A Framework for Video Annotation, Visualization, and Interaction

Daniel R. Goldman

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Washington

2007

Program Authorized to Offer Degree: Computer Science and Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Daniel R. Goldman

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Co-Chairs of the Supervisory Committee:

David H. Salesin

Brian Curless

Reading Committee:

David H. Salesin

Steven M. Seitz

Brian Curless

Date: JULY 25, 2007

In presenting this dissertation in partial fulfillment of the requirements for the doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to Proquest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, 1-800-521-0600, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature 

Date JULY 25, 2007

University of Washington

Abstract

A Framework for Video Annotation, Visualization, and Interaction

Daniel R. Goldman

Co-Chairs of the Supervisory Committee:
Professor David H. Salesin
Computer Science and Engineering

Professor Brian Curless
Computer Science and Engineering

Existing approaches to interaction with digital video are complex, and some operations lack the immediacy of interactive feedback. In this thesis, I present a framework for video annotation, visualization, and interaction that harnesses computer vision to aid users in understanding and communicating with digital video. I first review the literature concerning visual representations, navigation, and manipulation of video data, and explore the literature of professional film editing, summarizing some of the techniques applied by and operations performed by film editors. I describe a new approach for computing the motion of points and objects in a video clip, and I present my interactive system that utilizes this data to visually annotate independently moving objects in the video, including speech and thought balloons, video graffiti, hyperlinks, and path arrows. I also demonstrate an application of this interface to construct visualizations of a short video clip in a single static image, using the visual language of storyboards. The principal advantage of the storyboard representation over standard representations of video is that it requires only a moment to observe and comprehend but at the same time retains much of the detail of the source video. The layout of the storyboard can be optimized to place the elements in a configuration that maximizes the clarity of presentation. Finally, I also demonstrate two novel interaction techniques for random video frame access using either the natural spatial dimensions of a storyboard representation or an individual video frame.

Throughout the thesis, I discuss how these approaches simplify and streamline the understanding and manipulation of video materials.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I first wish to thank my collaborators and advisors for the projects comprising my thesis research, Professors David Salesin, Brian Curless, and Steven Seitz. The rules of the Graduate School limited the number of committee co-chairs to two, so in the interest of fairness I drew a random number to determine the odd man out. In reality, Steve is my third de facto co-chair.

Source code and software utilized in the construction of this dissertation were provided by Peter Sand, Michael Black, Chris Gonterman, Harshdeep Singh, Samreen Dhillon, Aseem Agarwala, and Krzysztof Gajos. Thanks for additional technical advice go to Dan Weld, Sameer Agarwal and Josef Sivic, while filmmaker Ann Coppel and editors Cindy Sangster and Nic Anastassiou gave helpful discussions about film editing practice.

Sample video footage was provided by Nick Irving at UW Intercollegiate Athletics. Special thanks to Peter Rubin for providing artwork and for improving our understanding of storyboard iconography, and to my brother Jon Goldman for the use of footage from his short film *Kind of a Blur*, which can be purchased online at Apple's iTunes Store.

Kevin Chiu, Wilmot Li, and Mira Dontcheva provided additional technical assistance with video production. Nodira Khoussainova, Nathan Goldman, and Chris Gonterman appear in my video demonstrations.

Many thanks to the purveyors of fine caffeinated beverages who provided essential resources utilized during the course of this research: Revolutions, Herkimer, Fremont Coffee, Zoka, Cafe Allegro, Chocolati, Wannabee, and Macrina.

I would also like to thank a number of additional advisors and mentors, both formal and informal. In pseudo-chronological order: Andy van Dam, Marc Brown, Marc Levoy, Eric Enderton, Joe Letteri, Doug Smythe, Ben Snow, John Knoll, Pablo Helman, and Aaron Hertzmann. All of them pushed me to do more or better when I probably would have otherwise chosen to take a nap. My belated return to academia was made possible by the generosity of Cliff Plumer, Andy Hendrickson,

Steve Sullivan, and many others at Industrial Light and Magic.

Finally, I thank my parents, Martin and Helen Goldman, for their support and encouragement.

DEDICATION

I dedicate this dissertation to Molly and Nathan Goldman. I owe everything to their patience, understanding, and support of late nights and abbreviated weekends. I love you with all my heart. "We are a problem-solving family!"

Chapter 1

INTRODUCTION

Film and video are expressive and realistic media for communication and entertainment, but the high cost and difficulty of acquiring, manipulating, and distributing these media have traditionally limited production to the domain of Hollywood. Recently, however, as cameras, bandwidth, and storage become increasingly affordable, a wider spectrum of users produces digital video in ever-increasing volumes. Unfortunately, existing approaches to interacting with digital video have not kept up with the deluge of raw material. Professional methods for searching, browsing, annotating, and editing video material are too complex and often lack the immediacy of interactivity required by novice users.

In this thesis, I present a framework for video annotation, visualization, and interaction that harnesses computer vision to aid users in authoring and visualizing digital video. I will discuss a series of related projects that address some of the specific tasks common to many video applications. These projects focus on the problems of video object annotation, static visualization, and temporal navigation within a video clip.

1.1 Challenges of video

Video is an inherently difficult medium to grasp because its temporally varying nature cannot be apprehended in a single instant. In video editing software designed for experts, video is represented using individual frames, but casual authors and consumers of digital video do not want to be troubled by the unintuitive notion of video frames. Instead, they are more interested in the higher level components of a video like motion, action, character, and story. Interfaces for viewing and manipulating digital video should therefore revolve around these higher-level concerns. Another characteristic of novice users is that in general they are less willing than experts to wait long periods of time for results: All the user's interactions should produce immediate tangible results.

It may appear that these two requirements are at odds, since computing even low-level features in video can be computationally intensive. However, since video acquisition and editing are not usually performed at the same time, we can perform some video analysis between acquisition and editing in order to enable higher-level operations and interactions. Many possible aspects of video are candidates for processing in this precomputation phase, but in this thesis I will principally address the motion of objects in the scene. Although not as high-level as character and story, this is a mid-level feature of video that we can think about as a building block toward those higher-level concepts. Computer vision techniques can be applied to understand how various points move about the scene and segment this motion into independently moving objects. Such data can be extremely rich, as seen in Figure 1.1: It can be used to segment independently moving objects even without explicitly considering color or texture.

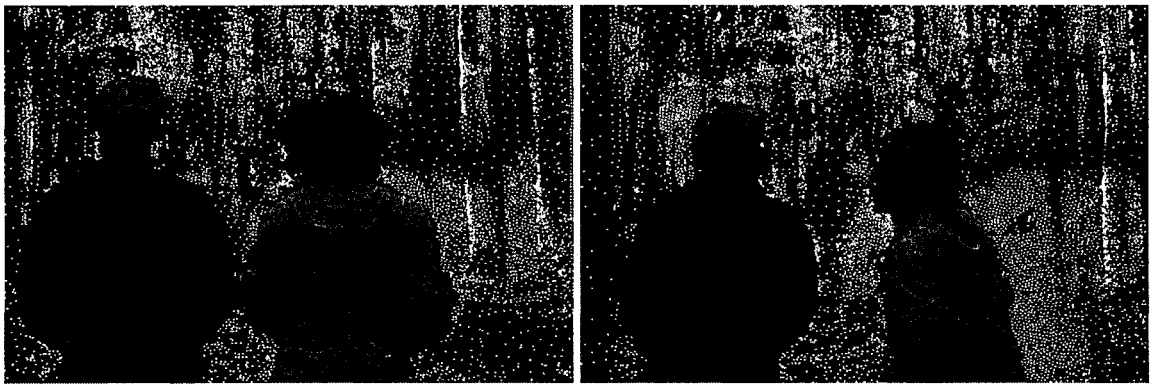


Figure 1.1: An illustration of motion data for two frames of a video clip. The points have been tracked on all frames and colored according to common motion. (video footage ©2005 Jon Goldman)

To demonstrate the utility of this data in interfaces for digital video, I address the problem of visual annotation: associating graphical objects with moving objects on the screen. In existing interactive applications, only still images can be annotated, as in the "telestrator" system used in American football broadcasting (see Figure 1.2). Using my system, such annotations can be attached to moving objects in the scene. The annotations supported by our system include descriptive labels, illustrative sketches, thought and word bubbles communicating speech or intent, path arrows

indicating motion, or even hyperlinked regions (see Figure 1.3). Given the precomputed object motion, the system can also determine when objects are occluded or change appearance significantly, and change the appearance of the annotations accordingly.

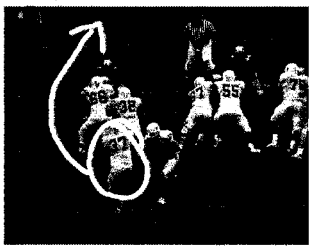


Figure 1.2: A mockup of the "telestrator" system used to annotate still frames of video during television broadcasts of American football games. (©Johntex, Creative Commons license [98])

Another challenge of digital video is how to represent video clips in a graphical user interface. Typical video interfaces today resort to using still frames as proxies for video segments. In this thesis, I explore a more expressive visual representation employed by filmmakers – the storyboard. This static representation has been developed by artists as a shorthand for expressing dynamic scenes in film using standard notational conventions. Although the language of storyboard notation has not previously been formalized, a number of common characteristics can be identified, such as the use of axis-aligned frame lines, 3D arrows that inhabit the 3D world of the video, and arrows breaking frame to indicate camera motion (see Figure 1.4). I demonstrate how my interactive annotation application can be used to author storyboards from general input video.

Finally, the use of motion analysis also permits novel approaches to video navigation through direct manipulation. The trajectories of objects in the scene can be employed as manipulation constraints: In my system, clicking and dragging on objects in a video frame or a storyboard representation causes the video to advance or rewind to a frame corresponding to that object's motion. This approach simplifies video tasks requiring selection of individual frames (for example, locating a "cut frame" on which to begin or end a shot).

In this thesis, I demonstrate that computational analysis of video content can be utilized to construct effective, intuitive interfaces for video tasks, including dynamic graphical annotation, static

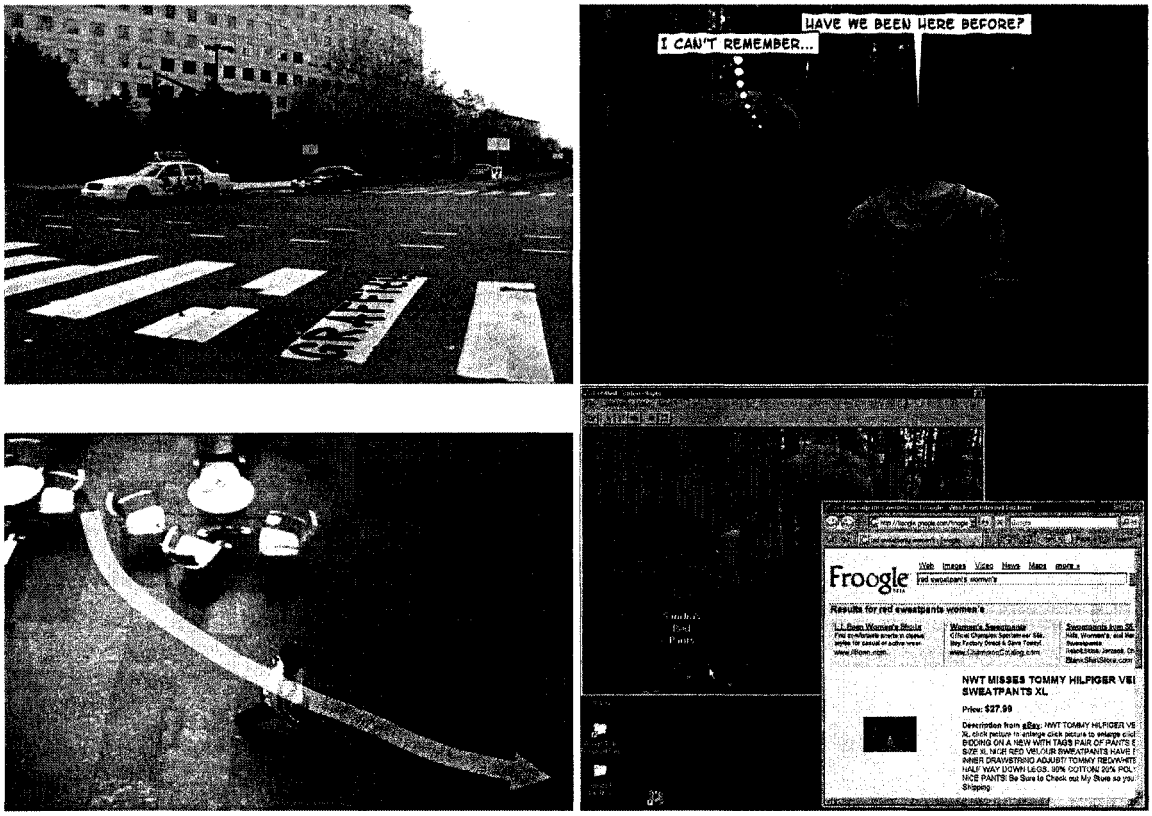


Figure 1.3: Several dynamic video object annotations. (video footage ©2005 Jon Goldman)

motion visualization, and frame navigation by direct manipulation. My contributions include a novel motion grouping algorithm, a fluid interface for user targeting of video objects for annotation, a formal summary of storyboard conventions, the first system for constructing storyboards from video, and an approach to video navigation using direct manipulation on video frames and storyboards.

1.2 Overview

In Chapter 2, I review the literature concerning visual representations, navigation, and manipulation of video data. In order to gain a better understanding of the ways that professionals think about browsing, searching, and editing large volumes of video material, I also explore the literature of professional film editing, summarizing some of the techniques applied by and operations performed by film editors. I focus specifically on the visual language of storyboards.

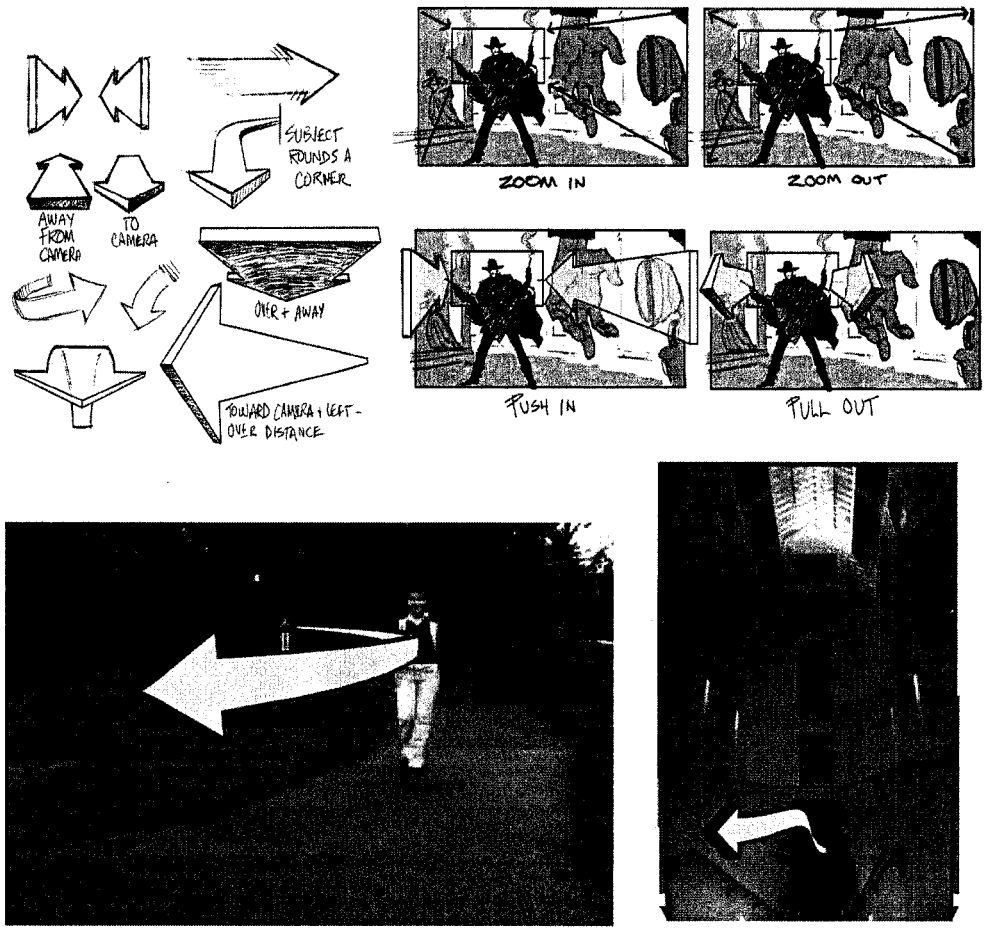


Figure 1.4: Examples of storyboard conventions.

In Chapter 3, I describe my approach for computing the motion of points and objects in a video clip. This system automatically tracks and groups features in the video in an off-line preprocess, so that manipulation and annotation can be implemented as interactive operations.

In Chapter 4, I present my interactive system for visually annotating independently moving objects in a video clip. The annotations in the system include speech and thought balloons, video graffiti, hyperlinks, and path arrows. This annotation interface can be employed in a variety of applications including surveillance, film and video editing, visual tagging, and authoring rich media such as hyperlinked video.

In Chapter 5, I discuss an application of my annotation interface for constructing visualiza-

tions of short video clips in a single static image, using the visual language of storyboards. These *schematic storyboards* are composed from multiple input frames and annotated using outlines, arrows, and text describing the motion in the scene. The principal advantage of this storyboard representation over standard representations of video – generally either a static thumbnail image or a playback of the video clip in its entirety – is that it requires only a moment to observe and comprehend but at the same time retains much of the detail of the source video. Chapter 6 describes a system for automatically refining the layout of storyboard elements for optimal placement.

Finally, in Chapter 7, I show how object motion can also be employed in a direct-manipulation interface for random frame access using spatial constraints. I also demonstrate an interaction technique to scrub through time using the natural spatial dimensions of a storyboard representation.

Throughout the thesis, I discuss how these approaches simplify and streamline the understanding and manipulation of video materials.

Chapter 2

BACKGROUND

2.1 Introduction

The availability of raw video material online is exploding, thanks to professional stock footage services like BBC Motion Gallery, as well as consumer-level services such as Google Video. Standard- and high-definition digital video cameras are becoming more affordable each year, and most still cameras and even some cell phone cameras are able to record short videos at low resolution. Video cameras are even sold as toys: Hasbro sells a video camera for $80 that can record over 7 hours of low-resolution video onto a Flash memory card [33].

Yet many video manipulation tasks – such as navigation, annotation, and editing – are still time-consuming processes, in some part due to the sheer volume of available video material that must be viewed and recalled. Commercial software for working with video generally offers only token mnemonic assistance, representing each video segment with only a single frame, which typically defaults to the segment's first frame. But this frame is often unrepresentative of the video content, and in any case does not illustrate other important aspects of the video, such as camera and subject motion. Consider the case of video editing packages such as Apple's Final Cut Pro or Adobe Premiere: In these systems, the organization of the video clips is left entirely to the user — typically using a hierarchical folder system — and the visual presentation of such folders is usually limited to either a list of text labels, or an array of thumbnails. Commercial offerings for video editing at the beginner and professional level look surprisingly similar, inheriting interface paradigms from the earliest digital editing tools created almost twenty years ago. The standard timeline visualization of even a few minutes of video is often a jumble of lines and colors.

In this chapter, I survey the literature related to graphical representations, navigation, and manipulation of video data, with an emphasis on their relation to the video editing process. I also assess the relative merits of these approaches, and identify new research directions. I will emphasize video editing over other video applications because of the much longer history of physical film editing.

This history provides a richer and more mature literature about how people think about moving images, as well as a body of work practices for managing the scale of large volumes of moving images that are now accessible to non-professionals.

Throughout this chapter I will address several specific user tasks relating to digital video that we wish to support and improve. The simplest operation to describe is simply finding or *selecting* video. One might use the word *search* to describe this operation, but search implies the existence of a specific target frame or segment of video, whereas the goal of finding may be somewhat less rigidly defined. In contrast, the term *browsing* implies a relatively undirected tour through some (possibly related) data. Therefore I use the more general term *selecting* to refer to the user's desires in invoking either or both of the two specific actions. Indeed, it may be common to do both searching and browsing operations: first search for some video, then browse through the results.

However, we also wish to support the important set of operations required to *edit* video. Raw video material may be low-quality, redundant, boring, or unintelligible. In addition to simply omitting less useful portions, edited video can convey a sense of cause and effect, or an entire narrative. Perhaps somewhat surprisingly, juxtaposing seemingly unrelated shots can communicate a novel meaning: expressing an emotion, a connection, a contrast, or sociopolitical ideas.

To a first approximation, video editing is just "cutting out the bad parts." But deciding what is a bad part is not always easy. For example, Francis Ford Coppola shot over 1,250,000 feet of film for *Apocalypse Now*: more than 230 hours. Yet the final film is 2 hours and 25 minutes. The ratio of source footage to final film, known as the *shooting ratio*, is 95 to 1 [70, 60]. (The average ratio is closer to 20 to 1.) Studio heads have long been concerned about shooting ratio since it is seen as a measure of a director's efficiency on location. But the editor Walter Murch once considered another ratio that studio heads had not yet considered: He took the total number of cuts that were in the finished product, divided it by the total number of days that the editors had worked on the film, and computed that the rate of cuts per editor-day came to about 1.47 [60]. For every splice made, 15 were made and then rejected. Even so, the overwhelming majority of a professional editor's time is not spent actually splicing film, but rather making visual decisions. (In fact, Murch claims, the editor is actually making twenty-four decisions a second: "No. No. No. No. No. No. No. No. No. No. No. Yes!") Seen this way, editing is not so much "putting together" as "discovery of a path": the more film, the more possible pathways.

Improving video editing interfaces potentially encompasses a great many problems of video analysis, visualization, and interaction. Although I will touch on each of these areas, the emphasis in this chapter is on visualizations of both raw and edited video material, and exploring how those visualizations can support interaction in general, and editing interactions in particular.

The remainder of this chapter is organized as follows: I begin in Section 2.2 with an overview of editing terminology. Section 2.3 explores the techniques and approaches of professional film editors. Finally, Section 2.4 discusses previous approaches to video visualization and editing.

2.2 Filmmaking terminology

In the text that follows, I will use some terminology common to film and video editors, but perhaps less familiar to computer scientists. Raw film or video that has not been edited is known as *footage*. A *shot* is the basic unit of an editor's film language, consisting of footage captured over a continuous duration of time without interruption. Another useful term for some narrative source material is the *take*, consisting of a portion of footage intended to form a shot in the final presentation. Thus, an actress repeating a single action many times at the instruction of a director is performing multiple takes, even though all of the takes may occupy a single unbroken stretch of footage. Finally, a number of shots using the same camera position, lighting, and set dressing may be captured in sequence at the same time, even though they do not appear in sequence in the film. This collection of shots is called a *setup*.

A *scene* refers to a sequence of adjacent shots in an edited film that take place at the same time and generally in the same place. However, some scenes can include simultaneous events in multiple locations by using the editing technique called *cross-cutting*, in which shots in alternate locations are interleaved.

The transition from one shot to another is called a *cut*, so named because it originally required physically cutting two pieces of film and splicing them together with clear tape. The frame at which a shot begins in the context of a finished piece is called the *in-point* or *in-frame*, and likewise the frame at which it ends is the *out-point* or *out-frame*. Thus, a cut joins the out-frame of the previous shot to the in-frame of the following shot. The unused portion of a take before and after the cuts are called the *head* and *tail* of the shot, respectively. These are sometimes known as *trims*.

Occasionally, editors may use *soft transitions* other than a hard cut between shots. The most common types are *fades* and *dissolves*, in which a shot gradually transitions to a solid color or to another shot. *Wipes* are less commonly used. In most cases, soft transitions indicate a passage of time.

Even as I adopt these terms from the film editing literature, I will depart from its terminology in one specific way: although film professionals would never refer to material captured on film and projected on a theater screen as "video," that term will here refer to any digitized images acquired or created at a sufficiently high frame rate to produce the illusion of motion when replayed, regardless of the images' original source or presentation medium. Likewise, most references to "film" footage can refer to electronically captured moving images as well, except where explicitly discussing physical media. Thus I will omit the awkward phrase "film and/or video" from the remainder of this thesis.

2.3 Film editing technique

In order to evaluate the suitability of different visualizations for selecting and editing video, we first explore the criteria by which a professional editor actually makes editing decisions. Although the goals of a home video editor are clearly more modest than those of the professional, a great deal of literature has been written about the minutiae of film editing, so it is instructive to consider the professionals first, and ask what criteria they apply in making their editing choices. By learning the aspects of video material that are most useful in making high-quality editing decisions, we can perhaps gain insight into the tools that might enable amateurs to make the same good decisions.

2.3.1 Realism, classicism, and formalism

Giannetti [26] describes film editing technique as a spectrum from *realism* through *classicism* through *formalism* (see Figure 2.1). Sequence-shots contain no editing at all, as in some of the earliest films. *Cutting to continuity* merely condenses time and space of a completed action, omitting for example the time that it takes one to drive home from work. *Classical cutting* interprets an action by emphasizing some details over others. *Thematic montage* is editing with a strong point of view, arguing a particular thesis, and connecting shots less with temporal continuity in mind but

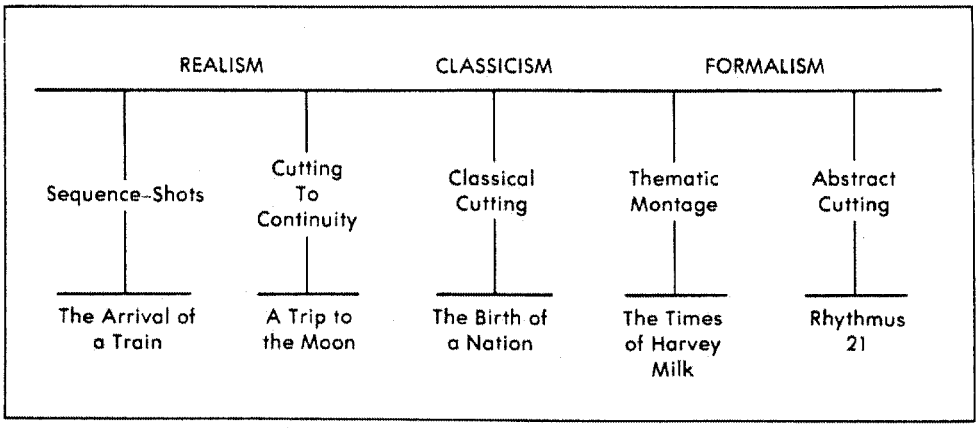


Figure 2.1: Giannetti's spectrum of editing techniques.

more with a subjective meaning. Finally, *abstract cutting* is purely formalistic, devoid of recognizable subject matter.

The importance of this spectrum for our purposes is that there are many common threads in editing, but no one uniform rule book. Furthermore, the choice of editing style is not just personal but also cultural, and even political. Over time the pendulum of style swings back and forth across this entire spectrum. Early filmmakers thought of editing as merely a technical way to join separate scenes shot all at once. Later filmmakers realized they could use a cut to bridge two parts of an action separated by uninteresting details.

Today, most Hollywood films lie somewhere near the middle of the spectrum, obeying the rules of classical cutting. The director D.W. Griffith is widely recognized as the father of classical cutting. Although the *close-up* shot tightly framing an actor's face had been used previously, Griffith applied it in novel contexts for purely psychological reasons, not only to convey emotions of characters but also to instill emotions in the audience. He developed the technique of *eyeline matching* — in which the direction of an actor's gaze in a close-up indicates to the viewer the physical location of the object being observed in a following shot — and the *180-degree rule*, establishing a virtual line in the scene that the camera rarely crosses. He also pioneered the use of cross-cutting, showing simultaneous events in multiple locations. Taken in combination, Griffith's techniques are sometimes known as

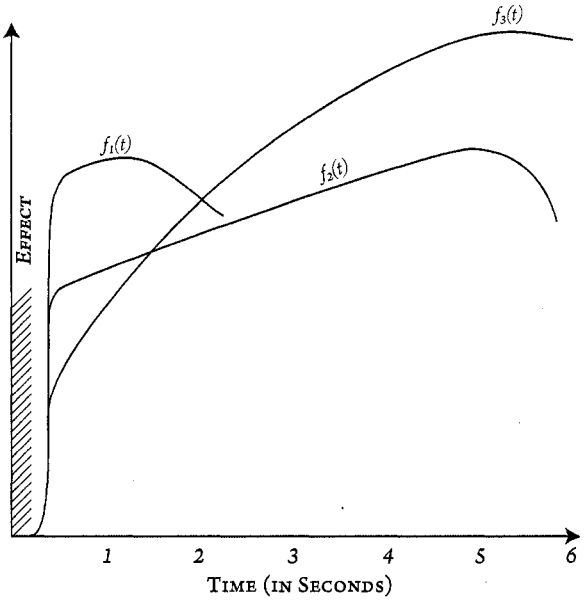


Figure 2.2: Four content curves, after an illustration by Spottiswoode [82]. The vertical axis signifies the effect of the content upon the viewer. Curve $f_1(t)$ represents a shot that is simple and striking, but becomes boring if held on the screen for too long. Curve $f_3(t)$ represents a shot that contains material of great intrinsic beauty or meaning in the film, for which the full understanding will be attained more slowly. Curve $f_2(t)$ represents an intermediate type. (The hatched region to the left of the graph indicates the duration of "subliminal images" that a human observer would not consciously perceive.)

"invisible editing" [26], because the cuts do not draw attention to themselves.

In Soviet Russia, filmmakers began to explore the use of film to communicate the ideals of Marxism, giving birth to formalist film editing, and swinging further right in Giannetti's spectrum of editing styles. One of the first formalist editors and theorists, V.I. Pudovkin, argued that new meanings cannot be created in a single shot alone, but only through the juxtaposition of shots [65]. He also identified a number of specific goals that may be achieved in editing: *contrast*, *parallelism*, *symbolism*, *simultaneity*, and *leit-motif*. Lev Kuleshov, a contemporary of Pudovkin, performed editing experiments [45] using a single closeup of an actor with a neutral expression, juxtaposed against several other shots: a bowl of soup, a coffin containing a female corpse, a little girl playing. When asked about the emotional state of the actor, audiences gave different descriptions depending on the specific juxtaposition they were shown: hungry, sad, proud. The formalists inferred that the

viewer him or herself can create emotional meaning, given the proper juxtaposition [26].

Sergei Eisenstein took this approach to an extreme in his theory of dialectical montage. According to Eisenstein, transitions should not be flowing, but rather jolting and even violent, and a smooth transition is an opportunity lost. Even today, Eisenstein's film, *Battleship Potemkin* (1925), seems radical in its construction [26].

Critics derided formalism as too explicitly manipulative, guiding the viewer too much toward a single intended emotional response, rather than allowing any nuanced interpretation. Even classical cutting subjectivizes an event because each shot represents what the filmmaker thinks is important, not necessarily what an impartial observer would think. The French film critic André Bazin countered classicism and formalism by championing a style that came to be known as *neorealism*, because it was felt to be a return to the realism of the earliest films, and a swing of the pendulum back to the left side of the editing spectrum. To Bazin, the essence of reality lies in its ambiguity, and in order to represent this ambiguity, the filmmaker must not try to impose a single point of view [10]. Neorealism was embraced by postwar filmmakers such as Roberto Rosselini and Vittorio De Sica, both of whom deemphasized the role of editing in their work. When asked about his distaste for a subjective style of editing, Rosselini reportedly replied, "Things are there, why manipulate them?" [26].

2.3.2 Why do cuts work?

In spite of their differences, all styles of editing take advantage in one way or another of the effective use of cutting to create or destroy continuity.

Film theorists and perceptual psychologists to this day struggle to understand how a cut — a complete temporal discontinuity of the visual field — can be so easy for a viewer to forgive and generally ignore. There seem to be points in a shot at which it is natural to cut, and those at which it is not. In an attempt to explain this phenomenon, Raymond Spottiswoode, an early film theorist, argued that every shot has its own *content curve*, representing the amount of information conveyed to the viewer at a given moment in time [82] (see Figure 2.2). This curve rises at the start of the shot, as the viewer adjusts to the sudden change of visual information, then generally begins to fall as the user scans the image repeatedly and sees the same scene again. According to Spottiswoode's

theory, a cut must be made at the peak of the content curve: the moment in a shot at which the audience has been able to assimilate most of its information. Cutting before the peak doesn't allow the audience to assimilate the action, but cutting after the peak of the content curve results in stilted, boring scenes that seem to drag. A complex scene requires more time to assimilate than a simple one, but once a setup has been established, the following shot in the same setup can be considerably shorter since the audience recalls the previous viewing [26].

