RWTH AACHEN UNIVERSITY

Object Selection and Adaptive Trajectories in DRAGON

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

by
Alisa Novosad

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Bastian Leibe

Registration date: Oct 10th, 2011
Submission date: Feb 6th, 2012

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, February 2012
Alisa Novosad

Contents

List of Figures

Abstract

Although the direct manipulation video navigation is a quite young research area, various independent software products such as DRAGable Object Navigation (DRAGON) and Direct manipulation Player (DimP) have already been offered for this purpose. Several reports on the similar approaches have been also published as a part of a research project (Interactive Video Object Annotation).

Considering the innovative nature of specific solutions and products in this area, a common weakness has been observed. As a matter of fact, the current direct manipulation video navigation systems still do not offer the possibility to navigate a specific part of a certain object present in a video such as a wheel of a car and not the whole car. Instead, the user was forced to find out on his own, which approach is appropriate to make a specific item in the video navigate along its movement trajectory.

The main purpose of this work is to eliminate all possible misunderstandings between the user and the direct manipulation video navigation system in terms of object selection and to adapt the system to the user's mental model as much as possible by implementing three new selection techniques for this purpose: an area selection and two realizations of a point selection with adaptive trajectories (pressure and dragging velocity). It is also important to find out whether any of the mentioned selection techniques has the potential to be used as a multipurpose.

Überblick

Obwohl das ”direct manipulation video navigation” (DMVN) ein relativ neues Forschungsgebiet ist, haben sich mehrere voneinander unabhängige Softwareprogramme, wie DRAGable Object Navigation (DRAGON) und Direct manipulation Player (DimP) in diesem Gebiet etabliert. Einige Veröffentlichungen bezüglich der ähnlichen Methoden und Techniken sind im Rahmen des Forschungsprojekts ”Interactive Video Object Annotation” erschienen.

Aufgrund der begrenzten Erfahrungen in diesem neuen innovativen Gebiet wurden mehrere Schwachstellen in den Softwareprogrammen festgestellt. Dem Anwender wird im Rahmen der aktuellen DMVN Systemen beispielsweise keine Möglichkeit eingeräumt, bestimmte Teile bestimmter Objekte in einem Video auszuwählen und navigieren (z.B. ein Autorad und nicht gleich das ganze Auto). Bisher war es Aufgabe des Anwenders, sich selbst um die Vorgehensweise zu kümmern, die die Navigation eines bestimmten Objekts entlang seiner Trajektorie erlaubt.

Ziel dieser Arbeit ist daher, alle möglichen Fehleinschätzungen der Anwenderbedürfnisse im DMVN System im Bezug auf die Objektselektion zu beseitigen und das System den Vorstellungen und Erwartungen des Anwenders dadurch anzupassen, dass drei neue Selektionstechniken implementiert werden. Es handelt sich hierbei um die Flächenselektion (Area Selection) und um die zwei Varianten der Punktselektion (Pressure und Dragging Velocity Point Selection). Es ist auch wichtig zu untersuchen, welche dieser Selektionstechniken sich unter den Anwendern durchsetzten wird.

Acknowledgements

This thesis would not have been possible without the help, support and guidance of my principal supervisor, Prof. Dr. Jan Borches, not to mention his advice and unsurpassed knowledge of human computer interaction. The good advice, support and assistance of my second supervisor, Prof. Dr. Bastian Leibe, has been invaluable on the academic and technical level, for which I am extremely grateful.

I am heartily thankful to my supervisor Thorsten Karrer and to Moritz Wittenhagen, whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of the subject. I am also very thankful for their excellent examples and suggestions for improvement.

My sincere thanks go to my fellow postgraduate students in the M.Sc. course "Software system engineering" for their friendship, help and valuable feedbacks during my stay at Aachen.

The members of the Media Computing Group have contributed immensely to my professional time at the RWTH Aachen University. The group has always been a source of not only good advices but also friendships and collaboration.

The user study performed and its results evaluated and discussed in this work would not have been possible without the participation and support of my fellow students and my dorm floor neighbors.

Lastly, I would like to thank my family for all their love and encouragement. For my parents Jurij and Natalija Novosad who raised me with a love of science and supported me in all my pursuits. And most of all for my loving, supportive, encouraging, and patient fiancé Salaheddine Rezgui whose faithful support during all stages of this thesis is very appreciated.

Thank you.

Conventions

Throughout this thesis we use the following conventions.

Text conventions

Definitions of technical terms or short excursus are set off in coloured 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

Source code and implementation symbols are written in typewriter-style text.

```
myClass
```

The whole thesis is written in Canadian English.

Chapter 1

Introduction

" Unless you are staying in an underground cave for more than a year without an internet connection, there's a healthy chance that you have at least watched, if not downloaded, an online video on Youtube or Google Videos."

— Amit Agarwal, creator of Digital Inspiration blog

How many photo manipulation software tools could you remember at the moment? Adobe Photoshop, Adobe Illustrator, GIMP, Paint , etc. – image processing has long become an ordinary task for an average user. In comparison, video manipulation software is often more difficult to use and it is only spread among professionals. Simplified versions of such programs might be easier to use, they have only limited functionality though.

The amount of video sharing websites such as YouTube[1] and Vimeo[2] is increasing. Over 140 Million videos have been counted on Youtube alone in August 2010. According to an on-line article,[3] twenty-four hours of video are uploaded every minute. Considering the content, one can discover that the overwhelming majority of these videos

[1] http://www.youtube.com/
[2] http://www.vimeo.com/
[3] "The Business Side of YouTube" by Barry Silverstein

are homemade. Thus, users have to handle video navigation and editing with increasing regularity. The navigation tools, used in the current video editing programs will be illustrated in the next section.

1.1 State of the art in video navigation

Digital video navigation Regardless of whether the user prefers Windows, Linux, or Mac operation system, media-players are mostly constructed in a quite similar way (Figure 1.1). Their distinctive element is a horizontal time-slider.

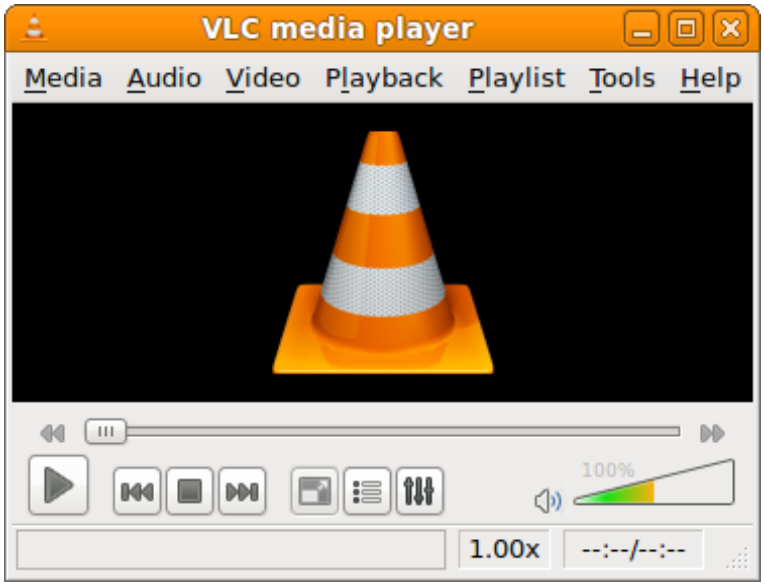


Figure 1.1: User interface of VLC player.

Definition:
Time-slider

TIME-SLIDER:
Time-slider is a time-based track bar, every position of which is mapped to the video timeline. One can start a playback from a certain moment of time by moving a slider to the new position.

Although this interface feature is well-known and widely used, following arguments have been found along its application. The key-disadvantage of the time-slider is associated with a non-linear time-position dependency of the objects in the video. The time-slider is linearly mapped to the absolute time of the movie. The objects, however, don't particularly move with a constant speed: they may accelerate or decelerate at times. This particular situation is represented on the figure 1.2. The car's trajectory is cut in uniform pieces, whose time values are shown on the time-slider below. It's obvious, that the car slows down when crossing the intersection, because the timeframe between points three and six is larger, compared to the other intervals on the slider.

Most difficulties in using time slider arise since it is linearly mapped to the video timeline. The objects velocity however is not linear and is not in proportion to the time slider's position

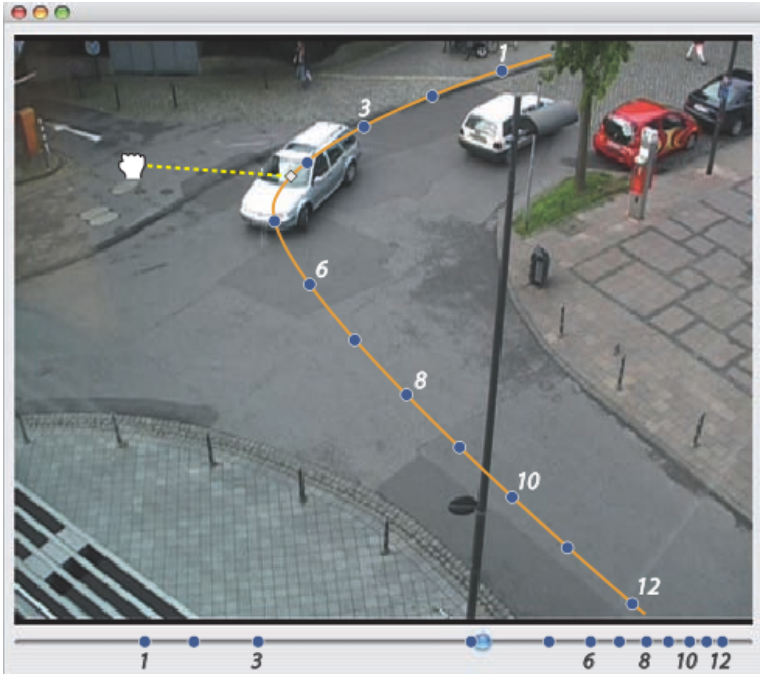


Figure 1.2: DRAGON: The image illustrates a non-linear position-time relation in the video.

Another disadvantage consists in the significant change of the time-slider's resolution by navigating through diverse videos (Figure 1.3). While the slider's length stays unchanged, it becomes more difficult to navigate through a several hours long movie rather then a three minute long video clip.

The resolution of the time slider strongly depends on the video size

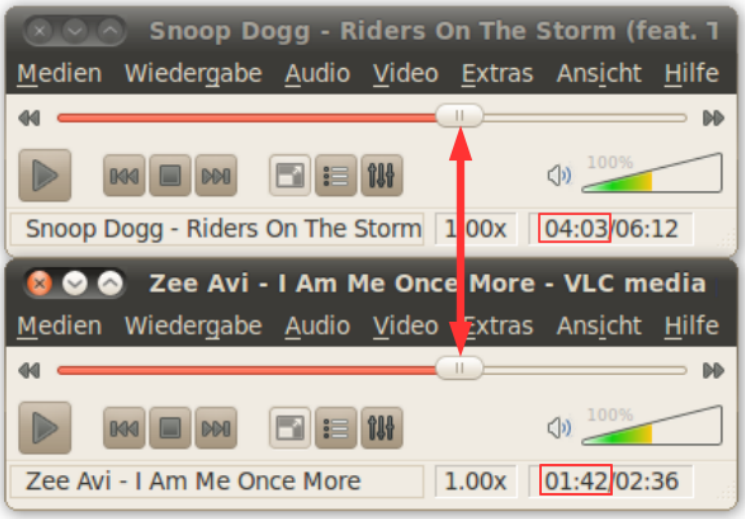


Figure 1.3: Play back of two different media files. A better resolution of the bottom time-slider allows to navigate a file more precisely.

Finally, despite the overwhelming popularity of the time-slider, user is constantly concentrated on controlling this interface feature, not the object's behavior and the time object is visible.

Alphaslider invented in 1994 was supposed to meet the notion of direct manipulation within the concept of the usual slider

Three decades ago the question about replacement of a complex user interface by direct manipulation of the object of interest was already arisen. Ben Shneiderman [1983], an american professor in field of human-computer interaction, has first defined the notion of direct manipulation system[4]. Alphaslider (Figure 2.11) was one of the first programs supposed to fulfill this notion for scanning and selection of large lists in graphical user interfaces [Shneiderman and Ahlberg [1994]].

It took a while until this issue was investigated for video navigation as well. This field of direct manipulation will be discovered in the following paragraph.

[4] more details in chapter 2.2—"Overview of existing selection techniques in direct manipulation video navigation (DMVN)"

Direct manipulation video navigation (DMVN) One of the first time-slider extensions appeared in 2003. Authors Ramos and Balakrishnan [2003] improved a standard video navigation by adapting the local slider resolution to the value of the pressure, exerted by the user on the attached touchpad.

Make use of the pressure value of the user's pointer in order to navigate through the video

The real interaction of the user's pointer with the video frame, however, was introduced by Goldman et al. [2007]. Using the presented system for video annotation, navigation and editing, the user has to paint over an object's silhouette with the mouse in order to track it. Due to the internal structure, the algorithm only works properly for big enough objects, that are not occluded while moving.

Draw a region over the video scene and drag it

To get rid of the object occlusion and merging problem, Karrer et al. [2008] found an elegant solution and introduced it at CHI'08[5] . The presented system called DRAGON (DRAGable Object Navigation)is a software product, implemented for MAC OS X systems, that performs video navigation by simple pointing at the object in the video. The pre-processing stage is hereby of crucial importance, because the video is prepared in such a way, that the motion trajectory of *every* pixel can be composed at runtime.

Click on an arbitrary point in the video and drag it

All currently existing DMVN-systems will be presented in more detail in section 2.2—"Overview of existing selection techniques in direct manipulation video navigation (DMVN)". However, the weakness of the DMVN in general will be investigated in the following section. The point of the present work is going along with it.

1.2 Problem description

Before starting with a problem description, the notion of the DMVN-system has to be explained.

[5] http://www.chi2008.org/

Definition:
DMVN system

DMVN SYSTEM:
DMVN system (Direct manipulation video navigation system) is a software product, that enables performing video navigation by moving objects directly, such as clicking on them and dragging them as opposed to indirect navigation, such as using time-slider (digital video navigation, e.g., VLC player) or using menus and buttons (analogue video navigation, e.g., VHS).

Following software components are supposed to be present in the usual DMVN system:

- Graphical User Interface (GUI);
- Object Selection Mechanism;
- Selection Visualization (e.g. visualization of the object's trajectories);
- Object Tracking Algorithm.

The GUI looks similar to the usual Media-Player (Figure 1.4), however it's equipped with an additional interaction functionality.

Figure 1.4: A typical user interface of the DMVN system. After clicking on some position the user's pointer gives a visual feedback that the dragging mode is turned on.

The Object Selection Mechanism defines how does the user select the object in the video if he wants to track it. Current DMVN systems provide a variety of selection patterns:

- Point-Selection. User only clicks on the object and the system starts tracking from the specified position. This selection pattern is used in DRAGON by Karrer et al. [2008] and in DimP by Dragicevic et al. [2008];
- Area-Selection. User has to paint over an object's shape or draw its simplified contour. This mechanism was presented by Goldman et al. [2007];
- Growing Bounding Box represents one of the possible auto-selection techniques. After the user has clicked on the object, system encloses an object into a bounding box and tracks it. This kind of selection was applied by Trichet and Merialdo [2006] for object tracking in the interactive television.

The DMVN system should provide a feedback to the user after he selected a region in the video. The bounding box around the object of interest and its corresponding trajectory might be highlighted for this purpose. Both elements are composed by the selection visualization algorithm.

The components described above are very important for DMVN system , however its core is a tracking algorithm.

OBJECT TRACKING ALGORITHM:
Object Tracking Algorithm is a tracking of an object specified by an initial data structure through the video sequence. Every DMVN system described in Section 2.2 has a different tracking algorithm. In fact, the different data structure used for the tracked entity:

- a single pixel for DRAGON;
- a set of SIFT-features around selected point for DimP;
- a group of particles inside the selected region for area-selection by Goldman;
- a positioned bounding box for the fast object selection by Trichet;
- etc.

Definition: *Object Tracking Algorithm*

Considering different DMVN systems one can realize that every tracking algorithm has own weak points. To conduct an in-depth analysis of the problem, suppose the tracking algorithm is perfect, no matter which DMVN system is used.

Definition: *Perfect Object Tracking Algorithm*

PERFECT OBJECT TRACKING ALGORITHM:
In the scope of this work we define the perfect tracking algorithm as an algorithm, which fulfills following conditions:

- the tracking is independent of the object intersection, merge, partial or full occlusion at runtime;
- the tracking results are independent of the light conditions in the video.

