Spatial Navigation for Context-aware Video Surveillance

Huib Piguillet, April 21, 2010
Spatial Navigation for Context-aware Video Surveillance

THESIS

submitted in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Huib Piguillet
born in Delft, the Netherlands

Visualisation and Virtual Reality
Computer Graphics and CAD/CAM group
Faculty of EEMCS
Delft University of Technology
Cover picture: The illustration shows a person that is walking in an area monitored by cameras. The screen at the bottom displays the prototype that is developed for this thesis. This prototype can be used to navigate through the camera recordings. The blue arrow represents such a navigation path. The illustration is created with Inkscape.
Spatial Navigation for Context-aware Video Surveillance

Author: Huib Piguillet
Student number: 1109952
E-mail: h.piguillet@gmail.com

Abstract

An increasing number of cameras is being used to monitor a growing range of environments. Consequently, surveillance systems consist of an increasing number of screens to display all incoming video streams, making it difficult for observers to maintain a mental model of spatial relations between videos. A number of systems is developed aiming at improving the observer’s spatial awareness by integrating videos with their spatial context, a 3D model of the monitored environment. In these systems, video content can be viewed from virtual cameras that correspond with the cameras in the real world. To switch between views on certain videos, the observer has to make a transition between the corresponding virtual cameras by some means of navigation.

While studying state-of-the-art 3D surveillance systems, we have observed that not much attention has been paid on exploring more sophisticated viewpoint transitions. In this thesis, different classes of camera relations are identified. For each class, a viewpoint transition mechanism is developed with the focus on reducing distortion of the video information during transitions.

In order to navigate through the virtual environment, viewpoint transitions have to be initiated to switch between videos. As part of this thesis, a number of concepts have been developed that provide controls to the viewpoint transitions in the form of 3D elements that are added to the virtual environment.

We have implemented our navigation concepts in a prototype, which was used for a user study. This thesis concludes with the results from this user study and our vision on future work in the field of context-aware surveillance.

Thesis committee:

Chair: prof. dr. ir. F.W. Jansen
Supervisors: ir. F.H. Post
            dr. ir. G. de Haan
Committee members: dr. ir. W.P. Brinkman
                  ir. M.G. van Elk
This thesis continues on earlier work by Josef Scheuer on advanced video surveillance systems. While further researching this subject, I learned a great deal in several areas, including computer graphics, virtual reality, human computer interaction and visual perception. I would like to thank a number of people who helped me throughout this last phase of my study.

In the first place, I want to thank my supervisors Frits Post and Gerwin de Haan. Frits, thank you for giving me the opportunity to work on this interesting subject. Your ideas and view on the subject refreshed my mind and encouraged me to keep being creative in finding solutions. Gerwin, your broad knowledge and insights in many different fields really inspired me to look at things from different points of view. I appreciate all the fruitful discussions we had, leading to such a great amount of ideas. Furthermore, I want to thank all the people from “the lab” and the rest of the twelfth floor, with whom I had a very pleasant time and who provided me with useful feedback after observing, playing with my work-in-progress or participating in my experiments. Finally, I would like to thank my parents, who have always supported me throughout my study.

Huib Piguillet

Delft, The Netherlands
April 21, 2010
Contents

Preface v

1 Introduction 1
  1.1 Related Work ........................................... 2
  1.2 Project Goals ........................................... 3
  1.3 Thesis Structure ......................................... 4

2 View Transitions 5
  2.1 Video Canvas Blending ................................... 6
     2.1.1 Canvas Positioning ................................ 6
     2.1.2 Canvas Blending ................................... 7
  2.2 Transition types ........................................ 7
     2.2.1 Zooming ........................................... 9
     2.2.2 Panoramic movement ............................... 9
     2.2.3 Orbiting ........................................ 11
  2.3 View Transition Graphs .................................. 15
  2.4 Summary ................................................ 15

3 Navigation 17
  3.1 Margin Videos ......................................... 17
  3.2 3D Cursor .............................................. 19
     3.2.1 Cursor Shape ...................................... 19
     3.2.2 Cursor Ghost ...................................... 19
  3.3 In-Video Navigation .................................... 20
     3.3.1 Glyph menu ........................................ 20
     3.3.2 Orbit Glyphs ....................................... 21
  3.4 Camera Selection Feedback .............................. 22
  3.5 The Navigation Tools as an Integrated Set ............ 22
  3.6 Summary ................................................ 23

4 System Architecture and Implementation 25
  4.1 System Architecture .................................... 25
     4.1.1 Controllers ........................................ 26
     4.1.2 Model ............................................... 27
     4.1.3 View ................................................ 28
  4.2 Implementation .......................................... 28
  4.3 Summary ................................................ 30

5 Evaluation 31
  5.1 Surveillance Scenarios ................................. 31
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.1</td>
<td>Office Hallway</td>
</tr>
<tr>
<td>5.1.2</td>
<td>PETS datasets</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Highway Traffic</td>
</tr>
<tr>
<td>5.1.4</td>
<td>Evaluation Environment</td>
</tr>
<tr>
<td>5.2</td>
<td>Experiment Description</td>
</tr>
<tr>
<td>5.3</td>
<td>Results</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Usage of Navigation Tools</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Task Results</td>
</tr>
<tr>
<td>5.4</td>
<td>Discussion</td>
</tr>
<tr>
<td>5.5</td>
<td>Summary</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions and Future work</td>
</tr>
<tr>
<td>6.1</td>
<td>Conclusions</td>
</tr>
<tr>
<td>6.2</td>
<td>Future work</td>
</tr>
<tr>
<td><strong>Bibliography</strong></td>
<td></td>
</tr>
<tr>
<td><strong>A</strong></td>
<td>Mathematical definitions</td>
</tr>
<tr>
<td>A.1 Bézier Curves</td>
<td>47</td>
</tr>
<tr>
<td>A.2 Common Perpendicular</td>
<td>47</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>Scenario Specifications</td>
</tr>
<tr>
<td>B.1 Scene Data</td>
<td>49</td>
</tr>
<tr>
<td>B.2 Camera Transition Graphs</td>
<td>52</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>Evaluation</td>
</tr>
<tr>
<td>C.1 Experiment Scenario</td>
<td>55</td>
</tr>
<tr>
<td>C.2 Questionnaire</td>
<td>56</td>
</tr>
<tr>
<td>C.3 Answers to the Closed Questions</td>
<td>58</td>
</tr>
<tr>
<td>C.4 Answers to the Open Questions</td>
<td>58</td>
</tr>
<tr>
<td><strong>D</strong></td>
<td>Article for IEEE CGA 2010</td>
</tr>
</tbody>
</table>
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Picture of a typical control room</td>
<td>2</td>
</tr>
<tr>
<td>2.1</td>
<td>An illustration of a smooth view transition</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>Positioning of a video canvas at several distances from its camera</td>
<td>6</td>
</tr>
<tr>
<td>2.3</td>
<td>Plot of canvas blending functions 2.1 and 2.2 over normalized time ( t )</td>
<td>7</td>
</tr>
<tr>
<td>2.4</td>
<td>Top view of a camera setup that illustrates the three transition that we classified based on their visual effects</td>
<td>8</td>
</tr>
<tr>
<td>2.5</td>
<td>Illustration showing the changing view that is the result of each of the view transition classes</td>
<td>8</td>
</tr>
<tr>
<td>2.6</td>
<td>Illustrations of the zooming transition path and canvas blending</td>
<td>10</td>
</tr>
<tr>
<td>2.7</td>
<td>Illustrations of the panoramic transition path and canvas blending</td>
<td>10</td>
</tr>
<tr>
<td>2.8</td>
<td>Comparison of viewpoint interpolation in world space to POI interpolation in image space</td>
<td>12</td>
</tr>
<tr>
<td>2.9</td>
<td>Illustrations of the orbit transition path and canvas blending</td>
<td>12</td>
</tr>
<tr>
<td>3.1</td>
<td>Example showing that the opacity of a canvas during an orbit transition depends on either the time ( t ) or the angle between the two end viewpoints</td>
<td>14</td>
</tr>
<tr>
<td>3.2</td>
<td>Example of a view transition graph</td>
<td>15</td>
</tr>
<tr>
<td>3.3</td>
<td>The distribution of video thumbnails on the margin region</td>
<td>18</td>
</tr>
<tr>
<td>3.4</td>
<td>Projection of the mouse pointer on the 3D scene</td>
<td>19</td>
</tr>
<tr>
<td>3.5</td>
<td>Video overlayed with two glyphs that can be selected to initiate a transition to a neighboring camera</td>
<td>20</td>
</tr>
<tr>
<td>3.6</td>
<td>Screen shots showing the 3D cursor before and after a mouse click</td>
<td>21</td>
</tr>
<tr>
<td>3.7</td>
<td>Top view of the 3D cursor showing the menu responding to mouse events</td>
<td>21</td>
</tr>
<tr>
<td>3.8</td>
<td>View frustum bounds appearing when the corresponding thumbnail is selected</td>
<td>22</td>
</tr>
<tr>
<td>3.9</td>
<td>Overview of the connections between navigation tools</td>
<td>22</td>
</tr>
<tr>
<td>3.10</td>
<td>Example of the interaction tools during the different navigation states</td>
<td>24</td>
</tr>
<tr>
<td>3.11</td>
<td>Overview of the connections between navigation tools</td>
<td>24</td>
</tr>
<tr>
<td>4.1</td>
<td>Diagram that shows how the Model-view-controller design pattern is applied to our surveillance system</td>
<td>25</td>
</tr>
<tr>
<td>4.2</td>
<td>Diagram depicting the flow of messages that are passed over the various components</td>
<td>26</td>
</tr>
<tr>
<td>4.3</td>
<td>The render order of the various scene graph objects</td>
<td>29</td>
</tr>
<tr>
<td>4.4</td>
<td>Class diagram that shows the connections between the important classes</td>
<td>29</td>
</tr>
<tr>
<td>5.1</td>
<td>The office scenario</td>
<td>32</td>
</tr>
<tr>
<td>5.2</td>
<td>Overview of the PETS scenarios</td>
<td>33</td>
</tr>
<tr>
<td>5.3</td>
<td>The traffic scenario</td>
<td>34</td>
</tr>
<tr>
<td>5.4</td>
<td>Impression and overview of the evaluation scenarios</td>
<td>35</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.5</td>
<td>Example of the evaluation environment showing the “orbit” scenario in two surveillance interfaces.</td>
<td>36</td>
</tr>
<tr>
<td>5.6</td>
<td>Picture of the experiment setup.</td>
<td>36</td>
</tr>
<tr>
<td>5.7</td>
<td>Histogram combining navigation tool preferences.</td>
<td>37</td>
</tr>
<tr>
<td>5.8</td>
<td>Confusion: before a transition, person A is right of person B, while after the transition, person A is left of person B in the next view (left).</td>
<td>38</td>
</tr>
<tr>
<td>5.9</td>
<td>Illustration of two cameras and their mutual coverage, introducing a dead area and a shared area.</td>
<td>38</td>
</tr>
<tr>
<td>5.10</td>
<td>Possible redesign of the orbit glyphs making them more consistent with arrow glyphs.</td>
<td>39</td>
</tr>
<tr>
<td>A.1</td>
<td>A bézier curve.</td>
<td>47</td>
</tr>
<tr>
<td>A.2</td>
<td>The common perpendicular of two lines.</td>
<td>48</td>
</tr>
<tr>
<td>C.1</td>
<td>Explanation of the navigation elements mentioned in Question 1.</td>
<td>57</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

While walking or driving through public places, one is likely to find a growing number of surveillance cameras. These are used to record situations in public places like roads, railway stations and shopping centers, as well as in private places like stores and offices. Although automatic methods exist to detect events or track moving objects, they are not suitable in all cases. Such cases include crowded places with poor lighting or other bad conditions. In these cases, it is impossible for current computer systems to segment persons and keeping track of them or to distinguish normal events from events that require further attention, such as aggression or suspicious behavior. Therefore, human observers remain necessary and they need a user interface that can at least show all videos. In most cases such an interface will offer some functionality to control which videos are shown and in which direction cameras are pointing, in the case of controllable (e.g. pan-tilt-zoom) cameras. In its simplest form, such a user interface consists of one or more screens that show a matrix view of all videos or a subset of them. Figure 1.1 shows the use such a user interface in a typical surveillance control room.

Tracking human activity between cameras (i.e. when a person moves from one camera view to another) can be a difficult task. In situations when there is not much activity, the observer’s attention is immediately drawn to the video in which a subject reappears after disappearing in one of the videos. But when there is constant activity going on in most of the videos, for instance in case of a crowded train station, it becomes difficult for both human observers and computer vision algorithms to keep track of the subject. In these cases, this tracking is a repeating process until it is not necessary to observe the subject anymore. Each step of this process involves several mental transitions to follow a person or moving object that disappears from camera A and reappears in camera B. First, the observer has to locate camera A on an environment map, which can be either a physical one or a map he has stored in his memory after some training. After observing the video footage of camera A in which direction the subject is moving, the map has to be consulted again to make a prediction on which camera the subject will appear next. This video then has to be located on the matrix view and the process will be repeated each time the subject moves out of sight of a camera.

As the tracking process involves a lot of mental transitions to relate the video content to positions on the environment map, it can cause cognitive overload, especially for people new to the environment who have therefore not built up a mental map in their minds. One way to reduce this problems is to integrate the videos with a spatial representation of the environment to reduce the mental effort needed to remain spatial awareness (i.e., the sense of knowing what one is looking at and from what perspective). A crucial part of such an integrated system is the set of navigation tools to control the view on the spatial representation, resulting in a different view on the videos.
1. Introduction

Figure 1.1: Picture of a typical control room, which contains a) a large shared screen; b) a matrix of screens; c) a matrix of videos on a single screen; d) a PTZ camera control; e) observers; and f) private video screens (courtesy of Chelmsford Borough Council, GB, [http://www.chelmsford.gov.uk/index.cfm?articleid=8882](http://www.chelmsford.gov.uk/index.cfm?articleid=8882)).

As part of the thesis project, a set of navigation concepts is designed that should make it easy to maintain spatial awareness while changing views on video streams. Before we go deeper on the matter of the project, a brief summary of related work will be given to point out the direction the project is driven to.

1.1 Related Work

Prior to this thesis project, a literature study [Piguillet, 2009] has been performed on the subject of navigation in 3D surveillance systems, which consists of two main parts. The first part focuses on state-of-the-art surveillance systems and the way videos are integrated in a 3D environment in these systems. The second part gives an overview of various navigation techniques that can be applied to the passive travelling between fixed viewpoints in 3D surveillance systems. These navigation techniques include selection with a 3D cursor [Hachet et al., 2008, Mackinlay et al., 1990, Zeleznik and Forsberg, 1999], region selection [Snavely et al., 2006] and gesture based navigation [Igarashi et al., 1998].

A simple step to add spatial relations to videos is described in [Girgensohn et al., 2007a,b]. In their Spatial Multi-Video player, the placement of videos in relation to one particular video of interest is chosen in such a way that they correspond the most with the camera positions on a map that is also included in the user interface. In addition, the videos have a colored border that corresponds with the camera colors on the map.

In other work, a third dimension is added to the spatial representation, by putting the videos in a 3D virtual environment model of the monitored area. [Sawhney et al., 2002, Sebe et al.]
2003] realised this by projecting videos as textures over the model, while [de Haan et al., 2009, Girgensohn et al., 2007a, b, Wang et al., 2007] introduce canvas objects on which the videos are attached to. [de Haan et al., 2009] involves several mechanisms to avoid distortions of the videos with the underlying 3D scene and adds guided navigation methods [Galyean, 1995] to restrict navigational freedom, ensuring that canvases are viewed from optimal positions. The canvas positioning and blending in this work is based on techniques used in Photo Tourism [Snavely et al., 2008, 2006], a program that enables users to browse photos as if they are flying through the photographed environment.

