(b) illustrates the relationship between the different paths.

5.3 Improving Track Selection

Selecting the appropriate set of tracks to guide the stabilization algorithm is crucial to the success of the stabilized video. Tracks that accurately track key feature points should be

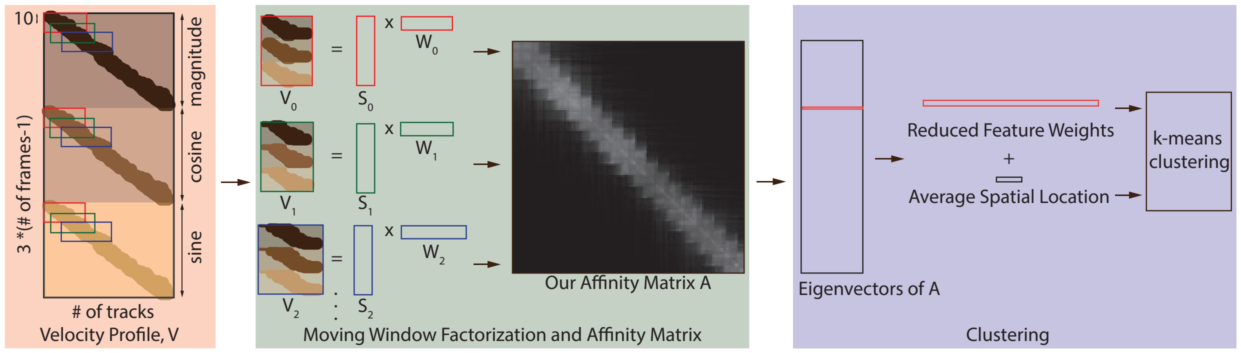


Figure 5.3: There are three steps in our track clustering algorithm. The velocity profile V is computed using the magnitude and direction of the tracks. It is sparse as the tracks are short. Next, we find sub-matrices V_i by selecting only velocity profiles that exist in a window of 20 frames. The step size for the window is 10, which gives us overlapping windows and therefore dependent motion bases S_i. The affinity matrix A is computed at this step using the weights in W_i. After the affinity matrix is computed, we project the affinity vector for each track onto the 20 largest eigenvectors of A. We append the average spatial location of each track to its feature vector for k-means clustering.

selected. In most cases, these key points should lie on a stationary background, but for specialized uses such as de-animation (Chapter 3), these key points may lie on a subject of interest.

Most video stabilization algorithms automatically prune tracks using RANSAC [42] to remove outlier tracks by fitting homographies between frames. While this approach gives acceptable results for simple scenes, selecting appropriate tracks for scenes with multiple moving objects can be difficult for an automatic algorithm.

Our solution is to have the user improve track selection by clicking on tracks to include or exclude when they are overlaid on the video. Note that we also automatically prune tracks using the technique described in Bundled Paths [77]. We cluster tracks based on affinities of their trajectories and average spatial location across time to alleviate the tedious task of selecting individual tracks for the user. A full cluster can be selected or de-selected with a single mouse click. Since clusters of tracks usually belong to the same object or region, the user can quickly ensure that selected tracks are now properly tracking the background. Tracks that should not be selected often have erratic trajectories in the warped video as their trajectories are not well explained by the camera motion, or they lie on dynamic objects (Figure 5.4).

Track Clustering

Track trajectories are usually short for handheld videos due to occlusions and dynamic scene content. Non-negative matrix factorization (NMF) [26] seeks a low dimensional motion basis that explains the speed and direction of the track trajectories for the entire video while handling incomplete track trajectories. However, a full factorization of all the incomplete track trajectories can take up to 6 minutes for a 10 second video. For our user-in-the-loop approach it is crucial that the factorization runs within about a minute on standard consumer hardware.

There are three main parts to our approach for track clustering. First, we construct the velocity profile V which stores the magnitude and directions for each track k over all frames. Second, we compute an affinity matrix A which measures the similarity between track trajectories using a moving non-negative matrix factorization that is faster than factoring V directly. Finally, we reduce the dimension of the affinity vector for each track by projecting it onto the eigenvectors of A. We cluster the tracks using the reduced feature vectors and their average spatial location with k-means. Figure 5.3 illustrates the three steps of our approach, and we consider each one in turn.

Velocity Profile

The first step is to compute the velocity profile V. If the video has T frames and K total tracks, then matrix V is size $3(T-1) \times K$. The velocity profile captures the instantaneous magnitude and direction of each track k. The k^{th} column of V is:

$$V^k = [m_1^k, \cdots, m_{T-1}^k, c_1^k, \cdots, c_{T-1}^k, s_1^k, \cdots, s_{T-1}^k]^T \tag{5.2}$$

where m_t^k is the magnitude of the velocity of the k^{th} track from frame t to $t+1$, while c_t^k and s_t^k are the cosine and sine of the angle of the k^{th} track. We add 1 to c_t^k and s_t^k to make the entries of V positive and we enter zeros into V to handle missing data at any particular frame. Figure 5.3 shows the structure of a typical velocity profile V.

Moving Window Factorization and Affinity Matrix

Non-negative matrix factorization seeks to factor V into its motion basis S and the corresponding weights W. The affinity matrix A of size $K \times K$ measures the similarity between all tracks in the video using their weights computed in W. However, the velocity profile is usually sparse because tracks in handheld videos are short as seen in Figure 5.3. Therefore the computation can be sped up by first breaking V into windows and factoring each window separately.

The entries of sub-matrix V_i correspond to velocity profiles of tracks that exist from frame $i * b$ to $(i+2) * b$, where b is the step size. For our application, we set b to 10. In other words, our window size is 20 and the windows overlap by 10 frames as we move the window. To create the windowed V_i, we simply select the appropriate rows from V (Figure 5.3) and discard any empty columns. We factor each V_i into its motion basis S_i and the corresponding weights W_i. The overlapping windows help to maintain coherence in the estimated motion basis S_i. We compute affinities between tracks in each sub-matrix V_i using W_i and update the overall affinity matrix A accordingly, as described later in equation 5.4.

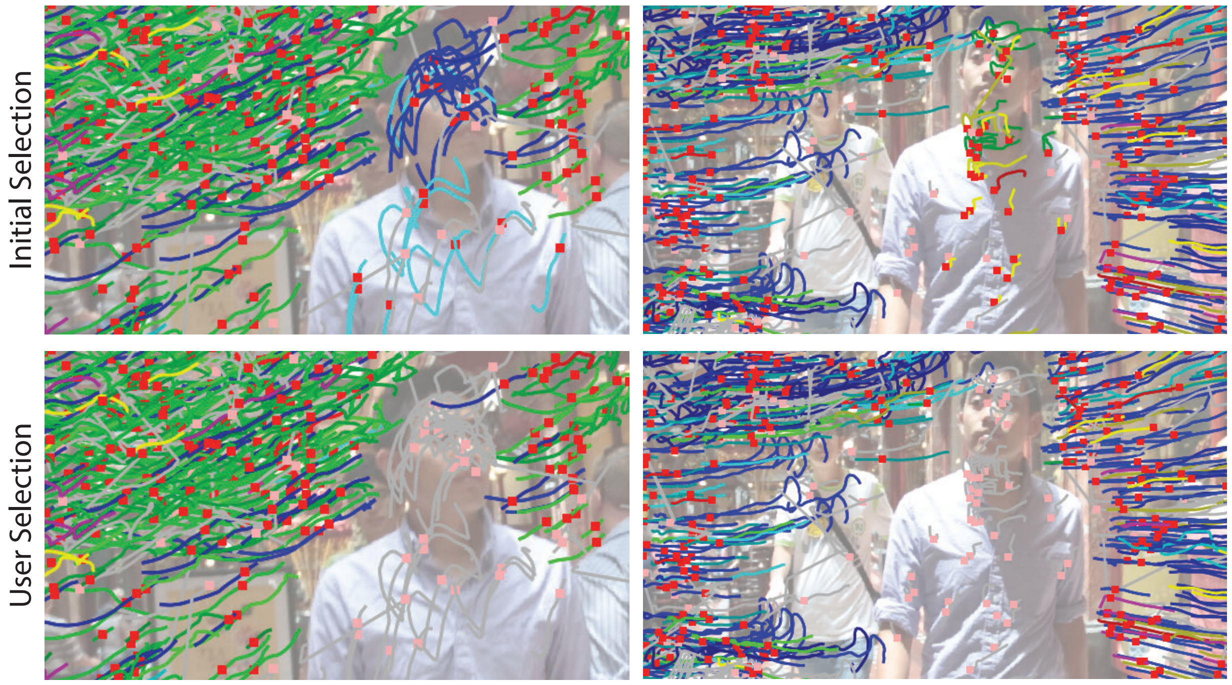


Figure 5.4: We cluster and visualize the selected tracks. Each cluster of tracks is assigned a bright color and their trajectories are overlaid on the warped video. Notice how tracks on dynamic objects can have an erratic appearance. The user then removes some clusters by clicking on the clusters to toggle them off. Tracks that are not used for the next warp are shown in light grey.

We seek to factorize V_i into non-negative matrices S_i $(6b \times r)$ and W_i $(r \times \hat{K})$ that minimize the following expression:

$$||V_i - S_i W_i||^2 \tag{5.3}$$

The matrix S_i is the motion basis for the window, and W_i contains the non-negative weights for constructing the trajectory of the tracks using the basis S_i. $\hat{K}$ is the number of tracks that exist in the window. We use alternating least-squares [45] to find S_i and W_i using the the algorithm described by Cheriyadat et al. [26]. We set the number of terms r to be 6 for our purposes as we expect a sparse set of motion basis vectors will be sufficient to represent the motions within a short window.

We can compute the affinity matrix A after the matrices W_i are computed. Let $\Phi_{q,s}$ be the set of indices of the matrices W_i that contain track q and s. Also, let $w_i(q)$ be the column of W_i containing the weights of the q^{th} track in the i^{th} window. The affinity between two tracks q and s is computed as follows:

$$A_{qs} = \frac{1}{|\Phi_{q,s}|} \sum_{i \in \Phi_{q,s}} \exp\left(\frac{-||w_i(q) - w_i(s)||^2}{\sigma}\right) \tag{5.4}$$

In other words, the affinity score between two tracks is the average affinity score computed over the windows where both tracks exist. We set σ to 5.0.

Our approach does not gives us the same affinity matrix as that obtained from a direct factorization of V, without breaking into windows. In particular, in our approach, tracks which do not exist on the same frame will have an affinity score of 0. This implicitly enforces tracks which do not overlap temporally to be clustered into different clusters. We compare and discuss our clustering results and performance using our approach against direct factorization in the results section.