That means, the perfect tracking algorithm follows the trajectory of the tracked item very precisely and doesn't crash from time to time, no matter how do the objects behave in the video. In the next step of explanation the real world examples will be manipulated in the described DMVN system.

Example 1: Consider the navigation of a simple tiny object with a complicated trajectory

First example is a little indivisible object that is randomly moving in the video scene (e.g., a pigeon on the figure 1.5, left). It doesn't matter, which tracked entity is used (e.g., one point or an area, that covers the whole pigeon), the resulting trajectory will look as shown on the picture in the middle.

Figure 1.5: Example 1: Trajectory of a pigeon on a sidewalk.

Now imagine, the user only wants to move the pigeon straight forward along the line, without dragging it along its curvilinear trajectory. Unfortunately, there has been no DMVN system until now, that can manage this situation in

the way it shown on the picture on the right: simplify the fine-scaled trajectory and move the object straight forward.

Example 2: Consider a complex object, that consists of many parts with different trajectories

In the next example a DMVN system using area selection is needed. Consider the situation, when the user wants to drag a bike with a person sitting on it (Figure 1.6). He is only interested in moving bike along its motion path. The problem is solved after the user has applied the area selection to the whole bike, or an object sitting on it.

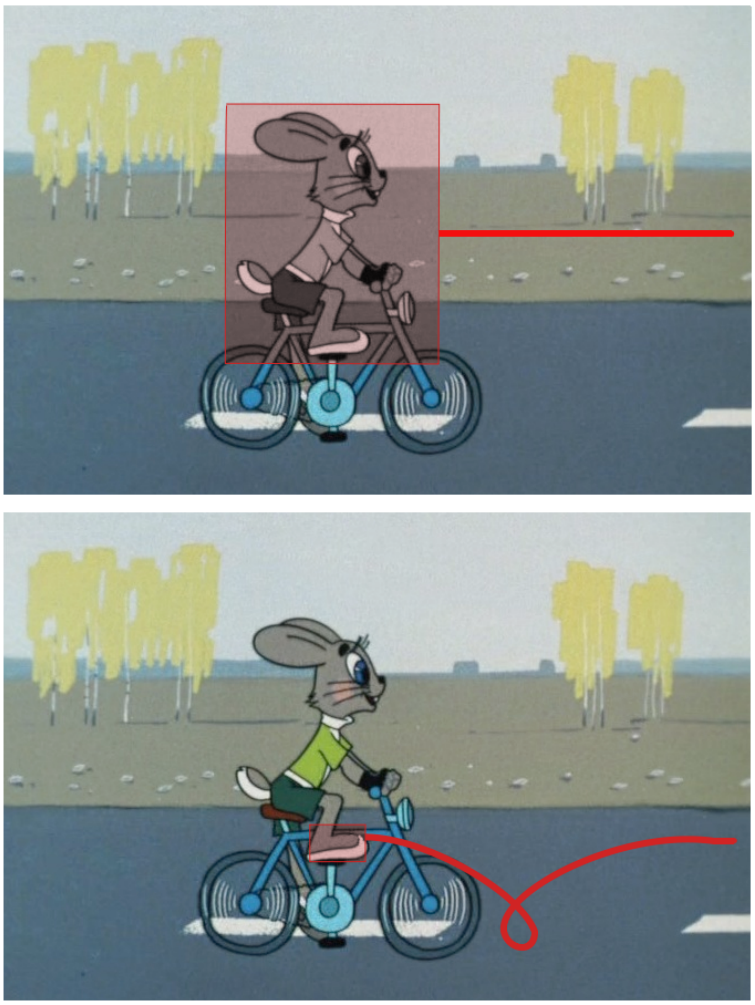


Figure 1.6: Example 2: The tracked entity in the video represents a complex object. Every part of it has its own trajectory: the bunny, the bike's frame, both wheels and pedals. Source: "Nu, pogodi!" (1969), Soviet cartoon.

Then again the next user is going to observe, how is the bike's pedal moving in the same video. Using the same selection type, user only has to draw a region around bike's pedal. The problems that arise in this case:

- the bike is moving to fast, drawing a region over the pedal is difficult;
- the pedal is to small, user has to draw a very small area.

The area selection exists only inside the project of Dan Goldman and is unstable, so it won't be applied here

One can suggest that after the training stage user will become high skilled and can apply the area selection everywhere even when this kind of selection is needed. But before considering these examples it was assumed, that the tracking algorithm of the DMVN system is perfect. However, the only one DMVN system that uses an area selection is the one developed by Goldman et al. [2007]. And its tracking algorithm fails for small areas due to the inner architecture.

Therefore, due to the current state of the object tracking in DMVN systems, two described cases (bike vs. pedal) stay unsolved. Even though this problem would have been hard to solve, if the tracking algorithm wouldn't fail in reality.

The point selection can fix the case in the example 2

But still, there is one more DMVN approach that uses a point-selection and a single pixel as a tracked item (e.g., DRAGON or DimP). As mentioned above, the main advantage of the point-selection consists in the correct tracking of even very tiny objects. That supposes to fix the problem, that the area-selection based DMVN system cannot solve. The moving bike is considered again. Pointing at the pedal with mouse, the desirable trajectory will be observed (Figure 1.6, bottom). The task is solved.

Example 3: Consider a large simple object with a complex texture

In addition to that, a new example is evaluated. The point-selection is now applied to the rolling football. The user clicks on the black field and gets a fine-scaled moving path of the chosen part of the ball. Unfortunately, things become more complicated, if the user wants to move a ball along the line aside. After spending hours with trying to solve the task one can realize, that every part of the ball moves in

different way and the only one point on the ball, that moves straight forward is its middle point (see Figure 1.7). This fact has not been obvious to the user before experiment.

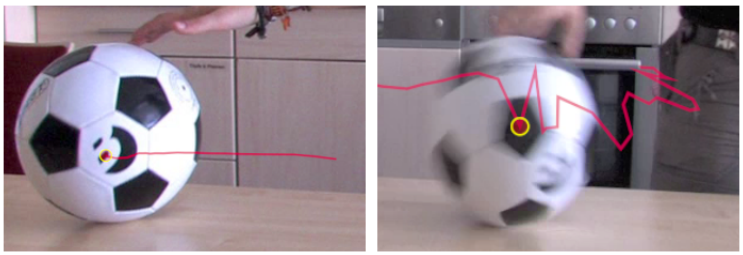


Figure 1.7: Example 3: A big rolling football.

In the examples described above user knows, how to interact with the system, and he can easily determine its functions. The question arises: why cannot the user get a desired response from the system then? To resolve the question, let's shift our attention to the theory of HCI (Human-Computer Interaction). The notion of **Seven Stages of Action** presented by Donald A. Norman [2002] is considered in order to explain, how do people do things and how do they interact with a computer system.

Apply the theory of the seven stages of action to the example 3

Before starting with a task, user has to determine, what does he want to achieve (Defining the goal). Next he has to do something to the system, e.g. click on the object or draw an area around it in the video frame (Execution). Finally, he has to perceive an output from the system and compare it to the goal he made before (Evaluation). This sequence is illustrated on the Figure 1.8.

Since the real tasks are not that simple, a new extended scheme was proposed (Figure 1.9). It consists of seven blocks and represents the seven stages of action mentioned above. We apply this scheme to the previous example of the football.

- **Goal**. Follow the black field on the ball using dmvn with point selection.
- **Intention to act**. Navigate a black field by selecting it with the mouse.

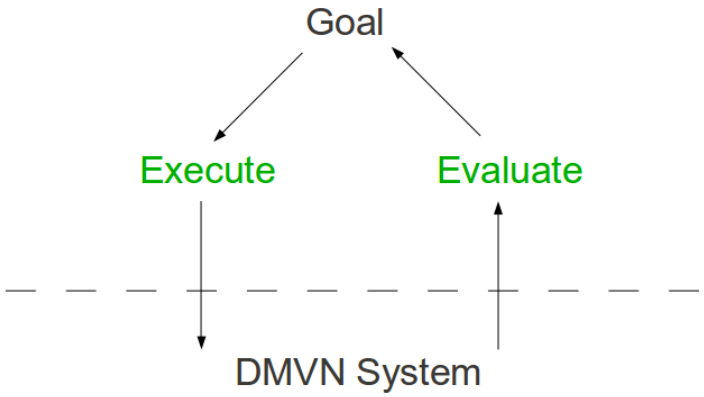


Figure 1.8: Different stages of an arbitrary action

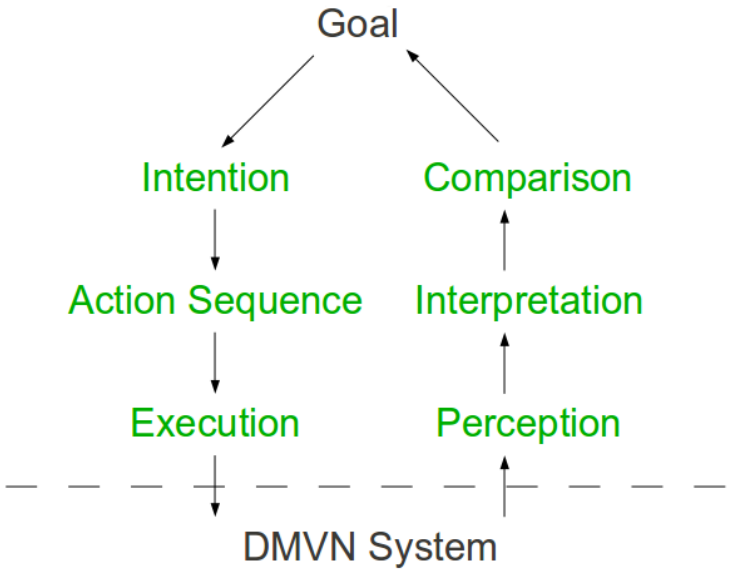


Figure 1.9: Seven stages of action

- **Action Sequence**. Click on the black field with the left button and drag it.
- **Execution of the action sequence**. Do it.
- **Perception** of the output state of the system. A curve is shown.
- **Interpretation** of the received information. A curvi-

linear trajectory of the black field is drawn.

- **Evaluation** of the interpretations. Curvilinear trajectory is equivalent to the desired result.

Consider one more goal: the ball has to be shifted aside. The execution part of the action won't change – click on the ball and drag it. However, the system reacts in a different way: instead of composing a linear movement trajectory it outputs a strange curve (e.g., Figure 1.7, right). The action fails during the evaluation stage, when the user realizes, that the system's answer isn't equal to the desired answer. This situation is called a **gulf of execution** in the field of HCI.

GULF OF EXECUTION:
It is a gap between the psychological language (or mental model) of the user's goals and the physical language of the system.

Definition: *Gulf of Execution*

To determine the physical language of the system, the DMVN system with the point-selection technique is considered. Its language is strictly limited. The system becomes the coordinates of the point and the current frame number as an input. The result is a movement trajectory of the single *selected point* starting from the input frame number.

Physical language of the system is represented by its algorithm

The psychological language of the user, however, is not hard coded compared to the system. If the user want's to follow some part of the ball, he clicks on this part and drag it. If he want's to move the whole ball aside, he has to point at the ball as well. Almost all users try selecting the middle of the ball in this case, but they cannot always click right in the middle of the ball. However, the middle point is the only one point on the ball, that is tracked by the system along the straight line. In all other cases users observe strange trajectories and cannot go any further, because they think they track the whole ball.

The psychological language of the user is his mental model of the system

If the action possibilities of the system don't match the intended actions of the user as in the described example, it always leads to the gulf of execution. There are different ways to resolve the arisen gulf [Norman [2002]]:

- *Visibility* of the system's state and the action's alternatives for user;
- A *good conceptual model*, i.e. consistent image of the system;
- *Good mappings*, i.e. clear relationships between actions and its results and between input parameter and their effects;
- Full continuous *feedback*, that the user receives about the results of his action.

In the scope of this master's thesis a new mapping space of the input parameters is proposed. As it was seen in the last example, the character of the user's gesture has to be put into relationship with a level of details of the motion trajectory. After a thorough research on video navigation and object selection techniques following mappings are considered:

1. **Area**. Level of motion details is mapped to the size of the region in the video, selected by user.
2. **Pressure**. This mapping supposes to resolve the gulf for those DMVN systems that use point-based selection. It requires an additional hardware though, such as a touchpad, which can help measure the pressure of the user's pointer. Depending on the pressure value, the motion trajectory of an object will either become more fine-scaled or stay poor-detailed.
3. **Dragging velocity** of the user's pointer is adjusted to the level of motion details. The user will then obtain a very precise trajectories after dragging an object slowly. On the other hand a very fast drag of the object leads to its mere common movement direction.

In addition to that, a new trajectory visualization mechanism will be developed, because the visibility of the DMVN system suffers one disadvantage in the examples above: the user doesn't know before experiment, how do the trajectories of different objects look like. However, every DMVN system has an understandable conceptual model and gives

a full feedback after every user's action. Therefore, two last criteria will not be considered further.

The implementation will be done for DRAGON, because this DMVN system is supported in the Media Computing Group of RWTH Aachen University since 2008.

1.3 Structure of the thesis

Chapter 2—"Related work" Modern DMVN systems existed at the moment of beginning the thesis are arranged in the second chapter according to the object selection type. Additionally, a group of techniques for the implementation of adaptive trajectories is presented.

Chapter 3—"Selection techniques for direct manipulation video navigation" In this chapter the contributions of our work to the direct manipulation video navigation (DMVN) system are described successively. Three selection techniques are explained: a region selection, a pressure sensitive selection, and a dragging velocity.

Chapter 4—"Evaluation" The results of two experiments are presented: evaluation of the area selection and adaptive trajectories applied to DRAGON. First experiment is a pure performance measurement (conducted automatically, without other users). Second experiment evaluates the addition of adaptive trajectories to the DRAGON's default point selection (carried out with help of 15 users).

Chapter 5—"Summary and future work" The last chapter contains a brief review of the work done on the object selection and adaptive trajectories in DRAGON. The second part provides an outlook on how the implemented algorithms might be improved and how the presented research area can be investigated further.

Chapter 2

Related work

" I have no special talent.
I'm only passionately curious. "

— Albert Einstein

2.1 Object selection in photos

Many object selection techniques for direct manipulation in video navigation have been influenced by similar approaches in the area of image processing.

Figure 2.1 represents a variety of selection techniques that can be found in the latest version of Adobe Photoshop. One can use primitive geometric shapes such as rectangles or ellipses (Figure 2.1, top left) to select simple convex objects. Selection masks of more complex shapes can be created using the *Lasso Tool* (Figure 2.1, top right). Tracing an object's outline is often time-consuming but one can assure that no redundant information has been included into the object silhouette. If faster selection is needed, a number of semi-automatic methods may be used as well (Figure 2.1, bottom).

Same techniques as for video object selection are available in every image processing software: point selection, rectangular selection, silhouette, etc.

Similar selection tools are available in GIMP, an image manipulation package for UNIX systems; apart from primi-

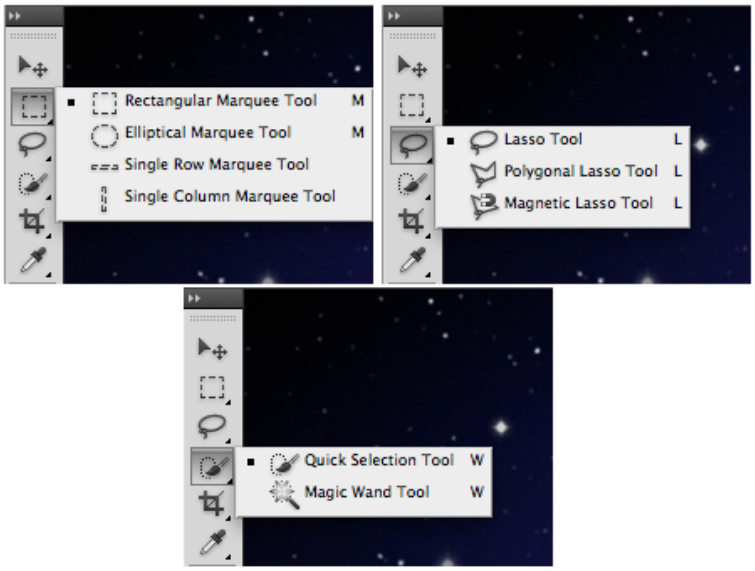


Figure 2.1: Selection techniques in Adobe Photoshop CS5.

tive selections such as rectangle, ellipse or free contour, one can apply advanced techniques like bezier and polygonal selections. Quick selection can be used as well.

Compared to photo manipulation, object selection in video often is not very precise. It is important to perform a fast selection and get consistent results. Since direct manipulation video navigation is a task where the video-object selection appears most often we will further discuss papers published in this research area until now.

