Dissertation

Visual Computing in Virtual Environments

Marcel Lancelle

A thesis submitted for the degree
of Doctor of Philosophy (PhD)
in Engineering Sciences (Computer Science)
by Dipl.-Ing. Marcel Lancelle,
Graz University of Technology, Austria

Graz, 2011

Supervisor: Univ.-Prof. Dr. Dieter W. Fellner
Advisor: Dr.-Ing. Sven Havemann

Institute of Computer Graphics and Knowledge Visualization
Graz University of Technology, http://www.cgv.tugraz.at/

Abstract

This thesis covers research on new and alternative ways of interaction with computers. Virtual Reality and multi touch setups are discussed with a focus on three dimensional rendering and photographic applications in the field of Computer Graphics.

Virtual Reality (VR) and Virtual Environments (VE) were once thought to be *the* future interface to computers. However, a lot of problems prevent an everyday use. This work shows solutions to some of the problems and discusses remaining issues.

Hardware for Virtual Reality is diverse and many new devices are still being developed. An overview on historic and current devices and VE setups is given and our setups are described. The DAVE, an immersive projection room, and the HEyeWall Graz, a large high resolution display with multi touch input are presented. Available processing power and in some parts rapidly decreasing prices lead to a continuous change of the best choice of hardware. A major influence of this choice is the application. VR and multi touch setups often require sensing or tracking the user, optical tracking being a common choice. Hardware and software of an optical 3D marker tracking and an optical multi touch system are explained.

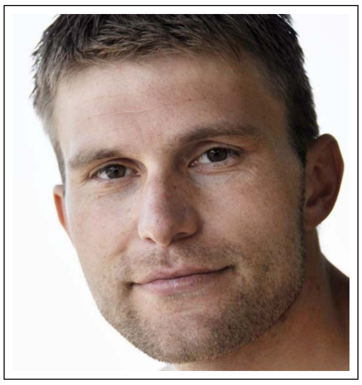
The Davelib, a software framework for rendering 3D models in Virtual Environments is presented. It allows to easily port existing 3D applications to immersive setups with stereoscopic rendering and head tracking. Display calibration and rendering issues that are special to VR setups are explained. User interfaces for navigation and manipulation are described, focusing on interaction techniques for the DAVE and for multi touch screens. Intuitive methods are shown that are easy to learn and use, even for computer illiterates. Exemplary applications demonstrate the potential of immersive and non-immersive setups, showing which applications can most benefit from Virtual Environments. Also, some image processing applications in the area of computational photography are explained, that help to better depict the captured scene.

Contents

Acknowledgements

Parts of the work for this thesis were done at ENSIMAG at the INPG, Grenoble, France in 2001-2002, the Computer Graphics lab of Braunschweig University of Technology, Germany in 2002-2004 and the HITLabNZ at the University of Canterbury, New Zealand in 2004-2005. The majority however was done at the CGV at the Graz University of Technology, Austria in 2005-2011. I thank my colleagues for collaboration and support. Fruitful discussions and talks with members of the neighboring ICG were helpful and interesting.

I thank my parents for their loving support as well as my girlfriend and friends for their understanding of my priorities for work, especially in the last year of this thesis. Finally, I thank my supervisors for advice and everyone involved in taking care of funding and administration.



The author. Photo courtesy of Oliver Sacherer.

1 Introduction

A desktop PC with keyboard and mouse is certainly quite universal. However, it is easy to see that for some tasks, it is not the best interface. Computing power has increased a lot in the past decades, but user interfaces have hardly changed. This well known finding among Human Computer Interaction (HCI) researchers has led to a number of user interface concepts for specific purposes.

This thesis focuses on the display and interaction of 3D content. How can visual perception of 3D data be improved? How is immersion into a virtual world possible? How can a user explore, navigate and manipulate the data in the best way? Which hardware is necessary? The content of this work shows steps on the way to answer these questions.

1.1 MOTIVATION

Users can comfortably work or play on a PC for many hours using a mouse and a keyboard. In most 2D and 3D applications, tasks can be accomplished quickly and precisely. Complex graphical user interfaces (GUIs) are possible and there are standard GUI elements such as text input fields or scroll bars that are commonly known. Especially pushed by the games industry, normal PCs are nowadays equipped with powerful 3D graphics hardware, developed and optimized for real-time rendering of shaded and textured triangle meshes.

Limitations of Conventional Desktop Setups. Many of such applications are or at least seem to be complex to novice users. The inhibition threshold may be too high, especially for people with limited experience with PCs. They may be overwhelmed by the interface and be afraid not to be able to use the system or even to break something. Capabilities of the programs are unclear and the interaction is often indirect and limited. Players of 3D games can be observed instinctively moving their head sideways in order to look around a corner, without the computer reacting on this subconscious request by the user. Two dimensional interfaces for three dimensional worlds lack a degree of freedom, resulting in a less intuitive control. Also, the interaction is performed dislocated from the displayed content, requiring a further indirection. Immersion, the sense of feeling to really be in a virtual world or 3D scene, is also limited. A very limited field of view and the lack of parallax and stereoscopic effects only allow a low level of spatial immersion.

Novel User Interfaces. For suited applications, we would like to immerse in 3D worlds, exploring and modifying them in an easy way, even for computer illiterates. We would also like to interact with large amounts of data for non-immersive applications in an intuitive way. Imaginative concepts for natural interaction in public spaces are given by Valli in [Val07], such as technology enhanced spaces that react on a user, provoking spontaneous interactions. A large variety of different devices exists, but how can they be used best? We tried out many different setups and applications in order to understand and analyze problems and find similarities to develop generalized solutions.

Computational Photography. In digital photography, the images are often still used like in analog film photography. However, with the help of image processing, some hardware restrictions can be bypassed by using multiple photos, usually taken with different settings. This area of *Computational Photography* slowly makes its way to the common users, panoramic image stitching being a good example. We would like to explore more of these new ways to make additional use of digital photos.

Contributions. This work presents three major contributions on the way to fulfill these goals:

- Hardware and software developments for the DAVE, an immersive environment.
- The HEyeWall Graz, a large high resolution multi touch screen.
- Improvements for applications in the area of Computational Photography.

1.2 COVERED WORK

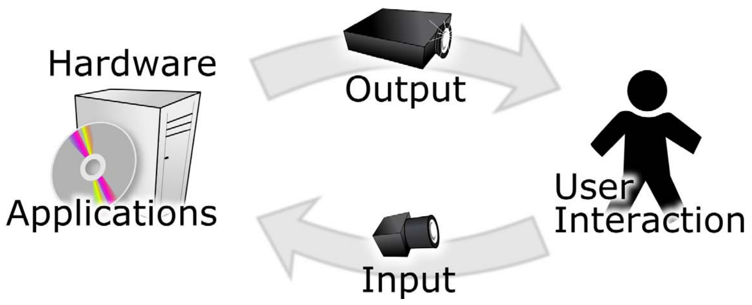


Figure 1.1: The basic components of a Virtual Environment. All of these aspects are covered in the following chapters.

The subjects illustrated in the figure above represent the building blocks of Virtual Environments and related applications. They are covered in the following chapters, but first, introductions to Virtual Reality and Computational Photography are given.

1.3 VIRTUAL ENVIRONMENTS AND VIRTUAL REALITY

The term 'Virtual Reality' appeared in the 1930*ies*, meaning a fictional world, later also described with cyberspace. Only for about 20 years it is also used for its second and now more common scientific meaning. Similar to the term *robot*, high expectations raised by science fiction literature and movies led to blatant misconceptions in the public, concerning the actual capabilities of VR. But also in science the term is fuzzy, as illustrated in the figure below.

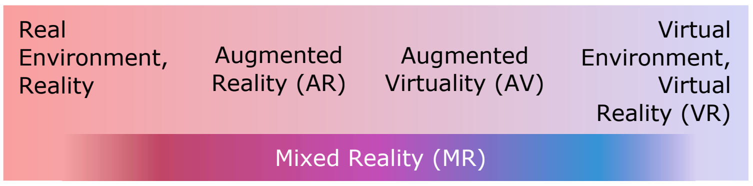

Real Environment, Reality
Augmented Reality (AR)
Augmented Virtuality (AV)
Virtual Environment, Virtual Reality (VR)
Mixed Reality (MR)

Figure 1.2: The *virtuality continuum*. Many varieties of mixed setups exist between complete reality to pure virtual reality. They are collectively referred to as Mixed Reality.

An example is a scene with a real person in a real room, reality. If with a semi transparent display, the real room is visible but instead

of the real person a virtual person, an avatar is virtually augmented into the room, it is called *Augmented Reality (AR)*. An opaque display with a virtual room and a camera image or 3D scan of a real person is called *Augmented Virtuality*. Finally, the term *Virtual Reality* is used when all visible objects are virtual. Combining both real and virtual objects, the environment can collectively be called *Mixed Reality*.
Virtual Environments have been around for a long time. The term itself is fuzzy, used both for describing the virtual world rendered by the displays as well as the complete setup including the hardware. They are used by the industry only for very specific problems and got rather out of fashion in the last decade. Important reasons are the requirement of a large room, high costs and effort for hardware and maintenance and a low software interoperability due to the diversity of setups. Often, not enough benefit is expected to justify the setup. However, these problems are worked on and hardware prices continuously drop. Eventually, VR will be more accessible to a lot more people and may see its revival. In fact, with 3D TVs, projectors and game console controllers and sensors, the basic hardware components already come into the living room. But which "killer application" will render the technology useful for a large number of people?
Augmented Reality is now much more popular, still being in the hype phase. Especially important is the fact, that a growing number of people owns a smartphone or a PC with a webcam and is already able to download and try an AR application. While we follow the developments in AR with interest, the field is sufficiently different to VR. Most notably, a live camera image is used and modified to superimpose virtual content. We do not employ such hardware and applications and in this work, do not cover any part unique to AR.
Commercial VR hardware and both free and commercial software frameworks are already available. Why can they not just be used? Unfortunately, it will not just work out of the box. We discuss and address the problems involved. Different ways for software development or ports are examined. The operation and maintenance of such systems is discussed. We improved the technology and interactions to solve the occurring problems.
With our software, building *and running* a virtual environment becomes *definitely affordable*.

1.4 COMPUTATIONAL PHOTOGRAPHY AND IMAGE PROCESSING

Photographic acquisition techniques become more and more popular for 3D scenes or high resolution 2D content. A few applications for image processing are shown that are loosely linked to VR. As an example, image alignment for high dynamic range (HDR) imaging is shown. HDR imaging and displays are also very interesting for VR. Similarly, image alignment is helpful to create content well suited for our multi touch setups.

2
Hardware Devices

Virtual environment hardware usually consists of several off-the-shelf components, combined with a few custom components for the remaining requirements. With a large variety, such setups in laboraties of research facilities and industry are individual, tailored to a specific problem or interest. Of course also availability and pricing influences the choice of the hardware.
This chapter first describes the purpose of the devices and lists a selection of such hardware setups. Then, in more detail, setups and devices are described that we tested or built on our own, mainly the DAVE and the HEyeWall Graz.

2.1 HARDWARE OVERVIEW

In this section, different types of both historical and current input and output devices are listed. With the emphasis on displays, devices for other human senses are presented in less detail.

2.1.1 The History of Virtual Reality

The ideas for today's VR devices are often surprisingly old. The historical and at that time revolutionary ideas are presented separately from the modern devices as it is easier to see the development in time as well as the combination of several devices resulting in a VR like setup.
The following description lists selected early important key inventions leading to modern audio and video technology, eventually making VEs possible. The development of computer technology is crucial, but out of scope of this document.

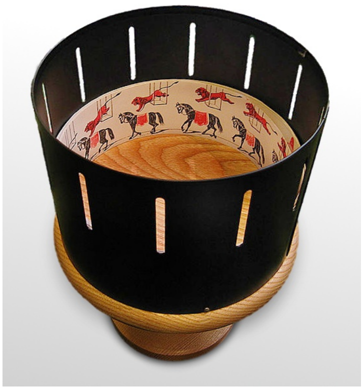


Figure 2.1: A zoetrope, a spinning cylinder with animation frames on the inside. Looking through the slits when it spins fast enough, an animation is perceived (modified image, original by [Dun04]).

An early form of a *zoetrope* (see side figure) was invented around 180 *AD* in China (see [Nee62]). In the mid 17th century the *magic lantern* was invented, the first projection system using a lens. In the 1830*ies*, animation using slits in rotating discs or cylinders with drawn animations were reinvented (*phenakistoscope* in 1832, *zoetrope* or *stroboscope* in 1834), founding the basics of today's animated displays. Sir Charles Wheatstone first presented a device to view a stereo image pair in 1838, the *stereoscope* [Cro92]. It lead to stereoscopic photography in 1844 and smaller viewing devices in 1850 by David Brewster. In the 1860*s*, 360 degree paintings were shown on panoramic murals surrounding the spectators.
Mechanical machines reproducing sound are around for over a millennium. In 1857 the earliest audio recording device was invented by Édouard-Léon Scott de Martinville. Surprisingly, it took until 1879 to combine projection and animation to a device called *zoopraxiscope*. Only later in 1882 the flip book animation was invented. In 1908 Lippmann introduced autostereoscopic photography using a lens array, called *integral photography* [Lip08]. In the 1920*s*, vehicle simulators were introduced [Ell94] and the Zeiss company used domes as projection screens [SL98].
Head Mounted Displays (HMDs) were first described for telepresence or stereo movie applications. A patent by Henry J. de N. McCollum [McC45] in 1945 first mentions an HMD and another patent by Morton L. Heilig [Hei60] in 1960 extends this idea. Even though both patents describe stereo HMDs, the first one that was physically built was monoscopic, the *Headsight Television System* [CB61] by Charles P. Comeau and James S. Bryan at Philco in 1961.
The Sensorama Simulator [Hei62], patented in 1962 by Morton L. Heilig is arguably the first VR installation, even though it is a mechanical device.

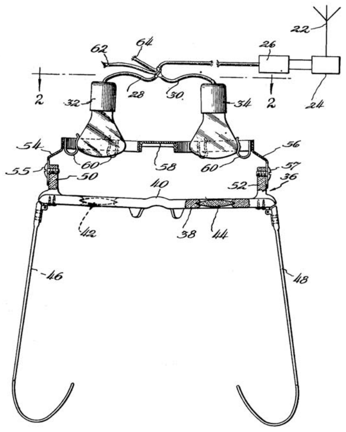


Figure 2.2: First patent for an HMD in 1945 by McCollum [McC45].

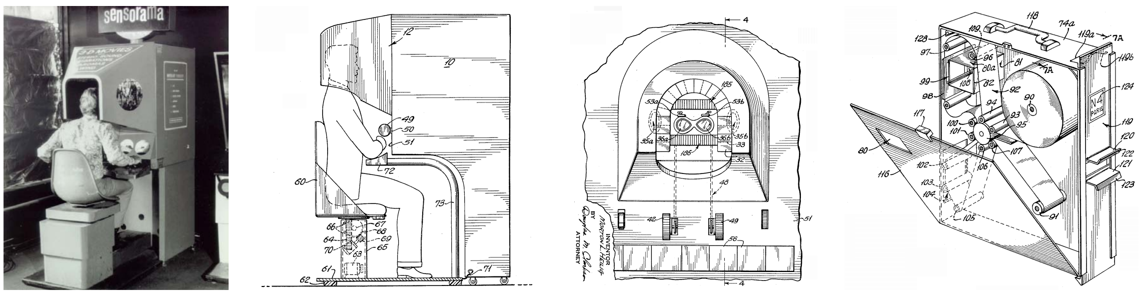


Figure 2.3: Sensorama by Morton L. Heilig in 1961/1962, the first interactive virtual reality setup, showing a motorbike ride through New York City. Four other films were available. The simulation with a stereo film also includes exhaust smell and wind. From left to right: photo, side view, frontal view of the head piece, the film box and mechanism [Hei62].

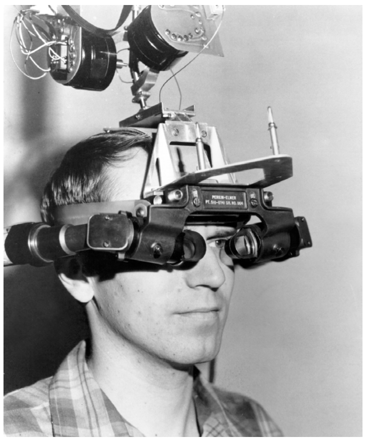


Figure 2.4: The *Sword of Damocles*, the first HMD to show computer generated content, by Sutherland in 1968 [Sut68].

In 1965, Ivan Sutherland presents his vision for future displays in *The Ultimate Display* [Sut65], pretty much describing a virtual environment. In 1968, he presents the *Binocular Omni Orientation Monitor (BOOM)* [Sut68], an early see through HMD. Together with its mechanical head position measurement system it is attached to the ceiling, therefore also called the *Sword of Damocles*. The 3D wand, still a state of the art input device in VEs, was first presented in the *Sorcerer's Apprentice system* [Vic72] by Donald Lee Vickers in 1972. It was used in combination with Sutherland's HMD.
From the 1970*ies* dome projection screens were used for immersive environments in research and military training [SL98].
In 1986 the virtual retinal display was presented by Kazuo Yoshinaka at Nippon Electric. The image is directly generated on the retina by a focused light beam. Today, technical problems still exist but the potential advantages of small, light weight devices with a low energy consumption are promising.
At SIGGRAPH in 1992 the CAVE system [CNSD*92],[CNSD93] was presented, an immersive cubic room with three rear-projection side walls and a down-projection floor. With HMDs, big problems of the time were issues due to tracking accuracy and latency. The CAVE solved these problems, as for a head rotation, the problems have a much smaller influence on the images. Further multi projector setups were developed to adapt to different needs, like the PowerWall in 1994 [Pow94], a high resolution tiled projector screen, the ImmersaDesk [CPS*97] in 1994, a 45° table setup and the InfinityWall [CPS*97] in 1995, an extension to the PowerWall.
In 2003 the first dynamic HDR display was presented [SWW03], providing improved contrast and brightness by several orders of magnitude compared to conventional displays.
Around 2004, hand held devices became powerful enough to be used as a platform for Augmented Reality. The low priced compact systems are also much more accepted in an every day situation than an HMD and quickly became the hardware of choice in that field.

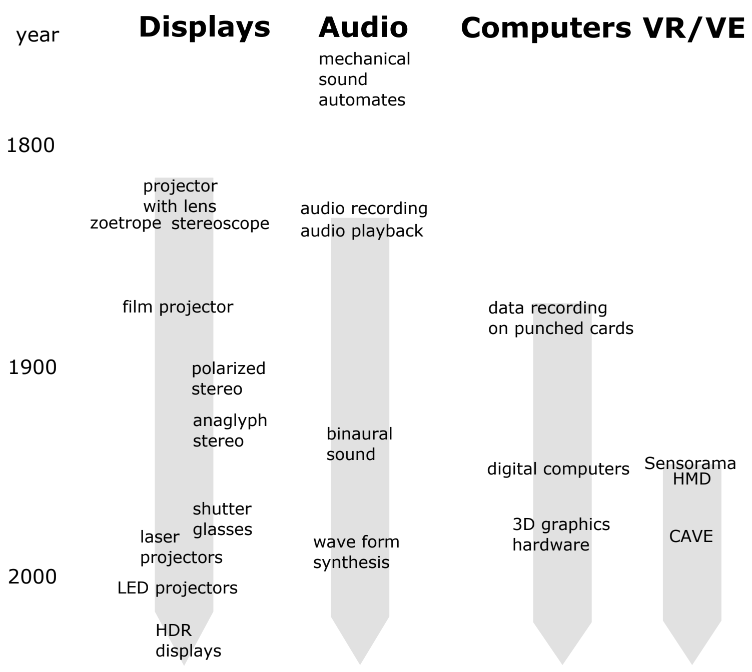


Figure 2.5: Time line of selected inventions that are important for VR.

The time line above summarizes the important historical developments.

2.1.2 Output Devices

Output devices enable the program to communicate its state to the environment. To express an information to a user, human senses must be stimulated. How this can be achieved is discussed below, focusing on vision.

2.1.2.1 Stimulating Vision: Interactive Displays

For a majority of VEs, the key aspect is the optical display, matching the importance of vision for people exploring an unknown environment. Today, there are two important categories of immersive VR displays: large display screen setups and HMDs, i.e. small screens directly in front of the eyes.
For human vision, many effects come into play. There is no display today that can match the capabilities of the human eye. Resolution, contrast, absolute brightness, field of view, stereoscopic view with accommodation and motion parallax are features that are available individually, but it is not yet possible to integrate all of them in a single device. However, many effects can be handled to a satisfying extend. The following figure gives an overview on important available display technologies and options.

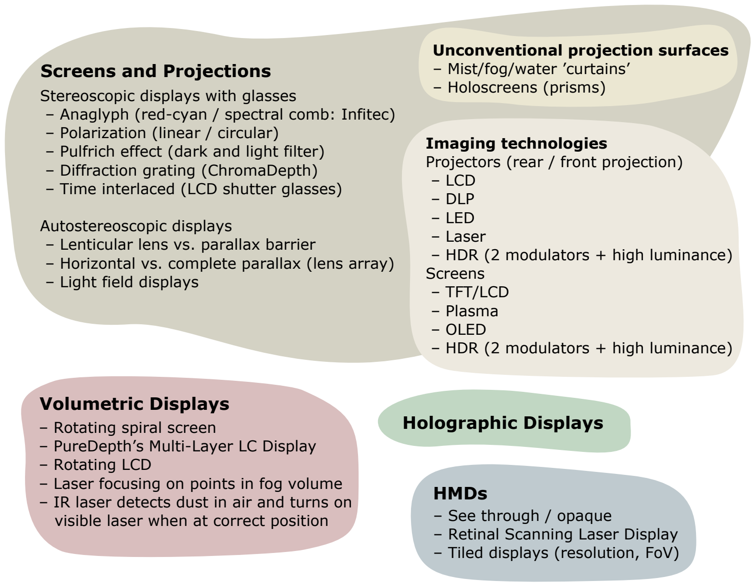

Screens and Projections
Stereoscopic displays with glasses
– Anaglyph (red-cyan / spectral comb: Infitec)
– Polarization (linear / circular)
– Pulfrich effect (dark and light filter)
– Diffraction grating (ChromaDepth)
– Time interlaced (LCD shutter glasses)
Autostereoscopic displays
– Lenticular lens vs. parallax barrier
– Horizontal vs. complete parallax (lens array)
– Light field displays
Unconventional projection surfaces
– Mist/fog/water 'curtains'
– Holoscreens (prisms)
Imaging technologies
Projectors (rear / front projection)
– LCD
– DLP
– LED
– Laser
– HDR (2 modulators + high luminance)
Screens
– TFT/LCD
– Plasma
– OLED
– HDR (2 modulators + high luminance)
Volumetric Displays
– Rotating spiral screen
– PureDepth's Multi-Layer LC Display
– Rotating LCD
– Laser focusing on points in fog volume
– IR laser detects dust in air and turns on visible laser when at correct position
Holographic Displays
HMDs
– See through / opaque
– Retinal Scanning Laser Display
– Tiled displays (resolution, FoV)

Figure 2.6: An overview of selected relevant technologies and options for (semi) immersive displays.

Many of these technologies can be combined. As an example, it is possible to build a stereoscopic or even an autostereoscopic HDR projection display, but to our knowledge it has not been constructed yet. A less useful example of an existing combination is a stereoscopic multi touch table. With the users standing next to the table, the large angle to the screen requires head tracking and two separate images for each observer. Furthermore, touch interaction can only occur on a single plane. Objects that appear above the screen surface are unsuited since a hand or arm will not be correctly occluded. It is hard to find an application to justify the amount of effort for this setup. Other interesting developments in display technologies are transflexive and e-paper displays that work well in bright environment light situations. By reflecting the existing light of the environment, they consume very little power.

Figure 2.7: Anaglyph and polarized glasses are common technologies to realize cheap stereoscopic displays.

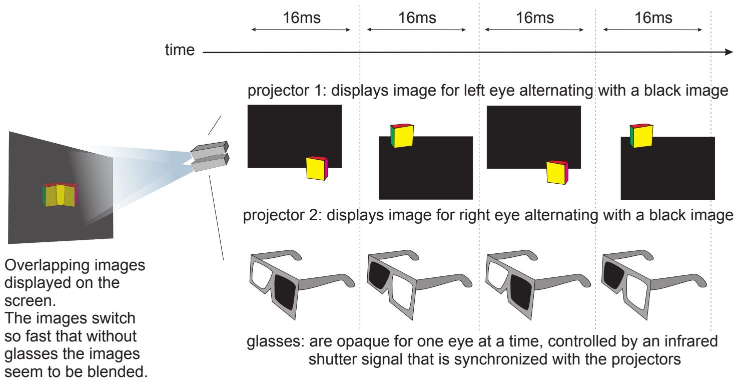

16ms
16ms
16ms
16ms
time
projector 1: displays image for left eye alternating with a black image
projector 2: displays image for right eye alternating with a black image
Overlapping images displayed on the screen. The images switch so fast that without glasses the images seem to be blended.
glasses: are opaque for one eye at a time, controlled by an infrared shutter signal that is synchronized with the projectors

Figure 2.8: Illustration of a time interlaced stereo display.

Projection technology. The major drawback of projectors is their limited brightness. Projectors operate within their thermal limits, light bulbs are expensive and need to be replaced after a few thousand hours of life time. Lasers as light source offer advantages but are even more expensive. However, this is about to change, as the laser life time increases significantly. LED projectors are now getting interesting, but still suffer from rather low light intensities.

Projection screen geometry. Most projection screens are flat, but also curved screens are used, e.g. in dome theaters or flight simulators. Compared to a CAVE, no visible edges exist. Even arbitrary geometry may be used as a projection surface. With cleverly designed content, this may lead to impressive results, as e.g. projections on buildings show with arts or advertisement. Finally, neither the projector nor the screen have to be fixed. With appropriate tracking, the projection can be adapted to remain registered relative to the screen surface.

Projection screen material. Screen materials can be divided into reflective screens for front projection and translucent screens for rear projection. Different gain factors are available and can be used to increase the brightness, but at the same time restricting the viewing angle. When using polarization, many translucent screens work more or less well, whereas for front projection silver screens are essential to retain polarization. Holographic screens are holographic grated prisms on a translucent sheet, directly reflecting light from a projection angle that can be quite different from the screen normal. Mist and fog can also be used as mostly flat screens or even volumetric screens.

Vergence-Accommodation conflict. Almost all current stereoscopic displays share a common problem: Accommodation is not handled correctly. While usually accommodation and vergence act together, in screen based or HMD devices the eyes focus on a fixed screen distance rather than the 3D object distance, causing discomfort and eye strain [RMWW94],[LIH07]. This is called the vergence-accommodation conflict.

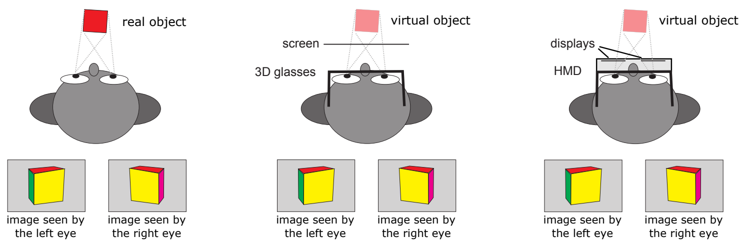

real object
screen
virtual object
3D glasses
displays
virtual object
HMD
image seen by the left eye
image seen by the right eye
image seen by the left eye
image seen by the right eye
image seen by the left eye
image seen by the right eye

Figure 2.9: Stereoscopic viewing (from left to right): the real stereoscopic effect, a stereoscopic display with screens and glasses and an HMD. The displays can provide correct vergence but fail to provide correct accommodation.

One possible solution is a fast switching lens and time interlaced rendering of several depth layers [LHH*09]. It uses a stack of lenses, each consisting of a birefringent material and a ferroelectric liquid-crystal polarization modulator. It seems quite possible to eventually integrate such lenses in glasses and to build e.g. a CAVE with this technology and fast projectors. A brighter display also reduces the problem, as the pupil gets smaller and the depth of field larger. Unfortunately, excessive costs make this simple idea for an enhancement infeasible for many projection setups.
Also, the retinal scanning laser displays have a related problem: Their image *always* appears in focus. Deformable mirror surfaces can provide physical defocus. Additional software generated defocus cues like software depth of field blurring may be a solution when vergence can be measured [SSKS03], [SSSF04].

Holographic and Volumetric displays provide better or correct accommodation and vergence cues [HGAB08], but need a lot of hardware effort. Also, they can only provide correct occlusion information for a single viewpoint at a time.

CAVE vs. HMD. CAVEs and HMDs are the primary choices for immersive VR setups. The following list shows the major differences.

In favor of the CAVE	In favor of the HMD
Tracking accuracy and latency problems have less impact in a CAVE.	CAVE needs a large room, takes up lots of space.
Size perception & level of immersion are better in a CAVE.	HMDs are easy to transport.
The CAVE provides a large field of view.	HMDs are usually less expensive.
In the CAVE the user can see the own body.	HMDs consume a lot less power.
A high resolution can easier be realized in a CAVE.	
HMD may need calibration before each usage.	
HMD is heavy and uncomfortable.	

Figure 2.10: Feature comparison of CAVEs versus HMDs.

High Dynamic Range Displays. HDR displays with constant illumination need two light modulators [See09]. Such a design needs lots of light and usually, most of the light is absorbed in the display. The key for efficient HDR displays is a modulated light generation combined with an additional modulator, like a coarse LED matrix as background illumination combined with an LCD [SHS*04]. Especially designed for HDR projection, a MEMS mirror array can be used to unevenly distribute the available light before it passes through the second modulator [HS08], [HSHF10]. Another option for an HDR display is to project an image on a modulated reflectance screen, like screens in e-books with e-ink, or just a paper print for a static image [BI08].

2.1.2.2 Output devices For Other Senses

Vision is arguably the most important sense to be stimulated in a virtual environment. However, it is reasonable to stimulate more senses than just vision in order to increase the level of immersion. There is no clear definition of a sense. The following list shows the human senses that are commonly recognized as such, together with important technologies that can be used to stimulate each sense, more details follow below.

Human Senses		Important Technologies
Vision (sight)	⇒	Displays, see above.
Audition (hearing)	⇒	Audio (synthesized or prerecorded) sound or voice: electro-mechanical speaker, ear phones, wave field synthesis, HRTF
Tactition (touch)	⇒	Haptic devices, fan wind, mist/ water spray, *Ultrasound Tactile Display* [HTNS09]
Gustation (taste)	⇒	Artificial aroma
Olfaction (smell)	⇒	Olfactory device/gun
Proprioception (limb locations and motions and muscular force within the body)	⇒	Locomotion devices, walking simulators (2D treadmill, VirtuSphere), swimming simulator
Kinaesthesia (acceleration)	⇒	Motion platforms and haptic displays, handheld force display by nonlinearly perceived asymmetric acceleration [AAM05]
Equilibrioception (balance)	⇒	Influence on balance by strong magnetic fields
Thermoception (temperature differences)	⇒	peltier hot and cold elements, electrical heating by current through resistor, infrared heat lamps
Nociception (pain)	⇒	Electrical shocks/stimulation
Sense of time	⇒	-

Figure 2.11: Human senses relevant to VR, with important technologies to stimulate them.

More senses exist, e.g. internal senses related to digestion. There are no devices that can perfectly stimulate even just a single human sense. Even though relevant, the details of the senses and how they work, especially in combination with the brain, are out of scope.

Audition. Natural sound is a sum of monophonic sound sources. However, distance and position of the sound source are important information for humans. The external ear and upper body act as a frequency filter. Also, the different positions of the ears lead in general to a time difference of the sound reaching each ear. Both effects are described by the *Head Related Transfer Function (HRTF)*. Convolving a virtual sound signal with the appropriate HRTF, the sound can be rendered and output with headphones for a 3D sound sensation. Using head tracking in a VE, the relative angle and distance of the sound source can be computed. However, sound transmitted through the body can not be simulated by that technique. Another approach is wave field synthesis, using a few dozen loudspeakers in an array. For a small target volume, the sound is approximately reconstructed. The major downside is a rather large and expensive hardware effort.

Olfaction. In general it is not possible to generate any odor by a combination of a few basic components. Usually, all odors to be displayed are produced individually and put in a different container. An overview, also discussing recording and transmission of smells, is given in [DHL01].
It takes a long time to replace one smell by another for a whole room. A solution for an olfactory display that is not attached to the user and allows some motion is shown in the image on the side. The smell is launched in the direction of the nose [YKNT03], [NNHY06].

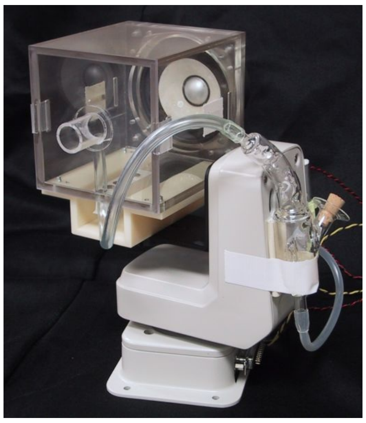


Figure 2.12: Olfactory device with nose tracking. Photo courtesy of [YKNT03].

Perceptual Illusions. The human senses are not perfect measurement instruments. Studies show a perception mismatch in VEs, e.g. objects appear larger or walked distances shorter than they should. Stimulating the senses in a clever way, a number of different side effects can be exploited to trick the perception. Senses can be manipulated in an unconscious way, as in walk redirection [RKW01] or exaggeration of head rotation [LFKZ01]. In some cases, such techniques allow to bypass or reduce hardware limitations.

2.1.3 Input Devices

The computer is supposed to react on input by users or other changes of the environment and thus needs to read sensor measurements of physical values. Sensors can be very simple, like a joystick button, or consist of a complex system like the *Global Positioning System (GPS)*. Below is an attempt to list the most important sensors and components for common measurement tasks that may arise in virtual environments and mixed reality setups.

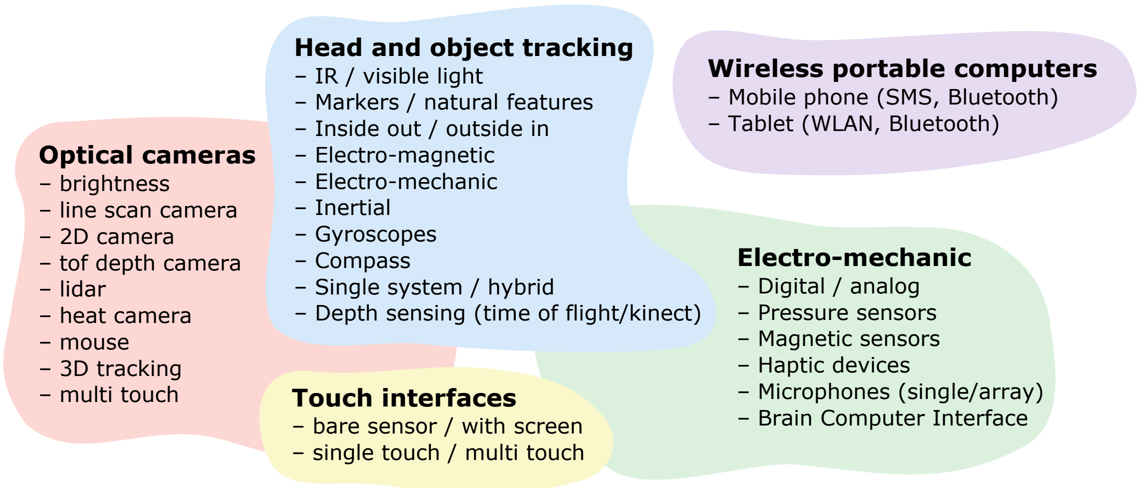


Figure 2.13: A selection of relevant technologies for input devices in Virtual Environments.

Of special interest are input devices for game consoles, as they are usually well tested, have a robust hardware, are available for a long time and are cheap. While they usually are not meant to be used in another context, game console controllers like the *Nintendo Wiimote* or the *Microsoft Kinect* are good examples for a large interest by the community that provides open source interface libraries and documentation by reverse engineering.

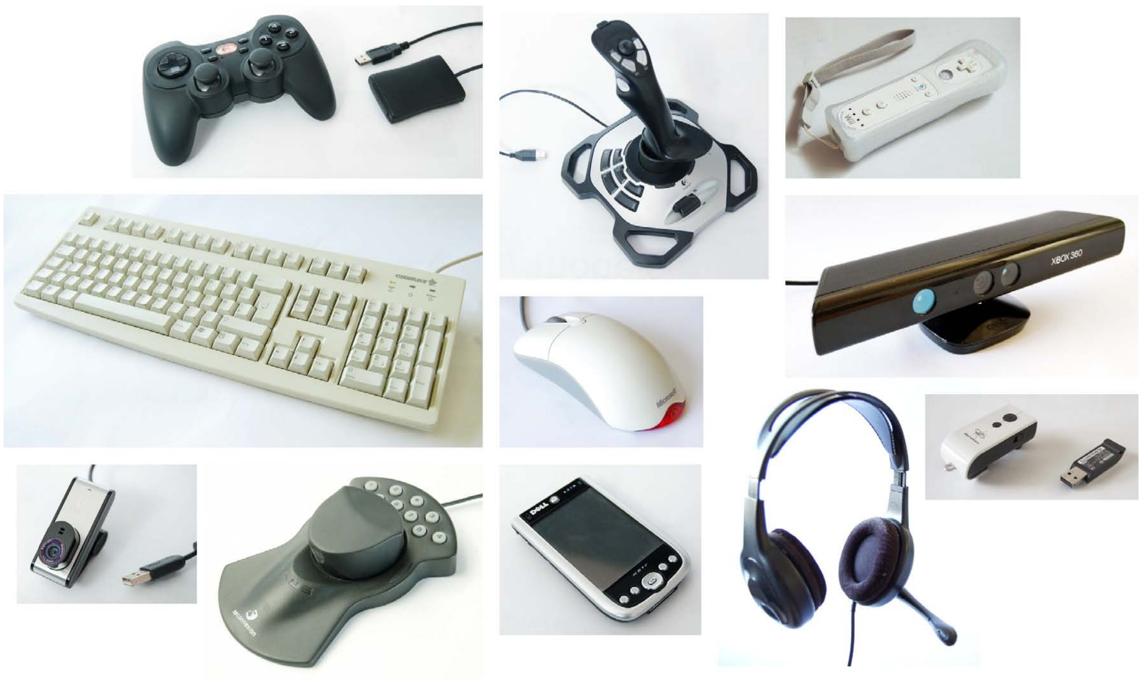


Figure 2.14: A selection of commercially available input devices. Depending on the setup, different devices make sense. Top row (left to right): wireless gamepad, joystick, wiimote. Center row: keyboard, mouse, kinect. Bottom row: webcam, spacemouse, PDA, headset, gyration mouse.

The gyration mouse and wiimote feature digital buttons and a 3D acceleromter, later enhanced by gyroscopes. The wiimote additionally contains an IR camera with onboard detection of up to four light source positions. The kinect has a color camera plus an IR camera and IR dot pattern projector for depth sensing. The spacemouse by 3Dconnexion enables a 6 degrees of freedom (DoF) input by rotating and translating the knob. To be able to use some of the devices for VR instead of for their intended purpose, the interface must be reverse engineered.

2.1.3.1 Single and Multi Touch

A variety of single and multi touch technologies exists. The ones that can be combined with a display are of primary interest for us. Referring to touch technologies, in this document we always imply the combination with displays, unless otherwise stated.
Around 2005, multi touch displays became popular again [SKO09]. New kinds of multi touch technologies are active topics in research and industry. The following table shows promising technologies at the time of writing, organized by the screen size.

Important Technology	Typical Screen Size	State of Development
Capacitive sensor array.	$\leq$ 32" (see text)	Works well and is commercially successful, e.g. Apple IPod.
LCD with each pixel also having an IR light sensor. (PixelSense)	5" - 40"	Samsung manufactures for Microsoft Surface 2, still expensive.
Line scan cameras at the edges of the display.	> 10"	Commercially available. Many fingers close together cannot each be detected.
LLP in combination with rear projection.	> 10"	Commercially available but no optimal technology invented yet.
FTIR in combination with rear projection.	> 10"	Commercially available but no optimal technology invented yet.
DI in combination with rear projection.	> 10"	Commercially available but lighting restrictions exist.

Figure 2.15: Promising touch and multi touch technologies at the time of writing. This table may be outdated soon as the development of new technologies is an active topic in research and industry.

Projective capacitance technology can be used on much larger screens than stated above. However, a visible coarse grid of horizontal and vertical wires run over the screen. A controller measures the change of capacitance in presence of e.g. a finger touch. To reduce moiré artifacts when placed in front of LCD screens, the wires can be layed out in a zigzag pattern. Only single or dual touch is possible with the current hardware and drivers, while controllers supporting six touches are worked on. Our tested product from *Visual Planet* also showed a noisy signal leading to errors. However, for public displays like in a shop window, this technology seems well suited. Also, the true multi touch capacitive technology with invisible sensor grids improves, announced is an increased screen size of 46" in late 2011.
The frustrated total internal reflection (FTIR) technology was first used for multi touch by Han et al. in 2005 [Han05]. A promising new technology is an LCD with a build in IR sensor for each pixel. Sharp, around 2007 (LCDs with Embedded Optical Sensors) and Samsung in 2011 (PixelSense, for MS Surface 2) presented such displays.
A choice of technologies is described in more detail in section 2.3.2. The illustrations on the next page show the basic principles of different kinds of optical touch technologies.