Clustering

The q^{th} row of the affinity matrix is the affinity vector for the q^{th} track. Instead of using the affinity vector to cluster the tracks directly, we reduce the dimension of the affinity vector by projecting it onto the 20 largest eigenvectors of A to reduce computation costs for clustering. In practice, the reduced feature vector is the row of the set of eigenvectors corresponding to each track. This reduced feature vector for track q is combined with the average spatial location of the track across time to form a length 22 vector. (The average spatial location is normalized by the width of the video and multiplied by 15.) We use k-means clustering with the number of clusters set to be the number of frames T. Examples for our track clustering are shown in Figure 5.4.

Track Visualization

Each track is drawn as a red square on the frame with its entire trajectory drawn as a line on the warped video (Figure 5.4). Selected tracks are displayed in bright colors, with each track cluster receiving a unique color. Tracks which are pruned and not used to guide the warp are displayed as pink squares with their trajectory in light grey.

The user interacts with the track clusters by clicking on them when the warped tracks are displayed. We find the closest track cluster to the mouse and highlight it in real-time. The user can toggle the track cluster by clicking on it to either include or exclude the cluster from guiding the warp. Once she is happy with her modifications, we stabilize the video with the user-selected tracks.

Occasionally, the user might wish finer-grain control to make edits within a cluster, or to refine the output of the clustering algorithm. We allow them to remove and turn on/off specific tracks from within a cluster.

5.4 Specifying Region Warps

Videos stabilized with tracks on the background might still be undesirable for a user like the example shown in Figure 5.1(d), where the background structures are tilted. In general, since the baseline stabilization algorithm does not have any semantic information of the content in the video, it might warp the frames in a way that the user finds undesirable. Our solution is to allow the user to directly specify how regions in the frame should deform. We then use the user specifications to guide a new optimization with new temporal weights that propagates the user edits. This new optimization tries to find a stabilized video that is close to the previous stabilized result yet conforms to the user-specified constraints as shown in Figure 5.1(b).

User Interface

The user scrubs through the video to find an offending frame she wishes to correct. We show the entire warped frame without cropping to allow the user to access the extreme edges of the warped frame. A green outline is overlaid to provide a visual cue on the boundaries of the initial frame size as shown in Figure 5.1(e-f).

The user draws a quadrilateral around the region she wishes to correct by clicking on the four corners of the region. The corners of the quadrilateral can be selected and dragged to warp the region. The region selected within the quadrilateral is deformed in real-time to provide feedback for the user as shown in Figure 5.1(f). Arrow keys on the keyboard translate the region to provide additional control. Note that while only one quadrilateral is drawn in Figure 5.1(f) which covers the entire frame, the user can draw multiple small quadrilaterals (which do not cover the entire frame) to make local corrections.

The user can specify how each edited frame will affect neighboring frames. She can choose to propagate the edit to frames both preceding and succeeding the edited frame (the default), or only propagate in one direction.

We introduce a global parameter α with which the user can specify the confidence of her region constraints. If the α value is high, her region constraints have more weight in the optimization routine when solving for a user-stabilized output. Conversely if the α value is low, her region constraints can be relaxed to generate a smoother video. We provide two recommended values for α, 1 and 10 which we found to balance well with the rest of the energy terms.

Stabilizing Video With User-Constraints

After the user is done specifying region constraints using quadrilaterals and deforming them, our system derives the user-specified camera path $Q(t)$. It also propagates the constraints smoothly across time and we achieve this by introducing a window function which modulates the temporal term in the video stabilization energy function. We also introduce a data term that controls which frames are penalized if they deviate from the user-specified path $Q(t)$. The following parts detail how $Q(t)$ is computed, how the energy functions are modified and how the final stabilized video $U(t)$ is computed.

User-Specified Camera Path

The user-specified camera path $Q(t)$ incorporates the region constraints the user has drawn. Therefore, we first identify which grid cells are affected by each region constraint. Grid cells which have more than 50% of their area within the drawn quadrilateral inherit any modifications the user makes to the quadrilateral. The k^{th} deformation the user makes on frame t is represented by a homography $H_k(t)$. We compute $H_k(t)$ by computing the homography that takes the original four corners of the quadrilateral to the corners of the modified quadrilateral.

If the user adds no constraints, then $Q(t) = P(t)$. However, if the user modifies regions by deforming them with the k^{th} quadrilateral, then for grid cell i in frame t, the user-specified camera path is $Q_i(t) = H_k(t)P_i(t)$.

Constraint Propagation

Since the user only specifies region constraints on a sparse set of frames, our algorithm must propagate the region constraints smoothly over time. Each region constraint is linearly interpolated to its neighboring frames as shown in Figure 5.5. The linear interpolation is performed for a maximum of 60 frames centered at the edited frame t, illustrated by the frames after the blue arrow in Figure 5.5.

We embed this interpolation in the original energy function in equation 5.1. We introduce a new window function σ_t which modulates the existing temporal filter $\omega_{t,r}$ in equation 5.1 by replacing $\omega_{t,r}$ with $\sigma_t \cdot \omega_{t,r}$. Our window function σ_t is a linear interpolation weight as shown in Figure 5.5. We set σ_t to 1 for the entire window by default. We use the corresponding linear interpolation weights as illustrated in Figure 5.5 if there are region constraints on frames within the window.

Warping With Constraints

We would like to reoptimize the stabilized video to find a smooth user-assisted stabilized video $U(t)$ which respects the user-specified camera path $Q(t)$. We use a similar energy function $O(U(t))$ to that in equation 5.1 used to find the initial stabilized video,

$$\begin{aligned} O(U(t)) &= \sum_t (\tfrac{1}{5}(1 - E(t))||U(t) - C(t)||^2) \\ &+ \sum_t (5\alpha E(t)||U(t) - Q(t)||^2) \\ &+ \sum_t (\alpha\lambda_t \cdot \sum_{r\in\Omega_t} \sigma_t \cdot \omega_{t,r}(C) \cdot ||U(t) - U(r)||^2) \\ &+ \sum_t \sum_{j\in N(i)} ||U_i(t) - U_j(t)||^2 \end{aligned} \tag{5.5}$$

The energy function we optimize is similar to equation 5.1. We reduce the weight of the first term which enforces the final stabilized video to follow the estimated camera motion $C(t)$. We introduce a new term so that the final stabilized video will follow the user-specified path

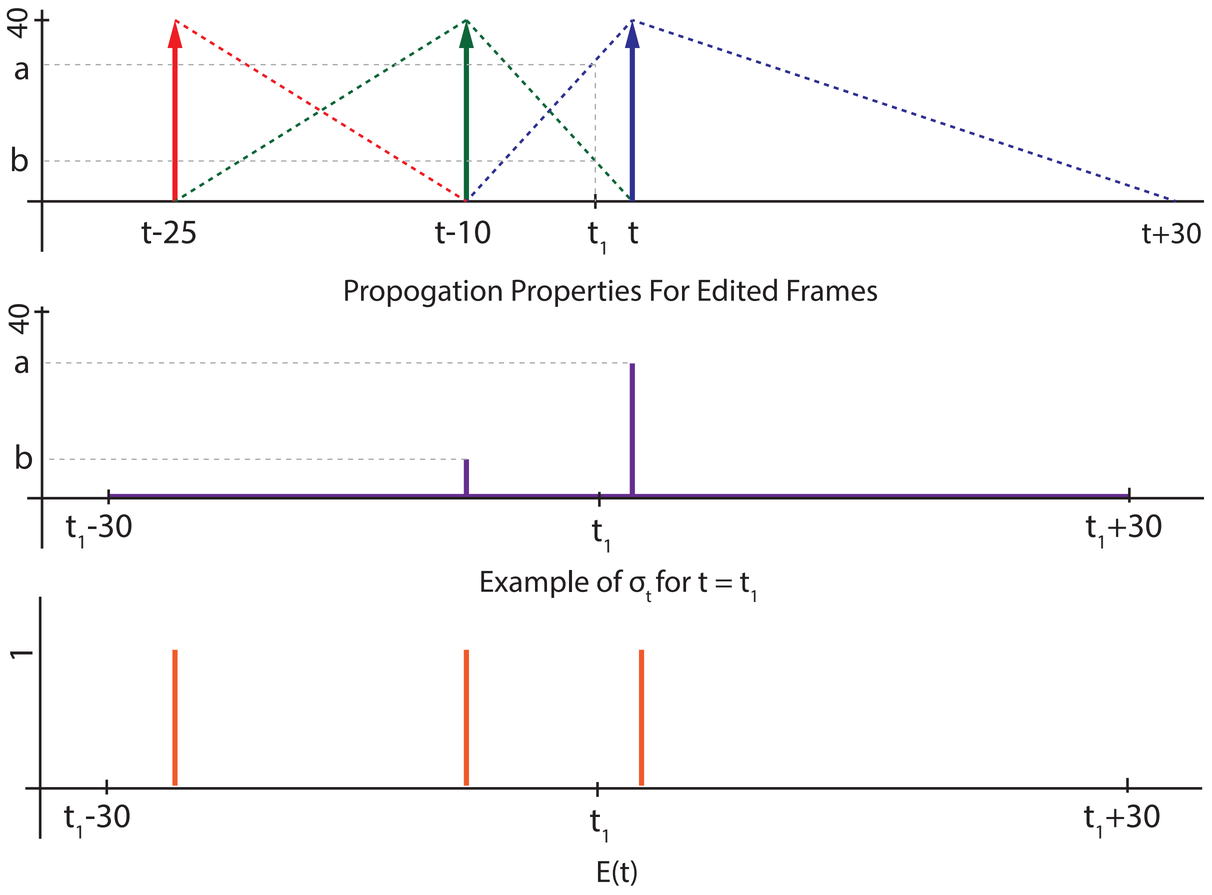
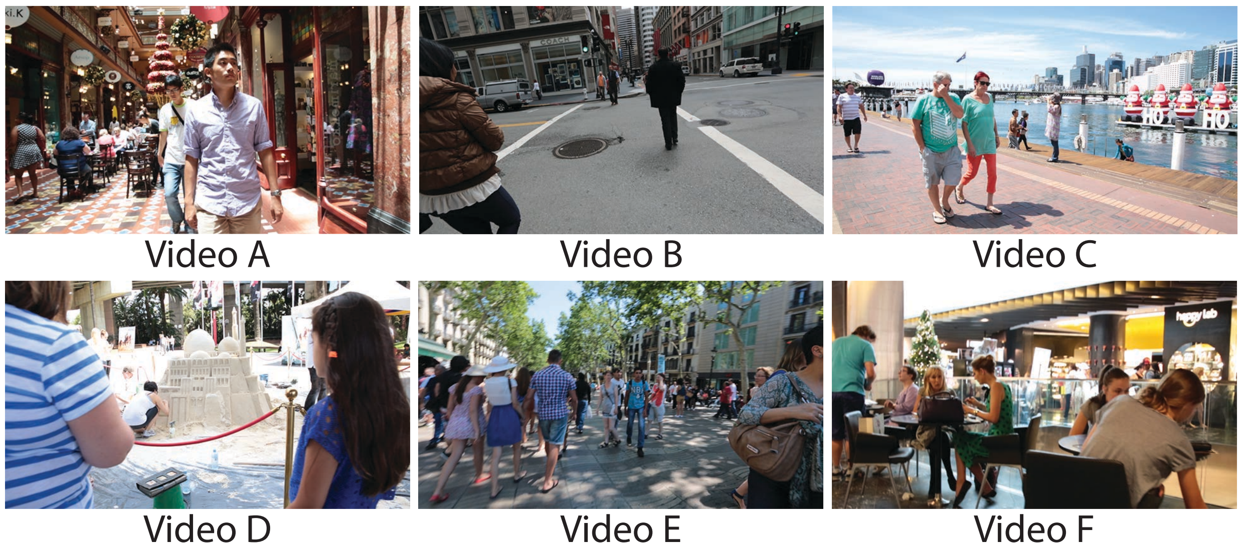