1.2 Project Goals

The basic components that the “3D surveillance systems” mentioned above have in common are a 3D model of an environment that is monitored by cameras, somehow overlaid with videos from those cameras. The parameters of the viewpoint, such as position, orientation and field of view, determine the projection of the 3D scene on the image plane (typically a monitor screen), and therefore the view on the videos as well. If the viewpoint parameters match those of the real cameras, the corresponding video will fill up the whole image plane.

Switching between videos requires a change of the viewing parameters to end up on the position of another camera. If we change these parameters smoothly over time, the function of the viewing parameters over time influences the path of the viewpoint and therefore the visual change in image space. As it should be as easy as possible for the observer to keep track of important video content, we need to find a transition function that minimizes image distortion and maximizes continuity of the view change. This problem leads to our first goal:

**Research goal 1.** Given two virtual cameras, each pointed at a video canvas, how must we define a smooth viewpoint transition such that it enables the viewer to keep track of video content without much mental effort?

It would be practical if we could define a general transition function that works in each case of two given cameras, or at least a small set of transition functions that covers all cases. For that reason an important part of Chapter 2 is the classification of camera relations.

When we have found and implemented such transition mechanisms, we would also need the means to control them. In [de Haan et al., 2009], transitions through a set of cameras can be initiated by pressing a “previous” or “next” button, similar to a camera rotor system that switches videos that are displayed on a screen. Photo tourism [Snavely et al., 2006] includes some other interaction tools, such as region selection on a photo to zoom in on a specific part and a set of thumbnails at the bottom of the screen to switch from the current photo to another one. Instead of having to search for the video that we have to switch to in order to keep track of certain video content, it would be more natural and intuitive to navigate to other videos from within the same 3D environment, by using spatial cues and controls. Since there is a wide variety of 3D navigation widgets and other navigation concepts, the problem is to find or invent the ones that can be used to control our viewpoint transitions and are intuitive and easy to use, focusing on reducing the user’s mental effort.

**Research goal 2.** Which navigation concepts can be added to control viewpoint transitions in a 3D surveillance system that reduce the effort it takes for the user to track activity between videos?

Aimed at reaching this second goal, we propose a surveillance system, containing a set of navigation tools that are highly linked to each other and can be used interchangeably to initiate
camera transitions. Each camera pair can be linked to one of the transition mechanisms that is included. The next chapters describe these transition mechanisms and navigation concepts and the motivation behind their design, as well as the evaluation setup and results, using a prototype that brings the navigation concepts together.

1.3 Thesis Structure

This thesis is structured as follows: Chapter 2 describes a system of view transitions that form the basis of our surveillance system. A conceptual design of the navigation tools that for the control of these view transitions is then given in Chapter 3. An implementation of these concepts in the form of a prototype is described in Chapter 4. This prototype was used for the user study we conducted to evaluate the usage of the navigation tools in specific scenarios. This evaluation is described in Chapter 5. Finally, in Chapter 6, we discuss if the research goals are met, to conclude with a vision on future work on the subject of this thesis.
In the previous Chapter, we described the context-aware surveillance system [de Haan et al., 2009] that serves as a basis that we have extended with the navigation tools we will describe in the following chapters. In this system, environment models of a monitored area can be loaded in a 3D environment, along with virtual cameras that correspond with the real surveillance cameras. Videos that come from the cameras are then displayed on canvas objects that are placed before the virtual cameras. This way, a full view on a specific video can be achieved when the viewpoint is matched with one of the virtual cameras. To switch between videos, one must somehow navigate the viewpoint from one virtual camera to another one. We will call such navigation steps view transitions. In this Chapter, we will further define the concept of view transitions and classify them based on their visual effects.

We can either choose to make each view transition instantaneously or interpolate the viewpoint of the view point in time between two virtual cameras, according to a certain function. The former method has the advantage that there is no “temporal gap” during the transition: after switching, the next video continues where the previous one has ended. The latter method has the advantage that there is no “spatial gap”: a smooth viewpoint trajectory enables the user to maintain continuity in perception, or as [Bowman et al., 1997] concludes: “jumping techniques can reduce the user’s spatial awareness”. Therefore, a view transition should be designed that it can enhance the user’s spatial awareness in an as short as possible time span.

Each virtual camera has three parameters that determine their view on the environment: its position, orientation and field of view (which is related to the lens angle). A transition function that corresponds to two given cameras must somehow interpolate these parameters over time. The resulting values can then be used to parameterize the viewpoint, resulting in a view transition. The interpolation of the viewpoint position forms a transition path. The velocity of the viewpoint as it moves over this transition path can be kept constant, but other velocity functions can result in improved spatial awareness such as a slowing down function (gradually decrease the velocity towards zero at the end point, see [Mackinlay et al., 1990]), or a slow-in, slow-out function (gradually increase the velocity from the start to halfway the transition, then decrease until the end is reached, see [Bowman et al., 1997]). The idea behind these advanced velocity functions is the velocity is small when the viewpoint is at a start or end position of a transition, when it is the most important to have a proper view on a video canvas. Figure 2.1 illustrates an example of a smooth view transition.
2. View Transitions

Figure 2.1: An illustration of a smooth view transition from camera A to camera B. The viewpoint is represented by a view frustum and a gray video canvas and is shown at $t = 0$ (at camera A), $t = 0.5$ and $t = 1$ (at camera B). Its position is denoted by $p$, and its orientation is depicted by the vector $r$. The transition path is represented by a blue arrow.

2.1 Video Canvas Blending

Each view transition starts with a full view on the video belonging to camera A and ends with a full view on the video belonging to camera B. To obtain a smooth transition of the visual information, the videos have to be blended into each other by adjusting the opacity of their canvases. To accomplish a blending that keeps distortion low while not discarding too much video content, the canvases must be carefully positioned and a balance of the canvas opacities must be found.

2.1.1 Canvas Positioning

To maintain the same view on a video while viewing it from a fixed camera position, its canvas has to be scaled to reach the bounds of the camera’s view frustum when the canvas’ distance to the camera is changed. Figure 2.2 shows a canvas that is placed on different distances from its camera and is scaled accordingly. We can specify a default focal distance for each camera; the distance from the camera where the objects of interest are located or the region where most activity takes place. This focal distance acts as the default position of a video canvas.

Figure 2.2: Positioning of a video canvas at several distances from its camera. Its size is scaled to reach the bounds of the view frustum, resulting in a view that exactly fills the screen.
2.1.2 Canvas Blending

At the beginning of a transition from camera A to camera B \((t = 0)\), the opacity of canvas A has to be 1 (fully opaque) and the opacity of canvas B has to be 0 (fully transparent). At the end of the transition \((t = 1)\), canvas A should have opacity 0 and canvas B should have opacity 1. The canvas opacities during a transition can be controlled by a blending function.

Any function that goes smoothly from 0 to 1 can be used, but some functions are especially suitable. Functions 2.1 and 2.2 can be found in [de Haan et al., 2009] as well. These functions have the properties that they are gradually increasing at the start and end of the transition and relatively low around \(t = 0.5\). The effect is high visibility of the canvases when the distortion caused by a large viewing angle is low, and low visibility when the distortion is large. These low opacities halfway a transition make the canvases blend smoother into each other and distortion less visible. A plot of these blending functions can be seen in Figure 2.3.

\[
f_A(t) = 1 - \sin^2\left(\frac{1}{2}\left((1 - t)^2 + 1\right)\pi\right) \tag{2.1}
\]

\[
f_B(t) = 1 - \sin^2\left(\frac{1}{2}(t^2 + 1)\pi\right) \tag{2.2}
\]

![Canvas Blending Functions](image)

**Figure 2.3:** Plot of canvas blending functions 2.1 and 2.2 over normalized time \(t\). Using these functions, the opacity of a canvas will be high when it is viewed from the “right” angle and much lower when viewed from different angles, when there is more distortion with the background scene.

2.2 Transition types

The main motivation behind using smooth view transitions is to help the user maintain spatial orientation. It is therefore important that the changing view that follows from a view transition is behaving in a predictable manner. If we classify view transitions based on their visual effect, we can distinguish three view transition types:

**zooming** viewpoint changes result in a widening or narrowing of the view, zooming in or out on the environment
**2. View Transitions**

**panoramic movement** viewpoint translations and rotations that result in a horizontal and/or vertical shift, of the view, scrolling through the environment to obtain a panoramic view. Linear interpolation often results in a *sweep* through the environment that is difficult to follow, due to the close distance of the viewpoint to the video canvases.

**orbiting** moving around an object that remains visible during the transition, linear interpolation in world space often results in very non-linear translation of objects in image space, which makes it hard to keep track of the object of interest and which can be uneasy on the eyes.

Figure 2.4 illustrates these three transition types. Going from camera A to camera D, we first make a zooming transition to camera B, then an orbit transition to camera C and finally a panoramic transition to camera D. The changing view that results from each transition is shown in Figure 2.5.

---

**Figure 2.4:** Top view of a camera setup that illustrates the three transition that we classified based on their visual effects. The top view surrounded by images that correspond with the four cameras. The red, green and blue arrows represent typical orbiting, panoramic and zooming transitions respectively.

---

**Figure 2.5:** Illustration showing the changing view that is the result of each of the view transition classes.
2.2.1 Zooming

As zooming only involves a small amount of rotation and translation in image space, linear interpolation of the viewpoint parameters, as described in Transition mechanism 1, is sufficient. Figure 2.6a illustrates this transition. Both canvases are placed at approximately the same position in the scene, which is at the focal point of the foremost camera. This way, the videos will smoothly blend into each other.

1. Compute the field of view by linear interpolating between $fov_A$ and $fov_B$:
   \[
   fov(t) = \text{lerp}(fov_A, fov_B; t)
   \]

2. Compute the orientation $r(t)$ of the viewpoint by linear interpolating between $r_A$ and $r_B$:
   \[
   r(t) = \text{Slerp}(r_A, r_B; t)
   \]

3. Compute the position by linear interpolating between $p_A$ and $p_B$:
   \[
   fov(t) = \text{lerp}(p_A, p_B; t)
   \]

---

**Transition mechanism 1**: Step by step description of the zooming transition function. The bottom figure illustrates the transition steps.

---

2.2.2 Panoramic movement

If we consider panoramic movement as “scrolling” through the scene, we can treat it like a 3D scrolling problem. If we compare this to 2D scrolling, for instance through a text document, regular scrolling (comparable to Transition mechanism 1) results in a rapid visual flow which can annoy and disorient the user. [Igarashi and Hinckley, 2000] introduces the concept of *speed-dependent automatic zooming*: by zooming out when the scrolling speed increases and zooming in when scrolling speed decreases, the perceived scrolling velocity on the screen remains constant. This concept is refined in [van Wijk and Nuij, 2003, 2004] in a search for an optimal viewpoint trajectory to keep the perceived speed constant, while [Tan et al., 2001] applies the concept to 3D navigation with a technique they call *speed-coupled flying*, coupling the height and tilt of the camera to the speed the camera is moving.

Speed-dependent zooming can be implemented in slow-in, slow-out panoramic movement by
2. View Transitions

(a) A zooming transition on the top view. The canvases are both positioned at the focal point of the foremost camera.

(b) An illustration of the zooming transition path, using Transition mechanism 1.

Figure 2.6: Illustrations of the zooming transition path and canvas blending.

incorporating a curved viewpoint trajectory, causing the camera to move backward during slow in and move forward during slow out. Bézier curves 1 are suitable to define such a curved path, because they can be parameterized by $t$ and a set of control points, which can be easily computed from the position and orientation of camera A and B, as we can see in Transition mechanism 2 which is illustrated in Figure 2.7b. The position of the viewpoint at a certain time is then determined by computing $B(t)$, the position of a point on the curve at time $t$. The control points influence the curvature of the trajectory, which allows for tweaking to obtain an optimal path. During a panoramic view transition, canvases are positioned at the predefined focal distance of the camera. Since objects of interest are located at this distance, such a canvas positioning will reduce mismatching of these objects with their real position, when the video is viewed from different angles during the transition.

(a) An illustration of panoramic viewpoint movement seen from the top view. The canvases are placed at the default focal distances of the cameras to reduce distortion in important areas.

(b) Speed-dependent zooming applied to panoramic viewpoint movement.

Figure 2.7: Illustrations of the panoramic transition path and canvas blending.

1See Appendix A for a definition of bézier curves.
1. Compute the field of view by linear interpolating between $fov_A$ and $fov_B$:

$$fov(t) = \text{lerp}(fov_A, fov_B; t)$$

2. Compute the orientation $r(t)$ of the viewpoint by linear interpolating between $r_A$ and $r_B$:

$$r(t) = \text{Slerp}(r_A, r_B; t)$$

3. The position $p(t)$ of the viewpoint is the point at $t$ on the bézier curve specified by the control points $P_A$, $C_A$, $C_B$ and $P_B$, where $C_A = -a r_A$ and $C_B = -a r_B$. The value of $a$ determines the curvature of the transition path, thus the amount the viewpoint moves back and forth. A value close to 0 results in a straight line, larger values result in more curvature.

$$p(t) = B(t)$$

**Transition mechanism 2:** Step by step description of the panoramic transition function. The bottom figure illustrates the transition steps.

### 2.2.3 Orbiting

Consider the case where an object, which we will call *point of interest* (POI) is visible in two cameras. As the focus will be on the POI, the canvases are positioned such that they intersect at the POI. Two problems are likely to arise when we would apply Transition mechanism 1 for a transition between the cameras:

1. The difference to the video canvases can vary much, resulting in visible content that increases and decreases and flying through the canvases when the cameras are opposing each other.

2. Although motion is linear in world space, it can be far from a straight line in image space, as illustrated in Figure 2.8a.

As a result, the object of interest can move in an unpredictable manner across the screen, making it difficult to follow. A solution can be found in stabilizing the view change, by interpolating the viewpoint position through image space instead of world space. The resulting viewpoint trajectory is a curved path around the POI, which is very similar to orbital motion in physics.
2. View Transitions

(a) Linear interpolation of the viewing parameters in world space can result in a curved movement of the POI on the screen, which is difficult to keep track of.

(b) The POI follows a straight line on the screen as a result of linear interpolation of its position in image space.

Figure 2.8: Two samples from frame sequences taken from a view transition, to compare the visual effect of interpolation in world space (a) to POI interpolation in image space (b).

The solution is an extension of orienting POI movement, described in [Mackinlay et al., 1990] and is described and illustrated in Transition mechanism 3 and Figure 2.9 respectively. The result is a smooth change of the viewpoint-to-POI distance and a straight path of the POI on the screen (see Figure 2.8b). By positioning both video canvases such that they intersect with the point of interest, the least distortion of the video with the underlying scene will occur in the region around the POI.

When no POI is specified, but we want to make a transition between two cameras that record much of the same area, a POI position can be computed in order to make Transition mechanism 3 applicable. The center of the common perpendicular (see §A.2) of the cameras’ optical axes is, the shortest possible line segment between these axes is likely to be visible in both cameras and would therefore be a suitable choice.

(a) Orbiting illustrated in the top view. Both canvases are positioned at the point of interest to reduce distortion in this region.

(b) An illustration of the zooming transition path, using Transition mechanism 3. The point of interest is depicted as a yellow cylinder.

Figure 2.9: Illustrations of the orbit transition path and canvas blending.

