Evaluation of direct manipulation techniques for in-scene video navigation

Diploma Thesis at the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Christian Brockly

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Martina Ziefle

Registration date: June 16th, 2008
Submission date: February 9th, 2009

Contents

List of Figures

List of Tables

Abstract

We provide a brief survey of current video navigation techniques and group them in a classification space. A detailed analysis of current interfaces shows disadvantages of timeline-based systems, especially for in-scene navigation tasks, including the spatial separation of content and controls, modal timeline resolution, and a missing natural mapping between input mouse movements and output object motion. An online survey gives us insights on how users currently interact with video material.
Based on these information, we motivate a direct manipulation technique for interaction with objects in a video, used in a variety of current research projects. In our analysis of this novel approach, we examine user performance with different combinations of movement classes (straight lines, curves, waves, and edges) and motion path visualisations (arrows and trajectories). Furthermore, we introduce density dots and variable width trajectories to represent object velocity.
Objects that pause for a certain amount of time can introduce problems and make interaction hard. A user study investigates advantages and disadvantages of approaches to handle such situations. We introduce the concepts of overlay crossers and pause loops.
Furthermore, we explore the use a of trajectory segmentation approach to handle pendulum-like trajectories. A study investigates the use of an inertia feature, fixed direction scrubbing, overlay crossers, and an extended trajectory visualisation.

Überblick

Wir geben einen kurzen Überblick über aktuelle Techniken zur Navigation in Videos und klassifizieren die entsprechenden Systeme. Eine detaillierte Analyse zeigt Nachteile von Zeitleisten-basierten Systemen bezüglich Frame-genauer Navigation auf. Darunter sind die örtliche Trennung des Videoinhaltes von den Steuerelementen, die modale Zeitleistenauflösung und das Fehlen einer natürlichen Übereinstimmung zwischen Mausbewegungen als Eingabe und Objektbewegungen als Ausgabe. Eine Onlineumfrage gibt uns Informationen darüber, wie Nutzer aktuell mit Videomaterial umgehen.
Darauf aufbauend motivieren wir die Nutzung von direkter Manipulation für die Interaktion mit in Videos vorhandenen Objekten. Diese Technik wird bereits in einer Vielzahl von Forschungsprojekten genutzt. Als Teil unserer Betrachtung dieses neuen Ansatzes untersuchen wir das Verhalten von Benutzern bei verschiedenen Kombinationen von Objektbewegungen (Geraden, Kurven, Wellen, Kanten) und deren Visualisierungen (Pfeile und Trajektorien). Darüber hinaus stellen wir Möglichkeiten vor, die Geschwindigkeit von Objekten mit Hilfe von Punkten und Trajektorien variabler Breite darzustellen.
Objekte, die für eine gewisse Zeit ihre Bewegung unterbrechen, stellen die Technik vor Probleme und können die Interaktion erschweren. In einer Studie untersuchen wir die Vor- und Nachteile von Ansätzen, die diese Situationen lösen können, und stellen dabei die Konzepte *overlay crosser* und *pause loop* vor.
Schließlich untersuchen wir die Nutzung eines Ansatzes zur Segmentierung von Trajektorien, der die Interaktion mit Objekten, die sich auf Pendel-ähnlichen Bahnen bewegen, verbessern soll. Eine Studie untersucht dabei die Nutzung eines Trägheitsansatzes, der Navigation mit fixierter Zeitrichtung, des Konzeptes des *overlay crossers* und einer erweiterten Trajektorienvisualisierung.

Acknowledgements

Thank you,
Christian Brockly

Conventions

Throughout this thesis we use the following conventions.

Text conventions

Definitions of technical terms or short excursus are set off in orange boxes.

EXCURSUS:
Excursus are detailed discussions of a particular point in a book, usually in an appendix, or digressions in a written text.

Definition:
Excursus

Implementation details are set off in yellow boxes.

ALGORITHM:
If user presses key RETURN, print "Hello World". Otherwise, exit.

Implementation:
Algorithm

The whole thesis is written in British English.

The plural "we" will be used throughout this thesis instead of the singular "I", even when referring to work that was primarily or solely done by the author.

Chapter 1

Introduction

"Imagination is more important than knowledge. For knowledge is limited to all we now know and understand, while imagination embraces the entire world, and all there ever will be to know and understand."

—Albert Einstein

Number of video software users is rising

The processing power of, both commercial and home, computers is constantly rising. Almost every new cell phone or digital camera has the ability to capture video material, and the access to video content due to broadband internet connections has become normality. The video portal YouTube[1] is one of the most successful sites on the internet, and it serves as a prime example for this development. In October 2006, more than 65.000 videos were uploaded and more than 100.000.000 viewed per day. As the number of nonprofessional video users and editors is rising, the demand for easy video editing systems and better video playback interfaces, i.e., more intuitive interaction techniques, is increasing every day.

There is CPU power for advanced interfaces

Today's commercially successful video systems use powerful algorithms for video compression, playback, audio/video synchronization, streaming, and other aspects. Although modern multimedia systems have processing

[1]http://www.youtube.com

power available for advanced user interfaces and interaction techniques, the user interface for video navigation remained more or less the same, compared to traditional video devices.

1.1 Analogue video navigation

Analogue devices provide buttons for start, stop, fast-forward and rewind

On VHS recorders, video projectors, and other video playback devices—some of them more than 30 years old—one can find at least five buttons: start, stop, pause[2], fast-forward, and rewind. These buttons provide the user with the necessary functions to watch a video and to perform basic reviewing tasks—to review a certain part, she uses the rewind function, and to search for a certain scene in the future, she uses the fast-forward function, respectively. Furthermore, the pause button is a useful command to examine the content of a single image or to just stop the playback.

However, when taking a deeper look at these functions, one can see that they are not sufficient for many purposes that go beyond watching an entire movie from start to end. For example, fine-tuning the video playback position or reviewing the same video interval repeatedly at different speeds and in different directions as mentioned in [Kimber et al., 2-5 July 2007] is hardly possible. The following example illustrates this problem:

> John is an amateur movie maker, and he is filming the high school basketball match of his son Jake. Jake's team is about to lose by one point and Jake is in possession of the ball. With the buzzer announcing the end of the game he shoots and scores. Jake just made a *buzzer beater*. But the referee decides that his shot was too late and so he does not give the points. To prove that the points were regular, John tries to show to the referee that the ball had left his son's hands before the buzzer.

[2]Many devices combine the start and pause function in a single play/pause button.

> However, he is not able to scroll the video to the right position on time. The camera rewinds and forwards the tape so fast that, when he presses the pause button, he always ends up a few seconds before or after the buzzer.

Finding playback position is unnecessarily hard

We can observe that finding the right playback position and reviewing the material, which is a common task and should be simple, turns out to be a real challenge when using only play, pause, fast-forward and rewind. Accomplishing the task is hard because of the following reasons:

- The playback speed can either not be controlled or only in form of predefined speeds like 2x, 4x, 8x.
- In case the user misses the right moment when fast-forwarding, he has to rewind and may miss the right position again.
- Pushing the pause button in the right moment requires huge attention and concentration.

Analogue devices do not offer many alternatives

Of course, the developers of analogue video cameras and projectors did not have many alternatives. As videos were recorded on tape in an analogue and continuous way, to reach a certain position or to get from the beginning to the end of a clip, one had to either forward or rewind the tape. Real-time random access was not possible. With digital media, the situation is different; random access is now possible.

1.2 Digital video navigation

Digital devices use both old and new interface components

It is not surprising that software video players like VLC (see figure 1.1) provide the traditional functions—they are useful, well-known, and easy to understand. However, we have seen that they have some inherent problems. Research groups and video equipment developers have already provided extensions in form of additional buttons or interface components such as timelines, velocity sliders [Ramos and Balakrishnan, 2003], jog wheels, and chapters, extending the navigation capabilities in professional systems as well as in home use appliances.

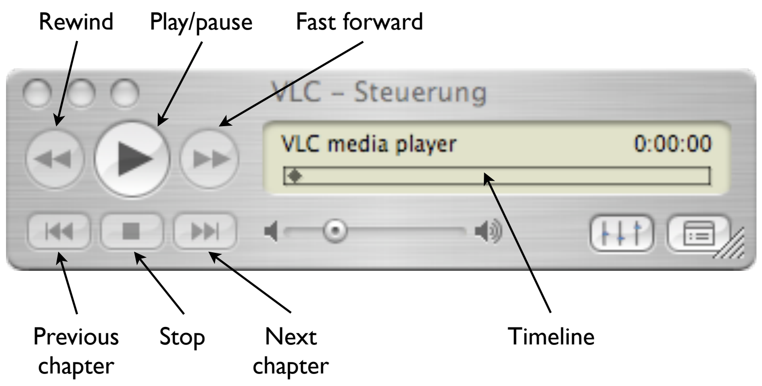

Figure 1.1: VLC media player interface with VCR-like controls and timeline.

1.2.1 Micro and macro navigation

For our further analysis, we have to define the concept of scenes and distinguish two different concepts—micro navigation and macro navigation.

Definition: *Scene*

SCENE:
A frame sequence $f_i, f_{i+1}, \ldots, f_j$ is called a scene, iff for all k, the frames f_k and f_{k+1}, with $i \leq k < j$, have been recorded in sequence by the same camera without pausing the recording process.

Examples for scenes in this sense are all recordings that have not been cut or combined with other recordings, while scenes in movies are usually understood as the parts of the action in a single location, cut together from several captures (of different angles and distances).

MICRO NAVIGATION:
A navigation from a frame f_s to a frame f_e is classified as micro navigation, also called in-scene navigation, iff there are no frames f_i, f_j with $s < i, j < e$ or $s > i, j > e$, where f_i is part of another scene than f_j.

Definition: *Micro navigation*

MACRO NAVIGATION:
A navigation is classified as macro navigation, iff it is not classified as micro navigation.

Definition: *Macro navigation*

Both navigation concepts can be found in modern video playback devices. There are usually timelines and chapters, working as follows.

Modern devices offer micro and macro navigation

Timeline A widely used interface component that has not existed in traditional devices is the timeline, allowing the user to jump to arbitrary positions in the video; thus, enabling both micro navigation and macro navigation.

Timeline visualises and controls the current position

The timeline provides a representation of the current video position relative to the length of the clip. The start of the timeline stands for the start of the clip and the end of the timeline for the end. It can be used to either jump to another position in the video by clicking at a certain position, or to scrub at differents speeds and directions through the stream by dragging the slider to the left or right side, rewinding or fast-forwarding, respectively.

Timelines are used in almost all video players for computers, including Apple's QuickTime Player (figure 1.2), Microsoft's Windows Media Player and YouTube's embedded Flash player.

Chapters Especially with DVDs and other commercially distributed video material, video producers have the possibility to divide movies into chapters, which can be accessed independently. Usually, there are two buttons in the interface to jump from one chapter to the next. These chapters are predefined sets of scenes and cannot be changed by users through video players. Thus, they serve for macro navigation only.

Chapters divide movies into fixed parts

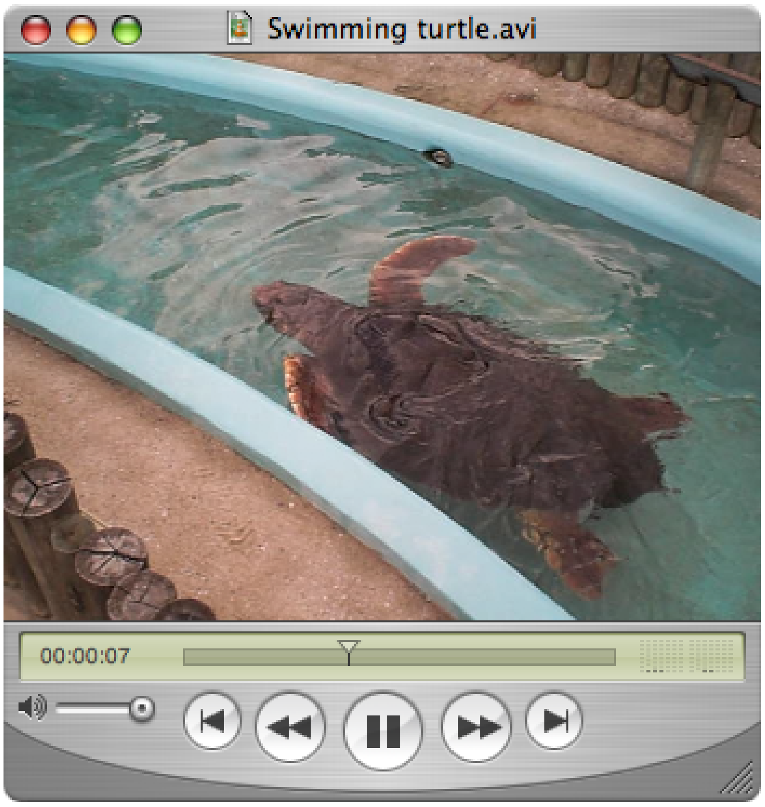

Figure 1.2: Apple QuickTime Player 7.4 resembling the interface of traditional video devices with a start/pause button, forward, rewind and providing modern components like chapter buttons for DVD playback and a timeline.

1.2.2 Micro navigation using the timeline

We concentrate on micro navigation

In this thesis, we will not take a deeper look at chapters and macro navigation but concentrate on improving in-scene micro navigation. We will see that the timeline, in conjunction with fast-forward and rewind buttons, has certainly improved the user's situation but still causes the following problems and disadvantages in the context of micro navigation.

Spatial separation of content and controls Users lose a lot of time and their *locus of attention* (Raskin [2000]) because they have to constantly switch between the spatially separated video view and playback controls. This introduces a high *degree of indirection* in the sense of [Beaudouin-Lafon, 2000].

Imprecise positioning control buttons As illustrated in the example of John and his son, it is hard to push the pause button at the right time.

Modality of timeline scrolling speed For videos of different lengths, the same subpart of the timeline corresponds to a different number of video frames making it hard to estimate the right position.

No natural mapping between timeline and video motion Dragging the timeline slider from left to right scrolls the video in forward (time) direction while the video content, i.e., the objects shown in the video, move in various directions and speeds.

In chapter 3—"Direct manipulation for video navigation" we will analyse these problems in more detail.

1.3 DRAGON

The system that we are going to evaluate in this thesis, called DRAGON (**DRAG**able **O**bject **N**avigation), presented at CHI 2008 [Karrer et al., 2008], is an approach to get rid of the described problems by introducing a direct manipulation technique.

DRAGON combines playback and control in one widget using direct manipulation

The DRAGON system combines video content and playback control in a single widget. The user is able to select an object directly inside of the video screen and drag it to another place. When an user selects an object, DRAGON calculates the corresponding trajectory, which consists of vertices of the form (x, y, t), where (x, y) are the Cartesian coordinates of the object at frame number t. As long as the mouse button is not released and the mouse is being moved, DRAGON calculates a weighted distance between (x_m, y_m, t_c) and (x_v, y_v, t_v) for all vertices v of the object's trajectory, where (x_m, y_m) and t_c are the current mouse cursor position and the current frame number, respectively.

The frame referenced by the vertex with the minimal distance is chosen as the next frame for playback (see figure 1.3 for illustration). Thereby, DRAGON creates the illusion that the user is able to move objects along their trajectories inside the video material (figure 1.4), providing a system with high *directness*, i.e., users navigate with *fewer cognitive resources* [Hutchins et al., 1985].

Definition: *Directness*

DIRECTNESS:
The feeling of directness is inversely proportional to the amount of cognitive effort it takes to manipulate and evaluate a system and, moreover, that cognitive effort is a direct result of the gulfs of execution and evaluation.

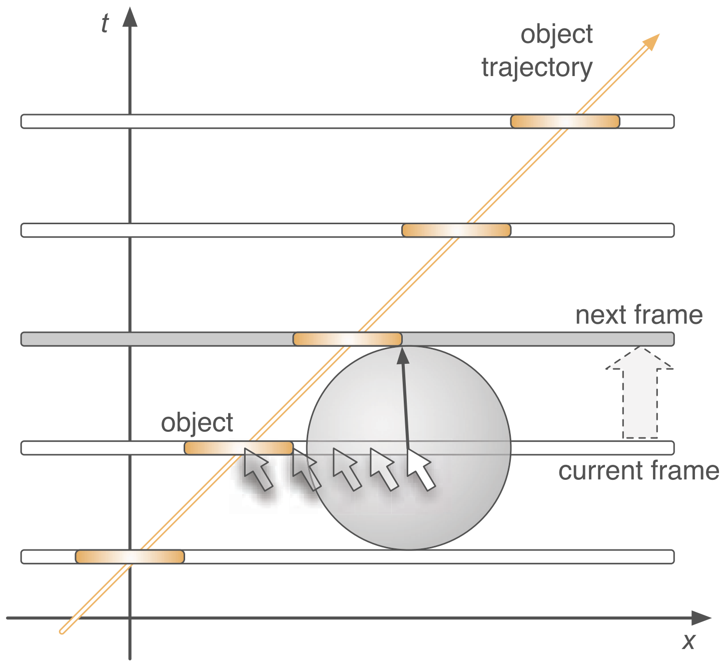

Figure 1.3: [Karrer et al., 2008]: Top view of a stack of frames (i.e., the y-axis is pointing out of the picture plane towards the reader). When the user clicks on the object and moves the mouse to the right, the video is scrolled to the frame where the (x, y, t) distance between mouse pointer and object is minimal. This distance is measured in both space and time, represented in the diagram by the shaded sphere, to avoid unwanted jumps along the video timeline.

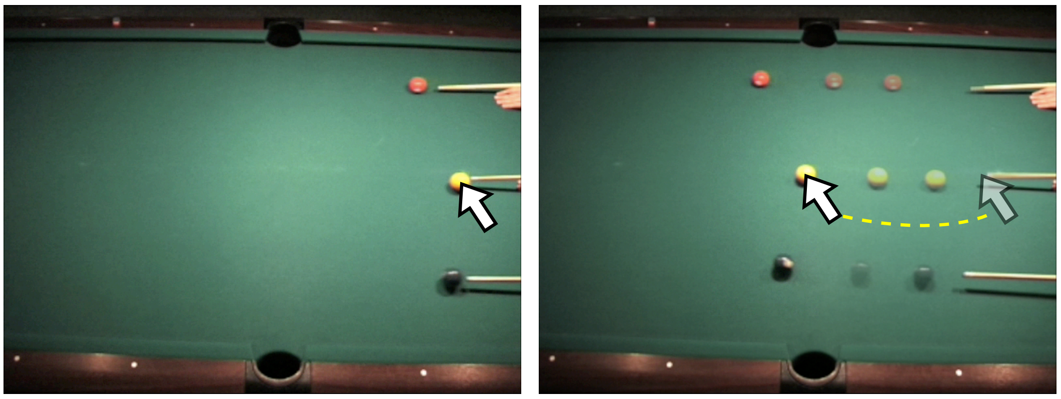

Figure 1.4: Example for DRAGON's direct manipulation technique. The user drags the yellow ball from the position shown in the left image to the one of the right image. The video stream is synchronized, so that the distance between mouse cursor and ball is minimal all the time. This results in the ball following the mouse movement. The transparent overlays of the balls' intermediate positions have been added for illustration purposes only.

Direct manipulation facilitates micro navigation

Remember the story of John and the buzzer beater in the basketball example. With DRAGON, the task of moving to the position where the hand leaves the ball can be accomplished much easier[3]. John can use the traditional controls to, more or less, find the interesting point in time and perform the fine-grained navigation using direct manipulation.

Direct manipulation usually does not facilitate macro navigation

Observe that for the navigation through direct manipulation the ball has to be visible, as an occluded object can neither be selected nor dragged. Therefore, DRAGON usually does not improve macro navigation since scene changes occur frequently. Even when an object is initially visible, scene changes introduce two major difficulties. First, dragging objects from one point to another during a scene change is very hard because of algorithmic limitations. The optical flow implementation used in DRAGON computes the optical flow between two subsequent frames. During a scene change, the algorithm will compute useless[4] flow fields and will not be able to keep on tracking the object.

[3]Of course, today's video cameras usually do not offer a touch-screen or allow the use of a mouse. However, considering the current development of (multi-)touch displays this might change quickly.

[4]The flow fields are useless for DRAGON because they do not represent object motion.

Second, even if it was possible to keep on tracking the selected object, dragging will often not make much sense because the trajectory can lose continuity. Therefore, this thesis focuses on in-scene navigation as used in short video clips, video cut applications and other areas like replays for live sports broadcasting.

Additional material can be found on project website

For the interested reader, additional material is available at the DRAGON project website.[5] This material includes a binary for Mac OS X 10.5 or later, a preprocessed sample video, the CHI 2008 paper by Karrer et al. [2008], a demonstration video. The page will be updated with current research. Part of this current research is a Diploma thesis on *DragonEye: Fast Object Tracking and Camera Motion Estimation* by Wittenhagen [2008].

1.4 Structure of this thesis

After this short introduction, the subsequent chapters are structured as follows.

2—"Related work" In the next chapter, we give a brief overview on research projects on video navigation, annotation, and retrieval. Moreover, we present a classification space to group similar projects.

3—"Direct manipulation for video navigation" The third chapter features an analysis of problems present in current timeline-based systems. Based on these considerations, we motivate the introduction of DRAGON's direct manipulation technique, and describe its implementation, benefits, and drawbacks. Moreover, we present an online survey to determine the demands of users.

4—"Object trajectories and visualisation" This chapter covers an examination of different types of object trajectories, of different levels of interaction difficulty, a presentation of approaches to user guidance, and techniques to represent object speeds.

[5]http://hci.informatik.rwth-aachen.de/dragon

5—"Motion gaps" At this point, we discuss the problems arising when objects pause or move very slowly for a certain amount of time, and we present possible solutions validated through user studies.

6—"Pendulum-like trajectories" The last problem we cover are trajectories of objects that reach and leave positions on similar trajectories, introducing ambiguity. As in the previous chapter, we evaluate different approaches in a user study.

7—"Summary and future work" At the end of this thesis, we summarise the approaches presented before, identify their contributions, and give an outlook on future work.

Chapter 2

Related work

"You do not need to know everything,
but you should know where you can find it."

—My math teacher

As described in the previous chapter, research groups have published approaches to improving and facilitating the use of video material. There are two main areas—video navigation and video retrieval. We now describe some of these works, those directly related to DRAGON in more detail than the others.

2.1 Video navigation

The following systems focus on how to improve video navigation, either by implementing direct manipulation techniques or by introducing extended timeline features.

2.1.1 *DimP*

DimP follows same idea as DRAGON

DimP [Dragicevic et al., 2008] (for **Di**rect **m**anipulation **P**layer) is a video player that uses direct manipulation for video navigation. The idea behind the system is basically the same as in DRAGON. However, the tracking algorithm is different (based on SIFT feature extraction and optical flow based feature-matching), and the interaction technique offers some features that, in DRAGON, are either not present or have been designed differently. We compare these differences in section 2.3.

Direct manipulation for video navigation is different

The *DimP* research group points out that, for several reasons, direct manipulation for video navigation is different from direct manipulation as used in traditional GUIs. First, there is no clear spatial segmentation of the interaction content, and, furthermore, this content can suffer arbitrary deformations during a dragging operation. Second, the trajectories are predefined, which constrains the object motion. Third, dragging an object is not independent of the other content, as the rest of the video moves simultaneously.

We can distinguish different classes of direct manipulation for video navigation

In addition, three different classes of direct manipulation are distinguished as shown in figure 2.1. *Curvelinear dragging* refers to dragging a point constrained to a 2D curve using an usual pointing device. *Flow dragging* generalizes this class to direct manipulation of *arbitrary motions having only one degree of freedom*, where "arbitrary" includes any visual transformation and "one degree of freedom" refers to the possibility of mapping the whole motion to a scalar variable, such as time (e.g., the deformation of a bouncing ball). The last class is *relative flow dragging*, where background motion is subtracted from object trajectories. This is used for the interaction with objects in situations where background motion is involved, because users perceive the relative motion of objects to their background rather than the absolute motion of the corresponding pixels inside the video screen.[1]

Interaction

DimP has various visualisation features

For preview, the cursor is changed from an arrow to a hand when the area under the cursor contains a significant amount of motion. After clicking on such a point in

[1]This phenomenon is also known as *Duncker illusion* [Zivotofsky, 2004].

