Faculteit Informatietechnologie

Vakgroep Informatica

Magic Lenses for Visualising Multimedia Data

door

Raf Ramakers

Promotor: Prof. dr. Kris Luyten

Begeleiders: Mieke Haesen, Jan Meskens

Eindwerk voorgedragen tot het behalen van de graad van

bachelor in de informatica/ict

Academiejaar 2009–2010

Acknowledgments

I would like to thank a number of people who have made this thesis possible.

First of all, I would like to thank Prof dr. Kris Luyten for supervising this thesis.

Besides, special thanks go to Mieke Haesen and Jan Meskens. They introduced me to the interesting problems discussed in this work. During the past half year, they guided me through this thesis with ideas, advice, and feedback. For every question I had, they made time for me.

More personally, I would like to thank my parents for giving me the opportunity to study, and for supporting me throughout my whole life and education. Finally, I thank my friends for the moments of humor and relaxation during the creation of this thesis.

Finally, I would like to show my gratitude to Luc Adriaens for recording videos about this work and to the EDM researchers who anticipated in the usability test.

Raf Ramakers, June 2010

Magic Lenses for Visualising Multimedia Data

door

Raf Ramakers

Eindwerk voorgedragen tot het behalen van de graad van

bachelor in de informatica/ict

Academiejaar 2009–2010

Promotor: Prof. dr. Kris Luyten

Begeleiders: Mieke Haesen, Jan Meskens

Faculteit Informatietechnologie

Universiteit Hasselt

Vakgroep Informatica

Summary

With a large amount of video data available, finding an event of interest can be very complex and time consuming. Even professional users have to skim many hours of stored video data before they find a suitable fragment.

In this thesis, direct manipulation tools called *magic lenses* will be used to explore video data very fast and intuitive. Furthermore, the composition of magic lenses will be examined to provide more complex queries for finding events of interest.

A magic lens is a generalization of the lens metaphor. Therefore, the application would have more potential if users can interact with these tools in the same way as they can interact with lenses in real world. For this reason, the application will be developed on a multi-touch screen. These displays are available in an increasing number of technologies and require new ideas for interaction, to improve User Experience.

To demonstrate the possibilities of magic lenses and multi-touch interaction, a new rich media application for video exploration will be created.

Contents

Acknowledgments **i**

Summary **ii**

Contents **iii**

1 Introduction **1**
1.1 Purpose of this Thesis 1
1.2 Overview of this Thesis 2

2 Multi-touch Displays **3**
2.1 History 3
2.2 Analysis of a Multi-Touch Display 5
2.2.1 Light Sources 6
2.2.2 Optical Sensors 7
2.2.3 Visual Feedback 8
2.3 Optical Multi-touch Technologies 9
2.3.1 Frustrated Total Internal Reflection (FTIR) 9
2.3.2 Diffused Illumination (DI) 11
2.3.3 Technique Comparison 14
2.4 Multi-touch detection and processing 16
2.4.1 Touchlib 16
2.4.2 FTIRCap 16

3 Multi-touch User Experience **18**
3.1 Gestures 18
3.1.1 Direct Gestures 19
3.1.2 Symbolic Gestures 20
3.1.3 Gestures and Multi-User Aspects 22

3.1.4 Problems using gestures 24
3.2 Input visualization techniques 25
3.2.1 Touch Feedback Ambiguity Problem 26
3.2.2 Input Visualization 27

4 AMASS++ Archive **33**
4.1 AMASS++ Summarized Data 34
4.1.1 Summarization Constraints 35
4.2 Parsing The Summarized Data 35

5 Magic Lenses **38**
5.1 The Magic Lens Concept 38
5.2 Magic lenses and Data Filtering 40
5.3 Advantages of Magic Lenses 41
5.4 Magic Lenses for Visualising Video Data 42
5.4.1 Searching for Fragments 43
5.4.2 Searching for Videos 45
5.5 Interacting with Magic Lenses 48

6 Implementation **52**
6.1 Technologies 52
6.1.1 .NET/C# 52
6.1.2 WPF/XAML 53
6.2 Data Structure 54
6.2.1 Model-View-ViewModel pattern 55
6.2.2 Handling Input 57
6.3 Implementing Lenses 63

7 Usability Tests **65**
7.1 The purpose 65
7.2 Methodology 66
7.2.1 The Procedure 66
7.2.2 Participants 66
7.2.3 Introduction 66
7.2.4 Tasks 67
7.3 Findings and Results 68
7.3.1 Results of the questionnaire 68
7.3.2 Findings 68
7.3.3 Overall Results 71

8 Future Work **72**
8.1 More Types of Lenses 72
8.2 Dept of the lenses 73
8.3 Visualizing lenses on top of a video 73
8.4 Creating an *AND*-relation 75
8.5 Improving the Visualization of Relations 76
8.6 Filtering on different levels 76

9 Conclusion **80**

Samenvatting (Dutch Summary) **82**

A Usability Test: Instruction Document (in Dutch) **84**
A.1 Inleiding 84
A.2 Verloop van de test 84
A.3 Taken 85
A.4 Vragenlijst 85

Bibliography **89**

List of Figures **94**

List of Tables **97**

Chapter 1

Introduction

Nowadays, videos are available from an increasing number of sources, such as webcams, home videos, surveillance systems and television retransmissions. With such a large amount of video data finding, analyzing and processing a fragment of interest in these videos can be very complex and time consuming. There are a number of automatic techniques available for event detection and object tracking but these do not solve the problem. While they may reduce the amount of data significantly by converting a raw video stream into a set of abstract objects, there is still a large amount of data for a person to deal with. On the one hand, when dealing with personal video data, it may be possible to exploit your own knowledge to help managing the search. On the other hand, when dealing with an arbitrary video or videos in the archive of a news station, finding an event is similar to looking for a needle in a haystack.

1.1 Purpose of this Thesis

Finding a suitable video fragment in a video archive is mostly a complex task. Information visualization techniques address these problems by providing graphical presentations of the data and direct manipulation tools for exploring this data. In this thesis, a novel interaction technique called *magic lenses* will be used to find an event of interest in a video.

Magic lenses are semi-transparent User Interface elements to visualize specific data underneath, for example parts of a video. When moving a lens over a video, fragments of interest can be highlighted. This is just one example of the possibilities that these direct manipulation tools can provide. In this work, advanced features of magic lenses such as building queries will be examined as well as the interaction with these tools for finding fragments of interest in a vast archive of video data. Furthermore, an application demonstrating these new interaction techniques will be created. Finally, a usability test will evalulate the value of these new ideas.

When working with magic lenses, all actions are performed in the same spatial area because these lenses manipulate objects that are located directly underneath. Accordingly, it is more intuitive if users perform their actions on that spatial area instead of using traditional input devices. Therefore, the application will be created on a multi-touch display.

1.2 Overview of this Thesis

First of all, chapter 2 gives an introduction into the currently available camera based multi-touch techniques and their components. These new platforms bring new challenges, some that can be partially solved using current software design paradigms, but many that will require applying new ideas from research in Human Computer Interaction. Therefore, chapter 3 will cover some essential parts of applications on a multi-touch device. In chapter 4 the video archive used in this work will be introduced and chapter 5 explains the power of magic lenses for filtering this archive. Eventually, chapter 6 describes the technologies and data structure needed to implement all these new interaction techniques. Finally, the results of the usability tests are analyzed and interaction problems are revealed in chapter 7. On top of that, chapter 8 and 9 describe respectively the future work that can be done in this research area and the conclusion of this thesis.

Chapter 2

Multi-touch Displays

Interactive graphics devices that combine camera and tactile technologies for direct on-screen manipulation, are known as multi-touch displays. This technology exists since 1970 and has been available in different forms. Multi-touch displays allows users to control multiple input pointers independently. Depending on the size of the display, multiple persons are allowed to interact with the same surface at the same time. Due to the improvement of processing power of modern desktop computers, multi-touch technology does not longer require expensive equipment.

2.1 History

Since 1970 (see figure 2.1), several research groups have done research on touch sensitive surfaces. Patents from that time [[23],[24],[25]] demonstrate how camera based touch sensitive surfaces can be constructed. Multi-touch technologies have a long history, which is comparable to the mouse that was invented in 1965 and became ubiquitous in windows 95 [3].

In 1982, the *University of Toronto's Input Research Group* developed the first real human-input multi-touch system [8]. This system used a frosted-glass panel with a camera placed behind this glass. When one or more fingers touch the glass, the camera detects the action as one or more black spots on

a white background, which the system considers as input. Since the size of a dot was dependent on pressure, the system was pressure-sensitive as well. Without this pressure sensitivity, a touch based system would only have two states: touch or no touch. When a touch table is able to sense different levels of pressure, it becomes a three-state model like all other input devices [11]:

- State 0, no physical contact is made with the input device;
- State 1, the input device is tracking;
- State 2, the input device is dragging.

A breakthrough occurred in 1991, when Pierre Wellner published a paper on his multi-touch *Digital Desk* [5]. This system uses an early front projection tabletop system that uses optical techniques to sense both hands/fingers as well as certain objects, such as paper-based controls and data. This work clearly demonstrated multi-touch concepts such as two finger scaling and translation of graphical objects using either a pinching gesture or a finger from each hand.

In 2005, Han presented a low cost camera based multi-touch sensing technique [12]. Han highlighted the potential for multi-touch interaction in the development of the next generation of interfaces. The system uses the technique called Frustrated Total Internal Reflection (FTIR, described in more detail in Section 2.3.1) which involves trapping infrared light in a sheet of acrylic. When touching this sheet, the touched spot will frustrate and trapped light cause it to leak out the sheet. A camera records where infrared light is leaking and recognizes touch. Since Han's work, there was an explosion of interest in multi-touch interaction. Hardware implementations of multi-touch interaction have allowed for the low cost development of surfaces and enabled much research exploring the benefits of multi-touch interaction.

In 2007, Apple launched a new cellular phone: the Apple iPhone. With this product, Apple introduced multi-touch technology to control the phone. The iPhone senses touch by employing electrical fields. On touch, changes in

this electrical field are measured. This allows the iPhone to detect the part of the phone that is being touched. With the iPhone, Apple reintroduced prior multi-touch concepts such as the pinch gesture.

Later in 2007, Microsoft demonstrated their version of a multi-touch table, the MS Surface. This table looks like a coffee table with an interactive surface. In this technique, the display is illuminated with infrared light from the back. When a user touches the table, it will reflect infrared light that can be captured by the cameras inside the table. Because of the use of multiple cameras, the input resolution is high enough to detect objects. Microsoft has demonstrated how fiducial markers can be used to tag objects and allow specific interaction with the table [29].

We can say that multi-touch input has hit the mainstream. The demand for these systems is pushing multi-touch to modalities ranging from camera-based wall-sized installations seen on CNN to the iPhone's capacitive screen. Figure 2.1 provides an overview of some very important points in the evolution of multi-touch displays.

2.2 Analysis of a Multi-Touch Display

Over the years many touch technologies have been developed. These technologies can be divided in two major touch technology trends: the non-optical and optical approach. Many non-optical approaches have already found their

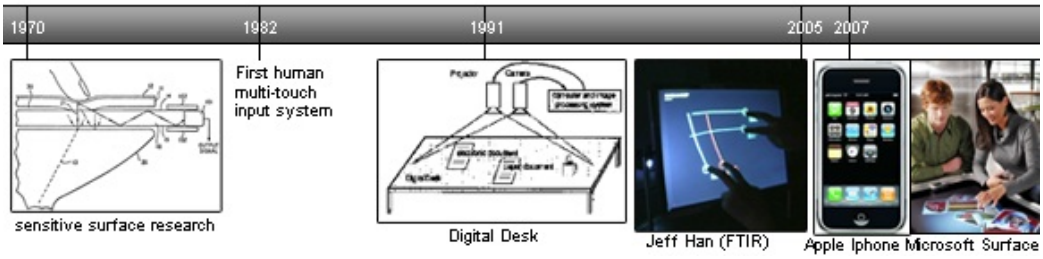

Figure 2.1: Some very important points in the technical evolution of the touch technology.

way into consumer products, albeit in smaller interactive surfaces (such as mouse pads on laptops and touch-screens on phones). On the other hand many of these technologies require industrial fabrication facilities. Therefore the focus in this work is entirely on the optical approach.

This section describes the most important parts of optical multi-touch approaches. In essential, there are four reasons for the popularity of this technology: scalability, low cost, ease of setup and due to their simple configuration they also have the potential to be very robust. Many technologies in this multi-touch approach shares some common parts: an optical sensor (typically one ore more cameras), a light source, and visual feedback in the form of projection or LCD.

2.2.1 Light Sources

If a hand touches a surface that is not overlaid on the screen it is called a touch tablet or touch pad. On the other hand, touch screens are systems where users are pointing exactly where they see the object. As a consequence, the camera in a touch screen records the same area where the projection occurs. It is important that the camera does not capture the projected images when trying to track the fingers/objects on the display. Image processing on each captured frame can be employed to overcome this problem. Captured frames are converted to a gray scale images and projected images are removed by subtracting the current frame with a reference frame. As a result, the recorded frames only include white contours (blobs) which are points of contact.

The performance of a multi-touch device depends on the underlying hardware and software. When a user touches a multi-touch device, she expects to get immediate visual response. The responsiveness of the device is a factor that indicates the time needed to process the user's input and present the user's result. To improve this responsiveness, many optical multi-touch technologies use infrared light instead of image processing to subtract the contact points from the (static) background.

Infrared (IR) is a portion of the light spectrum that lies just beyond what can be perceived by the human eye. It is an electromagnetic radiation with a wavelength longer than visible light, but shorter than microwaves [9]. This special portion of light can be created in different ways. There are technologies that uses LED infrared light, such as *Frustrated Total Internal Reflection* (section 2.3.1). Other technologies such as *Rear-side Illumination* (section 2.3.2) for example produces infrared light with basic illuminators. Furthermore, optical solutions such as *Front-side Illumination* (section 2.3.2) exists which do not require an infrared light source. Instead they use ambient light of the environment.

2.2.2 Optical Sensors

In concerning to multi-touch, infrared light is mainly used to cut out the visual image in the visible light spectrum that is being captured by the camera. Most digital cameras and webcams are fitted with a filter to remove the infrared part of the spectrum so they only capture the visible light spectrum. A camera that only captures infrared light can be created by removing this infrared filter and replacing it with one that removes the visible light. On some cameras it is possible to remove the IR filter, on other (more expensive) cameras, the lens has to be replaced with a lens without coating. Most cameras will show some infrared light without modification, but much better performance can be achieved if the filter is replaced.

Cameras able to detect infrared light that illuminates the objects or fingers on the touch surface are also capable to record all other colors of the spectrum. In order to block this light, a bandpass filter (see Figure 2.3(g)) can be used. This filter only allows light from a specific wavelength to pass through.

A very crucial property of the camera in a multi-touch table is the resolution. The higher the resolution, the more pixels are available to detect fingers or objects in the camera image. This is very influenced for the precision of the touch device and is always in relation with the size of the multi-touch

display. Another critical part is the frame rate of the camera, this should be at least 30 frames per second allowing smooth interactions.

2.2.3 Visual Feedback

The use of a projector is the most popular technique to display visual feedback on the table. Rear projection is used to project the interface image on the projection screen from behind the surface. There are two main display types of projectors: LDP (Digital Light Processing) and LCD (Liquid Crystal Displays). It is important that the projector has the appropriate resolution, throw and brightness, and that the lag between input and output is appropriate for the target application. The throw is the distance between the projector and projection surface that is required to display an image of a specified size. For example in order to have a screen size of 34inch, a distance of 60cm between the projector and projection screen may be needed. It is possible to use mirrors to reduce this distance (see figure 2.2), but this reduces the quality and brightness of the image and significantly complicates the design of the device.

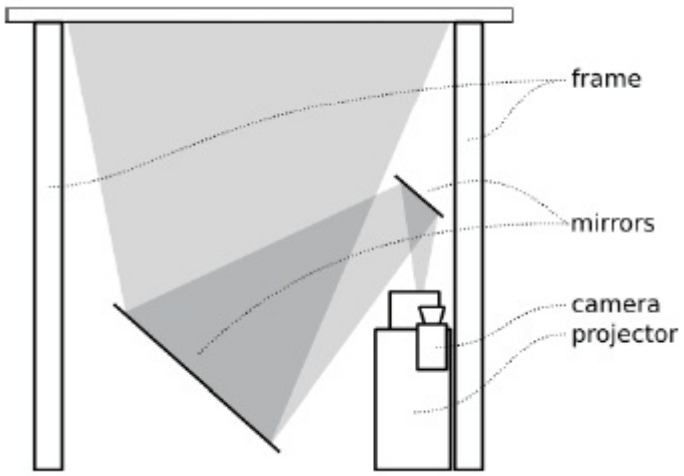

Figure 2.2: The use of mirrors to reduce the distance between the projector and the projection surface [20].

2.3 Optical Multi-touch Technologies

The previous section described the most important parts of optical based multi-touch tables. In this section the two most popular optical based solutions: *Frustrated Total Internal Reflection* and *Diffused Illumination* will be discussed and compared (section 2.3.3)

2.3.1 Frustrated Total Internal Reflection (FTIR)

FTIR is the first optical based multi-touch technology that will be discussed, it is also the technology that I am going to use for this thesis. This is a very popular optical multi-touch methodology since Jeff Han introduced it in 2005 [12]. FTIR actually refers to the well-known underlying optical phenomenon *Total Internal Reflection* [30]. This occurs when a ray of light strikes a medium boundary at an angle larger than a particular critical angle. This angle depends on the refractive indexes of both materials and can be calculated mathematically using Snell's law. If the refractive index is lower on the other side of the boundary, no light can go through and all light is reflected.

In a very basic FTIR construction, infrared LEDs (figure 2.3(a)) are mounted on four sides of a sheet of acrylic (figure 2.3(b)) [20]. When the LEDs are turned on, infrared light will become trapped in the sheet of acrylic. There is no refraction in the material so the light beam is totally reflected (figure 2.3(c)). When a user touches the surface, the light escapes (figure 2.3(d)) and is reflected at the finger's point of contact due to its higher refractive index. Now the reflection is no longer total at that point. The light exits the acrylic in a well defined area under the contact point which can be seen by the camera. A basic set of computer algorithms can be applied to the camera image to determine the location of the contact point. On the rear side of the acrylic sheet a diffuser is placed, which functions as a projection screen for the digital projector (figure 2.3(e)). Without a diffuser, the camera will not only see touch-points, but all objects behind the surface. So the

diffuser is not only a projection screen but the layer also ensures that only bright touches are visible to the camera.

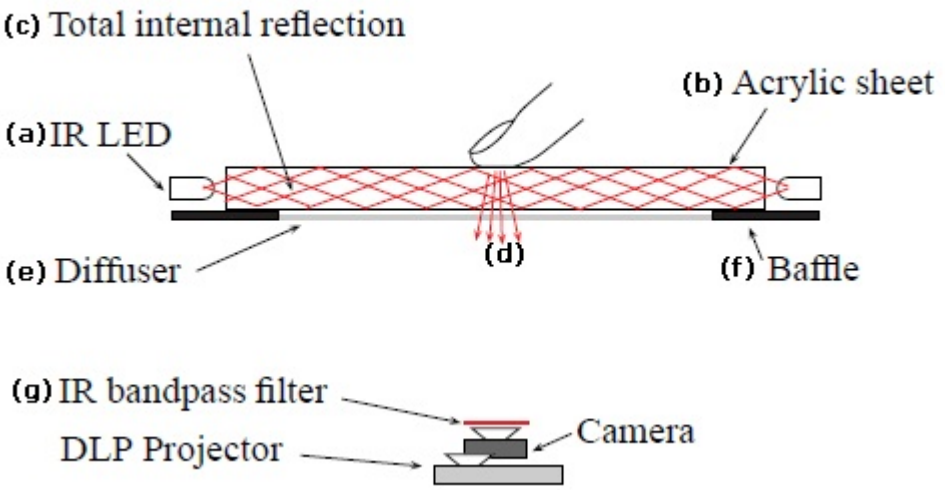

Figure 2.3: The bare minimum parts needed for a FTIR setup [32].