While orbiting around an object when the angle between the two cameras is larger than 90°, and using canvas blending functions based on the normalized time, the back of canvas A would become visible before \( t = 1 \) and the back of canvas B would be visible at \( t = 0 \). If the back
1. Compute the field of view by linear interpolating between $fov_A$ and $fov_B$:
   $$fov(t) = \text{lerp}(fov_A, fov_B; t)$$

2. Compute the image coordinates $s_A$ and $s_B$ of the POI in camera A and camera B, respectively:
   $$s_A = \text{worldToImage}(p_{POI}; \text{camera}_A)$$
   $$s_B = \text{worldToImage}(p_{POI}; \text{camera}_B)$$
   $$s(t) = \text{lerp}(s_A, s_B; t)$$

3. Compute the distance from the viewpoint to the lookat point, by taking a linear interpolation of $d_A$ and $d_B$:
   $$d(t) = \text{lerp}(d_A, d_B; t)$$

4. Compute the orientation $r(t)$ of the viewpoint by linear interpolating between $r_A$ and $r_B$:
   $$r(t) = \text{Slerp}(r_A, r_B; t)$$

5. Convert the image coordinates to coordinates of the viewpoint’s coordinate system, where $v$ is the vector from the viewpoint to the POI:
   $$p_{POI} = \text{imageToWorld}(s(t); \text{viewpoint})$$
   $$p(t) = p_{POI} - v$$

**Transition mechanism 3:** Step by step description of the orbit transition function. The bottom figure illustrates the transition steps.
of a canvas is visible, the video content has no connection with the underlying scene any more, destroying the effect of being in the videos and the 3D environment at once. To let the canvas opacity smoothly reach 0 at these large angles, \( t \) gets a new value using functions 2.3 and 2.4 (see below) before it is put into the canvas blending function. The effect of this correction step is illustrated in Figure 2.10 where \( t_A \) gets different values in two cases, although \( t \) is the same in both cases.

\[
t_A = \min(1 - t, 1 - \frac{\alpha}{\pi}), \quad \text{where } \alpha \text{ is the angle between the viewpoint and camera A} \quad (2.3)
\]

\[
t_B = \min(t, 1 - \frac{\beta}{\pi}), \quad \text{where } \beta \text{ is the angle between the viewpoint and camera B} \quad (2.4)
\]

\[\begin{align*}
(a) \quad \frac{\alpha}{\pi} &= 0.3; \quad t = 0.5; \\
& \quad t_A = \\
& = \min(1 - 0.3, 1 - 0.5) = 0.5
\end{align*}\]

\[\begin{align*}
(b) \quad \frac{\alpha}{\pi} &= 0.7; \quad t = 0.5; \\
& \quad t_A = \\
& = \min(1 - 0.7, 1 - 0.5) = 0.3
\end{align*}\]

**Figure 2.10:** Example showing that the opacity of a canvas during an orbit transition depends on either the time \( t \) or the angle between the two end viewpoints. Although \( t = 0.5 \) in both cases, the canvas in case (b) gets a smaller opacity since the angle between the viewpoints is larger than 90°.
2.3 View Transition Graphs

Given a surveillance scenario with a certain camera setup, we could determine for each camera pair if a view transition should be possible between these cameras and what kind of transition this should be. These camera pairs combined would then form a directed graph when we represent the cameras as nodes and the view transitions as edges. Such view transition graphs make it possible to determine from a given view, to which other views we can navigate, serving as a “road map” for the navigation tools with which we can control view transitions. Figure 2.11 shows a graphical representation of a view transition graph.

![View Transition Graph](image)

**Figure 2.11:** Example of a view transition graph. Red edges represent orbits, green edges represent panoramic movement an blue and purple edges represent zooming in and zooming out, respectively.

2.4 Summary

Our surveillance interface is based on smooth transitions of the viewpoint between virtual cameras that are positioned before video canvases. The main motivation to use smooth view transitions is to that the continuous change of perspective can help the viewer to maintain spatial awareness. However, it is important that the view changes in a predictable manner, so it can be easily followed. We have therefore classified view transitions into three types, based on their resulting view change: orbit, panoramic and zooming. For each of these transition types, we defined a transition function that describes the interpolation of the viewing parameters over time. These functions are designed with the aim at obtaining predictable view changes. For a given set of cameras, view transition graphs can be composed to define which transitions one can make between a set of cameras. The transition functions, together with the view transition graph, form the basis of our navigation system. the next chapter will discuss concepts that make it possible to interact with this system to initiate view transitions in a natural way.
Chapter 3

Navigation

The system of view transitions between virtual cameras that we described in the previous Chapter provides a basis for the tracking of activity over several cameras. If for instance it is necessary to follow a person that walks across a monitored environment, we would like to navigate to a relevant virtual camera, each time the person walks out of our current view. To be able to do this, we must provide a user interface with which view transitions to a certain virtual camera can be initiated. In this Chapter, we describe such a user interface for navigation.

When a subject’s appearance on camera views can change rapidly, navigation decisions need to be made quickly. Therefore, it should be immediately clear to which directions and by which means the user can navigate from a given video. We aimed at achieving this by augmenting videos with interactive 3D widgets that provide direct feedback about the result of their usage. To select and activate these 3D widgets, only 2D input device such as a mouse is needed.

3.1 Margin Videos

Video thumbnails are the most straightforward of the concepts we will describe in this chapter. The placement of selectable thumbnails of adjacent videos (according to the view transition graph) on a margin around the main view enables the user to directly navigate to a certain video. This is a concept similar to the SMV-player [Girgensohn et al., 2007b]. After the mouse button is pressed while hovering on a thumbnail, a transition to the corresponding camera is initiated and the margin is updated with the thumbnails for the new camera.

The margin can be scaled depending on the surveillance task, its size ranging from showing thumbnails the same size as the main view (for an overview of several cameras) to such a small size that there are no thumbnails at all (for fully ego-centric navigation). The thumbnails are put on the intersection of the margin with the line from the center of the screen to the projection of the camera on the image plane. This way, the position of a thumbnail indicates the direction of the corresponding camera. While this distribution can result in overlapping videos, Algorithm 3.1 spreads these videos until they are not covering each other. This algorithm considers the margin as a circle on which thumbnails have a one-dimensional position. It then checks iteratively if the distance between each group of thumbnails is larger than a threshold value and spreads the thumbnails when necessary.
3. Navigation

1: \( n \leftarrow \) number of thumbnails
2: \( \epsilon \leftarrow \) threshold value
3: for \( i = n \) to 2 do
4: for \( j = 0 \) to \( n \) do
5: \( l \leftarrow \) sub sequence of thumbnails of size \( i \), starting at index \( j \)
6: \( h \leftarrow \) first element of \( l \)
7: \( t \leftarrow \) last element of \( l \)
8: if distance\((h, t)\) \( \leq (i - 1)\epsilon \) then
9: \( m \leftarrow \) center of \( t \) and \( h \)
10: for \( k = 0 \) to \( i \) do
11: \( l[k] \leftarrow m + (k - \frac{i - 1}{2})\epsilon \)
12: end for
13: end if
14: end for
15: end for

Algorithm 3.1: Spreading overlapping thumbnails on the screen margin.

(a) The distribution without using the spreading algorithm.
(b) The distribution using the spreading algorithm.

Figure 3.1: The distribution of video thumbnails on the margin region, before (a) and after (b) applying Algorithm 3.1.
3.2 3D Cursor

Besides using video thumbnails, navigation can be controlled from within the main video by using the augmented 3D widgets. In order to do so, these widgets can be selected with the mouse pointer. The projection of the mouse pointer on the scene (see Figure 3.2) returns a 3D coordinate. This coordinate is indicated with a 3D cursor that can be used to select objects of interest and start a navigation that is focused on this object.

Figure 3.2: Projection of the mouse pointer on the 3D scene by shooting a ray from the viewpoint through the image plane onto the scene. This enables the user to select certain navigation widgets or parts of a video.

3.2.1 Cursor Shape

The simplest cursor shape would be a point, but this gives no indication of its depth in the scene. A 2D shape, such as a circle, gives a better indication as it scales with the distance from the camera and gets rounder when it is viewed from above, but such a cursor can still be misleading when it is placed on an object in the video that is not modeled in the scene. Therefore, a full 3D cursor would give the best indication of its position in the scene. To prevent the cursor from covering a large part of the video, it is important that the cursor is shown as a transparent object or wire frame outline of a 3D shape.

Meeting these requirements, we created a 3D cursor that consists of the outlines of a cylinder that would fit around the average person (1 meter in width, 2 meters in height). To create the illusion of a transparent cylinder, the cylinder is always rotated towards the viewer so the vertical lines always align with the contours of the cylinder. The cursor is only visible when the mouse is pointing at a specified floor part and not at a selectable interaction tool. By specifying certain floor parts, we can restrict the cursor to possible parts where persons can walk. It would for instance not be very useful and sometimes misleading to make all the walls and ceilings selectable by the cursor as well.

3.2.2 Cursor Ghost

During and after a view transition, it can be difficult to keep track of the part of the video that was selected with the 3D cursor. A “ghost” copy is therefore left behind at the position of the cursor before the transition. Combined with the transition mechanisms described in Chapter, the ghost copy serves as a focus guidance that moves in a predictable manner. After a transition, the cursor can be used again while the ghost is fading away. To determine the new angle from which
the POI is seen, a cross is added at the bottom of the cursor that represents its two horizontal axes, indicating its orientation.

A disadvantage of such a static ghost is the temporal gap that is mentioned in §2. The person or object that is followed is likely to have moved during a transition, appearing at a different place than the cursor ghost. This could be partly solved by involving a prediction based on the direction of the menu selection or other gestural movements but an exact prediction cannot be made without perfect automatic tracking.

3.3 In-Video Navigation

The addition of a mechanism to select certain areas or objects in the video, we can extend the 3D scene with navigation widgets. This enables navigation directly from within the videos, allowing the user to keep his focus right at the main video view. The widgets in our system are selectable arrow shaped glyphs, each representing a view transition to one of the neighboring cameras. The arrow points in the direction of the view change that a transition to the connected camera would result in. Arrow shapes are suitable for this, because their shape, size and other properties can be used to illustrate the motion path and the direction and velocity of moving objects. Nienhaus and Dollner, 2003 describes the use of glyphs to depict dynamics in illustrations of animations.

Figure 3.3: Video overlayed with two glyphs that can be selected to initiate a transition to a neighboring camera.

3.3.1 Glyph menu

By connecting glyphs to the 3D cursor, instant navigation from the selected point becomes possible, which allows for a more rapid camera selection than having to move the mouse pointer to one of the glyphs first. The connections of the glyphs at the cursor position form together a menu. Selection of one of its items is similar to selection in a pie menu: When the mouse button is pressed, connections to the currently available glyphs are made. These connections together form a menu around the selection point. The mouse pointer can now be dragged around the cylinder to select one of the glyphs from the cursor. The release of the mouse button initiates a transition to the selected camera. Figure 3.4 and Figure 3.5 illustrate the usage of the glyph menu.
3.3.2 Orbit Glyphs

Directed glyphs can be used to indicate the sideways or back and forth view changes that we see in panoramic and zooming transitions. But we need something else for orbit transitions since these transitions do not result in a view change in a certain direction, but rather a rotation. We chose to indicate an orbit transition to a certain camera by connecting the center of the glyph menu directly to the camera position. An orbit glyph consists of a dashed line from the top of the cursor to the relevant camera and a polygon from the bottom of the cursor to the projection of the camera position on the same vertical plane as the cursor bottom. An example can be seen in Figure 3.4.

Figure 3.4: Screen shots showing the 3D cursor before and after a mouse click.

Figure 3.5: A top view of the 3D cursor showing the menu responding to mouse events.
3.4 Camera Selection Feedback

When we can zoom in on one or more regions of the video, it is not exactly clear from the 3D tools which part of the video can be expected to be visible after the transition. We could intersect the view frustum of the corresponding camera with the model and project it on the image plane, but this projection would not be correct if many objects that are visible in the video are not included in the model. A simpler approach is to take a rectangular slice of the view frustum. The question then is at which distance from the camera this should be taken. A better estimate about the content that a camera sees can be made if the shape of its view frustum is shown, so that the position, orientation and visible area of the camera become clear. This is realized by the outlines of planes that intersect the view frustum at a number of distances till the predefined focal distance of the camera, see Figure 3.6.

![Figure 3.6: View frustum bounds appearing when the corresponding thumbnail is selected.](image)

3.5 The Navigation Tools as an Integrated Set

The navigation tools can be used interchangeably, but they are connected to each other to exploit the power of the visual hints the various tools can give. Figure 3.7 shows these connections: if a glyph or a thumbnail is selected, the other tools related to the same camera are highlighted as well.

![Figure 3.7: Overview of the connections between navigation tools. When a glyph or a thumbnail is selected, the other tools related to the same camera are highlighted as well.](image)
During each of the two navigation states (during a transition or fixed at a camera), the sensitivity and visibility of the interaction tools will change, depending on the state and the position in the camera graph. Each state change results in changing behavior of the interaction tools, as described in the following processes and in Figure 3.8.

During one camera view:

- Obtain the neighboring cameras from the transition graph.
- Show the glyphs that go from the current camera to one of the neighbors, hide the other ones.
- Show and position a video thumbnail for each neighbor.
- Show and enable the 3D cursor.
- During 3D cursor movement: gray out a video thumbnail when the cursor position is not visible on that video, otherwise show the cursor position on the video, indicated with a cross.

During a transition:

- Hide the 3D cursor, leave a ghost copy behind.
- Hide the margin videos.

3.6 Summary

In this chapter we have described a set of navigation tools. The tools can be used to directly interact with the view transition system, so the focus can always stay on the actual videos. Margin videos enable direct navigation to a neighboring camera, while glyphs can be used for navigation from within the video. Glyphs can be selected directly or reached from the 3D cursor, by using the glyph menu. Camera selection feedback is given by highlighting the widgets that correspond to the selected camera and by showing an indication of its view frustum.
Figure 3.8: Overview of the interaction tools when the system state is subsequently: fixed at a camera, when no glyph is selected (a) or when one glyph is selected (b); during a transition to the selected camera (c); right after the transition (d).
Chapter 4

System Architecture and Implementation

For the evaluation of the camera transitions and interaction tools described in the previous chapters, a prototype was developed. It consists of a viewer from which videos are viewed and interaction tools can be controlled and a graphical user interface (GUI) to load scenarios, control video playback and toggle or adjust transitions and interaction tools. This chapter describes the design and implementation of this prototype.

4.1 System Architecture

Our surveillance system is interactive application to view video streams attached to canvases in a 3D model, containing navigation tools that can be controlled by user input. The Model-View-Controller pattern (MVC, [Krasner and Pope 1988]) is chosen as a design pattern, since the three main components of this pattern (model, view and controller) can be directly applied to this type of application. As illustrated in Figure 4.1, the controllers process mouse events and any other input device that is connected to the application and, depending on the event, makes changes to the model. The model contains the environment model, camera data, video canvases and interaction tools, as well as a virtual camera that controls the viewpoint. The view determines which parts of the model are viewed on top of each other and is directly linked to the parameters of the virtual camera.

Figure 4.1: Diagram that shows how the Model-view-controller design pattern is applied to our surveillance system.
4. System Architecture and Implementation

4.1.1 Controllers

All user input is processed by controllers that modify parts of the model or trigger animations. As illustrated in Figure 4.2, all input from hardware devices — mainly mouse events — is caught by the Input Controller. This delegates the events to one of the specific controllers, depending on the type of event and on which part of the view the event took place. The specific controller then sends a message to a part of the model.

![Diagram depicting the flow of messages that are passed over the various components of the application after user input. Depending on the input, one of the controllers processes the input and sends messages to parts of the model.](image)

**Figure 4.2:** Diagram depicting the flow of messages that are passed over the various components of the application after user input. Depending on the input, one of the controllers processes the input and sends messages to parts of the model.

**Video Controller**

The Video Controller controls the videos that are played on the canvases. These videos can be real time streams from cameras, as well as recordings on storage media for investigation afterwards. Several graphical user interfaces (GUIs) and input devices can connect to the Video Controller, by adjusting the playback speed. The Video Controller updates the connected GUI according to the frame number and playback speed, as long as the GUI implements the update methods from the GUI base class.