2.2 Overview of existing selection techniques in direct manipulation video navigation (DMVN)

The following section contains an overview of the most significant papers published on the object selection in DMVN systems. Six selection techniques will be presented in this section: a point, an area, a storyboard, a trailblazing and two implicit selection techniques: a dragging velocity and a pressure.

2.2.1 Point

Using this technique the user can select an object by simply clicking on it

According to the object selection technique, the two systems considered in this section are similar. A *point* that the user's pointer is positioned on is considered to be an interaction pattern: the navigation takes place after the user has clicked by mouse on some object in the video and dragged it to an arbitrary direction. However, there are different implementation algorithms beside the two systems used point selection.

DRAGON (**DRA**gable **O**bject **N**avigation system, introduced by Karrer et al. [2008]) considers the optical flow through the video frames to compose movement trajectories for every point in the scene (Figure 2.2).

Definition: *Optical flow*

OPTICAL FLOW:
A common technique used for motion estimation in Computer Vision, defined by Horn and Schunck [1981] as an apparent motion distribution of brightness patterns in an image sequence. In other words, its a set of vectors, representing an inter-frame movement of every particular pixel.
This approach has a big advantage, dealing only with pixel movements not considering the global object transformations. But it might cause an incorrect velocity field, due to its lighting dependance: a fixed object might be obtained as a moving structure when being illuminated by moving light source.

Dragicevic et al. [2008] apply a relative feature flow algorithm for the same purpose. Reducing the amount of tracked points in every frame, the authors attempt to improve video navigation performance. However, the algorithm sometimes fails or doesn't work properly due to occlusion, splitting, merging or deformations of objects in video.

Dragicevic et al. [2008] implemented a special video player called DimP (**Di**rect **m**anipulation **P**layer), that allows to evaluate the described technique in practice (Figure 2.3).

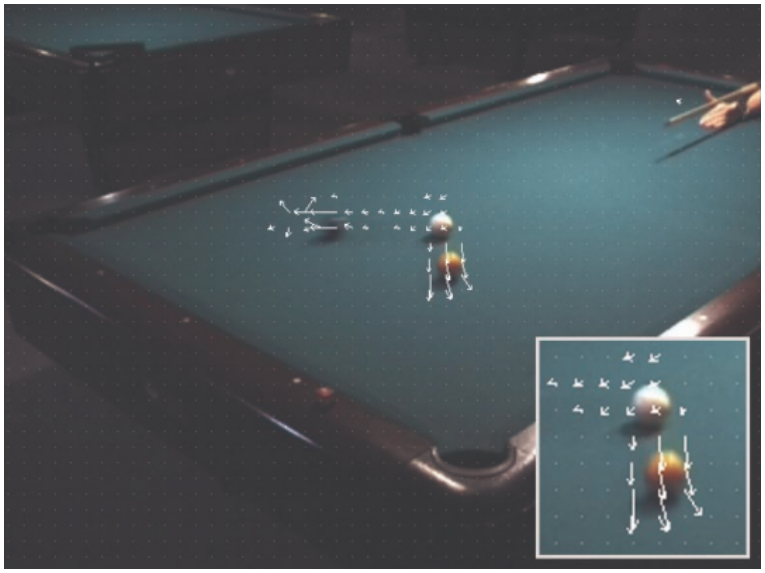


Figure 2.2: Optical flow computed inside DRAGON. Arrows show the pixel's movement direction between adjacent frames, a good opportunity to create a trajectory.

Definition:
Feature flow

FEATURE FLOW:
An extended version of the optical flow algorithm, intended for finding correspondences between video frames. Presented by Liu et al. [2008] it was supposed to reduce an amount of computations by considering only most important features in the frame. This algorithm is often called SIFT-flow, due to usage of SIFT-detector for feature extraction.

Despite of different realizations, both algorithms satisfy Sheiderman's concept of direct manipulation interface [Shneiderman [1983]]:

- usage of physical actions (mouse selection and movement) instead of complex syntax;
- fast feedbacks during interaction (trajectory visualization, mouse pointer's shape change);
- fast user's education (no large explanations needed).

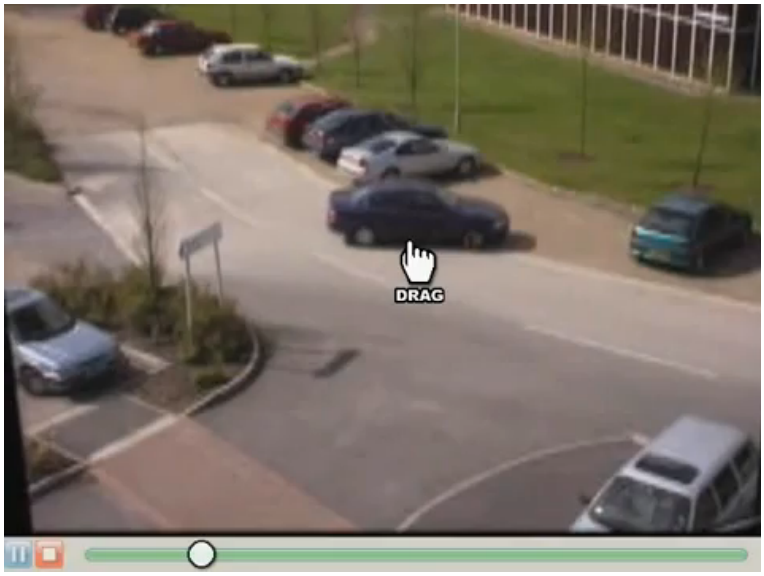


Figure 2.3: Direct manipulation Player (DimP) provided by Dragicevic et al. [2008].

Karrer et al. [2008] also provided a user study, which revealed that people tend to more natural way of video navigation. Once DRAGON and DimP were presented at the same conference, many areas of everyday life were found, where direct video manipulation might be applied successfully.

Point selection is now offered as a commercial software developed by ImageGraphicsVideo[1] department of Dynamic Ventures, Inc. At the first glance it reminds DimP player, described above (Figure 2.4).

Point selection can be applied to the concept of the interactive television

Trichet and Merialdo [2006] present an algorithm for fast object selection in the interactive television, that uses a point selection as well. After user selected a point in video, system starts growing a bounding box around this point. Frame color and contrast information as well as a motion field are defining a region-of-interest (Figure 2.5).

[1] http://imagegraphicsvideo.com/

Figure 2.4: Video manipulation software provided by Dynamic Ventures, Inc.

Definition: *Region-of-interest*

REGION-OF-INTEREST:
Region-of-interest (ROI) is an area of the image. Its size depends on the factors, that influence user's attention. Possible factors are color and contrast information, motion field and position of the point, selected by user. Good performance is achieved by computing some of these factors offline.

This selection technique has already been used in a commercial video of a famous jeans brand

Creators of the Wrangler® online shop [2] believe, that such intuitive way of scrubbing video might increase the shop's attractivity. Although the interaction space is restricted to only one character on the screen (Figure 2.6), it is amazing to play with brand's commercials. A short video report is available on YouTube.[3] The implementation is done with ActionScript 3. Initiated in summer 2010 the application is still in progress and already has prototype for video navigation on iPad.

[2] http://www.wrangler-europe.com/
[3] http://www.youtube.com/watch?v=2zuOdJQiljw

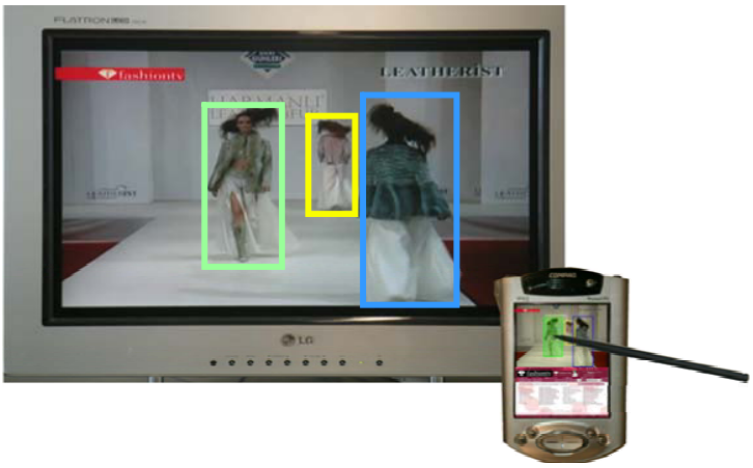


Figure 2.5: Bounding box selection used for interactive television. Source: Trichet and Merialdo [2006].

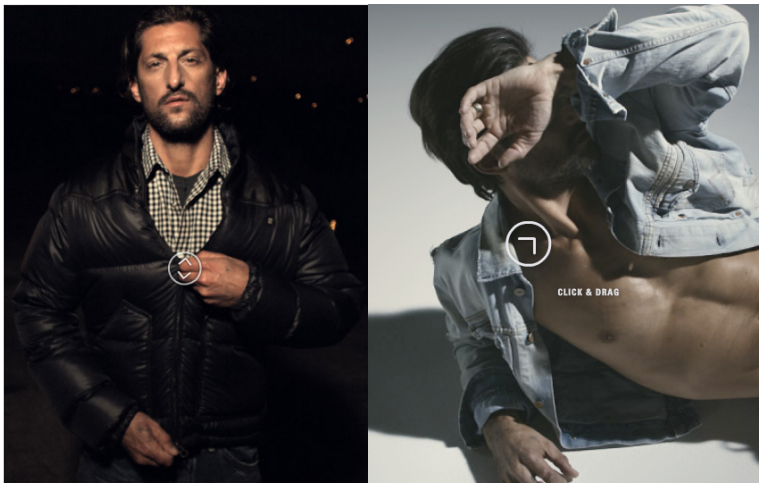


Figure 2.6: Wrangler® online shop. User can simply drag an little character in order to scrub a commercial video in time.

2.2.2 Area

User can select the whole object or a part of it

While navigating through the video scene whole objects can be selected, not only one point. In this chapter we will discuss the next selection pattern called area. According to object tracking survey of Yilmaz et al. [2006], the fol-

lowing shapes are used to represent an area of interest: sets of points, primitive geometric shapes (rectangle, ellipse) or object silhouettes.

The particle grouping approach allows a stable tracking of points inside the selected area

Goldman et al. [2007] implemented a system that is based on direct manipulation and allows performing video annotation, navigation and composition. A particle grouping approach is used as an algorithmical background. During the first stage of the algorithm, the system extracts motion information from a video and represents it by a set of points (particles). The points are grouped into sets according to the objects' behavior. Selection can be performed in two ways:

- User selects a point in a video frame. System looks for a closest group of particles and follows its movement direction.
- User paints over(or around) some region in a frame and system combines multiple groups of particles in the selected area (Figure 2.7).

We will consider five typical area selections in the scope of this work

Beside the common algorithmical background, only the second technique can be considered as an area selection. We've extended a set of possible selection techniques using section 2.1—"Object selection in photos":

1. Generally used mouse selection (rectangle).
2. Drawing a simplified contour (circle, ellipse) over an object of interest.
3. Drawing an object's silhouette.
4. Painting over an object.
5. Autoselection.

A good way is to conduct an experiment and to discover, whether the area selection is more useful then the usual point selection when considering a DMVN system. In the scope of this work we will also inspect, whether there is an another implementation of the area selection technique,

Figure 2.7: Possible selection techniques inside the direct manipulation system of Goldman et al. [2007].

which is much better than the Goldman's one. There might be a selection, that could be applied to almost every object in a video, so that it can be used as a multipurpose selection technique.

2.2.3 Storyboard

Another system presented by Goldman et al. [2006] creates schematic storyboards for short video files. With a small user input and a short preprocessing stage the system generates an informative picture, showing details from a given video footage. During the preprocessing stage a user has to look through the frames in order to identify key video frames and key features appearing in multiple frames. An output picture is sometimes combined from a couple of

This technique allows to convert a frame sequence into an interactive storyboard with the corresponding motion arrows

frames and consists of a motion information, represented with a motion arrow (Figure 2.8). This kind of visualization helps users perceiving and understanding a video content in a short time.

Based on this mapping a direct video manipulation mechanism was implemented. User selects an object in the storyboard and moves it along its movement arrow, while video is scrolling in time.

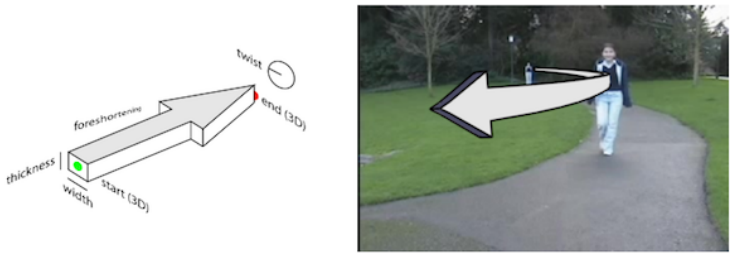


Figure 2.8: A motion arrow, constructed in system of Goldman et al. [2006].

2.2.4 Selection using external visualizations (Trailblazing)

Trailblazing is primarily suited for video surveillance systems

Unlike Goldman, Kimber et al. [2007] present a direct manipulation system for video navigation, that doesn't need any user input during tracking preprocess. The interaction mechanism is similar to the ones from section 2.2.1—"Point", but it is better suited for surveillance systems.

The trajectories of all objects are mapped onto a floor plan of a surveilled location

The GUI consists of three elements: a floor plan of surveilled location, a timeline, and a video window. The floor plan is used as an external metadata, defining objects in the video and their trails (movement trajectories). The system allows clicking on a random point in the scene due to 3d reconstruction and this action moves the selected object to the closest location. While clicking on the floor plan, the system shows a list of possible candidates, that can be navigated at this particular moment (Figure 2.9).

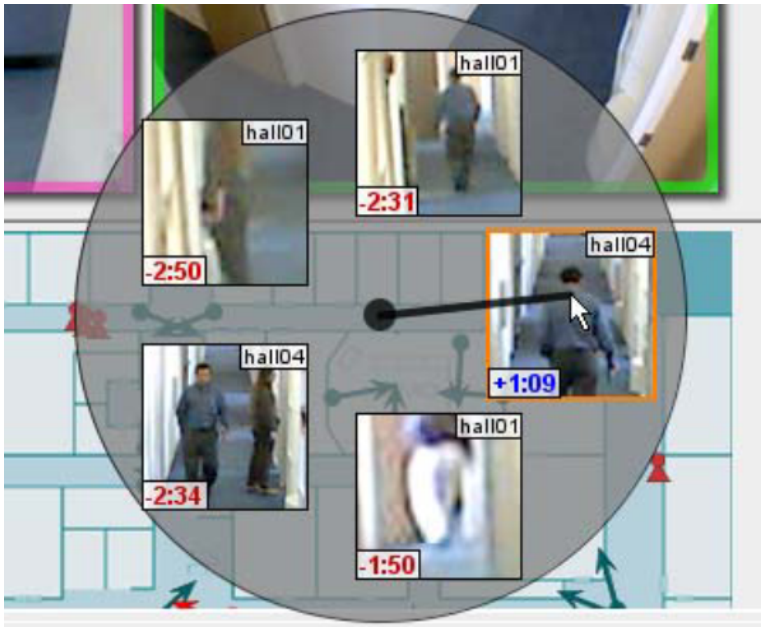


Figure 2.9: A list of objects, which are spatially and temporally close to each other.

2.2.5 Implicit selection techniques

In this section we will give a short overview of selection mechanisms, that don't use a general pointing at the object by finger or mouse.

TLSlider allows to dynamically adapt the time slider's resolution according to the pressure of the user's pointer

First approach was presented by Ramos and Balakrishnan [2003] as Twist Lens Slider (TLSlider) and was supposed to improve a video navigation. The system divides a video stream into a set of short pieces and creates a slider, by putting their preview frames in a line together. The interaction between the user and the system takes place, however, over an additional hardware such as a graphics tablet and a stylus. After user has clicked on the one from preview frames, playback jumps to a corresponding moment of time. Increasing pressure at the interaction point the user increases the amplitude of the twist lens too, so that the resolution of the time slider is getting higher (Figure 2.10).