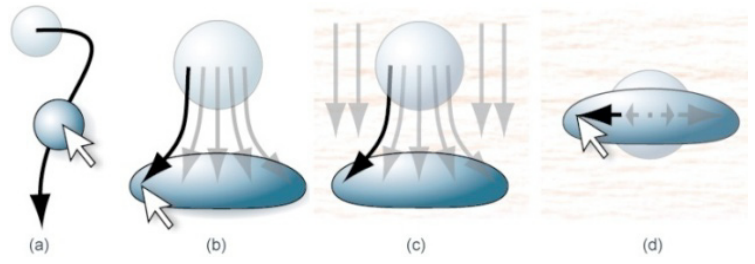

Figure 2.1: [Dragicevic et al., 2008] Three classes of direct manipulation for video navigation. (a) *curvelinear dragging* for 2D curves, (b) *flow dragging* for a set of trajectories, and (c)/(d) *relative flow dragging* for situations involving background motion.

the video screen, *DimP* visualizes the point's estimated trajectory. Video navigation is then invoked by dragging the point along its trajectory. When the distance between the mouse cursor and the dragged point, represented by a red cursor on the trajectory, gets larger, the trajectory is emphasized to show that the interaction is constrained. If a user drags the mouse cursor far away from the trajectory for more than 2 seconds, interaction terminates.

DimP recognises background motion

Interaction involving background motion is performed by dragging the object along its relative motion path instead of its absolute motion part (figure 2.2). *DimP* recognises the upward camera pan and transforms the absolute trajectory (figure 2.2a) into a relative trajectory (figure 2.2b) by subtracting the background motion from the absolute motion. To maintain the video context, the system creates a stitched image of the corresponding frames during dragging operations (figure 2.2c).

Mapping between cursor position and trajectory

DimP uses 3D approach for calculation of next frame

Dragicevic et al. [2008] define some requirements for curvelinear dragging. Users should be able to perform fine-grained as well as coarse dragging. Furthermore, it should be possible to follow loops and cusps. Last, the spatial distance between the mouse cursor and the selected point on

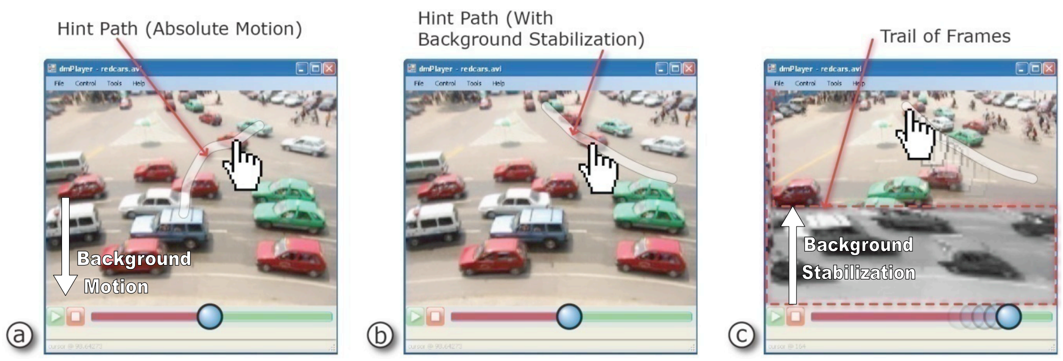
Hint Path (Absolute Motion)
Hint Path (With Background Stabilization)
Trail of Frames
Background Motion
Background Stabilization
a
b
c

Figure 2.2: Background stabilisation as implemented in *DimP* (Dragicevic et al. [2008]). (a) Absolute motion of the selected object (red car). (b) Hint path (trajectory) with background stabilisation. The system subtracts the background from the absolute motion. (c) Trail of frames during a dragging operation to maintain the background stable. Old frames are greyed out. When interaction terminates, the current frame is moved to the centre of the screen.

the trajectory should always be minimal. To achieve these goals, a 3D approach is used to calculate the next frame to be shown when performing a dragging operation. Therefore, every trajectory vertex consists of three coordinates (x, y, z), where x and y are the Cartesian coordinates and z is the arc-length distance from the curve's origin. Furthermore, every point is tagged with the corresponding frame. For every point $C = (C_x, C_y)$ of the curve, *DimP* calculates the 3D distance D to the pointer position (p_x, p_y) using

$$D = \sqrt{(p_x - C_x)^2 + (p_y - C_y)^2 + (k \cdot \overline{C_aC})^2 + k_D}$$

where $\overline{C_aC}$ is the arc-length distance between C_a and C. The point minimising the equation is chosen as the next C_a. $k_D > 0$ is added when $\overline{C_aC}$ and $\overline{C_{a^{t-1}}C_a}$ have different signs. This preserves directional continuity in cusps.

Drawbacks of the evaluation function are related to possible leaps and long trajectories

There are some drawbacks. First, jumps can occur, when users drag the cursor far away from the trajectory. For smaller k these jumps occur more frequently than for higher values. The group suggests to smooth large jumps visually using animations. Furthermore, tracking objects around long trajectories can get hard as cluttered trajectories may occur. A suggested solution is clipping trajectories to predefined arc-lengths[2].

[2]In chapter 6—"Pendulum-like trajectories", we propose a similar

User study

In a study, users were asked to perform video navigation tasks with a timeline slider and the direct manipulation techniques on two different videos. The dependent variables were *trial time* and *error*, where the trial time is the time between the first mouse-down and the last mouse-up events of a trial, and the error is calculated as the distance between the target frame and reached frame divided by the total frame count. Two videos were used. For each of them, users had to perform the same task using relative flow dragging and the timeline slider.

Relative flow dragging showed to be at least 250% faster than the timeline slider, while there was no significant difference between the error rates. Furthermore, users found that the tasks could be performed more easily and faster using relative flow dragging.

2.1.2 Schematic Storyboarding

Users can create schematic storyboards of videos

The system presented in [Goldman et al., 2006] enables users to create schematic storyboards—usually used in the production of movies—over existing video material. In these storyboards, a variety of arrows is used to express changes in the video. Furthermore, the system creates panoramic images combined from the single frames, when background motion is involved. This visualization in conjunction with motion arrows allows an easier and faster comprehension of a scene than single frames only. Figure 2.3 shows an example for the motion of a child. The motion arrows are not created automatically but require some user interaction. Users need to select the key points in the different frames of a video.

Objects and background can be dragged

Based on this data, a direct manipulation technique was implemented. Users can select objects and drag them along the motion arrows, scrolling the video to the corresponding positions in time. Moreover, users can drag the background as well. A click at a point, that does not belong to a moving

approach by dividing trajectories into disjunct subtrajectories.

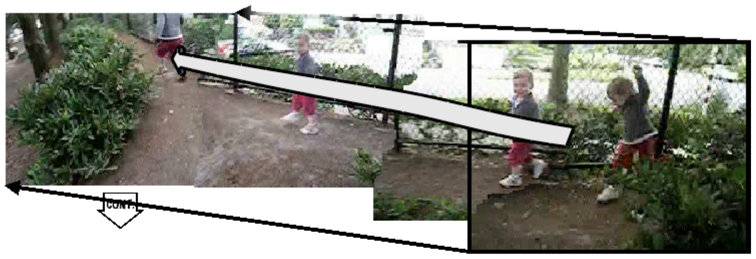

Figure 2.3: Schematic storyboarding by [Goldman et al., 2006]. A motion arrow as known from storyboards on a stitched panorama build from single frames of the video. Users can select the child and drag it through the scene.

object retrieves the frame in which the selected point is as close as possible to the centre of the frame.

2.1.3 Video Object Annotation, Navigation, and Composition

Goldman et al. [2008] present a system based on motion data and direct manipulation to perform annotation, navigation, and composition tasks in videos.

Particle tracking and grouping recognises object motion

The system extracts the necessary motion data in a pre-processing step where *particle tracking* outputs a cloud of particles that represent the motion of visible points. In a second step, an algorithm segments these particles using an affine grouping procedure. Although the tracking is not perfect, it is sufficient for the implemented interaction techniques. The user interface uses the segmentation information for various purposes.

Two modes for object selection are available

The selection of an object for example can be done in two ways. First, a simple mouse click on an object searches for the closest group of points. Second, users can draw a selection area to select points from multiple groups. This enables users to overcome eventual over-segmentation by the grouping algorithm. Furthermore, partial occlusions can be handled because not the points themselves are selected but the corresponding group. As long as this group is at least partially visible, the system keeps tracking the region.

Navigation

Discontinuous leaps are explicitly allowed

The system provides video navigation by direct manipulation. However, in contrast to *DimP* and DRAGON, the system uses neither temporal information nor arc-lenght distances to avoid discontinuous leaps in time because, for some tasks, these leaps may be desirable. Moreover, when it is not possible to continue tracking the selected region, the next closest region is chosen which allows for longer trajectories.

Implementation is as follows (figure 2.4): First, a user clicks at a location x_0 on frame f_0. Then, the system calculates the closest track i^* (motion path of a particle), and saves the offset $d = x_0 - x_{i^*}(t_0)$ to this track. When dragging the mouse to position x_l, the next temporal position is selected using:

$$t^* = \text{argmin}_{\{t \in T(i^*)\}} ||x_l + d - x_{i^*}(t)||$$

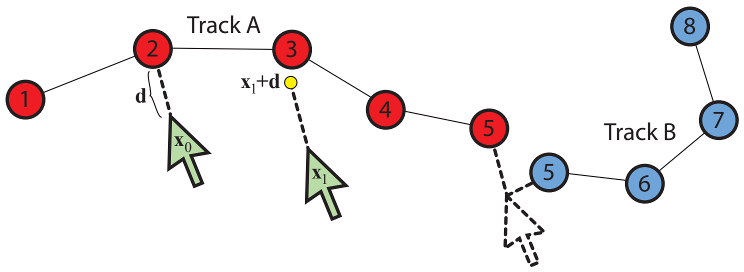

Figure 2.4: [Goldman et al., 2008]. A mouse click at position x_0 on frame 2 selects track A. When the mouse moves to position x_l, the frame 3 is closer to the offset mouse position $x_l + d$, so the video is scrubbed to frame 3. Track A ends at frame 5, but the virtual slider is extended to later frames using the next closest track (track B).

Users can constrain object motion

Additionally, the system allows for spatially fixing points in the video and, thus, constraining the number of accessible frames. For instance, in figures 2.5 (d), the position of the

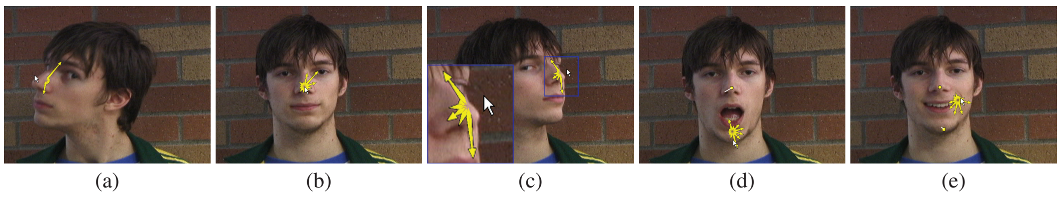

Figure 2.5: [Goldman et al., 2008]. The scrubbing interface is used to interactively manipulate a video of a moving head by dragging it to the left and right (a-c). Nearby frames are indicated by longer starburst arms. At the extremes of motion (c), some arms of the starburst disappear. Additional constraints are applied to open the mouth (d), or keep the mouth closed and smile (e).

nose is fixed. Hence, it is not possible to scrub to the positions of figures 2.5 (a), 2.5 (b), and 2.5 (c). In 2.5 (e), nose and chin are fixed such that direct manipulation is constrained to the frames where the head remains in the same position.

Starburst widget visualises possible dragging directions

The figures 2.5 (a)-(e) show another feature of the system. The so-called *starburst* widget (see (c) for zoomed-in view) visualises the range of motion. Short spikes represent the dragging directions to reach temporally distant positions; long spikes represent the motion in the clos temporal neighbourhood of the current frame.

Inertia is used to overcome full occlusions

Though the grouping information is sufficient to handle partial occlusions, it is not possible to overcome full occlusions, yet. Therefore, the system includes an inertia feature that provides a looser long-distance scrubbing technique. During dragging, it calculates the scrubbing velocity in frames per second, and, when the mouse button is released, this velocity decays over the next few seconds. However, for trajectories folding back or having a spiral form, a linear mouse motion produces a sequence of discontinuous frames. In such cases, *temporal inertia* does not work. Instead, the system uses *spatial inertia*. A heuristic is used to distinguish when to use which type of inertia.

Annotation

Various annotation styles are supported

Unlike existing applications where only still images can be annotated, it is possible to attach information to mov-

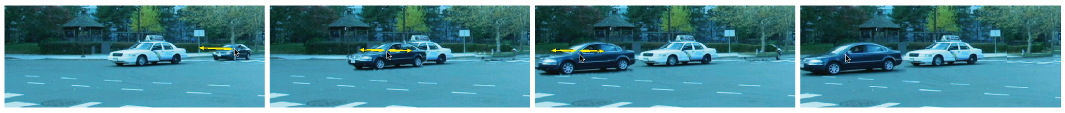

Figure 2.6: [Goldman et al., 2008]. Drag-and-drop composition of a still from video. The black car is to the right of the white car on all frames of the input video. From left to right, the black car is dragged from its location on frame 1 to a new location on frame 66. The first three panels show the appearance during the drag interaction, and the fourth panel shows the final graph cut composite.

ing objects—graffitis, scribbles, word balloons, video hyperlinks, marking occlusions, and path arrows. Graffitis are annotations whose location is specified by defining four or more anchor points. These points follow the underlying particle motion, and inherit their perspective deformation. Scribbles are simple textual or sketched annotations that move with the associated object. Word balloons, either speech or thought balloons, similar to those in comics, stay at a fixed position with an indicator following the object they have been attached to. Hyperlinks extend moving objects with links to websites and external resources. Marking occlusions allow users to mark a region that changes its colour when the associated object gets occluded or when it appears again, respectively. Last, path arrows show the motion path of a selected object.

Composition

Video-to-still composition works with drag-and-drop

The video-to-still composition feature included in the system allows to seamlessly combine multiple frames from a video clip into a single image. A drag-and-drop approach allows for dragging an object to any position it passes in the video while the rest of the video stays fixed. By using background stabilisation, camera pans can be handled. When the users drops the object, i.e., releases the mouse button, the area occupied by the object in the anchor frame is being substituted with the contents of the corresponding area of the current frame, in order to remove its original appearance. Although this area might have changed in the meantime, the approach usually produces reasonable results (figure 2.6).

Key point selection for schematic storyboards is now automatic

Moreover, the system includes the *schematic storyboards* from Goldman et al. [2006]. However, the key point selection is now simplified. Users navigate through the video using either direct manipulation or the timeline slider. A hot key marks frames as storyboard keyframes. No further interaction for key frame selection is necessary.

2.1.4 *Trailblazing*

Trailblazing offers visualisation of floor plans and direct manipulation

A group at FX Palo Alto Laboratory presented *Trailblazing* [Kimber et al., 2-5 July 2007], a software similar to DRAGON for surveillance systems. Users are enabled to control the video playback position by selecting objects in the video and dragging them to another position. The system provides two different modes. First, it supports single camera video analysis. Second, the systems allows for registering a floor plan and camera positions to enable multiple camera analysis. In this case, the system additionally shows trajectories and positions of the persons in the floor plan, which is synchronised to the video view such that users are enabled to drag objects in the video or on the floor plan to review the video clip.

Object interaction is classified by appear, disappear, continue, merge, and split

For single camera tracking, the group has followed previous work from Cucchiara et al. [2004] to classify an object's interaction with other objects. The five different classifications are *appear*, *disappear*, *continue*, *merge*, and *split*. While appear, disappear and continue can be easily handled, merge and split are more complex. To be able to determine the candidate objects when merged regions are split during the video, merges and splits are tracked and saved in a database.

The UI offers direct manipulation in all video views

The user interface of Trailblazing, shown in figure 2.7, consists of several video displays in different resolutions and of different locations in the observed area, a floor plan or map showing the camera and object positions, and a timeline enabling the user to control all the videos synchronously. Clicking on an object, independently of the video display, selects the object and shows its trajectory. In case of clicking with no object under the mouse cursor, candidate objects are being highlighted and can be chosen by cycling through them using multiple clicks. Once an object is selected, user can scrub through the video by dragging it along its trajectory. The group has observed that, depending on the situation, different portions of the object trails should be shown instead of showing the full trajectory. Furthermore, objects can be dragged from one video display to another, interpreted as a command to move to the video position where the object appears in the other video display.

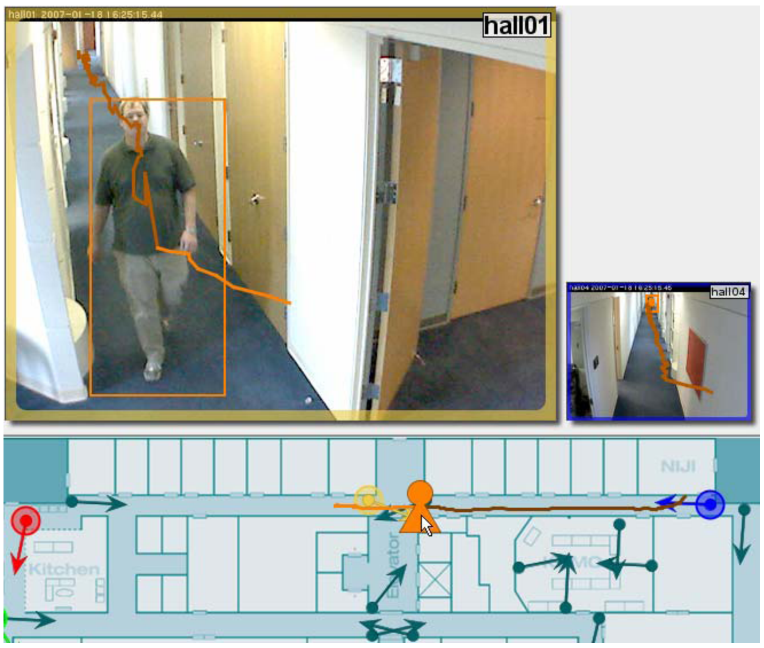

Figure 2.7: *Trailblazing* (by Kimber et al. [2-5 July 2007]) Viewer showing trails in video and floor plan view. The user may scrub video by dragging a person along their path in the video window or on the floor plan.

The system uses a weighted evaluation function for dragging

While dragging, problems can arise. For instance, if an object crosses its own trajectory, the system has to decide whether the video has to be scrolled to the first time, the object was in that place, or to the second time. To solve this problem, the group uses an evaluation function taking into account weighted distances of time and location and a cost for changing the temporal direction.

Trailblazing provides advantages to slider-based video scrubbing

The developers of *Trailblazing* conclude, that it is a system providing several advantages to slider-based video scrubbing. First, direct object manipulation is more natural than moving a slider as objects move at different speeds and in different directions. Second, for long videos, sliders do not offer a sufficiently fine control, and third, users easily see the start and end of the interval in which the object is visible. Finally, the system can be used for means of retrieval, e.g., to check when a person was in a particular position or to find all people nearby.

2.1.5 *LEAN*

LEAN provides extensions for slider-based systems

The *LEAN* system proposed by Ramos and Balakrishnan [2003] does not allow direct manipulation of the video content, but tries to improve interaction with sliders by introducing the *Twist Lens* (figure 2.8) and the *PVSlider* (figure 2.9).

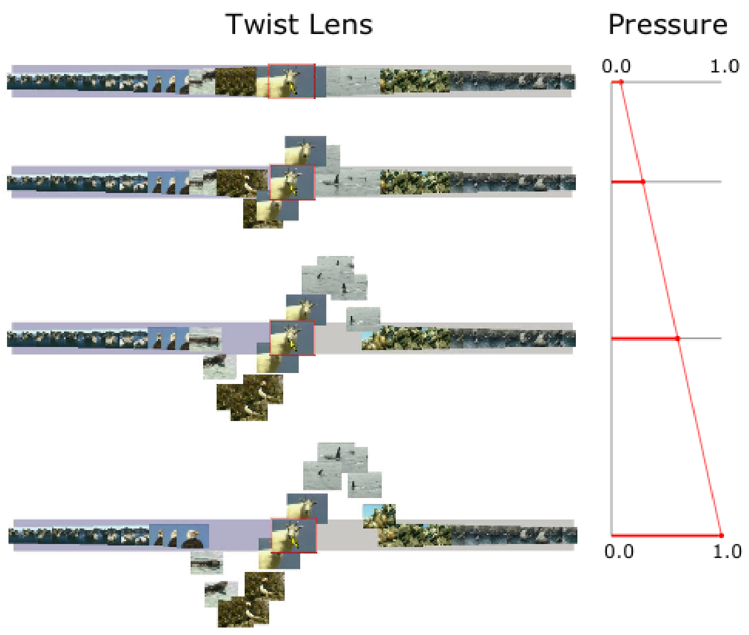

Figure 2.8: *Twist Lens* (by Ramos and Balakrishnan [2003]). The figure shows, from top to bottom, how the amplitude of the lens changes according to the pen's pressure, which is displayed on the right.

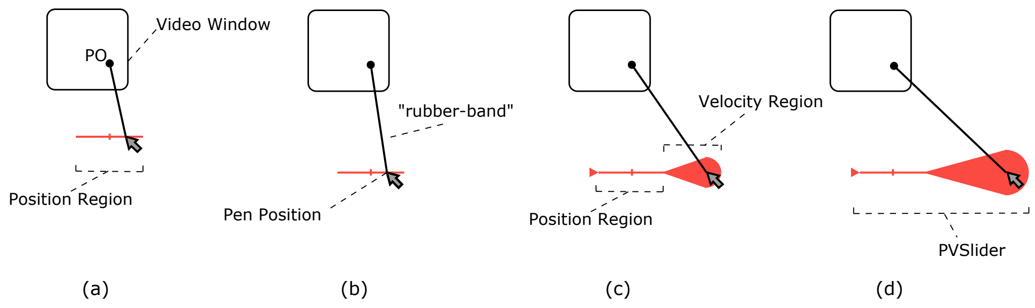

Figure 2.9: *PVSlider* (by Ramos and Balakrishnan [2003]). (a) Position region for frame-accurate scrolling over a range around the current frame. (b) Increasing the size of the position region by increasing the distance to the point of origin (PO). (c) Velocity region for fast-forward scrolling. (d) Increasing the velocity with visual feedback.

Twist Lens

Twist Lens increases timeline resolution dynamically

LEAN dynamically divides the video stream into single preview frames. Instead of providing a timeline slider as described in section 1.2.1, a *Twist Lens* slider consisting of these preview images is shown. When the users clicks on an image in the *Twist Lens* slider, the video jumps to the corresponding frame. By selecting a range of frames, the user creates a new slider with a higher resolution, showing preview images of the frames inside the selected range. A graphics tablet with a stylus serves as the input device, enabling users to increase the amplitude of the *Twist Lens* by using more pressure (with stylus), i.e., the slider opens in form of a spiral showing more preview frames and, thus, increasing the precision.

PVSlider

PVSlider allows for frame-accurate and macro navigation

The system uses no forward or rewind buttons, but enables the user to dynamically create *PVSliders* by clicking at arbitrary points of the screen. These sliders have two regions. First, the *position region* represent a range of frames around the current frame. Horizontal dragging of the slider inside this region accesses the corresponding frames. Vertical dragging changes the size of the mapped interval (higher distance increases the range). Second, the *velocity region*, which starts at the end points of the position region, allows for accessing the rest of the frames. In this region, dragging to the right performs fast-forward while dragging to the left rewinds the video. Again, the distance to the slider origin has an impact on the scrolling speed. The figures 2.9 (a) and 2.9 (b) show scrubbing in the position region, and 2.9 (c) and 2.9 (d) illustrate the use of the velocity region.

2.2 Automatic video indexing and retrieval

Research groups have proposed techniques for automatic video indexing and video retrieval by queries. These systems analyse the video content and allow users to perform queries by asking questions or by sketching the content. Though these systems do not offer techniques for video navigation, they show that the use of motion based data as input can facilitate video retrieval.

2.2.1 AVI

AVI analyses video content and allows for asking questions to retrieve videos

The *AVI* (Automatic Video Indexing) system proposed by Courtney [1996] provides a tool to perform analyses of video content, especially for surveillance purposes. By extracting a *motion graph* from the video, i.e., motion analysis and object tracking, it is possible to classify several events like appearance/disappearance (for example due to occlusion), deposit/removal (of objects), entrance/exit (into and out of the scene) and motion/rest (of objects). Figure 2.10 shows a sample analysis where a person removes an object from the scene. Queries are made by selecting a person or an object and giving commands like *"show me all objects that this person removed from the scene"*. The system then retrieves the corresponding video clips and highlights the events that match the query.

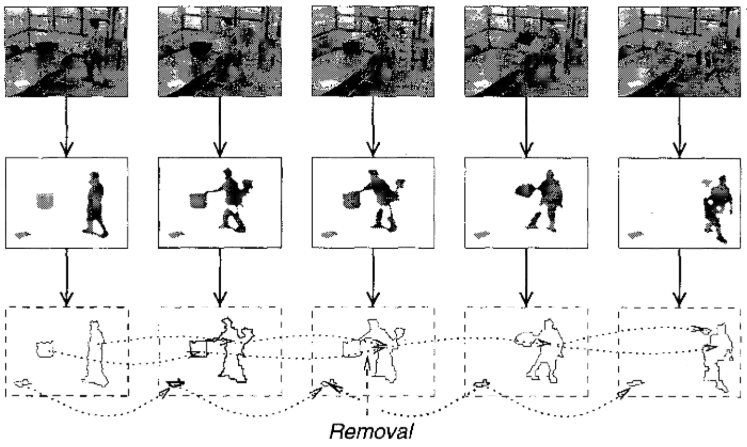

Figure 2.10: *AVI* (by Courtney [1996]. Relation between video data, motion segmentation information, and a symbolic motion graph.

