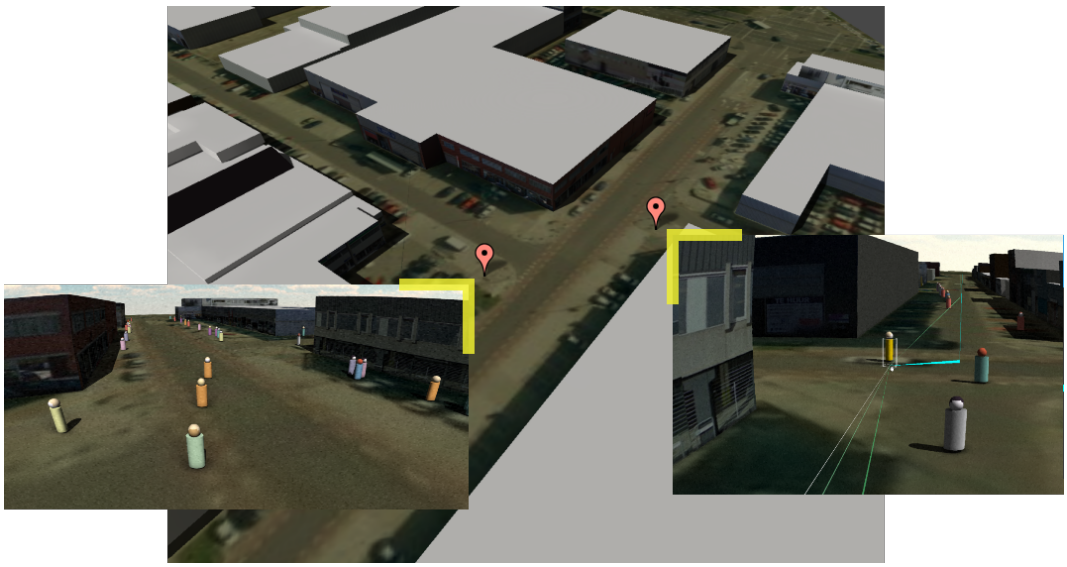
Active Multi-Camera Navigation in Video Surveillance Systems

Master's Thesis

July 2012



Marnix Kraus

Active Multi-Camera Navigation in Video Surveillance Systems

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Marnix Kraus
born in Delft, the Netherlands

Computer Graphics Group
Department of Media and Knowledge Engineering
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

Active Multi-Camera Navigation in Video Surveillance Systems

Author: Marnix Kraus
Student id: 1358332
Email: m.kraus@student.tudelft.nl

Abstract

Active cameras are increasingly used to extend the area coverage and to improve image quality in video surveillance. With such a *pan-tilt-zoom* camera, operators can control its rotation around two axes and can use an optical zoom to get a better shot. Although multiple video streams are monitored simultaneously on a large grid of displays, they lack spatial correlations that the operators can rely on. In stressful situations, such as during the pursuit of a target, it is hard for the operator to maintain situation awareness while controlling cameras and switching between them.

In this work, an interface is presented that helps the operator in controlling the cameras while maintaining spatial knowledge of the environment. Its design extends the work of De Haan et al. [de Haan 10] on augmented surveillance interfaces, which only uses static video cameras. First, pan-tilt-zoom camera operation is integrated by providing the user with new camera controls and visual feedback methods. Second, the special hardware joystick is replaced by a simple mouse and keyboard control in order to have a seamless integration with the overall system. Finally, the user is assisted in steering multiple active cameras by automatically determining and focusing an optimal camera based on user-indicated target location and movement direction.

As active cameras allow live interaction by the operator, it is difficult to experiment and consistently perform evaluations using real-life surveillance systems. Therefore, the prototype interface is implemented upon a simulator which generates virtual scenarios and live content. Evaluation measures are designed to measure situation awareness and performance of the operator during the pursuit of a target. In the evaluations performed, the prototype was found to have a better learning curve, a better ease of use and user performance is increased in comparison to a classical interface. These findings indicate that the presented interface can enhance operator performance in surveillance control rooms that monitor vast and complex areas.

Thesis Committee:

Chair:	Prof. Dr. Ir. F.W. Jansen, Faculty EEMCS, TU Delft
University supervisor:	Dr. Ir. G. de Haan, Faculty EEMCS, TU Delft
External supervisor:	Ir. M.G. van Elk, Managing Director at S&T Vision
Committee Member:	Prof. Dr. M. Neerincx, Faculty EEMCS, TU Delft

Preface

This project is the third episode of a series on interactive video surveillance. It involves interactive images and pan-tilt-zoom camera controls. When I started on finding a subject for my thesis, I was mainly interested in computer game development. Although the available topics were fine, they were not really appealing to me. Fortunately, the TU Delft has a separate computer graphics group that also performs research in other types of visualizations. Gerwin de Haan introduced me to some of his topics of research. Especially video surveillance caught my attention, because of its high level of user interaction. I was eager to find new solutions that could be used in real life.

This thesis went exactly as I expected how it would be like to write such a large report about only one topic. The difficulty was at a good level and the amount of work could easily be altered to fit inside the planned hours for this project. I had the most fun evaluating the system with the kids from ROC Mondriaan, supervised by Rob Elsing.

The field trip to video surveillance center CIV in collaboration with Michel van Elk from S&T and TNO has given me new insights on my own project by letting me into one of their brainstorm sessions. I hope that they have also found some interesting pieces to continue their own study.

Special thanks to my parents and sister for being the first ones to test my prototype. We had a good laugh.

Marnix Kraus
Delft, the Netherlands
July 13, 2012

Contents

Chapter 1

Introduction

Camera surveillance systems are widely utilized in the daily routines of companies that want to keep their safety standards high. The number of surveillance cameras worldwide comes to a rough estimate of five million devices and that amount is still rapidly expanding [Keval 09]. The main purpose of video surveillance is to ensure safety and security in a given area. Not only by preventing crime, but also by observing the area for other calamities that could occur. It is the aim for the operators that control and survey these cameras to maintain overview of the situation and respond swiftly when anything exceptional happens. A common task that comes with major difficulties is the pursuit of persons via cameras in particular. This project makes an effort in researching and overcoming the problems that the camera supervisors have during their work. This introductory chapter provides a first glance at the domain of the project, the problem description and the research goals of interest.

1.1 Domain Analysis

Camera operators work in small teams behind their desks. They look out for calamities and suspicious acts and persons. Each operator focuses on a particular region by viewing a number of monitors displayed in a matrix (Figure 1.1). The average number of simultaneous views differs from five to fifteen and is still increasing over the years [Keval 09]. Some TV monitors are split into even smaller images, depending on the importance and detail the operator needs. The monitors themselves show no spatial relationship of any kind, only the numbering of the cameras is in a somewhat logical order. Expanding the system with extra cameras in between would already disturb this logical ordering. So, expansion of the initially installed set of cameras is not easy to perform.

1.1.1 Surveillance Equipment

On the operator's desk is a spot monitor and a special camera joystick. These two tools are used to control each camera independently. The installed cameras in most

Figure 1.1: A matrix of CCTV monitors. Static monitors show the overview, a PTZ monitor stands in front of the operator. From `http://ipcctv-solutions.com`.

CCTV systems nowadays are *pan-tilt-zoom (PTZ) cameras* and are also referred to as *active cameras*. These devices can be controlled in two rotational axes and also have an optical zoom functionality that can be used to enhance the objects on screen by decreasing the field of view. The pan-axis is used to rotate around the camera's up-axis and the tilt-axis is used to pitch the camera's lens up and down (Figure 1.2). The joystick controls the pan and tilt function by pushing the stick around in its socket. The zoom functionality is often controlled by rotating the stick around its center.

A small numerical keypad next to the joystick is used to type in the numbers of the different cameras. Whenever the operator wants to control a certain camera, that camera has to be displayed in the spot monitor first. To do so, the user types in the number of the camera and confirms the number with a return statement (i.e. by pressing the Enter-button). The camera numbers are displayed on the images in the large matrix view, so it is easier for the operator to access them.

1.1.2 Task Analysis

During the shift of a camera surveillance operator there are a number of tasks that the operator performs. The activities can be divided into reactive and proactive tasks [Keval 09]. While proactive tasks involve observation and gaining a global

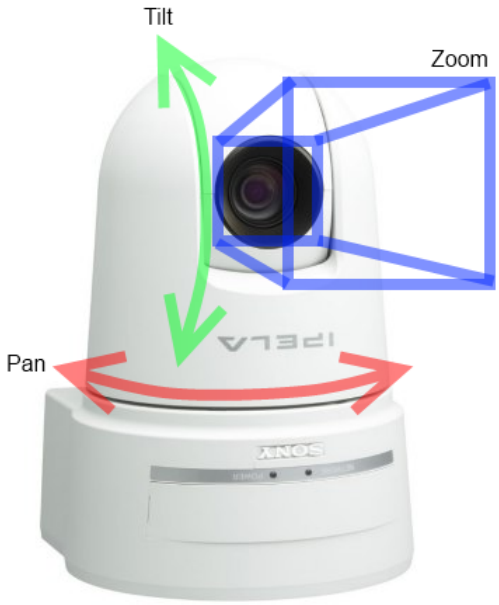


Figure 1.2: The pan-tilt-zoom camera can pan horizontal, tilt vertical and zoom its optical lens.

overview of the activities in the area, reactive tasks involve following targets and locating direct calls on screen from officers in the field. A task inventory performed by TNO Soesterberg led by C. Schilder and W.D. Ledegang shows the different tasks that operators can perform during their daily operations (Table 1.1).

It can be seen that the reactive tasks all include fast handling of controls and require multitasking qualities from the operator. The operators are bound to time and have a high workload during these tasks. They use different communication means to inform the officers on the site, which makes multitasking an even more desired quality for an operator.

The time that each of the different types of task occupy depends on the day of the week and the time of day. The weekends are often full of reactive tasks, while the rest of the week is mostly occupied by proactive tasks. This implies that the weekends bring along higher stress levels to operators. Figure 1.3 shows the ratios between weekends and weekdays in an average control room. In the weekends, reactive tasks may even take up to 65 percent of the total work shift.

1.2 Problem Description

Now that the domain of the camera surveillance operator is defined, a problem description can be created from the trouble that the operators have during their daily work. Especially the reactive tasks that the operators have to go through are highly stressful [Keval 09] and therefore interesting. The users have a hard time

Task	Description	Type
Monitoring	Look out in the area via the video screens.	Proactive
Surveillance	Perform a focused round through an area based on expectations and hot spots.	Proactive
Search	Try to find persons based on a description or picture.	Reactive
React to alarms	React to an alarm and navigate to the alarm on screen.	Reactive
Follow	After detecting a person on suspicious behavior, follow that person through the area. Also: • Communicate with people on site. • Pass camera images to other operators. • Register incidents.	Reactive
Rewind and look back	Directly rewind to a certain point in the video and provide a description of the suspect or situation.	Reactive
Search for old events	Rewind through old images whenever someone asks for old incidents.	Proactive.

Table 1.1: The task routine of the camera operator. Courtesy of TNO Soesterberg.

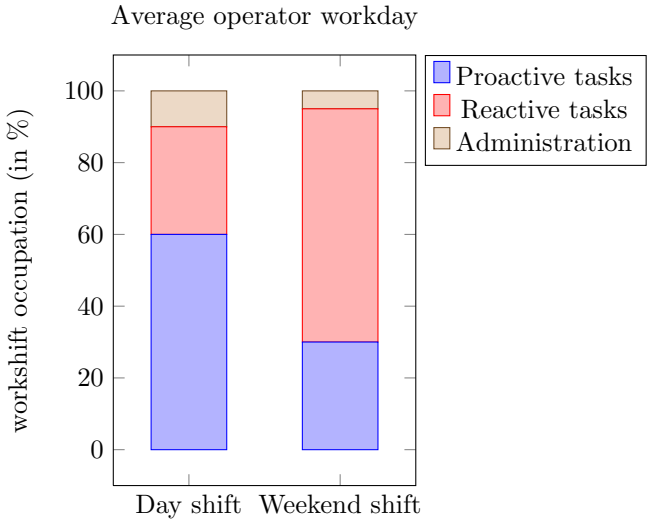


Figure 1.3: The average day of a surveillance operator. Weekend shifts consume more time for reactive jobs. Data from Keval [Keval 09].

doing their job when performing reactive tasks, like following a running person on screen. It was found that a number of problems cause the high workload level of the employees: Participants in the tests by Keval indicated that the cameras are illogically ordered, that some cameras are placed at bad positions, and the ratio of operator to camera is too high. Not only does failure result in low efficiency and performance; losing a target that is being chased embarrasses operators in front of their colleagues and employer.

1.2.1 Definitions

The causes that induce failure can be derived from the comprehensive concept of *situation awareness* (SA). Situation awareness is the level of perception that a person has about the environment and everything in the area of interest [Matheus 03]. A concept that contributes to situation awareness is *spatial knowledge*. Already in 1982, Thorndyke [Thorndyke 82] proposes two mental models within the spatial knowledge domain that are also present in video surveillance: *orientation* and *navigation*. In video surveillance, the word navigation is equivalent to steering and switching to the correct location in the area in order to find things. This is an action with focus on solving a route from the current location to the new location on screen, where orientation is a more mental concept of awareness. The operator has to figure out what the current screen is showing and where that could be in the real world. Figure 1.4 shows both concepts in a figurative way, emphasizing on the differences between the two. If the operator is not able to use both concepts of orientation and navigation in a correct manner during a stressful situation, failure in task is likely to happen.

Two other definitions of interest that occur in literature are *exocentric* and *egocentric*. Exocentric systems keep the user separated from the interface, as in a third-person view or a map overview. Egocentric systems make the user of the system a part of the world. The user is an entity in the world in some way or another (e.g. by means of a first-person camera or virtual reality).

1.2.2 User Scenarios

A typical example of a problematic situation is the following scenario. A person is walking down the street and the operator has a view on the target from behind. By zooming in, the person can still be spotted on a long distance. Suddenly, the target decides to walk around the corner and the operator has lost its target. When such a scenario occurs, the operator has to switch to a new camera that is just around that corner. By means of orientation, the operator has to think of which camera is a good choice. If he knows the cameras by heart, he can easily switch around the corner. Afterwards, the camera still has to be positioned correctly to find the target once more (Figure 1.5). In another case, if the operator is on unfamiliar territory, he will have to return to the large matrix view of images and search for a camera where he can actually see the target. It is possible that there is no such camera,

(a) In orientation issues, one is looking for information in relation to itself. In this example: "In which street is this camera that I am looking through?"

(b) In navigation issues, one searches for the route to see a certain object. In this example: "How do I steer my cameras to get a closeup of the face?"

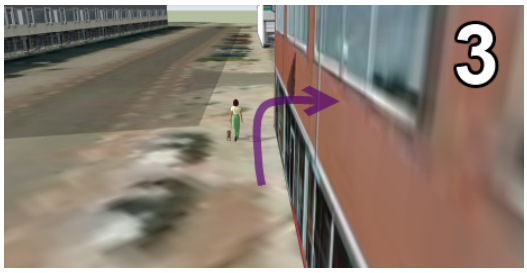
Figure 1.4: Two examples of issues that operators face in the concepts of (a) orientation and (b) navigation.

because cameras are incorrectly oriented. The operator is completely dependent on the current rotation of the cameras (Figure 1.6). Alternatively, the operator could look at a map and find the number of the camera that is around the corner.

During the latter scenario, the chance at mode errors is high. This means that there is a high risk at doing an action that does not comply with the current state of the system (e.g. the user looks at the matrix view and tries to steer one of those images with the joystick, instead of the spot monitor). The sequence of actions is long and there is a high chance of losing the target.

1.2.3 System Centralization

Currently most control rooms have their own regions to control. There are a number of regions that can be viewed on the monitors so that the operators can do their surveillance rounds. Only a limited number of areas is assigned per control room. Future plans of the surveillance companies are to centralize and merge these small control rooms into larger ones. There will be more different regions that can be viewed and multiple operators can share their regions. This is a good idea for optimization of the work flow, because the work can be shared among more persons. But a disadvantage of this approach is the fact that the operators should memorize more regions and camera numbers. The learning trajectory is relatively long and will only increase this way. If there would be a way to not learn all camera numbers



(a) The operator sees a woman with a dog in camera 3. The woman is about to walk around the corner.

(b) The operator knows the camera number of the camera that has a better view and types in '5'.

(c) The camera still needs adjustment, so the camera is panned, tilted and zoomed to get a closeup of the woman.

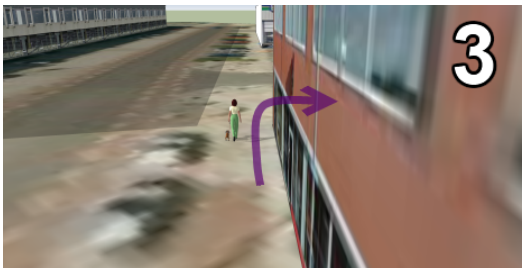
Figure 1.5: An example problem in the daily work of the operator in unfamiliar areas: The operator does not know the area and has to search for the target once lost.

by heart, it would be an improvement to the system. Learning the actual camera numbers would be obsolete for this particular task.

1.3 Project Overview

To narrow down the scope of the project, the task of following will be the our focus. There is a lot to enhance to the task sequence and a more advanced system is wanted by operators to improve their efficiency. The operators have issues with switching to the correct cameras, especially when the environment is unknown to them. The goal of this project is to design a new prototype that aids the operators in their task of following. The system should not take over, but only enhance the possibilities for the user.

The project is approached from a computer science point of view with a practical perspective on solutions. First, a literature research is performed to investigate the already existing alternatives for the matrix view. A prototype implementation is designed and created by laying focus on the difficulties that the operators have with PTZ cameras. Finally, user evaluations take place to see if the newly designed interface is better than the old matrix system by means of performance, usability



(a) The operator sees a woman with a dog in camera 3. The woman is about to walk around the corner.

(b) The operator does not know the camera number and has to look through all images in the grid.

(c) The operator types in the number if he has found the subject. The camera still needs to be adjusted to get a good shot.

Figure 1.6: An example problem in the daily work of the operator: The operator knows the environment and can switch easily, but has to readjust the camera.

and learning curve. The questions that are of interest to come to such a system are the following.

Research Question 1. *What intuitive types of camera controls can be used to steer a single PTZ camera? How do these controls mix with a complete system that also lets the user switch between other cameras?*

We are looking for a new way of steering to see if the joystick controls that are used nowadays can be replaced or enhanced with something intuitive. This new style of control should be able to be integrated in the overall system.

Research Question 2. *What alternative navigation methods can be used to improve active multi-camera video surveillance systems during intense situations and in which ways do they improve video surveillance systems or the user?*

This project aims for a new way to navigate through the range of cameras available without remembering all camera numbers. A number independent method is of interest that can improve the performance and the usability of the system.

Research Question 3. *In what way can the new system be effectively tested on low costs and a high number of iterations?*

PTZ cameras are expensive and require live interaction with the user. A way to effectively perform user tests is of interest. Not only for this test, but also for evaluations in the future.

In the upcoming chapters, we will show how these questions evolve into new interface concepts for operators and how requirements can be met to actually come to an overall prototype, that implements both solutions for the concepts of navigation and orientation in spatial knowledge and situation awareness.

This report defines and structures the literature study and user analysis, and designs documentation and implementation features of the final product. Chapter 2 will describe a literature study of published references and other interesting public material. System ideas are formed into a plan and features are defined to implement into the prototype. Chapter 3 introduces a low cost system to perform user evaluations that is used as an input for the system to be designed.

The system is defined into three main features, which are described in Chapters 4 to 6. Chapter 7 will elaborate on the implementation of the system. Evaluation results from user tests and the setup to perform these tests can be found in Chapter 8. Chapter 9 finalizes this report with discussions, conclusions and recommendations for the continuation on this branch of research.

Chapter 2

Analysis and Literature Study

Before the new system can be designed to aid the operators in their task, there are a number of subjects to discuss first. Preparations have to be made by analyzing the currently used systems, users and scenarios, and by studying literature, research, and products that are already on the market. This chapter will first provide a literature survey of related work, then some extra sources of inspiration and afterwards an initial design approach for the new prototype.

2.1 Related Work

Before designing the new system, we will have a look at research that has already been done by other researchers with a similar vision towards video surveillance. The subjects described in this section do not only show the work on the scope of pursuing targets by cameras, but also focuses on other tasks that can be of influence on the design.

2.1.1 Multi-Camera Interfaces

The demand for both a better cognition on orientation and navigation creates a wish for new interfaces that provide spatial information on the multitude of cameras in an area. Numerous studies have led to alternative views for the matrix view and have shown that especially orientation problems can easily be addressed.

2D and 3D Image Representations

The use of 2D maps is a concept that is proven to work for over centuries. It gives the reader a certain feeling of awareness, because there is a relation between the image and the spatial context of the real world. Extra information in the map (e.g. icons and colors) can make it clearer to understand or make it more confusing to read, depending on the density of data shown on the map. The use of maps is a clear example of an exocentric approach.

In 2006, Girgensohn et al. [Girgensohn 06] presented a map structure showing relevant cameras in a certain area. The cameras of relevance differ dynamically due to the user's input position on the map. The cameras that can view that position the best will be highlighted and enlarged. Also, association lines are drawn between the camera images and the positions on the map. The colors on the map also comply with the colors around the images (Figure 2.1).

In addition to the associated views in a 2D map, Wang et al. [Wang 07] made use of a 3D model of the environment. Because the videos are inside the 3D model, instead of associated outside like in Figure 2.1, this technique is referred to as *embedded video* (Figure 2.2). Subsequent to their research in 2007, Wang et al. [Wang 08] also performed tests that combined association lines with 3D views. It was a combination of their own work and that of Girgensohn et al. (Figure 2.3). Their research shows an exocentric way to represent a spatial relationship in 3D. This means that multiple stories were also possible to use in this design by Wang et al. The orientation of the cameras was only displayed by a simple arrow, which did not seem to be sufficient. Subjective tests showed that the methods were 'easy to learn' and that the mental workload was low.

Relevant Cameras

When pursuing a target through video images, it is often hard to switch from one camera to the other while keeping the destined target in sight. After switching from one camera to another, there is always a time interval of reorientation. Even if the user knows the camera numbers by heart and types in the correct number for the new camera, as described in Section 1.2.2, he does not know what to expect exactly in the new image. The cameras could be oriented in any direction.

To prevent this from happening, new interfaces have been designed that use previews or in-view images. Figure 2.4 shows how to look through a camera that is actually behind a wall. The original camera image is enhanced with an augmented image of the hidden camera. The 3D orientation between both cameras is taken into account, which makes it logical for the operator what to expect in the following camera image if he/she wishes to switch the spot monitor. This technique was introduced by Furmanski et al. [Furmanski 02] and was later introduced as *tunneled video* by Veas et al. [Veas 10].

The technique of tunneled video only handles camera images that can be viewed inside each other. The previewed camera has to be inside the viewport of the main camera and has to have a similar view direction. But there are more cases where this is not the case, but these cases cannot be solved by tunneled video. Cameras are set up to see a wide range of the environment. It is not likely that two cameras have a similar scope. An alternative technique is a mosaic approach. Multiple articles [Girgensohn 07, de Haan 10, Veas 10] conclude that a mosaic technique as in Figure 2.5 shows clearly which cameras are available in the neighborhood, based on their relative position to the current camera. It solves the issue for cameras being outside of the current viewport.

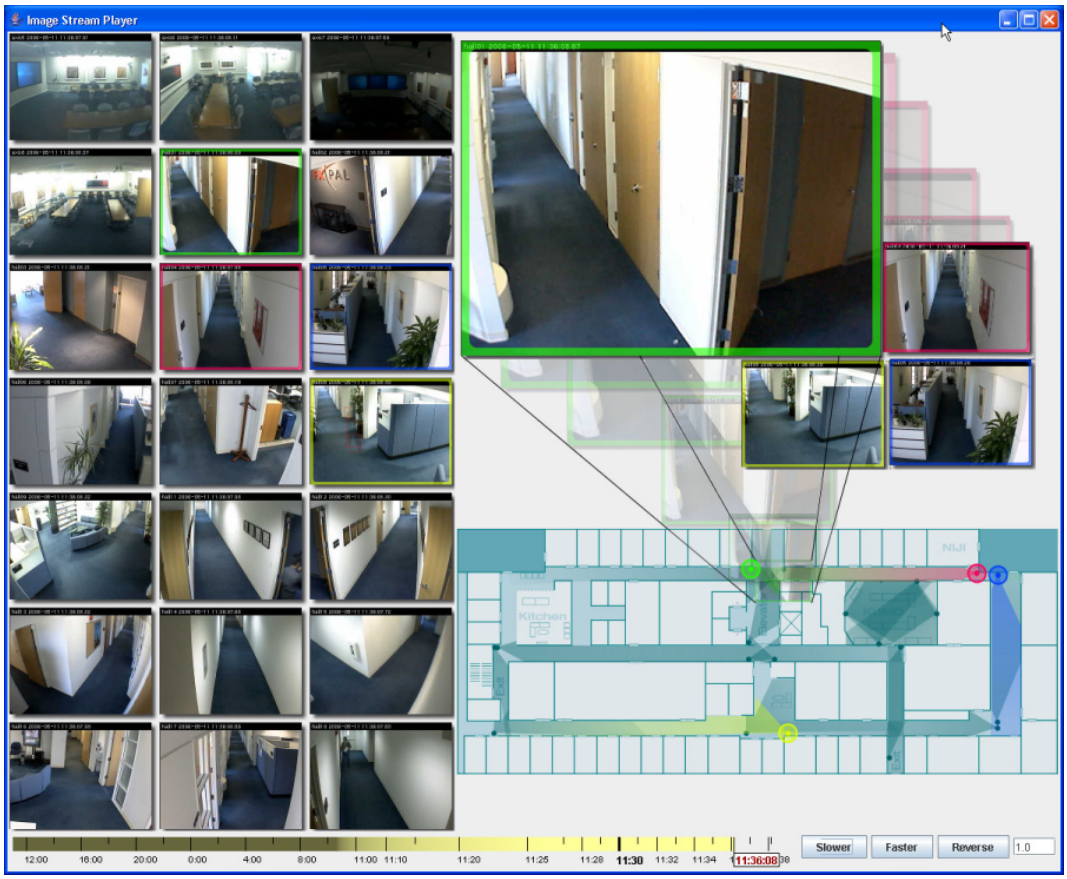


Figure 2.1: A map with associated cameras and video streams. The color codes reappear in borders of the view and indicators on the map. From Girgensohn et al., 2006.