Figure 5.5: **Top**: The user adds three region constraints on frame t, $t-10$ and $t-25$. The dotted lines show how the the region constraints should be linearly propagated. The support of the interpolation is determined by the proximity of other constraints. If there are region constraints that are within 30 frames, such as the red-green and green-blue pair, the linear interpolation is performed within the frames between them. Otherwise, the interpolation is performed for 30 frames like the blue arrow to the right. The user can also specify one-sided propagation like the red arrow on the left. **Middle:** We show an example of our window function σ_t for frame t_1. **Bottom:** The function $E(t)$ controls if the new data term is applied for each frame, allowing frames to follow the user-specified path $Q(t)$ or not.

$Q(t)$. Note that the temporally varying function $E(t)$ controls when the final stabilized video should be similar to $Q(t)$. Also, $\omega_{t,r}$ is replaced with $\sigma_t \cdot \omega_{t,r}$ and with $P(t)$ replaced with $U(t)$. The value α (default value is 10) is the global variable which controls the confidence of the user's region constraints.

We set $E(t)$ to have a value of 0 for all frames initially. For each frame t that the user provides an edit, we set $E(t)$ to 1.



The update rule for the Jacobi-based iterative solver [23] is:

$$\begin{aligned} U_i^{(\xi+1)}(t) &= \tfrac{1}{\gamma'}(\tfrac{1}{5}(1-E(t))C_i(t) + 5\alpha E(t) Q_i(t) \\ &+ \sum_{\substack{r\in\Omega_t \\ r\neq t}} 2\lambda_t \omega_{t,r}\sigma_t U_i^{(\xi)}(t) + \sum_{\substack{j\in Ni \\ j\neq i}} 2U_j^{(\xi)}(t)) \end{aligned} \tag{5.6}$$

where

$$\gamma' = 2\lambda_t\sigma_t \sum_{\substack{r\in\Omega_t \\ r\neq t}} \omega_{t,r} + 2N(i) + \frac{1}{5}(1-E(t)) + 5\alpha E(t)$$

The $(\xi+1)^{th}$ iteration of the user-specifed camera path $U_i^{(\xi+1)}(t)$ is computed with the update rule in equation 5.6. The initial solution $U^{(0)}(t)$ is initialized with $Q(t)$ and we use 15 iterations. The transformation $D(t)$ is used to correct $Q(t)$ and it is computed as $D(t) = U(t)Q^{-1}(t)$.

5.5 Results

We test our system on a collection of very challenging videos and compare our user stabilized result against the other methods as shown above. Specifically, we compare against our baseline [77], YouTube's video stabilizer [50, 51], and Adobe's warp stabilizer [74, 75]. We use a MacBook Pro 2013 2.8 Ghz i7 with 16GB of RAM. Our code is unoptimized in C and Matlab. OpenGL is used to display the video and user interface.

User Input and Performance.

Depending on the complexity of the video, and desired control over the final output video, the user typically takes around 10 minutes to select tracks and provide constraints. Our selection of 6 videos contains severe camera shake with wide field-of-view and large depth range. This makes our videos very challenging for video stabilization algorithms. We remove 29 track clusters and provide 25 region constraints on average for our videos. The region constraints are distributed over the length of the entire video. Our results have smoother camera motion with less background distortion when compared to the other methods. For detailed comparisons, please see the main video and supplementary videos.

It takes 0.67 seconds per frame (1.5fps) to stabilized the video using the initial baseline ($C(t)$ and $P(t)$) and 0.52 seconds per frame (1.9fps) to optimize with user constraints ($U(t)$). Our computation for the affinity matrix is on average 12 times faster than using direct factorization. Our total clustering time on average is 33 seconds. Table 5.1 has the detailed breakdown of the times and edits.

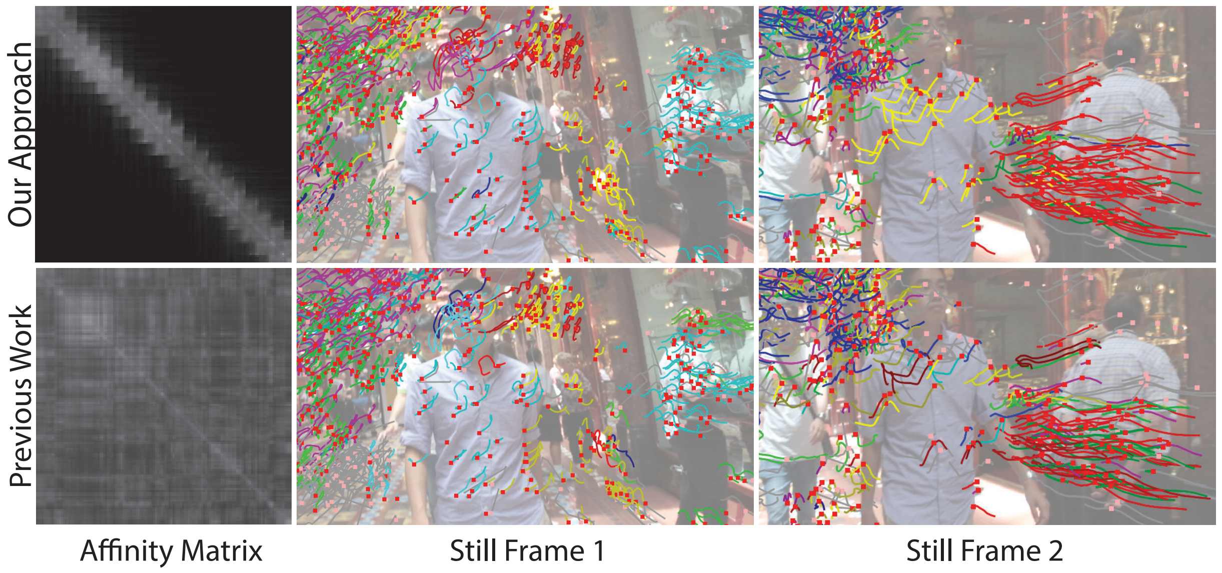


Figure 5.6: We compare the clustering result of our method against using the direct factorization of V. Tracks in the same clusters are drawn with the same color. Notice that while the affinity matrices are different, the quality of the track clusters are similar. In particular, tracks belonging to the dynamic objects are rarely clustered with the background tracks. In still frame 1, our result correctly clusters the tracks on the arm with the tracks on the torso. Also, in still frame 2, our result clusters the tracks on the torso as a single cluster. Our clustering method is also 12 times faster.

ID	# frames	# tracks	$C(t)$	$P(t)$	$U(t)$	Direct A	Our A	Total Clustering	# Cluster Edit	# Region Const.
A	207	16144	38.9 s	102.9s	106.5 s	225.2 s	14.85 s	30.1 s	24	23
B	298	19385	57.6 s	146.1 s	153.9 s	156.0 s	17.78 s	44.6 s	49	23
C	167	11050	28.9 s	79.2 s	84.5 s	86.0 s	10.4 s	19.1 s	10	23
D	203	15001	34.8 s	103.9 s	103.6 s	253.36 s	13.5 s	28.6 s	23	24
E	312	23936	54.6 s	163.1 s	162.9 s	371.3 s	25.4 s	60.5 s	68	36
F	147	10186	23.2 s	73.8 s	73.7 s	60.3 s	8.7 s	15.0 s	2	23

Table 5.1: Our computation times for our examples as well as the number of user edits. Notice that our method for computing the affinity matrix is on average 12 times faster. Our average clustering time is 33 seconds. Computing the baseline video stabilization takes 0.67 seconds per frame (1.5fps). Optimizing for our user-assisted video takes 0.52 seconds per frame (1.9fps). On average the user removes 29 clusters and provides 25 region constraints.

Quality of Track Clustering.

Factoring the velocity profile matrix directly can be computationally intensive since the size of V can be up to 1000×20000 for our examples. Our method clusters small overlapping windows of V_i instead and computes affinities based on local weights W_i from local motion bases S_i as described in Section 5.3. While the affinities computed with the two techniques are different, the clustering results are similar as shown in Figure 5.6.

Notice that in both methods, the tracks on dynamic objects are rarely clustered with the background tracks. In still frame 1, our method clusters tracks on the subject's arm on the right with the torso while the previous work clusters them with the background. In still frame 2, our method successfully clusters tracks on the torso as a single cluster.

In addition, our approach assigns tracks that do not exist in the same frame an affinity score of 0. As a result, the likelihood of tracks that do not exist on the same frame being assigned the same cluster is much lower than the previous approach.

Quality of Stabilization.

There are two ways the user can improve video stabilization in our framework. First by improving track selection, and second, by adding additional region constraints. In Figure 5.7, the use of user-selected tracks reduces distortion in both examples. In the top example, the horizontal stretch of the subject is reduced after using the appropriate tracks for stabilization. In the bottom example, the walls of the building on the left, after using user-selected tracks, have less skew than the initial baseline. For video comparisons please see the supplementary videos.

When proper track selection is not sufficient to improve the stabilization results, the user can correct distortions directly as shown in the example in Figure 5.8. The user drawn constraints are on the right and are applied to frame 150. Notice that after stabilizing the video with the user constraints, the neighboring frames inherit the user modifications properly. However, in some cases, slight wobbles are introduced in the video after correcting for distortions.

In Figure 5.9, we compare video stills from the baseline stabilization against corresponding frames after the user-specified constraints are used to stabilize the video. Notice that in both examples, the stabilized video with user-constraints has milder distortions as vertical structures in the scene remain vertical. In Figure 5.10- 5.15, we show selected still frames to compare the quality of stabilization between the baseline stabilization and our final result. Notice that for the examples, our result has upright facades and subjects are less distorted. For better comparison for all results, please see our main video and supplementary videos.