2.2.2 A Motion-Flow-Based Fast Video Retrieval System

This system introduces *query-by-sketch*

Su et al. [2005] use motion vectors from MPEG videos to perform motion analyses and to fill a database of *motion-flows* for fast video retrieval. Instead of using events and queries in text form as in section 2.2.1—"AVI", a method called *query-by-sketch* is presented, working as follows. Users can draw simple sketches of what they are searching for. These sketches are then matched against the information in the database. For instance, sketching a line from left to right searches for videos with objects moving from the left to the right. Figure 2.11 shows an example query and the results.

2.2.3 VideoQ

VideoQ extends query-by-sketch by temporal information

VideoQ, proposed by Chang et al. [1997], is a system similar to one presented in the previous section. Again, videos again are retrieved by sketching the content. However, VideoQ allows users to use temporal information as well. Arrival and departure of objects can be modelled, enabling

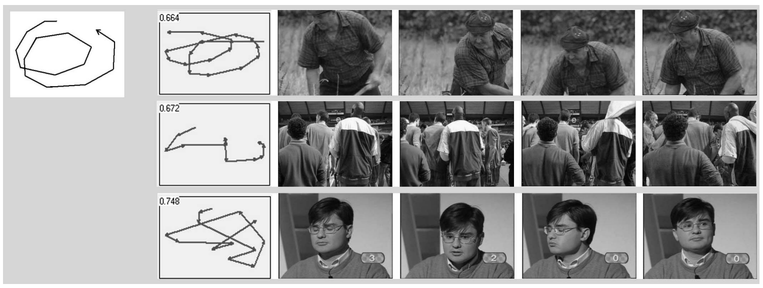

Figure 2.11: Example of a *query-by-sketch* result as proposed by Su et al. [2005].

users to search for more complex scenarios. Even changes in the size of objects are representable.

2.3 Discussion

Direct manipulation systems focus on micro navigation

DimP, *Trailblazing*, *Schematic Storyboarding*, and the system by Goldman et al. [2008] use direct manipulation to scrub through videos, and they focus on micro navigation. The first two systems use automatic motion extraction and trajectory creation. Usually, these trajectories are very close to the real object motion. The storyboarding approach requires user interaction to create the trajectories and does not offer such a fine-grained and exact navigation. This system serves rather for providing an overview of a video scene. Background motion is handled by *DimP*, *Schematic Storyboarding*, and Goldman et al. [2008]. *Trailblazing* does not need background motion recognition because it bases on fixed camera positions and floor plans. While *DimP* uses background stabilisation and shifts frames in and out of the video screen, *Schematic Storyboarding* creates panoramic images and, thereby, shows the full scene at once. At this point, we want to mention an improved version of DRAGON by Wittenhagen [2008], called DRAGONEYE, that supports camera motion estimation. However, as this version has been implemented in parallel to this work, we have not evaluated the features in this thesis.

LEAN focuses on micro and macro navigation

LEAN introduces extensions to slider-based systems and improves micro navigation as well as macro navigation. Users can scroll through the video material at different speeds by using both frame-exact and coarse navigation.

Query-by-sketch matches input motion to video content motion

Query-by-sketch has some similarities with direct manipulation video navigation systems because mouse movements are matched against motion in the video content. Furthermore, the MPEG vectors used in the extraction of motion can help to improve the performance of the DRAGON prototype.

We remember the systems from the introduction

Though not mentioned in this chapter, we remember that other techniques for scrolling through a video are available. In 1—"Introduction" we have seen the basic features provided by commercial systems—forward/rewind buttons and timelines.

2.3.1 Classification space

We propose to classify the systems

As we have seen, the presented systems differ in various aspects. We propose to classify the systems by the following conditions.

Micro navigation Are all frames easily accessible?

Macro navigation Can the user jump from one position of the video stream to another?

Based on spatial information Is the scrubbing process based on spatial information?

Based on temporal information Is the scrubbing process based on temporal information?

Background motion Does the system handle background motion (important for navigation through direct manipulation only)?

Based on this classification we define a classification space as shown in figure 2.12. Observe, that we do not claim this classification to space be complete. The space rather serves as a survey of the closely related systems.

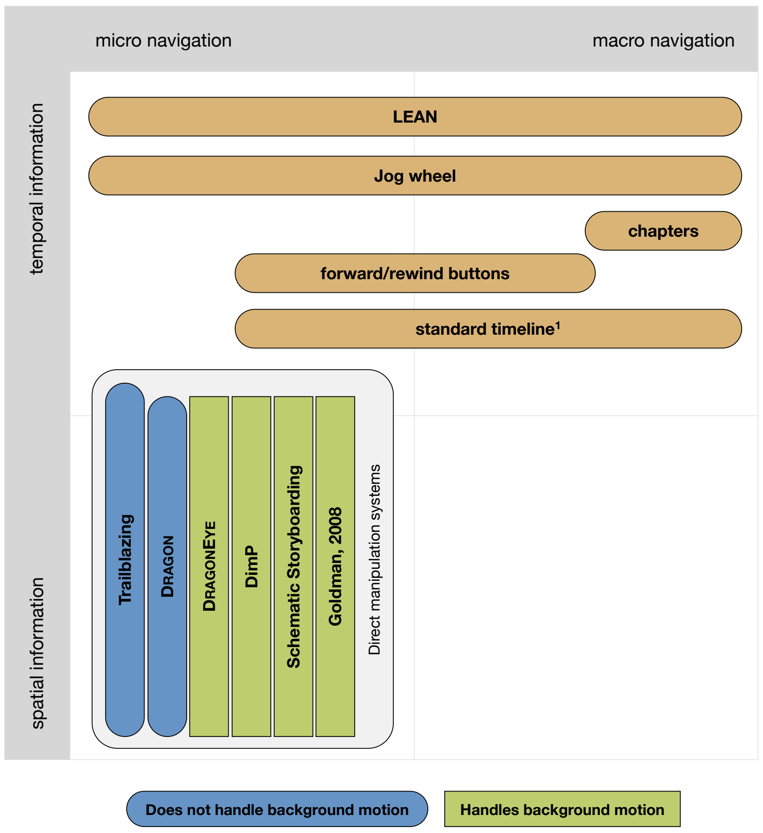

Figure 2.12: Classification space for video navigation ([1]timeline resolution depends on video length). The size and position of horizontal boxes represents the suitability for micro and macro navigation, while the size and position of vertical boxes represent the use of spatial and temporal information.

Time based system range from limited to powerful

Though timeline-based systems[3] support both macro and micro navigation, the classification space uses a smaller box for these systems than for the *LEAN* system and jog wheels, because the timeline's scrolling resolution depends on the video length. For long videos, the timeline loses its micro navigation capabilities while jog wheels and *LEAN* allow fluid interaction for all videos lengths and a dynamic range of scrolling speeds. Furthermore, chapters and forward/rewind buttons have limited functionality and are visualised by smaller boxes.

[3]These systems include commercial video players like Apple QuickTime Player and Windows Media Player.

Direct manipulation video navigation makes less uses of time

Obviously, systems based on direct manipulation use temporal information to create the trajectories. Therefore, we have not positioned them entirely inside of the box micro navigation/spatial information. *Trailblazing* makes more extent use of temporal information as it presents object candidates when the user clicks on a point in background; therefore, it is assigned a larger box (*Trailblazing* performs a search in the video and shows the events tagged with a time stamp). We assign a different box to *DimP*, the system from [Goldman et al., 2008] (see section 2.1.3), DRAGONEYE, a version of DRAGON by Wittenhagen [2008] with camera motion estimation, and *Schematic Storyboarding* because these systems use background stabilisation techniques. Moreover, all systems are inside another box to group them and express that they do not differ regarding micro-navigation capabilities.

No system is available for macro navigation based on spatial information

We observe, that no system supports the combination of spatial information and macro navigation. This may be an inherent problem of macro navigation. The spatial information changes too much and can usually not be used to scroll through the video. We thought of positioning the video retrieval systems in this cell. However, as they are not used for video navigation, but for video retrieval, we have chosen to not include them in this classification space.

Chapter 3

Direct manipulation for video navigation

"Ignorance more frequently begets confidence than does knowledge: it is those who know little, and not those who know much, who so positively assert that this or that problem will never be solved by science."

—Charles Darwin

In this chapter, we analyse the disadvantages and problems that arise in timeline-based and VCR-like systems, and we describe how users can benefit from a direct manipulation technique for video navigation.

3.1 Disadvantages of current systems

As illustrated in the basketball example in 1.1—"Analogue video navigation", forward and rewind buttons have serious disadvantages; it is hard to push the button at the right time when the scrolling speed is high. Though the timeline is already an improvement, systems combining start, fast-forward and rewind buttons with the timeline still have disadvantageous properties, especially for micro navigation tasks.

We consider spatial separation, modality, and natural mappings

In our analysis, we first consider the spatial separation of content and controls, followed by a discussion about the modality of the timeline regarding scrolling speed. Last but not least, we observe that there is no natural mapping between a user input motion and the resulting object motion in a video. For illustration and better understanding, we relate our observations to the concept of the *seven stages of action*, introduced in *The Design of Everyday Things* [Norman, 2002].

3.1.1 The seven stages of action for video navigation

Every action can be split up into seven stages

Every action in computer interfaces[1] can be split up into two parts—execution and evaluation. To perform any task, one must formulate a goal, execute an action sequence based on this goal, perceive the changes in the world, and then evaluate whether the goal has been achieved or not. Thus, execution consists of forming the intention, specifying an action sequence, and the actual execution of this action sequence, while evaluation is divided into perceiving the state of the world, interpreting this perception, and evaluating the outcome. Together with the formulation of the goal itself, we consider *seven stages of action*. At different positions, problematic points can be identified—*gulfs of evaluation* and *gulfs of execution*. *Gulfs of evaluation* arise, when, for instance, the comparison of an action's outcome with a particular goal is hard, or when the evaluation of the state of the system is difficult or impossible. *Gulfs of execution* arise, when action sequences cannot be formulated, or when it is impossible to execute actions with the tools offered in the environment or world.

The action of using the timeline slider or the VCR-like controls to skip to a certain moment in a video can be split up according to these stages. For illustration, remember the buzzer beater example from chapter 1—"Introduction". This time, we change the situation a little. John is now at home and reviews the video material with his favourite media player. This media player provides the standard con-

[1]In fact, every action in the real world can be split up the same way.

trols for video navigation, i.e., a timeline, a start/pause button, and buttons for fast-forward and rewind. For our analysis, we focus on the timeline slider.

We observe seven stages for timeline navigation

Again, John's *goal* is to see whether the referee's decision had been correct or not. In other words, his *intention* is to review the moment of the buzzer. To achieve that, his *action sequence* consists of finding the position in time of his son shooting the ball and of seeing whether there is still time left. To *execute the action sequence*, he needs to grab the timeline slider and move it to the left or right, respectively, until he finds the exact position. He does this through interaction with the video playback interface—the *world*. Through this interface, John first *perceives* the frame currently shown, and then *interprets* the content by realising what is happening, i.e., by checking the time left and the ball's position. Finally, he *evaluates* whether he is seeing the desired frame, and he tries to make his decision based on this evaluation. It may happen that he has to repeat the action several times until he finds the correct position.

We have now split up John's action into seven stages. Figure 3.1 shows a graphical representation of the different stages for a general video navigation task. We have already marked two unbridged gulfs of execution and evaluation present in timeline-based systems. In the following sections, we give further explanations on these gulfs, their causes, and their consequences.

3.1.2 Spatial separation of content and controls

Spatial separation works well for static content

Timeline-based and button-based video players separate the video content and the video controls spatially from each other (remember the screenshot of Apple's QuickTime Player in figure 1.2). Therefore, to navigate through a scene, the user has to concentrate on two distinct parts of the interface. As Ramos and Balakrishnan [2003] point out, this separation works well with static and non-temporal content, where switching the view between a document and some controls has no influence on the position inside of the document; word processing is an example for this class of applications. However, for time-based media like audio

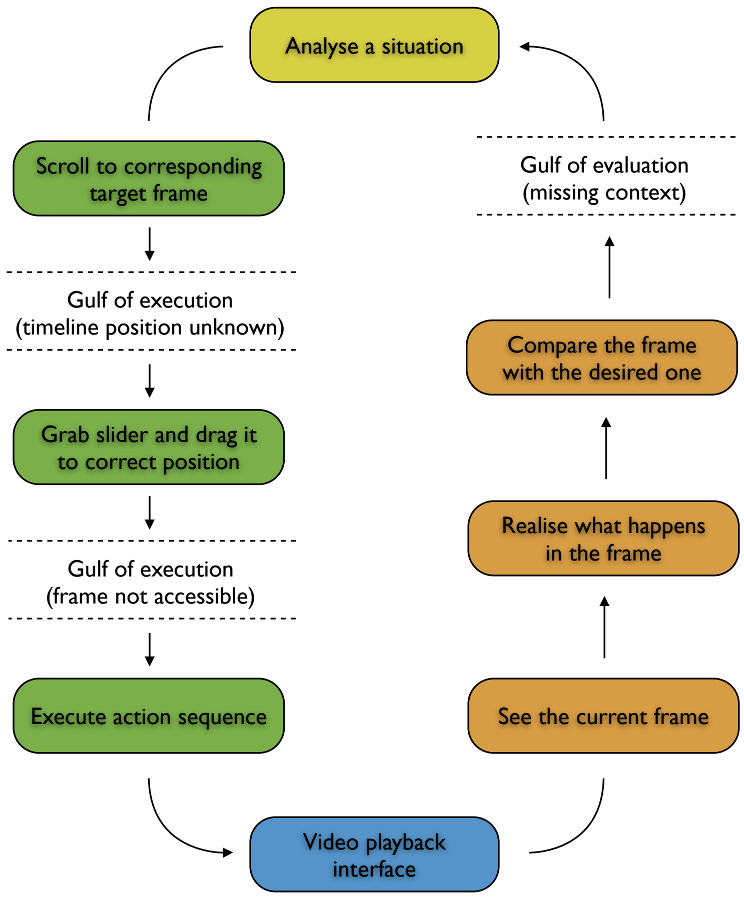
Analyse a situation
Scroll to corresponding target frame
Gulf of execution (timeline position unknown)
Grab slider and drag it to correct position
Gulf of execution (frame not accessible)
Execute action sequence
Video playback interface
See the current frame
Realise what happens in the frame
Compare the frame with the desired one
Gulf of evaluation (missing context)

Figure 3.1: The *seven stages of action* for timeline navigation. Gulfs of execution arise if the position of the target frame is unknown or if it is not possible to form the action sequence due to an insufficient timeline resolution. Moreover, a gulf of evaluation is possible when there is not enough information to evaluate the current position in the video.

and video, the situation is usually different—the content constantly changes. For a better understanding, we use the concept of *loci of attention*, introduced by Raskin [2000].

Definition:
Locus of attention

LOCUS OF ATTENTION:
The single source or location of sensory input that a person attends to at a given time, such as the point in space that they are looking at and able to devote mental resources to interpreting.

When working with audio and video material, the switches between content and controls create a *'game' of target acquisition* [Ramos and Balakrishnan, 2003]. For example, to review the last three seconds of a video, the user has to move his *locus of attention* from the video content to the control devices, and vice versa. In the time the user needs to do this context switch, the video keeps moving on. In consequence, the object at the point she was looking at can have moved to another position, disappeared or been occluded. Moreover, the user has to handle two *loci of attention* which increases the cognitive load.

Spatial separation does not work well with dynamic content

3.1.3 Modal timeline resolution and imprecision

The scrolling speed of timelines depends on the video length and screen resolution. For short videos, one timeline pixel can correspond to one video frame, while, for long videos, one pixel may correspond to a few hundred frames. Figure 3.2 shows two screenshots of VLC,[2] a popular audio/video player. Observe that, in both screenshots, the timeline slider is at the same position, though, in the upper image, the audio track is at 02:01 minutes, and, in the lower image, another (longer) audio track is at 03:42 minutes. Hence, dragging the timeline slider for the same distance has to different effects. In other words, both the timeline scrolling speed and position visualisation are *modal* according to the following definition by Raskin [2000].

Scrolling speed is modal

MODAL:
A human-machine interface is modal with respect to a given gesture, when

1. the current state of the interface is not the user's locus of attention and
2. the interface will execute one among several different possible responses to the gesture, depending on the system's current state.

Definition: *Modal*

[2] www.videolan.org/vlc

The described modality is an inherent problem of timelines and other systems that use controls of fixed size for documents of different lengths (for example scroll bars).

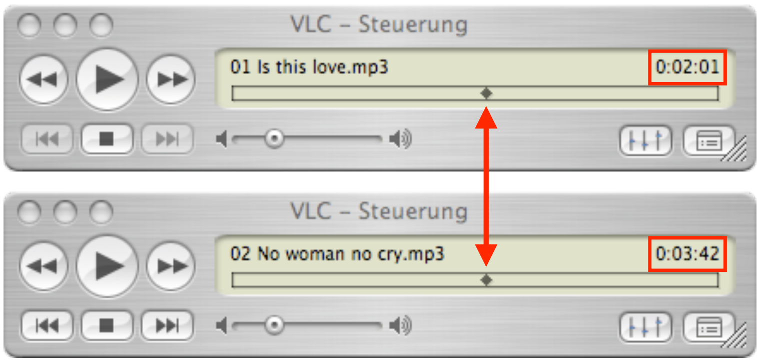

Figure 3.2: Modal timeline resolution and scrolling speed. Because the scale is relative to the length of a sample, the same timeline position (indicated by the red arrow) can represent different positions in time (see the red boxes).

The timeline may represent only one percent of the frames

Furthermore, the timeline is, in general, not able to represent every frame of a video. Consider a video with 25 frames per second and a computer screen with a resolution of 1024x768 pixels. If the timeline uses the whole screen width[3], this results in a maximum timeline resolution of 1024 different positions. To be able to access all frames of a video clip separately, the clip duration may not exceed 40 seconds, because $40 \text{ sec} \cdot 25 \text{ fps} = 1000$ frames. In a two minute clip, only one third of all video frames can be accessed directly, because, in this case, one timeline pixel corresponds to three frames. In a 90 minute video, the 1024 different timeline positions represent less than one percent of all frames. See table 3.1 for details.

The limited timeline resolution introduces two major issues

We conclude that the dependence of the timeline resolution on the video length introduces two major issues. First, the number of frames that is skipped when using the timeline for fast-forwarding and rewinding is modal; for every video, the user has to learn how far she has got to drag the mouse to skip a certain amount of time. Second, in general, the timeline resolution is not sufficient to enable users to do

[3]Usually, this is not the case as, in general, the timeline does not use the whole screen size.

Video length	**Represented frames**	**Frames per position**	**Seconds per position**
20 sec	100.0%	1.0	0.04
40 sec	100.0%	1.0	0.04
1 min	66.7%	1.5	0.06
2 min	33.3%	3.0	0.12
3 min	22.2%	4.5	0.18
5 min	13.3%	7.5	0.30
20 min	3.3%	30.0	1.20
90 min	0.7%	135.0	5.40

Table 3.1: Timeline resolution. Percentage of independently represented frames, frames per position ratio and seconds per position ratio (for a timeline with 1000 different positions, and videos with 25 fps).

frame-accurate analysis of single scenes in longer videos because only a part of the frames is represented. Thus, users often do not have the possibility to set the playback position to every single frame and to accurately review certain scenes.

Some players offer slow motion playback

Some systems provide modes to set the scrolling speed of the forward and rewind buttons to slow motion playback. This technique increases precision but can waste the user's time as she may have to wait a long time for the right moment to stop the video.

Gulfs of execution and evaluation are possible

Returning to our model of the seven stages of action, we can identify two gulfs for long videos. When the goal is to navigate to a certain point in time (target frame), both a gulf of execution and a gulf of evaluation can arise. First, it can be difficult to compare the target frame with the currently shown frame, when frames are skipped and the context does not give sufficient information about the current position in the video. Second, it is impossible to formulate a correct action sequence if the target frame is one of the non-accessible frames.

3.1.4 Missing natural mapping

Timelines use mapping between space and time

Due to cultural conventions, the leftmost point of a timeline represents the start, and the rightmost point represents the end. Timelines as used in history and archeology follow this scheme[4] as do timelines used in video playback systems. We observe that this introduces a mapping between time (video position) and space (timeline position). Unfortunately, this mapping can complicate users' lives. Although videos are time-based, usually the video content and not the position in time is the interesting part in a navigation task. Imagine a video of a train entering the screen on the right and leaving on the left side. In a reviewing process, the use of the timeline slider to accompany the train's motion is unnatural, because the (spatial) left-to-right movement on the timeline causes the train to move in the opposite direction, from right to left.

Object move at different speeds and directions

Another problem of both the timeline and the fast-forward/rewind buttons arises when the video shows objects changing the direction and speed of their motion. Standard video navigation controls use a constant scrolling speed, and, therefore, the video moves from frame to frame linearly. Objects usually do not move at constant speed. Consider the car shown in figure 3.3. First, the driver accelerates, then brakes, and stops the car. Afterwards, he accelerates again. Observe the positions marked on the timeline slider and the object trail. These positions correspond to each other (when the car is at position i, the timeline slider is at position i, as well). While the points on the object trail are equidistant, the distance between two consecutive positions on the timeline slider varies. We observe that this is similar to modal behaviour because the same amount of input motion (timeline slider) produces different amounts of output motion (object).

Equal temporal distances have distinct spatial distances

More mathematically expressed, a distance $\Delta t = t_j - t_i$ on the timeline slider corresponds to a distance $\Delta s = f(\Delta t) = f(t_i, t_j)$ on the object trail. In general, f is a non-linear function, i.e., for three temporal positions t_1, t_2, and t_3 with $t_2 - t_1 = t_3 - t_2$, the distances $\Delta s_1 = f(t_1, t_2)$ and

[4]There are exceptions: Some timelines start at the top and go to the bottom.

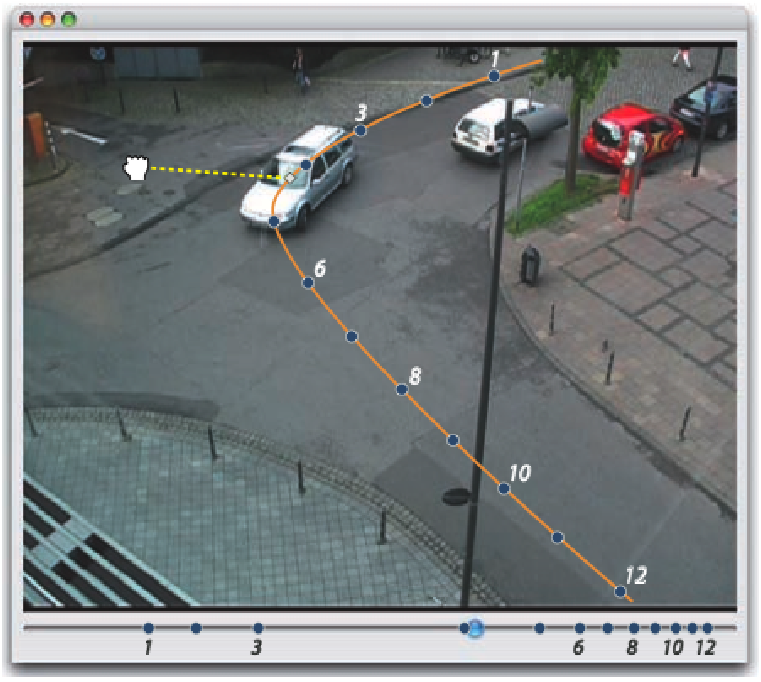

Figure 3.3: Non-linear movement [Karrer et al., 2008]. The numbered positions on the car's trajectory and the timeline slider mark the same points in time. The car moves slow between 3 and 4 to let the other white car pass. The highest speed is reached between 10 and 12.

$\Delta s_2 = f(t_2, t_3)$ are not equal. We call this phenomenon *non-linear movement*.

Non-linear movement causes gulf of execution

Non-linear movement introduces a gulf of execution when users want to find the position where an object passes a certain point in the video. Guessing the right Δt for a Δs is hard, and the task of finding the correct timeline position reduces to a trial and error task. It may therefore be necessary to perform several cycles of the seven stages of action. In each cycle, users have to guess the position on the timeline slider which, often, makes it impossible to form an action sequence that accurately scrolls the video to the target frame.