The director John Huston believed that cuts were analogous to the way our mind processes visual information by ignoring things it has already seen, even drawing a connection between the physiological action of a blink to the physical mechanism of a cut [84]:

> "To me, the perfect film is as though it were unwinding behind your eyes, and your eyes were projecting it themselves, so that you were seeing what you wished to see. Film is like thought. It's the closest to the thought process of any art.
>
> "Look at that lamp across the room. Now look back at me. Look back at that lamp. Now look back at me again. Do you see what you did? You *blinked.* Those are *cuts.* After the first look, you know that there's no reason to pan continuously from me to the lamp because you know what's in between. Your mind cut the scene. First you behold the lamp. *Cut.* Then you behold me."

One principal factor in the effectiveness of a cut is directing the audience's focus of attention. Walter Murch — the editor of *Ghost* (1990), *The English Patient* (1996), *The Talented Mr. Ripley* (1999), and *Jarhead* (2005) — describes the importance of this phenomenon succintly:

> "If you think of the audience's focus of attention as a dot moving around the screen, the editor's job is to carry that dot around in an interesting way. If the dot is moving from left to right and then up to the right-hand corner of the frame, when there's a cut, make sure there's something to look at in the right-hand corner of the next shot to receive that focus of interest.... After each cut it takes a few milliseconds for the audience to discover where they should now be looking. If you don't carry their focus of interest across the cut points, if you make them search at every cut, they become disoriented

> and annoyed, without knowing why. Now, if you are cutting a fight scene, you actually want an element of disorientation—that's what makes it exciting. So you put the focus of interest somewhere else, jarringly, and you cut at unexpected moments." [63]

Bruce Block describes this editing principle as *continuum of movement* [12]. If the focus of attention is stationary or moves in a smooth curve across a cut, he calls this *affinity* of continuum of movement. If the focus of attention jumps from one place to another or changes direction of motion abruptly across the cut, this is called *contrast* of continuum of movement.

According to Block, affinity of continuum will disguise cuts with continuity errors or that break the 180-degree rule. However, editors may intentionally use contrast of continuum to create events that are visually jarring, or create a feeling of intensity. Block attempts to visualize continuum of movement using a series of storyboards, displayed first sequentially, then superimposed, but the results are fairly hard to decipher.

Continuum of movement is only one of many considerations in a cut. Murch gives six criteria for determining how to make a good cut [60], along with a somewhat tongue-in-cheek weighting of each criterion according to its importance:

- Emotion (51%)
- Story (23%)
- Rhythm (10%)
- Eye-trace (7%)
- Two-dimensional plane of screen (5%)
- Three-dimensional space of action (4%)

Emotion and *Story* are easily understood. *Rhythm* refers to repeated patterns of cutting, such as a regular metrical cutting of similar length shots, or alternating short and long shots. *Eye-trace* refers to Griffith's principle of eyeline matching, defined earlier in this section. *Two-dimensional plane of the screen* refers to the practice of allotting different characters or objects different focus points on the screen across multiple shots, so that for example one person is always to the left of frame and the other always to the right. Finally, *three-dimensional space of action* refers to traditional spatial continuity, so that, for example, if a woman is sitting in one shot, she should not be standing in the next.

Since emotion is at the top of the list, and is weighted more heavily than all the other criteria combined, a cut should never betray the emotion of a scene, even if it means breaking all the other rules. Similarly, editors should violate the other criteria starting from the bottom up. Note that Murch considers three-dimensional spatial continuity the *least* important when editing a scene.

Murch also takes John Huston's ideas about blinking a bit more literally. While editing *The Conversation* (1974), he discovered that he was always placing cuts at the exact frame at which the actor Gene Hackman was blinking [60]. Murch reasoned that both he and Hackman were internalizing the thought processes of Hackman's character, Harry Caul, and the transitions from one thought to the next were marked by blinks for an actor, and cuts for an editor. Murch considers the rate of a person's blinking to be closely related to our emotional state and to the nature and frequency of our thoughts. So, a blink is either something that helps us internally separate thoughts, or an involuntary reflex that goes along with the separation. When two people are conversing, the listener blinks when she "gets" the gist of what the speaker is saying. If this dialogue were being filmed, that's where the cut belongs.

Finally, there are holistic considerations in cutting that span beyond the individual shot or pair of neighbouring shots. Film director Sidney Lumet asserts that the two principal elements of editing are juxtaposition and creating *tempo* [55]. He explains that changing the tempo of a film throughout its length is the key: "If a picture is edited in the same tempo for its entire length, it will *feel* much longer. It doesn't matter if five cuts per minute or five cuts every ten minutes are being used. If the same pace is maintained throughout, it will start to feel slower and slower. In other words, it's the *change* in tempo that we feel, not the tempo itself."

Murch makes another musical analogy in describing underlying patterns of editing: "When it works, film editing — which could just as easily be called film construction — identifies and exploits underlying patterns of sound and image that are not obvious on the surface. Putting a film together is, in an ideal sense, the orchestrating of all those patterns, just like different musical themes are orchestrated in a symphony" [63].

2.3.3 Visualization and the element of chance

Perhaps most interesting for our purposes are Murch's thoughts on nonlinear editing and the creative process. He points out that one advantage of editing with physical film is that it forces the editor to constantly see footage other than what he is looking for, as he fast-forwards or rewinds through a reel in search of a particular shot. In the course of this search, the editor sees unexpected juxtapositions, sometimes finding what he *really* wants instead of what he *thinks* he wants. Each shot is embedded somewhere on a 10 minute reel, so all of the footage is constantly being reviewed, like a farmer turning over earth. This constant review is lost in digital editing, which makes such fortuitous accidents more rare.

Murch believes that cinema is at a similar historical point to music before musical notation was invented, "stumbling in the pre-notational phase" [63], and has even suggested applying the notation of the *I Ching* to narrative structure. Part of the difficulty of editing is the range of scales upon which the editor must act [63]:

> "The editor works at both the macroscopic and the microscopic level: ranging from deciding how long precisely each shot is held, to restructuring and repositioning scenes, and sometimes to eliminating entire subplots."

In an attempt to visualize a film as he assembles it, Murch has adopted a practice of printing out still frames from the film and assembling them on large pasteboards [60]. He chooses the best takes for each setup, and — adopting the language of Cartier-Bresson — tries to find the "decisive moment" in each of them: the one frame that best captures the emotion of the image. (He notes that this frame is often close to the cut point for that shot when assembled.) He then groups shots by scene, but places them in the order in which they were filmed, an order often completely unrelated to their order in the story. The benefit of this approach is that Murch can still see some juxtapositions that he might not have anticipated when looking at the script. In this way Murch restores some of the randomness to film editing that has been lost in the age of digital video editing.

In this section I have covered both the different styles of editing and the approaches used by specific editors in thinking about their profession and making visual decisions. Perhaps the most important point is that different cuts and editing approaches can have specific effects on an audience.

As we will see in the following sections, little attention has been given to the consideration of editing effects in visual interfaces to video.

2.4 Video visualization

In this section I will summarize some of the different ways in which video material — and moving images in general — have historically been visualized, browsed and edited.

2.4.1 Pictorial representation of motion

Even before the advent of film or video, artists were faced with the task of representing moving objects using static images or sculptures. We begin with a quick survey of pictorial representations of motion in art.

Cutting [20] catalogued five distinct solutions employed by artists for describing motion in an image: Dynamic balance, multiple stroboscopic images, affine shear/forward lean, photographic blur, and image and action lines. Cartoonists have their own toolbox for depicting the temporal axis, including adjusting the shape and size of panels, and bleeding panels off the page [58].

Ward [95] theorized that the most effective depiction of ongoing action is when both the means of arrival (i.e., the subject's past trajectory) is obvious and the subsequent motion appears inevitable. Action will appear suspended or incoherent when neither past nor future movements are clearly indicated, or if there is an equally likely alternate action. Ward identifies several means of indicating past and potential action: pose that is a consequence of motion, is difficult to maintain, and capable of clear development in the direction of apparent previous motion; involuntary consequences of figure motion such as flowing hair or clothing; direct depiction of changes of position such as streak, action lines, and stroboscopic repetition; and interaction with the environment, such as the wake of a boat or reaction to a punch. Each of these indications enhances the illusion of motion in a still image.

Of particular interest for our visualization purposes are the art and techniques of production storyboards. Storyboards have been used since the dawn of filmmaking [31] to articulate and communicate concepts of image composition and scene blocking. Although the basic concept of a storyboard is simply a sketch of selected representative frames in sequence, many storyboards em-

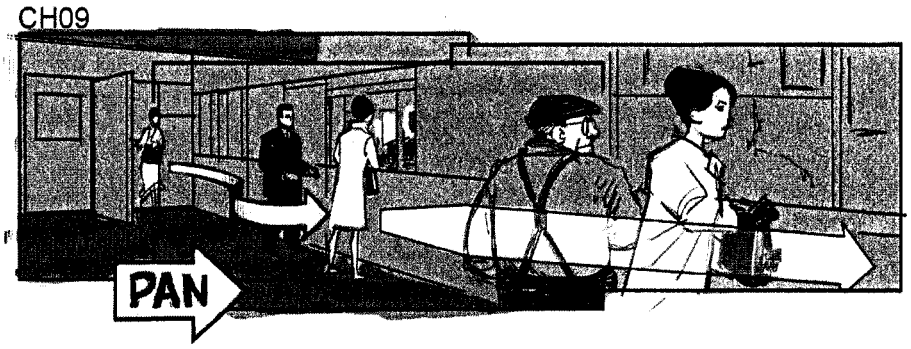


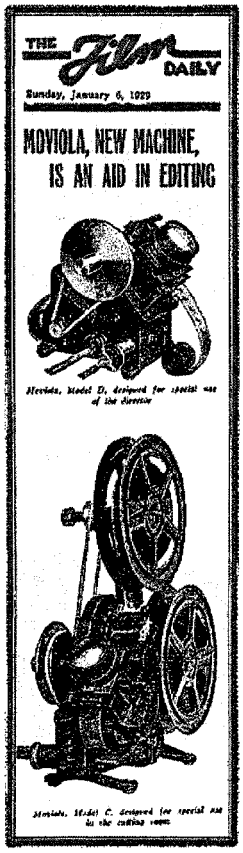
Figure 2.3: An example of a production storyboard featuring extended frame and motion arrows. Credit: Peter Rubin.

ploy more sophisticated visual notations to represent motion, including extended frames, motion arrows, and zoom lines [11, 76] (see Figure 2.3).

Computer graphics researchers have only recently begun to consider automatic depiction of motion in still images. Masuch *et al.* [57] and Kawagishi *et al.* [42] exploit cartoon illustration techniques such as speedlines and streaking to enhance the appearance of motion in computer animation. Joshi and Rheingans [40] apply similar techniques to illustrations of time-varying volume data. Nienhaus and Döllner [62] use storyboard notations — including 3D arrows and radial lines indicating collisions — to illustrate simple motions in computer animation. Kim and Essa [43] utilize motion analysis to apply stroboscopy, speedlines, and other nonphotorealistic motion effects to images and videos.

2.4.2 Physical film editing

Prior to the advent of digital video, professional film editing was performed using a cutting block with pins for sprocket alignment, grease pencils for annotation, and clear tape for splicing. Source footage and cuts could only be viewed using a separate playback device: first projectors, then later a vertical rear-projection device called the Moviola (see Figure 2.4). The Moviola is still in use by a few editors today: Michael Kahn edited the 2005 film *Munich* using a Moviola, for which he received an Academy Award for film editing.


THE Film DAILY

Sunday, January 6, 1929

MOVIOLA, NEW MACHINE, IS AN AID IN EDITING

Figure 2.4: An early advertisement for the Moviola.

Later devices were built specifically for editing, notably the German-made KEM and Steenbeck flatbed editors (see Figure 2.5, left). These machines feature multiple sound and picture heads and multiple screens, allowing two or more pieces of film to be examined before deciding upon the frames at which they would be cut and spliced together. A set of moving optical elements enables smooth variable speed control for frame-by-frame playback, fast forward, and rewind, without the flickering associated with a shutter mechanism. The takeup reels can also handle larger reels, enabling an editor to work with 20–30 minutes of footage, considerably more than the 10 minutes possible using a Moviola [97].

Using either the Moviola or the flatbed editors, finding relevant portions of film is made possible by voluminous textual logging of footage. Each piece of negative film has *edge codes*, unique

Figure 2.5: Left: KEM flatbed editor. Right: trim bin.

numeric identifiers printed between or adjacent to the sprockets of the film. This edge code is optically transferred to a print along with the image content during the developing process, so that numerous prints of the negative can be made without losing track of the source negative from which it was trimmed. But since a cut is made on a physical print of the film (called a *workprint*), adding length to a shot that has been shortened requires finding the small discarded pieces in a large *trim bin* filled with many such scraps and reattaching them [70] (see Figure 2.5, right), or recutting from a new (and expensive) workprint. Prior to the advent of digital image processing, transitions between scenes other than an ordinary cut — principally fades, dissolves, and wipes — were not actually made in the editing room at all. Instead, the editor handed off written instructions for the transition to a separate department, where they were executed on an "optical printer" designed for these and other special visual effects.

By the 1970s, a new generation of filmmakers foresaw that improving technology could radically change the editing process. Francis Ford Coppola announced, "I want editing equipment that will allow a director to edit a film without rushing, in a couple of weeks. Computers can do the cutting and splicing. The director and the editor will be able to concern themselves solely with the *meaning* and composition of the juxtaposed scenes" [70]. While an assistant editor was hunting for trims in a trim bin during editing of *The Empire Strikes Back* (1980), George Lucas announced, "Some day we'll have a digital trim bin."

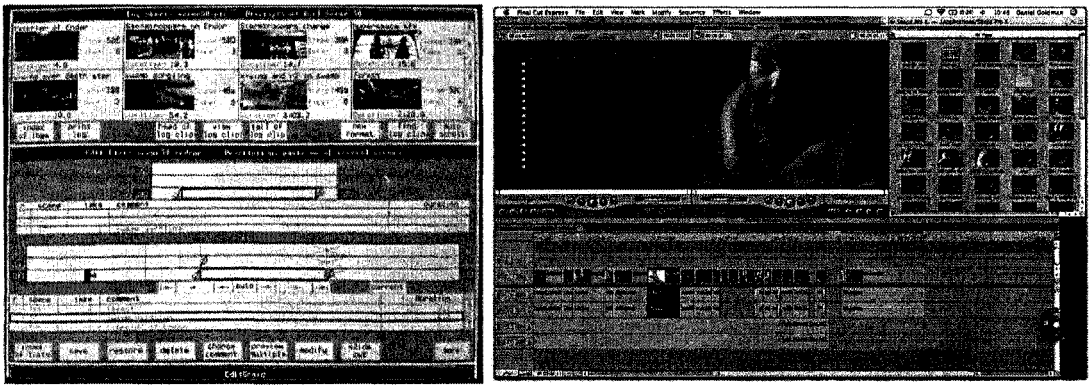


Figure 2.6: Left: Lucasfilm's EditDroid GUI, 1985. Right: Final Cut Express GUI, 2005.

2.4.3 Nonlinear editing

Eventually, LaserDiscs and magnetic disks enabled random access to large amounts of video data. Several editing systems were developed to take advantage of this random access, including the CMX 600, the Montage Picture Editor, and Lucasfilm's EditDroid [70]. Using one of these *nonlinear editing* systems, any piece of footage can be accessed in under a second, without having to scan through a reel (or worse, dig through a trim bin). Footage was represented first using text (on the CMX), then later using image thumbnails (on the Montage and EditDroid systems). The EditDroid system also adapted the physical layout of a flatbed editing table — including the orientation of film and sound tracks running from right to left — into a *timeline* visualization.

Present-day commercial video editing tools generally utilize the same basic visualization techniques for video editing. The major differences between Lucasfilm's 1985 EditDroid and the 2005 Final Cut Express (shown side by side in Figure 2.6) is that the timeline has been flipped into a left-to-right ordering, large previewing windows (also adapted from flatbed editors) occupy most of the screen space, icons have been made smaller, less textual information is shown, and the timeline now includes image thumbnails.

2.4.4 *Video abstraction*

In the 1990s, researchers began to explore alternate visualizations of video afforded by increasing computing power and decreasing cost of storage. Such visualizations are often known as *video abstracts*, since they condense a large volume of video data into visual summaries that can be apprehended more rapidly. Some literature distinguishes between *dynamic* abstracts, in which the output abstract is a video itself, and *static* video abstracts, in which the visualization is a static image that could be printed onto paper.

A few examples of dynamic abstracts include the work of Lienhart [51] and Pfeiffer *et al.* [64], who cluster shots hierarchically into meaningful units and balance the length of shortened shots using the hierarchical structure. Smith and Kanade [78] characterize video using both image and audio cues, including motion, language, face detection, and text overlay detection, and use these characterizations to identify the most salient sections to retain in their dynamic abstracts.

However, a key advantage of static representations is that they can be observed nearly instantaneously. In contrast, the simple act of observing a dynamic abstract takes a certain length of time. One can always fast-forward through a dynamic abstract (in effect scaling the data along the temporal axis), but as playback speed increases, it becomes more difficult to observe details.

In addition, static abstracts can show many video clips in parallel. One can imagine a dynamic abstract showing multiple video thumbnails at once, but such a display would be awkward to use in video editing applications since the human visual system can be quickly overwhelmed by even small numbers of video streams playing simultaneously: A rapid motion in one video may distract the observer's attention from small but important motions in another video playing simultaneously. Pylyshyn [66] has determined that observers can track a maximum of five independently moving objects at the same time. As the speed and number of the moving objects increases, the performance of the observer drops precipitously.

Although I am unaware of any user study featuring direct comparison of static and dynamic abstracts, Komlodi and Marchionini [44] compared several types of keyframe displays and found some improved object identification performance using side-by-side displays over sequential displays. In the remainder of this work I will cover principally static, side-by-side presentations, and those in which dynamic elements are initiated and controlled by explicit user interaction.

Many video abstraction systems first segment the video into its constituent shots by detecting cuts. A simple cut is straightforward to detect by comparing various image statistics between successive frames. However, such approaches can result in false positives when there is very fast camera or object motion, and false negatives for non-instantaneous cut transitions, such as dissolves, fades, or wipes. Lienhart [52] summarizes several approaches with improved robustness to dissolves and fades.

The Hitchcock system of Girgensohn *et al.* [27] attempts to separate *raw* home video into usable shots by constructing a measure of image unsuitability due to low brightness, rapid or jerky camera motion, and segment length. The video is segmented into shots by finding valleys in the unsuitability function that are longer than a minimum shot length. However, the user can request that individual shots or an entire assembled video be shortened or lengthened, and the unsuitability function will be used to re-optimize the in- and out-frames of one or more shots.

Once the video has been segmented, many systems attempt to identify a number of *keyframes* from the source video and display thumbnails of them side-by-side. A variety of approaches have been proposed to identify the best keyframes for such a display, including clustering [83, 102] and curve simplification [23]. Vermaak *et al.* [92] applied a decision-theoretic approach, designing a utility function rewarding sequences of keyframes that are maximally distinct and individually carry the most information, using histogram-based feature vectors to estimate the information content of each frame. The sequence of keyframes can then be optimized using dynamic programming.

None of the previous works specifically consider the effective depiction of motion as discussed in Section 2.4.1. However, Assa *et al.* [8] apply an analysis of human pose to determine the most representative keyframe thumbnails. In this work, the animation sequence is represented as a high-dimensional curve, and then embedded in a low-dimensional space. Curve analysis is applied to locate extrema of the low-dimensional embedding. User studies confirm that the automatically-selected key poses matched naïve users' interpretations of the most salient frames in a sequence, but it is not clear how these poses might fit the measures suggested by Ward.

Keyframe-based abstraction systems emphasize image content but often fail to illustrate the continuity of a video sequence. Continuity is retained in the two video visualizations suggested by Tonomura *et al.* [89], the *VideoMAP*, and the *VideoSpaceIcon*. The VideoMAP is a virtual "EEG" of a section of video that may include multiple shots. It takes the form of a table of data graphs

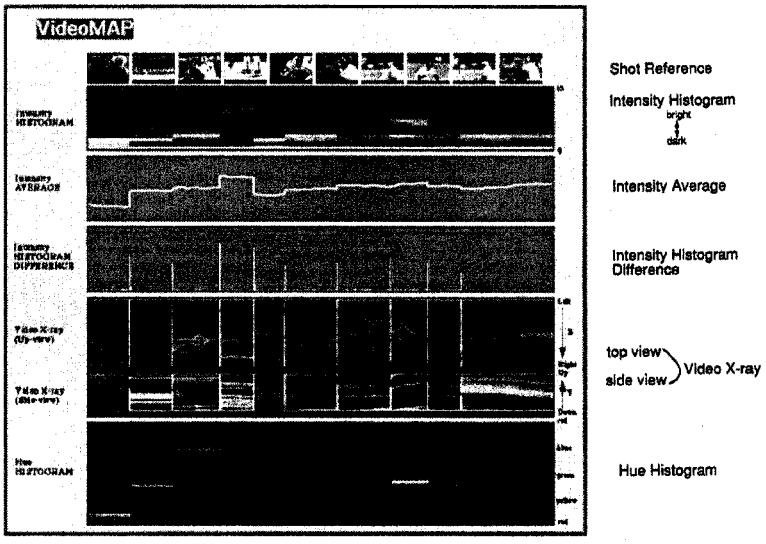


Figure 2.7: A sample VideoMAP.

(see Figure 2.7) in which multiple aspects of the video can be perceived. The top row contains key frames from each shot. Successive rows contain graphs of various pixel statistics of the video, such as histograms, histogram difference between frames, average brightness, number of edge pixels, etc. Although difficult for a novice to read, this visualization does show some salient aspects of the video clearly, such as the location of cuts (seen as spikes in the histogram difference graph), and the general direction of motion in a shot (seen in the "Video X-ray" graph).

2.4.5 Image mosaics

One approach for showing the continuity of an individual shot is to align the frames into a continuous image mosaic. Typically, these mosaics only apply to cameras in a fixed position with changing camera orientation or focal length. Under such constraints, all frames of a shot can be mapped to a common background coordinate frame. Teodosio and Bender [87] developed a system called "Salient Stills" to generate such mosaics from video, noting that the output mosaic can have multi-resolution patches, a larger field of view, or higher overall resolution than any of the constituent frames. In addition, noise levels can be reduced, and transient objects are located and extracted using a weighted median operator. They construct a visualization of object motion called a "timeprint,"

Figure 2.8: Left: A Salient Still "Timeprint" [87] showing the motion of a basketball player. Note that it is hard to determine the actual direction of motion without close examination. Right: Assa *et al.* [8] demonstrate a spatially extended thumbnail layout.

in which a foreground object is repeated stroboscopically across the mosaic image using various degrees of transparency, color distortions, and added blur (see Figure 2.8).

Although the construction of the Salient Stills mosaic is largely automatic, reintroducing foreground images into timeprints is a mostly manual process. User input is reduced by the system of Agarwala *et al.* [3], who demonstrate several approaches to generating image mosaics from video using an energy minimization framework. Under this framework, a noisy or incompletely specified objective, such as, "include all pixels different from the median background color," or, "include these specific pixels," is balanced against a smoothness prior ensuring that contiguous blocks from the same image are preferred over scattered isolated pixels.

2.4.6 Layout

Once shots have been segmented using cut detection and keyframes or mosaics extracted, systems showing multiple shots typically arrange thumbnails of the keyframe or mosaic images in a regular pattern on the screen, left-to-right (and sometimes in rows top-to-bottom), in temporal order [83, 102, 23, 92, 67]. However, several researchers have explored alternate arrangements.

Assa *et al.* [8] utilize several different approaches to display their selected keyframes. In addition to side-by-side keyframe displays and mosaic images, they also demonstrate a spatially-expanded

layout for physical actions. The goal of this layout is to maintain successive keyframes at a predefined distance from each other, at an orientation similar to the direction of the depicted object's motion. Simulated annealing optimization is used to meet these goals while also ensuring that keyframes do not overlap. The benefit of this approach is that the focus of the viewer's attention travels in the same direction as the object movement.

Taniguchi *et al.* [85] and Chiu *et al.* [16] observed that when multiple thumbnails are used to represent multiple shots, the resulting layout may result in poor utilization of screen space. Both works lay out the frames in temporal order (left-to-right and top-to-bottom) but address the screen space utilization problem using packing approaches. In PanoramaExcerpts [85], mosaics and keyframes are packed together using a greedy algorithm that explicitly attempts to minimize "dead space" in the layout. Their algorithm is fast enough for interactive layout, but does not find the optimal packing. Chiu *et al.* [16] note that not all portions of a keyframe are equally salient, and instead take the approach of identifying regions of interest (called *germs*) in the keyframe, laying them out in a regular grid, and filling the interstices with the less salient image content (called *supports*). This approach can theoretically pack more thumbnails in the same screen space, but in the absence of comparative user studies it is hard to assess the merits of one over the other.

2.4.7 Interaction

Irani and Anandan [37] observed the capability of mosaic representations to be used as a "visual table of contents" for direct manipulation indexing into the source video material. They describe interaction techniques for the case of a static background mosaic with no moving objects, and a synopsis mosaic that overlays a stroboscopic image of foreground moving objects with color-coded dotted lines indicating their trajectories. Each point in the mosaic is mapped to the frames during which that point was visible in the video, either one or more subsequences of the video. An annotation written on the mosaic can automatically be inserted into the source video, and static objects can even be inserted or deleted. Since the trajectories signify the passage of an object over time, marking segments of a trajectory is equivalent to marking a time interval. Irani and Anandan point out that their method does not work well for objects that rotate, or cameras that rotate about the objects. Their 2D motion estimation also assumes that only small objects in the field of view are in

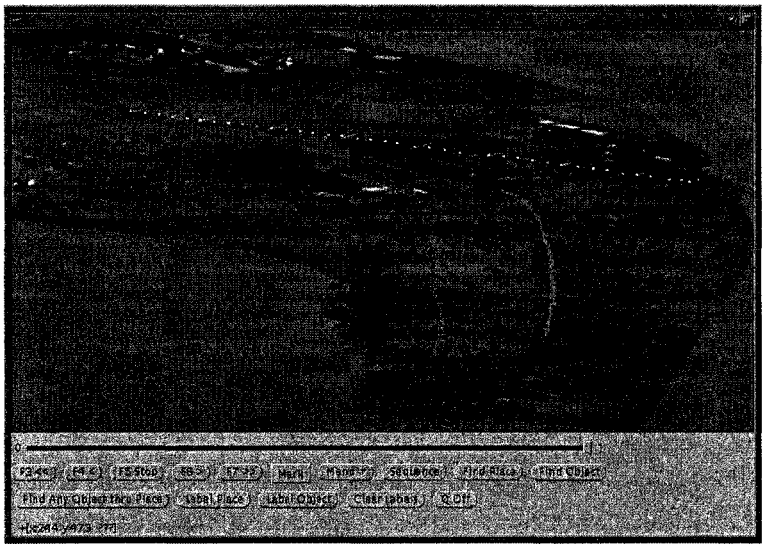


Figure 2.9: Indexing into a video using a mosaic [37]. Yellow dots mark the trajectory of the airplane. Green, magenta, and cyan dots mark the trajectory of three parachuters. Pink dots (near the right) mark the motion of a car.

motion.

Interaction in the system of Irani and Anandan is limited to spatial frame and segment selection. In contrast, Ramos and Balakrishnan's LEAN system [67] emphasizes interaction techniques for linear navigation and annotation of arbitrary video using a pressure-sensitive tablet. A *position-velocity slider* combines the video jog and shuttle wheels into a single linear virtual device: Up to a certain threshold the slider behaves just like a typical timeline slider, in which points on the slider map to frames of video. Beyond that threshold the slider behaves as a fast-forward or fast-rewind control, where the distance from the pen-down location (the "origin") determines the temporal velocity. A *twist-lens slider* shows a sequence of keyframe thumbnails, with a fish-eye distortion near the pen location making the small thumbnails more visible. Since this distortion causes overlap between the thumbnails, an interface is provided by which increasing the pen pressure causes the nearby thumbnails to move out of a straight line into an S-curve, eliminating the overlap and incidentally improving visibility when using a tablet LCD (see Figure 2.10).

Most video systems organize thumbnail images according to their temporal ordering, but the

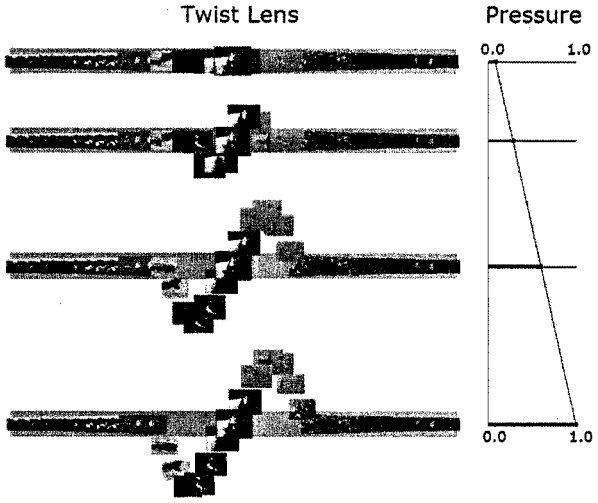


Figure 2.10: Ramos and Balakrishnan's "twist-lens slider" for visualizing video streams.

Hitchcock system [27] clusters shots according to visual features, and represents the clusters using stacks called *piles*. Large piles can be recursively expanded into multiple subpiles, but all of the higher level piles remain visible "behind" the currently-active pile (see Figure 2.11). Hitchcock also features a timeline interface in which the entire thumbnail is shown for each shot, uniformly scaled according to its duration. Since the duration of a shot can be very short or very long, the scaling function is nonlinear. In contrast to timelines in commercial video editing software, these thumbnails remain reasonably readable while still indicating relative shot length and cutting rhythm.

Previously discussed methods decompose videos only across the time domain, but Jojic *et al.* [39] applied graphical models to decompose shots into multiple sprite layers and a scene background as well. They claim that this decomposition has an intuitive meaning even to non-technical users, and describe an interface incorporating shot clustering using panoramic mosaic matching. One can interact with the interface by clicking on a thumbnail to find all the video segments containing that object or background. Using their system it is also reportedly possible to edit object motion and even add or remove objects using drag-and-drop interaction. However, due to the brevity of this workshop presentation the capabilities and usability of the actual implementation remain unclear.

Some researchers have looked beyond traditional interfaces to explore more physical interactions

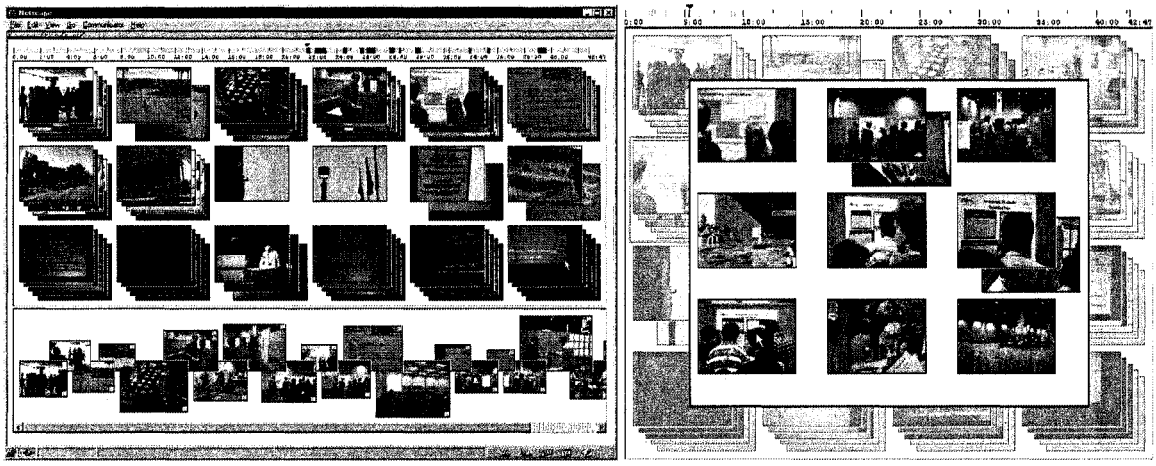


Figure 2.11: Girgensohn *et al.* [27] visualize related shots in "piles," which can be expanded into subpiles or individual shots.

with video material. Aner-Wolf and Wolf [7] and Zhu *et al.* [103] use physical photo albums as interfaces for browsing and assembling video clips. Aner-Wolf and Wolf use mosaic representations of shots for matching photos of events to corresponding video clips from a separate camera, applying a mosaic matching technique due to Aner and Kender [6]. The approach of Zhu *et al.* is more specialized, requiring a custom-made album with image masks in unique shapes that can be matched to a database using active shape models. Snibbe *et al.* [80] explore the use of haptic interfaces for video control and feedback.