Since the quality of the stabilized video is strongly correlated with the user constraints, the more constraints the user provides across the frames, the better the stabilized video will be. Our results are generated with a set of user constraints across the video to correct for distortion and camera paths. Note that if the user only wishes to fix major artifacts in isolated regions, she only would have to provide constraints in those regions.

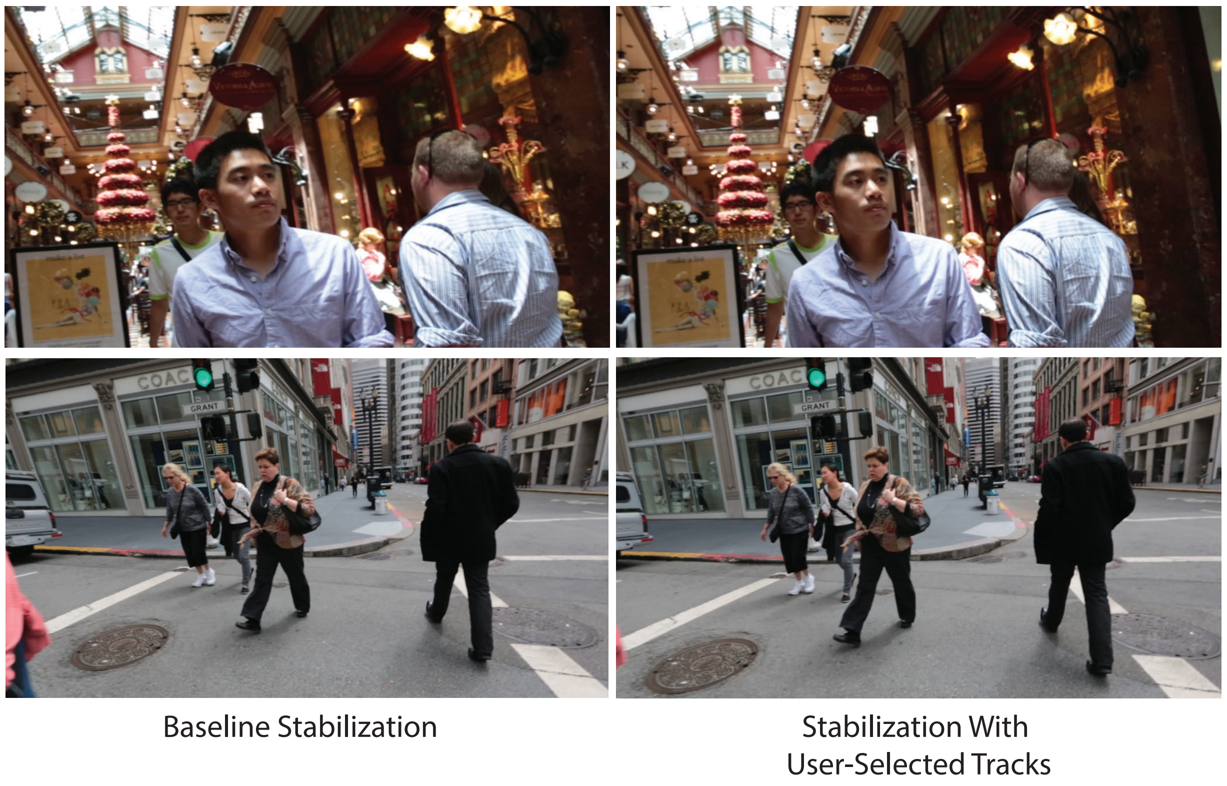


Figure 5.7: Comparison of initial baseline with result using user-selected tracks. The face and torso of the subject have less horizontal stretch in the top example. The walls of the buildings on the left are less skewed in the bottom example.

Limitations.

While our method allows the user to have control over how the stabilized video will look, our system will not generate good results if the quality of the user constraints is bad. In particular, if the user constraints significantly deviate from the camera path and are in conflict, the output video will be jerky. Our system does not infer the intent of the user. For example, if the user wants to level the horizon in the stabilized video, she would still have to manually correct regularly spaced frames as opposed to just specifying her intent. However, there is also a drawback to our method; in some cases, new artifacts are introduced in the effort to reduce existing artifacts.

5.6 Conclusion and Future Work

We have demonstrated that our system can improve video stabilization by allowing the user to customize the result to her preferences. The two modes of interactions, track selection and

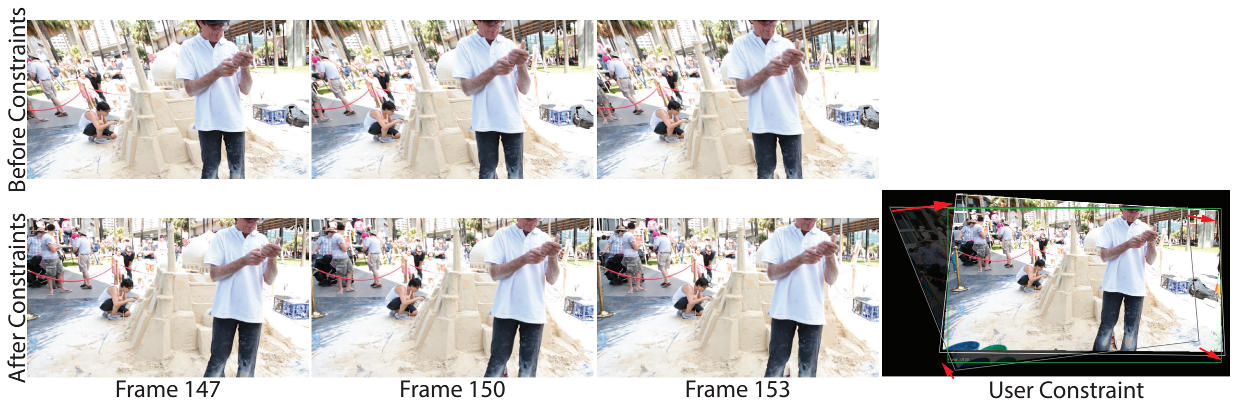


Figure 5.8: Before region constraints were applied (top row), the frame is skewed. The region constraint is drawn on frame 150 and it corrects for this deformation. The red arrows show the change in the quadrilateral drawn in white for the region constraint. After the video is stabilized with the constraints, note that the neighboring frames are corrected appropriately as the background is upright and there are fewer distortions.

region specification, directly relate to how automatic stabilization algorithms work. With a set of correctly selected tracks and region constraints, the output video exhibits smoother motion while respecting the user's intent.

One avenue for future work is to explore how more tools can be used for improving stabilization. For example, higher level controls such as automatic vertical and horizontal rectification could be explored to minimize the user interaction needed. Similarly, the user could also potentially specify constraints by providing exemplar frames and the algorithm could compute the transformations necessary to match scene properties such as vanishing points. Finally, there is definitely room for improvement as our technique might introduce unwanted artifacts such as small wobbles in the final result.

In this chapter, we have presented a new end-to-end system that allows users to improve automatic video stabilization algorithms. While our system still has artifacts in the output video, we believe that this is a first step towards video stabilization algorithms that leverage user guidance and expertise to solve for a stabilized output, beyond the capabilities of fully automatic methods.

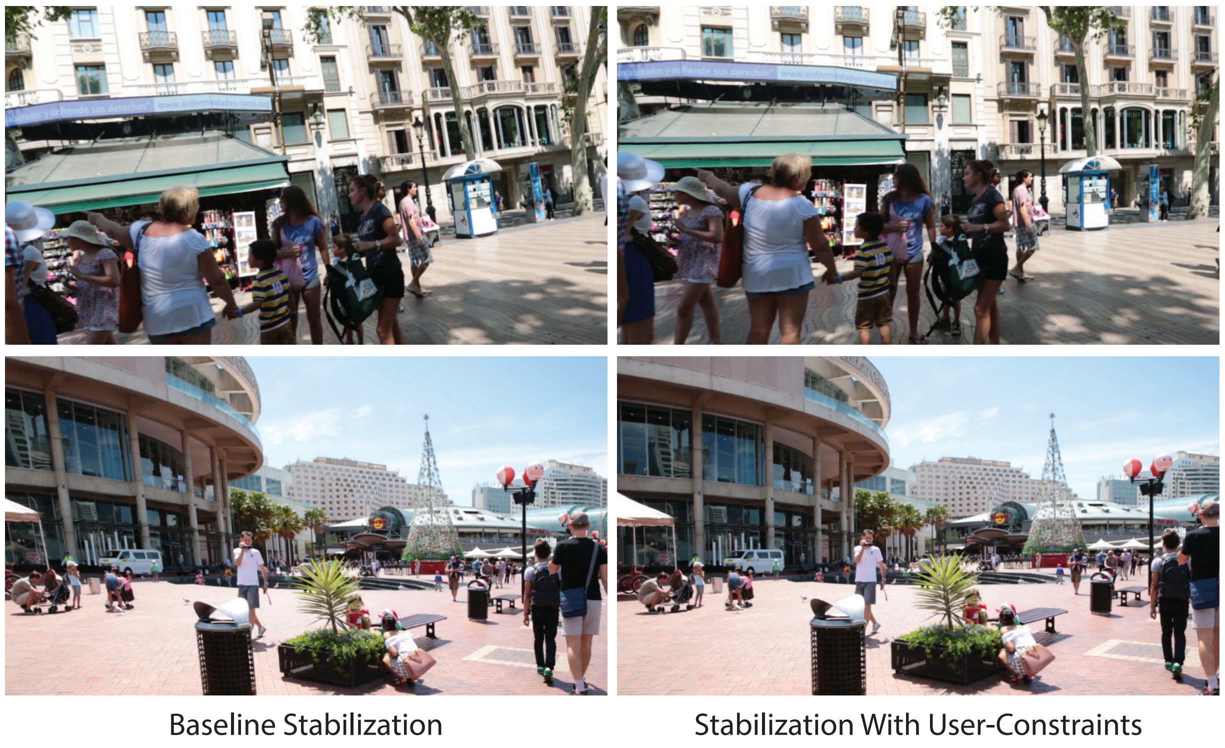


Figure 5.9: Comparision of initial baseline with result using user constraints. In all examples on the left, the baseline stabilization is not satisfactory as the scene is distorted and the horizon is not level. The same frames in our result are less distorted as the structures are vertical.