The previous paragraph described just the bare minimum parts needed for a FTIR setup. To improve the quality of the multi-touch table some optimalizations are needed to the elaboration. First of all, baffles (figure 2.3(f)) are required to hide the light that is leaking from the sides of the LEDs. With the basic setup, the performance mainly depends on how greasy the fingertips of the user are. Wet fingers are able to make better contact with the surface. Dry fingers and objects will not be able to frustrate the TIR. To overcome this problem a *compliant layer* (figure 2.4(a)) is needed on the top of the surface. These compliant surfaces are typically composed of a soft and transparent material. On top of the compliant layer, a rear projection material is placed. This material prevents the compliant layer of becoming damaged and also functions as a projection screen. So the diffuser on the rear side is no longer needed because the projection layer on top has all its functionalities. Additionally, this has the advantage that the fingers touch the projected images directly.

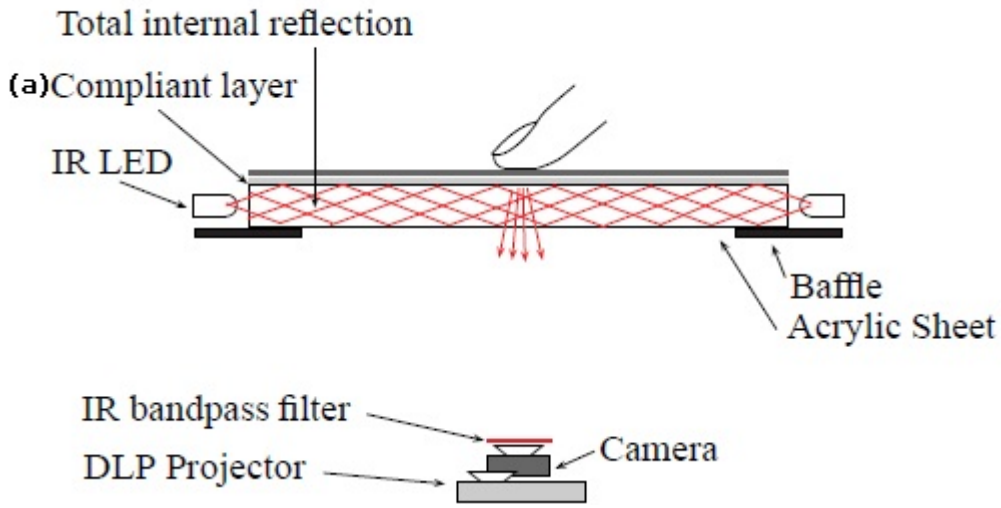

Figure 2.4: FTIR schematic diagram depicting some improvements to increase touch sensitivity [20].

2.3.2 Diffused Illumination (DI)

Diffused Illumination [20] is the second very popular optical based technology that this chapter is focusing on. DI comes in two main forms: Front Diffused Illumination and Rear Diffused Illumination. The basic difference between FTIR and Diffused illumination technologies is manifested in the fact that FTIR is based on refractive indexes whereas DI is based on the contrast between the silent image and the finger that touches the surface.

Rear-side Illumination (RI)

In this technology, there is no need to attach many LEDs on the border. This kind of optical based multi-touch surface uses one or more infrared light illuminators behind the display (see figure 2.5). A part of infrared light will be diffused on the diffuser, another part will pas through. The diffuser is located on the top or the bottom of the surface and also functions as a projection layer.

When the display is touched, the fingertips reflect less light back to the camera than the objects in the background. This allows the system to detect touches. Because the technique is based on reflection rather than changing the refractive index, it works with wet and dry fingers. In contrast to FTIR, the compliant layer is not needed anymore. furthermore, because sensing does not rely on surface contact, any transparent surface (like glass) can be used.

The quality of the diffuser is a significant factor for the performance of the surface. It is important that the diffuser does not absorb too much infrared light, if it happens, the contrast between the background and the fingertip will be very low. If the diffuser does not absorb enough infrared light, it will be illuminating objects which are near but not on the screen. This would result in false touch detection.

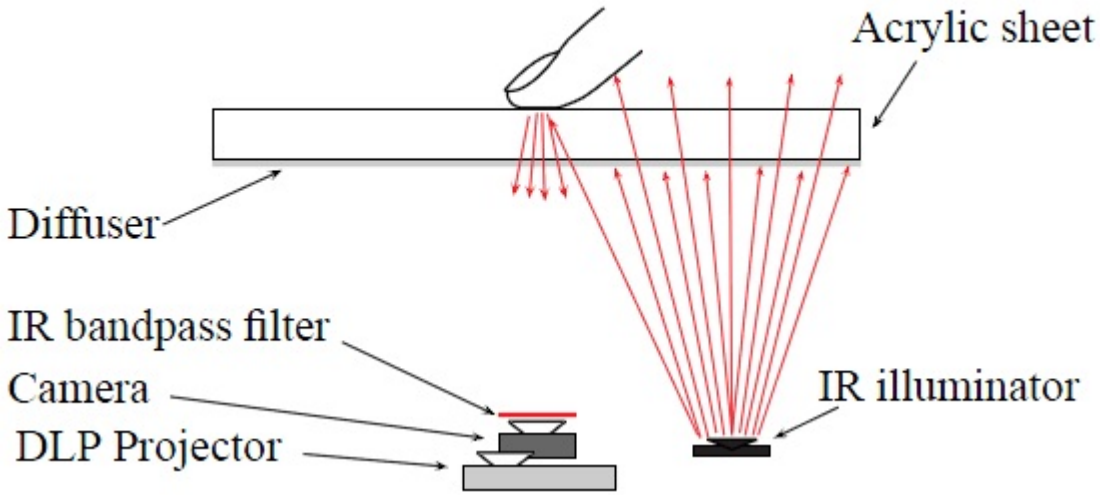

Figure 2.5: General setup of a Rear-side illumination system [19].

Besides fingertips, it is also possible to detect other objects that reflect light on the surface (for example: cellular phones, cards, etc.). These objects can be recognized based on their shape or fiducials [7] (easily recognizable markers) printed on their bottom surfaces. Depending on the resolution

of the underlying infrared camera, it is possible to allow tracking object's position and orientation. This opens up new interaction capabilities in which physical objects can be detected and recognized to trigger specific software functions.

Front-side Illumination (FI)

The second important technique used within the diffused illumination topic is called Front-side Illumination. Similar to all other optical multi-touch technologies, a light source is used. FI does not need an infrared light source, but uses the ambient light of the environment. FI shares some common properties with rear-side illumination. First, any transparent surface can be used. Second, the diffuser is very important because it takes care of ambient light to spread evenly over the surface. By touching the surface, shadows will appear underneath the fingertips which the camera can track because the ambient light can not reach it (see figure 2.6).

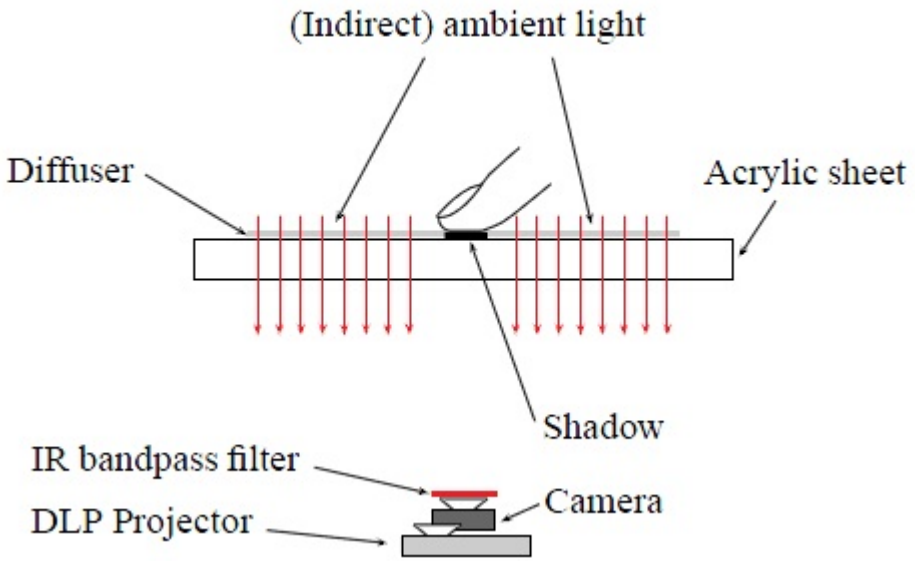

Figure 2.6: General setup of a Front-side illumination system [19].

Front-side illumination can be seen as the cheapest way of camera based

multi-touch devices. Nevertheless, the performance of the surface highly depends on the quality of evenly distributed light and therefore it is less reliable and precise than FTIR and RI. Even traditional objects or fiducial markers cannot be tracked because the technology is based on shadows instead of reflection of infrared light.

Since the Diffused Illumination technique relies on contrast differences, the environment in which the multi-touch display is placed has a large influence. Direct sunlight or light sources overhead can decrease the performance and tracking.

2.3.3 Technique Comparison

In sections 2.3.1 and 2.3.2 three essential optical based technologies have been described in detail. Figure 2.7 provides a quick summarization of the pro en cons of these multi-touch solutions [19].

	FTIR	Rear DI	Front DI
Advantages	• Blobs have strong contrast • Allows for varying blob pressure • When a compliant surface is used, input can be given with something as small as a pen tip	• No need for a compliant surface • Can use any transparent surface like glass • No LED frame required • Can track objects, fingers and fiducals	• No need for a compliant surface • Can use any transparent surface like glass • No LED frame required • Simples setup
Disadvantages	• LED frame is required (soldering) • Cannot recognize objects or fiducial markers • Requires a compliant surface for proper interaction • Special material (like acrylic) is needed to get TIR	• Blobs have lower contrast • Greater chance of false blobs • Quality of the diffuser is a significant factor	• Cannot recognize objects or fiducial markers • Greater chance of false blobs • Performance relies heavily on amount of ambient light in the environment

Figure 2.7: Advantages and disadvantages of FTIR, Rear DI and Front DI compared to each other.

2.4 Multi-touch detection and processing

With the hardware in place, there is also a pipeline of image processing operators needed that transform a camera image into user interface events. Only the FTIR tracking pipeline will be described because this is the multi-touch technology I will be using.

In the case of FTIR, the video processing chain can be very short. There are three major steps that can be distinguished:

1. The pre-processed step: images captured by the camera are processed to remove unchanging parts using previous images (history substraction).

2. Bright regions are detected in the pre-processed image. These are areas where something is touching the surface.

3. The postprocessing step, here corresponding touches in different camera frames are found and transformed in screen coordinates.

There exists libraries that take care of images captured by the camera, notable examples are Touchlib, an open source library and FTIRCap, a library that I will be using.

2.4.1 Touchlib

Touchlib stores a copy of the current frame and uses it as a reference frame to perform background substraction on the next frames. By interfering between these frames, Touchlib sends events to the user program such as *finger down*, *finger moved* and *finger released* [31].

2.4.2 FTIRCap

FTIRCap [21] takes frames captured by the infrared camera as input and creates touch-points by performing computer vision algorithms. Each touch-point consists of its screen coordinates, its size and intensity. After calibrating camera/projection, the touch-points are streamed over UDP at a

frequency of 60Hz. This results in a very fluent input interaction. Because the FTIRCap application sends its touch-points over UDP, the multi-touch program can run on another computer.

In this work, FTIRCap will used to retrieve the touch-points from the frames captured by the camera.

Chapter 3

Multi-touch User Experience

The video analysis application created in this work tries to obtain a natural multi-touch interaction experience. Multi-touch techniques can be employed to perform tasks more intuitively by using gestures, supporting multiple users and combining displays with real physical objects. However, designing such natural user interfaces brings new challenges, which are difficult to solve with current interaction paradigms. This chapter discusses research in Human Computer Interaction to create more natural interactions and how these techniques are applied in the video analysis application (see figure 3.1).

3.1 Gestures

A gesture is a form of non-verbal communication in which visible bodily actions (hand moves) communicate particular messages to the system. These gestures can replace static icons in (desktop metaphor based) programs to perform specific actions. The real world is already multi-touch and we use gestures all of the time. If done right, these systems should be as easy to interact with as picking up a pencil, drawing a picture and showing it to someone.

Gestures can be classified in two groups based on the moment of interpretation of the movement: *direct gestures* and *symbolic gestures*.

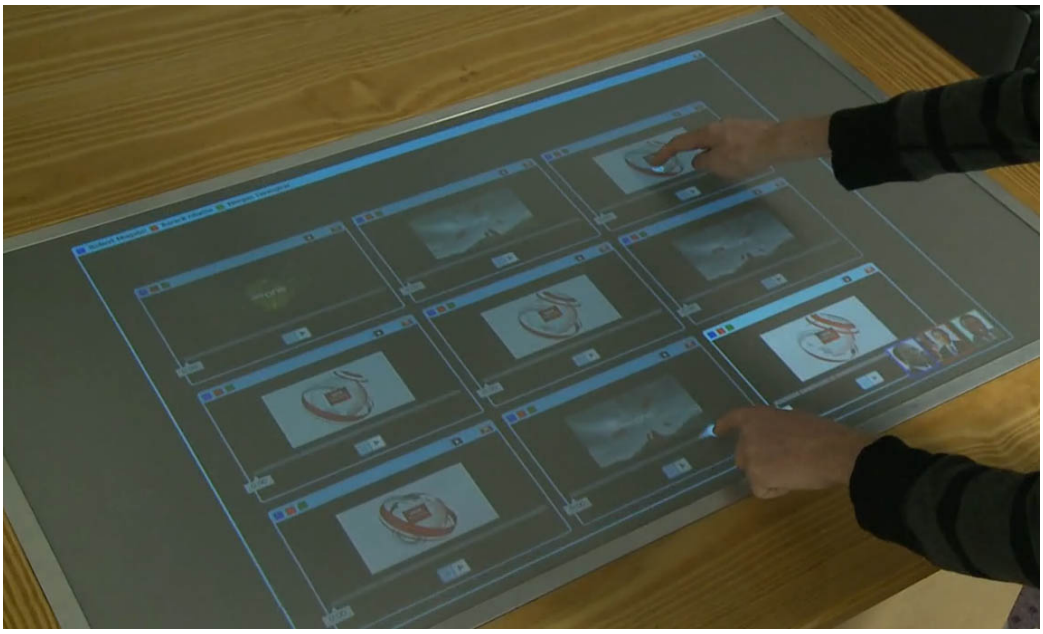

Figure 3.1: The multi-touch video analysis application.

3.1.1 Direct Gestures

A direct gesture is a gesture where each part of the movement is interpreted at a predefined interval. These gestures describe patterns that allow users to manipulate objects directly. Using gestures to perform manipulations such as rotate, scale or translate results in a very fast, easy and intuitive interaction. Therefore, this type of gestures is called *natural gestures*. Due to the fact that it is difficult to provide affordances that shows the number of touch-points needed to perform a gesture, the number of fingers needed to perform the direct gestures are not limited. A couple of these type of gestures are integrated in the application created in this work:

- To move an object, one or more fingers can be used to provide forces in the same direction.
- Enlarging or shrinking an object requires a minimum of two fingers moving to or away from each other to respectively shrink or enlarge the

object. Essentially, there is only one moving finger needed to perform a scale gesture, the other touch-points can be non-moving.

- To rotate an object, a minimum of two fingers is necessary. In essence, rotation can occur in two situations. In the former, non-moving touch-points exists. In this case the center of these touch-points is the center of the rotation. The other moving fingers causing the force of rotation around this point. In the latter situation, there are only moving touch-points which will all perform a part of the rotation around the center of these fingers.

3.1.2 Symbolic Gestures

Symbolic gestures are patterns based on the full trajectory of the movement. For this reason, these gestures can only be interpreted when the trajectory is completed and the finger is released from the surface. Patterns can be made in any shape such as triangles, circles or even text characters. In this work, two symbolic gestures are implemented: a polygon gesture and a line gesture. The later will be described in the section 3.1.3.

When working with many objects on a multi-touch display, manipulating each individual object (for example: scaling, rotating, translating, aligning, deleting, etc.) can be an annoying task. To optimise this task, the user can select multiple objects by painting a polygon around them (see figure 3.2(a)). Objects located inside or intersecting this polygon will be selected (see figure 3.2(b)). The selection can be transformed, resulting in a manipulation of all elements in the selection. Furthermore, all objects can be deleted or the selection can be organized with just one click on a button. The organization tool will align the videos in a grid and place the lenses on top (see figure 3.2(c)).

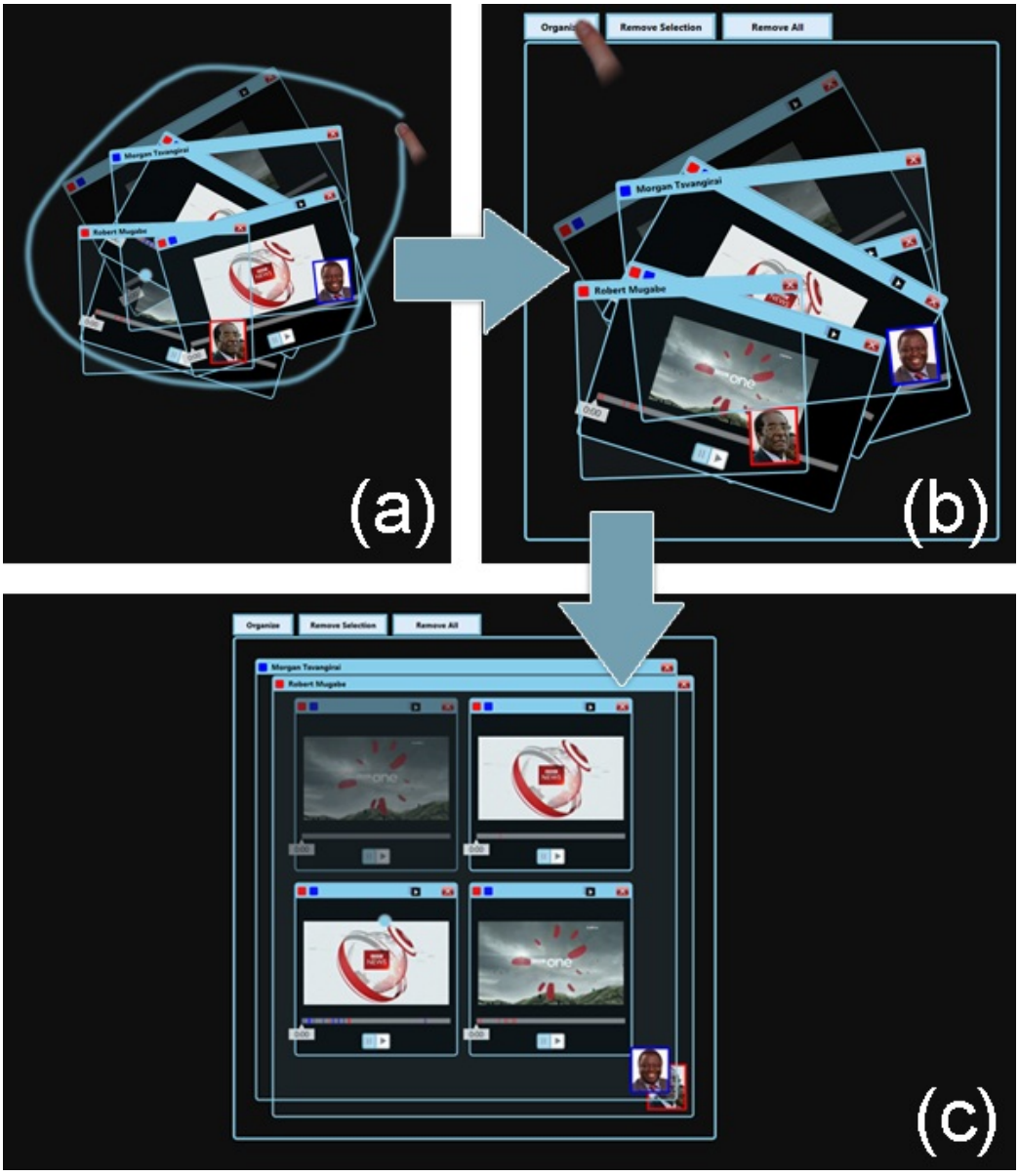

Figure 3.2: Selecting multiple objects(a), (b) and organizing all these objects(c).