Since lots of touchscreen devices are now available for the private use and many of them can already measure the

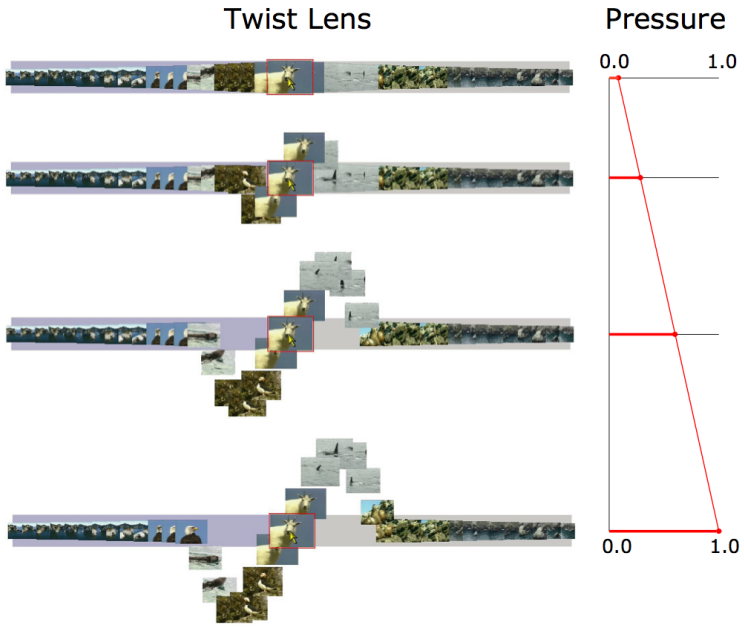


Figure 2.10: Twist Lens Slider by Ramos and Balakrishnan [2003]. The figure shows how the amplitude of the lens changes according to the pen's pressure, displayed on the right.

However, an additional hardware, such as a pressure sensitive touchpad, is needed for this technique

pressure too (e.g., Wacom tablets[4]), it is possible to navigate through the video without any additional hardware. Thus, in the scope of the direct manipulation video navigation considered in this master thesis we will combine the level of video details with the pressure, exerted by user on the touchscreen. The object navigation in the video proceeds in such a way that user's pointer follows a common movement direction of an object in the video, if the pressure is very low. Otherwise, user has to follow an object very precisely.

Another selection technique uses a dragging velocity of the user's pointer in order to adapt the slider's resolution

Second implicit selection technique considered in this section is about mapping the level of movement details to the velocity of user's gestures. In the scope of the document navigation Igarashi and Hinckley [2000] invented a mechanism, that integrates a speed depending automatic zooming for large documents. If the user scrolls very fast,

[4]http://www.wacom.eu/

the view zooms out automatically so that it does not disorient him. This implicit selection technique, where the user controls only the movement speed and the system adjusts the level of details, was inspired by Alphaslider, extended by Shneiderman and Ahlberg [1994] (Figure 2.11). For the purpose of the direct manipulation video navigation we will apply the velocity based selection in a similar way as a pressure based one. Thus, the momentary velocity of the user's pointer must be estimated in every video frame. The level of details in the video should get lower if the user's pointer slows down while dragging.

In the scope of this master thesis we will implement both implicit selection techniques and compare them to the general (explicit) selection techniques such as region selection, described in the section 2.2.2—"Area". We are going to prove, whether the implicit way of selecting objects is more useful than the explicit one or not. Furthermore, we will check these selection techniques against each other.

We will implement and compare both DMVN techniques, the *pressure* and the *dragging velocity*

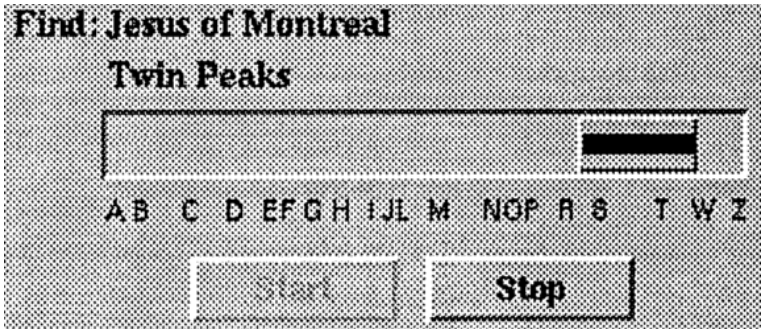


Figure 2.11: Alphaslider by Shneiderman and Ahlberg [1994]: granularity of the so called acceleration interface is proportional to the movement speed of the mouse pointer.

2.3 Adaptive trajectories

In this section we will show, how to adjust trajectories to the performed selection. We consider two main selection techniques: a point selection (section 2.2.1—"Point") and an area selection (section 2.2.2—"Area"), since the other approaches discussed in section 2.2 use the one of this two.

Different approaches can be applied to the motion trajectory in order adapt it to a certain level of motion details

According to Ramos and Balakrishnan [2003], user can change an additional input parameter of the direct manipulation system in order to get different levels of video details. Talking about motion trajectories we decided to define a level of motion details as a level of trajectory smoothness. Adapting the trajectory to the current state of the system might be achieved in three ways: trajectory smoothing, usage of image pyramid and trajectories clustering.

Yan and Kankanhalli [2002] demonstrate two smoothing methods, that might be applied to the motion trajectories: a polyline simplification and a bezier curve based filtering. A discrete curve evolution from Treetasanatavorn et al. [2004] might be used as well. The trajectories adaptation looks now as following:

- Precompute fine-scale trajectories for all objects in the video.
- Define input levels of details.
- Smooth the trajectory according to the input level value.

The second adapting approach described by Cheng and Ho [2009] uses an image pyramid, that consists of a motion at different resolutions.

Definition: *Image pyramid*

IMAGE PYRAMID:
Image pyramid is a hierarchical structure of pictures, that represent the same image at different resolutions. The highest resolution picture is on the bottom level.

This mechanism is similar to the previous one: once the input parameter has been changed - the level of resolution has to be switched and the trajectory has to be recomputed.

The last method for computing adaptive trajectories uses a trajectories clustering concept [Buzan et al. [2004]]. In the context of this method, an input parameter defines a size of the cluster.

Chapter 3

Selection techniques for direct manipulation video navigation

"I'd rather live with a good question
than a bad answer."

—Aryeh Frimer,
professor at Bar Ilan University

In this chapter we will present the contributions of our work on object selection to the direct manipulation video navigation (DMVN) system. A gap in the user interaction, which couldn't be solved by any DMVN system, has already been observed. Inspired by the "seven stages of action" theory [Norman [2002]], this gap was called gulf of execution. After a wide literature review in chapter 2—"Related work" it was clear that the object selection mechanism of DMVN systems should be extended. For further implementation, three new selection techniques have been chosen. They are supposed to close the gap between the user's mental model and the system model:

- Region selection (explicit selection technique);
- Pressure based selection (implicit);

- Dragging velocity based selection (implicit).

As mentioned in section 1.2—"Problem description", the system DRAGON in its last version (DRAGimation) will be deployed as a DMVN system. The system DRAGON [Karrer et al. [2008]] was developed in 2008 at the RWTH Aachen University and has been maintained by Thorsten Karrer, Malte Weiss, Moritz Wittenhagen, Jan Borchers and others ever since. Both the graphic user interface of the old and the new version after the extension are illustrated in figures 3.1 and 3.2 respectively.

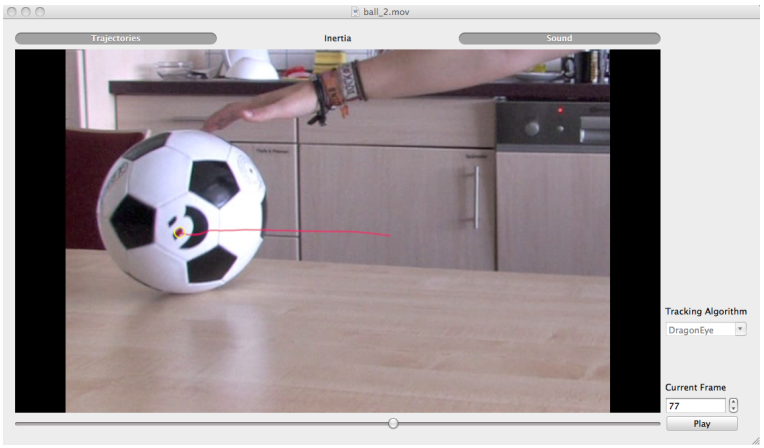


Figure 3.1: The former graphic user interface (DRAGON).

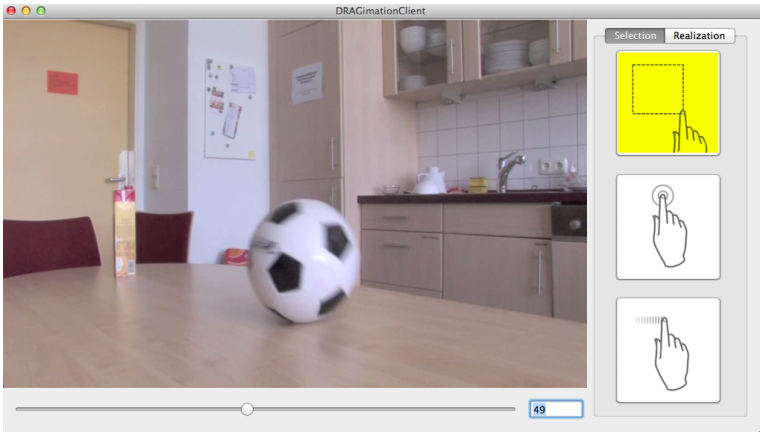


Figure 3.2: The new graphic user interface (DRAGimation), which has been extended in the scope of this work.

The old version offers only a point-based video navigation, which allows the user to select a random point of the object and drag it through the frame sequence. The new version (DRAGimation) provides three selection techniques on the whole. These three new techniques were integrated in the scope of this work and have been discussed in detail in section 2.2. The following sections deal with object selection in DRAGimation and provide a detailed explanation of the performed program extension.

3.1 Region selection

This section describes the region selection both from the user's point of view and the one of a developer. First and furthermost it is defined, how should the user apply this selection technique in a DMVN system. Afterwards, the main algorithm of the region selection is explained stepwise, giving a deep overview of the DRAGON functionality itself.

3.1.1 Selection behavior

In order to define a specific region of the frame, which has to be processed for navigation purposes, following techniques might be used (Figure 3.3):

- Classic rectangular selection (First picture from the left);
- Simplified contour (Second picture from the left);
- Drawing a silhouette (Third picture from the left);
- Drawing over an object shape (Technique from Goldman et al. [2007], fourth picture from the left).

A user survey has revealed, that the majority prefers the rectangular selection, as they are used to it from the daily use of the PC. This kind of selection seems to be a reasonable approach, and will be therefore implemented in this work.

Figure 3.3: Different area selection techniques.

3.1.2 Implementation

The last version of DRAGON is designed as a client-server application

The computation workflow in DRAGimation benefits a lot from its architecture. Unlike DRAGON, the latest version (DRAGimation) is designed as a client-server application, so that the most CPU demanding operations such as a trajectory computation are now executing on the server's side. The data exchange between the server and the client is performed over multiple channels, which allow many trajectories to be computed in parallel[1].

The simplest way is to compute the mean trajectory from the bundle of trajectories beginning inside the selected area

Assuming the intuitive way of implementing area selection will simply consist of generating random points inside the selected area and requesting their trajectories. The average calculated over these trajectories is supposed to represent the movement of the selected area. Obviously, the size of the selected area does have a significant influence on the computation success. If the selected area is much bigger than the object itself, or does not cover the object properly, the generated points will belong most probably to the background and not as planned to the object. This issue will be illustrated later on in this subsection.

2D coordinates of the points inside the area are generated via gaussian distribution

In order to produce 2D random points in our algorithm, a random number generator based on a gaussian distribution has been integrated into DRAGON. Two parameters of the gaussian kernel function should be defined for such a 2D object as an area: the mean value and the variance. The mean value is obviously the middle point of an area. The variance, however, should be computed in a certain way. To make sure that the majority of the generated points will

[1] After trying the different numbers we have estimated a most stable amount of requested trajectories n = 10.

be placed inside the defined region, the variance should be equal to 25% of its edge length (Figure 3.4).

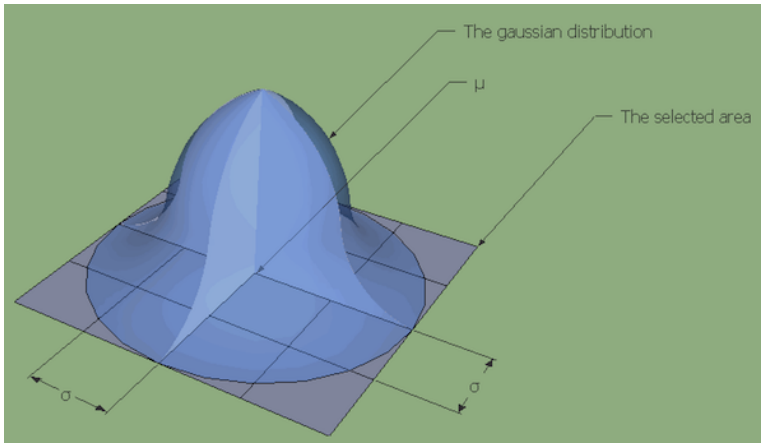


Figure 3.4: The variance σ in the Gaussian distribution. According to the properties of the gaussian distribution, 95.5% of the random generated points fall into a square with an edge length $2 \times 2\sigma$.

To generate a random point by means of the Gauss distribution inside a region the following approach is used [Press et al. [1992]]. The coordinates (x_i, y_i) of the random point are computed based on two independent gaussian numbers (η_1, η_2) with the mean values $\mu_{\eta_1} = \mu_{\eta_2} = 0$ and unit variances $\sigma^2_{\eta_1} = \sigma^2_{\eta_2} = 1$ as follows:

$$x_i = \mu_1 + \sigma_1 \cdot \eta_1 \tag{3.1}$$

$$y_i = \mu_2 + \sigma_2 \cdot (\rho \cdot \eta_1 + \sqrt{(1 - \rho^2)} \cdot \eta_2), \tag{3.2}$$

where

(μ_1, μ_2) are normalized 2D coordinates of the region center,

σ_1, σ_2 are corresponding standard variances for x and y axis,

ρ is the correlation coefficient.

Once all points have been generated, the client application requests the trajectories from the server. As mentioned in

the beginning of this subsection, the resulting mean trajectory strongly depends on the area size. In order to examine the impact of the area selection on the resulting trajectory, an experiment has been carried out. In this experiment the one and the same object is selected by using three different area sizes. The results are shown in the figure 3.5.

The described algorithm does not work if the selected area is larger then the object of interest

The area selection is supposed to be successful, if the majority of random points (initial nodes of trajectories) are located inside the object's edge or borders. This condition is for instance not satisfied in the second and third case (the blue star) of figure 3.5. Even though the small area selection of "bad-shaped" objects, in this case the blue star of figure 3.5 prevents errors during the trajectory computation. It is nearly impossible to obtain an acceptable smooth curve when enclosing the object in a much bigger rectangle.

To illustrate this issue, the random points distribution inside the selected region has been shown in figure 3.6. As a matter of fact, if the area becomes larger, most points are located beyond the object's edge. The prepared set of computed trajectories will include by implication many outliers, which negatively influence the mean trajectory.

The randomly distributed initial points of trajectories should better not fall onto the background

To conclude, it is important to point out, that the area selection remains stable as long as the selected area comprises a good bit of points from the affected object and not from the background (or at least very few points belonging to the background) as depicted in figure 3.7 (top row). It is furthermore important to reduce this constraint on the user's selection and to keep the algorithm stable even though the input selection is wrong. This is the case when the selected area comprises too many points from the (white) background as shown in figure 3.7 (bottom row). To avoid such an error, the input point sets must be treated within an additional preprocessing stage.