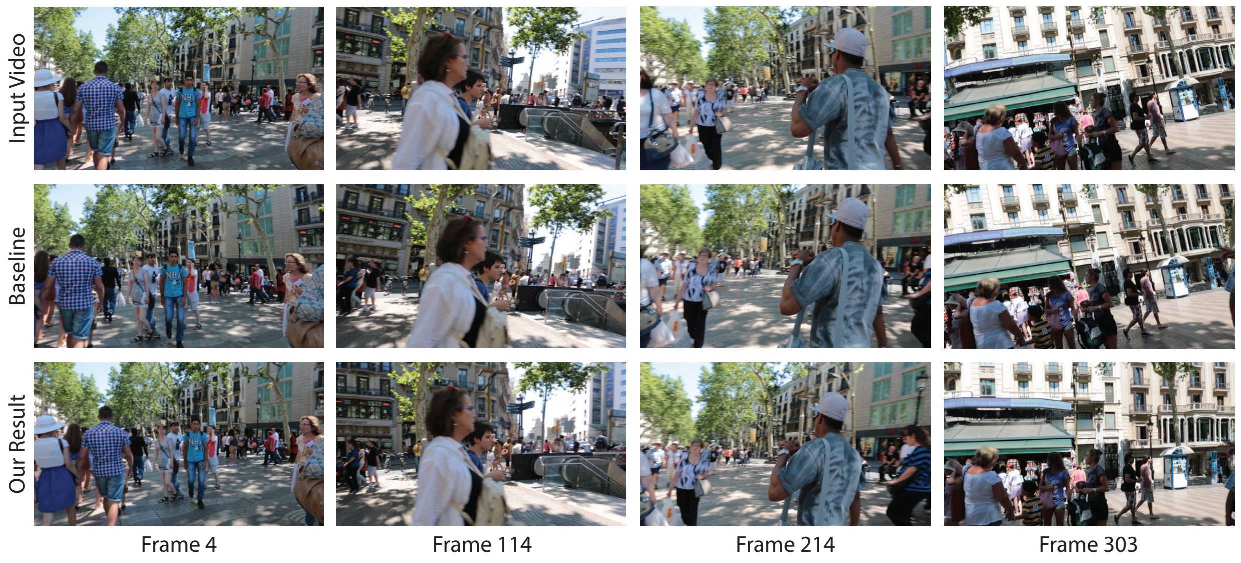


Figure 5.10: Notice that in our result, frame 4, 114 and 214 have facades that are more vertical compared to the baseline stabilization. In particular, the baseline stabilization did not correct for the slow rotation for the camera in frame 303 while our method corrects for it as the user specified that the rotation should be removed.

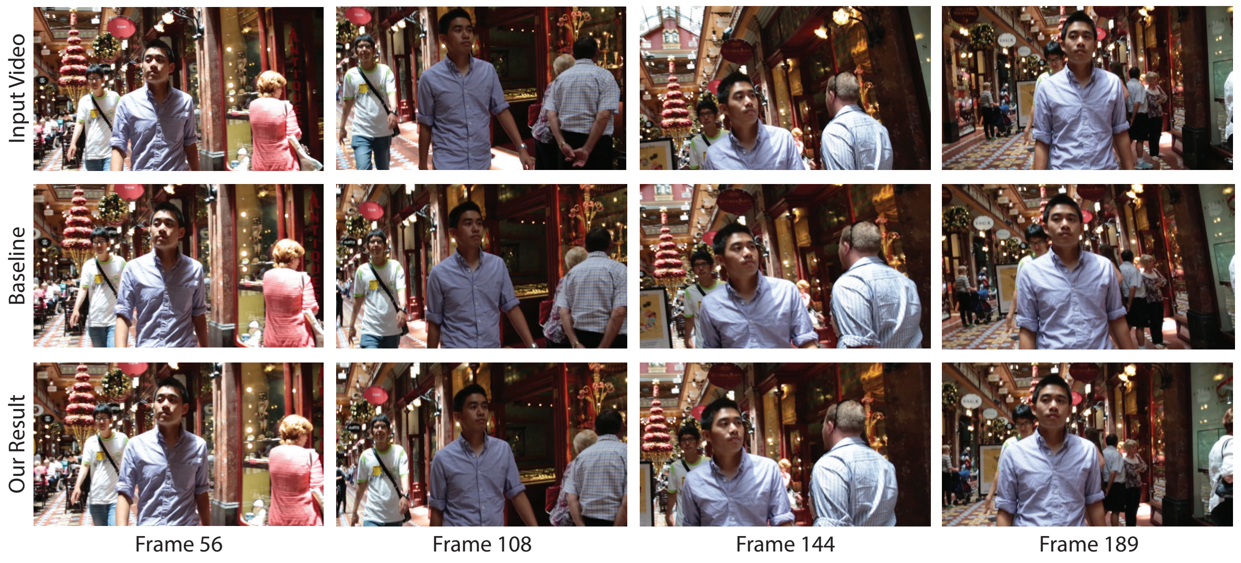


Figure 5.11: In this example, the baseline stabilization skews the subject towards the end of the video as tracks from other moving people are selected. In our result, the skew of the subject is drastically reduced. Notice that the facades are vertical in our result as well.

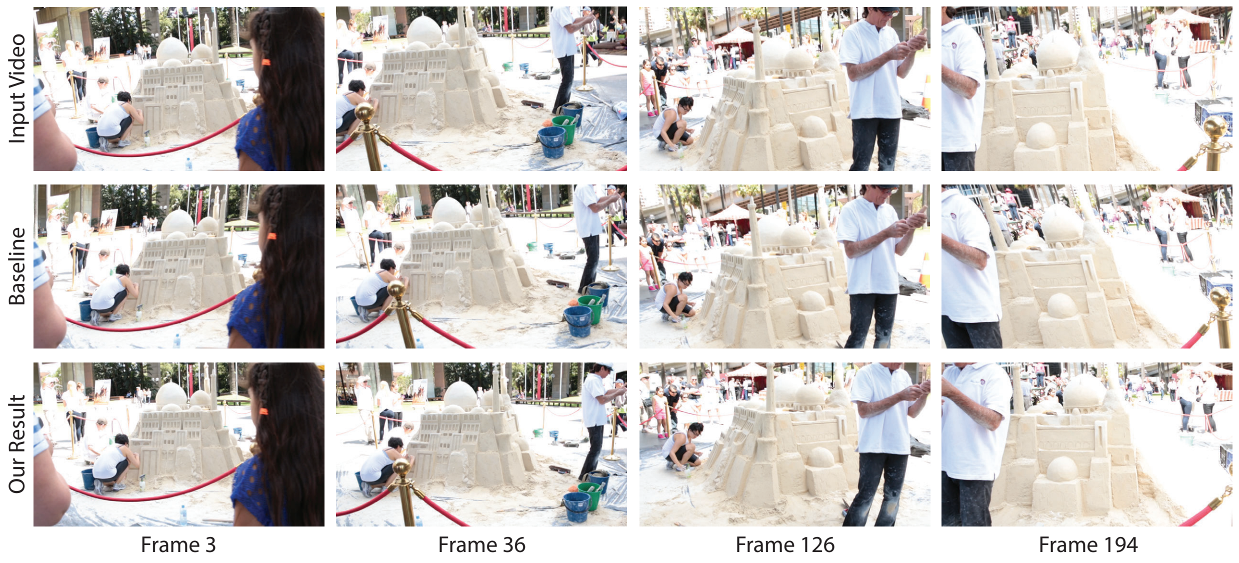


Figure 5.12: In this example, the skew of the sandcastle is distracting in the baseline stabilization. Our result correctly preserves the shape of the sandcastle.

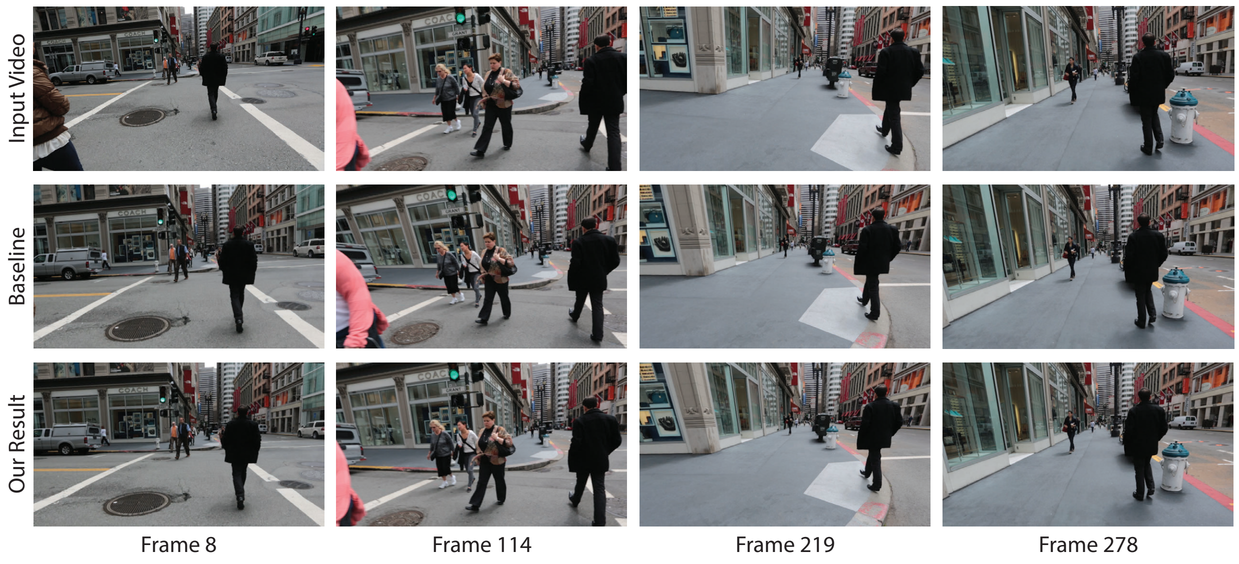


Figure 5.13: The suddenly jerk in the video at frame 114 results in the left facade being heavily distorted in the baseline stabilization. Notice how our result reduces the distortion.