3.1.3 Gestures and Multi-User Aspects

Multi-touch systems can recognize multiple gestures at the same time. The possibility of tracking so many fingers at once opens up the system for multiple concurrent users all standing around the table and interacting with the surface at the same time, this is called *multi-user*. With all of these new input capabilities come new design challenges. When two users standing over a surface, are facing each other, the system does not know how to orient objects because the orientation of the users are not known. Furthermore, it becomes difficult to find an appropriate location for menus in order to support multiple concurrent users. Gestures can help to identify the position of a user around the table. For example, when the user wants to add a rectangle, she can draw a rectangular sketch on the surface. The system interprets this gesture and creates a rectangle with the specific orientation at that position. To generalize this method and not only support drawings, all objects can be grouped on a sheet which is visible when performing a gesture.

In this thesis, the user interface is totally invisible, resulting in a maximal support of multiple users. Users can start working by performing a line gesture (symbolic gesture). This simple action (see figure 3.3(a)) displays a menu between begin and end point of the line with the specific orientation(see figure 3.3(b)). In this main menu, users can discover all available features. Notice users are supported because the right affordances are provided to make visible which actions and features that are possible through the use of the visible clue that list all potential actions. From this menu, sheets of objects (lenses or videos) can be displayed by tapping on a menu item. Dragging an object to the surface will add this object with the same orientation as the menu (see figure 3.3(c)).

When many objects are grouped on a sheet, a scrollable area can reduce the size resulting in a larger workspace. In a traditional GUI, dragging the scroll-handler downward causes the content to move upward, keeping with the customary GUI metaphor that one is moving the viewing window, not the content. Realizing the power of gestures, scrolling can be accomplished by dragging a finger through the content area (see figure 3.4). This type

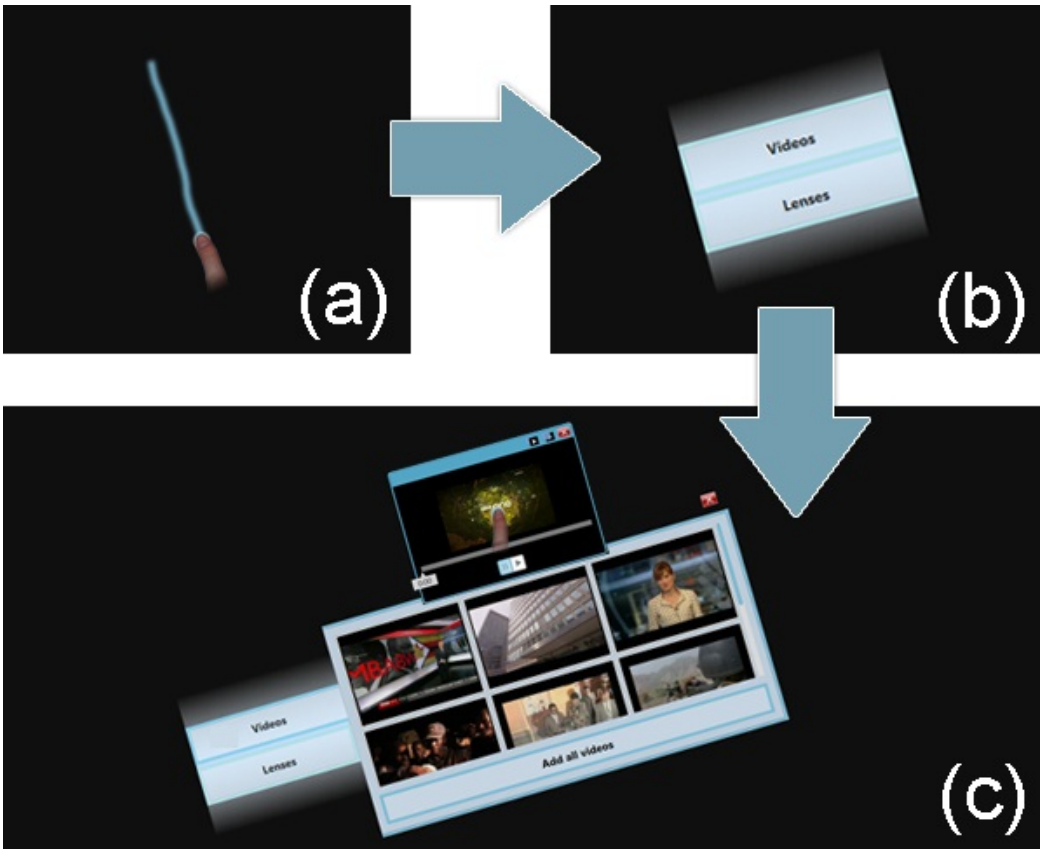

Figure 3.3: Popping up a menu(a),(b) and add a video(c).

of interaction is more intuitive because the user can move the content itself not the viewing window. Further, scrolling can be realized by touching any content, instead of dragging a scrollbar at the border of the area.

3.1.4 Problems using gestures

Many natural user interfaces are entirely gesture based. However, gestures are not easy to learn and difficult to remember. Only a few are innate or readily predisposed to rapid and easy learning. Even simple gestures such as hand-waving gestures of hello, goodbye and come here are culture specific.

One of the key challenges in a gesture-based system is offering affordances that will sufficiently guide users to make the right gesture. Showing what is possible is difficult, because the desktop metaphor based on icons is replaced with gestures. Consequently, the visibility in a gesture-based system is mostly very low as in the video analysis application. While touching an object might be intuitive, providing the right affordances to let the novice users know that they can actually use multiple fingers to shrink or grow the size is a difficult task. A good start is to make these actions as explicit as possible by removing constraints at the number of fingers and making the User Interface attractive. When the interface seems inviting, fun and playful, people get engaged. All it takes is for a user to accidentally touch an object with multiple fingers and see it change in size to understand that such things are possible. After one such unexpected, users will likely become more comfortable with just trying things out to see what is possible.

Gesture and touch-based systems are already well accepted. As a result, people try to perform traditional gestures to systems that do not understand them: waving hands in front of sinks that use old-fashioned handles, not infrared sensors, to dispense water, tapping the screen of non-touch-sensitive displays, etc. Therefore, standard conversions need to be developed so the same gestures mean the same thing in different systems [22].

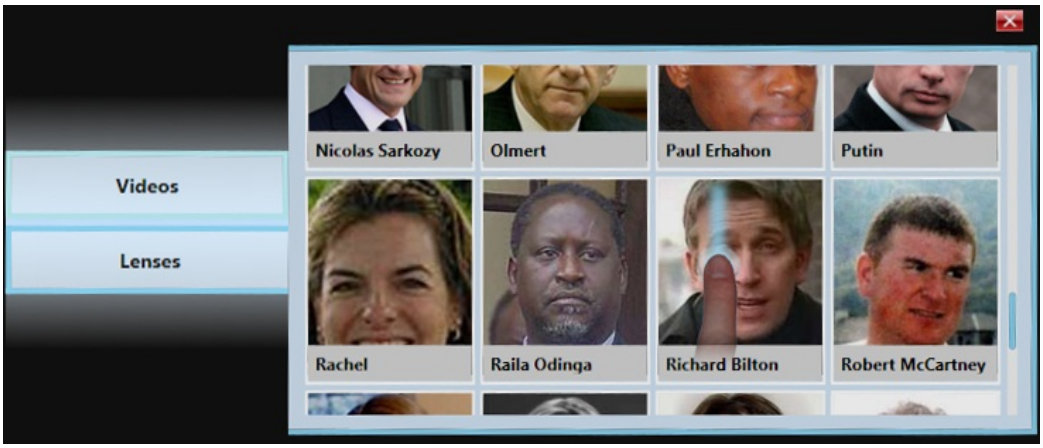

Figure 3.4: Scrolling through content by performing a natural move gesture.

3.2 Input visualization techniques

Multi-touch systems are no different from any other form of interaction. They need to follow the basic rule of interaction design: providing some sort of noticeable feedback for each action taken by the user. The idea of feedback is more than just showing the result of a specific action. Feedback from any different action adds up to provide information about the current state of the system. This becomes even more relevant with multiple users interacting with the system concurrently. Feedback related to a user's action may be relevant to the other users.

In traditional systems with a mouse as input device, there are two levels of feedback. First of all, the Operating System provides universal feedback for the input device. Moving the mouse will result in a movement of the arrow on the screen, this reassures the user that the system is still working. Furthermore, physical activation of a mouse button affirms that the input is delivered, and the position of the mouse pointer makes it apparent where the input is delivered. The secondary form of feedback is provided by the applications running on top of these systems. As a result, each application provides its own feedback independent of the feedback supported by the Operating System. In contrast, when using customary multi-touch systems,

these systems provide only the second type of feedback: feedback from the application.

3.2.1 Touch Feedback Ambiguity Problem

Most applications on touch based systems do not integrate any form of feedback to a user's input. They only give feedback when a functionality triggered by the user is successfully executed. Interacting with these kind of systems can be very confusing in a number of situations where user's input results in unexpected behaviour:

- The hardware failed to detect the touch. In this case users will wait to get response to the execution of the functionality. In addition, they will never get the response for that action and they will not be able to distinguish if the hardware failed, the action was unsuccessful or the input does not map to the expected function.
- Accidental activation: The user did not noticed an accidental touch because the lack of touch feedback. Therefore, she can get confused when this touch changes the state of the system.
- Non-responsive contact: The user wants to interact with an object not enabled for touch interaction. As a result, she wont get feedback because the touch did not result in a successfully executed action. Therefore, she cannot determine in a glance if the hardware failed or if the object is stationary.
- Selection problem: When the user missed a target and gets no feedback, she could be confused. Visualizing whether a user has successfully touched an onscreen target is essential.
- Fat-finger problem: Touching a surface reduces the contact area to a single point. This can be a problem when a physical touch on the border of an element is mapped to a single point near this object.

- Activation event: Depending on the particular hardware, the moment of activation can vary: with some multi-touch systems, activation occurs before the finger reaches the display, this might result in a different initial position of the touch contact than where the user thinks the contact occurred.

In all these situations, the user left to deduce the cause of error from no feedback, this is referred to as the *touch feedback ambiguity problem*. For this reason, the user could not correct himself and makes the same mistakes over and over again. For example, if a user is used to touch a surface with low pressure, it could take a many trail and errors before she notices that a certain degree of pressure is required when working with a resistive technology. Eventually, the user will loss the sense of control, gets frustrated and disconnect from the system [16][6].

3.2.2 Input Visualization

To increase user confidence in touch-screens, touch-feedback can be integrated in the system to eliminate the error ambiguity problem. An early attempt to visualize the input was the *TouchLight* project [33]. This system displayed raw sensor data captured by the camera back to the screen. On the one hand, this approach addresses the problem of accidental activation and non-responsive systems. On the other hand, the fat finger and non-responsive contact problem are unattempted. Furthermore, only optical based multi-touch systems can integrate this visualization principle because a camera capturing the raw sensor data is required.

In this work, touch feedback will be produced based on the raw captured data from the camera. Consequently, this provides a wide variety in the visualization of the touches to cover all situations of ambiguity. The mentioned touch visualizations are all inspired by *Microsoft Ripples* [16] integrated in the *Microsoft Surface* [17].

Visualizing Touch-points

In traditional systems with a mouse as input device, the pointer is always visible on the screen as long as the mouse is physically connected. Accordingly, the touch-points only need to be visible when a physical contact with the surface takes place. In principle, two states can be distinguished: state 0, precedes the detection of a touch-point and in the state 1, the system is touched (see figure 3.5).

State 0 cannot be visualized, as it precedes the detection. In contrast, the visualization of state 1 addresses some ambiguity problems:

- **Activation Event**: When touching a surface, the moment of activation can vary in different multi-touch technologies. The visible touch-point should clearly indicate the activation moment.
- **Accidental activation**: Unintended contacts causing individual responses. The user perceives these touch-points and eventually corrects the resulting actions to the system.

Transition B in figure 3.5 is an animation that highlights the appearance of a new pointer, causing an activation event or an accidental activation to be more striking. Furthermore, the animation when touching a movable object is slightly different from the animation when touching a stationary object such as a background image. If an object is captured by a touch, a circle

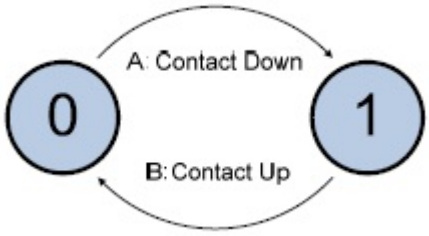

Figure 3.5: Touch-points states and transitions. 0: not yet touching 1: stationary contact [16].

shrinks around the contact, visualizing the connection between the object and the finger. If not, a circle *splashes* outward (see figure 3.6), representing the metaphor of a search for objects in infinity. These animations prevent two ambiguity problems:

- **Selection**: The animation which shrinks around the contact point will notice the user that she selected a movable object. Obviously, this animation does not indicate which object is selected. As a result, it can be very confusing when working with transparent objects and not knowing which object has the focus. This problem will be discussed in section 5.5.
- **Non-responsive content**: A user can determine in a glance if she interacts with an object which is not intend to respond to touch, because of the *splash* animation.

To address the fat finger problem, transition B is animated 3.7. This animation emphasizes the point/finger mapping. After the user lifts their finger, an animation shrinks the visible touch-point to the actual hit-point. This animation notices the user which object is eventually touched. Users working with the system frequently can improve their touches because they notice the point/finger mapping each time.

Visualizing Movements

Integrating state 1 in an application will address the basic ambiguity problems. In contrast, many other input problems can occur when interacting with a multi-touch system. Dragging a contact point to another place will causes the touch-point visualization to move with it. While dragging, the state of the touch-point remains in state 1. This can be confusing when resizing an object to its minimum or maximum size and only a small touch-point is visible underneath the finger. In this case, the user might think that the contact point is released. Preventing this problem requires a new state: state 2, in which the contact point is moving (see figure 3.8).

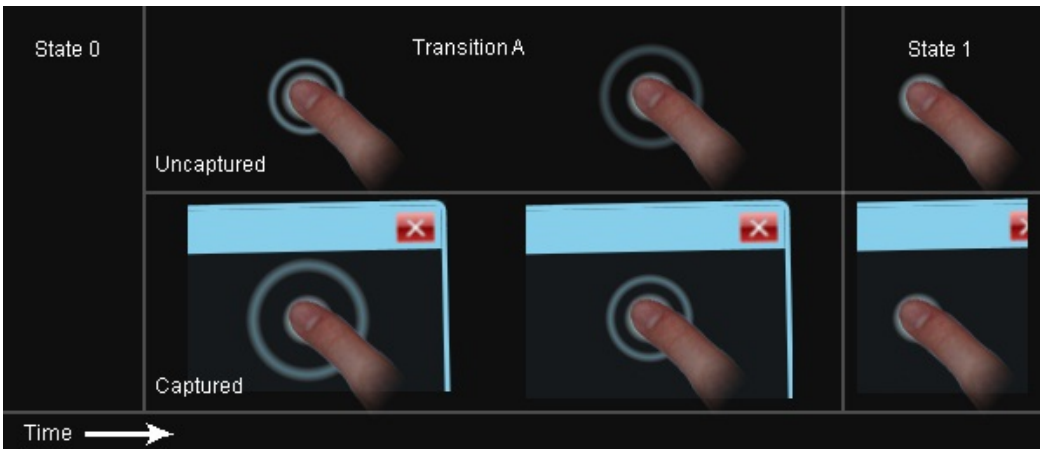

Figure 3.6: Two animations are shown for transition A. If an object is captured, a circle shrinks around the contact. If not, it *splashes* outward.

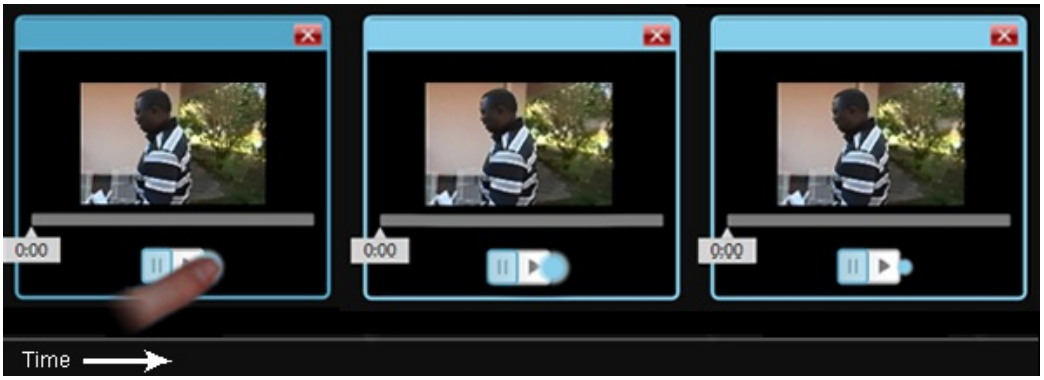

Figure 3.7: Transistion B: emphasizing the point/finger mapping to address the fat finger problem.

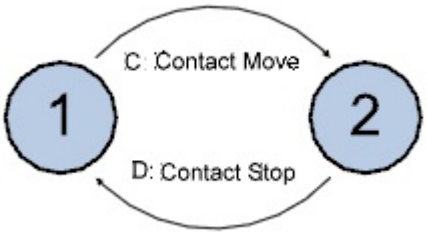

Figure 3.8: Touch-points states and transitions. 1: stationary contact 2: moving contact [16].

Movements of a touch point are visualized as a short trail behind the finger. When attempting to resize an object past its size limits, the user can still notice the tether and conclude that the movements are being captured without a release.

In addition, when performing a gesture and no record of the path is left behind, the user gets no response when that gesture does not perform an action. Furthermore, there is little information available for the user to help understand why. Replacing the short trail in state 2 by a tether to the initial touch-point results in a visualization of the gesture path at any time. This path reminds individual users of their own gesture as well as it provides feedback to all other users. A user has a better chance of knowing what somebody is thinking, if she sees what that person is doing. Especially, it is easier for a user to notice large gestures than it is for here to follow someone else mouse movements around the screen. Consequently, the task of keeping track of other collaborating partners is easier in gesture-based systems.

In summary, state 2 has two visualizations that are slightly different 3.9. First, a movement of a captured object is displayed as a short trail behind the finger. Second, movements of a touch-point at a stationary object produces a path representing the performed gesture.

Figure 3.9: Two visualizations are shown for state 2. If an object is captured, a short trail is displayed behind the finger while moving. If not, the full path will be visualized.

Chapter 4

AMASS++ Archive

In some cases, like filtering a video, there is a risk of exhausting the whole system if appropriate resources are not available or if resources are not appropriately managed. To solve this problem it is highly desirable to present appropriate summarization or abstraction of the raw video data to the users. Video summarization is the process where videos get annotated. The annotated data can be used later in combination with lenses to find the desired information. In this thesis the AMASS++ summarization archive will be used to parse the media data.