**Interaction Controller**

When the viewer receives mouse events, they are passed to the Interaction Controller, which determines which part of the view is selected by intersecting a ray from the viewpoint through the mouse pointer on the image plane with the scene. Depending on the selected part, a part of the model is modified. On mouse movements, the following modifications are made in the model, depending on the selected region in the view.

**floor parts:** Show the 3D cursor and position it on the intersection point, update the cursor position in the video thumbnails on the border. In other cases hide the 3D cursor.
interaction tools: Store the camera related to the selected interaction tool in memory, highlight the connected interaction tools. The stored camera can then be accessed by the transition controller (see next paragraph).

Transition Controller

When the mouse button is pressed, a transition to the selected camera is initiated by passing it to the transition controller. If no camera is stored in memory, nothing happens. On the initiation of a transition, a timed loop starts running, interpolating the viewpoint’s position, rotation and field of view at each step. The duration of the loop is determined by a metric, depending on the transition type. The metrics are aimed at keeping the perceived velocity of visible information constant:

- **Zoom** The view narrows or expands by approximately the same amount in most cases, so a constant duration suffices.
- **Panoramic** The extent of motion of the view depends on the distance between the focal points of the cameras.
- **Orbit** The perceived change of the view depends mainly on the angle between the cameras’ optical axes.

A multiplier can be adjusted in each case to change the duration per transition type. As mentioned in Chapter 2, transitions with longer durations result in a “temporal gap”, but better spatial awareness, whereas transitions that are (almost) instant result in a “spatial gap”.

4.1.2 Model

The model is a conceptual representation of a recorded environment, cameras, video canvases and added interaction tools, which are stored in a scene graph. We can see in Figure 4.1 that the model is constructed from a data set. This data set consists of three components:

- an environment model, stored in an OpenScenegraph \(^1\) model file
- a description of the cameras, stored in an XML file (see \(\text{§B.1}\) for an example)
- a set of videos, stored as video files or image sequences, or as URLs to live streams from cameras

A work flow is developed to create each of these components from within the Blender 3D modeler \(^2\), by using its Python \(^3\) scripting facilities:

1. Create a model of the monitored site. This can either be an existing model that is imported in Blender, or a model that is made from scratch.

---

\(^1\)http://www.openscenegraph.org
\(^2\)http://blender.org
\(^3\)http://python.org
2. Add and calibrate each of the available cameras as a virtual camera to the model. Calibration can either be done by manually adjusting each camera’s position, orientation and field of view, but algorithms such as the one described in [Tsai, 1987] exist that can auto-calibrate the cameras fairly well for some scenarios.

3. Add a group “Floors”, and assign objects to it over which the cursor can slide.

4. Add a group “Animation” and assign animated objects to it that should not appear in the application model, but only in video animations. This group includes all objects that don’t have fixed positions, such as persons or vehicles.

5. Add glyph objects. A glyph is represented by a rectangular cuboid. The desired glyph size can be controlled by adjusting the length of the cuboid. The default length is 1.0 units.

6. Run the export scripts to produce a dataset that can be loaded into the application. This dataset consists of a model file, an XML specification of the camera setup (see §B.1) and eventually a number of animations that are rendered from the virtual cameras.

Optional:

7. Animate some of the objects, for instance persons or cars.

8. Run the export script to render a video from each of the cameras.

The optional steps can be taken to create artificial data sets, which we used to create scenarios with several different camera setups for our experiments (see Chapter 5). The possibility to create artificial scenarios can take away the need to set up a real environment with cameras.

4.1.3 View

The view on the scene graph containing the model is established by the OpenSceneGraph render loop, from the viewpoint, which is linked to the controllable camera in the model. It is important that certain objects are always on top of certain other objects, no matter what their differences to the viewpoint are. For instance, videos should always be on top of the model and interaction tools should come on top of video canvases. To ensure this rendering order, all scene graph objects are classified in render bins, as shown in Figure 4.3.

4.2 Implementation

The prototype is built upon the OpenSceneGraph graphics toolkit and the VRMeer virtual reality library. It is entirely written in the interpreted language Python, using the osgSWIG wrappers. With this combination, objects could be added or modified in a straightforward manner while the application was running.

Deriving from OSG classes, additional classes in the geometry package provide the basic functionality to adjust the color of objects and show, hide, select or deselect them, to create new interactive elements easily. The parent Transform class contains methods to determine its position and orientation in the coordinate system of other Transform objects, to determine visibility of objects in cameras and to keep the the object always rotated towards the viewer (which is used for the cursor and the cursor ghost).

---

4http://code.google.com/p/osgswig
Figure 4.3: The render order of the various scene graph objects, which is, from bottom to top: model, video canvases, interaction tools, border with thumbnails.

Figure 4.4: Class diagram that shows the connections between the important classes.
Although the use of Python and SWIG wrappers add overhead to the system performance, we managed to let the application run at interactive speed (>15 FPS) while using our test scenarios (see §5.1). The application could be optimized by rewriting it in a compiled language such as C++ and by using advanced libraries to display video textures, which are included in the latest OpenSceneGraph versions. But the focus for this prototype was to make it just usable at interactive speed.

4.3 Summary

We implemented the view transition techniques and navigation concepts, as described in Chapter 2 and Chapter 3, in a prototype application. The Model-View-Controller pattern, together with high level graphics toolkits allow us to rapidly add new navigation widgets or modify existing ones. The next chapter describes the usage of this prototype in an experiment for the evaluation of our navigation concepts.
To get more insight and feedback on the usability and effectiveness of the view transition and navigation concepts that are described so far, we conducted a first evaluation study. This evaluation consisted of experiments that involved performing typical surveillance tasks using the prototype described in Chapter 4. During the experiments, participants had to track a person as it moves across several videos. Such a task is an obvious choice, since the main motivation behind developing our spatial navigation interface was to improve the effectiveness of tracking activity between surveillance cameras. For these experiments, we created a synthetic surveillance scenario in which we had the freedom to completely specify the camera setup and generate suitable video content. Control over the camera setup allowed us to study the influence of the camera setup on the usage of navigation tools. Before creating these synthetic scenarios, we studied a number of existing surveillance datasets. The following sections further describe the scenarios we studied and created, the experiments we conducted and the results of these experiments.

5.1 Surveillance Scenarios

We have created several scenarios that can be explored by the prototype. Some of these scenarios are based on real surveillance videos. But since surveillance data that is relevant for our research is not widely available and too restricted in terms of video content and camera setups, we created artificial scenarios as well. This section summarizes the scenarios that we have created along with their relevant properties.

5.1.1 Office Hallway

The office scenario consist of videos from six different cameras. The environment model is created out of CAD drawings. The virtual cameras are manually calibrated to let the videos match with the model. The camera setup features all three transition types that are defined in Chapter 2, which makes it a suitable data set to test the different transitions of the prototype. Although, the data set is not suitable to compare our navigation concepts with a traditional matrix view, due to the simplicity of the model, which consists of only one main room and two corridors and the small number of persons at a time appearing in the videos. Tracking persons in the video on a matrix view appeared to be an easy task with the office scenario. See Figure 5.1 for an impression of the office scenario.
5. Evaluation

**Setting:** Office

**Activity:** small number of persons passing by or behaving suspicious

**Transitions:** orbit / panoramic / zoom

![A screenshot of the office scenario loaded in the prototype.](image1) ![A map view of the office scenario.](image2)

Figure 5.1: The Office scenario. This scenario contains a view transition graph featuring all our three transition types.

5.1.2 PETS datasets

The Computational Vision Group of the University of Reading \(^1\) has made several datasets that are used as benchmark data in their workshops publicly available. We have created scenarios out of three of these datasets. Figure 5.2 gives an overview of these scenarios.

The scenarios have in common that the cameras are focused on the same area in the scene so orbiting is the most appropriate transition between them. The presence of many different people and events to keep track of makes the sets interesting to evaluate the effectiveness of the 3D cursor.

**Setting:** train station, airport, road crossing

**Activity:** events in crowds (loitering, unattended luggage)

**Transitions:** orbit

As it is hard to distinguish people from each other in such a crowded area, and people look different from other angles, the 3D cursor and cursor ghost shows its use to keep the focus on certain people while switching between cameras.

\(^1\)http://www.cvg.rdg.ac.uk/
Surveillance Scenarios

(a) The PETS 2006 scenarios take place in a large hall of a railway station. Four cameras monitor roughly the same area from different angles and distances. The purpose of these scenarios is to keep track of persons who leave luggage unattended.

(b) Map view of the PETS 2006 scenario.

(c) The videos of the PETS 2007 workshop are shot on a crowded airport hall, seen from four cameras. Here the task is to track events such as loitering, attended luggage removal and unattended luggage.

(d) Map view of the PETS 2007 scenario.

(e) For the PETS 2009 data sets, scenarios on a road crossing are recorded with eight cameras. The task is to track individuals within a crowd or certain crowd events.

(f) Map view of the PETS 2009 scenario.

Figure 5.2: Overview of the PETS scenarios. These scenarios are set in public places. Each of their camera setups consists of cameras that are focused on the same area in the scene.
5. Evaluation

5.1.3 Highway Traffic

Camera setups used to monitor traffic events differ from the cases mentioned above in that they consist mostly of sequential cameras that are placed at large distances from each other along a highway road. We created a synthetic scenario that incorporates a piece of a highway and animated some car models in it. The resulting dataset could then be used to smoothen out the zooming transitions.

Setting: Highway

Activity: cars passing by

Transitions: zoom

![Image](a) A screenshot of the traffic scenario loaded in the prototype.  
(b) A map view of the traffic scenario.

Figure 5.3: The traffic scenario, consisting of a highway part with animated cars. Navigation between cameras can be done by zooming transitions.

5.1.4 Evaluation Environment

The surveillance scenarios that we studied were useful for informal experiments to iteratively design our viewpoint transitions and navigation concepts. However, none of these scenarios are suitable for a more structured evaluation. First, we would need the possibility to design our own camera setups to be able to study usability of navigation tools in a specific setup. Second, control over the video content is needed to influence the difficulty of a task. When it is either trivial or too difficult to complete a task. This difficulty can be influenced by parameters such as the number of animated persons, their appearance relative to each other and how closely they walk to each other.

Therefore, we have created synthetic scenarios for which we could completely specify the camera setup and person animations (see Figure 5.4). Each of these scenarios contains a simple model, to which simplified person models (or actors) were added that are animated along a path to simulate a walk across the scene. Each scenario model contains a camera setup of four cameras, connected with one of the three transition types. The actors are animated such that they walk in an unpredictable manner across the scene, which is important for our tracking task, as described in the next section.
5.2 Experiment Description

The main goal of the experiments was to gather qualitative feedback on the usability of our surveillance interface in specific scenarios. We therefore let each participant perform a simplified surveillance tracking task. Such a task involved tracking one specific animated actor as he walks across an office scene. This actor has to followed by the participant until he leaves through a labeled door. The identification marker of the door should then be written down by the participant. The actors are made identical to ensure that the actor cannot simply be spotted or re-tracked by visual recognition.

As one of the purposes of the evaluation is to investigate the effect of view transitions on the usability of our interface, we created a scenario for each of our three view transition types (“orbit”, “panoramic” and “zoom”). Maps of these scenarios are given in Figure 5.4b. The scenarios were kept very basic to avoid uncontrollable effects that a complex scene can introduce. For each of the scenarios, a number of synthetic videos were prepared which show a number of animated “actors” that walk across the scene, as seen from the cameras.

There were 12 people who participated in the experiment. Each participant performed a tracking task with two interfaces. The first interface showed the videos in a traditional matrix lay-out. The motivation behind the first task is to get insight in which scenarios are difficult in general, independent of the surveillance interface. For the first task, our spatial navigation system was used. The difference between the two interfaces is shown in Figure 5.5. Before performing a task, the participant could get used to the task environment by practicing with one of the PETS scenarios or the “Office” scenario, depending on the transition type of the task setup. For the first task, this involved looking at a set of videos while trying to keep track of certain individuals. Before performing the first task, the participant was given a map of the scenario. For the second task this meant getting used to each navigation concept that is also used in the task (for instance, practicing with glyphs is not necessary before performing the “orbit” task, since glyphs are not included in this scenario). Each tracking task took about two minutes to complete, a complete experimental session took about half an hour. The scenarios were divided over the all participants such that every scenario was used an equal number of times. As a result, each scenario was performed four times per task. The participant got two chances, so he can start over again when he thinks that he has lost the subject.

While performing a task, the participant were asked to provide direct feedback by thinking-aloud. This feedback was recorded with a microphone. Simultaneously, the task environment is

Figure 5.4: Impression and overview evaluation scenarios. These scenarios are based on synthetic videos and are used for the experiments.
5. Evaluation

recorded, as well as participant’s actions. These recordings were combined in one video that could
be analyzed afterwards. After finishing the tasks, the participants had to fill out a questionnaire
(see §C.2).

Figure 5.5: Example of the evaluation environment showing the “orbit” scenario in the two interfaces
that were used for the experiment.

The experimental setup (as can be seen in Figure 5.6) consists of a computer with two screens
attached, one for the participant and one for the experiment leader. The participant’s screen
shows a full view of the task environment, whereas the other screen contains the tools to control
the recordings. Since all controls can be reached from the keyboard, the mouse is available for the
participant, which he will need for the second task. A detailed description of the experiment can
be found in §C.1.

Figure 5.6: Picture of the experiment setup, showing a) a participant; b) the experiment leader; c) the
task screen; d) the control screen; e) a mouse to perform the 3D navigation task; f) a
keyboard to control the experiment; and g) a microphone for recording.
5.3 Results

Looking at the results of the experiment, along with the recordings and questionnaire answers (see §C.3 and §C.4), we have seen some noticeable differences between the different scenarios and surveillance interface (“matrix” vs. “spatial”). For each scenario, we will summarize these results and give possible explanations.

5.3.1 Usage of Navigation Tools

The questionnaire started with the question how useful each navigation tool was found and why (see §C.2). Figure 5.7 shows the results to this question combined in a histogram. This histogram shows that glyphs were found the most useful. The main reason for this was that the direction of the view transition is immediately clear from the arrow shape so it felt natural to navigate through the environment by using glyphs. The 3D cursor was found useful as well, but was only used actively in the orbit scenario, where it was a crucial tool for not confusing different persons. In the other scenarios, the 3D cursor was mainly used passively to mark a person and compare it with its instances on neighboring video thumbnails. Video thumbnails were mostly used to check if a person appears on one of the other videos than the main video, some participants used the cross indication. View frustum bounds were found the least useful and sometimes were found confusing in which direction they were pointing or misleading in how they occluded with the videos and the underlying 3D model.

![Figure 5.7: Histogram combining navigation tool preferences.](image)

5.3.2 Task Results

The orbit scenario appeared to pose the most difficulties on the tracking task in both interfaces. For the matrix interface, this was due to the fact that camera angles are opposite and spatial reasoning is difficult. In real images one would resort to “matching” appearance and image features amongst multiple cameras, but as the scene was symmetric and all actors were identical, they
could not be directly compared by appearance alone. One participant succeeded by constructing a mental model of the environment by comparing door numbers with the ones on the map. In the spatial interface the 3D cursor was found very helpful, especially when combined with the dynamic feedback on which transitions were available. However, difficulties still originated from the requirement to make quick decisions choosing one of the orbit glyphs while actively tracking the subject with the 3D cursor. If one of these is done wrong or concentration is lost for a moment, the subject is lost very easily. Participants sometimes experienced difficulties in matching the 3D cursor ghost with the correct actor in cases where actors walk closely together.

In contrast to the orbit scenario, participants in the panoramic scenario performed quite well with both interfaces. In this simple configuration with 4 cameras, the matrix interface is very straightforward and motion predictions were easy. In both interfaces, some difficulties arise when two persons leave the video at the same moment in the same area (see Figure 5.8) or when people walk in a dead area (see Figure 5.9). For the spatial interface, timing proved sometimes difficult when transitions were slow. Thumbnails could be used to determine the right moment for a transition but most people are focused on the main view too much. Most participants preferred simple clicking on arrow glyphs to navigate, and avoided the use of thumbnails and drag-selection with the 3D cursor.