Finally, Adams and Venkatesh [2] attempt to provide assistance to home videographers earlier in the video creation process by issuing recommendations for shot composition at the time of image capture using a PDA interface.

Table 2.1 highlights some of the characteristics of a selection of the video visualizations surveyed.

Table 2.1: Table of characteristics for different video visualization techniques.

Approach	**Organization**	**Timescale**	**Motion Repr.**	**Selections**
VideoSpaceIcon [89]	spatiotemporal	1 shot	camera sweep	N/A
Salient Stills [87]	spatial	1 shot	stroboscopic	N/A
Assa *et al.* [8]	spatial	1 shot	stroboscopic/blur	N/A
Irani and Anandan [37]	spatiotemporal	1 shot	camera sweep/action lines	segments
VideoMAP [89]	temporal	multiple shots	spacetime	(frame)
PanoramaExcerpts [85]	temporal	multiple shots	camera sweep	shot
Jojic *et al.* [39]	objects/backgrounds	whole video	camera sweep	objects
LEAN [67]	temporal/user-defined	whole video	none	frame
Hitchcock [27]	image statistics	whole video	none	shot
Aner-Wolf and Wolf [7]	user-defined	whole video	none	shot

Chapter 3

SCENE TRACKING AND GROUPING

3.1 Introduction

Many different features of a video can in theory be analyzed to improve video interfaces and visualizations. The more features that can be automatically understood in a video — faces detected, scenes recognized, objects segmented — the better we can shape the user's experience of working with digital video.

In this thesis, however, I focus specifically on motion analysis for user interaction and visualization. I restrict this focus for two reasons: First, motion is a distinguishing feature of video as opposed to still imagery, and poses unique challenges not found in systems dealing with still image understanding and manipulation. Second, motion analysis is a relatively mature sub-field of computer vision with a large body of literature from which to draw algorithms and insights.

Tracking the motion of objects in the scene facilitates video manipulations because operations on a single still frame can act as proxies for operations on an entire clip: Manipulations on one object can be propagated to other frames of the video. Furthermore, as we will see in Chapter 7, object tracking also enables novel interaction modes, by making it possible to navigate the time axis of a video using the motions of objects in the scene.

The design of this system assumes that the user will tolerate a long preprocess in exchange for very rapid user interaction. My motion analysis subsystem therefore first analyzes the video in a fully automatic preprocessing step that tracks the motion of image points across a video and segments those tracks into coherently moving groups.

3.2 Related work

Tracking has previously been used for video manipulation and authoring animations. For example, Agarwala *et al.* [4] demonstrated that an interactive keyframe-based contour tracking system could be used for video manipulation and stroke animation authoring. However, their system required

considerable user intervention to perform tracking. In contrast, my application does not require pixel-accurate tracking or object segmentation, so I can use more fully-automated techniques that do not produce pixel segmentations. In a related vein, the systems of Li *et al.* [50] and Wang *et al.* [94] can be used to segment videos into independently moving objects with considerable accuracy, but do not explicitly recover the transformations of objects over time, and therefore cannot be used to affix annotations. My system, on the other hand, performs tracking and segmentation as an off-line preprocess, so that new annotations can be created at interactive rates.

Buchanan and Fitzgibbon [14] recently presented a near-real-time tracker that employed pre-processing of the video sequence to reduce the computational burden of tracking. In principle such an approach could also be used to author annotations that translate along with objects in the video. However, my system associates annotations with entire regions, not just tracked points, allowing for a variety of transformations of the annotation, including perspective transformations.

My method utilizes the particle video approach of Sand and Teller [72] to densely track points in the video. Object tracking is a widely researched topic in computer vision, and many other tracking approaches are possible; Yilmaz *et al.* [100] recently surveyed the state of the art. Particle video is especially well suited to interactive video applications because it provides a dense field of tracked points that can track fairly small objects, and even tracks points in featureless regions.

My grouping preprocess accomplishes some of the same goals as the object grouping technique of Sivic *et al.* [77], which tracks features using affine-covariant feature matching and template tracking, followed by a grouping method employing co-occurrence of tracks in motion groups. That method has shown significant success at grouping different views of the same object even through deformations and significant lighting changes. However, after some experimentation I found that it has several drawbacks for our application: First, the field of tracked and grouped points is relatively sparse, especially in featureless areas of the image. Second, affine-covariant feature regions are sometimes quite large and may therefore overlap multiple moving objects. Finally, the grouping method relies on segmenting a feature similarity matrix using connected components. This process does not scale well to tens of thousands of tracked particles, not only because of the increased memory requirements but also because the connected components approach is not robust to particles that transition from one group to another due to tracking errors. However, unlike Sivic's approach, my grouping method requires that the number of motion groups is given *a priori*.

Liu *et al.* [53] presented a related approach to motion grouping using a normalized correlation metric between feature trajectories, with spectral clustering applied to the resulting similarity matrix. Like Sivic's method, the memory usage of this approach scales quadratically with the number of features.

A number of previous methods cluster feature motions using either a translation-based distance metric [34, 71, 13] or a piecewise constant motion model [19]. In contrast, my approach uses an affine model that properly groups features in approaching or receding objects.

3.3 Pre-processing

My system consists of two off-line preprocessing stages. In the first off-line preprocess, point particles are placed and tracked over time (Section 3.3.1). Subsequently, I employ a novel grouping mechanism to aggregate particles into consistent moving groups (Section 3.3.2).

3.3.1 Particle tracking

To track particles, I apply the "particle video" long-range point tracking method [72, 71], which I briefly recapitulate:

First, the system computes optical flow on pairs of consecutive frames, using an energy function that includes a smoothness term modulated by the image gradient magnitude, and an occlusion factor that selects occluding boundaries using the divergence of the flow field and pixel projection differences. Bilateral filtering is applied near flow boundaries to improve boundary sharpness.

Then, particles are propagated from one frame to the next using an optimization process that considers the flow field, image intensity, and color channels, and a weighted smoothness term with respect to nearby particles. At each frame, particles with high post-optimization error are pruned, and new particles are added in gaps between existing particles.

The key advantage of the particle video approach over either template tracking or optical flow alone is that it is both spatially dense and temporally long-range. In contrast, feature tracking is long-range but spatially sparse, and optical flow is dense but temporally short-range. Thus, particle video data is ideal for the purpose of attaching annotations, as we can estimate the motion of any pixel into any frame by finding a nearby particle.

In the sections that follow, I will use the following notation: A particle track i is represented by a 2D position $\mathbf{x}_i(t)$ at each time t during its lifetime $t \in T(i)$. The total number of particles is denoted n.

3.3.2 Particle grouping

For certain annotation applications, I find it useful to estimate groups of points that move together over time. My system estimates these groupings using a generative K-affines motion model, in which the motion of each particle is generated by one of K affine motions plus isotropic Gaussian noise:

$$\mathbf{x}_i(t+\Delta t) = A_{L(i)}(t)[\mathbf{x}_i(t)] + \mathbf{n} \tag{3.1}$$

$$\mathbf{n} \sim N(0,\sigma) \tag{3.2}$$

Here $A_k(t)$ represents the affine motion of group k from time t to time $t+\Delta t$, and $\mathbf{n}$ is zero-mean isotropic noise with standard deviation σ. (For notational convenience we write $A_k(t)$ as a general operator, rather than separating out the linear matrix multiplication and addition components.) In my system, $\Delta t = 3$ frames. Each particle has a group label $1 \leq L(i) \leq K$ that is constant over the lifetime of the particle. The labels $L(i)$ are distributed with unknown prior probability $P[L(i) = k] = \pi_k$. I denote group k as $G_k = \{i | L(i) = k\}$.

Our system optimizes for the maximum likelihood model

$$\Theta = (A_1, \ldots, A_K, \pi_1, \ldots, \pi_K, L(1), \ldots, L(n)) \tag{3.3}$$

using an EM-style alternating optimization. Given the above model, the energy function Q simplifies to:

$$Q(\Theta) = \sum_i \sum_{t \in T(i)} \left(\frac{d(i,t)}{2\sigma^2} - \log(\pi_{L(i)}) \right) \tag{3.4}$$

where $d(i,t)$ is the residual squared error $||\mathbf{x}_i(t+\Delta t) - A_{L(i)}(t)[\mathbf{x}_i(t)]||^2$.

To compute the labels $L(i)$, I begin by initializing them to random integers between 1 and K. Then, the following steps are iterated until convergence:

1. Affine motions A_k are estimated by a least-squares fit of an affine motion to the particles G_k.

2. Group probabilities π_k are estimated using the number of particles in each group, weighted by particle lifespan, then normalized so that $\sum_k \pi_k = 1$.

3. Labels $L(i)$ are reassigned to the label that minimizes the objective function $Q(\Theta)$ per particle, and the groups G_k are updated.

In the present algorithm I fix $\sigma = 1$ pixel, but this could be included as a variable in the optimization as well.

The output of the algorithm is a segmentation of the particles into K groups. Figure 3.1 illustrates this grouping on one of our input datasets. Although there are some misclassified particles, the bulk of the particles are properly grouped. My interactive interface, discussed in the following chapter, can be used to overcome the minor misclassifications seen here.

Figure 3.1: Four frames from a video sequence, with particles colored according to the affine groupings computed in Section 3.3.2. (video footage ©2005 Jon Goldman)

3.4 Conclusion

This method is fairly simple and perhaps seems obvious in retrospect, but nonetheless covers new ground. In particular, the long range stability of the particle video tracks simplifies the algorithm with respect to competing techniques. Contrary to expectation, it is not necessary to explicitly include either a temporal regularization of the affine motions or an outlier rejection term in the energy function. Indeed, some more elaborate algorithms performed more poorly than the one described in this chapter: I initially implemented the approach described by Sivic *et al.* [77], but as described in Section 3.2, I found that the relatively large affine-invariant features often overlapped multiple object regions, resulting in those objects being grouped together. Furthermore, the relative scarcity of affine-invariant features makes them less suitable for the types of interactive applications that we hope to support, as I will describe in Chapters 4 and 7 of this thesis.

One important limitation of our system is the length of time required to preprocess the video. In my current implementation, the preprocess takes about 10 minutes per frame of 720×480 input video, which is prohibitive for some of the potential applications described here. Although most of the preprocess is heavily parallelizable, and moderate accelerations can be attained by tuning the code, novel algorithms will be necessary for applications requiring a tighter loop between video capture and interaction.

Another limitation is that our grouping mechanism does not enforce spatial coherence, imposing some additional burden of effort on the user in cases where the motion alone is not sufficient to separate multiple objects. I would like to explore the use of spatial constraints and image properties in our grouping algorithm, to combine the benefits of existing pixel segmentation algorithms with my motion segmentation approach.

My method's operating range is largely constrained by that of the underlying particle video method. This approach works best on sequences with large textured objects moving relatively slowly. Small moving objects are hard to resolve, and fast motion introduces motion blur, which causes particles to slip across occlusion boundaries. Although the particle video algorithm is relatively robust to small featureless regions, it can hallucinate spurious coherent motion in large featureless regions, which may be interpreted as separate motion groups in the object grouping stage. Tracking and grouping videos for all the examples shown in the paper and video are available on-

line [29].

The particle video approach can also exhibit some "drift" of particles from their correct positions over very long frame ranges, especially on strongly deforming objects or perspective shifts. Repairing such drift using information from distant frames is an interesting topic for future work.

Chapter 4

VIDEO ANNOTATION

4.1 Introduction

Graphical annotations of video are sometimes used by video authors to augment media for informational or entertainment purposes. But although there are many practical approaches for graphically annotating static images, applying such annotations to arbitrary *video* material is a more challenging task, since the objects or regions move across the screen over time. One approach is the "telestrator," the device used by American football broadcasters to overlay hand-drawn diagrams over still images of sports video (see Figure 4.1). The telestrator simply allows a static diagram to be drawn atop the video. Typically, these diagrams are removed from the screen while the video is playing, because their location on the screen is aligned with the field or the players only for the single frame upon which they were originally sketched. Another approach is employed by visual effects software such as Adobe After Effects, in which user-specified regions of the image are tracked — typically using normalized cross-correlation template tracking — and various annotations can subsequently be attached to transform along with the tracked regions. In the hands of a skilled artist, this approach can result in annotations that appear "locked" to objects in the video, but it can require substantial manual labor to select the points to be tracked and to correct tracking errors.

In the previous chapter I described an approach to track a large number of points across a video sequence and group them according to common motion. In this chapter I describe a system that makes it easy to author annotations that transform along with objects or regions in an arbitrary video. The precomputed motion and groupings of the tracked points are then used to drive an interactive annotation interface. I call these annotations "video object annotations" (VOAs) because they are associated with specific objects or regions of the video, unlike telestrator-style annotations that are simply overlaid at a given screen location.

I envision video object annotations being used in any field in which video is produced or used

to communicate information. Telestrator-like markup can be useful not only for sports broadcasting but also for medical applications, surveillance video, and instructional video. Film and video professionals can use VOAs to communicate editorial information about footage in post-production, such as objects to be eliminated or augmented with visual effects. VOAs can also be used to modify existing video footage for entertainment purposes with speech and thought balloons, virtual graffiti, "pop-up video" notes, and other arbitrary signage. In this chapter, I demonstrate results for several of these applications. Finally, my interface also naturally lends itself to a variety of other applications, such as hyperlinked video authoring.

In this chapter I describe a pipeline for video processing that greatly enriches the space of interactions that are possible with video. These interactions are enabled in part by the feature grouping algorithm described in the previous chapter.

4.2 Related work

The telestrator [98] is a key point of reference for my system. The telestrator was invented in the late 1960s by physicist Leonard Reiffel for drawing annotations on a TV screen using a light pen. It first became popularly known in 1982 when it was used by American football commentator John Madden during the televised instant replays for Super Bowl XVI and is therefore often colloquially known as a "John Madden-style whiteboard." A similar approach has also been adopted for individual sports instruction using systems like ASTAR [9] that aid coaches in reviewing videos of athletic performance. However, as previously mentioned, annotations created using a telestrator are typically

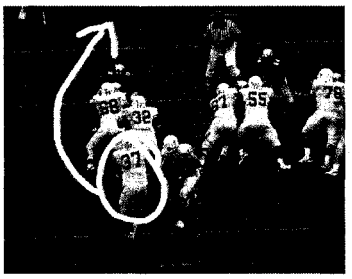


Figure 4.1: A typical telestrator illustration during a football broadcast. (©Johntex, Creative Commons license [98])

static, and do not overlay well on moving footage.

In recent years, broadcasts of many professional sporting events have utilized systems supplied by Sportvision [81] to overlay graphical information on the field of play even while the camera is moving. Sportvision uses a variety of technologies to accomplish these overlays. In most cases, the playing or racing environment and cameras are carefully surveyed, instrumented, and calibrated before play begins. In addition, objects such as hockey pucks and race cars are instrumented with transmitting devices of various kinds so that their locations can be recovered in real time. Finally, chroma keying is sometimes used to matte the players, so that the annotations appear to be painted directly on the field of play. Although this instrumentation and calibration allows graphics to be overlaid in real time during the broadcast, it requires expensive specialized systems for each different class of sporting event, and is not applicable to pre-existing video acquired under unknown conditions.

The visual effects industry has also adopted the concept of the telestrator for facilitating communications between directors and visual effects artists. A product called cineSync [18] allows individuals in multiple locations to share control of a video file that has been previously transmitted to each location. Both parties can scrub the video and draw annotations on the screen. Because the actual video data is preloaded onto the client computers, no video data is being transmitted during the session. Therefore very little network bandwidth is required: sessions can even be run over 56K modem connections. However, as with a telestrator, the annotations are only associated with still frames.

Thought and speech balloons have previously been employed in virtual worlds and chat rooms [59, 47], in which the associated regions are known a priori. Kurlander *et al.* [47] specifically address the problem of balloon layout. However, my system allows association of thought and speech balloons with video objects for which the position and movement is not known beforehand.

My system includes an optimization of annotation location (Section 4.3.2) that balances proximity to the target with overlap of important features of the image. Related systems have been developed by Thanedar and Höllerer [88] for pre-recorded video, and by Rosten *et al.* [68] for augmented reality displays. My approach in this respect is similar to the work of Rosten *et al.*, but adapts the optimization to the case of pre-recorded video while retaining interactive speeds. Unlike Thanedar and Höllerer, who apply low-level video features to detect regions of low importance, our

system uses feature groupings to explicitly detect moving objects in the scene.

I am not the first to propose the notion of hyperlinked video as described in Section 4.3.3. The earliest reference of this to my knowledge is the Hypersoap project [21]. However, the authoring tool proposed in that work required extensive user annotation of many frames. I believe my system offers a significantly improved authoring environment for this type of rich media.

4.3 Interaction

The tracked particles and group labels described in Chapter 3 can be used for a variety of applications. My interactive annotation interface is patterned after typical drawing and painting applications, with the addition of a video time-line. The user is presented with a video window, in which the video can be scrubbed back and forth using either a slider or a novel direct manipulation interface described below. A toolbox provides access to a number of different types of VOAs, which are created and edited using direct manipulation in the video window.

4.3.1 Video selection

To annotate an object, the user must specify a type of annotation and the object to which the annotation is to be attached. The object specification task is closely related to the problems of interactive video segmentation and video matting [4, 17, 50, 94]. However, these other methods are principally concerned with recovering a precise matte or silhouette of the object being selected. In contrast, the goal of our system is to recover the *motion* of a particular object or region of the image, without concern for the precise location of its boundaries.

In my system, the user creates VOAs simply by painting a stroke s or dragging a rectangle over the region $\mathbf{R}_s$ of the image to which the annotation should be attached. This region is called the annotation's *anchor region*, and the frame on which it is drawn is called the *anchor frame*, denoted t_s. The anchor region defines a set of *anchor tracks* that control the motion of that annotation. For some applications, it suffices to define the anchor tracks as the set of all particles on the anchor frame that lie within the anchor region:

$$\mathbf{A}(s) = \{i | t_s \in T(i), \mathbf{x}_i(t_s) \in \mathbf{R}_s\} \tag{4.1}$$

However, this simplistic approach to selecting anchor tracks requires the user to scribble over a

potentially large anchor region. The system can reduce the amount of user effort by employing the particle groupings computed in Section 3.3.2. The interface uses the group labels of the particles in the anchor region to infer entire group selections, rather than individual particle selections. To this end, the system supports two modes of object selection. First, the user can click once to select the group of points of which the closest track is a member. The closest track i' to point $\mathbf{x}_0$ on frame t_0 is located as:

$$i'(\mathbf{x}_0, t_0) = \text{argmin}_{\{i|t_0 \in T(i)\}} ||\mathbf{x}_0 - \mathbf{x}_i(t_0)||, \tag{4.2}$$

and the selected group is simply $G_{L(i'(\mathbf{x},t))}$. Second, the user can make a "sloppy" selection that includes points from multiple groups. The resulting selection consists of the groups that are well represented in the anchor region. Each group is scored according to the number $||\mathbf{A}_k(s)||$ of its particles in the anchor region s. Then any group whose score is a significant fraction T_G of the highest scoring group is accepted:

$$\mathbf{A}_k(s) = G_k \cap \mathbf{A}(s) \quad \forall 1 \leq k \leq K \tag{4.3}$$

$$S_k(s) = \frac{|\mathbf{A}_k(s)|}{\max_{1 \leq k \leq K} |\mathbf{A}_k(s)|} \tag{4.4}$$

$$\mathbf{G}(s) = \bigcup_{k|S_k(s) >= T_G} G_k \tag{4.5}$$

The threshold T_G is a system constant that controls the selection precision. When $T_G = 1$, only the highest-scoring group $\text{argmax}_k |\mathbf{A}_k(s)|$ is selected. As T_G approaches 0, any group with a particle in the selected region will be selected in its entirety. I have found that $T_G = 0.5$ gives very intuitive results in most cases.

My system's affine grouping mechanism may group particles together that are spatially discontiguous. However, discontiguous regions are not always appropriate for annotation. To address this only the particles that are spatially contiguous to the anchor region are selected. This is achieved using a precomputed Delaunay triangulation of the particles on the anchor frame.

By allowing more than one group to be selected, the user can easily correct the case of oversegmentation, such that connected objects with slightly different motion may have been placed in separate groups. If a user selects groups that move independently, the attached annotations will simply be a "best fit" to both motions.

Using groups of particles confers several advantages over independent particles. As previously

mentioned, user interaction is streamlined by employing a single click or a loose selection to indicate a moving object with complex shape and trajectory. Furthermore, the large number of particles in the object groupings can be used to compute more robust motion estimates for rigidly moving objects. The system can also display annotations even on frames where the original particles no longer exist due to partial occlusions or deformations. (However, my method is not robust to the case in which an entire group is occluded and later becomes visible again. This is a topic for future work.)

When an object being annotated is partially occluded, the system should be able to modify its annotation's appearance or location, either to explicitly indicate the occlusion or to move the annotation to an un-occluded region. One indication of occlusion is that the tracked particles in the occluded region are terminated. Although this is a reliable indicator of occlusion, it does not help determine when the same points on the object are disoccluded, since the newly spawned particles in the disoccluded region are not the same as the particles that were terminated when the occlusion occurred. Here again we are aided by the grouping mechanism, since it associates these points on either side of the occlusion as long as there are other particles in the group to bridge the two sets. To determine if a region of the image instantiated at one frame is occluded at some other frame, the system simply computes the fraction of particles in the transformed region that belong to the groups present in the initial frame. An annotation is determined to be occluded if fewer than half of the points in the region belong to the originally-selected groups. Figure 4.2 shows a rectangular annotation changing color as it is occluded and disoccluded.

4.3.2 Video annotations

My system supports four types of graphical video object annotations. The types are distinguished both by their shape and by the type of transformations they use to follow the scene. In each case, the transformations of the annotation's anchor tracks are used to determine the appearance and/or transformation of the annotation.

Given the anchor tracks, transformations between the anchor frame and other frames are computed using point correspondences between the features on each frame. Some transformations require a minimum number of correspondences, so if there are too few correspondences on a given frame — for example because the entire group is occluded — the VOA is not shown on that frame.

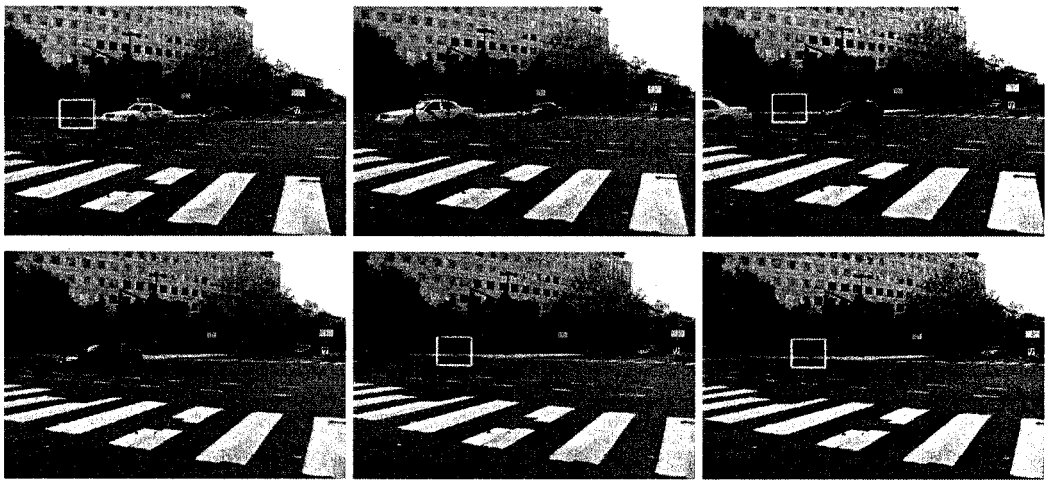


Figure 4.2: A rectangle created on the first frame remains affixed to the background even when its anchor region is partially or completely occluded. The annotation changes from yellow to red to show that it is occluded.

At present, I have implemented prototype versions of "scribbles," "graffiti," "speech balloons," and "path arrows."

Scribbles. These simple typed or sketched annotations just translate along with the mean translation of anchor tracks. This annotation is ideal for simple communicative tasks, such as local or remote discussions between collaborators in film and video production.

Graffiti. These annotations inherit a perspective deformation from their anchor tracks, as if they are painted on a planar surface such as a wall or ground plane. Given four or more non-collinear point correspondences, a homography is computed using the normalized DLT method described by Hartley and Zisserman [32]. An example of a graffiti annotation is shown in Figure 4.3.

When the user completes the drawing of the anchor regions, the transformations of graffiti annotations are not computed for all frames immediately, but are lazily evaluated as the user visits other frames. Further improvements to perceived interaction speed are possible by performing the computations during idle callbacks between user interactions.

Word balloons. My system implements speech and thought balloons that reside at a fixed location on the screen, with a "tail" that follows the annotated object. The location of the speech balloon is optimized to avoid overlap with foreground objects and other speech balloons, while remaining

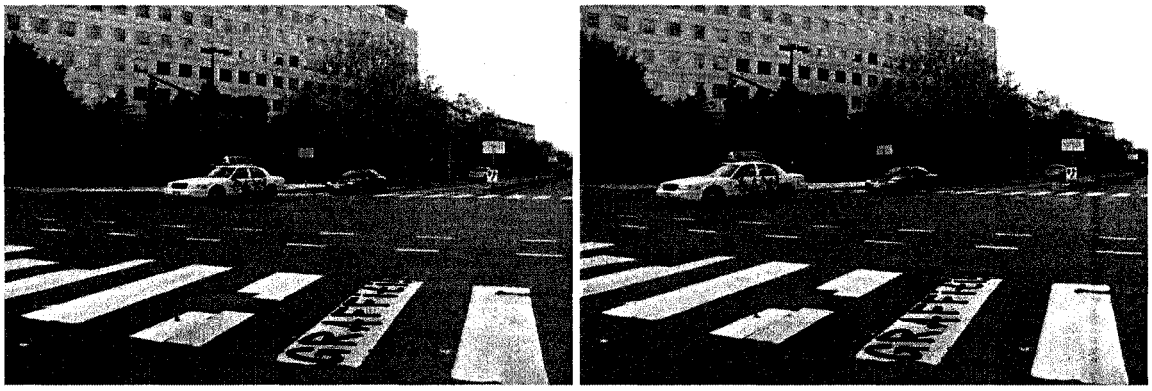


Figure 4.3: Two "graffiti" annotations attached to a car and a road at two different frames in a sequence.

close to the anchor tracks. Inspired by Comic Chat [47] and Rosten *et al.* [68], I optimize an energy function with a distance term E_d, overlap term E_o, and collision term E_c:

$$E = c_d \sum_a E_d(a) + c_o \sum_a E_o(a) + c_c \sum_{a,b} E_c(a,b) \tag{4.6}$$

where a and b index over the speech balloon annotations.

The distance term E_d simply measures the distance of the speech balloon's mean position $\mathbf{x}_a$ from its anchor tracks:

$$E_d(a) = \sum_t \sum_{i \in \mathbf{A}_a} ||\mathbf{x}_a - \mathbf{x}_i(t)||^2 \tag{4.7}$$

The overlap term E_o measures the amount by which the annotation overlaps foreground objects:

$$E_o(a) = \sum_{t \in a} \sum_{\mathbf{x} \in a} f(\mathbf{x},t)/V(a) \tag{4.8}$$

$$f(\mathbf{x},t) = \begin{cases} 1, & i'(\mathbf{x},t) \in \mathbf{G}_{fg} \\ 0, & i'(\mathbf{x},t) \notin \mathbf{G}_{fg} \end{cases} \tag{4.9}$$

where $V(a)$ is a normalization term representing the spatio-temporal volume (width $\times$ height $\times$ duration) of the annotation and i' is the closest track as defined in equation (4.2). Here I use the notational shorthand $\mathbf{x} \in a$ to refer to points inside the balloon region, and $t \in a$ to refer to the balloon's duration of existence. By default, my system defines background points as those belonging to the largest group $\mathbf{G}_{bg} = G_{\text{argmax}_k ||G_k||}$, and all other points belong to the foreground ($\mathbf{G}_{fg} =$

Figure 4.4: Two word balloons with screen position optimized to minimize overlap with the actors throughout the shot. (video footage ©2005 Jon Goldman)

$\{i|i \notin \mathbf{G}_{bg}\}$), but the user can easily indicate different foreground and background groups using the selection mechanism described in Section 4.3.1.

Finally, the collision term E_c measures the amount by which a pair of annotations overlap, and it is computed analogously to E_o.

Many terms in the energy function can be factored and/or approximated in order to accelerate computation. For example, notice that Equation 4.7 can be rearranged as:

$$E_d(a) = N||\mathbf{x}_a||^2 - 2\mathbf{x}_a^T \sum_{t,i} \mathbf{x}_i(t) + \sum_{t,i} ||\mathbf{x}_i(t)||^2 \tag{4.10}$$

The third term is a constant over the optimization, and can therefore be omitted from the energy function, and the sum in the second term can be computed once before the optimization process. The computation of E_o can be accelerated by precomputing summed area tables for the expression in the inner loop, and by approximating the shape of a word balloon using its bounding rectangle. E_c can also be computed in constant time for the case of rectangular regions. I apply Nelder-Mead simplex optimization to find the optimum balloon locations. Using the aforementioned accelerations, this optimization takes only a few seconds for the two word balloons in Figure 4.4. (See the supplementary video [29] for a real-time demonstration.) In our implementation, $c_o = c_c = 10000$, and $c_d = 1$.

I also experimented with animated word balloons that only translate or translate and scale with their anchor tracks, and also using a global optimization over all frames with a constraint to enforce

smooth motion. However, I found that word balloons moving about the screen were difficult to read, even when moving quite slowly. My present implementation is therefore designed to maximize legibility at the risk of some overlap and awkward tail crossings. In the rare case in which annotated objects change location dramatically on the screen, e.g., by crossing over each other from left to right, this implementation may result in undesirable layouts with crossed tails or balloons that do overlap foreground objects. However, I note that it is extremely atypical in modern cinematography for characters to swap screen locations while talking. In reviewing several full length and short films I found fewer than a dozen such shots. In every case the dialog was not fully overlapping, so that word balloons could appear over different frame ranges in order to avoid occlusions and crossed tails.

Path arrows. These annotations highlight a particular moving object, displaying an arrow indicating its motion onto a plane in the scene. To compute the arrow path I transform the motion of the centroid of the anchor tracks into the coordinate system of the background group in each frame. The resulting path is used to draw an arrow that transforms along with the background. By clipping the path to the current frame's edges, the arrow head and tail can remain visible on every frame, making it easy to determine the subject's direction of motion at any time. My system's result can be seen in Figure 4.5. I believe this type of annotation could be used by surveillance analysts, or to enhance telestrator-style markup of sporting events.

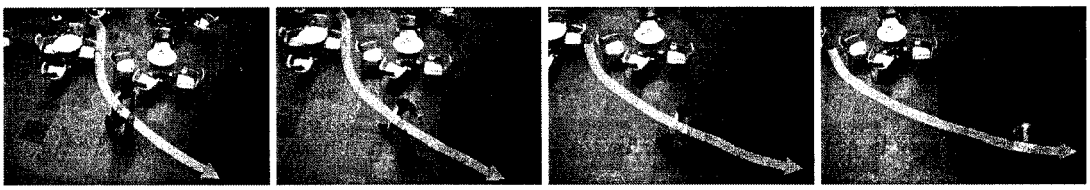


Figure 4.5: An arrow highlighting the motion of a walking person.

4.3.3 Multimedia authoring

My system can also be used to author alternative media representations of the original video, such as hyperlinked video [21].