The AMASS++ project [1] [2] tries to increase the usefulness and usability of multimedia archives by combining text, audio and visual data. The archive is based on daily news from the BBC. Not only video data is captured, but also subtitles are extracted by capturing text from the subtitle teletext page. Furthermore timing information for subtitles is preserved so that realignment with the video is possible. To extract meaningful information from the video media, the videos frames are first captured at a high frequency. Later some image detection techniques are used to detect keyframes and frontal faces by fitting a 3D morphable face model to the data. Also the subtitles are processed to extract keywords related to topics. The data capture methods are all oriented towards the collection of news media concerning specific events such as the presidential election in Zimbabwe.

First of all, the raw video data is analyzed, resulting in a set of sum-

marized data elements. Afterwards when the data is needed, the reduced data set will be parsed. This results in a set of objects created in a specific programming language.

4.1 AMASS++ Summarized Data

The AMASS++ project not only touches the problem of finding materials relevant to queries, but also on the importance of presenting them in the most productive manner. Therefore the analyzed data is stored in a file with a specific syntax independent of any programming language. The analyzed video is called a *MediaStream*, this mediastream contains a collection of *Faceblocks* and *Stories*.

A Faceblock represents a particular person that is recognized in a video. Mostly, a person is detected over more than one frame, therefore each faceblock is identified with a begin and end point indicated with the sequence numbers of the two specific frames. The most relevant frame is marked as the *Keyframe* for that faceblock and contains a reference to the image taken from that frame.

Stories are a parts of a video that treats about the same content. In a news broadcast, a story generally starts with the news reader that introduces the news fragment and ends with the news reader that introduces the next topic. Because a story is a part of a video, each story includes the sequence number of the begin and end frame. Also subtitles are included in a story completed with timing information so that realignment with the video is possible (see figure 4.1).

Finally, a story is provided with a couple of keywords (wordmaps) related to specific topics for example *war*, *political*, *military*, *party*, etc. With these keywords, different stories about the same topic are related.

```
00:00:02:430 Tomorrow's election in Zimbabwe will go ahead as
Mugabe refuses to back down.
00:00:07:150 He warns fellow African leaders not to intervene
and hits out at Britain for being an evil influence.
00:00:24:870 In hiding, on the eav of polling day, the
opposition supporters braced for more brutality and violence.
00:00:33:030 We will have the latest from inside Zimbabwe.
00:00:36:430 Also: Thousands of new wind turbines to be built
across Britain.
00:00:40:710 The Government calls it "a green revolution.
00:00:43:030 The British man who murdered his wife and baby is
sent o prison for life.
00:00:48:260 His wife's mother expresses her grief.
```

Figure 4.1: An example of the subtitles in a story in the AMASS++ archive.

4.1.1 Summarization Constraints

When a video gets analyzed there is always a strong reduction in the amount of data because a video summarization program (such as AMASS++) is always oriented to track specific elements in a video. The AMASS++ archive only supports tracking for events of interests and does not support tracking of objects. So only temporal queries can be executed on the summarization archive like finding peoples in a specific period. Consequently, spacial queries are not yet supported like finding people that entered a particular room. This kind of queries would actually be very useful when querying for example videos captured by a surveillance camera. Therefore, this kind of data will be introduced in the next version of the archive.

4.2 Parsing The Summarized Data

After the video's are summarized and the associated files are created, these files need to be parsed before they can be used in a specific programming language. In this work a uniform parser will be used to parse the summarized data in the AMASS++ archive. This parser is written in C# and is not related to the application created in this thesis, consequently it can be used in any application. After parsing the summarized data, an in memory data

structure(see figure 4.2) is created representing the elements described in section 4.1.

The generated objects are associated in a way that they can be accessed quickly. For example, when a specific story is given, the related mediastream can be obtained immediately.

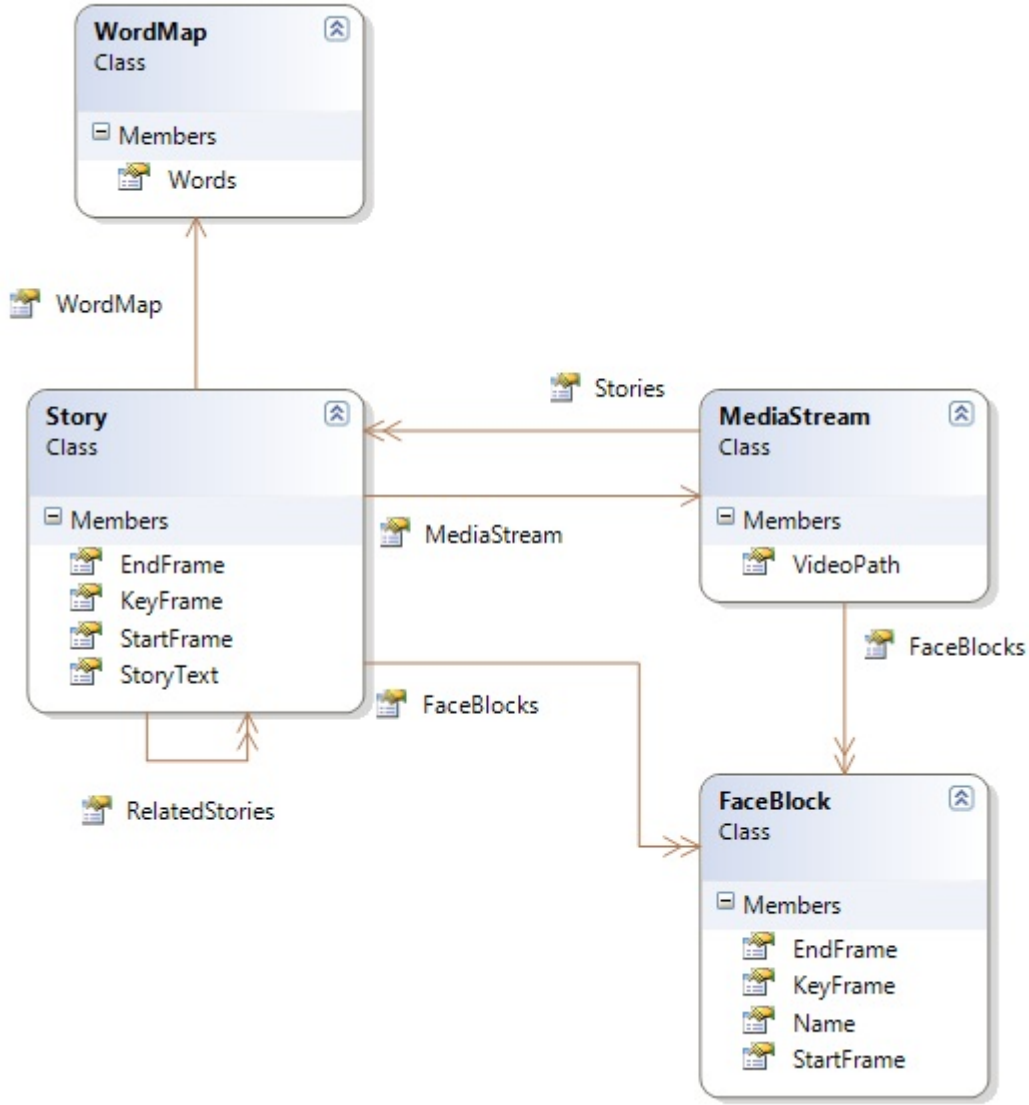

Figure 4.2: In memory data structure of the annotated AMASS++ data.

Chapter 5

Magic Lenses

Video data is available from an increasing number of sources, and yet analyzing and processing it is still a manual, tedious task. With such a large amount of video data finding, analyzing and processing a fragment of interest in these videos can be very complex and time consuming. In order to optimise this task, more advanced video browsers and visualizations are currently being employed in several research projects [34] [28]. In this thesis, direct manipulation tools called *Magic Lenses* will be introduced to explore data in a video.

5.1 The Magic Lens Concept

The *magic lens* concept was introduced by Eric A. Bier as a see-through interface in 1993 [27]. A magic lens filter is a generalization of the lens metaphor and can be used in any screen-based application. A magic lens is a screen region that transforms the content underneath it. Users usually drag a lens over the screen to decide which region is affected. A lens may act as a magnifying glass, zooming in on the content on which it is placed. It may also function as a x-ray tool to reveal hidden information.

When multiple lenses are stacked on top of each other, the individual functionalities are composed. In some cases, different lens orderings will

generate different results. Figure 5.1 is an example of a lens that functions as a x-ray tool to make the curvature of an underlying 3D model visible.

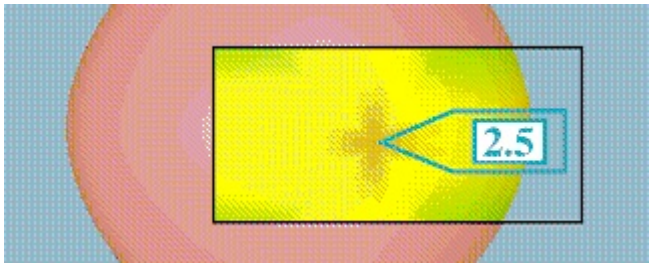

Figure 5.1: Curvature pseudo-color lens with overlaid tool to read the numeric value of the curvature [27].

Magic Lenses are a subset of the graphical user interface tools called *see-through interface* tools. Another semitransparent interactive tool in this category is a toolglass, which manipulates the object underneath permanently when they are applied. Figure 5.2 shows an example of a toolglass which changes the color of the object underneath it when that object is selected with the toolglass on top. In contrast to toolglasses, a magic lens only provides a preview of what changing the property would look like.

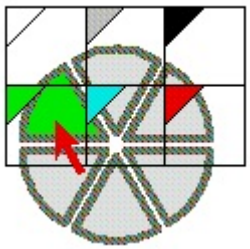

Figure 5.2: A color toolglass applies its style to an underlying part of a circle by *clicking through* the fill-color button [27].

5.2 Magic lenses and Data Filtering

Fishkin et al. [26] illustrated the value of magic lenses when finding specific information. Their work analyzed the usability of database query systems. While such systems are very powerful, they are not easy to use for browsing or exploring the data. To solve this problem, the program provided a graphical view of database values as a *starfield display* and magic lenses for exploring the data (see figure 5.3). The former, starfield displays, combine 2D scatterplots for visual information presentation with additional features to support filtering of individual points, scale, zoom, etc. in order to explore the dataset. The latter, movable filters, support queries on underlying data by encoding each operand of the query as a magic lens filter.

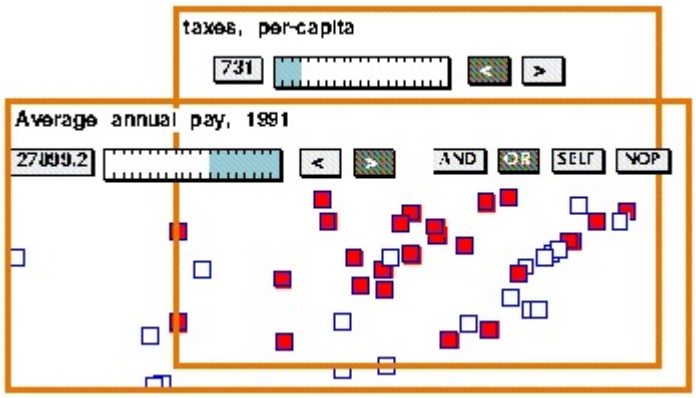

Figure 5.3: All cities with high salaries OR low taxes are highlighted as a result of two disjunctive lenses [26].

Scatterplots, as used in the starfield displays, are techniques for displaying data as points in a 2-D field. Because a scatterplot is just a collection of points, this visualization technique can encode large amounts of data. In this work a database of US census data is used, in which each row represents a city and each column describes the city along various census metrics: population, crime rate, etc. This data is especially appropriate for visualization in a scatterplot because the physical location of a city on a map is an intuitive mapping to the 2D plane.

Each lens acts as a filter that screens on some attribute of the cities (for example, extracting all cities with a crime rate lower than 100 inhabitants per km^2). The threshold value for this attribute is controlled by a slider on the filter. When the lenses overlap, their operations are combined in a *OR*, *AND* or *NOT*-relation. The application also provides grouping of filters to support complex queries.

Each lens can be represented by a function with two parameters: L(F, M). The first parameter is the filtering function. The second parameter is the composition mode that describes how the result of the filtering function is combined with the output of the lenses underneath. For example, if two filters F1 and F2 need to be applied in an *OR*-relation to the data underneath, it is necessary to create two lenses: L1=(F1, AND) and L2=(F2, OR). The result of positioning L2 over L1 is (F2 OR F1). The composition mode of the L1 lens is not important if there are no lenses underneath. Figure 5.3 shows an example of two lenses in an *OR*-relation, the cities which have high annuals OR low taxes are highlighted.

5.3 Advantages of Magic Lenses

In most applications a control panel with widgets (such as buttons) is an important area in a program. This panel is used for almost all functionalities in the application and therefore it needs to be accessed very quickly and easy. For these reasons a control panel always competes for screen space with the work area. In contrast, when using magic lenses, once the lenses are added, there is no need for a control panel. Furthermore, a lens is a manipulation tool that normally would have existed in the control panel but now is a part of the work space. As a result, the work area can take up the entire space and lenses can still be used because they are semi-transparent. They can even be scrolled partially off the screen to provide an unobstructed view of the application.

Magic lenses can be used in tiny displays, such as notebook computers or PDA's, that have low resolutions. It can also be used on wall-sized multi-

touch displays, where a fixed control panel might be physically out of reach from some screen positions. Lenses can move with the user to stay close at hand.

In Interfaces based on magic lenses, manipulation can only be accomplished on objects underneath a desired lens. These lenses perform manipulations directly on the underlying objects so the user's attention can remain focused on this area. In almost every other traditional interface, user attention cannot remain focused on the area that she manipulates. For example, when she wants to manipulate an area, the user needs to drag another tool from the control panel and apply it to the object. In other tools, manipulations can be accomplished by selecting an object in the workspace and clicking on a button in the control panel. In this case, the result of the manipulation is applied to another area of the screen.

In short, magic lenses reduce dedicated screen space while providing the ability to view context and detail simultaneously.

5.4 Magic Lenses for Visualising Video Data

Finding a suitable video fragment in a video archive is mostly a complex task. Information visualization techniques address these problems by providing graphical presentations of the data and direct manipulation tools for exploring the data. In this thesis, magic lenses are integrated into a video analysis application to support video exploration.

In the previous examples, the see-through interface tools operate in real time to manipulate the objects. In some cases, like filtering a video, there is a risk of exhausting the whole system when appropriate resources are not available or when resources are not appropriately managed. To solve this problem, a pre-processing step is introduced where the video gets annotated first. The video summarization data can be used later in combination with lenses to find the desired information. In this work, the AMASS++ summarization archive will be used to analyze the media data (see chapter 4).

5.4.1 Searching for Fragments

Most magic lenses contain a filter that represents a specific person. This lens basically represents a simple query for retrieving fragments containing this person. Moving such a lens over a video will apply the filter to the video. As a result, fragments containing the represented person on the lens are visualized. Because the AMASS++ archive is especially specialized in face recognition, the lenses are mainly intended to find fragments of specific persons. Nevertheless other lenses can be integrated in the application to support other queries (see chapter 8).

The AMASS++ archive has a great support for temporal queries (see section4.1.1), therefore the results will be represented in a timeline slider (see figure 5.4). To integrate the query results and to support multi-touch on this slider, a specialized widget is created instead of using general sliders. The time dimension is very obvious in this widget because the time in the slider thumb (see figure 5.4) represents the current position in the video. The slider thumb is extended with a sharp corner for precise fragment selection and is positioned below the bar, therefore the widget remains free for displaying the query results.

Moving a lens, representing a person (a filter), over the timeline of a video applies the filter to the overlapping part. When results are found, specific frames containing this person are highlighted in the slider. Because multiple lenses can be applied to the same spacial area, a unique color for each filter is introduced to distinguish the results of the different queries. In figure 5.5, the lens representing Morgan Tsvangirai (color: blue) entirely overlaps the timeline. As a consequence all frames in the video containing Tsvangirai are highlighted in blue. On the other hand, the lens representing Barack Obama (color: red) partially overlaps the timeline, consequently only the frames containing Obama in the overlapping area are highlighted. Whereas in figure 5.6, the Barack Obama lens entirely overlaps the timeline accordingly more results are found. Furthermore, a text balloon with the name of the person is visible above the timeline (see figures 5.5 and 5.6) to improve the feedback when the media player is playing a highlighted frame.

Figure 5.4: A Media player with a specialized timeline widget to integrate query results.

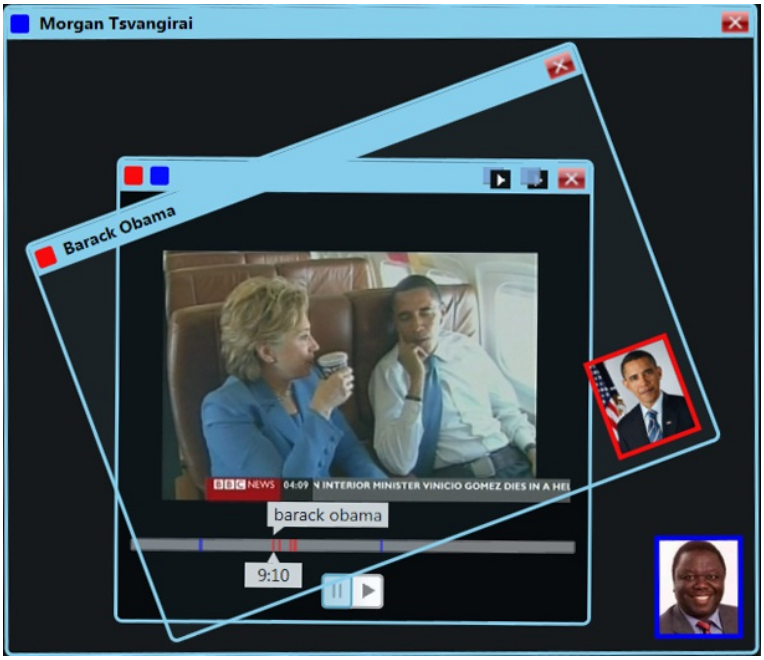

Figure 5.5: The Morgan Tsvangirai lens entirely overlaps the timeline, the Barack Obama lens partially overlaps the timeline.

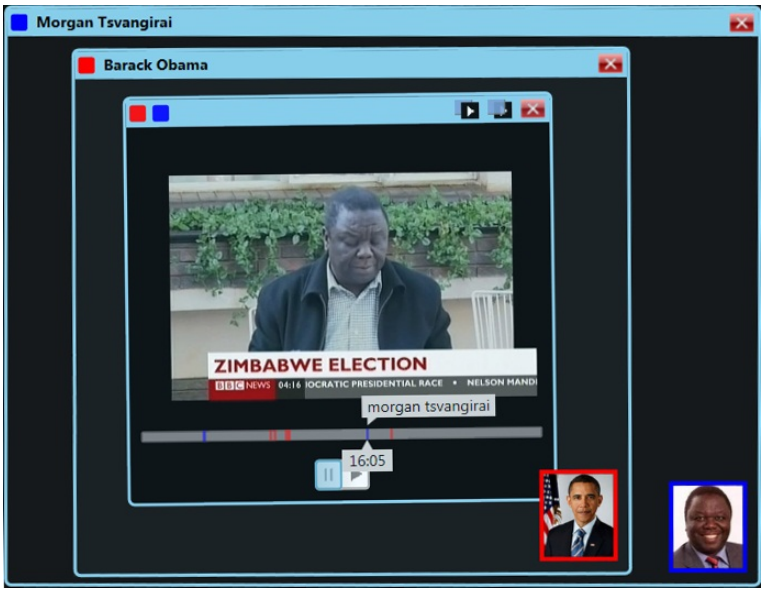

Figure 5.6: The Morgan Tsvangirai and Barack Obama lenses both overlapping the timeline.