Gulfs of evaluation occur when objects move unpredictably

Moreover, a gulf of evaluation turns up in longer scenes with objects that move on unpredictable trajectories. For better understanding, consider figure 3.4. We suppose, that the video is at the frame marked by the red dot, and that the goal is to find the target frame, marked by the green dot. The shown trajectory could, for instance, represent the movement of a person captured by a surveillance camera. As we can see, the movement is unpredictable as various direction changes are involved. In a timeline-based system, it is hard to decide whether the target frame is in the future or in the past (relative to the current frame) unless one knows or watches the entire video. Thus, in such videos, the evaluation of the current state in relation to the goal state is difficult. For the car in figure 3.3, this is not the case; the trajectory is short and predictable. If the task is to drag the car to the position where it enters the lower half of the screen, evaluation is easy, and no gulf arises.

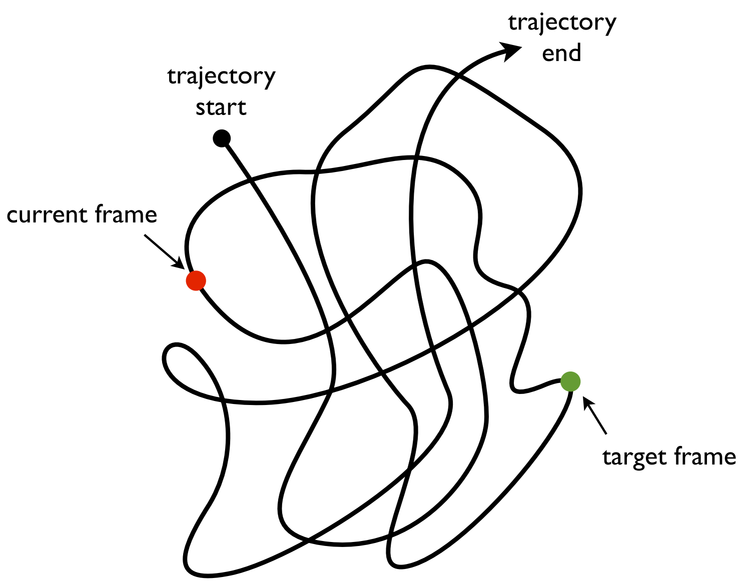

Figure 3.4: Gulf of evaluation in timeline-based systems. When objects move on long and unpredictable trajectories, it is hard to evaluate whether a target frame is in the future or in the past.

3.2 Dragon

Humans interact directly with objects

In interactions with objects in the real world, people do not use abstract controls to express their ideas and to move objects—they use their hands and grab the objects of interest. Direct manipulation interfaces transfer this behaviour to computer interfaces. In general, such interfaces are more intuitive and easier to understand than those using abstract controls.

DRAGON enables users to interact directly with object in the video

DRAGON is build upon a direct manipulation metaphor that enables users to grab objects in the video and to drag them along a virtual slider consisting of the positions the object passes. In the rest of this chapter, after a brief explanation on the implementation of DRAGON, we take a deeper look at how such a direct manipulation interface can benefit users, improve video navigation, and bridge the gulfs of execution and evaluation.

3.2.1 Algorithmic issues

DRAGON uses flow fields to calculate trajectories

The DRAGON prototype consists of two main parts. First, we use an optical flow implementation by Weiss [2007] to compute flow fields between the consecutive frames of a video file. These flow fields contain information about the motion of pixels and areas from a frame to its predecessor and successor frames. Second, the DRAGON user interface reads these flow fields and the video file. Based on the flow information, it computes the trajectory of a selected pixel by creating a chain of flow vectors. This is done on-the-fly. Therefore, the drag interaction starts as soon as the user selects a pixel (by mouse click).

DRAGON calculates spatio-temporal distance

In 1—"Introduction", we have described informally how DRAGON matches a video frame to the current mouse position. More formally, we search for the vertex on the object trajectory with the minimal spatio-temporal distance. This means, that instead of using two-dimensional vertices that contain only the x and y coordinates, we add the corresponding frame number f to the tupel. Then, we calculate the current spatio-temporal distance with respect to the mouse cursor position (x_m, y_m) and the current frame f_c for all vertices (x_v, y_v, f_v) of the trajectory using the following definition.

$$d(x_v, y_v, f_v) = \sqrt{(x_v - x_m)^2 + (y_v - y_m)^2 + (f_v - f_c)^2 \cdot t^*}$$

where t^*, the so-called time scale, is a scalar to define the impact of the frame distance. We use $t^* = 1.0$ as default. Note that for $t^* = 0$ the mapping reduces to a standard minimal distance calculation.

3.2.2 Direct manipulation

According to Shneiderman [1987], we define direct manipulation interfaces as follows.

Definition: *Direct manipulation*

DIRECT MANIPULATION:
Direct manipulation interfaces require the following properties.

1. Continuous representation of the object of interest.
2. Physical actions or labelled button presses instead of complex syntax.
3. Rapid incremental reversible operations whose impact on the object of interest is immediately visible.

DRAGON is a direct manipulation interface

DRAGON is a direct manipulation interface in this sense. The object of interest is continuously represented as the video proceeds (1), dragging is a physical action (2), and, in general, the dragging operations are reversible (3). Furthermore, Shneiderman highlights the following advantages and properties of direct manipulation interfaces.

1. Novices can learn basic functionality quickly, usually through a demonstration by a more experienced user.
2. Experts can work extremely rapidly to carry out a wide range of tasks, even defining new functions and features.
3. Knowledgeable intermittent users can retain operational concepts.
4. Error messages are rarely needed.

5. Users can see immediately if their actions are furthering their goals, and if not, they can simply change the direction of their activity.

DRAGON benefits from direct manipulation metaphor

For DRAGON, all properties except (2) are valid. Dragging objects does not require much explanation; the technique is well-known from a variety of applications, including drag-and-drop operations with icons and modification techniques for visual elements in graphics software. Finally, error messages are rarely needed as users receive immediate feedback. Moreover, we observe some additional properties contrasting the disadvantageous properties of timeline systems. First, the video content and positioning controls are not spatially separated. Second, the screen resolution has no influence on the precision of navigation as long as it is possible to show the video in its original size (or magnified). Third, there is a strong natural mapping between the input mouse movement and the resulting object motion.

The following sections describe these properties in more detail.

3.2.3 No separation between content and controls

DRAGON bridges gulfs of execution and evaluation

The most obvious difference between DRAGON and traditional video players is that, in DRAGON, the video controls are not separated from the video content. The entire video screen serves as both output view and input control.As a direct consequence, users do not need to change their locus of attention to perform navigation tasks; they can simply select an object and drag it around in the scene. Furthermore, the direct access to the moving objects reduces what Hutchins et al. [1985] call *semantic distance*.

Definition: *Semantic distance*

SEMANTIC DISTANCE:
Semantic distance concerns the relation of the meaning of an expression in the interface language to what the user wants to say.

Reduction of semantic distance bridges gulfs of execution and evaluation

The reduction of this semantic distance bridges the gulfs of execution and evaluation; the object can be dragged to every position of its trajectory at every speed, and the user receives continuous feedback as the video accompanies the motion. The user decides, by demonstration, to which position the video should go.

3.2.4 Natural mapping between mouse input and object motion

Direct manipulation creates natural mapping and solves problem of non-linear movement

According to Norman [2002], a "natural mapping, [. . .] taking advantage of physical analogies and cultural standards, leads to immediate understanding". Direct manipulation for video navigation as implemented in DRAGON introduces such a natural mapping between the mouse input and object motion. Instead of mapping space to time as in timelines, we establish a 1:1 mapping from space to space. Then, neither direction nor velocity changes of the object have influence on the drag interaction. As the position of the selected object positions is being synchronised to the mouse cursor position, a motion from the left to the right results in the video scrolling such that the selected object moves from left to right as well, and a faster movement of the mouse scrubs through the video at a higher speed.

The natural mapping bridges gulf of execution

This bridges the gulf of execution mentioned before. In figure 3.4, object dragging enables us to drag the object to the green position without any notion of time, and we do not have to guess the correct position on the timeline slider. The gulf of evaluation persists as long as trajectories are unpredictable and we do not show the object's trajectory.

3.2.5 Independence of screen resolution and video length

Object dragging is independent of the video length

In general, the scrubbing precision is independent of the widget size and video length. This is a direct consequence of the natural mapping. Exceptions occur, when the video is not exhibited at its original size because, in such cases, not all pixels are visible and accessible by the mouse. On the other hand, this is not really a disadvantage as, for the same reason, the user cannot see the difference. Moreover, the video length has no influence on the precision.

The precision now depends on the object velocity

Observe that, when all frames are accessible, we do not have a gulf of execution in the formulation of the action sequence. However, we cannot always conclude that all frames are directly accessible. When objects move slow, the video resolution has an impact on the scrubbing precision, and when objects pause, several frames correspond to the same object position. Hence, in these situations, not all frames are always accessible. We return to this problem and present approaches to bridge the related gulf of execution in chapter 5—"Motion gaps".

3.3 Online survey

Users have different demands

We have seen that the use of direct manipulation for video navigation is a promising approach that, in general, bridges the gulfs of execution and evaluation introduced by timeline-based systems. However, we do not know the user's demand for such a technique, yet. Therefore, we conducted an online survey to get to know usual tasks, and to determine the necessities of users. 27 users participated with the following profile (not all participants answered all questions).

Age 21 of them were younger than 30 years and four older than 30 years.

Gender 24 male participants, three female participants.

Computer use Two participants used computers between five and six times per week, the rest every day.

Video playback tools Two participants used video playback tools up to two times per week, six users between four and six times, and 14 users every day.

Video editing 26 participants use video editing tools never or at most once per week, one participant five to six times per week, and one participant every day.

This online survey was used to obtain information on other topics related to DRAGON. Therefore, we will not repeat the participant profile in the subsequent chapters, but refer to this section.

3.3.1 Video reviewing

Over 60% review parts of videos

We wanted to know whether users review videos, and, if yes, how they perform this task. Users were allowed to select multiple answers. In total, 71% watched videos from beginning to end, 61% of all participants reviewed video material and 32% watched only parts of videos (figure 3.5). Some participants marked only one answer; we believe that they might have misunderstood the question. Of these, about 29% answered that they watched videos only from the beginning to the end, i.e., they did not review certain events and had no demand for a video navigation tool. Another 21% answered that they only reviewed videos and did not watch videos from beginning to end. 7% did neither watch entire clips nor review scenes, but skipped parts to watch interesting parts only.

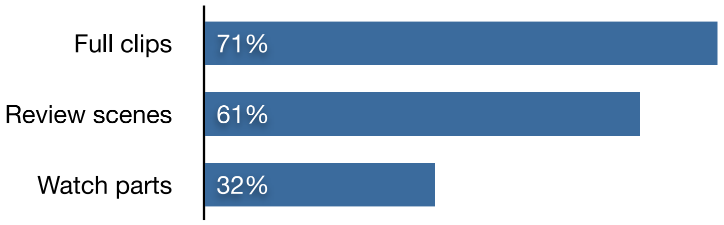

Figure 3.5: Percentage of users watching full clips, reviewing scenes, and watching parts of video clips. Users were allowed to select multiple answers.

Frame-exact navigation is not always necessary

Furthermore, users were asked to answer whether frame-exact navigation was important for them. 18% percent of the participants answered that they need frame-exact navigation, and 71% percent answered that a few frames before or after an interesting event are acceptable. 11% did not use review features (figure 3.6).

Timeline slider is most used control

In the last question about review strategies, users were asked to answer how they work with the timeline and the

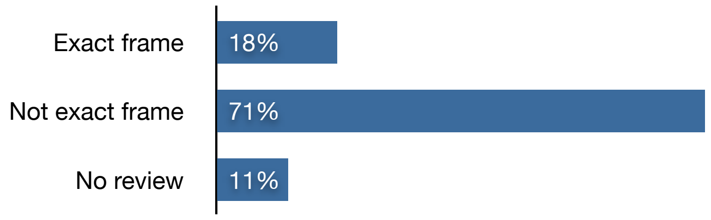

Figure 3.6: Necessity for frame-exact navigation tools. Users had to choose one option.

forward/rewind buttons. 11% of all participants use the buttons, while 54% guess a position on the timeline slider, and 71% drag the slider until they find the desired position. 32% use a combination of guessing the position and dragging the slider (figure 3.7).

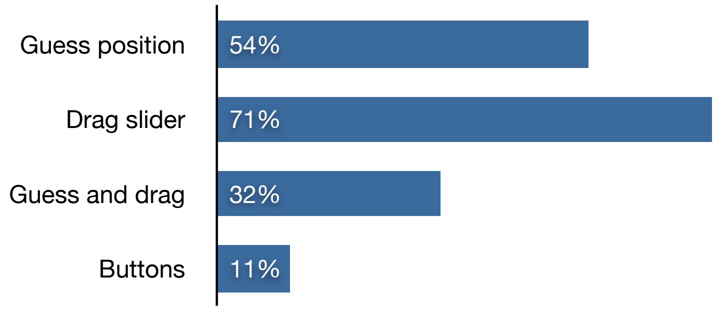

Figure 3.7: Video navigation interface components used by participants. Only a 11% use fast-forward and rewind buttons. Users were allowed to select multiple answers.

There is necessity for in-scene navigation tools

We conclude that there is demand for in-scene video navigation tools. Over 60% of the users review scenes, though only 18% look for an exact frame. Usually, users perform these reviews either by guessing a position on the timeline or by dragging the slider until the correct position is found. As only 11% use the forward and rewind buttons, we further conclude, that it is important to have random access and to be able to review the material at different speeds and directions.

3.3.2 User satisfaction

Users are more or less satisfied with the timeline

In the online survey and during our test sessions, we asked users to decide whether the timeline is an appropriate tool for micro and macro navigation. We used a 5-item Likert scale, where 1 means "strongly disagree" and 5 "strongly agree". For the question *"the timeline slider is appropriate for micro navigation"* we measured an average value of 3.21, and for the question *"the timeline slider is appropriate for macro navigation"* an average of 3.94.

Users find object dragging appropriate for micro navigation

We presented screen recordings of interaction sequences with the DRAGON prototype to the participants of the online survey, and we asked whether *"object dragging is appropriate for micro navigation"* and *"object dragging is appropriate for macro navigation"*. Users answered with an average of 4.25 that object dragging is appropriate for micro navigation, while the average value for the macro navigation case was 2.52. Furthermore, we asked whether object dragging was useful for the participant, answered in average with 3.57, and whether they thought that it would be useful for others, answered in average with 4.17.

The results match our expectations

These results match the expectations induced by our prior analysis of timelines and direct manipulation. For macro navigation, the timeline is a reasonable interface component and users are satisfied with the functionality. Moreover, users have taken a correct decision in saying that object dragging is not suitable for macro navigation. For micro navigation tasks, DRAGON was the preferred solution, though many users are satisfied with the timeline.

Users prefer object dragging

In another test session, seven users could actually test the system and were asked, which technique they preferred for video navigation. Six of seven users preferred object dragging, four users said that object dragging felt to be faster to perform navigation tasks, and five users found object dragging easier than using the timeline.

3.3.3 Professional users require high precision

Video cuts require exact navigation tools

A test session with four professional users, all working at Westdeutscher Rundfunk Aachen, a German TV station, has shown that high precision is a key factor in video cut applications. To perform a video cut, the cutter searches for the so-called *in* and *out points*. These points determine the position in time where one video is cut together with a second video. Often, a *master scene*, for example a recording of a dialogue captured from a distance of a few meters, is cut together with several *close-ups* that show further details of the scene. For illustration, suppose that a persons takes a cup of tea which is on a table. We are looking at the whole setting from a certain distance, and the director wants a close-up of the hand reaching the cup. To create the impression of a fluid motion, the hand has to be at exactly the same position in both the master scene and the close-up. Therefore, professional video editors need a tool that provides frame-exact navigation. Usually, jog wheels are used for this purpose.

Professional users need precision

We conclude that a direct manipulation interface, at least for professional users, has to make sure that all frames are always directly accessible. If frame-exact navigation is not made possible, the system becomes useless. Furthermore, interviews with these professional users revealed that it is important to play the audio contained in the video file during navigation, and that, though they found that object dragging is easy, the jog wheel remains their preferred tool. However, they found object dragging useful for video cuts that involve a high amount of motion.

3.4 Problems and drawbacks with direct manipulation for video navigation

Direct manipulation is complex and can have drawbacks

An user interface has to take into consideration four crucial criteria to provide a good and satisfying user experience—visibility of state and actions, full and continuous feedback, natural mappings, and a good conceptual model [Norman, 2002]. We have seen that direct manipulation helps us to achieve these design goals, but "it turns out that the direct manipulation concept is complex. Moreover, although there are important benefits there are also costs. Like everything else, direct manipulation systems trade off one set of virtues and vices against another" [Hutchins et al., 1985]. Though DRAGON, as the motion of video objects is strongly connected to the motion of the mouse cursor, provides a good natural mapping, immediate feedback, continuous visibility of the current state, and an easily understandable conceptual model, there are situations and trajectories that require special handling and extended features.

Different shapes of trajectories are possible

First, trajectories can appear as straight lines, curves, waves, circles or any other 2D shape, and some trajectories can be followed more easily than others. E.g., in general, dragging objects along waves is more difficult than dragging along a straight line trajectory (when we use a mouse as input device). Moreover, objects move at different speeds. In chapter 4—"Object trajectories and visualisation", we present problematic trajectories, different visualisations capable of conveying information about object speed (figure 3.8) and creating different affordances. Additionally, we present a user study evaluating the user performance with different types of visualisations.

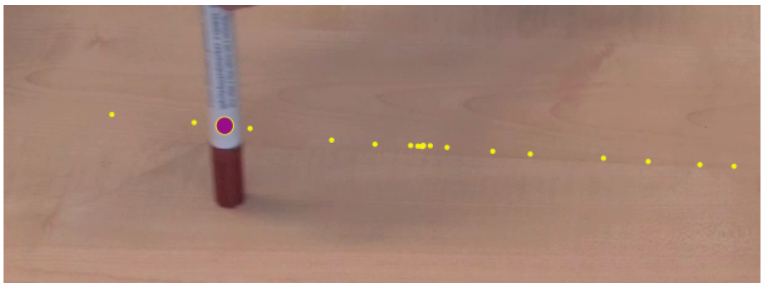

Figure 3.8: Example for a velocity visualisation. The points' density allows for conveying the object's velocity.

Objects can interrupt their movement

Second, objects can interrupt their motion for a time while other objects keep moving. Then, these objects have trajectories where a number of vertices have the same Cartesian coordinates but different frame numbers (figure 3.9). In chapter 5—"Motion gaps", we describe that dragging objects over such points can lead to leaps in time, disturb the mapping between mouse and object position, and even impede video scrubbing. We propose approaches to overcome this problem and present three studies.

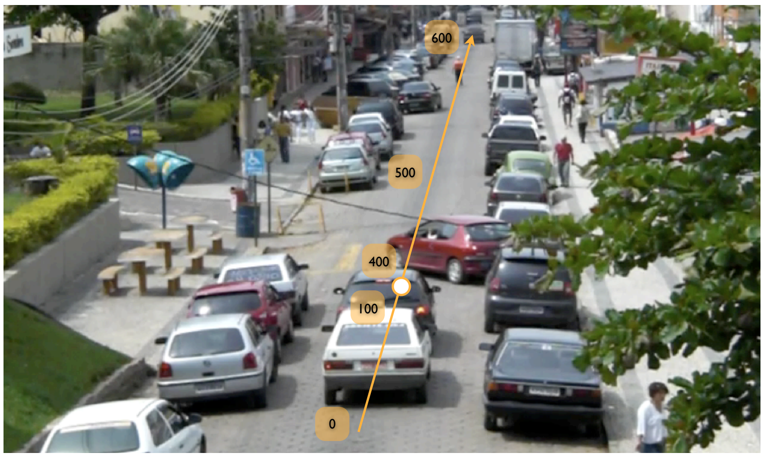

Figure 3.9: Example for a trajectory of an object that pauses its motion. The black car stops for 300 frames (frames 100 to 400). Hence, the trajectory vertices for the frames 100 to 400 have same Cartesian coordinates. This can disturb the mapping between mouse and object position

Objects can pass the same points twice

Third, pendulum-like trajectories are possible, i.e., objects move to and return from a certain point on the same or a simliar trajectory. At these turning points, the gesture for forward and backward scrubbing is identical because the object moves along the "same" trajectory in two distinct moments and directions. Nevertheless, the system has to be determistic and choose one of the two subtrajectories (figure 3.10). We analyse such situations and present a user study on different approaches in chapter 6—"Pendulum-like trajectories".

Note on our user studies

Five users are sufficient to find most of the usability errors in the design

Nielsen and Landauer [1993] propose a *mathematical model of the finding of usability problems* and estimate the number of found usability problems with n test users by

$$\text{problems found} = N(1 - (1 - L)^n)$$

where N is the number of all usability problems in the design, and L is the proportion of usability problems found

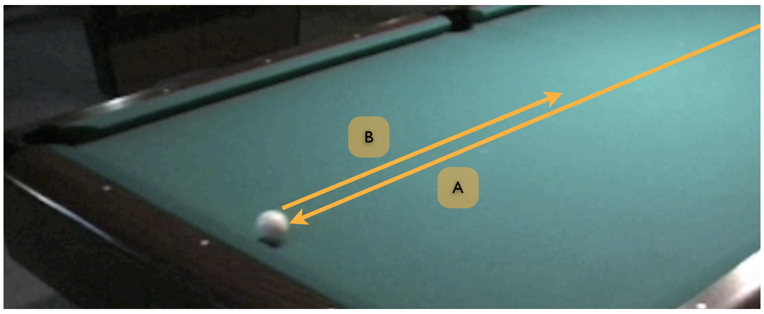

Figure 3.10: Example for a pendulum-like trajectory. The billiard ball arrives and returns on a very similar trajectory, coming in direction A, hitting the border, and returning in direction B. At the lowermost position, DRAGON has to decide which part of the trajectory should be used for the mapping between mouse positions and trajectory vertices.

by a single user (acording to Jakob Nielsen's Alertbox – Usability Testing with 5 test users,[5] the typical value of L is 31%). Furthermore, Nielsen advises that *after the fifth user, you are wasting your time by observing the same findings repeatedly but not learning much new.* Therefore, we recruited at least participants for each user study, and expect to find 80% of the usability problems present in our approaches.

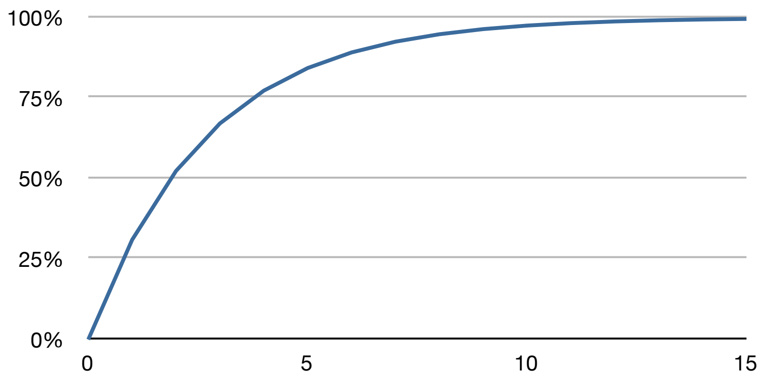

Figure 3.11: Percentage of found usability problems for different numbers of study participants. Five users are sufficient to find 80% of the usability problems in the design.

[5]http://www.useit.com/alertbox/20000319.html

Chapter 4

Object trajectories and visualisation

"The greatest enemy of knowledge is not ignorance, it is the illusion of knowledge."

—Stephen Hawking

In this chapter, we explore different types of trajectories, including straight lines, curves, and waves. Through a user study, we see that the difficulty of interaction depends on the trajectory's shape, and that users do not always follow the exact object motion. Moreover, we consider trajectories of objects with non-linear movement and present possible visualisations of both directions and object velocity.

4.1 Different shapes of trajectories and the representation of direction

We use full trajectories, arrows, and hidden trajectories

Additionally to not showing the trajectory of an object at all, we implemented two types of trajectory visualisation—full trajectories and arrows. While the first option draws the full trajectory of an object, arrows show the possible dragging directions in temporal forward and backward direction around the current frame (figure 4.1).