![Figure 5.8: Confusion: before a transition, person A is right of person B, while after the transition, person A is left of person B in the next view (left).](image)

![Figure 5.9: Illustration of two cameras and their mutual coverage, introducing a dead area and a shared area.](image)

In the zooming scenario we observed remarkable improvements in the spatial interface. For this scenario, most participants failed to keep track of the subject in the matrix interface. This was mainly caused by the difficulty to verify which persons are the same ones on multiple videos. When attention has shifted from one video to another, it can happen that the subject has not appeared on that video yet. At that moment, it is already too late to regain focus on the first video, which means that the subject is lost. In the spatial interface, the participant could navigate
from within the same view. As the videos smoothly merged into each other, the attention could always be on the subject if the right moment for a transition is selected. Some participants achieved this by following the subject with the 3D cursor until the cross-hair indicator cursor appeared on the thumbnail of the next video.

### 5.4 Discussion

From the task performance results, we see that the participants were able to track a subject as well as, or sometimes better with our spatial interface than with the matrix interface. Furthermore, we have seen from our recordings and questionnaire answers that the navigation tools helped the participants to keep track of the subject and maintain a mental model of the spatial context. The limited amount of training the participants received had a large influence on the experiment. Studying the recordings, it often occurred that the participant did not respond quickly enough to keep track of the subject. Future experiments should therefore incorporate more extensive training sessions to assure that the participants are accustomed enough to both the navigation tools and the actual task scenario environments.

The orbit scenario turned out to be very difficult for the participants to perform correctly. Some of this could be attributed to some misleading aspects of, or inconsistencies between some navigation tools, as reported by the participants in their feedback. Besides, from the task results on the matrix interface we can conclude that the orbit scenario is difficult by itself, because actors on the video were easily confused with each other.

While arrow glyphs point in the same direction as their corresponding view transition, orbit glyphs point in the opposite direction of their view transition (i.e. in the direction of their corresponding camera). This makes behavior of the orbit glyphs exactly the opposite to the arrow glyphs, since the user has to make a gesture in the opposite direction of where he wants his view to change to. To resolve this issue, the orbit glyphs could be redesigned to make the orbit transitions and corresponding camera choosing less difficult. Figure 5.10 shows a possible redesign, in which the orbit glyphs have arrow shapes and are pointing in the direction of the corresponding view change.

![Figure 5.10: Possible redesign of the orbit glyphs making them more consistent with arrow glyphs. The orbit glyph is shown in red.](image)

(a) The current situation: the orbit glyph points in the direction of its corresponding camera.

(b) A possible redesign: the orbit glyph points in the viewing direction of the camera.
5. Evaluation

Besides the orbit glyphs, the view frustum indications could be improved as well. Especially in the zooming scenario, it is not exactly clear which area of the video is visible in other cameras, since the view frustum bounds are fully rendered on top of the video canvas without intersecting the underlying model. An indication of the intersection of a camera’s view frustum with the model would therefore be an improvement.

The camera setups that we used for this experiment were kept deliberately small, e.g. only 4 cameras, and focused on a single view transition. Such setups are not very realistic, since real configurations are usually much larger and incorporate various camera relations to ensure sufficient camera coverage. Therefore, a more extensive evaluation on large complex surveillance scenarios would enable more realistic comparison between our interface and a matrix interface, as we expect that the real advantage of our interface will become visible in such scenarios.

5.5 Summary

As a first structured evaluation, we have conducted a user study. This user study was focused on getting insight in the usage of our navigation interface in specific scenarios that were based on our view transitions. During the experiments, participants had to perform a surveillance tracking task using our interface and was asked to provide feedback. From the user feedback and task results, we have identified a number of difficulties, some of which possibly could solved by further refinement or redesign of the navigation tool set.
This chapter summarizes the material that is described in the previous chapters, to see if our initial research questions have been resolved. In addition, suggestions for possible directions of future research in the subject of spatial surveillance are stated.

6.1 Conclusions

In the introduction we have seen various examples of 3D surveillance systems, which have in common that they incorporate videos in a 3D model of the monitored environment. The main goal of this approach is to take away the need of switching between videos and map views while tracking activity between videos, by combining the two. Focusing on different videos in these systems is done by 3D navigation, although this is a part that has not gained much attention. The main goal of this thesis project was therefore to improve navigation for 3D surveillance systems, subdivided in two research goals. Restating these goals, a discussion is added to evaluate if these goals were achieved and what the advantages and disadvantages of the solutions are.

Research goal 1. Given two virtual cameras, each pointed at a video canvas, how must we define a smooth viewpoint transition such that it enables the viewer to keep track of video content without much mental effort?

The navigation concepts proposed in this thesis are based on viewpoint transitions between the virtual cameras, resulting in a switch from one full view on a video canvas to a full view on another video canvas. What happens on the image plane during such a transition is influenced by the way the viewpoint parameters are interpolated. We identified three types of camera configurations, that share the same behavior regarding a changing projection of the 3D scene and video canvases on the image plane, which we called the “view change”. For each of these classes, a mechanism has been developed with the aim to keep the visual flow predictable, making it easier to keep track of important video content.

Concluding from user feedback and evaluation results, smooth view transitions were found helpful to keep track of important video content during view transitions. Especially during zooming transitions, the smooth transitions showed their use during the evaluation experiments. A comparison of our advanced transition functions compared to linear interpolation of the viewpoint position was not included in our evaluation. However, during informal experiments we could see
that the POI can be very difficult to track or sometimes disappear for a moment when we did not
use the improved functions.

After having designed a system of view transitions, we asked ourselves which navigation tools
we could add on top of this system:

**Research goal 2.** Which navigation concepts can be added to control viewpoint transitions in a
3D surveillance system that reduce the effort it takes for the user to track activity between videos?

To resolve this goal, a number of interactive 3D widgets have been designed that can be added
to the scene, rendered on top of the video canvases. These widgets can be selected to initiate view
transitions. Their placement and shape indicate the direction of corresponding view transitions,
to allow rapid navigation decisions while switching between virtual cameras. The main widget
type are glyphs, selectable arrows that point in the direction of views of the cameras they are
connected to. Glyphs can also be selected from the 3D cursor, the widget that follows the mouse
pointer, enabling the user to select a point of interest in a video. View frustum bounds provide
an indication of the camera that is selected as a candidate to navigate to, giving an impression of
which direction the resulting view change will have and what video content will be visible in the
next view.

The margin around the main view contains video thumbnails from other cameras. It provides
a direct view on videos that can be navigated to and provide a means for navigation if there is
no part in the scene where the 3D cursor can be active (i.e. no floor parts). Depending on the
environment model and preferences, the margin can be scaled to change the thumbnail size or
remove them completely.

The advantage of navigation from within the scene is that the focus can be kept on the same
area on the screen while preserving spatial orientation the same way some one would when he
walks through the monitored environment. But since that same person could get lost when the
environment is rather complex, so could some one navigating through the virtual environment. In
these situations, the addition of a map or the annotation of objects could be helpful.

A downside of rendering all 3D widgets on top of the video canvases is that this can result
in spatially incorrect situations, for instance when glyphs go through a wall or a another object,
which can be confusing. Although this behavior could be corrected for objects that are included
in the environment model, the problem still remains for other objects and it could hide large parts
of a widget, for instance the end of a glyph, reducing or destroying its effectiveness.

A user study was conducted to evaluate the usage of the spatial interface with simple, single view
transition scenarios. Due to the small number of participants and the nature of the experiments,
strong statistical conclusions could not be drawn from the results. However, we could analyze
from the participants’ task performance, questionnaire answers and recordings in which cases the
spatial navigation system performs well and which parts should be improved. Arrow glyphs were
found helpful in panoramic and zooming scenarios. The 3D cursor, together with the glyph menu
was found very helpful in orbit scenarios, although the direction in which the orbit glyphs are
pointing worked out counter intuitive for some users. Margin videos proved its use to keep track
of the selected point of interest over multiple videos. Feedback on the camera selection was found
misleading and distracting in its current form (view frustum bounds), although this concept could
be helpful when further worked out.
6.2 Future work

The interaction tools that are included in the prototype provide a basis for intuitive navigation through videos, but some possible extensions came to mind that can help the user maintain his orientation, finding his way through the scene or use the tools more intuitively. The labeling of certain objects with interactive annotations could speed up the process of navigating to certain locations. Think of selectable buttons at an elevator in a parking garage to navigate to a floor instantaneously, or a selectable region at the exit of a building to go to another building of the company.

Glyphs can be selected from the glyph menu by selecting one of the connections from the cursor position to a glyph. Such a selection can be seen as a gesture where only the starting point and end point count. The handling of more complex gestures could make it more intuitive to indicate navigation in a certain direction, for instance a curved draw to the left to navigate to a camera around the corner. Another option is to combine gestures to steer the selection of one of the glyph menu items, which can be difficult when there are many cameras to choose from. Such dynamic gesture based menus are described in [Bau and Mackay, 2008].

One of the problems with the view frustum bounds is that the direction of the camera is not always clear. A subtle animation that moves the bounds away from the camera would possibly enhance the indication of this direction. Another problem is that the view frustum bounds do not give an accurate indication of the parts of the scene that can be seen from the corresponding camera, as they are rendered in top of the scene and the video canvases. To solve this issue, the intersection of the view frustum with the the model (or at least the floor parts) should be visible as well.

Navigation with our prototype is mainly egocentric: camera transitions appear as a flight through the environment, starting and ending in a full view on a video. The surrounding videos in the border region of the screen allow for a somewhat more egocentric navigation as they provide an overview for the part of the environment that is close to the current camera. At a given time the screen shows at most a node and its direct neighbors in the transition graph. This overview could be extended to show all videos in the transition graph, the video sizes for instance based on geometric distances or distances in the graph. Smooth transitions between egocentric views and exocentric views, such as “zooming in” to a selected video from an overview, enable the user to easily switch between surveillance tasks.

Another aspect of the margin videos is the indication of the 3D cursor position in the main view. If we extend this concept of 3D tool visibility in multiple videos at once, we could make all videos interactive in the same manner, making them sensitive for mouse events as well. The combination with an extended exocentric system results gives the user a new level of navigational freedom but adds a new level of complexity for the user as well.

Only recorded video material from fixed cameras was used during this project. To use the concepts described in this thesis in real surveillance environments, better connection to camera hardware and support for controllable “pan-tilt-zoom” cameras would be necessary. While recent releases of OpenSceneGraph offer better support for video control, support for PTZ cameras is less straightforward. The freedom in rotation of these cameras would require synchronization with the virtual cameras in the 3D environment and result in a dynamic set of camera relations. This opens a range of new possibilities, compared to our static camera graph. For instance, cameras could all be focused at a selected spot in the environment, or cameras could be adjusted to each other to decrease spatial gaps between videos.

It is not always an easy task to create a complete data set that can be loaded in our application,
out of an environment model and a set of videos and some camera data. In our current work flow, which is done in Blender, cameras are calibrated by manually adjusting the parameters of a camera object until a frame of a video that is placed in the background aligns with the model. Algorithms for automatic calibration could be useful to integrate in the work flow. For further research, artificial data sets are important, as there is not many surveillance data available, mainly due to legal issues. Crowd simulation techniques [Ulicny and Thalmann, 2001] could contribute to more realistic evaluation videos. The work flow itself can be improved by adding tools for the calibration of virtual cameras and graphical editing of transition graphs.


Appendix A

Mathematical definitions

A.1 Bézier Curves

A bézier curve is a curve that is parameterized by the control points $P_0, \ldots, P_n$ as follows:

$$B(t) = \sum_{i=0}^{n} b_{i,n}(t) P_i, \quad t \in [0, 1],$$

where

$$b_{i,n}(t) = \binom{i}{n} t^i (1-t)^{n-i}, \quad i = 0, \ldots, n$$

![Figure A.1: A bézier curve.](image)

A.2 Common Perpendicular

The common perpendicular of two skew lines\(^1\) is the shortest line segment between them, from point $P_A$ on $l_A$ to $P_B$ on $l_B$:

$$l_A = p_A + t r_A$$
$$l_B = p_B + s r_B.$$  

\(^1\)Two lines are skew lines if they are not parallel and do not intersect.
A. Mathematical definitions

The common perpendicular can then be described as:

$$\overrightarrow{PA} \overrightarrow{PB} = \mathbf{l}_B - \mathbf{l}_A = (p_B + tr_B) - (p_A + tr_A)$$

$$= \begin{pmatrix} p_{B1} \\ p_{B2} \\ p_{B3} \end{pmatrix} + s \begin{pmatrix} r_{B1} \\ r_{B2} \\ r_{A1} \end{pmatrix} - \begin{pmatrix} p_{A1} \\ p_{A2} \\ p_{A3} \end{pmatrix} + t \begin{pmatrix} r_{A1} \\ r_{A2} \\ r_{A3} \end{pmatrix}$$

$$= \begin{pmatrix} p_{B1} - p_{A1} + sr_{B1} - tr_{A1} \\ p_{B2} - p_{A1} + sr_{B2} - tr_{A2} \\ p_{B3} - p_{A3} + sr_{B3} - tr_{A3} \end{pmatrix}$$

The direction of the common perpendicular can be computed by taking the cross product of $r_A$ and $r_B$:

$$\mathbf{c} = r_A \times r_B$$

$\overrightarrow{PA} \overrightarrow{PB}$ is parallel to $\mathbf{c}$, which gives us three equations with two variables $t$ and $s$:

$$\frac{p_{B1} - p_{A1} + sr_{B1} - tr_{A1}}{c_1} = \frac{p_{B2} - p_{A1} + sr_{B2} - tr_{A2}}{c_2} = \frac{p_{B3} - p_{A3} + sr_{B3} - tr_{A3}}{c_3}$$

$\mathbf{p}_A$ and $\mathbf{p}_B$ can then be found by using the solutions of $t$ and $s$ in the equations of $\mathbf{l}_A$ and $\mathbf{l}_B$, respectively.

Figure A.2: An illustration of the common perpendicular of two lines $\mathbf{l}_A$ and $\mathbf{l}_B$, shown in red.
Appendix B

Scenario Specifications

B.1 Scene Data

Cameras and glyphs can be added to a data set by defining nodes in an XML file. The `<scene>` node attributes describe information about the video frame rate and length, while camera data is stored in `<camera>` nodes and glyph data is stored in `<glyph>` nodes.