Actually, when multiple lenses are stacked on top of a video an *OR*-relation is performed between them: all frames containing one of the persons represented by the lenses on top are visualized in a particular color. To support more complex queries, for example, finding frames containing multiple persons, requires the availability of more summarized events in the AMASS++ archive. In this situation, it would be more likely that multiple detected events exist in the same frame. Luckily the archive will be upgrated soon to support more events. Because many events are detected in an entire video, these advanced queries are currently more applicable to retrieve relevant videos (see section 5.4.2.

5.4.2 Searching for Videos

When users explore video data, it is often desired that they want to examine if one or more persons appear in a collection of videos. To fulfill this task, users need to determine if there are highlighted frames in each video. The application would have more potential if users could tell at a glance if a video

satisfies the query.

To optimise this task, videos are obscured if they do not fit one of the lenses on top of it. Figure 5.7 illustrates this principle. On the one hand, three videos are obscured because they do not contain frames including Alastair Yates *OR* Robert Mugabe. On the other hand, six videos are highlighted because frames including Alastair Yates *OR* Robert Mugabe exist. In conclusion, videos are highlighted when they have results for at least one of the lenses above.

The default *OR*-relation is a simple method to examine if a video contains a specific person. However, to determine which videos incorporate a couple of persons, an *AND*-relation is needed. Switching the relation type from *OR* to *AND* between lenses can be accomplished one after another by pressing on the lenses with one hand and moving the sliding widget to the AND-state with the other hand (see figure 5.8). The filters now appear in the same lens, indicating that they are related with an *AND*-relation. The highlighting condition of a video is still the same: a video is highlighted when it has results for at least one of the lenses above. Furthermore, filters in an *AND*-relation are compounded in one lens, therefore the video has to satisfy all the filters in the lens.

Figure 5.9 demonstrates the same situation as the example in figure 5.7, however the Alastair Yates and Robert Mugabe lens are in an *AND*-relation. Consequently only two videos contain frames including Alastair Yates *AND* Robert Mugabe, seven videos are obscured.

Switching the relation type back to an *OR*-relation, can be accomplished by pulling the pictures of the persons out of the lens one at a time (see figure 5.10).

The relation functionalities in this work are different to the relations in the application of Fishkin et al. (see section 5.2). The relations described in the latter are applicable to each individual part (city) of the starfield display. In the context of filtering a video, a single frame does not contain much information. Consequently it is not relevant to provide complex queries on

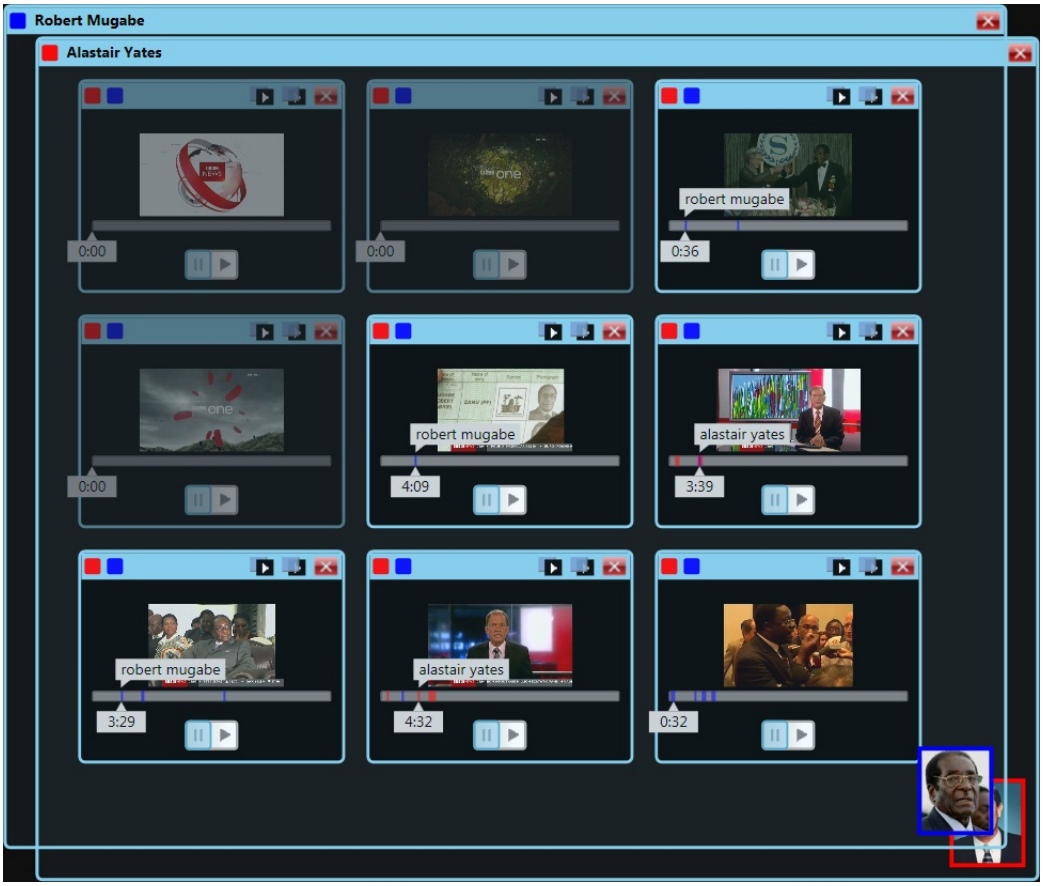

Figure 5.7: Highlighting all frames containing Robert Mugabe *OR* Alastair Yates.

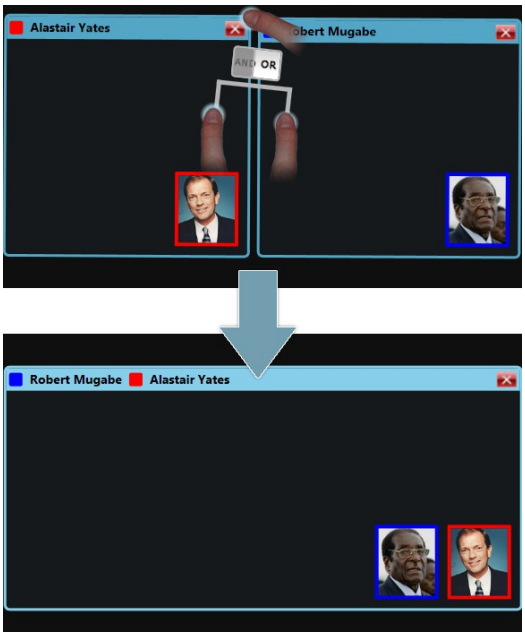

Figure 5.8: Switching the relation type between two lenses to an *AND*-relation by performing a three-finger gesture.

individual frames, therefore the relations are only incorporated to complete videos.

5.5 Interacting with Magic Lenses

Magic lenses are semi-transparent User Interface elements. In some cases this transparency makes it hard to distinguish the ordering of the lenses, particularly when using much lenses on top of each other. Manipulating an object in a stack of objects not knowing which is on top, can be very confusing. Changing a visible property (such as the border color) of an object when the user is touching it, results in an improvement of the responsiveness of the system. Furthermore, the user knows immediately which object she manipulates.

On the one hand, the ordering of the lenses has no influence on the results in this work. On the other hand, the ordering of lenses towards videos has

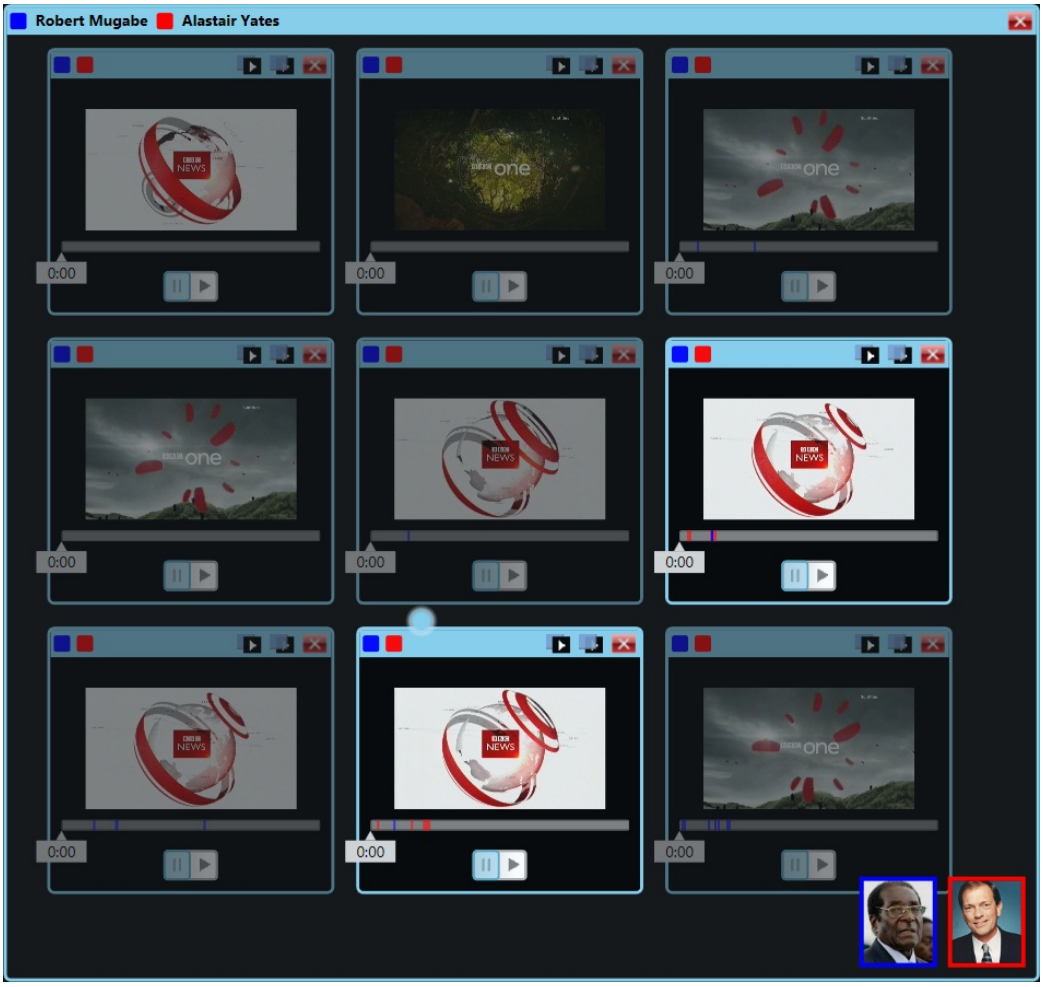

Figure 5.9: Highlighting all frames containing Robert Mugabe *AND* Alastair Yates.

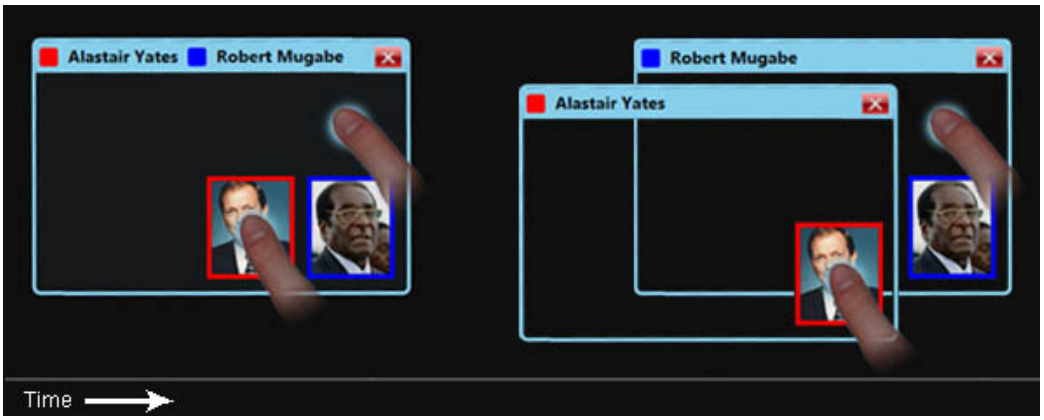

Figure 5.10: Switching the relation type between two lenses to an *OR*-relation.

a very big influence. In addition, when a user cannot determine at a glance if a lens is on top or behind a video, she can doubt about whether the video has no results or the lens does not affect the video. In this thesis, a colored square representing a lens is added to the border of the video when a new lens is hovering this video. This solution is demonstrated in figure 5.11, the video contains a red and a green square because it is affected by the lenses representing Barack Obama (red) and Morgan Tsvangirai (green). The media player does not contain a blue square because the lens representing Robert Mugabe (blue) is behind the video.

Working with magic lenses can be very intuitive because of the lens metaphor. In contrast, the users can get frustrated when they have to move all the lenses before they can work with the underlying video such as start, pause, play forward, rewind, etc. To provide these basic media player options, user interface elements (such as buttons, sliding widgets and the media timeline) are created to support manipulation with semi-transparent elements on top.

With all these media player functionalities, manipulating a video by gestures with a stack of lenses on top, can still be improved. To optimise this process two buttons (see figure 5.11) are provided in the border of each video to push it one position forward or one position backward in the stack. Con-

sequently no other objects need to be replaced when a video needs to be manipulated with lenses on top.

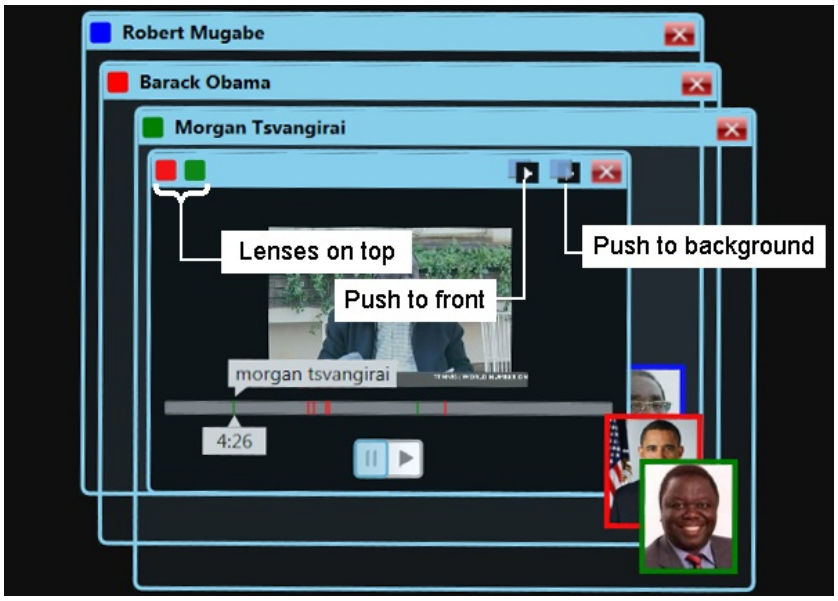

Figure 5.11: The *Push to front* button and *Push to background* button are positioned at the right side of the video. Two squares representing the lenses on top of the video are located on the left side of the video.

Chapter 6

Implementation

Implementing the video analysis application involved research in many technologies as well as fundamental decisions in the architecture of the software. In this chapter, used technologies and considerations in the design of the application will be described.

6.1 Technologies

Many technologies are suitable for multi-touch applications such as .NET/C#, Flash and C++/openGL. In this work, the .NET/C# programming interface will be used.

6.1.1 .NET/C#

The Microsoft .NET Framework [14] is a software framework available on several Microsoft Windows operating systems. On the one hand, it includes a large library of coded solutions to prevent common programming problems. On the other hand, a virtual machine manages the execution of programs using this library. .NET is intended to be the main framework for creating applications for the Windows platform [15].

The framework can be used in a variety of languages including C#, Visual

Basic, J# and C++. In this work, C# will be used. When programming applications using the .NET Framework, these applications are guaranteed to run on other machines containing this framework. The programs are transferable because the compiler generate a common intermediate language instead of binary code.

The virtual machine that manages the programs created in .NET insures that the code runs *managed*. This means that it cannot crash a system, nor can it make it less stable.

6.1.2 WPF/XAML

The .NET Framework supports many graphical interfaces for example Windows Forms (WF), Windows Presentation Foundation(WPF), Windows Communication Foundation (WCF), etc. In this thesis, WPF will be used to integrate graphical elements in the application.

Windows Presentation Foundation

Windows Presentation Foundation (WPF)[18] is a subsystem of the .NET Framework for rendering user interfaces in Windows-based applications. This graphical system is designed to remove all dependencies with the prehistoric GDI (Graphics Device Interface) subsystem. Instead, WPF is built on DirectX. Therefore, all content in a WPF application is converted to 3D triangles, textures or other Direct3D objects and then rendered by hardware. This means that WPF applications get the benefits of hardware acceleration for smoother graphics and all-around better performance.

WPF covers all areas of video, speech, rich document viewing, 2D and 3D graphics with a consistent programming model. Furthermore, many techniques and effects in one area apply to all the other areas. As a result, the controls are also extremely composable. For example, an animation can be integrated in a combobox or buttons and menu's can be filled with video clips. In addition, WPF makes it quite easy to *skin* applications with radically different looks [35].

In summary, this graphical system is appropriate for describing the higher level of visual effects that is expected in today's applications.

Extensible Application Markup Language

XAML [36] is a relatively simple declarative programming language suitable for constructing and initializing .NET objects in .NET 3.0. It is a XML-based language to define a user interface with an incredible range of expressiveness. Although XAML is originally designed to be a XML representation of WPF, it is a general-purpose language and therefore it can be applied to other technologies as well. Furthermore, using XAML with WPF is optional, everything done with XAML can be done entirely in other .NET languages. On the other hand, it is rare to see WPF used in real world without XAML for a couple of reasons [37]:

- XAML is the most concise way to represent a user interface.
- The use of XAML encourages a separation of front-end appearance and back-end logic.
- The design of a user interface created in XAML can be viewed without compilation.

6.2 Data Structure

In the application, two major parts can be distinguished that required some Software Engineerings principles: the pipeline from the abstract data to the visualization and the pipeline from the capturing of the input points to the handling of events. The former process is implemented using the Model-View-ViewModel pattern. The latter is implemented in a process with multiple layers.

6.2.1 Model-View-ViewModel pattern

A well-designed application is an application that is easy to develop, test, maintain, and evolve. Creating such an application typically involves some form of separation and encapsulation of responsibilities. A common approach involves separation of the User Interface (the views) from the business logic (the model). In fact, that approach is so common that there are a number of stock solutions, collectively known as the *model-view-x* paradigm.

The oldest concept is the Model-View-Controller (MVC), dating back to 1979. In this case, the Controller handles all the interaction between the User Interface and the business model. Besides this, the controller contains no essential data.

The Model-View-ViewModel (MVVM) is a derivative of the MVC that takes advantages of particular strengths of the WPF architecture to separate the Model and the view by introducing an abstract layer between them: a *Model of the View*, or *ViewModel*. This third layer (see figure 6.1) provides an abstraction of data that a view needs to visualize the information. Furthermore, the ViewModel can store visual states and policies that are shared by a group of views.