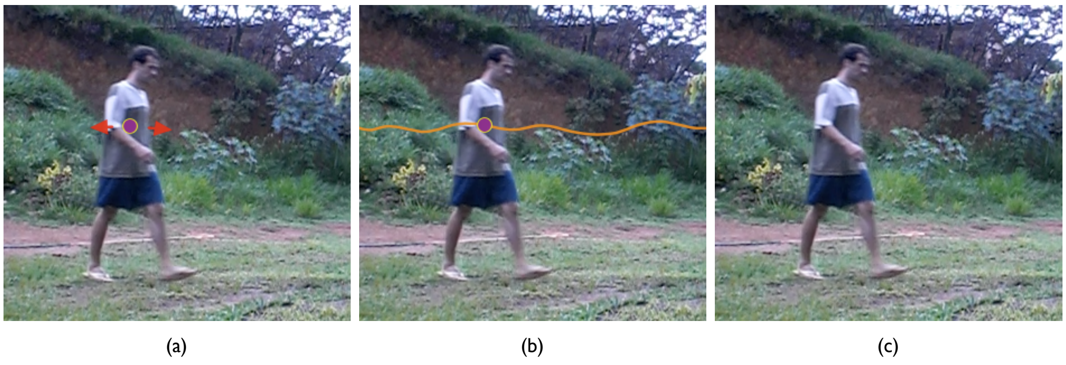

Figure 4.1: Three approaches to guiding users. (a) show arrows indicating the motion in the previous and next frames, (b) show the full trajectory, and (c) show neither trajectory nor arrows.

Full trajectories bridge gulf of evaluation

Remember the gulf of evaluation introduced by long and unpredictable trajectories (see figure 3.4). With full trajectories, we augment the user's notion about the development of a motion, and, hence, bridge the gulf. With hidden trajectories and arrows, users do not have this extra information. However many navigation tasks often do not require this knowledge.

The figures in this chapter show real mouse movements

In the following sections, we present different types of trajectories and the results of a user study conducted with seven test users. For illustration, we use figures to visualise the mouse movements users have performed while dragging objects and the trajectory vertices that were chosen by the matching function. To create these figures, we saved the mouse positions and the coordinates of the matched trajectory vertices to a text file, each time a user dragged an object. Later on, we created graphical representations of the mouse positions and trajectory vertices using coloured points (red points for the trajectory, blue points for the mouse positions) in a Cartesian coordinate system. In the last step, we chose a frame from the video used in the test session and inserted the previously created representation as an overlay. Hence, the following figures show the natural mouse movements and the trajectory vertices that were chosen by the matching algorithm. Note, that the points do not necessarily form a continuous line, neither the mouse positions (due to mouse acceleration, response times of the operating system, and other factors, the mouse cursor usually does not move smoothly), nor the trajectory vertices (due to the matching function).

We conducted a study on the interaction with trajectories and visualisations

In a user study, participants were asked to perform simple dragging tasks. In random order, either DRAGON visualised the trajectory, using arrows or the full trajectory, or hid it. We were particularly interested in whether the mouse movement would always follow the trajectory, whether the different visualisations had an impact on the distance between the mouse cursor and trajectory vertices, and whether the currently used mapping function for mouse positions to vertices was suitable for all situations. Furthermore, we wanted to know whether the presented approaches were sufficient to guide users. Remember that guidance may be necessary because direct manipulation for video navigation is constrained to object trajectories.

4.1.1 Straight lines

Interaction with straight lines is simple and works well

Interaction with straight line trajectories is the simplest case. Even when users do not follow the exact shape of a trajectory, the spatio-temporal mapping works well; all frames are shown during a drag interaction, usually without leaps. Furthermore, the different visualisations have no impact on how users drag the mouse to find a certain position in the video. In figure 4.2 we see a typical interaction sequence performed by one participant. The blue and red dots represent the recorded mouse and object positions, respectively. We can see that the blue mouse trajectory has approximately the same shape as the red object trajectory. Note that in the image labelled “Trajectory”, the previously described phenomenon of a non-continuous line appears; the object trajectory shows some leaps (in the centre of the image) because the user performed a fast drag interaction.

4.1.2 Curves

Interaction with curves depends on visualisation

Curved trajectories can introduce problems. As long as a user drags an object close to its trajectory, the calculation of the next frame works well. However, in figure 4.3 we see

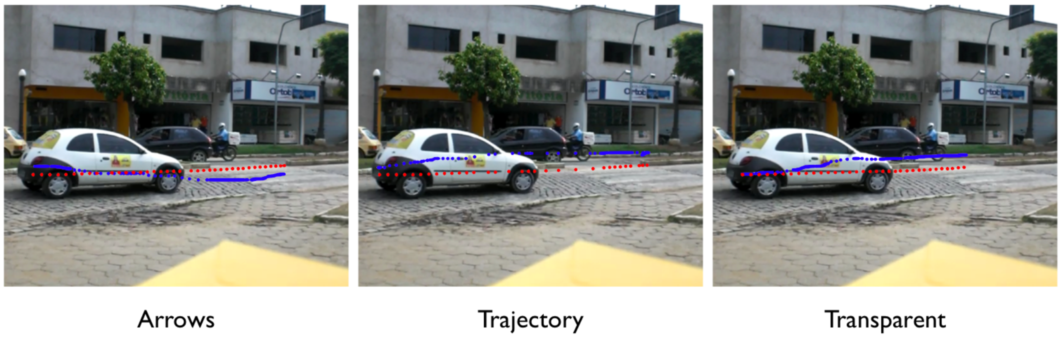

Figure 4.2: Mouse (blue) and corresponding object (red) motion with three different visualisations. For straight trajectories, the mouse motion usually follows the shape of the trajectory, and the distance is, in general, small. Tracking works well in these cases.

that this is not always the case. In fact, all test users dragged the mouse along a shorter, more straight, path on the inner side of the curve. The distance to the real object path increased when the visual guidance decreased, i.e., when using arrows or hiding the trajectory. We believe that this has three reasons.

Three reasons cause a higher distance to the object path

First, the trajectory affords users to follow the exact motion while the other two options do not introduce such an affordance. Second, the *Gestalt Law of Simplicity* tells us that users reduce the complexity of shapes, i.e., when dragging an object along a curve, users tend to flatten the curve and use this simpler path. Third, as we asked users to drag the silver car along its trajectory to the position where it is about to leave the screen, using a shorter path is a reasonable approach to reach the target position more efficiently.

Flattened mouse paths can produce leaps in the playback

Unfortunately, due to the evaluation function used for the mapping between mouse and object position, the dragging along this shorter path leads to leaps in the playback of the video and can confuse users. In figure 4.3 we see that, when arrows or no visualisation is used, mouse movements were such, that DRAGON jumps the frames in the upper left part of the object's trajectory. We have marked this part with an orange line. Some users were so surprised of this sudden context change, that they scrubbed the car back to its initial position and restarted the interaction, that time closer to the trajectory. In this example, the white car was responsible

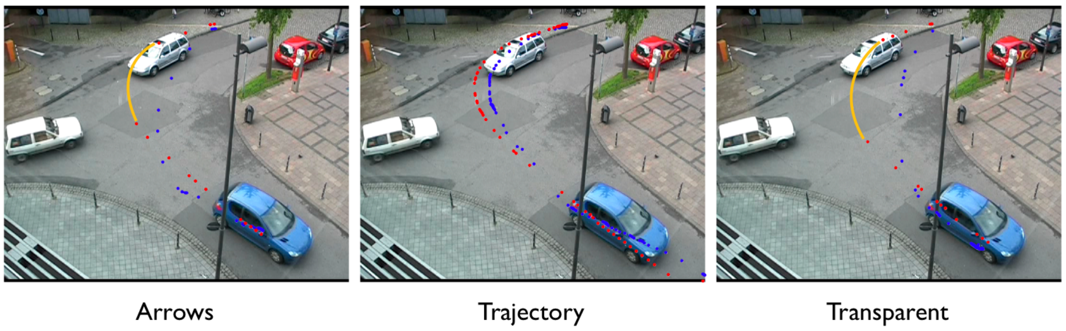

Figure 4.3: Mouse (blue) and corresponding object (red) motion with three different visualisations for curved trajectories. Usually, users use a shorter path on the inner side of the curve to drag the object. When less obtrusive or no visualisation is used, this mouse motion path gets shorter and the distance to the trajectory increases, thus provoking leaps in the playback (orange lines).

for the surprise, because it is at the left side of the screen before the leap and at the right side afterwards (this creates the impression that the white car passes through the silver car, or vice versa).

In chapter 5—"Motion gaps", we handle situations where objects pause their motion for a certain amount of time. In such cases, similar leaps in the playback may occur.

4.1.3 Waves

Interaction with waves is hard

Waves are a restricted class of curves, where the curve direction changes repeatedly. In figure 4.4, a basketball flies against a wall and then hits the ground several times until it leaves the screen on the left side. We asked users to analyse the ball's motion, i.e., drag it to the left side of the screen while trying to hit the ground as often as possible without producing leaps. Users found it hard to accomplish this task. This is due to the current implementation of the calculation of the next frame.

Shortest distance mapping produces leaps

For illustration, consider figure 4.5. To access or to see all frames, users have to keep a small distance to the trajectory during the drag interaction. When users move the mouse as shown in the two pictures, the shortest distance algo-

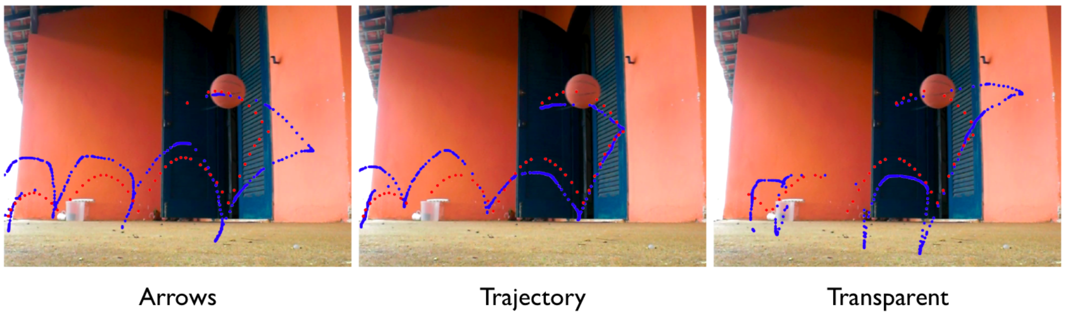

Figure 4.4: Mouse (blue) and corresponding object (red) motion with three different visualisations. For wave-like trajectories, the spatio-temporal approach to calculate the next frame frequently leads to jumps in the video playback during drag interactions. Hiding trajectories increases the level of difficulty while full trajectories allow for high accuracy.

rithm jumps the part of the trajectory under the imaginary line between the two mouse positions. This effect disturbs the natural mapping between mouse and object positions because a very small mouse movement causes a much bigger object motion. Observe that the time scale (see 3.2.1—"Algorithmic issues") and the frame distance between the two moments has an impact on the size of the leap that will occur; with both a higher frame distance or a higher time scale more frames will be shown during playback, preserving the natural mapping.

Interaction gets harder with hidden trajectories

Furthermore, scrubbing the video without producing leaps is harder when we hide the trajectories (figure 4.4), because the missing visual guidance provides less information on the development of the motion and, consequently, makes it harder to stay on the correct part of the trajectory. Apparently, for users preferring unobtrusive visualisation, arrows are suitable, though full trajectories enabled users to scrub through the video with the highest accuracy[1]. Based on our observations, we believe that arrows afford users to anticipate the object motion by moving the mouse in front of them, while trajectories afford—and enable—users to follow the trajectory with a higher accuracy.

[1]Note, that full trajectories enable users to analyse the ball's motion without even having to drag the object. We expect, that showing the whole trajectory is particularly useful for annotation tasks because, then, finding interesting points is often possible without having to watch the video.

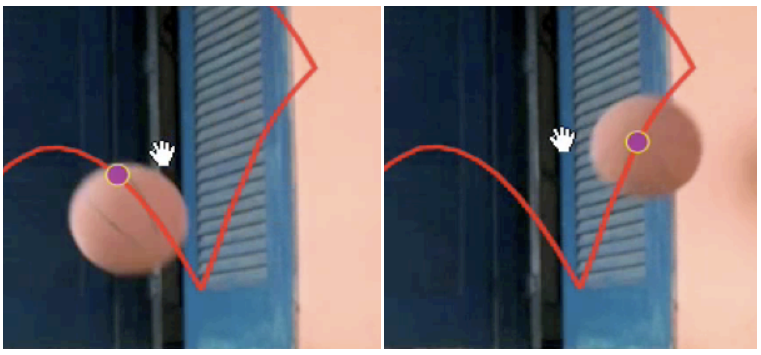

Figure 4.5: Problem when using shortest distance. A very small mouse movement causes a huge leap in the playback. This disturbs the natural mapping between input and output motion, and surprises users.

Linear regression mapping

LRM matches against linear regression of trajectory

For future work, we propose to investigate the use of *linear regression mapping* (LRM), a modified algorithm to determine the next frame (for illustration, see figure 4.6). Instead of matching the mouse position against the exact trajectory, the algorithm calculates the linear regression of parts of the trajectory and matches the mouse position to these new trajectories. A point of the regression trajectory represents the point of the original trajectory which is the vertex of the original trajectory and the orthogonal to the regression trajectory.

A preliminary study on LRM showed promising results

We have conducted a preliminary test with two users on the basketball example. Instead of calculating the correct linear regression, we have approximated the mapping by ignoring the y-distance. The results were promising; scrubbing through the video and accessing the frames around the positions where the ball hits the ground is easier than with the original spatio-temporal mapping; leaps do not occur (observe that this technique would also eliminate the leaps in the curves from the previous section). For a future evaluation it is necessary to implement an algorithm to efficiently extract the parts of the trajectories where linear regression mapping should be used, because, in general, full trajectories can not be represented by their linear regression—in

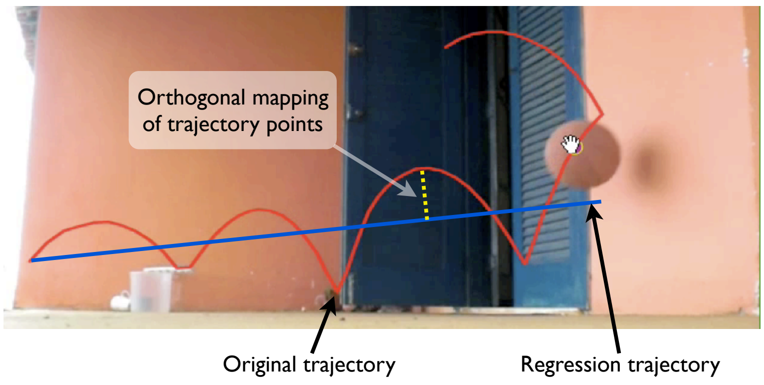

Figure 4.6: Linear regression mapping. Along dedicated parts of the trajectory, we exchange the spatio-temporal mapping against the original trajectory with a mapping to a linear regression trajectory. The frame where the object crosses the orthogonal line to the regression trajectory is chosen as the next frame.

figure 4.6, the blue regression trajectory represent only the part of the trajectory after hitting the wall.

4.1.4 Edges

Interaction with edges is usually easy

Edges occur frequently in videos of objects that hit limits like walls or that collide with other objects. In the previous section, for instance, the basketball's trajectory contains edges at the points where it hits the ground. Figure 4.7 shows billiard balls that change the movement direction due to a collision. We asked users to drag the yellow ball to its final position. All users performed this task by first dragging the ball to the collision position in the centre of the screen. As in figure 4.4, the distance between the mouse cursor and the object trajectory increases after arriving at the edge, because users need some time to realise the direction change. In general, we observed that this distance is lower when the full trajectory is shown. However, this makes no difference for the interaction because the frame does not change, and the context remains the same. Moreover, scrubbing to the edge of a motion is one of the simplest tasks because the user only needs to drag the mouse

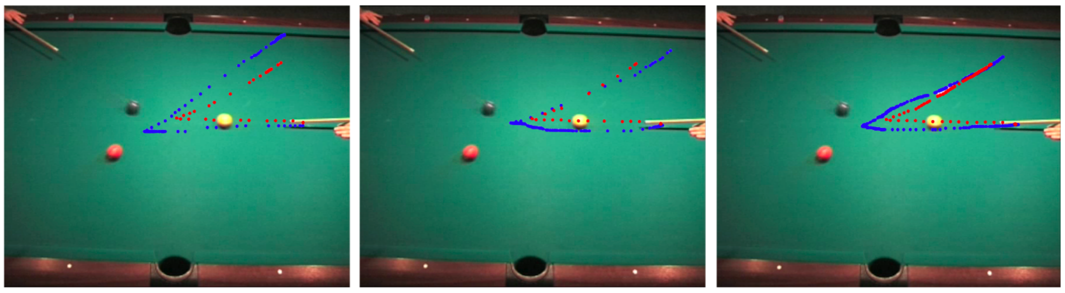

Figure 4.7: Mouse (blue) and corresponding object (red) motion with three different visualisations. For trajectories with edges, the mouse motion usually follows the shape of the trajectory until the edge. At this point, independent of the visualisation, the distance increases dramatically until the user perceives the direction change. Nevertheless, tracking works well in these cases.

away from the edge to a position where the edge position is nearer than any other point of the trajectory. In figure 4.7, this would be the area left of the imaginary line between the two middle holes.

Scrubbing to the end of a trajectory is the simplest case

The same holds for scrubbing an object to the end of its trajectory. For the video of figure 4.8, we have asked users to reestablish the order of cubes using the red block. Most of the users accomplished the task by dragging the mouse to the right side of the blocks on an approximated trajectory. This is a signal for the understanding of the direct manipulation metaphor; the user knows that the interaction is constrained to the trajectory and uses this knowledge to facilitate the dragging.

Depending on the task, the algorithm does not need to be exact

Observe that, in this example, the tracking algorithm loses the red block at some point of the trajectory and starts to track the blue one (due to motion blur and illumination changes). However, in this task, the tower is the *locus of attention* and not the single block. Therefore, and as long as the result of the dragging operation is correct, users do not perceive the change; we have asked seven users to perform this task and no user noticed the selection change, even after being questioned whether there was a difference to previous tasks.

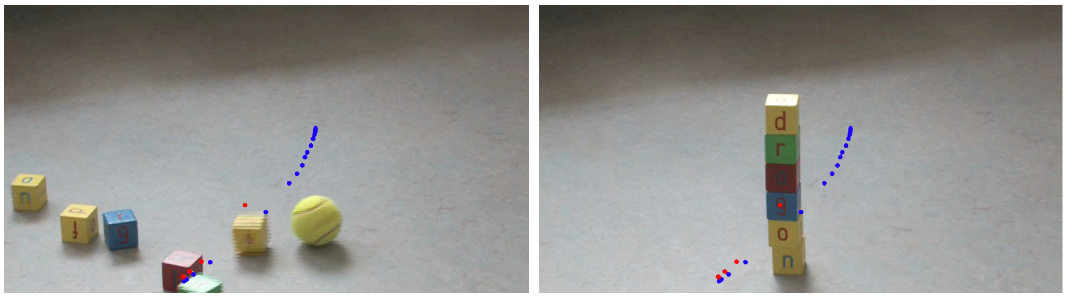

Figure 4.8: Mouse (blue) and corresponding object (red) motion for the task of dragging an object to the end of its trajectory. When users know the direction, they drag the mouse to a position behind the end of the trajectory. Observe that, in this case, the tower is the *locus of attention*. Therefore, the tracking algorithm does not need to be exact (at the left the red cube is focused, at the right frame the blue cube is selected).

4.1.5 Conclusion

Some tasks benefit from certain visualisation while others are independent

We have presented two types of guidance—arrows and full trajectories. Arrows are less obtrusive and, in most cases, sufficient to help the user to recognise the constraints and perform the dragging accordingly. Showing the full motion path provides more information about the whole scene and affords users to keep the mouse closer to the trajectory. This is especially useful for curved and wave-like trajectories, because, first, users tend to simplify the trajectory (drag along inner side of the curve), and, second, interaction sometimes requires high precision (because of shortest distance mapping). Finding edges and ends of trajectories is easy and does not depend on the visualisation.

4.2 Representation of velocity

We propose two simple approaches to represent the velocity of an object

We propose two different types to represent the speed of an object through the appearance of its trajectory—dots of different density and variable width trajectories. The implementation of both features is simple. To create the density dots, instead of drawing lines from a trajectory vertex to its successor, we draw every trajectory vertex as a single dot. When objects move slow, the dots get closer, and when objects move faster, the distance between two dots increases. However, for very slow objects, dots can overlap. In such cases, we draw only every i-th dot to avoid the creation of a continuous trajectory. To create trajectories of variable width, we first compute the average spatial distance between succeeding vertices. Then, when drawing the connection lines, we compare the average distance with the current distance and increase or decrease the line width, respectively. We can choose whether a smaller distance should correspond to a higher or a lower velocity.

The results were obtained through an online survey

As one part of the online survey from 3.3—"Online survey", we conducted a study on how users perceive these visualisations. We used five DRAGON screenshots, one using dots of different density (figure 4.9) and the other four using variable trajectory widths (figure 4.10). In each of the pictures, we have labelled parts of the trajectory, and users knew that the selected object changed its speed between the labelled positions. They were then asked to decide on which part of the trajectory the objects moved faster or slower, respectively.

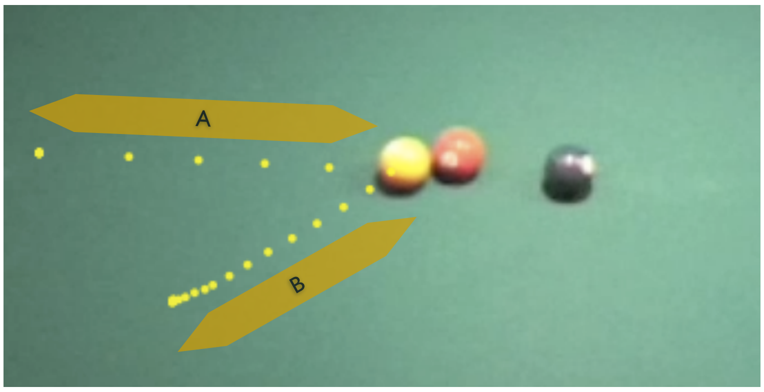

Figure 4.9: Visualisation of object velocity by dots of different density. 19% associate dense dots with faster motion, 81% with slower motion.

Most user associate thinner lines with faster motion

The user study revealed that most of the users associate thinner lines with faster motion. For the situations in figure 4.10 (b)-(d), 67% of all given answer indicate thinner lines for faster motion. In total, 26% of the answers propose the association of thicker lines to faster motion, 62% to slower motion, and 14% are undecided. Only example

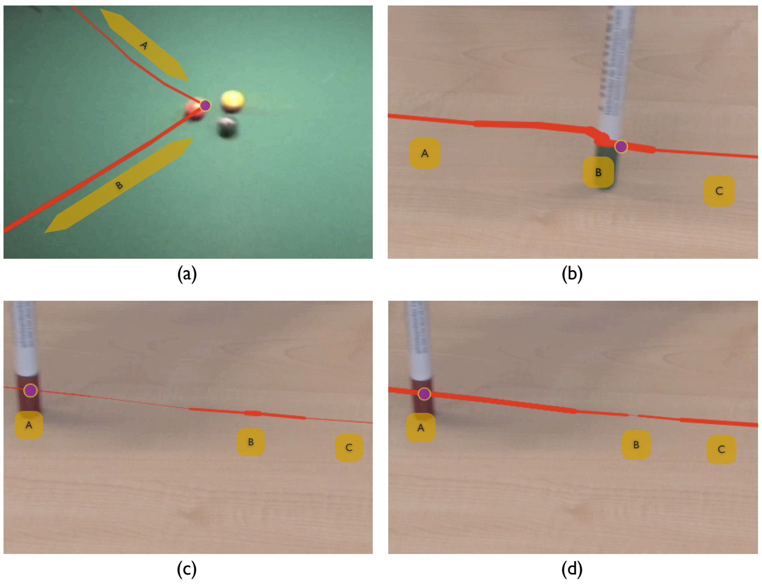

Figure 4.10: Visualisation of object velocity by variable trajectory widths. Study participants decided where the objects moved faster or slower, respectively. In 26% of all cases, thick lines are associated with faster motion, in 63% with slower motion. Picture (d) provoked deviating results: 52% associate slow motion with the thin line (A), 30% with the thick line (B).

4.10 (a) showed contrary results (see table 4.1); in this case, only 30% associated faster motion to thinner lines. We cannot explain why the result for this example is different. For further evaluation, we have applied a two-tailed binomial test at 5% significance level with the null hypothesis N_0 : "there is no preference for the association of thinner lines to either faster or slower motion" and the alternative hypothesis N_1 : "there is a preference for the association of thinner lines to either faster or slower motion". With a total of 79 gives answers (we do not count undecided users), and 56 answers for "thinner line for faster motion", we obtain a p-value of $p = 0.0001 < 0.025$. This is a significant result.

Real world connections may explain preference

Users commented that brush tools in graphics suites usually produce thicker lines when the brush motion is slower. Furthermore, we can observe the same effect when taking long-exposure photographs of light sources in dark environments. Due to blur, objects that move slower than others leave a thicker trace. These real-world connections may explain the preference.