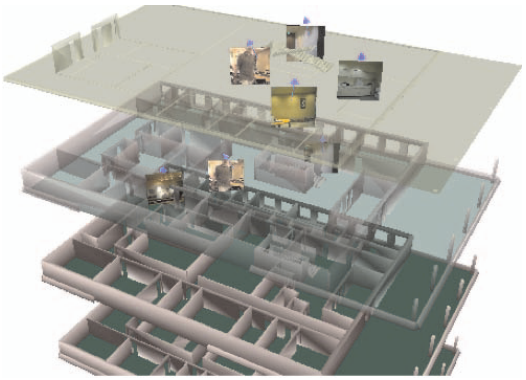


Figure 2.2: A multi-storey model with video embedding. The viewports are placed on billboard planes, which results in high situation awareness, but removes the orientation of the camera. From Wang et al., 2007.

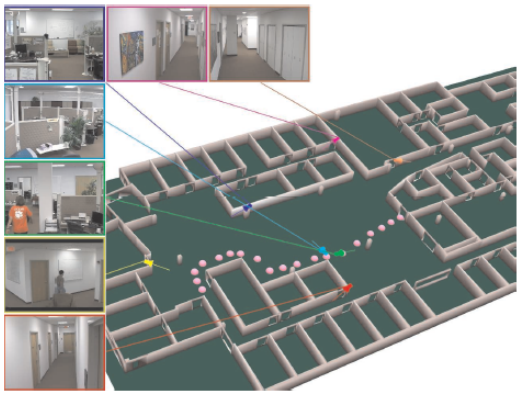


Figure 2.3: A 3D associated view. The images are associated with the positions of the corresponding cameras. From Wang et al., 2008.

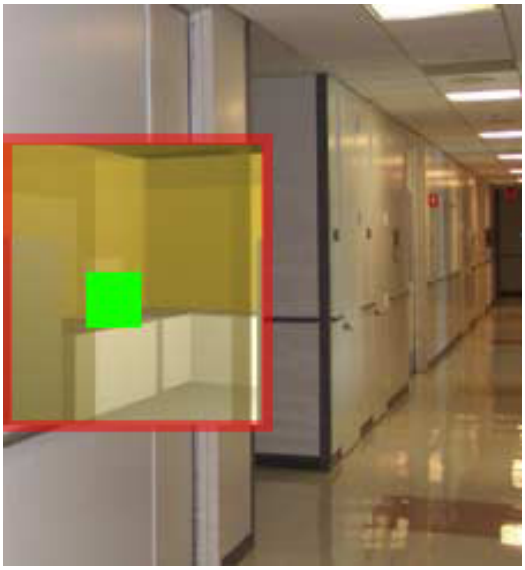


Figure 2.4: The wall on the left is marked as see-through. The user can see what is in the next camera view without actually going there. From Furmanski et al., 2002.

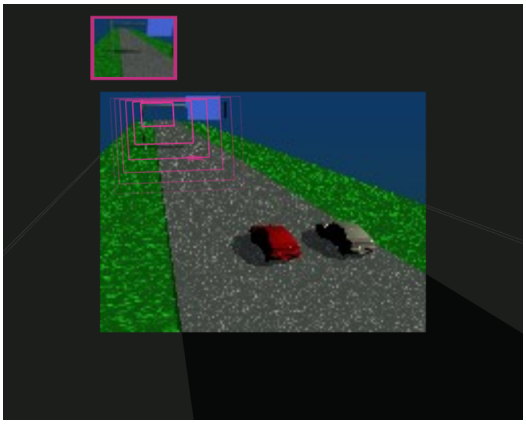


Figure 2.5: The mosaic view shows the current camera, but also cameras that are in the vicinity. The small image represents a preview of the camera that is somewhere in front of the current camera. From De Haan et al., 2010.

Instead of showing a preview image of the camera, it can also be represented by an object, or glyph. De Haan et al. [de Haan 10] show how to use arrow glyphs in a perspective view to indicate how to navigate to neighboring cameras. The cameras themselves are coupled via a graph to make sure that not all cameras in the scene are shown at the same time, but only the relevant ones. But as already shown by Girgensohn et al. [Girgensohn 07], the connections between neighbors can also be calculated on the basis of their view frustum, position and rotation. Figure 2.6 shows how the user is able to navigate to neighboring cameras by clicking on the glyphs in the screen.

The use of these glyphs was later extended to a dynamic cylindrical shape that could be moved over the ground. A 3D model of the environment was used as a basis to register mouse events on the ground. When the user confirmed to be on a certain *point of interest* (POI), the arrow glyphs that were already in the scene got attached to the cylinder. The user could now click on the desired camera. The system would also show a preview of that camera before transferring to there. Color association made it easier to separate all the different preview images on screen. Figure 2.7 shows the use of the cylindrical glyph. What is interesting about this technique, is the fact that it is not image based anymore. The user picks a point in 3D space and the system calculates the interesting cameras for that point.

Figure 2.6: The arrows shown in the image correspond to neighboring cameras. The user can click the arrow to fly to that camera.

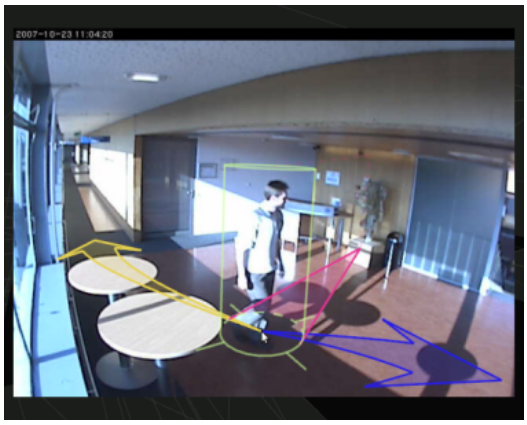


Figure 2.7: The cylinder glyph is placed on the ground place of a 3D model. When the user clicks, it shows the available cameras that have the point of interest in their field of view. From De Haan et al., 2010.

Camera Transitioning

In the current video surveillance management systems camera switches are implemented as direct changeovers. The new camera is instantly shown when the operator types in the camera number to go to on the spot monitor. This extremely fast transition lowers the level of situation awareness and lets the user reorient every time a switch takes place. We have already seen that previews can solve this problem partially, but there is still no physical transition that makes a relation with the real world and cameras among each other.

With the three techniques of mosaics, tunnels, and glyphs described above in Figure 2.3 come also new ways of transitioning. Instead of a direct replacement of the image, a smooth transition is made that gives a better understanding of orientation and navigation. Orientation because the new image comes slowly into focus, which gives the user time to settle down. Navigation because the new transitions show an animation where spatial knowledge is kept alongside. Figure 2.8 shows how both tuneling, mosaics, and glyph transitions can be animated over time from one camera to the other.

De Haan et al. [de Haan 09] describe three different ways of transitions: zooming, panoramic (i.e. strafing) and orbiting (Figure 2.9). These three types of transition have different approaches to keep situation awareness optimized and use different animation curves. The technique of orbiting, which is a special case because of the large distance between the cameras, was later improved by De Haan et al. [de Haan 10]. The transition was not animated in linear world space, but in linear screen space. This gives more ease and rest in the image and the user is able to keep focused on the point of interest (Figure 2.10).

The techniques described above have not been researched with any of the asso-

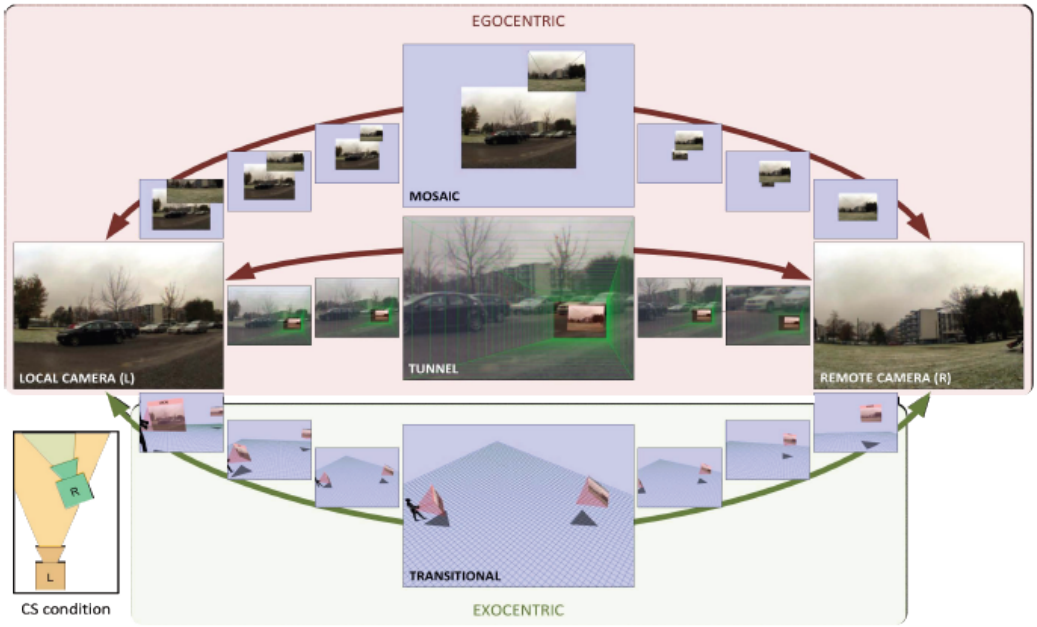


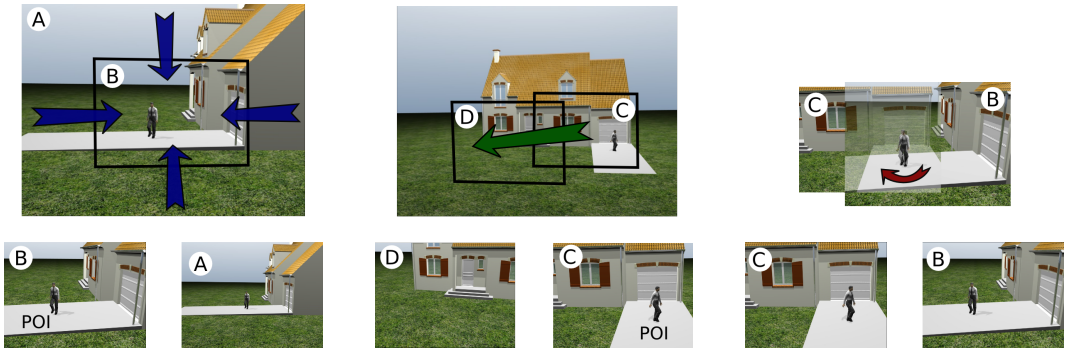
Figure 2.8: Three different transition techniques to get from one camera to another. By Veas et al. divided into egocentric and exocentric, while De Haan et al. describe all three techniques as egocentric. From Veas et al., 2010.

ciated or embedded videos yet. It does not seem to be plausible to do so either, because of the large difference in approach between the exocentric maps and egocentric transitions. Although all the above techniques look promising, there have been no attempts of integration with PTZ cameras yet.

2.1.2 Single PTZ Camera Controls

Active cameras are used in most surveillance systems nowadays, because of its high flexibility and wide range of view. The camera can rotate around 360 degrees and can narrow its viewport angle down to get objects up close. But so far, the studies performed on multi-camera interfaces have been focusing on static cameras, as active cameras also come with more complexity. The operator should be able to steer all three axes of the camera and does not always know how the camera is oriented. So both concepts of orientation and navigation return in a single PTZ camera as well.

The default control for PTZ cameras is the joystick (Figure 2.11). These are custom made joysticks that have a rotating stick as a third axis, which is often used for zooming. The joystick is beneficial for high precision steering commands, but also requires good skills to use it. Research on joysticks has mainly been performed in the field of teleoperations. Zhu et al. [Zhu 11a] designed smaller joysticks that were less unwieldy and required less effort than the usual joysticks. The joystick is similar to a half gamepad and showed that it gives at least the same results as the



(a) Zooming in: a view transition results in a narrowing of the view.

(b) Panoramic: a view transition results in a sliding view.

(c) Orbiting: a view transition results in a rotation of the view.

Figure 2.9: The three different actions as described by De Haan et al. From left to right: zoom, translation, rotation. From De Haan et al., 2010.

(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 2.10: Linear interpolation gives a strange result relative to the POI, but transfers in a more logical way in screen space. From De Haan et al., 2010.

Figure 2.11: A common PTZ joystick controller with extra function buttons that can pick the active camera to be controlled and set camera to preset orientations. From `http://worldeyecam.com`

Figure 2.12: A virtual PTZ joystick. Zooming is implemented as simple buttons on the right. The mouse is controlled within the left square. From `http://serialporttool.com`.

common joystick, while the ergonomics of the device have improved.

Also virtual versions of the standard joystick have been manufactured (Figure 2.12). They can be controlled by using the mouse as a stick. The third axis for zooming is often implemented with simple buttons, which degrades the precision and ease of use of the device. One advantage of the digital joystick is the possibility to easily upgrade the device, while hardware is always bound to its initial form.

Other options for hardware support include Wii remotes [Goh 08], eye tracking devices [Zhu 11b], haptic devices [Ott 06], speech recognition [Chen 07], and gestures [Iannizzotto 05]. All of which are still in an experimental stage and do not have the precision of the joystick or are cumbersome to work with. The operator is dependent on the hardware he/she is working with and these should therefore be reliable devices.

2.1.3 Automated Systems

The majority of the research in video surveillance is on automation. Research groups try to create systems where task load is minimized by taking away tasks from the user and assigning them to the system. Multiple studies [Krahnstoever 08, Shah 07, Singh 07, Yao 08] have created systems that could identify persons and follow them through the environment by tagging the persons. Figure 2.13 shows how persons in a subway station are labeled on three different cameras, based on image detection and spatial data. The information can be used to calculate which camera has to be shown to the operator in order for him/her to get a good shot at the suspect. A tracking system by Calderara et al. [Calderara 09] is also able to use path prediction. Based on already tracked persons, the likelihood of the path of a newly tracked person is calculated. The information could be used to calculate the next camera to show (Figure 2.14).

Figure 2.13: The targets are identified in three different cameras and labeled with the same number and color. The system contains a shared mental model. From Yao and Odobez, 2008.

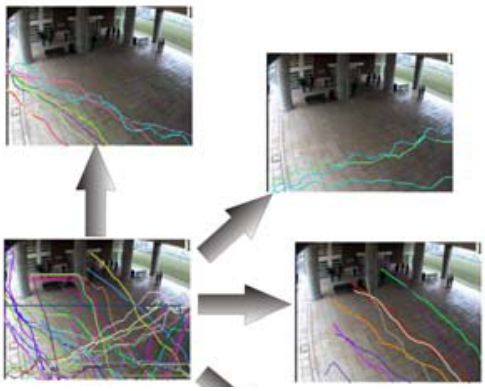


Figure 2.14: The system is given a dataset of paths. From there, the system learns predictive routes. From Calderara et al., 2009.

One research that does take PTZ cameras into account was performed by Qureshi and Terzopoulos [Qureshi 07, Qureshi 09, Qureshi 11]. They presented a system that let PTZ cameras to be steered automatically; each camera has its own target to follow. Whenever another camera could have a better view in the target, responsibility was taken over by the other camera. This is referred to as *camera hand-off*. The system could operate fully automated, which makes the user somewhat obsolete. Figure 2.15 shows how the responsibilities between three overlapping PTZ cameras are handled. The system only works in the current time frame and does not take any predictions into account; something that Kim and Kim [Kim 08] did include later on.

2.2 Sources of Inspiration

Besides from the scientific work described in Section 2.1, other sources of inspiration can be of use to create a new prototype for video surveillance interfaces. This section will provide a number of globally used software applications that show interfaces based on navigation through maps, documents, and 3D worlds.

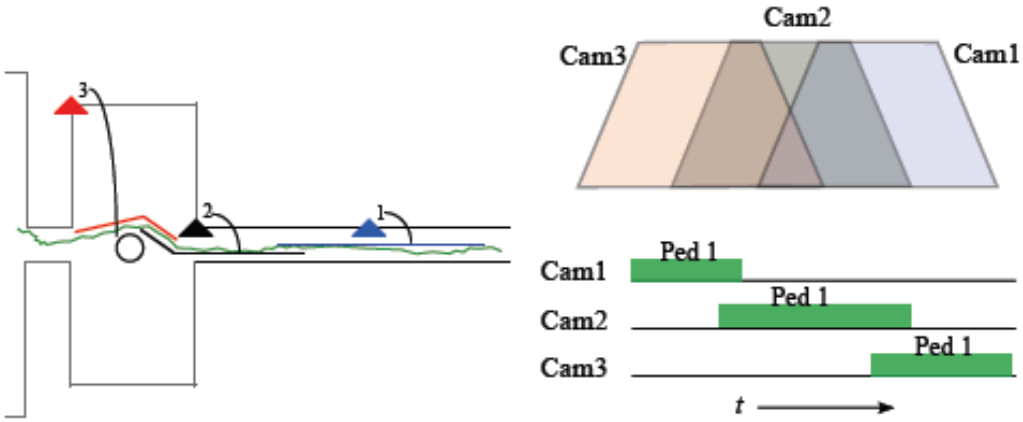


Figure 2.15: The cameras are able to cooperate on the scheduling of focus. Left: The part of a person and the visible parts in each camera. Right: The final scheduling of the target, while the cameras have wider overlapping viewports. From Qureshi and Terzopoulos, 2009.

2.2.1 Document Navigation

Common software applications for document reading use the mouse to interpret hand gestures. These gestures are then translated into document navigation to scroll the document around. This technique is also used in image editing programs.

The hand tool is an example that can be found in Adobe Reader[1]. The hand tool is used like an actual hand; holding the paper firmly will let the paper move in the same direction as the hand (Figure 2.16). The paper moves absolute to the mouse.

A contrary technique can be found in every program that uses a scroll bar (e.g. Microsoft Word, Adobe Reader, or Mozilla Firefox). By clicking the middle mouse button in the view, a reference point appears on the spot of the mouse. Moving the mouse pointer away from the reference point increases the speed at which the viewport moves around. In this technique (shown in Figure 2.17), the viewport is moved and not the document. The viewport moves relative to the mouse by increasing the speed vector of the viewport.

An often used zoom technique that focuses around the mouse is called *box zooming*. The user picks a point on the screen and stretches a box from that point. The resulting view is confirmed by releasing the mouse button. The technique is often found in CAD applications and image editing programs as shown in Figure 2.18.

2.2.2 Google Street View

One particular piece of software that has similarities to video surveillance is Google Street View (GSV), but whit a different utilization purpose. The application shows

[1]Part of Adobe Systems Incorporated

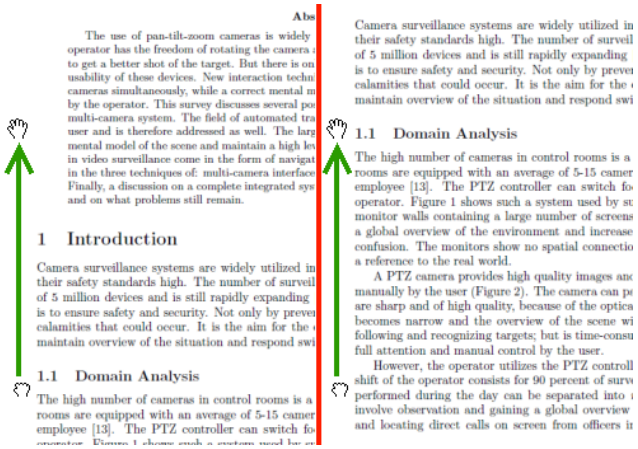


Figure 2.16: The common hand tool. Grab the paper and drag it around. The hand stays at the same place on the paper as it was initially grabbed.

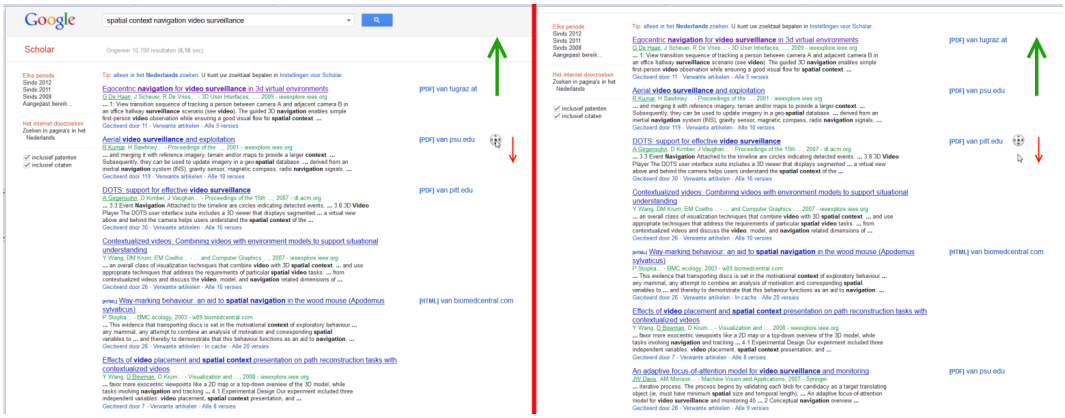
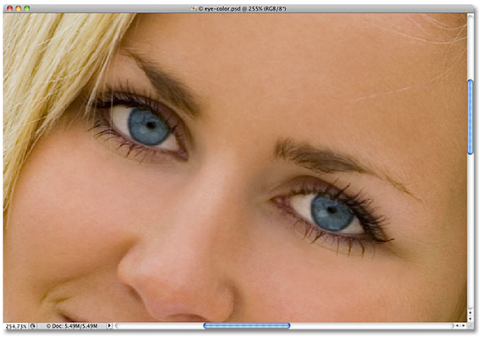


Figure 2.17: While reading a document, the middle mouse button is used to move the viewport around. The user picks a reference point on screen and moves the mouse away from that point. The viewport comes along with the mouse. Or in other words, the paper moves in the opposite direction.

(a) The user spans a rectangle from the upper left corner to the lower right.



(b) The system makes a new viewport that only shows the selected area.

Figure 2.18: The box zoom functionality lets a user choose an area of interest, which is then enhanced as the only visible area. From `http://www.photoshopessentials.com`

a way to navigate from camera to camera in a 3D world by means of mouse and keyboard. The images used by GSV are photographs in a panoramic bubble, which gives a similar look and feel as PTZ cameras.

Camera Orientation

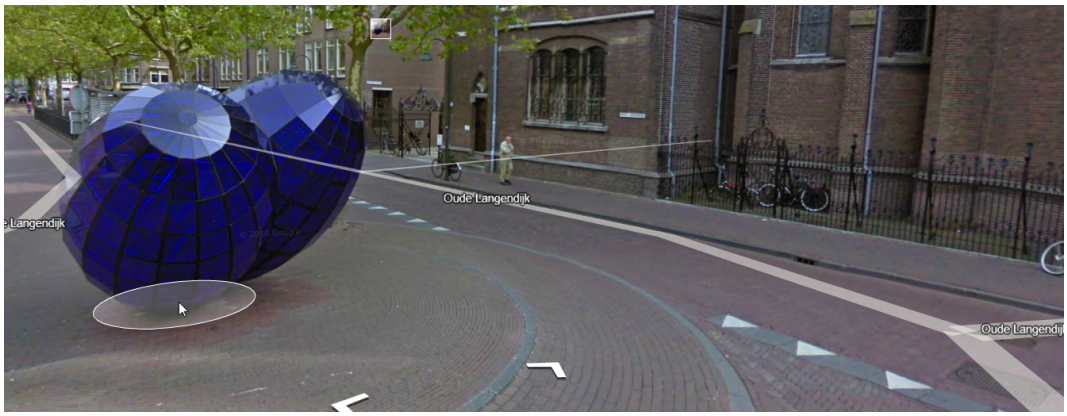
Navigating around by rotating in a single bubble feels (and in essence is) the same as rotating a PTZ camera around its axes. The technique used by GSV originates from the hand tool described in Section 2.2.1, but is extended to a 3D perspective. The user picks a point in space, which can be the ground or a wall. Moving the mouse will rotate the camera in a way that the mouse is still on that same position in space after movement has taken place. It feels as if you move the earth, instead of the camera itself. The control is focused on the point of origin, instead of the destination. An example of such a motion can be found in Figure 2.19.[2]

Multi-Camera Navigation

In order for the user to go from one place to another, GSV uses a circle that is placed on the ground surface of an underlying 3D model. By clicking on that position, the system will find the best possible panorama photo close to that point. The best possible camera is simply calculated by finding the closest panorama bubble in world space, which does not always give the expected result.

When GSV performs a transition to a new bubble, the rotation of the camera is maintained. So the user is still looking in the same direction after the transition. This gives good results on a straight road, but if a user would like to turn around

[2]For a reproduction of the images: `http://bit.ly/LW6adF`



(a) The user clicks and drags with the left mouse button to the right of the screen.

(b) The user releases the mouse button when the preferred rotation is reached.

Figure 2.19: Google Street View lets the user drag the environment around the current bubble. After dragging, the mouse is still on the same position in the environment. From `http://maps.google.com`

the corner of a street, the maintained rotation is useless. Figure 2.20 demonstrates how such a transition around the corner can give erroneous results.[3] A system with a higher workload than GSV could never permit to lose the target in this manner.

The user is not only able to click on the ground surface, but can also perform a similar action by placing the mouse on a wall segment of a building. The underlying 3D model calculates the normal vector of the wall and the circular cursor is transformed into a square for user feedback. When the user clicks on the segment, a different algorithm is used to calculate the best bubble. Again, the closest bubble to the point is picked. But since the system knows the normal of the wall, it can also calculate the rotation of the camera to look at the segment.

The bubble selection technique of GSV can be compared to the work by De Haan et al. [de Haan 10].

[3]For a reproduction of the images: `http://bit.ly/LyQSMs`

In that work, the cylindrical shape is present and available cameras are calculated with the use of the point of interest. However, GSV does not ask for a confirmation on where to go, but picks a new bubble by itself. GSV is designed for non-stressful situations, which makes it possible to just browse around the environment without being in a hurry. The advanced system of De Haan et al. cannot afford to lose track of the target.

Image Transitions

GSV is not a real 3D application, but a web application that provides algorithms to display panorama pictures. When performing a transition from one bubble to the other, it cannot actually leave the bubbles. The algorithm performs an interpolation from one image to the other. The transition is not precise and shows glitches during the animation. The advantage however is the egocentricity of this approach. The user does not leave the images and the interpolation does animate some kind of speed by skewing and stretching the image. Figure 2.21 shows a frame of the animation used in GSV.