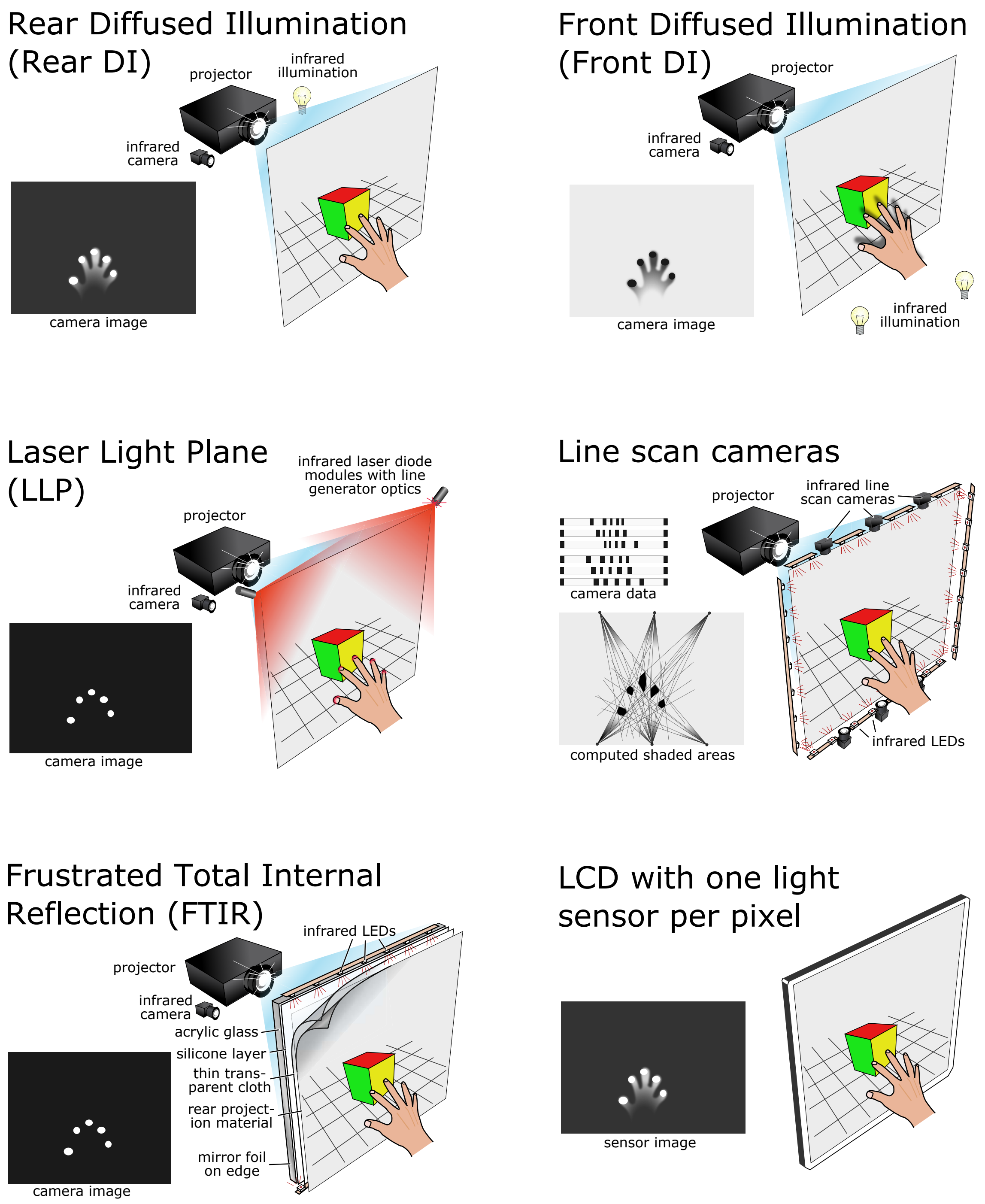


Figure 2.16: A variety of optical multi touch technologies, especially useful for large screens where capacitive technology is not available.

Technology	Environment Light	Force	Trigger Errors	Scalability	Passive Fiducials	Other Notes
Rear DI	needs controlled lighting	zero	clothes or other bright large objects near screen falsely trigger	good	yes	hard to find suitable screen material
Front DI	needs controlled lighting	zero	occlusion leads to fewer recognized fingers	good	yes	hard to find suitable screen material
LLP	robust	zero	triggers already 1–3 mm above screen surface	good	no	-
Line scan cameras	very robust	zero	several objects close to each other are detected as a single one	good	no	-
FTIR	ir blocking filter can be used to work better with ambient light	force sensitive	needs enough force to trigger	average	no	hard to manufacture large silicone layer
LCD with light sensor per pixel	probably ok except for direct sun light	zero	unknown	poor, directly depends on LCD size	yes	-

Figure 2.17: None of the optical touch technologies for large screens is perfect. This is an overview of their capabilities and properties.

Note that often setups may detect fiducial markers for tangible objects even if this is not directly supported, see section 3.2.4.

Back-of-Device Interaction Hardware A very interesting concept for very small touch screen devices is the back-of-device interaction [BC09], where the finger touches are sensed on the rear side of the device. This was invented to overcome the so called *fat finger problem*: A finger can easily covers the item of interest or even the whole screen, a precise click is not possible. With back-of-device interaction, the touch position is displayed on the display, e.g. the outline of the touching finger may be rendered as a virtual see through feedback.

2.2 VIRTUAL ENVIRONMENT SETUPS

This section describes arrangements of displays and projection screens in combination with additional technologies in order to realize a setup for a specific purpose. We tested a few promising setups, that were both interesting in terms of their capabilities as well as feasible in terms of cost and effort, but first we list a few existing setups at other institutions.

2.2.1 Related VR Setups

Current common VE setups in research and industry are still CAVEs and PowerWalls. The C6 at the Iowa State University is a high resolution six sided CAVE (see also [Fri99]). Per side, four 4k projectors are used for active stereo. Compared to a wall of our DAVE, it has 11× as many pixels and is 2.4× as bright.

Large high resolution walls use tiled displays. An example with stereoscopic rear projection is the HEyeWall at Fraunhofer IGD, Darmstadt, featuring a resolution of $8k \times 4k$ pixels. In case of LCD screen arrays, flat and curved arrays exist, like the *HIPerSpace* [DLR*09] wall display at CalIT2, a PowerWall providing 287 Megapixels. Besides the visible bezels, the LCD screens have the advantages of compact and cheap high resolution and high contrast displays with low maintenance costs. Vehicle simulators and dome theaters usually have a monoscopic projection on a curved surface. Most full dome projection systems are still monoscopic. However, many new installations are likely to support stereoscopic images. The CyberDome [NVH*04] is an immersive dome setup. Today, laser projections are available that provide a high contrast and sharp images of $4k \times 4k$ pixels, claiming that $8k \times 8k$ pixels are also possible. However, the currently used LCOS (Liquid Crystal on Silicon) modulators still have problems with synchronization for time interlaced stereo.

Figure 2.18: The Toshiba bubble helmet, one of many curiosities of VR hardware.

More exotic setups are e.g. cylindrical or spherical screens or variations of with partially planar screens. Examples are the *Star-CAVE* [DDS*09] with five side walls, or the *AlloSphere* [HKMA07], a large 360 degree projection on an almost spherical surface with the spectators standing on a bridge in the center. The *Elbe Dom* is a large almost cylindrical screen with a diameter of 16m and a height of 6.5m, where the users also stand on a raised platform. The *Varrier* wall [SMD*01] is a high resolution curved array of autostereoscopic LCDs. A fixed parallax barrier is installed in front of each display. The respective content is shown on the displays using head tracking to provide the correct stereoscopic view.

Cheap 3D consumer LCDs and projectors exist that work with time interlaced stereo and shutter glasses. They are described with *3D ready* and work with NVIDIA 3D Vision frame sequential signals. Few LCDs exist with polarization, where alternating columns have an accordingly oriented polarization filter. Also some autostereoscopic TV sets are on the market. Desktop setups with a stereo monitor and with or without head tracking are also referred to as *fish tank VR*.

2.2.2 DAVE Hardware Setup

As discussed earlier, the CAVE technology provides the most immersive VR experience. A first CAVE in Braunschweig was developed in order to make that technology more affordable and thus more available by greatly reducing its cost. It is called DAVE, for Definitively Affordable Virtual Environment. Building on our knowledge from the first DAVE in Braunschweig, a second improved DAVE was

constructed in Graz, also targeting continued research, education and eventually commercial applications.
The key hardware components of the current DAVE are eight render PCs and a master PC, whose graphics cards can be easily exchanged every few years, time interlaced stereo projectors that were modified to run in synchronization, a self made tracking system consisting of four infrared cameras connected to a PC and the screens held by a wooden frame.

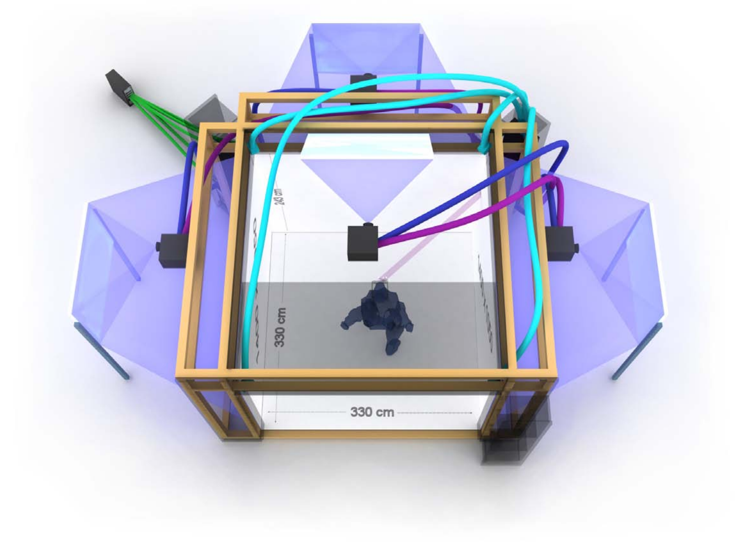


Figure 2.19: Second DAVE setup: a four-sided CAVE.

Topology and Geometric Setup. We considered a few geometrical possibilities for the DAVE in Graz. A setup with both floor and ceiling projections requires a rear projections and thus a raised transparent floor. We decided against such a setup out of room and cost restrictions. Also, without a ceiling projection it is easy to mount cameras for the tracking system. We eventually decided to once again use three rear projection side walls plus a floor projection from above. A disadvantage is the restricted field of view, especially for content displayed above the user and for tall people standing upright.

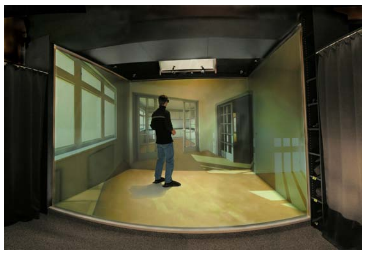


Figure 2.20: A user in the DAVE.

To maximize the DAVE size and still be able to use a part of the remaining room for a different setup, we use a different location of the projectors compared to the original CAVE and the first DAVE in Braunschweig, placing them just above the top center of the screens pointing outwards. They reflect their image on large coated mirrors back to the screen. Advantages are that the projectors are positioned rather close together, allowing short cables. They are attached to the wooden frame of the DAVE. Being up high, they are out of the way, do not receive a lot of dust and are not so much in risk of being touched while cleaning or during maintenance of the DAVE room. Unfortunately, our projectors needed a lot of maintenance in the first years and demounting the projectors takes some time. With a new version being available, we upgraded the projector hardware which made problems less severe. The mirror for the floor projection is placed above the front screen, so that the users' shadows are cast to the rear side where they are least noticed.
The DAVE is rotated by 45 degrees with respect to the room walls in

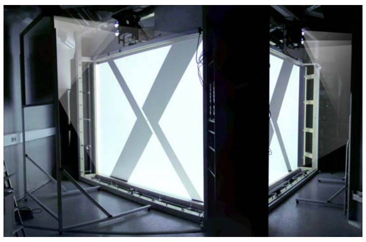


Figure 2.21: The DAVE seen from behind the screen. The light beams were added for illustration purposes.

order to minimize the used space. Another design choice was to build a visually pleasing surrounding entrance, completely hiding the technology and computers. While this is great for demonstrations, some parts of development and debugging are unnecessarily complicated.

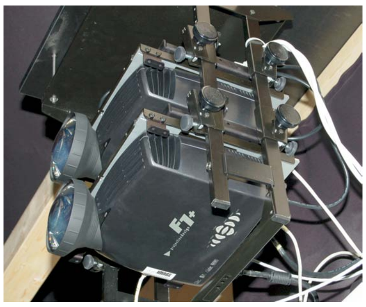


Figure 2.22: First version of our stereo projector hardware, still with two projectors in separate housings.

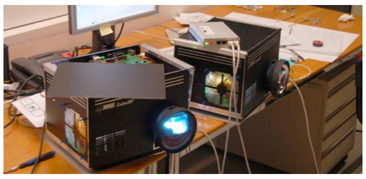


Figure 2.23: Debugging of stereo projector hardware in the second version. Only one lens and one lamp per screen are needed, but the housing still contains two projector mainboards.

Projector Issues. The projectors buffer images from unsynchronized video streams, thus not requiring expensive genlock graphics cards. The projectors are modified by *digitalImage* to synchronize to a chosen master projector by slowing down the system clock of the slaves. The parameters have to be calibrated by the manufacturer and locking to the correct signal usually takes around four minutes. This modification is also the reason why the communication to the projectors via the serial port is often corrupted.
A previous attempt to access the projectors via their LAN connections failed, as only some of the projectors reacted. Hardware and software was changed to use the serial port instead.
Another issue with the projectors is that they may loose some settings after a power cut, sometimes requiring the access of the system menu. This is only possible by pointing the remote control through the ventilation slots at the correct angle, only hitting one of the two projector units, typing a key combination as password with a fast timeout, while standing on a ladder. After a power cut, two hours must be scheduled for resetting the projectors. For the reproduction of colors, a lookup table is used. From time to time it may reset to a wrong default and the correct one must be reloaded. Other issues include lamps and contacts of connections. After a failed firmware update, an EPROM had to be replaced. A few times we had to send in a projector because of a broken transformer. Looking at the price for a projector or even the price for insured shipping, these experiences were rather cost intensive. Since mid 2008, another updated version is available, developed in collaboration with the projector manufacturer. We assume that most problems mentioned are solved in that product.

Screen Material and Frame. Our first solution for gapless projection in the corners was a custom made welding of the material including a loop held by an aluminum pipe which is pulled by rubber straps (see figure below, left side). The front wall of our first screen was 1cm too wide, what we only noticed when everything was set up, so the screen needed to be replaced. To ease the setup and realize a more transportable version, *digitalImage* developed a new frame with acrylic glass corner with a 45 degree phase (see figures below). Each projection screen is independent from the other ones. The setup of such a DAVE only takes one day.

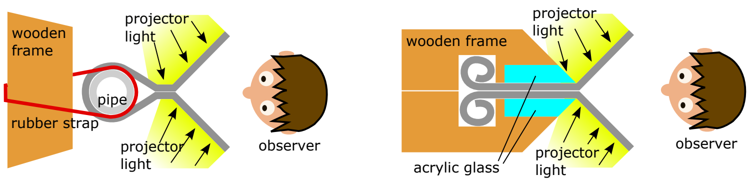


Figure 2.24: Old design on the left, with a single large frame. New design on the right, each side can easily be separated, e.g. for transport or maintenance.

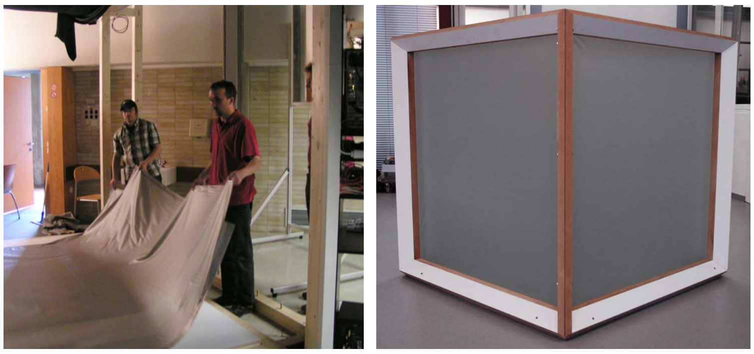


Figure 2.25: Left: With the old frame design, exchanging the screen material is a lot of effort. In this case, the front side was manufactured 1cm too wide and the whole screen had to be exchanged. Right: New frame design by digitalImage. Photo courtesy of Armin Zink.

Screen Color and Material. The frontal floor projection screen is reflective and should match the color of the rear projection screen of the side walls to get a similar brightness and dark level. With our gray rear projection material this was not respected at first. A photometric calibration to a common range may mean to loose a lot of contrast and dynamic range. To solve this problem, we tested a few materials and replaced the floor material to match the rear projection screen as good as possible. Following Majumder et al. [MS05a], smooth intensity transitions allow to retain a higher brightness. However, the strongest problem is that the projection surfaces are not Lambertian, i.e. the intensity depends an the view angle to the surface. Even though the rear projection screens have a low gain factor of 0.8, they suffer from hotspot problems, especially for wide angle projection like with our 1:1 lens. To lessen this effect, large fresnel lenses could be mounted just behind the screen. Finally, a dynamic software attenuation controlled by head tracking could effectively hide the transitions. However, even with our setup, many visitors do not or hardly perceive these edges and must be stopped from walking into the screen.
As the screens for the side walls are tightened also on the bottom, the floor of the DAVE is raised to about 10cm above ground. To make the step clearly visible, we highlight it with stripes in the background image and with an additional LED illumination. The lights are not directly visible from the inside and do not disturb the users. We ask users to wear felt slippers in order to keep the floor clean. This is inconvenient, especially for elderly people who might need to sit down to put on the shoes. A minor positive side effect is that visitors may be more aware of the sensitive equipment and may behave in a more cautious way.

Figure 2.26: An ipod touch is used for system control, realized with a web browser.

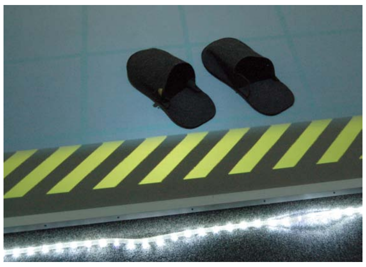


Figure 2.27: An LED strip is used to highlight the small step into the DAVE to avoid injuries. The floor projection is protected against dirt and scratches by felt slippers.

Mirrors. One problem we encountered with the mirrors is their nonplanarity. After loosening the tight mounting in their aluminum frame, it got a lot better. Still, this is a concern because our current software calibration can only correct linear distortions (see section 4.2.3.1).

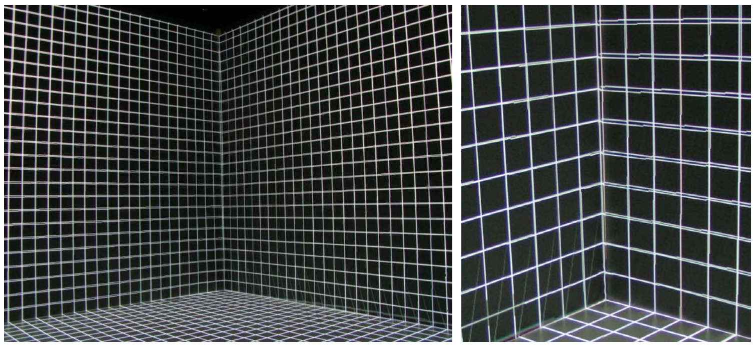


Figure 2.28: The linear calibration is not correct along the complete edge. Here, the corners are aligned but towards the center of the edge, the effects of nonplanarity of our large mirrors become visible. In addition, the screen is not exactly planar either.

Floor Projection Clipping. In the DAVE, none of the rear projected images shine on the same surface. Only the floor projection is a front projection and may also shine on the side and front walls. The angle is very steep and thus the light is considerably dimmer. We mechanically adjust the projector so that the image accurately aligns with the front wall. We use hardware blend masks for both sides, cropping the original image with a 4:3 aspect ratio to the square shaped floor.

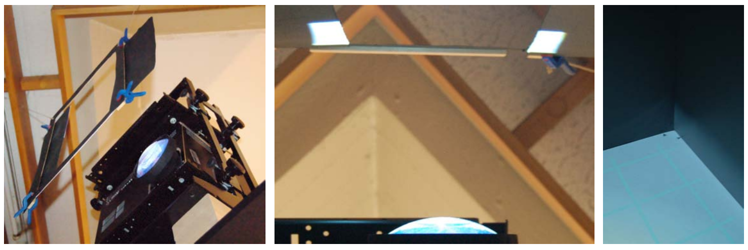


Figure 2.29: A wooden frame is suspended from the ceiling in front of the projector (left). Black cardboards can be adjusted to block unwanted light (center). Here, only the projector for the floor is turned on, demonstrating that no unwanted direct light reaches the front projection (right image, left side) and some still shines on the side wall (right image, right side).

A further and sharper reduction of unwanted direct light may be achieved with an additional software mask. It can easily be generated with a slight modification of the tool that we use to compute an undistorted background image for the operating system (see section 4.2.3.1). The image can be displayed on top of the image content, attenuating unwanted light. This is already supported by our frameworks (see section 4.2.3.4), but as this effect is rarely noticeable, we have not implemented this yet.

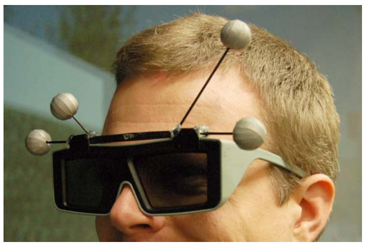


Figure 2.30: Lightweight shutter glasses with retroreflective markers for head tracking.

Stereoscopic Shutter Glasses. When the DAVE was build, we used relatively heavy and very expensive shutter glasses. Around 2007 much cheaper and lighter glasses were available. We developed an exchangeable mounting of the reflective markers, so that glasses can be cleaned or swapped easily.

Cheap Home Theater 3D Projectors. An interesting recent change is caused by the home cinema market, the now rapidly growing stereo sector. In around 2009, the price for time interlaced 3D TV sets and projectors dropped dramatically, because consumer devices were introduced, thanks to another recent 3D movie hype. While resolution and brightness do not match high end professional projectors yet, they are orders of magnitudes cheaper and lead to a vaster availability of stereo hardware in public. As they synchronize on the video signal, multi projector setups require genlock graphics hardware or a multi headed graphics card with several outputs.
For a setup like the DAVE in Graz, prices are compared between the existing setup and a fictional one using the new home theater projectors, with prices from 2011. For enhanced resolution, we consider a setup with two tiled projections per wall for the fictional setup.

	Professional	Home Theater
Number of projectors	one per side	two per side
Total number of pixels	5.5M	7.5M
Total brightness	7,500 AL	24,000 AL
Total price for projectors	75,000 EUR	8,000 EUR
Total price for graphics cards	2,000 EUR	8,000 EUR

Figure 2.31: Price calculation comparing a DAVE setup with professional 3D projectors and with home theater projectors.

It is clearly visible, that for a much brighter system and slightly higher resolution, the price is much lower. Including also the price of updating graphics cards three times over the following years, the costs are still halved.

Comparison to an HMD. We once had the occasion to borrow an HMD for a week. We directly compared it to our DAVE for an indoor architecture application. Especially, visual immersion and size perception were of interest. The HMD was an eMagin Z800 3DVisor with 800x600 pixels per eye, a horizontal field of view of about 33 degrees and a weight of 227g. In the short time we did not manage to get the included inertial tracker to work but used the DAVE tracking instead. While the stereoscopic effect was visible, the impression was far inferior to the DAVE. The exact reasons for this are not known. Speculations by HMD experts are a combination of imperfect optics, a small field of view, a larger influence of tracking latency and the vergence-accommodation conflict.

Brain Computer Interface. In collaboration with the Laboratory of Brain-Computer Interfaces at the Institute for Knowledge Discovery, TU Graz, we connected a brain computer interface (BCI) to the DAVE. For psychological experiments, BCI signals were evaluated and simple commands send to the DAVE server via a network connection, allowing limited navigation trough a scene (see section 5.1.1.5 and section 6.3.10.1).

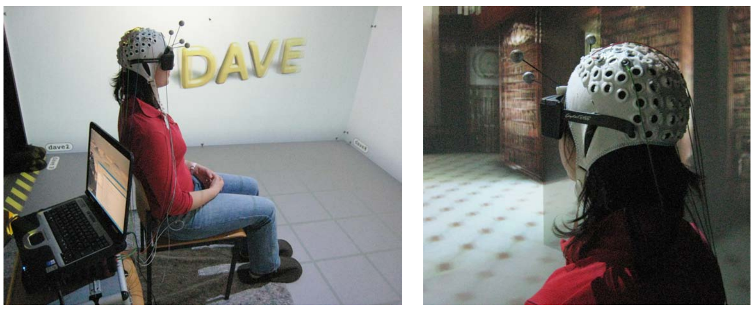


Figure 2.32: The brain computer interface (BCI) and a test subject in the DAVE. Electrodes are placed on the head to measure electroencephalogram (EEG) signals. Recognizing previously learned patterns, the subject is able to navigate through a 3D scene in a very limited way.

2.2.3 HEyeWall Hardware Setup

As of today, projector arrays are the best way to realize a scalable seamless display. Large high resolution displays are possible. Common drawbacks include the colorimetric and photometric differences of the individual projectors, caused by lamp manufacturing tolerances, lamp wear and LCD color changes with LCD projectors. This results in tiling artifacts so that the transitions from one projector to the next become visible, especially for low frequency content like a single color background. While this can be partially corrected by a software calibration, the brightness also depends on the viewing angle in case of a rear projection setup. This is worse for wide angle lenses and projectors without lens shift that cannot be put closely together. The HEyeWall in Darmstadt is an example for such a display, one of the leading large screen installations, a $5m \times 2.5m$ stereo screen using 24 projectors. To address the problem, we build the HEyeWall Graz, a smaller scale experimental setup as a possible successor. It has 3 by 2 projectors on a $4m \times 2m$ screen, plus an additional bright low resolution projector that covers the whole screen. The figure below shows an illustration of the setup.

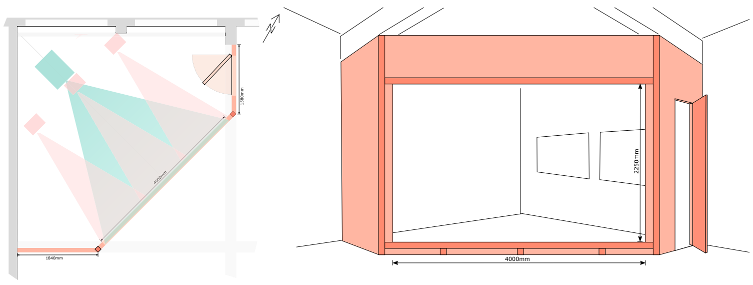


Figure 2.33: The HEyeWall setup in Graz, from the top (left) and from the front without the screen (right). A large projector (green) with 1024×768 pixels and 15,000 AL projects on the whole screen. 3×2 tiled projectors (red) with 1400×1050 pixels and 3,500 AL each are set up in an array.

The idea is that the large projector handles the low frequencies, with the small tiled projectors filling in the high frequency details (see section 4.3 for more information). The HEyeWall Graz is designed for an interactive display of high resolution 3D content for multiple people. For input without the need for additional devices we chose a multi touch interface solution. A rear projection setup fits well to the needs of multi touch input, as users of a front projection would occlude the content when standing in front of it or interacting with it. Due to the short throw lens of the large projector, we use a very low gain screen, like in the DAVE.

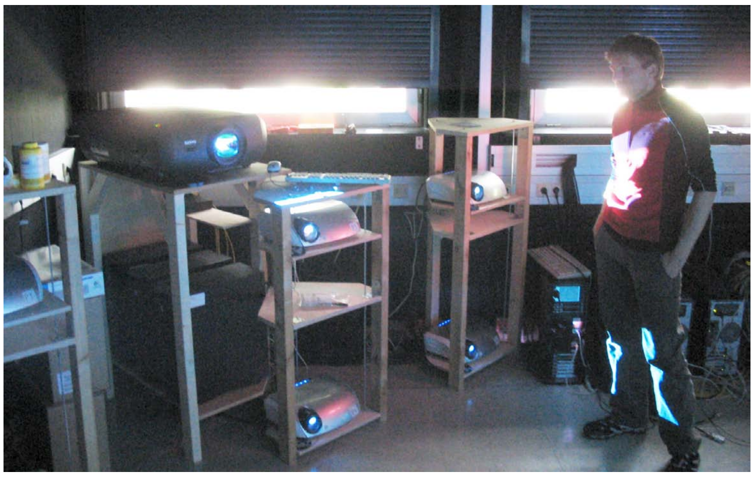


Figure 2.34: Most of the HEyeWall projectors: 6 small tiles and one large projector covering the whole screen.

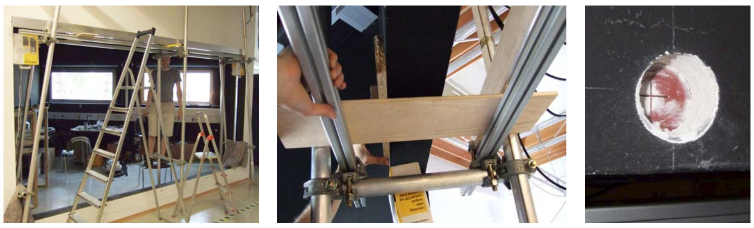


Figure 2.35: To attach the screen to the frame, we drilled holes in the screen and the frame. Here, preparations are shown for drilling the holes. It was a challenge to be accurate enough.

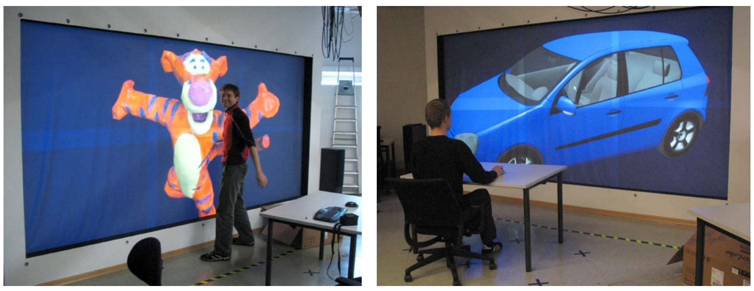


Figure 2.36: The first applications tested on the HEyeWall.

Another early idea for the HEyeWall was to use HDR projection. We experimented with an LCD as secondary light modulator. However, the display has an anti glare surface that diffuses the light too much. We also realized that it would not be feasible for us to achieve a sufficient light intensity.

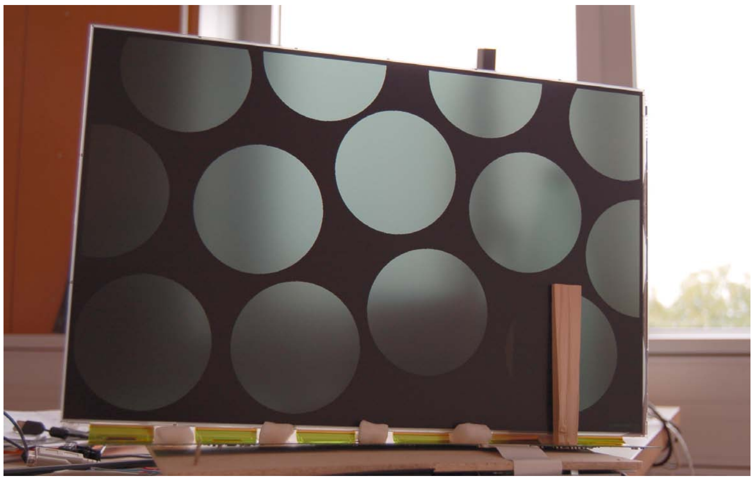


Figure 2.37: An LCD was disassembled to be used as a component for an HDR display. Here, white circles on a black background are displayed.

2.2.4 Polarized Stereo Rear-Projection Wall

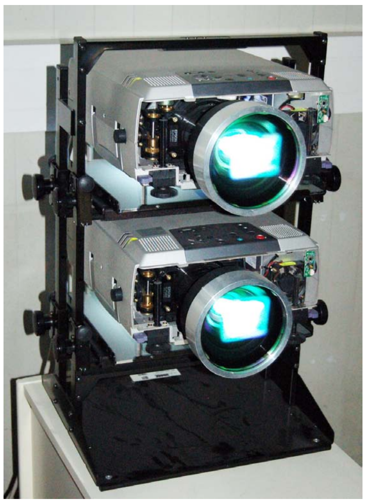


Figure 2.38: A stereo rear-projection setup with two projectors and linear polarization filters attached to the lenses.

A cheap alternative to expensive 3D projectors used to be a setup with two identical projectors but different polarization. Usually, these projectors are vertically stacked due to the form factor of the projector housings, trying to get the lenses together as close as possible. In that case, lens shift capability may not be required, a feature often only available on expensive projectors. Caution must be taken with the ventilation, such that the heat dissipation is not blocked with such stacked setup. Polarized filters are a cheap way to provide stereo separation. A silver screen is necessary when used with front projection to retain the polarization. Similarly, not all rear projection screens retain polarization and a suitable material must be used.

In a minimalistic setup we use two small projectors with washers glued on the bottom projector to hold the feet of the top projector. Polarized filters from photographic equipment are superglued to the lens. The setup is very portable. The images are quick to align, with basically one degree of freedom depending on the screen distance. We also have a very similar static rear projection setup with larger projectors (see image on the side), where we added a head tracking system as in the DAVE. However, we use the wall for larger groups of visitors and thus, almost never activate the head tracking.

In both setups we do not achieve an exact alignment of the images. When wearing the polarized glasses, the eyes can easily compensate for the small misalignment. However, when working with monoscopic applications and not wearing the glasses, the image looks blurred.

For linear polarization, ghosting appears if the head is tilted. To avoid this, circular polarization can be used. A projector with time interlaced projection does not have this problem and also does not require a special silver screen.

2.2.5 Future Setups

Two setups are presented that were designed for particular requests and will hopefully be realized in the near future.

2.2.5.1 Mini Dome Concept

Price drops for ultra short throw projectors make new setups feasible. One special problem from the industry comes from film production studios for full dome (hemispherical) projections. During production, a preview on a hemispherical projection is very important to test the impression. However, the studios are not always located close to a respective dome installation, the venue is expensive and coding the video to the required format takes a long time.
We designed a small cost effective projection setup covering a hemispherical viewing angle, that can be installed in an office and allows a quick preview. The idea is to use a front projection in an upper corner of a room, as illustrated below.

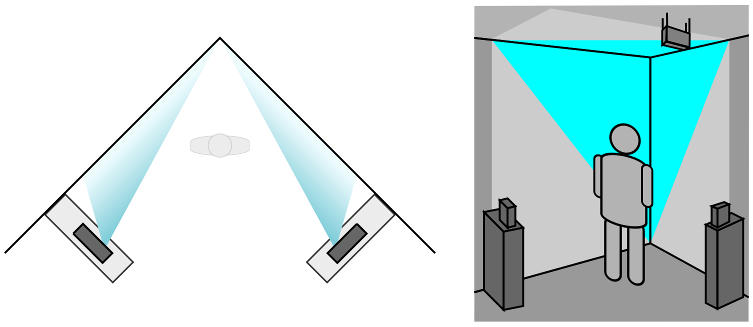


Figure 2.39: The Mini Dome setup, our novel concept suited for testing hemispherical film productions.

With only three off-the-shelf off-axis ultra short throw projectors it is possible to get a hemispherical projection that is not shadowed by the viewers or other projectors. Also, using flat surfaces, the image is perfectly sharp without expensive optics. By using front projection, very little space is necessary for the setup. The same software as in the DAVE can be used both for calibration and display. For testing stereoscopic movies, anaglyphic stereo is probably good enough for testing the 3D effect. Note that for stereoscopic rendering, the footage can not be the same as for the theater, as the screen distance is different and consequently, the eye separation must be different.

2.2.5.2 Car Simulator Concept

We design the display setup for a car simulator of the Institute of Automotive Engineering. If the funding works out, the final system will be mounted on a hexapod motion platform to partially simulate acceleration forces. Unconventional requirements are a stereoscopic setup as well as gaze tracking. Also, a very large room may not be available. As driver reactions at high driving speeds will be evaluated, distant objects are of interest and thus, the resolution must be reasonably high.
To estimate the necessary field of view and the number of displays for different configurations, we use a 3D modeling application and place

a cylindrical screen around a car model, centered on the driver's head position. By placing light sources at a few common head positions, the field of view can be observed on the shadow map of the cylinder. The most promising configuration seems to be a tiled display with 8×3 LCD screens and parallax barrier in combination with head tracking, a Varrier display [SMD*01], [PKG*07]. A drawback are the visible bezels, even when using monitors that are designed for tiled displays. By using the parallax barrier, no genlock hardware is necessary, an important cost factor for this setup.

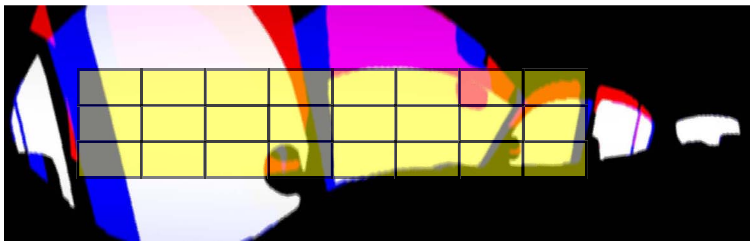


Figure 2.40: Visible field of view from a driver in a car, projected on a cylinder and unwrapped for this illustration. White, red and blue represent slightly different head positions. With this display setup with 8×3 autostereoscopic monitors (yellow), most of the important field of view is covered.

2.3 CUSTOM DEVICES

To explore new ways of interaction and to better meet the requirements in a VE, we also build custom devices. In some cases commercial solutions are available, but too expensive. Our goal is to use affordable hardware if possible and also demonstrate interaction with a prototype.

2.3.1 Optical 3D Tracking System

For many VR displays, the user's eye positions must be measured to provide motion parallax and an undistorted view. It is common to actually only measure the head position and orientation and derive estimated eye positions.

6 DoF Tracking Technologies Electromagnetic 6 DoF tracking systems are available for a long time and today also wireless systems exist. They are portable and easy to set up and do not require a line of sight to the sensor. A major problem are metallic objects or other magnetic fields near the tracking device, leading to distortions. For acoustic tracking systems, the number of emitters does not easily scale while maintaining a high update rate. Mechanic tracking devices restrict the motion of the user. Currently, optical tracking systems are the best choice for tracking in a CAVE. They commonly use multiple cameras to triangulate markers. A good system with six cameras costs around 5,000 EUR (mid 2011), see section 3.1.1 for more details. Important applications for the high end systems are motion capturing and virtual camera systems for movie production.

In 2005 as the DAVE in Graz was built, the cost was about 50,000 EUR for a commercial tracking system. As there was no affordable optical tracking system available, we decided to build our own system, with hardware cost of around 5,000 EUR. For a single system, the necessary software development is a large effort, but for multiple systems it pays off. An idea was to release the system as open source to enable research and profit from developments of other groups. The software will be described in section 3.1. Compared to the electromagnetic tracking system of the DAVE in Braunschweig, we achieve a notably higher update rate and lower latency, a higher accuracy, and only need lightweight wireless markers for tracking.

2.3.1.1 Illumination Setup

By far the easiest way to detect markers is to setup illumination so that on a black background, the markers appear as bright spots (or vice versa). There are many ways to differentiate markers, e.g. patterns, natural features, different colors or time coded active markers. Limitations of these approaches are the maximum number of different markers, physical size or image resolution, limited viewing angles or frame rate limitations. Using active markers, i.e. IR LEDs require little power and only a short exposure time is necessary for the cameras. However, markers that should be visible from all sides require multiple LEDs. Moreover, cabling and batteries are necessary and make the tracked object heavier and less convenient to use. Thus, we choose to use a common and simple but effective passive marker setup.

Passive Markers. For markers, wooden or cotton balls, taped with retro reflective material are used. They are attached to the tracked object with lightweight and robust carbon sticks. To differentiate tracked devices, we use so called *targets*, a fixed constellation of three or more markers (see Figure 2.8). Besides identification we also compute the orientation, giving all six degrees of freedom. In the DAVE, we mostly use one target attached to the glasses and another one attached to a handheld input device, e.g. a joystick. In the past years, our joystick was dropped about a dozen of times, mostly by visitors. Sometimes, the carbon sticks break but can be replaced easily. The target then needs to be recalibrated.

Infrared Illumination. To make the markers visible, infrared (IR) LEDs (see section 2.3.3 for details) are placed closely around each camera lens, so that their light is reflected well into the lens by the retro reflective material.

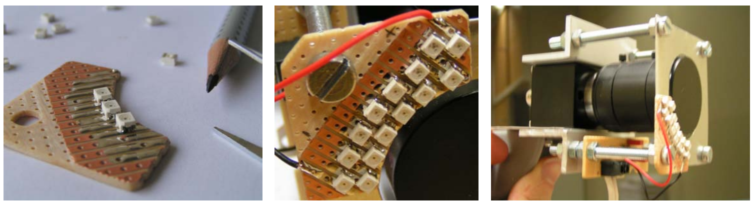


Figure 2.41: Infrared illumination with SMD LEDs. In addition to the housing on the right, an IR filter is attached in front of the lens to filter out visible light.

Retroreflective Material. To test the importance of the LEDs being close to the camera lens, we measured the reflectivity of our 3M^{TM}Scotchlite^{TM}tape. For practical reasons we used a torch with visible light, pointed the focused light to the tape and took a photo of the reflected light. The results in the figure below show that it is indeed very important to place the LEDs as close around the lens as possible. As an example, for a marker distance of 2m, the high reflectivity within 0.3 degrees is equivalent to a radius of 1cm. The retro reflective tape surface contains glass beads made out of dense glass that focus light on the surface just behind, so that the light is reflected back almost to its origin. Since we use light with a higher wavelength, we expect a slight defocus due to a lower refractive index, slightly flattening and broadening the curve shown.