The trajectory clustering approach allows to exclude faulty trajectories from the bundle

Inspired by the particle grouping technique [Goldman et al. [2007]] we decided to apply a K-means clustering to the input set of points inside the area. It is an iterative technique for separating elements into clusters. The initial number of clusters has been set to two, since there are at least two groups: points belonging to the object and points lying on the background. In order to use a trajectory for the clus-

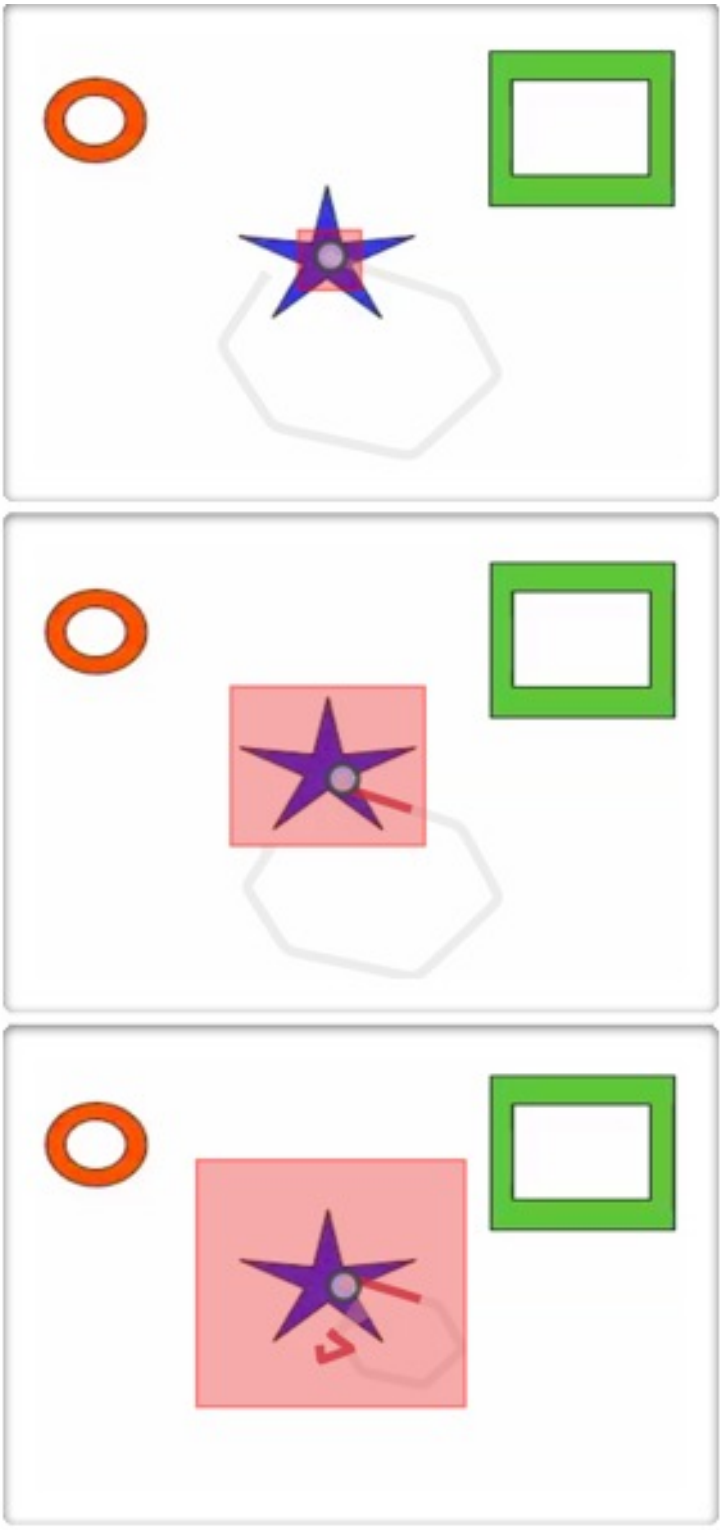


Figure 3.5: Region selection in DRAGimation: the mean trajectory for the different areas.

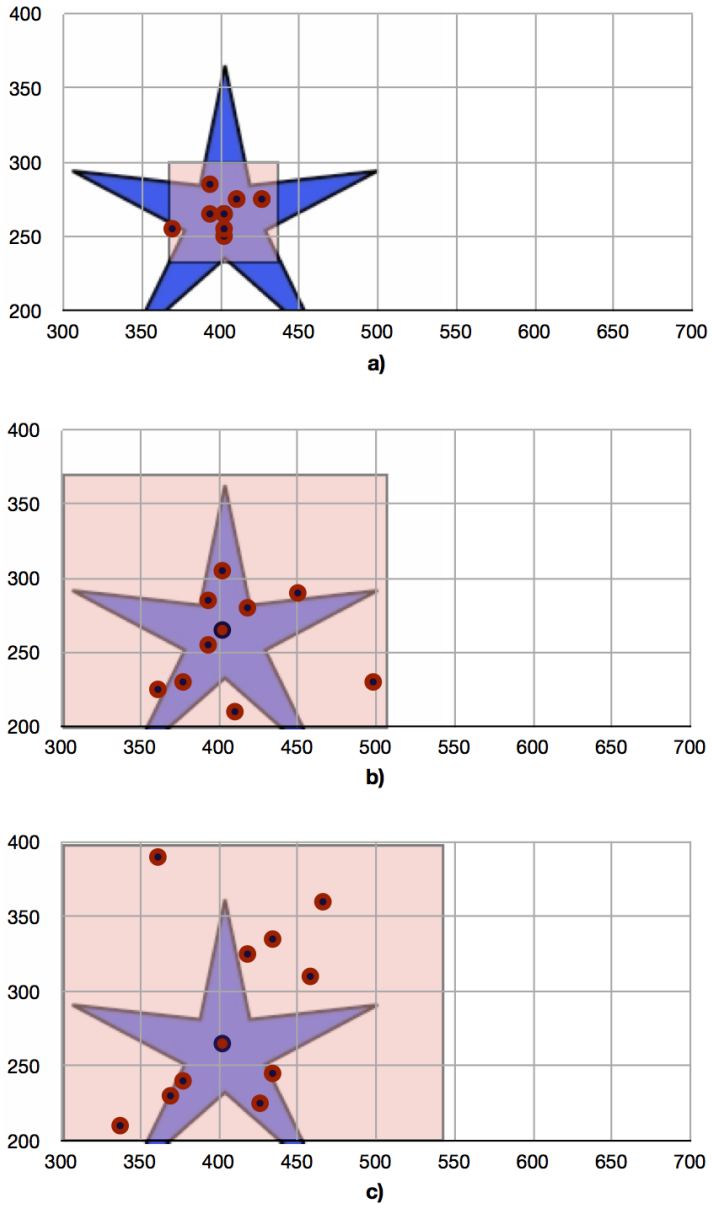


Figure 3.6: The ten randomly generated points (by the Gauss generator) inside selected areas with different sizes: (a) 62×59 px, (b) 193×158 px, (c) 267×246 px. Refer to figure 3.5 for the corresponding video frames.

tering approach, we have to find a way to represent it as a cluster element. Every trajectory in the video frame can be mapped to a 2D translation vector in an affine motion space by taking the difference between its start and end node coordinates. From now on the corresponding 2D vectors will be clustered instead of trajectories.

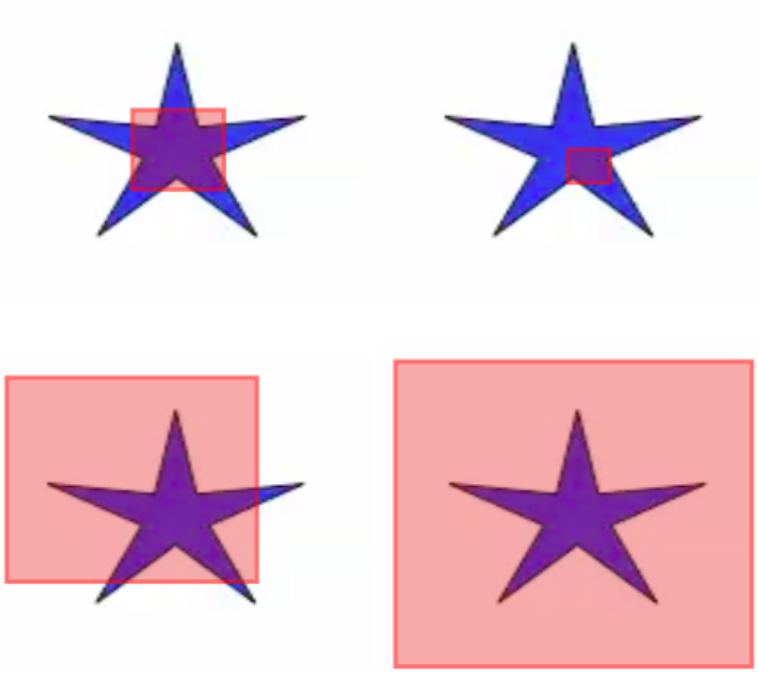


Figure 3.7: DRAGimation: possible selections of an object.

Our algorithm is called every time the new piece of trajectory comes from server, and it consists of following steps:

1. Initialization. Assign two random vectors from the initial set to the cluster centers a and b.
2. For all trajectories
   - compute the distance to every cluster center. Here the distance between vector endpoints is meant, since all translation vectors start from origin (Figure 3.8);
   - push the trajectory into a cluster which center is the closest to it.

3. Recompute the cluster centers a and b as the centers of gravity of the corresponding clusters. Start again from 2. and repeat until convergence.

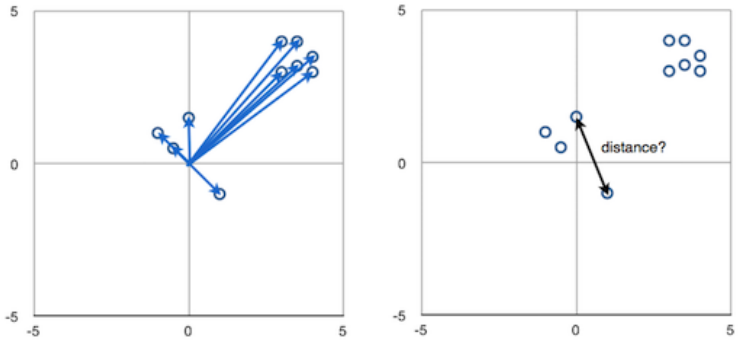


Figure 3.8: K-means clustering inside DRAGimation: the distance between each trajectory and a cluster's center is computed as a distance between the endpoints of the corresponding translation vectors

Consistent results have been achieved even after five iterations of the described algorithm

The algorithm is supposed to converge as soon as the clusters don't change their content anymore. But it is still possible that a certain trajectory will oscillate between two clusters and keep the algorithm running. To prevent that, we decided to limit an amount of iterations to some number. We realized that five iterations are enough to converge the algorithm inside the given restrictions in all observed cases (ten trajectories and two clusters).

It is important to choose the right cluster for the estimation of the mean trajectory

The described algorithm distinguish trajectories into two clusters: moving pixels belonging to the real trajectory and pixels belonging to the background. It is now important to take the cluster with only "good" trajectories for the further processing. As long as the object is moving, the endpoints of trajectory are further away from origin of the affine motion space and build therefore an own cluster. Unusable and unstable trajectories does not move at all (background pixels) or make just little movements into random directions (noise). Thus, they fall into the second cluster and must not be considered in an average trajectory estimation. An intuitive way to distinguish "good" trajectories from "bad" ones is to measure how far are the corresponding cluster centers from the origin, and choose the furthest one.

However, there is a movement type, for which this approach will definitely not work. Assuming, the object moves along a circle or a spiral, consider the way of mapping a trajectory into an affine motion space. At the moment it is represented by a translation vector from its start to the end node. But if the trajectories bundle follows the circular path, almost every end point of every trajectory will be very close to its start. The respective distance will be small, but according to the logic described above it may not be considered as a "good" trajectory. The algorithm will always take the furthest cluster for the further computations. It is impossible to predict the algorithm's behavior in this situation since one of two clusters might be taken for the mean trajectory estimation. This can result in a significant instability of the whole program system (Figure 3.9, bottom).

Additionally, our algorithm has been improved and has become more stable in processing circular and spiral movements

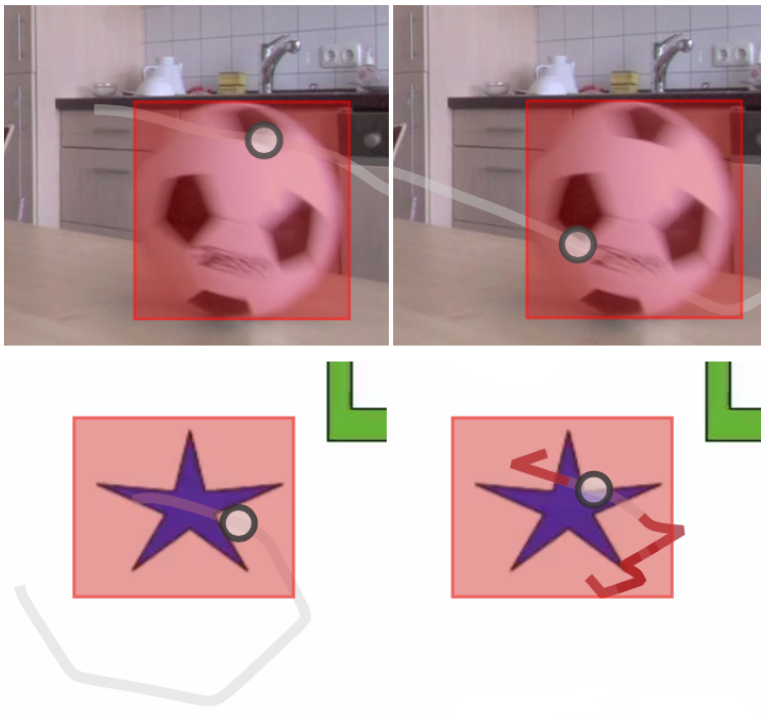


Figure 3.9: DRAGimation: the clustering results. Left column represents the average trajectory after a few seconds. On the right side is shown, how can the final curved trajectory look like after all pieces have come from the server.

An intuitive and obviously the simplest solution is to split the whole trajectory into pieces and apply the clustering to every one of them. Goldman et al. [2007] have used a similar strategy when applying particle grouping to video annotation and navigation. Taking empirical data in account the authors decided to cluster points every $\Delta t = 3$ frames. This slight modification in the algorithm has resulted in much better stability.

The estimation of the mean trajectory takes place over that part, which appears in all trajectories in the bundle

Once all "good" base trajectories have been clustered, the mean trajectory has to be computed. One must keep in mind, that the server needs some time to proceed and sends trajectory pieces as soon as they computation is finished, so the length of the acquired (base) trajectories may be sometimes different in the client's side. Thus the estimation of the mean trajectory in the next step should be done over that part of the curve, which appears in all other trajectories in the cluster. This part is defined by the minimum common length of these trajectories. To compute the mean trajectory, the algorithm considers base trajectories in parallel. In every step i only nodes with the index i are taken into account. The algorithm computes an average node, appends it to the resulting mean trajectory, and continues with the next value of i until the minimum common length is reached. The mean trajectory is updated this way every time new data is provided on the server and the user's side.

To test our algorithm on many videos, these have to be prepared with a special software offered by the i10-chair

The algorithm might be finally implemented and a user study can be conducted afterwards. To run the algorithm on the other videos, certain preparations are necessary. DRAGON considers the optical flow field through the video frames in order to create a movement trajectory (for an explanation of the optical flow please refer to the section 2.2.1—"Point"). The preparation phase lasts in some cases a couple of hours since all the flow fields for all video frames have to be estimated. The necessary software is therefore provided by the i10-chair at the RWTH University. The test video and accompanying flow packages are converted into a single DRAGON-package.

3.1.3 Selection visualization

Visibility is one of the seven principles of design [Norman [2002]]. It is important to give the user feedback, what is happening in the system even during an object selection: whether the selection was successful or not, may the user already drag an object or should he select a new region.

Thus the area should have been drawn at least. From all region selection types mentioned on figure 3.3 we chose the first one as the most users are used to it. The area is highlighted by a red rectangle and disappears when the user starts dragging an object through the frame sequence.

In particular the user has to complete two operations every time: select an area over an object and drag it. To keep interface simple we haven't added any features for choosing between these operations. We simply gave the user only one try for selection and one for dragging (Figure 3.10). After the user selected a region (Figure 3.10, middle), the system computes an approximated trajectory of the whole area (Figure 3.10, right) starting in a circle. User can now drag an object along its trajectory.

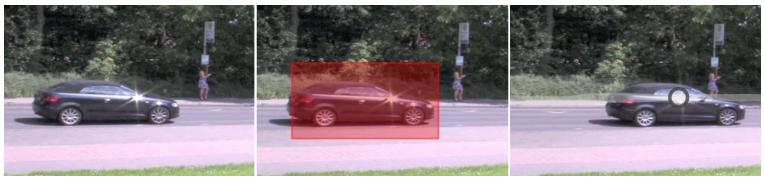


Figure 3.10: DRAGimation: the initial video-frame view and two stages of the object navigation via region selection.

3.2 Dragging velocity

Unlike using the area selection, with the next technique user doesn't select an object explicitly. Therefore, the dragging velocity is considered as an implicit object selection technique. The following section describes its approach and the algorithm.

3.2.1 Selection behavior

Dragging velocity is an implicit selection technique

The concept of dragging velocity selection differs from the one of area selection. It is not the usual object selection where the user explicitly defines what do we want to navigate and in what way. This is an implicit selection. A basic idea is to control the level of motion detail though dragging speed of a user's pointer.

Assume the point based video interaction is considered (more about point selection in paragraph 2.2.1—"Point"). The video navigation is then separated into two phases:

1. User clicks on the certain point in the video and gets a corresponding trajectory.
2. He can now drag an object along the trajectory as far as it is supported by its length.