Onscreen Information

Besides from the circular pointer on screen, a couple of other indicators are present to aid the user in its navigation task. GSV is used to display streets and is embedded in the Google Maps application. GSV has access to map data and uses this information to display an overlay on the panorama picture (Figure 2.22). The information is not accurate, but gives the user extra information about accessible areas. The data used by GSV has been recorded by a car, and it is therefore convenient for the user to click on the streets and not on the sidewalks. Doing so will make the expectations of the user more trivial, which is a good thing. Clicking in small corners will not give good results, because the chance that there is a bubble in that position is small.

2.2.3 Google Panoramio

Another service by Google is called Panoramio. This service makes use of GPS data among similar photographs. The technique is similar to the tunneling effect described in Figure 2.3. By moving the mouse over the photograph, different quadrangles appear where similar pictures have been taken. The GPS data is used to place the picture on the correct position and with the correct orientation. Extra images that do not overlap with the current picture are indicated by arrows on the edges of the screen. Figure 2.23 shows how a transition takes place within Panoramio to get closer to a building. Research on similar techniques have been performed by Snavely et al. [Snavely 08].

(a) The user is on an intersection and wants to take a left turn. The red rectangle indicates the position of the mouse cursor.

(b) Google Street View maintains the orientation after a bubble transition. This is an unwanted effect.

(c) The desired rotation of the camera should be similar to this view.

Figure 2.20: Google Street View does not handle rotation, since it only receives a position as input and cannot calculate the desired rotation. From `http://maps.google.com`

Figure 2.21: A frame capture of the light tunnel that Google Street View uses to animate the transition between bubbles.

Figure 2.22: Google Street View shows an extra overlay of the streets and shows clickable glyphs for the user to go to the nearest bubble down that street.

Figure 2.23: An image transfer in Google Panoramio. Left: Panoramio shows a quadrangle of the new image orientation and a clickable arrow that goes to another picture. Center: A transition is made by blended two images and shearing them over linear time. Right: The new image shows what was in the quadrangle on the left.

2.2.4 TNO Slim Toezicht

During the analysis phase, a visit was paid to S&T Vision. This company cooperates with TNO to form new video surveillance control rooms and user interfaces. Their focus is similar to the work of Girgensohn et al. [Girgensohn 07], where they use map structures and indicators to give operators more situation awareness. Their first design was manufactured for demonstration purposes in 2007 and can be found at Centrum voor Innovatie & Veiligheid[4]. The system makes use of the orientation of the cameras and displays the information on an interactive map that is used in combination with the old matrix view (Figure 2.24). The design is an enhancement on the current video surveillance systems.

CIV uses data from the industrial area Overvecht for the demonstration system. The environment contains only six cameras in an area of 1 km^2. This data can be used for real-life scenarios during the design process.

2.3 Approach

This section provides an initial approach for the design of the project. The designed solutions are a continuation on the research project supervised by De Haan and therefore takes the work of De Haan et al. [de Haan 10] as starting point for the design.

The system from De Haan et al. is a project that can be implemented in the near future and is a full aid for the user. There are no automated processes to take tasks away, but only new interface options. The current issues involve the complexity of the system and the visual feedback that should show how a camera is oriented and what the user will see in the next camera.

[4]Also known as CIV: `http://www.hetciv.nl`

Figure 2.24: An early version of the TNO Slim Toezicht project in association with S&T Vision. All video screens are color coded. The color codes are used to associate the images with points on the screen. Bottom left: A low resolution image of the final version that is currently active in Utrecht. The camera cones show the view direction of the PTZ camera.

Google Street View does not seem to have a problem with the expectancy by users, because the cameras inside the panoramic bubbles can freely rotate. GSV does not show any previews of the camera images or other feedback on where the user will go, but the density of bubbles in the area is so high that it is no real issue.

A logical next step would be to integrate PTZ cameras into a similar interface of De Haan et al., which they also indicate as future work in the paper. Some of the features should be redesigned to lower the complexity of the system. GSV is a perfect source of inspiration to do so, although there are still some unsolved issues in GSV as well. The complexity of the PTZ camera should be avoided by making a system that is driven by the point of interest and not by images and direct camera controls. The point of interest will be the focus of this design.

The system of De Haan et al. can be combined with PTZ cameras if it is possible to show the user how the camera is rotated and what it will show, and if the system can calculate a good rotation for the upcoming camera. The active cameras should still be able to be controlled manually. Evaluation research will show if it is still an option to use the joystick or if an integration with mouse and keyboard would be a better choice.

The design can be divided into three main components. The following chapters will elaborate on their algorithms and design. The following features will define the main focus of the system:

- **PTZ Camera Control**: Controlling a single active camera by some means of hardware manually.
- **Multi-Camera Navigation**: Picking a point in space and decide which camera has a better perspective on that view.
- **Inter-Camera Transitioning**: The way that the user receives feedback from the system on how he got from one camera to the other. This can be done by a direct switch, image manipulation, or by animating a flight from one camera to the other in world space.

Another important factor that has already been mentioned in Section 1.3 is the high cost of a surveillance system. Chapter 3 therefore describes how a simulator is implemented to lower the costs for user evaluations.

2.4 Discussion

The three subjects described in this section (i.e. multi-camera interfaces, PTZ camera controls, and automated systems) show the research fields that are currently active in video surveillance. Multi-camera interfaces have been discussed with both exocentric and egocentric fundamentals. The egocentric approaches keep a high level of spatial knowledge, which makes the user more maneuverable through the area. The exocentric approaches make good use of the total overview, which lets

users make correct replications of the situation over time (e.g. the user is able to reconstruct the path that the target walked through the area).

Control rooms are often equipped with simple joysticks. They provide a high precision mechanism to control the PTZ camera around its axes. Other experimental hardware options will not be ready for use in the short term. Software that is controlled by the common mouse and keyboard is simple to update. The mouse and keyboard are also well known by a large amount of the target group.

Automated tracking is an interesting concept, but is not flawless. The starting point of this thesis is to aid the user in their daily routines and especially in their tasks of pursuing and following. Taking the task away from the user will not aid them, but replace them. The techniques are still in an experimental phase, which makes that usage on the short term is unreliable.

Document reader software uses the mouse as prior navigation tool. Although the implementations of documents readers are made in 2D, a 3D representation is a small step. The PTZ camera only has two axes to rotate about, which means that a PTZ camera only has a 2D rotation. A mapping from the document reader tools to a PTZ interface is therefore one-to-one.

Google Street View shows an inventive way to navigate between 3D oriented panorama bubbles. It would be an idea to replace the static panorama images with live camera feeds instead. The navigation techniques are proven to be intuitive, because of the wide global use of the product. Although, it is not certain if the interface still holds in situations with higher stress levels. The extra onscreen information gives users an extra tool to maintain situation awareness. If the images are not clear enough for any reason, the user can also orient on the streets provided by the Google Maps overlay.

By creating a follow-up on the system of De Haan et al. and taking some techniques from Google Street View, a simple, intuitive and interactive PTZ camera system can be created. Especially mouse features are of interest, so that the gap between new and old gets larger for evaluations.

Chapter 3

Video Surveillance Simulator

Before going deeper into the main features of the system design, a basis is needed to build the system on. A major issue with PTZ cameras is the ability for testing. Fast high quality cameras are expensive, especially when you need more than one. Static cameras in a research project still have the possibility to use old video streams that run in sync, but PTZ cameras are dynamic and depend on the user's input. An alternative is required to have a dynamic test environment that is reusable. This chapter shows how to implement such a system.

A game engine will be used to implement a simulated environment with crowds and scenes. The characters will be simplified to minimalist characters that are still distinguishable from one another. The images can then be streamed to the interface in order to let the user interact with them. Using a simulator reduces all costs for camera hardware and specific surveillance systems and video managements systems. There are no difficulties with special interfaces for specific cameras, which is normally a time consuming job to implement.

The game engine can make the images more realistic by adding some post effects to the images that will correspond to real camera images. Examples of useful post effects are fish eye cameras, Gaussian noise, and blur. Figure 3.1 shows the kind of images that the implemented simulator provides to the new system and how post effects are used to improve realism.

3.1 Characters

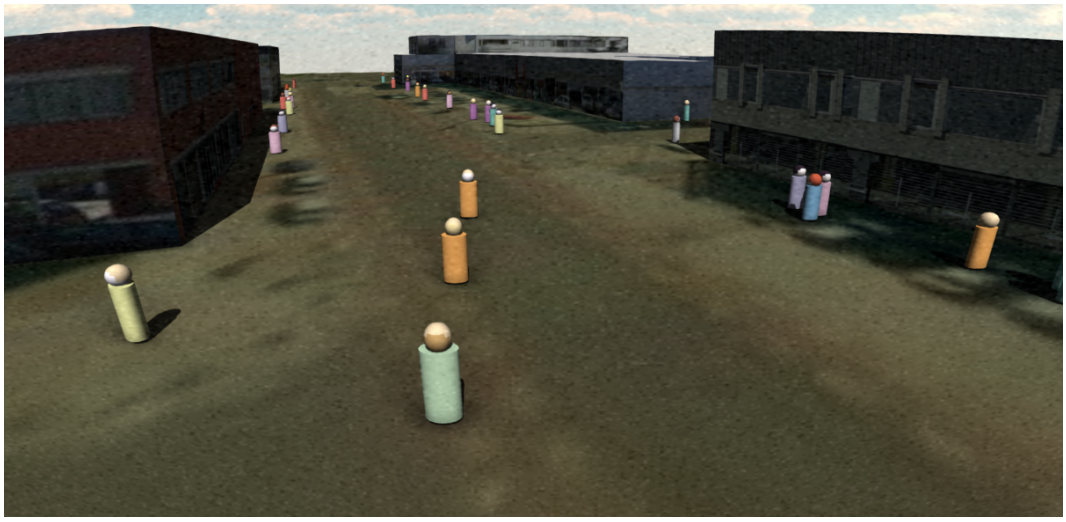
The persons walking around in the scenario are simple 3D models that consist of a cylinder and a sphere on top of it. They have separate materials for hair, face, and body. The three colors are standardized to specific diverse colors.

The skin colors are specified by means of a chart called the Von Luschan scale[1]. Colors 7, 13, 19, 26, and 34 have been used from Figure 3.2 to provide some variety in skin color. The hair colors are five distinct colors from an image that describes different hair colors in RGB colors for Adobe Photoshop (Figure 3.3). The body

[1] http://en.wikipedia.org/wiki/Von_Luschan%27s_chromatic_scale

(a) A clean image without any effects. The video stream looks unnatural for a surveillance camera.

(b) The same image as used in (a), but with a fish eye effect added to it. The center is enlarged, the sides are pushed towards the edges of the image.



(c) The final video stream that will be used by the interface for simulation. The image contains both fish eye and noise effects.

Figure 3.1: The simulator filmed out of one camera. The persons walking around are simple, but distinguishable. Top: Different stages of post effects. Bottom: The final video stream. These images originate from the new prototype itself.

colors are qualitative colors from the ColorBrewer[2] application. These colors are very distinct from one another. The used colors can be found in Figure 3.4.

All persons in the environment are able to perform actions as well. Instead of animating the person, a shiny star is summoned above their head (Figure 3.4). This was implemented to have more dynamics in the environment and can also be used to let the operator perform extra tasks.

The persons are also equipped with a path finding and collision avoidance algorithm. The algorithms are already implemented by the game engine and only require a navigation mesh to operate. The navigation mesh is calculated by the game engine as well, given a 3D mesh of the environment.

[2] `http://colorbrewer2.org`

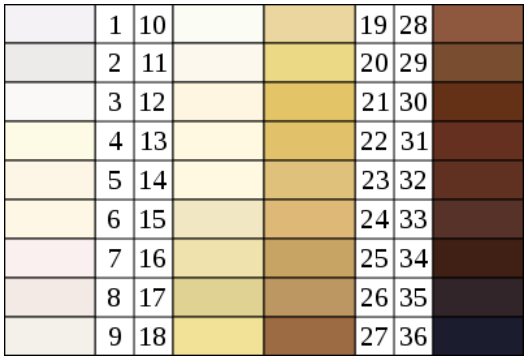


Figure 3.2: The Von Luschan scale indicates types of skin color from all over the world.

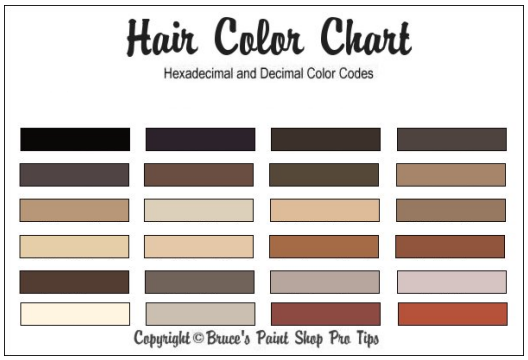


Figure 3.3: Diverse hair colors used by the simulator. Courtesy of Bruce Beard.

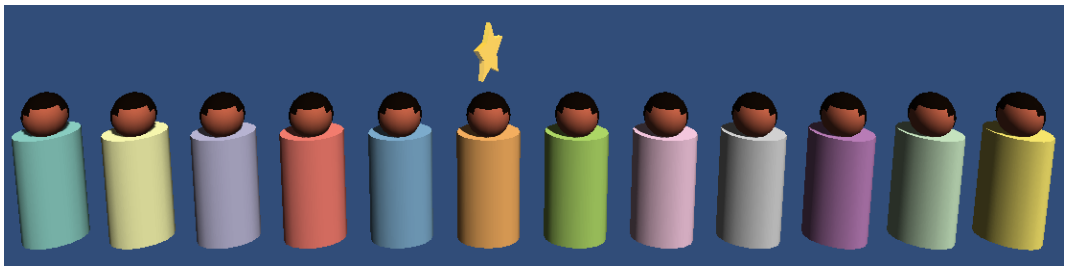


Figure 3.4: All shirt colors represented in the simulator. The colors are derived from the ColorBrewer application. The person in the middle shows a rotating star above its head to indicate that it is performing an 'action'.

3.2 Cameras

The cameras themselves are the interactive objects in the simulator. The user is able to communicate with the simulator via the cameras by rotating and zooming with the cameras and requesting the selected camera. The cameras are similar to real world cameras and have the same properties for speeds and angles. The camera can be constrained to an angular range, which is especially used to only tilt the camera 180°, instead of going upside down. The diversity of constraints and speeds can be found in Figure 3.5. The values used are derived from the Bosch AutoDome 800 Series HD PTZ Camera.[3] One thing that will not be implemented is the delay that is caused by the distance from camera to control room and the delay that is caused by the motors inside the PTZ camera. These forms of constraints are important to take into account when going towards a real system. But for now, direct responding cameras will have more benefit during evaluations. The costs of implementing motor delays costs time and there are higher priorities for first prototype testing that should be done first, before spending too much time into

[3]Instruction manual of the Bosch AutoDome 800 Series HD: `http://stna.resource.bosch.com/documents/Data_sheet_enUS_2474750603.pdf`

pan speed	1 – 200°/s
pan range	360° continuous
tilt speed	1 – 200°/s
tilt range	−90° – 90°
field of view	1° – 55°
zoom speed	60°/s

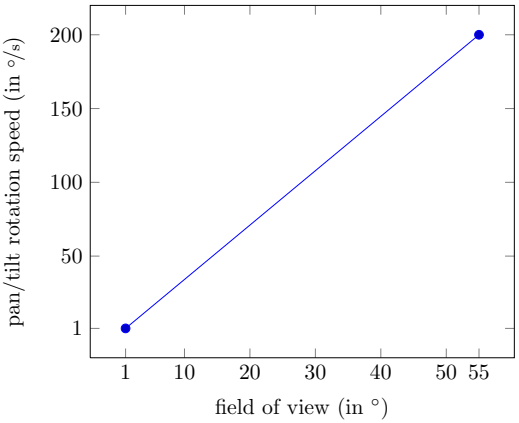


Figure 3.5: Specifications of the PTZ camera used in the simulator. Pan and tilt speed vary with the angle of the field of view. The graph shows the linear function between field of view and the rotation speed.

something that is not that beneficial in this phase of the research.

3.3 Implementation

The simulation is a feature that should be able to run by itself, only outputting images to use. That is, it should not be dependent on user input or implementation features that are especially designed for the prototype. The simulator should also be able to run with a classic matrix view. Therefore, an MVC model is a logical choice. Here, the model is the simulation and the system can run the simulator via an API. In a perfect design, the simulator would be a separate system that could run by itself only providing images through a network port or something similar. That would make it possible to also let other computers perform the rendering. But to avoid the high cost for building a perfect design, a local simulation is run inside the prototype. More details on the implementation of the overall system and the camera representatives can be found in Chapter 7.

3.4 Discussion

The high costs of real life video surveillance systems can be caught easily by creating a simulator. The simulator has a higher replay value, which means that users can perform the same scenario multiple times. There is no need to let a real person walk through the environment just for a test. One could also alter weather and lighting conditions or make special events happen, like fire hazards or car accidents. Other reasons for simulation involve privacy issues and the data size of the streamed videos.

The simulator used in this project can only be run locally, but networking facilities would make it possible to create a special simulation server that only focuses on delivering images to the interface.

Delays are not implemented into the system, because the time that it takes cannot be wasted on something that is not yet beneficial in this stage of the research.

Chapter 4

Single PTZ Camera Control

To be sure that the complete system has a good work flow, research has to be performed on the alternatives of the joystick. The joystick itself will also be taken into account as an option for the final prototype. Chapter 2 already showed that mouse and keyboard can be considered as an alternative with many possibilities in control.

4.1 Joystick

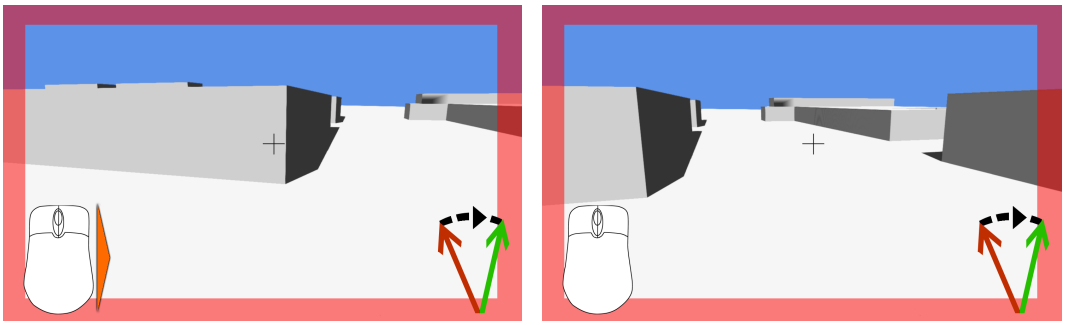
Currently almost only hardware joysticks are used in video surveillance systems to steer an active camera. The joystick is highly accurate and has three axes to steer the three axes of the camera. The hardware controller demands a lot of effort, because the stick to apply force on is long and requires all fingers. Zooming occurs by pushing against a rotational spring, which can be experienced as hefty.

Another downside to this device is the lack of focus on screen. There is no cursor that aids the user in fixation on the point of interest. A crosshair in the center of the screen would overcome this problem partially, but the camera should therefore always be able to rotate in complete freedom. The user cannot simply point to something interesting.

4.2 Mouse Alternatives

This section shows three different interfaces that use the mouse to control the camera, which are inspired by the documentation provided in Section 2.2. The techniques for zooming are discussed separately, since they can all be applied to the mouse techniques as separate modules.

The mouse is an effective tool for interactive software, because the focus point can be controlled in absolute ways, while the joystick can only apply relative forces to an object.



(a) The red border indicates that the mouse is locked and the camera will be moved instead of the mouse.

(b) Moving the mouse to the right will also let the camera rotate to the right.

Figure 4.1: The shooter game control maps the movement of the mouse directly to a rotation of the camera. The point of focus is always centered on screen. The orange arrow shows the direction of the mouse. The arrows on the right show the rotation that the camera makes. The icon inside the mouse is the icon displayed as mouse cursor.

4.2.1 Shooter Game Control

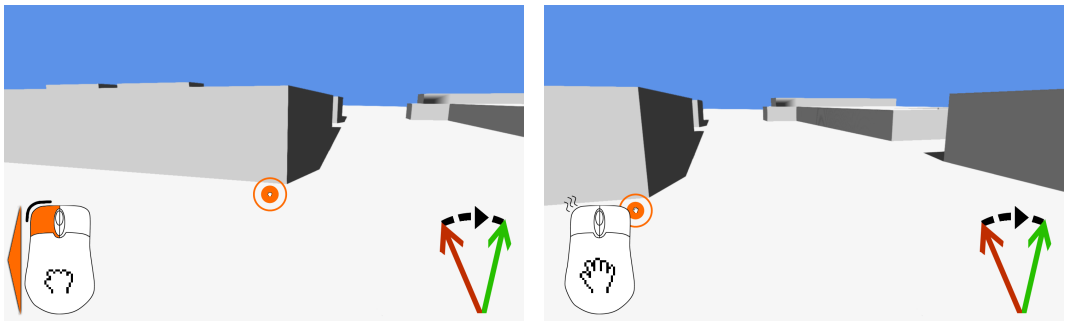
One could imagine a camera as the head of a person. This is often done in first-person shooter games. The controls in these games are simple mouse movements. The controls are absolute, which means that keeping the mouse still will also keep the camera still.

The controls are clear and simple and people that play games surely know how to work with this interface. This technique has a similar downside to the joystick interface; the point of interest is in the center of the screen. To overcome this issue, the point of interest can be unlocked by pressing the spacebar. But doing so will stop the camera from moving. There are two different states where the user can be in; the steering mode and the focus point mode. As a visual feedback on the difference in the two states, the image is provided with a large red border when controlling the camera as seen in Figure 4.1.

4.2.2 Hand Tool Control

The next control is similar to the control discussed in Section 2.2.2 that was designed by Google. The user is able to grab the world and rotate the camera by moving the mouse. After rotating, the mouse is still on the reference point in the world. Figure 2.19 in Section 2.2.2 already showed how this was achieved in GSV. Figure 4.2 explains the algorithm in an illustrative way on how the tool was implemented in the prototype.

In addition to the solution provided by Google Street View, an inertia factor was added to give the camera a smoother feel. The user can give the camera a small whip to have it move in a certain direction. The interface is designed to use a focus



(a) The user grabs a position in the world with the left mouse button.

(b) The user releases the mouse button and the cursor opens the hand. The rotation is finished.

Figure 4.2: The hand tool control lets the user grab the environment and drag it around. The cursor keeps pointing to the same position in world space. The orange arrow shows the direction of the mouse. The arrows on the right show the rotation that the camera makes. The icon inside the mouse is the icon displayed as mouse cursor.

point, which is in line with the focus of this research. Disadvantages of the algorithm have to be found during user evaluations.

4.2.3 Document Viewer Control

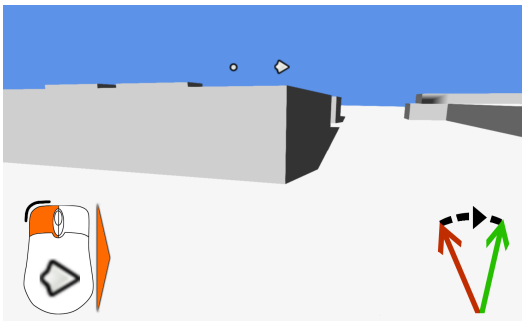
The opposite behavior of the hand tool can be achieved with the middle-mouse tool described in Section 2.2.1. Instead of altering the ground, the viewport moves along with the mouse. Usually, the left mouse button is used for text selection, but since there is no need to select text, the left mouse button is used as movement button.

Whenever the user clicks and drags with the left mouse button, a small dot is shown on the clicked screen position. This is the reference point for the user and is used to calculate the acceleration, depending on the mouse distance to that point. The mouse pointer will change in an arrow that shows the acceleration direction of the camera. Pulling the mouse farther away from the point will increase rotation speed to that direction. The steering direction behaves opposite to the hand tool control. Figure 4.3 shows how the tool is used to rotate the camera and how it differs from the hand tool.

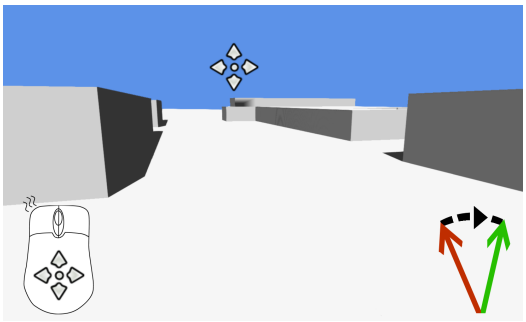
This technique has the same properties as the hand tool control and will probably also have similar advantages and disadvantages.

4.2.4 Zoom Controls

Panning and tilting can be controlled by moving the mouse around, since it has exactly two axes, but there is no third axis on the mouse by simple hand movement or rotation. However, most mice have a scroll wheel that can be used for this purpose. Scrolling the wheel can be directly mapped onto changing the field of view of the camera. Using inertia will make that process smooth for the user.



(a) The user grabs a screen position with the mouse. A dot appears on that position and the user drags the cursor to the right from that point. Acceleration is given towards the direction of the small gray arrow cursor.



(b) The user releases the mouse button and the cursor shows a compass cursor, indicating that the camera is still decelerating.

Figure 4.3: The document tool uses a relative control scheme by adding an acceleration vector to the camera relative to the point indicated by the user. The orange arrow shows the direction of the mouse. The arrows on the right show the rotation that the camera makes. The icon inside the mouse is the icon displayed as mouse cursor.

Secondly, the zoom could also occur towards the mouse cursor on screen. This technique is often used in document viewers as well. The downside is that zooming then also alters pan and tilt levels of the camera in order to center the mouse cursor. Doing so could confuse the user, since the axes are no longer separate control parts.