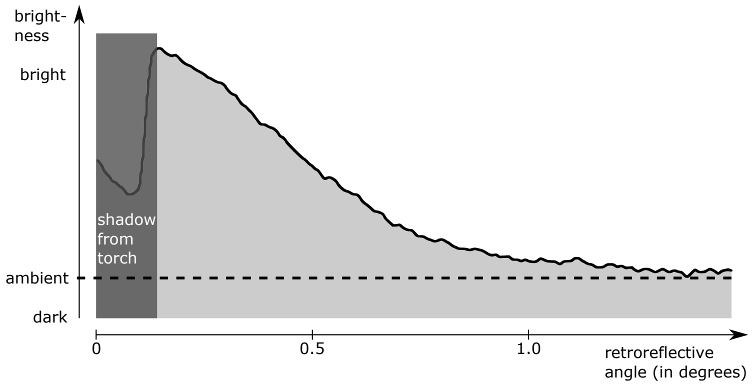


Figure 2.42: To get an idea on the angle dependent reflectivity, a photo of a light beam reflection from the retro reflective material was taken. The scan line shows the small reflection angle. On the left, the shadow of the light source hides the brightness. Note that the camera response curve is not calibrated.

The results above confirm the assumption, that the outer ring of LEDs helps considerably less than the inner ring for close markers. However, the far markers appear less bright and are the ones that need additional light, so that arrangement has the positive effect to help getting a more equal brightness for close and distant markers. Since we use distances of up to 4m in the DAVE, it would still help to move both rings closer to the lens.

2.3.1.2 Outside-In vs. Inside-Out Tracking

Most VR setups use outside-in tracking. This means that cameras outside the tracking volume track an object inside the volume. Another approach is to use a camera on the tracked object that recognizes markers attached outside the tracking volume. With that camera image, the camera position can be computed. This approach is better suited for accurate orientation but less accurate translation values. It is often used for Augmented Reality. A notable exception is the HiBall tracking system [WBV*99]. It is precise and fast and it detects a coded light by IR LEDs integrated in the ceiling. Using more of these LEDs, the system can be easily scaled to large rooms. In combination with an HMD, this enables natural walking in a large area.

2.3.1.3 Sensor Fusion

No tracking system is perfect. A clever combination of multiple technologies can lead to much better results, especially when sensors are likely to fail temporarily. This approach is often used in robotic navigation or AR. For VR, a good combination to improve the optical tracking in the DAVE might be accelerometers and gyroscopes. A higher update rate and reduced lag are possible, as the acceleration sensors and gyroscopes always provide data and can easily send data with over 60 Hz and a low latency. Integration errors over time can be compensated whenever the optical tracking detects the respective target.

2.3.2 Single and Multi Touch Setups

On small devices, capacitive touch technology works very well, but for large screens there is no perfect touch screen technology available. A few promising technologies were tested in order to explore and improve their capabilities.

2.3.2.1 Diffused Illumination setups

For Aichi, the world exhibition *Expo 2005* in Japan, a five screen setup with diffused rear illumination and rear projection was built. Behind each screen is a projector, an infrared light source and an infrared camera. Ambient light is a big problem with such a setup. It is not easy to find a usable screen material and even with fully controlled lighting it is not easy to reliably track hands. With simple thresholding, bright cloths of the user also incorrectly trigger a detection. In combination with a local contrast filter, whole hands can be tracked reliable, whereas single fingers from the same hand cannot be distinguished on that setup. An issue that was not thought of in the planing phase is the hygienic concerns in the Japanese society. As an alternative to using the bare hand, paper hands on sticks were offered at the venue. See section 6.2.1.1 for a description of the software part.

2.3.2.2 NextWindow Frame

In collaboration with *NextWindow* we modified their single touch frame to get a larger scale version. The frame consists of stripes with infrared LEDs at 3 sides and two line scan cameras in the corners. A big advantage over all other optical touch technologies is the independence to ambient light. Even direct sunlight on the screen seems not to be a problem. No touch force is required for a detection. A disadvantage is that sleeves or almost touching parts of the hand also trigger a detection.

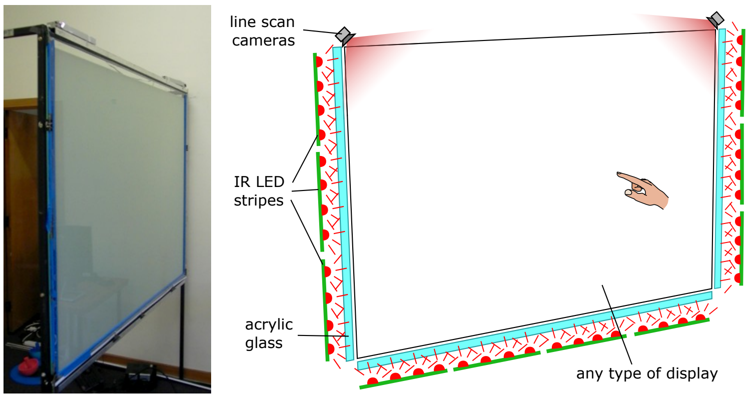


Figure 2.43: The NextWindow setup (in 2005) that we scaled and fitted to a larger rear projection screen. Two line scan cameras see a finger's shadow and can triangulate it reliably.

Figure 2.44: Transportability of VR and multi touch setups is an important issue in practice, as they are well suited for exhibitions and presentations. This image shows the rather difficult transport of the multi touch table constructed by the Fraunhofer IGD.

In the design above, multiple touches cannot be localized as ambiguities cannot be resolved. The *dreaMTouch* by *Citron* mostly handles this by employing many more (we estimate a dozen) line scan cameras. When fingers get close to each other, the system fails to distinguish individual fingers and only detects one large contact. Together with the furniture company *bene Büromöbel* it was integrated in an existing table with an additional screen in a side wall (see Figure 6.10 on the right).

2.3.2.3 HEyeWall Multi Touch Tracking

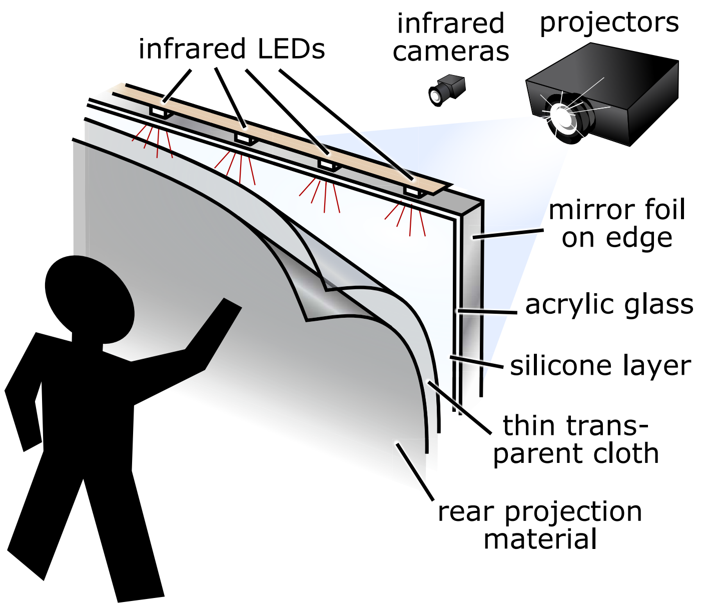


Figure 2.45: Multi touch setup for the HEyeWall with the FTIR technology. The silicone layer as compliant surface on the acrylic glass is necessary to make the touches more visible. A thin cloth is used to prevent the rear projection material from sticking to the silicone layer.

In the diffused illumination setup described above, there are difficulties getting reliable touch information for single fingers. Thus, the promising FTIR (frustrated total internal reflection) technology (see Figure 2.16) was tested. With positive experience on a first small test screen, it was scaled up for the HEyeWall.

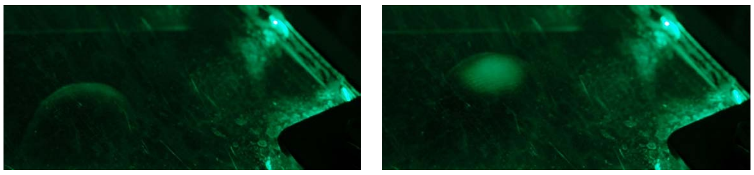


Figure 2.46: Frustrated Total Internal Reflection (FTIR). Light enters the acrylic glass from the side. Here, a small fraction is diffused on the surface and visible in the top right of the photos. The major amount is refracted and reflected internally. When a finger is close to the glass, it is not lit (left image), but when it touches, the refraction index changes and the light diffusely reflects from the finger (right image).

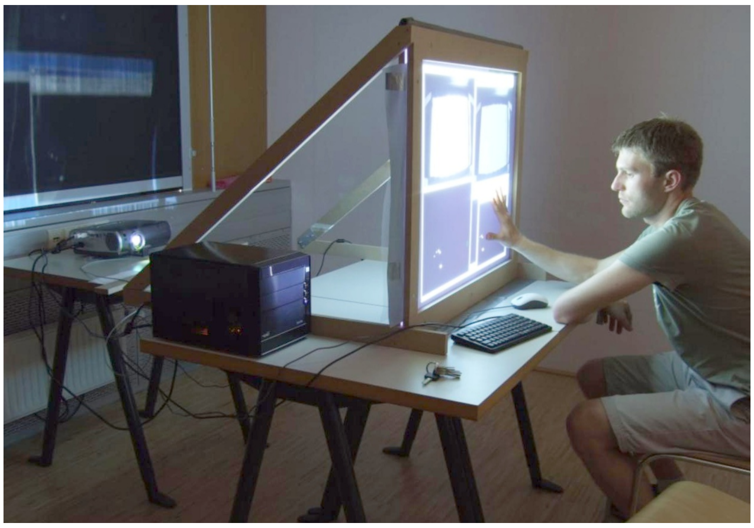


Figure 2.47: An FTIR test setup for the HEyeWall multi touch input. An infrared camera is placed next to the projector to capture the screen. With infrared illumination from the side, the screen lights up when touched.

Scaling the Technology to the Large HEyeWall. We tried to extrapolate the amount of necessary light for the large screen. As it turns out, we had largely underestimated the number of necessary LEDs. By using more LEDs, polishing the edges and fitting them with reflective mirror foil everywhere except for the LED positions, we managed to increase the amount of light by a factor of about six.
Another problem is the silicone coating. While the coating of the small test screen worked well, the high viscosity and quick drying of the silicone lead to serious problems. Our first attempt failed and the next one still lead to surface defects in form of enclosed air bubbles and an uneven surface.

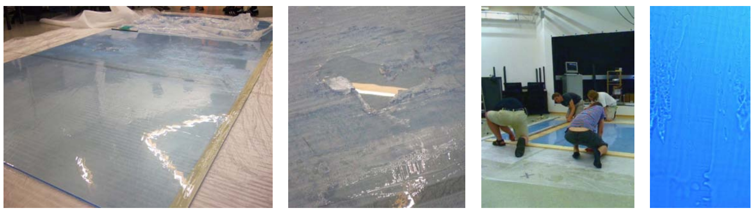


Figure 2.48: **Silicone coating the large screen did not work well. The left two images show our first attempt that failed. Even using silicone thinner, our second attempt was not perfect, resulting in air bubbles and sometimes unregular spots as seen in the close up on the right.**

Also, a small fraction of the projector light is visible to the cameras. The direct reflections on the acrylic glass lead to blind spots without further precautions.

2.3.3 Illumination with Infrared LEDs.

For both tracking systems for the DAVE and the HEyeWall, infrared LEDs are employed.

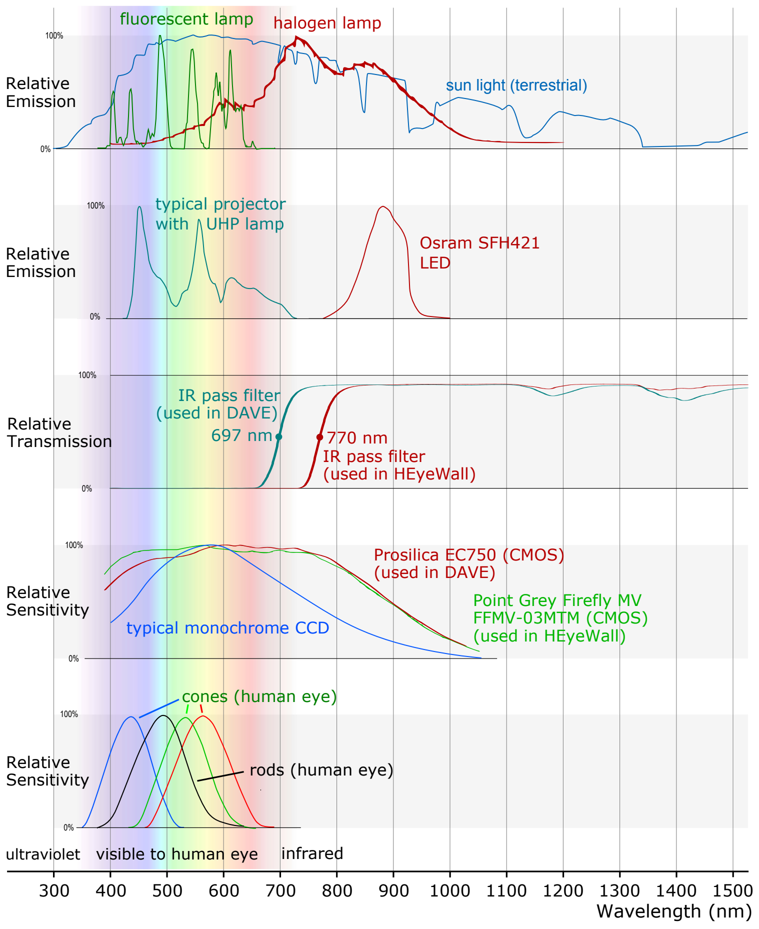


Figure 2.49: Spectral curves: interfering light sources (reproduced from [DQ],[Spea],[Speb]), projector (reproduced from [Sped]) and infrared illumination (reproduced from [Spec]), IR pass filter (self measured), used CMOS cameras (reproduced from [Gre],[Pro]) and a typical CCD camera (reproduced from [Opt]) for reference, human eyes (reproduced from [Dow87]).

Spectral Curves and Infrared Light. In order to avoid blinding of the users or contrast reduction, the tracking works with wavelengths outside the visible spectrum. As common camera sensors are sensitive to the near infrared spectrum and LEDs with these wavelengths are particularly cheap, this is by far the most cost effective solution. Unfortunately, other light sources may interfere. If ambient light is a problem, one solution is to use laser illumination and a matching narrow bandwidth filter that blocks a large amount of the ambient light. A second solution is to use bright short flashes synchronized with the exposure. However, the pulsed light from DLP projectors may lead to flickering. Of course, both methods can be combined. As we use a rather long exposure of about 4ms in the DAVE, motion blur occurs at fast moving markers. Also, the projector light in the DAVE is slightly visible to the camera. Both problems can be minimized by increasing the amount of IR illumination. Besides adding more LEDs or using powerful devices, pulsed LEDs

synchronized to the camera exposure can drive higher currents for a short time. As an example, in our configuration, LED pulses with only 0.1ms allow a ten times higher maximum current. Motion blur and influence by other light sources are reduced by the factor of 40, but the light also gets about five times dimmer. Also, there is a risk to influence the pulsed infrared signal used by the shutter glasses. Finally, the driver circuit is more complex. For these reasons, we use a constant illumination.
For the HEyeWall multi touch setup a constant illumination with about 30ms exposure time is used. With multiple pulsed flashes with a duration of 0.2ms each and ten times the current, and with a synchronized multi exposure, the illumination intensity could stay the same, but the influence of ambient light from room illumination, daylight and projector light could be reduced by a considerable factor of 150. This solution requires camera hardware supporting such multi exposures and a much more complex LED driver. A small error may quickly lead to thermal destruction of the LEDs.
An uneven illumination may lead to a dynamic range higher than the camera can record with a single image. Multiple images with different exposures could be recorded but this requires additional overhead and processing time. Instead, the illumination should be evenly distributed over the image.

Eye Safety. Using infrared illumination for tracking is an elegant way. Adding more LEDs for the DAVE is relatively simple. In contrast, for the HEyeWall, much more work is required. However, especially in the DAVE little ambient light is present and the pupils of the users are big. As the LED light is barely visible to the human eye, its protective reflex does not work. A high dosage over a long time may lead to opacity of the lens over the years.
Little information is available to accurately compute safety limits. An estimation shows that at normal distances, the radiation level is not harmful. However, one should not look towards the LEDs from a close distance. Also, other 3D optical tracking systems use IR illumination with much higher intensities.
Of course, focussed IR lasers should be treated with special care. For our multi touch systems this is an important point against IR laser light plane setups.

2.3.4 Hand Held Interfaces

We developed several hand held interfaces, both for the immersive DAVE and for non-immersive interaction such as at a desktop or in an exhibition or museum. Available interfaces are not ideal and often not well suited for VR. Experimenting with new interfaces requires basic IO capabilities, presented below.

2.3.4.1 DAVE Interfaces

Several hardware interfaces were tested for navigation and interaction in the DAVE. For localization and orientation, the devices use the same 3D tracking that is also used for head tracking functionality.

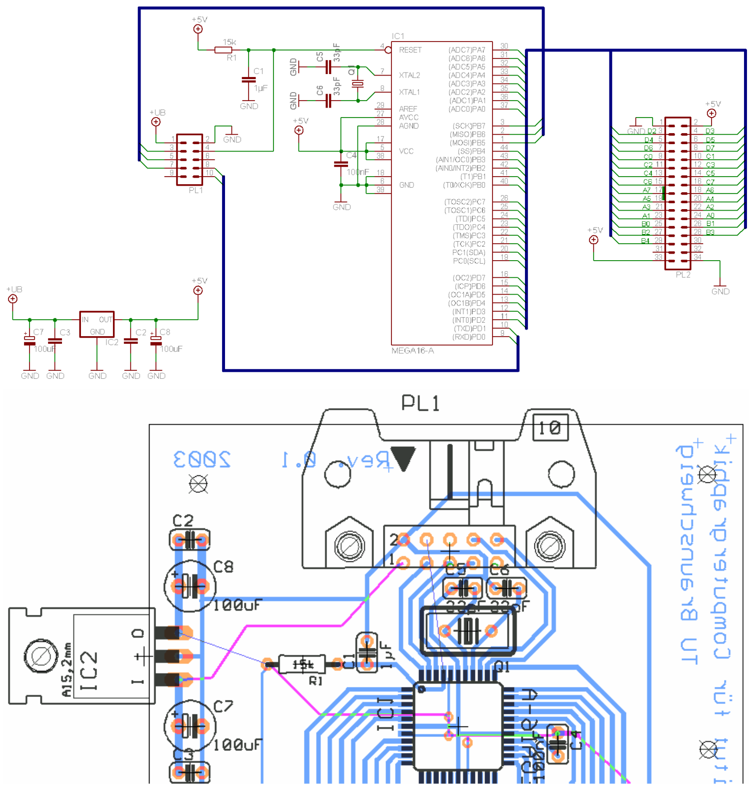


Figure 2.50: Schematics and layout of an electronic interface to connect different input devices for the DAVE in Braunschweig. The Atmel Mega16 microcontroller is programmed to communicate with a PC, offering digital and analogue inputs and outpus.

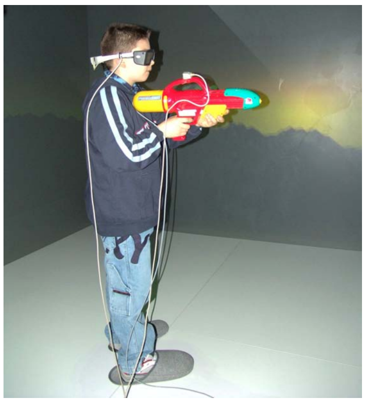


Figure 2.51: A user with the water pistol IO device in the DAVE in Braunschweig.

For the the electromagnetic tracking of the DAVE in Braunschweig, sensors are expensive and only one was available to us for custom devices. To quickly be able to swap one custom device with another, a general purpose input and output interface is used. The interface cable and sensor can easily be detached from one device and attached to the next one. 27 IO pins are available for free use, providing digital inputs and outputs, a maximum of eight analog inputs and one analog pwm (pulse width modulated) output. A couple of digital inputs are used for automatic device identification, also allowing hot plugging. In contrast, for devices using the optical tracking in Graz, each device gets an own target attached and in theory, all devices can be used at a time. In practice, the system only checks for one to two devices at a time, as the performance of the tracking system drops with more targets and most of the applications use the same input device.

Gamepad. In the DAVEs in Braunschweig and Graz, gamepads are used, offering many inputs (16 digital buttons, 4 analog axes) in a robust and ergonomic housing. Navigation with 6 independent degrees of freedom is possible, still allowing to trigger additional events with the remaining channels. However, most of the functionality is not used in our applications. The high number of controls presents a challenge to novice users, as they can accidentally mix up the buttons, or trigger a reaction by accident. For interaction with a gamepad, usually both hands are required. In our experience, pointing with stretched arms while holding the gamepad feels awkward.

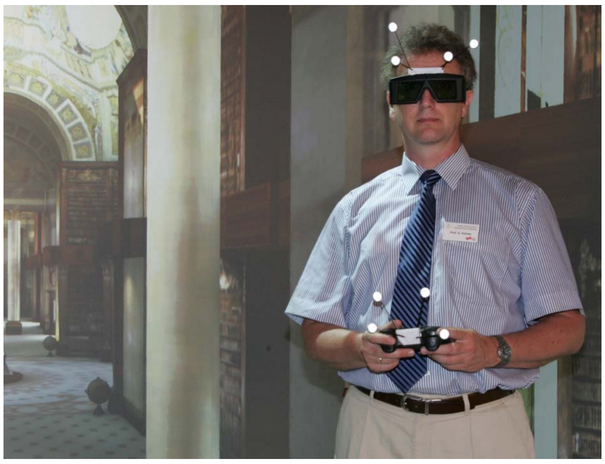


Figure 2.52: Tracked gamepad and 3D glasses as input devices in the DAVE in Graz.

Joystick. To free one hand and enable comfortable pointing, and to avoid the complexity of a gamepad, a custom joystick was built. Similar commercial alternatives are very expensive (e.g. the FlyStick2 by A.R.T. costs 3,100 EUR, mid 2011) and do not offer an analogue trigger, like our device. This trigger state together with the 6 DoF tracking information is usually sufficient for our applications. Four additional digital inputs are placed on top of the device to offer further input options when necessary. For the electronics, a cheap wireless gamepad (Logitech Rumblepad 2, 22 EUR, mid 2011) was disassembled and unnecessary parts of its board cut away to fit in a small box under the grip. This box also contains the batteries, giving the device a good balance. A physical arrow is glued to the front to indicate the principal direction. This allows a quick explanation to novice users, e.g. controlling movement by the arrow pointing direction. The main drawback in daily use is that a custom device is hard to replace in case it breaks.

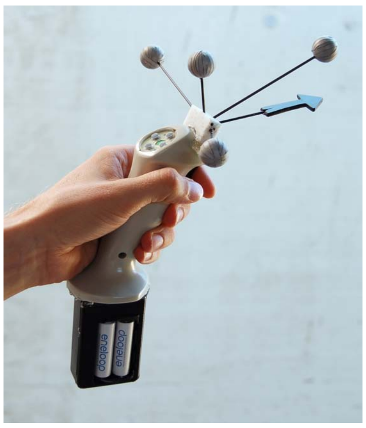


Figure 2.53: A custom joystic, a versatile input device used in most DAVE applications.

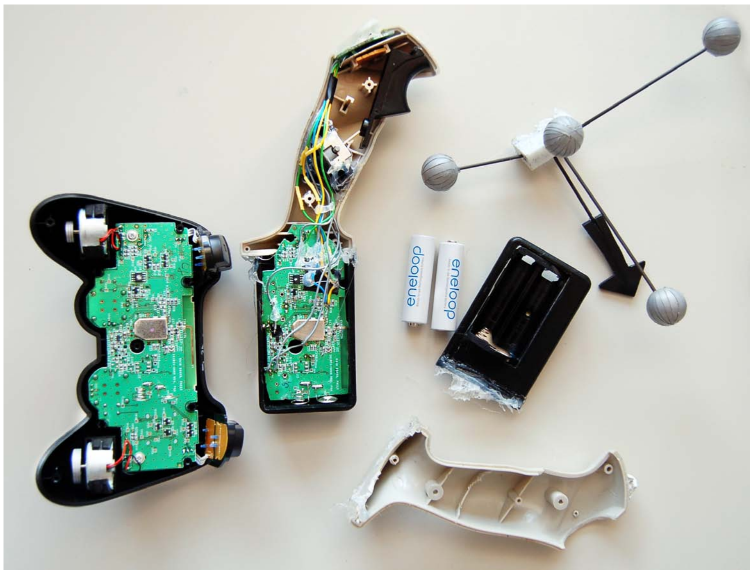


Figure 2.54: The electronics of a gamepad (left) are cut and integrated into a joystick grip (center), with a battery compartment on the bottom.

2.3.4.2 Wireless sensors and buttons

For custom user interface prototypes, especially for the DAVE, wireless handheld devices are useful. While the first versions were still using a microcontroller connected via the serial port, it also turned out to be very cost effective to use the electronics of an existing joystick, featuring a USB wireless communication that is very easy to use via the available driver. Joysticks are often supported by software already. On the other hand, a custom USB device is expensive and a lot of effort is required to implement the firmware and the drivers on both of our supported operating systems.
Another possibility is trying to use wireless presenter devices or gyration mouses, that have accelerometers and now even gyroscopes build in. Unfortunately, there is no software API available as they are intended to just control a 2D mouse cursor. Maybe, the use of such devices is possible by sniffing the USB data and reverse engineer the communication protocol to access the raw data. Hackers have achieved this for the *Wiimote* by *Nintendo*, as explained in the next section.

2.3.5 Accelerometer

In a museum exhibition, small or unaccessible items may be hard to see from all sides. One solution may be to show a 3D rendering of a 3D scan on a large screen next to the real physical item. To inspect the item from all sides, visitors should be able to rotate them in an intuitive way. One idea to solve this is a proxy object that may have a similar shape as the item. By picking the proxy item up and by rotating and moving it around, the 3D representation on the screen can be controlled.
For the *EPOCH* [EPO] project we built such a prototype. A first attempt was to use computer vision to detect features on an object and derive its six degrees of freedom. At the time, algorithms for efficient natural feature detection were still in development by experts in the field. After a few months of work it showed that too much time was still necessary to finish the implementation. However, most of our goals could be achieved using a new three axis accelerometer integrated in a single chip. To interface that chip with a PC, we use a microcontroller. While joystick hardware usually supports a dozen of binary buttons plus a few analog potentiometer levels, they may not be useful to directly measure analog voltages, as often provided by sensors. Using a microcontroller, the analog voltage can be converted and sent to the PC via the serial port. For moderate motion, most of the acceleration consists of gravity, thus the vector pointing down can be measured and used to control tilt and rotation of a 3D artifact on the computer screen. Later, this chip was also used in the Wiimote, making it more available as it is integrated in a mass product for a low price. The Wiimote is wireless but also larger than our small device.

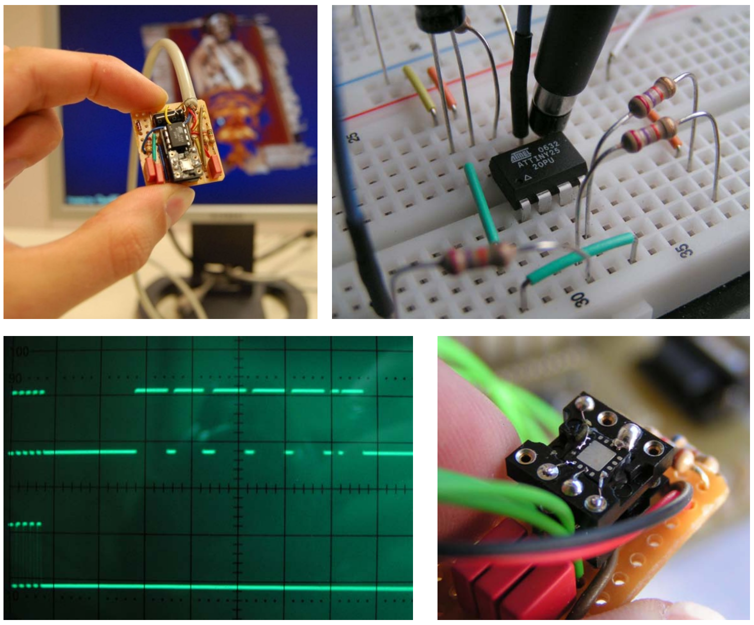


Figure 2.55: Acceleration sensor and micro controller board as PC interface to control 3D object rotation on the screen. The microcontroller is shown in the top right photo, the accelerometer is visible in the center of the bottom right image. The serial communication with the PC needed debugging with an oscilloscope (bottom left). The complete device is shown in the top left image.

2.3.6 Turn Table for Automated 3D Scanning

Scanning 3D objects like items in museum collections becomes more and more important. However, such scanning often requires manual work. Only with fully automated 3D scanning it is feasible to digitize large collections of museum items. In an early work to address this problem, we built a low cost silhouette carving based 3D scanner. The hardware is a self made turn table which is motorized by a stepper motor that is controlled by the PC. A backdrop, lighting and a webcam connected to the PC complete the setup. Refer to section 3.3.3 for more details.

2.3.7 Camera and Projector Mounting

Some mechanical details sound trivial from a research point of view, but still need to be done. One example is mounting of projectors and cameras. It is advisable to be able to control the degrees of freedom independently. While for a camera the position is usually fixed, two or three angles have to be adjusted. In addition, the lens controls must be reachable to be able to adjust zoom, focus and aperture.

Camera Mounts. The camera mounts for our 3D optical tracking in the DAVE are commercial mounts for loud speakers. They are very unpractical: All rotational axes are fixed with a single screw while one axis is a lot looser than the others, making small adjustments very tedious and hard to control. To solve this, we designed a custom simplistic camera mounting. Independent tightening of each degree of freedom is easy. However, since the *Firefly MV* cameras by *Point Grey* did not come in a housing, precise work is necessary. As now a camera housing is available that is equipped with a standard tripod mount, very cheap and good mounts can be bought.

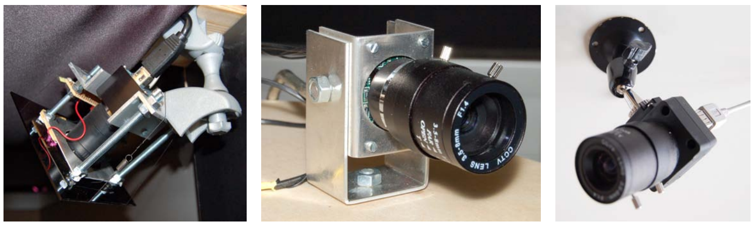


Figure 2.56: The camera mounts in the DAVE are hard to adjust (left). Custom camera mounts for the HEyeWall work much better (center). A cheap and good option is shown on the right.

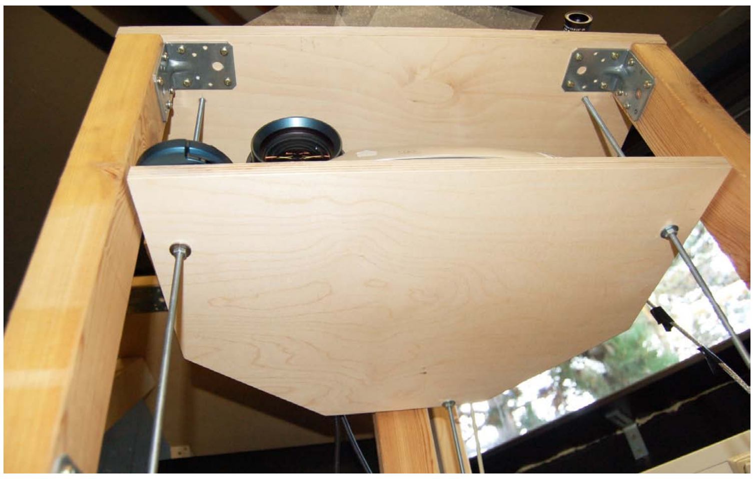


Figure 2.57: The custom stands allow an easy and precise adjustment of the projectors with three screws for each platform.

Projector Stands. A precise mechanical alignment of projector images to a screen is hard and often impractical. With additional software calibration, only a rough alignment is necessary, as long as the image covers the whole part of the screen. However, the better the mechanical alignment, the less light and resolution is wasted. Rotation angles as well as translation should be independently controllable. With the DAVE projector mounting, adjusting the projectors is tedious.
We designed a better solution for the small projectors of the HEyeWall. Each platform for a projector is held by three screws that can be directly adjusted with a high precision. Projector weight and friction are enough to keep the screws in place.
An idea for a helpful application for positioning a projector is a guidance system. A calibrated camera may continuously analyze a projected pattern to be able to advise the operator in which direction the projector should be moved.

2.3.8 Outlook: The Ideal Hardware

For all presented devices there is much room for improvements. For a technology to stimulate all human senses and react to the user's intentions, science fiction stories have an answer for a long time: the brain interface. A major advantage of this idea is, that if this one technology really worked one day, all other hardware would become obsolete.
Sutherland's thoughts for the *Ultimate Display* [Sut65] were already mentioned. Three years later, the science fiction author Bob Shaw

described a different but compelling idea of a *scenedow*, a combination of scene and window with *slow glass* in his novel *'The Light of Other Days'* [Sha68]:

> Light took a long time to pass through a sheet of slow glass. [. . .] One could stand the glass beside, say, a woodland lake until the scene emerged, perhaps a year later. If the glass was then removed and installed in a dismal city flat, the flat would - for that year - appear to overlook the woodland lake.

3

Input: Optical Tracking

A lot of input devices mostly consist of digital sensors and hardware for data transmission. This raw data can often more or less directly be used in the interaction logic. However, optical input devices using cameras typically record sensor data that cannot directly be used. Clever software is needed to interpret the data in a number of ways. This is also the reason for their versatility. A whole research area is dedicated to this task, the field of *Computer Vision*. It is particularly hard for a computer to "see" and understand its environment. Many tasks that are simple for a human are unsolved problems in Computer Vision.
Systems for user and object tracking are an essential part of Virtual Environments. Many of the applications can be realized using optical tracking systems. Two types of optical tracking systems were built from scratch in order to affordably realize and improve the systems and experiment with new ideas: 3D tracking based on reflective markers and 2D multi touch systems for large rear projection screens. The software part of these systems is the content of this chapter.

3.1 3D MARKER TRACKING

At the start of the project, we had three major goals: We wanted to fill the gap of not having a good tracking system for the DAVE, build a cost effective system in line with the idea of an affordable DAVE and intended to create a benefit for the community.

3.1.1 Optical 3D Marker Tracking - Related Work

Tracking systems based on other hardware technology are listed in the previous chapter. Here, only work is presented that is closely related to our approach.
State of the art commercial tracking systems provide very precise and fast measurements. Vicon Motion Systems offers cameras with up to 16 Megapixels and 120 Hz capturing and on board processing, or lower resolution images with up to 2000 frames per second. *Advanced Realtime Tracking GmbH (A.R.T.)*, *OptiTrack* by *NaturalPoint* and the *iotracker* offer systems with less powerful hardware specifications. The software tools for all of these systems are developed for a convenient calibration and use. However, the high price is a problem for many installations. In mid 2011, the approximate costs for systems suited for tracking in the DAVE are 50,000 EUR (Vicon), 20,000 EUR (A.R.T.), 10,000 EUR (iotracker) and 5,000 EUR (OptiTrack, only max. 58° hfov). The OptiTrack bare camera boards without housing, lens or flash but with imaging sensor and onboard 2D point processing are available for only about 250 EUR, making this a very attractive choice for own developments.
Organic Motion offers a markerless motion capture system. Using convex hull carving, it can fit and show 3D skeleton data in real-time. The *Kinect* sensor by *Microsoft* computes a depth image by triangulation with a projected IR pattern. Time of flight cameras also provide a depth image by measuring the time of travel of a very short IR flash. These methods can also provide the user pose as skeleton information. However, the head rotation cannot be obtained yet. This is important for VR in order to estimate the eye positions especially for the stereo effect.
A cheap active optical tracking with a single camera is *TrackIR* by *NaturalPoint*, intended for desktop gaming. Fishtank VR and exaggerated head rotation are used in games e.g. to look around in a cockpit. An open source alternative using the same software interface is *FreeTrack*. Hay et al. describe low cost tracking using *Nintendo Wiimote* cameras [HNH08]. As the devices can send pixel positions of up to four bright spots in the infrared camera image, four active markers can be tracked with relatively little effort. A similar system with passive markers is presented in [ASLP10]. A drawback is the relatively small field of view (42° hfov, 32° vfov) of the camera, making it less useful for the DAVE. A reported problem in practice is the Bluetooth pairing that may not always work reliably.
The ARToolkit [KB99] is a 6 DoF marker tracking for a single camera. A square containing an identifying pattern is used as marker. This is also possible using natural features, as described by Wagner et al. in [WRM*08]. Such tracking is well suited for AR. For outside-in

tracking in a CAVE, the required high camera resolution or large marker size is not practical.
For marker identification, active markers can be switched on and off with a time code pattern, similar to the technique employed by the HiBall tracking system [WBV*99]. For such a method, the LEDs need a electronic controller and some sort of synchronization. Another approach is to use different wavelengths, i.e. different colored LEDs with an RGB camera. This may be a good option for a limited number of markers when using visible light. However, the Bayer pattern leads to a reduced resolution of each color channel compared to a monochrome image with the same imaging sensor.
Research at the Vienna University of Technology with passive marker tracking [DU02] led to the already mentioned iotracker system that is now commercially available. Mehling et al. [MEK06] and Pintaric et al. [PK07] describe the system. End of 2005, when the DAVE was built, we saw a working prototype. Unfortunately, the system was not yet available. Results are superior compared to our system in terms of accuracy and speed, also allowing more targets at a time.
Hervé et al. show with the Cyclope tracker [Mat05], that even with a single camera, the 6 DoF of a target can be recovered, e.g. by using the POSIT algorithm presented by DeMenthon et al. [DD95] which computes a first solution without perspective as starting point for a subsequent iterative refinement. The cyclope tracker was also developed into a commercial product which was released end of 2006. The application of this idea to our tracking system is possible with the same hardware and will lead to a higher robustness against occlusions.
Hogue et al. present a hybrid tracking system for a six sided CAVE using an inertial sensor in combination with a set of laser rays pointing in different directions to the display screens [HJA04]. Cameras outside the CAVE track these points.

3.1.2 Tracking System Components

In order to fulfill the goals stated above, optical tracking is the best option we found. We chose the method of tracking targets consisting of spherical markers because it was already demonstrated that this is a robust, accurate method and there are no obvious unsolved problems. Our further design decisions were to use passive (reflecting) markers instead of active ones, the choice of camera hardware considering cost and benefit and also the decision to do all image processing on a single PC rather than on dedicated DSP hardware. Finally, for easy distribution and high performance, it should not depend on external mathematical tools like *Matlab* by *MathWorks*.
The software is split up into modules with defined input and output data, as illustrated on the following page. These modules can easily be exchanged for future improvements and tests or recording and playback of data. A configuration file allows to choose and configure the modules.

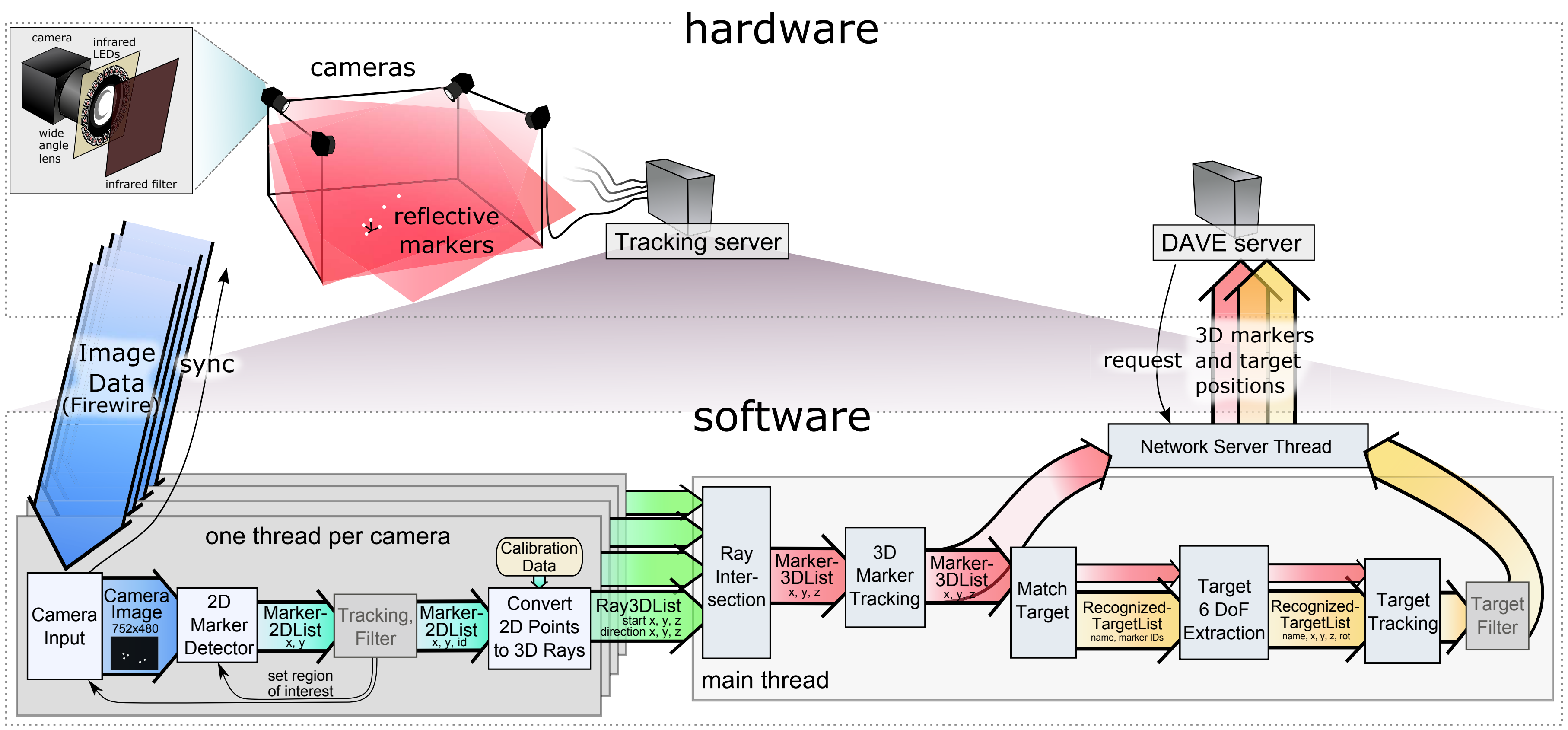


Figure 3.1: Marker based optical 3D tracking system overview.