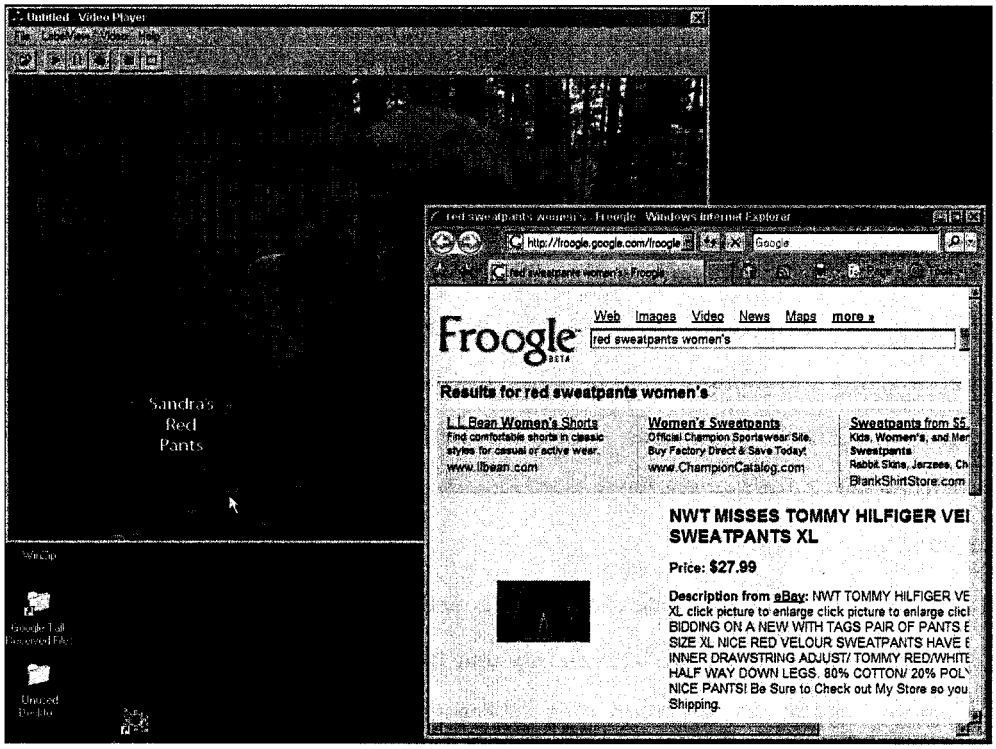


Figure 4.6: A video with highlighted hyperlinks to web pages. (video footage ©2005 Jon Goldman)

A prototype hyper-video player using our system as a front end for annotation is shown in Figure 4.6, and can also be seen in the supplementary video [29]. When viewing the video on an appropriate device, the user can obtain additional information about objects annotated in this way, for example, obtaining price information for clothing or other depicted items, or additional references for historically or scientifically interesting objects. As a hyperlink, this additional information does not obscure the video content under normal viewing conditions, but rather allows the viewer to actively choose to obtain further information when desired. The hyperlinked regions in this 30-second segment of video were annotated using our interface in about 5 minutes of user time.

4.4 Discussion

I have presented a system for interactively associating graphical annotations to independently moving video objects. My contributions include the application of an automated preprocess for video interaction, and a fluid interface for creating graphical annotations that transform along with associated video objects. The assembly of a well-integrated system enabling heretofore unseen approaches to video markup and interaction is, in itself, an important and non-obvious contribution. My system performs all tracking and grouping offline before the user begins interaction, and my user interface hides the complexity of the algorithms, freeing the user to think in terms of high level goals such as the objects to be annotated and the types and contents of annotations, rather than low-level details of tracking and segmentation.

A primary benefit of this approach over existing methods for annotating video is that we perform tracking as an off-line preprocess. This means that the user does not need to be in the loop for selecting regions to be tracked and correcting the results. Instead, the system requires the user only to perform higher level tasks such as selecting objects to be annotated and the types of annotations to be used. Furthermore, the grouping preprocess allows for rapid and intuitive selection of coherently moving regions.

Although traditional image mosaicking techniques can be used to scrub video by clicking on points in the background [37], my approach permits manipulations that can't be achieved using previous mosaicking approaches, such as those shown in Figure 7.2.

To assess the performance of my annotation approach, I asked a novice user to create a translating text annotation using After Effects, a commercial motion graphics and compositing tool with a tracking component. With the assistance of a moderately experienced After Effects user, the novice spent 20 minutes initially preparing the single-point (translation-only) tracked text. The interaction required dozens of mouse clicks and motion all over the screen, whereas my interface requires just 1 click (to select the text tool), one click-drag motion (to select anchor region and text position) and a text entry box, and can be performed in seconds. With three additional clicks, the text can be placed (and transformed) in perspective, as shown in the demonstration video [29]. Furthermore, an After Effects user must manually locate trackable features and select the search region, wait several seconds (or more for long sequences) while tracking is performed, and, if the tracking fails, must

manually correct it or remove the badly tracked frames from further computations. An expert After Effects user might be able to accomplish this in a few minutes, still orders of magnitude slower than what is possible using my interface. This is because my system has performed all tracking and grouping offline before the user begins interaction, and the only remaining task for the user is to gesturally indicate video objects, annotation styles and contents.

4.5 Future work

I believe my approach to interactive video annotation may have a number of applications in a variety of domains. I envision the following scenarios as a sampling of future extensions to my work:

In film and video production, interactive annotations can be used by editors and directors to communicate about objects and individuals by making markings directly on their footage. A director can simply circle the objects she wants removed or emphasized, and the annotation is viewable on all frames of the video instantly, remaining aligned with the object of interest. My technique is easily adapted to remote review settings like that supported by cineSync, since the data associated with each annotation is quite small and can be transmitted in real time over a network connection.

In sports broadcasting, video object annotations could be used to supplant existing telestrator technologies, with the advantage that the annotations can remain on the screen while the video is playing. Furthermore, the lines drawn to illustrate player motion could be generated automatically like a storyboard. In contrast to the Sportvision technologies, no complex and expensive instrumentation of the field of play and the competitors is necessary, so my approach is potentially applicable to historical sports video, lower-budget high school and collegiate sporting events, or even individual sports instruction. However, as noted in Section 3.4, the processing time required by the current preprocessing algorithm is prohibitive for "instant replay" types of applications, and further work is needed to reduce the computational burden to a manageable scale.

Additional applications may be possible by integrating my system with the approach of Sivic *et al.* [77] to associate the same object across occlusions, in multiple shots, or in multiple videos taken from different cameras. For example, one can imagine extending the Photo Tourism concept [79] to Video Tourism. A viewer could nonlinearly travel through multiple videos that captured the same environment or event, like Michael Naimark's "Moviemaps" of Aspen and Banff. In addition, text

or graphical annotations could be propagated from photos to videos, or from one video to another.

Video object annotations can be used to annotate the objects and processes in an assembly instruction video. If the end user also has a camera, the user's current stage of assembly could be detected, and the instructional video synchronized to the appropriate step. A similar technique could be used to synchronize a video of a walking tour to the user's current location.

In another application, an analyst reviewing surveillance video could easily mark individuals of interest for further review using our system. The video may be automatically re-centered or cropped and zoomed on these individuals for easier review. If the scene has been filmed with multiple surveillance cameras, it may also be possible to click on an individual and propagate the annotation into the other videos. Finally, the path of a single individual over time might be followed across multiple cameras automatically.

In conclusion, I believe interactive video object annotations can become an important tool for augmenting video as an informational and interactive medium.

Chapter 5

SCHEMATIC STORYBOARDING

5.1 Introduction

In this chapter we will see a specific instance of video annotation for the purpose of quickly producing a visualization of a video clip.[1] Such a visualization can act as a proxy for the video clip itself in video editing interfaces.

Video editing is a time-consuming process, due in part to the sheer volume of video material that must be repeatedly viewed and recalled. Video editing software offers only token mnemonic assistance, representing each video segment with only a single frame thumbnail, which typically defaults to the segment's first frame. But this frame is often unrepresentative of the video content, and does not illustrate other important aspects of the video, such as camera and subject motion.

Indeed, this is not a new problem: since the beginning of motion pictures, it has been necessary for filmmakers to communicate with each other and their crew members about moving compositions, and they have invented a special type of diagram — the *storyboard* — for this purpose. Although the dictionary definition of a storyboard is just a sequence of still frames representing a moving sequence, storyboard artists have developed a distinct visual vocabulary to concisely summarize and annotate moving compositions. Filmmakers may use the term, "storyboard," to refer to either type of illustration, so I employ the term, *schematic storyboard*, specifically to describe the annotated storyboard.

Schematic storyboards are static — like a filmstrip — but organized and annotated to convey continuity and directionality — like an animation. A key advantage of schematic storyboards is that a significant time interval of the video content can be expressed all at once. In contrast, the simple act of observing an animated display takes a certain length of time: A ten-minute shot generally takes ten minutes of a user's time to observe in its entirety. Of course, one can always fast-forward through a video (in effect scaling the data along the temporal axis), but as playback speed increases,

[1] The work described in this chapter was originally presented as a paper [28] at SIGGRAPH 2006.

it becomes more difficult to observe details of the motion.

In addition, schematic storyboards are particularly well-suited for applications such as video editing, in which many clips must be observed and mentally processed in parallel. An animated display is awkward to use in such applications, since the human visual system can be quickly overwhelmed by even small numbers of video streams playing simultaneously: A rapid motion in one video may distract the observer's attention from small but important motions in another video playing simultaneously.

In this work, I restrict myself to the problem of visualizing footage already segmented into *shots*, where a shot is a sequence of frames captured over a continuous interval of time. I have developed a system to assist the user in creating a schematic storyboard that summarizes an entire shot in a still image, using a visual representation that can communicate high-level aspects of motion of camera and subject, without requiring a moving display. As a still image, it can be printed out on a piece of paper or a booklet for reference on a soundstage or on location. It can also be used as an interface for navigating the temporal axis of the original footage in an intuitive fashion.

The problem of automatically producing storyboards from video is a difficult and wide-ranging challenge, spanning aspects of computer vision, artificial intelligence, visualization, and human-computer interfaces. This work addresses only a small part of that broad spectrum, interleaving some user interaction for frame and object selection with automated tracking, layout, annotation, and compositing. Because of this user interaction, I emphasize applications such as stock footage search, in which video clips may be re-used in multiple settings over long periods of time, thus amortizing the cost of the user interaction over many uses.

My contributions include introducing a formal summary of the visual language of storyboards as a tool for visualization, and incorporating many of these principles as part of a semi-automated layout and rendering system. Although there are many books on storyboards, I believe that the summary in this chapter is the first concise "rule book" for the use of storyboard annotations.

In Section 5.2 I discuss some previous work in related areas. Section 5.3 summarizes some of the more common notations of storyboards, and Section 5.4 describes the translation of these notations into an automated system, which is elaborated in Section 5.5. Results and applications are given in Sections 5.6 and 7.3. Section 5.7 concludes with applications, limitations, and future work.

5.2 Related work

Artists and scientists employ a variety of methods to illustrate motion in static images. Cutting [20] catalogued five distinct solutions historically employed by artists: dynamic balance, multiple stroboscopic images, affine shear/forward lean, photographic blur, and image and action lines. Ward [95] discussed the role of body pose in the depiction of motion. Cartoonists employ these and other tools for depicting motion, including "speedlines," adjusting the shape and size of panels, and bleeding panels off the page [58]. Masuch *et al.* [57] have applied speedlines to computer animation, and Kawagishi *et al.* [42] incorporate additional techniques such as geometric deformation. Joshi and Rheingans [40] explored the use of speedlines to illustrate the evolution of features in volumetric simulations. Kim and Essa [43] use these and other techniques to create a non-photorealistic expression of motion in a video.

I have chosen the art and techniques of production storyboards as an ideal iconography for video visualization purposes. Storyboards have been used since the dawn of filmmaking [31] to articulate and communicate concepts of image composition and scene blocking. This is not the first work to explicitly adopt this visual language — Nienhaus and Döllner [62] previously adopted storyboard-style 3D arrows to depict dynamics in 3D scenes — but my work is the first to apply storyboard techniques to video visualization.

Salient Stills [86] represents one of the first works to attempt video summarization in a single image. In particular, Massey and Bender [56] noted the applicability of Salient Stills for conveying camera and subject motion. More recently, Freeman and Zhang [24] used stereo imaging to merge multiple frames of video into a single image as if they occupied the same space simultaneously. Agarwala *et al.* [3] seamlessly merged multiple images using a variety of user-specified criteria, and demonstrated the application of their system to several types of time-varying scenes. A related approach was proposed by Wexler and Simakov [96]. Assa *et al.* [8] considered the problem of pose or keyframe selection for composing a storyboard-like extended frame. Like my work, all these systems combine multiple moments in time into a single still image, but from a single viewpoint and without the schematic annotations of our system.

Although it is not the focus of my work, the problem is related to that of video abstraction or summarization, which attempts to create a compact abstract (either still or animated) of a large col-

lection of video. The literature on this topic is large, but Li *et al.* [49] surveyed the field. Irani and Anandan [37] created a system for summarizing and indexing surveillance video that shares some common goals with my work. The PanoramaExcerpts system [85] summarizes large collections of video using both single frames and panoramic mosaics. My work attempts to extend the expressiveness of these types of static summaries using storyboard annotations.

I assume that our video material has already been segmented into individual shots. This segmentation can be done manually, but many automatic methods have also been developed [61, 35, 15, 48].

5.3 The visual language of storyboards

I propose visualization of video in a single static storyboard diagram. This schematic storyboard is designed to communicate high-level motion of the observer and observed objects, abstracting away details that may be less important for understanding motion. At the same time, the storyboard should relate in an intuitive way to the original video. I use the term, *schematic storyboard*, to describe storyboards that combine both pictorial and diagrammatic elements such as arrows and frame outlines.

The rules of composition for these elements are complex and fluid, but I have identified some of the key stylistic conventions shared by several books on traditional storyboarding [76, 11, 31, 12, 41]. I also received assistance from Peter Rubin, a visual effects art director and former professional storyboard artist, who created the hand-drawn storyboards in this chapter and refined my understanding and implementation of storyboard notation.

I attempt to formalize some of the specific techniques used by storyboard artists in the remainder of this section. My summary spans a broad range of storyboard conventions. In Section 5.5 I will narrow my scope, applying a few key idioms to generate storyboards from video with a small amount of user input.

In the discussion that follows, I refer to the primary object of interest as the "subject" of a shot. The subject is often — but not always — in the foreground, and may be moving relative to the static environment.

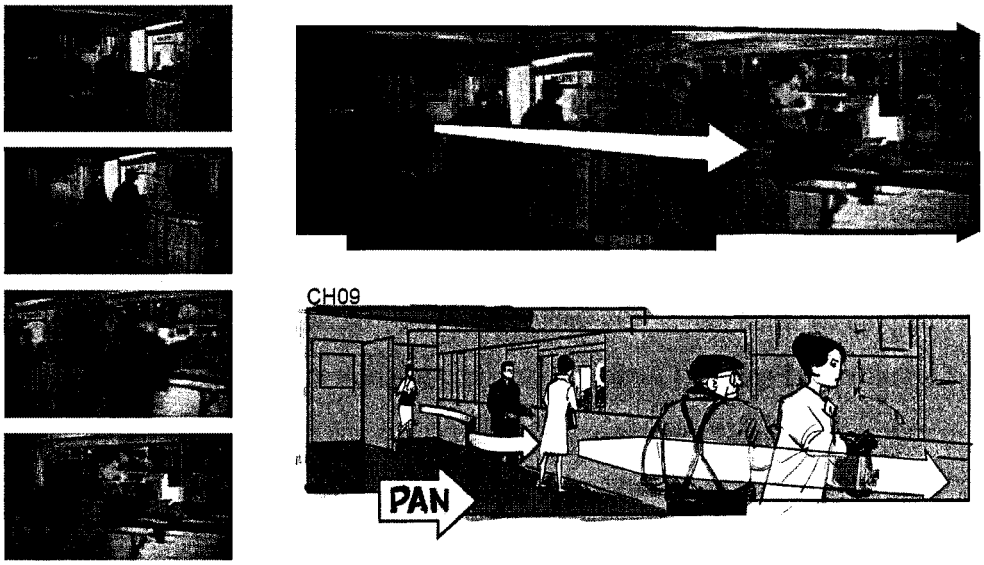


Figure 5.1: Left: Four still frames of a shot from the 1963 film, *Charade*. Top: Schematic storyboard for the same shot, composed using the four frames at left. The subject appears in multiple locations, and the 3D arrow indicates a large motion toward the camera. The arrow was placed and rendered without recovering the 3D location of the subject. Bottom: Hand-drawn storyboard for the same shot, composed using Adobe Photoshop and Corel Painter by a professional storyboard artist. (Credit: Peter Rubin.)

Key frames. Storyboards typically depict several "key" moments in the time span of a shot. The depicted moments in time represent some or all of the following qualities:

- extrema of motion;
- "representative" poses of a subject, which are in some sense similar to a large range of the observed poses;
- clarity of expression or pose; and
- dynamic balance, suggesting the motion in progress.

Also, different objects or individuals in the scene may be depicted at different moments in time in order to more fully optimize these criteria.

Extended frame. An extended frame is an arrangement of multiple frames on a screen or page. The frames are arranged so that the background appears continuous. Typically, standard planar projections are used, but different regions of the extended frame may have different perspective

projections. Changes in perspective are typically hidden in featureless regions or at architectural boundaries. Figure 5.1 features one such extended frame composition.

In contrast to multiperspective panoramas [99], a storyboard is intended to be viewed in its entirety at a single orientation. Therefore, even if the best alignment between multiple frames includes a rotation, all frames in an extended frame are placed on the page or screen without rotation. (One rare exception is when the camera undergoes a large and rapid change of roll angle over the duration of a shot.) See Figure 5.4 for a comparison of alignments with and without rotations.

Note that arranging many frames in a single extended frame may sacrifice clarity and legibility. Therefore, storyboard artists use extended frame sparingly, typically only when the camera and/or subject are moving smoothly and the image composition changes from the beginning to the end of the shot. However, extended frame compositions are split into smaller segments or even individual frames if the resulting composition would sacrifice clarity. Such a confusing composition may result from:

- poor alignment between frames;
- large scale changes between frames due to changes in focal length or motion of the camera; or
- motion of the camera or subject that "backtracks," so that distant moments in time would obscure each other.

In addition, storyboard artists may break an extended frame into several segments in order to avoid wasted space on the page. Figure 5.10 uses multiple segments due to large forward motion of the camera, while Figure 5.6 uses multiple segments due to the subject backtracking in her path, first travelling left to right and then right to left.

Motion arrows. Storyboard artists often augment the subjects of the frames with 3D arrows that roughly follow the path of motion of the camera or subject. These arrows are usually rendered as if they were solid or semi-transparent objects in the scene itself, using different line styles or shading to distinguish the motion paths of different subjects. Motion arrows provide a more definitive sense of direction of motion than speedlines, motion blur, or some of the other mechanisms previously discussed. Furthermore, they can describe additional degrees of freedom, having both thickness and "twist" that may vary over the length of the arrow. See Figure 5.2 for a few of the many styles

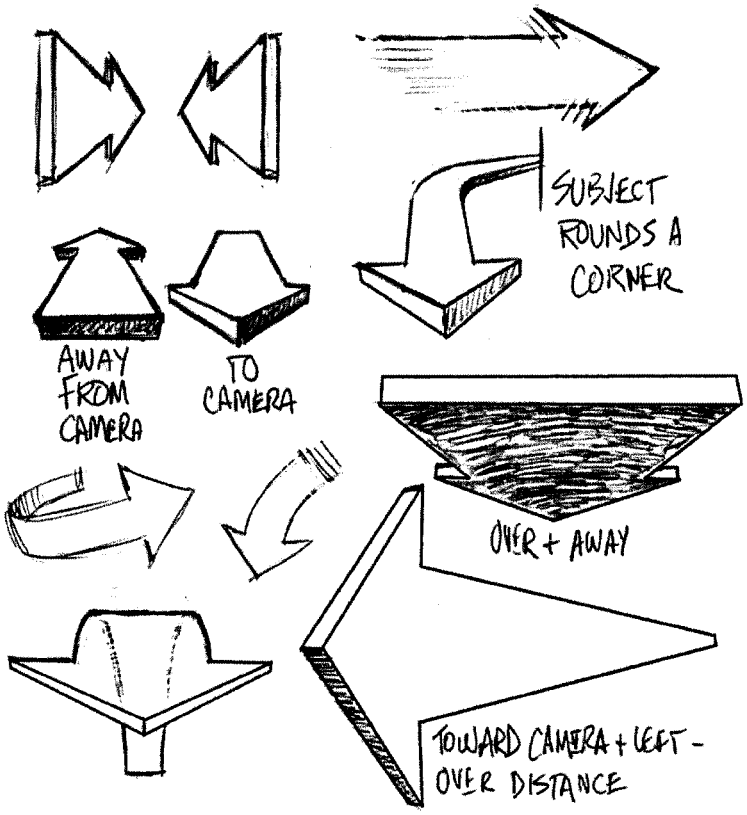


Figure 5.2: **Arrows.** A variety of arrow styles used by storyboard artists. Credit: Peter Rubin.

of motion arrows in storyboards. Many storyboards observe the following conventions for motion-arrow depiction:

- Arrows are piecewise smooth, emphasizing broad motion rather than small details.
- Arrows never obscure important objects in the scene.
- For subjects rotating about their direction of translation ("rolling") — e.g., a banking aircraft — the arrow twist varies over its length, and maintains alignment with the horizontal or vertical plane of the subject.
- For subjects that do not roll — e.g., a person or animal — the arrow twist may either be aligned to the subject's horizontal or vertical plane, or aligned so as to maximize the thickness of the arrow as seen from the camera.

- The arrow's width in the image is approximately proportional to the subject's size in the image. A change in perspective may be exaggerated to emphasize the motion.
- Arrows are omitted if the motion is short or self-evident.
- When multiple subjects move in the same general path, a single arrow may be used to represent their aggregate motion.
- If the subject referred to by an arrow is ambiguous, the arrow may include a textual label. (This is often the case for arrows indicating camera motion, since the camera itself is not visible.)
- Arrows indicating camera motion typically cross or meet the frame boundary.

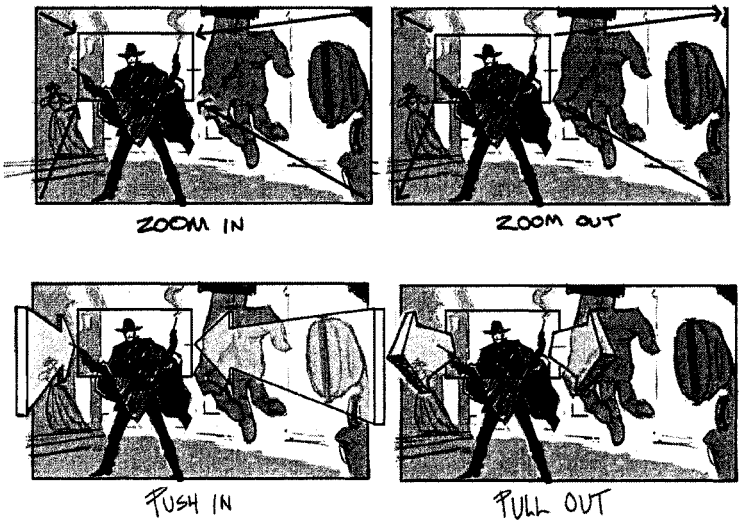


Figure 5.3: **Zoom lines and dolly arrows.** Storyboards illustrating changes in focal length (top) and camera motion (bottom). Credit: Peter Rubin.

Zoom lines. Changes in focal length (zooms) are denoted by concentric sets of frame lines, using 2D arrows to indicate the direction of zoom (see Figure 5.3). The frame lines are typically unbroken (even if they intersect important details of the image), but the arrow heads and tails may be offset from the corners of the frames in order to improve legibility.

Depth ordering. The subjects depicted in extended frames are composed in depth order, with closer subjects appearing in front of more distant subjects, regardless of temporal ordering. Motion arrows are also rendered in depth order, unless they would be too heavily obscured by closer objects. The background of the storyboard is depicted as a continuous scene, hiding changes in perspective in featureless areas or along architectural boundaries.

Sketch style. Storyboards are typically rendered by hand using pencil or charcoal. These are often rendered quickly without significant detail, texture, or shading. Often the dominant subject is rendered in the most detail, with static background objects rendered more loosely.

5.4 Computer-generated schematic storyboards

Generating storyboards automatically is a broad challenge, entailing difficult problems in computer vision, non-photorealistic rendering, visualization, and even activity and emotion recognition. As a step towards solving this problem, I first propose the following conceptual pipeline:

- Tracking: Determine the motion of the camera and subject.
- Segmentation: Classify pixels as belonging to a rigid background or moving subject.
- Keyframe selection: Determine frames to appear in the composition.
- Extended frame layout: Arrange the frames into one or more extended frame segments.
- Annotation layout: Determine the placement of arrows and zoom lines.
- Compositing: Combine the elements in a plausible depth ordering.
- Sketch rendering: Optionally output the results in a non-photorealistic style.

In this chapter I largely limit my scope to the latter four items of this pipeline, that of arranging and rendering storyboard elements in a clear and appealing composition that effectively communicates camera and subject motion. I will also present a user interface for user selection of keyframes. I identify and address the following challenges:

- *Frame alignment.* In a storyboard extended frame, we must create a continuous composition from multiple frames in which the center of projection varies. This is related to panoramic mosaics and multiperspective imaging, with the additional constraint that the final composition may not rotate the images with respect to the page or screen. However, since my goal is expressiveness, not realism, my system may permit some distortion of objects in the scene.

- *3D annotations with unknown camera.* My system must introduce 3D annotations with incomplete 3D information. The challenge here is determining the salient elements of motion; note that even when the camera is calibrated, it is not straightforward to determine the 3D trajectory of all the moving, deforming objects in the scene.
- *Maximize visual clarity.* I seek to lay out elements and render them in styles that maximize the visibility and clarity of the presentation.

Finally, I point out that while a human storyboard artist can use arbitrary visual elements to improve a composition, our system is constrained to represent images and motions that actually appeared in the input video.

In Section 5.5, I discuss in detail the algorithms employed to meet these goals.

5.5 System overview

My system is broken into the following stages, which are performed in sequence: extended frame layout, keyframe selection, compositing, sketch rendering, and annotation layout. The algorithms for each stage are described in detail in subsections that follow.

The system considers all objects in the scene to belong to either the background (rigid, possibly viewed from a moving camera), or a single subject. The case of multiple subjects is a natural extension of our framework.

In my prototype system, tracking is performed offline as described in Chapter 3. The user can navigate forward and backward in the video using either a standard timeline slider or the "virtual slider" interface described in the next chapter (Section 7.1).

5.5.1 Extended frame layout

When the first keyframe is selected, the interface enters "storyboard mode," in which the video frame is shown translating and scaling so that the background pixels are roughly aligned between frames. To accomplish this, the system first solves for the transformation of each frames into a single coordinate system. That is, it finds transformations $\mathbf{M}_t$ for each frame t, such that the background features are nearly aligned, under the constraint that $\mathbf{M}_t$ is comprised of a uniform scale and translation $s_t\mathbf{x} + \mathbf{t}_t$. Rotations are intentionally omitted from this transformation. Figure 5.4 shows

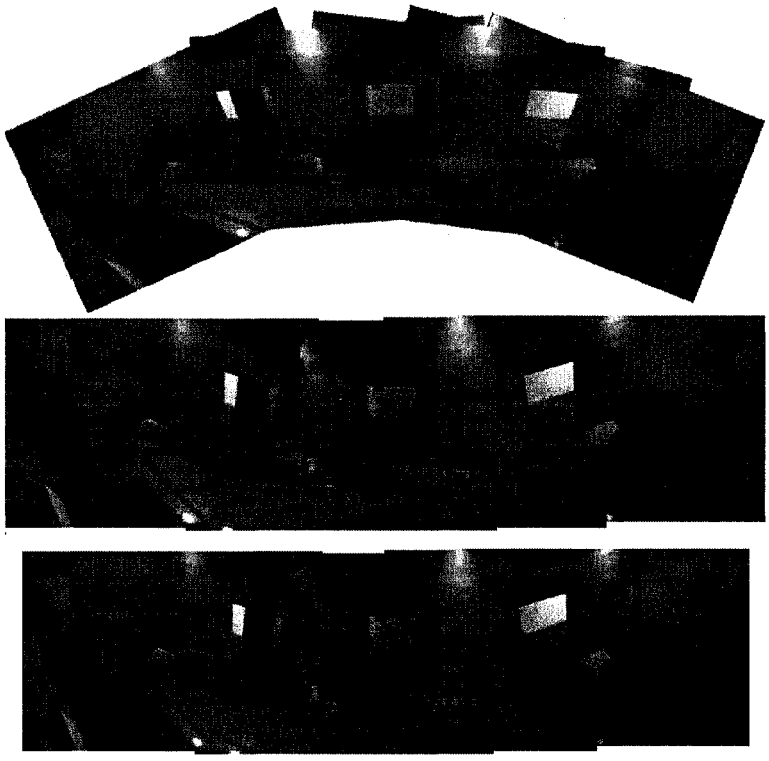


Figure 5.4: Top: extended frame layout with rotations. Middle: extended frame layout with no rotations. The rotationless composition has poorer alignment but gives a more accurate impression of the camera motion. Bottom: Many alignment artifacts can be attenuated using gradient-domain blending.

the difference between a composition using rotations and a composition without rotations. Note that although the composition with rotations has better geometric alignment between features, it gives an impression that the camera is rolling with respect to the horizon, even though the camera is held at a constant orientation with respect to the horizon. The composition without rotations has poorer alignment, but it gives a more correct impression of the camera motion.

Consider a single pair of frames s and t, with n background features. Let us denote the location of feature i in these frames as $\mathbf{x}_{si}$ and $\mathbf{x}_{ti}$. The obvious approach is to solve for the least-squares transformation between the sets of points $\mathbf{x}_s = \{\mathbf{x}_{si} \forall i\}$ and $\mathbf{x}_t = \{\mathbf{x}_{ti} \forall i\}$ using only uniform scale and translation, but this can produce degenerate solutions. Consider for example the case in which two points undergo a ninety-degree rotation with respect to the camera, appearing on a horizontal

line in one frame and a vertical line in the next; the best scale between these frames in the least-squares sense is 0, even if the distance between the points is unchanged.

An alternate approach is to find correspondences with rotations, then "undo" the rotations. My approach is a modification of a method due to Horn [36] and refined by Umeyama [90], in which I have substituted the optimal rotation R with the identity matrix. Indeed, when there is no rotation between the two sets of feature points, this transformation is the optimal least-squares uniform scale and translation between the points.

Specifically, the system computes the centroids and standard deviations of the features in each frame in the standard way:

$$\bar{\mathbf{x}}_t = \sum_i \mathbf{x}_{ti}/n \tag{5.1}$$

$$\sigma_t = \sum_i \|\mathbf{x}_{ti} - \bar{\mathbf{x}}_t\|^2/n \tag{5.2}$$

Then the relative scale between frames s and t is computed as the ratio of the standard deviations of the feature positions, and the relative translation is computed as the difference of scaled centroids:

$$S_{s\to t} = \sigma_t/\sigma_s \tag{5.3}$$

$$\mathbf{t}_{s\to t} = \bar{\mathbf{x}}_t - S_{s\to t}\bar{\mathbf{x}}_s \tag{5.4}$$

I denote this transformation $\mathbf{M}_{s\to t}$. (Note that all features are not visible in all frames, so for each pair of frames the system recomputes $\bar{\mathbf{x}}$ and σ using the subset of feature points visible in both frames.)

After computing the transformations between temporally adjacent pairs of frames, the system assigns each frame a transformation $\mathbf{M}_t$ in a global extended frame coordinate system. The first frame is arbitrarily assigned to lie at the origin with a scale of 1, and each successive frame is transformed by appending the transformation from the previous frame:

$$\mathbf{M}_0 = \mathbf{I} \tag{5.5}$$

$$\mathbf{M}_t = \mathbf{M}_{(t-1)\to t} \circ \mathbf{M}_{t-1} \tag{5.6}$$

This placement is not globally optimal, since small errors between pairs of frames may accumulate over the length of a sequence. But I have found these errors to be acceptable as long as tempo-

rally distant frames do not overlap — that is, as long as the camera does not pass over the same background multiple times.

As noted in Section 5.3, a single extended frame composition is not always the optimal representation for a given shot. I therefore split a shot into multiple *segments* as needed to handle both large changes in scale, and self-intersecting trajectories of the subject. Each segment is represented as a single extended frame.

My system provides several tests to determine how to break a shot into segments. One is a *scale* test that determines if the scale ratio between the largest and smallest frame in a segment is greater than a threshold T_s. The second is an *overlap* test that determines if the subject's path will cross over itself. The overlap test sweeps out the path of the subject in the extended frame coordinate system, failing when the percentage of pixels overlapping between successive intervals of the path is greater than a threshold T_o. For all results shown in this chapter, I used the constant thresholds $T_s = 1.5$ and $T_o = .25$.

My system employs a greedy algorithm, accumulating successive frames into a segment until one of the above tests fails, at which point the current frame becomes the first frame of a new segment. In addition, a failure of the overlap test often indicates an extremum of subject motion. Therefore, the system duplicates the last frame of the previous segment as the first frame of the new segment, so that both segments visually represent a time interval including the extremum.

When all the segments have been computed, the segments are rescaled so that the first frame t of each segment has scale $S_t = 1$.

In a typical storyboard, separate extended frame segments are laid out left to right and top to bottom. My segment layout approach adopts the spirit of extended frame layout: we offset each segment either to the right of or below the previous segment, choosing the offset direction that is closer to the direction of camera movement. This avoids segment layouts in which the segment layout and extended frame layout proceed in opposite directions.