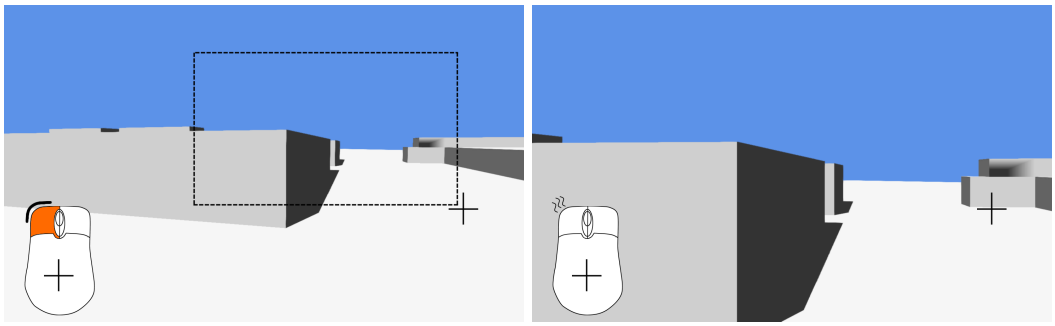
Similar to the second option is the box zoom technique explained in Section 2.2.1. From a rectangle drawn by the user, the viewport can be calculated and the camera can be oriented to exactly fit that viewport as shown in Figure 4.4.

Scrolling the wheel and box zooming can exist next to each other as long as there is a free button to use for the box zoom. For a simple implementation, one could first press ‘B’ on the keyboard and then draw the rectangle.

4.3 Discussion

The controls that use the mouse instead of the joystick will have more freedom in creating an overall system that is based on De Haan et al. The mouse is flexible to use and has different modes by means of holding a certain mouse button. One could also steer the mouse pointer with the joystick, but that has been proven to become unhandy. For example, game consoles almost never use a mouse pointer to control the menu. Also, combining mouse and joystick will cause mode errors, because the user has to switch hardware and can therefore be holding the wrong device.

In the final implementation, camera controls will be controlled with the left mouse button and evaluations will show which of the implementations is preferred by the target group.



(a) The user drags a box from the upper left of the screen to the bottom right of the screen. A selection box is drawn to indicate the area to zoom towards.

(b) The selected area is fit inside the viewport once the user releases the mouse. The zoom mode is left, the user can perform other tasks with the mouse again.

Figure 4.4: The box zoom functionality lets the user pan, tilt, and zoom with only one gesture. The user indicates a screen area of interest. The camera tries to fit its viewport around it. The icon inside the mouse is the icon displayed as mouse cursor.

Chapter 5

Multi-Camera Navigation

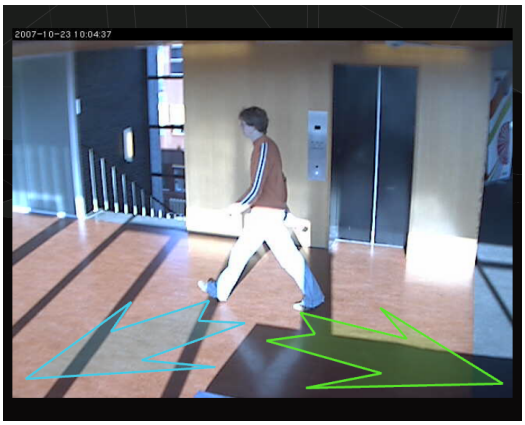
The solutions for single cameras provided in Chapter 4 do not solve the scenarios described in Section 1.2 to navigate through multiple cameras. Whenever someone walks in screen and then disappears behind a wall, there is no way to see that person with the current camera. The user needs to switch its view on the spot monitor to another camera that is able to see behind the wall.

This chapter first elaborates on user input, then continues on which cameras to show to the user, in what ways the user can control the system, and finally on what visual feedback is provided to aid the user in its choices.

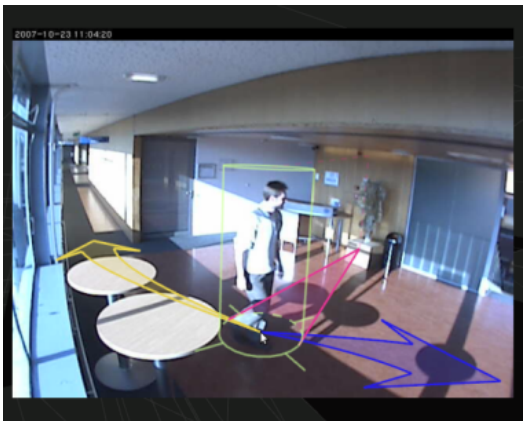
5.1 Background

Since this project is targeting for a tool to aid the users in their tasks, a new interface is designed to let the operator make their choices swift and determined. The current system that is used in most control rooms lets the user pick a number and the image on the spot monitor changes. But changing the camera is not always enough, since the cameras used in the system are PTZ cameras that are always oriented in a default rotation. It is often the case that the operator picks a camera and subsequently has to change the orientation of the camera before getting a good shot and retrieving the target back on screen. Camera switching is only image based and has nothing to do with spatial knowledge or spatial relationships between cameras. The operators only rely on their knowledge of the environment.

The system by De Haan et al. [de Haan 10] shows two features to get from one camera to another by means of spatial selection: Clicking on projected glyphs in space, and selecting a point of interest on the ground and clicking one of the connected glyphs that show up (Figure 5.1). The first technique is useful for cameras that do not have overlap in their view range. This is often the case in environments that only have static cameras installed. An underlying graph is used to manually calibrate the system for neighboring cameras, which takes time to install per environment. The use of PTZ cameras alone makes the use of projected glyphs obsolete. PTZ cameras can rotate around in 360 degrees, which implies an overlap with all



(a) The arrow glyphs in the image are clickable and correspond to the neighboring cameras.



(b) The cylinder attaches neighboring arrows that have the point of interest in their viewport. The user selects one of them by making a direction gesture towards the desired camera and releases the mouse button.

Figure 5.1: The two techniques used for camera selection by De Haan et al. Both figures show that the glyphs used can be drawn in perspective to the actual world.

other cameras in the scene. With this assumption, the focus can be put solely on point of interest-navigation.

This new design uses user input, position and direction of the target, to filter and calculate interesting cameras in the neighborhood. It is up to the user which camera is the best to look at. So the system aids the users in their choices, but does not take over the full task of automated following and targeting. A confirmation from the operators is always required by the system before changing the spot monitor.

5.2 User Input

The cylinder cursor designed by De Haan et al. is reused in this project for user input. It represents a human target to follow and is useful for interaction with the operator. It has the size of a person and is used as an overlay on the camera image, but in real world perspective. The user indicates the point of interest with the cylinder, but actually indicates a small area of interest. The new design spans a collision model over the cylinder that will later be used for visibility calculations.

When a person is walking or running in the environment, it is often not enough to only know the position of the target. Therefore, a second input parameter of direction is required to make better calculations in what camera is interesting. Section 2.1.3 already showed that automated tracking software makes a good effort on indicating the walking direction of the target. The human eye is even more advanced in estimating the direction of a walking person by just looking at the video screen. And since the user has already indicated the POI, it should not be too much work

to also give a direction to the cylinder by means of a gesture.

Section 5.4 shows how these controls work in detail. For now, the input parameters of position and movement direction are defined to use by the system as input. The input is used to filter the cameras in the environment by means of a metric. That score is then used to give feedback to the user on which cameras are probably interesting.

5.3 Camera Filtering and Metrics

The number of available cameras on any point in space is significantly bigger within an environment with active cameras than in one with static cameras. But that does not imply that all cameras can see all points. There could be occluding objects that make it impossible for a camera to see a certain point. To avoid that occluded cameras are shown as a logical option for the user, a filtering should be performed to only include good cameras.

There are three criteria to filter out the cameras that are not of interest: distance, visibility and direction. All three filter options are elaborated and refined to an algorithm in this section. The filtering calculations will use a metric score and weigh the three measures per camera. The user receives feedback on the final metrics of the camera in a direct way and will be able to make a choice instantly.

5.3.1 Distance

A camera that is close to the POI is assumed to be a good camera to pick. The operator is able to see the target up close while maintaining a large field of view (Figure 5.2). The metric for distance to the POI is normalized between the farthest and the closest camera to the POI. Every camera receives a metric score from 0 to 1 with the following formula found in Equation (5.1). The scores are later used in the overall weight calculations.

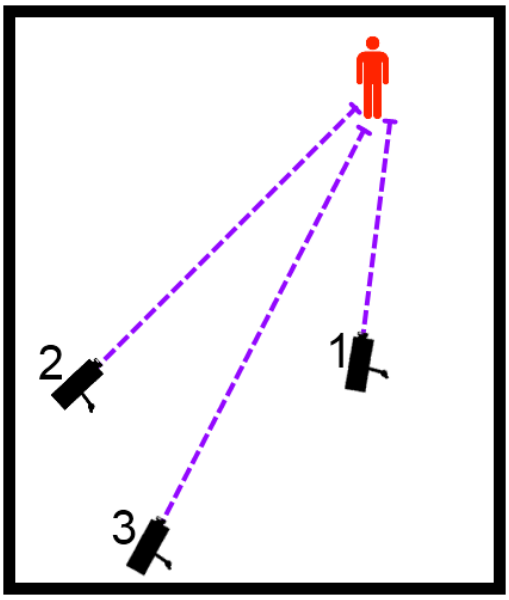
$$p_{dist}(\boldsymbol{cam}, \boldsymbol{poi}) = 1 - \frac{d(\boldsymbol{cam}) - d(\boldsymbol{min})}{d(\boldsymbol{max}) - d(\boldsymbol{min})} \tag{5.1}$$

Where min and max are the closest resp. farthest camera positions in the environment to the POI and d is the distance function:

$$d(\boldsymbol{c}) = \|\boldsymbol{c} - \boldsymbol{poi}\|$$

5.3.2 Visibility

The second measure takes the obstacles in the world into account. For this technique to work, a 3D model of the environment is required, which reveals the occlusions by sampling rays in the collision model. If one would only look to the POI itself,



#	distance score (0–1)
(closest) 1	1.0
2	0.4
(farthest) 3	0.0

Figure 5.2: Three cameras looking at the same target. The scores per camera are listed in the table on the right. The camera that is closest receives the highest score, because that camera can get a better closeup of the target and has a better future for zooming in at the target. The camera that is farthest away receives the a score of 0.

one ray would be sufficient. But that would not tell much about the visibility of the target, since a person or any other object is bigger than a single point. The operator is often interested in a small area instead. In this case, the cylinder.

Points in the collision object shown in Figure 5.3 are uniformly taken and rays are shot from the camera to the target points. Then, the number of hits back on the cylinder is counted and the number of rays that hit something else or even nothing. The rays that hit the cylinder indicate what the camera can see of the cylinder. This method is not completely accurate, but is fast and gives a good estimate of the visibility of the area of interest. The ratio of good against incorrect rays can be expressed as a normalized metric between 0 and 1 with the following formula found in Section 5.3.2.

Normally, the user is not only interested in the position of the target, but also in the future path of the target. To take into account any short term future positions, the direction indicated by the user is used to span a larger collision object similar to the one in Figure 5.4. By sampling rays in this larger collision object, cameras with less occlusion with respect to the future direction of the target score higher. So taking the future into account does not change the algorithm, only the area of interest. This method could be improved by taking other samples from the collision model that are uniformly spread over screen space, instead of world space.

Algorithm 1 The visibility is calculated by performing raycasts to a random point in the collider and taking the percentage of rays that is not obstructed by other objects.

$$p_{vis}(\boldsymbol{cam}, col) = \frac{\sum hit}{n}$$

Where n is the number of raycasts and a hit is calculated as follows:

$$hit = \begin{cases} 1 & \text{if } raycast(\overrightarrow{ray}) = \boldsymbol{r} \\ 0 & \text{otherwise} \end{cases}$$
$$\overrightarrow{ray} = \overrightarrow{\boldsymbol{cam}\ \boldsymbol{r}}$$
$$\boldsymbol{r} = \begin{bmatrix} U(x_{min}, x_{max}) \\ U(y_{min}, y_{max}) \\ U(z_{min}, z_{max}) \end{bmatrix}$$

Where $\boldsymbol{r}$ is a random vector inside the collider col and $\overrightarrow{ray}$ is the direction vector from the camera $\boldsymbol{cam}$ to $\boldsymbol{r}$.

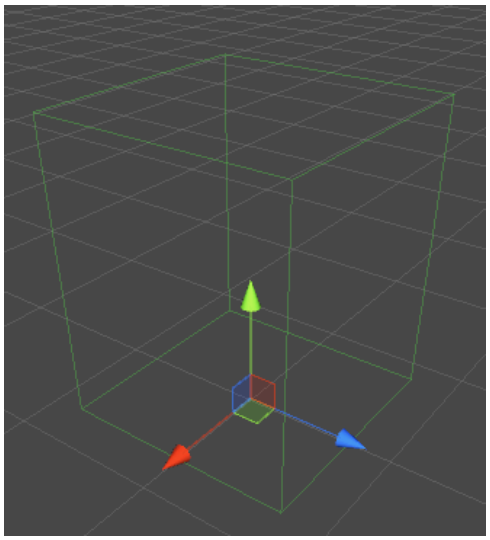
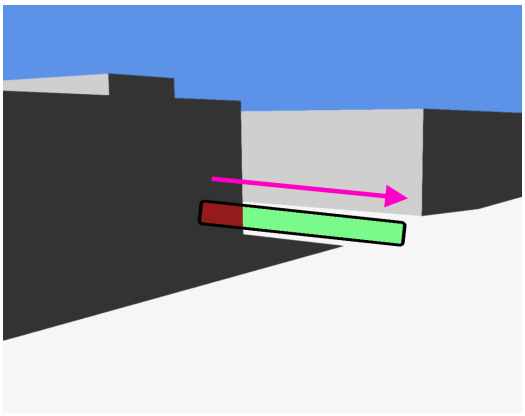
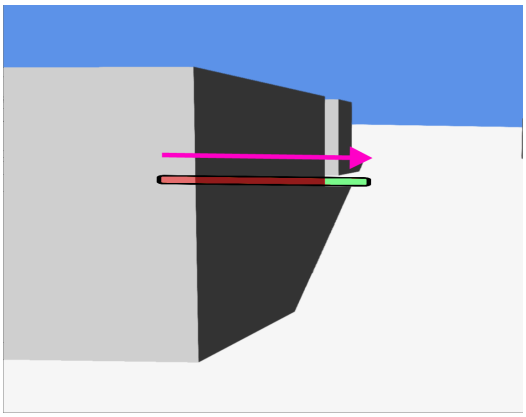


Figure 5.3: The bounding box of the point of interest. Dimensions: $1.5m \times 1.5m \times 1.8m$.



(a) Sampling rays onto this area will return a visibility of around 70%.



(b) Sampling rays onto this area will return a visibility of around 20%

Figure 5.4: The visibility of every camera is calculated by sampling rays onto the user selected area that indicates the movement direction of the target. The arrows indicate the gesture that the user has made.

5.3.3 Angle

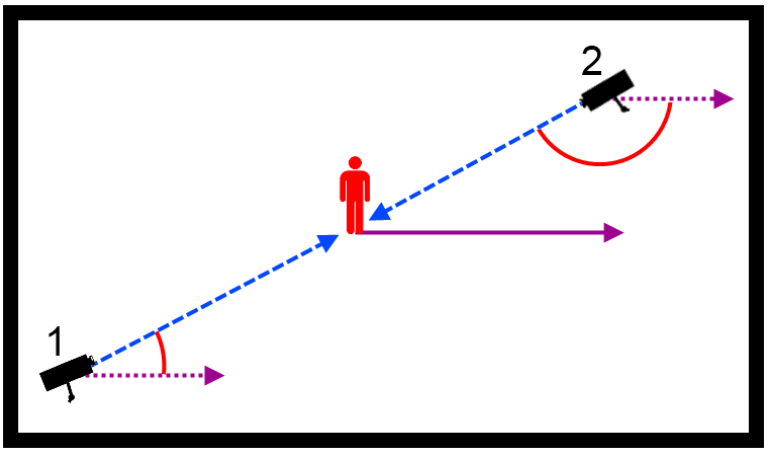
The operator is often not only interested in seeing a target, but would also like to influence from which side the object is viewed. Given that the task of the operator is to pursuit a target, it is likely often desired to view the person from behind. Viewing a person from the back makes it easier for the operator to anticipate on quick actions by the target and implies that a future camera is easier to access, since it is in the field of view.

The angle between the target's direction and camera can be used to calculate a metric for the viewing direction. Aligning the camera parallel to the target's direction will give a high score, while aligning the camera in the complete opposite direction will give a low score (Figure 5.5). To exclude the height of the camera, the angle is calculated in the XZ-plane only. The formula to get a normalized metric score for the angle is depicted by Equation (5.2).

$$
\begin{aligned}
p_{ang}(\boldsymbol{cam}, \boldsymbol{dir}, \boldsymbol{poi}) &= 1 - \frac{\theta_{d,dir}}{\pi} \\
\theta_{d,dir} &= \arccos(\hat{\boldsymbol{d}} \cdot \hat{\boldsymbol{dir}}) \\
\hat{\boldsymbol{d}} &= \frac{\boldsymbol{poi} - \boldsymbol{cam}}{\|\boldsymbol{poi} - \boldsymbol{cam}\|}
\end{aligned}
\tag{5.2}
$$

5.3.4 Final Metric Score

Given the above three measurements, a metric score for each camera is calculated. This is achieved by weighing the three measures to come to a final score. If the



#	angle score (0–1)
1	0.8
2	0.2

Figure 5.5: Two cameras that look towards the target that goes to the right. The camera that is best aligned with the direction vector receives a higher score, because the user expects to get behind the target. The figures in the table on the right give an estimate of their respective scores.

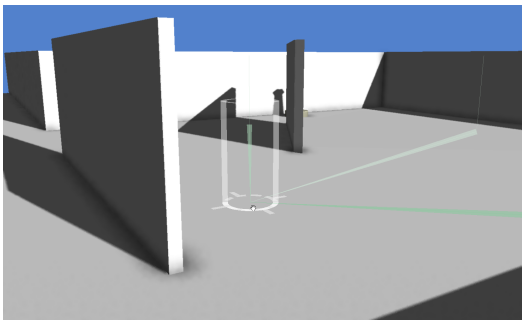
visibility measure would be more accurate, it could have been used to filter out cameras that are totally occluded. Since this is not the case, visibility is also taken into the overall weighting factor.

$$p = w_{dist} \cdot p_{dist} + wvis \cdot p_{vis} + w_{ang} \cdot p_{ang} \qquad \text{with} \sum_{x} w_x = 1$$

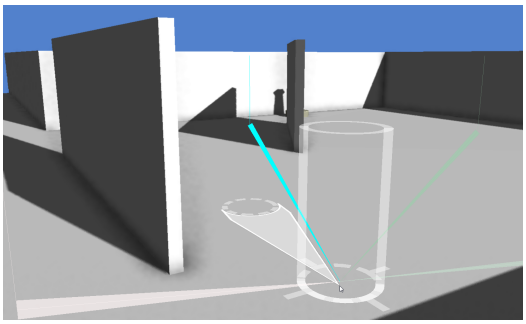
A grounded reason for the chosen weights may not be valid. Experimental testing resulted in the following values as found in Table 5.1. The value for visibility and angle are high. The high visibility weight is useful to filter out cameras that are behind walls or other occluding objects. The high angle weight is required to make the direction measure do its work. Leaving this value low would result in high scoring cameras that look in a complete opposite direction. The low weight for distance is fine, because the cameras have zoom options to bring objects closer and have no real hinder of being far away. These weights were tested in environments between $20m^2$ and $1000m^2$ and gave similar results in means of giving the best expected camera the highest score. One could tweak the weights to their own preferences while using the system.

w_{dist}	w_{vis}	w_{ang}
0.172	0.420	0.408

Table 5.1: Weight scores defined by experimental gathering.



(a) The user selects a point to start the arrow to drag.



(b) By holding the right mouse button, an arrow shows up. The user drags the arrow in the direction the target is walking (i.e. in the direction he wants to look).

Figure 5.6: By dragging an arrow from the point of interest, the user inserts position and direction into the system. The algorithm instantly calculates the winning camera. Releasing the right mouse button will initiate the transfer to the camera marked in cyan.

5.4 Interaction

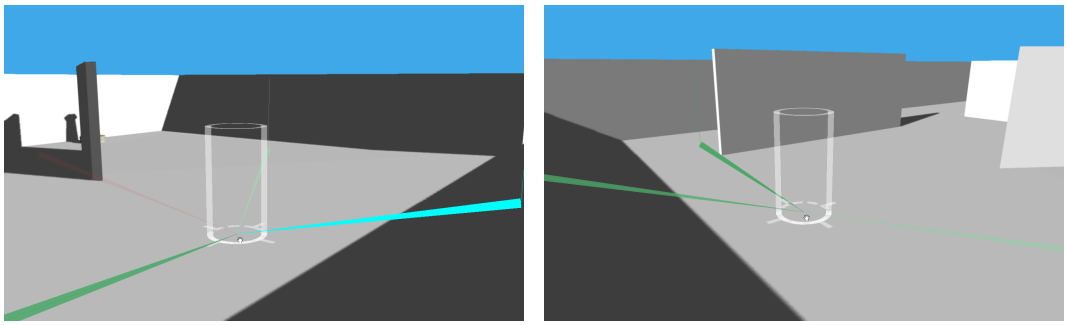
While working with the system, the user is able to perform one out of three actions. All actions result in a switch of camera, but use different input parameters for the calculations of the filtering system.

5.4.1 Direction Gesture

The first action takes all three parameters of visibility, direction and distance into account. The user picks a point in space with the cylinder and confirms the position by holding the right mouse button. At that moment, a directional arrow is drawn from the initial position to the mouse. When the user moves the mouse, the size and direction of the arrow move along. The action can be confirmed by letting go of the right mouse button, which will initiate the camera transfer. The action can also be canceled by pressing the left mouse button while holding the right mouse button. This input method is illustrated in Figure 5.6. The camera that will be chosen by the system is depicted in cyan.

5.4.2 Direct Jump

The user can also restrict the filter calculations to only distance and visibility; the user does not care from which angle the target is viewed. Whenever the user performs a double click with the left mouse button, the angle parameter is left out of the scope, since there is no direction vector provided by the user. The dedicated camera will perform a rotation to look at the POI and the spot monitor will switch camera. This user input is very quick, because the user only has to perform two fast clicks after one another.



(a) The user selects a point to double click on. Double click will only work if one of the arrows is colored in cyan, indicating a winning camera.

(b) The new camera focuses the point of interest in the center of the screen.

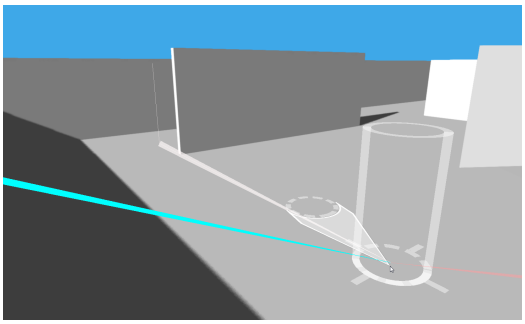
Figure 5.7: Double clicking on the point of interest while a cyan arrow is shown is faster than dragging an arrow. It costs less effort and can still give the desired expected results.

In this mode, the user is not able to pick a camera by hand, but the camera is automatically chosen by the system. To avoid confusion, interactive real-time user feedback has to be provided before the double click takes place. Figure 5.7 illustrates what to expect from this method.

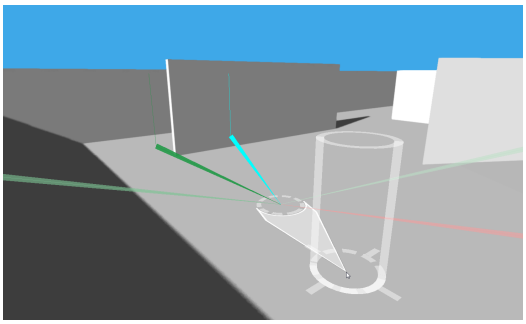
5.4.3 Free Selection

One could imagine the user is not always satisfied with the choice that the system makes, because he/she would like to see something from a specific point of view. This can be achieved by holding the 'Shift'-key while performing the right mouse button drag method. When doing so, only the angle metric is taken into account. The visual feedback that the user gets shows a disc of all choices of all cameras. Since visibility is not used as a scoring factor anymore, all cameras in the environment are shown to the user. By letting go of the right mouse button, while holding shift, the highlighted camera is selected.

The issue with this approach is that the user often does not know that the selected camera is almost completely occluded. By providing more options to the user the system becomes more complex. The issue could be solved by showing some kind of preview of what the camera will see. But doing so would imply that all cameras need to rotate towards the POI, which is not desired. For instance, when the cameras are recording other events at that moment. Controlling all cameras at the same time is prone to conflicts. Another idea is to cache a preview that will be showed to the user instead. This optimization was not implemented in the system, because of time issues and the low number of cameras per environment that were used for testing. Figure 5.8 shows how this method differs from the normal right mouse button drag.



(a) Dragging an arrow with the right mouse calculates camera scores by visibility, angle and distance.



(b) Dragging an arrow with the 'Shift'-button and the right mouse button calculates scores by angle only.

Figure 5.8: By holding the 'Shift'-button during a right mouse drag, all cameras show up as a disc. The user can directly pick which camera is desired by drawing the arrow in the same direction as the colored line.

5.5 Visual Feedback

As already stated, the cylinder from De Haan et al. [de Haan 10] is used in this project to give the user a feeling of depth in the image. It makes the point of interest stand out and lets the user focus on that point.

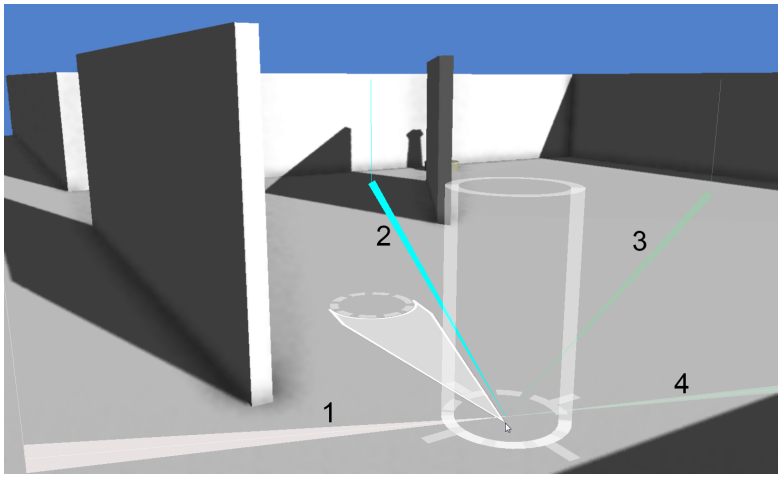
The glyphs created by De Haan et al. that attached themselves to the cylinder on confirmation have been left out. Instead, new simple arrows are connected to the cylinder, coming from all available cameras in the environment. To be sure that the scene does not get too cluttered with arrows, the lines of cameras with a score lower than 0.1 are clipped. This value can always be altered and was only used for testing purposes.