For performance testing, all modules can be run a thousand times in a row each to be able to more accurately measure the run time. The measurements are still not exact due to the influence of the cache, but are helpful to judge runtime differences of different approaches or implementations in relation to each other. Also, each module has a simulation mode to send out fake, simulated data to test subsequent modules. Some modules can be disabled, e.g. target detection. Most modules can record or playback incoming or outgoing data, useful for debugging or development.

3.1.2.1 Image Capture and Transfer

All cameras are triggered by software. When an image arrives, a time stamp is added. This module directly communicates with the camera API and is the only operating system dependent part of the whole system.
For one type of cameras that we use, the *EC750* by *Prosilica* cameras, we can only use synchronization by software at the moment. For hardware synchronization, the camera API does not fit well to our multi threaded system model. It would be advisable to split up the camera capture module into two parts, moving the synchronization part to the main thread.
We also developed a module for the more affordable *Firefly MV* cameras by *Point Grey*, when they became available.
To avoid memory allocation for each frame, a double buffer is used. While one image is processed, the data transfer for the next image is already started.

3.1.2.2 Marker Detection

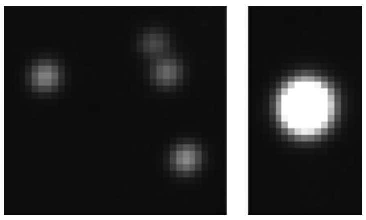


Figure 3.2: Examplary close-ups of markers in a camera image.

A simple approach in a computer vision pipeline is to first use a brightness and contrast enhancement of the image, followed by a thresholding pass and then a connected components labeling step. As only small circular bright spots are searched, the above steps can be reduced and simplified to a single pass in order to improve the performance.
A pointer to the raw image data and the image dimensions are the inputs of this module. An additionally specified region of interest (ROI) may further reduce the image area where markers are searched. The image is processed line by line in a single pass and only uses data of the current and last row at a time, so that the data fits easily in a fast processor cache.
In the camera image, light from the DAVE projectors is slightly visible. A sparsely sampled 1D Laplace impulse response on the rows is calculated and compared to a threshold. It is invariant to background illumination with low spatial frequencies and is very fast to compute. If the value is above a certain threshold, is is accepted as part of a detected marker. In that case, the left and above pixels are checked in order to use the same ID when the marker was already detected before. The weight (instead of number of pixels) and weighted position is summed up to a total for each ID.
After the whole region of interest is processed, all weighted positions are divided by the accumulated weight to gain an average position with sub pixel accuracy.

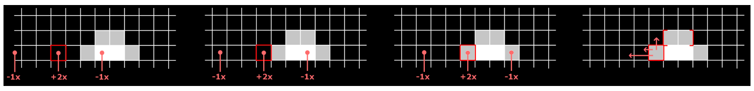


Figure 3.3: A fast marker detection algorithm, processing a row of the image from left to right. In the two left images, not marker is found. The third image shows a detection, as the vale is above a configured threshold. The right image shows which pixels are searched to extend a label of a potentially already detected marker.

Robustness. Infrared light from other sources can lead to false positive detections. As an example, when the door of the DAVE is open, some ambient light enters the room. Also, a halogen ceiling light may be switched on and leads to partially very bright spots. If during processing, too many markers are counted, the processing of the image is aborted. This is done to prevent the whole system from blocking while waiting for the thread.

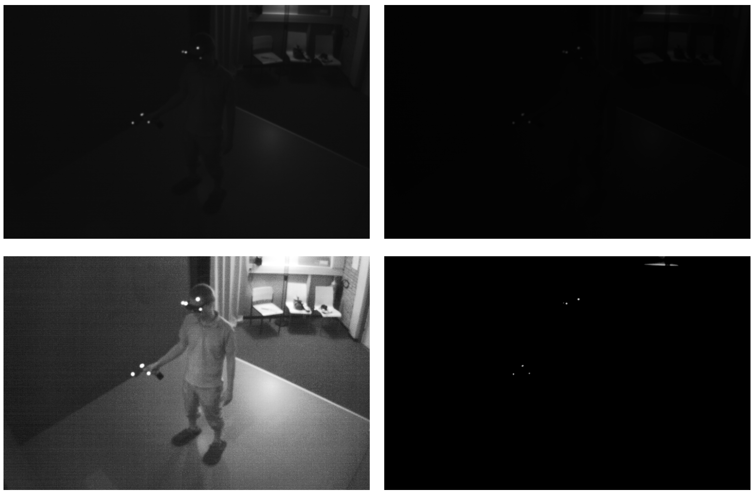


Figure 3.4: An exemplary camera image (top left, enhanced levels in the bottom left) with some ambient light present. The result of filtering and thresholding is shown in the top right (enhanced levels in the bottom right). One marker of the glasses is not recognized. The ambient light leads to five additional detections which does not break or significantly slow down the tracking of the glasses and the joystick.

The simplified pseudo code example on the next page illustrates the marker detection. Failure cases are too dark or partially occluded markers, resulting in a wrong position. As markers touching the image border are also likely to return the wrong position, they are discarded.

Subsequent Position Refinement. To find even more precise center positions of markers, a subsequent and more complex algorithm was tested to refine the positions. As the marker positions are already known, the overhead is not high for the few markers. In that approach, the marker center is computed multiple times for different intensity thresholds and the average position of accepted values is calculated. However, in practice no noticeable improvements result and this approach is not used.

```
const int hDist = 3; // horizontal distance of samples
const int threshold = 30; // absolute threshold for detection
for (each line/row y in the region of interest) {
  for (each column x in the region of interest minus a hDist pixel wide border) {
    float value = 2*img(x, y) − img(x − hDist, y) − img(x + hDist);
    if (value > threshold) {
      if (does left pixel belong to a known−marker?) {
        m = known−marker; // yes, use that marker
      } else {
        // its a new marker in this line
        if (does it touch a known−marker in the previous line?) {
          m = known−marker; // yes, use that marker
        } else {
          // add new marker
          MarkerStruct m = {0, 0, 0, false};
          tMarkers.push_back(m);
          if (maximum number of blobs exceeded) return;
        }
      }

      weight = value − 0.95*threshold;
      m.posSumX += x * weight;
      m.posSumY += y * weight;
      m.weightSum += weight;
      if (marker touches image border) m.touchesBorder = true;
    }
  }
}

for (each marker m) {
  m.posSumX /= m.weightSum;
  m.posSumY /= m.weightSum;
  if (m.touchesBorder) m.erase(); // remove from list
}
```

Figure 3.5: Simplified code for marker detection. The code is developed for fast processing and works reliably in the common case. Markers that overlap, are partially occluded or are too close or too far away from the camera will lead to wrong positions.

Possible Improvements. One possible solution to address the overlapping marker problem is to use circular Hough transform, as mentioned in [PK07].
When all markers have the same physical diameter, the pixel area in the camera image gives a rough estimation of the distance from the camera. This additional information may help to reduce ambiguities in the subsequent ray intersection, but is currently not used.
Instead of only detecting points, other shapes could be used. As an example, reflective stripes taped to the glasses are probably sufficient for 6 DoF tracking in the DAVE. However, they will be occluded more often and require a different, more complex code for marker detection and subsequent modules.

3.1.2.3 2D Marker Tracking

For the initial idea of using asynchronous camera images (see section 3.1.5), tracking is essential for extrapolating the 2D marker position at a common time. In the current scenario, it is less important. While in some cases it may help to disambiguate marker ray intersection over time, the following 3D marker tracking module is sufficient in general.
With a simple assumption of linear 2D motion, a region of interest for the currently visible markers can be predicted. This region should

be enlarged to be on the safe side, to also detect the markers when the prediction is wrong. Currently in the DAVE, the region of interest is always set to the complete image, as the marker detection is quick enough.

3.1.2.4 3D Ray Generation

The 2D marker positions are converted into 3D rays in world coordinates by using the camera model with the respective calibration data (see section 3.1.6.2). The ray is defined by the camera nodal point and the direction vector pointing into the marker's direction.

3.1.2.5 Triangulation by Ray Intersection

A simple method for triangulation was implemented. Hartley et al. present several alternative methods in [HS95] that can improve the accuracy, especially for rays that are almost parallel, e.g. by minimizing the sum of the squared reprojection errors.

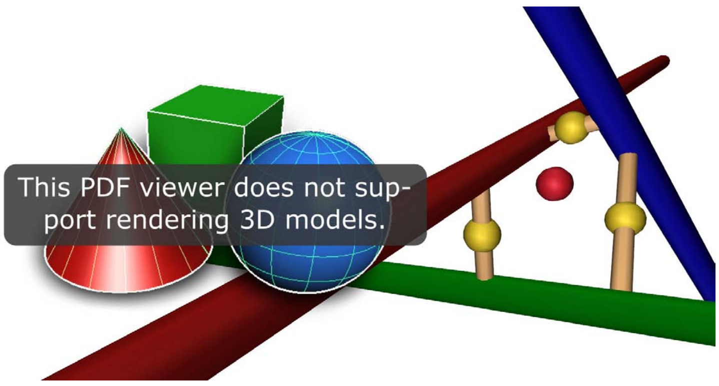


Figure 3.6: A closeup view of three skew rays. The shortest connection between two skew lines is shown as thin yellow cylinder. Its center is the assumed real intersection position (yellow sphere) of these two rays. For more than two rays, the average position of all yellow spheres is used (red sphere).

After all 3D rays are collected from each thread and copied to a new list to be used by the main thread, the camera threads can start processing their next images. Due to a number of errors adding up, like camera noise and imprecise calibration, the lines will in general be skew lines that do not intersect in the exact mathematical sense. The ray to ray distance is compared to a threshold in order to decide whether the rays are considered to intersect. The assumed position of the real intersection is computed as the center of the shortest connection of the two lines. For practical reasons, the assumed intersection positions are simply called intersections in this document. For more than two lines, the average of all pairwise intersection points is used. With these definitions the intersections of all rays in question are computed and 3D markers are generated for the respective positions. While the algorithm can deal with all combinations of a variable number of cameras, it does not scale very well.
The following example demonstrates how the rays for the intersection computation are selected.

```
Camera 1: ray 1.1, ray 1.2
Camera 2: ray 2.1
Camera 3: ray 3.1, ray 3.2
Camera 4: ray 4.1

Step 1: find pairs of rays (of different cameras) that intersect (note:
     for the intersection test of two rays the distance of the two rays
     is used: if they are closer than e.g. 1cm they are considered to
     intersect with each other):

  pairs:   (1.1  3.2) (1.2  2.1) (1.2  3.1) (1.2  4.1) (2.1  3.1)
           (2.1  4.1) (3.1  4.1)
  triples:
  quads:

Loop (1st pass):
 Find groups to merge.
 Remove groups of previous level that were merged.
Result:
 pairs:   (1.1  3.2)
 triples: (1.2  2.1  3.1) (1.2  2.1  4.1) (1.2  3.1  4.1) (2.1  3.1  4.1)
 quads:

Loop (2nd pass):
 Find groups to merge.
 Remove groups of previous level that were merged.
Result:
 pairs:   (1.1  3.2)
 triples:
 quads:   (1.2  2.1  3.1  4.1)

Loop (3rd pass):
 Find groups to merge.
  None found -> exit loop.
```

Figure 3.7: Example demonstrating the algorithm to find each possible combination of rays from different cameras for triangulation. Groups of intersecting rays are enclosed in brackets (). The resulting two groups represent one 3D marker each. Its position is computed via triangulation.

Once the groups of rays are found, they are triangulated as illustrated above.
All possible ray intersections are possible positions of 3D markers. We choose to allow multiple intersections for a ray, because multiple markers in the same line of sight would indeed only generate one ray. We found that getting more false markers is better for most applications than missing a correct one. It can happen, that multiple markers are generated instead of a single one, e.g. when for three rays, two pairs of rays intersect and the remaining pair does not. For that reason, markers that are very close are merged to a single marker at their average position.
When the same marker is detected in many camera images, as many rays are generated. While more rays per intersection should improve the accuracy, it is also slower to compute. Also, for a more general setup of cameras, it will be useful to give a higher weight to intersections with a lower estimated error, i.e. perpendicular rays.

3.1.2.6 3D Marker Tracking

Using information of the 3D markers of the previous time steps, we try to match the markers of the previous frame with the newly generated markers of the current frame, passing on the ID. For a very simple prediction of the new position, the marker speed is used, assuming linear motion with friction. A problem occurs if more than one new marker is within the threshold distance. Our design decision is to use unique IDs, so only one marker gets the ID, the others a new one. In that case the ID of a marker may change.

Filtering. Noise perturbs the marker position results. For some applications that directly use a marker position, e.g. to draw a line in space, filtering may be advantageous for smoother results. A well suited method is a Kalman filter [WB95]. As a filter also introduces latency depending on the amount of filtering, it is advisable to let each individual application configure the trade-off between smoothness and latency. Currently, we do not use such a filter e.g. to reduce noise or smooth a path of a marker.
The information on all individual 3D markers is complete and is copied to the network thread, providing the data with the lowest latency as possible.

3.1.2.7 Target Recognition

The pairwise distances of all markers are computed in a first step. All six distances of each combination of four markers are then compared to the pairwise distances of each of the learned targets. Possible matches within a threshold are chosen, giving a number of possible targets with the respective marker IDs for each of its vertices. Thus, for a new recognition of a target, all its markers must be visible in our implementation.
This step scales linearly with the number of known targets, but computation time grows a lot quicker with the number of detected markers with $O(n^2)$. One idea to speed up the computation was to cluster close markers and only search within a cluster. In an early attempt we used a modified version of Kruskal's minimum spanning tree algorithm, were long edges above a threshold are not added. However, using that algorithm does not make any sense because still all pairwise distances need to be computed. Instead, the clustering should have a lower complexity. Some sort of local hashing or bucket sort with overlapping buckets may be a solution but was not yet implemented. Pintaric et al. describe an additional quick rejection of wrong target constellations to further improve the performance [PK07].

3.1.2.8 Target 6 DoF Computation

To compute the best fitting 6 DoF pose of the learned target to the measured markers, the Kabsch algorithm [Kab76], [Kab78] is used. It is designed to find the root mean square deviation (RMSD) for applications in molecular biology. The translation is computed using the center of mass of both point sets. Subsequently, the rotation matrix that minimizes the mean square distance is computed. Shinji [Ume91] presents a similar solution with singular value decomposition (SVD) of the covariance matrix.
In rare cases, multiple marker constellations may have been matched to the same learned target. Of all these possible matches the one with the lowest RMSD is chosen.

3.1.2.9 Target Tracking

When one or more markers of a target are not detected, in our implementation the target will not be detected. This choice was made to improve the robustness of target detection. However, the remaining markers can often still be tracked, when the target was recognized in the previous time step. When three markers of a target are detected and their IDs still tracked, all 6 DoF are computed. If only one or two markers are visible, only the translation of the target is updated.

3.1.2.10 Target Filter

For similar reasons as for marker filtering, the 6 DoF of a target may be filtered. This is especially interesting for the case when recovering from a lost tracking. Without filtering, the target jumps to its new state. It may be useful for some applications to smooth this jump. Also, for a pointing device, the angular errors may lead to unwanted jumping of the direction. Filtering can help to smooth the orientation, or the position for a path of a stroke while sketching.
At the moment no filter is employed. For the 3D glasses we prefer the lowest possible latency instead of smooth motion. As in daily use a quick head motion causes a temporal loss of target recognition, a prediction with a linear model is likely to fail. However, when suddenly no new information on the target is available, a quick but smooth slowing down of the motion may be more comfortable. We have not tested this yet.

3.1.2.11 Network Transfer

Finally, the target data is also copied to the network thread. While the main thread restarts its loop waiting for the results of the camera threads, the network thread takes care of sending the marker and target data to the recipients. Currently this is not done each time new data is available, but rather on request. The original idea was that the tracking server might extrapolate the data, trying to reduce the latency. We have not implemented an extrapolation, as we fear that overshooting may lead to more artifacts than the latency.

3.1.3 Latency and Timing

The diagram on the next page shows timings on a quadcore PC from early 2006.

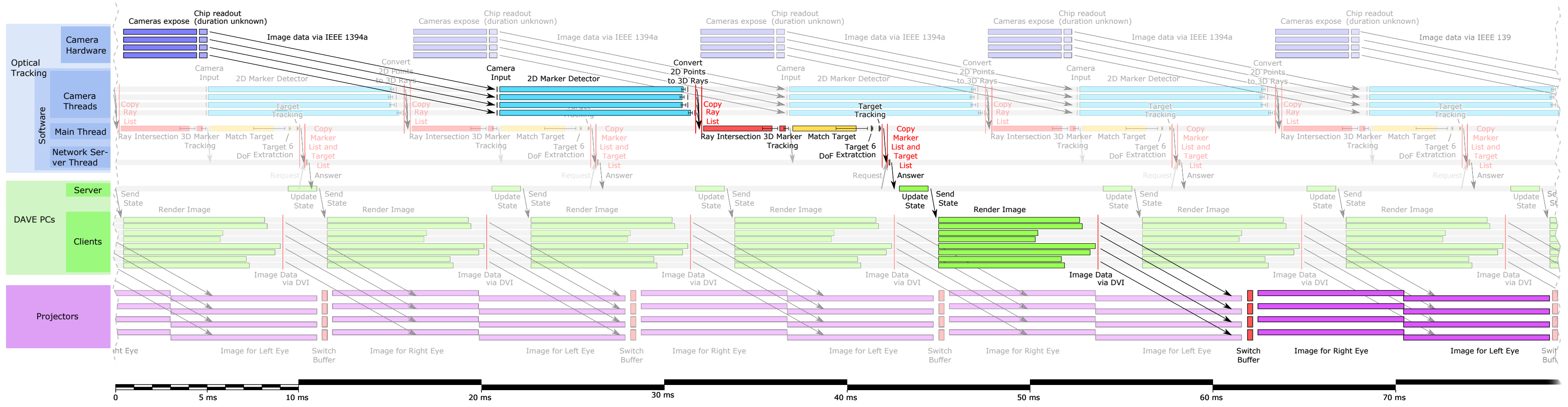


Figure 3.8: Idealized and simplified timing diagram of our optical tracking system and its relation to the DAVE timing. Processing durations have been measured repeatedly in a static scene with two targets with four markers each. Black bars at the end of each module show the variance. For a precise measurement, each module ran 1000 times in a row instead of once. Currently, multi thread synchronization and scheduling issues introduce additional delays not shown here.
The path from the top left to the bottom right shows a close to optimal timing, with about 70 ms delay. A worst case in this diagram would lead to almost 95 ms latency. Due to multi thread synchronization and scheduling issues, the tracking is in reality slower (43 Hz) than shown here (64 Hz). In fact, the latency measured in the DAVE averages to around 100 ms. Note that some values in the diagram could not be measured and were derived from computations and theoretical reasoning. Here, the rendering synchronization is shown for a Davelib program, see section 4.2.2.2 for more details.

The total latency in the DAVE was measured by rolling markers on the floor and projecting a point at the predicted position, assuming linear motion. Approximately 100ms latency is measured. This can be perceived, e.g. with the joystick and a rendered pointing ray for picking, when quickly moving the joystick. There are a number of possible improvements to reduce the latency of the tracking system:

- A brighter illumination for a shorter exposure time.
- Cameras with faster image transfer, e.g. a gigabit ethernet interface.
- Cameras with onboard marker detection (with a reported latency of 8.3ms, the affordable *V120:SLIM* cameras by *OptiTrack* with onboard processing are three times faster than our system).
- Better usage of temporal coherence.
- A region of interest of the camera that can be set quickly.
- A faster PC (the current one is five years old).
- A GPU implementation for target detection.
- An improved accuracy so that less rays must be handled by the ray intersection.
- A hybrid tracking, e.g. a combination with inertial sensors and gyroscopes.

Additionally, to reduce overall latency in the DAVE, rendering could be synchronized with the tracking system, starting once the new tracking data arrives. Finally it would help if the projectors could switch to the new image at once instead of buffering it, similar to disabling the v-sync of monitors.

3.1.4 Alternative System Using Epipolar Geometry

Instead of ray intersection in 3D space, another common approach is to transform the rays to lines in 2D image space of the other cameras. These epipolar lines are actually curves due to the radial lens distortion. The framework was modified to also allow this approach. However, this was never implemented to a degree where it worked well. This approach is used e.g. by the *io tracker*.

3.1.5 Asynchronous Image Capture

The original idea for image capture was to use a dynamic region of interest of the cameras. Delivering only a subwindow, the Prosilica EC750 cameras can transmit images with up to 200 Hz. The minimum and maximum 2D coordinates of currently visible markers, plus a safety border can be used to compute the region of interest for the next frame. As each camera has a differently sized region of interest, the capture rate is also different in general. With each camera measuring as quickly as possible, the 2D marker positions can be tracked and extrapolated to a common time step, raising the system frame rate in total.

When this was implemented (except for the extrapolation) and tested, to our surprise, the frame rate was much lower than without using the region of interest feature. This is due to an undocumented hardware limitation, causing a delay of 30ms - 40ms to set a new region of interest. On request the manufacturer informed us, that only the much more expensive cameras could instantly switch to a new region of interest. Of course, the region of interest could be set less frequently, but as this is necessary especially at fast movements, it does not make sense, as this is the time when the tracking is most important. As mentioned, the region of interest was also tested for just the software marker detection, but it does not result in a large speed up of the whole system.

3.1.6 Calibration for Optical Marker Tracking

Camera calibration techniques for intrinsic and extrinsic camera calibration are presented by Tsai [Tsa92] and Zhang [Zha99]. There, one or more calibration images are used. Often checkerboard patterns are employed, allowing a simple and precise automatic detection of the reference points. However, in our setup a large calibration pattern is necessary for accurate results. For practical reasons, commercial 3D tracking systems use a different method, consisting by two steps. First, a calibration target is wanded randomly in the tracking volume to register the cameras to each other, then a reference target is used to define the coordinate system. Motivated by that process we also use wanding but just with a single marker. This idea is also described in [SMP05]. Differences between one and two point targets for calibration are discussed in [DU02]. With bundle adjustment and especially outlier detection, the calibration with two markers at a fixed distance will be quicker and more robust, see [TMHF00] for a comprehensive description the subject.
One of the weakest points of the tracking system is its calibration. It is not very user friendly and takes some time, but it is also only rarely necessary, i.e. about once every two years in the DAVE.

3.1.6.1 Calibration Procedure

The calibration procedure consists of several steps. First, a single marker is recorded while being placed at a number of different positions, one after another, whose world coordinates are known or measured with a tape measure. Second, that marker is slowly wanded, trying to cover most of the important tracking volume. During the measurements, its 2D positions in all camera images are recorded. Third, the necessary data is specified in the calibration configuration file, i.e. the time codes when the markers were recorded at the known world coordinates plus the respective coordinate values, as well as the start and end time of the wanding. This step may be avoided, e.g. by analysing the recorded data or by pushing a button of a wireless device each time the marker is on a reference position or the wanding starts or stops. It is also helpful to specify a rough position of the cameras. Fourth, the interactive calibration software is started, where the camera viewing direction can be rotated with the mouse for a rough initialization of the orientation in order to provide good enough start values for the optimization.

3.1.6.2 Camera Models

A standard camera model with intrinsic and extrinsic parameters is used, as illustrated below.

```
Ray get3DRay(float pixelX, float pixelY) {
  int maxWidthHeight = max(width, height);

  // normalized pixel coordinates
  double nrmX = (marker[i].x−imgCenterX) / maxWidthHeight;
  double nrmY = (marker[i].y−imgCenterY) / maxWidthHeight;
  double distSqr = (nrmX∗nrmX) + (nrmY∗nrmY); // squared distance from image center

  // undistorted pixel coordinates
  double undistX = nrmX ∗ (1 + distSqr ∗ (radialDistortionDistP2 + distSqr∗radialDistortionDistP4));
  double undistY = nrmY ∗ (1 + distSqr ∗ (radialDistortionDistP2 + distSqr∗radialDistortionDistP4));

  Ray ray;
  ray.x = position [0]; // camera position
  ray.y = position [1];
  ray.z = position [2];
  ray.dx = imgPlaneDir.x + undistX∗imgAxisX.x + undistY∗imgAxisY.x; // direction
  ray.dy = imgPlaneDir.y + undistX∗imgAxisX.y + undistY∗imgAxisY.y;
  ray.dz = imgPlaneDir.z + undistX∗imgAxisX.z + undistY∗imgAxisY.z;
  return ray;
}
```

Figure 3.9: A standard camera model with second order radial lens distortion, used for the 3D optical tracking.

The model above is restrictive. Lens distortion that cannot be described by the model can not be corrected for. As the mean backprojection error of 0.6 to 0.9 pixels is larger than we had expected, an additional local lens deformation model was tested, but no improvements were achieved. Ricolfe-Viala et al. show a more sophisticated realization of this idea [RVSS10], requiring a separate calibration of lens distortion with a large number of known point positions on a planar calibration target.
The literature suggests that the employed camera model should be accurate enough.
Another attempt was to use a generalized camera model, where the direction is computed by a general function with 36 parameters, as shown below.

```
Ray get3DRay(float pixelX, float pixelY) {
  // normalized pixel coordinates
  int maxWidthHeight = max(width, height);
  double u = (pixelX−imgCenterX) / maxWidthHeight;
  double v = (pixelY−imgCenterY) / maxWidthHeight;

  Ray ray;
  ray.x = position [0]; // camera position
  ray.y = position [1];
  ray.z = position [2];

  // direction of the ray
  ray.dx = c[0];
  ray.dy = c[1];
  ray.dz = c[2];

  ray.dx += u∗c[3] + v∗c[4];
  ray.dy += u∗c[5] + v∗c[6];
  ray.dz += u∗c[7] + v∗c[8];

  ray.dx += u∗u∗c[9] + u∗v∗c[10] + v∗v∗c[11];
  ray.dy += u∗u∗c[12] + u∗v∗c[13] + v∗v∗c[14];
  ray.dz += u∗u∗c[15] + u∗v∗c[16] + v∗v∗c[17];
   ...
  ...
  ray.dx += u*u*u*c[18] + u*u*v*c[19] + u*v*v*c[20] + v*v*v*c[21];
  ray.dy += u*u*u*c[22] + u*u*v*c[23] + u*v*v*c[24] + v*v*v*c[25];
  ray.dz += u*u*u*c[26] + u*u*v*c[27] + u*v*v*c[28] + v*v*v*c[29];

  ray.dx += u*u*u*u*c[30] + v*v*v*v*c[31];
  ray.dy += u*u*u*u*c[32] + v*v*v*v*c[33];
  ray.dz += u*u*u*u*c[34] + v*v*v*v*c[35];

  return ray;
}
```

Figure 3.10: An attempt of an alternative generic camera model.

This camera model works, but the calibration is less robust. As an example, there is no assumption that the pixel coordinate system is orthogonal. This knowledge is not used and must be introduced with an additional error term. Also, more parameters need to be optimized and the optimization is trapped easier in local minima.

3.1.6.3 Optimization of Camera Parameters

For the optimization of the camera parameters, we developed an own tool that can visualize the result. For debugging purposes, this can be manually triggered in small steps and visually controlled, to better understand what happens during optimization. This proved as a valuable tool to find the right optimization conditions. The calibration uses a simple to implement, steepest descend method, that unfortunately suffers from local minima problems. To help escape from such minima, we can also modify the parameters by small random values, usually achieving better results after a new optimization. However, with a good initialization, like the values of a previous calibration, the whole procedure is quite quick and soon leads to good results.

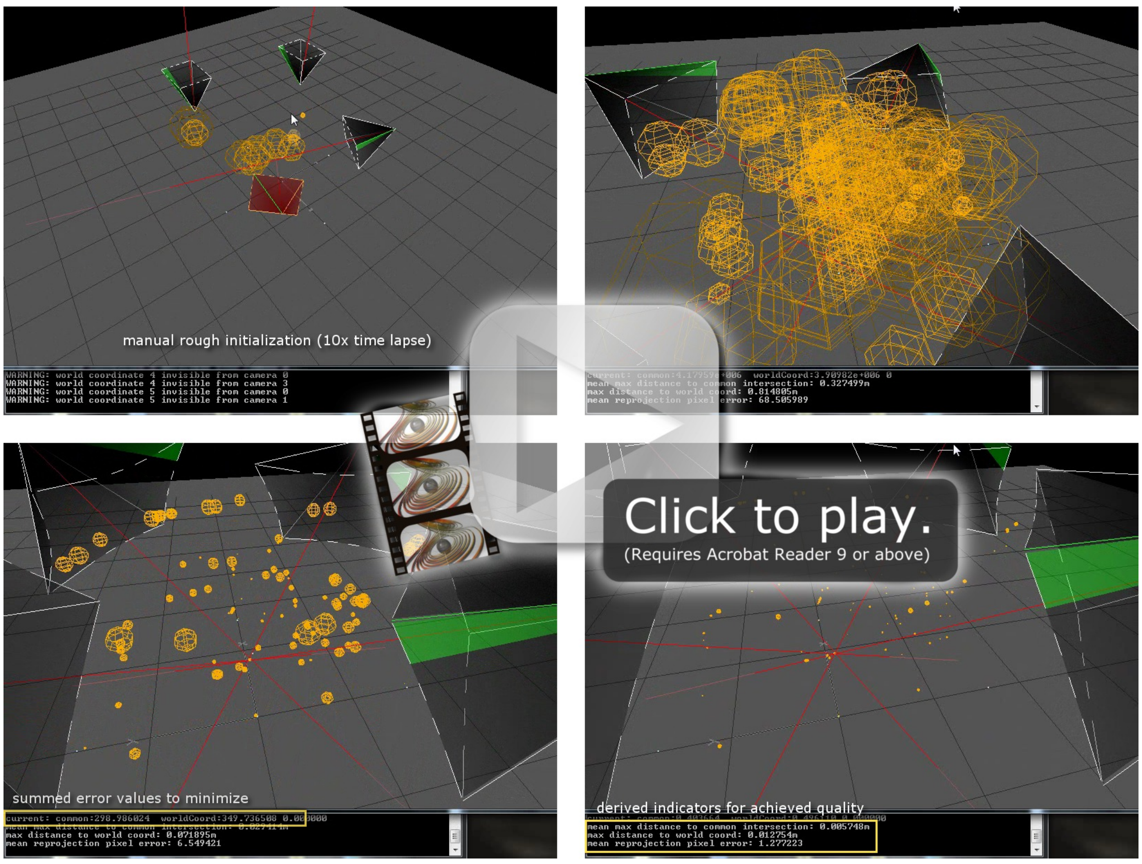
Minimization Terms and Conditions. It is not easy to intuitively define the correct error terms to minimize. As an example, our first attempts minimizing the ray intersection distance caused all cameras just to move very close together, soon leading to numerical instabilities. A few optimization criteria were tested to find a robust optimization. Our final solution is a weighted sum of two error values. Both of them sum the fourth powers of distances. The high exponent stronger penalizes large errors. The first set of distances is used to snap the solution to the measured world coordinates. The distances of the rays to these coordinates are computed and their fourth powers are summed up to give the first error value. The second set of distances uses the data from wanding. A few hundred measurements are randomly selected. To avoid a possible error at fast motion due to software synchronization, only slowly moving marker positions are accepted. only accepting measurements during a slow motion of the marker. The distances of each ray to their common intersection point are computed and their fourth powers are summed up, resulting in the second error value. The choice of these distances to the common intersection point proved to be much more stable for optimization than the direct ray to ray distance which tends to move the cameras together. The two weights should be adjusted so that these two error values, normalized to the number of used points, should lie in the same order of magnitude. Outliers are not rejected but are not a problem, as the lighting can be well controlled during calibration. In

all the years, only a single time an outlier occured, which could easily be seen in the visualization and ignored by restarting the program, thereby choosing a different set of random markers.
The steepest decent minimization needs many iterations and gets stuck in local minima. Also, a good initialization is required. It is used because of its simple implementation. The pseudo code below illustrates the optimization. As this minimization does not converge very well, it is run several times with different step lengths. To avoid local minima, a low amount of random noise may be added.

```
float computeError() {
  // distance of rays to predefined world coordinates should be minimal
  float error1 = 0;
  for (all worldCoordinateDefs w)
    for (int cam=0; cam<cameras.size(); cam++) {
      Ray ray = cameras[c].get3DRay(w);
      // squared distance of point to ray
      float distSqr = pointRaySquareDist(ray, worldCDefs[w]);
      error1 += distSqr*distSqr; // power of 4
    }

  // distance of rays to common intersection 3D position should be minimal
  float error2 = 0;
  for (all randomly chosen frames f from wanding) {
    // add distances of each line per frame to each other line
    Point point = computeIntersection(f);
    for (all cameras c) {
      Ray ray = cameras[c].get3DRay(f);
      // squared distance of point to ray
      float distSqr = pointRaySquareDist(ray, point);
      error2 += distSqr*distSqr; // power of 4
    }
  }
  return error1 + error2;
}

void minimizeOneStep() {
  // compute gradient numerically
  float current_value = computeError();
  for (all cameras c) {
    for (all camera parameters p) {
      original_parameter = camera[c].parameter[p]; // backup parameter
      camera[c].parameter[p] += small_value;       // modify a little
      new_value = computeError();
      gradient[c][p] = new_value - current_value;
      camera[c].parameter[p] = original_parameter; // restore parameter
    }
  }
  gradient.normalize();

  // step along gradient
  for (all cameras c) {
    for (all camera parameters p) {
      camera[c].parameter[p] -= stepSize * gradient;
    }
  }
}
```

Figure 3.11: Pseudo code for one step of a simple steepest gradient minimization.

The video below shows the tool for calibration, interactively showing the results. To get an impression of the quality of the calibration, statistics are computed and printed for the mean of the maximum distance to a common intersection (4mm in the example), the maximum distance to a reference coordinate (8mm in the example) and the mean reprojection error in pixels (0.9 pixels in the example).


manual rough initialization (10x time lapse)
summed error values to minimize
derived indicators for achieved quality
Click to play.
(Requires Acrobat Reader 9 or above)

```
WARNING: world coordinate 4 invisible from camera 0
WARNING: world coordinate 4 invisible from camera 3
WARNING: world coordinate 5 invisible from camera 0
WARNING: world coordinate 5 invisible from camera 1
```

```
current: common:4.17959e+006  worldCoord:3.90982e+006 0
mean max distance to common intersection: 0.327499m
max distance to world coord: 0.814805m
mean reprojection pixel error: 68.505989
```

```
current: common:298.986024  worldCoord:349.736508 0.000000
mean max distance to common intersection: 0.029414m
max distance to world coord: 0.071895m
mean reprojection pixel error: 6.549421
```

```
current: common:0.403664  worldCoord:0.496110 0.000000
mean max distance to common intersection: 0.005748m
max distance to world coord: 0.012754m
mean reprojection pixel error: 1.277223
```

Figure 3.12: Video (without audio) showing a time lapse calibration procedure with the interactive optimization application. After manually setting a rough initial guess, the evolution of camera parameter optimization to a precise calibration is shown. Note the distorted camera frustum. The whole video is sped up by a factor of ten.

3.1.6.4 Target Creation and Learning

To distinguish targets, they need to be sufficiently different to each other. But also the pairwise marker distances within a target should be sufficiently different in order to minimize computation time. For four markers there are already six pairs of markers and make the task harder than it may first appear. But also if e.g. two distances are very similar, the tracking will still work, because the best fitting target of multiple options is chosen in the end. The *iotracker* software suite even provides a tool to help generate good targets, presented by Pintaric et al. in [PK08].
To add a new target, its marker positions are measured. This assumes the target is placed at the origin of the coordinate system. If e.g. an additional translation is necessary, the values can be modified manually. It can happen that an additional marker is measured which has to be removed manually. Since the positions are based on a single measurement, they may not be very precise. It is possible to activate a target optimization module, that computes continuous measurements of a moving target. Average pairwise distances are used to appropriately adjusts the target vertex positions.

3.1.6.5 Possible Improvements of the Calibration

We found that it is hard to simplify the whole calibration procedure for the operator. Other tracking system calibrations are similar, but often use a special target to define the origin and orientation of the world coordinate system, rather than measuring the positions with a tape measure. Usually, such targets are relatively small for practical reasons, thereby limiting the absolute precision. The camera parameter optimization of commercial systems usually works orders of magnitudes faster than our current implementation. A reduced number of unknown parameters will lead to a more robust optimization with possibly more accurate results. This can be achieved by a separate calibration of the lens distortion parameters. Often, printed checkerboard patterns are used for this purpose. In the DAVE setup this is unpractical, as a small pattern is not in focus close to the camera, but the pattern should cover a large part of the image. Also, additional illumination or retroreflective calibration patterns are necessary. Instead, several images of a straight line can be recorded for each camera. A retroreflective stripe can be stretched under tension by a bow to ensure linearity. The line in the camera image appears curved, and in the camera image points on this curve can easily be extracted automatically to optimize the lens distortion parameters, as described in [AF01].

3.1.7 Related Applications for 3D Tracking

Tracking System for Magnetic Resonance Imaging. For a different application we assessed the accuracy of the tracking with two cameras for a small working volume of about $30cm \times 30cm \times 20cm$. In Magnetic Resonance Imaging (MRI) of a head, the scan takes some time and motion is an issue. Slices of the head are scanned with progressing time. With head tracking, the excitation plane can be corrected according to the current head position. With that method, a registered scan without gaps results. Measuring a target composed of reflective markers attached to a mouthpiece, with cameras several meters away, the accuracy is not high enough. Details can be found in [Dol10]. To increase the accuracy, the idea is to place two cameras with the least amount of metal as possible very close to the head inside the scanner, hoping not to interfere with the scan results too much. The PC for signal processing should be outside the shielded room and communication between cameras and PC must be achieved with an optical link, as electrical wires would act like antennas and reduce the scan quality. Unfortunately, with our implementation, the required accuracy cannot be achieved yet. To further investigate the reasons, a lot more effort is necessary to individually assess the accuracy of the different components. As an example, we suspect that the lens distortion model may be insufficient for the lens focused at a close range.

Camera Tracking. There is another application for the algorithms above, that at the first sight has not much in common: camera tracking software, alias matchmoving software. For a 2D movie of a mostly static scene, it computes the camera position and orientation relative to the scene. Using natural features instead of markers and treating each image of the movie as if it was taken by another camera, the task is very similar to the calibration described above. Subsequently, the 3D position of the natural features can be computed to help composition of 3D elements. In fact, *Vicon*, one of the leading companies of 3D tracking systems, also offers a high end professional camera tracking software called *boujou*.

Motion Capturing. To record human motion with optical tracking, markers are placed close to joints to recover a skeleton animation. As the tracking of single markers can fail due to occlusions or fast motion, this is not a trivial task. An editor was implemented to automatically remove wrong detections and connect trails in a first step. Length constraints and heuristics are used. If the automatic processing cannot recover all information, a manual editing is possible. A short movie was produced with a human animation recorded with this method. A real time pose detection was planned, but the data provided by the tracking system is not reliable enough.

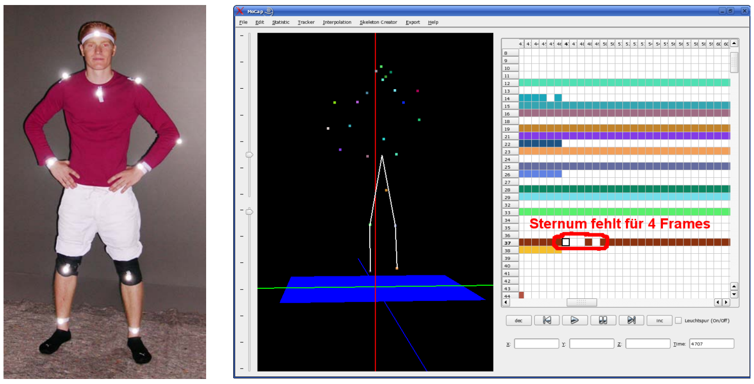


Figure 3.13: Motion Capture editor.