5.5.2 Additional keyframe selection

In "storyboard mode," the current frame is overlaid atop the storyboard as a ghosted image, in the proper extended frame coordinate system. Changing frames causes this ghosted image to move

about the window, panning as the camera moves. The user can add new keyframes at any time, and the layout is automatically recomputed. Some steps in this iterative process are shown in Figure 5.5.

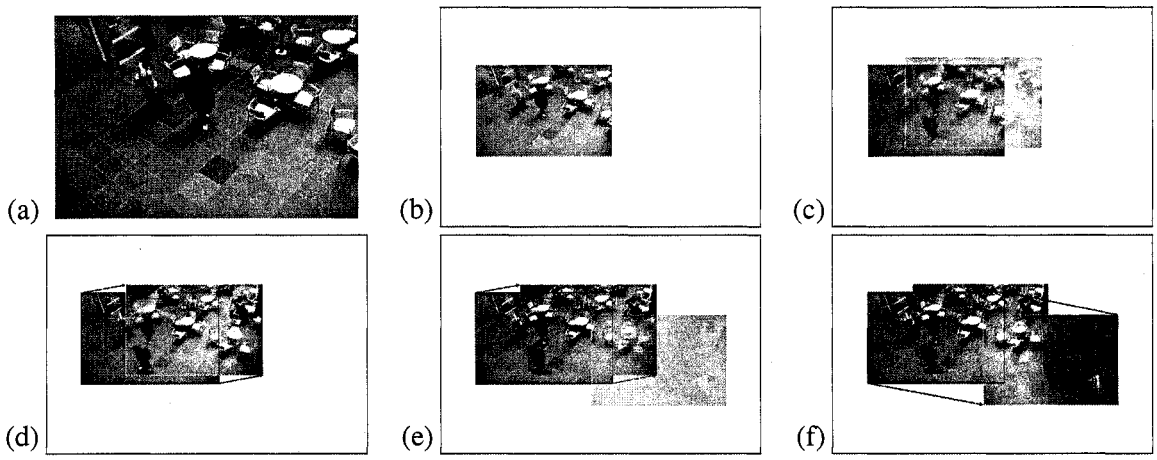


Figure 5.5: Several steps in the interactive authoring of a storyboard. The user scrubs to a frame of interest (a) and marks it as a keyframe using a menu item (not shown). The interface enters a "storyboard mode (b)," showing the keyframes in a global coordinate system in which the background is roughly aligned. When the user scrubs to another frame (c), the current video frame is shown ghosted above the storyboard. A second keyframe is specified (d), and frame arrows appear to show the motion of the camera. The user scrubs to a new frame (e) and marks it as the third keyframe (f).

In the event that the tracking and grouping computation fails due to motion blur, compression artifacts, or other challenging aspects of the input video, feature tracking can also be performed manually, using the following approach. In this case, keyframe selection must be completed before extended frame layout: The user first chooses the frames to be included in the composition using a standard linear timeline. Then she identifies corresponding points that appear in multiple frames. To assist in the layout and segmentation phases, she also labels each such feature as belonging to a subject (moving) or background (static) object. Extended frame layout is then computed only for the keyframes. Relatively few points are needed (about a dozen per keyframe). The results in this chapter were produced using manual tracking due to low-quality input sources, but for each of the results presented in this chapter, the time required for user input was no more than 5 to 10 minutes. For results produced using the automated tracking approach, please see Appendix A.

5.5.3 *Segmentation and compositing*

My system composites the extended layout before adding other 3D annotations such as arrows.

When assembling the storyboard composite, the system must balance competing goals. Where regions of background overlap, the system should create a seamless composite, as in a panoramic mosaic. Where regions of subject and background overlap, the system should ensure that the subject appears in the final composite. Finally, where the subject overlaps in two such frames, the multiple instances of the same subject should appear in the proper depth priority, such that instances in which the subject is closer occlude more distant ones.

I treat compositing as a labelling problem, in which the label assigned to a pixel determines the frame from which the color of that pixel is drawn. My system first computes mattes for the subjects using the GrabCut matting algorithm [69], initialized using the bounding boxes of the subject feature points. These mattes are then transformed into the extended frame coordinate system, where they are used as hard constraints for a second graph cut optimization, using the Photomontage approach of Agarwala *et al.* [3] to determine the locations of seams between frames. Where multiple mattes overlap, I define a depth ordering of the frames using the relative 2D size of the subject to indicate its distance to the camera. Note that this approximation for depth ordering is not generally a valid one: for example, a car viewed from the side appears smaller than when viewed at the same distance from the front. However, I have found that it works well in practice. Finally, as in the Photomontage system, I apply gradient-domain blending as needed, in order to eliminate visible seams between separate frames.

5.5.4 *Sketch rendering*

For some applications, this literal composite of the input frames is the ideal representation of the video content. However, I also provide a non-photorealistic (NPR) filter that can be applied to the composite before annotations are added, giving the image an appearance similar to a hand-drawn sketch (see Figure 5.6). This NPR filter "inks" high-contrast edges using a thresholded gradient-magnitude image, and shades the background using a combination of color transformations and image masks. This representation improves the visibility of arrows and other notations under a wide variety of image contents. Since it abstracts the contents of the images, it may also be used in

applications for which the animatic video content may be a proxy for the final footage. For example, in animation preproduction, two types of rough cuts provide an early guide to the dynamics of a scene: a "rip-o-matic" is a cut composed of footage from other films, and an "animatic" is a cut composed using very rough models and animation.

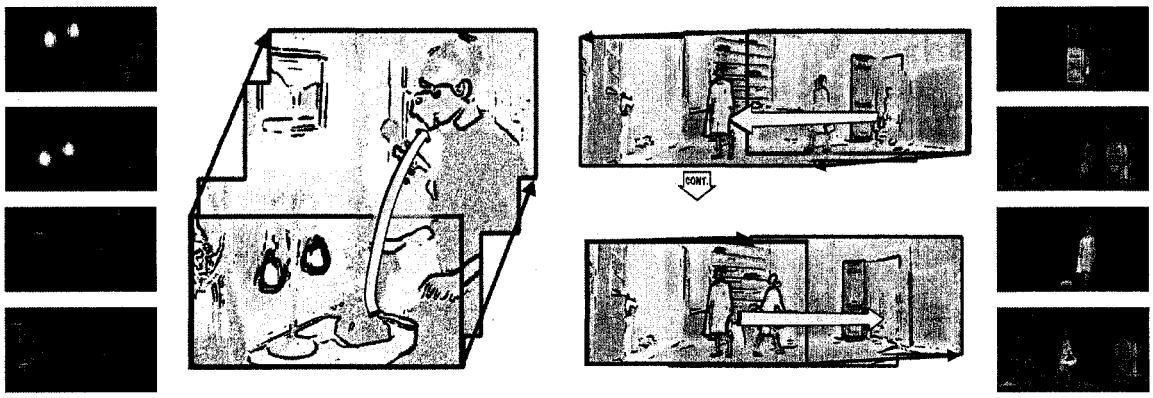


Figure 5.6: Four frames from the "telephone" and "walkleftright" sequences, and schematic story-boards composed using these frames.

5.5.5 Annotation layout: camera motion

Camera motion within a segment is annotated using an outline of the first frame boundary and arrows leading from the corners of this outline to the corners of the boundary of the last frame of the segment. For translational motions, only two outermost arrows are drawn, so that the internal arrows do not cross over the center of the frame.

5.5.6 Annotation layout: subject motion

My system annotates subject motion using 3D curved arrows, composited atop the extended frame. Such arrows have many degrees of freedom. Even a straight arrow with no curvature or varying twist has ten geometric degrees of freedom: three each for starting and ending position, two for breadth and thickness, and one for twist angle. In addition, each arrow could be rendered using a different amount of foreshortening, determined by the camera's focal length. (See Figure 5.7.)

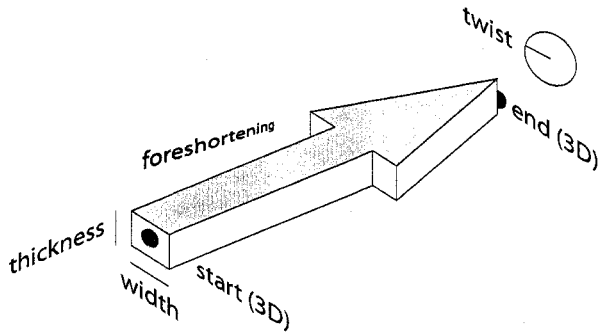


Figure 5.7: The ten degrees of freedom of a straight 3D arrow.

To simplify, in my current system I choose a single 3D coordinate system and camera projection for all motion arrows, with the eye point **E** at the origin, the negative z-axis pointing into the page, and a user-selected focal length.

The system creates a separate arrow for each frame range in a segment in which the subject is continuously visible. The arrows are placed using 3D non-uniform B-splines, with control vertices constrained by the locations of the 2D subject features. However, the system does not generally have enough information about the scene to compute true three-dimensional locations of the features. Furthermore, the keyframes do not share a common 3D coordinate system. Therefore I employ a pseudo-3D estimation using the 2D distributions of the subject features in the extended frame coordinate system.

First, the 2D centroids $\bar{\mathbf{x}}$ and standard deviations σ of the subject features in each frame are computed in the global extended frame coordinate system. I assume that the dominant changes in size of the subject are due to motion towards or away from the camera. I use the standard deviation of the subject feature points as an estimate of the size of the subject in each frame. If we know the distance to the subject in one frame, the system can estimate its distance in any other frame using similar triangles: $d_t/d_s = \sigma_s/\sigma_t$. Therefore, one needs only to provide the distance to the subject in a single frame. I assume the subject is at its closest in the frame with the largest standard

deviation, $t_{min} = \text{argmax}_t \sigma_t$. The distance $d_{t_{min}}$ of the subject at this closest frame is specified as a system constant. (This constant affects only the scale of the reconstructed scene along the z-axis, and therefore does not affect the final rendering, modulo numerical precision. I used $d_{t_{min}} = 500$ pixels for all our results.) The subject centers are approximated as the point along the ray from the camera through the feature centroid $\bar{\mathbf{x}}_t$ at distance d_t.

These estimated 3D points form the control vertices of a cubic non-uniform B-spline, using multiple end knots in order to interpolate the starting and ending position. For splines with fewer than 4 control vertices, the spline degree is reduced accordingly.

The resulting 3D spline forms the backbone of the 3D arrow. In order to align the arrow geometry to this backbone, I also construct a coordinate frame at each point along the spline. As described in Section 5.3, a twisting arrow signifies a subject rotating around its velocity vector, and since this is an uncommon case, I presently generate arrows that do not twist along their length. My system determines these coordinate frames as follows: Given the 3D point $\mathbf{P}_H$ and tangent $\mathbf{T}_H$ at the head of the arrow, I compute a perpendicular vector $\mathbf{U}_H = \mathbf{T}_H \times (\mathbf{E} - \mathbf{P}_H)$, where $\mathbf{E}$ is the center of projection. This vector is used to construct the normal vector everywhere along the arrow: $\mathbf{N}_t = \mathbf{U}_H \times (\mathbf{E} - \mathbf{P}_t)$. (Degeneracies of these cross products are rare, but can be handled as special cases.)

Finally, the arrow endpoints are offset from the subject positions along the spline by a small amount to improve visibility of the subjects.

The width w and thickness h in pixels of the arrows (i.e., the dimension along which the arrowhead flares, and the perpendicular dimension) are set to be linearly proportional to the standard deviation of the features at the closest frame:

$$w = \alpha \sigma_{t_{min}} \tag{5.7}$$

$$h = \beta \sigma_{t_{min}} \tag{5.8}$$

For all results shown in this chapter, I used $\alpha = 0.5$ and $\beta = 0.125$. Figures 5.8 and 5.11 show arrows constructed by our system for an approaching figure.

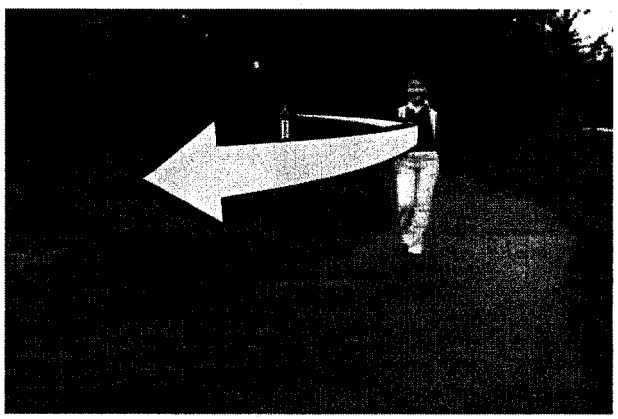
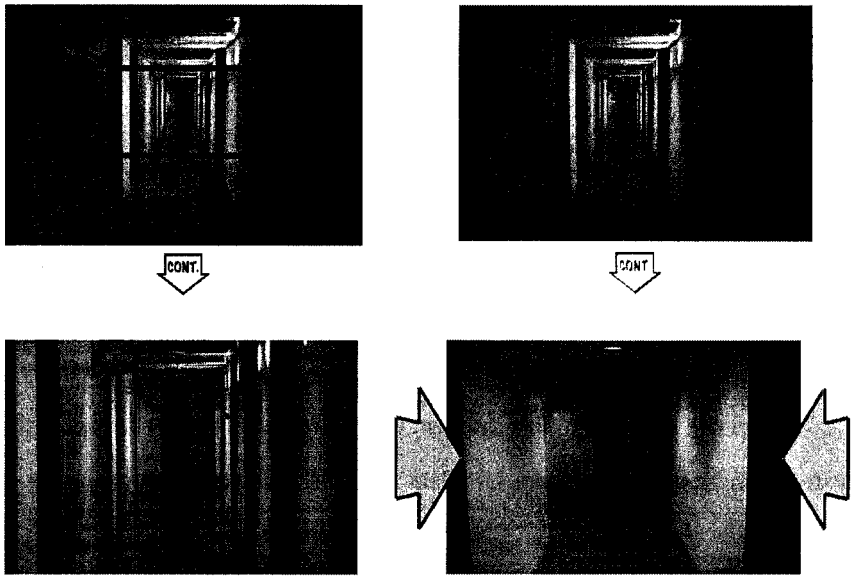


Figure 5.8: A motion arrow for an approaching figure.



(a) (b)

Figure 5.9: (a) Storyboard notation for a zoom in (increasing focal length). (b) Storyboard notation for a dolly in (camera traveling forward).

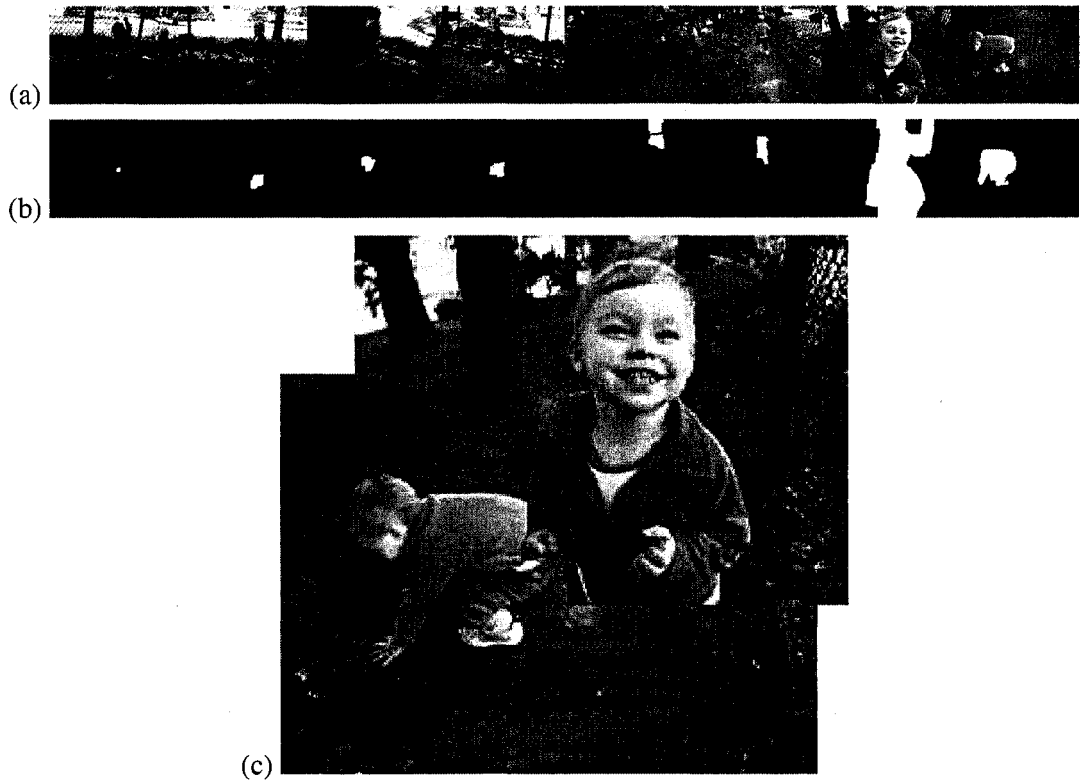


Figure 5.10: Source frames (a) and GrabCut mattes (b) for the "Running Boy" shot. The mattes are not very accurate, but the system can still use them to produce the storyboard shown in Figure 5.12. Note that the mattes only need to be precise where the subjects overlap, as in this portion of the composite (c).

5.5.7 Inter-segment motion annotations

A shot may be split into multiple extended frame segments, either because of widely varying scales across the shot, or because of overlapping motion of the subject. For both types of segment transitions, my system renders small arrows labelled "CONT." to indicate continuity between these segments.

Scale changes between segments can occur either because of motion into or out of the image plane (known to cinematographers and camera operators as a *dolly*), or because of a change in focal length (known as a *zoom*). At present my system does not detect the difference between these camera operations, but allows the user to choose the appropriate annotations for each scale change.

Scale changes due to zoom are annotated using zoom lines. Consider two adjacent frames t and

$t+1$ that have been placed in successive segments A and B. To represent a zoom-in between A and B, the system draws the outline of frames t and $t+1$ in the coordinate frame of t. The transformation between the two outlines is simply $\mathbf{M}_{t\rightarrow(t+1)}$. For a zoom-out, the procedure is similar, but this time the system draws the outlines atop frame $t+1$, using the transformation $\mathbf{M}_{(t+1)\rightarrow t} = \mathbf{M}^{-1}_{t\rightarrow(t+1)}$. Finally, the corresponding corners of the frame outlines are connected using 2D arrows (see Figure 5.9(a)).

My system denotes scale changes due to dollying using 3D arrows at both left and right sides of a frame. For camera motion into the plane of the image (forward motion), receding arrows are drawn on frame $t+1$ in segment B. For camera motion out of the plane of the image (backward motion), approaching arrows are drawn on frame t in segment A. The placement of these arrows is such that either the tail of the arrow touches the earlier frame or the head of the arrow touches the later frame, always pointing "forwards" in time across the transition (see Figure 5.9(b)).

5.6 Results

In previous sections I have shown a few preliminary results from our system. In this section I provide several more illustrating the range of our system. A few additional results are shown in the supplementary video [29].

Many complex shots can be depicted very gracefully and completely using only single-panel storyboards. Figure 5.11 shows a storyboard for a shot in which the camera starts framing the skylight, then tilts down to reveal a woman walking up a stairway toward the camera. It takes quite a bit of mental effort to parse the written description in the previous sentence, and any single still frame from this shot would be incomplete, but the storyboard is clear and unambiguous.

My approach method can even handle long shots filmed with a hand-held camera. The "Running Boy" sequence is a 20-second home video taken using a hand-held digital camera at 320×240 resolution, 15 frames per second, heavily compressed with MPEG-4. Figure 5.10(a) shows the 8 user-selected key frames, and Figure 5.10(b) shows the mattes produced automatically using GrabCut. About a dozen feature points were manually selected in each of the key frames and labeled as subject or background. Although the mattes automatically produced by the GrabCut algorithm are low quality, often cutting out large sections of the subject's body, they are sufficient to seed the

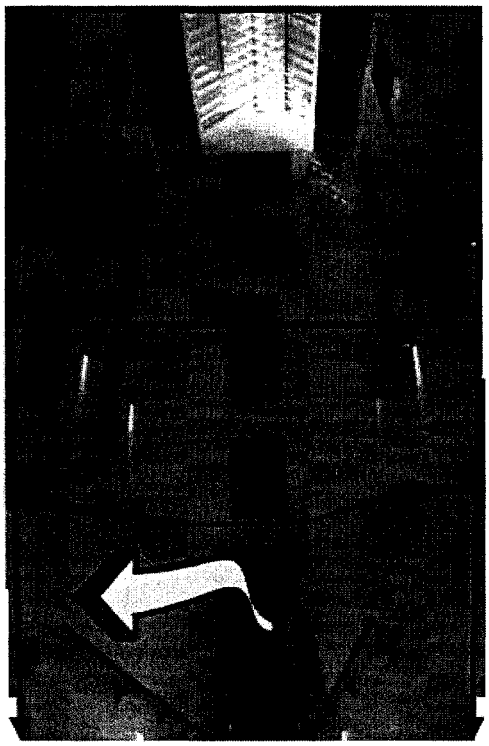


Figure 5.11: A storyboard for a shot in which the camera tilts down to a stairway, revealing an actress walking toward the camera.

graph-cut composite. The final schematic storyboard, with 3D subject and camera arrows overlaid atop the composite, is shown in Figure 5.12.

Figure 5.6 shows two non-photorealistic examples. In the "walkleftright" example, the system has broken this scene into two extended frame segments (connected with a "CONT." arrow) since the subject crosses over her own path in the latter half of the shot.

Additional examples are shown in Figure 5.13. These shots were all extracted from the film, *Charade*, digitized at 320×240 resolution and 30 frames per second. Users spent 5–10 minutes on each storyboard, whereas the sketches on the right of Figure 5.13 took a professional storyboard artist 10–30 minutes each to complete, using Corel Painter and Photoshop.

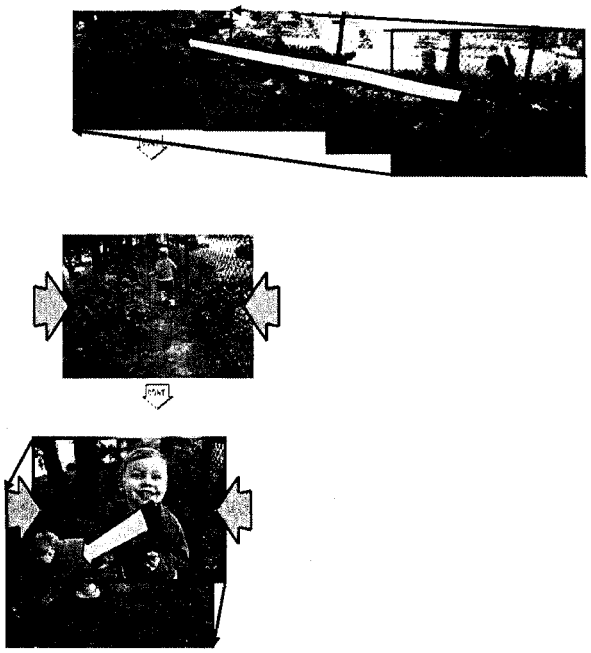


Figure 5.12: The complete schematic storyboard produced from the frames in Figure 5.10.

5.7 Discussion

I have presented a system for transforming video clips into static visualizations using the visual language of storyboards. My contributions include: the introduction of the iconography of storyboards as a visualization tool for video, and the representation of visual motion using 3D arrows without requiring true 3D reconstruction.

I believe that schematic storyboards can provide effective visualizations for presentation in both inherently static print media and dynamic digital media. In contrast with animated or filmstrip displays, a storyboard depiction allows rapid visual comprehension for a variety of tasks involving selection, comparison, and manipulation of motion data.

Schematic storyboards offer many other avenues for further research. My principal aim is to improve the quality of our results and bring them closer to the appearance of hand-drawn storyboards. Although I believe my image layout approach is satisfactory for visualization purposes, additional manual or automated methods could be applied to eliminate remaining artifacts. For example, the segmentation process can be improved by using layered representations such as that proposed by

Kumar *et al.* [46]. I can also hide changes of perspective or parallax by encouraging seams along architectural [101] or subject/background boundaries, or applying small amounts of distortion to the images in order to improve their alignment [38].

My system provides a non-photorealistic filter, but I believe these results can be improved by taking advantage of motion cues to render differently moving regions in different styles. In any event, the image quality of my system's storyboards is presently limited by that of the input video, but an alternate avenue for improvement is to utilize other image sources — such as still photos or hand sketches — to replace the degraded video frames.

My prototype implementation could be improved by adding support for multiple subjects, depth estimation that is more robust to changing orientation and occlusions, and improved schematic layout. We believe most of these extensions are relatively straightforward, and in the following chapter we demonstrate one possible approach to arrow simplification and layout.

In principle, schematic storyboards could also be extended as adaptive entities that change their appearance depending on context or size. For example, when surveying a large number of storyboards for editing purposes, the storyboards might be better visualized by using a more abstract rendering emphasizing one or two frames with simplified color schemes and 2D motion arrows.

Since it is based predominantly on existing storyboard iconography, my system shares some of the fundamental limitations of storyboards themselves. For example, storyboards do not use a standard indication for the duration of a shot or the speed of a motion, other than generic text labels such as "fast" or "slow" or the duration in seconds or frames. Furthermore, long and complex shots may be split into many segments, with confusing annotations and a spatial layout that may waste screen space.

In spite of these issues, I believe that a wide range of tasks involving video can benefit from the concise summary of motion afforded by schematic storyboards. In a few instances, automated storyboard generation could replace illustrations presently drawn by hand. For instance, textbooks on film studies already use production storyboards to illustrate film techniques [41], and where the original storyboards are unavailable, they may be reproduced by hand [31]. Similarly, instruction manuals typically contain illustrations of a physical process that must clearly represent the assembly of an object or the operation of a device. Storyboards can be especially useful for tasks in which multiple videos must be viewed, compared, and selected, such as logging and indexing of stock

footage galleries.

Although the level of expertise required to create a storyboard using my system is much lower than that of a professional storyboard artist, some user effort is still required. Therefore, our system is less well-suited to situations in which the video or storyboard is only viewed once or twice, such as the initial "assembly" stage of editing documentary footage, in which large volumes of data must be screened and culled to select shots that will be used in the first cut [73], or surveillance, in which the majority of acquired video has little or no value. However, fully automating our system could enable its use for these applications and also at the consumer level, for applications ranging from Internet video search to home video editing.

For some real-world videos containing motion blur, occlusions, and severe compression artifacts, our tracking and grouping algorithms fail, and in this case we provide an interface allowing the user to perform manual object tracking and labeling. Since only a few foreground and background features are required, this manual process usually takes a small amount of time, but it is nonetheless a tedious task. We believe that future progress in computer vision will enable robust, fully automated approaches for creating schematic storyboards from video.

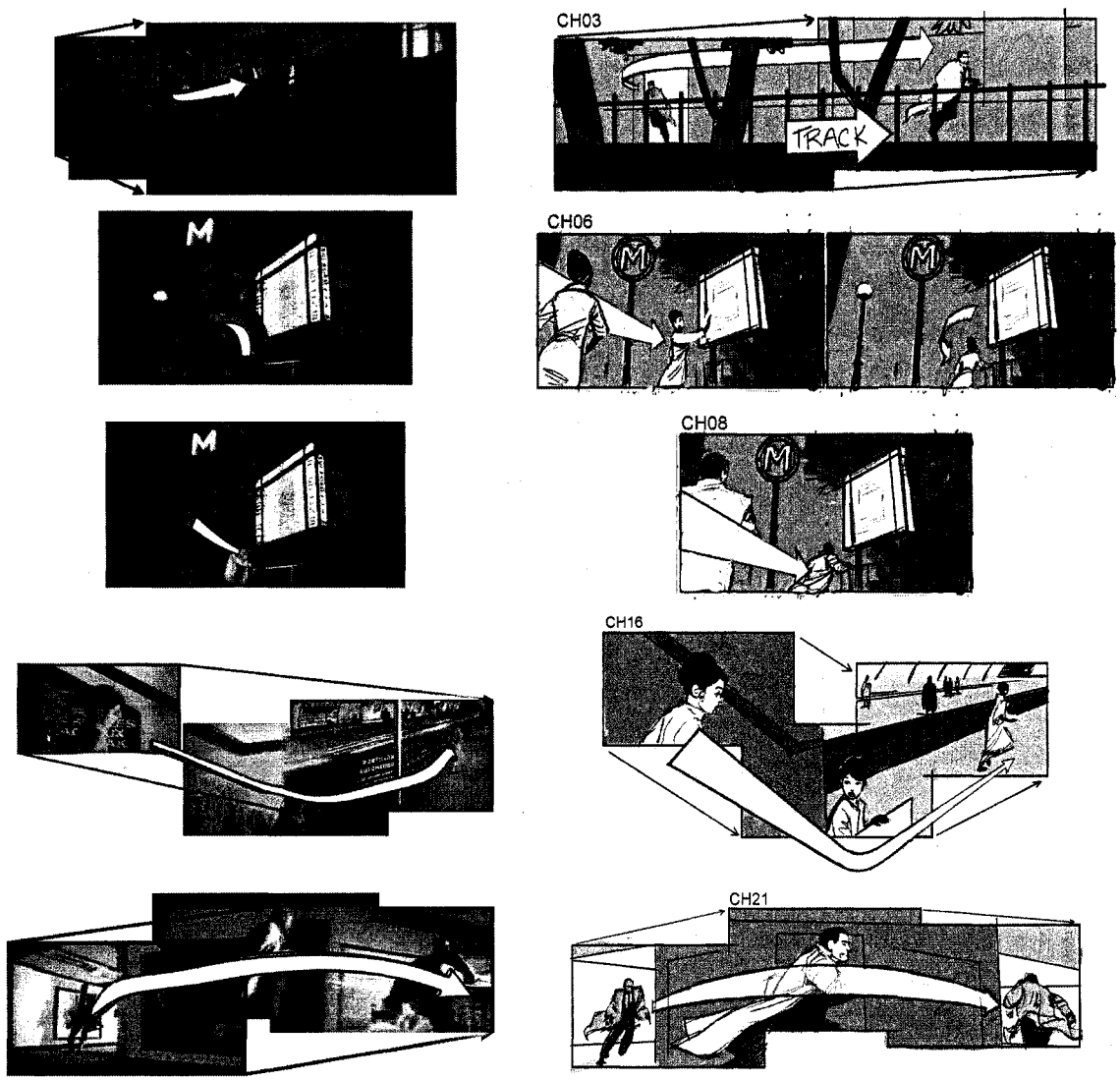


Figure 5.13: Schematic storyboards for a chase sequence from *Charade*. Left: results produced by my system. Right: results produced by hand. (Credit: Peter Rubin)

Chapter 6

OPTIMIZED STORYBOARD LAYOUT

6.1 Introduction

In this chapter I apply optimization as a tool for improving the layout of annotated storyboards.

In the previous chapter, I described a system to assist the user in creating a schematic storyboard that summarizes an entire continuous video clip (a "shot") in a still image, using a visual representation that can communicate high-level aspects of motion of camera and subject, without requiring a moving display. As a still image, it can be printed out on a piece of paper or a booklet for reference on a sound stage or on location. It can also be used as an interface for sliding a viewer along the temporal axis of the original footage in an intuitive fashion, as I'll discuss in Chapter 6.

The storyboards generated by this system contain the following graphical elements, illustrated in Figure 6.1:

- One or more *extended frame* images, composed by image processing and combining multiple frames of the input video.
- *Frame outlines* denoting the framing of the camera motions.
- Solid, black, 2-dimensional *camera arrows* associated with the frame outlines, denoting the motion of the camera across the extended frame.
- White, outlined, 3-dimensional *subject arrows* within the frame lines, denoting the motion of objects in the scene.
- White, outlined, 2-dimensional *continuity arrows* denoting temporal continuity between multiple extended frame segments.

In the previous chapter, the layout algorithms did not consider visual conflicts between multiple graphical elements, and therefore the resulting layouts often include arrows that overlap and obscure important parts of the extended frame, or in some instances arrows that overlap and occlude each other.