The scores for each camera are indicated by a color and transparency scheme. After all, the user is not interested in the specific scores of each camera, but in the differences between the cameras and the eventual camera that will be selected by the system. The divergent color scheme from Figure 5.9 was used with a specific color for the winning camera.[1] Cameras with a high score become green, average scoring cameras are white, and bad cameras become red. Figure 5.10 shows an example case in the implemented design. The best camera that will eventually be selected by the system, given the position and/or direction is given a cyan like color to distinguish it from the rest.

The cylinder is enhanced with a directional arrow. The direction is used for the calculations of the camera filtering system described above. The arrow can be found in Figure 5.10 as well. The arrow is also drawn in perspective to the camera, which gives more dimension to the image. The length of the arrow also indicates the size of the collision object. In other words: the longer the arrow, the more future path

[1] The color scheme was generated by the Color Brewer Tool from Pennsylvania State University: `http://colorbrewer2.org`

Figure 5.9: Scoring colors from 0 to 1. Cyan is used to indicate the actual winner that will be the next camera on the spot monitor.



#	Total Score (0–1)
1	0.4
2	0.9
3	0.7
4	0.6

Figure 5.10: All cameras receive a score, based on distance, visibility and angle. The colors match with the color scheme in Figure 5.9. The white arrow drawn by the user indicated the movement direction of the target to follow. The dotted circle is the origin point from where the user starts the drag motion.

and certainty is ensured by the user as input for the system. The arrow is only drawn while the user holds the right mouse button. At the origin of the arrow, only a small circle is left where the cylinder was. The cylinder itself stays attached to the mouse, since the end of the arrow is used as the look-at-point.

For extra situation awareness, a small minimap was also added to the screen. The minimap rotates along with the view direction of the camera and centers on the camera position. The user gets a better understanding of how the cameras are connected in an overview. Although, the implementation of the minimap is still primitive and should be developed further before usage. The minimap was not used during the evaluations described in Chapter 8.

5.6 Discussion

This chapter showed how the user is able to switch between cameras with a simple click-and-drag method. It takes minimal effort and still gives the user enough freedom to pick the camera he/she wants. Depending on the method used by the user, holding the 'Shift'-key or by double clicking, different sets of cameras are highlighted by the system. The different methods can be interpreted as different levels of freedom for the user.

The diverging color scheme with an outlying cyan color for the optimum gives the user clarity on what to expect, depending on the user's input. The cylinder and arrow give perspective to the image and give the user feedback on the input parameters for the calculations.

Other measures than distance, visibility and angle were investigated as well, but these three are significant to the user and are the only ones that can be calculated with just a position and direction. Otherwise, extra input has to be provided by the user, which also takes more time. Since these three measures give sufficient freedom, the other options were left out. Other measures that were considered during the design process are: size of the target, other users that use the cameras as well, control rights, paths and speed.

Chapter 6

Inter-Camera Navigation

So far, a system has been defined that lets the user rotate a single PTZ camera around and navigate between multiple cameras. The system is able to switch images on the spot monitor, but the change is still instant and shows no relationship between the cameras. This can cause confusion. The user's situation awareness is kept, because the user assumes himself to be inside the environment. Traveling from one camera to another should therefore take some traveling time.

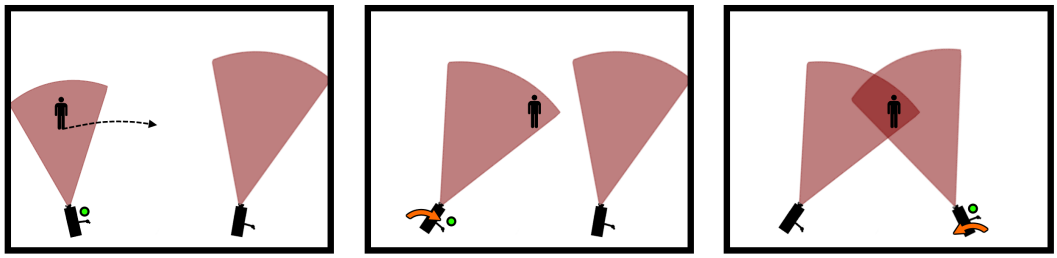
To gain some time for the camera to look at the correct look-at point and for the user to gain a better understanding of the spatial relations between the cameras, an exocentric animation is used to get from one camera to another as described by Veas et al. [Veas 10]. Instead of a direct switch, the user is taken on a flight by a turtle (actor) that is separated from the spot monitor image and which leaves the camera image on a 3D plane with correct rotation. This technique was also implemented in the systems from De Haan et al. [de Haan 09] and later improved in De Haan et al. [de Haan 10].

This chapter briefly describes the technique and indicates new improvements that have to be made to integrate PTZ cameras into this technique.

6.1 Transition Technique

The already existing technique that was implemented by De Haan et al. couples two cameras to each other by flying through the underlying 3D model. The camera images are drawn in the 3D model as if it were large TV screens. De Haan et al. describe three methods to transition from one camera to another, as already shown in Figure 2.7: zooming, panoramic (strafing), and orbiting. The three ways of transitioning have been considered as separate cases up until now, because static cameras have less freedom in motion. The use of PTZ cameras gives more freedom and more overlap to the cameras. This means that both system and user can have their focus on the POI alone, instead of extra glyphs on the screen that define various transition cases that do not overlap in viewport.

Only the orbiting technique is suitable for a POI-driven transition that is used

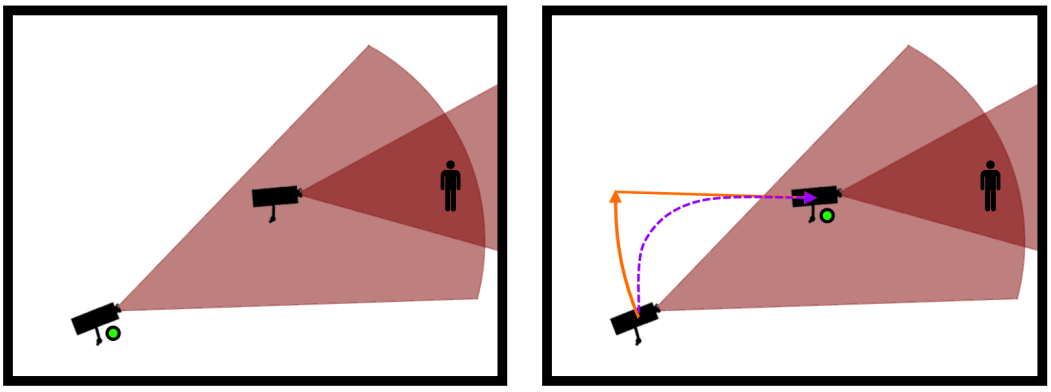


(a) The target walks to the right, the left camera is currently selected on the spot monitor.

(b) The user can rotate along with the target.

(c) The user double clicks on the target to get a closer view. The new camera rotates directly to the point of interest.

Figure 6.1: Because the camera is able to pan along with the target, there is always an overlap between two cameras. There is no arrow glyph required to represent a camera that is off screen. The camera can steer it in screen again.



(a) The spot monitor shows the camera on the left. The user double clicks on the target. The next camera is inside the viewport of the current camera.

(b) The transition can be performed by orbiting around the point of interest while the distance is linearly interpolated to the new camera.

Figure 6.2: Because the user is always forced to input a point of interest, a zoom operation is the same as an orbit operation, but the angle of the orbit is small.

in the new design and can be reused in both zooming and panoramic transitions as well. Figures 6.1 and 6.2 show why both zooming and panoramic transitions are made obsolete when using PTZ cameras only.

Now that every transition is an orbital transition, the transition from one camera to the other always keeps the point of interest on screen for the user to reference. This keeps the user conscious of the current position of the actor. The POI is translated in linear screen space, while the actor makes a circular motion around the POI in world space. The animation equation can be found in Algorithm 2 and is illustrated in an example in Figure 6.3.

Algorithm 2 The following steps are performed each frame of the animation with a time span of [0–1]. a and b define the current resp. destination camera. The values of b update over the time span. org and $dest$ define 'original' and 'destination'. These values are pre-calculated and do not change during the animation.

1. Animate the rotation of the actor by a spherical linear interpolation:

$$r = \text{slerp}(r_{org}, r_{dest}, t)$$

2. Animate the field of view of the actor:

$$fov = \text{lerp}(fov_{org}, fov_b, t)$$

3. Animate the position of the actor:

Once) Calculate viewport coordinates of the POI through camera a:

$$\boldsymbol{vp}_a = \text{worldToViewport}(\boldsymbol{poi})$$

a) Calculate viewport coordinates of the POI through camera b:

$$\boldsymbol{vp}_b = \text{worldToViewport}(\boldsymbol{poi})$$

b) Interpolate the viewport coordinates of the POI. The distance between camera a and b is calculated separately to maintain a linear distance animation:

$$\boldsymbol{vp}_t = \text{lerp}(\boldsymbol{vp}_a, \boldsymbol{vp}_b, t)$$
$$\boldsymbol{vp}_t = \begin{bmatrix} \boldsymbol{vp}_t.x \\ \boldsymbol{vp}_t.y \\ \text{lerp}(\boldsymbol{vp}_a.z, \boldsymbol{vp}_b.z, t) \end{bmatrix}$$
$$\boldsymbol{poi}_t = \text{viewportToWorld}(\boldsymbol{vp}_t)$$

c) Define the final position by adding the difference between the POI's over time:

$$\boldsymbol{pos}_{actor} = \boldsymbol{pos}_{actor} + \boldsymbol{poi} - \boldsymbol{poi}_t$$

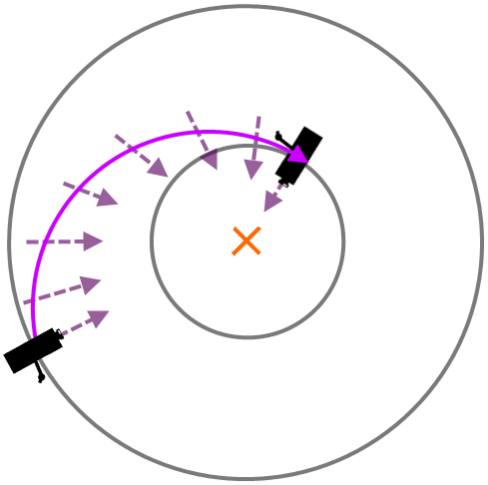


Figure 6.3: The animation from the outer camera to the inner camera. The distance is linearly interpolated, the actor keeps focusing on the POI.

6.2 Zoom Integration

The camera image that is left by the actor is placed in 3D space. Since the user is focusing on the POI, the canvas' is placed at a distance to exactly intersect with the POI and is given the size to exactly span the viewport. The canvas of the new camera to transfer to is placed in the same manner. The images themselves are blended by a transparency function, also defined by De Haan et al. as defined in Figure 6.4.

The system is not finished by only adjusting the pan and tilt levels of a camera to look at the correct position. The PTZ camera also has a zoom axis that can be user to get a better focus. When transitioning from one camera to another, the user expects the new camera to show the target at a similar size as it was in the previous camera. Since the distances are different between the cameras, the zoom level has to be adjusted to gain an equal target size.

The plane of the current camera has been set to the POI and so is the plane of the new camera. The plane of the current camera has a certain size to fill the field of view completely, which can be calculated by Algorithm 3. The inverse of Algorithm 3 can be used to calculate the field of view for the new camera, while keeping the size of the new camera's plane equal to the one of the current camera (Algorithm 4).

6.3 Visual Feedback

Extra visuals are implemented to gain situation awareness during a transition. Not only the canvasses that show the camera images are drawn, but there is also a

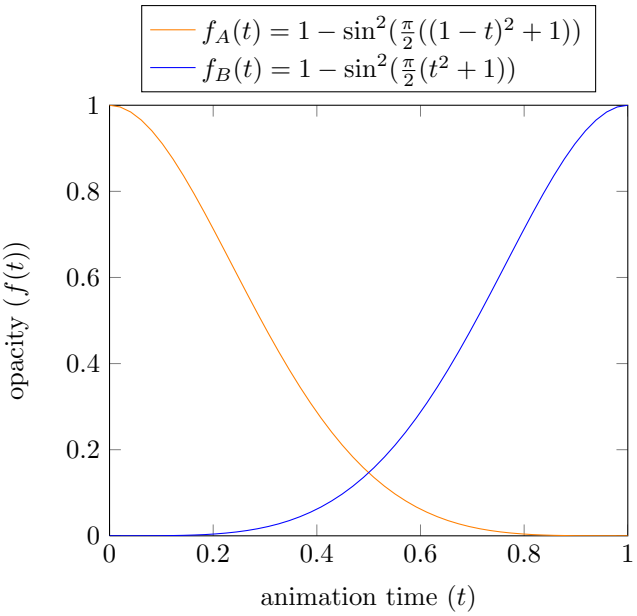
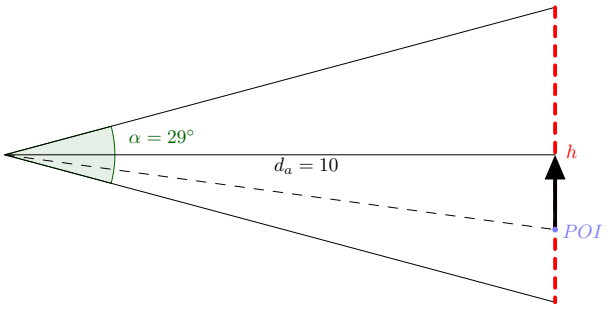


Figure 6.4: Opacity blending of the two canvasses that view the video streams during a transition. The opacity of camera a is reduced to 0, while the canvas of camera b shows itself.

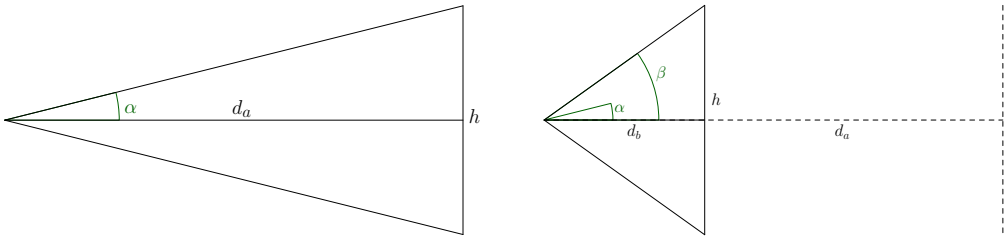
Algorithm 3 The height of the canvas size can be calculated by taking the distance to the POI and current field of view.



$$d = \|\boldsymbol{cam} - \text{proj}_{fwd}(\boldsymbol{poi})\|$$
$$h = 2d \tan \alpha$$

The red canvas height can be calculated by taking the distance to the projected POI and the current angle of the field of view.

Algorithm 4 Proof that fov_B can be calculated from distances to the POI and the old field of view fov_A alone.



(a) The view frustum of camera A with the image canvas at distance d_A

(b) The view frustum of camera B with the image canvas at distance d_B

To find β from α:

$$\begin{aligned} \alpha &= known & \beta &= ? \\ d_A &= known & d_B &= known \\ h &= known \\ \alpha &= \frac{h}{2d_A} & \beta &= \frac{h}{2d_B} \end{aligned}$$

$$\begin{aligned} h &= \alpha 2 d_A \\ \beta &= \frac{\alpha 2 d_A}{2 d_B} \\ \beta &= \alpha \frac{d_A}{d_B} \end{aligned}$$

wireframe representation of the underlying 3D model present (Figure 6.5). This makes it easier for the user to reference cameras in the world.

During the transition, the direction arrow and the cylinder stay visible as ghost objects. Whenever the transition is over, the ghost object is slowly faded out. This makes that the user can keep a reference and is not directly dependent on the image itself. During the fade out, a new cylinder is attached to the mouse, so the user can already move on with its tasks (Figure 6.6).

6.4 Discussion

The transition model defined by De Haan et al. is a good alternative for the direct image switch on the spot monitor and keeps situation awareness levels higher during the transition. The user is able to keep a better focus on the point of interest and is able to make a spatial relation between two cameras.

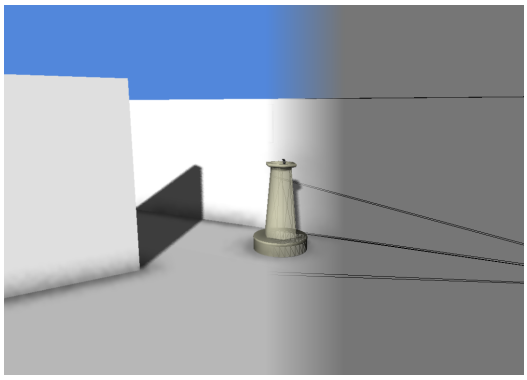


Figure 6.5: A blended image to show how the underlying wireframe model corresponds with the simulated real world.

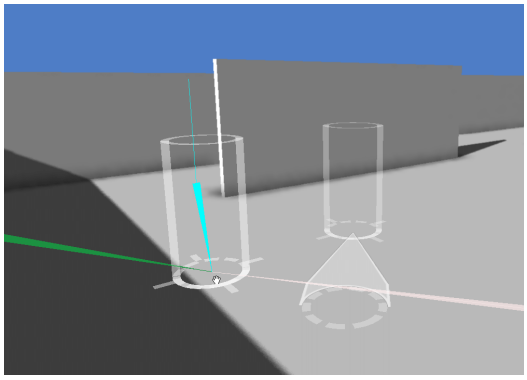


Figure 6.6: The old cylinder fades away, while a new cylinder is already attached to the mouse.

Extensions have been made to resolve the zoom levels of the camera and the different types of transitions have been brought down to only one, which makes the expectation pattern less extensive. Zoom level adjustment makes it possible to keep a target at the same size after the transition. This comes in handy when someone has zoomed in very closely and transitions to a camera that is closer.

Chapter 7

Implementation

To give a bit more in-depth information on the system prototype, this chapter elaborates on the implementation of the application. The cameras run on a simulator, which makes the choice for a development environment more flexible. There are no dependencies on hardware camera interfaces, which makes that every aspect of the system can run locally. Therefore, a game engine is a good choice, since it has many 3D features already included, which saves up a lot of time.

7.1 Game Engine

Most projects created in the Computer Graphics group of the TU Delft are done in open source applications. The mainly used product is OpenSceneGraph[1]. Although OpenSceneGraph is a great solution for making 3D interfaces, it does not provide the desired visual features and tricks that the simulator needs. Another less important issue is my lack of experience in C++. A game engine in another language would make me perform much more efficient.

The choice has been made to create the prototype in Unity3D[2]. This is a game engine that contains many cool features like: crowd simulation, light mapping, real-time shadows, and post-effects. These extras can be used to create convincing environments and people for the operators to perform their tasks. The scenarios do not have to be perfect or look like real life images, but the images must be sufficient for the user to estimate depth and perspective and they have to give the user a feeling of looking through an actual surveillance camera.

The Unity3D engine compiles both C# and Javascript files. For this project, C# was chosen as the primary programming language. The engine can be used for free, but some important features are only available in the pro version. A student license was bought for 80 euros granting access to the pro features for a year. A screenshot of the development environment can be found in Footnote 2.

[1] `http://www.openscenegraph.org`
[2] `http://www.unity3d.com`

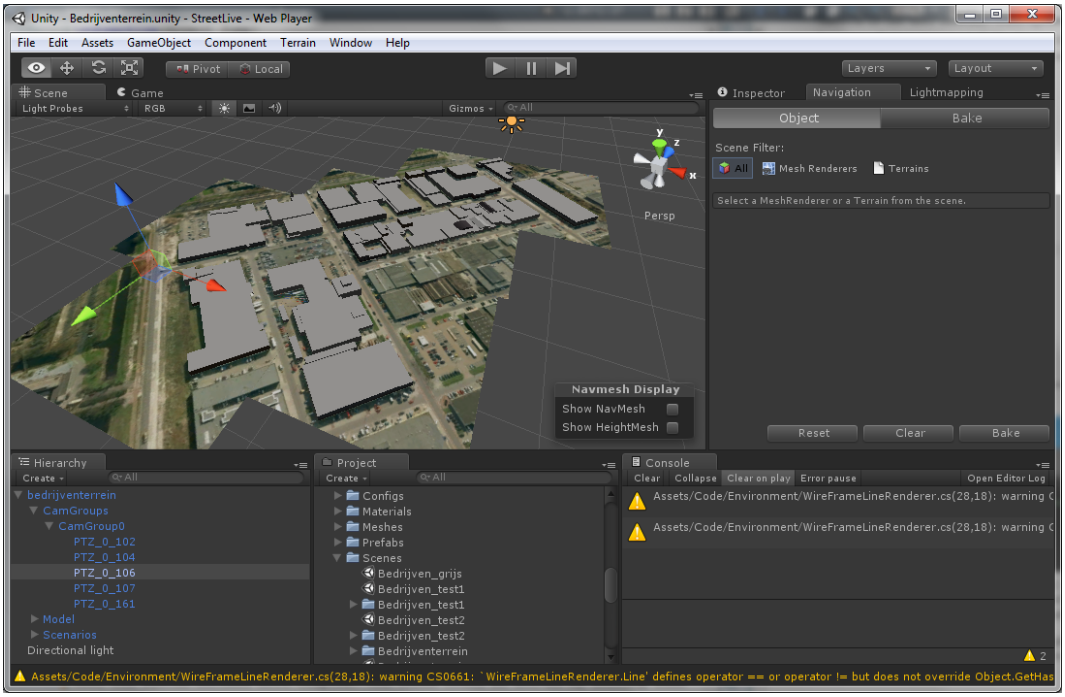


Figure 7.1: The development environment of Unity3D includes a custom compiler for C# code and also lets the user interact with simple tools to set up scenes.

7.2 Content Pipeline

In order to get the system to work with an environment, a 3D model of that scenario is required. The 3D model is used for collision detection with the mouse, as well as for collision detection with the visibility rays described in Section 5.3.2. The simulator also needs a 3D model in order to draw the area that the people walk in, so the simulator also needs materials and textures attached to the 3D model. The simulator also requires some extra information inside the model. Namely, waypoints to indicate scenario paths that the persons should follow and camera objects to position and orient the cameras.

The 3D model is created in a 3D editing application. In this case, in Blender[3]. It is then exported to a format of choice.[4] The exported model is imported into Unity3D and is used to build a scene. A little script creates a configuration file that contains data about the desired camera controls, the starting camera, and the number of persons in the scenario. It is also possible to set different collision models for mouse and visibility collision. This is convenient for clicking through thin walls, while scores for visibility are still maintained. The scene is saved and a navigation mesh is generated. The scene can be loaded into the simulator at startup via the

[3] `http://www.blender.org`

[4] Unity3D supports various 3D modeling extensions: .fbx, .dae, .3ds, .dxf, .obj

configuration files.

7.3 Vital Classes

This section describes a couple of code classes that are important for the system to work properly. The classes have a tight relationship, but the simulator is kept separate from the rest. Appendix A shows a class diagram with the most important features to give an overview of the system.

7.3.1 Camera Management

One of the two controllers that is used in the MVC model is the `CameraManager` class. The camera manager maintains the cameras and controls which images are shown to the user. It also maintains the scores that each camera receives each frame relative to the point of interest.

It contains the imported method: `GoToPoint()`. This method inputs position and direction and performs the actual transition of cameras. It orients and zooms the new camera to the correct position and fires an event to let the actor start its animation.

7.3.2 PTZ Camera

The `PTZCamera` class contains the information that a real surveillance camera also contains. It is the model in the MVC model and maintains maximal and minimal speeds and angles. The camera can be steered by angle and speed, but the system can also give a direction vector where the camera should rotate towards. The camera class fires events for certain happenings like zooming, panning, and tilting. But also when stopped, since the cameras have a small inertia factor to break their motions.

The cameras could have been implemented even more realistic by inserting a delay into the motors, but this project requires a clean environment for the first evaluations to see if the designed approach would even work in the first place.

7.3.3 Interaction Manager

The `InteractionManager` and camera manager have a tight relationship, since the interaction manager wants to know everything about the camera scores and the cameras in the environment. The interaction manager maintains all visuals and is the second important controller in the MVC model. It controls the actor that animates through the environment and lets the user take control of the camera in the spot monitor for manual steering. It receives the user input from keyboard and mouse via the View classes.

The `Actor` class contains the algorithm described in Section 6.1 for animating the transition and tells the system when the animation is finished to unblock the manual camera controls.

The view part of the system consist of a `ViewManager` class and some classes that contain GUI elements (e.g. buttons for options, the minimap, and the weighting factors). It also registers the input from the user and propagates it to the interaction manager.

7.4 Discussion

The Unity3D game engine gives a huge advantage during the implementation phase, because desired, but hard to implement, features are already implemented into the system. Especially crowd simulation and lighting has saved a large amount of time.

The main infrastructure is not complicated and uses a simple MVC model to maintain a separation between the user, the visuals and the simulator. The simulator is designed to work with other interfaces as well. This is convenient for doing comparison testing with the classic matrix wall system.

Chapter 8

Evaluation

The only real measure that one could make for this system is the usability for the user and the performance of the user during his/her task. User tests will have to show if the system is significantly better than the current systems and if the user prefers working with it. This chapter shows such an evaluation performed by a specific target group, how the tests are set up, what measures are of interest, and how the test results can be interpreted.

8.1 Test Setup

The tests performed for this prototype are evaluations to see if the basic features described in the previous chapters are useful for the user. There is a lot more to test than what is raised here, but it is a showcase of what can be done with the system and the simulator. The limitation in time does not allow us to perform any more evaluations at the moment. The following test sequence only takes 20 minutes to complete.

8.1.1 Test Sequence

To evaluate the main features of the system, a within-subject evaluation was performed to measure the difference between the current system and the new prototype. The tests to perform include the task of steering a single camera and doing a stress test of the overall system to see if a target can be followed throughout the scene. An evaluation will be performed in four tasks for the participant. The first two tests are about steering a camera, the last two are about transitioning between the cameras. Both sets of tests are provided to the participant in a random order to prevent an order effect (Figure 8.1). The tasks involving the old system show a replica of the matrix view that also uses the simulator to generate images. This makes the tests more balanced. The simulator's spot monitor can be controlled with the joystick.