Listing B.1: The specification for the TNO dataset.

```xml
<scene frames="540" framerate="9" name="tno" imagetype="png" first="Camera_001">
  <camera lens="26.6846694946" rotation="[1.1603871585893859, -0.045259095728397369, 1.21065044403078617]" name="Camera_001" focus="9.04942017283" />
  <camera lens="24.2194385529" rotation="[1.2685822248458862, -0.1013299971818924, 4.04682117401123]" name="Camera_002" focus="9.24657629796" />
  <camera lens="21.4613456726" rotation="[1.3965858221054077, -0.044946812093257904, 1.0036171674728394]" name="Camera_003" focus="8.48281406195" />
  <camera lens="37.2410087585" rotation="[1.2794926166534424, 0.017251072451472282, 0.35417526960372925]" name="Camera_004" focus="6.00364824864" />
  <camera lens="32.7326202393" rotation="[1.303638219833374, -0.022431494668126106, 0.42505234513282776]" name="Camera_005" focus="3.57312170599" />
  <camera lens="30.8671302795" rotation="[1.4933465719232022, 0.017278756946325302, -0.3564921740185899]" name="Camera_006" focus="4.34840731183" />

  <glyph rotation="[0.0, 0.0, -1.5611919164657593]" scale="1.0" location="[−23.36732292175203, 4.2070536613464355, 11.415270805358887]" name="Glyph_1_3" />
  <glyph rotation="[0.0, 0.0, 1.5660220188140869]" scale="1.0" location="[−23.057138442993164, 4.4181962013244629, 11.41527175903203]" name="Glyph_1_4" />
  <glyph rotation="[0.0, 0.0, -1.003138930130005]" scale="1.0" location="[−16.007078170776367, 7.252052993774414, 11.41527175903203]" name="Glyph_1_5" />
  <glyph rotation="[0.0, 0.0, -2.5724976062774658]" scale="1.0" location="[−14.162900924862617, 6.438455816650391, 11.41527175903203]" name="Glyph_1_6" />
  <glyph rotation="[0.0, 0.0, -1.762900590866064]" scale="1.0" location="[−10.829598426818884, 7.506248741210938, 11.41527175903203]" name="Glyph_1_7" />
  <glyph rotation="[0.0, 0.0, 1.9751911163330078]" scale="1.0" location="[−11.536643981933594, 6.81349228162207, 11.41527175903203]" name="Glyph_1_8" />
  <glyph rotation="[0.0, 0.0, -1.3275841474533081]" scale="1.0" location="[−12.239776611328125, 8.766441345218438, 11.41527175903203]" name="Glyph_1_9" />
</scene>
```
Listing B.2: The specification for the PETS2006 dataset.

```xml
<scene frames="3020" framerate="25" name="PETS2006" imagetype="jpg"/>
<camera lens="50.0" rotation="[1.3993545770645142, 0.030557125806808472, 0.0]" location="[23.9205252546, 1.8730578422546387, 0.0]" name="Camera.001" focus="27.8235679024" />
<camera lens="59.0" rotation="[1.2995181083679199, 0.017180601134896278, 0.0]" location="[24.74758389282227, -3.192675829357158, 0.0]" name="Camera.002" focus="34.3560452052" />
<camera lens="35.4103355408" rotation="[1.46120977401734, 0.004966648228795796, 0.0]" location="[7.94050287315343, 6.807651519775391, 0.0]" name="Camera.003" focus="27.8235679024" />
<camera lens="33.526309967" rotation="[0.9656356136322021, 0.035474737136179276, 0.0]" location="[-2.8917784690856934, 10.009918212890625, 0.0]" name="Camera.004" focus="16.8506908547" />
<camera lens="33.7460441589" rotation="[1.5284435898582458, 0.0507224477648735, 0.0]" location="[-25.93504334960938, 9.9810447692871094, 0.0]" name="Camera.005" focus="34.3560452052" />
<camera lens="35.0" rotation="[1.0]" location="[23.74758389282227, -3.192675829357158, 0.0]" name="Camera.006" focus="27.8235679024" />
<camera lens="35.0" rotation="[1.0]" location="[23.74758389282227, -3.192675829357158, 0.0]" name="Camera.007" focus="27.8235679024" />
<camera lens="35.0" rotation="[1.0]" location="[23.74758389282227, -3.192675829357158, 0.0]" name="Camera.008" focus="27.8235679024" />
</scene>
```

Listing B.3: The specification for the PETS2007 dataset.

```xml
<scene frames="1090" framerate="25" name="PETS2007" imagetype="jpg" first="Camera.001"
</scene>
```

Listing B.4: The specification for the PETS2009 dataset.

```xml
<scene frames="794" framerate="9" name="PETS2009" imagetype="jpg"/>
<camera lens="50.0" rotation="[1.3993545770645142, 0.030557125806808472, 0.0]" location="[23.9205252546, 1.8730578422546387, 0.0]" name="Camera.001" focus="27.8235679024" />
<camera lens="59.0" rotation="[1.2995181083679199, 0.017180601134896278, 0.0]" location="[24.74758389282227, -3.192675829357158, 0.0]" name="Camera.002" focus="34.3560452052" />
<camera lens="35.0" rotation="[1.0]" location="[23.74758389282227, -3.192675829357158, 0.0]" name="Camera.003" focus="27.8235679024" />
<camera lens="35.0" rotation="[1.0]" location="[23.74758389282227, -3.192675829357158, 0.0]" name="Camera.004" focus="27.8235679024" />
<camera lens="35.0" rotation="[1.0]" location="[23.74758389282227, -3.192675829357158, 0.0]" name="Camera.005" focus="27.8235679024" />
<camera lens="35.0" rotation="[1.0]" location="[23.74758389282227, -3.192675829357158, 0.0]" name="Camera.006" focus="27.8235679024" />
<camera lens="35.0" rotation="[1.0]" location="[23.74758389282227, -3.192675829357158, 0.0]" name="Camera.007" focus="27.8235679024" />
<camera lens="35.0" rotation="[1.0]" location="[23.74758389282227, -3.192675829357158, 0.0]" name="Camera.008" focus="27.8235679024" />
```

Listing B.5: The specification for the evaluation scenarios.

```xml
<scene frames="1" framerate="15" name="All" imagetype="jpg" first="Camera_006">
  <camera lens="25.0" rotation="[ 0.326197147369285 , -3.1510660648435494 , 0.7654237813949585 ]" location="[ -0.004676804732376 , 9.238967955078125 , 3.121391773238777 ]" name="Camera_001" />
  <camera lens="25.0" rotation="[ -0.01958122253418 , -3.152433872229004 , -2.012939966583252 , -3.137313604345848 , 2.9561547633972168 ]" name="Camera_002" />
  <camera lens="25.0" rotation="[ -0.254308204650879 , -2.59495861258027 , -3.138709535326246 , -1.238999628607017 , 2.9561547633972168 ]" name="Camera_003" />
  <camera lens="25.0" rotation="[ 1.128893522338867 , -0.0045807729524371 , -2.059485349273682 , -3.1415927410125732 , 0.0 ]" location="[ -8.514032368910161 , 20.5 , 2.90000006953674316 ]" name="Camera_004" />
  <camera lens="22.0" rotation="[ -0.59485349273682 , -3.1415927410125732 , 0.0 ]", location="[ 4.190454231622207 , 20.3911081371582 , 2.90000006953674316 ]" name="Camera_005" />
  <camera lens="22.0" rotation="[ -0.59485349273682 , -3.1415927410125732 , 0.0 ]", location="[ 16.55970954895195 , 20.4126395095312 , 2.95614528665060586 ]" name="Camera_006" />
  <camera lens="22.0" rotation="[ 1.911718487730953 , 3.1415927410125732 ]", location="[ 22.050034547668457 , -1.298661231904629 , 3.0802597999572754 ]" name="Camera_007" />
  <camera lens="35.0" rotation="[ 1.9101276397705078 , -3.1415927410125732 , 0.0 ]", location="[ 45.569318122558594 , -2.64529304100762 , 2.9561457130433213 ]" name="Camera_008" />
  <camera lens="35.0" rotation="[ -1.92087707866339404 , -3.1415927410125732 , 0.0 ]", location="[ 1.5707963705062866 ]" name="Camera_009" />
  <camera lens="35.0" rotation="[ -1.919966293518751 , 3.1415927410125732 , 0.0 ]", location="[ -3.825298509326217 , -2.914074008585011 , 2.95614528665060586 ]" name="Camera_010" />
  <camera lens="35.0" rotation="[ -1.911718487730953 , 3.1415927410125732 , 0.0 ]", location="[ 22.050034547668457 , -1.298661231904629 , 3.0802597999572754 ]" name="Camera_011" />
  <camera lens="35.0" rotation="[ -1.9101276397705078 , -3.1415927410125732 , 0.0 ]", location="[ 45.569318122558594 , -2.64529304100762 , 2.9561457130433213 ]" name="Camera_012" />
</scene>
```

Scene Data
B. Scenario Specifications

B.2 Camera Transition Graphs

The prototype reads the transition graph for each data set from an XML file. Each parent `<node>` in the XML tree has one or more child `<node>`s. The `transition` attribute of the child node defines a transition from the parent node to the child node.

Listing B.6: Transition graph for the TNO data set.

```xml
<graph name="root">
  <node name="Camera.002">
    <node name="Camera.003" transition="orbit"/>
    <node name="Camera.004" transition="orbit"/>
  </node>
  <node name="Camera.003">
    <node name="Camera.001" transition="zoomIn"/>
    <node name="Camera.002" transition="orbit"/>
    <node name="Camera.004" transition="pano"/>
  </node>
  <node name="Camera.004">
    <node name="Camera.002" transition="orbit"/>
    <node name="Camera.003" transition="pano"/>
    <node name="Camera.005" transition="pano"/>
  </node>
  <node name="Camera.005">
    <node name="Camera.004" transition="pano"/>
    <node name="Camera.006" transition="zoomIn"/>
  </node>
</graph>
```

Listing B.7: Transition graph for the “Orbit” evaluation scenario.

```xml
<graph name="root">
  <node name="Camera.001">
    <node transition="orbit" name="Camera.002"/>
    <node transition="orbit" name="Camera.003"/>
    <node transition="orbit" name="Camera.004"/>
  </node>
  <node name="Camera.002">
    <node transition="orbit" name="Camera.001"/>
    <node transition="orbit" name="Camera.003"/>
    <node transition="orbit" name="Camera.004"/>
  </node>
  <node name="Camera.003">
    <node transition="orbit" name="Camera.001"/>
    <node transition="orbit" name="Camera.002"/>
    <node transition="orbit" name="Camera.004"/>
  </node>
  <node name="Camera.004">
    <node transition="orbit" name="Camera.001"/>
    <node transition="orbit" name="Camera.002"/>
    <node transition="orbit" name="Camera.003"/>
  </node>
</graph>
```

Listing B.8: Transition graph for the “Panoramic” evaluation scenario.

```xml
<graph name="root">
  <node name="Camera.005">
    <node transition="pano" name="Camera.006"/>
  </node>
  <node name="Camera.006">
    <node transition="pano" name="Camera.005"/>
    <node transition="pano" name="Camera.007"/>
  </node>
</graph>
```
Listing B.9: Transition graph for the “Zoom” evaluation scenario.

```xml
<graph name="root">
  <node name="Camera.009">
    <node transition="zoomIn" name="Camera.010"/>
  </node>
  <node name="Camera.010">
    <node transition="zoomIn" name="Camera.011"/>
  </node>
  <node name="Camera.011">
    <node transition="zoomIn" name="Camera.012"/>
  </node>
</graph>
```
Appendix C

Evaluation

This appendix contains the documents that were used for the evaluation, as well as the raw results. This includes the experiment scenario and a questionnaire about the experiment each participant was asked to fill out, along with the answers.

C.1 Experiment Scenario

Preparation:

1. Allocate two camera setups to the participant
2. Start the application, load the practicing setup, configure in "grid mode"
3. Start the webcam application ("Cheese")
4. Start the desktop / audio recording application ("gtkRecordMyDesktop")

Experiment:

1. Supervisor: Explain the role of the participant as a surveillance video observer and his task to track a person through an environment
2. Supervisor: Explain the map with camera setup and correspondence with videos
3. Participant: Try to follow people through videos
4. Task 1:
   a) Supervisor: Load the first camera setup
   b) Supervisor: Explain the first task: follow the person that starts at door X and write the door where he leaves in
   c) Supervisor: Instruct user to "think aloud"
   d) Participant: Put on the headset
   e) Supervisor: Start the recording
C. Evaluation

f) Supervisor: Start video playback
g) Participant: Perform the task
h) Supervisor: End the recording
i) Participant: Write down end door

5. Supervisor: Switch off "grid mode"

6. Supervisor: Explain the 3D navigation system.

7. Supervisor: Load the practicing setup

8. Participant: Try out all navigation elements

9. Participant: Try to follow people through videos

10. Task 2:
    a) Supervisor: Load the second camera setup
    b) Supervisor: Ask to handover mouse
c) Supervisor: start the recording
d) Supervisor: Start video playback
e) Participant: Perform the task
f) Supervisor: End the recording
g) Participant: Write down end door

11. Participant: Take off the headset

12. Participant: Fill in questionnaire

Afterwards:

1. Administration: archiving of video files

C.2 Questionnaire

Scenario task 1: Orbit / Panoramic / Zooming
Scenario task 2: Orbit / Panoramic / Zooming
End door 1: .......
End door 2: .......
Age: .......
Occupation: ................................................
Sex: male / female

Experience with computers in general: 0 0 0 0 0
Experience with camera surveillance (being the observer)
Experience with 3D navigation (for instance computer games): 0 0 0 0 0

not much ↔ much
1. How useful did you find the different navigation elements? (see the printed image)

<table>
<thead>
<tr>
<th>#</th>
<th>Navigation element</th>
<th>not useful</th>
<th>a bit useful</th>
<th>neutral</th>
<th>useful</th>
<th>very useful</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(clicking on) arrow glyphs</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3D cursor</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>border thumbnails</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>view frustum bounds</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

![Navigation Elements](image)

**Figure C.1:** Explanation of the navigation elements mentioned in Question 1.

Can you explain your preference?

2. Did you encounter any difficulties or obstacles while performing the first task (monitoring videos in a matrix view)?

3. Did you encounter any difficulties or obstacles while performing the second task (3D navigation)?

4. Did the application behave like you expected? (please explain if not)

5. Did you encounter any other difficulties or obstacles while using the application in general?

6. Do you have any additional remarks or ideas?
C. Evaluation

C.3 Answers to the Closed Questions

<table>
<thead>
<tr>
<th>Participant Number</th>
<th>#</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenerio (Orbit / Panoramic / Zoom)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task Results</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Door 2</td>
<td>C</td>
<td>A</td>
<td>G</td>
<td>C</td>
<td>C</td>
<td>G</td>
<td>C</td>
<td>?</td>
<td>C</td>
<td>C</td>
<td>?</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>Door 2 (2)</td>
<td>C</td>
<td>A</td>
<td>G</td>
<td>C</td>
<td>C</td>
<td>G</td>
<td>C</td>
<td>?</td>
<td>C</td>
<td>C</td>
<td>?</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>Age / Occupation / Sex</td>
<td>Age</td>
<td>27</td>
<td>27</td>
<td>29</td>
<td>27</td>
<td>27</td>
<td>31</td>
<td>25</td>
<td>23</td>
<td>28</td>
<td>26</td>
<td>34</td>
<td>26</td>
</tr>
<tr>
<td>Occ.</td>
<td>MSc</td>
<td>PhD</td>
<td>MSc</td>
<td>PhD</td>
<td>PhD</td>
<td>MSc</td>
<td>res.</td>
<td>MSc</td>
<td>PhD</td>
<td>PhD</td>
<td>Phd</td>
<td>MSc</td>
<td></td>
</tr>
<tr>
<td>Sex</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>F</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>Experience with computers / surveillance / 3D navigation</td>
<td>computer</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>surv.</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3D nav.</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Usefulness of Navigation Elements (Question 1)

| glyphs | 5 | - | 5 | 4 | - | 5 | 4 | - | 5 | 4 | - | 4 |
| 3D cursor | 2 | 4 | 4 | 4 | 5 | 1 | 5 | 4 | 4 | 5 | 5 | 3 |
| thumbs | 5 | 2 | 4 | 5 | 5 | 1 | 2 | 4 | 4 | 5 | 5 | 4 |
| vfb | 3 | 5 | - | 4 | 1 | - | 2 | 2 | - | 2 | 1 | - |

C.4 Answers to the Open Questions

**Question 1.** How useful did you find the different navigation elements? (see the printed image) Can you explain your preference?

1. *(3D cursor)* too much click / dragging normal glyphs easy enough for good navigation

2. *(3D cursor)* keeps focus on person; *(border thumbnails)* no time to look at them! *(view frustum bounds)* determines next camera

3. I found very useful the glyphs because they give you the idea of the direction and you can follow the man in the room easily

4. By looking at the border thumbnails it was immediately clear when the best moment for switching cameras was. Clicking on glyphs and the 3D cursor were both useful for navigation, but the 3D cursor was also useful for tracking the character.