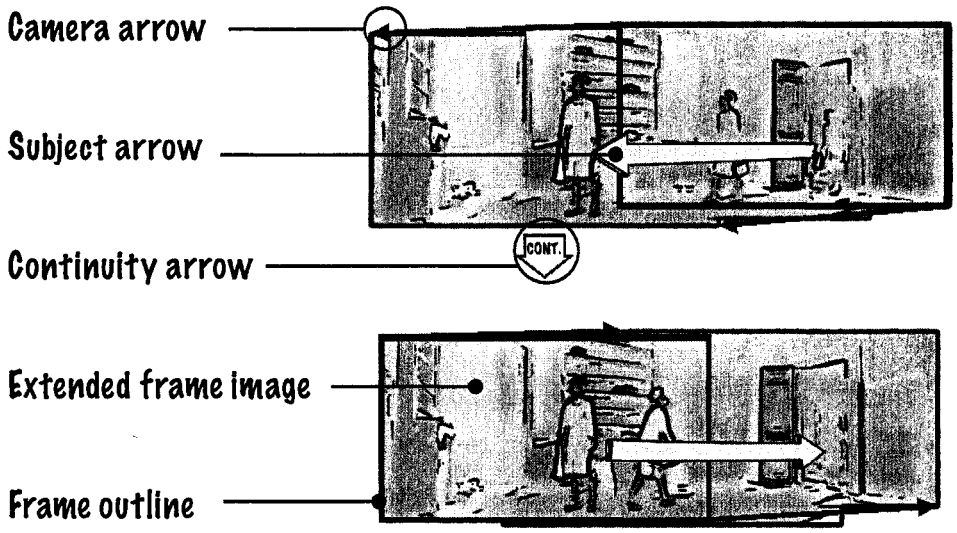


Figure 6.1: Elements contained in a typical storyboard.

In this chapter, I attempt to improve such layouts by introducing several degrees of freedom to certain elements, and try to solve for the optimal layout given those degrees of freedom.

Of the elements listed previously, extended frames and frame outlines are considered fixed, static elements. The extended frames are initially computed using a series of least-squares optimizations for positioning and a graph cut optimization for pixel selection. These optimizations could in principle be subsumed by the global layout optimization described in this chapter. However, as the quality of these layouts is already quite good in isolation, I don't believe the potential improvements are worth the additional computational complexity.

In any annotation layout some pixels will be occluded by the annotational arrows, but the visibility of all pixels is not equally important. Therefore, an *importance image* is associated with each extended frame, assigning each pixel a level of importance for that pixel's visibility. The importance image is computed using a matting algorithm, so that only occlusion of foreground moving objects in the extended frame is penalized. The system also convolves the importance image with a Gaussian kernel to smooth the optimization landscape and allow small overlaps along edges of important

regions.

The rest of the elements are considered free *dynamic* elements that can be freely repositioned to improve visibility and readability of the overall layout by avoiding overlaps between elements. Each dynamic element has one or more *target* locations suggested by the existing storyboards system. In the absence of other constraints each element would be positioned as close as possible to its targets.

Each type of arrow has a slightly different set of degrees of freedom:

Continuity arrows are treated as having fixed shape, defined using bounding boxes and a variable offset from the target location.

Straight-line camera arrows are defined by a variable starting and ending location and a line width.

Curved object arrows are defined by n control vertices and a width at each vertex. The resulting curve is as an order 2 or 3 b-spline that passes through the first and last control vertex, and approximates the intervening vertices. In my experiments I restricted n to 3, effectively limiting the interpolation to a best-fit quadratic curve, but I have also implemented higher-order splines.

Camera and object arrows may also be split into multiple intervals. I denote these intervals parametrically using a starting and ending parameter along the curve, where $t = 0$ denotes the start of the original path, and $t = 1$ denotes the end of the path.

6.2 Layout by optimization

The structure of the cost function for storyboards is not self-evident. For simplicity, I take the approach of LineDrive [5] treating the cost function C as a sum of terms

$$C(L) = \sum_k w_k F_k(L) \tag{6.1}$$

in which each term k has a constant weight w_k and a "factor" F_k, which may be a nonlinear function of the layout L. In general the F functions will represent the fitness of L separated according to various design criteria.

I make the further simplification that each factor can be expressed as a sum over each element of the layout, or each pair of elements in the layout. Functions of a single element are called *data factors* (denoted $F_D(e)$), as they encode the fitness of that element to its intrinsic goal or target data. Functions of pairs of elements are called *interaction factors* (denoted $F_I(e, f)$), as they encode the

penalty for conflicting interactions between two elements. Formally:

$$F(L) = \sum_{e \in L} F_D(e) + \sum_{e,f \in L} F_I(e,f) \tag{6.2}$$

where e and f are distinct elements ($e \neq f$) of the layout L.

6.2.1 *Interaction penalties*

My implementation includes only one type of interaction penalty, corresponding to the notion that elements should not obscure each other. For the interaction of two arrow elements, the overlap penalty is the area of overlapping pixels:

$$F_I(e,f) = ||\text{Pixels}(e) \cap \text{Pixels}(f)|| \tag{6.3}$$

where $||X||$ counts the number of pixels in the set X.

For the interaction of an arrow elements with the extended frame image, the overlap penalty is the weighted integral of the overlapping area, using the importance map M as the weighting function:

$$F_I(e, \text{Extended Frame}) = \sum_{p \in \text{Pixels}(e)} M(p) \tag{6.4}$$

6.2.2 *Data penalties*

The simplest data penalty term is just the squared distance of the object from its target(s) $T(e)$:

$$F_D^{\text{dist}}(e) = \sum_{t \in T(e)} d(t,e)^2 \tag{6.5}$$

where $d(t,e)$ computes the distance from t to the closest point on e. Subject and camera arrows may have multiple targets, and for these arrows I define two separate distance penalty terms, one measuring the distance to the endpoints and a separate one measuring the closest distance along the arrow. However, in practice I have used the same weights for both of these terms.

Camera and subject arrows may also be split into several intervals (t_0, t_1), where values of t range from 0 to 1 and intervals cannot overlap. Each interval is represented visually using a separate arrow, and I define penalties for complexity and also for gaps along the target path. The first rewards parsimony, by penalizing the number of intervals:

$$F_D^{\text{parsimony}}(e) = ||I_e|| \tag{6.6}$$

The second rewards full coverage of the path, by penalizing longer gaps:

$$F_D^{\text{gap}}(e) = 1 - \sum_{[t_0,t_1]\in I(e)} t_1 - t_0 \tag{6.7}$$

Finally, I also found that very short arrows are highly undesirable, so a shortness penalty is defined:

$$F_D^{\text{short}}(e) = \sum_{[t_0,t_1]\in I(e)} (t_1 - t_0)^{-2} \tag{6.8}$$

6.2.3 Perturbations

The cost function in Equation 6.1 is optimized using simulated annealing. Simulated annealing has been previously applied to layout optimization by Davidson and Harel [22] and by Agrawala *et al.* [5]. (Lok and Feiner [54] have surveyed a variety of presentation layout approaches.) We apply the specific algorithm described by Agrawala *et al.*. At each iteration, a random element is perturbed. Each element has several degrees of freedom along which these perturbations may act, some continuous and some discrete, including position, number of intervals, and interval locations.

All element types can have their location perturbed. The amount of the perturbation is a random variable drawn from a 2D Gaussian distribution, with 0 mean and standard deviation proportional to the size of the element. For straight and curved arrows I originally experimented with allowing each separate control point of a straight or curved arrow to be perturbed independently, but I found that even a small deviation from the initial shape or direction of the arrow could be perceived as a significant visual distortion. After eliminating these additional degrees of freedom I achieved much more pleasing results, as well as faster convergence.

Camera and object arrows may also have multiple intervals, and my system therefore supports these additional perturbations:

- *Split*: Choose a random interval and split it into two separate intervals.
- *Merge*: Randomly choose two successive intervals and merge them into a single interval.
- *Insert*: Randomly choose two successive intervals and insert a new interval in the gap between them.
- *Delete*: Randomly delete an interval.
- *Adjust*: Randomly choose an endpoint of an interval and slide it to a new location between neighbouring endpoints with uniform probability.

Some of these perturbations have preconditions: For example, a merge or insert operation can only occur when an arrow has more than one interval. Therefore, at each step of the optimization I determine which preconditions are met and randomly choose from the set of allowable perturbations.

6.3 Preference elicitation with Arnauld

One of the difficulties of constructing the cost function in Equation 6.1 is determining the appropriate weights w_k to balance the tradeoffs between different design goals. I attempted to optimize the weights w_k using the Arnauld system for preference elicitation [25].

To facilitate the application of the Arnauld system to general utility maximization problems, we isolated the max-margin solver component into an XML-RPC server. This made it trivial to integrate Arnauld, written in Java, with the existing storyboards system, written in Python. The Arnauld interface consists of just six methods, with very little data transfer required:

- *getNewSession()*: Initiates a new dialogue, and returns an integer session ID.
- *terminateSession(ID)*: Terminates the given dialogue.
- *setPrior(ID, weights, equivalentSampleSize)*: Set a prior using the given weights and equivalent sample size.
- *addConstraint(ID, factorsA, factorsB, cmpAB)*: Add a constraint declaring *factorsA* better than, worse than, or equivalent to *factorsB*. The sign of integer *cmpAB* encodes the preference order.
- *containsEquivalentConstraint(ID, factorsA, factorsB)*: Returns true if an equivalent constraint has already been declared.
- *getWeights(ID)*: Compute the optimization and return the optimal set of weights.

Weights and factors are passed as name/value pairs, so that meaningful labels can be assigned to each weight or factor, and unneeded factors can be omitted.

Query selection, which is performed on the client side, proved to be the most significant challenge in applying preference elicitation to this problem. Gajos and Weld [25] propose two methods for query selection, and both require enumeration of the space of possible queries. However, many of the variables in this application are continuous, and so all possible queries cannot be enumerated. We can discretize the space of possible queries, but even so, the space of queries is still extremely

large for any reasonable discretization. Therefore we discretely sample the local neighborhood of the current best guess.

In our preliminary experiments, I applied outcome-based question generation to determine the next query. However, this method resulted in repeated presentation of nearly identical queries to the user. I believe this repetition is due to our application having many visually similar layouts with numerically distinct sets of factors. Although Arnauld provides a method to determine if a query will be redundant, it checks only for exact scaled copies of a factor vector pair, and therefore does not detect visually similar but non-identical queries. It should be noted that repeated queries are not just an inconvenience for the user: Repeated queries bias the optimization, so the optimized weights actually get worse as more queries are made.

I believe that the "weight-based question generation" suggested by Gajos and Weld would not suffer from this deficiency, but I have not yet pursued this query selection strategy. Also, the present system considers only a single target layout at a time. I believe that drawing queries from more than one sample layout at a time is necessary to fully explore the design space.

6.4 Results

Several "before and after" results are illustrated in Figures 6.2, 6.3, 6.4 and 6.5. All of these results were produced using the same manually-selected weights:

$$\begin{aligned} w_{\text{interaction}} &= 0.1 \\ w_{\text{dist}} &= 1.0 \\ w_{\text{parsimony}} &= 10.0 \\ w_{\text{gap}} &= 1.0 \\ w_{\text{short}} &= 100.0 \end{aligned}$$

Some features of the layouts have been improved: For example, in Figure 6.3, the subject arrow is split in two pieces so that objects in the extended frame image can be seen better, and the cam-

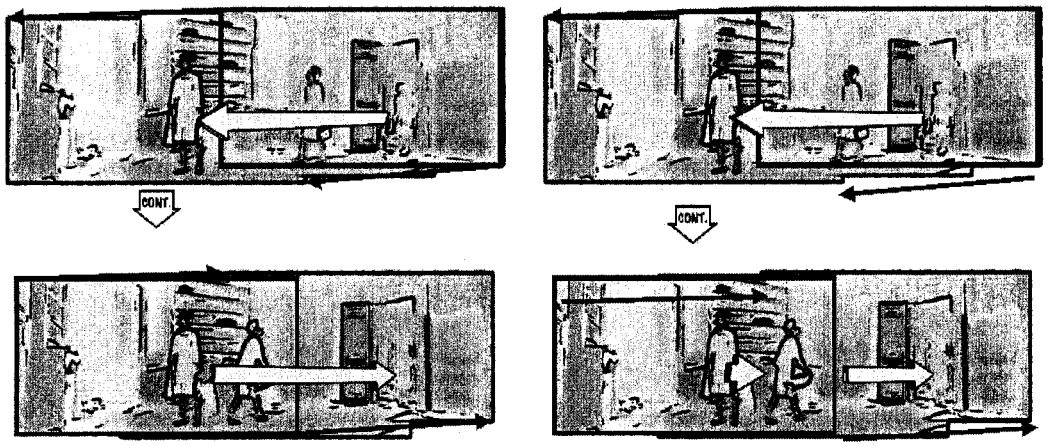


Figure 6.2: "Walk left/right" dataset. Left: before layout optimization. Right: after layout optimization

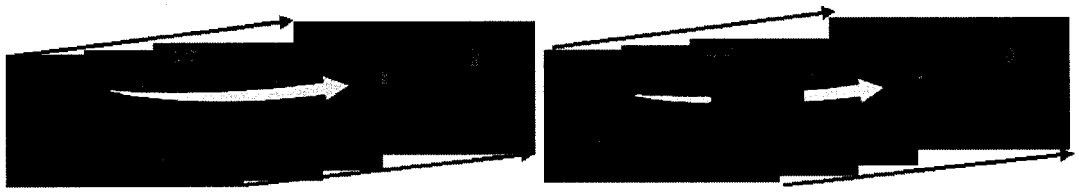


Figure 6.3: "Running woman" dataset. Left: before layout optimization. Right: after layout optimization.

era arrows have been offset from the frame outlines, improving their visibility. In Figure 6.6, the topmost continuity arrow has been displaced to avoid overlap with a nearby camera arrow.

However, some features of the layout are less desirable. For example, in spite of the very steep shortness penalty, many very short arrows persist in the resulting layouts (see bottom segment of Figure 6.6). Also, pairs of matching camera arrows are not generally moved symmetrically away from the frame lines, leading to less pleasing compositions. In Figure 6.2, one camera arrow has been moved into the frame itself, which might cause it to be mistaken for a subject arrow. In Figure 6.4, the subject arrow has been split without appreciably improving visibility of the extended frame.

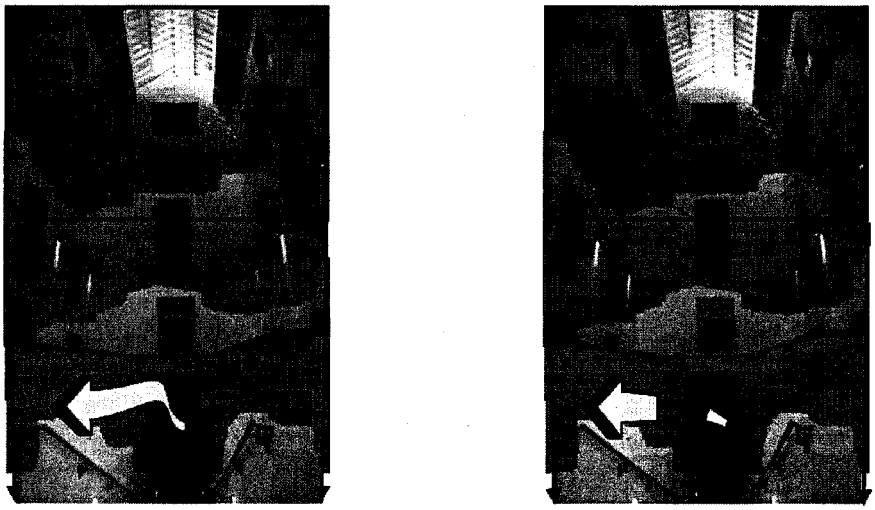


Figure 6.4: "Tilt down/approach" dataset. Left: before layout optimization. Right: after layout optimization.

Figure 6.5: "Car" dataset. Left: before layout optimization. Right: after layout optimization.

It should also be noted that the modified layouts are not really dramatically different from the originals, suggesting that simulated annealing may be overkill for optimizing the cost function: in hindsight it seems probable that a local sampling of layouts could likely find the same solutions with significantly less computation.

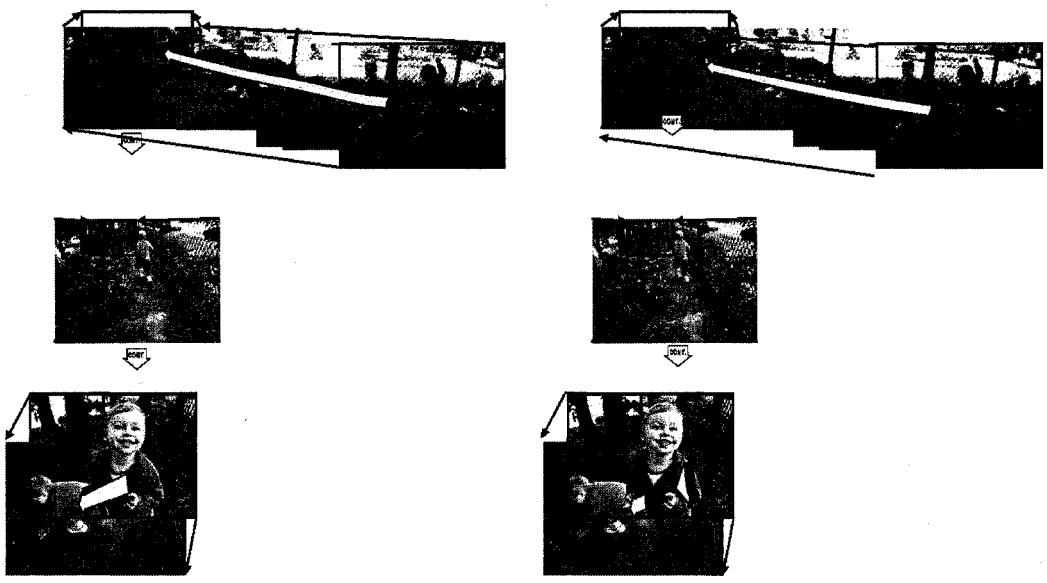


Figure 6.6: "Running boy" dataset. Left: before layout optimization. Right: after layout optimization.

Chapter 7

DIRECT MANIPULATION FOR TEMPORAL NAVIGATION

In the preceding chapters, I described an approach for analyzing the motion of points and objects in a video, an annotation system utilizing that data for interactive annotation, an application of that approach for constructing schematic storyboard visualizations, and a layout optimization approach for those storyboards. In this chapter, I build upon the video analysis and storyboard visualizations to present several fluid and intuitive direct-manipulation interfaces for *scrubbing*, or moving along the time axis of a video clip.

In existing video viewing tools, a user clicks and drags on a horizontal slider to scrub forward and backward in time. But the left-to-right motion of the slider is not necessarily related to the motion of the camera or subject, which may move vertically, along a complex curve, or in the opposite direction: right to left. I propose the use of a still video frame or a storyboard as a temporal interface, such that clicking and dragging on different elements of the frame or storyboard scrubs through time in a manner that creates the impression of directly manipulating the camera or subject. Schematic storyboards enable an alternative to the timeline or jog/shuttle dial as a graphical interface for "scrubbing" through the time axis of a video clip.

First, I demonstrate direct manipulation on video frames, utilizing raw point tracking data as virtual sliders. I call this first approach *virtual slider scrubbing*. Next, I show how virtual slider scrubbing can be extended to multi-point constraints for objects with complex motion or deformations. Finally, I describe a similar approach using the abstracted graphical motion description of a schematic storyboard. I refer to this as *storyboard scrubbing*.

Although long-range temporal navigation is a goal of the video summarization and abstraction literature reviewed in Chapter 2, there appears to be relatively little prior work in video navigation on shorter time scales. Irani and Anandan [37] observed that a static video summary could be used as an interface for selecting frames in a video, and Jojic *et al.* [39] have also demonstrated a user interface for video by decomposition into layers.

Direct manipulation control has previously been demonstrated for constructing new animated sequences from video in the Video Textures work of Schödl *et al.* [75, 74]. Our work adapts this idea to arbitrary video clips, without requiring controlled greenscreen capture.

7.1 Virtual slider scrubbing

In this interface, the user views a single frame of video, or a video playing back at some constant speed, and can scrub to a different frame of the video by directly clicking and dragging on any moving object. This UI is implemented as follows: When the user clicks at location $\mathbf{x}_0$ while on frame t_0, the closest track i' is computed as in equation (4.2), and the offset between the mouse position and the track location is retained for future use: $\mathbf{d} = \mathbf{x}_0 - \mathbf{x}_{i'}(t_0)$. Then, as the user drags the mouse to position $\mathbf{x}_1$, the video is scrubbed to the frame t' in which the offset mouse position $\mathbf{x}_1 + \mathbf{d}$ is closest to the track position on that frame:

$$t' = \text{argmin}_{\{t \in T(i')\}} \|\mathbf{x}_1 + \mathbf{d} - \mathbf{x}'_i(t)\| \tag{7.1}$$

Figures 7.1 and 7.2(a-c) illustrate this behavior.

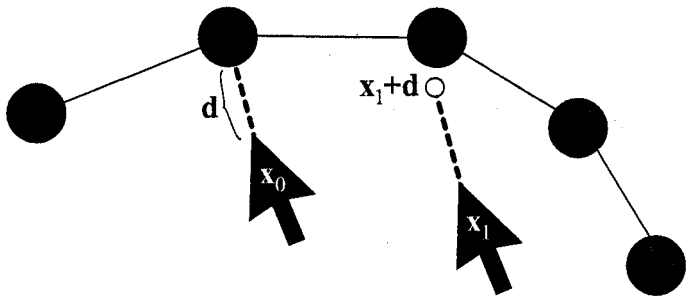


Figure 7.1: When the mouse is depressed at position $\mathbf{x}_0$ on frame 2, the feature track shown in red is selected as the closest track. When the mouse moves to position $\mathbf{x}_1$, the feature at frame 3 of that track is closer to the offset mouse position $\mathbf{x}_1 + \mathbf{d}$, so the video is scrubbed to frame 3.

7.2 Constraint-based video control

By extending the virtual slider scrubbing technique described above to multiple particle paths, I also implement a form of constraint-based video control. The user sets multiple point constraints on different parts of the video image, and the video is advanced or rewound to the frame that minimizes the sum of squared distances from the particles to their constraint positions. Here, c indexes over constraint locations $\mathbf{x}_c$, offsets $\mathbf{d}_c$, and constrained particles i'_c:

$$t' = \text{argmin}_{\{t \in T(i')\}} \sum_{c \in C} \|\mathbf{x}_c + \mathbf{d}_c - \mathbf{x}_{i'_c}(t)\|^2 \tag{7.2}$$

In my current mouse-based implementation, the user can set a number of fixed-location constraints and one dynamic constraint, controlled by the mouse. However, multiple dynamic constraints could be applied using a multi-touch input device. Figures 7.2(d) and 7.2(e) illustrate facial animation using multiple constraints.

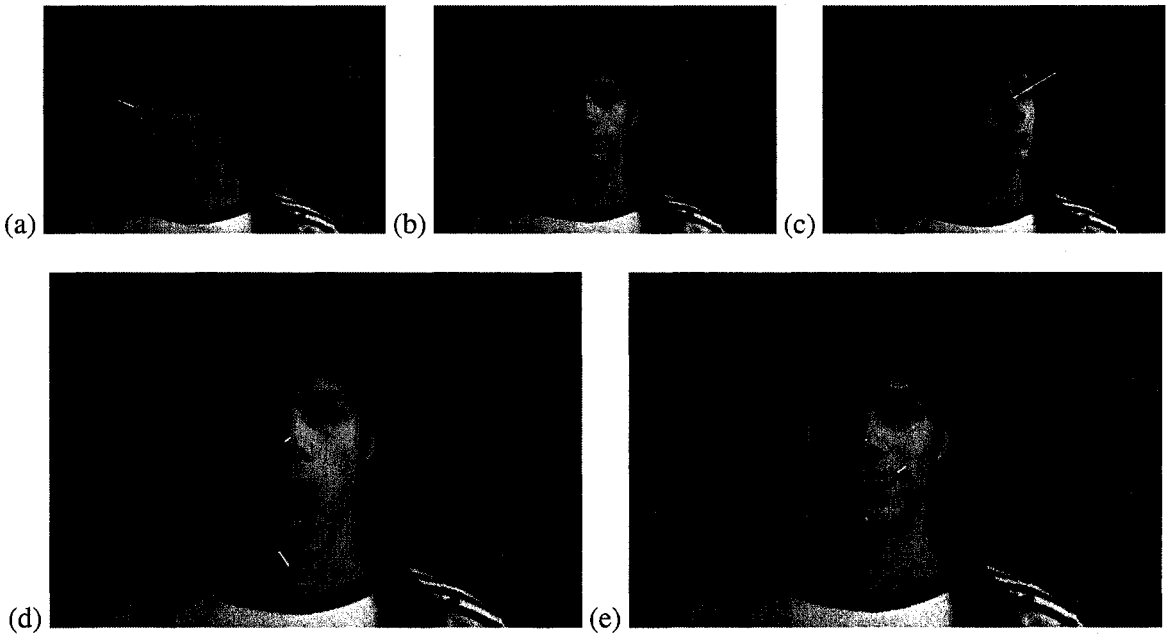


Figure 7.2: My scrubbing interface is used to interactively manipulate a video of a moving head by dragging it to the left and right (a-c). Additional constraints are applied to open the mouth (d), or keep the mouth closed and smile (e).

7.3 Storyboard scrubbing and editing

The virtual slider approach has several drawbacks. First, it relies on the trajectories of individual particles that may not survive through an entire shot due to occlusions, deformations, or varying illumination. Therefore it may not always be possible to drag an object through its entire range of motion using a single click and drag. Second, when a video features objects with repetitive or small screen-space motions — like a subject moving back and forth along a single path, or moving directly toward the camera — it may be hard or impossible to reach a desired frame using virtual sliders.

To address these limitations, we can use the aggregated motion and layout of schematic storyboards as an alternate temporal navigation interface. Storyboards feature aggregated motion, represented by motion arrows, and they explicitly handle overlapping or oscillating motion by splitting into multiple extended frame panels.

I have developed an intuitive interaction paradigm for rapidly selecting moments in a video clip that leverages the spatial relationships and representation of motion offered by storyboards. Clicking on any point in the storyboard with the mouse displays a specific frame in the video, and dragging the mouse with the mouse button depressed results in a continuous playback either forwards or backwards in time. Different parts of the storyboard invoke different actions:

Motion arrows. Clicking or dragging on a subject motion arrow retrieves a frame of the video when the subject appeared at that position of the arrow. Thus, dragging towards the head of an arrow moves forward in time, while dragging towards its tail moves backwards in time.

Background pixels. Clicking or dragging on a pixel not on a motion arrow retrieves the frame of the video in which the selected pixel is closest to the center of frame. Thus dragging the mouse left or right across a storyboard representing a panning camera motion gives the impression of dragging the camera itself left or right.

When rendering the storyboard, my system pre-renders selection buffers for these different regions and the associated temporal transformations so that the retrieval of the selected frame occurs at interactive rates.

I have prototyped a video editing interface using this technique, enabling the user to select in and out points of a clip and drag them to a timeline.

7.4 Conclusion

Each of the direct-manipulation scrubbing interfaces described here has some drawbacks, but I believe the approaches are complementary. Using virtual sliders, it may not always be possible to drag an object through its entire range of motion using a single click and drag. However, the object groupings can be employed using storyboard scrubbing: Particles in the same affine motion group are aggregated into a single motion arrow. Similarly, videos with repetitive or small screen-space motions may be hard to navigate using virtual sliders. Again, storyboard scrubbing does not suffer from this weakness as the storyboards are explicitly constructed so as to avoid overlapping arrows. However, virtual sliders have one key advantage over the storyboard interface: They permit easier fine-scale manipulations, since a motion arrow may be too simplistic a representation for small motions. In any case, I propose these interfaces not as a replacement for traditional scrollbars and jog/shuttle widgets, but rather as supplementary modes of manipulation.

One unresolved drawback of both navigation approaches is that over very long frame ranges the particle tracking has poorer accuracy, which can results in navigation to the wrong video frames. For example, in the constraint-based video control section of the demonstration video [29], the constraint placed on the nose actually can be seen to drift as far as the cheeks over several hundred frames. Finding a method to repair the particle tracks across long frame intervals is a topic for future work.

In this chapter, I have proposed novel interaction techniques for scrubbing through video using video objects and the natural spatial relationships of a storyboard.

Chapter 8

CONCLUSION

8.1 Contributions

In this thesis, I have demonstrated that computational analysis of video content can be utilized to construct effective, intuitive interfaces for video tasks, including dynamic graphical annotation, static visualization of motion, and temporal navigation by direct manipulation. My contributions include a novel motion grouping algorithm, a fluid interface for video annotation that is accessible to novice users, a formal summary of storyboard conventions, the first system for constructing storyboards from video, and an approach to video navigation using direct manipulation on video frames and storyboards.

8.2 Future work

I believe the work demonstrated in this thesis represents the first steps toward dynamic video interfaces of the future. It is my hope that some of the principles of this system — such as the use of precomputation to enable richer interaction and visualization techniques, and the application of direct manipulation interaction to video material — can be extended to future research. In this section I will enumerate some of the problem domains that have been opened by this line of research, and illustrate one or more potential projects within each domain:

Perception and computer vision for video interaction. The particle tracking and grouping algorithm described in Chapter 3 works for a wide variety of input shots, but there is room for improvement in terms of both quality and speed. Furthermore, there are other aspects of a video beyond raw motion that could be analyzed to facilitate video interfaces. For example, it remains to be seen whether Spottiswoode's "Content Curve" corresponds to any real model of visual attention, but if this concept could be approximated computationally, it would provide a powerful tool for computational video.

Figure 8.1: The first several feet of the Bayeux tapestry [1].

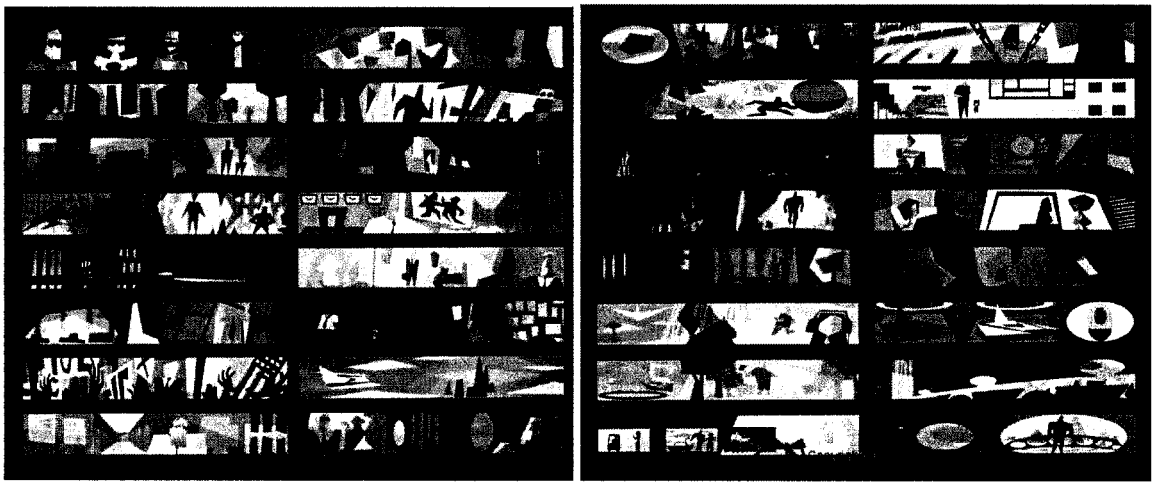


Figure 8.2: A section of the color script for the animated Pixar/Disney film *The Incredibles* [91]. This section represents roughly half of the film. Although someone familiar with the film can pick out scenes and characters, the purpose of this particular color script is to visualize the color palette and compositional style of the film. Other color scripts may emphasize tone and texture.

Visualization and navigation of long video segments. My current system focuses on individual shots of video. However, by analyzing higher-level aspects of the video such as location, character, and story arc, we can imagine visualizing longer video streams, and interactive methods for navigating their time axis.

Present interfaces for edited video do not have any effective visualization of longer sequences (more than 30 seconds). Although it might be impossible to visually represent all aspects of such a sequence, we might be able to focus on specific aspects of the video, such as color, texture, composition, character, or location, to construct an effective visualization along some subset of these dimensions. In such a visualization, one would ideally like to retain the coherent image content of a

thumbnail, with the spatial continuity of a VideoMAP [89]. We can think of such a visualization as a digital approximation of the Bayeux Tapestry, a medieval embroidery that illustrates the invasion of England and the Battle of Hastings in 1066. It explains events in chronological order from left to right over 231 feet of cloth, but unlike a comic book with separate panels, the narrative is told essentially in one long continuous panel (see Figure 8.1) [1]. Time moves left to right, though not necessarily at a constant speed. Filmmakers have used a related visualization called a *color script* — sometimes split into one panel per scene — to visualize and refine the color palettes, compositional style, tone and texture of films in preproduction (see Figure 8.2) [91]. If it were possible to approximate these types of visualizations from input video, I believe both professionals and consumers would have a new way to navigate long videos much more rapidly than is currently possible. Figure 8.3 is a hand-composed illustration showing one concept for how such a long-form video summary might appear. Automating the construction of such a visualization will require novel approaches in computer vision, layout, and compositing.