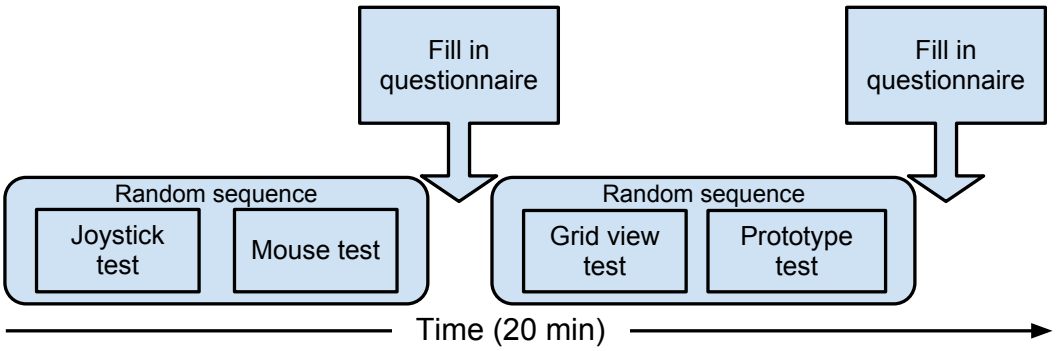


Figure 8.1: The sequence of tests that every participant takes. The participant performs the single camera tests in a random order, fills out the questionnaire and does the other two multi-camera tests in a random order as well.

8.1.2 Measures

In order to find a significant difference between the common system and the new prototype, measurements are required. The interesting topics of this project involve the objective statements: productivity, performance and speed. And the subjective measures: stress levels, situation awareness, and ease of learning. The objective measures are defined in Table 8.1. The table gives an overview of tasks that the user performs and relevant information that comes from the task. The subjective information can be retrieved by letting the participant fill in a questionnaire, which compares the old system to the new. The questionnaire can be found in Appendix B.

8.1.3 Scenarios

In the first two tests, the participant is evaluated on steering and precision skills. The participant gets to see the environment through one camera and controls it, waiting for the target person to appear. The target person is distinguishable from the rest by its distinctive yellow body color, where all the other persons wear pastel colored shirts (Figure 8.2).

All persons show shiny stars above their heads about every ten seconds. A star represents an action in real life. The user should be able to recognize the stars, because it shows that the user is paying attention to the target. The task for the test is therefore to count the number of stars that the target shows over its lifetime.

The environment used for the test is an intersection in an industrial area in Utrecht at the Mississippidreef and Nevadadreef. The joystick test and the mouse test both have a different camera position, but they film the same streets. The cameras are 20 meters away from each other. Figure 8.3 shows a map of the intersection and a screenshot of the environment.

In the second set of tests, the same task is assigned to the participant to count the stars of the target. But this time, the participant has access to multiple cameras. The cameras are set up in a logical order, because this is usually the case in real

Measure	Observation Method	Achieved Information	Claim to Prove
Total time to find the target.	Use a timer to check the time.	If it is easy to stroll around with the camera and the user is not lost, the person will be found much easier.	Navigating the camera around is easy to perform and flexible.
Number of times that the target is out of sight.	Count the number of times that the target is not visible.	If the user is able to keep the camera focused on the target.	The camera is easy to steer in a stable way.
The total time that the user has lost sight of the target.	Use a timer to measure the time intervals between visible and not visible.	If the user is able to retrieve the target when lost.	The user is aware of the orientation of the camera during steering operations.
Let the user count the number of stars that the target spawns.	Let the user count, write down the final number.	If the user pays attention and is focused on the target.	The user is still capable of performing visual tasks while steering the camera.

Table 8.1: The measures to be performed by the observer.

life environments as well. All cameras look in the same direction at start up. The environment used for this particular test is a model of the complete Nevadadreef in Utrecht. The scene holds six cameras spread along the street. Figure 8.4 shows a map of the street and the cameras and a screenshot through one of the cameras. Both tests in this set use the same environment and the same cameras. This could lead to a learning effect, but the time to create an extra environment with similar properties takes too much time and effort.

8.1.4 Matrix View Replica

In order to get a good comparison between the new prototype and the old system, a similar control scheme was built on top of the simulator. The user is able to steer the cameras with the joystick and can switch the cameras with the keyboard. By typing in a number on the keypad and pressing the 'Enter'-button, the camera will change directly to the new number. The other camera images are shown next to the spot monitor, as shown in Figure 8.5.

8.1.5 Hardware

The test was set up in pairs for faster testing. So two computers were required to let both participants work at the same time. The monitors were set up close to each

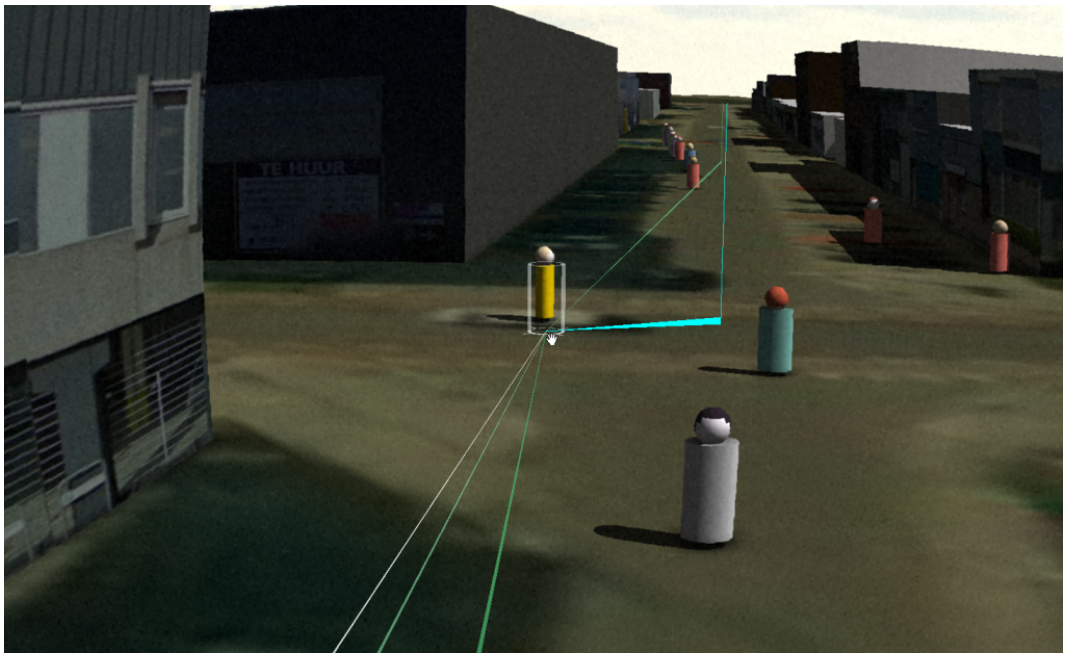
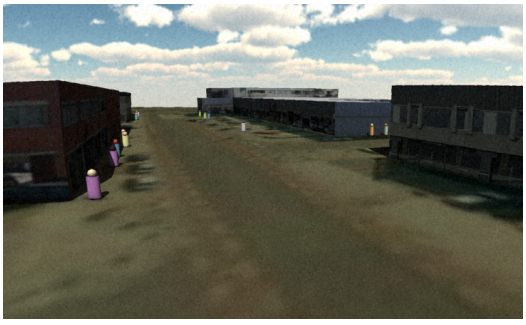


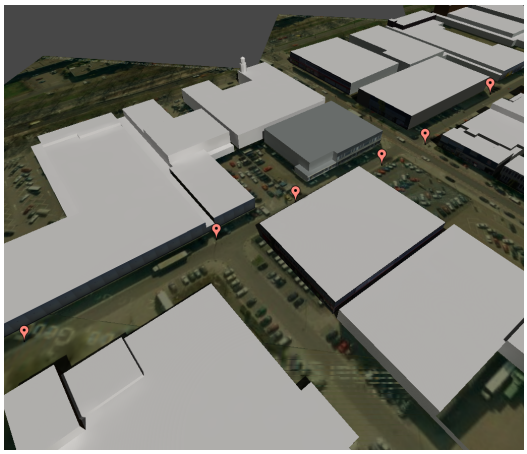
Figure 8.2: The target person to be followed has a distinctive yellow color.

(a) A picture of the environment in bird's-eye view. Left marker: camera for mouse techniques. Right marker: joystick camera.

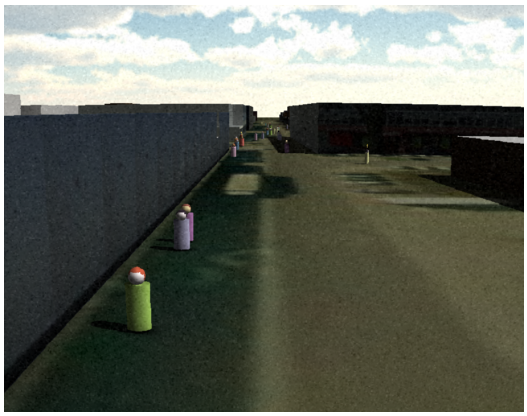


(b) A screenshot from out of the mouse camera.

Figure 8.3: The first test takes place in a single street on a crossroad.



(a) A picture of the environment of interest in bird's-eye view. The markers represent camera 1–6 from left to right.



(b) A screenshot from out of camera 1 into the main street.

Figure 8.4: The second test takes place in a long street with a number of side alleys.

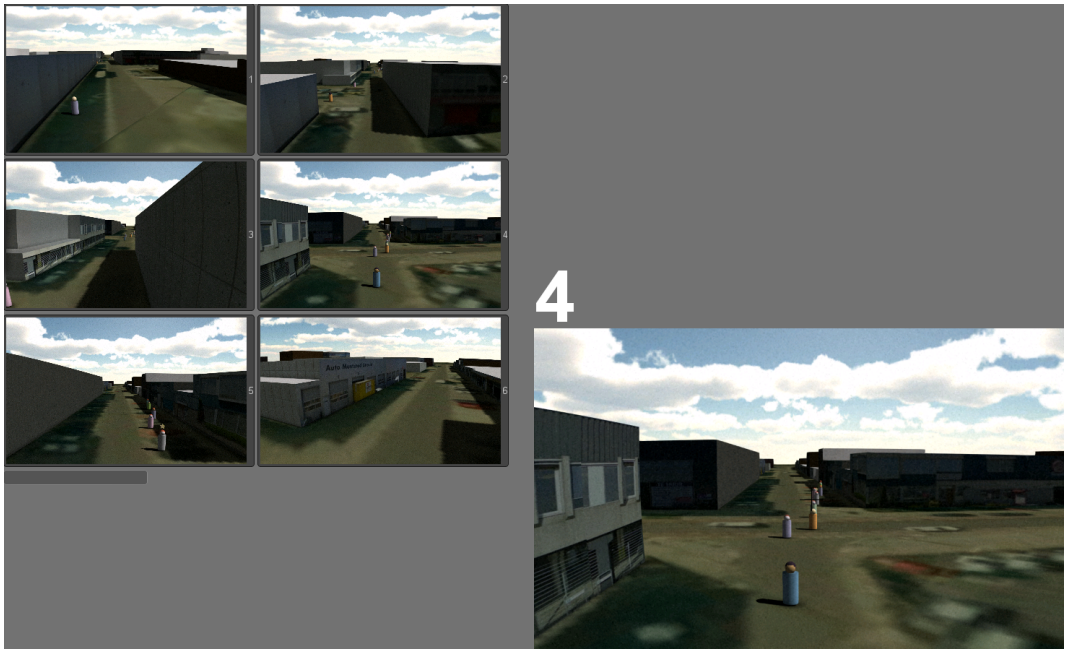


Figure 8.5: The replica of the matrix view shows the available cameras with their numbers on the left. The spot monitor is on the bottom right of the screen.

Figure 8.6: The test setup in a classroom at ROC Mondriaan. The tests are performed in couples. A video camera is placed behind the participants to film the monitors.

other so that only a single camera was required to film the actions of the participants while performing the tasks. The image of the setup can be found in Figure 8.6. The joystick used has three axes, which made it possible to perform the zoom actions with a turn in the wrist, just like in the actual systems.

8.2 Logging

The measurements required to make conclusions require timing of events. Doing this manually as an observer for pairs would be too intensive and probably not even feasible. To avoid stress on the observer's side, a logging system was built in the simulator. The logger keeps track of visibility of the target person and actions that the user performs.

The logger returns an XML file containing the actions of the user and includes a timestamp on every entry. The log file can then be used to compare the old system to the prototype for one user.

A log viewer was created to show a quick overview of the events during a session. The information is displayed in a timeline and each different sort of event has its own row (Figure 8.7). The overview is used to make the XML more human readable. The logger is also able to synchronize two XML files for comparison.

Visibility was measured by performing three raycasts to the target person origination from the camera, going to the bottom, middle, and head of the target. This is the only coupling that the simulator has with the rest of the system. This means that not only images are outputted from the simulator, but also extra information about the scenario. One could argue about the design, since there is no clear separation anymore. But on the other hand is the information vital for evaluation purposes.

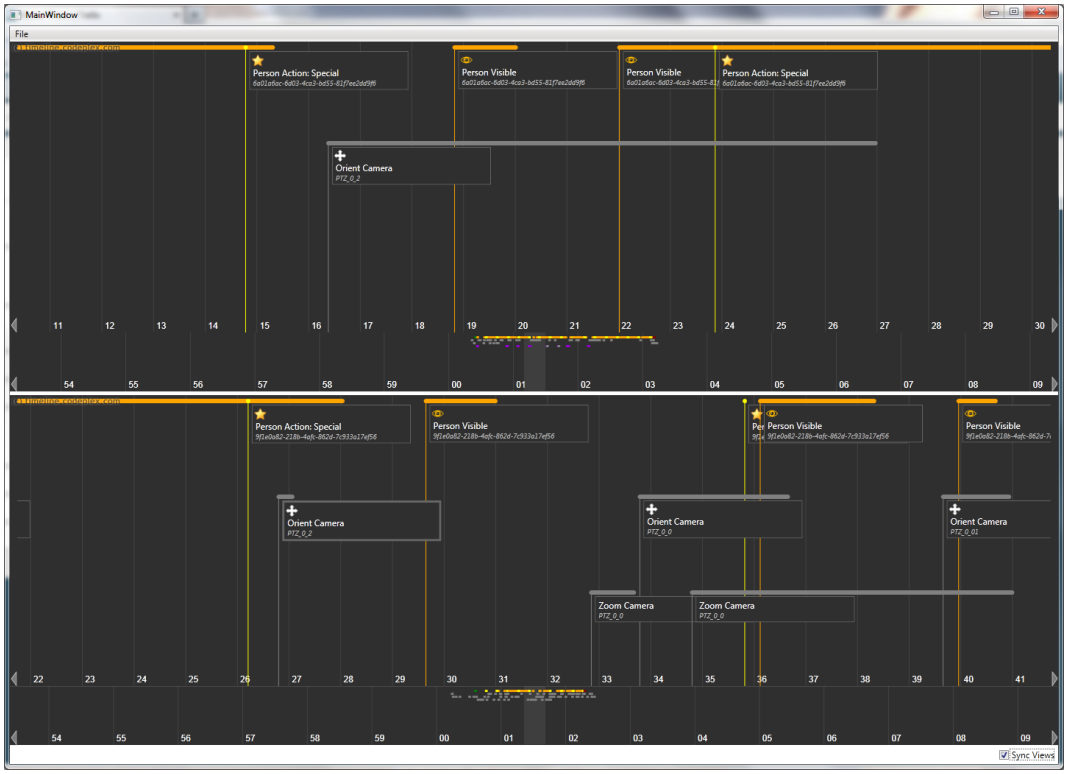


Figure 8.7: The log viewer is able to sync two log files in time in order to compare the old techniques with the newly designed ones.

8.3 Target Group

The archetype users described in Section 1.1 is still a large and diverse group of CCTV operators. The professional operators that use the current system are occupied people that do not have a large amount of time to participate in user evaluations. They are the actual target group to evaluate, but are often occupied in their daily work and most companies only have a small number of employees available. Therefore, an alternative group has to be approached that has an eye for detail; potential future video surveyors.

The ROC Mondriaan in The Hague is a school that facilitates a large group of students in security and surveillance. This educational institute is specialized in private security services which involve the field of surveillance, administrative tasks and handling in calamities. Video surveillance is not a core subject in the course program, but is often taught during internships. The students in this facility have a good eye for detail and obscure behavior, but do not have much experience with the current systems on the market. This makes them perfect as a test group and can be regarded as representatives for the common video operator with limited experience.

The students of the institute have a majority of men (i.e. about 70%) and have

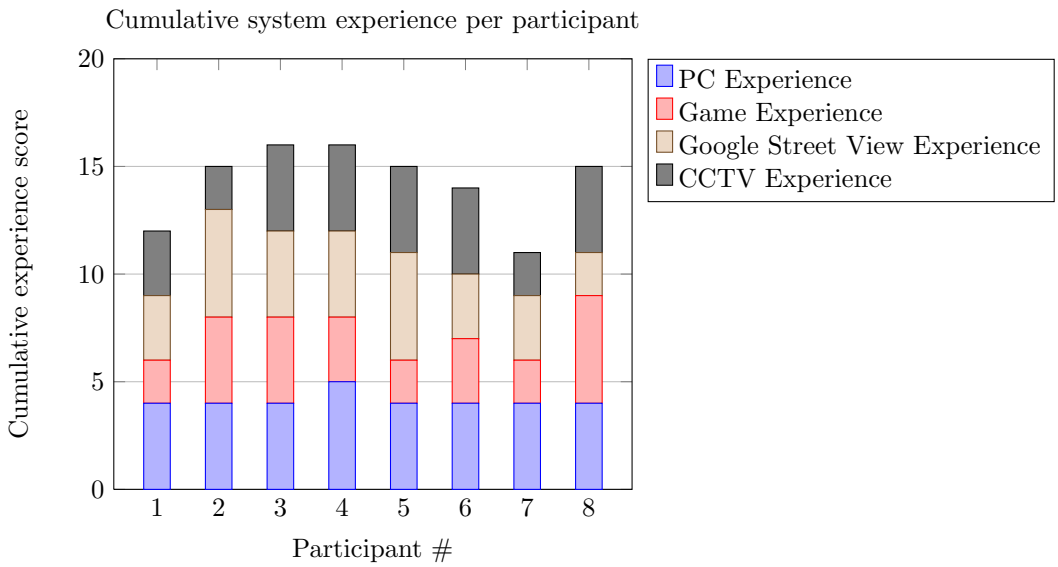


Figure 8.8: Cumulative scores on experiences with related systems. Each category has a range of 1–5.

a range of ages between 16 and 21 years old. In this range of ages, a majority is likely to know how a computer works and most of the students have probably played computer games in their life. This could be effecting the results, but probably in a good way. The test results in Section 8.4 will reveal more on this topic.

8.4 Test Results

The tests at the institute resulted in eight participants in the user evaluations. This is not a large number, but there were not enough students available at the time. This section elaborates on the test results from the questionnaire and the measurements performed by the logger. The questionnaire was filled out during the evaluations to save time, so the system could reload the application for another session.

8.4.1 Questionnaire

The questionnaire found in Appendix B consists of statements about the different tests that the participants had to perform. The questions compare the current system with the new prototype. Every question covers a concept that was enhanced by the new design.

Beforehand, a couple of experience questions are asked to see if there are any differences in the target group in computer and game experience. Figure 8.8 show that the level of computer experience is high; every person has daily experience with the pc. There is less diversity in the group than expected, but that is fine with this small number of participants.

Single Camera Control: Joystick vs. Mouse

The statements asked in the questionnaire check which functionalities of the PTZ camera are best controllable by either one of the interfaces. Besides, statements were asked on precision, learning curve, and stability of the camera. Since all participants did not have much experience with the joystick interface either, the learning curve question is valid for them to answer. Figure 8.9 displays the average results of the statements comparing both joystick and mouse facilities. Both joystick and mouse controls score high in all statements and have a high average, indicating that both methods are sufficient to work with. Panning, tilting and zooming work well in both cases. The camera reacts as the user expects and both controls are easy to learn. Both controls are smooth and stable to handle and are swiftly enough to work with. The user is aware of its situation, although the mouse control scores higher by half a point on average.

By performing a paired sample t-test, it is possible to see if the differences between the scores of each question are significantly different or not. The output from this test performed in SPSS can be found in Appendix C. The significance scores of the t-test give high results at a confidence interval of 95% on all variables. It is likely that there is no significant difference in preference ($t_{average}(7) = -0.348, p = 0.738$) between the joystick and the mouse controls for a single camera (Table 8.2). The significance levels for the correlations between the variables are also low ($p(7) \geq 0.150$) due to the low number of participants. Only correlations for ease of learning are of a significant level ($r(7) = -0.731, p = 0.039$), but the t-test indicates that there is probably no significant difference in learning effect between joystick and mouse ($t(7) = -0.0188, p = 0.857$).

The participants indicated during the tests that the mouse control is not well adjusted in its zoom functionalities, because it zooms too swiftly. The mouse controls are also known to take more effort when used in comparison to the joystick. Some users indicated that the mouse is easy to pick up, since there is no confusion on what can happen, while the joystick could easily be confused by inverting the axes mentally.

Multi-Camera Control: Matrix View vs. New Prototype

The questions to be answered on multi-camera navigation focus on handling and switching between multiple cameras. All statements in the questionnaire start with: "When I switch from one camera to another...". Again, the questions incorporate all features that should have been improved in the new prototype.

The differences in average scores between the matrix view and the new prototype are more divergent that the single camera control scores. Figure 8.10 show the differences in scores between both interfaces. It can be seen that the matrix view scores below average on most items, where the new interface scores in ranges from 'sufficient' to 'good'. The matrix view makes users confused on their whereabouts and focus target. It seems that the prototype aids the user in keeping focus on the target object and looking on the right position. The user is also less dependent on

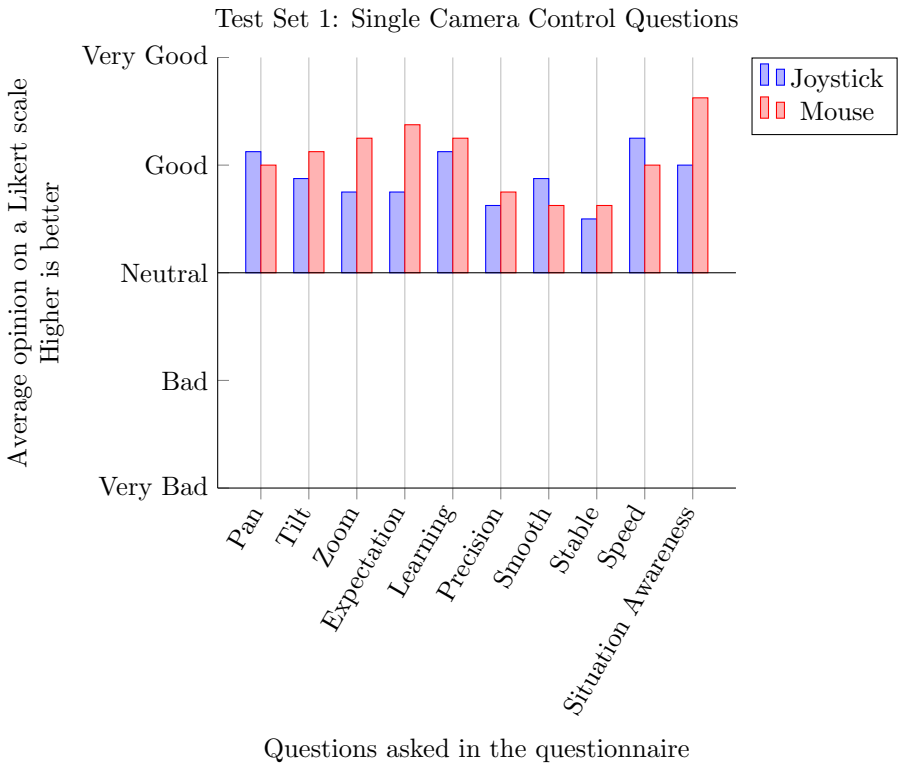


Figure 8.9: Average scores of the separate questions in the questionnaire on single camera control.

Variable	Correlation	Correlation Significance	t ($df = 7$)	T-test Significance
Pan	−.241	.566	.284	.785
Tilt	−.185	.660	−.370	.722
Zoom	−.415	.307	−.764	.470
Expectation	−.378	.356	−.957	.370
Learning	−.731	.039	−.188	.857
Precision	−.184	.663	−.196	.850
Smooth	−.426	.292	.344	.741
Stable	−.557	.152	−.205	.844
Speed	−.426	.292	.509	.626
Situation Awareness	.298	.473	−1.930	.095
Average	−.559	.150	−.348	.738

Table 8.2: T-Test, Correlation and Significance values of the paired sample t-test for the single control questionnaire. None of the significance values are low enough to conclude on hard measures.

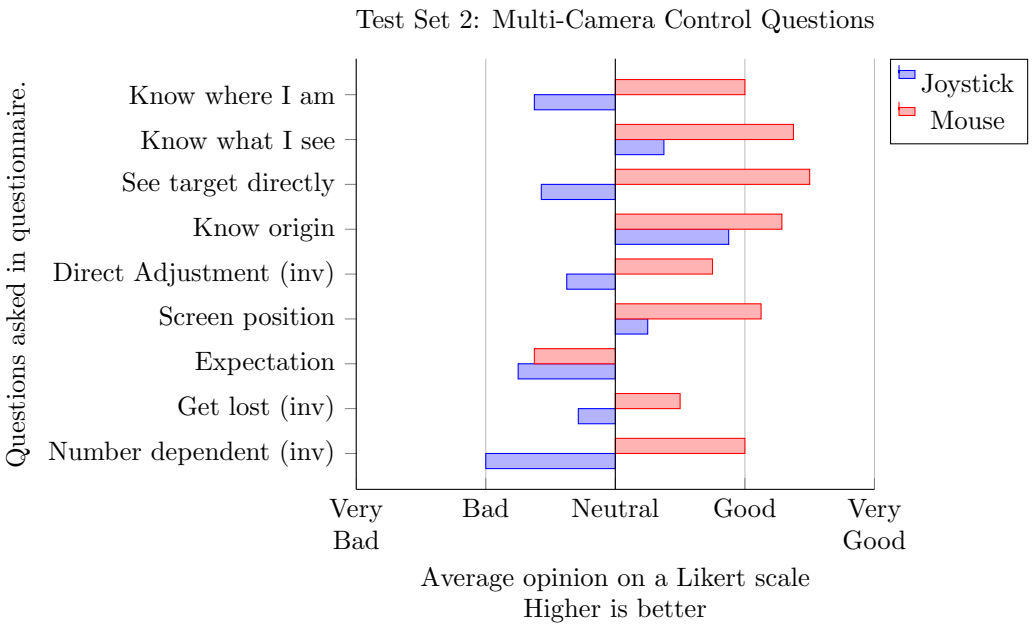


Figure 8.10: Average scores of the separate questions in the questionnaire on multi-camera control.

camera numbers, which means that they have to remember less information about the environment, so they can have more focus on their actual task.