Answer	Situation				Total	%
	a	b	c	d		
Thicker line for faster motion	12	4	3	4	23	26%
Thinner line for faster motion	7	15	18	16	56	62%
User does not know	4	4	2	1	11	14%

Table 4.1: Understanding of variable width trajectories. Most users associate thin lines with fast motion.

Most users associate denser dots with slower motion

83% of the participants associate dense dots with slower motion and only 17% with with faster motion (figure 4.11). No one was undecided. In an analogous binomial test with the null hypothesis N_0 : "there is no preference for the association of dense dots to either faster or slower motion" and the alternative hypothesis N_1 : "there is a preference for the association of dense dots to either faster or slower motion", in this case, a total of 23 answer and 19 preferences for the mapping of dense dots to slow motion, gives us $p = 0.0013 < 0.025$. Hence, there exist a significant preference for this mapping. However, a drawback of the dot visualisation is that the dots can appear as a cloud, and thus unconnected, for fast objects on non-straight trajectories. To overcome this problem, we propose to draw connection lines between the dots. Otherwise, users might get confused.

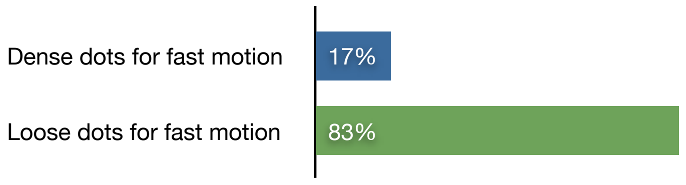

Figure 4.11: Understanding of density dots. The majority of users interpret dense dots as a visualisation for slow motion.

4.2.1 Conclusion

The results were statistically significant

We conclude, that both approaches are reasonable to visualise different speeds. Both results were statistically significant. However, many users (14%) were undecided regarding variable widths.

Chapter 5

Motion gaps

"Mind the gap!"

—London tube announcement

Often, objects pause their movement for some time while other objects keep moving. Figure 5.1 shows an example for such a situation; a car stops at an intersection, waits for a woman to enter, and, afterwards, leaves the scene. The car's stop time creates a series of trajectory points with the same (or very close) spatial coordinates. We call this event *motion gap*.

MOTION GAP:
An object has a *motion gap* from frame i to frame $i + k$, $k > 0$, at position (x_i, y_i) with threshold $\varepsilon > 0$, iff $\forall j : i < j < i + k \Leftrightarrow (x_i - x_{i+k})^2 + (y_i - y_{i+k})^2 < \varepsilon$.

Definition:
Motion gap

In other words, a *motion gap* is a position on an object's trajectory where the object either does not move for a certain time or moves at most ε units (can be measured in pixels, for example). The threshold ε is needed because the optical flow algorithm may produce noise and because objects can move such slow, that they produce a disadvantageous impact on the evaluation function used in the mapping of mouse positions to trajectory vertex.

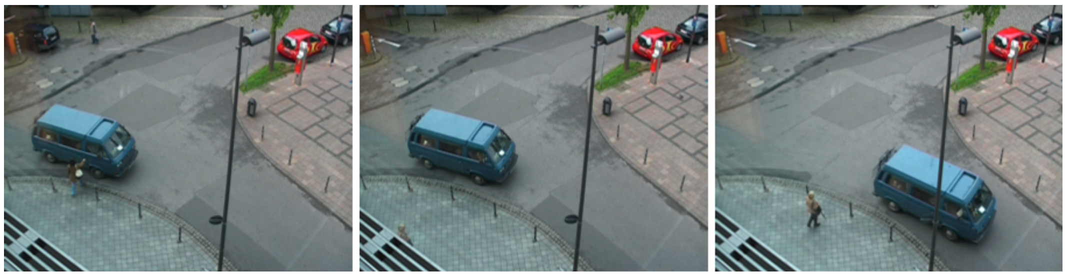

Figure 5.1: Motion gap. When the user drags the car, she either does not see the frames between the left and middle frame, or the drag interaction stops at the left frame (when using the normal spatio-temporal mapping).

As described in section 3.2.1—"Algorithmic issues", the next frame during a dragging operation is defined to be the one with the minimal d-value calculated by

$$d(x_v, y_v, f_v) = \sqrt{(x_v - x_m)^2 + (y_v - y_m)^2 + (f_v - f_c)^2 \cdot t^*}$$

where (x_v, y_v, f_v) is the corresponding vertex, (x_m, y_m) is the mouse position, f_c is the current frame, and t^* is the time scale. This calculation may lead to an undesirable effect when motion gaps exist. If the motion gap persists for a long time, the term $(f_c - f_v)^2$ can get considerably larger than the spatial distance terms. This makes interaction difficult as, for $t^* > 0$, the user has to drag the object far beyond the motion gap position to make the evaluation function choose frames on the other side of the motion gap.

We distinguish hard and soft motion gaps

We call such gaps *hard motion gaps* whereas *soft motion gaps* are gaps where objects stop but the frame distance is not so large, that dragging operations need to be interrupted. Observe that in extreme cases, the frame distance can increase such, that even the available screen area is not sufficient to drag the object over the gap.

5.1 First approaches

Hard motion gaps can destroy direct manipulation metaphor

If a hard motion gap occurs, the system could simply ignore the problem and force the user to choose a different object or interface component, for example the timeline, to scrub through the video until the object motion continues.

Obviously, this approach is neither very intuitive nor consistent with the definition of direct manipulation, because the user cannot move the object along its full trajectory. Furthermore, if a user wants to examine the motion of an object with several consecutive hard motion gaps (for example people taking a walk, cars in a traffic jam, or very slow objects, etc.), the direct manipulation technique loses its advantages if these interrupted trajectories are not joined in a way, that enables users to access all positions using a fluid motion. As in traditional interfaces, the user then has to focus on other objects or interface components to pass the motion gaps, changing the locus of attention consequently.

We need a fast method to determine motion gaps

Before we can start our analysis, we need a detection procedure. The following algorithm gives a fast approximation for motion gaps on a trajectory, where $\Delta v_{i,i+1}$ is the spatial distance between the vertices v_i and v_{i+1}.

Implementation: *Motion gap detection*

MOTION GAP DETECTION:
We use a simple method for the detection of motion gaps.

1. Calculate the full trajectory with vertices $v_1, \dots, v_n$, and set $i = 1$.
2. If $i = n$, terminate.
   If $\Delta v_{i,i+1} < \varepsilon$, create a new motion gap beginning at i.
   Otherwise set $i = i + 1$.
3. Iterate over v_j, $j = i + 1, \dots, n$, until $\Delta v_{j,j+1} > \varepsilon$ or $j = n$. Close the motion gap at j, tag it with the size $\delta = j - i$, and stop the iteration.
4. If $j < n$, go to 2 and set $i = j + 1$. Otherwise terminate.

Several techniques are thinkable

There are several possibilities to handle motion gaps. For a first evaluation, we have investigated the options *time scale zero*, *time scale timer*, and *velocity slider*, described in the following sections. We switch to motion gap mode, when objects enter a motion gap whose size δ exceeds a predefined frame distance δ_0. We have used $\delta_0 = 10$ frames for our studies.

5.1.1 Time scale zero

Time scale zero jumps the motion gap

The first and, from an implementation point of view, easiest possibility, is to jump the gap by setting $t^* = 0$. If an object interrupts its motion at frame f_s and resumes it at $f_r \neq f_s$, a dragging operation starting at frame f_i and ending at frame f_j, $i < s, j > r$, will then produce a frame sequence of the form

$$f_i, \ldots, f_{s-1}, f_s, f_r, f_{r+1}, \ldots, f_j.$$

Implementation: *Time scale zero*

TIME SCALE ZERO:
If the current frame is in a motion gap, set $t^* = 0$. Otherwise set t^* to its default value.

Time scale zero can confuse users

We expect that this approach is a reasonable method to avoid that users have to change the locus of attention and concentrate on other components while following an object trajectory. They can continue a dragging operation and do not need interrupt their movement to pass the motion gaps. However, on the other hand, we expect, that this solution might confuse users when other visible objects are moving, because jumping the motion gap then causes a sudden position change of the other moving objects. For example, dragging a car in a traffic jam will lead to non-continuous motion of the other cars.

5.1.2 Time scale timer

Time scale timer jumps the motion gap after a time-out

As an addition to setting the time scale to zero as soon as an object enters a motion gap, we propose another method. This method is very similar to the time scale zero approach with the difference, that the DRAGON jumps the motion gap only after a certain time, to be configured by the parameters Δt and f.

TIME SCALE TIMER:

1. If the current frame is in a motion gap, start a timer with frequency f.
2. For every timer tick, set t^* to $(2t^* + (1 - \Delta t))/3$.
3. If a frame change occurs, reset time scale, stop the timer and go to 1.

Implementation: *Time scale timer*

When no frame change occurs, a timer fires f times per seconds and decreases the time scale by Δt. Otherwise, DRAGON resets the time scale to its default value, in our case 1.0. We performed the study using $\Delta t = 0.05$ and $f = 10$ Hz.

5.1.3 Velocity slider

We have implemented a *velocity slider* similar to the *PVSlider* (see figure 2.9) in Ramos and Balakrishnan [2003]. For DRAGON, we use the velocity region, only.

MOTION GAP VELOCITY SLIDER:
δ_{min} is a distance threshold, f maps distances to speeds.

- If the current frame is in a motion gap, switch to velocity slider mode. If not, continue spatio-temporal tracking.
- If velocity slider mode is active, compute the distance Δ_{mouse} between the mouse cursor and the position of the dragged object.
- If $\Delta_{mouse} > \delta_{min}$, check whether the mouse cursor is at the side of the motion gap start or end.
  - If mouse cursor is at side of motion gap start, scroll backward with speed $f(\Delta_{mouse})$.
  - If mouse cursor is at side of motion gap end, scroll forward with speed $f(\Delta_{mouse})$.

Implementation: *Motion gap velocity slider*

Velocity sliding is enabled just-in-time

When an object reaches a motion gap, the spatio-temporal tracking is suspended, and DRAGON switches to the velocity slider mode. As feedback, we change the mouse cursor from an open-hand cursor to a resize cursor as shown in figure 5.2. When the system is in the velocity slider mode, a user can move an object over a motion gap. The farer she tries to drag the object beyond the motion gap position, the faster the object will pass the gap as the scrolling speed increases.

A velocity slider gives more control than time scale features

There are two fundamental differences between the velocity slider and the previously described time scale features. First, the user sees how the context changes during the motion gap without switching to another object or using the timeline, and, second, she can control the playback speed and direction dynamically.

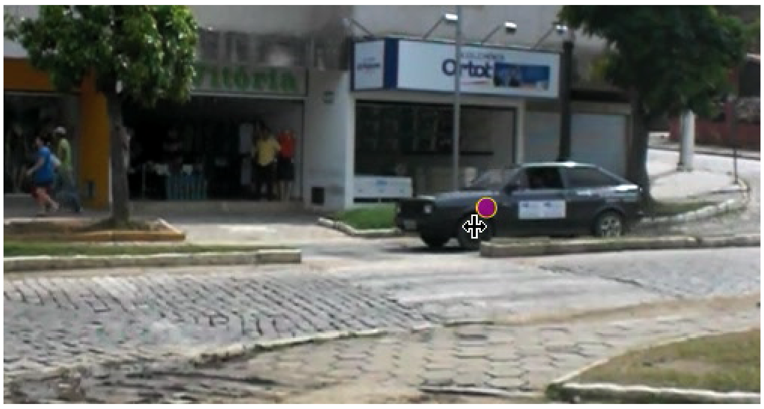

Figure 5.2: Motion gap velocity slider with resize cursor. In this example, moving the mouse cursor onto the left side of the position indicator forwards the video, while moving it onto the other side rewinds the video. DRAGON automatically enters and leaves the velocity slider mode, when the object enters or leaves a motion gap, respectively.

5.2 First user study

In the first user study on motion gaps four users were asked to drag objects over motion gaps using the three different techniques *time scale zero*, *time scale timer* and *velocity slider*.

5.2.1 Tasks

We have used two videos for the study

We have used two videos for our evaluation (see A—"Movies"). The first one initially shows a blue chair in the right half of the screen; then, a few moments later a green chair is put in left half of the screen. Again, a few seconds later, the blue chair is being removed and later on the green chair is being removed as well. The second video shows a scene of a busy street. There are two cars, a black one and a white one, entering the scene at the right side and leaving the scene on the left side of the screen. While the white car moves without stopping, the dark one has to wait for a pedestrian to cross the street, continuing its movement only a few seconds later. Observe, that the chair video has the larger motion gaps while the car stops for a few seconds only.

Users were asked to drag object over the motion gaps

For each of the techniques, the users were asked to drag the following objects from one side of the screen to the other: the green and blue chairs in the first video, and the dark car in the second video. Through a screen recording tool and experiment evaluation features in DRAGON, we were able to analyse the users movements and reactions using the different techniques that were presented in random order. Afterwards, users were asked whether they had any preference for one of the options, to comment their decisions, and to give further comments on both the motion gap problem and DRAGON in general.

5.2.2 Observations

Time scale zero

Time scale zero works well with short motion gaps only

Using the time scale zero technique, users react differently in the two videos. In the chair video, when dragging the blue or green chair over the motion gap, the video context changes a lot, as the other chair disappears without leaving a trace. This context change surprised all users. In the street scene video, the situation is different. The context remains more or less stable—only small objects like pedestrians in the background jump from one position to another, and the movement of the white car still appears quite continuous. In this video, one of the participants did not see the leap in time at all. Another user realised the context change but commented, that the overall movement is still visible and that, for this case, the time scale zero option is the most logical option, even though some parts of the video might be skipped. This user also found that the feature is easy to use and reacted always like expected.

Time scale timer

Time scale timer does not work well

Of all presented approaches, users had the most difficulty with the time scale timer feature. The time-out between the mouse movement and the object movement breaks the causal relationship between input and output, and, hence, the direct manipulation metaphor does not work anymore. Users often did not wait for the timer to decrease the time scale. For example, one participant tried to drag the green chair coming from the left (early trajectory vertices) and going to the right (later vertices) over the motion gap but always returned to vertices in front of the motion gap position before the timer could decrease the time scale. The return to these earlier trajectory vertices always stopped and reset the timer. As a consequence, she was not able to pass the gap. Another participant claimed that the system response time was very low (again, because of the missing causal relationship between input and output). In the questionnaire, three of four users stated that dragging using the time scale timer did not always react as expected and that the feature was hard to use.

Users try to apply more force when DRAGON does not react

When users managed to pass the gap using the feature, they had often moved the mouse cursor far away from the motion gap position. It seems that they tried to apply more "dragging force" by increasing the distance between the mouse cursor and object (like a rubber band). After leaving the motion gap, the spatio-temporal evaluation function then chooses a frame, where the object is far away from the motion gap position as well. This introduces a new problem—not only the frames of the motion gap are skipped, a lot of frames behind are skipped as well, making users miss even more of the video context.

Velocity slider

Although the velocity slider was the favourite technique for three of the four test users, and all users were able to pass the motion gap without losing the video context, they have encountered some difficulties. When they used the feature for the first time, most of them did not realise the mode change visualised through the change of the mouse cursor; the dragged object is the *locus of attention*, and, thus, users did not concentrate on the mouse cursor.

The change to velocity slider mode is not prominent enough

In the car example, one of the users released the mouse button after half a second of velocity sliding because she did neither see that the other objects were moving nor that the mouse cursor had changed. She thought, that the system was not reacting anymore (she had used the time scale zero feature before). The same situation arose in the chair video, with the difference, that, in the last instant before releasing the button, one of the chairs moved. Due to this visual feedback, the user understood the metaphor. Others had similar problems, though after a short period of training, all participants understood the metaphor and were able to easily scroll through the video. The one user who rejected the velocity slider feature stated that it was not logical to control the motion of other objects when the currently selected object is in a motion gap.

Users think that DRAGON does not react

As described above for the time scale timer, users tend to move the mouse cursor far away from the motion gap position, in case the selected object does not move. Depending on the length of a motion gap and the amount of motion in a scene, objects do not move even when *velocity sliding*. Furthermore, when users grab an object that currently is in a motion gap and the system activates the velocity slider mode, it seems, that the system does not react. This could be observed especially in the tasks that involved dragging the green chair as this scene has a very low amount of motion and a large motion gap. In such cases, users need an immediate visual feedback.

Again, users try to apply more force when DRAGON does not react

5.2.3 Conclusions

Time scale zero is reasonable

The time scale zero feature is a reasonable approach. User satisfaction depends on the video, i.e., on the amount of motion and the length of the motion gap.

Time scale timer is not reasonable

The time scale timer technique was rejected by most of the users and imposed the highest level of difficulty. Using a time-out to pass the motion gap has shown to be no reasonable approach. Users have to wait for a reaction of the system, and the drag interaction gets interrupted. Therefore, the technique will not be investigated any further.

The velocity slider needs more feedback

The velocity slider seems to be a reasonable approach but has a low level of feedback. For the next user study, we have added more visual feedback and an advanced control interface.

5.3 Online survey

An online survey determines users' opinions

In parallel to the first user study, we conducted the online survey from section 3.3—"Online survey". The part on motion gaps had two objectives—first, obtain information on whether users understand what happens when the time scale zero feature is used and non-selected objects jump from one position to another while dragging an object over a motion gap, and, second, get to know what users expect from DRAGON to do with motion gaps.

5.3.1 Questions

Users were asked to explain effect of time scale zero dragging

The object dragging technique was explained at the beginning of the survey. Prior to the study, we had recorded DRAGON sessions of the chair and car videos with activated time scale zero feature. In the first video, we dragged the green chair over the motion gap, and in the second video we did the same with the dark car. These videos were shown to the participants, who then were asked to explain why the blue chair disappeared and the white car jumped, respectively. On the next page of the survey, we explained the effect as follows.

> [. . .] When we drag the car and pass the point where it stops, the video jumps from the moment where the car stopped to the point where the car started to move again. [. . .] After putting the green chair on the floor, the blue chair is being removed. Only afterwards, the green chair is being removed as well.

The users were then asked to choose one or more of the following options.

> When I am dragging the dark car/the green chair...
>
> 1. . . . I want to see what happens to the white car or the blue chair [. . .],
> 2. . . . I am interested only in the dragged object and the video should follow my mouse cursor movement [. . .]
> 3. . . . I want to see the full motion and be able to control the playback speed and direction [. . .]
> 4. . . . the video should follow my mouse cursor movement until the point where the chair/car stops and let me decide what to do (ignore the pause, control the pause etc.).
> 5. I do not know.
> 6. I do not understand the problem.

5.3.2 Answers

Eight of 26, about 31%, users gave a correct explanation, the rest did not have any explication or gave incorrect answers, ranging from “the video might just be cut and therefore discontinuous” (four users thought there was a cut) over “the controls of the videos aren't very fine" to "[the] user used [the] timeline to skip".

From 21 participants, eight marked the first option (show gap content), six the second (jump the gap), nine the third one (control playback), and seven the fourth one (stop and choose). No participant did not answer and all of them understood the problem.

5.3.3 Results

There is demand for advanced motion gap handling

Taking a closer look at the marked answers, we observe that 14 participants chose either *show gap content* or *control playback* or both options, which are closely related (to be able to control the content, we need to show the content), while 7 users chose neither one of these two options. Counting up, 18 participants chose *show gap content, control playback* or *stop and choose* (figure 5.3).

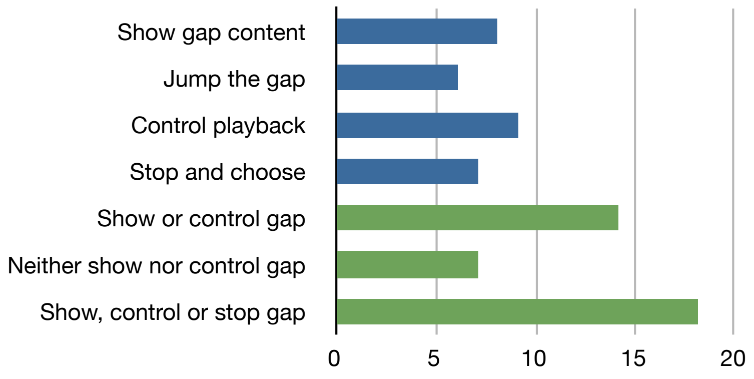

Figure 5.3: Online survey results for motion gap handling. 21 participants have been interviewed.

Motion gaps have to be visualised

The time scale zero option can confuse users. Only 31% of the participants had a correct explanation for the leap in time when an object passes a motion gap. Hence, motion gap handling should include a visualisation mechanism. In the second user study, we evaluate the previously mentioned advanced velocity slider that covers the three options *show, control* or *stop* the gap. According to the survey results, this should be a reasonable approach to handle the motion gap problem.

5.4 Second user study

For the second user study, we implemented an improved version of the *velocity slider* and another, previously not tested, technique called *pause loop*.

5.4.1 Improved velocity slider

Main problem was insufficient feedback

The main problem with the velocity slider tested in the first user study was the insufficient feedback—sometimes users did not realise the mode change from spatio-temporal tracking to velocity sliding. Furthermore, the online survey revealed that almost 70% of the participants get confused by leaps in time, and 85% of the participants want to either be informed of a motion gap or control it (when considering stopping the video as a kind of control). As an improvement, we propose an extended velocity slider (figure 5.4), that shows up as soon as an object enters a motion gap.

We introduce overlay crossers to improve feedback and control

We introduce the concept of *overlay crossers*. A crosser is a semi-transparent button that fires repeatedly while the mouse cursor hovers over it or an assigned area. Furthermore, while a crosser fires, it is highlighted to give immediate feedback. As we can see in figure 5.4, the visual part of the slider consists of three such crossers—one for forward scrolling, one for pausing and one for backward scrolling. The left crosser (depending on the direction of the object movement, the left crosser is the forward or backward crosser) fires while the mouse cursor is positioned left of the pause crosser. We hide the forward crosser when the object is at end of the gap to indicate that the object is about to move. The rewind crosser works analogously. The pause crosser stops the playback and fires when the other two crossers do not fire. Moreover, we attach an indicator to the pause crosser that keeps moving clockwise for forward scrolling and counter-clockwise for backward scrolling. Depending on the scrolling speed, the indicator moves faster or slower to give feedback on the scrubbing speed. When the pause crosser is active, i.e., playback is stopped, the indicator remains at a fixed position.

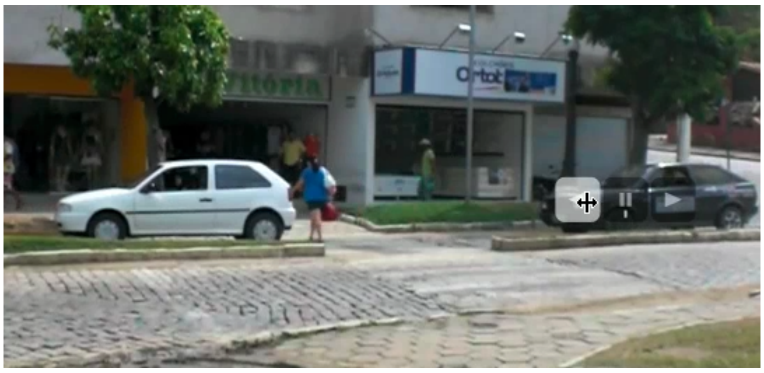

Figure 5.4: Improved velocity slider with three overlay crossers, that fire if the mouse hovers over them. Inside the pause crosser, an indicator turns clockwise or counter-clockwise at different speeds to represent the current scrolling direction and speed.

We expect to overcome the initial difficulties using this improved version of the velocity slider as we now offer the functions *jump the gap, control the gap* and *show the gap* through one interface component, while continuously giving feedback on whether a motion gap is entered and the video is being scrolled.

5.4.2 Pause loops

We introduce pause loops for extended motion gap control

Initially planned to be tested in the first user study, but unfortunately not implemented on time, we have evaluated another possible technique for the solution of the motion gap problem—so-called *pause loops*. The figures 5.5 and 5.6 show how pause loops work. When the user drags an object from right to left, and the objects enters a motion gap, a circle is shown on top the trajectory. From this moment, the angle between the mouse cursor and the imaginary line between the motion gap position on the trajectory and the centre of the circle is measured until the object leaves the motion gap. For a motion gap from frame f_s to f_r and an angle α (measured in clockwise direction), the next frame f_{next} is defined as

$$f_{next} = f_s + \left\lfloor \frac{\alpha}{360} \right\rfloor (f_r - f_s).$$

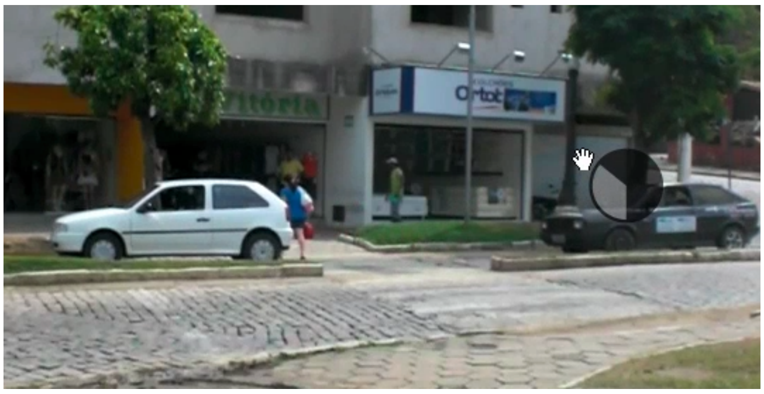

Figure 5.5: Pause loop example for the street video. When the dark car reaches its motion gap, DRAGON shows a circle with an arc inside. The user can control the size of the arc by dragging the mouse around the circle. In this example, the arc is at 135°, i.e., it has 37.5% of the circle's size. Thus, the video is at 37.5% of the motion gap.