The motion is more detailed if the dragging velocity is small

The trajectory is computed in two directions starting from the object's location and represents its movement before and after it came to the selected position. Thus the user can drag an object only forward or backward along the curve. There is one more degree of freedom which can be manipulated during the video manipulation. It is a dragging velocity of the user's pointer what we now can map to the level of motion details. The mapping takes place as follows:

- Don't change an initial trajectory that came from server if the dragging is slow;
- Show the less detailed trajectory if the dragging speed is increasing.

How to define whether the dragging is fast or slow, we will define in the following paragraph. But obviously the interaction will not make lots of changes to the latest version of DRAGON. However the system will be sensible to the dragging speed. We now have defined the interaction routine from the users point of view and will continue to explain how can be this idea implemented based on the latest version of DRAGimation.

3.2.2 Implementation

The dragging velocity is computed based on the signals (events) produced by the user's pointer

The most important statement, which has been made before any implementation is done, is a definition of the dragging velocity itself. From now on we will talk about a current velocity of the user's pointer (mouse pointer in case the mouse is used). This value is easy to compute because of the basic :mouseDown and :mouseDragged events that are described in the DGNView class of the DRAGimation application. After we have initialized the time value and a distance in the first one, we can simply update them every time :mouseDragged is called.

Three levels of motion details have been defined for the implementation

The next question that arises: which values does the current point velocity have? Due to a computation routine, we will save them as n px/sec. Using this unit of measurement we have got following values on the MacBook Pro with a 1440×900 px desktop resolution:

- $[0, 200]$ for slow dragging;
- $(200, 800]$ for the faster dragging;
- (800, ..) for the fastest motion when it is still possible to control the direction of the object movement .

This measurement is of prime importance for the following velocity levels estimation. The level of dragging velocity corresponds to the level of motion details that we want to control. We decided to consider only three levels: high, middle, and low. It is a preliminary version of the algorithm and we only want to see, whether this feature will work at all to not. Now we have to correlate the levels of details with the levels of velocity.

- Don't change a trajectory, if velocity $\leq$ x;
- Show the motion at the medium level of details, if velocity value is between x and y;
- Schematically represent a trajectory, if velocity $\geq$ y.

Different approaches can be used to change the level of motion details

The main point of this paragraph however is the implementation of the dragging velocity based navigation. Namely the controlling of level of the motion detail. There are three main approaches, briefly introduced in paragraph 2.3—"Adaptive trajectories":

- **Trajectory smoothing**. The simplest one according to the implementation. The level of motion details is correlated with a level of the trajectory smoothing.
- **Trajectories clustering**. During initialization a certain cluster is defined around the interaction point. If the dragging velocity is increasing, the point's neighbors have to be investigated and therefore the cluster should be enlarged. The clustering algorithm mentioned in paragraph 3.1—"Region selection" might be used as well.
- **Video pyramid based on the image pyramid**. The video is saved on different resolutions into a package of the trajectory levels with corresponding flow fields. During interaction the algorithm has to switch the levels of the pyramid according to the dragging speed.

In the scope of this work only the trajectory smoothing will be implemented

The first approach is the simplest one regarding the implementation. Trajectory clustering is already done for the area selection and due to a lot of additional computation has caused a lack of overall performance. The clustering has to be done very often, namely every time the main trajectory is updated. However, due to limitations of dragimation by 10 trajectories it may not work correctly for big clusters. Same is for the trajectory pyramid, which is first complicated by a long preprocessing stage for every video. Therefore, in the scope of this master's thesis we will develop the trajectory smoothing as the simplest one. because we are concentrated on the interaction itself not on the algorithmical background. Others ideas we will keep for the future work.

Fro now on we will concentrate only on the possible curve smoothing approaches. We already mentioned a couple of them in 2.3—"Adaptive trajectories":

- Low-pass filtering;
- Polyline simplification (from Yan and Kankanhalli [2002]);
- Bezier curve based filtering (from Yan and Kankanhalli [2002]).

The low-pass filtering will be implemented as a trajectory smoothing approach

Applying a low-pass filtering as the simplest approach to the trajectory smoothing we will get preliminary results much faster and can already conduct a user study. It s more important to make a statement whether this kind of selection is useful at all or not. We can take care of the computation stability with advanced methods afterwards. But if the dragging selection will not work for the users, it is reasonable to break up without losing time.

Varying levels of motion details are represented as a trajectory smoothed by an FIR filter with different convolution masks

We will use a simple finite impulse response (FIR) filter with different realizations for any level of trajectory smoothing. It requires a short amount of parameters and is more suited for trajectory smoothing. The output signal of the filter is produced as a weighted sum of a finite number of the input signals (points on the initial trajectory). The number of points to be taken and their weights make up a so called convolution mask of the filter. We call it *kernel* and keep unchanged during the whole trajectory processing. Considering trajectory as a set of points we will apply the filter mask for every point on the curve. Depending on different convolution masks we will get different level of motion detail:

- Initial trajectory stays unchanged (high level);
- Mask $[\frac{1}{6}, \frac{1}{6}, \frac{2}{6}, \frac{1}{6}, \frac{1}{6}]$ (middle level);
- Mask $[\frac{1}{36}, \frac{3}{36}, \frac{4}{36}, \frac{5}{36}, \frac{5}{36}, \frac{5}{36}, \frac{5}{36}, \frac{4}{36}, \frac{3}{36}, \frac{1}{36}]$ (low level).

FIR filter may negatively impact curves or surfaces (shrinkage problem), but it is easy to implement

The FIR filter however should be carefully applied to circular trajectories. The circular movement may shrink significantly under certain circumstances (depending on the convolution mask or on how many times it has been applied to a trajectory). This problem has been described in Taubin [1995] for curves and surfaces. Nevertheless, the big

advantage of this filter is its simple implementation. Moreover it is easy to integrate trajectory smoothing in existing code of DRAGimation: an additional trajectory processor (DGNTrajectoryProcessor) will take care of an initial trajectory that is piecewise coming from the server. It should be noted however, that after testing our algorithm with the convolution masks described above, no material adverse effects of the filter have been noticed.

The final algorithm for the dragging velocity object selection is looking as follows:

1. Initialization after user clicked on the object (:mouseDown event)
   (a) save start position of the user's pointer,
   (b) start timer,
   (c) initial smoothing level = 1 (no smoothing);
2. While user is dragging an object through the scene (:mouseDragged event)
   (a) get current position of the user's pointer,
   (b) get current time,
   (c) compute a current dragging velocity using previous value pair (time, position),
   (d) update smoothing level according to the resulting value.

3.2.3 Selection visualization

An additional element displaying the dragging velocity value has been added to the GUI

The selection behavior didn't change in general comparing to the latest version of DRAGimation. The one more parameter needed to be shown along with an interaction point and a trajectory itself. It is a velocity value and its influence on the resulting trajectory smoothness. Thus we added an item to the graphic interface that displays a current dragging velocity value. Moreover, because of the fast computation speed of the smoothing algorithm the trajectory is smoothed in the real time and changes very time the dragging speeds up or slows down.

3.3 Pressure

The pressure based selection is considered as the second implicit selection in the scope of this work. The following section contains a detailed explanation of how was this technique applied to DRAGON as a DMVN system.

3.3.1 Selection behavior

Pressure based selection is an another implicit selection technique

After the dragging velocity was described in the previous section, we will introduce the next implicit selection technique. The motion detail of the object however is now defined by a pressure value exerted by user on the screen.

Again, three levels of the motion's resolution are presented

The point based interaction is taken as basic interaction pattern again. Assume, the pressure value is announced to the system every time user interacts with a screen – touches it or drags a pointer along it. In order to correlate the pressure values with levels of detail we again define three levels of the pressure value and their mapping principle to the motion trajectory smoothness:

- Low pressure - high resolution of movement detail (trajectory stays unchanged);
- Middle pressure - medium resolution;
- High pressure - schematically represented movement.

The values assigned to each level however depend on the hardware used for the pressure measurement.

3.3.2 Implementation

A modern pressure sensitive tablet has been used for measurements

Along the modern touchscreen devices we have chosen the one with a pressure-sensitive screen from Wacom. It is a 21.3 inch sized model called Cintiq 21 UX (Figure 3.11), which is widely used for painting. In this master's thesis

however the tablet will be used for the pressure based selection in direct manipulation video navigation.

Figure 3.11: Wacom Cintiq 21UX 54,1 cm (21,3 inch) tablet chosen for the direct manipulation video navigation. Source: Amazon online shop[2] .

We tested the tablet with a simple Cocoa application offered on the Wacom developers support page.[3] This simple program enables a pressure-sensitive painting on the Wacom tablet (Figure 3.12). We have exerted different pressures on the screen, measured the corresponding values, and separated all pressure values into three following intervals afterwards:

- $[0, 0.2)$ for the low pressure;
- $[0.2, 0.5)$ for the medium pressure;
- $[0.5, 1]$ for the high pressure.

The pressure value of the user's pointer can be easily acquired using the tablet's API

The tablet's pressure measurement functionality is equipped with useful API functions so that it can be easily integrated into the DRAGimation code. The core algorithm that creates adaptive trajectories doesn't differ from the one of dragging velocity based selection. It has to smooth trajectories according to the current level of motion detail.

[3] http://www.wacomeng.com/mac/index.html

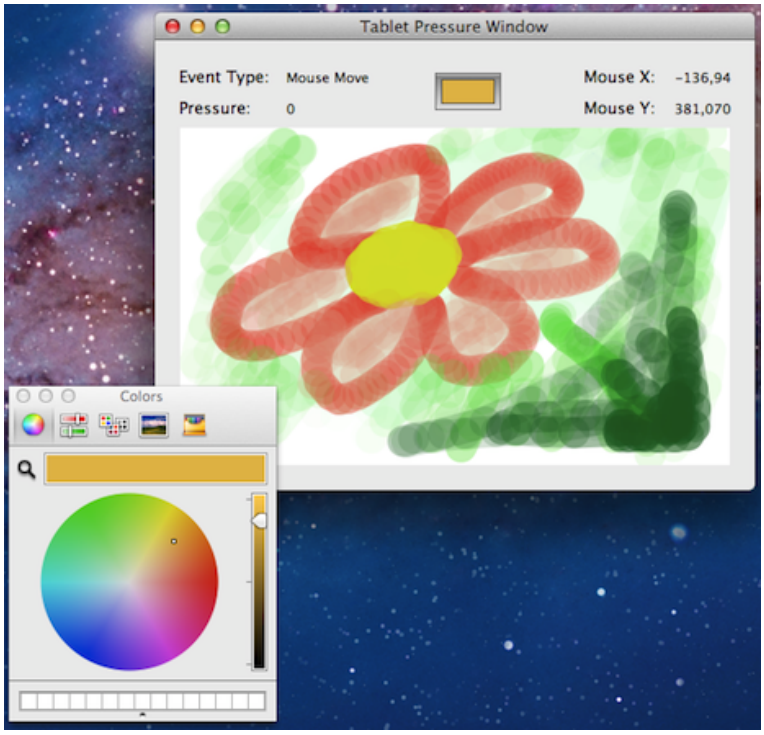


Figure 3.12: Cocoa Simple: sample code for pressure-sensitive drawing on the Wacom tablet.

If the needed level of motion detail is computed every time the pressure or dragging velocity is changed, both selection techniques can use the algorithm for trajectory processing in common.

The final algorithm for the pressure based object selection is looking as follows:

1. Initialization after user clicked on the object (:mouse-Down event)
   (a) get an initial pressure value,
   (b) compute an initial smoothing level,
   (c) ask for trajectories from the server,
   (d) process a trajectory with a corresponding smoothing-mask;
2. While user is dragging an object through the scene (:mouseDragged event)

(a) get current pressure value,

(b) update smoothing level according to the resulting value.

3.3.3 Selection visualization

Along many possible visualizations of the pressure value, a simple numerical indicator has been chosen for GUI

We tried to keep the interface as simple as possible and did not want to complicate it with additional components. First we decided to represent a pressure using a simple circle around the user pointer, which increases if the user exerts a greater pressure on the screen and gets smaller otherwise. However, after implementing this pressure indicator and testing it with other unexperienced users, we have got feedback that the navigation became less enjoyable. The circle started distracting users while dragging, since the pressure may vary at runtime and the corresponding circle size should be always up to date. Despite continuous size variation even at small pressure the circle covered a lot of video details. So we changed the strategy and placed a simple text display on the right panel of the main DRAGimation window (Figure 3.13).

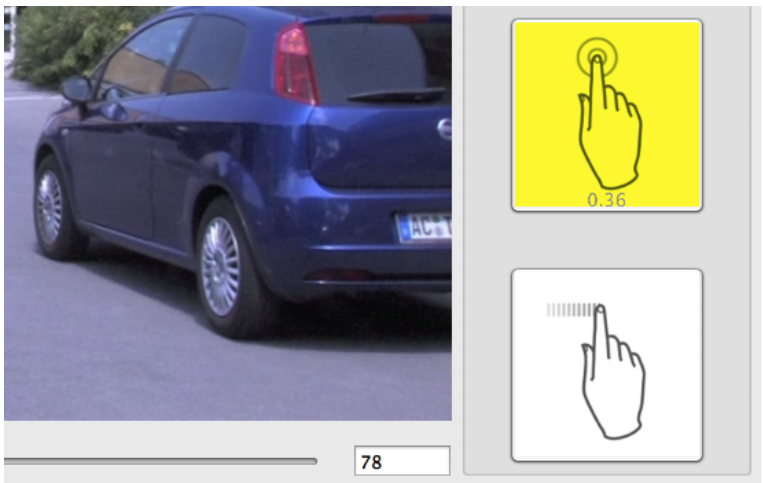


Figure 3.13: DRAGimation: Current pressure value is displayed on the right main window panel (yellow button).

We tested the algorithm on the couple of videos and it has shown trustful stability and high pressure sensitiveness.

Chapter 4

Evaluation

"It doesn't matter how beautiful your theory is,
it doesn't matter how smart you are.
If it doesn't agree with experiment, it's wrong"

— Richard Feynman,
american theoretical physicist

In the first experiment the area selection will be compared to the default point selection

As it has been mentioned in the thesis topic, two issues should be evaluated: object selection and adaptive trajectories applied to DRAGON. First and foremost the new object selection technique (area selection) will be compared to the default one - point selection. This is a pure performance measurement and no user will be needed for this experiment. It is important to conclude, whether the implemented area selection is stable enough to be included into the standard DRAGON package.

In the second experiment both techniques with adaptive trajectories will be evaluated

The second experiment handles the second issue and is supposed to evaluate the addition of adaptive trajectories to the DRAGON's default point selection. Unlike the area selection analysis, this experiment will be carried out with help of a group of users.

4.1 Area selection performance analysis

The goal of the experiment is analyze the quality of the area selection algorithm implemented in the scope of this thesis. Another aim is to compare this algorithm with the point selection used in DRAGON by default.

4.1.1 Experimental setup

Both selection techniques has been compared based on their output trajectories

Before starting the first experiment, we agreed on an output value, which is present in both selection techniques, area and point selection: a **trajectory**. It is therefore reasonable to evaluate the selection technique by comparing the trajectory with a certain standard trajectory: the best possible trajectory of an object, which mostly represents its real object movement, is referred to as "ground truth".

The deviation between the output trajectory and the ground truth has been chosen a comparison criterion

The deviation between the ground truth and any trajectory computed by means of the point selection algorithm might be considered as reference for its quality. Obviously, the average distance of the particular trajectory to the ground truth is inversely proportional to its quality in terms of selection technique. This approach for quality evaluation of computed trajectories might be also used in terms of area selection.

The object selection has been executed many times in order to imitate the user's input error

After applying the point selection to many test videos, it was notice that the object's trajectory varies widely depending on the selected position in the object itself. Thus, the selection should be executed many times to achieve convincing statistics. The random point distribution on the object's surface provides a wide variety of object trajectories. For the generation of random points the gaussian generator class, as used in the area selection, has been implemented (refer to section 3.1).

During the experiment five different values of a standard deviation (variance σ) were used. This includes both cases:

- the random points are concentrated around the initial position of the ground truth trajectory;

- the random points are spread out over the object's surface (or even outside the object's edge).

To reproduce different area selections, not the position of the area, but its size has been computed as a random value.

The experiment was conducted on 15" MacBook Pro with Intel Core 2 Duo processor and 3GB SDRAM. In order to carry out the experiment in the shortest time possible and as automatically as possible, an additional tool was developed for DRAGON (refer to appendix A—"Area selection performance analysis: an additional Cocoa application"). The steps of the following algorithm were performed for this purpose:

1. Initialize experiment:
   (a) compose a set of five different values for the standard deviation σ for the gaussian generator;
   (b) take initial standard deviation value for the point distribution;
2. Request from server and save internally the standard trajectory used as ground truth.
3. Save the initial position of the ground truth trajectory as a mean value for the gaussian function of the generator.
4. Generate ten randomly distributed points using the current standard deviation value and the mean value. Request the corresponding trajectories from the server.
5. Process the received trajectories and compute their average distance to the standard trajectory (ground truth).
6. Generate ten different sized areas using the appropriate variance σ. Request the corresponding trajectories.
7. Compute the average distance to the standard trajectory.
8. Take the next value for σ and start again with step 4.