5. *(3D cursor)* I used the 3D cursor as a tool to follow the subject, *(border thumbnails)* allows me to preview my next camera move, *(view frustum bounds)* I did not use it or notice it was there

6. *(glyphs)* Felt very natural to navigate through the environments, except that I had no clue about the camera orientation related to the selected arrow, *(3D cursor)* Maybe I did not use it, but subconsciously, *(border thumbnails)* Too distracting to keep an eye on different
Answers to the Open Questions

views. Made me lose track of the subject, (view frustum bounds) not immediately apparent what camera orientation they represented

7. (border thumbnails) Were not very noticeable, (view frustum bounds) orientation was not very clear (backward / forward direction)

8. View frustum bounds not particularly useful, but for some experienced or trained persons might help in some way. I found useful the highlighting of the border thumbnails, but the colors surrounding them may not be seen by everybody.

9. The 3D cursor was useful because I could follow one person with (mark a person). Especially in the second scenario it helped. The border thumbnails I only used for checking the yellow cross. The arrows I used mainly for the navigation.

10. Frustums are hard to interpret (many rectangles) and relying on them makes it easy to lose someone (clicking too early). Waiting they appear in the thumbnail better. The cursor really helps when switching between overlapping views, especially crowded scenes and “zooming out”.

11. The 3D cursor helped me follow the subject and anticipate which camera to select next, especially when “the cross” (cursor position on thumbnail view) and ”fading old cursor” (cursor ghost) was present to assist with the mental rotation.

12. (glyphs) it’s easy to use and it’s clear (3D cursor) didn’t know what it was for at first (border thumbnails) it’s easy and clear

Question 2. Did you encounter any difficulties or obstacles while performing the first task (monitoring videos in a matrix view)?

1. (panoramic) No, with 4 cameras it is easy.

2. (zooming) Yes, very difficult to distinguish the cameras AND persons.

3. (orbit) You first need to memorize the position of the cameras.

4. (orbit) With the first task, I didn’t have a mental image of the camera placement, which made me lose track of the character quite fast.

5. (panoramic) No

6. (zooming) Not really, as the subject was clearly unique.

7. (panoramic) Tricky when you can’t follow the subject, because he is not visible in any of the views. Nice to get an overview.

8. (zooming) Matrix view was more appropriate to my skills than 3D view.

9. (orbit) It was very difficult to switch between the camera views and find the person again. Especially since they all looked the same.

10. (orbit) Keeping track of camera positions relative to each other is hard. Keeping track of landmarks (doors) relative to each other is easier.

11. (panoramic) Not really, but had to make sure I was following the right guy which was sometimes difficult as equal people changed cameras almost at the same time.

12. (zooming) A bit. Because of how the camera’s images are placed on the monitor.
C. Evaluation

Question 3. Did you encounter any difficulties or obstacles while performing the second task (3D navigation)?

1. (zooming) No, though better 3D position perception for camera position (current view)
2. (orbit) Other than Question 4, not really.
3. (panoramic) No
4. (zooming) No
5. (orbit) The animations when changing camera induced me in terror sometimes. Only because extra movement distracts me from the movement of the subject to follow.
6. (panoramic) Yes, view delays and identical subjects made me unsure in whether I was tracking the right subject.
7. (zooming) First time use: difficult to see when the mean subject was in the next view. Sometimes a short moment without information on the subject. This was difficult when a lot of bystanders stood next to the subject.
8. (orbit) It was not clear from the beginning how the doors where standing to each other.
9. (panoramic) I only had one difficulty when switching between 2 cameras and the person I followed was very close to another person.
10. (zooming) Clicking too soon when zooming in can make one loose track of the person being followed.
11. (orbit) Not really, I had to learn the change of cameras a bit.
12. (panoramic) When changing to another camera, it shows a strange effect on the monitor. Not enough overlap between the cameras.

Question 4. Did the application behave like you expected? (please explain if not)

1. Yes, camera switching very natural and easy to follow.
2. Not at first. I started with dragging the cursor in the direction the subject was walking. Which is the opposite direction of the camera that gives the optimal view.
3. Yes
4. Yes
5. Yes
6. Almost, although canceling the view change after clicking was not apparent.
7. At first, I thought that the subject was visible in a video when it was walking over an arrow, but later I understood the cursor indication in the thumbnails.
8. In matrix view you should be able to select on the screen the position of the views and the order in which you want to view it.
9. Sometimes I expected to see the yellow cross in the next camera view earlier, but it showed up too late.
10. Yes

11. It did most of the time, but sometimes the change of the camera went very quickly (because the distance between the cameras was large), this confused me.

12. Yes

**Question 5.** Did you encounter any other difficulties or obstacles while using the application in general?

1. Cursor in border thumbnails could be improved? Better 3D perception could improve target following. Also point marking while camera switching would be useful.

2. No

3. No

4. No

5. (see Question 3)

6. It did feel a bit sluggish, not to be used when quickly succeeding view changes are required. This, to my feeling, has to do with the rather difficult reorientation, especially in the 3D rendered environment.

7. (audio log)

8. No

9. No

10. No

11. No

12. No

**Question 6.** Do you have any additional remarks or ideas?

1. No

2. Cool! Very functional application.

3. No

4. Maybe it helps to put a map of the cameras on screen.

5. Possibility to attach the 3D cursor to a subject: only clicking is enough, because the cursor moves by itself.

6. Delay in view transitions, were the most difficult to overcome.

7. No

8. 3D view and matrix view were not applied in the same simulated 3D world. Maybe you should consider this in future experiments! I didn’t get a map of the area used during the 3D view so that I can validate what I see in the simulation.
C. Evaluation

9. No

10. For the "zoom in" (and maybe flip) scenarios, consider not completely removing the previous camera view, to make it easier to adjust to the changing perspective, and help when clicking too early.

11. From experimental perspective: make it a repeated measure experiment as I expect a huge learning effect.

12. No
Appendix D

Article for IEEE CGA 2010

This appendix contains the article we have written for the special issue of IEEE Computer Graphics & Applications on Multimedia Analytics 2010. This article is a revision of our first submission, which was conditionally accepted (“with minor revisions”).
Spatial Navigation for Context-Aware Video Surveillance

G. de Haan         H. Piguillet         F.H. Post

Abstract

User interfaces for video surveillance networks are traditionally based on arrays of video displays, maps and indirect controls. To address the usability-limitation of this interface when the number of cameras increases, novel video surveillance systems aim at improving context awareness. However, interactive spatial navigation still proves to be difficult: unconstrained, free 3D control is too complex while a predefined camera path limits flexibility. Especially for live tracking of complex events along cameras, navigation decisions need to be made quickly and accurately based on the actual situation. Here, we present our novel spatial navigation interface that facilitates this type of video surveillance tasks. To keep attention focused on the action instead of an external navigation interface, one can directly navigate “in” the visible video by using the mouse. While tracking the action with the mouse, interactive 3D widgets augmented in the video update visual feedback on the available camera transitions. Optimized visual transition between individual videos ensure context-awareness while focusing on the action. We demonstrate our interface on several surveillance datasets and present results from a pilot user evaluation.

1 Introduction

Video surveillance control rooms have to deal with an ever-growing amount and diversity of data from complex environments. Currently, operators observe the video streams directly on large matrix display arrangements, combined with an interactive camera layout plan, e.g. see Figure 1. This basic layout is mainly useful for a general overview and for first detection of abnormal events. However, it is less suitable for coherently following the course of an event through time and space. In a complex environment with densely placed cameras or in, it is hard to maintain orientation and spatial context between individual streams. Here, cognitive overload can easily occur, especially in time-critical situations where live video streams are involved. For example, it can become extremely hard to perform person tracking tasks between different screens or to find an uncluttered viewing angle of an important object in a crowded scene.

In an effort to overcome some of these problems, recent systems have started to adopt the concept of displaying individual video streams within their spatial context, for example in a 2D map or 3D Virtual Environment (VE). See the sidebar “Towards context-aware surveillance.
Figure 2: View transition sequence of tracking a person between camera A and adjacent camera B in an office hallway surveillance scenario (from [1]). The guided 3D navigation enables simple first-person video observation while ensuring a good visual flow for spatial context. This method dynamically embeds and blends video canvases in a 3D VE which consists of characteristic model landmarks and perspective lines.

In our previous work [1], we argue that major bottlenecks in many surveillance tasks lies in the difference between view reference frames [8] in different camera views and the map view, and the shifting of visual attention. Where operators can naturally reason and act upon the first-person or egocentric views from a single camera, they have to mentally “translate” their reasoning back to the other camera views or the map, a third-person or exocentric representation. We therefore proposed a novel, first-person viewing and navigation interface for integrated surveillance monitoring in a 3D VE. Because most of the spatial information and overlap of all cameras is available, the system performs the mental spatial mapping for the user and provides this as feedback in the first-person 3D view. During navigation along images, the visual flow is enhanced to maximize coherence between video streams and spatial context, as illustrated in Figure 2.

In our current work, we look beyond the visual characteristics of spatial transitions and extend the context-awareness to the interactive navigation and visual feedback. First, our original interaction is restricted to “back-and-forth” navigation along a predefined 3D guidance path along series of camera views as defined in the “context graph”. Although guidance is essential, we do experience that this is often too limited for live tracking of events in a complex camera network: persons or vehicles simply do not follow fixed guidance paths. For this, it is important that navigation decisions can be made quickly, accurately and based on the actual situation. Secondly, we found that the manual construction of view transition parameters guidance paths for all camera-pairs is not only labor-intensive but also difficult to specify correctly for different events: as different events may happen in different regions of an observed video of a single camera pair, the transition parameters for a smooth transition can be completely different for each region. Therefore, it is important to improve the automatic and adaptive creation of transition parameters.

In this article we present our novel, context-aware navigation interface that addresses these two issues. First, based on a classification of camera configurations in a camera network, we provide different view transition schemes and optimize view transition parameters based on the selected point of interest. Second, we enhance navigation flexibility by allowing transitions to any camera that is “related” to the current view. Because this requires many new interaction possibilities, we propose a rich set of interaction tools for in-video interaction and augmented interface elements for direct visual feedback. By combining the improved view transitions with these novel navigation tools, we enhance the integrated experience for the more demanding egocentric video surveillance tasks. In the remainder of this text we elaborate on the designs of these tools, demonstrate their application on various scenarios and provide results from our user evaluation.

2 View Transitions

As described in the introduction, the smooth view transitions aim to help the operator in preserving spatial awareness while navigating from camera to camera in the context graph. In addition to the spatial awareness, we now add the requirement of keeping attention focussed at a certain point of interest in the scene. Therefore, attention should be paid to finding a transition function that results in a changing view that is perceptual the least difficult to follow. This means that for a given view transition, the resulting camera movement should both be simple in movement and velocity to make it predictable for the observer.
2.1 Classification of View Transitions

From the analysis of various surveillance scenarios (see “Sidebar: Surveillance Scenarios”), we identified three types of view changes: broadening or narrowing views, sliding views and rotating views. We have called the corresponding view transitions “zooming”, “panoramic” and “orbiting”, respectively. For each of these transitions, we have designed a function that results in a steady view change.

We define view transition functions as adjustments of the viewpoint’s position, orientation and field of view, all parameterized over time. In each of the three functions, both the orientation and field of view are interpolated linearly over time. In contrast, they do differ in how the viewpoint’s position is determined.

2.2 Improving the View Transition Function

Although a linear interpolation of the viewpoint’s position and orientation would be trivial, this does not result in desirable view changes in many cases. For zooming, this method works fine, since the view will broaden or narrow in a predictable manner and there will not be a lot of movement. When this method is applied to a panoramic transition, it often results in a rapid visual flow which can disorient the user. For orbiting, things will get worse, because the distance to the point of interest can vary much and the POI can move in an unpredictable path over the screen, e.g. see Figure 4(top row).

We can solve the issue that is introduced by panoramic movement by increasing the distance to the video canvases gradually so we get a broader view that slides through the screen at a lower speed. Such a mechanism is applied to a document viewing problem in [4] and further refined in [9, 10] in a search for an optimal viewpoint trajectory to keep the perceived speed constant, while [7] applies this concept to 3D navigation with a technique they call speed-coupled flying. In this technique, the height and tilt of the camera is coupled to the movement speed of the camera. We have developed a similar technique to define panoramic transitions by using Bézier curves. Bézier curves are suitable for this, because they can be parameterized by $t$ and a set of control points, which can be easily computed from the position and orientation of camera A and B, which is illustrated in Figure 3c. The control points influence the curvature of the trajectory, which allows for tweaking to obtain an optimal path. The position of the viewpoint at a certain time is then determined by computing $B(t)$, the position of a point on the curve at time $t$.

Orbit transitions are based on a rotation around a certain point of interest (POI). We can enhance the view transition by interpolating both the distance between the viewpoint and the POI and the position of the POI on the screen. From the resulting values, together with the orientation and field of view of the viewpoint, the position of...
Figure 4: Comparison of visual transition results between two viewpoint interpolation schemes. The top row shows how a simple camera viewpoint interpolation results in a curved screen trajectory of the POI. The bottom row shows how the viewpoint interpolation can be optimized to result in a smooth line of the POI on screen.

The viewpoint can then be calculated. This approach is an extension of orienting POI movement [5]. The result is a smooth change of the viewpoint-to-POI distance and a straight path of the POI on the screen. Figure 4 illustrates the difference between direct linear interpolation of the viewpoint position and implicit determination based on the POI.

3 Navigation Tools

The main purpose of integrating a set of videos with a 3D environment is to reduce the mental effort it takes to execute certain surveillance tasks such as person tracking. Many of these tasks are based on controlling the sequence of view transitions, which corresponds to navigation through the 3D environment. Based on formative evaluation and early experiments on various surveillance scenarios, we describe two navigation interface requirements. In order to perform interactive navigation from the current camera view to other cameras, the user first needs to be given information on availability and pose (position, orientation) of the neighboring cameras on a certain point of interest, e.g. a person. Second, the user must be able to select and initiate predictable transitions to these cameras, preferably while focusing on the point of interest.

With these requirements in mind, we developed a comprehensive set of interaction tools that is described in the following sections. An overview of visible interaction tools during a transition is given in Figure 5. We selected a simple 2D mouse to be the main input method as it provides fast and absolute 2D pointing at a point of interest in a video. The visual interface tools are either connected to cameras (glyphs, view frustum bound, border thumbnails) or connected to other tools (the 3D cursor). The connection to a camera is made clear by labeling the interaction tools with distinctive colors for each camera. The tools related to the cameras can be used interchangeably and compensate each other when certain tools cannot be shown or are preferred for any other reason.

3.1 Glyphs

Arrow shapes are widely used to represent motion. [6] uses arrow glyphs to depict dynamics in illustrations of animations, using their shape, size and other properties to illustrate the motion path and the direction and velocity of moving objects. Each glyph corresponds to a transition between two cameras. While viewing from a given camera, only the glyphs that are starting from this camera are made visible and can be directly controlled from within the camera view by selecting them with the mouse pointer.

3.2 3D Cursor

To enable navigation that is based on predictions about activity in videos, a tool is necessary to select a point of interest in the video. Such a 3D cursor can be "shot" upon the scene by shooting a ray from the mouse pointer to the 3D scene. In our case, we specifically use "floor" parts of the scene, which are the parts persons or vehicles can move across. In this way, the cursor appears to slide over the model when the mouse pointer is moved over the screen.

Although a simple cursor shape would be a point, this gives almost no visual feedback on its 3D position in the scene. A circle-shaped cursor already gives a better indication, as perspective viewing scales with the distance from the camera and gets rounder when it is viewed from above. Still, such a cursor can be misleading when it is placed on an object in the video that is not modeled in the scene. Therefore, a full 3D cursor would give the best indication of its position in
the scene. To prevent the cursor from covering a large part of the video, it is important that the cursor is built only from the parts that are necessary to span a 3D shape.