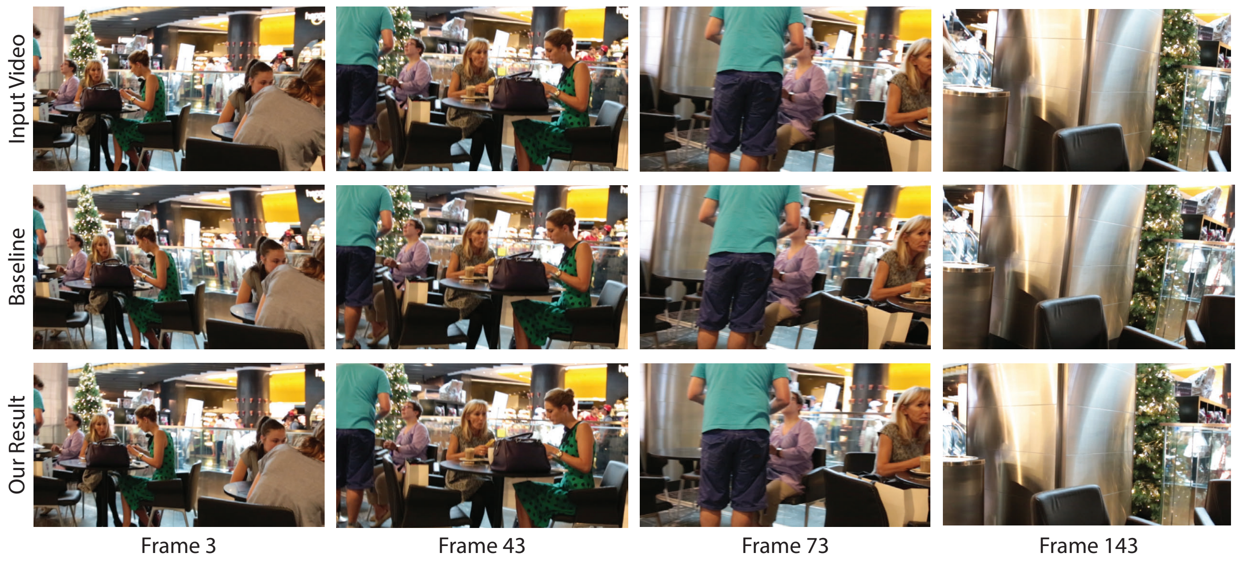


Figure 5.14: Our result for this example exhibits fewer distortions and the vertical structures largely remain vertical compared to the baseline result.

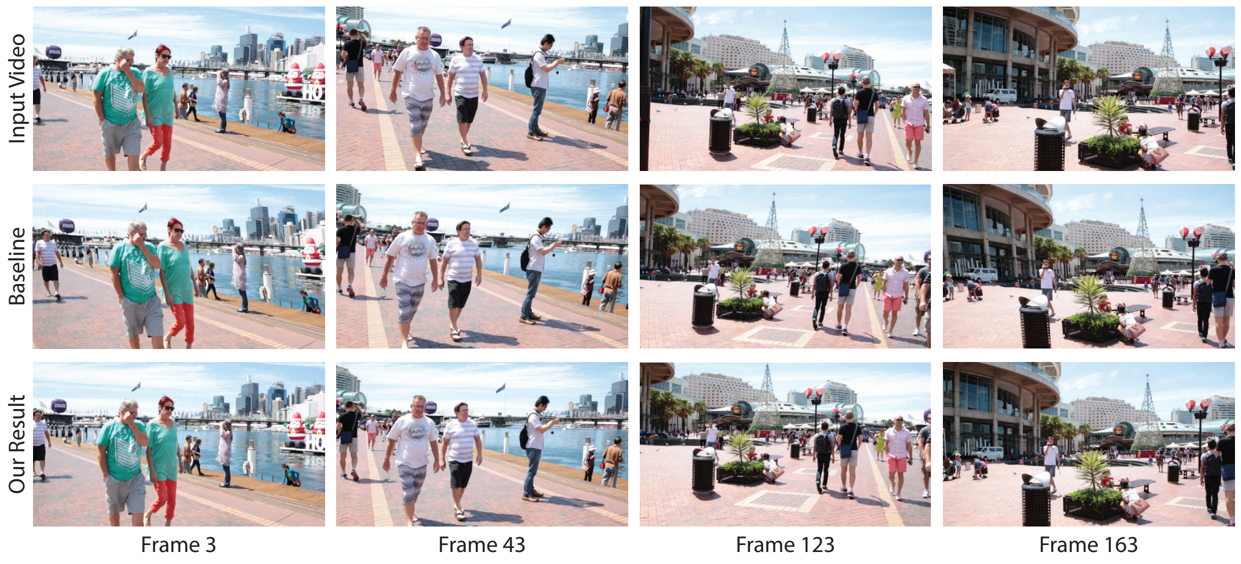


Figure 5.15: Notice how subjects and faces in the baseline results are skewed in either direction due to the extreme shake of the camera. In our result, these distortions are more pleasing to the eye as structures are largely vertical and pedestrians are not skewed.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we have demonstrated how computation along with user-desired goals can generate output media that better express what the creator experienced when she captured the input video. In De-animation, our system is able to isolate finer-scale motions by removing large-scale motions and compositing them into a still image by leveraging on technologies such as image warps and graph-cuts. The flexibility of our system is that it is goal driven, as the user has control over what the algorithm will optimize. This allows the user to produce surreal results that are captivating and better express the moment than the input video. Our system can also be used as a tool for visualizing motion as it isolates a target motion from the rest of the video. We have also demonstrated that our system allows new approaches for video editing tasks on moving objects.

When we restrict the domain of input videos to only portraits and assume that the goal is to create portrait cinemagraphs, we show that it is possible to automatically generate convincing results with input videos from DSLRs and handphones. We leverage on facial feature detection algorithms to estimate local areas of motion and composite them into a still image using automatically generated constraints. Our fully automatic system generates results within 30 seconds and we think that our system is a viable option to produce portraits for everyday use.

Video stabilization is a useful tool to help remove camera shake to improve the quality of the video. However for extremely challenging videos, automatic algorithms struggle to generate good results without distorting the scene. We show that by using user guidance in the form of correcting erroneous frames and improving track selection, we can generate a better stabilized output video. Our user-assisted video stabilization tool uses user-input as constraints to help guide an new camera path that satisfies her desires. In particular, the facades in our output videos are not distorted, camera motion is smoother, horizons are rectified, and low frequency camera sways are removed.

In summary, we believe new algorithms for video (and photo) editing should be designed to take into user's goals and/or media content into account. This allows higher-level editing operations to be performed by novice users without expert knowledge in video editing software. In our work, we developed end-to-end systems that allow users to create high quality results for de-animation, cinemagraphs, and video stabilization without professional video editing software and tedium. We hope that our work inspires continued progress in creating easy-to-use tools to empower novice users to create high-quality content.

6.2 Future Work

There are specific areas for improvement for each of our systems. However, in general there are other areas where video content should be used to inform video editing operations.

Current video stabilization algorithms only seek to stabilize the footage by removing camera shake. While some algorithms attempts to emulate cinematic rules by constraining the output video to have either specific camera paths such as linear or parabolic, there has been little effort in constraining the output video based on its content. Specifically, video cropping or video retargeting can be jointly optimized with video stabilization. Professional videos do not only have pleasant camera paths; they are also aesthetically pleasing. One future work would be to use common rules of thumb for video aesthetics as a guideline for video stabilization and cropping.

Another application for content-based video stabilization and cropping is for ultra-wide angle head mounted cameras. While these cameras capture everything from the point of view of a person, they do not capture what the person is actually focusing on. For example, a person might track a subject of interest with his eyes and not with his head. A content-based video stabilization and cropping might generate a stabilized video crop that follows what the person is focusing on instead of just stabilizing the shaky video.

Head mounted video cameras not only capture live action, but also possibly have large pose variations due to quick head motions. These quick pose transitions can be difficult to stabilize and can be disorientating for a viewer. One interesting direction of future work would be to explore ways to convey the action as well as the pose change by using suitable transition frames.

While having a high quality footage is useful for telling stories, it is often more important to have a sequence of multiple video clips to convey a moment. Unfortunately, there has been minimal research on how sequences of different clips could be automatically combined to better tell a story with pleasing results. Interaction with audio voice overs and graphical animations could also help to better convey the moment. As such, tools to aid novice users to composite clips with audio for story telling is a future work.

Videos are essentially a sequence of photographs taken consecutively which samples the scene at reular time intervals. An interesting direction would be to design a system that allows the user to modify the exposure of an output photograph based on a video of the scene. This manipulated exposure can also vary spatially to create images that cannot be captured with a regular camera. One benefit of this is that long exposure photography can be taken without the use of a tripod.

6.3 De-animation

Our system to de-animate videos at the moment requires a significant amount of computation. In particular, it still takes several minutes on a desktop to compute a result. It will be challenging and useful to find computationally efficient methods while achieving similar results. This will greatly reduce the time it takes for each result which will allow the user to iterate and explore different goals interactively. With an efficient algorithm, the system can also be implemented on mobile devices which will allow everyday users to capture and create de-animated videos easily.

The algorithm minimizes several energy functions in succession as it computes an initial and then a refined warp, and then a composite. If the problem of de-animation is modeled as a single energy function, the computation required might be reduced. More importantly, the current system has the warping and compositing as two different stages with no interaction between them. As such, it is likely that our solution is a local minimum. One interesting work would be to jointly optimize the warp and composition to give a better result.

The user strokes that are required to create high quality results are dependent on the skill of the user. In some cases, this might be difficult for a novice user to use our system. While we have addressed the class of portrait videos with our followup work, there are other approaches that should be considered. For example, suggestions of regions that can be de-animated can be computed and provided to the user. Also, the user strokes that are used in our algorithm are hard constraints. This makes the system very sensitive to small changes to the input. One future work would be to explore how to allow mistakes in the user input by using soft constraints.

6.4 Portrait Cinemagraphs

The limitation of our system is that only facial features are animated in our final output. One avenue for future work would be to automatically detect and composite additional moving regions such as cloth or hair onto the final cinemagraph. As these regions are much more difficult to handle, one alternative would be to synthesize subtle motions in these regions using a motion database.

The cinemagraph portraits may be unnerving for some viewers as the head is static while the facial features are animated. One solution to remedy this is to add small motions back onto the head so that the results appears more natural. As an extension, new motions can also be synthesized by leveraging on a large database of human facial expressions.

An extension to allow pose changes will admit longer and more interesting cinemagraphs to be created. This can be achieved by first solving for good cinemagraphs for each pose, and generating suitable pose transition frames.

6.5 Video Stabilization

We have shown that video stabilization can be improved by using information about how the scene should look like from the user. There are several areas for improvement from our current system. Drawing quads can be unintuitive for casual users as there is a large degree of freedom. Simpler techniques such as drawing horizon lines or building facades might be easier. In fact, computer vision techniques to detect horizon lines, people, and building facades can be utilized to provide information instead of depending on a user. This might allow the next generation of scene-aware automatic video stabilization algorithms to perform better than our user-assisted system.

Bibliography

[1] URL: `http://cinemagraphs.com`.

[2] URL: `http://www.filmindustrynetwork.biz/nyc-photographer-jamie-beck-cinemagraph/12173`.

[3] URL: `http://kinotopic.com`.

[4] URL: `http://cinemagr.am`.

[5] URL: `http://www.icinegraph.com`.

[6] Aseem Agarwala et al. "Interactive digital photomontage". In: *ACM Transactions on Graphics* 23.3 (Aug. 2004), pp. 294–302.

[7] Aseem Agarwala et al. "Panoramic video textures". In: *ACM Transactions on Graphics* 24.3 (Aug. 2005), pp. 821–827.

[8] Jackie Assa, Yaron Caspi, and Daniel Cohen-Or. "Action synopsis: pose selection and illustration". In: *ACM Transactions on Graphics* 24.3 (Aug. 2005), pp. 667–676.