PAUSE LOOP:

1. If the current frame is in a motion gap, create a pause loop.
2. Calculate the angle α between the mouse position and the line orthogonal to the trajectory and go to frame $f_{next} = f_s + \lfloor \frac{\alpha}{360} \rfloor (f_r - f_s)$.
3. Go to 1.

Implementation: *Pause loop*

Pause loops allow for forward and backward scrubbing with dynamic precision

Thus, dragging the mouse around the circle allows access to the different frames of the motion gap, where, for example, an angle of 180° scrolls the video to the middle of the gap. When we drag a full circle in clockwise direction, we jump the gap in forward direction (counter-clockwise dragging jumps the gap in backward direction). Observe,

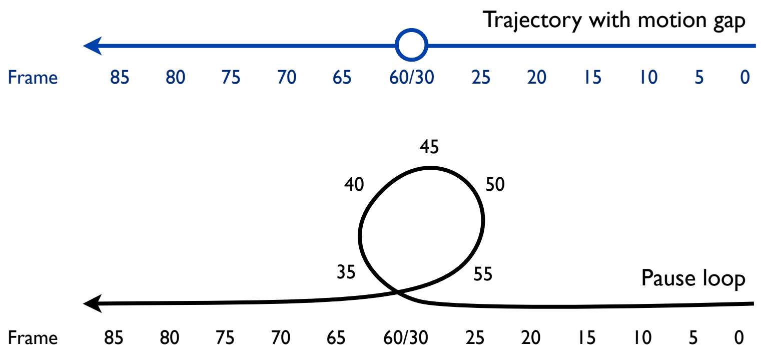
Trajectory with motion gap
Frame 85 80 75 70 65 60/30 25 20 15 10 5 0
45
40
50
35
55
Pause loop
Frame 85 80 75 70 65 60/30 25 20 15 10 5 0

Figure 5.6: Trajectory with motion gap and a mouse movement to jump the gap using a pause loop. After some training, users may develop an automatic movement in form of a mouse gesture to jump the gap.

that, though pause loops offer the same functionality as the velocity slider—the user can stop the playback during a motion gap and scroll in both forward and backward direction—loops enable the user to vary the scrubbing precision through the distance to the loop, and the time to pass a motion gap does not depend on its size. Moreover, we expect, that after some training the users will develop automatic behaviour in form of a mouse gesture to jump the motion gap.

5.4.3 Tasks

The second study compares velocity sliders and pause loops

We have used the same videos as in the first study. Users were asked to pass the motion gap and, additionally, find its start and end. Both velocity slider and pause loop allow users to exactly determine the start and end of a motion gap. We explained the problem of motion gaps in an introductory talk on DRAGON, though we did not explain the two different techniques to not influence these first-time users.

5.4.4 Observations

Improved velocity slider

Improved feedback facilitates use

Users performed well with the improved velocity slider, the visual feedback helped to understand the mode change, and users were able to find the the beginning and end of the motion gap, without any explanation on how to use the slider. Moreover, users were always aware of the scrolling speed and direction. As negative points, we have to mention, that when the crossers appeared, one user released the mouse and tried to click the buttons. In our implementation, we hide the crossers as soon as the mouse button is released. This surprised the user, though, after a few tries, she understood that hovering was sufficient to perform the action. Apparently, the button-like style creates an *affordance* (for affordances, see Norman [2002]). Moreover, as with the velocity slider evaluated in the first study, users cannot estimate the start and end of a motion gap. Furthermore, depending on the motion gap size, they want to use high scrolling speeds. Consequently, the mouse can be considerably far away from the motion gap position when normal tracking continues, leading to leaps in the playback.

Pause loops

Pause loops work well

User performance with pause loops was good. Users were able to find the start and end of the motion gap. When pause loops appeared for the first time, users tried to pass the gap by dragging the mouse cursor away from the gap position (as they would do with velocity sliders). Without further explanations, it took only a few seconds for them to have the idea of dragging the mouse around the circle. A drawback of pause loops became apparent, when one user released the mouse button in the moment that the circle appeared. This hid the pause loop. Then, when the user grabbed the object again, the video jumped to the motion gap start because the angle became $0°$. This effect happens every time a user selects an object that is in a motion gap.

5.4.5 Conclusions

Feedback for velocity slider can still be improved

The improved velocity slider is a good approach to the motion gap problem. The main drawback, that users may drag the mouse far away from the motion gap position to accelerate the playback, remains. We propose two extension for future work. First, the orthogonal distance to the trajectory (for a left to right movement, this would be the vertical distance) should be used to change scrolling speeds, and, second, the position indicator should be changed to a circle as used in pause loops. We expect that these extensions allow for better estimation of the necessary playback speed and avoid long distances between the mouse cursor and motion gap positions.

Pause loops are intuitive

Pause loops are a successful approach. Users intuitively began to drag the cursor around the circle. We believe that the immediate feedback visualised by the inner arc that follows the mouse cursor creates the affordance of dragging the mouse around the loop. The advantages over velocity sliders are a higher precision and that the start and end point are at the same position, thus avoiding leaps due to a high mouse distance after passing the gap. The main disadvantage shows up when users release the mouse button inside the gap because DRAGON jumps to the start of the motion gap when the object is grabbed again. For future work, we propose to not hide the pause loop as long as the selection does not change.

Table 5.1 shows a summary of all presented techniques.

	Timescale		Velocity slider		Pause loop
	Zero	Timer	Simple	Extended	
Control (in gap)	no	no	yes	yes	yes
Scroll	-	-	yes	yes	yes
Pause	-	-	yes	yes	yes
Pass gap	immediately	immediately	time depends on gap size		constant time
Feedback	-	-	yes, but low	yes	yes
Position	-	-	no	no	yes
Direction	-	-	no	indicator, crossers	yes
Advantages	Fast method		Full control	Full control Intuitive Improved feedback	Full control Intuitive High precision Automatizable
Disadvantages	No control Context loss	Time-out Context loss	Post-gap distance Low feedback	Post-gap distance No position feedback	Jumps to motion gap start when grabbed

Table 5.1: Summary of motion gap techniques. *Pass gap* refers to the necessary time to pass the motion gap. *Post-gap distance* refers to the possibility of having a huge mouse distance when leaving the motion gap, leading to a leap.

Chapter 6

Pendulum-like trajectories

"Do never admit there is a problem unless you have got a solution for it."

—Lea Nagel

Different trajectory vertices can be equally suitable for the mouse position matching

In chapter 4—"Object trajectories and visualisation", we have seen two examples of trajectories with edges; in the first one, a basketball repeatedly hits the ground, and, in the second one, billiard balls collide. In both cases, the direction of the object reverses, and the parts of the trajectory before and after the direction change are distinct enough to accurately access every single trajectory vertex by a corresponding mouse movement. However, when the angle between the parts of the trajectory preceding and following the edge decreases, the interaction may be too imprecise; it gets harder to select certain vertices in the neighbourhood of the outermost point. In extreme cases, two distinct trajectory points, one before and the other after the turning point, turn out to be equally suitable for the mouse position to trajectory matching (figure 6.1). Then, moving the mouse back and forth near a turning point leads to an ambiguous situation where the algorithm could select either a vertex on the subtrajectory that ends or on the subtrajectory that starts at the turning point. Hence, the video could scroll forward or backward.

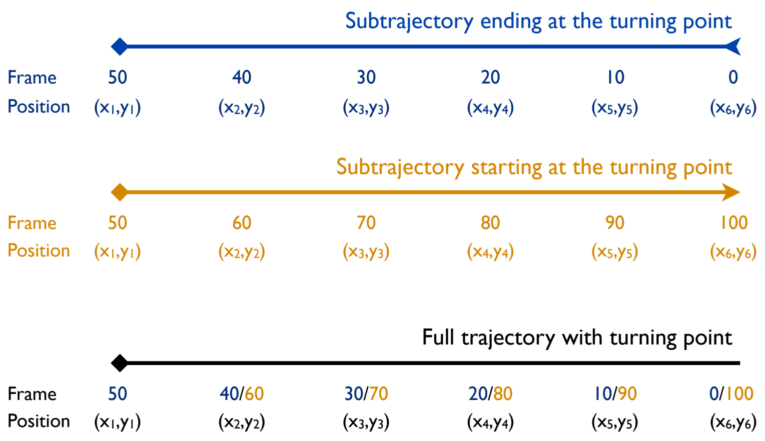
Subtrajectory ending at the turning point
Frame 50 40 30 20 10 0
Position (x1,y1) (x2,y2) (x3,y3) (x4,y4) (x5,y5) (x6,y6)
Subtrajectory starting at the turning point
Frame 50 60 70 80 90 100
Position (x1,y1) (x2,y2) (x3,y3) (x4,y4) (x5,y5) (x6,y6)
Full trajectory with turning point
Frame 50 40/60 30/70 20/80 10/90 0/100
Position (x1,y1) (x2,y2) (x3,y3) (x4,y4) (x5,y5) (x6,y6)

Figure 6.1: Illustration for pendulum-like trajectories. Reaching the turning point, the spatial and temporal distances for the trajectory vertices in forward and backward direction are equal. DRAGON has to resolve this ambiguity or allow the user to decide the scrubbing direction.

At turning points, dragging is ambiguous

In figure 6.1, the black trajectory, composed of the blue subtrajectory ending and the orange subtrajectory starting at frame 50, has an angle of size zero at the turning point. When the drag interaction reaches this point, DRAGON has to decide, or at least enable the user to decide, which part of the trajectory, in this case the blue or orange one, should be chosen for the next frames. As we can see, when the video is at the turning point, the distance, both spatial and temporal, is equal for both subtrajectories.

We do not know the user's intention

Furthermore, even if there was no computational ambiguity, dragging an object away from a turning point remains ambiguous in terms of determining the user's intention because we do not know whether the user wants to review how the object *reached* the point, or whether she wants to see how it *left* the point. In such situations, which arise frequently a reversing motion is involved[1], simple object dragging is modal; the same gesture has two different outcomes.

[1]Examples: *Music:* the motion of the hand of a guitar player; *sports:* tennis players running at the baseline; *transit:* cars parking in and out; *machines:* parts of motors

We refer to this kind of trajectory, in particular to the ambiguous parts, as *pendulum-like trajectories*. In this chapter, we present approaches to handle pendulum-like trajectories with and without additional user interaction. In the case of no additional user interaction, DRAGON resolves the ambiguity, i.e., eliminates the modality, by using a fixed scrubbing direction, while additional user interaction allows the user to decide. Moreover, we present a user study to investigate how users interact with our approaches, and to determine the related usability problems.

We introduce *pendulum-like trajectories*

6.1 Edge detection

First of all, as for motion gaps, we need a procedure to find the positions where interaction can complicate. For our evaluation, we have implemented a simple approach that has been sufficient for our test videos. $d(v_i, v_j)$ refers to the spatial distance between the trajectory vertices v_i and v_j. The algorithm is to be configured by d_{min}, a minimal distance, and α_{max}, a maximal angle. In our implementation, we have used $d_{min} = 30$ pixels and $\alpha_{max} = 20$ degrees.

We use a simple edge detection method

ToDo: der algorithmus ist ein bisschen anders... nochmal nachsehen

EDGE DETECTION:

1. Calculate the full trajectory.
2. Iterate over all vertices v_j
3. For each vertex v_j, find the two spatially closest vertices v_i and v_k with $i < j < k$ and $d(v_i, v_j) > d_{min}$ and $d(v_j, v_k > d_{min}$.
4. Mark v_j as an edge, if the angle α, spanned by v_i, v_j and v_k, is smaller than α_{max}.

Implementation: *Edge detection*

For more general videos, we expect other methods to achieve better results.

6.2 Approaches

DimP uses arc-length distance to fix temporal direction

As described before, either DRAGON decides the scrubbing direction at a turning point or we have to enable the user to do this. In *DimP*, [Dragicevic et al., 2008] use the arc-length distance to keep the scrubbing direction constant, i.e., in ambiguous situations, they do not change from forward to backward scrubbing or vice versa.

DRAGON uses trajectory segmentation at edges

In DRAGON, we propose to segment the trajectory based on the additional edge information obtained by the previously described method. Our implementation works as follows: For $k > 2$ subsequent edges $e_1, \ldots, e_k$, DRAGON automatically divides the full trajectory T into subtrajectories $T_0, \ldots, T_{k+1}$, where T_i ranges from e_i to e_{i+1}. For simplicity, we consider the end points of the trajectory as the edges e_0 and e_{k+1}. At every instant, only one of the subtrajectories is active while the others are disabled and are not being taken into account for the calculation of the next frame. At every frame, the object is on the currently active subtrajectory.

Figure 6.2 illustrates the trajectory segmentation for a damped oscillation of a basketball that repeatedly hits a ground. All pictures (a) to (h) show positions where the ball is either at the end of the trajectory or at an edge. Moreover, the straight arrows stand for active segments while the dotted arrows symbolise disabled segments. For clarity reasons, we omit the full trajectory and show only segments that either start or end at the corresponding turning point.

6.2.1 Fixed direction

Fixed direction switches active trajectory at turning points

Observe that, in figure 6.2, we switch the active segment at every turning point (pictures (b) to (g)). Fixing the temporal direction like this, enables us to perform an unambiguous drag interaction through the whole scene. Reaching at an edge e_i, DRAGON deactivates the previously active T_{i-1} and activates T_i (for backward scrubbing we deactivate T_i and activate T_{i-1}, accordingly). Hence, dragging the object away from the turning point scrolls the video in the last used temporal direction.

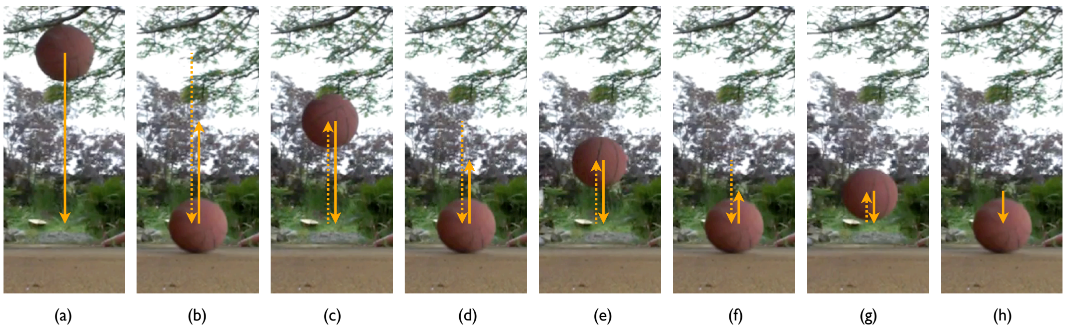

Figure 6.2: Example for the trajectory segmentation of a damped oscillation. When using *fixed direction*, DRAGON switches the active subtrajectory when reaching the turning points in pictures (b)-(g). Then, it takes into account only the points from the currently active trajectory (straight arrows) and ignores the points of the previously active trajectory (dotted arrows).

To avoid multiple switches close to edge positions, caused by noise or slow cursor motion, we do not switch between subtrajectories, until the temporal distance between the current trajectory vertex $v = (x, y, f)$ and the frame f_{edge} of the edge position (x_{e_i}, y_{e_i}) is greater than a predefined threshold value d_{fix}. Usually, a small value is sufficient to avoid switches and to accurately set playback directions (in our studies, we have used $d_{fix} = 5$ frames).

For subsequent trajectory switches, users must pass a threshold distance

6.2.2 User-decided direction

Fixed direction scrubbing is sufficient to scrub through the whole scene. However, when the task is to examine frames shortly before *and* shortly after the edge position, the d_{fix}-threshold can be larger than the number of frames that are to be reviewed. Then, users get stuck on one subtrajectory, and they have to drag the object farer than necessary to force the switch. This wastes the user's time. Furthermore, if the task requires to repeatedly review a certain number of frames either *only before* or *only after* a turning point, the user does not need the automatic segment switches. Reviewing the instant before a collision of two billiard balls would be an example for a task that becomes difficult and tedious and where it is desirable to temporarily fix one subtrajectory.

Automatic subtrajectory switches make some tasks difficult

It is necessary to allow users to choose the subtrajectory dynamically

Thus, for several applications, it is necessary to enable the user to decide whether the turning point should be passed or not, or in other words, whether DRAGON should switch the subtrajectory. For a good direct manipulation solution, the system has to provide at least three properties.

1. Usage of normal object dragging when the object is not at an edge.
2. Possibility to choose the temporal direction without interrupting the current drag interaction.
3. Immediate feedback on direction changes.

Overlay crossers satisfy the conditions for a direct manipulation interface

To achieve these goals, we have implemented overlay crossers similiar to the ones used for motion gaps (see section 5.4.1—"Improved velocity slider"). At each edge, we use two crossers, one for each adjacent subtrajectory. Again, we show the controls only when they are necessary; the two crossers appear or disappear when an object reaches or leaves an edge, respectively, thus satisfying property 1. Users select the direction by clicking a crosser or by hovering over it; the latter option satisfies property 2. Moreover, the highlighting of the currently active crosser satisfies 3.

Overlay crossers allow for automatic behaviour

As for pause loops (see section 5.4.2—"Pause loops"), we want users to be able to develop automatic behaviour in form of gestures. Therefore, we position the crossers such that a clockwise mouse motion at an edge activates the forward crosser, and that a counter-clockwise motion activates the backward crosser, respectively. Moreover, we divide the area behind the turning point (where the crossers are) in two parts, and associate one to the forward crosser and the other to the backward crosser. We have implemented the crossers to fire as soon as the mouse hovers over these areas. For illustration, see figure 6.3 which shows an example for a circular mouse movement to scrub through a scene in forward direction.

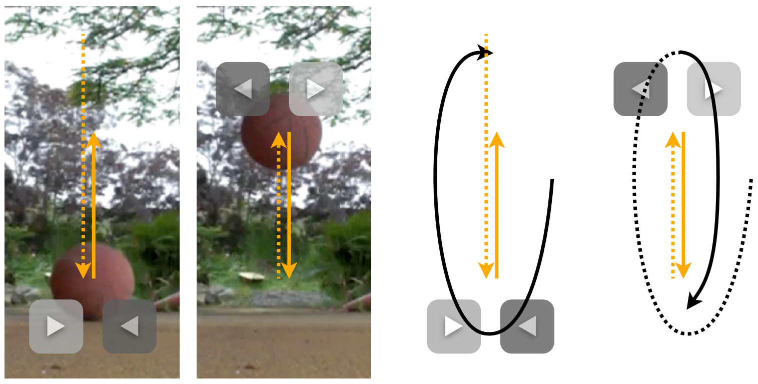

Figure 6.3: Overlay crossers and mouse gestures. The crossers are positioned such that a clockwise turn with the mouse around the edge switches to forward scrubbing and vice versa. On the left, we see two pictures from figure 6.2 with the corresponding crossers. On the right, we see the a mouse gesture to scrub through the scene.

6.2.3 Inertia scrubbing

We have implemented another option, that does not depend on trajectory segmentation. We call it *inertia scrubbing*. Through a minimal physics engine, DRAGON calculates the current object velocity during a drag interaction. If the user releases the object and the velocity is non-zero, the physics engine keeps scrolling the video in the last used temporal direction and decreases the object's velocity gradually. This enables users to "push" an object over an edge. Hence, inertia does not provide frame-exact control but enables users to scrub through the whole scene[2]. The Apple iPhone[3] uses a similar feature to scroll through contact lists.

Inertia allows for "pushing" objects

[2]Furthermore, inertia is useful for handling object occlusions. Due to the optical flow algorithm, users cannot drag objects along the full trajectory if the objects get occluded. With inertia, users can push objects such that the video scrub to the position where the object is no longer occluded. See Goldman et al. [2008] for a similar approach.

[3]http://www.apple.com/iphone

6.3 Preliminary study

A preliminary study gave initial insights

During implementation, we ran an informal preliminary study with two users, and asked them to scrub through a scene with a pendulum-like trajectory (figure 6.4) using fixed directions, overlay crossers, and inertia.

Inertia requires training

The inertia feature showed to be easy to understand but required some training. Both users had never used inertia features before (in computer interfaces), and they had difficulties with invoking the feature and estimating the necessary velocity to push the object to a certain point.

Fixed direction is intuitive but accidental subtrajectory switches occur

With fixed direction scrubbing, users were able to rapidly scrub through the scene, both in forward as in backward direction. At this point, we had not used the d_{fix}-threshold yet, and slow mouse movements lead to multiple, perceptually non-determistic, subtrajectory switches as described before in section 6.2.1.

Symbols on overlay crossers are confusing

Users understood the functionality of overlay crossers. However, we found out, that the visual feedback, i.e., highlighting the active crosser, is not sufficient to communicate the switch between two subtrajectories. Sometimes, users confused the crossers. It seems that, though the symbols for forward (▶) and backward direction (◀) are well-known from all sorts of audio and video playback devices, they are less intuitive when not appearing in left-to-right order (as in the leftmost picture of figure 6.3). Hence, highlighting the symbol only is not sufficient to represent the selected temporal direction.

6.3.1 Extended trajectory visualisation

We avoid the use of temporal information in DRAGON's

To overcome the usability breakdown introduced by this insufficient feedback, we first thought of labelling the crossers with "forward" and "backward". We have opted not to do that because, first, introducing a label to avoid misunderstandings usually does not make the interface more intuitive, and, second, we want to avoid that users have to concentrate on temporal information.

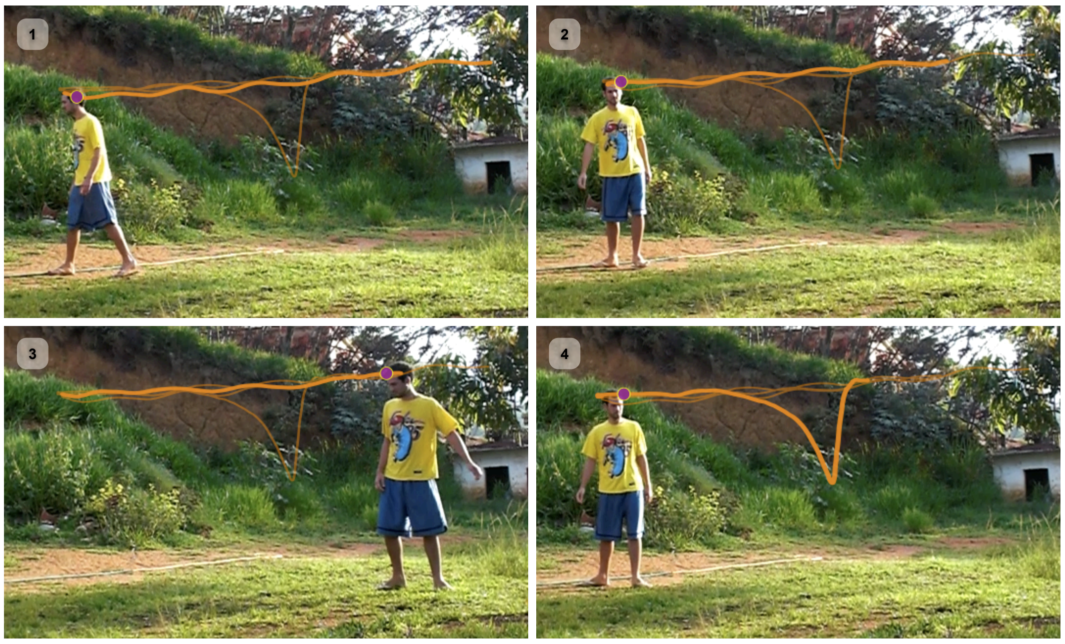

Figure 6.4: Visualisation of trajectory segmentation in the video that has been used for our user studies. The currently active subtrajectory has a thicker line and full opacity. After each subtrajectory switch, DRAGON automatically updates the video, thus providing immediate feedback. We expect that the feature is useful for fixed direction scrubbing and overlay crossers.