3.1.8 Results and Future Work

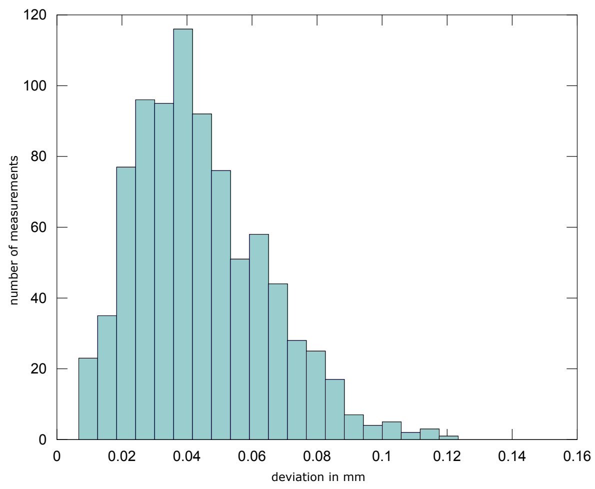


Figure 3.14: Jitter of a static marker near the center of the tracking volume. The histogram of the distances to the mean position of the marker is illustrated. The jitter is caused by the influence of sensor noise of the camera (mean distance error = 0.044mm).

Accuracy. Measurements are provided by the calibration software to in order to get an overview of the quality of the calibration. These values are not directly optimized and by using a set of random measurements, different to the measurements used for optimization, the following results are achieved. For these measurements, the system is used as it is configured in daily use, with the calibration done three months before. Motivated by an analysis of the iotracker in [PK07], we also performed similar measurements.

The jitter of a static marker close to the center of the tracking volume was measured. The jitter occurs due to sensor noise and does have a very small influence on the position, as expected. Our results are similar to [PK07].

We also measured the absolute distance between two markers rigidly attached to a stick, similar to [PK07]. The stick wa moved with different speeds in the center and at the border of the tracking volume and occlusions were present. The results below indicate that the measured error consists of a term related to the position within the volume and a systematic constant term. These errors can probably be largely decreased by using such a stick for calibration, imposing the fixed distance as an additional optimization criterion. Our results seem to be slightly better than the ones in [PK07]. In the measurements, partial occlusions of multiple markers hardly occurred. In practice however, markers of a target lead to such constellations and decrease the accuracy.

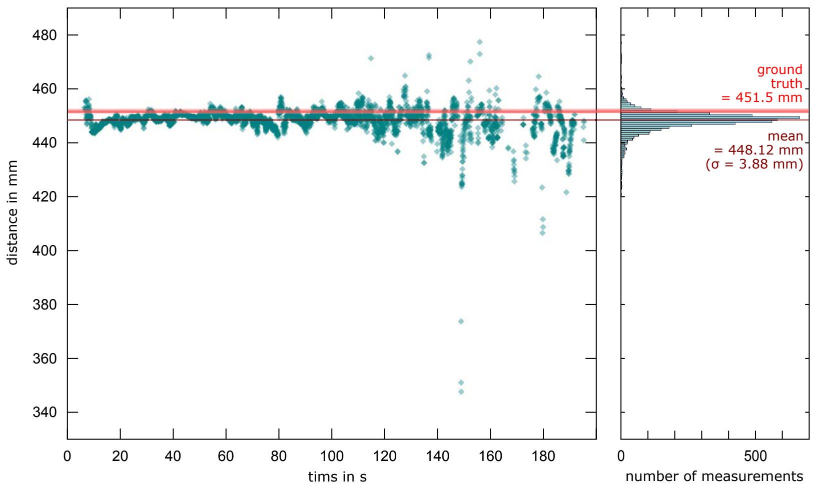


Figure 3.15: Distance measurements between two markers on a stick. The stick was moved slowly near the center of the tracking volume first, then quicker and further towards the borders. At the end of the measurements, sometimes one marker was occluded and the other one was detected twice. These outliers were removed for the computation of the mean and standard deviation. The systematic scaling error of 0.75% is caused by the slightly biased calibration method.

In another experiment a marker was moved around on the floor of the DAVE. The floor is based on a wooden construction and is not exactly planar, but is still good enough for an impression of the absolute accuracy of the measured height. The results show a clear offset towards the up direction. We assume that this offset is mostly due to the scaling error. Again, this measurement could be integrated into the calibration and largely improve the results. Such a criterion should be optional, as not all setups may have a tracking volume that extends to a floor or another planar surface.

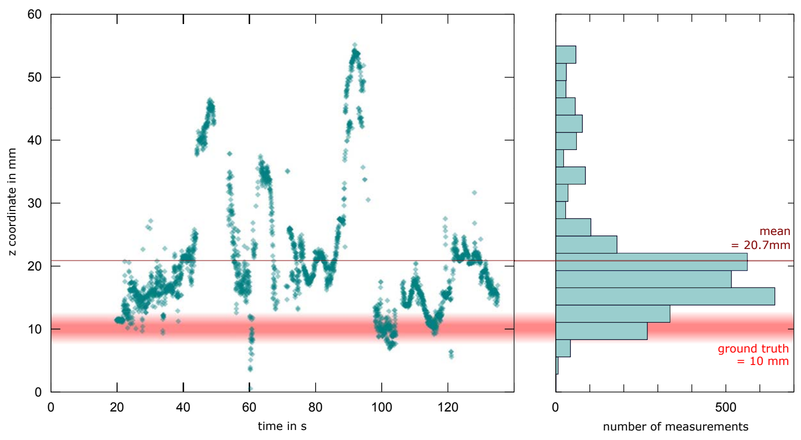


Figure 3.16: Assessment of the absolute error. A marker was slowly moved along the floor of the DAVE, i.e. at the border of the calibration volume.

For the angular jitter of a static target we recorded the orientation (as in [PK07]) for several positions in the tracking volume. In one corner, no jitter was present at all. We assume that all markers were only detected with a single pixel above the threshold. This means that this analysis is not meaningful. Better meaningful relative accuracy measurements may achieved by recording a target with a known motion. This may be realized e.g. by mounting it on a large turntable or the wheel of a bike. The perfect circle may be fitted from the data, the deviation from the interpolated position may give a more meaningful information on the error of position and orientation.

Update Rate. As the threads for capturing the camera images and processing the 2D information start with some delay, they are not synchronized at the program start. A number of locks, guards and barriers are placed to synchronize the threads while trying to keep the latency as low as possible. The maximum frame rate is limited by the image transfer of the cameras to around 64 Hz. The actual framerate with few markers is only around 50 Hz, even though processing should be just fast enough. A possible improvement may be to move the camera triggering into the main thread by adding an additional module. In that way, the synchronization by hardware will better fit to the *Prosilica* API. When also showing graphical debugging information, the framerate drops to about 30 Hz, as OpenGL is used by each thread but the same context only allows exclusive access to one thread at a time. As this mode is only used for debugging, this is no problem. For tracking more than two or three targets at that framerate, further performance improvements are necessary.

Conclusion and Future Improvements. We succeeded in creating an affordable tracking system that is used in the DAVE. We also had the vision to release the software as open source to the community to provide a basic system, ultimately hoping to also profit from development by other contributors. Unfortunately, this could never be realized due to conflicting commercial interests of our lab.
The most valuable possible improvements are a better calibration procedure, hardware synchronization of the cameras and brighter flashes, and a GUI for a simpler configuration.

3.2 MULTI TOUCH TRACKING

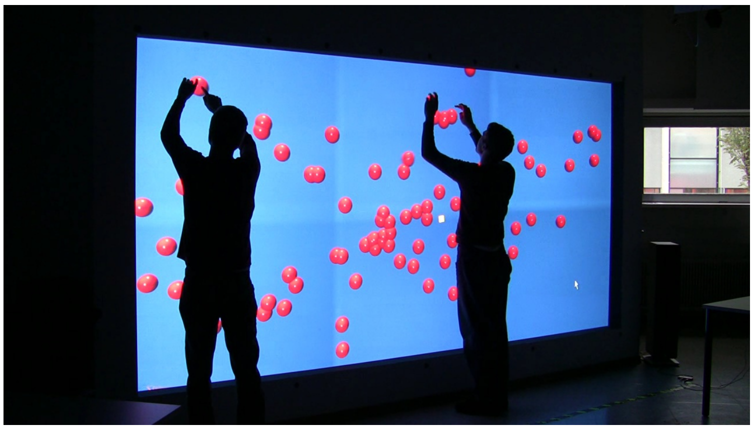


Figure 3.17: Multi touch test on the HEyeWall.

3.2.1 Related Work

Freely available and open source libraries exist to detect bright spots in a camera image and send the positons over a network interface. *Tbeta* (formerly *touchlib*) and *reacTIVision* are examples. Also, more flexible vision tools like VVVV can be used for the task. In some libraries, performance is not optimal or difficult lighting conditions cannot be handled.
The *TUIO* protocol [KBBC05] is established as a quasi standard as a network interface for the detected points. However, recent commercial hardware also or only supports the Windows 7 multi touch messages.
Freely available multi touch tracking frameworks already existed when we built our first setup. However, none of them supported multiple cameras natively. A possible work-around was to resample and stitch several camera images into a single large image to then pass it to an existing framework. The popular *touchlib* was written using a set of modular image filters. Greyscale conversion, brightness, contrast, thresholding and blob detection were each individual modules having to access and modify the whole image data for each filter. With that work-around, stitching and all filters are processed by a single thread, resulting in low update rate.

3.2.2 Tracking System Software Components

For the sake of speed and accuracy, own software was written, reusing large parts of the 3D tracking software. It natively supports multiple cameras and allows remote calibration when the cameras are not connected to the displaying computer. Furthermore, the IR illumination can be controlled, e.g. in order to capture FTIR and DI in alternating frames.

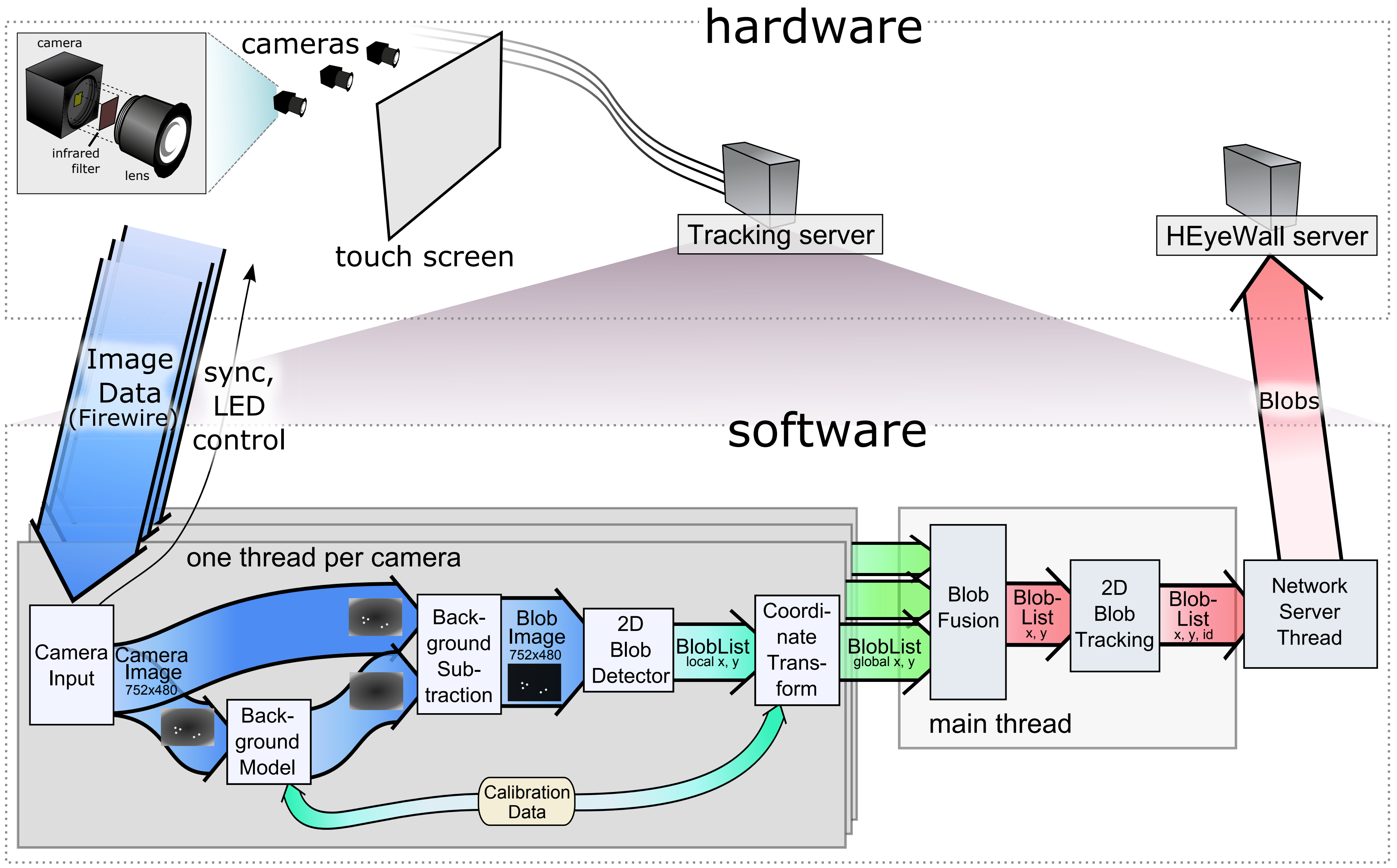


Figure 3.18: Multi touch tracking software overview for the HEyeWall Graz. For smaller setups with one camera, a single PC is sufficient for tracking, program control and image generation.

A naïve approach to use multiple camera images is to first undistort each image for radial lens distortion and then stitch the individual images into one larger image. This step takes a lot of time and resampling artifacts occur. Instead, in our system, finger touches are detected individually, with one thread per camera image. Subsequently, lens distortion correction and transformation into a global coordinate system are performed. The main thread fuses this information and passes the results to the network thread. This approach does not lead to resampling artifacts in the camera images, needs a lot less processing and parallelizes all time consuming computations with multiple threads. Again, the idea of open sourcing the software could never be realized.
As large parts of the 3D optical tracking software were reused, only the differences are described below.

3.2.2.1 Image Capture and Transfer

The image capture module supports controlling an IO port of the camera that is used to switch the IR illumination on or off. This allows to capture a background image with only the ambient light for a subsequent background subtraction. Especially to allow adaption to slowly changing ambient lighting, this may be useful. Currently, the illumination is constantly turned on.

3.2.2.2 Background Model

A problem of the hardware setup described above is that the reflection of the projector in the acrylic glass is visible in the camera image as a very bright spot. Without further hardware filters or redundant cameras, this area is a blind spot to the touch input. Also the IR illumination might be directly visible or lead to bright areas in the camera image. The figure below illustrates the situation.
To avoid wrong detections caused by that light, background subtraction may be used. Unfortunately, the acrylic glass slightly bends under the pressure of a finger and the reflection of the projector moves slightly. With the imperfect silicone coating of the HEyeWall, air bubbles appear as small bright spots that also move when touching the screen. They are very hard to automatically distinguish from bright spots caused by touching fingers. Also, the camera may move slightly so that the LEDs or the frame surrounding and holding the screen may cause wrong detections, e.g. when users lean on the border of a table setup.

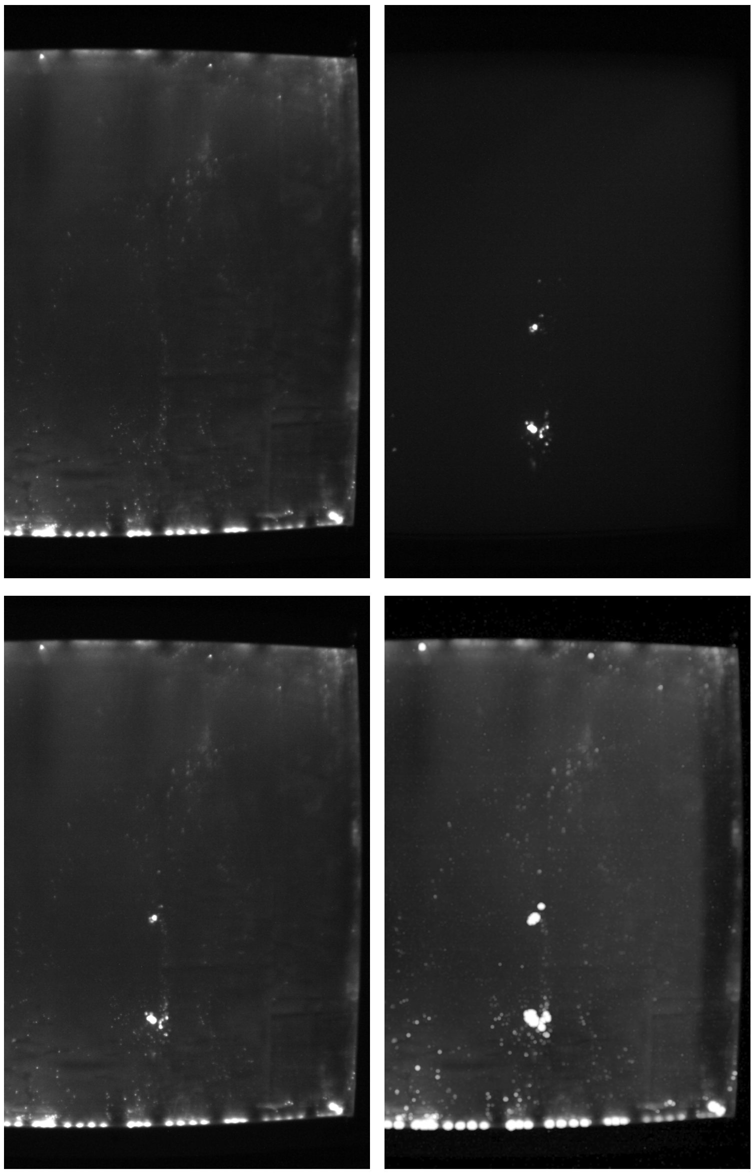


Figure 3.19: Camera images of the setup, showing the illumination. LEDs only (top left), projectors only (top right) and both together (bottom left) as the condition in normal operation. For calibration, using the 3^{rd} brightest pixel of some 20 captured images, pushing on different parts of the screen, moving reflections can be taken into account without also recording the touches. The resulting eroded image for background subtraction is shown on the bottom right. Note that white regions are blind spots for that camera. For better illustration, the images are brightened with a gain factor (first three images) and gamma correction (last image).

We use three methods to solve these problems, two of them are done in this module and described below.
First, to obtain a background image, twenty images are recorded with half a second delay between each capture. During this time the projectors display a white image. A short beep indicates the moments when images are captured. The operator pushes on different locations during that process. For each pixel, the third brightest measured value of all images is used. In that way, the touched positions will not lead to blind spots, but the moving reflections are taken into account. The numbers can be changed in the configuration file.

Second, a morphological dilation is performed with the structuring element shown on the side. This operation smoothly extends bright regions to allow a slight motion of the bright spots without an erroneous detection.
The third method is a mask image that is applied in the next module and is described there.
Over time, the lighting situation may change slightly, leading to wrong marker detections. This is a common problem in practice. A dynamic update of the background image would be advisable to adapt to such slow changes automatically. This may be possible when the screen is in use, so that locations where no touch is detected are used to learn the updated background. However, if the screen is not used, the projector reflections are static and can not be adapted to the situation when they move again. This issue should be addressed by improved hardware as suggested in the previous chapter.

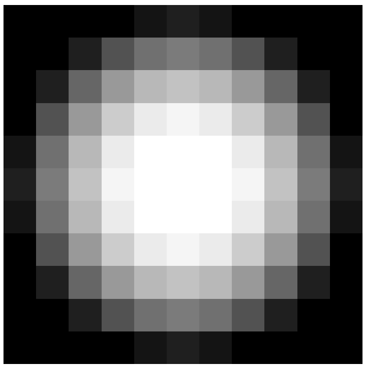


Figure 3.20: Structuring element for morphological dilation. This leads to extended bright regions in the background image.

3.2.2.3 Blob Detection

The blob detection module is almost identical to the marker detector of the 3D tracking system (see section 3.1.2.2). The approach is more effective compared to the *touchlib* that sacrifices performance with many passes (five for our test setup).
In addition to the described detection, an additional check is performed per pixel, comparing the gray value of a mask image to a configured threshold. Only if the value is higher than the threshold, marker detection is performed for that pixel position. This allows to mask out previously defined regions, addressing the problem of detections caused by the frame of the screen. This is the third method to avoid unwanted detections. The generation of this mask is done by the calibration module and is described in section 3.2.3.2.
The blob detection usually performs very well, but for a fast motion of a single finger, in rare cases two blobs may result that are close to each other. This may break the tracking and cause a change of the ID.

3.2.2.4 Transformation to World Coordinate System

The detected blob coordinates are transformed from the camera coordinate system to the common world coordinate system. We chose the same world coordinate system that is used for the geometric calibration of the display (see section 4.2.3.1). Two parameters for radial lens distortion and a homography are computed by the calibration. All of the camera parameters are listed below.

```
 float a, b, tx, c, d, ty, e, f, g; // homography
 float radialDistortionDistP2; // radial  distortion , degree 2
 float radialDistortionDistP4; // radial  distortion , degree 4
 float imgCenterX, imgCenterY; // image center in pixels, for radial
     distortion
int width, height; // camera image dimensions
```

Figure 3.21: Parameters of camera model for the multi touch tracking system.

The application of the transformation is straight forward, as the following code shows.

```cpp
Vec2 pixel2World(float pixel.x, float pixel.y) {
  int maxWidthHeight = max(width, height); // of camera image

  // lens distortion
  double nrmX = (pixel.x − imgCenterX) / maxWidthHeight; //
      normalized pixel coordinates
  double nrmY = (pixel.y − imgCenterY) / maxWidthHeight;
  double distSqr = (nrmX*nrmX) + (nrmY*nrmY); // squared distance
      from image center
  double undistX = nrmX * (1 + distSqr * (radialDistortionDistP2 +
      distSqr*radialDistortionDistP4)); // undistorted pixel coordinates
  double undistY = nrmY * (1 + distSqr * (radialDistortionDistP2 +
      distSqr*radialDistortionDistP4)); // undistorted pixel coordinates

  // homography
  double w = e*undistX + f*undistY + g;
  float worldX = (a*undistX + b*undistY + tx)/w;
  float worldY = (c*undistX + d*undistY + ty)/w;
  return Vec2(worldX, worldY);
}
```

Figure 3.22: Transformation of detected points from the camera coordinate system to the world coordinate system, using the camera parameters from the previous listing.

Note that during measurements for calibration of the camera parameters, no transformation is applied in this module and the coordinates stay unmodified.

3.2.2.5 Blob Fusion

Fusing the information of multiple camera images is realized by fusing close blobs from different cameras. When the same finger is detected by two cameras, the transformation to the common world coordinate system results in the same position except for a small error by noise and imprecise calibration. These blobs from different cameras within a threshold distance are fused to a single blob.
It is also possible to use two cameras looking at the same area from different viewpoints. This can be useful e.g. to avoid the blind spots from a projector reflection. It also allows to handle occlusions in setups where the camera and user are on the same side of the screen, like it is the case for a front projection setup as described in section 3.2.5.1.

3.2.2.6 2D Blob Tracking

To reassign the IDs known from the previous time steps to the newly detected blobs, a more sophisticated algorithm is used than for the 3D tracking.
As before, the position of a circular search area of a predicted blob is derived by old known blobs. However, the radius is not constant but grows with the blob's speed. This allows a higher tolerance for quickly moving fingers. Again, the problem arises that several newly detected blobs might be within the search area of a single old blob, so only the closest blob is chosen get that ID assigned. Even with a single erroneous frame with two detections instead of one, the wrong one may get the ID, causing a release event of the old ID in the next frame when a single marker is detected again.
It is desirable for some applications that the same ID is kept for the same finger. To avoid a change of the ID when no finger is detected only for a few frames, the information is kept for a few frames, mov-

ing into the predicted direction with a growing search radius. The prediction assumes linear motion plus drag to quickly slow down motion of predicted markers.

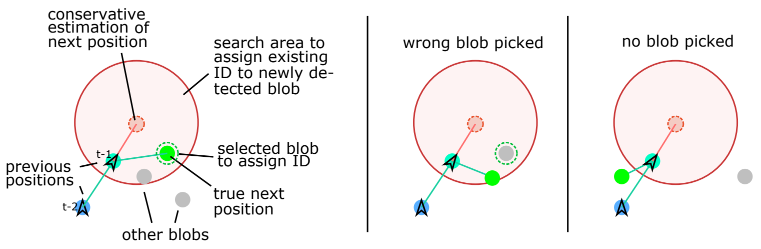

conservative estimation of next position
search area to assign existing ID to newly detected blob
selected blob to assign ID
previous positions
t-1
t-2
true next position
other blobs
wrong blob picked
no blob picked

Figure 3.23: Blob tracking working correctly on the left. The center and right figures show failure cases.

Some external applications assume that the first touching finger always has the ID 0. To enhance compatibility with these applications, the ID counter is reset after a short while without any detections.

3.2.2.7 Network Transfer: TUIO and Proprietary Protocol

The TUIO protocol is used to send the information to the client applications and is widely used. For the calibration, a proprietary protocol is used that can transmit the camera number for each detected point for the calibration. A new version of TUIO could also be used, as it allows to embed custom data fields.
To work with applications that only support the Windows 7 messages, freeware bridges exist that translate the messages from TUIO to Windows 7 without a noticeable delay.

3.2.3 Calibration

3.2.3.1 Geometric Calibration

As the cameras cannot see the display content, the easiest way for geometric calibration is to display a few points that the operator has to touch, one after another. To add robustness, we chose to require that the touch must be one second long. During that time, an arc is displayed that completes a circle, providing a progress feedback (see section 4.3.4.2). In case of the HEyeWall, the measurements rely on geometrically calibrated projector images.
The first eleven parameters of the camera model are optimized, i.e. homography and some radial lens distortion parameters. The center of the distortion is assumed to be at image center, a good approximation that helps to avoid overfitting. With two variables per measured point, at least six calibration points are necessary for each camera. The more points are added, the better the calibration gets. At the moment, no outlier detection and rejection is performed, assuming perfect measurements.
As display and tracking for the HEyeWall is split up to several computers, the program measuring these calibration points is a normal Davelib program (see section 4.2), with the tracking running on a different computer in a special calibration mode. The already described custom network protocol is used that includes the camera number for every touch point.

The calibration itself is again computed with the steepest gradient optimization to minimize the squared distances. For this case, it is stable and works well. Even though a good initialization is not hard to compute, for our setups it is sufficient to initialize with the identity and no lens distortion, also for the HEyeWall setup where the cameras are rotated by 90^{circ}.

3.2.3.2 Mask Image Generation

To address the problem with the frame generating detections outside of the screen, a mask image is computed only using the geometric calibration data as input. It is realized as a distance transformation, where the gray values represent the distance from the border in pixels.

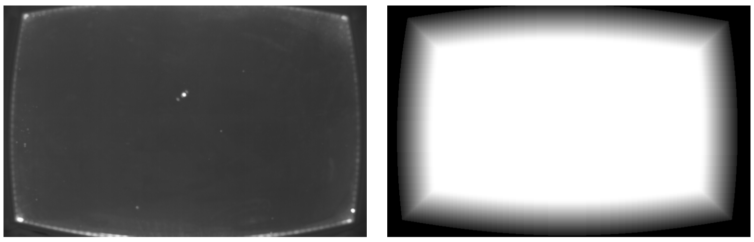


Figure 3.24: Background image of our table setup on the left. In this setup, leaning on the table can easily cause the frame to generate several touch points on and outside the border of the visible region. To efficiently ignore these signals, we automatically compute a distance map that is used as a lookup to only generate blobs inside the area visible to the users, with a border width that can be configured. It is shown on the right with adjusted levels for better illustration.

In that way, the border width can be configured without having to redo the computation by simply changing the threshold distance. The mask image may also be edited manually to mask out additional regions, e.g. for setups that do not have a rectangular screen.

3.2.3.3 Calibration Procedure

The calibration procedure consists of the following steps:

- Background training
- Pushing on a few points that are displayed
- Computation of the calibration computation (geometric and mask image)

For the HEyeWall, the second step is performed separately for each camera. A script calls the same program with different parameters. In that way, no specialized software is necessary that only works for the HEyeWall (see also section 4.3.4.2).

In some setups, the second step may be automated when the projector light is at least slightly visible to the camera. The projector shows the point to be calibrated as a white circle on a black background. A long time exposure may show the difference in the image and can directly be used as camera input, resulting in a detected blob and therefore a fully automatic calibration.

3.2.4 Target Detection

Some multi touch hardware can directly see objects on the display surface, similar to a camera. In a table setup, tangible objects may be used for interaction. This is often achieved by using patterns on their bottom side, so called fiducials. Our setups can only sense touch points like fingers. Normal fiducial objects are too light for our FTIR setups in order to allow a recognition. The dreaMTouch hardware can only deliver areas that are shaded, i.e. an approximate convex hull of objects.

However, this is enough to allow a recognition of a fixed configuration. Similar to the targets of the 3D tracking, 2D markers can be attached to the bottom of an object. For the dreaMTouch setup with line scan cameras at the display edge, we use felt gliders that block light like finger tips would. For the FTIR setup this is unpractical, as the high required pressure would require a very heavy object. Instead, infrared LEDs can be used under the object, facing down.

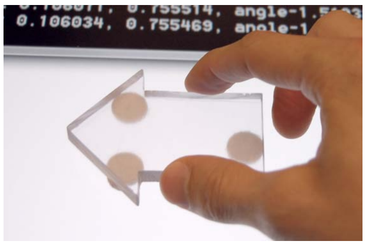


Figure 3.25: A tangible object, an arrow with felt gliders used as fiducial. Using pairwise distances of the single detected touch points, the touch screen can be retrofitted with such a target recognition. The object is made out of acrylic glass so that not too much screen content is hidden.

To enable target recognition also on commercial setups, an own software was written to filter a TUIO stream and replace all recognized touch points with the respective TUIO objects. The result is output to another TUIO port.

As in the 3D target recognition, pair wise distances of markers, or here touch positions, are used for the recognition of an object. The compared distances tolerate an absolute distance error that should be slightly higher than the device resolution, including signal noise. Since the TUIO protocol uses a normalized coordinate system with values from zero to one for both x and y axes, the y components of distances are scaled to match the aspect ratio of the screen. Although an arbitrary number of blobs per object is supported, our dreaMTouch setup cannot distinguish several blobs that are very close to each other. Thus, for the small tangible arrow in Figure 3.25, only three felt gliders are used.

To learn a new object it is placed at the center of the screen and all blobs are recorded to a file. Ideally, all pair wise blob distances should be significantly different, leading to asymmetric constellations. For each constellation of pair wise distances that fit to a previously learned object, the position and orientation of the recognized object are computed. Finally, the detected objects and all blobs that do not belong to any detected object are collected and forwarded.

During our tests, accidental recognition of a set of fingers as an object rarely occurred. Even intentionally trying to trick the system by placing fingers at distances of the felt gliders, it is not easy to trigger object recognition. A single measurement is taken to learn a new reference object. For a better detection and a tighter edge distance tolerance value, an average distance of many measurements on different positions on the screen should be used, like in the 3D optical tracking software. Currently, objects are tracked by detection in each frame, without using temporal coherence. When an object is placed or removed from the screen, only some blobs may be detected and add unwanted blob messages. For applications that use both objects and blobs, this may lead to unintentional behavior. Objects should probably be only detected when all blobs appear within a short time period. Blobs that formerly belonged to an object should be suppressed once the object is lifted. It is also possible that a finger grabbing the fiducial object is accidentally recognized as a blob. Other physical shapes of objects can easily solve this problem. We think that a higher object is likely to be grabbed at a higher position and should solve this issue. In summary, this method allows to retrofit fiducial object recognition to multi touch devices that do not natively support them, without modifying neither the multi touch driver nor the application.

3.2.5 Alternative Setups

3.2.5.1 Laser Pointer on Projection Screen

As shown in figure 2.16, most of the optical multi touch technologies rely on blob detection in 2D camera images. With no or little modification, our software also works for other hardware setups than FTIR.
An example is an experiment with a laptop, a projector, a webcam and a laser pointer that is pointed on the projected image. As the camera can see the light of the display, the geometric calibration is performed automatically. After that calibration, the camera exposure time is turned down and the background image is captured with the projector showing a white image. The laser pointer dot needs to appear sufficiently brighter than the projector image. This can be achieved with a bright laser pointer or with a matching narrow bandwidth filter in front of the camera lens. In general, the user will be a few meters away from the screen. It is not easy to perform precise tasks and accurately aim with a hand held laser pointer. One idea to improve that application is to constantly use a lower powered preview point or an additional laser diode with a different color to help aiming. Also, a Kalman filter may help to stabilize the recognized cursor, at the cost of a delayed signal.
For the HEyeWall setup, an infrared laser pointer with a visible laser pointer as a guide for aiming may be used additionally to the touch input, without any change of the tracking system.

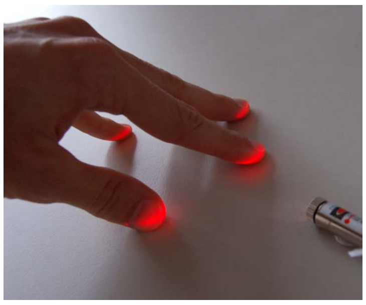


Figure 3.26: Laser Light Plane setup. Fingers entering the plane light up. As they shadow fingers behind, multiple laser line modules can be placed around the screen. Here, a normal table surface turns into a multi touch surface with cheap components.

3.2.5.2 Laser Light Plane

The Laser Light Plane (LLP) setup uses a few laser line modules. They consist of a laser diode and a grid of small acrylic cylindrical lenses and are very cheap (around 3 EUR). With a mechanical setup allowing precise adjustments, large areas can be lit. For safety reasons, we prefer the visible red wavelength for our experiments. Again, our software can directly be used, without the need of special adaptions except for configuration parameters.

3.2.6 Performance and Latency

The PC used for tracking at the HEyeWall from 2009 has four cores. With the system running at 43 Hz, three cores have a load of about 50%, with one core at about 10% or less. The full frame rate of the cameras of about 60 Hz is not achieved due to multi threading and camera synchronization issues. As currently the illumination is not bright enough, the exposure time is set to 30ms, thus limiting the update rate.

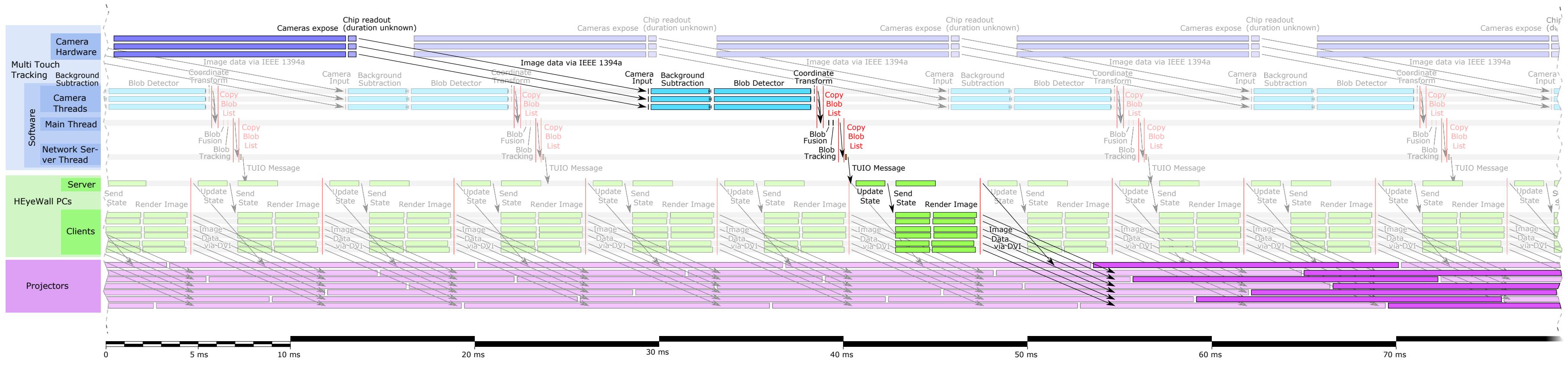


Figure 3.27: Idealized and simplified timing diagram of multi touch tracking and its relation to the HEyeWall timing. Processing durations have been measured repeatedly in a static scene with a few touch points. Black bars at the end of each module show the variance. For a precise measurement, each module ran 1000 times in a row instead of once. The path from the top left to the bottom right shows an exemplary timing, with about 80 ms latency. Due to multi thread synchronization and scheduling issues, the tracking is in practice slower (up to 43 Hz) than the possible speed shown here (64 Hz). Also, currently the illumination is not bright enough and the exposure time is set to 30ms, further limiting the update rate, which is not illustrated here.

3.2.7 Results and Future Work

Four main algorithmic differences make the development superior to most other multi touch software:

- Most time consuming operations are multi threaded, with one thread per camera.
- For image processing, only two modules, thus two passes are used.
- Remapping or stitching of camera images is not necessary, as lens distortion and stitching are only computed for the 2D marker positions.
- The stable background subtraction can robustly deal with imperfect hardware and lighting conditions.

3.2.7.1 Future Improvements

Solving illumination issues with hardware changes and reducing synchronization overhead of the multi threading are the most important issues to address. With the lighting issues under control, a dynamic, slowly updating background model may be used, if still necessary.

3.3 MISCELLANEOUS

3.3.1 Monocular 2D and 3D Marker Tracking

An early test with a simple marker tracking in an AR application was performed, as shown below.

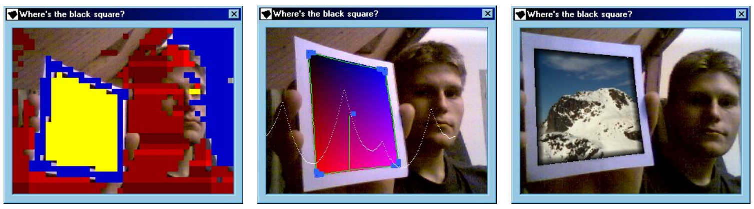
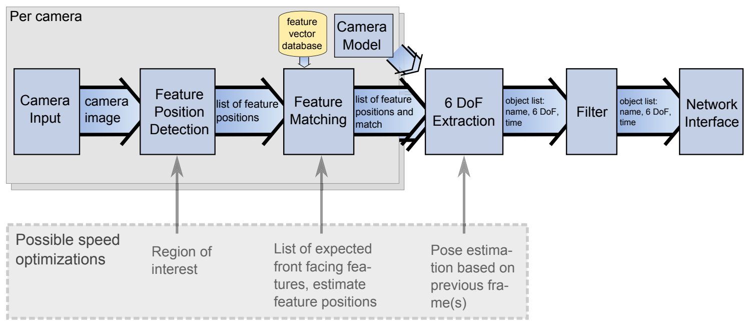

Where's the black square?
Where's the black square?
Where's the black square?

Figure 3.28: Own implementation of a 2D AR marker application. Marker identification (left), corner detection (center) and compositing of a respectively warped photo on the marker (right).

Later, a framework for tracking input devices with 6DoF from a single camera was partially implemented. Previously learned natural feature points on the object are especially interesting, as they do not require artificial looking patterns on the object. With known object coordinates for each feature, the 6DoF of the object relative to the camera can be computed. This opens up the field of Augmented Reality applications. We worked on such a system, with the goal of being able to use a proxy object as an input device, only requiring a webcam.



Per camera
feature vector database
Camera Model
Camera Input
camera image
Feature Position Detection
list of feature positions
Feature Matching
list of feature positions and match
6 DoF Extraction
object list: name, 6 DoF, time
Filter
object list: name, 6 DoF, time
Network Interface
Possible speed optimizations
Region of interest
List of expected front facing fea-tures, estimate feature positions
Pose estima-tion based on previous fra-me(s)

Figure 3.29: Tangible Six Degrees of Freedom Input Device - system overview. The system was never finished, as at the time no suitable natural feature tracking library was available. Instead, a different hardware solution with an accelerometer was used (see section 2.3.5).

We realized the large amount of effort that is necessary for an own implementation of natural feature point detection and eventually solved the problem in a different way, as described in section 2.3.5.

3.3.2 Camera Based Automatic Display Calibration

A display setup consisting of multiple projectors usually needs calibration so that individual images fit well together. Especially if a calibration needs to be performed often (e.g. in case of a mobile setup) or is done by non-experts, an automatic geometric and colorimetric calibration using a camera can be quick and accurate.
A survey of camera based calibration techniques for photometric and geometric calibration is presented by Brown et al. in [BMY05]. Zhou et al. show continuously self calibrating projectors, using one rigidly attached camera for each projector [ZWAY08]. Grundhöfer et al. show real-time radiometric compensation [GB08]. A comprehensive overview of projector-camera systems is given by Bimber et al. in [BIWG08].
However, many of these approaches above rely on special hardware and algorithms are often not very tolerant to changes in environment, like illumination conditions. For a quick calibration, the camera needs to be synchronized with the projector, requiring additional effort. The dynamic range of the camera may not be sufficient, requiring an HDR capture with multiple images at different exposures. As we rarely need to recalibrate our display wall, the effort of setting up a camera and implementing automatic calibration algorithms is not justified. Instead, we perform the geometric calibration manually, as described in section 4.2.3.1, [Ras00] and [LOUF06]. An example is the geometric calibration for the DAVE. The non-linear distortions by the mirrors and non-planar screens make an exact calibration with only a 4×4 matrix impossible. For a manual calibration, the operator can easily decide which corner is more important and judge where errors have less impact. This could also be done by software, again with more effort. As a geometric calibration in the DAVE and HEyeWall are rarely necessary, using the manual calibration that takes just a few minutes is not a problem.