In general, the views know of the ViewModel and bind to its data, to be able to reflect any changes in it. The ViewModel has no reference to the

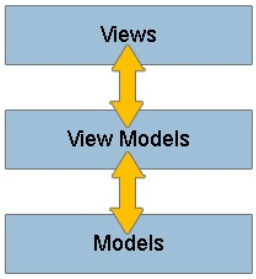

Figure 6.1: The Model-View-ViewModel pattern.

views, it holds only a reference to the Model. This makes it easy to replace views while using the same ViewModel. For the Views, the ViewModel acts both as a facade to the model and insures that visual states are shared between views binded to the same data in the ModelView.

Data Binding

WPF introduced the concept of binding User Interface elements to data from a variety of data sources in the form of common language runtime objects or even XML [4].

The classic scenario is providing a visual representation of a list of items in an object located in the ViewModel. Instead of iterating through the list and manually adding each item to a collectionview (for example an *listboxItem* to a *listbox*), WPF introduced the *ObservableCollection*. Basically, this is just a list of objects located in the ViewModel. A widget in the View, for example a *listbox* can be binded to the collection in the ViewModel. The binding mechanism makes sure that the data is always up to date in the View. The type of the items in a *ObservableCollection* can be everything. Therefore, a *Value Converter* can be used to format the items in a way they can be represented.

Benefits of using MVVM

Developing a WPF application based on the MVVM design pattern provides some structural benefits:

- The pattern separates the User Interface from the business logic, making it easy to modify one without affecting the other. Furthermore, it improves the reusability of the model and the replaceability of the view.
- The ViewModel makes it easy to share visual states between multiple views.

- The Model-View-ViewModel design is a very loosely coupled design, making it easy to maintain especially when different teams are developing the application. The View holds a reference to the ViewModel, and the ViewModel holds a reference to the Model. The rest is done by the data-binding infrastructure of WPF.

MVVM-example: Videos

Because of its benefits, the MVVM-pattern is the global pattern in the application. Figure 6.2 shows the basic data structure of a video. The entire view is created in XAML and binded to the ViewModel representation of a video. In addition, the view could be divided in multiple XAML-files each binded to its own ViewModel. The ViewModel consists of a couple of objects representing the visible state of a video, such as transformations, minimum and maximum sizes, a slider, etc. The real video data and data parsed by the *AMASS++ data parser* (see section 4.2) are included in the Model.

Figure 6.2 shows only a minimal representation of the real data structure in the application. Normally, the transformation, z-index, size ranges, etc. are included in a base class of the *VideoViewModel* class. This is because the information is needed in all movable objects (see figure 6.3). Section 6.2.2 gives an introduction to this layered structure.

6.2.2 Handling Input

The used multi-touch table sends UDP-packets containing information about the touch-points to the network controller, as described in section 2.4. As a result, it is not possible to use traditional event handlers for handling input on objects. Hence, there is a need for algorithms to perform three essential steps:

1. Recognizing events
2. Mapping of recognized events to events on objects

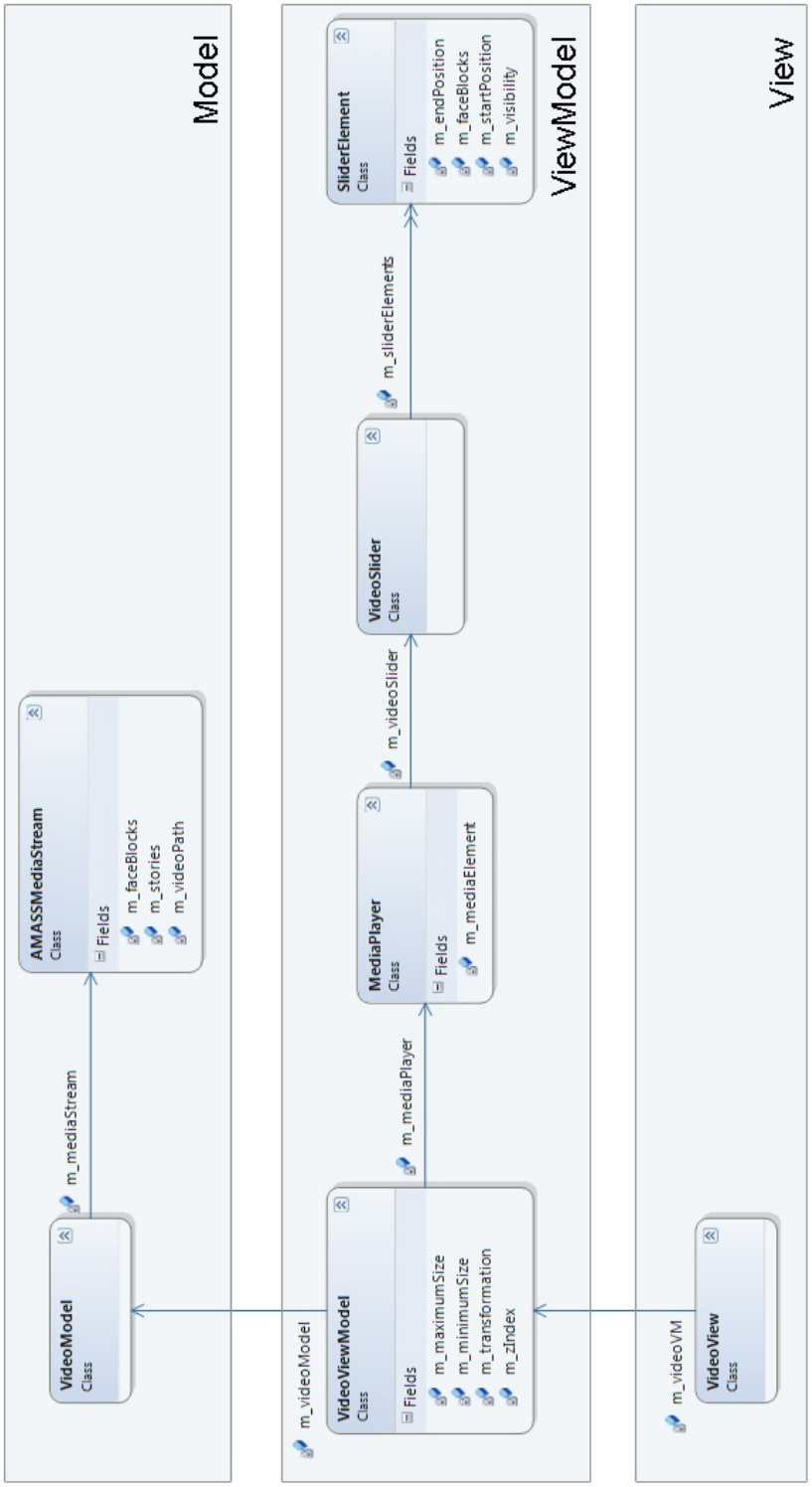

Figure 6.2: Basic data structure of a video. (Figure 4.2 shows the full class diagram of the AMASS++ *MediaStream*).

3. Recognizing gestures on specific objects

Each step is implemented as a separate library (see figure 6.3) and uses the underlying library to add extra features on top. Basically, each library in this stack helps to control the complexity. Furthermore, the components are easy to maintain and reuse. In this section, the principles of each step will be described. The first library is created at the University Hasselt. The others are created by myself during this thesis.

Recognizing events

In section 2.4 is described how images captured by a camera are transformed in screen coordinates and transported over UDP using FTIRCap. This layer, integrated in the *MTInput* library (see figure 6.3), analyses the packets and converts the touch-points into control-points (*Pointer* objects). Control-points are touch-points extended with a unique identifier, persistent over one continuous stroke on the screen. Basically, this layer compares the set of touch-points to the set of control-points, update their properties and emit events such as *PointerAdded*, *PointerRemoved*, , *ReleaseEvent*, *MoveEvent*.

Mapping of recognized events to events on objects

The previous step recognizes touches independent of objects in the application. Obviously, touches at specific area's such as a button, a scroll area, a slider, etc. need to be recognized as events on these objects before any significant application can be made. For this reason, the *MTInputRipples* library is created (see figure 6.3).

On the one hand, to recognize a touch as a touch on an object, requires that the library has knowledge of all the touchable objects in the application. On the other hand, this would break the intension of a library. Therefore each tangible object in the application contains an instance of the *InputEntity* class. This object consists of basic events such as a press-event, release-event

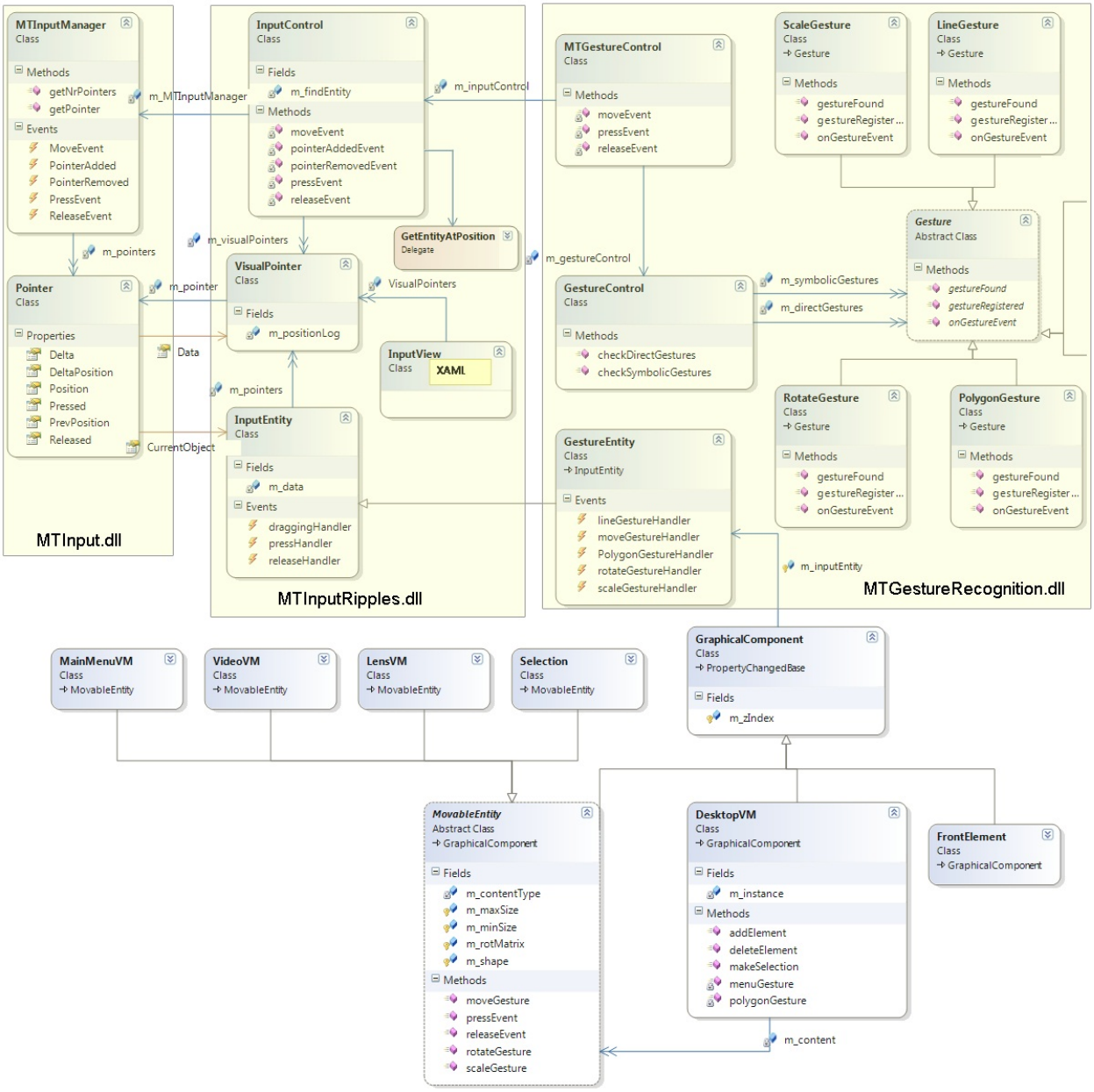

Figure 6.3: The basic input architecture of the application.

and drag-event. Handlers for each event can be created in the application objects.

Before events on a specific *InputEntity* instance can be triggered, the library needs to know which *InputEntity* has the focus. A *Delegate* function (`InputEntity GetEntityAtPosition(Point pos)`) can be used to retrieve the *InputEntity* instance of the object at a specific position (the touch-point). This delegate function needs to be implemented in the application tier.

To return an *InputEntity* instance of an object at a specific position, the location of each object in the application can be checked. This algorithm is very inefficient when many objects exists. Furthermore, it is difficult to check if an object, not regular shaped, contains a point. In contrast, a quad-tree can be used to optimise the search-algorithm. Fortunately, WPF has implemented such a process to find visual objects, called *Hit Testing*.

The Hit test algorithm searches for elements in the *WPF Visual Tree* at a specific position and returns the element at the top (highest z-index). This behavior can be changed by implementing the *hitTestCallback* function. This function is called for every object at the requested position. As a result, a priority mechanism can be created to return the element that deserves the focus. This mechanism can be very useful when supporting basic media player options with semi-transparent elements on top (a problem stated in section 5.5). To generalize this method, all elements that are manipulable behind semi-transparent elements are derived from the *FrontElement* class (see figure 6.3).

Apart from triggering events on objects the *MTInputRipples* library also deals with input visualizations (introduced in section 3.2.2) using XAML. Because the input visualization runs on top of the application, it can be reused in other applications. This provides a consistent visualization over multiple programs. Once the library is integrated in a system, the programmer does not have to worry about input feedback, it is all done by the library. Figure 6.4 summarizes all the steps from recognizing an event to the creation of events on specific objects.

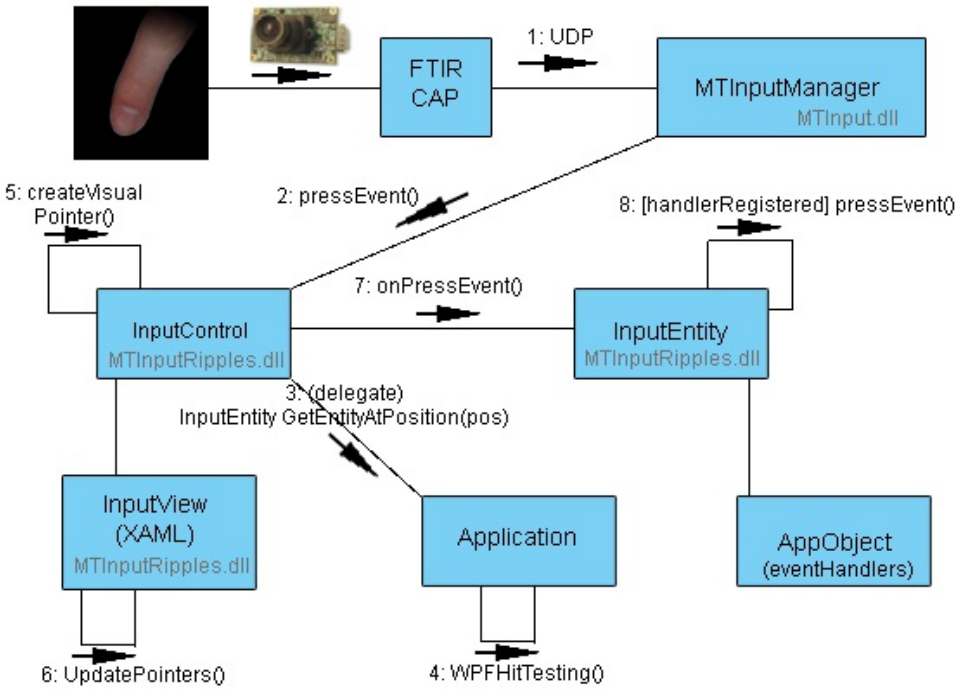

Figure 6.4: Mapping of recognized events to events on objects.

Recognizing gestures on specific objects

When gestures are supported on an object, an instance of the *GestureEntity* class located in the *MTGestureRecognition* library 6.3 can be used. This class is derived from the *InputEntity* class and adds special events for gestures. Handlers can be registered on gesture events that are provided. A gesture event is triggered when this gesture is performed on a particular object.

6.3 Implementing Lenses

Lenses can be positioned over many videos, they can also be dragged over videos to move the lens to the other side of the surface. In both situations system delays can be very frustrated. Therefore, it is necessary that the calculation of results is very fast. On the one hand, the AMASS++ data parser (described in section 4.2) solves the problem partially by loading all the video data when the application is started. On the other hand, efficient collision detection algorithms are needed to detect an overlap between a video and a lens.

Moving or adding a lens performs a process that exists of four steps:

1. Detection of videos overlaid by the lens: When a video is totally overlaid by a lens, a collision detection algorithm is not suitable. A Random point (for example the left top) of the video can be used to detect if this point is located inside the lens. In this case, the video intersects the lens or the video is overlaid by the lens. In both circumstances, the filter function is executed and thereby the results in the video are calculated. Each result (a person detected in the video over a period) is represented by an instance of the *SliderElement* class, shown in figure 6.2. Furthermore, step 2 and 3 can be skipped.

2. Bounding box collision: This very fast approximate collision detection algorithm is executed to retrieve the videos that are probably intersecting the lens.

3. Precise collision detection: When an intersection between the bounding boxes of a video and the lens exist, a more precise collision detection algorithm is performed to determine which videos are intersecting the lens. First, the borders of the video and the lens are converted to lines. Second, the intersections between these 8 lines are calculated. When one intersection is found, the collision algorithm terminates and results in the video will be calculated.

4. Collision with the video timeline: When a video is overlaid by a lens and the results are calculated, each part of the result only needs to be visible when the lens is on top of that result in the timeline (see section 5.4.1). Basically, the intersections between the time-line and the lens are needed. These intersections can be retrieved using the line collision algorithm in step 3 to calculate the intersections between the line described by the timeline and the four borderlines of the lens.

Chapter 7

Usability Tests

This chapter describes the test process used to evaluate the application created in this work, as well as the results. Chapter 8 describes suggestions for future work based on findings and results from this test.

7.1 The purpose

The usability test is mainly intended to determine the extent of the lenses to facilitate a user's ability to explore videos. Hence, the test is mainly focused on the use of magic lenses, described in chapter 5. Before the test, a couple assumptions had been created that can be substantiate afterwards:

- Users are aware of the goal of lenses.
- The results are clearly visualized in the video's.
- Relations between lenses can be created easily.
- The visualization of relations between lenses is intuitive.
- Creating relations between lenses simplifies the process of finding relevant videos.

7.2 Methodology

This section describes the course of the test session, the profile of the participants and their role in this usability test.

7.2.1 The Procedure

Five participants took part in the usability test at the Expertise Centre for Digital Media. In 30 minutes, each subject evaluated the system in a couple of steps. First of all, an instruction video was provided to show the basic functionalities of the system. Next, the participant performed two tasks on the surface while thinking aloud. During the interaction with the table, the subjects were observed and their behavior and comments were noted down. After the test, the user completed a questionnaire.

7.2.2 Participants

Three male and two female participants took part in the test. The age range is displayed in table 7.1.

Age	**Freq.**
20-25	1
25-30	3
30-35	1
Total (participants)	**5**

Table 7.1: Age range of the participants.

7.2.3 Introduction

Before the actual evaluation, participants where asked to watch an introduction video. This video showed basic features of the system for:

- Adding lenses
- Manipulation of lenses
- Basic operations of a video
- Viewing the results in a video
- Finding relevant videos as a result of a query
- Creating an *AND*-relation
- Removing the *AND*-relation

The user could learn these actions by performing the same operation after each step on a multi-touch system running the video analysis application.

7.2.4 Tasks

After watching the instruction video, the participant performed two tasks requiring similar actions as demonstrated in the instruction video:

1. How many videos contain fragments of Gordon Brown *AND* Nicolas Sarkozy *AND* Morgan Tsvangirai? Play a fragment that contains Nicolas Sarkozy.
2. Break the *AND*-relation between the lenses in task 1 and examine how many videos contain fragments of Gordon Brown *OR* Nicolas Sarkozy.