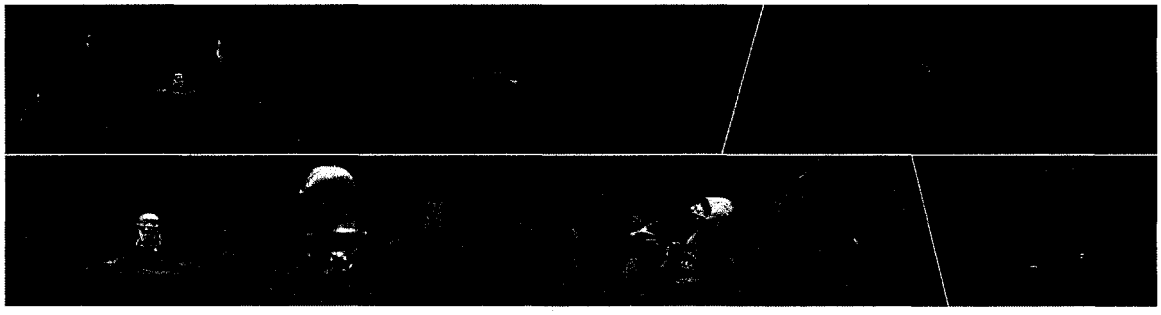


Figure 8.3: A tapestry visualization of a short segment of the film *Raiders of the Lost Ark*, constructed manually using Adobe Photoshop.

Shot visualization for video editing. The system I have presented has some utility for video editing tasks such as choosing cut points, but one can imagine a much broader spectrum of visualization tools informed by the concerns of professional editors, some of which were enumerated in Chapter 2. Video editors must often make choices between large collections of shots, and being able to visualize different juxtapositions without watching each possible edit would be invaluable. Some editorial concerns — such as emotions displayed by actors in the source footage — seem unlikely

to be understood by computers in the near future. But others — for example, object and camera motion, or continuum of movement — seem well-suited to automated understanding and improved visualization with current algorithms.

Bruce Block [12] has manually illustrated continuum of movement by superimposing storyboard arrow annotations from several shots (see Figure 8.4). I propose automating the presentation of this information, and possibly also assigning a score to the affinity or contrast of continuum of movement in a given sequence. This information can help editors identify cuts that could be improved, and possibly suggest improvements. Similar visualizations could assist in matching eyelines across cuts.

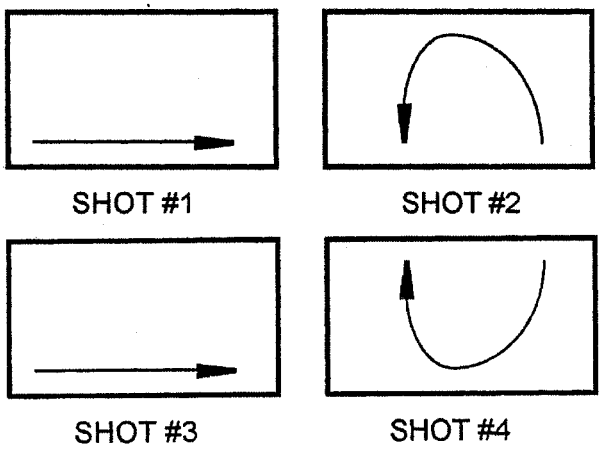


Figure 8.4: Bruce Block illustrates continuum of movement [12] by showing the path of the center of the viewer's attention across the frame within individual shots. In the upper pair of shots (#1 and #2), the center of attention remains in the same location across the cut, resulting in affinity of continuum of movement. In the lower pair of shots (#3 and #4), the center of attention jumps from one location to another across the cut, resulting in contrast of continuum of movement.

Although it is not yet feasible to automatically detect emotion from facial expressions, it may be possible to recognize changes in facial expression across multiple takes, for example to enable an editor to see a range of alternative line readings. This type of interface could be seen as a special case of keyframe selection [8, 92] for faces. Given multiple takes of the same line being spoken, it may be possible to synchronize several takes using both image and audio cues. Often it is desirable to use different takes for different portions of a line reading. Beyond simply improving visualization

of these editorial concepts, I hope that novel visualizations may reveal patterns to editors that were not visible in any previous form.

Automatically organizing shots by content. Current video editing interfaces allow the user to organize source footage into hierarchical folders. I envision a "smart bin" — akin to the smart folders in Apple's iPhoto and iTunes software — that could automatically select and order clips according to search criteria. For example, the user might create a "pan left" folder that would automatically identify clips containing a pan to the left, sorted by the speed of the panning motion. Additional useful search criteria might be minimum/maximum shot length, image-based or histogram matching [6], and face detection [93]. Context-dependent search criteria are also possible, such as finding shots similar to the currently displayed frame or shot. This would enable an editor to quickly find alternate takes for the current shot. Scrubbing the timeline to a different frame could automatically initiate a search for new matching shots. A simple modification — searching for shots similar to the *previous* shot — quickly identifies candidate reaction shots in a dialogue scene.

Video editing for novices. Taking the ideas in the previous sections several steps further, a video editing system for novices could abstract away the specifics of editorial choices, and simply suggest good places to cut. Such an interface need not be completely automatic, but could perhaps encourage traditional editing choices by using "snapping" types of interaction for choosing cut frames and juxtapositions, in which a mouse dragged near an optimal frame would snap to the right frame, or a group of shots dragged to an empty location on the timeline would automatically sort themselves into the best order of juxtaposition.

Taken to the logical conclusion, I envision an expert smart bin encompassing a recommender system that suggests candidate shots to insert at the current timeline location, based on the context and a rule book of film editing technique.

8.3 Summary

In summary, I believe that high-level visualization and user interface paradigms for searching, browsing, and editing video footage are in their infancy. In addition to the immediate open problems, I believe this area provides a rich vein of future research, since our ever-improving understanding of

human vision and perception will enable researchers to make better tools for editing videos based on higher-level goals. By focusing on the goals and practices of professional film editors I hope to uncover new ways that large volumes of video can be visualized and manipulated. It is my belief that such approaches can improve the ability of amateurs to work more easily with video as well, not only for editing but also for other tasks involving understanding and communicating properties of large video volumes, and transforming them in ways that we have only begun to explore.

BIBLIOGRAPHY

[1] The Bayeux Tapestry. `http://www.angelfire.com/rnb/bayeux_tapestry/`, 2006. [Online; accessed 4-April-2006].

[2] Brett Adams and Svetha Venkatesh. Director in your pocket: holistic help for the hapless home videographer. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 460–463, 2004.

[3] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. Interactive digital photomontage. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 23(4):294–301, 2004.

[4] Aseem Agarwala, Aaron Hertzmann, David H. Salesin, and Steven M. Seitz. Keyframe-based tracking for rotoscoping and animation. *ACM Trans. Graph.*, 23(3):584–591, 2004.

[5] Maneesh Agrawala and Chris Stolte. Rendering effective route maps: Improving usability through generalization. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 241–250. ACM Press / ACM SIGGRAPH, 2001.

[6] Aya Aner and John R. Kender. Video summaries through mosaic-based shot and scene clustering. In *Proc. ECCV '02*, pages 388–402, 2002.

[7] Aya Aner-Wolf and Lior Wolf. Video de-abstraction or How to save money on your wedding video. In *WACV '02: Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision*, page 264, 2002.

[8] Jackie Assa, Yaron Caspi, and Daniel Cohen-Or. Action synopsis: Pose selection and illustration. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 24(3):667–676, 2005.

[9] ASTAR Learning Systems. Video analysis for coaches, instructors, and more. `http://www.astarls.com`, 2006. [Online; accessed 3-September-2006].

[10] André Bazin. *Bazin at work: major essays & reviews from the forties and fifties.* Routledge, New York, 1997.

[11] Marcie Begleiter. *From word to image: storyboarding and the filmmaking process.* Michael Wiese Productions, 2001.

[12] Bruce A. Block. *The Visual Story: Seeing the Structure of Film, TV, and New Media.* Focal Press, 2001.

[13] Gabriel J. Brostow and Roberto Cipolla. Unsupervised bayesian detection of independent motion in crowds. In *Proc. CVPR*, volume 1, pages 594–601, 2006.

[14] Aeron Buchanan and Andrew Fitzgibbon. Interactive feature tracking using k-d trees and dynamic programming. In *Proc. CVPR*, pages 626–633, 2006.

[15] L.F. Cheong and H. Huo. Shot change detection using scene-based constraint. *Multimedia Tools and Applications*, 14(2):175–186, June 2001.

[16] Patrick Chiu, Andreas Girgensohn, and Qiong Liu. Stained-glass visualization for highly condensed video summaries. In *IEEE International Conference on Multimedia and Expo*, pages 2059–2062, 2004.

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

[18] cineSync. share your vision. `http://www.cinesync.com`, 2006. [Online; accessed 29-August-2006].

[19] Daniel Cremers and Stefano Soatto. Motion competition: A variational approach to piecewise parametric motion segmentation. *International Journal of Computer Vision*, 62(3):249–265, 2005.

[20] James E Cutting. Representing motion in a static image: constraints and parallels in art, science, and popular culture. *Perception*, 31:1165–1193, 2002.

[21] Jonathan Dakss, Stefan Agamanolis, Edmond Chalom, and Jr. V. Michael Bove. Hyperlinked video. In *Proc. SPIE*, volume 3528, pages 2–10, January 1999.

[22] Ron Davidson and David Harel. Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics*, 15(4):301–331, 1996.

[23] Daniel DeMenthon, Vikrant Kobla, and David Doermann. Video summarization by curve simplification. In *MULTIMEDIA '98: Proceedings of the sixth ACM international conference on Multimedia*, pages 211–218, New York, NY, USA, 1998. ACM Press.

[24] William T. Freeman and Hao Zhang. Shape-time photography. In *Proc. Computer Vision and Pattern Recognition*, pages 151–157, June 2003.

[25] Krzysztof Gajos and Daniel S. Weld. Preference elicitation for interface optimization. In *Proceedings of UIST '05*, 2005.

[26] Louis Giannetti. *Understanding Movies*, chapter 4, "Editing", pages 117–184. Prentice Hall, 6th edition, 1993.

[27] Andreas Girgensohn, John Boreczky, Patrick Chiu, John Doherty, Jonathan Foote, Gene Golovchinsky, Shingo Uchihashi, and Lynn Wilcox. A semi-automatic approach to home video editing. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 81–89, New York, NY, USA, 2000. ACM Press.

[28] Dan B Goldman, Brian Curless, Steven M. Seitz, and David Salesin. Schematic storyboarding for video visualization and editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3):862–871, July 2006.

[29] Daniel R. Goldman. *A Framework for Video Annotation, Visualization, and Interaction.* PhD thesis, University of Washington, 2007. [Available online at `http://grail.cs.washington.edu/theses/dgoldman/`].

[30] Jon Goldman. Kind of a Blur. `http://phobos.apple.com/WebObjects/MZStore.woa/wa/viewMovie?id=197994758&s=143441`, 2005. [Short film available online; accessed 21-May-2007].

[31] John Hart. *The art of the storyboard: storyboarding for film, TV and animation.* Focal Press, 1999.

[32] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*, page 109. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[33] Hasbro. Hasbro.com: VCAMNOW personal video recorder. `http://www.hasbro.com/tiger/default.cfm?page=hot&product_id=16726`, 2006. [Online; accessed 1-April-2006].

[34] B. Heisele, U. Kressel, and W. Ritter. Tracking non-rigid, moving objects based on color cluster flow. In *Proc. CVPR*, page 257, 1997.

[35] W.J. Heng and K.N. Ngan. An object-based shot boundary detection using edge tracing and tracking. *Journal of Visual Communication and Image Representation*, 12(3):217–239, September 2001.

[36] B. Horn, H. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America A*, 5(7):1127–1135, 1988.

[37] Michal Irani and P. Anandan. Video indexing based on mosaic representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 86(5):905–921, May 1998.

[38] Jiaya Jia and Chi-Keung Tang. Eliminating structure and intensity misalignment in image stitching. In *Proc. ICCV*, 2005.

[39] Nebojsa Jojic, Sumit Basu, Nemanja Petrovic, Brendan Frey, and Thomas Huang. Joint design of data analysis algorithms and user interface for video applications. presented at NIPS 2003 workshop on Machine Learning in User Interface, extended abstract at: `http://research.microsoft.com/workshops/MLUI03/jojic.html`, 2003.

[40] Alark Joshi and Penny Rheingans. Illustration-inspired techniques for visualizing time-varying data. In *IEEE Visualization*, page 86, 2005.

[41] Steven D. Katz. *Film directing shot by shot: visualizing from concept to screen.* Michael Wiese Productions, 1991.

[42] Y. Kawagishi, K. Hatsuyama, and K. Kondo. Cartoon blur: nonphotorealistic motion blur. In *Proc. Comp. Graph. Intl.*, pages 276–281, 2003.

[43] Byungmoon Kim and Irfan Essa. Video-based nonphotorealistic and expressive illustration of motion. In *Proc. Comp. Graph. Intl.*, pages 32–35, 2005.

[44] Anita Komlodi and Gary Marchionini. Key frame preview techniques for video browsing. In *DL '98: Proceedings of the third ACM conference on Digital libraries*, pages 118–125, New York, NY, USA, 1998. ACM Press.

[45] Lev Vladimirovich Kuleshov. *Kuleshov on film: writings.* University of California Press, Berkeley, 1974.

[46] M. Pawan Kumar, P. H. S. Torr, and A. Zisserman. Learning layered motion segmentations of video. In *Proc. ICCV*, pages 33–40, 2005.

[47] David Kurlander, Tim Skelly, and David Salesin. Comic chat. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 225–236, New York, NY, USA, 1996. ACM Press.

[48] M.S. Lee, Y.M. Yang, and S.W. Lee. Automatic video parsing using shot boundary detection and camera operation analysis. *Pattern Recognition*, 34(3):711–719, March 2001.

[49] Y. Li, T. Zhang, and D. Tretter. An overview of video abstraction techniques. Technical Report HPL-2001-191, HP Laboratories, 2001.

[50] Yin Li, Jian Sun, and Heung-Yeung Shum. Video object cut and paste. *ACM Trans. Graph.*, 24(3):595–600, 2005.

[51] Rainer Lienhart. Abstracting home video automatically. In *MULTIMEDIA '99: Proceedings of the seventh ACM international conference on Multimedia (Part 2)*, pages 37–40, New York, NY, USA, 1999. ACM Press.

[52] Rainer Lienhart. Comparison of automatic shot boundary detection algorithms. In *Proc. Image and Video Processing VII 1999*, 1999.

[53] Ce Liu, Antonio Torralba, William T. Freeman, Frédo Durand, and Edward H. Adelson. Motion magnification. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 24(3):519–526, 2005.

[54] Simon Lok and Steven Feiner. A survey of automated layout techniques for information presentations. In *Proceedings of SmartGraphics 2001*, 2001.

[55] Sidney Lumet. *Making Movies*, chapter 9, "The Cutting Room: Alone at Last", pages 148–169. Bloomsbury Publishing, 1995.

[56] M. Massey and W. Bender. Salient stills: process and practice. *IBM Systems Journal*, 35(3,4):557–574, 1996.

[57] Maic Masuch, S. Schlechtweg, and R. Schultz. Speedlines: depicting motion in motionless pictures. In *ACM SIGGRAPH 99 Conference abstracts and applications*, page 277, 1999.

[58] Scott McCloud. *Understanding Comics: The Invisible Art.* HarperCollins, 1993.

[59] C. Morningstar and R. F. Farmer. The lessons of Lucasfilm's Habitat. In M. Benedikt, editor, *Cyberspace: First Steps*, pages 273–301. MIT Press, Cambridge, MA, 1991.

[60] Walter Murch. *In the Blink of an Eye: A Perspective on Film Editing.* Silman-James Press, Los Angeles, 1995.

[61] H. Nicolas, A. Manaury, J. Benois-Pineau, W. Dupuy, and D. Barba. Grouping video shots into scenes based on 1D mosaic descriptors. In *Proc. Intl. Conf. on Image Proc.*, pages I: 637–640, 2004.

[62] Marc Nienhaus and Jürgen Döllner. Dynamic glyphs – depicting dynamics in images of 3D scenes. In *Third International Symposium on Smart Graphics*, pages 102–111, 2003.

[63] Michael Ondaatje. *The Conversations: Walter Murch and the Art of Editing Film.* Knopf, 2002.

[64] S. Pfeiffer, R. Lienhart, S. Fischer, and W. Effelsberg. Abstracting digital movies automatically. *Journal of Visual Communication and Image Representation*, 7(4):345–353, December 1996.

[65] V. I. Pudovkin. *Film Technique and Film Acting*. Lear, New York, 1949.

[66] Zenon W. Pylyshyn. *Seeing and Visualizing: It's not what you think (Life and Mind)*. Bradford, 2003.

[67] Gonzalo Ramos and Ravin Balakrishnan. Fluid interaction techniques for the control and annotation of digital video. In *Proc. UIST '03*, pages 105–114, 2003.

[68] Edward Rosten, Gerhard Reitmayr, and Tom Drummond. Real-time video annotations for augmented reality. In *Proc. International Symposium on Visual Computing*, 2005.

[69] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "GrabCut" – interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 23(3):309–314, 2004.

[70] Michael Rubin. *Droidmaker: George Lucas and the digital revolution*. Triad Publishing, 2005. 327,338.

[71] Peter Sand. *Long-Range Video Motion Estimation using Point Trajectories*. PhD thesis, Massachusetts Institute of Technology, 2006.

[72] Peter Sand and Seth Teller. Particle video: Long-range motion estimation using point trajectories. In *Proc. CVPR '06*, volume 2, pages 2195–2202, 2006.

[73] Cindy Sangster. Personal communication, 2005.

[74] Arno Schödl and Irfan A. Essa. Controlled animation of video sprites. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 121–127, 2002.

[75] Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa. Video textures. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 489–498, 2000.

[76] Mark Simon. *Storyboards: Motion in Art*. Focal Press, 2000.

[77] J. Sivic, F. Schaffalitzky, and A. Zisserman. Object level grouping for video shots. *International Journal of Computer Vision*, 67(2):189–210, 2006.

[78] Michael Smith and Takeo Kanade. Video skimming and characterization through the combination of image and language understanding. In *Proc. IEEE International Workshop on Content-Based Access of Image and Video Databases*, pages 61 – 70, January 1998.

[79] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3D. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3):835–846, 2006.

[80] Scott S. Snibbe, Karon E. MacLean, Rob Shaw, Jayne Roderick, William L. Verplank, and Mark Scheeff. Haptic techniques for media control. In *Proc. UIST '01*, pages 199–208. ACM Press, 2001.

[81] Sportvision. Changing The Game. `http://www.sportvision.com`, 2006. [Online; accessed 29-August-2006].

[82] Raymond Spottiswoode. *A Grammar of the Film*. University of California Press, Berkeley, 1950.

[83] Xinding Sun and Mohan S. Kankanhalli. Video summarization using R-sequences. *Real-Time Imaging*, 6(6):449–459, 2000.

[84] Louise Sweeney. John Huston; profile. *The Christian Science Monitor*, page 13, 11 Aug 1973.

[85] Yukinobu Taniguchi, Akihito Akutsu, and Yoshinobu Tonomura. PanoramaExcerpts: extracting and packing panoramas for video browsing. In *Proc. ACM Intl. Conf. on Multimedia*, pages 427–436, 1997.

[86] Laura Teodosio and Walter Bender. Salient video stills: Content and context preserved. In *Proc. ACM Intl. Conf. on Multimedia*, pages 39–46, 1993.

[87] Laura Teodosio and Walter Bender. Salient stills. *ACM Trans. on Multimedia Comp., Comm., and Appl.*, 1(1):16–36, February 2005.

[88] Vineet Thanedar and Tobias Höllerer. Semi-automated placement of annotations in videos. Technical Report 2004-11, UC, Santa Barbara, 2004.

[89] Yoshinobu Tonomura, Akihito Akutsu, Kiyotaka Otsuji, and Toru Sadakata. VideoMAP and VideoSpaceIcon: tools for anatomizing video content. In *Proc. SIGCHI*, pages 131–136, New York, NY, USA, 1993. ACM Press.

[90] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.

[91] Mark Cotta Vaz. *The Art of The Incredibles*. Chronicle Books LLC, San Francisco, California, 2004.

[92] Jaco Vermaak, Patrick Pérez, Michel Gangnet, and Andrew Blake. Rapid summarization and browsing of video sequences. In *British Machine Vision Conference*, volume 1, pages 424–433, 2002.

[93] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR*, volume 1, page 511, 2001.

[94] Jue Wang, Pravin Bhat, R. Alex Colburn, Maneesh Agrawala, and Michael F. Cohen. Interactive video cutout. *ACM Trans. Graph.*, 24(3):585–594, 2005.

[95] J.L. Ward. *Perception and Pictorial Representation*, volume 1, chapter 13, "A piece of the action: Moving figures in still pictures", pages 246–271. Praeger, New York, 1979.

[96] Y. Wexler and D. Simakov. Space-time scene manifolds. In *Proc. ICCV*, pages 858–863, 2005.

[97] Wikipedia. Flatbed editor — Wikipedia, The Free Encyclopedia. `http://en.wikipedia.org/w/index.php?title=Flatbed_editor&oldid=41671972`, 2006. [Online; accessed 1-April-2006].

[98] Wikipedia. Telestrator — Wikipedia, The Free Encyclopedia. `http://en.wikipedia.org/w/index.php?title=Telestrator&oldid=64269499`, 2006. [Online; accessed 28-August-2006].

[99] Daniel N. Wood, Adam Finkelstein, John F. Hughes, Craig E. Thayer, and David H. Salesin. Multiperspective panoramas for cel animation. In *Proc. SIGGRAPH*, pages 243–250, 1997.

[100] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):13, December 2006.

[101] Lihi Zelnik-Manor, Gabriele Peters, and Pietro Perona. Squaring the circle in panoramas. In *Proc. ICCV*, pages 1292–1299, 2005.

[102] D. Zhong, H. Zhang, and S.-F. Chang. Clustering methods for video browsing and annotation. In I. K. Sethi and R. C. Jain, editors, *Proc. SPIE, Storage and Retrieval for Still Image and Video Databases IV*, volume 2670, pages 239–246, March 1996.

[103] Cai-Zhi Zhu, Tao Mei, and Xian-Sheng Hua. Natural video browsing. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 265–266, New York, NY, USA, 2005. ACM Press.

Appendix A

COMPLETE STORYBOARDS: "KIND OF A BLUR"

To assess the performance of my system on real-world data, I attempted to use it to create schematic storyboards for every shot in the short film, "Kind of a Blur" [30]. The main body of the film (not including head and tail credit sequences) was manually segmented into 89 separate shots. Of these, 25 were representable as single-frame storyboards with no annotations. For the remaining 64 shots, I performed preprocessing on uncompressed 720×480 video frames, although the original footage was captured on DV with 4:2:2 chroma compression.

Of the 64 shots attempted, 35 shots resulted in complete and acceptable storyboards. The remaining 29 were not completed satisfactorily for the following reasons (some shots had multiple problems). The particle video algorithm failed significantly on seventeen shots: Eleven due to motion blur, three due to large-scale occlusions by foreground objects, and three due to moving objects too small to be properly resolved. Of the remaining shots, the grouping algorithm failed to converge properly for five shots. Six shots included some kind of turning or rotating object, but our system only explicitly handles translating objects. Three shots featured complex camera moves in which the camera both translates and rotates: Such shots are not easily represented using standard storyboard notation, sometimes requiring additional schematic diagrams. Three shots featured unsatisfactory composites, due to failures of the heuristics for compositing depth order. Three shots featured key poses that were occluded due to compositing, which can be viewed as a failure of the extended frame segmentation algorithm. Two shots featured oscillating motion (a hand moving back and forth), which is not well represented using our system.

The completed storyboards are shown on the following pages in Figures A.1-A.9. For reference, the original storyboards drawn for the film are shown in Figures A.10-A.15.

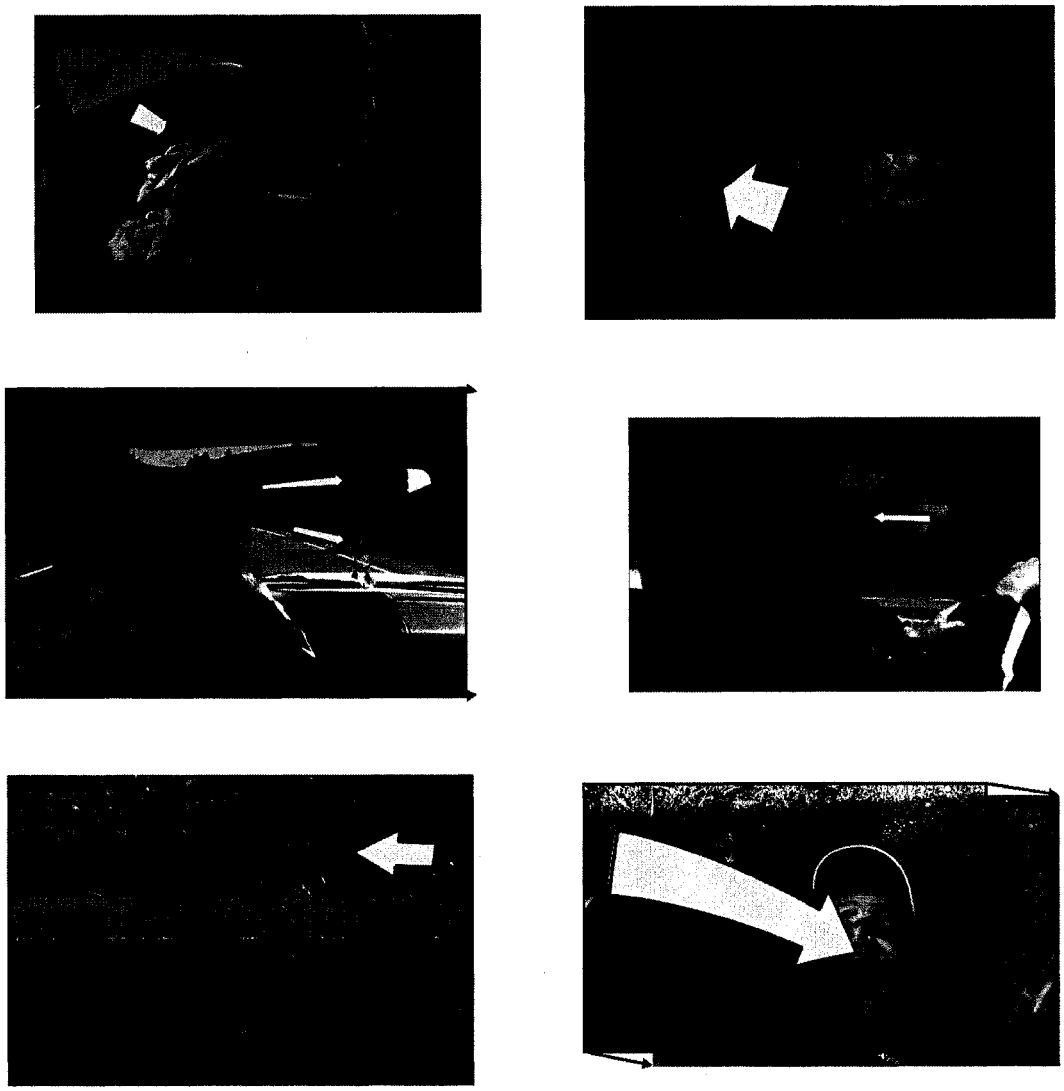


Figure A.1: Storyboards from the "Road Side" sequence.

Figure A.2: Storyboards for the "Road Side" sequence (continued).

Figure A.3: Storyboards for the "Glove Compartment" sequence.

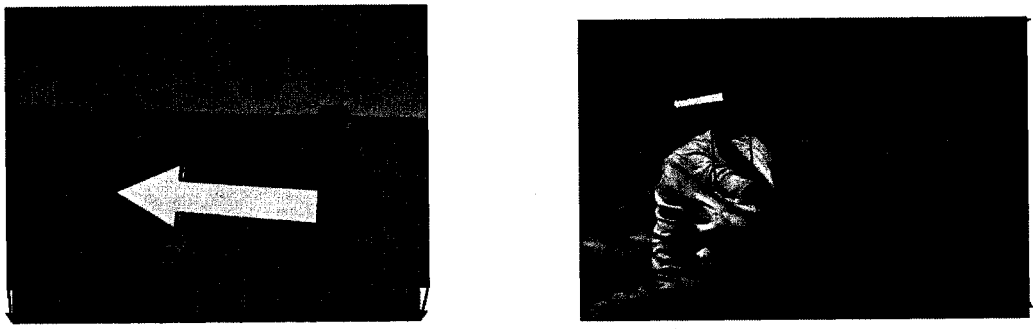


Figure A.4: Storyboards for the "Cow Search" sequence.

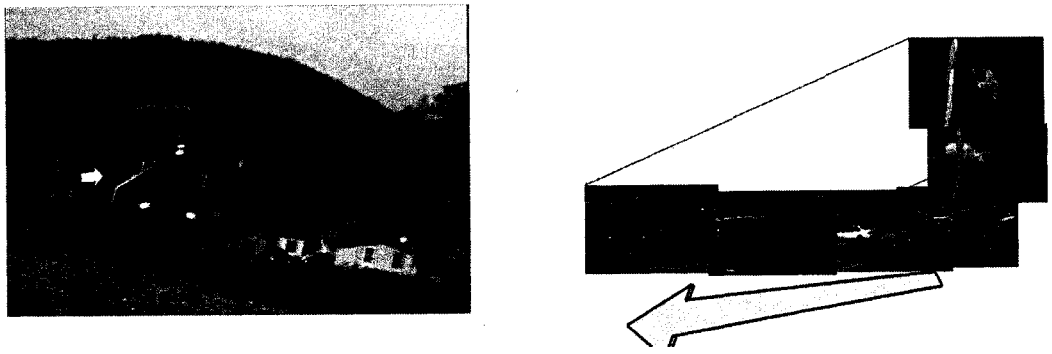


Figure A.5: Storyboards for the "Drive Talk" sequence.

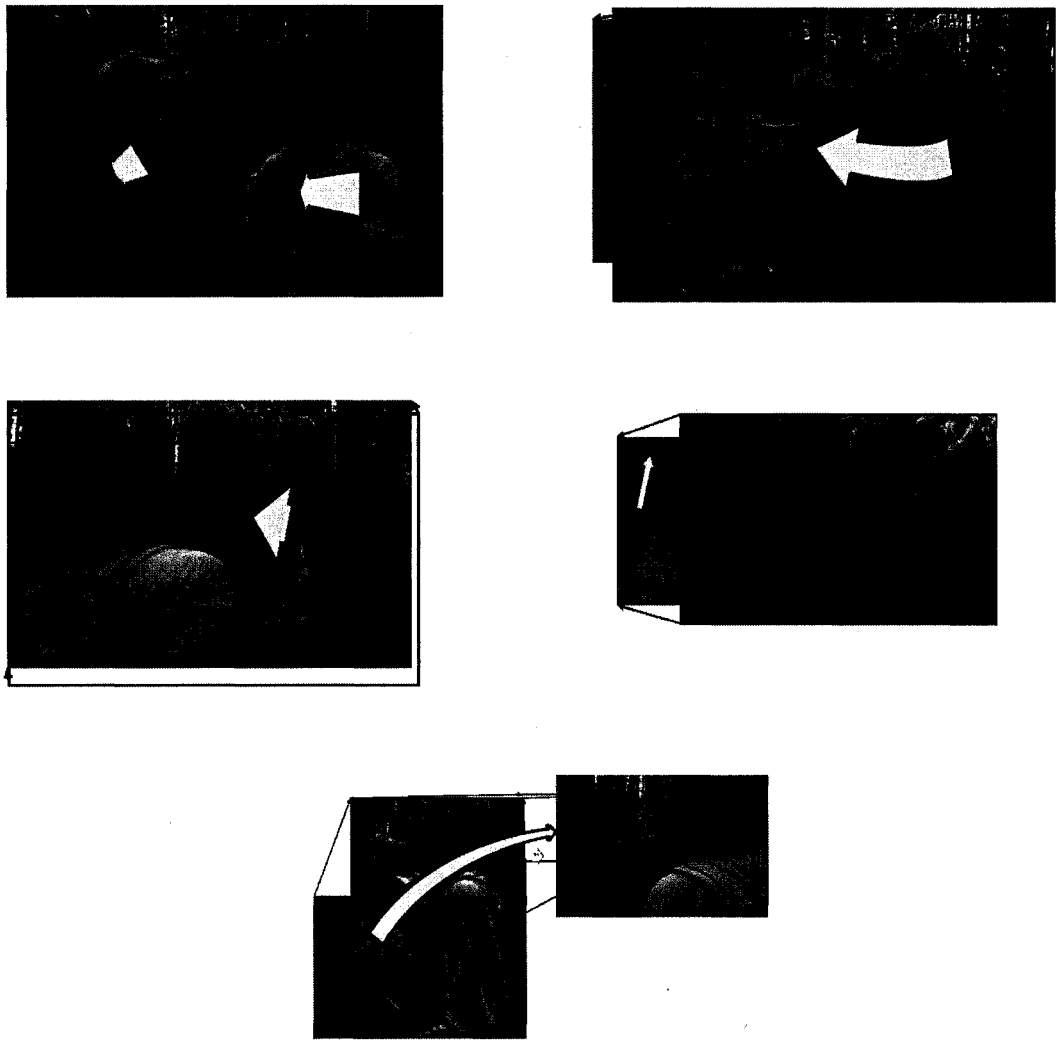


Figure A.6: Storyboards for the "Forest Walk" sequence.