4.1.2 Results

The area selection showed more stability than the point selection

Generally speaking the area selection showed more stability than the point selection. This fact is mostly obvious in the case of widely distributed initial points (large values of σ). To compare both selection techniques a two-dimensional graphic was composed. The values of the standard deviation used by the gaussian generator are shown by the x-axis. The y-axis on the other hand represents the average distance of a trajectory produced by a particular selection technique to the ground truth.

The experiment has been performed by means of two different videos (Figure 4.1). The corresponding graphs are shown on figures 4.2 and 4.3.

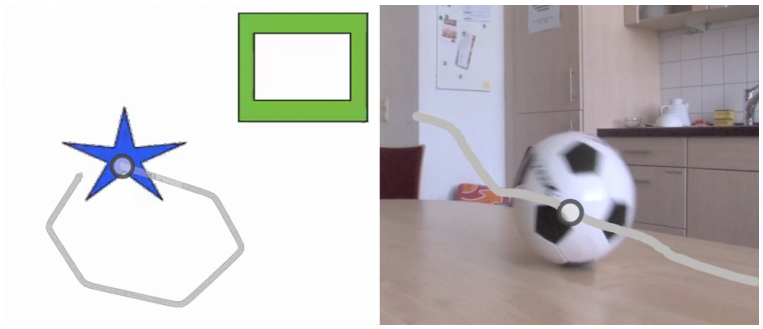


Figure 4.1: Two sample videos used during experiments. The depicted curve represents the ground truth trajectory.

Unlike the point selection, the area selection did not show a strong dependency on the input error σ

Evaluating the graph, one can conclude that the area selection technique slightly depends on the standard deviation σ, whereas the point selection shows a very strong dependence on σ. It means that no matter how proper the object has been selected and which area size has been used, as long as it is not much bigger than the object itself – our algorithm will produce an acceptable trajectory.

Quite the opposite happens to the point selection at large deviances ($\sigma \geq 0.06$). If the input error σ is getting larger, in other words, if the user acts in a hurry and accidentally hits the points on the object's border or even outside it, he might obtain an unpredictable trajectory which may belong to some other object present in the background.

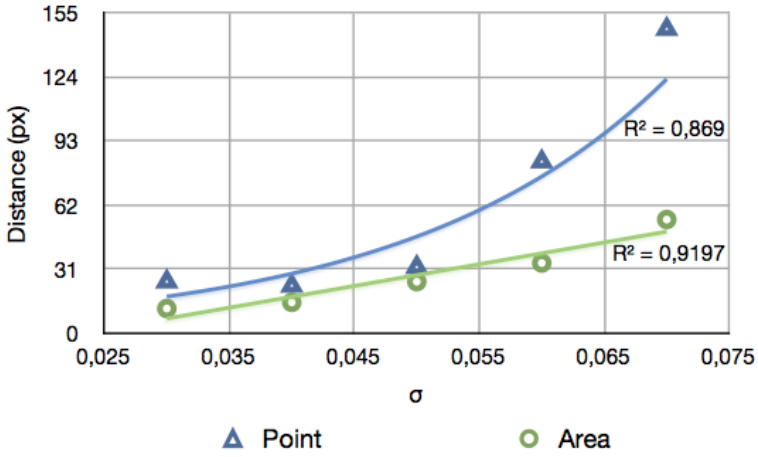


Figure 4.2: The graphical representation of the experiment results (first video, figure 4.1, left).

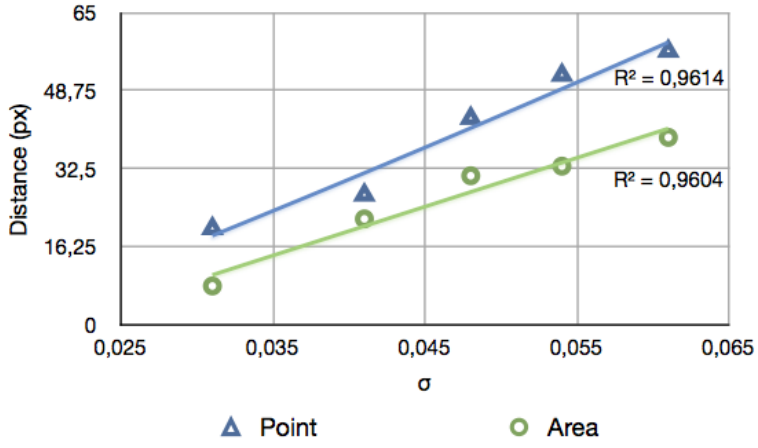


Figure 4.3: The graphical representation of the experiment results (second video, figure 4.1, right).

4.2 Adaptive trajectories evaluation

A user study has been conducted with a group of users

In this experiment only two selection techniques with included adaptive trajectories (pressure and dragging velocity) will be compared with a default point selection. Unlike the evaluation of the area selection, this experiment will be executed on a group of users.

4.2.1 Hypothesis

The user study has been conducted on 15 participants. The final goal of the user study consisted in verifying two following hypotheses:

H1 Using adaptive trajectories in a DMVN system, any video navigation task can be solved in a shorter time than using a default point selection.

H2 The usage of pressure based selection leads to a faster task completion compared to dragging velocity.

In the following paragraphs the methodology and the results of the experiment will be presented.

4.2.2 Participants

In order to evaluate adaptive trajectories performance, 15 test persons with different backgrounds have been invited to the user study.

The participants identified their demographic and educational characteristics. The average age of respondents is approx. 22.2 years old. 8 respondents are female; 7 are male.

The majority of the participants has been studied Computer Science

When asked to identify their field of expertise, the majority of the participants indicated that they are Students of (or have studied) Software Systems Engineering (33.33%), followed by those who identify as Computer Sciences (26.66%), Economics (20%), Electrical Engineering (13.33%) and Physics (6.66%) (Figure 4.4).

Most participants of the user study did not have any experience in DRAGON

It is important to point out that only 20% of the participants were graduate students. Eleven respondents indicated that they are not familiar at all with DRAGON. Other two persons are in an intermediate command of it (used DRAGON approximately 3-5 times). Only two persons reported an expertise on DRAGON. They have already used DRAGON more than 10 times and extended its source code (Figure 4.5).

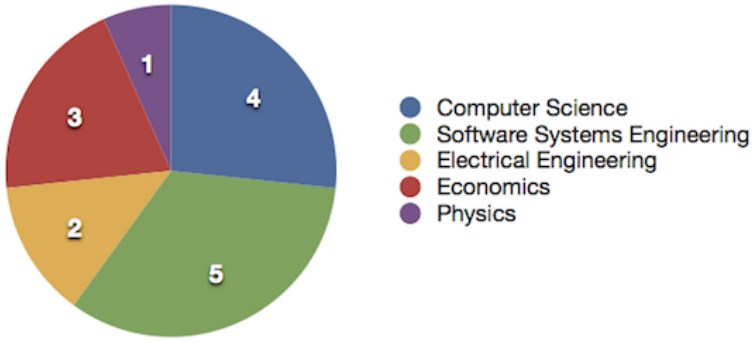


Figure 4.4: Field of study of the experiment participants. The overall majority belongs to M.Sc. Computer Science or M.Sc. Software Systems Engineering.

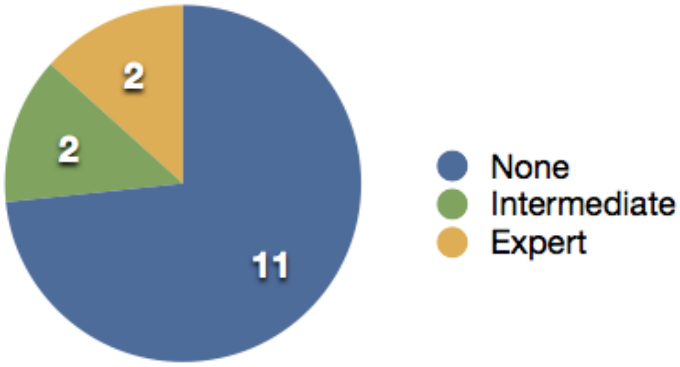


Figure 4.5: Experience level of the users participated the study related to DRAGON usage.

4.2.3 Tasks and methodology

For the user study we prepared a set of four videos (Figure 4.6). A simple object was presented in each video, which has to be navigated along its movement trajectory:

- a red ball;
- a yellow sport car;
- a white fish;
- a black cat.

Figure 4.6: Sample videos used for navigation via DRAGON during the user study.

Each navigation task should be completed three times, using the one of three selection techniques each time (Figure 4.7):

1. Default point selection.
2. Pressure based selection.
3. Dragging velocity based selection.

To avoid the learning effect, we asked users to apply the selection techniques each time in a diverse order. The performance of each selection technique was evaluated on the basis of the task completion time. The task is considered successfully accomplished if the following conditions are satisfied:

- Video 1: The red and the blue ball touch each other;
- Video 2: The yellow sport car passed the blue "RBS"-billboard;
- Video 3: The white fish achieves the left pane of the aquarium and returns to its start position;
- Video 4: The black cat throws the white ball and the ball touches the left side of the parked white car.

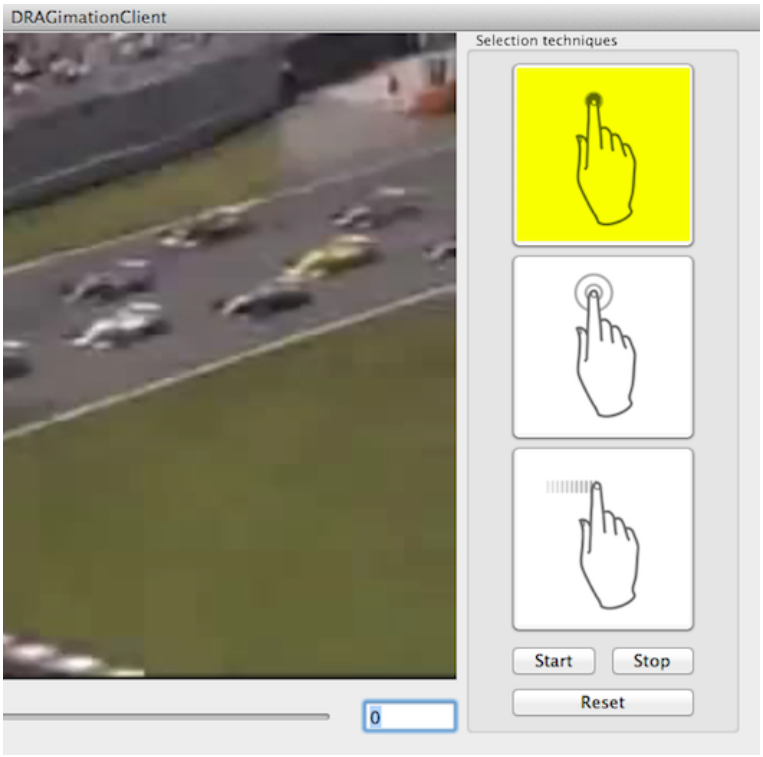


Figure 4.7: DRAGON: The graphic user interface with three buttons indicating the chosen selection technique: default, pressure, or dragging velocity.

Also short introduction of DRAGON functionality was made, since most users did not have any experience working with it. Users could also navigate a training video for a couple of minutes in order to get used to the three selection techniques. A screen recording was made during the interaction.

Following hardware has been used for the experiment:

- 15″ MacBook Pro with Intel Core 2 Duo processor and 3GB SDRAM;
- 21.3″ Wacom tablet (model Cintiq 21UX).

4.2.4 Questionnaire

A short post-session questionnaire has been used in order to evaluate the users' perception of the new techniques

Beside a pure performance evaluation using the time of the task completion, both new selection techniques with adaptive trajectories must undergo a qualitative analysis. This analysis should take into account a user acceptance and a personal opinion of the participants regarding adaptive trajectories.

In accordance with principles introduced by Converse and Presser [1986], a set of seven multiple-choice questions has been prepared for this purpose. Further questions, inspired the System Usability Scale (SUS) [Brooke [1996]], have also been included into the questionnaire. The complete post-study questionnaire is enclosed in appendix B—"User experiment questionnaire".

After finishing all tasks, a short post-session interview face-to-face is held. During this post-session users were asked to fill out the questionnaire and express their personal opinion. All user comments were then gathered for the later analysis.

Since most people would prefer anonymity, all questionnaires do not contain any personal data except the elements listed below:

- gender;
- age;
- field of study;
- experience working with DRAGON.

Every participant as given a personal ID to simplify the data analysis in accordance with the provisions of protection of personal data.

4.2.5 Results

The users completed the tasks 3 and 4 significantly faster using adaptive trajectories

Completion times A one-way ANOVA has been conducted to evaluate the task completion times, which have been minuted during the user study. Tasks one and two did not show any significant differences while comparing the task completion time of every single technique (Figure 4.8). A significant difference however was observed for the third task: $F(2, 42) = 5.889, p = 0.00557$. A pairwise t-test conducted afterwards revealed that users could accomplish the task much faster using the dragging velocity based selection:

1. Dragging velocity: $M = 17.95$, $SD = 4.94$;
2. Pressure: $M = 23.31$, $SD = 6.88$ ($p = 0.0315$);
3. Default: $M = 24.94$, $SD = 5.51$ ($p = 0.0063$).

The fourth task showed also significant differences: $F(2, 42) = 8.415, p = 0.000845$. Post analysis with a pairwise t-test revealed that both techniques, dragging velocity ($p = 0.0017$) and the pressure based selection ($p = 0.0035$), are significantly faster than the default point selection:

1. Dragging velocity: $M = 13.62$, $SD = 3.80$;
2. Pressure: $M = 14.26$, $SD = 3.53$;
3. Default: $M = 19.75$, $SD = 5.81$.

The significant difference was observed only in two cases for the dragging velocity and in a single case for the pressure. Therefore, the hypothesis H1 cannot be confirmed.

No significant differences between the pressure based selection and the dragging velocity have been observed

To prove the second hypothesis, both selection techniques with adaptive trajectories were compared using a pairwise t-test. The significant deference between the dragging velocity and the pressure was observed only in the third case ($p = 0.0315$). But on the other hand the participants were faster when using dragging velocity. Tasks 1, 3 and 4, however, did not show significant differences in completion

time between both selection techniques. Thus, the hypothesis H2 was rejected as well.

We also have analyzed the screen recordings, which have been made during the user study in order to understand why these techniques did not bring significant effort.

A new set of levels of motion details is needed to improve the overall effort of adaptive trajectories

We observed that the users often became disoriented in the video, if they increased the exerted pressure or the dragging velocity. Since there are only three levels of motion detail (low, medium, high), the object's trajectory changed quite abruptly during video navigation. To prevent this effect, a bigger amount of these levels is needed. Further experiments are needed to reevaluate the adaptive trajectories with the new levels of motion detail.

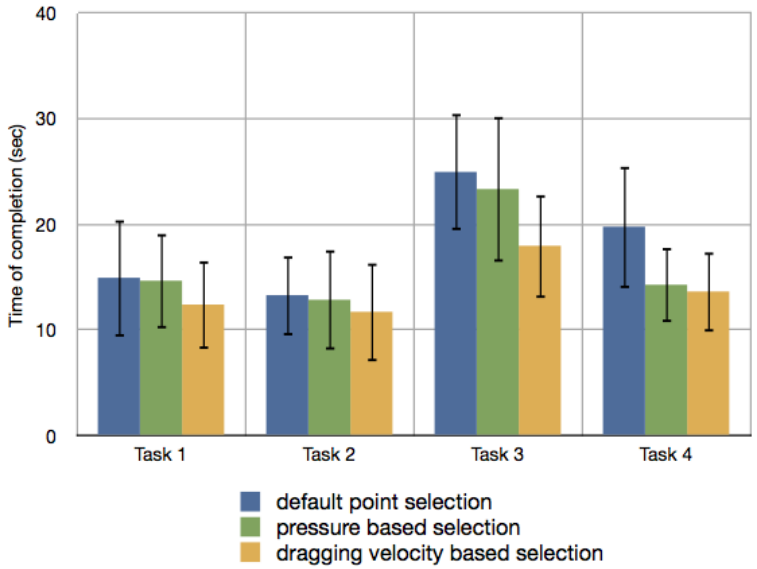


Figure 4.8: Graphic representation of the evaluated task completion time: the chart shows the average time of the corresponding task completion in seconds. The measurements were carried out over 15 participants with and without DRAGON experience.