[9] Shai Avidan and Ariel Shamir. "Seam carving for content-aware image resizing". In: *ACM Transactions on graphics (TOG)*. Vol. 26. 3. ACM. 2007, p. 10.

[10] Jiamin Bai et al. "A dual theory of inverse and forward light transport". In: *Computer Vision–ECCV 2010*. Springer, 2010, pp. 294–307.

[11] Jiamin Bai et al. "Automatic Cinemagraph Portraits". In: *Computer Graphics Forum (EGSR 2013)* (2013).

[12] Jiamin Bai et al. "Selectively De-Animating Video". In: *ACM Transactions on Graphics* (2012).

[13] Jiamin Bai et al. "User-Assisted Video Stabilization". In: *Computer Graphics Forum (EGSR 2014)* (2014). URL: `http://graphics.berkeley.edu/papers/Bai-UVS-2014-06/`.

[14] Xue Bai et al. "Video SnapCut: Robust Video Object Cutout Using Localized Classifiers". In: *ACM Transactions on Graphics* 28.3 (July 2009), 70:1–70:11.

[15] Connelly Barnes et al. "PatchMatch: A randomized correspondence algorithm for structural image editing". In: *ACM Transactions on Graphics-TOG* 28.3 (2009), p. 24.

[16] Connelly Barnes et al. “Video Tapestries with Continuous Temporal Zoom”. In: *ACM Transactions on Graphics* 29.4 (July 2010), 89:1–89:9.

[17] Herbert Bay et al. “Speeded-Up Robust Features (SURF)”. In: *Comput. Vis. Image Underst.* 110.3 (June 2008), pp. 346–359. ISSN: 1077-3142.

[18] Eric P. Bennett and Leonard McMillan. “Computational Time-Lapse Video”. In: *ACM Transactions on Graphics* 26.3 (July 2007), 102:1–102:6.

[19] M. Bertalmio et al. “Simultaneous structure and texture image inpainting”. In: *Image Processing, IEEE Transactions on* 12.8 (2003), pp. 882–889. ISSN: 1057-7149. DOI: `10.1109/TIP.2003.815261`.

[20] Marcelo Bertalmio et al. “Image Inpainting”. In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques.* SIGGRAPH ’00. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 417–424. ISBN: 1-58113-208-5. DOI: `10.1145/344779.344972`. URL: `http://dx.doi.org/10.1145/344779.344972`.

[21] Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. “Tools for placing cuts and transitions in interview video”. In: *ACM Transactions on Graphics (TOG)* 31.4 (2012), p. 67.

[22] Yuri Boykov, Olga Veksler, and Ramin Zabih. “Fast Approximate Energy Minimization via Graph Cuts”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.11 (Nov. 2001), pp. 1222–1239. ISSN: 0162-8828. DOI: `10.1109/34.969114`. URL: `http://dx.doi.org/10.1109/34.969114`.

[23] I. N. Bronshtein and K. A. Semendyayev. *Handbook of Mathematics (3rd Ed.)* Ed. by K. A. Kirsch. London, UK, UK: Springer-Verlag, 1997. ISBN: 3-540-62130-X.

[24] Thomas Brox and Jitendra Malik. “Object segmentation by long term analysis of point trajectories”. In: *European Conference on Computer Vision (ECCV).* Lecture Notes in Computer Science. Springer, 2010.

[25] Yaron Caspi et al. “Dynamic stills and clip trailers”. In: *The Visual Computer* 22.9-10 (2006), pp. 642–652.

[26] Anil Cheriyadat and Richard J. Radke. “Non-negative matrix factorization of partial track data for motion segmentation”. In: *ICCV.* 2009, pp. 865–872.

[27] Pei-Yu Chi et al. “Democut: generating concise instructional videos for physical demonstrations”. In: *Proceedings of the 26th annual ACM symposium on User interface software and technology.* ACM. 2013, pp. 141–150.

[28] Taeg Sang Cho et al. “The patch transform and its applications to image editing”. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on.* IEEE. 2008, pp. 1–8.

[29] Yung-Yu Chuang et al. "A bayesian approach to digital matting". In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on.* Vol. 2. IEEE. 2001, pp. II–264.

[30] Yung-Yu Chuang et al. "Animating pictures with stochastic motion textures". In: *ACM Transactions on Graphics* 24.3 (Aug. 2005), pp. 853–860.

[31] Yung-Yu Chuang et al. "Video Matting of Complex Scenes". In: *ACM Transactions on Graphics* 21.3 (July 2002), pp. 243–248.

[32] Michael F. Cohen and Richard Szeliski. "The Moment Camera". In: *Computer* 39.8 (2006), pp. 40–45.

[33] Carlos D. Correa and Kwan-Liu Ma. "Dynamic Video Narratives". In: *ACM Transactions on Graphics* 29.4 (July 2010), 88:1–88:9.

[34] A Criminisi, P. Perez, and K. Toyama. "Object removal by exemplar-based inpainting". In: *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on.* Vol. 2. 2003, II–721–II–728 vol.2. DOI: `10.1109/CVPR.2003.1211538`.

[35] Antonio Criminisi, P. Perez, and K. Toyama. "Region filling and object removal by exemplar-based image inpainting". In: *Image Processing, IEEE Transactions on* 13.9 (2004), pp. 1200–1212. ISSN: 1057-7149. DOI: `10.1109/TIP.2004.833105`.

[36] Soheil Darabi et al. "Image Melding: Combining Inconsistent Images using Patch-based Synthesis". In: *ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2012)* 31.4 (2012), 82:1–82:10.

[37] Paul E Debevec and Jitendra Malik. "Recovering high dynamic range radiance maps from photographs". In: *ACM SIGGRAPH 2008 classes.* ACM. 2008, p. 31.

[38] Alexei A. Efros and William T. Freeman. "Image Quilting for Texture Synthesis and Transfer". In: *Proceedings of SIGGRAPH 2001.* Computer Graphics Proceedings, Annual Conference Series (2001). Ed. by Eugene Fiume, pp. 341–346.

[39] Alexei A. Efros and Thomas K. Leung. "Texture Synthesis by Non-parametric Sampling". In: *IEEE International Conference on Computer Vision.* Corfu, Greece, 1999, pp. 1033–1038.

[40] Ehsan Elhamifar and Ren Vidal. "Sparse subspace clustering". In: *In CVPR.* 2009.

[41] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. *Distance transforms of sampled functions.* Tech. rep. Cornell Computing and Information Science, 2004.

[42] Martin A. Fischler and Robert C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782.

[43] Martin A. Fischler and Robert C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782.

[44] Matthieu Fradet, Philippe Robert, and Patrick Pérez. "Clustering Point Trajectories with Various Life-Spans". In: *Proceedings of the 2009 Conference for Visual Media Production*. CVMP '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 7–14. ISBN: 978-0-7695-3893-8.

[45] K. Ruben Gabriel and S. Zamir. "Lower Rank Approximation of Matrices by Least Squares with Any Choice of Weights". In: *Technometrics* 21.4 (1979), pp. 489–498.

[46] Michael L. Gleicher and Feng Liu. "Re-cinematography: Improving the Camera Dynamics of Casual Video". In: *Proceedings of the 15th International Conference on Multimedia*. MULTIMEDIA '07. Augsburg, Germany: ACM, 2007, pp. 27–36. ISBN: 978-1-59593-702-5.

[47] Dan B Goldman et al. "Schematic storyboarding for video visualization and editing". In: *ACM Transactions on Graphics* 25.3 (July 2006), pp. 862–871.

[48] Amit Goldstein and Raanan Fattal. "Video Stabilization using Epipolar Geometry". In: *ACM Trans. Graph.* 32.5 (2012).

[49] M. Grosse et al. "Coded Aperture Projection". In: *ACM Trans. Graph.* 29.3 (2010), pp. 1–12.

[50] Matthias Grundmann, Vivek Kwatra, and Irfan Essa. "Auto-Directed Video Stabilization with Robust L1 Optimal Camera Paths". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011)*. 2011.

[51] Matthias Grundmann et al. "Calibration-Free Rolling Shutter Removal". In: *International Conference on Computational Photography [Best Paper]*. 2012.

[52] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, ISBN: 0521540518, 2004.

[53] James Hays and Alexei A Efros. "Scene Completion Using Millions of Photographs". In: *ACM Transactions on Graphics (SIGGRAPH 2007)* 26.3 (2007).

[54] Felix Heide et al. "Diffuse Mirrors: 3D Reconstruction from Diffuse Indirect Illumination Using Inexpensive Time-of-Flight Sensors". In:

[55] Felix Heide et al. "Light-in-flight: Transient Imaging Using Photonic Mixer Devices". In: *ACM SIGGRAPH 2013 Emerging Technologies*. SIGGRAPH '13. Anaheim, California: ACM, 2013, 9:1–9:1. ISBN: 978-1-4503-2340-6. DOI: 10.1145/2503368.2503377. URL: http://doi.acm.org/10.1145/2503368.2503377.

[56] Y. Hitomi et al. "Video from a Single Coded Exposure Photograph using a Learned Over-Complete Dictionary". In: *IEEE International Conference on Computer Vision (ICCV)*. 2011.

[57] Jia-Bin Huang et al. "Image Completion Using Planar Structure Guidance". In: *ACM Trans. Graph.* 33.4 (July 2014), 129:1–129:10. ISSN: 0730-0301. DOI: 10.1145/2601097.2601205. URL: http://doi.acm.org/10.1145/2601097.2601205.

[58] Arjun Jain et al. "MovieReshape: Tracking and Reshaping of Humans in Videos". In: *ACM Trans. Graph. (Proc. SIGGRAPH Asia 2010)* 29.5 (2010).

[59] Neel Joshi et al. "Cliplets: juxtaposing still and dynamic imagery". In: *Proceedings of the 25th annual ACM symposium on User interface software and technology*. UIST '12. Cambridge, Massachusetts, USA, 2012, pp. 251–260. ISBN: 978-1-4503-1580-7.

[60] Alexandre Karpenko et al. "Digital Video Stabilization and Rolling Shutter Correction using Gyroscopes". In: *In Stanford CS Tech Report*. 2011.

[61] Natasha Kholgade et al. "3D Object Manipulation in a Single Photograph using Stock 3D Models". In: *ACM Transactions on Computer Graphics* 33.4 (2014).

[62] B. Kim and I. Essa. "Video-based nonphotorealistic and expressive illustration of motion". In: *Computer Graphics International 2005*. 2005.

[63] Gustav Kirchhoff. "Ueber das Verhältniss zwischen dem Emissionsvermögen und dem Absorptionsvermögen der Körper für Wärme und Licht". In: *Annalen der Physik* 185.2 (1860), pp. 275–301.