The paired sample t-test in Appendix C confirms the differences in the questionnaire. The significance levels are low enough ($t(7) \leq -2.646, p \leq 0.033$) to show that there is a significant difference in most of the questions (Table 8.3). But also in this case, the correlation significance levels ($p(7) \geq 0.233$) between the variables is too high, which is confirmed by the low number of participants. The only three questions that are likely to have no significant difference are about the animation path from one camera to another ($t(7) = -0.703, p = 0.508$), the direct adjustment of the camera after the transition ($t(7) = -0.228, p = 0.826$), and getting completely lost after switching ($t(7) = -0.730, p = 0.493$). This could also be of missing values in the entries.

Conversations with the participants made clear that the new system was definitely preferred. Although it uses a mouse instead of a joystick, it is still fairly integrated into the rest of the interface. Some would prefer an integration with the joystick, although they could not directly tell how that would be possible to implement. Only one participant insisted on using the matrix view. The person found out how all cameras were oriented in the same direction and found a trick to easily switch a number lower every time the target got off screen. The participant did not need to readjust the camera afterwards and minimized the number of switches and steering by learning quickly about the environment.

Variable	Correlation	Correlation Significance	t ($df = 7$)	T-test Significance
Know where I am	.000	1.000	−3.265	.014
Know what I see	.183	.664	−2.646	.033
See target directly	.181	.698	−3.055	.022
Know origin	−.519	.233	−.703	.508
Direct adjustment (inv)	.029	.946	−.228	.826
Know screen position	.717	.045	−4.965	.002
Expectation	.277	.506	−2.966	.021
Get lost (inv)	−.035	.941	−.730	.493
Number dependent (inv)	−.289	.488	−3.528	.010
Average	−.062	.883	−3.351	.012

Table 8.3: T-Test, Correlation and Significance values of the paired sample t-test for the multi-camera questionnaire. Correlation values are too low for most variables, but the t-test does have some significant values.

8.4.2 Quantitative Results

The log files produced by the logger module in the system can be used to calculate the measures defined in Section 8.1.2. The information on visibility and number of stars can be calculated by a script, but the problem is that not all information is as accurate as it should be. Therefore, the calculations are manually calculated to make sure that the data represents what the user has actually seen. The recorded videos of the tests were used as a reference while doing so. The four measures are visualized in Figures 8.11 to 8.14, which show the time to find a person, the number of times that a person was lost, the ratio of stars noticed by the user, and the ratio of time that the target was off screen.

Single Camera Control: Joystick vs. Mouse

The questionnaire already showed that there is no large difference in ease of use between the joystick and mouse. The results in the log files show similar results.

The time to find the target person was biased by cooperation between the two participants and a learning effect of the environment. The participants expected that the target would appear from the same direction, and it did. Therefore, the majority of the entries was excluded for the calculation of 'time to find a target'. Figure 8.11 shows that the time to find the person is lower in the joystick situation, but the videos show that this is not a case of better handling of the camera. A paired sample t-test on these three entries show that the significance level ($t(2) = -0.848, p = 0.486$) is too high to say that there is a significant difference. The other variables measured include all eight entries, but also show that none of the measurements contribute

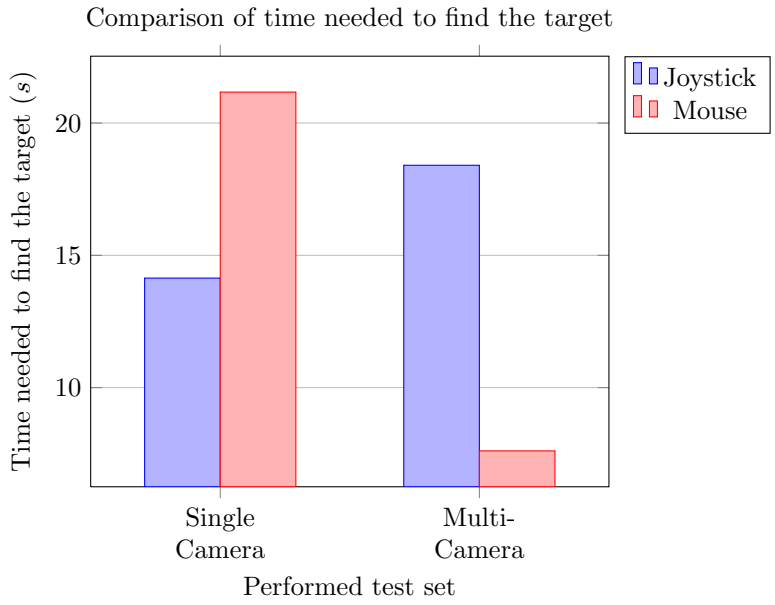


Figure 8.11: A comparison of the average time that is needed to find a target person in the environment. In the multi-camera test set, users are more likely to find the person fast with the new prototype.

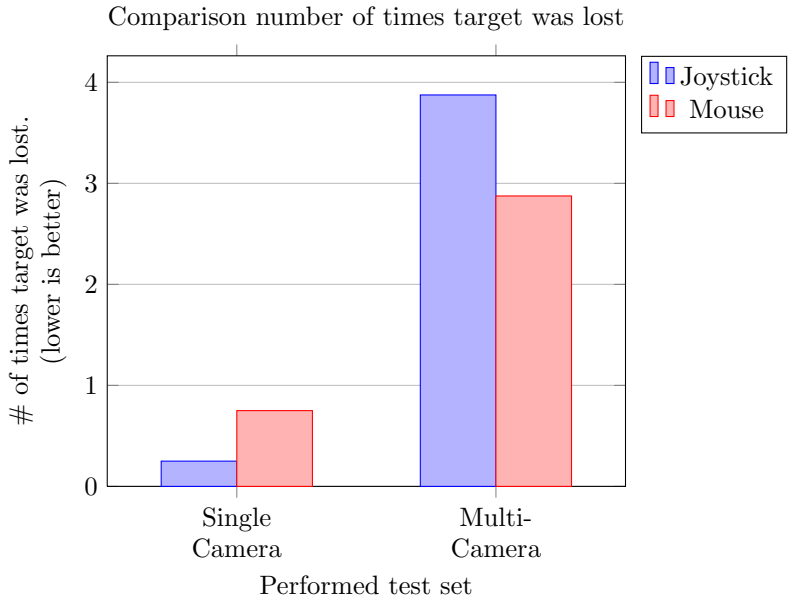


Figure 8.12: A comparison of the average number of times that the user loses its target out of sight. Lower is better.

Figure 8.13: A comparison of the numbers of stars that is found divided by the total number of stars that the participant could have seen. In the multi-camera test set, users miss out on stars more easily in the matrix view. Higher is better.

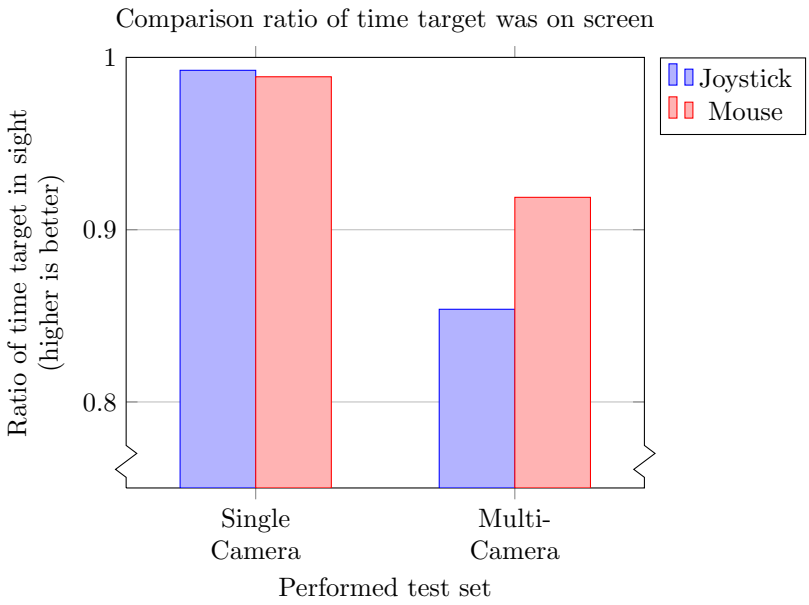


Figure 8.14: A comparison of the time ratio that the target is on screen and visible by the operator. In the multi-camera test set, users spend more time in retrieving the target with the matrix view. Higher is better.

significantly ($p(7) \geq 0.170$) to the performance of both methods. Figures 8.12 to 8.14 also shows that the differences between both methods are not that big.

Multi-Camera Control: Matrix View vs. New Prototype

One of the largest differences between the matrix view and the prototype is the active attitude of the user. The time needed to find the target is significantly lower $t(7) = 3.398, p = 0.011$) in the new prototype (Figure 8.11). This also accounts for the number of losses during a session ($t(7) = 2.646, p = 0.033$). And while Figure 8.14 would infer that the total off screen time is significantly lower in the prototype, the t-test tells otherwise and shows that there is no real difference ($t(7) = 2.068, p = 0.077$). And like in all other measures above, the correlations between the variables are actually too low ($p \geq 0.258$) to accept these assumptions. Only the correlation of the number of losses can be seen as acceptable ($r(7) = 0.762, p = 0.028$).

8.5 Discussion

The test results and questionnaires indicate that the mouse and joystick are both as preferable to use in a single camera. Although, some participants commented that the mouse worked more intuitive, but also with more effort. If any of the other designed controls will have a better influence on this factor has still to be researched. The difference in performance of both control is negligible small. More research should be performed to find the real differences in performance and ease of use.

The new prototype seems to do a good job in both performance and ease of use. Both bar charts and significance levels show that the new prototype has better scores on all fronts. The user approaches that target with a more active attitude and switches more between cameras in order to find the target. A feeling of defeat occurs more often in the matrix view. The prototype system scores significantly higher in the questionnaire and users seem to be more aware of their situation, whereabouts and point of focus. An important improvement is the ease of learning, which is probably due to the independence of the camera numbers.

8.5.1 Reliability

The statements in the questionnaire answer the same construct of usability per system. By using the method of Cronbach's alpha, it is possible to see if the questions asked are reliable for the construct to measure. Appendix C contains the output that SPSS has generated on the Cronbach's alpha calculations. A reliability of 0.7 or higher can be considered as good reliability. Table 8.4 shows that all constructs measured in the questionnaire have acceptable alpha values.

	Classic System	Prototype System	Mean
Single Camera Test	0.903	0.945	0.924
Multi-Camera Test	0.811	0.706	0.759

Table 8.4: The Cronbach's alpha values for each set of questions in the questionnaire. All values are greater than 0.7 and so are the mean alpha values, indicating that the questions are reliable.

8.5.2 Future Testing

The evaluations performed for this research only prove that a part of the system has improved the task description so far. The number of tests that can still be performed with the new system and simulator is endless. This evaluation has only shown one mouse control, while two others were also implemented. The measures only indicate the performance and attention of the users, bot not the activity. This could be measured, since the logger also outputs information on camera activity and the number of actions that the user performs.

As for an idea of future evaluations, the following subjects could be of interest to improve the system even further. The list below is just a reference list that could be useful for more evaluations and it shows the power of the simulator, where no hardware cameras are required and where the number of environments and scenarios has no limit and low costs.

- Test the other two camera controls and compare all measures between the four types of controls.
- Repeat the tests performed above with more users to make the correlation significant.
- Evaluate the activity of the user and the effort it takes to perform certain actions.
- Let the participants learn how to user either system for a longer period of time and evaluate once more on the familiar environment, but also on new environments.
- Use more occluded and less occluded environments to see if there are any differences in usage.
- Evaluate environments with more and with less cameras.
- Evaluate environments with a sparse number of cameras or with a denser coverage
- Test for any differences between gamers and non-gamers and other target groups that involve video surveillance.
- Evaluate the differences between known and unknown areas.

Chapter 9

Discussion and Conclusion

9.1 Summary

In this work, an effort was made to improve the performance and usability of video surveillance systems. In particular, for the task of pursuing and following a person around the area. The task analysis showed that this task is one of the hardest and most stressful tasks that the operator can perform during the day. The user of the surveillance system lacks situation awareness and spatial knowledge and is often ashamed for losing the target out of sight. Future plans of system centralization make it vital that a new system does not rely on camera numbers anymore.

The work of De Haan et al. [de Haan 10] was used as a starting point and was extended by the integration of PTZ cameras. The navigation tools from Google Street View were used as inspiration to create similar techniques for PTZ cameras. Document viewer techniques were also used as a reference for camera controls.

The system simulates video images through a game engine. Which cuts the costs for live PTZ cameras, video management systems and actors to play test scenarios repeatedly. Properties of actual cameras were used to reenact the behavior of the cameras in the environment.

The PTZ controls to steer with the camera that have been implemented are: the simple joystick, a shooter game control, the hand tool control, and a document viewer control that simulates the middle mouse button found in document viewers. Only the joystick and hand tool have been tested during evaluations, because both are familiar to people who play games and have used Google Street View. Zooming with the mouse was implemented as a simple scroll wheel that alters the field of view.

Navigating from one camera to another is achieved by clicking on the ground of a 3D model that lies in sync with the image. The user picks a point of interest and makes a gesture with the mouse to indicate in which direction the target to follow is moving. The system calculates the most interesting cameras and returns visual feedback continuously. The user can already see what camera will be chosen before releasing the mouse. The user only needs to use the mouse and two buttons

to control the entire system.

The measures of interest that the system uses to calculate the best camera are: distance, visibility and view angle. The distance is used to rate the closest camera as best. The visibility measure is calculated by sampling rays to the point of interest from each camera to see which camera has the best view. To make sure that the gesture of the user is used as an input, a final measure of angle is used to see if the camera aligns with the direction of the gesture. The scores are weighted to give an overall score. The winning camera will be switched onto the spot monitor.

The transition techniques used by De Haan et al. were re-implemented, but also simplified. The three different cases have collapsed into a single case of orbiting, since PTZ cameras always have an overlap with one another. The destination camera calculates its field of view from the field of view of the origin camera and the size of the 3D canvas that is used to draw the video stream on. The target to be followed stays at equal size after transitioning to the new camera.

The Unity3D engine was used to implement both simulator and prototype for reasons of performance and already implemented features. Post-effects were used on the images from the simulator to make the effect of a real surveillance camera more realistic.

The evaluations have taken place at ROC Mondriaan, an educational institute for safety and security, with a group of security students that have minor experience in video surveillance, but major experience in field surveillance. Within-user test sessions revealed that the new mouse interface is just as preferable as the joystick control and that the new prototype is significantly better in both performance, ease of use and learning with multiple cameras. The user has a more active attitude in finding the target and changing between cameras. The user is not afraid to do so. The user is not reliable on camera numbers anymore, which makes the learning curve less steep for users that have no experience in a particular environment yet.

9.2 Conclusion

The project introduces a new way of interacting with multiple cameras at the same time in the environment without being dependent on specific orders or numbers. Evaluations showed that the learning curve is not as steep as for the matrix view and that mouse interaction is preferred by the user over typing numbers. The research questions asked in Section 1.3 will now be answered to come to a final conclusion.

Research Question 1. *What intuitive types of camera controls can be used to steer a single PTZ camera? How do these controls mix with a complete system that also lets the user switch between other cameras?*

The joystick control is the most common control in video surveillance nowadays. Alternatives are found in document viewers. They provide an equal number of degrees of freedom and are therefore capable to also steer a PTZ camera. The evaluations performed in Chapter 8 showed that there are no significant differences

between the joystick and mouse controls, making it assumable that both controls are just as easy to use for steering a PTZ camera.

The mouse controls can easily be implemented into a multi-camera system. Only a mouse is required to control the complete system. The evaluations showed that a complete system with the mouse is easier to learn and use than the classic grid system with a joystick.

Research Question 2. *What alternative navigation methods can be used to improve active multi-camera video surveillance systems during intense situations and in which ways do they improve video surveillance systems or the user?*

Instead of typing in a number to get to the following camera, users can now make a transition based on spatial knowledge and relationships. A point of interest is chosen in the perspective of the camera and the user flies towards a newly selected camera that was indicated beforehand by colored arrows. The point of interest is always visible during the transition. This system lays focus on point of interest instead of manual image recognition.

The new prototype improves interaction with the user on a basis of performance, ease of use, and situation awareness. The user is more aware of its position in relation to other cameras and prefers to use the new system rather than the old matrix view. The user is less likely to lose its target while using the new prototype.

Research Question 3. *In what way can the new system be effectively tested on low costs and a high number of iterations?*

The costs of the prototype were reduced by implementing a simulator that provides real-time video images. Characters walk through the environment performing actions. These characters can be used for scenarios during evaluations. The cameras in the scenarios act like real life cameras with similar response properties. Scenarios are reduced in cost, since there is no need to hire real people that have to be followed through the cameras. Another advantage is the ability to replay scenarios infinitely many times, resulting in a less biased evaluation.

9.3 Future Work

Enhancing this research even further mostly involves evaluations and tweaking of this new prototype. This section sums up interesting topics that are still to be researched. Some of the improvements that still have to be done do not require much research and can easily be implemented, others are harder and need some more investigations on the situation.

Short-term Improvements

At the moment, the user is only able to pick a place on the ground and drag a gesture from there. When the user places the mouse on a wall, nothing will happen. The

user is obstructed in its controls. This problem should be solved by implementing some kind of a wall transition as well.

The weighted measures to filter cameras works, but is not fully optimized in what it should do. By improving the visibility measure to a more accurate instrument, it would be possible to actually clip cameras that do not see the object at all. Creating such a technique will have to map the collision object to screen space and measure the total surface ratio on screen.

Visibility could also be improved by involving a gradient in the movement gesture. Now, rays are sampled over the whole arrow object and only a ratio is checked. But the user creates an uncertainty by creating longer arrows. Therefore, the tip of the arrow is a less confident point than the base of the arrow. One could think of extra weights in the ratio of each ray sample.

Sometimes, users make a wrong decision. In the current system, it is possible to return to the previous camera by typing in the previous number. The new prototype sometimes requires a full 180° turn and then a correct mouse gesture. An undo-action could optimize user performance.

The transition that is used to get from one camera to another works fine for cameras that are quite close to each other, but cameras that have a far range sometimes make a circular motion with a radius of $100m$. The data from CIV that covers 1 km^2 with only six cameras shows exactly this problem. The number is large and sometimes confusing when both cameras are low near the ground. This could be improved with a flyover that takes off the ground first, still keeping the point of interest on screen. How this track should be calculated precisely has to be researched.

As already mentioned in Section 4.2.4, the scroll zoom technique only adjusts the field of view. It does not focus on the mouse cursor, as commonly implemented in document viewers. This could be another improvement to make to the zoom controls.

Also, the previews that were implemented by De Haan et al. have been left out. It would be an idea to cache all cameras with panorama images to show a static preview of the scene.

The side products of this projects can definitely be improved. The debugger could even output more information and calculate more precise information on visibility and camera transfers. But as mentioned before, the visibility calculations should be improved first.

The simulator could be improved with animated characters that actually do real life actions, instead of spawning stars.

Long-term Improvements

In this project, the assumption was made that all cameras are active cameras and that all cameras therefore have an overlap. This simplifies the problem and gives more freedom for implementations. But some systems have both active and static cameras. Some of the glyph implementations of De Haan et al. should probably return to make a more complete system.

Similar to that, real camera images have been left out for ease of implementation. Using real cameras will probably resolve in other problems that have not been foreseen just yet, like camera calibration, precision and delay. The simulator could be implemented with a delay to see if the interaction is the same, but it will never cope with the randomness of real electric motors.

Final Words

It can be seen that most of the implementations needed here are small tweaks, or large features that should first be tested. The simulator could help in doing so to improve the design further before actually implementing expensive features with real cameras and such.

Especially improving the debugger could help in creating a more streamlined evaluation phase for large groups of people. A lot of things were still to be done manually in this project's evaluation phase and that could definitely be optimized.

Most of the time will be required to get better test results on a variety of interesting topics as described in Section 8.5.2. Hopefully interesting results will show up that will provide more information on improving video surveillance even further. As a final thought, I would like to recommend merging the new interface with the classic matrix grid. The new system only uses the mouse, but a combination with number ticking looks like a plausible all-round solution.

Bibliography

[Calderara 09] Simone Calderara, Andrea Prati & Rita Cucchiara. *Video surveillance and multimedia forensics.* In Proceedings of the First ACM workshop on Multimedia in forensics - MiFor '09, page 13, New York, New York, USA, 2009. ACM Press.

[Chen 07] Jessie Y. C. Chen, Ellen C. Haas & Michael J. Barnes. *Human Performance Issues and User Interface Design for Teleoperated Robots.* IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews), vol. 37, no. 6, pages 1231–1245, November 2007.

[de Haan 09] Gerwin de Haan, Josef Scheuer, Raymond de Vries & Frits H. Post. *Egocentric navigation for video surveillance in 3D Virtual Environments.* In 2009 IEEE Symposium on 3D User Interfaces, pages 103–110. IEEE, 2009.

[de Haan 10] Gerwin de Haan, Huib Piguillet & Frits Post. *Spatial Navigation for Context-Aware Video Surveillance.* Technical Report 5, TU Delft, September 2010.

[Furmanski 02] Chris Furmanski, Ronald Azuma & Mike Daily. *Augmented-reality visualizations guided by cognition: perceptual heuristics for combining visible and obscured information.* In Proceedings. International Symposium on Mixed and Augmented Reality, pages 215–320, Maliby, CA, 2002. IEEE Comput. Soc.

[Girgensohn 06] Andreas Girgensohn, Frank Shipman, Anthony Dunnigan, Thea Turner & Lynn Wilcox. *Support for effective use of multiple video streams in security.* In Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks - VSSN '06, page 19, New York, New York, USA, 2006. ACM Press.

[Girgensohn 07] Andreas Girgensohn, Tony Dunnigan, Don Kimber, Jim Vaughan, Tao Yang, Frank Shipman, Thea Turner, Eleanor Rieffel, Lynn Wilcox & Francine Chen. *DOTS*. In Proceedings of the 15th international conference on Multimedia - MULTIMEDIA '07, page 423, New York, New York, USA, 2007. ACM Press.

[Goh 08] A. H. W. Goh, Y. S. Yong, C. H. Chan, S. J. Then, L. P. Chu, S. W. Chau & H. W. Hon. *Interactive PTZ camera control system using WII remote and infrared sensor bar*. In Proceedings of World Academy of Science, Engineering and Technology, numéro Cii, pages 127–132, 2008.

[Iannizzotto 05] Giancarlo Iannizzotto, Carlo Costanzo, Francesco La Rosa & Pietro Lanzafame. *A multimodal perceptual user interface for video-surveillance environments*. In Proceedings of the 7th international conference on Multimodal interfaces - ICMI '05, page 45, New York, New York, USA, 2005. ACM Press.

[Keval 09] Hina Uttam Keval. *Effective design, configuration, and use of digital CCTV*. PhD thesis, University College London, 2009.

[Kim 08] Jiman Kim & Daijin Kim. *Probabilistic camera hand-off for visual surveillance*. In 2008 Second ACM/IEEE International Conference on Distributed Smart Cameras, numéro 7-11 Sept. 2008, pages 1–8, Stanford, CA, September 2008. IEEE.

[Krahnstoever 08] Nils Krahnstoever, Ting Yu, Ser-Nam Lim, Kedar Patwardhan & Peter Tu. *Collaborative real-time control of active cameras in large scale surveillance systems*. In Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications - M2SFA2 2008, 2008.

[Matheus 03] Christopher J. Matheus, Mieczyslaw M. Kokar & Kenneth Baclawski. *A core ontology for situation awareness*. Information Fusion, 2003. Proceedings of the Sixth International Conference of, vol. 1, pages 545–552, 2003.

[Ott 06] Renaud Ott, Mario Gutiérrez, Daniel Thalmann & Frédéric Vexo. *Advanced virtual reality technologies for surveillance and security applications*. In Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications - VRCIA '06, volume 1, page 163, New York, New York, USA, 2006. ACM Press.

[Qureshi 07] Faisal Z. Qureshi & Demetri Terzopoulos. *Surveillance in Virtual Reality: System Design and Multi-Camera Control*. In 2007 IEEE

Conference on Computer Vision and Pattern Recognition, pages 1–8. IEEE, June 2007.

[Qureshi 09] Faisal Z. Qureshi & Demetri Terzopoulos. *Planning ahead for PTZ camera assignment and handoff.* In 2009 Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC), pages 1–8. IEEE, August 2009.

[Qureshi 11] Faisal Z. Qureshi & Demetri Terzopoulos. *Distributed Video Sensor Networks.* In Bir Bhanu, Chinya V. Ravishankar, Amit K. Roy-Chowdhury, Hamid Aghajan & Demetri Terzopoulos, editors, Distributed Video Sensor Networks, chapitre 19, pages 273–287. 2011.

[Shah 07] Mubarak Shah, Omar Javed & Khurram Shafique. *Automated Visual Surveillance in Realistic Scenarios.* IEEE Multimedia, vol. 14, no. 1, pages 30–39, 2007.

[Singh 07] Vivek K. Singh, Pradeep K. Atrey & Mohan S. Kankanhalli. *Coopetitive multi-camera surveillance using model predictive control.* Machine Vision and Applications, vol. 19, no. 5-6, pages 375–393, July 2007.

[Snavely 08] Noah Snavely, Rahul Garg, Steven M. Seitz & Richard Szeliski. *Finding paths through the world's photos.* ACM Transactions on Graphics, vol. 27, no. 3, page 1, August 2008.

[Thorndyke 82] Perry W. Thorndyke & Barbara Hayes-Roth. *Differences in spatial knowledge acquired from maps and navigation.* Cognitive Psychology, vol. 14, no. 4, pages 560–589, October 1982.

[Veas 10] Eduardo Veas, Alessandro Mulloni, Ernst Kruijff, Holger Regenbrecht & Dieter Schmalstieg. *Techniques for view transition in multi-camera outdoor environments.* In Proceedings of Graphics Interface 2010 on Proceedings of Graphics Interface 2010, pages 193–200, 2010.