3.3.3 3D Scanner

A different motivation leads to another use of a camera: 3D scanning. Especially with the background of digitization and archiving of 3D museum exhibition items, large amounts of objects need to be scanned. This is only feasible with fully automatic scanning. Often, laser scanners are used, with a camera capturing a laser line that is moved over the object. We built a scanner with less hardware effort based on silhouette carving, including a computer controlled turn table. A fully automatic camera calibration was added. Other than the turn table, only a web cam is necessary, leading to a low cost hardware. While with that approach some concave shapes cannot be recovered correctly, always a water tight model results and per vertex colors are obtained from the webcam images. As this work is a student project from 2002 and rather dated, the basic steps are only briefly covered.

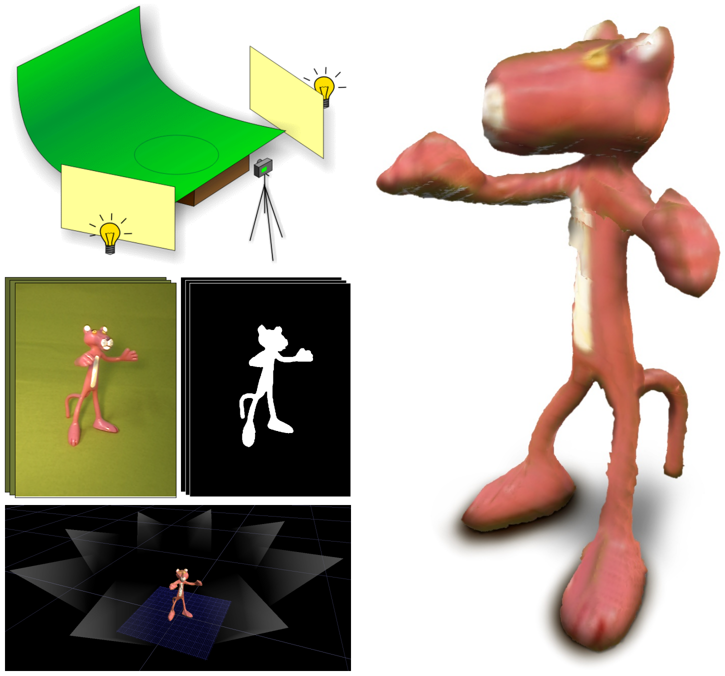


Figure 3.30: 3D model (right) obtained with our low cost 3D scanner (top left). The camera images and automatic segmentation results are shown in the left center. The camera frustums relative to the object together with the result are visible in the screen capture at the bottom left.

3.3.3.1 Automatic Camera Calibration for the 3D Scanner

The automatic calibration uses two calibration patterns on perpendicular planes. To automatically find the calibration points in the camera image, a series of image processing filters is applied (see illustration below). The camera parameters are obtained using Tsai's method [Tsa92]. Then the turn table rotates by about 15 degrees and the process is repeated, in order to compute the rotation axis. As long as the turntable and camera are not moved, several objects can be scanned with the same calibration.

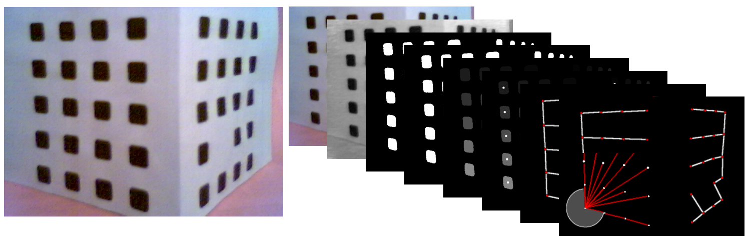


Figure 3.31: Image processing for automatic calibration. The image of the calibration object is converted to greyscale, binarized, eroded and labeled. The centers of each label are connected with a minimum spanning tree, removing its longest edge splits the two sides. The different number of markers identify the side. Using the angle of the convex hull of each set of points, the corners are found and the points can finally be registered. After a rotation, a second photo is processed to automatically compute the rotation axis of the turntable.

3.3.3.2 Scanning Process

The object is put on the turntable and photos are taken for a specified number of views. A watershed segmentation in the HSV color space is used to get a binary segmentation of the object. In a voxel volume, the visibility is computed for each voxel by backprojection into each segmented camera image. Afterwards the marching cubes algorithm [LC87] is used. It is modified to obtain a water tight mesh. Each vertex is backprojected in the camera images to obtain color information.
For some objects it is useful obtain views from different angles. An additional camera may be used, or the object may be scanned a second time lying on the side. However, experiments with the latter approach were not successful as the merging of two scans had to be estimated by the user without any visualization, which is not precise enough. The software was made freely available in the internet. Considering the poor image quality of the webcam, we obtained good results. For more details, please refer to [LH03].

Possible Improvements. Image based reconstruction has progressed since. The same hardware and images can be used for a higher quality reconstruction and without the need of calibration images. The *Arc3D*[CCD*08] webservice or the *PhotoFly* project by *Autodesk* allow such reconstructions.
A research project called *KinectFusion* [INK*11] additionally employs the kinect depth sensor to acquire even more reliable geometry in real time.

4

Output: Image Rendering

For rendering 3D graphics in a Virtual Environment, additional requirements arise compared to common non-immersive setups. Many of them can be addressed easily, provided that the source code of the application is available. The Davelib is described in this chapter, a lean and minimalistic, yet fully functional framework to develop new programs and to easily adapt existing programs. Also, the necessary extensions for a novel display concept are presented, a tiled seamless back projection display, the *Frequency Split Display* realized with the HEyeWall Graz.

4.1 RELATED WORK

4.1.1 VR Frameworks and Tools

The large variety of VR setups and hardware is described in chapter 2. Unfortunately, no VR operating system exists that could run an application in any VE, given the appropriate drivers for the hardware. No commonly accepted interfaces exist to specify display setups, input methods, or how communication of a group of computers and devices is realized.

Instead, individual solutions seem to be developed for almost each setup. A number of VR frameworks exist that try to facilitate development by providing standard functionality and allow a configuration of the specific setup. This works well for displays, and for head tracking to some extend, but other input devices are more diverse and must be adapted for each setup.

But even if a better framework or configuration standard was available than the currently used one, it often means a lot of effort to switch to the new system. Only a significant improvement justifies this step. The ideal situation is that a properly written application can be distributed as executable and can be run in any VE without further modifications.

4.1.1.1 Call Interception on Graphics Driver Level

A good idea to transfer any 3D content in a VE without modifying the executable of an OpenGL application is to intercept the API calls. Chromium[HHN*02] achieves this by providing a fake OpenGL driver that handles the API calls before forwarding them to the real OpenGL driver of the graphics card. In that way, all OpenGL commands can be distributed over the network to other machines in a cluster and can be executed there as well. For the correct view on each screen, the projection matrix is modified accordingly. In addition, additional stream filters may be applied in order to modify the rendering style. While this seems to be a universal solution at first sight, a number of possible problems exist. Most importantly, view frustum culling is used in many interesting applications. Objects that are not visible in the original camera frustum are not even send to the OpenGL driver. When running such an application in a CAVE, many objects that should be displayed on the side and floor screens will be missing. However, it should work well with many applications on 2D tiled displays walls and performs a lot better than first rendering one large resolution image that is then split up and distributed via network. WireGL [HEB*01] is a similar predecessor with the intention to compute small tiles of an image in a cluster and reassemble them on a single machine. This may result in a very uneven load balance. To better distribute the work load, another method is to distribute the rendering on multiple graphics cards by splitting up the content. Subsequently, the resulting framebuffers are merged.

A related method is used by the *NVIDIA 3D Vision* stereo driver, especially targeted at games. The driver takes care of rendering two images with a different stereo offset by automatically duplicating render targets, attaching an additional footer to each vertex shader and by duplicating render calls. The user can adjust eye separation and convergence, as the screen size and distance are unknown to the driver. To speed up rendering, not the whole rendering is duplicated. Instead, a heuristic is used, trying avoid computing the same intermediate results twice [McD10]. As an example, shadow maps are independent of the left or right eye and must be only rendered once. Again, software usually has to be adapted (see section 4.2.5.1) and the driver configured for each application. A special issue arises for the calculation of a 3D position for the respective fragment position during deferred shading that must be addressed [NVI11]. The 3D Vision requires NVIDIA graphics cards, so called *3D ready* displays and Windows PCs.

4.1.1.2 VR Frameworks

Many VR frameworks exist and it seems that most VR research groups maintain their own code. This is probably due to the few and very different hardware setups and the required flexibility. Also, a growing code base and experience with the own system means more effort to switch to a new system.
VR Juggler is a comprehensive open source suite including libraries for device management, math, threads, network, sound, navigation and configuration functions. It is very powerful but also rather complex and takes some time to get started. It is compatible with several options for rendering, such as OpenGL, DirectX, OpenSG, OpenSceneGraph and OpenGL Performer^{TM}.
Scene graphs are useful for a hierarchical organization of the spatial content, allowing object transformation and meta data information for the nodes. They often provide useful methods, e.g. a ray intersection request can deliver the first object or node that is hit by the ray.
OpenSG is an open source scene graph system with a rich set of functionality. One main feature is its integrated network synchronization for clusters. Building on OpenSG, instantreality provides a high level access using the 3D content file format X3D and scripting to quickly develop VR applications. It is closed source but freely available with small limitations. OpenSceneGraph is another scene graph system similar to OpenSG. For many of our own applications we use both OpenSG and instantreality.
Many more VR and AR frameworks exist that run on Windows, Linux and often more platforms. They can run on clusters and in CAVEs and usually support a range of specific input devices. Many of them are free or even open source. Many support an interface to a scripting language. Even though the main features are often very similar, the software architecture and support by tool or libraries can be quite different.
A lot of these frameworks try to provide a comprehensive set of tools and functions, intended to develop new applications. However an integration into a existing 3D applications is often not easy. The Cross-Platform Cluster Graphics Library (CGLX) is rather lean and similar to our Davelib: the same application runs on each computer and is synchronized via application state events. It is used for large tiled displays and does not support other display geometries. The interface is similar to GLUT.
A few further examples of free frameworks are *OpenGL Performer* by Silicon Graphics International (SGI), *DIVERSE* by Virginia Tech,

OpenRM by the R3vis Corporation, Avango^{TM} by Fraunhofer, *Functional Reactive Virtual Reality (FRVR)* by the University of Hamburg, the *Virtual Reality Virtual Rendering System (VR2S)* by the University of Münster, *Virtual Reality for Scientific Technical Applications (VISTA)* [SGvR*03], [KBG09] by RWTH Aachen, *Equalizer* by the University of Zurich, the *Virtual Rendering System (VRS)* by the University of Potsdam, Maverik by the University of Manchester, *Studierstube* by Graz University of Technology or *Syzygy* by the University of Illinois, to name a few.
Examples of commercial frameworks are *3DVIA Virtools* by Dassault Systems, the Visual Decision Platform (VDP) by IC.IDO, products by Mechdyne such as the Cavelib, OpenInventor®, COIN by Kongsberg Oil & Gas Technologies or COVISE by Visenso.
The lists are by no means comprehensive and the large number shows that there is no single perfect solution. While a few frameworks seem not to be maintained any more, new frameworks and extensions are still developed, like a VR framework specialized for molecular simulations [FDGB08] or *VR JuggLua* [PV11], a combination of VR Juggler, Lua and OpenSceneGraph.

4.1.1.3 Libraries in the Context of VR and AR.

Some libraries exist to solve a specific problem relevant to the context of AR and VR. The *ARToolkit* enables 6DoF marker detection especially designed for AR applications. Reitmayr et al. present *OpenTracker* [RS01], [RS05], an interface developed to provide a unified access to tracking systems and to take care of different coordinate systems. This can be modeled with a data flow graph defined in an XML file. It was extended to support more input modalities like joystick buttons, a keyboard or a mouse. Such input events can be remapped by filters. The Virtual-Reality Peripheral Network (VRPN) [THS*01] provides access to remote devices via the network.
Many applications use scripting functionality. *Lua* is a small and fast cross platform scripting language that is a common choice, especially for games.

4.1.1.4 Game Engines.

State of the art game engines are powerful frameworks for realistic rendering of large models. They are often specialized to terrains and sophisticated tools exist to model levels for a game that are rich of visual features and efficient to render. Advanced graphics effects are implemented, mostly trying to approximate global illumination. Physic engines are integrated for interaction with the objects. Network support for multiple players is a common feature. High quality content is often available from commercial games or the community. OGRE is an open source engine. More advanced are two important commercial engines, the Unreal Engine and the CryEngine. Both of them can be used for free in an educational context and allow modification by plugins and can likely be ported to a CAVE.
In fact, we have done that with the dated *Quake III Arena* engine when the source code was released (see Figure 6.3.3.1).

4.1.1.5 Desktop and Office Applications

For desktop and office applications in VR setups and distributed displays, conventional 2D windows are often displayed as textured quadrangles or flat boxes within 3D space. To be able to use all applications as they are available at a desktop setup, and to be able to use geometric calibration and blending of projector images, it is desirable to use a 3D compositor to display the windows. The *Compiz* compositing manager for Linux is such an example. Yet another approach for cluster rendering is to use a large desktop window and distribute the image data over the network so that clients can render it as a texture. It is often used for 2D desktop applications on a tiled display setup, as presented e.g. in our publication [USOF09].
The deskotheque project aims at providing both a personal workspace as well as a large display surface in order to support collaboration [PWS09].

4.1.2 Display Calibration

Often, multiple displays working together require calibration in order to avoid artifacts. The *geometric* calibration is used to correct the position of the screen content. The *colorimetric* calibration adapts the color range of the displays. The *photometric* or *radiometric* calibration is used to adapt bright and dark levels of the displays and may implicitly contain a colorimetric calibration.

4.1.2.1 Edge Blending for Overlapping Projections

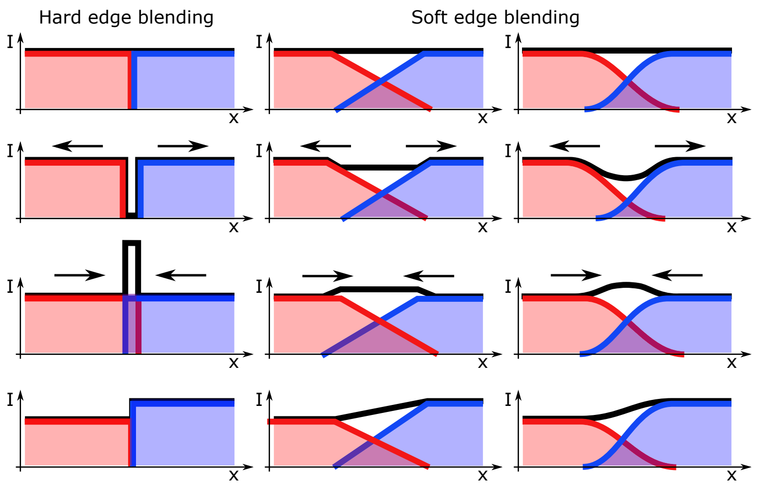


Figure 4.1: The intensities *I* of two projectors, across a transition along *x* are shown. Edge blending options: hard (left), soft with piecewise linear intensities (center), soft with smooth intensities (right). The top row shows that in all cases, both intensities sum up to a constant (black line) as intended. However, when the projectors are slightly misaligned (2nd and 3rd row, or when projector intensities are different (4th row), hard edge blending leads to visible artifacts.

As an exact mechanical alignment of projectors is often impractical. Adjacent projected images can be setup to slightly overlap. In the overlapping area the projected intensities add up. This can be avoided with hardware blend masks that physically block the light or software blend masks that attenuate the image content accordingly. The latter approach has the disadvantage, that the projector dark levels cannot be attenuated, resulting often visible artifacts in areas where multiple images overlap, such as inner corners in a 2D array setup (see the right image in Figure 4.27 for such an example). This can be only compensated by raising the common dark level, reducing dynamic range and contrast.
A sharp transition of one image to the next is called *hard edge blending*, as illustrated in the left column of the figure above. It is hardly used, as small misalignments lead to visible artifacts. A notable exception is the HEyeWall at IGD, Darmstadt, where physical blends are lo-

cated close to the screen [KRK03]. However, color non-uniformity of projectors leads to visible transitions, often even with a colorimetric calibration.
Another important reason for overlapping projector images is to achieve a smoother transition from one image to the next. A smooth blending is used in that case, called *soft edge blending*, as illustrated in the center and right columns of the figure above. A larger overlap leads to a less visible transition but decreases the overall usable brightness and resolution. A physical mask close to the projector lens casts its shadow out of focus, resulting in a soft transition. Metal stripes held by magnets can be used as masks, allowing an easy adjustment. Software attenuation masks can also be adjusted manually. However, especially for a larger number of projectors, manual adjustments are impractical and methods are used that automatically compute the masks.

Automatic Computation of Soft Edge Masks Raskar et al. show automatic geometric calibration using a camera per projector [RBY*99]. This method leads to small artifacts in the corners of the blend masks that are discussed in more detail in section 4.2.3.4.
Harville et al. present geometric and photometric calibration for a curved screen with special attention on its use in practice [HCS*06]. They also address the problem of computing a smooth blend mask in a very similar way to one of our methods. Their idea is to use a weighting function with a continuous first-order derivative inside the projected region. However, they overlook the discontinuity at the region border. Their approach is shown in the center column of the figure above, whereas we further modified the weighting functions to achieve the results shown in the right column (see section 4.2.3.4 and our publication [LF11b] for more details).
The larger the overlap of the projectors, the less visible is the transition. But a larger overlap also results in less bright and lower resolution projection. To address this problem, a super resolution projection approach exists where all projector images are superimposed [DVC07]. This method needs very accurate information about the pixel position of each projector. Small mechanical misalignment requires a new calibration.

4.1.2.2 Geometric, Photometric and Colorimetric Calibration

For curved screens, such as cylindrical screens or domes, non-linearly warped projector images are required. This can be achieved in a vertex shader [BR04], but this is just an approximation which only works well if the objects in image space are tesselated high enough.
A different, more common approach uses two passes for rendering [RWF98]. First, the image is rendered to a frame buffer object instead of the frame buffer itself. The second pass draws an appropriately deformed mesh, textured with the frame buffer object, to the frame buffer. This requires additional memory and processing time and can lead to resampling artifacts. However, all image content can be handled in a generic way without further modifications.
Smooth projection surfaces can be recovered with a camera, recording and processing calibration images [RWF98]. To achieve a correct geometric registration even for non-smooth surfaces, Bimber et al. use a look up for each pixel [BIWG08].

Majumder et al. show that photometric and colorimetric calibration can be achieved very well with the help of a camera [MS04]. However, the overall dynamic range of the display is greatly reduced. Later, they show that this reduction can partially avoided by taking perceptual thresholds into account [MS05b]. They allow non-uniform black and white levels with spatially smooth functions. Bimber et al. demonstrate that even very structured diffuse surfaces can be used to display arbitrary images while hiding the original structure to a high degree [BIWG08].
A serious issue especially of rear projection screens is a view dependent transmission, or a view dependent reflection for front projection screens. This effect is especially pronounced for high gain screens. But even the very low gain rear projection screens with a gain factor of 0.8 show wide but prominent hot spots. When using a calibration with a camera, the calibrated display will only look good from the camera position.
Bimber et al. compensate indirect scattering between projection surfaces [BGZ*06], e.g. of a CAVE. However, this effect has rather low frequencies even across screen borders and is less important.
Brightness and color temperature of a projector lamp change over its lifetime. With LCD projectors the situation is even worse, as also the LC modulator changes color. Especially in a projector array, the effects can become annoyingly visible. As an example, the first version of the HEyeWall at IGD, Darmstadt used LCD projectors with two projectors per tile for a stereo setup [KRK03] using spectral filters by Infitec. These filters sharply block several parts of the spectrum and therefore make the situation even worse. After a few years lifetime the display of a single color resulted in very visible color differences.
A relatively recent overview of projector camera systems is given in [BIWG08].

Semi-Automatic Calibration. Although many automatic calibration methods were presented, it does not justify the overhead for our setups and purposes. Thus, we use a manual registration of the corners to compute a geometric calibration with a 4×4 calibration matrix as described in section 4.2.3.1 and in [LOUF06]. It is easy to set up, does not require additional hardware and is flexible to use. Also no automatic photometric calibration is used. Instead, the lamp power settings are used in the DAVE and the intensity transfer functions for the HEyeWall projectors are approximated by brightness and gamma values that are adjusted with an interactive feedback.

4.1.3 Depth Perception

A combination of different visual effects leads to a human perception of depth. Most important are occlusion, motion parallax, perspective, stereoscopic viewing and accommodation clues. Objects in the distance have less contrast due to haze and lighting and shadows help as well. All these effects should be reproduced to achieve visual immersion.

Stereo Vision. As the eye positions are necessary for setting up the view and only head tracking is available in the DAVE, we estimate the eye positions and use an interpupillary distance of 6.3cm, the mean value of adults according to [Dod04]. As children are frequent visitors, a user dependent configuration was considered to set the distance accordingly. However, a quick test in an appartment model in the DAVE, showed a low influence for realistic results in the DAVE, even with a large error.About 5% to 10 % of people have limited or no stereoscopic abilities, the literature is vage about these numbers. Accommodation is correctly handled with help of the eye positions and respective display. Vergence however is not handled correctly in the DAVE (see section 2.1.2.1, a major cause of discomfort [HGAB08]. Note that stereoscopic vision with projective geometry on planar screens is correctly achieved with asymmetric frustums, opposed to toeing in the virtual cameras which rotates the projection plane and leads to unwanted vertical parallax.

Occlusions. Using standard real-time rendering techniques, occlusions are handled with the help of a depth buffer. However, e.g. in the DAVE, objects that should appear in front of the body will incorrectly be occluded by the body.

Motion Parallax. Motion parallax is as important as stereo vision, especially for close objects. When moving the head relative to a static scene, objects further away appear to move slower due to perspective. This effect can be well handled by head tracking but some delay occurs due to the latency of tracking system, rendering and display. Only light field or true volumetric displays do not suffer from such a delay.

Haze, Lighting and Shadows. Haze can be realized using the fixed function fog by OpenGL. As this is computed using the distance along a single vector, the visibility distance increases to infinity perpendicular to the viewing direction. Instead, the euclidean distance should be used instead with an own shader implementation. Most lighting and Shadow algorithms can be used without modification. Note that screen space based effects like screen space ambient occlusion or blooming may lead to visible edges at the screen borders.

Perceptual Deviations Loomis et al. have carried out a large number of experiments to study perception mismatches. Examples are that in their setups, a walked distance in VR is underestimated [LK03]. Similar studies showed that head rotation and scale are also affected in the experiments [SBJ*10]. While estimated values are also wrongly estimated in the test condition in reality, the VR condition shows a clear offset in most experiments. The reasons for these effects are unclear. Studies with HMDs showed, that the field of view and interpupilary distances that are perceived as natural differ from the correct values [BS11]. The results of such studies may help to compensate for such effects e.g. by slightly scaling the 3D model.

4.2 DAVELIB: A VR FRAMEWORK

4.2.1 Motivation and Goals

Simple Porting of Existing Applications to the DAVE. Porting an existing 3D application to the DAVE can be very simple. To demonstrate the lean concept of the Davelib, a minimal code example for a *GLUT* or *OpenGLEAN* program is shown below. The same program is executed on the server as well as the clients, each with different parameters. The additional lines that are necessary to port the application to the DAVE are highlighted in red.

```
#include <GL/openglean.h>
#include "Dave.h"
DAVE::Dave dave;

struct UserSyncStruct { // synchronize to clients
  float cameraPosition[3];
};

void display() {
  glClear(GL_COLOR_BUFFER_BIT |
  GL_DEPTH_BUFFER_BIT);
  dave.setProjectionMatrix();
  glLoadIdentity();

  UserSyncStruct* usd = (UserSyncStruct*) dave.getSyncBufferUserDataPtr();
  glTranslatefv(usd->cameraPosition);
  glutSolidTeapot(1.0);
  glutSwapBuffers();
}

void update() {
  // handle input devices (pseudo code)
  if (joystickButton[0].isPressed()) cameraPosition[2] += 0.1;

  // send/receive UserSyncStruct data
  dave.update();

  glutPostRedisplay();
}

int main(int argc, char* argv[]) {
  dave.parseCmdLine(argc, argv);
  dave.init(sizeof(UserSyncStruct));

  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DOUBLE |
  GLUT_RGB | GLUT_DEPTH);
  glutCreateWindow("Davelib 4");
  // ignore Davelib settings and always go into fullscreen
  glutFullScreen();
  glutSetCursor(GLUT_CURSOR_NONE);
  glutIdleFunc(update);
  glutDisplayFunc(display);
  glutMainLoop();
  return 0;
}
```

Figure 4.2: A minimal program with sample modifications for a CAVE setup (in red) using the Davelib.

The Davelib has evolved in the past 9 years of usage and is employed mainly in our DAVE. The main difference to many other VR frameworks is the lean design especially suited to be integrated in existing 3D applications. With the Davelib, only few function calls are necessary. Little time is required to learn how to use it and only one to two external dependencies exist (OpenGL and optionally Lua), simplifying integration in existing applications. Especially, the few external libraries help to simplify maintenance for developers of Davelib applications, as compiler versions and subsequently compatible library versions change over the years and the code using the libraries may need adaption to keep working. It is an open source cross platform (only Windows and Linux are tested) C++ library for OpenGL applications, but it should easily be portable to Java or modified for the use with DirectX. A wide range of setups is supported via configuration files and Lua scripting. This means that the application itself does not need to be recompiled and the same executable can be used. The last and only partially implemented step to fulfill this goal is the IO device library, a part of the Davelib that was developed primarily to support input devices that are setup specific, but it can also be used to control output devices.
The Davelib was initially written to provide the minimal necessary set of functions to run an application in our DAVE. It does not provide support for a GUI, sound, physics simulation, scene graphs, haptics, etc. but rather coexists to libraries written for these purposes.
The Davelib is a light weight set of basic functions that are necessary for OpenGL applications to work in a VR environment. While the focus stays on graphics functions for correct viewing, IO device scripting has been added for an easy adaption to a wide range of setups, including CAVE like setups, tiled displays and just normal PCs.

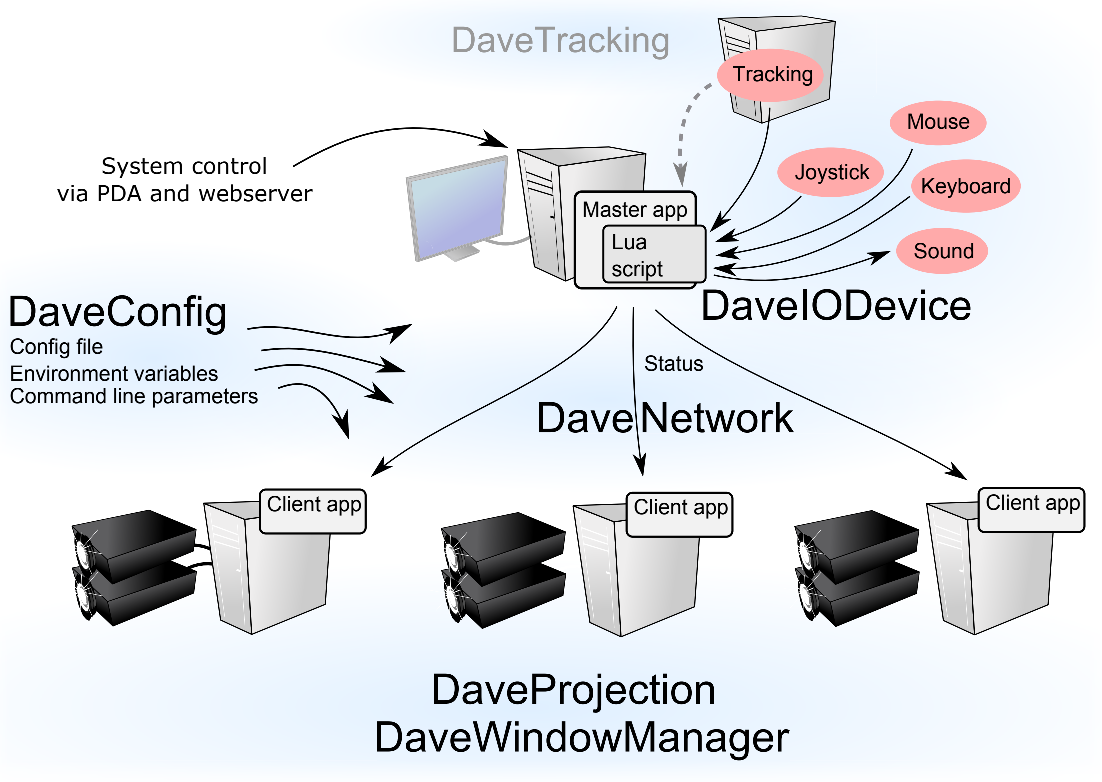


Figure 4.3: A simplified overview showing an example setup with the involved parts of the Davelib.

4.2.2 Davelib Core Functions

The Davelib consists of a carefully designed set of functions. There are no requirements or dependencies to use any of the Davelib functions, but the more are implemented, the more setups are supported. First, the basic components are described that are likely to be necessary for most VR setups. Afterwards, additional extensions are presented.

4.2.2.1 Screen Geometry and View Matrix Setup

The central technical aspect of a VE is the different view of the scene for each individual screen. In a conventional program for desktop setups, the OpenGL projection matrix is used to set up a perspective view looking into the virtual world. With the Davelib this matrix can be replaced by another matrix

$$P_{OpenGL} = P * R * T \qquad (4.2.1)$$

in order to give a correct view for a stereoscopic head tracked VR display. The OpenGL model view matrix can be used as normally, e.g. to change the camera position and orientation in the 3D scene and transform the specified vertices to local coordinate systems. An orthogonal view is currently not supported, as this only makes sense on 2D displays. When writing an application specific for a 2D tiled display, this may be a useful future extension which can easily be added.

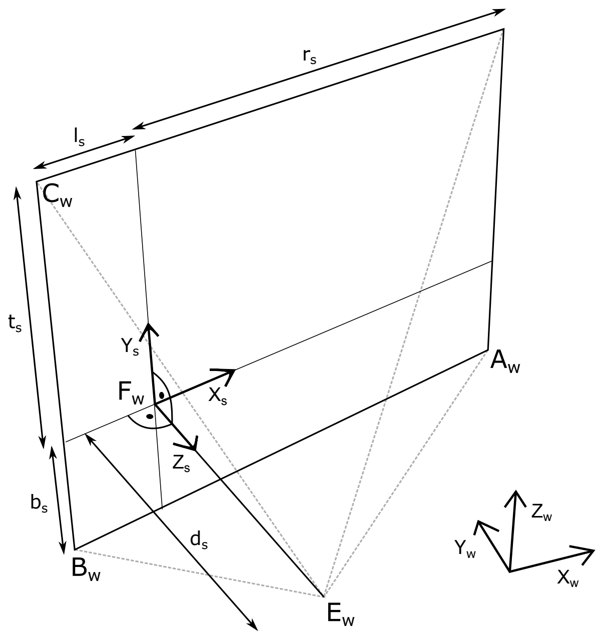


Figure 4.4: The asymmetric projective frustum for fixed screens and head tracking. A_w, B_w, C_w are corners of the rectangular screen and E_w is the eye position specified in world coordinates.

Static Screens. For projection screens, the camera is set up with a perspective frustum given by

$$P = \begin{pmatrix} \frac{2d_s}{r_s+l_s} & 0 & \frac{r_s-l_s}{r_s+l_s} & 0 \\ 0 & \frac{2d_s}{t_s+b_s} & \frac{t_s-b_s}{t_s+b_s} & 0 \\ 0 & 0 & \frac{z_{near}+z_{far}}{z_{near}-z_{far}} & \frac{2z_{near}z_{far}}{z_{near}-z_{far}} \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{4.2.2}$$

Note that in general the frustum for screens in a VE is asymmetric and depends on the eye positions that are estimated from head tracking [CNSD93].
The frustum may also be rotated to account for the rotation of the screen, e.g. for the side walls of a CAVE, which is realized with the rotation matrix

$$R = \begin{pmatrix} & & & 0 \\ \vec{X_s} & \vec{Y_s} & \vec{Y_s} & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.2.3}$$

Note that this rotation is independent of the rotation of the camera or navigation. Finally, a translation is applied with the negative eye position values.

$$T = \begin{pmatrix} & I & & -\vec{E_w} \\ & & & \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.2.4}$$

with the 3×3 identity matrix I.
To allow a simple configuration, only three corners A_w, B_w, C_w of the rectangular screen in the world coordinate system $\vec{X_w}, \vec{Y_w}, \vec{Z_w}$ and the distances of the near and far clipping planes z_{near} and z_{far} must be specified. From these values, the necessary parameters can be computed:

$$\begin{aligned} \vec{X_s} &= \|A_w - B_w\| \\ \vec{Y_s} &= \|C_w - B_w\| \\ \vec{Z_s} &= A_w \times B_w \\ r_s &= \overrightarrow{E_w A_w} \cdot \vec{X_s}^T \\ l_s &= \overrightarrow{E_w B_w} \cdot \vec{X_s}^T \\ t_s &= \overrightarrow{E_w C_w} \cdot \vec{Y_s}^T \\ b_s &= \overrightarrow{E_w B_w} \cdot \vec{Y_s}^T \\ d_s &= \overrightarrow{B_w E_w}) \cdot \vec{Z_s}^T \end{aligned}$$

The assumption for these equations is a rectangular screen, but only a part of it can be used for a planar screen with an arbitrarily shaped border, like a circle.

Head Mounted Display. For HMDs, P is set to a symmetric frustum:

$$P = \begin{pmatrix} \frac{1}{l_s} & 0 & 0 & 0 \\ 0 & \frac{1}{b_s} & 0 & 0 \\ 0 & 0 & \frac{z_{near}+z_{far}}{z_{near}-z_{far}} & \frac{2z_{near}z_{far}}{z_{near}-z_{far}} \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{4.2.6}$$

with

$$
\begin{aligned}
l_s = r_s &= tan\left(\frac{\alpha\pi}{360}\right)\\
b_s = t_s &= \frac{l_s}{a}
\end{aligned}
$$

where α is the horizontal field of view and a the screen aspect ratio. The rotation matrix R is directly set to the orientation matrix from head tracking. Like for static screens, afterwards a translation is applied as in Equation 4.2.4.

4.2.2.2 Synchronization Across Multiple Computers

As mentioned above, there are different approaches to drive multiple displays. One approach is to compute the images on a single PC that either directly outputs them, potentially by using multiple graphics cards, or sends the images to other PCs via network. This method does not result in a high performance and in the former case does not scale with the number of displays. For a high rendering performance, a better approach is to send only rendering commands or synchronization information over the network to distribute the rendering on a cluster, both reducing network transfer and parallelizing the rendering. The latter approach is used by the Davelib, with an instance of the same executable running on each PC.
An example is a user flying through a virtual world. Position and orientation of the navigation are copied to a synchronization buffer in the master application. After the network transfer it is read from the buffer in the client application. The Davelib internally also transmits the eye positions of a single head tracked user. Support for multiple head tracked users can be added easily.
The Davelib provides two types of network transfers. One is intended for the mentioned information like status updates, it is sent via UDP, preferably on a multicast address. It is fast but unreliable and only the last message is kept in the case that multiple messages are received at a time. For the multicast case, the clients only need to know the address of the server. The other type of network message is used for events, implemented with TCP for a reliable connection.
Currently, rendering is not explicitly synchronized. While applications can be written to synchronize the buffer swapping, waiting until image renderings on all other clients are complete, it is hard not to waste time or introduce latency from synchronization overhead. Also, especially in a CAVE, the workload and framerate of different machines may greatly differ, depending on object visibility on the different screens. In our VR setups we currently only use software that renders with high framerates. A fast rendering is very important with head tracking, as head motion otherwise leads to disturbing jumping of the content. With the high framerates, we do not notice problems without explicit synchronization. However, the latency may increase, as the rendering of a frame is not instantly started when new information is available, see Figure 3.8 for a graphical representation of the timings and latency.

4.2.2.3 Configuration

A Davelib application can render on a wide range of setups without adapting the program. A simple configuration file allows to use the

same executable, e.g. on the DAVE, the HEyewall or a normal desktop PC. The configuration mainly contains the configuration for each display, like the position of the four display corners in space. Further parameters like like network addresses or file names of calibration matrix or blend mask image can be specified, too.
The default configuration file name can be read from an environment variable, so on a configured PC, an application can be just started and automatically uses the correct parameters. Command line parameters can also be used to temporarily define or override individual settings of the configuration. String pairs of key and value are used, with helper functions for reading integer or double values. We also implemented a hierarchical definition. As an example, a value that should be the same for all computers can omit the machine name, acting as a default in case it is not defined in that deeper level of the hierarchy.
Unfortunately, there are no commonly accepted cross framework standards for setup configuration, e.g. display configuration or projector calibration. If for the first time a Davelib program should run on a new setup like a CAVE that uses display calibration, either the calibration has to be redone or an existing calibration from a different framework has to be transferred. Equally, the configuration or calibration may be automatically translated to enable usage of the same information in other software packages. As an example, after calibration of the DAVE, the configuration for Davelib applications are generated as well as the engine definition for instantreality.

```
multicastAddress = 224.245.132.123
multicastPort = 33001

// optical tracking
trackingServer.address = 10.52.10.1
trackingServer.port = 33002

/////// window on master ///////
master.windowPosX = 10
master.windowPosY = 10
master.windowWidth = 400
master.windowHeight = 300
master.hmdHFov = 60

/////// LEFT WALL ///////
dave1.lowerLeftX = -1.65
dave1.lowerLeftY = -1.65
dave1.lowerLeftZ = 0
dave1.lowerRightX = -1.65
dave1.lowerRightY = 1.65
dave1.lowerRightZ = 0
dave1.upperLeftX = -1.65
dave1.upperLeftY = -1.65
dave1.upperLeftZ = 2.47
dave1.windowMode = FULLSCREEN
dave1.cameraStereoOffset = LEFT_EYE
dave1.calibMatrix = config/LEFT.cal

dave2.lowerLeftX = -1.65
dave2.lowerLeftY = -1.65
dave2.lowerLeftZ = 0
 ...
```

Figure 4.5: An excerpt of the DAVE configuration file for programs using the Davelib. To describe the properties for a screen, only a few lines are necessary.

4.2.2.4 Window management

The window manager has the task to open one or more windowed or fullscreen OpenGL contexts, as specified by the configuration data. Since this operation largely depends on the way the windows are opened, it has to be implemented for each framework. We mainly use *OpenGLEAN* as a *GLUT* replacement. We slightly modified it to support quadbuffer contexts for the *clone* mode with quadro graphics cards and borderless windows for e.g. a side-by-side configuration. A less complete window manager for Simple DirectMedia Layer (SDL) is also implemented.
Another example is an external software opening a full screen window with native system calls. Not touching this part of the code, the values of the window configuration are ignored, reducing its portability to other setups. It runs correctly in the DAVE where a single fullscreen window is used for each PC, but does not work on the HEyeWall without further changes.
Often, 3D applications are written to render only to a single window. On a setup with multiple displays connected to a computer, the application can be started several times with different parameters. However, it is more memory conserving and more efficient to render to multiple contexts from the same application. As an example, a large 3D model only has to be only once loaded from disk and stored to main memory. This may require some more effort to port existing applications, since some initializations like texture and shader definitions need to be done per window, keeping in mind that the windows may be rendered by different graphics cards.
When rendering of multiple windows is done on the same card, textures and vertex buffers can be reused and some intermediate results like shadow maps must only be computed once per frame. It may make sense to use these advantages and start an instance for each graphics card.

4.2.3 Davelib Extensions

Useful add ons are described below, some of them require additional effort during implementation. We suggest to also support them within a Davelib program for a flexible usage on many different setups.

4.2.3.1 Linear Geometric Projector Calibration

To avoid a tedious and time consuming mechanical calibration of the projectors, they can be only roughly aligned with the image slightly overlapping the screen. An additional calibration is applied during rendering to predistort the images so that the image content is geometrically aligned as if the projector was aligned perfectly.
Raskar et al. propose a linear projector calibration [Ras00] by modifying the OpenGL projection matrix. We independently developed an almost identical solution that is used for all applications in the DAVE. The modification is realized by a simple matrix multiplication, so that Equation 4.2.1 extends to

$$P_{OpenGL} = C * P * R$$

Assuming perfect projector optics and a planar projection surface, a simple homography is sufficient to correct each of the 3D vertex

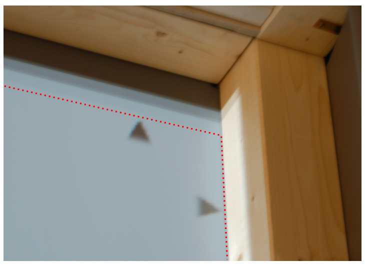


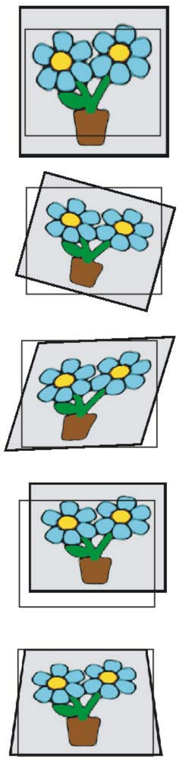
Figure 4.6: The projection slightly overlaps the screen area visible from the inside of the DAVE, indicated by a red dotted line. The current calibration is also visualized by the system background image, that is recomputed accordingly after each calibration.

positions. As straight lines stay straight after transformation, all 3D content is corrected in this way. For at least four reference points q_{s_i} the user specifies corrected points r_{s_i}. The calibration matrix

$$C = \begin{pmatrix} a & b & 0 & d \\ e & f & 0 & h \\ i & j & 1 & 0 \\ m & n & 0 & 1 \end{pmatrix}$$

is calculated such that all q_{s_i} will be projected as close as possible to the target positions r_{s_i}, minimizing the sum of the squared distances. This matrix is a combination of the matrices shown in the figure on the side. A semi-automatic calibration can be achieved by projecting a mouse cursor on the screen and clicking on physical reference positions like the corners of the target screen. To get precise physical locations of the reference points we use the corners of the projection. As a normal mouse cursor is not visible at each corner and only allows pixel precise acquisition, multiple lines at different angles are used for each corner. Their intersection defines the corner position accurately. The lines can be moved with a wireless joystick from within the DAVE. We have improved this calibration so that it is possible to interactively move a corner and instantly see the calibration result, also showing markers on each edge that should match to the respective markers on a neighboring screen.