[64] Dilip Krishnan and Rob Fergus. "Dark flash photography". In: *ACM Transactions on Graphics, SIGGRAPH 2009 Conference Proceedings*. Vol. 28. 2009.

[65] Vivek Kwatra et al. "GraphCut Textures: Image and Video Synthesis Using Graph Cuts". In: *ACM Transactions on Graphics* 22.3 (July 2003), pp. 277–286.

[66] Anat Levin, Dani Lischinski, and Yair Weiss. "A closed-form solution to natural image matting". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 30.2 (2008), pp. 228–242.

[67] Anat Levin et al. "4D frequency analysis of computational cameras for depth of field extension". In: *ACM Transactions on Graphics (TOG)*. Vol. 28. 3. ACM. 2009, p. 97.

[68] Anat Levin et al. "Image and depth from a conventional camera with a coded aperture". In: *ACM Transactions on Graphics (TOG)*. Vol. 26. 3. ACM. 2007, p. 70.

[69] Marc Levoy and Pat Hanrahan. "Light field rendering". In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM. 1996, pp. 31–42.

[70] Tommer Leyvand et al. "Data-Driven Enhancement of Facial Attractiveness". In: *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2008)* 27.3 (Aug. 2008).

[71] Zicheng Liao, Neel Joshi, and Hugues Hoppe. "Automated Video Looping With Progressive Dynamism". In: *ACM Transactions on Graphics* (2013).

[72] Ce Liu. "Beyond Pixels: Exploring New Representations and Applications for Motion Analysis". PhD thesis. Massachusetts Institute of Technology, May 2009.

[73] Ce Liu et al. "Motion magnification". In: *ACM Transactions on Graphics* 24.3 (Aug. 2005), pp. 519–526.

[74] Feng Liu et al. “Content-Preserving Warps for 3D Video Stabilization”. In: *ACM Transactions on Graphics* 28.3 (July 2009), 44:1–44:9.

[75] Feng Liu et al. “Subspace Video Stabilization”. In: *ACM Trans. Graph.* 30.1 (Feb. 2011), 4:1–4:10. ISSN: 0730-0301.

[76] Feng Liu et al. “Subspace Video Stabilization”. In: *ACM Transactions on Graphics* 30.1 (Jan. 2011), 4:1–4:10.

[77] Shuaicheng Liu et al. “Bundled Camera Paths for Video Stabilization”. In: *ACM Trans. Graph.* 32.4 (July 2013), 78:1–78:10. ISSN: 0730-0301.

[78] Bruce D. Lucas and Takeo Kanade. “An Iterative Image Registration Technique with an Application to Stereo Vision”. In: *International Joint Conference on Artificial Intelligence* (1981).

[79] Yasuyuki Matsushita et al. “Full-frame video stabilization with motion inpainting”. In: *IEEE Trans. Patt. Anal. Mach. Intell* (2006), pp. 1150–1163.

[80] Audrey R Nath and Michael S Beauchamp. “A neural basis for interindividual differences in the McGurk effect, a multisensory speech illusion”. In: *Neuroimage* 59.1 (2012), pp. 781–787.

[81] Shree K Nayar et al. “Fast separation of direct and global components of a scene using high frequency illumination”. In: *ACM Transactions on Graphics (TOG)*. Vol. 25. 3. ACM. 2006, pp. 935–944.

[82] Ren Ng et al. “Light field photography with a hand-held plenoptic camera”. In: *Computer Science Technical Report CSTR* 2.11 (2005).

[83] Matthew O’Toole et al. “Temporal frequency probing for 5D transient analysis of global light transport”. In: *ACM Trans. Graph.* 33.4 (2014), p. 87. DOI: `10.1145/2601097.2601103`. URL: `http://doi.acm.org/10.1145/2601097.2601103`.

[84] Amy Pavel, Björn Hartmann, and Maneesh Agrawala. “Video digests: a browsable, skimmable format for informational lecture videos”. In: *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM. 2014, pp. 573–582.

[85] Barbara Pease and Allan Pease. *The definitive book of body language*. Random House LLC, 2008.

[86] Patrick Pérez, Michel Gangnet, and Andrew Blake. “Poisson image editing”. In: *ACM Transactions on Graphics (TOG)*. Vol. 22. 3. ACM. 2003, pp. 313–318.

[87] Georg Petschnigg et al. “Digital photography with flash and no-flash image pairs”. In: *ACM transactions on graphics (TOG)*. Vol. 23. 3. ACM. 2004, pp. 664–672.

[88] Y. Pritch, E. Kav-Venaki, and S. Peleg. “Shift-Map Image Editing”. In: *International Conference on Computer Vision*. Kyoto, 2009.

[89] Yael Pritch, Alex Rav-Acha, and Shmuel Peleg. "Nonchronological Video Synopsis and Indexing". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.11 (Nov. 2008), pp. 1971–1984. ISSN: 0162-8828. DOI: `10.1109/TPAMI.2008.29`. URL: `http://dx.doi.org/10.1109/TPAMI.2008.29`.

[90] Shankar Rao et al. "Motion Segmentation in the Presence of Outlying, Incomplete, or Corrupted Trajectories". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.10 (2010), pp. 1832–1845. ISSN: 0162-8828.

[91] Ramesh Raskar, Amit Agrawal, and Jack Tumblin. "Coded exposure photography: motion deblurring using fluttered shutter". In: *ACM Transactions on Graphics (TOG)* 25.3 (2006), pp. 795–804.

[92] Alex Rav-Acha et al. "Unwrap Mosaics: A new representation for video editing". In: *ACM Transactions on Graphics* 27.3 (Aug. 2008), 17:1–17:11.

[93] Dikpal Reddy, Jiamin Bai, and Ravi Ramamoorthi. "External mask based depth and light field camera". In: *ICCV '13 Workshop Consumer Depth Cameras for Vision* (2013). URL: `http://graphics.berkeley.edu/papers/Reddy-EMB-2013-12/`.

[94] Dikpal Reddy, Ashok Veeraraghavan, and Rama Chellappa. "P2C2: Programmable pixel compressive camera for high speed imaging". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE. 2011, pp. 329–336.

[95] Michael Rubinstein et al. "Motion Denoising with Application to Time-lapse Photography". In: *IEEE Computer Vision and Pattern Recognition (CVPR)* (2011), pp. 313–320.

[96] Jason M. Saragih, Simon Lucey, and Jeffrey F. Cohn. "Deformable Model Fitting by Regularized Landmark Mean-Shift". In: *Int. J. Comput. Vision* 91.2 (Jan. 2011), pp. 200–215. ISSN: 0920-5691.

[97] Arno Schödl et al. "Video Textures". In: *Proceedings of ACM SIGGRAPH 2000*. Computer Graphics Proceedings, Annual Conference Series. July 2000, pp. 489–498.

[98] Jianbo Shi and C. Tomasi. "Good features to track". In: *Computer Vision and Pattern Recognition*. 1994, pp. 593 –600. DOI: `10.1109/CVPR.1994.323794`.

[99] B.M. Smith et al. "Light field video stabilization". In: *Computer Vision, 2009 IEEE 12th International Conference on*. 2009, pp. 341–348.

[100] Jian Sun. "Video Stabilization with a Depth Camera". In: *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. CVPR '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 89–95. ISBN: 978-1-4673-1226-4.

[101] Jian Sun et al. "Poisson matting". In: *ACM Transactions on Graphics (ToG)*. Vol. 23. 3. ACM. 2004, pp. 315–321.

[102] Kalyan Sunkavalli et al. "Video snapshots: Creating high-quality images from video clips". In: *Visualization and Computer Graphics, IEEE Transactions on* 18.11 (2012), pp. 1868–1879.

[103] Gregory P. Sutton and Malcolm Burrows. "Biomechanics of jumping in the flea". In: *J Exp Biol* 214.5 (Mar. 2011), pp. 836–847. DOI: `10.1242/jeb.052399`. URL: `http://dx.doi.org/10.1242/jeb.052399`.

[104] Richard Szeliski. *Image Alignment and Stitching: A Tutorial*. Tech. rep. MSR-TR-2004-92. Microsoft Research, 2004, p. 89. URL: `http://research.microsoft.com/apps/pubs/default.aspx?id=70092`.

[105] Michael W Tao, Micah K Johnson, and Sylvain Paris. "Error-tolerant image compositing". In: *International journal of computer vision* 103.2 (2013), pp. 178–189.

[106] Michael W Tao et al. "Depth from Combining Defocus and Correspondence Using Light-Field Cameras". In: *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE. 2013, pp. 673–680.

[107] James Tompkin et al. "Towards Moment Imagery: Automatic Cinemagraphs". In: *Visual Media Production, Conference for* 0 (2011), pp. 87–93. DOI: `http://doi.ieeecomputersociety.org/10.1109/CVMP.2011.16`.

[108] Ba Tu Truong and Svetha Venkatesh. "Video abstraction: A systematic review and classification". In: *ACM Trans. Multimedia Comput. Commun. Appl.* 3.1 (Feb. 2007). ISSN: 1551-6857. DOI: `10.1145/1198302.1198305`. URL: `http://doi.acm.org/10.1145/1198302.1198305`.

[109] Ashok Veeraraghavan et al. "Dappled photography: Mask enhanced cameras for heterodyned light fields and coded aperture refocusing". In: *ACM Transactions on Graphics* 26.3 (2007), p. 69.

[110] Andreas Velten et al. "Femto-photography: Capturing and visualizing the propagation of light". In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), p. 44.

[111] Andreas Velten et al. "Recovering three-dimensional shape around a corner using ultrafast time-of-flight imaging". In: *Nature Communications* 3 (2012), p. 745.

[112] Neal Wadhwa et al. "Phase-Based Video Motion Processing". In: *ACM Trans. Graph. (Proceedings SIGGRAPH 2013)* 32.4 (2013).

[113] Yu-Shuen Wang et al. "Spatially and Temporally Optimized Video Stabilization". In: *IEEE Transactions on Visualization and Computer Graphics* 19.8 (Aug. 2013), pp. 1354–1361. ISSN: 1077-2626.

[114] Bennett Wilburn et al. "High performance imaging using large camera arrays". In: *ACM Transactions on Graphics (TOG)*. Vol. 24. 3. ACM. 2005, pp. 765–776.

[115] Hao-Yu Wu et al. "Eulerian Video Magnification for Revealing Subtle Changes in the World". In: *ACM Trans. Graph. (Proceedings SIGGRAPH 2012)* 31.4 (2012).

[116] Mei-Chen Yeh and Po-Yi Li. “A tool for automatic cinemagraphs”. In: *Proceedings of the 20th ACM international conference on Multimedia*. MM ’12. Nara, Japan, 2012, pp. 1259–1260. ISBN: 978-1-4503-1089-5.