[Wang 07] Yi Wang, David M. Krum, Enylton M. Coelho & Doug A. Bowman. *Contextualized videos: combining videos with environment models to support situational understanding.* IEEE transactions on visualization and computer graphics, vol. 13, no. 6, pages 1568–75, 2007.

[Wang 08] Yi Wang, Doug Bowman, David Krum, Enylton Coalho, Tonya Smith-Jackson, David Bailey, Sarah Peck, Swethan Anand, Trevor Kennedy & Yernar Abdrazakov. *Effects of video placement and spatial context presentation on path reconstruction tasks with*

contextualized videos. IEEE transactions on visualization and computer graphics, vol. 14, no. 6, pages 1755–62, 2008.

[Yao 08] Jian Yao & Jean-marc Odobez. *Multi-Camera Multi-Person 3D Space Tracking with MCMC in Surveillance Scenarios.* In M2SFA2, page 12, 2008.

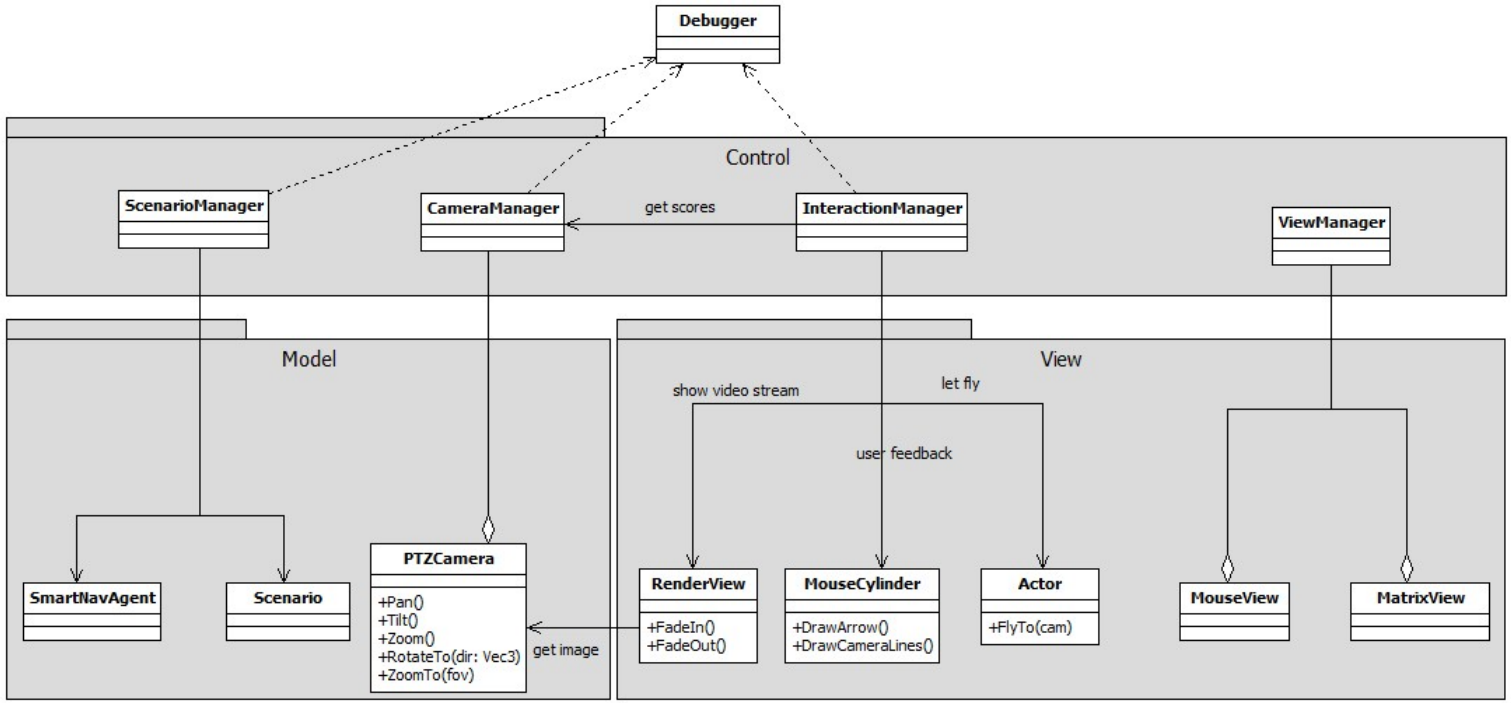
[Zhu 11a] Dingyun Zhu, Tom Gedeon & Ken Taylor. *Exploring camera viewpoint control models for a multi-tasking setting in teleoperation.* In Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11, page 53, New York, New York, USA, 2011. ACM Press.

[Zhu 11b] Dingyun Zhu, Tom Gedeon & Ken Taylor. *Moving to the centre: A gaze-driven remote camera control for teleoperation.* Interacting with Computers, vol. 23, no. 1, pages 85–95, January 2011.

Appendix A

Class Diagram

This appendix shows a simplified version of the class structure used in the prototype.


Debugger
Control
ScenarioManager
CameraManager
get scores
InteractionManager
ViewManager
Model
View
show video stream
let fly
user feedback
SmartNavAgent
Scenario
PTZCamera
+Pan()
+Tilt()
+Zoom()
+RotateTo(dir: Vec3)
+ZoomTo(fov)
get image
RenderView
+FadeIn()
+FadeOut()
MouseCylinder
+DrawArrow()
+DrawCameraLines()
Actor
+FlyTo(cam)
MouseView
MatrixView

Appendix B

Evaluation Questionnaire

Evaluatieformulier na tests camerabeveiligingssysteem

31 mei 2012 TU Delft

Je hebt nu zowel een systeem gebruikt zoals het nu in het bedrijfsleven wordt gebruikt en een nieuw systeem dat is ontwikkeld door de TU Delft. Door middel van deze vragenlijst kunnen we zien wat je beter vond werken en waarom.

Eerst stellen we een aantal vragen over je achtergrond, daarna gaan we in op de twee systemen.

1. Geslacht

☐ Man ☐ Vrouw

2. Leeftijd: _____

3. Opleiding

☐ MBO Niveau 1
☐ MBO Niveau 2
☐ MBO Niveau 3
☐ MBO Niveau 4
☐ Anders, namelijk: _____

4. Maak je weleens gebruik van de computer?
Het maakt niet uit voor welke doeleinden. Het is van belang om te kijken of je een beetje ervaring hebt met computers.

☐ Minder dan 1 keer per maand
☐ Minder dan 1 keer per week
☐ Meer dan 1 keer per week
☐ Bijna elke dag
☐ Echt elke dag

5. Speel je weleens computerspelletjes?

☐ Nee / Bijna nooit
☐ Ja, kleine spelletjes op het internet / smartphones
☐ Ja, ik speel weleens een groot computerspel / consolespel
☐ Ja, ik speel best vaak een groot computerspel / consolespel
☐ Ja, ik zie mezelf als een echte gamer

6. Heb je weleens gebruik gemaakt van Google Maps en het daarin gebouwde Google Street View?
Google Street View is dat oranje mannetje in Google Maps waarmee je foto's op straat kunt bekijken. (Zie plaatje voor een voorbeeld)

- ☐ Nee, nog nooit
- ☐ Ja, maar ik zou niet meer weten hoe het ook alweer werkt
- ☐ Ja, dat gebruik ik heel soms
- ☐ Ja, ik weet wel hoe het werkt
- ☐ Ja, en ik kan er goed mee overweg

7. Heb je weleens achter een beveiligingssysteem gezeten?
Heb je weleens achter camera's gezeten en ze bestuurd? Bijvoorbeeld op stage.

- ☐ Nee, nog nooit
- ☐ Nee, alleen een keer meegekeken
- ☐ Ja, ik heb zelf camera's gestuurd
- ☐ Ja, ik heb zelf ook verdachte mensen gevolgd

8. Vond je het oude systeem lijken op het beveiligingssysteem dat je zelf al eens hebt gezien?
Alleen invullen als je Ja had bij vraag 7 of als je weleens hebt meegekeken.

	1	2	3	4	5	
Nee, verre van	☐	☐	☐	☐	☐	Ja, het kwam in de buurt

Sturen met de camera

De volgende twee vragen vergelijken de joystickbesturing en de muisbesturing met elkaar.

9. Geef aan wat je vindt van de *joystick* op de volgende onderdelen.

	Slecht	Matig	Neutraal	Voldoende	Goed
Zijwaarts bewegen	☐	☐	☐	☐	☐
Op en neer bewegen	☐	☐	☐	☐	☐
In- en uitzoomen	☐	☐	☐	☐	☐
Het doet wat ik verwacht	☐	☐	☐	☐	☐
Makkelijk te leren	☐	☐	☐	☐	☐
Precisie	☐	☐	☐	☐	☐
Soepele beweging van de camera	☐	☐	☐	☐	☐
Stabiliteit (of het veel slingerde)	☐	☐	☐	☐	☐
Snelheid	☐	☐	☐	☐	☐
Ik weet waar ik naar kijk tijdens het sturen	☐	☐	☐	☐	☐

10. Geef aan wat je vindt van de *muisbesturing* op de volgende onderdelen.

	Slecht	Matig	Neutraal	Voldoende	Goed
Zijwaarts bewegen	☐	☐	☐	☐	☐
Op en neer bewegen	☐	☐	☐	☐	☐
In- en uitzoomen	☐	☐	☐	☐	☐
Het doet wat ik verwacht	☐	☐	☐	☐	☐
Makkelijk te leren	☐	☐	☐	☐	☐
Precisie	☐	☐	☐	☐	☐
Soepele beweging van de camera	☐	☐	☐	☐	☐
Stabiliteit (of het veel slingerde)	☐	☐	☐	☐	☐
Snelheid	☐	☐	☐	☐	☐
Ik weet waar ik naar kijk tijdens het sturen	☐	☐	☐	☐	☐

Wisselen tussen camera's

De volgende vragen gaan over de tweede test die je hebt gedaan met het volgen van het mannetje.

11. Geef aan wat passend is voor het *oude systeem.*
Als ik wissel van camera, dan...

	Helemaal mee oneens	Een beetje mee oneens	Neutraal	Een beetje mee eens	Helemaal mee eens
weet ik precies waar ik ben	☐	☐	☐	☐	☐
weet ik precies wat ik zie	☐	☐	☐	☐	☐
zie ik direct het doel dat ik aan het volgen was	☐	☐	☐	☐	☐
zie ik hoe ik van de ene camera naar de andere kwam	☐	☐	☐	☐	☐
moet ik de camera direct weer bijsturen	☐	☐	☐	☐	☐
weet ik naar welke plek op het scherm ik zou moeten kijken	☐	☐	☐	☐	☐
zie ik wat ik wilde zien	☐	☐	☐	☐	☐
raak ik snel de weg kwijt	☐	☐	☐	☐	☐
ben ik afhankelijk van de cameranummers en hoe de camera's opgehangen zijn	☐	☐	☐	☐	☐

12. Geef aan wat passend is voor het *nieuwe systeem.*
Als ik wissel van camera, dan...

	Helemaal mee oneens	Een beetje mee oneens	Neutraal	Een beetje mee eens	Helemaal mee eens
weet ik precies waar ik ben	☐	☐	☐	☐	☐
weet ik precies wat ik zie	☐	☐	☐	☐	☐
zie ik direct het doel dat ik aan het volgen was	☐	☐	☐	☐	☐
zie ik hoe ik van de ene camera naar de andere kwam	☐	☐	☐	☐	☐
moet ik de camera direct weer bijsturen	☐	☐	☐	☐	☐
weet ik naar welke plek op het scherm ik zou moeten kijken	☐	☐	☐	☐	☐
zie ik wat ik wilde zien	☐	☐	☐	☐	☐
raak ik snel de weg kwijt	☐	☐	☐	☐	☐
ben ik afhankelijk van de cameranummers en hoe de camera's opgehangen zijn	☐	☐	☐	☐	☐

13. Bij het nieuwe systeem gebruikte ik graag:
Meerdere antwoorden mogelijk.

- ☐ Het dubbelklikken om te wisselen
- ☐ De rechter muisknop om een pijl te tekenen
- ☐ De muisbesturing om de camera te sturen

Appendix C

SPSS Output

This appendix includes the SPSS output used in Chapter 8. The following list is the list of variables that relate to the questions found in Appendix B.

Single Camera Questionnaire	
Variable	Description
js_pan/m_pan	User satisfaction on camera panning.
js_tilt/m_tilt	User satisfaction on camera tilting.
js_zoom/m_zoom	User satisfaction on camera zooming.
js_expect/m_expect	Does the system do what the user expects?
js_learning/m_learning	Is this control easy to learn?
js_precision/m_precision	Does the camera listen precisely to your gestures?
js_smooth/m_smooth	Do the camera controls feel smooth while moving?
js_stable/m_stable	Is the camera image rumbling while moving?
js_speed/m_speed	Is the control responsive to user's gestures?
js_sa/m_sa	Situation Awareness: Is the user aware of its position in the world?
js_avg/m_avg	The average score of the above variables.

Multi-Camera Navigation Questionnaire	
Variable	Description
old_sa/new_sa	Situation Awareness: Is the user aware of its position in the world?
old_see/new_see	If the user knows what he/she is looking at.
old_seeTarget/new_seeTarget	Does the user see the target person directly after a camera switch?
old_seePath/new_seePath	Does the user know how he got from one camera to another?
old_adjust_inv/new_adjust_inv	Is the user likely to readjust the camera after a transition?
old_screenPos/new_screenPos	Does the user know where to look on screen after a transition?
old_expect/new_expect	Is the system behaving as expected by the user?
old_lost_inv/new_lost_inv	If the user was easily lost during the session.
old_numbers_inv/new_numbers_inv	Dependency on the camera numbers.
old_avg/new_avg	The average score of the above variables.

Test Measures (* = 1,single-cam / 2,multi-cam)	
Variable	Description
c*_tFind/n*_tFind	Time needed to find the target person.
c*_nLost/n*_nLost	Number of times that the user loses the target out of sight. Measured from the first finding to a point where the user is unable to see the target anymore.
c*_rStars/n*_rStars	The ratio of stars that the user has counted against the total number of stars that the user could have seen.
c*_rLost/n*_rLost	The ratio of time that the target was off screen.

Test Set 1: Single Camera Controls
Questionnaire T-Test

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	js_pan	4,13	8	,641	,227
	m_pan	4,00	8	,926	,327
Pair 2	js_tilt	3,88	8	,991	,350
	m_tilt	4,13	8	1,458	,515
Pair 3	js_zoom	3,75	8	1,035	,366
	m_zoom	4,25	8	1,165	,412
Pair 4	js_expect	3,75	8	1,035	,366
	m_expect	4,38	8	1,188	,420
Pair 5	js_learning	4,13	8	,991	,350
	m_learning	4,25	8	1,035	,366
Pair 6	js_precision	3,63	8	1,061	,375
	m_precision	3,75	8	1,282	,453
Pair 7	js_smooth	3,88	8	1,126	,398
	m_smooth	3,63	8	1,302	,460
Pair 8	js_stable	3,50	8	,756	,267
	m_stable	3,63	8	1,188	,420
Pair 9	js_speed	4,25	8	,886	,313
	m_speed	4,00	8	,756	,267
Pair 10	js_sa	4,00	8	,926	,327
	m_sa	4,63	8	,518	,183
Pair 11	js_avg	3,8875	8	,69783	,24672
	m_avg	4,0625	8	,91016	,32179

Paired Samples Correlations

		N	Correlation	Sig.
Pair 1	js_pan & m_pan	8	-,241	,566
Pair 2	js_tilt & m_tilt	8	-,185	,660
Pair 3	js_zoom & m_zoom	8	-,415	,307
Pair 4	js_expect & m_expect	8	-,378	,356
Pair 5	js_learning & m_learning	8	-,731	,039
Pair 6	js_precision & m_precision	8	-,184	,663
Pair 7	js_smooth & m_smooth	8	-,426	,292
Pair 8	js_stable & m_stable	8	-,557	,152
Pair 9	js_speed & m_speed	8	-,426	,292
Pair 10	js_sa & m_sa	8	,298	,473
Pair 11	js_avg & m_avg	8	-,559	,150

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	js_pan - m_pan	,125	1,246	,441	-,917	1,167	,284	7	,785
Pair 2	js_tilt - m_tilt	-,250	1,909	,675	-1,846	1,346	-,370	7	,722
Pair 3	js_zoom - m_zoom	-,500	1,852	,655	-2,048	1,048	-,764	7	,470
Pair 4	js_expect - m_expect	-,625	1,847	,653	-2,169	,919	-,957	7	,370
Pair 5	js_learning - m_learning	-,125	1,885	,666	-1,701	1,451	-,188	7	,857
Pair 6	js_precision - m_precision	-,125	1,808	,639	-1,636	1,386	-,196	7	,850
Pair 7	js_smooth - m_smooth	,250	2,053	,726	-1,466	1,966	,344	7	,741
Pair 8	js_stable - m_stable	-,125	1,727	,611	-1,569	1,319	-,205	7	,844
Pair 9	js_speed - m_speed	,250	1,389	,491	-,911	1,411	,509	7	,626
Pair 10	js_sa - m_sa	-,625	,916	,324	-1,391	,141	-1,930	7	,095
Pair 11	js_avg - m_avg	-,17500	1,42302	,50312	-1,36468	1,01468	-,348	7	,738

Test Set 2: Multi-Camera Controls
Questionnaire T-Test

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	old_sa	2,38	8	,916	,324
	new_sa	4,00	8	1,069	,378
Pair 2	old_see	3,38	8	,916	,324
	new_see	4,38	8	,744	,263
Pair 3	old_seeTarget	2,43	7	1,618	,612
	new_seeTarget	4,43	7	,976	,369
Pair 4	old_seePath	3,71	7	,951	,360
	new_seePath	4,29	7	1,496	,565
Pair 5	old_adjust_inv	2,25	8	1,035	,366
	new_adjust_inv	2,38	8	1,188	,420
Pair 6	old_screenPos	2,63	8	,916	,324
	new_screenPos	3,75	8	,707	,250
Pair 7	old_expect	3,25	8	,463	,164
	new_expect	4,13	8	,835	,295
Pair 8	old_lost_inv	2,71	7	1,380	,522
	new_lost_inv	3,29	7	1,496	,565
Pair 9	old_numbers_inv	2,00	8	,926	,327
	new_numbers_inv	4,00	8	1,069	,378
Pair 10	old_avg	2,7691	8	,67394	,23827
	new_avg	3,8681	8	,59683	,21101

Paired Samples Correlations

		N	Correlation	Sig.
Pair 1	old_sa & new_sa	8	,000	1,000
Pair 2	old_see & new_see	8	,183	,664
Pair 3	old_seeTarget & new_seeTarget	7	,181	,698
Pair 4	old_seePath & new_seePath	7	-,519	,233
Pair 5	old_adjust_inv & new_adjust_inv	8	,029	,946
Pair 6	old_screenPos & new_screenPos	8	,717	,045
Pair 7	old_expect & new_expect	8	,277	,506
Pair 8	old_lost_inv & new_lost_inv	7	-,035	,941
Pair 9	old_numbers_inv & new_numbers_inv	8	-,289	,488
Pair 10	old_avg & new_avg	8	-,062	,883

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	old_sa - new_sa	-1,625	1,408	,498	-2,802	-,448	-3,265	7	,014
Pair 2	old_see - new_see	-1,000	1,069	,378	-1,894	-,106	-2,646	7	,033
Pair 3	old_seeTarget - new_seeTarget	-2,000	1,732	,655	-3,602	-,398	-3,055	6	,022
Pair 4	old_seePath - new_seePath	-,571	2,149	,812	-2,559	1,416	-,703	6	,508
Pair 5	old_adjust_inv - new_adjust_inv	-,125	1,553	,549	-1,423	1,173	-,228	7	,826
Pair 6	old_screenPos - new_screenPos	-1,125	,641	,227	-1,661	-,589	-4,965	7	,002
Pair 7	old_expect - new_expect	-,875	,835	,295	-1,573	-,177	-2,966	7	,021
Pair 8	old_lost_inv - new_lost_inv	-,571	2,070	,782	-2,486	1,343	-,730	6	,493
Pair 9	old_numbers_inv - new_numbers_inv	-2,000	1,604	,567	-3,341	-,659	-3,528	7	,010
Pair 10	old_avg - new_avg	-1,09896	,92771	,32799	-1,87454	-,32338	-3,351	7	,012

Test Set 1: Single Camera Controls
Measures Statistics and Correlations

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	c1_tFind	14,1417	3	1,10837	,63991
	n1_tFind	21,1690	3	15,12279	8,73115
Pair 2	c1_nLost	,25	8	,463	,164
	n1_nLost	,75	8	,886	,313
Pair 3	c1_rLost	,0067	8	,01320	,00467
	n1_rStars	,9123	8	,07853	,02777
Pair 4	c1_rLost	,0067	8	,01320	,00467
	n1_rLost	,0105	8	,01646	,00582

Paired Samples Correlations

		N	Correlation	Sig.
Pair 1	c1_tFind & n1_tFind	3	,719	,489
Pair 1	c1_nLost & n1_nLost	8	,174	,680
Pair 2	c1_rStars & n1_rStars	8	-,451	,262
Pair 3	c1_rLost & n1_rLost	8	-,169	,689

Test Set 2: Multi-Camera Controls
Measures Statistics and Correlations

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	c2_tFind	18,4035	8	7,95839	2,81372
	n2_tFind	7,6091	8	2,28796	,80891
Pair 2	c2_nLost	3,88	8	1,642	,581
	n2_nLost	2,88	8	1,356	,479
Pair 3	c2_rStars	,6823	8	,27519	,09730
	n2_rStars	,8311	8	,25773	,09112
Pair 4	c2_rLost	,1457	8	,09955	,03519
	n2_rLost	,0808	8	,04347	,01537

Paired Samples Correlations

		N	Correlation	Sig.
Pair 1	c2_tFind & n2_tFind	8	-,334	,418
Pair 2	c2_nLost & n2_nLost	8	,762	,028
Pair 3	c2_rStars & n2_rStars	8	,341	,408
Pair 4	c2_rLost & n2_rLost	8	,454	,258

Test Set 1: Single Camera Controls
Measures Paired Samples T-Test

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	c1_tFind - n1_tFind	-7,02737	14,34673	8,28309	-42,66663	28,61188	-,848	2	,486
Pair 2	c1_nLost - n1_nLost	-,500	,926	,327	-1,274	,274	-1,528	7	,170
Pair 3	c1_rStars - n1_rStars	,05645	,14228	,05030	-,06250	,17540	1,122	7	,299
Pair 4	c1_rLost - n1_rLost	-,00386	,02277	,00805	-,02290	,01518	-,479	7	,646

Test Set 2: Multi-Camera Controls
Measures Paired Samples T-Test

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	c2_tFind - n2_tFind	10,79444	8,98603	3,17704	3,28194	18,30695	3,398	7	,011
Pair 2	c2_nLost - n2_nLost	1,000	1,069	,378	,106	1,894	2,646	7	,033
Pair 3	c2_rStars - n2_rStars	-,14880	,30617	,10825	-,40476	,10716	-1,375	7	,212
Pair 4	c2_rLost - n2_rLost	,06486	,08869	,03136	-,00929	,13901	2,068	7	,077

Test 1 Reliability

Joystick

Reliability Statistics

Cronbach's Alpha	N of Items
,903	10

Item-Total Statistics

	Scale Mean if Item Deleted	Scale Variance if Item Deleted	Corrected Item-Total Correlation	Cronbach's Alpha if Item Deleted
js_pan	34,75	42,500	,692	,894
js_tilt	35,00	42,286	,421	,908
js_zoom	35,13	40,125	,572	,899
js_expect	35,13	36,982	,845	,880
js_learning	34,75	38,214	,775	,885
js_precision	35,25	42,500	,367	,913
js_smooth	35,00	36,286	,821	,881
js_stable	35,38	41,125	,722	,891
js_speed	34,63	38,268	,879	,880
js_sa	34,88	40,411	,631	,895

Mouse

Reliability Statistics

Cronbach's Alpha	N of Items
,945	10

Item-Total Statistics

	Scale Mean if Item Deleted	Scale Variance if Item Deleted	Corrected Item-Total Correlation	Cronbach's Alpha if Item Deleted
m_pan	36,63	69,411	,815	,938
m_tilt	36,50	61,143	,858	,936
m_zoom	36,38	66,554	,785	,938
m_expect	36,25	65,071	,854	,935
m_learning	36,38	66,268	,920	,932
m_precision	36,88	64,982	,785	,939
m_smooth	37,00	63,714	,838	,936
m_stable	37,00	66,286	,783	,938
m_speed	36,63	74,839	,568	,947
m_sa	36,00	76,857	,630	,947

Joystick/Matrix View

Case Processing Summary

		N	%
Cases	Valid	6	75,0
	Excluded[a]	2	25,0
	Total	8	100,0

a. Listwise deletion based on all variables in the procedure.

Reliability Statistics

Cronbach's Alpha	N of Items
,811	9

Item-Total Statistics

	Scale Mean if Item Deleted	Scale Variance if Item Deleted	Corrected Item-Total Correlation	Cronbach's Alpha if Item Deleted
old_sa	22,83	34,967	,404	,804
old_see	21,83	29,767	,899	,745
old_seeTarget	22,50	23,500	,935	,717
old_seePath	21,33	30,267	,900	,748
old_adjust_inv	22,83	44,567	-,313	,887
old_screenPos	22,50	32,700	,610	,781
old_expect	21,83	41,367	-,100	,834
old_lost_inv	22,67	29,067	,673	,768
old_numbers_inv	23,00	32,000	,719	,769

Mouse/Prototype View

Case Processing Summary

		N	%
Cases	Valid	7	87,5
	Excluded[a]	1	12,5
	Total	8	100,0

a. Listwise deletion based on all variables in the procedure.

Reliability Statistics

Cronbach's Alpha	N of Items
,706	9

Item-Total Statistics

	Scale Mean if Item Deleted	Scale Variance if Item Deleted	Corrected Item-Total Correlation	Cronbach's Alpha if Item Deleted
new_sa	31,29	28,571	,135	,726
new_see	30,86	30,810	,016	,730
new_seeTarget	30,86	26,810	,377	,682
new_seePath	31,00	24,000	,364	,689
new_adjust_inv	32,86	30,476	-,037	,763
new_screenPos	31,43	26,619	,629	,658
new_expect	31,00	26,000	,649	,651
new_lost_inv	31,71	17,238	,872	,537
new_numbers_inv	31,29	22,238	,735	,607