The values a, b, d, e, f, h, m and n are numerically optimized with the steepest descent algorithm so that the sum of squared distances from projected points to their target location is minimal. Unfortunately, to correct keystoning or trapezoid distortion, the parameters for perspective division are influenced, leading to different depth results after perspective division and thus most notably different and skew clipping planes. We partially counteract possible resulting artifacts by setting the values i and j to m and n respectively, so that the near clipping plane stays the same but the far clipping plane is stronger distorted. This is motivated by the typical DAVE applications with close objects where clipping at the near plane occurs and leads to otherwise inconsistent clipping at screen borders. Raskar et al. address this problem differently [Ras00]. They set $i = j = 0$ and $k = 1 - |m| - |n|$ and thus extend the frustum to include the complete original frustum. However, the resolution of the depth buffer may not be used very well in that case, and front clipping planes are skew.



$$\begin{pmatrix} a & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} a & b & 0 & 0 \\ e & f & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & b & 0 & 0 \\ e & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & d \\ 0 & 1 & 0 & h \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} a & 0 & 0 & d \\ 0 & f & 0 & h \\ i & j & 1 & 0 \\ m & n & 0 & 1 \end{pmatrix}$$

Figure 4.7: The 4×4 matrix transformations shows which matrix values must be changed in order to correct the respective effects.

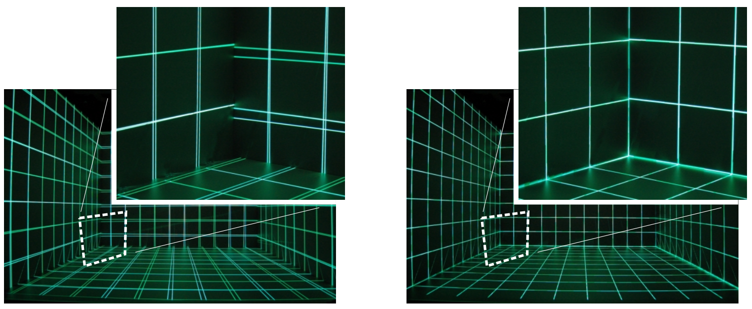


Figure 4.8: Software calibration of projectors with minimal overhead. Left: before calibration, right: after calibration.

The necessary values are stored in an ASCII file for each screen containing the 16 values. In OpenSG the same file is read, in instantreality these values are stored in a *MatrixViewModifier* in the engine definition, so that currently the engine definition must be rebuilt. This is done automatically after each calibration.

Discussion. It is helpful to place the projectors in a way to keep the changes by the calibration small, especially to avoid keystoning and thus reduce the potential problems with the depth calculations and clipping. Note that bitmap operations are not transformed by the OpenGL projection matrix and are not handled correctly. Some applications may draw a head up display in that way. Also, screen space fragment shaders that work with information from neighboring pixels may lead to slightly different results. Unfortunately, our mirrors and rear projection screens are not exactly planar. When the calibration is done to fit at the corners, other parts may be off by a few pixels. This is visible especially at the edges. With the manual calibration, this can be partially compensated.
The main advantages of this method are obvious: Only minimal code changes are necessary to apply the calibration and during rendering, no extra computation time is required. For more details, please refer to our publication [LOUF06]. If any of the mentioned issues are a problem in practice, the following approach can be used.

4.2.3.2 Non-Linear Geometric Projector Calibration

For smoothly curved screens, a mesh for a two pass approach can be generated with 2D freeform deformation tools developed in section 6.2.2.1. As curved screens, especially for rear projection, are not easy to build, and common projector optics are made for planar screens, we have not tested a calibration of such a setup yet.

4.2.3.3 Photometric and Colorimetric Calibration

Lamp differences and more importantly non-Lambertian projection surfaces and diffuse interreflections lead to visible edges in the DAVE. With multiple calibrations from different camera positions and with head tracking, this problem may be mostly compensated for. However, this is a large effort and only works for a single user in our setups.
At the moment, a very rough approximation is used in the DAVE. Only the brightness in the DAVE is regulated manually using the lamp power setting of the projectors. Interestingly, many visitors have problems to locate the physical limitations of the screens when using the DAVE for the first time. An interesting and high frequency content seems to lower the perception of the screen edges significantly.

4.2.3.4 Edge Blending

Automatic edge blending by software can be achieved in the Davelib by drawing an edge blend mask over the final image.
The main drawback of this method is that black levels of the projectors are not blended. Especially in a projector array, four overlapping images may lead to a visible increase of brightness when showing a dark image. However, the effect is not very obvious and only visible for very dark content. Currently, we do not yet support dark level adaption of the rendered images.
After having implemented the soft edge blending approach by Raskar et al. [RBY*99], we observe artifacts in the corners of the blend masks

(see Figure 4.9 below).
They use soft edge blending and use the following equation to compute the attenuation value of projector m at each pixel position (u,v)

$$A_m(u,v) = \frac{\alpha_m(m,u,v)}{\sum_i \alpha_i(m,u,v)} \tag{4.2.8}$$

with a weight function

$$\alpha_m(m,u,v) = w_i(m,u,v) * d_i(m,u,v) \tag{4.2.9}$$

where $d_i(m,u,v)$ is the distance to the closest edge and $w_i(m,u,v)$ is defined as 1 inside and as 0 outside the projection area of projector m. Finally, they use a gamma lookup table to correct for non-linear projector brightness curves.
To address the artifacts, the weighting function of equation (4.2.9) is modified. The resulting intensities will still sum up to 1 by construction of the equation (4.2.8).
Harville et al. come to the same conclusion [HCS*06] and suggest to use a multiplication of all edge distances (see below). In practice, the blending area for our setup is considerably smaller than 50% of the image. Thus, it is sufficient to only consider the closest horizontal and vertical edges e_1 and e_2, with respective distances $f_1(m,u,v)$ and $f_2(m,u,v)$. Also, we do not consider edges that should not contribute to the blending, i.e. distances to edges on the outside of the screen.
We follow the notation of [RBY*99] and present the following variants, replacing the edge distance related term d_i of the weighting function:

$$d_i^1(m,u,v) = \min(f_1(m,u,v), f_2(m,u,v)) \tag{4.2.10}$$

$$d_i^2(m,u,v) = \left(\frac{1}{\frac{1}{f_1(m,u,v)} + \frac{1}{f_2(m,u,v)}} \right)^p \tag{4.2.11}$$

$$d_i^3(m,u,v) = f_1(m,u,v) * f_2(m,u,v) \tag{4.2.12}$$

$$d_i^4(m,u,v) = (f_1(m,u,v) * f_2(m,u,v))^p \tag{4.2.13}$$

To compare the functions, the blend masks for a corner of four overlapping projectors are examined. d^1 is the original function by Raskar et al. By using the minimum edge distance, there are two regions where only the smaller value is used while the other one is completely ignored. This results in a discontinuity of the first order derivative of the weighting function, leading to visible artifacts. To avoid this, we introduce a new function d^2 with a double fraction, always respecting both values. To also produce a smooth function at the window border, we also introduce an exponent p. Trying to reproduce the attenuation pattern of physical blends in a corner with d^3, the resulting function is very similar to the one described in [HCS*06]. Again we apply the idea of adding an exponent p and get the new function d^4. With $p > 1$, the first order derivation of the weighting function also becomes continuous at the image border.
The figure below shows a detailed comparison of the aforementioned weighting functions.

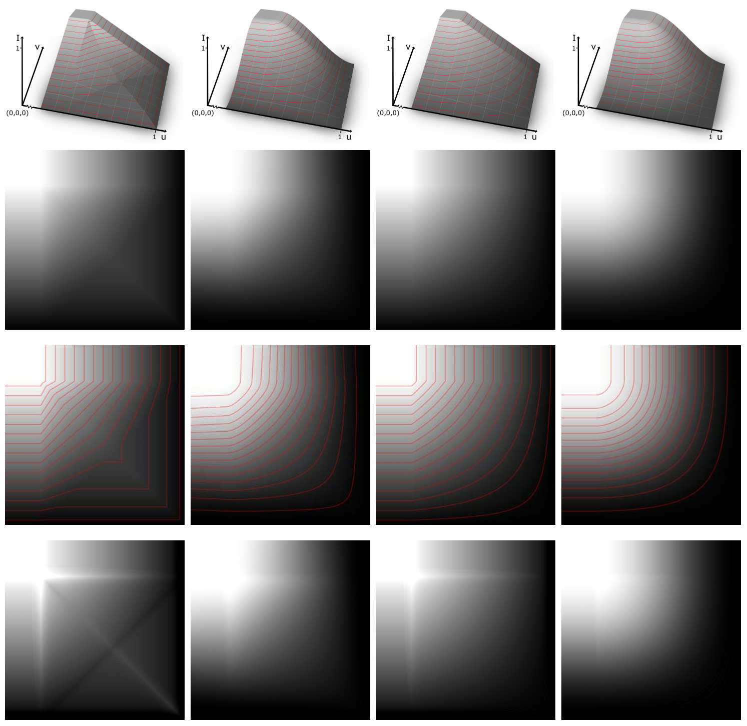


Figure 4.9: Comparison of different weighting functions. Only a crop of the bottom right corner of the mask of the top left projector is shown. The inner corners of other projectors are rotational symmetric. From left to right: d^1 by Raskar et al., d^2 with $p = 2.0$, d^3 by Harville et al., d^4 with $p = 1.5$. From top to bottom: 3D graphs showing the resulting intensities in a corner, attenuation map coded in gray values, attenuation map with contour lines of equal brightness, attenuation map with an unsharp mask (sharpening) filter applied to highlight the type and amount of artifacts.

Looking at the two bottom rows of the figure above, both d^2 and d^4 lead to the least artifacts due to their first order continuity inside and at the borders with $p > 1$. This is well visible as the iso-intensity curves in the 3^{rd} row are smooth, whereas the curves in the case of previous work (d^1 and d^3) suffer from sharp bends. As both of our new functions d^2 and d^4 seem to produce similar results, we further analyze their differences as well as the influence of p, shown in Figures 4.10 and 4.11. Increasing p leads to smoother iso-intensity curves but also narrows the transition width. However, a large transition width is desired. By experiment, we determine a good trade off for p for both functions. Finally, we choose d^4 with $p = 1.5$ because of its slightly visually smoother appearance at a similarly broad transition compared to d^2.

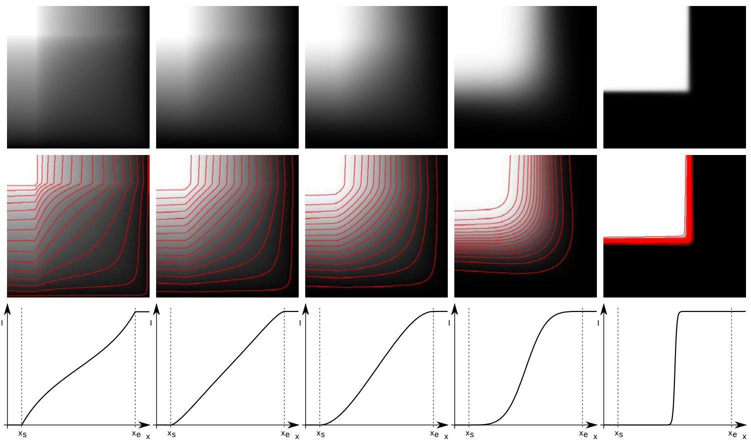


Figure 4.10: d^2 with varying exponents. From left to right: $p = 1.0$, $p = 1.5$, $p = 2.0$, $p = 4.0$, $p = 40.0$. The graph in the last row shows the edge transition between two projectors. By experiment, we found that $p = 2.0$ leads to good results (center column).

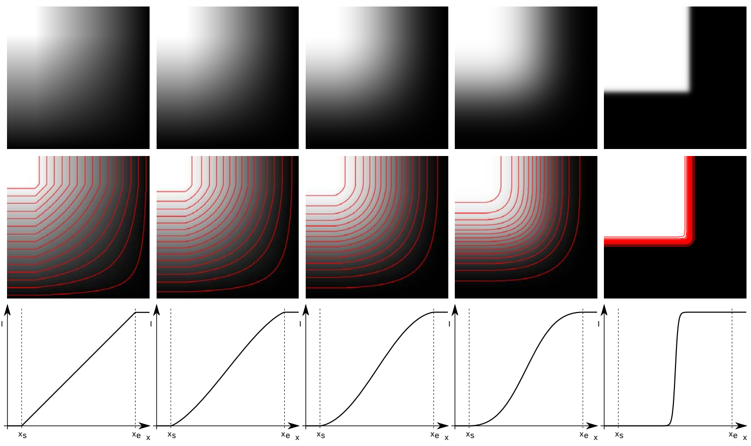


Figure 4.11: d^4 with varying exponents. From left to right: $p = 1.0$, $p = 1.25$, $p = 1.5$, $p = 2.0$, $p = 20.0$. By experiment, we found that $p = 1.5$ leads to good results (center column).

To compute the blend mask for each projector, the input is given as geometrical calibration file in an ASCII format with 16 float values, as descibed in section 4.2.3.1. The photometric calibration file contains only a single gamma value for now. The same files are used for the freqency split display (see section 4.3). The mask generation additionally needs the following input parameters: output file name, native width and height of the projector, and a flag for each of the four edges if they should be considered for blending, i.e. whether the edge is on the inside of the screen.
To get a pixel precise value, instead of forward warping the mask like Raskar et al., we compute the inverse transformation for each mask pixel into the screen space.
To be compatible with all frameworks we use, we write out both just the mask as a PGM file as well as a PNG file with the respective alpha value and the rgb color components set to black.

Figure 4.12: A close up view of a setup with only roughly mechanically aligned projectors. No blending, d^1 by Raskar et al. and d^4 with $p = 1.5$ (from left to right).

While we get inferior results than sophisticated camera based methods for full photometric calibration, we avoid their complexity and effort with a simple and easy to implement approach. Considering the intended use for only high frequency content (see section 4.3), this method is sufficient. Even with fully calibrated projector arrays, when using non-Lambertian display surfaces, the calibration is only perfect for the point of view of the camera. Soft edge blending helps to hide the transitions. Compared to the commonly used soft edge blending, we obtain blend masks with considerably smoother results at corners. Thus, in practice we achieve a slight reduction in perceived corner artifacts. Furthermore, only a small modification of the algorithm or program code is necessary.
For more details, please refer to our publication [LF11a].

4.2.3.5 IO Device Library

As indicated above, almost all VR setups have different input devices and many new devices are produced with a variety of input options. Writing an application so that it can easily be modified to work with a different device greatly increases the number of setups it will run on. Ideally, recompilation of the whole program is not necessary, using plugins or scripts.
Scripting is a powerful and flexible solution, especially useful when developing a cross platform application. A commonly used scripting language in the game industry is *Lua* [IdFC07], a fast, lightweight and free scripting language. It can easily be embedded in a Davelib application. Our new method of handling input devices for application control is to use Lua scripting. Also, additional output devices may be controlled, such as haptic and olfactory devices or a system for spatial sound rendering. As this part of the Davelib has little to do with image rendering, it is described in more detail in section 5.4.1.
Even though Lua has a small footprint, embedding a scripting language into a program, especially in an external program that is to be ported, does not fit well to the minimalistic idea of the Davelib. Nevertheless we consider it a good approach to become less setup dependent.
Currently, both a Davelib version with and one version without Lua exist. If in future, the input device library with Lua will be an integral part of the Davelib, it will also make sense to handle the configuration with Lua.

4.2.3.6 DaveTracking

This interface provides a unified access to data from 6 DoF tracking devices. Head tracking data is used internally by the Davelib. Only one interface is implemented so far, for our own tracking system (see section 3.1). It makes sense to also implement support for *OpenTracker* and for the IO device library. In the latter case, a separate program can use *OpenTracker* as an additional abstraction layer and forward the data to the Lua script in a unified way. With that method, Davelib applications do not have the large number of additional dependencies of *OpenTracker* when linking.

4.2.3.7 Future Extensions

The Davelib does currently not provide help for the following features that may be supported in future versions. Nevertheless, an application can of course implement such functionality on its own.

Non-Linear Geometric Calibration. Non-planar screens are not supported. A good way for projecting on smoothly curved surfaces is to render the necessary view to a frame buffer object in a first pass and render this as a texture on a warped mesh in a second pass (see section 4.2.3.2). In the second pass, a colorimetric calibration and a tone mapping for HDR image content can be easily applied as well.

Spatial Sound. Spatial sound synthesis for auralization of soundscape can be realized e.g. with wireless headphones. The relative position of the sound sources to the ears is required. A supporting function may be added, using the already available head tracking.

2D GUI. In previous work, many attempts were presented trying to display a 2D GUI within a VE. However, VR pointing devices and a 3D rendering of a flat menu are often not the best combination. For complex menus, sliders, lists etc., an extra window with a standard 2D GUI may be a better solution. It can be displayed on the local or a remote machine, e.g. a wireless handheld or tablet device in the DAVE. In other setups, a 2D GUI window on top of the 3D context may be used, e.g. at the HEyeWall. Such a GUI can be realized e.g. by using a browser and web technologies or *XAML*. The information can be communicated via the IO device library.

4.2.4 Testing and Simulation.

Rooms with VR equipment often have lighting restrictions and usually receive no or very little day light. For comfort it is often preferable during application development to work for most of the time in the office. While most aspects can be tested and problems debugged there, experiments on the real hardware are also necessary. This is mostly related to user interfaces and stereoscopic effects that cannot easily be simulated. Also, multi touch simulators can only be used to some extend. In rare cases, programs are especially hard to debug because the error only occurs on the VR equipment and not in the test or simulation. As this is a rare case, debugging infrastructure and experience is often poor on the actual VR hardware, i.e. on render clients a compiler may not even be installed and debugging via console output or log files is the quickest ad hoc solution but still uncomfortable. It is advisable to install the same software packages for development for the respective operating system on all machines, i.e. on the render clients, the server and the office PCs used for development.

4.2.5 Rendering Aspects

This part is very interesting for the practical usage of the Davelib, both for porting existing applications as well as guiding information for the development of new applications.

4.2.5.1 Necessary Modifications to Port Existing Applications

Applications that were not written with stereoscopic rendering in mind may not use a correct depth for all objects. An example that we observed in multiple applications is a skybox as background with a relatively small size, e.g. a radius of one meter, centered at the view position. As the depth buffer is disabled to render the skybox as the first part of the image rendering in these applications, it will look good on a normal monitor. With stereoscopic rendering however, scene objects in the distance appear further away than the sky, leading to an irritating impression.
Screen space processing effects are often written with the assumption that the projection matrix does not rotate the camera. Also, such effects may be interrupted at the screen borders. As an example, a blooming filter or point splatting in a CAVE are likely to look wrong at the edges. Such errors are partially compensated by our setup with slightly cropped images of the larger projected area, as an additional border is available.
Advanced techniques that render with screen resolution in mind, like state of the art shadow mapping algorithms, may not lead to optimal results and should be adapted in that case.

4.2.5.2 Best Practice for Implementation

Some simplifications and approximations do not work very well in a CAVE. An example are plants that may be approximate by very few textured quads. Due to the better depth perception, such tricks may be more obvious. A solution may be to better approximate the geometry by using more textured quads [DDSD03]

Stereoscopic Rendering. 3D graphics with uniform colors may not be well suited for stereoscopic rendering. If a large area of the screen has a rather uniform color, e.g. when the object is close, the stereoscopic depth perception of the model may be lost and instead, the screen is visible. It greatly helps to texture objects or at least provide realistic lighting.
For stereoscopic rendering, a 2D cursor or crosshair appears at the screen distance and may be perceived at a very different depth than the 3D object distance under the cursor. Better is to show a cursor just in front of the objects, either as a quad or as if projected.
Reflective materials or other effects using raycasting, like parallax occlusion mapping, will need a special adaption to take the head tracking and stereo offset into account, see section 6.3.6 for details.
An interesting related study showed that strong highlights with a correct reflection may lead to more eye strain than an incorrect

quicker approximation, where the highlights appear on the object surface [McD10].
For stereo displays with a limited field of view, like for fishtank VR setups, objects coming out of the screen, i.e. towards the viewer, should be carefully used. Clipping of such objects by the viewport should be avoided. Especially clipping by the left and right screen borders may lead to a loss of stereo fusion.

Particle Rendering. Rendering of particles may be approximated, not taking perspective distortion into account. A sphere may be approximated with a circular shape on the screen. This leads to visible artifacts especially at the edges of a CAVE. We developed a pair of shaders to quickly render a sphere for each specified point. The vertex shader generates coordinates for a quad tightly enclosing the shape, the fragment shader is used to fill the respective pixels.

Setup. For multiple windows rendered by a single PC, either two instances of the same program can be started, or a single program can render both images. The first approach is easier to implement but may not use system resources very well. The latter method can reuse memory and sometimes intermediate results like shadow maps if the windows are rendered on the same graphics card. Note that some systems like *SLI* by NVIDIA or *Crossfire* by ATI allow to interconnect multiple graphics boards of a PC and may allow e.g. to share texture memory.

Synchronization. In some applications like simulations, pseudo random numbers are used. As such numbers will in general be different on each PC, a Davelib master should compute and distribute them to the clients. Even better, the simulation state after all computations should be synchronized, so that is ensured that each program is in the same state. This means, that absolute values should be send instead of differential changes, relying on deterministic behavior. If however, the simulation result is a large amount of data that cannot easily be transmitted over the network for each frame, some compromise may be found. The same simulation could run on all machines but parts of the results could be synchronized each frame, hoping that if an inconsistency occurs, it will eventually be recovered from.

Camera Setup. By using an absolute coordinate system for our setups with the z axis pointing up, all real world scenes following that convention will also be oriented correctly. However, if the content is more of a 2D nature, like photos for the multi touch setups, it is not practical to have a correct coordinate system for a table setup, but when rendering the scene on the HEyeWall it is visible from the side, i.e. so that the photos only show up as lines. Some sort of 2D projection or camera setup may be useful, such as an orthographic camera for planar screens and a cylindrical projection for the DAVE (see Figure 6.75). We have not implemented a general solution yet, but use a configurable default camera position and orientation as an acceptable solution.

Development. During development on a desktop PC, a configuration file can be used e.g. to open two windows with different views.

Another option is to start two instances, one as master and one as client, to test the network synchronization.

Porting Effort. In summary, to port an application, larger modifications that require more effort are usually to add support for multiple windows, non-planar displays (e.g. for a CAVE), the network synchronization across multiple PCs, interaction, shader tricks and post processing effects.

4.2.6 Projector Calibration Procedure

With the old projectors for the DAVE, we had eight individual calibration files. The new projectors have a single lens and thus, only four calibration files are needed. In Linux we run the same calibration program four times on the server, each forwarding the display to the left eye projector image. Each application reacts to a different gamepad command, specified by a command line parameter. We also run four times a program displaying a black image, that is forwarded to the right eye projector images. Unfortunately, this solution is not platform independent. In future, we will adapt the solution as implemented for the HEyeWall.
After the projector calibration is complete, the desktop background images are updated, using the calibration. Small arrows at the edges show to the operator quickly, whether the projectors are still calibrated. Also, the engine definition file for instantreality is rewritten to include the new calibration values.
For the HEyeWall, the calibration also allows to set a maximum brightness and a gamma factor. There, the soft edge blend masks are recomputed after the calibration.

4.2.7 Conclusion

The Davelib, is a minimalistic open source library providing functions to port applications with relatively little effort to a range of VR environments. A small set of external dependencies helps to avoid problems during compiling, linking and long term maintenance. The latest change of the Davelib is the flexible IO device handling concept via scripting. For small projects that do not require a sophisticated scene graph or complex frameworks, or for testing new features and algorithms, it can also be used to quickly develop new VR software.
Many 3D applications are written with only desktop setups in mind. The Davelib helps our mission to promote VR and convince people to write software that also runs in VR environments.
The ideal situation is that a properly written application can be distributed as executable and can be run in any VE without further modifications. This is achieved for the display of interactive 3D graphics with a perspective camera. A possible partial solution is shown for the interaction.

4.3 FREQUENCY SPLIT DISPLAY

The idea of the HEyeWall is to provide a large, seamless, high resolution interactive wall, allowing multiple users to view and manipulate 3D data. The technology should scale, allowing a larger display by adding more components, and interaction should not require special input devices, allowing a simultaneous multi user collaboration.

These requirements leave a rear projection setup as the only choice. However, transitions are very visible on these setups (see images below and on the side). A sophisticated camera based calibration that takes several hours on the HEyeWall in Darmstadt can mostly solve the problems, but the dynamic range is reduced and the tiles are still noticeable especially for low frequency content such as plain colors. To address this problem, the idea for a possible successor is explained here. It is an attempt to build a large high resolution rear projection interactive display without visible artifacts from tiled projectors. As a proof of concept, a smaller prototype was built, the HEyeWall in Graz.

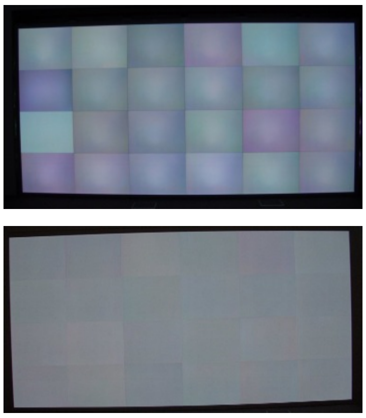


Figure 4.13: The HEyeWall at IGD Darmstadt before (top) and after calibration (bottom).

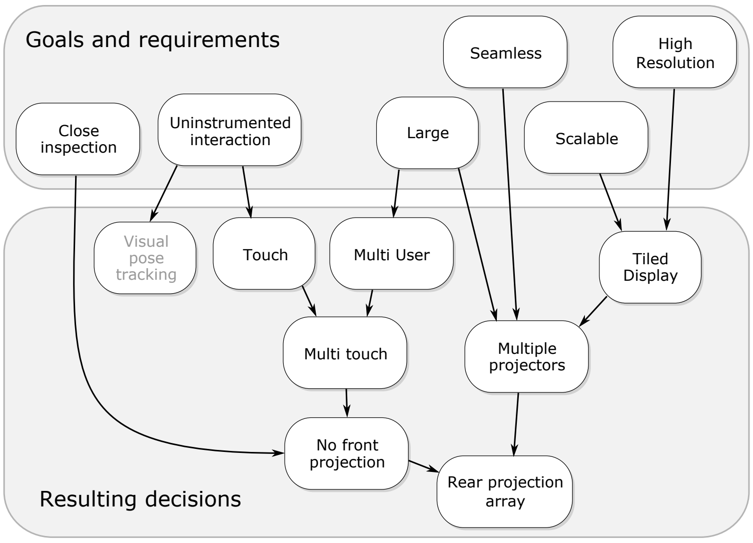


Figure 4.14: Visualization of reasons and decisions leading to the choice of technologies for the HEyeWall in Graz.

4.3.1 Related Work and Motivation

Tiled Displays. The maximum resolution of todays commercially available projectors is $4k \times 2k$ pixels. Such projectors are produced for cinemas and are very expensive. To increase the resolution or achieve a more affordable setup, tiled displays must be used.

Projection vs. LCDs. LCDs provide a high contrast and have a low price. However, bezels lead to a visible black grid between the displays. With the current technology, a minimum gap of 6mm to 8mm is necessary. Smaller gaps can be achieved with small projection units assembled into a self contained box each with a screen at the front, such as the Christie MicroTiles^{TM} LED projectors. Internal color calibration is used to minimize colorimetric differences. However, a 1.3mm visible gap exists and they are much more expensive than LCDs. Only free standing projectors are able to provide a truly seamless display.

Front Projection vs. Rear Projection. For direct inspection and interaction, users should be able to go close to the screen. Interactive front projection screens suffer from shadows cast by the users. With multiple projectors, some shadows may be eliminated [SCS01]. However, at least twice the number of projectors is necessary that must also support adjustable lens shift, increasing the cost. Also, for a large screen and many users, not all shadows can be avoided. Ultra short throw front projections can be used for an array with one or two rows (or columns) but further rows cannot be realized. Rear projection is well suited for such an interactive display. We combined the display with a large multi touch interface, as explained in section 3.2.

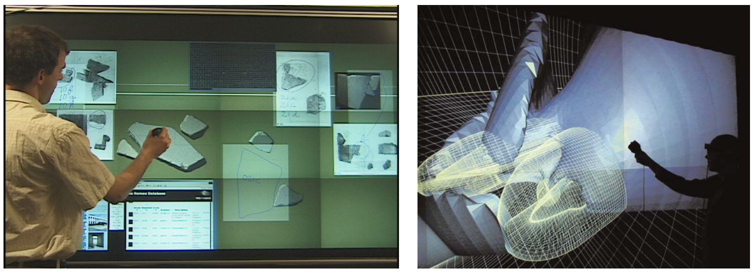


Figure 4.15: Left: Stanford Interactive Mural. Photo courtesy of HCI Group, Stanford University. Right: Display Wall at UC Davis VR Lab. Photo courtesy of Oliver Staadt, UC Davis and University of Rostock. Both tiled screens have visible tiling artifacts.

Ni et al. present further large high resolution displays in a survey from 2006 [NSS*06].

4.3.2 Frequency Split Idea

Considering tiled rear projections, we observe that the tile borders are visible for low resolution content. As an example, a plain bright color will show them best, while a region with many details makes the transition less visible.
From that observation we derive the idea to split the image into two components. The low frequencies of the image are displayed seamlessly by a single large low resolution projector spanning the whole screen. The missing high frequencies are filled up by small tiled projectors, thus adding the fine details. For a white thin line on a black background, the large projector shows black and the tiled projectors show the white line. For the inverted case however, the tiled projectors cannot subtract a thin black line from a projected white image. Instead, the large projector displays white with a thick black line and the tiled projectors fill the missing white, leaving only the thin black line.

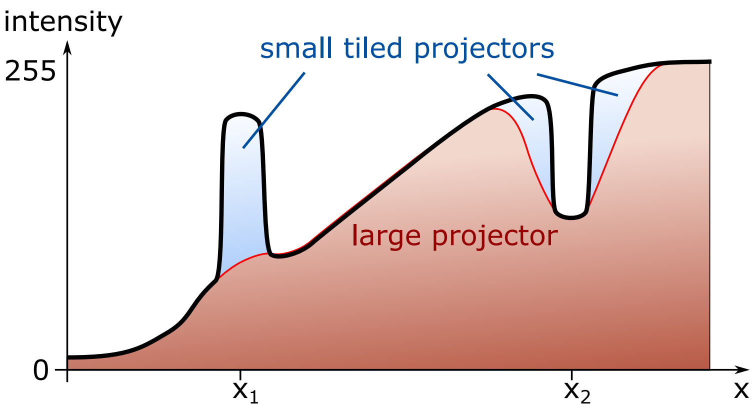


Figure 4.16: Frequency split 1D example signals. The black line shows the intensity along a line on the display, the colors show the contributions of the projectors. A small bright spot at x_1 can easily be added by the high resolution tiled projectors. A small dark spot at x_2 however requires more complex processing.

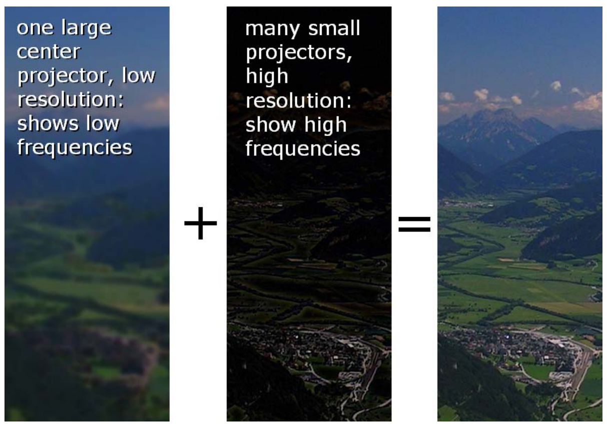

one large center projector, low resolution: shows low frequencies
+
many small projectors, high resolution: show high frequencies
=

Figure 4.17: Frequency split example images, showing the contribution of each frequency range, adding up to the final image.

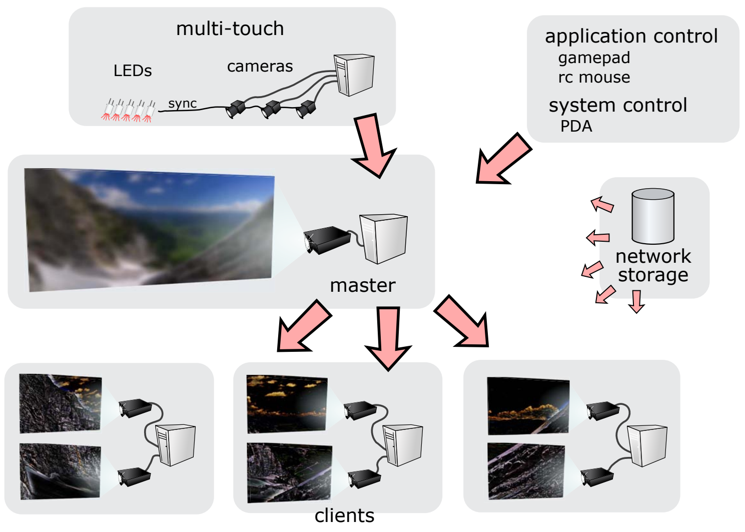


Figure 4.18: The HEyeWall in Graz is a setup to realize the idea of splitting image frequencies over multiple rear projectors. A multi touch interface is realized for input.

4.3.2.1 Soft Filter

Without modification, the large projector shows the image signal with sharp pixel boundaries, i.e. pixels are displayed as sharp big squares and a black grid is in between. This high frequency signal is not desired for the low frequency image. In addition, the pixels of the large projector are not aligned to the tiled projector pixels. A similar problem exists in HDR projection, where the light of a projector is further modulated by an LC panel [PS05]. Pavlovych et al. avoid moiré and aliasing artifacts by slightly defocusing the projector. With our large projector this is not advantageous, as it contains four lamps. Their image is not aligned any more when the projector is defocused (see image below). Furthermore, each color channel has a different image size and the black grid vanishes only with a stronger amount of defocus. A modification of the light engine to avoid these problems is impractical. Instead of defocussing, an optical soft filter is used. A clear spray paint on a transparent sheet of plastic works quite well, however the total contrast is lowered notably. A high quality soft filter may have a less negative impact, which has not been tested yet.

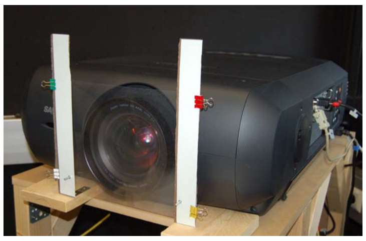


Figure 4.19: A cheap optical soft filter is mounted in front of the large projector. It is a transparent foil with clear transparent spray paint.

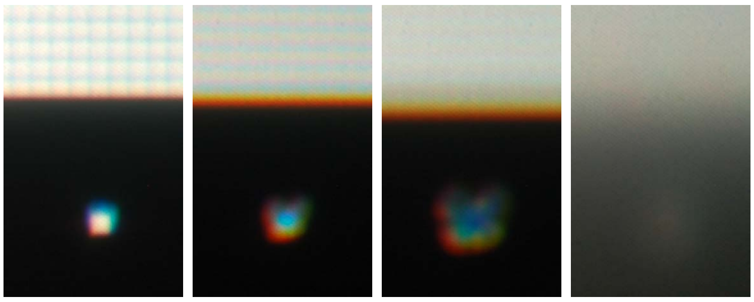


Figure 4.20: Close up of a test image of the large projector. From left to right: In focus, a black pixel grid is visible. With a slight defocus, still a horizontal grid is visible. A strong defocus mostly hides the grid but leads to an unsuited pixel image. An optical soft filter solves the problems. Here, the employed filter is stronger than necessary and also reduces the total contrast notably.

4.3.2.2 Photometric Calibration and Issues

Like for soft edge blending, when adding light intensities from overlapping projectors, their intensity transfer functions must been taken into account. However, transforming both image signals to linear intensities to split up the signal is not the goal. Instead, the low frequency signal should remain the same, as otherwise the tiled projectors would also have to show a part of the low frequency images, which we want to avoid by design. This means that only the high frequency image must take the transfer functions into account.

From the high resolution image H to be displayed, an eroded image E is computed. The radius of the structuring element should be set to the perceived radius of a blurred pixel by the optical soft filter, in order to get a large enough dark area and not to waste computation time. The intensities for the large projector I_l and for the tiled projectors I_t are computed as

$$I_l(x,y) = e(H) I_t(x,y) = t_t^{-1}(t_l(H - e(H))) \tag{4.3.1}$$

with the intensity transfer functions t_l of the large projector and t_t of the tiled projector at the image position (x,y). The intensity transfer functions are roughly approximated by a gamma curve and a maximum brightness value to reduce the intensity of the tiled projectors in order to match the large projector maximum intensity.

4.3.2.3 Using Perceptual Thresholds

The human eye can perceive high local contrasts only to a certain degree. Scattering in the eye reduces the contrast. In addition, relative brightness differences are much easier to see than absolute ones. Considering a thin black line on a white background, running over a transition of two tiled projectors, they must fill in a lot of the white light and the transition may become visible. To better hide the transition, the line can be shown as dark gray, i.e. a little brighter. Now the large projector can display a higher percentage of the white next to the line, making the transition less visible. We have not tested this idea in practice yet.
Following Equation 4.3.1, the PC computing the image for the large projector must compute E with the full resolution of all tiled projectors and then downscale the image to the native resolution of the large projector. This is time consuming and becomes even worse when building a wall with a higher resolution. An obvious optimization is to compute an approximation of the low resolution image with a much lower image size. A visible error results for fine dark image details. Standard anti aliasing for textures or in screen space averages the fine dark with brighter surrounding structures, leading to a brighter region. This error reduces the local contrast but may acceptable, as discussed above. Approximating the image H with L at a lower image dimension, the computation for the image of the large projector is much faster. The PCs computing the image content for the high resolution tiled projectors still need to render the high resolution images, but only in a subregion. For them, the low resolution image L is approximated by downsampling their high resolution image H to A. This is an approximation, because the anti aliasing method is different and the pixel grid is different.
With these approximations, Equation 4.3.1 is modified to

$$I_l(x,y) = e(L)I_t(x,y) = t_t^{-1}(t_l(H - e(A))) \qquad (4.3.2)$$

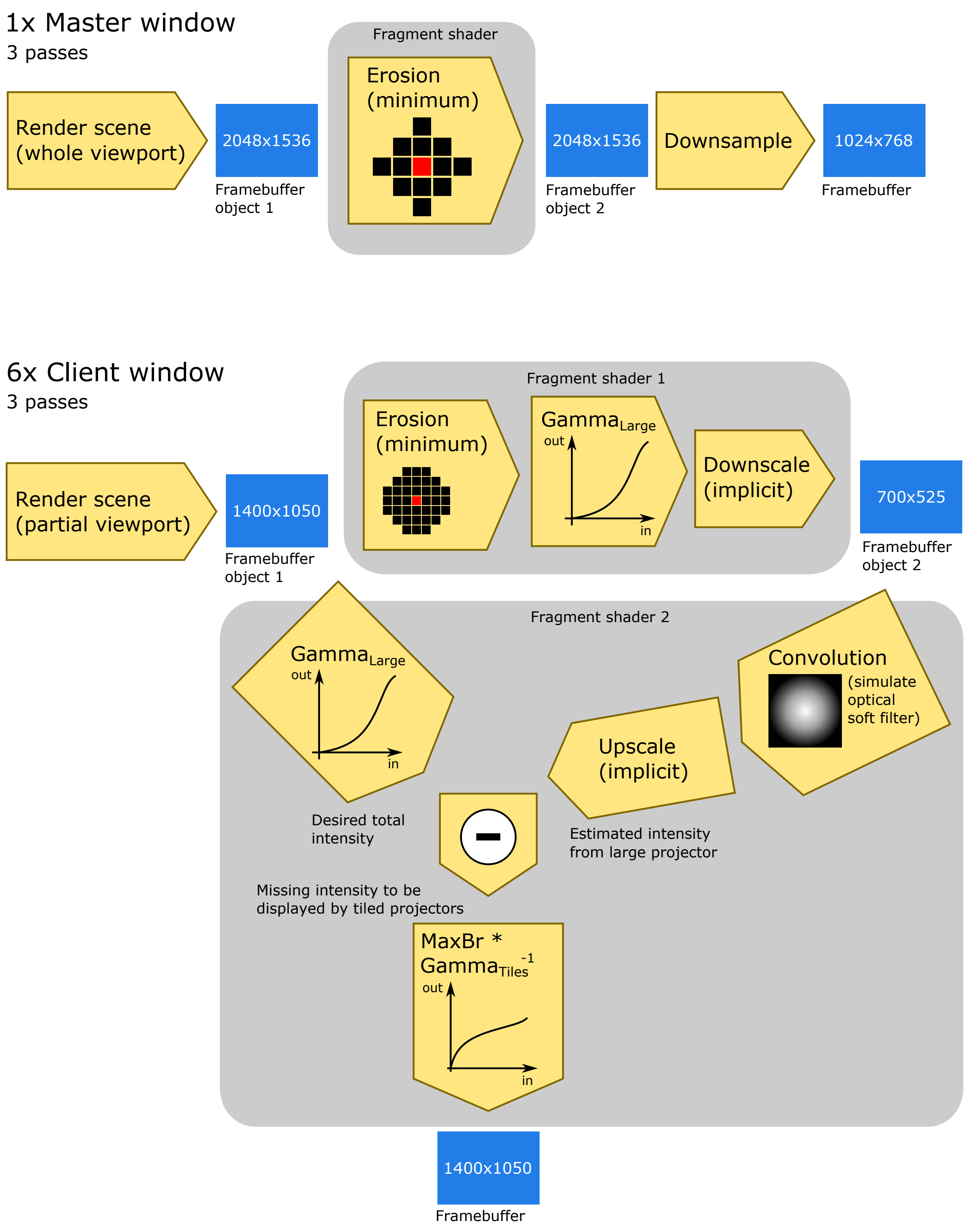


Figure 4.21: Implementation details of the frequency split rendering. The master renders the image for the large projector in the double resolution. The image is eroded and finally downsampled. The client windows for the tiled projectors render their part of the image in full resolution. Fragment shader 1 and the top right branch of fragment shader 2 compute the estimated light intensity contributed by the large projector. The top left branch in fragment shader 2 computes the desired light intensity. The difference is to be displayed by the tiled projectors. Here, the intensity transfer functions are approximated by a gamma curve and a constant *MaxBr* for the brightness compensation of the tiled projectors.