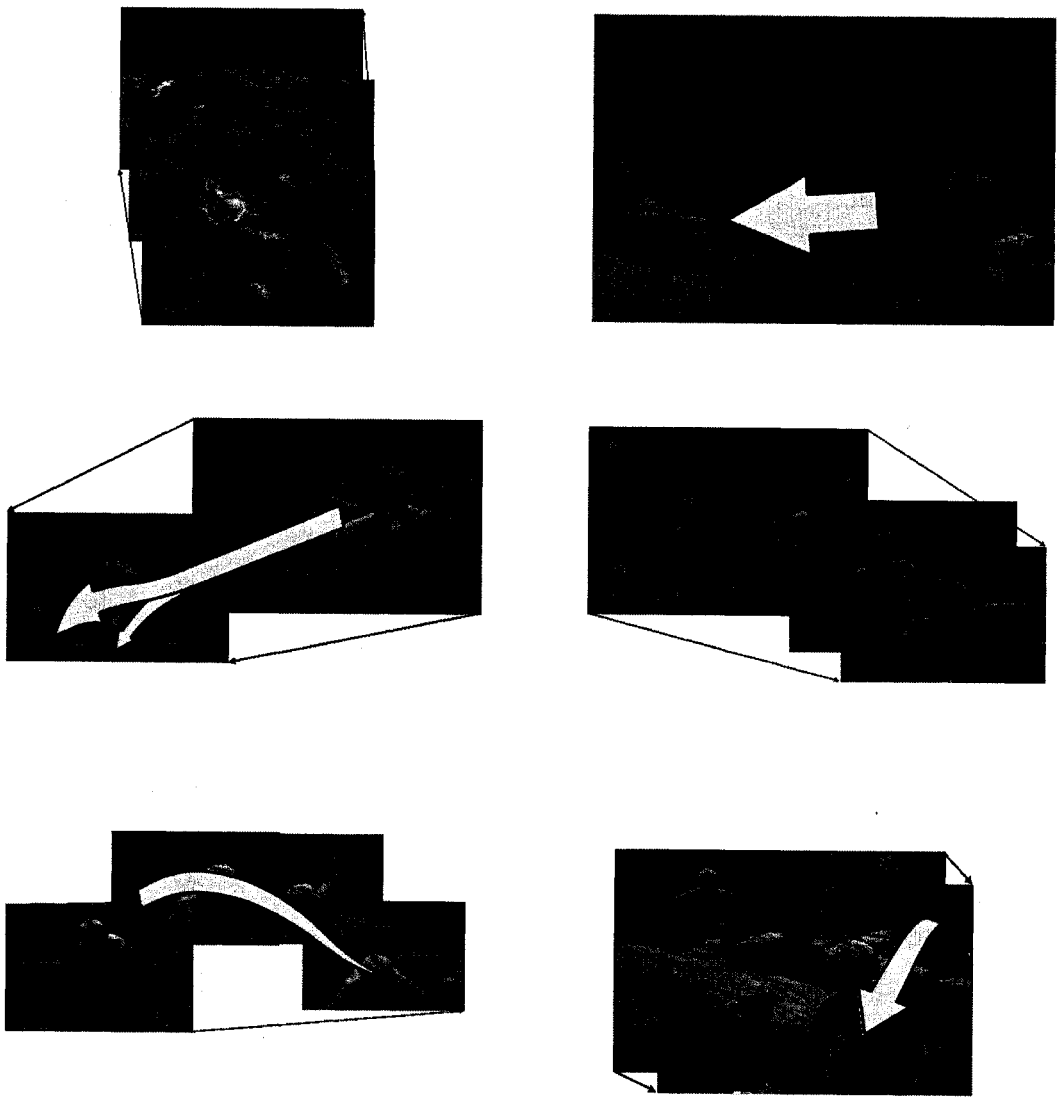


Figure A.7: Storyboards for the "Stream Swim" sequence.

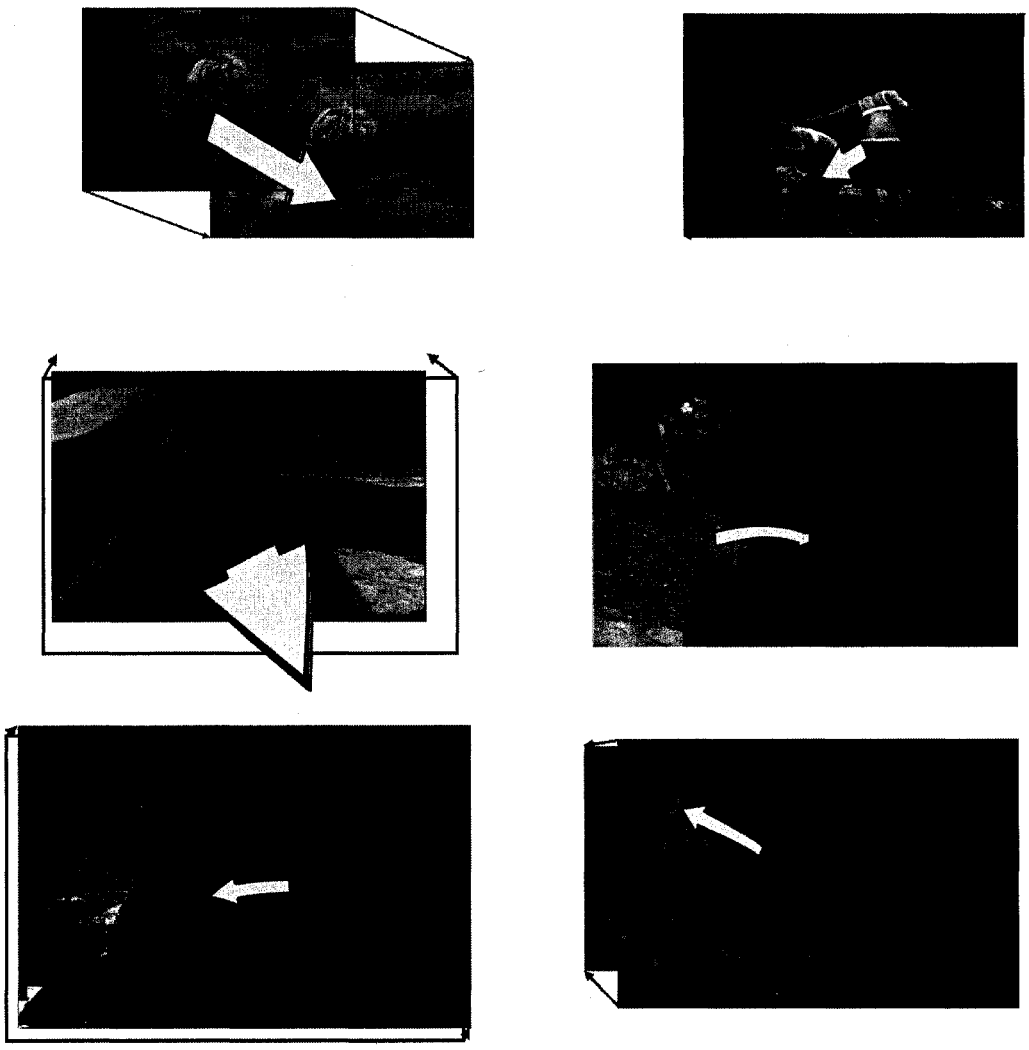


Figure A.8: Storyboards for the "Stream Swim" sequence (continued).

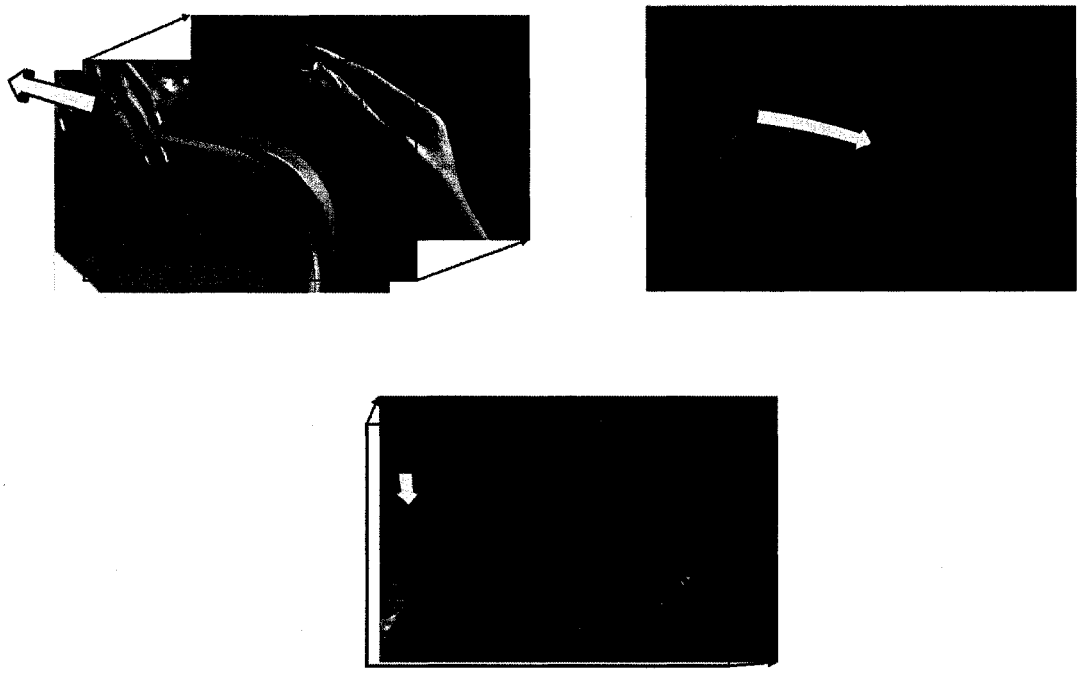


Figure A.9: Storyboards for the "Stream Swim" sequence (continued).

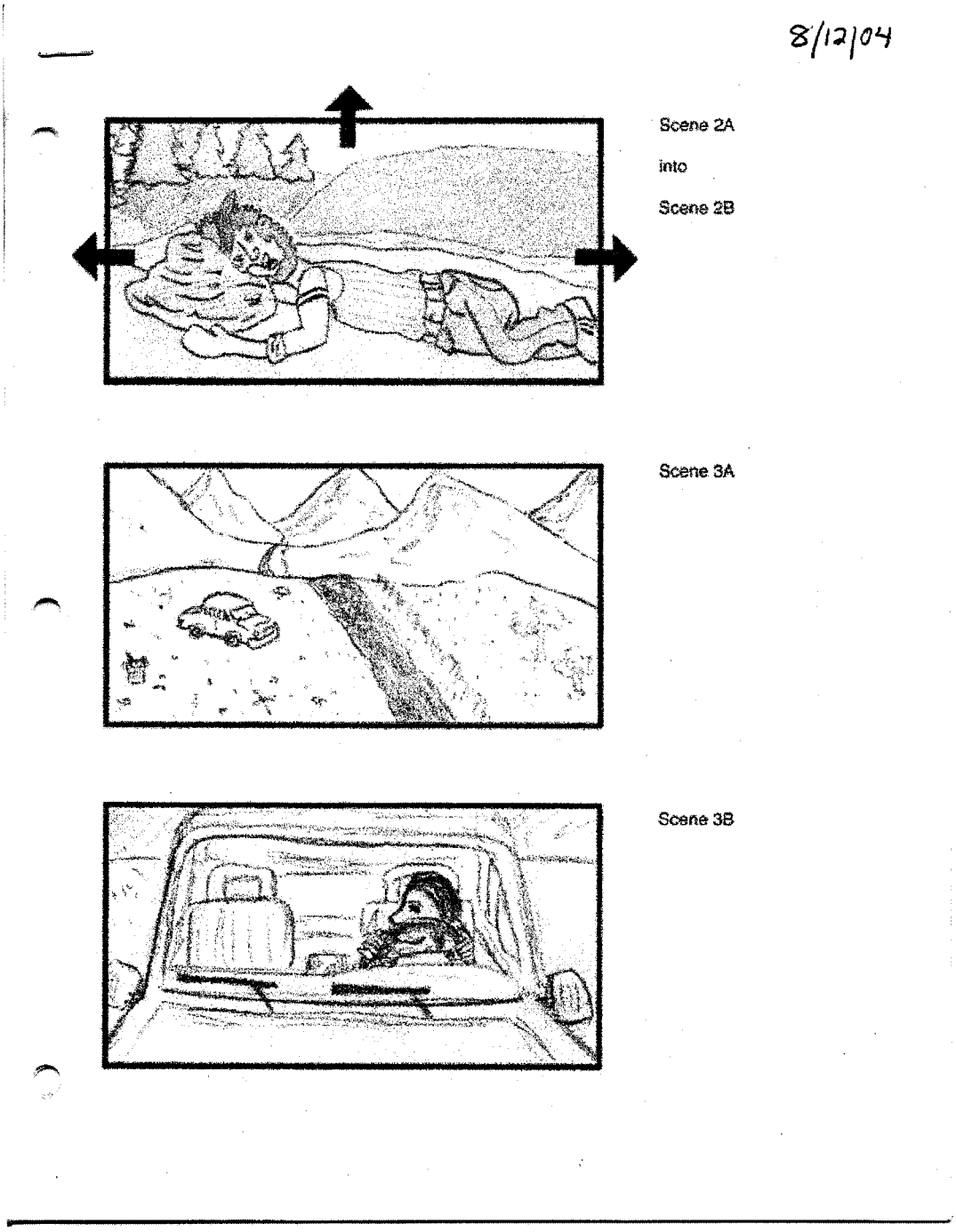


Figure A.10: Original hand-drawn storyboards for *Kind of a Blur*. ©2005 Jon Goldman

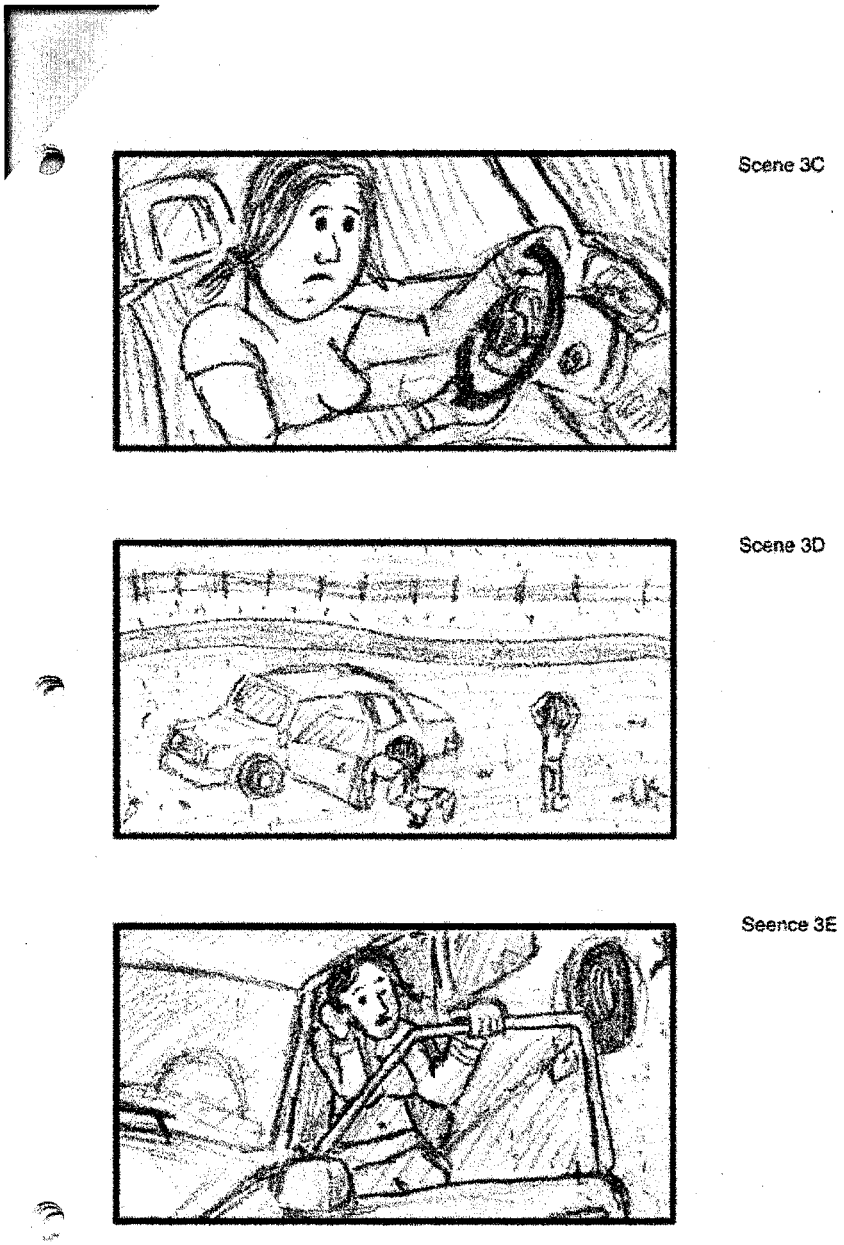


Figure A.11: Original hand-drawn storyboards for *Kind of a Blur*. ©2005 Jon Goldman

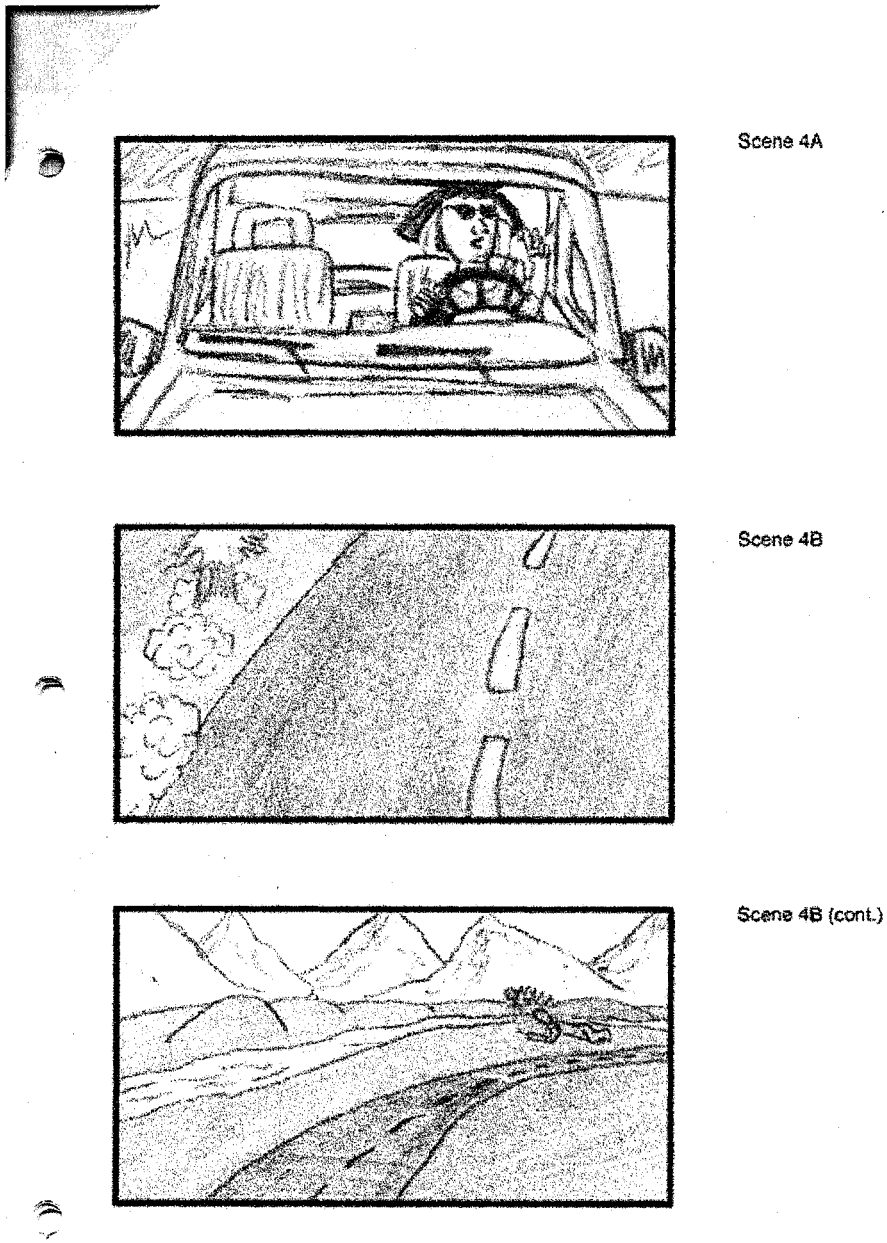


Figure A.12: Original hand-drawn storyboards for *Kind of a Blur*. ©2005 Jon Goldman

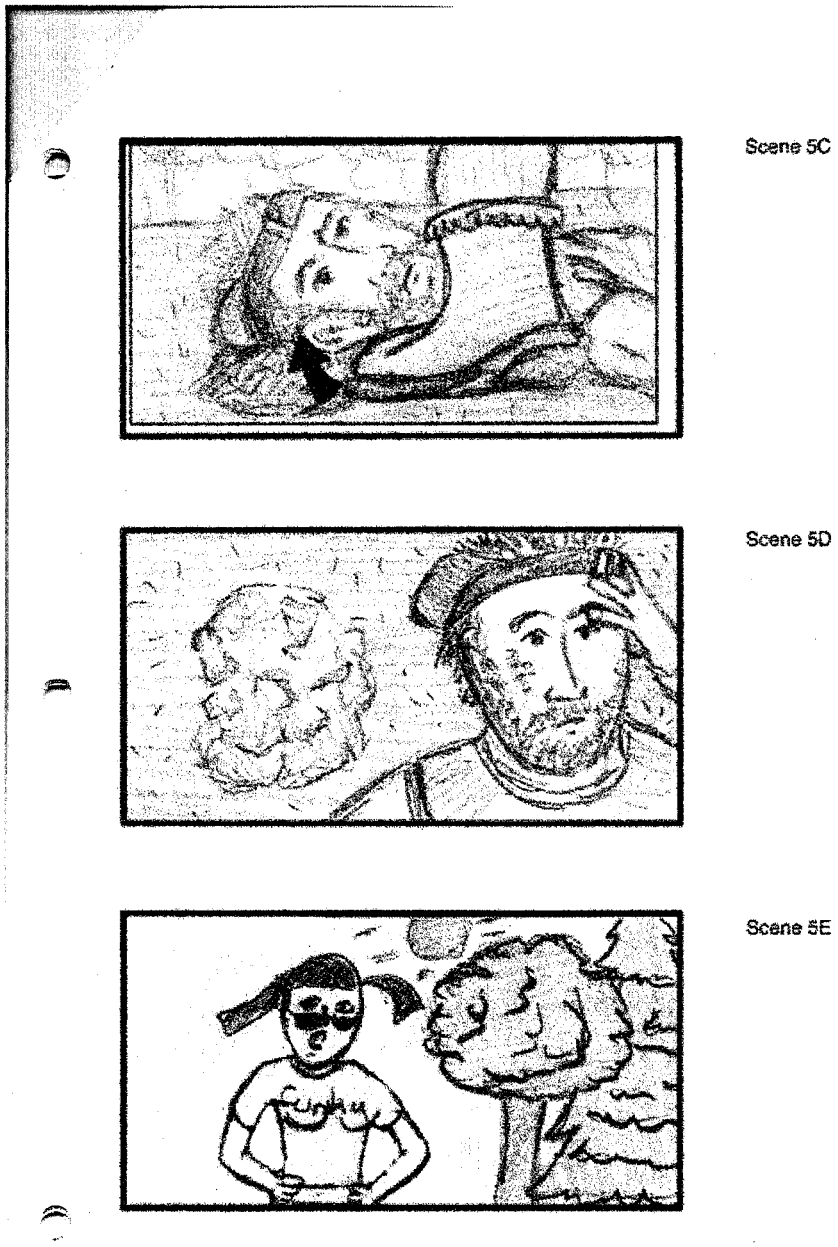


Figure A.13: Original hand-drawn storyboards for *Kind of a Blur*. ©2005 Jon Goldman

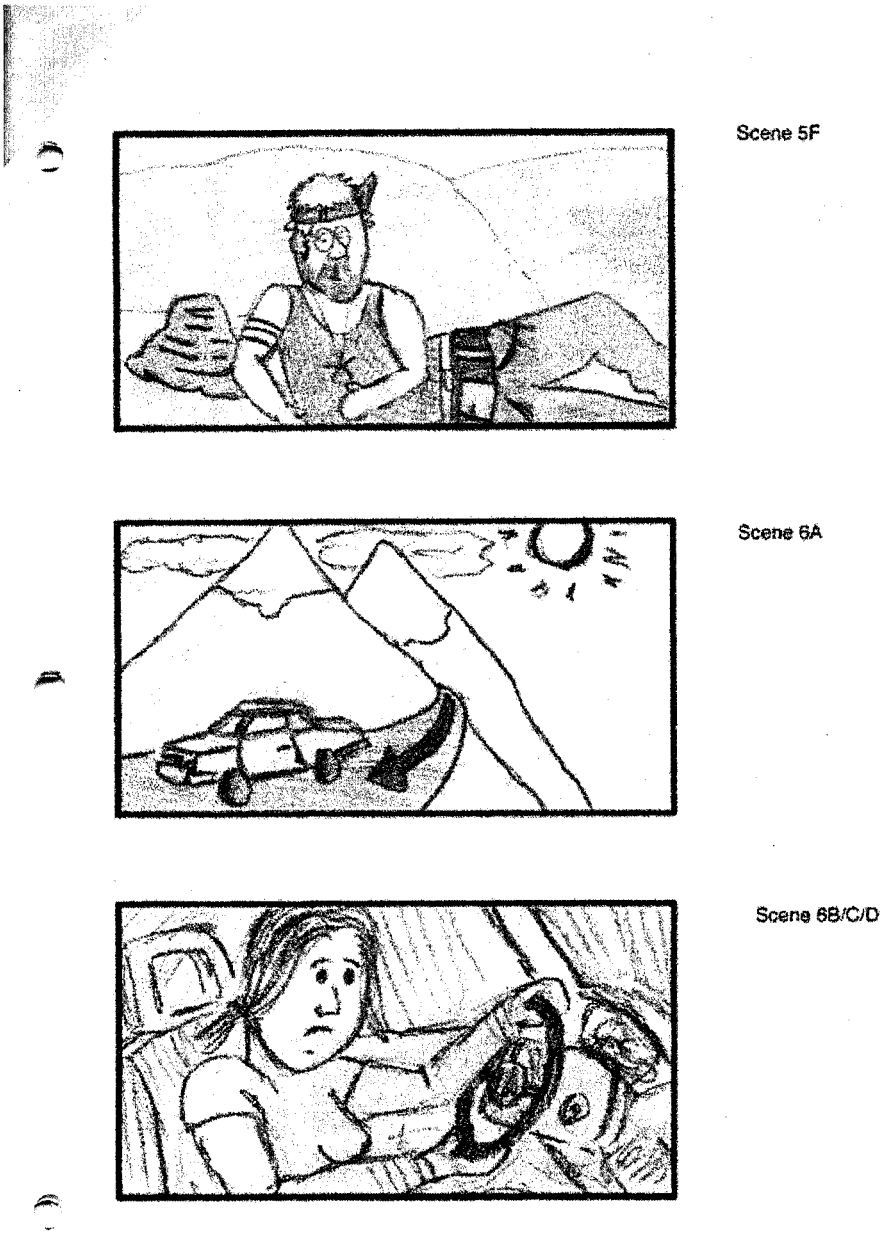


Figure A.14: Original hand-drawn storyboards for *Kind of a Blur*. ©2005 Jon Goldman

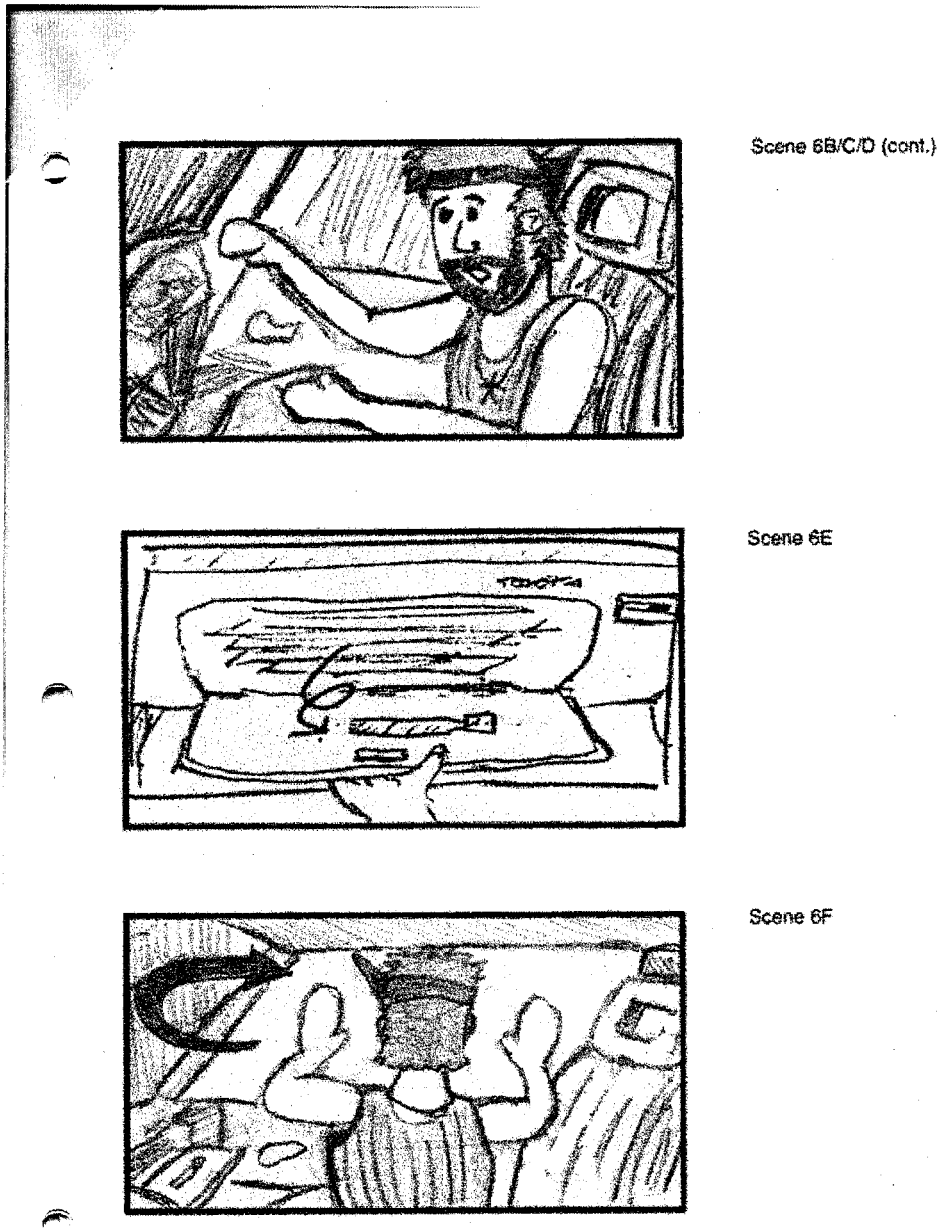


Figure A.15: Original hand-drawn storyboards for *Kind of a Blur*. ©2005 Jon Goldman

VITA

Dan Goldman received his B.Sc. in 1994 and M.Sc. in 1995 from Stanford University's Computer Science department. After graduation he worked for seven years as an artist, programmer, and researcher at Industrial Light & Magic, a division of Lucasfilm. In 2002, he joined the Computer Science and Engineering department at the University of Washington as a graduate student. After five and a half years of study, he received the Doctor of Philosophy degree in June 2007, and he is now employed as a senior research scientist at Adobe Systems, Inc., in Seattle, Washington.