Meeting these requirements, we designed a 3D cursor that consists of the outlines of a cylinder that would fit around the average human (1 meter in width, 2 meters in height). To create the illusion of a transparent cylinder, the cylinder is always rotated towards the viewer so the vertical outlines are always on the outer left and right of the cylinder. The cursor is only visible when the mouse is points at a specified floor part and not at a selectable interaction tool.

When the cursor is visible, connections to cameras that can be directly reached with an orbit transition are shown with orbit glyphs. These glyphs are added because normal glyphs only indicate a direction of the transition, while orbit transitions mainly involve a rotation around a point in the scene. In this case, there is no particular direction that can be depicted by our straight glyphs. An orbit glyph consists of a dashed line from the top of the 3D cursor to the relevant camera and a polygon from the bottom of the cursor to the projection of the camera position on the same vertical plane as the cursor bottom.

When the mouse button is pressed, connections to the regular glyphs are made. The mouse pointer can now be dragged around the cylinder to select one of the glyphs from the cursor, which is similar to the selection of an item in a pie menu. The release of the mouse button initiates a transition to the selected camera. Figure 6 illustrates how the cursor menu can be used.

During a transition initiated from the cursor menu, it can be difficult to keep track of the selected point of interest in the scene. We therefore spawn a “ghost” copy in the scene just before...
the transition and position it at the point of interest. The human brain is well trained to follow the motion of distinct objects, in particular if the object follows a predictable path (which is achieved through our view transitions functions). With the aid of the cursor ghost, it is immediately clear where to focus on when the transition is finished. The change of orientation is also shown by the ghost, as the cross at the bottom circle stays aligned with the two horizontal axes of the world. While the ghost fades away, the 3D cursor can then be used again.

A disadvantage of such a static ghost is the time gap. The person or object that is followed is likely to have moved during a transition, appearing on another place as the cursor ghost. This could be partly solved by involving a prediction based on the direction of the menu selection or other gestural movements, but an exact prediction cannot be made without perfect automatic tracking.

3.3 Video Thumbnails

It can occur that a camera has connections to other cameras while it is not viewing any floor part of the model only a part or none of the glyphs. In these cases, navigation directly from within the camera view is not possible and raises the need for other interaction tools. Inspired by the SMV-player [3], thumbnails of the videos recorded by the neighboring cameras are put in a border that surrounds the main view. After the mouse button is pressed while hovering a thumbnail, a transition to the corresponding camera is initiated and the border is updated with the thumbnails for that camera.

3.4 View Frustum Bounds

When there can be zoomed in on one or more regions of the video, it is not exactly clear from the 3D tools which part of the video can be expected to be visible after the transition. We could intersect the view frustum of the corresponding camera with the model and project it on the image plane, but this projection would not be correct if there are many objects visible in the video that are not included in the model. A simpler approach is to take a rectangular slice of the view frustum, but the question would then be on which distance from the camera this should be done. A better estimation about the content that a camera sees can be made if the shape of its view frustum is shown, so that the position, orientation and visible area of the camera become clear. This is realized by the outlines of planes that intersect the view frustum on a number of distances until the focal distance of the camera.

4 Evaluation

As described in the previous section, the proposed set of navigation tools was designed iteratively based on early formative evaluation and informal experiments on various surveillance scenarios and camera layouts (see also Sidebar “Surveillance Scenarios”). To get more insight and feedback on the current usability and effectiveness of these tools, we performed a first evaluation study. Here, we describe the experiment setup, discuss some notable results and present a discussion.

4.1 Experiment Description

The main goal of our evaluation was to get qualitative results and feedback on the various aspects of the navigation interface in basic scenarios and detect difficulties in the task. Therefore, participants in our experiment performed a simplified surveillance tracking tasks. The main task is to track one specific actor — the subject — as he is recorded on several video cameras while walking randomly in an indoor office scene. The user should track the actor along the camera views and, once the actor walks through one of the doors, its identification marker should be written down.

As the emphasis of the experiment lies on how well people track actors during various transitions, we chose to keep the used scenarios and video material as basic as possible to avoid uncontrolled influences. First, we selected three very basic surveillance scenarios, each of which were based on a camera setup for a specific view transition, namely “orbit”, “panoramic” and “zoom”, see Figure 7. Second, the number of cameras was deliberately kept small to four cameras. Larger scale tracking task would be very difficult in the simple matrix interface. Third, we chose to use pre-rendered synthetic videos with simplistic and identical virtual actors. These actors are made identical on purpose, this to ensure that the subject needs to be followed from the beginning and cannot simply be spotted or re-tracked solely by visual recognition. For each of the scenarios, a number of synthetic videos were prepared which show a number of animated “actors” that walk across the scene, as seen from the cameras.

There were 12 people who participated in the experiment. Each participant performed a tracking task with two interfaces. In the first interface, the subject had to be tracked on a traditional static matrix lay-out, whereas for the second interface, our the navigation tools of our spatial interface were used. The visual difference between the two is illustrated in Figure 8. Although the results from the task in the matrix interface can be used as a comparison to our interface, we mainly include this to get user feedback on
which scenarios are difficult in general, independent of the interface that is used. Before the tasks were performed, the users were introduced to the system and could get used to the two interfaces. For the matrix interface, the users were presented with a printed plan view of the camera layout. Each tracking task took about two minutes to complete. The scenarios were divided over the all participants such that every scenario was used an equal number of times (each scenario+task was performed four times per task). The participant got two chances, so he could start over again when he thought that he had lost the subject.

During each session, participants were asked to think-aloud while they were performing a task, which was recorded with a microphone. At the same time, synchronised videos of the users in front of the screen were recorded, as well with the users actions and screen content of the interactive session. After the sessions, the participants had to fill out a questionnaire on their experiences during the tasks.

4.2 Experiment Results

Here we describe the results from our the evaluation. From the experiment results, the recordings and questionnaire answers, we distilled some noticeable issues, including differences between the different scenarios and the surveillance interface (“Matrix” vs. “Spatial”) used.

The questionnaire started with the question how useful each navigation tool was found and why. Figure 10 shows the results to this question combined in a histogram. This histogram shows that glyphs were found the most useful. The main reason for this was that the direction of the view transition is immediately clear from the arrow shape so it felt natural to navigate through the environment by using glyphs. The 3D cursor was found useful as well, but was only used actively in the orbit scenario, where it was a crucial tool for not confusing different persons. In the other scenarios, the 3D cursor was mainly used passively to mark a person and compare it with its instances on neighboring video thumbnails. Video thumbnails were mostly used to check if a person appears on one of the other videos than the main video, some participants used the cross indication. View frustum bounds were found the least useful and sometimes were found confusing in which direction they were pointing or misleading in how they occluded with the videos and the underlying 3D model.

The orbit scenario we constructed appeared to pose the most difficulties on the tracking task in both interfaces (Matrix/Spatial: 1 correct out of 4). For the matrix interface, this was due to the fact that camera angles are opposite and spatial reasoning is difficult. In real images one would resort to “matching” appearance and image features amongst multiple cameras, but as the scene was symmetric and all actors were identical, they could not be directly compared on appearance alone. The one participant succeeded by comparing door letters with the ones on the map and reasoning from that which of the persons is the subject. In the spatial interface the 3D cursor was found very helpful, especially when combined with the dynamic feedback on which transitions were available. However, difficulties still originated from the requirement to make quick decisions choosing one of the orbit glyphs while actively tracking the subject with the 3D cursor. If one of these is done wrong or concentration is lost for a moment, the subject is lost very easily. Participants sometimes experienced difficulties in matching the 3D cursor ghost with the correct actor in cases where actors walk closely together.

In contrast to the orbit scenario, participants...
in the panoramic scenario performed quite well with both interface (Matrix: 3 correct out of 4, Spatial: 2 correct out of 4). In this simple configuration with 4 cameras, the matrix interface is very straightforward and motion predictions are easy. In both interfaces, some difficulties arise when two persons leave the video at the same moment in the same area or when people walk in a dead area. For the spatial interface, timing proved sometimes difficult when transitions were slow. Thumbnails could be used to determine the right moment for a transition but most people are focused on the main view too much. Most participants preferred simple clicking on arrow glyphs to navigate, and avoided the use of thumbnails and drag-selection with the 3D cursor.

In the zooming scenario we observed remarkable improvements in the spatial interface (Matrix: 0 correct out of 4, Spatial: 4 correct out of 4). For this scenario, most participants failed to keep track of the subject in the matrix interface. This was mainly caused by the difficulty to verify which persons are the same ones on multiple videos. If the attention changes from one video to another, it can happen that the subject has not appeared on that video yet. At that moment, it is already too late to regain focus on the first video, which means that the subject is lost. In the spatial interface, the participant could navigate from within the same view. As the videos smoothly merged into each other, the attention could always be on the subject if the right moment for a transition is selected. Some participants achieved this by following the subject with the 3D cursor until the cross-hair indicator cursor appeared on the thumbnail of the next video.

4.3 Discussion of the Evaluation

What we have seen from our recordings and questionnaire answers is that the augmented navigation tools helped the participants to keep track of the subject and maintain a mental model of the spatial context. The limited amount of training the participants received had a large influence on the experiment. Studying the recordings, it often occurred that the participant did not respond quickly enough to keep track of the subject. Future experiments should therefore incorporate more extensive training sessions to assure that the participants are accustomed enough to both the navigation tools and the actual task scenario environments.

We have seen that the orbit scenario was very difficult for the participants to perform correctly. We attribute some of this to some inconsistencies between different navigation tools. For example, while arrow glyphs point in the same direction as their corresponding view transition, orbit glyphs point in the opposite direction of their view transition (namely in the direction of their corresponding camera). This makes behavior of the orbit glyphs exactly the opposite to the regular glyphs, since the user has to make a gesture in the opposite direction of where he wants his view to change to. We therefore expect that an redesign of these inconsistencies can make the orbit transitions and corresponding camera choosing less difficult.

The view frustum indications need some im-
of small events and individual persons or vehicles. Zooming in however goes at the expense of camera coverage, and context is easily lost by the operator. Also, views of mobile cameras used in the field should be dynamically integrated with static and adjustable surveillance camera configurations. We see interesting opportunities for the dynamic integration of these cameras in our user interface.

The work presented here provides support for either direct observation of scenes and events, or for reconstruction of events using recorded surveillance video footage. In real-time observation, an “instant replay” functionality would be very useful, for back-tracking of missed details.
and events. Time management and navigation in time is in itself an important topic, and techniques must be developed to preserve the spatial and temporal context during such operations.

For tracking of persons, automatic techniques from computer vision are being investigated [2]; when these techniques are sufficiently fast and reliable to be used in real-time surveillance, they can be easily integrated to enhance navigation by the operators.

As the perceptual and cognitive processes involved in video surveillance are not completely understood, the improvement of new techniques claiming to support effectiveness or efficiency should always be evaluated experimentally. But for the evaluation of surveillance interfaces, a variety of surveillance video footage would be necessary. However, due to privacy concerns, usually forced by legislation, authentic video materials are very hard to obtain. Therefore, we will also have to revert to artificial scenarios, such as self-generated movies or synthetically generated scenes. Setting up good test scenarios and designing performance metrics for this purpose is an interesting challenge by itself.

References

Sidebar: Surveillance Scenarios

For evaluation purposes, we have used several surveillance scenarios. Before they can be loaded into the surveillance prototype, a scenario must be prepared first. The preparation is done from within the Blender 3D modeller in combination with a python script. This script is capable of rendering animations to videos, which makes it possible to create artificial datasets. Because the availability of real camera data is limited, such artificial datasets can be very useful to create any possible scenario that suits specific needs.

The preparation of a scenario involves creating a model of the environment, in which virtual cameras then are aligned with the real cameras. Subsequently, arrow glyphs can be put in the model, as well as eventual animated objects. Using a python script, spatial properties of the cameras and glyphs can then be exported to an XML file. Optionally, when we want to create an artificial scenario, videos can be rendered from the virtual cameras. The model, XML file, and videos form a scenario that can be loaded into the prototype.

PETS datasets

The Computational Vision Group of the University of Reading has made several datasets that are used as benchmark data in the PETS workshops publicly available. We have prepared three of these datasets for our system. Examples can be seen in Figure 11[a, b, c]. These datasets contain video footage from cameras in public environments where many different people walk in and out. The datasets were useful to investigate crowds and certain events when viewed from different angles and distances, although they were less interesting to evaluate our set of navigation tools as a whole, since all cameras are focused at the same area.

Office Hallway

The office scenario, which can be seen in Figure 11d, offers a suitable environment to test all navigation tools. It contains a camera setup that includes various camera relations. There is not much activity in the videos, which makes this scenario less interesting for tracking experiments.

Highway Traffic

Camera setups used to monitor traffic events differ from the cases mentioned above in that they consist mostly of sequential cameras that are placed at large distances from each other along a highway road. We created a synthetic scenario that incorporates a piece of a highway and animated some car models in it. The resulting dataset could then be used to smoothen out the zooming transitions.

Evaluation Environment

To evaluate our prototype, we needed an environment that features various camera setups, as well as video activity that is not trivial to follow. Since surveillance videos are not widely available, especially the ones that suit our needs, we created our own synthetic evaluation environment, which can be seen in Figure 11f.

---

1http://www.blender.org/
2http://www.cvg.rdg.ac.uk/
3IEEE International Workshop on Performance Evaluation of Tracking and Surveillance
Sidebar: Towards Context-Aware Surveillance Systems

New types of interfaces were proposed for spatial context in multi-camera video surveillance systems [6, 8]. The spatial context of camera streams is generally conveyed through placement of video thumbnails, camera icons and coverage indicators on a 2D map of the environment. In the Spatial Multi-Video player [7], the coherence between video streams is provided by carefully arranging spatially related video thumbnails around the current, selected camera stream. In a sense, these interfaces convey spatial context through embedding multiple egocentric representations (video streams) into the exocentric 2D map representation. Their user study revealed that spatial context, provided by either a map or spatially related videos, improved user performance and acceptance for tracking persons walking on an office floor across multiple cameras.

Other researchers experimented with adding a third dimension to this spatial representation, by putting the video streams directly in a 3D model of the monitored area. In the Augmented Virtual Environment (AVE) system [9, 11, 16] and the Video Flashlight technique [10], live video streams are integrated with an accurate 3D environment model using Projective Texture Mapping. These approaches aim to provide sufficient information to afford seamless transitions to the 3D world for observation from arbitrary viewpoints. Other approaches use 3D canvas objects on which the videos are projected, and focus more on user navigation interface [15, 5, 3]. To explore the design space of combining videos with 3D environment models, the testbed of [15] combines existing, static rendering techniques and provides basic navigation and report on possible usage patterns. Their user study indicates that 3D integration can indeed be a useful addition in surveillance tasks [14].

In [3] complete video streams are spatially embedded in a first person view, with the motivation being to alleviate duality in the navigation interface. Dynamically transforming texture billboards are used, inspired by Photo Tourism [13], and focus on improved real-time navigation. The user should be able to smoothly navigate to this position instead of teleporting, in order to maintain spatial context and orientation [2]. Experiments performed in [1] reveal that for inspecting videos arranged in space, animations and smooth transitions of the viewpoint can help the user to build up the mental map of spatial object relations. The use of a 3D environment does introduce the problem of fast and easy 3D navigation, e.g. losing orientation and awkward controls. The important role of automated and guided navigation for a complex surveillance context was already emphasized in [15]. Although the AVE-based V-Sentinel system [16] reports on automated fly-throughs with optimal and natural display along predefined paths and generated alarms, details of these transitions and constraints are not reported. The 3D video player in [5] provides arbitrary 3D viewing or camera transitions to follow automatically tracked persons. Our method focuses on simplifying interactive navigation, such that the user is guided or constrained along the views in a comprehensible manner, see also [4]. In an extension of Photo Tourism, Snively et al. also combine path-based, egocentric navigation and blended photographs [12]. In our current work, we extend the navigation interface by augmenting the videos with context-aware feedback.

References