Visualisation highlights the active subtrajectory

Object dragging should use spatial information instead. Therefore, we have extended the trajectory visualisation (figure 6.4). Instead of drawing the entire trajectory in the same style, the extended visualisation uses the segmentation information and automatically highlights the active subtrajectory by decreasing the opacity and line width of the inactive segments. As soon as the active trajectory changes, DRAGON updates the view, and, hence, provides immediate feedback. We expect that this feedback avoids confusion and helps users to efficiently select the desired subtrajectory. Of course, this works only when the subtrajectories have, at least slightly, different shapes. When the segments are equally shaped and equally positioned, as in figure 6.1, the highlighting does not add any more information.

6.4 User study

A user study gave more insights

We have conducted a test session with five participants, three of them male, two female, all between 20 and 30 years old. They were asked to scrub through the video from figure 6.4 in both directions using inertia, fixed direction, and overlay crossers. For the latter two cases, the task was given both with the extended trajectory and without visualisation. We did not use the trajectory visualisation for the inertia tasks. We used a randomisation procedure to choose the order of the experiments. We were particularly interested in whether users would always be aware of the current subtrajectory, and whether they would be able to pass the turning points at different speeds, from slow to fast. We used a screen recording tool for later evaluation.

6.4.1 Observations

Inertia

Inertia is intuitive but requires training

Though all users understood the inertia feature, we observed that, frequently, they have had problems in estimating the necessary velocity to achieve a certain goal. Often, after releasing the object, either the video scrolled so fast, that the object ended up far beyond the turning point, or users applied insufficient force to push it over the edge at all. Furthermore, users often stopped the mouse movement first and only then released the mouse button. Thus, the velocity was zero, and inertia did not work. However, the difficulty might depend on the input device. During the implementation of DRAGON, we used trackpads, mouses, and stylus pens (on graphics tablets). In our opinion, among these, pens are the easiest input devices to work with inertia. Nevertheless, in this work, we focused on mouse input as it is the most commonly used device for average users, and we did not conduct a formal study to compare the usability of input devices regarding inertia tasks. Although, with the rising number of touch-sensitive devices and the success of the iPhone, we expect inertia to be an appropriate and intuitive method for the future.

Fixed direction

With fixed direction scrubbing, users scrubbed through the scene without notable difficulties. They were able to scrub both at different speeds and in different directions. Participants intuitively switched the current subtrajectories. However, as they did not try to repeatedly change the direction very close to the edges and scrubbed over longer intervals, they always stayed on a particular subtrajectory for some frames, and, therefore, they did not perceive the d_{fix}-threshold. This matches our expectation, that the threshold has no negative impact on reviewing tasks that do not involve frame-by-frame analysis of the video around the turning point position.

Fixed direction scrubbing is an intuitive and reasonable approach

The extended trajectory visualisation changed the user's performance only slightly. However, though we have not asked users explicitly, we observed, that the visualisation encouraged participants to examine the closer area around a turning point, probably to learn when a trajectory switch would occur. Moreover, users did not comment the d_{fix}-threshold; it was intuitively clear—without any explanations—that, to reverse the scrubbing direction, it is necessary to leave the turning point first and to return to it subsequently.

Fixed direction scrubbing works well without extended visualisation

Overlay crossers

For overlay crossers, the study confirmed that the symbols for forward (▶) and backward (◀) are not sufficient to communicate the temporal directions to the user. Without the extended trajectory visualisation, scrubbing over the turning points reduced to a trial and error task. Moreover, some users showed automatic behaviour different from the one we tried to develop. Instead of using circular turns, they tried to keep the direction fixed by staying on the same side of the trajectory (figure 6.5).

The symbols ▶ and ◀ are not sufficient, and participants invent other possible technique

With enabled trajectory visualisation, the performance increased dramatically. The feedback helped users to choose the direction more efficiently. When they chose the wrong

Extended trajectory increases usability and understanding

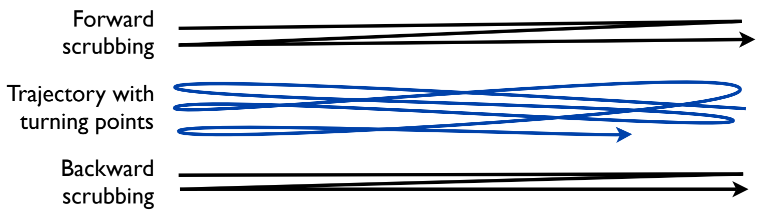

Figure 6.5: Possible technique for scrubbing over turning points on pendulum-like trajectories (when not using fixed directions). Movements on one side of the trajectory cause forward scrubbing at the turning points, movements on the other side cause backward scrubbing, respectively.

crosser, they were able to correct the error and choose the desired scrubbing direction immediately. Though the overall scrubbing speed was lower than with fixed directions, participants easily scrubbed through the scene. Again, the visualisation encouraged them to examine the turning points more closely.

Accidental trajectory switches may occur due to movements near turning points

We observed a usability problem in the current implementation of the overlay crossers. Previously, we have described that the crossers fire as soon as the mouse hovers over a designated area. We have designed these areas such that they start at the turning point and are divided through the imaginary line that passes between the two crossers. Sometimes, users crossed one area to select a direction, and when returning to the trajectory, they unintentionally crossed the opposite area. To avoid this, we propose to add neutral areas both between the two original areas and to the turning point (figure 6.6). We expect to reduce the number of accidental direction changes with this technique.

6.4.2 Results

The different techniques have advantages and disadvantages

Fixed direction scrubbing is the easiest way to scrub along a pendulum-like trajectory. For most purposes, the technique is sufficient. Moreover, it requires no training and trajectory visualisation. For special purposes like reviewing a certain subtrajectory for several times or examining frames nearby a turning point, overlay crossers are

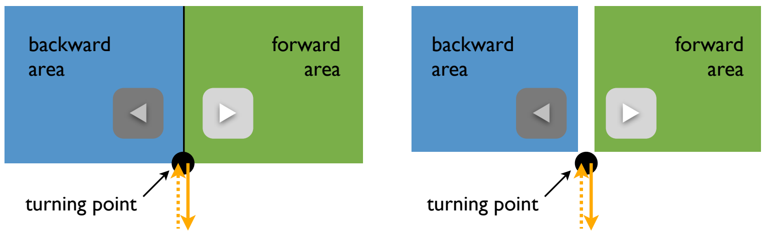

Figure 6.6: On the left side, the original implementation of overlay crossers for pendulum-like trajectories, users may accidentally change the scrubbing direction when dragging on a path close to the turning point. We expect to overcome this problem with neutral areas as shown on the right side.

a suitable approach. However, the labelling is not sufficient to communicate the scrubbing direction. Extended trajectory visualisation overcomes this problem, if adjacent subtrajectories have different shapes. Moreover, we expect that, after some training (which was not possible in this study), users will develop automatic behaviour regarding the clockwise/counter-clockwise turns for forward and backward scrubbing. Inertia is intuitive but requires training to estimate the correct velocity. However, we believe that it is a reasonable approach for navigation task that do not require frame-accuracy around the turning points.

Chapter 7

Summary and future work

""Forty-two," said Deep Thought, with infinite majesty and calm."

—The Hitchhiker's Guide to the Galaxy

7.1 Summary and contributions

Many groups work on improving video navigation

The importance of digital video is constantly rising. Timeline-based systems are suitable for macro navigation tasks. However, for in-scene navigation other techniques promise better performance. In chapter 2—"Related work", we have seen several approaches to improve video navigation, for example, by introducing dynamic sliders [Ramos and Balakrishnan, 2003] or direct manipulation techniques [Dragicevic et al., 2008], [Goldman et al., 2006], [Kimber et al., 2-5 July 2007].

Direct manipulation bridges gulfs of execution and evaluation

In 3—"Direct manipulation for video navigation", our coverage based on the seven stages of action has shown the disadvantages of the timeline based system regarding in-scene navigation. Gulfs of execution and evaluation are present, and there is demand for better interfaces. DRAGON's direct manipulation technique is a reasonable approach to facilitate video navigation, and it is capable of bridging the gulfs by introducing a strong natural mapping between mouse and object movement, at the same time avoiding the spatial separation of content and controls.

Some motion classes require trajectory visualisation

For simple trajectories and tasks, the spatio-temporal evaluation function used in DRAGON provides a good conceptual model. However, certain types of trajectories require special handling. In chapter 4—"Object trajectories and visualisation", we have seen examples of different types of trajectories. Straight lines and edges are the easiest cases where the different trajectory visualisations *full trajectory* and *arrows* have only a small impact on the user performance. In the case of curves and waves, tasks become more difficult if the visual guidance decreases. Finally, we have seen that *density dots* and *variable width trajectories* are capable of transmitting information about object direction and velocity changes.

We handle motion gaps with velocity sliders and pause loops

In chapter 5—"Motion gaps", we considered *motion gaps* that occur when objects pause or move very slow for a certain amount of time. In such cases, it is necessary to suspend the spatio-temporal mapping and to use advanced interaction techniques. A simple detection procedure allows for estimating hard motion gaps and the implementation of extended features. We have investigated several approaches. First, *time scale zero* jumps the motion gap and only works well for small motion gaps, while the use of a *time scale timer* is no good option. *Velocity sliders* and *pause loops* provide more control, and they avoid context losses in form of leaps in the playback, thus, decreasing the cognitive load.

DRAGON disambiguates pendulum-like trajectories with overlay crossers and fixed directions

Chapter 6—"Pendulum-like trajectories" covered trajectories of objects that move along similarly shaped and positioned subtrajectories at different moments in time. In such cases, dragging becomes ambiguous at the turning points between one subtrajectory and another. Therefore, we have introduced trajectory segmentation that constrains the mouse trajectory matching. This allowed us to provide advanced features. Our user studies have shown that, depending on the tasks, *fixed direction scrubbing* and *overlay crossers* are reasonable approaches to eliminate ambiguity. However, simple labelling of overlay crossers is not sufficient to communicate the temporal direction, and user tests confirmed that the extended trajectory visualisation is suitable to avoid user confusion.

Study participants find DRAGON intuitive and easy to use

Throughout all user studies, most participants found object dragging easy, intuitive, and quick to perform navigation tasks. If it was available in their favourite video players, many would prefer object dragging to timeline interactions. Our observations were positive as, usually, all participants understood the metaphor and were able to successfully accomplish all tasks.

7.2 Future work

There are still many open questions about direct manipulation for video navigation. At this point, we want to give some ideas for possible future work.

7.2.1 Pause loops for pendulum-like trajectories

Pause loops may resolve problems of pendulum-like trajectories

We propose to investigate the use of pause loops at the turning points of pendulum-like trajectories. Often, we encounter a combination of motion gaps and turning points, because, when objects pass a turning point, they slow down, stop, and speed up again. Moreover, pause loops have shown to be intuitive. We even expect that the extended trajectory visualisation could become redundant as the pause loop changes its appearance during interaction. Hence, the subtrajectory switch would automatically be made visible.

7.2.2 On-demand pause loops and overlay crossers

Pause loops and overlay crosser are not always necessary

In the current implementation, once the pause loop or overlay crosser features are active, one has to use them to perform navigation tasks. For motion gaps, for example, it is not always necessary to stop the video because it might increase the task completion time. There is necessity for a heuristic to determine whether a user wants to examine the motion gap or just jump it.

The same holds for overlay crossers for pendulum-like trajectories. DRAGON could use increased neutral areas (see figure 6.6) and use fixed direction scrubbing as default. We need a heuristic to distinguish between accidental and deliberate subtrajectory switches.

7.2.3 Subtrajectory selection for pendulum-like trajectories

Currently, arbitrary subtrajectory switches are not possible

The current implementation of our trajectory segmentation approach for pendulum-like trajectories does not allow for arbitrary switches to disabled subtrajectories; switches are possible at the turning points only. When there are many segments, it can be useful to provide additional techniques to enable users to skip a certain number of subtrajectories.

7.2.4 Pen-based and multi-touch interaction

Pen-based and multi-touch interaction allow for advanced interaction

With stylus pens or multi-touch, advanced interaction techniques gestures could be implemented easily. With multi-touch, we could constrain the motion of certain areas. Moreover, pens allow for another possible solution to the motion gap problem. Increasing the pen pressure could decrease the time scale.

7.2.5 Object recognition

Object recognition allows for object highlighting

As DRAGON uses optical flow to track and identify movement, it does not recognize the objects themselves in the video stream but only the movement of the corresponding pixels. If an algorithm for object detection was used, a lot of further questions could be researched. For example, the highlighting of moveable objects by mouse-over effects like semi-transparent overlays or object shapes could be useful. Furthermore, objects could be entered in a database and recognized in succeeding frames even when they disappear in the meantime or get crossed by other objects.

7.2.6 First-person dragging

Relative instead of absolute mouse movement can be interesting

It is hard to interact with small objects on curved trajectories as small mouse movement require a much higher precision and more attention than large-scale movements. We propose to examine the use of what we call *first-person dragging*. This technique does not use a spatio-temporal mapping against the absolute mouse position but uses the relative change of the mouse position, i.e., when the user moves the mouse, the object moves in the direction that best fits the mouse direction.

7.2.7 Real-world studies

Larger field studies have to be conducted

Due to computational and space complexity only short video clips have been used for the user studies. Of course, these videos do not represent reality. In our controlled experiments we made sure, that all necessary functions would work, and users had to accomplish clearly defined tasks. Thus, we have not evaluated how a real user in his real work or home environment would use the object dragging technique. Especially in the case of motion gaps and pendulum-like trajectories, further long-term field studies with continuous user feedback have to be conducted.

Appendix A

Movies

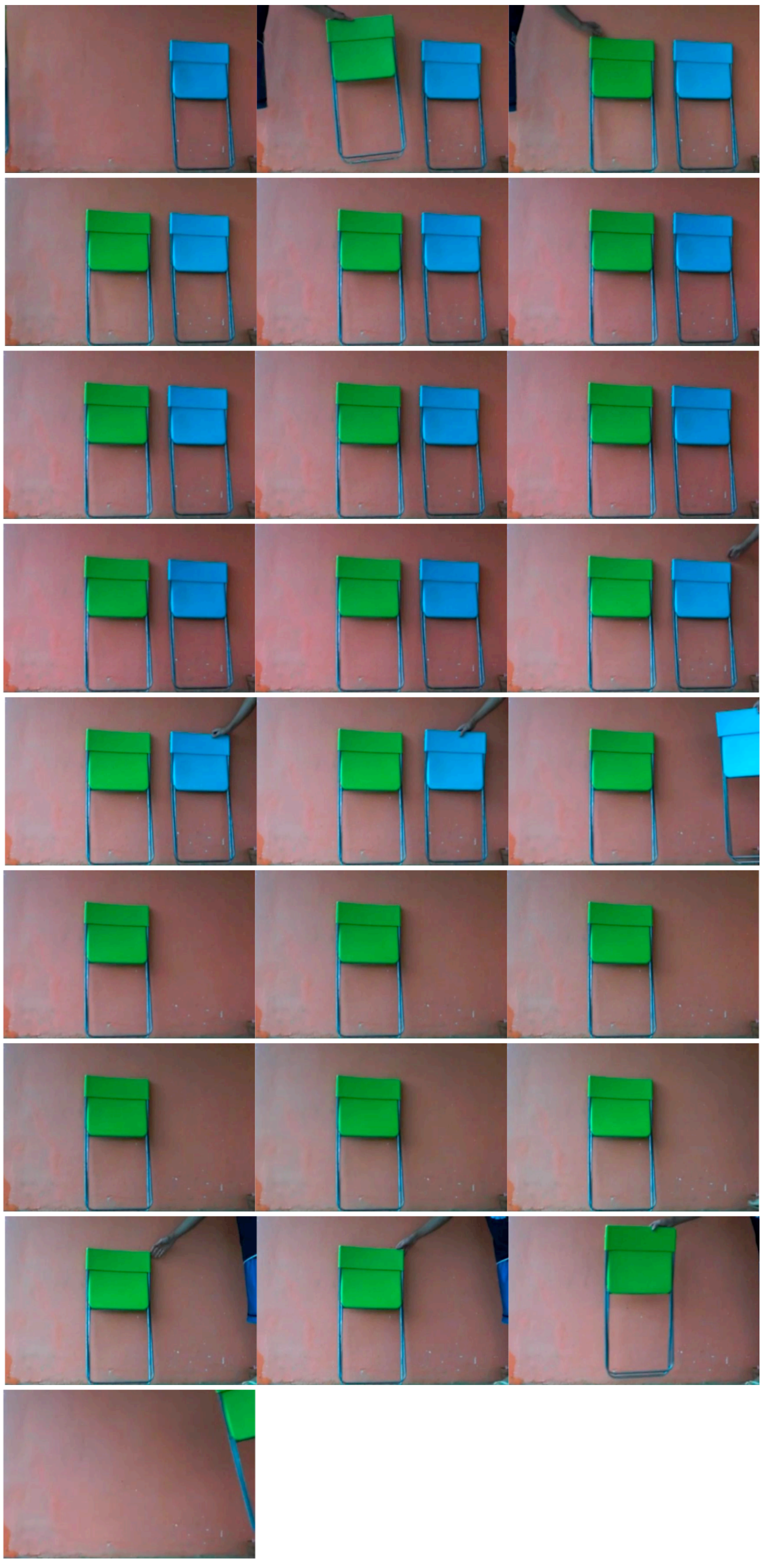

Figure A.1: Chairs video used for motion gap studies.

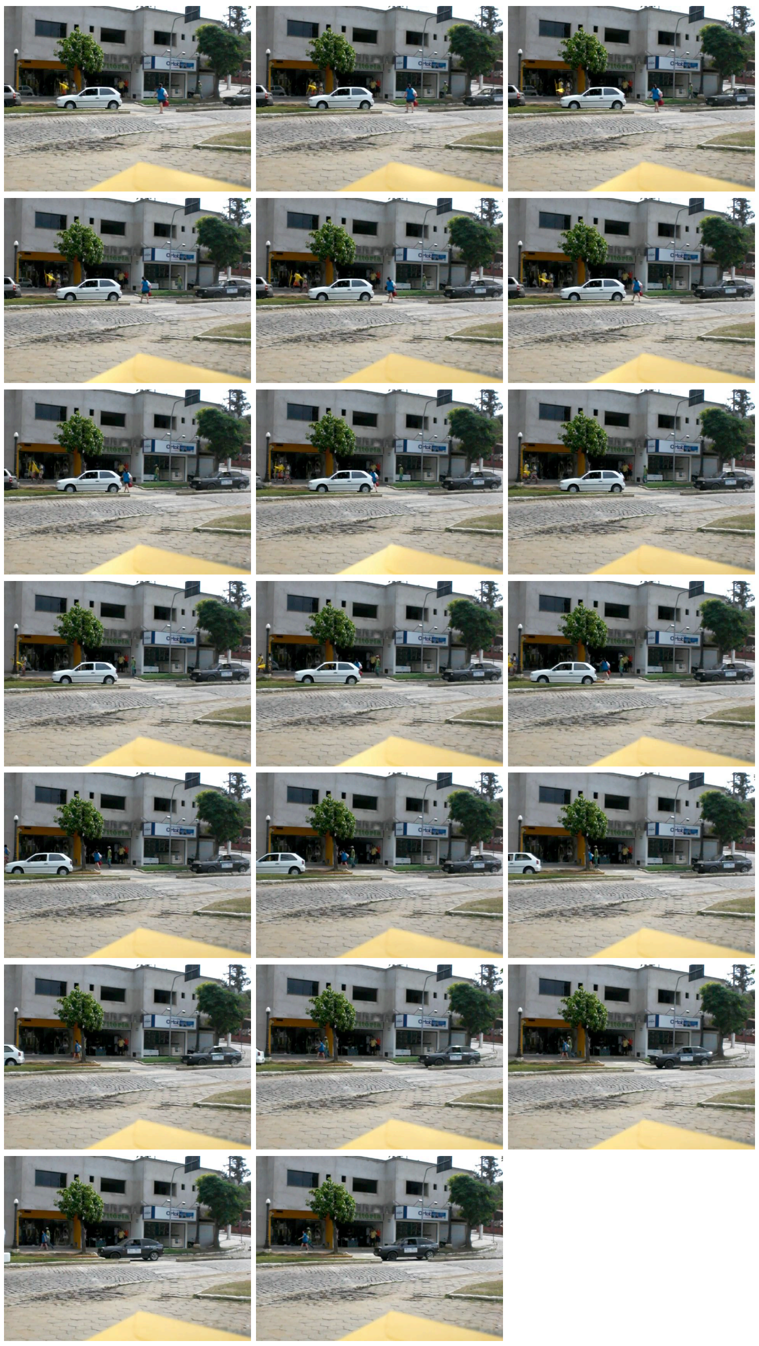

Figure A.2: Street scene used for motion gap studies.

Bibliography

Michel Beaudouin-Lafon. Instrumental interaction: an interaction model for designing post-wimp user interfaces. In *CHI*, pages 446–453, 2000.

Shih-Fu Chang, William Chen, Horace J. Meng, Hari Sundaram, and Di Zhong. Videoq: an automated content based video search system using visual cues. In *MULTIMEDIA '97: Proceedings of the fifth ACM international conference on Multimedia*, pages 313–324, New York, NY, USA, 1997. ACM. ISBN 0-89791-991-2. doi: http://doi.acm.org/10.1145/266180.266382.

Jonathan D. Courtney. Automatic, object-based indexing for assisted analysis of video data. In *MULTIMEDIA '96: Proceedings of the fourth ACM international conference on Multimedia*, pages 423–424, New York, NY, USA, 1996. ACM. ISBN 0-89791-871-1. doi: http://doi.acm.org/10.1145/244130.244452.

R. Cucchiara, C. Grana, and G. Tardini. Track-based and object-based occlusion for people tracking refinement in indoor surveillance. In *VSSN '04: Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, pages 81–87, New York, NY, USA, 2004. ACM. ISBN 1-58113-934-9. doi: http://doi.acm.org/10.1145/1026799.1026814.

Pierre Dragicevic, Gonzalo Ramos, Jacobo Bibliowitcz, Derek Nowrouzezahrai, Ravin Balakrishnan, and Karan Singh. Video browsing by direct manipulation. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 237–246, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-011-1. doi: http://doi.acm.org/10.1145/1357054.1357096.

Dan B Goldman, Brian Curless, David Salesin, and Steven M. Seitz. Schematic storyboarding for video visualization and editing. *ACM Trans. Graph.*, 25(3):862–871, 2006. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1141911.1141967.

Dan B. Goldman, Chris Gonterman, Brian Curless, David Salesin, and Steven M. Seitz. Video object annotation, navigation, and composition. In *UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 3–12, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-975-3. doi: http://doi.acm.org/10.1145/1449715.1449719.

Edwin L. Hutchins, James D. Hollan, and Donald A. Norman. Direct manipulation interfaces. In *HUMAN-COMPUTER INTERACTION*, University of California, San Diego, 1985. Lawrance Erlbaum Associates, Inc.

Thorsten Karrer, Malte Weiss, Eric Lee, and Jan Borchers. Dragon: A direct manipulation interface for frame-accurate in-scene video navigation. In *Proceedings of the CHI 2008 Conference on Human Factors in Computing Systems*, Florence, Italy, April 2008. ACM Press. Best CHI 2008 Note Award.

Don Kimber, Tony Dunnigan, Andreas Girgensohn, Frank Shipman, Thea Turner, and Tao Yang. Trailblazing: Video playback control by direct object manipulation. *Multimedia and Expo, 2007 IEEE International Conference on*, pages 1015–1018, 2-5 July 2007. doi: 10.1109/ICME.2007.4284825.

Jakob Nielsen and Thomas K. Landauer. A mathematical model of the finding of usability problems. In *CHI '93: Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, pages 206–213, New York, NY, USA, 1993. ACM. ISBN 0-89791-575-5. doi: http://doi.acm.org/10.1145/169059.169166.

Donald A. Norman. *The design of everyday things*. Basic Books, [New York], 1. basic paperback ed., [nachdr.] edition, 2002. ISBN 0-465-06710-7.

Gonzalo Ramos and Ravin Balakrishnan. Fluid interaction techniques for the control and annotation of digital video. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 105–114, New York, NY, USA, 2003. ACM. ISBN 1-58113-636-6. doi: http://doi.acm.org/10.1145/964696.964708.

Jef Raskin. *The Humane Interface: New Directions for Designing Interactive Systems*. Addison-Wesley Professional, 2000.

B. Shneiderman. Direct manipulation: A step beyond programming languages. pages 461–467, 1987.

Chih-Wen Su, Hong-Yuan Mark Liao, and Kuo-Chin Fan. A motion-flow-based fast video retrieval system. In *MIR '05: Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, pages 105–112, New York, NY, USA, 2005. ACM. ISBN 1-59593-244-5. doi: http://doi.acm.org/10.1145/1101826.1101845.

Malte Weiss. *Depth-Discontinuity Preserving Optical Flow Using Time-Multiplexed Illumination*. 2007.

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

A. Z. Zivotofsky. The duncker illusion: intersubject variability, brief exposure, and the role of eye movements in its generation. *Investigative Ophthalmology and Visual Science*, 45:2867–2872, 2004.

Index

Typeset February 3, 2009