Quality analysis Apart from the qualitative analysis of the DRAGON performance, we also analyzed the user's perception of the new selection techniques integrated in DRAGON by means of adaptive trajectories.

The users were asked to fill out a post session questionnaire after the experiment (refer to appendix B—"User experiment questionnaire"). All the participants were confirmed in fact, that DRAGON benefits from adaptive trajectories (Appendix B, question 1). Test persons felt comfortable using proposed selection techniques and did not think they were overwhelmed while executing video navigation tasks. All 15 users did not find these techniques excessively complex (Appendix B, question 2).

The users concluded that DRAGON benefits from adaptive trajectories

Answering the third question 13 participants were confident about the success of these techniques on their establishment in the everyday's life of average users (in regards to the familiarity with computer sciences) and did not think they would require any training. However, two users claimed, they needed some more time to understand and keep going with pressure and dragging velocity.

Both new selection techniques, the pressure and the dragging velocity, are easy to understand

The majority of the users (10 persons) identified the pressure sensitive point selection as more enjoyable and usable selection technique when compared to the dragging velocity in question 6. However, answering the last question three participants thought, dragging velocity has the potential to become a multipurpose selection technique, and other five were fairly confident, that an appropriate selection technique has to be chosen depending on a task definition (default selection for simple trajectories, dragging velocity otherwise). Other seven persons have chosen the pressure selection, but even they believed that the dragging velocity might evolve into a global technique since it can be implemented on every portable device. Further users' comments are presented in the next paragraph.

According to the users' answers, the dragging velocity might be used as a multipurpose selection technique for DMVN systems since it does not require any additional hardware

Users' comments To avoid ambiguity while analyzing the post-session questionnaire, all users' comments were acquired. Answering the last question, seven participants has chosen the pressure selection. They found much easier to reproduce a particular pressure level than a dragging speed. Only three users mentioned, however, that it has been uncomfortable for them to press one the screen and to move a pointer simultaneously.

Some users found it easier to reproduce a certain level of detail using pressure

Five users mentioned that the dragging velocity is better suited for simple straight trajectories

Other five users concluded, the dragging velocity can be successfully applied to simple straight trajectories (such as in task 2, or 3), whenever only the pressure based selection can help while navigating along complicated trajectories such as in task 4. There is no need to apply the adaptive trajectories for simple straight movements such as in task 1.

All users concluded that the dragging velocity should be included in DRAGON as an additional selection technique

To conclude, there is one important aspect all participants did agree on (even those, who liked and chose the pressure selection at the end of the experiment): there are not many devices with pressure sensitiveness, so if DRAGON should be popularized, it has to contain the dragging velocity. Not to mention the area selection, this kind of point selection with adaptive trajectories is definitely supported by all portable devices and desktop computers.

Chapter 5

Summary and future work

"Big results require big ambitions."

—Heraclitus,
greek philosopher

Finally, a brief review is given in the first part of this chapter to summarize the work done within this thesis. The second part provides an outlook on how the implemented algorithms, which might be further improved to provide more functionality.

5.1 Summary and contributions

The goal of this work was to eliminate all possible misunderstandings between the user and the DMVN system in terms of object selection

The aim of this work was to close the gap existing in the interaction between the user and the direct manipulation video navigation (DMVN) system. During the initial research phase we investigated all direct manipulation video navigation systems existing at that moment. As a result, the gap, referred to as "gulf of execution", has been found in every system, not to mention an instability of their tracking algorithm under certain conditions. For further investigations DRAGON has been chosen as an example of a DMVN system.

First and foremost the area selection has been implemented

To achieve the aimed goal, an area selection has been implemented first. A DMVN system for video annotation presented by Goldman et al. [2007] provides an area selection as well. Its tracking algorithm, however, fails for small or occluded objects. On the other hand, DRAGON considers movements of every pixel and is therefore resistant to occlusions and small moving objects. Analogous to the particle grouping approach of Goldman et al. [2007], a K-means trajectories clustering is implemented to prevent unstable trajectories inside the selected area.

The object selection in the DMVN system became more transparent and predictable after extending it by the area selection

Our algorithm allows the user to always keep navigation under control and gives him a choice on which part of an object will be tracked by a DMVN system. As a result, the DMVN system became more transparent and predictable. An experiment, conducted to compare the area with the default point selection, showed more stability of the area selection. In case of the example with the rolling football in chapter 1—"Introduction" (Figure 1.7) one can say, that using the new algorithm, the user is always able to move the ball along the straight line by selecting the entire ball, not the point only.

Afterwards, DRAGON has been extended by two point selection techniques with adaptive trajectories

Another way to provide users with more freedom of action, was to extend the default point selection by means of adaptive trajectories. Two approaches has been integrated into DRAGON for this purpose: the pressure sensitive navigation and the dragging velocity based navigation. Both methods allow the user to control the level of motion details. To reproduce a certain level of motion details, the object's trajectory has been smoothed via low-pass IIR filter.

The user study revealed that DRAGON benefits from the pressure based selection and the dragging velocity

The user study conducted afterwards revealed that it makes perfect sense to attach adaptive trajectories to the default point selection. A great deal of important feedbacks has been gathered and analyzed in section 4.2.5—"Results". The participants of the user study claimed, that the navigation using adaptive trajectories is more predictable and enjoyable than a usual point selection. The participants were convinced that these techniques, especially the dragging velocity, can be successfully integrated into every DMVN system.

5.2 Future work

Even though the intended tasks have been successfully accomplished, there is still a high need for improvement, especially when it comes to the object selection techniques in a DMVN system, such as DRAGON. Potential and feasible improvement possibilities are thoroughly presented below.

5.2.1 Region selection: advanced clustering

To achieve a better stability, another trajectory clustering approach is needed

Regarding the implementation of area selection in section 3.1.2—"Implementation", the trajectory clustering algorithm is considered again. For the sake of a simple implementation and creating a first prototype in a shortest period of time, the K-means clustering has been implemented. However, the overall functionality might have been improved by an another clustering approach, e.g., a mean shift clustering.

5.2.2 Dragging velocity: further implementation possibilities

Other implementations, such as a trajectory pyramid or a trajectory clustering, have to be compared with the one we presented in this thesis

In the second part of the thesis the implementation of the adaptive trajectories in DRAGON was treated. The user study revealed, that the dragging velocity unlike pressure sensitive dragging, has a big potential to evolve into a multipurpose selection technique for DMVN systems. Thus, it makes sense to investigate this technique deeper. Along with the trajectory smoothing via low-pass filtering other approaches can be applied to distinguish between levels of motion details, such as a trajectory pyramid, or trajectory clustering. Both techniques are briefly described in section 3.2.2—"Implementation".

In case of the trajectory pyramid, a sample video should undergo an additional preprocessing stage. Before using the video in DRAGON, all pixel trajectories should be prepared in several copies. Each copy of the motion trajectory is smoothed according to the level of the video pyramid.

It is quite important to investigate, whether this technique will result in a faster real-time navigation.

On the other hand, the trajectory clustering does not require any additional preprocessing steps. The entire computation, however, takes place at runtime. It would be a good idea to compare both described approaches with the first implementation of the dragging velocity that was made in the scope of this work.

5.2.3 Further user studies on object selection techniques in DRAGON

Further evaluations are needed to investigate the implemented selection techniques

The first experiment (refer to 4.1—"Area selection performance analysis") has already shown that the implemented area selection provides more stability over the default point selection in DRAGON. The experiment, however, aimed to prove the robustness of an algorithm, not the users' perception. The question of, whether users' would prefer this technique over the point selection, or a clever combination is needed, must be answered.

Appendix A

Area selection performance analysis: an additional Cocoa application

An additional software tool has been developed within this work (Figure A.1). It is supposed to assist the authors while executing the area selection performance analysis.

Once all input parameters have been set, a new point (or area) selection can be generated. The buttons ”Next Point” (or ”Next Area”) are used for this purpose. The average distance is calculated automatically after the button ”Avg Distance” has been pushed.

The field ”Error” represents a current value of a variance used within the random numbers generator. The new value can be set after ”Update” has been pushed.

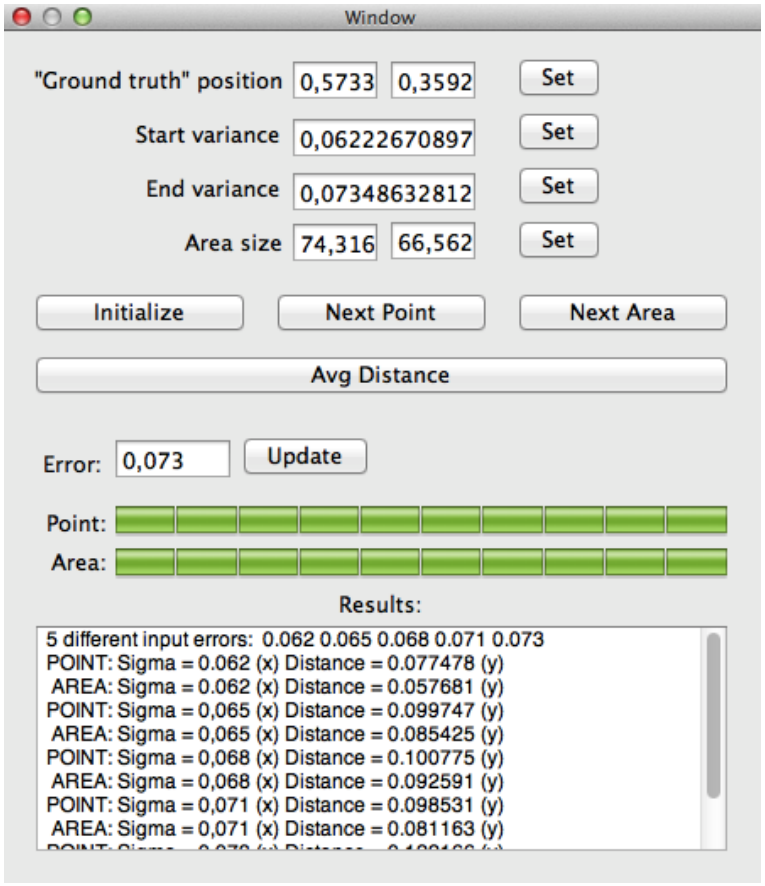


Figure A.1: DRAGON: An additional software tool developed for the area selection analysis.

Appendix B

User experiment questionnaire

After the user executed four navigation tasks, he was asked to fill out the following questionnaire. While filling out the questionnaire users were allowed to think aloud. Their comments were analyzed and presented in paragraph 4.2.5—“Results”.

User test **Adaptive Trajectories in DRAGON**
Participant ID: _____
Date: ___.___.___

I hereby agree that my data can be evaluated anonymously for research purposes at RWTH Aachen University. I have been informed that I may refuse to answer certain questions. I am also allowed to quit the user study at any time and without any penalty.

_____
(signature)

Personal information	
Sex	☐ Male ☐ Female
Age	
Field of study	
DRAGON Experience	☐ None ☐ Intermediate ☐ Expert

Post session questionnaire

1. I do not think DRAGON benefits from adaptive trajectories.

   ☐ Yes ☐ No

2. I found the new selection techniques (pressure and dragging velocity) unnecessarily complex.

   ☐ Yes ☐ No

3. I would imagine that most people would learn to use these new selection techniques very quickly.

   ☐ Yes* ☐ No

   * If you have chosen “Yes”, please continue with 6.

4. I needed to train a lot before I could get going with the **pressure** based selection.

   ☐ Yes ☐ No

5. I needed to train a lot before I could get going with the **dragging velocity** based selection.

   ☐ Yes ☐ No

6. I think the pressure based selection was easier to use than the dragging velocity.

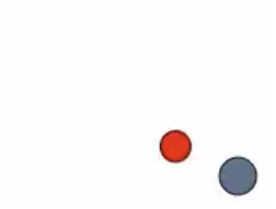
   ☐ Yes ☐ No

Figure B.1: Post-session questionnaire (page 1 of 2).

User test **Adaptive Trajectories in DRAGON**
Participant ID: _____
Date: ___.___.___

7. I think that

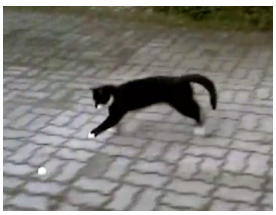
- ☐ **pressure** based selection has the potential to evolve into a global technique, which might be used in every selection case.
- ☐ **dragging velocity** based selection has the potential to evolve into a global technique, which might be used in every selection case.
- ☐ the choice of the appropriate technique must be made depending on the case at hand:



☐ Default point selection
☐ Pressure
☐ Dragging velocity

☐ Default point selection
☐ Pressure
☐ Dragging velocity

☐ Default point selection
☐ Pressure
☐ Dragging velocity



☐ Default point selection
☐ Pressure
☐ Dragging velocity

Figure B.2: Post-session questionnaire (page 2 of 2).

Bibliography

J. Brooke. *SUS: A quick and dirty usability scale*. Taylor and Francis, London, 1996.

Dan Buzan, Stan Sclaroff, and George Kollios. Extraction and Clustering of Motion Trajectories in Video. In *Science*, 2004.

Chi-Cheng Cheng and Chung-Hsing Ho. Improved Visual Tracking Using the Technique of Image Pyramid. In *2009 IEEE International Conference on Robotics and Biomimetics*, pages 659 – 664, Guilin, China, 2009.

Jean Converse and Stanley Presser. *Survey Questions: Handcrafting the Standardized Questionnaire*. Sage University Paper series onQuantitative applications in the Social Sciences, Beverly Hills, CA, 1986.

Pierre Dragicevic, Gonzalo Ramos, Jacobo Bibliowitcz, Derek Nowrouzezahrai, Ravin Balakrishnan, and Karan Singh. Video browsing by direct manipulation. *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, pages 237 – 246, 2008.

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

Dan B Goldman, Brian Curless, David Salesin, and Steven M Seitz. Interactive Video Object Annotation. *ACM Computing Surveys*, pages 1–7, 2007.

Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, August 1981.

Takeo Igarashi and Ken Hinckley. Speed-dependent automatic zooming for browsing large documents. *Proceedings of the 13th annual ACM symposium on User interface software and technology - UIST '00*, pages 139–148, 2000.

Thorsten Karrer, Malte Weiss, Eric Lee, and Jan Borchers. DRAGON: A Direct Manipulation Interface for Frame-Accurate In-Scene Video Navigation. *Direct*, pages 247–250, 2008.

Don Kimber, Tony Dunnigan, Andreas Girgensohn, Frank Shipman, Thea Turner, and Tao Yang. Trailblazing: Video Playback Control by Direct Object Manipulation. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 1015–1018. Ieee, July 2007.

Ce Liu, Jenny Yuen, Antonio Torralba, Josef Sivic, and William T Freeman. SIFT Flow: Dense Correspondence across Different Scenes. *Proceedings of the 10th European Conference on Computer Vision: Part III*, pages 28–42, 2008.

Donald A Norman. *The Design of Everyday Things*, volume 16. Basic Books, 2002.

William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical Recipes in C*, volume 29. Cambridge University Press, 1992.

Gonzalo Ramos and Ravin Balakrishnan. Fluid interaction techniques for the control and annotation of digital video. In *Proceedings of the 16th annual ACM symposium on User interface software and technology - UIST '03*, volume 5, pages 105–114, New York, New York, USA, 2003. ACM Press.

Ben Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. *Human-Computer Interaction*, 08:461–467, 1983.

Ben Shneiderman and Christopher Ahlberg. The Alphaslider: A Compact and Rapid Selector. *Human Factors in Computing Systems*, pages 365–371, 1994.

Gabriel Taubin. Curve and surface smoothing without shrinkage. *IEEE International Conference on Computer Vision*, page 852, 1995.

Siripong Treetasanatavorn, Jörg Heuer, Uwe Rauschenbach, Klaus Illgner, and Andre Kaup. Temporal video segmentation using global motion estimation and dicrete curve evolution. In *2004 IEEE International Conference on Image Processing*, pages 385–388, 2004.

Remi Trichet and Bernard Merialdo. Fast Video Object Selection for Interactive Television. In *2006 IEEE International Conference on Multimedia and Expo*, pages 989–992. Ieee, 2006.

Wei-Qi Yan and Mohan S Kankanhalli. Detection and removal of lighting & shaking artifacts in home videos. In *Proceedings of the tenth ACM international conference on Multimedia - MULTIMEDIA '02*, page 107, New York, New York, USA, 2002. ACM Press.

Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking. *ACM Computing Surveys*, 38(4):13–es, December 2006.

Index

Typeset February 9, 2012