Finally, the test user completed a questionnaire about the experiences using the system. The full instruction document including this list of questions can be found in appendix A.

7.3 Findings and Results

A usability test delivers not only results from the questionnaire, much information can be revealed by observing user's behavior during the interaction with the application. Furthermore, the success rate and comments deliver great information about the satisfaction while using the program.

7.3.1 Results of the questionnaire

After task session completion, participants rated the application for eleven measures. A detailed report of each individual question is provided in table 7.2. The results of the questionnaire are also visualized in a barchart in figure 7.1. A value of one indicates that the users did not agreed with the statement. In contrast, a value of five indicates the users entirely agreed with the statement.

7.3.2 Findings

Based on our observation during the test, we had the opportunity to evaluate the application very critically.

- Every participant used the basic functionalities of the media player with lenses on top.
- Many participants said: "It is easy to create complex queries with these lenses."
- Some participants requested a faster way to move lenses on top of a video such as snapping the lenses around the videos. On the one hand, a mechanism to place lenses on top of a video can speed up the search process. On the other hand, a snapping tool can be very frustrating when there are many videos on the surface, when multiple persons work on the same touch-screen or when the user just wants to filter a couple of videos. For these reasons a better tool called *the organization*

	Median rating	**Mean rating**
1. The objective of the lenses (visualizing relevant fragments) is clear.	5	4,8
2. Lenses are very useful for finding fragments in a video.	4	4,2
3. Moving a lens over a video to view the results is very intuitive.	4	4,2
4. Using colors result in a very clear visualization of the relevant fragments.	4	4
5. It is very intuitive to use the basic features of a media player (play, pause, rewind, forward, etc.) when lenses are on top.	4	4,2
6. The colored squares in the border of a video provides a clear visualization of the lenses op top.	3	3
7. Lenses in an *AND*-relation are very useful to find videos containing multiple persons.	4	4
8. The gesture to create an *AND*-relation is easy to learn.	2	2,6
9. The gesture to break the *AND*-relation is easy to learn.	4	4
10. Displaying two persons in one lens provide a clear visualization for an *AND*-relation.	5	4,6
11. It is a great visualization to obscure videos that do not contain results.	4	4

Table 7.2: Detailed report of the questionnaire.

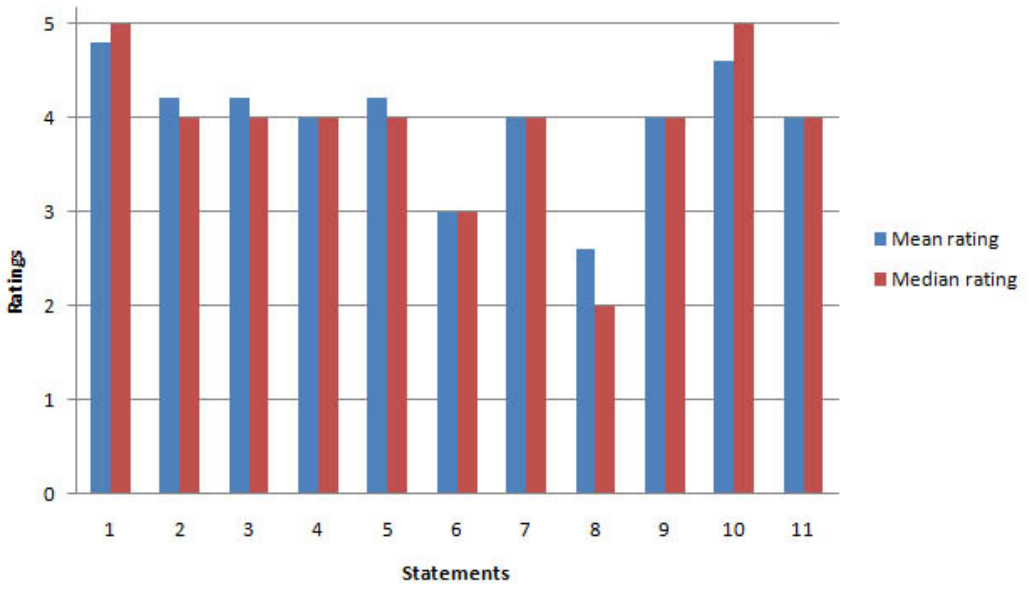

Figure 7.1: A barchart of the results of the questionnaire (The statements can be found in table 7.2).

tool is already integrated in the application. This tool is described in section 3.1.2 and provides a fast way to align an arbitrary number of objects in the application. Unfortunately this functionality was not tested because the focus was entirely on magic lenses.

- When lenses overlap each other, it is not clear which lens is on top.
- Due to the fact that it is not clear which lens is on top, performing a three-finger gesture to create an *AND*-relation is very difficult. More specifically, it is very difficult to perform this gesture when one lens entirely overlaps the other. In this situation, the user can only reach the overlapping lens and therefore she performs the three-finger gesture on just one lens. Furthermore, when the lenses are separated, some participants tried to make this gesture with one hand. Others used both hands to click on the two lenses, as a result they had no hand left to move the slider. In both situations the user could not complete the gesture.

- Some participants thought that the colored squares in the border of a video represented the persons for which the video contains fragments. Once they noticed that these squares represented the lenses on top, they found this representation very useful to determine which lens is on top or behind the video.

7.3.3 Overall Results

Based on the result of the questionnaire and the findings during the observation of the test users, a list of essential pros and cons can be provided:

Pros:

- Using the basic functionalities of videos when lenses are on top is intuitive.
- The objective of lenses (finding fragments and moving them above the videos) is clear.
- Highlighting frames in a different color for each person is very useful.
- Visualizing an *AND*-relation in a compound lens is logical.
- The gesture to break an *AND*-relation is easy to learn.

Cons:

- The depth of the lenses are difficult to notice in many cases because the lenses are transparent. This causes frustration among the users when they manipulate an unintended lens.
- It is difficult to perform a three-finger gesture such as creating an *AND*-relation.
- The colored squares do not clearly represent the lenses on top of a video.

Chapter 8

Future Work

Results of the usability test are very crucial for improvements to the application. Therefore this chapter provides recommended changes and justifications driven by the participant success rate, behaviors, and comments discussed in section 7.3. Furthermore, some general suggestions for future work related to magic lenses will be provided.

8.1 More Types of Lenses

The current version of the application supports only lenses representing a person. Nevertheless, other lenses could be created, for example:

- A lens that visualizes all stories in a video (parts of a video that treats about the same content).
- A lens for visualizing the subtitles of a video.
- Lenses that visualize the existence of specific words in a video.
- Lenses for visualizing the occurrence of specific events in a video.

8.2 Dept of the lenses

Many participants in the usability test had a problem with the lack of visualizations indicating the dept of the lenses (see section 7.3.2). To improve this feedback the frame-border of a lens can be filled with the same color as the color linked to the person it represents. Possibly, the overlap of lenses can be made even more visible by enlarging the colored border of each lens (see figure 8.1).

Section 5.5 described a solution for the dept problem of the lenses in which the border color of the lens was changed when the user is touching it. During the usability test nobody noticed this change in color because a lens can be very large (to cover many videos). In this case, the users do not focus on the border but rather on the videos inside the lens. The responsiveness of the system can be improved by providing a color overlay when the user is interacting with a lens (see figure 8.2).

8.3 Visualizing lenses on top of a video

The improvement in the spacial character of the lenses described in section 8.2 can help to detect the ordering of lenses towards a video if this lens intersects the video. In contrast, when a lens entirely overlaps a video, the colored squares in the border of this video are intended to visualize the lenses on top (see section 5.5). This feedback is very essential because the user can determine at a glance if a lens is on top or behind a video. As a result there is no doubt about whether the video has no results or the lens does not affect the video.

Results of the usability test showed that it is not clear for many users if the colored squares represent the lenses on top or the lenses for which a video contains results. Therefore, a visualization in these squares can be provided to indicate if the video contains a result for the lens (see figure 8.3). Consequently, the user will experience that all lenses on top of a video are visualized as a colored square in the video.

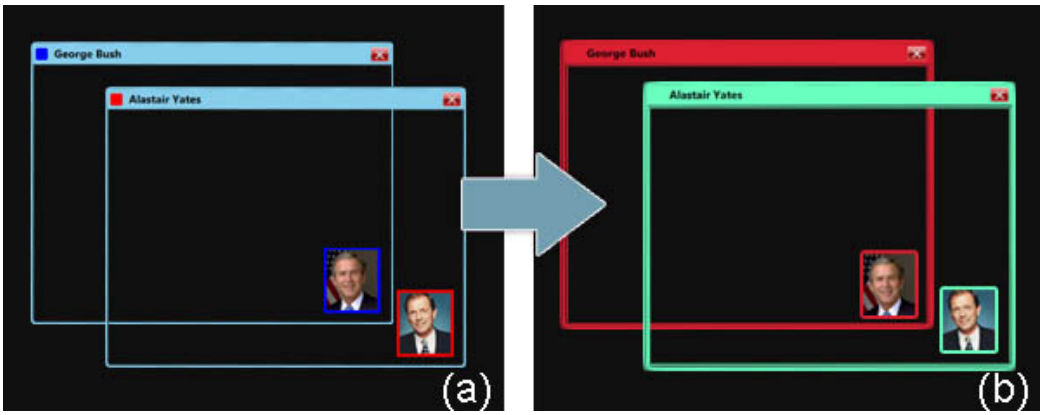

Figure 8.1: Improving the visualization of the dept of the lenses from the current situation (a) to situation (b).

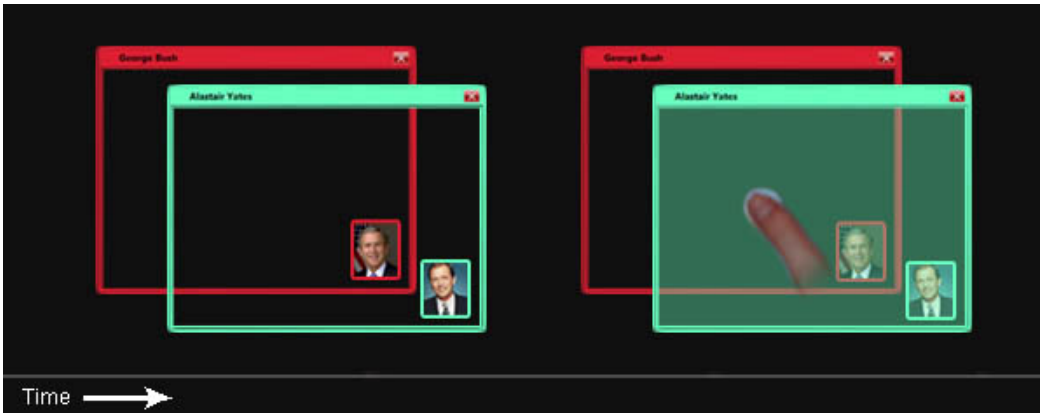

Figure 8.2: Improving the visualization of the dept of the lenses with a color overlay.

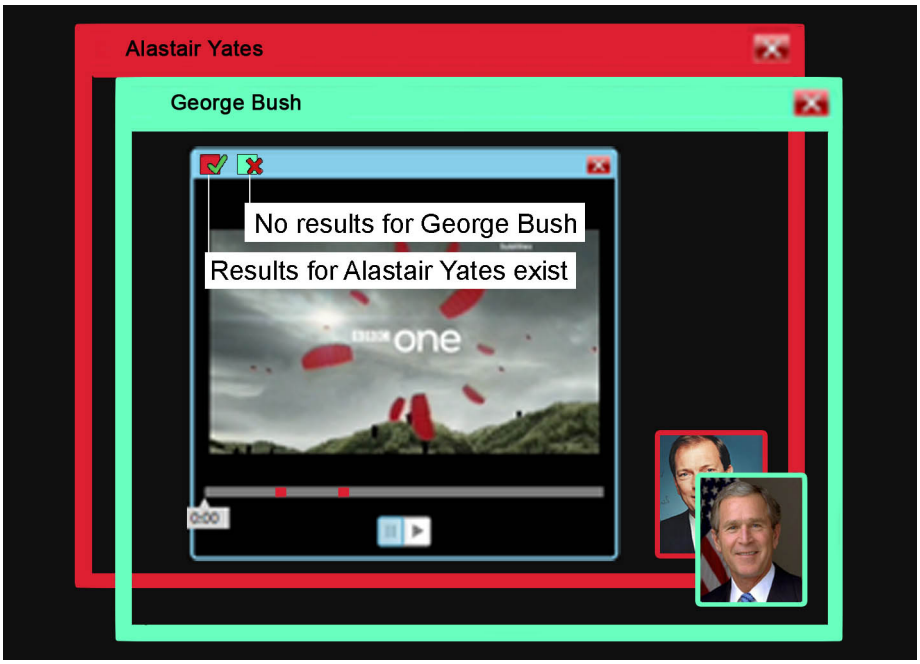

Figure 8.3: Improvements for visualizing the lenses on top of a video.

8.4 Creating an *AND*-relation

On the one hand, because the dept of the lenses was not clear, it was not easy for the test users to create an *AND*-relation between lenses. On the other hand, performing a three-finger gesture appears to be a very difficult task because it can only be completed when using both hands. In contrast, the usability test shows that the visualization of an *AND*-relation in a compound lens is very intuitive. Therefore the original tree-finger gesture can be replaced with a gesture that represents a composition, for example, dragging the images in the lenses on top of each other. Furthermore, this gesture is more consistent with the gesture that breaks an *AND*-relation (see figure 5.10).

8.5 Improving the Visualization of Relations

Based on the experiences obtained during this work, I would recommend to visualize both *AND* and *OR*-relations between lenses in a compound lens. Due to the fact that there is a spacial connection between lenses in a compound lens, the relation will be represented more clearly. Furthermore, this improvement provides a more consistent visualization for the *AND* and *OR*-relation. Switching between these two relation types can be achieved with a sliding widget in the compound lens (see figure 8.4).

To make a distinction between separated lenses and lenses in an *OR*-relation in this improvement, I suggest to visualize only the results for the query created by the compound lens and to remove the visualization of fragments in this case (see figure 8.4). In the initial version of the application, query results of compound queries are only entire videos. Accordingly, complex queries on different levels can be supported (see section 8.6).

8.6 Filtering on different levels

In this work, moving a compound lens over a video results in a visualization of the same frames as moving all these lenses separately over this video. As a result, the relation type (*AND* or *OR*) has no effect on the visualization of frames. In contrast, switching the relation type can affect the visualization of the media player frame if the video contains relevant fragments.

Actually, the relations in this thesis only support queries to retrieve entire videos. Accordingly, these relations can be extended to support queries on three different levels: to retrieve relevant videos (current situation), stories or frames. To support these kind of queries more events need to be detected in the archive so the occurrence of more events in one story or even one frame are more likely. This will open up new possibilities to support queries for retrieving much more precise results, for example:

- Retrieving all frames containing George Bush in a car.

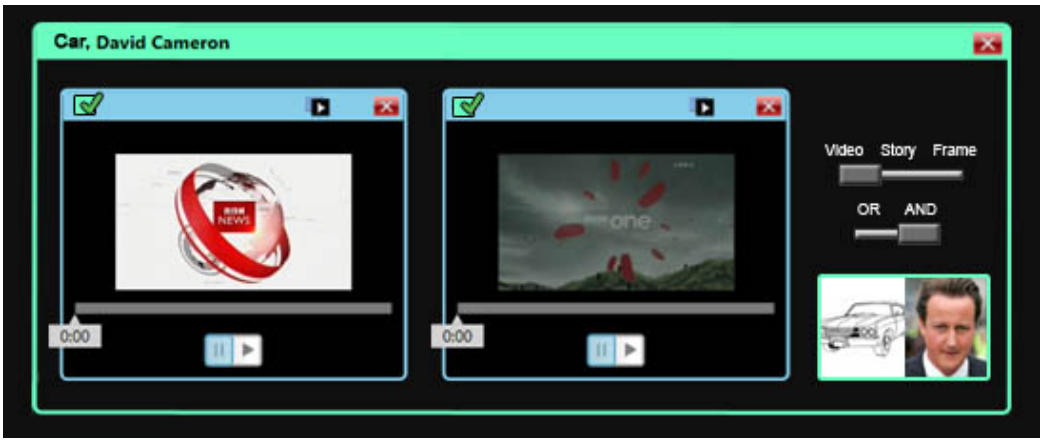

Figure 8.4: Finding videos containing fragments of a car *AND* David Cameron.

- Retrieving all stories containing the army and Hillary Clinton.

To switch the level (videos, stories or frames) on which a compound lens is filtering data, a sliding widget in the lens can be provided (see figures 8.4, 8.5 and 8.6). Since separate lenses no longer represent an *OR*-relation (see section 8.5), stacking lenses on a video will result in an execution of independent queries as well as an independent visualization of their results (see figure 8.7).

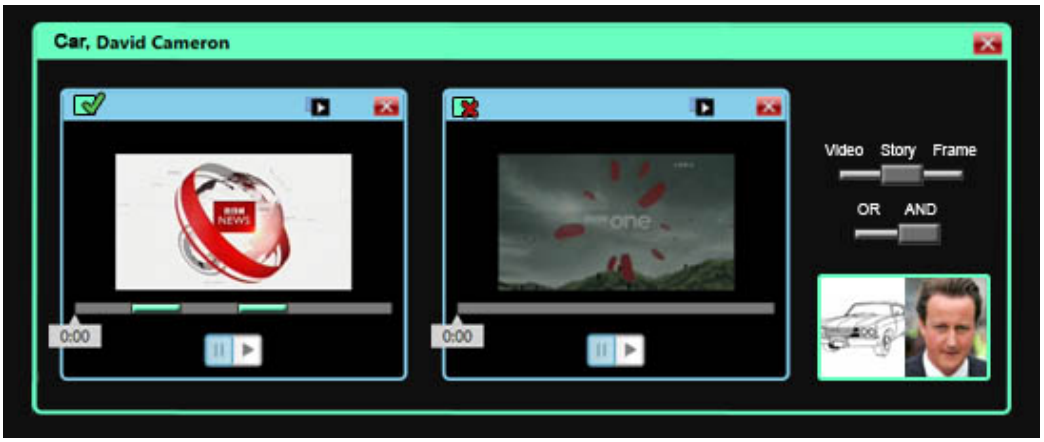

Figure 8.5: Finding stories containing fragments of a car *AND* David Cameron.

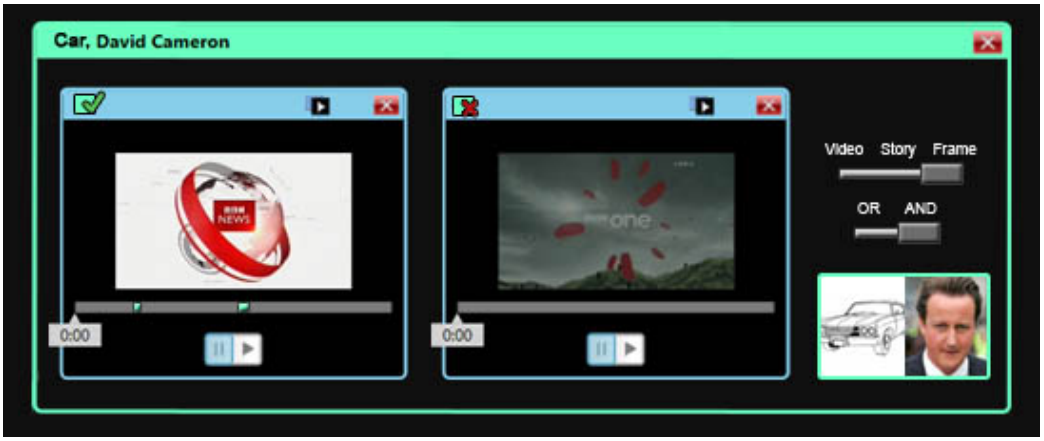

Figure 8.6: Finding frames containing a car *AND* David Cameron.

Figure 8.7: Multiple lenses stacked on top of each other results in an execution of independent queries.

Chapter 9

Conclusion

In this thesis an application is created to demonstrate the possibilities of lenses for analyzing video data. Not only the advantages when using lenses are revealed but also the problems when interacting with lenses are identified. Furthermore, a media player is created to support the right widgets for visualizing the occurrence of interested events in time. On top of that, some advanced features are implemented such as relations between lenses to speed up the process of finding results for more complex queries.

The application is implemented on an *FTIR* multi-touch screen to support a natural interaction with the application. First of all, this technology is compared to other optical based technologies to determine the possibilities. Later in this work, an overview of the detection and processing of the multi-touch input points is provided. A couple of other steps were added to this chain to provide the detection of events on specific objects in a program.

Multi-touch platforms bring new challenges such as input feedback ambiguities and problems when more users are interacting with the same surface. These problems are analyzed and solutions are provided. Furthermore, some direct and symbolic gestures are created to make the interaction more natural. These gestures can replace static icons in traditional interfaces and therefore support multiple users to work with the system at the same time. As a result, a robust layer is added to the input system to support an arbitrary number of gestures.

Finally, a usability test delivered great feedback about the interaction with the lenses. The pro and cons of the magic lenses could be clearly identified. Even improvements are provided to make the interaction more intuitive and easier.

Samenvatting (Dutch Summary)

Tegenwoordig zijn er veel video's beschikbaar door de opkomst van webcams, beveiligingssystemen, het internet, televisie opnames, etc. Het vinden van fragmenten in dit grote aanbod kan zeer moeilijk en tijdrovend zijn. Er bestaan methoden voor het indexeren van videofragmenten maar dit levert nog steeds een grote hoeveelheid data op waar de persoon zelf mee overweg moet kunnen. Het introduceren van interactieve tools in programma's voor het analyseren van video's kan het zoekproces naar fragmenten versnellen. In deze bachelorproef zal onder andere onderzocht worden hoe magic lenses gebruikt kunnen worden om fragmenten te zoeken in een grote hoeveelheid video data.

Magic lenses (zie hoofdstuk 5) zijn semi-transparante User Interface elementen die onderliggende data op een snelle en intuïtieve manier kunnen visualiseren. In onze applicatie zal elke lens een bepaalde persoon voorstellen. Wanneer de lens over een video geplaatst wordt, zullen de fragmenten waarin deze persoon voorkomt, opgelicht worden. Om complexere zoekopdrachten te vergemakkelijken, zoals het zoeken naar meerdere personen, kunnen er relaties gemaakt worden tussen de verschillende lenzen.

Het zoeken naar gewenste fragmenten in een video moet op een snelle mannier gebeuren. Gebruikers mogen immers geen vertraging ondervinden bij het verplaatsen van een lens. Om het detecteren van fragmenten te versnellen wordt daarom het AMASS++ videoarchief (zie hoofdstuk 4) gebruikt. Dit archief bevat gedetailleerde informatie over een heel aantal BBC nieuws-

opnames en is voornamelijk gespecialiseerd in het herkennen van gezichten in video's.

Het grote voordeel bij het gebruik van magic lenses is dat de tool fysiek over de inhoud van het programma geplaatst kan worden een toolbar die alle tools verzamelt is dus overbodig. Dit heeft als gevolg dat de aandacht van de gebruiker volledig op de inhoud van de applicatie gericht kan zijn. Wanneer ervoor gezorgd kan worden dat gebruikers de lenzen en andere objecten direct kunnen manipuleren in plaats van het gebruik van een traditionele muis en toetsenbord zal de gebruiker zich volledig kunnen concentreren op het werkvlak. Dit kan uiteindelijk leiden tot een zeer intuïtieve interactie. De applicatie zal daarom geïmplementeerd worden op een Multi-touch tafel (zie hoofdstuk 2).

Multi-touch interfaces brengen enkele nieuwe interactie technieken met zich mee zoals bijvoorbeeld het gebruik van gestures en het ondersteunen van meerdere gebruikers die samenwerken op eenzelfde tafel. Bovendien moeten er in multi-touch applicaties extra visualisaties voorzien worden om bijvoorbeeld feedback te geven over input punten. In Hoofdstuk 3 zal uitvoerig besproken worden welke elementen voorzien zijn in de applicatie om een intuïtieve interactie te verkrijgen.

Al deze elementen zullen geïntegreerd worden in een programma waarmee video's geanalyseerd kunnen worden op een multi-touch tafel. Tenslotte zullen gebruikerstesten de positieve en negatieve kanten van de applicatie aan het licht brengen (zie hoofdstuk 7).

Appendix A

Usability Test: Instruction Document (in Dutch)

A.1 Inleiding

Deze gebruikerstest speelt zich af binnen het kader van mijn bachelor thesis. In deze thesis wordt er voornamelijk onderzocht hoe er op een snelle en gemakkelijke manier videofragmenten uit een grote hoeveelheid video data gezocht kan worden.

Het zoeken naar fragmenten gebeurt met behulp van *magic lenses*. Deze semi-transparante tools stellen elk een persoon voor, en maken het mogelijk fragmenten te filteren waarin de desbetreffende persoon voorkomt.

De applicatie ondersteunt momenteel een klein BBC archief van volledige nieuwsuitzendingen en is beschikbaar op een multi-touch tafel.

A.2 Verloop van de test

Allereerst zal een handleiding over het gebruik van de applicatie gegeven worden aan de hand van een video. Bij het bekijken van deze video zullen regelmatig pauzes ingelast worden zodat je de verrichte handelingen zelf kan uitvoeren op de multi-touch tafel.

Vervolgens zal je gevraagd worden 2 taken uit te voeren die gelijkaardig zijn aan de voorbeeldjes uit de video.

Tenslotte kan je je mening geven over deze applicatie aan de hand van een vragenlijst.

A.3 Taken

Taak 1

In hoeveel video's komen Gordon Brown *EN* Nicolas Sarkozy *EN* Morgan Tsvangirai voor? Speel een fragment van Nicolas Sarkozy af.

Taak 2

Verbreek de *AND*-relatie tussen de lenses uit taak 1 en ga na in hoeveel video's fragmenten van Gordon Brown *OF* Nicolas Sarkozy voorkomen.

A.4 Vragenlijst

- Algemene informatie:
  Geslacht: M / V
  Leeftijd: .....

- Het doel van de lenses (visualiseren van gevonden fragmenten) is duidelijk.

  Volledig niet akkoord 1 2 3 4 5 Volledig akkoord

- Het gebruik van lenses voor het zoeken naar fragmenten vind ik zeer nuttig.

  Volledig niet akkoord 1 2 3 4 5 Volledig akkoord

- Het is duidelijk dat lenses over een video geplaatst moeten worden om resultaten zichtbaar te maken.

  Volledig niet akkoord 1 2 3 4 5 Volledig akkoord

- Het gebruik van kleuren bij het visualiseren van fragmenten vind ik duidelijk.

  Volledig niet akkoord 1 2 3 4 5 Volledig akkoord

  Opmerkingen: .....
  .....
  .....

- Ik vind het zeer intuïtief dat de functionaliteiten van de media player (afspelen, pauzeren, terugspoelen, vooruitspoelen) bediend kunnen worden wanneer er zich lenses over de video bevinden

  Volledig niet akkoord 1 2 3 4 5 Volledig akkoord

- De gekleurde rechthoekjes in de linkerbovenhoek van een video (zie afbeelding A.1) visualiseren op een duidelijke manier de lenses die zich bovenop de video bevinden.

  Volledig niet akkoord 1 2 3 4 5 Volledig akkoord

  Opmerkingen: .....
  .....
  .....

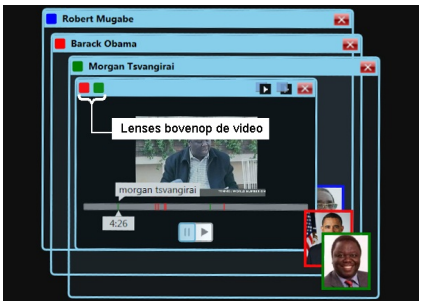

Figure A.1: Het visualiseren van lenses die zich bovenop de video bevinden.

- Het gebruik van *AND*-relaties tussen de lenses vergemakkelijkt het zoekproces naar video's waarin meerdere personen voorkomen.

  Volledig niet akkoord 1 2 3 4 5 Volledig akkoord

  Opmerkingen: .....
  .....
  .....

- De gesture voor het definiëren van een *AND*-relatie, was gemakkelijk aan te leren.

  Volledig niet akkoord 1 2 3 4 5 Volledig akkoord

- De gesture om lenses terug in een *OR*-relatie te brengen was gemakkelijk aan te leren.

  Volledig niet akkoord 1 2 3 4 5 Volledig akkoord

- Het samenvoegen van lenses is een duidelijke manier voor het weergeven van een *AND*-relatie.

  Volledig niet akkoord 1 2 3 4 5 Volledig akkoord

- Het minder zichtbaar worden van video's is een duidelijke manier om video's aan te duiden die geen resultaten voor de zoekopdracht bevatten.

  Volledig niet akkoord 1 2 3 4 5 Volledig akkoord

HARTELIJK DANK VOOR UW MEDEWERKING!

Bibliography

[1] SBO Program of the IWT. Advanced multimedia alignment and structured summarization (amass++). http://www.cs.kuleuven.be/ĩir/projects/amass/, Last visited: May 04, 2010.

[2] T. Tuytelaars S. Martens, J. H. Becker and M.F. Moens. *Multimodal Data Collection in the AMASS++project*. Katholieke Universiteit Leuven, Belgium, 2008.

[3] Microsoft Research Bill Buxton. Multi-touch systems that i have known and loved, jan 2007. http://www.billbuxton.com/multitouchOverview.html, Last visited: May 10, 2010.

[4] Microsoft Windows Presentation Foundation. Data binding overview. http://msdn.microsoft.com/en-us/library/ms752347.aspx, Last visited: May 09, 2010.

[5] Pierre Wellner. The digitaldesk calculator: tangible manipulation on a desk top display. In *UIST '91: Proceedings of the 4th annual ACM symposium on User interface software and technology*, pages 27–33, New York, NY, USA, 1991. ACM.

[6] Kathy Ryall, Clifton Forlines, Chia Shen, Meredith Ringel Morris, and Katherine Everitt. Experiences with and observations of direct-touch tabletops. In *TABLETOP '06: Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pages 89–96, Washington, DC, USA, 2006. IEEE Computer Society.

[7] Enrico Costanza and John Robinson. A region adjacency tree approach to the detection and design of fiducials. In *VVG*, pages 63–69, 2003.

[8] N. Metha. *A Flexible Machine Interface*. M.A.Sc. Thesds, Department of Electrical Engineering, University of Toronto., 1982.

[9] the free encyclopedia Wikipedia. Infrared. `http://en.wikipedia.org/wiki/Infrared`, Last visited: April 10, 2010.

[10] the free encyclopedia Wikipedia. Apple iphone. `http://en.wikipedia.org/wiki/Apple_iPhone`, Last visited: April 15, 2010.

[11] Ralph Hill William Buxton and Peter Rowley. Issues and techniques in touch-sensitive tablet input. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 215–224. ACM Press, 1885.

[12] Jefferson Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118, New York, NY, USA, 2005. ACM.

[13] the free encyclopedia Wikipedia. Heideggerian terminology. `http://en.wikipedia.org/wiki/Heideggerian_terminology`, Last visited: May 08, 2010.

[14] Microsoft .NET. `http://www.microsoft.com/net/`.

[15] the free encyclopedia Wikipedia. .net framework. `http://en.wikipedia.org/wiki/Net_Framework`, Last visited: May 09, 2010.

[16] Daniel Wigdor, Sarah Williams, Michael Cronin, Robert Levy, Katie White, Maxim Mazeev, and Hrvoje Benko. Ripples: utilizing per-contact visualizations to improve user interaction with touch displays. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User*

interface software and technology, pages 3–12, New York, NY, USA, 2009. ACM.

[17] Microsoft. Microsoft surface. `http://www.microsoft.com/surface/en/us/default.aspx`, Last visited: May 02, 2010.

[18] Microsoft. Windows presentation foundation. `http://windowsclient.net/wpf/`, Last visited: May 09, 2010.

[19] Johannes Schöning, Jonathan Hook, Tom Bartindale, Dominik Schmidt, Patrick Oliver, Florian Echtler, Nima Motamedi, and Peter Brandl. Building interactive multi-touch surfaces. *journal of graphics, gpu, and game tools*, 14(3):35–55, 2009.

[20] Johannes Schöning, Peter Brandl, Florian Daiber, Florian Echtler, Otmar Hilliges, Jonathan Hook, Markus Löchtefeld, Nima Motamedi, Laurence Muller, Patrick Olivier, Tim Roth, and Ulrich von Zadow. Multi-touch surfaces: A technical guide. techreport, Institute for Geoinformatics University of MÃ¼nster, 2008.

[21] Tom Cuypers, Jan Schneider, Johannes Taelman, Kris Luyten, and Philippe Bekaert. Eunomia: toward a framework for multi-touch information displays in public spaces. In *BCS-HCI '08: Proceedings of the 22nd British CHI Group Annual Conference on HCI 2008*, pages 31–34, Swinton, UK, UK, 2008. British Computer Society.

[22] Don Norman. Natural user interfaces are not natural. `http://www.jnd.org/dn.mss/natural_user_interfaces_are_not_natural.html`, Last visited: May 29, 2010.

[23] Leonard R. Kasday. Touch position sensitive surface, 1984. `http://www.google.com/patents?vid=USPAT4484179`.

[24] James B. Mallos. Touch position sensitive surface, 1982. `http://www.google.com/patents?vid=USPAT4346376`.

[25] Robert E. Mueller. Direct television drawing and image manipulation system, 1974. `http://www.google.com/patents?vid=USPAT3846826`.

[26] Ken Fishkin and Maureen C. Stone. Enhanced dynamic queries via movable filters. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 415–420, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[27] C. Stone E. A. Bier, W. Buxton K. Pier, and T. D. DeRose. *Toolglass and Magic Lenses: The see-Through Interface*. University of Toronto, University of Washington, Xerox PARC, 3333 Coyote Hill Road, Palo Alto, CA 94304, 2003.

[28] Kai-Yin Cheng, Sheng-Jie Luo, Bing-Yu Chen, and Hao-Hua Chu. Smartplayer: user-centric video fast-forwarding. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 789–798, New York, NY, USA, 2009. ACM.

[29] the free encyclopedia Wikipedia. Microsoft surface. `http://en.wikipedia.org/wiki/Microsoft_Surface`, Last visited: April 20, 2010.

[30] the free encyclopedia Wikipedia. Frustrated total internal reflection. `http://en.wikipedia.org/wiki/Total_internal_reflection`, Last visited: April 13, 2010.

[31] NUI Group. `http://nuigroup.com/touchlib/`, Last visited: April 22, 2010.

[32] NUI Group. `http://wiki.nuigroup.com/FTIR`, Last visited: April 22, 2010.

[33] Andrew D. Wilson. Touchlight: an imaging touch screen and display for gesture-based interaction. In *ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces*, pages 69–76, New York, NY, USA, 2004. ACM.

[34] K. Coninx M. Haesen, J. Meskens. *Visualising Digital Video Libraries for TV Broadcasting Industry: A User-Centred Approach.* Expertise Centre for Digital Media, Belgium, 2009.

[35] the free encyclopedia Wikipedia. Windows presentation foundation. `http://en.wikipedia.org/wiki/Windows_Presentation_Foundation`, Last visited: May 09, 2010.

[36] Microsoft. Xaml overview. `http://msdn.microsoft.com/en-us/library/ms752059.aspx`, Last visited: May 09, 2010.

[37] the free encyclopedia Wikipedia. Extensible application markup language. `http://en.wikipedia.org/wiki/XAML`, Last visited: May 09, 2010.

List of Figures

2.1 Some very important points in the technical evolution of the touch technology. 5
2.2 The use of mirrors to reduce the distance between the projector and the projection surface [20]. 8
2.3 The bare minimum parts needed for a FTIR setup [32]. 10
2.4 FTIR schematic diagram depicting some improvements to increase touch sensitivity [20]. 11
2.5 General setup of a Rear-side illumination system [19]. 12
2.6 General setup of a Front-side illumination system [19]. 13
2.7 Advantages and disadvantages of FTIR, Rear DI and Front DI compared to each other. 15

3.1 The multi-touch video analysis application. 19
3.2 Selecting multiple objects(a), (b) and organizing all these objects(c). 21
3.3 Popping up a menu(a),(b) and add a video(c). 23
3.4 Scrolling through content by performing a natural move gesture. 25
3.5 Touch-points states and transitions. 0: not yet touching 1: stationary contact [16]. 28
3.6 Two animations are shown for transition A. If an object is captured, a circle shrinks around the contact. If not, it *splashes* outward. 30
3.7 Transistion B: emphasizing the point/finger mapping to address the fat finger problem. 30

3.8 Touch-points states and transitions. 1: stationary contact 2: moving contact [16]. 30
3.9 Two visualizations are shown for state 2. If an object is captured, a short trail is displayed behind the finger while moving. If not, the full path will be visualized. 32

4.1 An example of the subtitles in a story in the AMASS++ archive. 35
4.2 In memory data structure of the annotated AMASS++ data. . 37

5.1 Curvature pseudo-color lens with overlaid tool to read the numeric value of the curvature [27]. 39
5.2 A color toolglass applies its style to an underlying part of a circle by *clicking through* the fill-color button [27]. 39
5.3 All cities with high salaries OR low taxes are highlighted as a result of two disjunctive lenses [26]. 40
5.4 A Media player with a specialized timeline widget to integrate query results. 44
5.5 The Morgan Tsvangirai lens entirely overlaps the timeline, the Barack Obama lens partially overlaps the timeline. 44
5.6 The Morgan Tsvangirai and Barack Obama lenses both overlapping the timeline. 45
5.7 Highlighting all frames containing Robert Mugabe *OR* Alastair Yates. 47
5.8 Switching the relation type between two lenses to an *AND*-relation by performing a three-finger gesture. 48
5.9 Highlighting all frames containing Robert Mugabe *AND* Alastair Yates. 49
5.10 Switching the relation type between two lenses to an *OR*-relation. 50
5.11 The *Push to front* button and *Push to background* button are positioned at the right side of the video. Two squares representing the lenses on top of the video are located on the left side of the video. 51

6.1 The Model-View-ViewModel pattern. 55

6.2 Basic data structure of a video. (Figure 4.2 shows the full class diagram of the AMASS++ *MediaStream*). 58
6.3 The basic input architecture of the application. 60
6.4 Mapping of recognized events to events on objects. 62

7.1 A barchart of the results of the questionnaire (The statements can be found in table 7.2). 70

8.1 Improving the visualization of the dept of the lenses from the current situation (a) to situation (b). 74
8.2 Improving the visualization of the dept of the lenses with a color overlay. 74
8.3 Improvements for visualizing the lenses on top of a video. . . . 75
8.4 Finding videos containing fragments of a car *AND* David Cameron. 77
8.5 Finding stories containing fragments of a car *AND* David Cameron. 78
8.6 Finding frames containing a car *AND* David Cameron. 78
8.7 Multiple lenses stacked on top of each other results in an execution of independent queries. 79

A.1 Het visualiseren van lenses die zich bovenop de video bevinden. 87

List of Tables

7.1 Age range of the participants. 66
7.2 Detailed report of the questionnaire. 69