Results are shown in the images below for a difficult test scene.

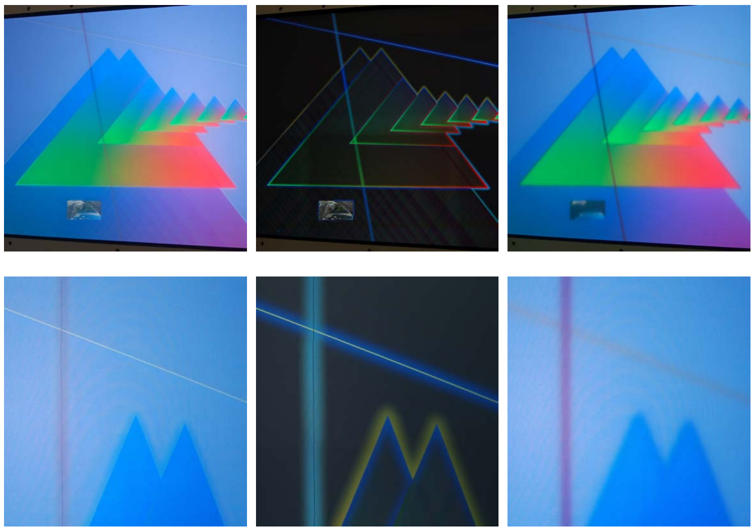


Figure 4.22: A difficult test scene (top row) and a close-up (bottom row). The left column shows the result, the high frequencies are in the center column and the low frequencies in the right column. Here, the calibration is not perfect. The thin bright line turns out good, the dark line has an unwanted large dark halo around it. Here, the 8 bit depth of each channel of the color buffer is too limited and leads to Mach band effects at wide gradients.

Update Rate. In our experimental application, the master renders with 60 Hz, limited by the refresh rate. The clients however only achieve 10 Hz. We assume that the bottleneck is the blur pass which performs a 32×32 pixel convolution for each output pixel. As with the optical filter the convolution is separable, two 1D passes can be used. This simple optimization will lead to a major speed improvement.

4.3.3 Rendering Aspects

In addition to most aspects already mentioned in section 4.2.5, additional issues arise. Synchronization is more important, as rendering times may differ and only the clients have an additional latency caused by the network synchronization of the scene updates.
With the above approximations, the pixel sizes differ greatly and the widths of OpenGL points or lines must be adapted. When rendering the image for the large projector, the line width should be about a quarter of the normal width. For thin lines this results in a lower opacity and thus a lower contrast.

4.3.4 Calibration Procedures

4.3.4.1 Projector Calibration

We implemented a projector calibration as a Davelib application. Specific to the HEyeWall is a grid for the master projector, that once calibrated is used for defining the corners of the tiled projectors. A preview of the influence of gamma and maximum brightness values is available, allowing a rough manual photometric calibration. For the gamma value, test patterns are displayed. Again, after calibration, the engine definition file is rewritten for instantreality and the soft edge blend masks are recomputed.

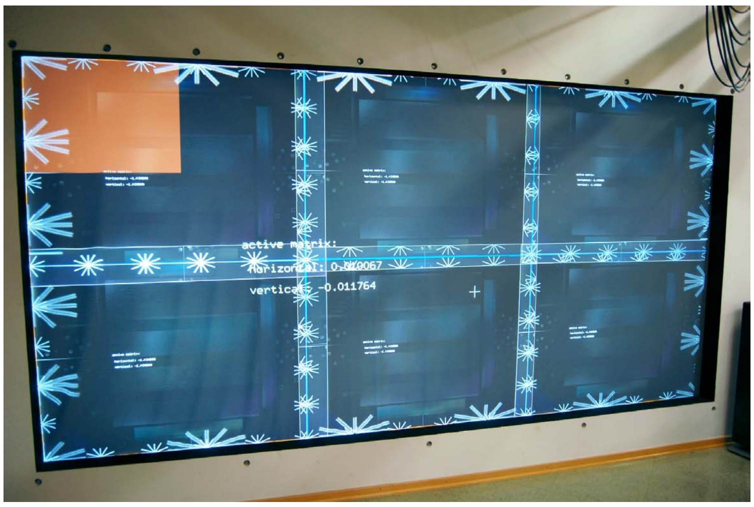


Figure 4.23: Manual geometric calibration of the six high resolution tiled projectors plus one large projector.

Photometric Projector Calibration. To achieve better results, the idea was to obtain a intensity transfer function of each projector by a set of differently exposed photos of a calibration pattern (see side image). The photos are taken with long exposure times in order to avoid color distortions of the DLP projectors that display each color one after another. A manual white balance is selected. To get linear intensity values, we use *qtpfsgui* that automatically estimates camera response curves from the provided set of photos and outputs a linear HDR image. However, due to hotspot problems described below, the approach is flawed.

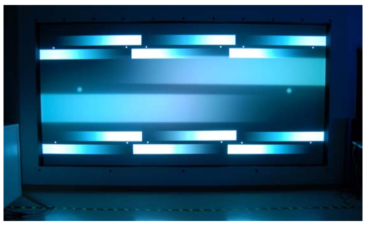


Figure 4.24: Our first idea for a calibration pattern to calibrate intensity transfer functions of the projectors.

Screen Gain and Hotspots. Even though a low gain screen is used, especially the wide angle projection of the large projector is prone to problems with hotspots.

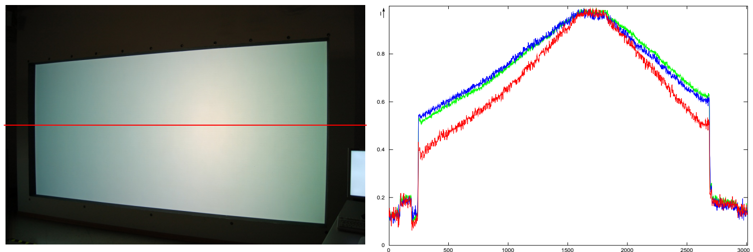


Figure 4.26: Just the large projector showing plain white. A serious hotspot exists, even with the low gain screen. As the intensity changes gradually, it is not very distracting for a human observer. In contrast, for splitting up light contributions between projectors with hotspots at different positions, the issue is severe. The intensities along the red line are plotted on the right side. For this purpose, multiple exposures were used to generate a linear HDR image with an autocalibrated camera response curve. An intensity difference of about 50% is clearly visible, red is attenuated most for large angles.

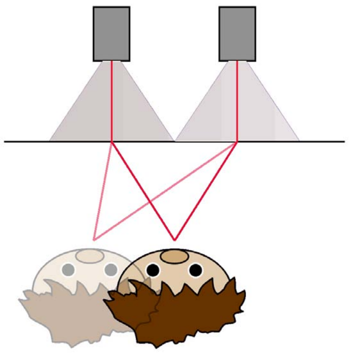


Figure 4.25: The major problem of rear projection is the brightness changing upon the viewpoint.

Recovery of the Projector Intensity Transfer Functions. A better method may be to temporarily install a diffuse front projection screen like white paper on the projector side of the permanent screen, taking the photos from the projector side. We have not tested this idea yet. Unfortunately, the problems with view dependent intensities will still exist.

4.3.4.2 Multi Touch Calibration

The calibration for the multi touch input requires a geometrically calibrated display as just described. The procedure for the calibration is described in section 3.2.3.

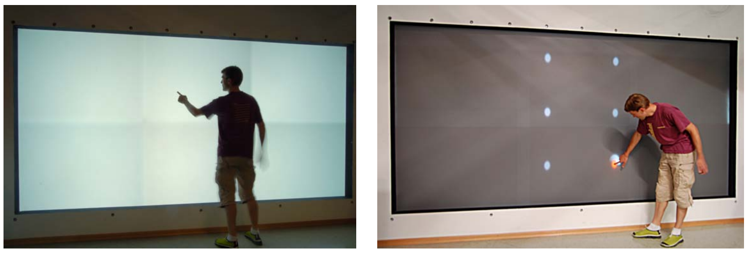


Figure 4.27: Manual multi touch calibration steps for the HEyeWall. 1. All projectors show a white image and the background images are captured. The operator taps on a few random locations so that moving projector reflections can be taken into account. 2. The operator taps on the center of each of the six circles for all three cameras.

4.3.5 Discussion

Although the idea to split up the frequencies is good, some drawbacks remain. In favor of a high performance scalable system, the image computation is split up to several PCs. The effort of computing and splitting the video signals adds some complexity to the rendering systems. But more importantly, the low frequency projector should have a similar brightness to the sum of all tiled projectors. In our case, the low frequency projector is huge and costs the same as the six tiled projectors but the effective brightness is only half of the tiles. The frequency split system has the same brightness as the low frequency projector alone, while adding the black level of the tiled projectors, thus decreasing the contrast of the image. However, some brightness of the small projectors may be added, at the cost of seam visibility. Another option is to simply display the same normal image with both sets of projectors, resulting in a partially blurry image with lower local contrast but maximum brightness. As in practice no Lambertian rear projection screens exists, the brightness can only be computed for a single tracked user. Without tracking, hot spots lead to different brightness and thus image artifacts.
The figure below shows a comparison of different options for image generation. In the top left image, all projectors show the image, with the soft filter used for the large projector, leading to a bright image with low contrast. The left center image shows only the tiled array with a bright high contrast image with a high resolution, but with a transition artifact visible in the blue sky. The bottom left shows the image of the large projector with a high contrast but a low resolution.

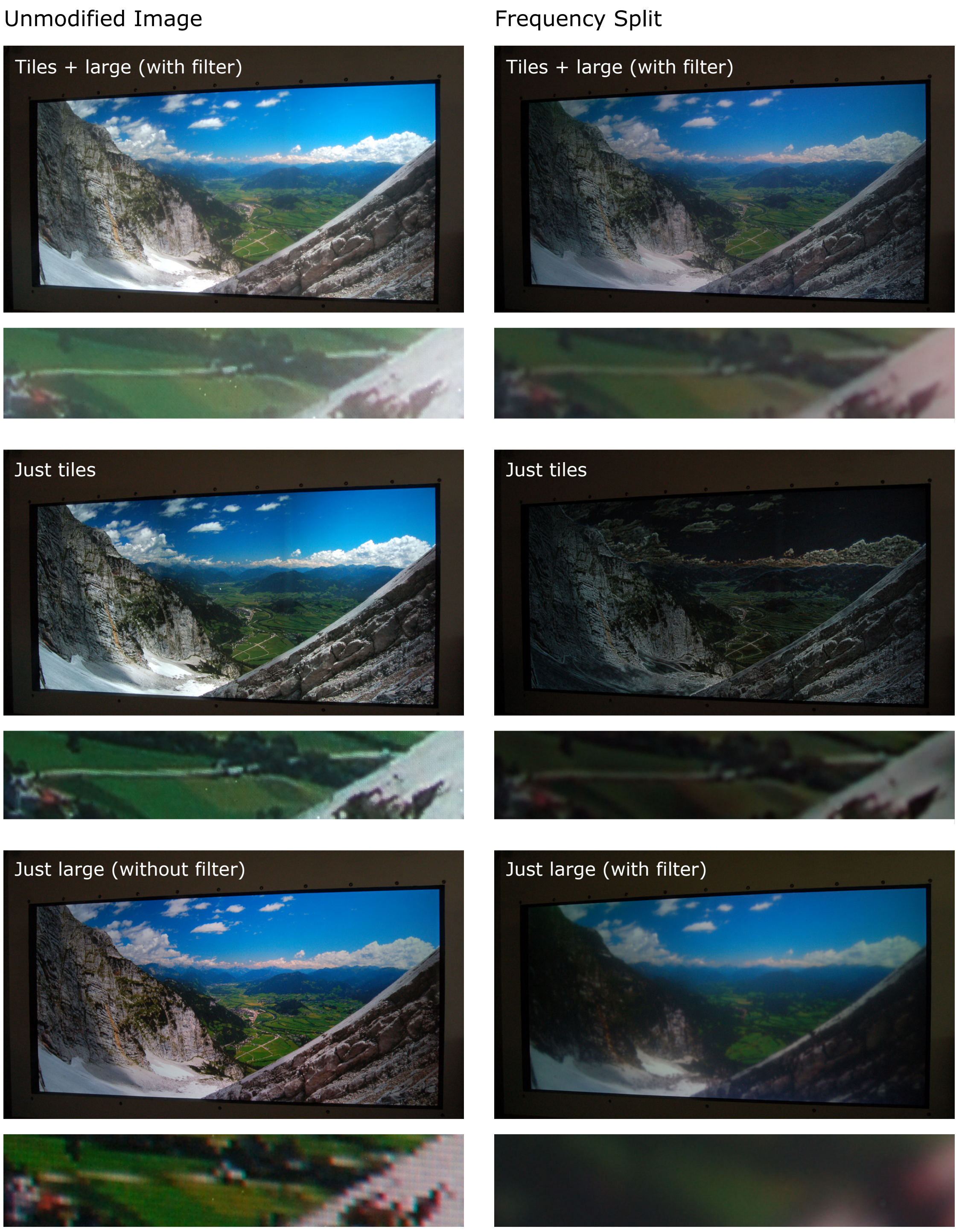

Unmodified Image
Frequency Split
Tiles + large (with filter)
Tiles + large (with filter)
Just tiles
Just tiles
Just large (without filter)
Just large (with filter)

Figure 4.28: Results and comparison with an interactive rendering of a texture, each showing the complete image and a zoomed in region. The left column shows the projection of the unmodified content image. Note that soft edge blending is used for the tiled projectors. The right column shows the frequency split result and its components from the tiled and large projector.

The top right shows the result with the frequency split approach with a high resolution and no visible artifacts from tiling. However, the overall image brightness and contrast is reduced. The image components are illustrated below.

5 User Interaction

Computers with different form factors than a desktop PC are often less suited for input methods that were developed for desktop PCs. As an example, keyboard and mouse are usually not the best choice for small handheld devices, that are often used away from a desk. Instead, recent hardware often provides touch screens. The same is true for virtual environments. In most cases, text input is not important and also, there is no desk for a standard mouse to sit on. Thus, other interaction methods are necessary. The field of Human Computer Interfaces (HCI) is dedicated to improve such interactions. Here, especially the interaction with the DAVE, the HEyeWall and the touch screen setups are discussed.

In many of our VR applications, the most common task is navigation. It is composed of wayfinding, the cognitive part, and travel, the motor part. Direct and indirect travel techniques are discussed. Additionally, selection and object manipulation may be of interest. Especially, 2D multi touch screen interaction is investigated in that regard. At first sight, the task of moving the camera or the objects may seem trivial. However, many details are important for intuitive behavior and usability.

Travel and object manipulation result in the change of position and orientation. This is usually implemented by modification of a transformation matrix in the scenegraph. They can be realized with the same metaphor or interface. However, their meaning is fundamentally different to the user and it is often more convenient to use a specialized interaction method for each task. The software interfacing of IO devices, GUIs for immersive VEs and system control are also discussed in this chapter.

As mentioned before, a wide variety of devices exists, but only the common ones have widely accepted standardized software interfaces. Even just to write a program that can control the exposure and read images from a normal webcam requires a lot of effort, as different driver types and operating systems exist. Devices that are not as common may need special software development. Examples are the now more and more popular multi touch screens or the Kinect sensor. Even though a mapping from those devices to a mouse exists, their potentials can hardly be used without a special adaption of each software.
As diverse as the hardware setups and rendering systems are, own implementations and adaptions to the individual needs are implemented in many VR labs, also trying to innovate. There is no such thing as a standard device for Virtual Reality. Even for touch screens, their technology still dictates how to use them best. This is as inconvenient for developers as it is for the users.

5.1 INTERACTION IN THE DAVE

Many interaction techniques for VEs used today were already implemented surprisingly early. A good example is the work by Vickers from 1972, showing a complete 3D wireframe modeling system in a VR setup [Vic72]. Much later, but still already in 1993, Fairchild discusses user interfaces for navigation and object manipulation [FLL*93].
A good overview on most topics on 3D user interfaces is given by Bowman et al. in [BKLP04] and [BKL*09].

5.1.1 2D and 3D Travel in the DAVE

5.1.1.1 Related Work

Viewpoint changes by physical walking in the VE are often very limited due to screens, tracking volume, room sizes or cable lengths. To allow moving beyond the physical dimensions of the VE, travel techniques exist for immersive VEs, allowing to modify an additional offset and orientation. An overview is given in [BKH97], [BKH98a].

Gaze. The direction of travel can be derived from the gaze direction. However, unwanted motion may result when looking around during travel [BKH97].

Pointing. By pointing a wand and pressing a button, the user travels into the pointing direction [Min95]. This allows independent travel and viewing, opposed to gaze based traveling. Smart methods for velocity control are interesting especially for large scaled scenes to quickly cover large distances. Trivially, velocity or acceleration can be controlled by pushing a button. Mine et al. control velocity by measuring how far the user's arm stretches away from the body [Min95]. Jeong et al. allow the user to control velocity by the applied force on a button [JJK*04], also comparing it to other methods [JSCH09].

Leaning. Leaning can also be used for travel, either measured by a tracking system [LFKZ01], with force sensors measuring the weight distribution of a board the user stands on [dHGP08] or with contact mats triggered by stepping on an area off the center [Aus03]. Such leaning or stepping techniques are developed to free the user's hands for other tasks, such as object manipulation with a hand held device.

Locomotion Devices. Physical locomotion devices are a way to allow physical motion, simultaneously traveling. One or two dimensional treadmills are a common choice [DCC97]. To compensate for balance problems, some 1D treadmills can be rotated. Especially the 2D treadmills require a large mechanical effort. The *Cyberwalk* project shows such a setup consisting of a number of small 1D treadmills mounted on top of another large 1D perpendicular treadmill. The *Cybercarpet* uses a number of small rotating spheres the user can walk on. The *Cybersphere* is a large sphere suspended on an air cushion [FRE03] in that a user can walk in any direction. The sphere also serves as a rear projection screen. The *VirtuSphere* is a similar device with a wireless HMD and a plastic sphere on bearings.
Some simulators allow a similar motion to the real motion, like the swimming simulator [FKC*05]. Similar devices are often used for training muscles. However, these walk locomotion devices never feel natural and require a large mechanical effort. Stationary cycles are easy to build and intuitive to use, like the Georgia Tech *VR Bike* or the experiments described in [SPDAM*02].

Virtual Vehicles. Travel can be achieved by steering a virtual vehicle. A car, plane or magic carpet could be used, or an animal or walking person may be controlled. The camera or view can be realized by an automatically following external camera on a smooth path. Care must be taken so that a sensible part of the scene is visible and the viewpoint does not enter scene geometry by accident. This sort of travel is often used in computer games. Such motion may be restricted. As an example, walking is often restricted to a 2D or 2.5D surface due to gravity. Predefined paths may be used similar to e.g. a roller coaster.

Walking in Place. With *Walking in Place* [SUS95], the user lifts the feet alternatingly in order to travel, similar to real walking. According to their study, this improves subjective presence opposed to more indirect ways of traveling. In addition, the user's hands are free for other tasks. To allow faster motion in large scale scenes, Interrante et al. present the *Seven League Boots* interface [IRA07], scaling the motion along the intended walking direction.

Grabbing the Air. Panning with one or two pointers can be used for object manipulation as well as for travel. An example is *Grabbing the Air* [Mul98], where a user wears two gloves and can pinch the fingers to close a contact, thereby grabbing that location in order to move the world location, allowing rotations with two hands.

World in Miniature, Maps. The *world in miniature (WIM)* metaphor [SCP95] can be used to get a static map or overview of the world and move a representation of the observer in it, thus traveling. Quick motion or teleportation is possible but the motion is not very precise. Also, teleportation is reported to have negative effects on building a cognitive map, i.e. wayfinding.

Automatic Rotation. Many CAVEs have only three vertical walls, leaving one wall without projection. To lower the number of times the user looks at the missing vertical wall, La Viola et al. suggest to automatically rotate the scene so that the user faces the front wall [LFKZ01]. A different approach for real walking and HMDs is to increase the perceived room size in the VE by *redirected walking* [RSS*02], where the scene is rotated unconscious to the user. This really works only for a known path or at least a direction. For relatively free walking, a room size in the order of $20m \times 20m$ is necessary.

Orbiting Viewpoint. A common way to inspect objects at desktop setups is an orbiting viewpoint. This is rarely used in immersive VEs. A problem may occur especially with a uniform background and a display with a small field of view: An error in navigation, e.g. an inconvenient pivot point for orbiting, can lead to the user being *lost in 3D space*, i.e. the object is not visible on the screen. It is often hard to recover in order to see the object again, a frustrating experience. Fitzmaurice et al. present several ideas to overcome such problems by providing more feedback and implementing an undo slider for navigation [FMM*08]. To explore a virtual world, velocity or acceleration are often controlled by the keyboard, joystick or gamepad, sometimes referred to as walk or flight mode. The changes are relative to the current state. An effective way used in many PC games is to use the keyboard for translation with one hand and the mouse for rotating the view with the other hand.

5.1.1.2 Navigating 1:1 Scaled Scenes

The majority of our immersive 3D applications are real world scenes, where interactive navigation is realized by walking or flying through the scene. They are displayed with metric values and their real scale. Also, they usually have a static up direction. As the DAVE volume is quickly reached by walking inside the volume, additional means for traveling further in the scene are necessary. Our goal is to provide a very general travel method that works for all such models in order not to confuse people when changing the application. A precise and slow but also a fast motion should be possible. Finally it should be quick to learn and easy to use.

Pointing with the Joystick. A straight forward approach to move about is to use a pointing device and move into that direction when pushing a button. This was implemented in the DAVE from its early days. We constructed a joystick with an analogue trigger for navigation and four additional buttons for additional input (see section 2.3.4.1). During the years we adapted the navigation to both our needs as well as the observed behavior of first time users. Lots can be learned from the visitors, who often start testing during our explanation. From their reactions it is often possible to see if the behavior meets their expectations. Usually, we just teach the basics and provide assistance only when further information is necessary. As an example, some users do not figure out themselves how to fly backwards or to turn and ask how to achieve the task. However, not all requirements mentioned above are met with the described system. Our solution is a novel combination of several velocity control techniques, partially similar to the techniques by Mine et al. and Jeong et al. In addition, a simple automatic rotation is implemented following the idea of LaViola et al. An additional explicit rotation is also possible. Our current implementation is described in the following paragraphs.

Speed control. Three speed factors determine the velocity. The first is the analogue joystick trigger, the second one is the distance of the joystick to a position below the user's head, and the third is a current maximum velocity. This current maximum velocity itself is also dynamic, growing or shrinking exponentially over time, depending on the first two factors. It is limited to a highest value derived from the bounding box of the scene. In that way, the user is able to precisely navigate at walking speed as well as quickly flying over a city. To slow down, we use inertia to brake continuously and quickly. This avoids an unnatural sudden stop when releasing the trigger, which feels too sudden and disturbing especially at a high speed. The user's arm length can be estimated using the maximum glasses' position above the floor over a period of time, assuming an upright standing user. We have not implemented this idea yet and only assume a constant. Currently, the arm stretch factor is computed using the distance from the joystick to the position 30cm below the glasses. When a novice user intends to travel fast, we observe that the speed increases too slowly at first or may increase to quickly after a short while. We assume that parameters can be chosen to counteract these effects.

Rotation. As the DAVE does not have a projection wall on the back side, a user will not see the virtual scene in that direction. When also wishing to move in that direction this is rather inconvenient. We solve this problem with two further control options. The first is stretching out the arm and rolling the joystick around the arm axis. In fact, we implemented this method because we observed novice users making that motion, which had no effect at the time. In a way, this is similar to playing with a toy plane in the hand, or rotating a steering wheel. For some users, it feels natural to initiate rotation in that way. Only rarely this leads to unwanted motion.
The second technique is motivated by LaViola et al. [LFKZ01] and Razzaque et al. [RSS*02]. Their goal is to use the torso direction to determine the principal direction and rotate the scene so that the user faces the front wall. They use a slow constant rotation and additionally increase the rotation velocity during a quick motion or rotation of the head. As we do not want to employ torso or hip tracking out of practical reasons, we rotate the scene depending on the pointing direction of the joystick, projected on the ground plane. The rotation is only applied when also moving, using the analog value of the trigger and the hand distance from the body. To avoid quick turning when moving backwards, we use the sine of the angle between the forward vector and the projected joystick vector, multiplied by a factor found by experimentation. In that way, a straight backward motion is possible.

Exaggerated Rotation. The described interaction methods allow for an easy to learn, precise and quick control to cover small and large distances. They use only two 6DoF and an additional trigger value. The implementation of travel has shown to be more complex than first expected. The biggest remaining problem for new users is that they get stuck in confined rooms.
To avoid such problems, we also tested the idea of non-linearly exaggerating the head rotation, as independently described in [MVL00] and [LFKZ01]. We use a simplified approach, again only using the head rotation as tracked input value, i.e. without using hip tracking. While this approach generally works well and users are hardly aware of the exaggerated rotation, the latency is much more noticeable. Similar to HMDs, with this technique, static content now moves a lot on the screen depending on the head rotation. Especially quick head rotations increase discomfort.
Note that in all described rotation methods, a simple rotation in the navigation code leads to a rotation around the origin, which is the center of the DAVE in our case. To rotate around the user as intended, an additional correcting translation is necessary.

Leaning Into a Curve. We also tested implementations of an aircraft or motor bike style navigation, mainly with a rolling camera. Trying to implement such a technique that simultaneously allows the user to freely move and turn in the DAVE, all of our attempts have failed. We assume that with additional tracking information like the hip orientation, ambiguities can be resolved. However, balance problems of some users may increase. Another idea is to use a seat, constraining the user's principal direction, similar to sitting in an air plane cockpit.

Portability. Unfortunately, each application needs a slightly different way of implementing the navigation algorithms. The code is short but it may be worth moving it to a library. At the moment we have individual implementations for each application, allowing an easy development. However, once we are satisfied with a change, we need to adapt all other applications. To facilitate development and maintenance in future, the same code for travel may be used with the IO device library, see section 5.4.1. We already implemented a navigation library that can be used e.g. for orbit, flight and walk navigation, with no special relation to the DAVE. It uses logarithmic scaling and momentum and damping for a continuous motion.
For instantreality applications, standard navigation methods exist for a desktop setup, whereas in the DAVE, our joystick navigation is loaded.

Observations. Sometimes, novice users trying to control the up and down directions do not understand at once, that the direction of the joystick or the arrow attached to it matters. Instead, they assume that the angle of the arm is relevant. However, the respective shoulder is not tracked, so using the orientation of the joystick is a more robust and controllable way.

Gamepad. To provide more buttons and control options, we also modified a wired gamepad, and later on used a common wireless gamepad. The idea is to allow more motion control, such as 3D translation, rotation around 3 axes and scaling. The latter degree of freedom is arguably not an element of navigation, but can also be implemented by modifying the same camera transformation matrix. Throughout a few years of experience with the gamepad, we observed that most of the additional controls are hardly ever used. As an example, it is rather disturbing to tilt the horizon in a real world scene, which even bears the potential risk of users loosing balance. In our experience, the gamepad is not the ideal interface for the DAVE. Pointing with a two handed gamepad, as necessary for our velocity control, feels rather awkward. Also, as there are many first time users and we aim at reducing the explanation time to a minimum, we eventually replaced the gamepad with the joystick.

5.1.1.3 Navigating Scaled or Artificial Scenes

Sometimes, a realistic scale is not optimal. As an example, a 3D model of the solar system would appear very large and objects far away. Better suited might be a scaled down model, so that e.g. the earth appears as a sphere with a diameter of 1m. More obvious is the inspection of e.g. nano structures. At the real scale, they would just not be visible at all. A magnified version, again to approximately human scales, is probably much more useful.
Another idea is to scale eye distance instead of the model. However, we found that it is less confusing to keep all tracking and navigation related coordinates in real metric values and only scale the model to an appropriate level. This is as simple as applying the scale to the model view matrix. Note that this also affects normal vectors for OpenGL fixed function lighting, so that the vertex normals should be normalized before usage.

5.1.1.4 User Position and Pose

Absolute Values. Without a physical device, the position or pose of the user can be directly used as input values. The idea of using a pose or gesture more similar to a pose in the real life is promising, hoping to achieve a more intuitive way of navigation by using implicit knowledge or reasoning. With the PPRacer game (see figure below and section 6.3.9.2) as an example, the user controls a penguin which gets faster when the user lies down, a pose that copies the penguin and seems better suited for gliding down the slope than standing upright. We simply use a thresholded signal of the absolute head position to control the penguin. Sometime users may unconsciously move with their body off the center and therefore not be able to move the head far enough

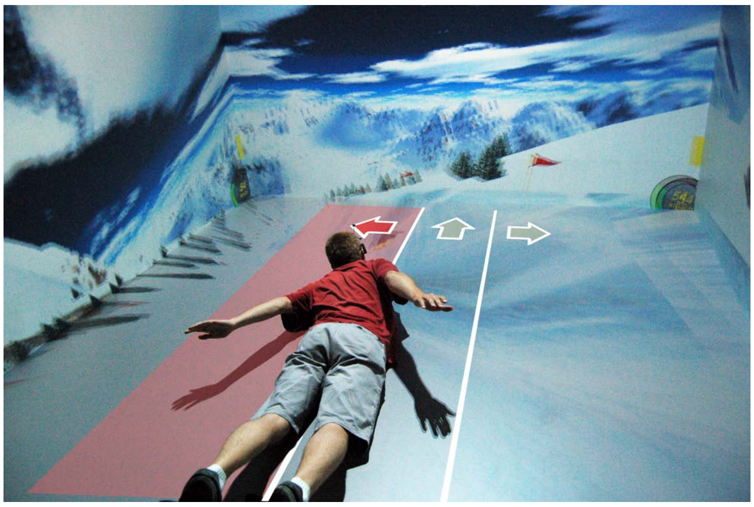


Figure 5.1: A simple control only using the absolute position of the glasses. Left, center and right positions are used for steering as indicated by arrows that are overlaid on the photo for illustrative purposes. When the glasses are higher than 1.4m above the ground, the penguin brakes. When they are lower than 0.8m above ground, the penguin speeds up.

to steer in the intended direction. An additional feedback or at least a projection of a center line may be useful.

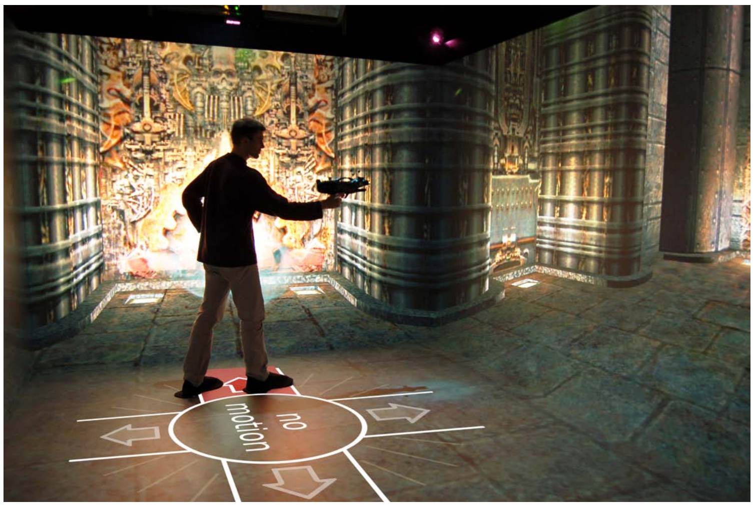


Figure 5.2: In this Quake III arena game, the absolute position of the user is employed to control the walking speed. It turned out that this method is rather hard to use. The arrows were added for illustrative purposes. Also note that the photo was taken in mono, from a point of view near the camera and the user does not wear the glasses, for a nicer photo.

Leaning. We conducted experiments with travel triggered by the head position offset from the center of the DAVE. This is similar to velocity control by leaning, and the rotation speed around the up axis by the head orientation, as described first by Fairchild in 1993 [FLL*93] and was later refined in [FSG98] and [LFKZ01]. As we do not have tracking information of the lower body part, we simplify the leaning to an absolute offset from the DAVE center. We were skeptical about its outcome and indeed it proved rather hard to control. As an example, the game *Quake III Arena* in the DAVE is a lot harder to control with that technique than with a normal PC with a mouse and a keyboard.
For more details, please refer to our publication [SRO*08].

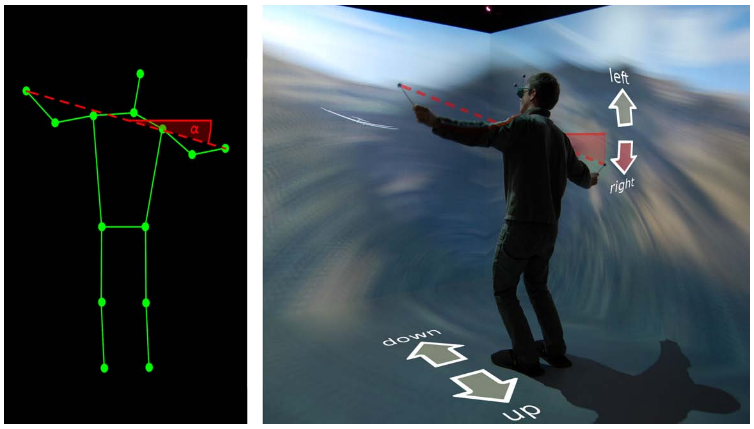


Figure 5.3: Using the four markers on the glasses plus a stick with a marker in each hand, a pose is estimated. This pose serves as input to control an application, like this glider simulation. As indicated by the illustration, the user lowers the right arm and raises the left arm in order to fly a curve to the right. A step forward results in tilting down, stepping back leads to flying up.

Skeleton Based Interaction. For the input of the glider application, we use a two stage approach. The first part is is a skeleton estimation of the user. Different methods can be used to estimate the skeleton from the few input values. One uses a model constructed by reasoning, another uses the most likely pose from a database which we obtained with the help of motion capturing (see section 3.1.7).
This skeleton is then used as input to control the glider (see figure below). This approach makes it easy to use a different method for estimating the skeleton without modification of the game control logic or allows other applications to use the same interface.
In fact, due to the well working Kinect hardware together with e.g. the *openNI* library that quite robustly estimates user skeletons, there is an extended interest in such skeleton based interaction. For more details, again refer to our publication [SRO*08].

5.1.1.5 BCI

Together with the Department of Psychology, we conducted several studies with a Brain Computer Interface (BCI) in the DAVE. Unfortunately, reality is far behind hopes expressed in science fiction. At the moment, with this setup very few distinctive patterns can be recognized with about a couple of seconds delay. Also, a tedious wiring and a long training period are required. We tested simple start and stop commands to navigate on a predefined path. In another experiment, binary commands were interpreted to trigger an event. Even though this is very restrictive, it may be an interesting way to communicate for some severely disabled persons. The experiments and setup are described in more detail in section 6.3.10.1.

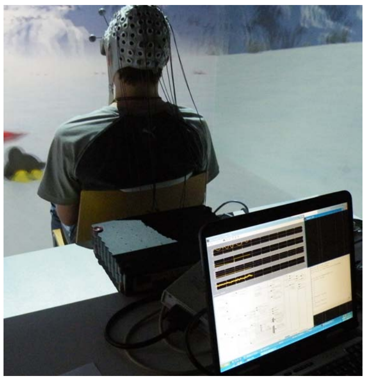


Figure 5.4: Brain computer interface (BCI) used as input to control navigation in the DAVE. A state change is detected within about two seconds and send to the DAVE server via UDP or TCP. See section 6.3.10.1 for a description of the applications.

5.1.2 Object Manipulation in the DAVE

5.1.2.1 Related Work

Object selection is often realized by ray casting [PFC*97]. Gaze direction, pointing direction or bearing from the eye to a hand may be used. However, resting the hand is not possible and natural tremor makes it difficult to precisely control pointing. Additionally, when pushing a button, the hand may move, which makes a selection of a small target difficult.
Once an object is selected, translation and rotation changes can be applied in a relative way, copying the motion of the input device.
Objects can also directly be picked at their location (i.e. without ray casting), with physical limitations of the VE often requiring additional travel. A technique to non-linearly extend the arms is go-go presented by Poupyrev et al. [PBWI96].
Kaiser et al. support and disambiguate selection and object translation by additional voice commands [KOM*03].
Especially for small working volumes for 3D interaction, haptic input devices are well suited. Motorized arms like the *Phantom* device [SS97] or points in 3D controlled by wires like Spidar [BIS01] work well for a desktop setup. For collision detection a high update rate of 1000 Hz is common.
For an intuitive and natural behavior of objects with respect to manipulations, gravitation can be used and objects may not be allowed to interpenetrate. A physics engine with collision detection can be used to meet these requirements.
Fröhlich et al. present the *Responsive Workbench*, where virtual springs can be attached to objects to apply forces, while the mentioned requirements are met [FTB*00]. This also enables simultaneous collaboration.

5.1.2.2 Picking with the DAVE Joystick

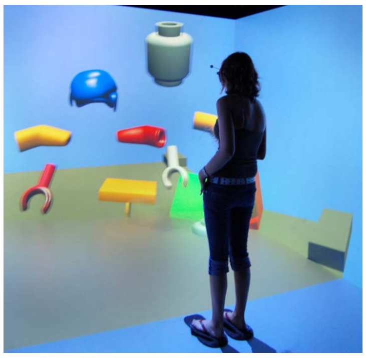


Figure 5.5: Picking and moving objects. Picking is started by ray intersection on a joystick button press. Relative motion and rotation of the joystick are then applied to the object.

The most successful object manipulation application that we have implemented for the DAVE, is repositioning and rotating items in an OpenSG application (see section 6.3.3.1 and side figure). A ray from the joystick in the pointing direction is used to pick the closest intersecting object. To simplify aiming, this ray is visualized before picking. A method similar to the *go-go* interaction technique [PBWI96] proves successful, where a large translation is scaled non-linearly. We also tried to use the same idea of non-linear scaling for rotation, as for some rotations, the user must twist the hand uncomfortably or split the rotation into several small rotations. However, we found that this is rather hard to control, as a user seems to expect consecutive rotations to be independent. But also, undoing a physical rotation of the hand should undo the rotation of the virtual object. As an example, after picking up an object and slowly rotating it for inspection, the user may position the hand back into the initial state, also expecting the virtual object to do so.

As in the current implementation the physics simulation for the selected object is turned off during manipulation, it can also be moved through other objects like walls. In some applications, this may be an unwanted behavior or result in unwanted motion due to collision response once the object is released. However, similar to [FTB*00], we could move an invisible triangular frame instead, temporarily connecting each of its vertices with a spring to the closest object surface. This would indirectly allow a 6DoF manipulation of the object, as well as preventing the object from penetrating other objects. We have not implemented this idea yet.

A related object manipulation is an indirect control of parameters for a *Generative Modeling Language (GML)* [Hav05] object, which allows parametric modeling. While a 2D GUI on a handheld device can be a good way, another option is to move sliders in 3D space, positioned close to the object itself. This makes sense especially for parameters controlling the spatial extend of the object, like for resizing the wall of a house or a window in that wall. A small colored sphere indicates a handle and a path may show a restriction where it can be moved (see section 6.3.4.4).

We have also tested triangle mesh manipulation and coloring, see section 6.3.4.2. However, for sculpting we found that a desktop setting with a haptic device is much better suited.

At the moment, we use the same input device for either traveling or object manipulation and switch the mode with a button. A dedicated second device could be used, however this is likely to confuse novice users. A second device may also be interesting in order to test two handed 6 DoF interactions like object deformations with *Twister* [LKG*03].

5.1.3 Menus and GUI for DAVE Applications

Many designs for 3D menus and GUIs were tested by other researchers. However, the additional third dimension compared to a desktop setup does not seem to improve the overview and usability. We only tested a simple menu in one application to select a furniture item (see section 6.3.3.1 and side figure). A problem occurs when the 2D menu, which is rendered on a plane in the 3D scene, intersects with the scene geometry, partially hiding the menu. When disabling the OpenGL depth test, the menu will never be hidden, however its depth will still be perceived as behind the object surface. Instead, it may make more sense to use a portable device with a conventional 2D GUI. The interface to the application can be realized with the IO device library (see section 5.4.1).

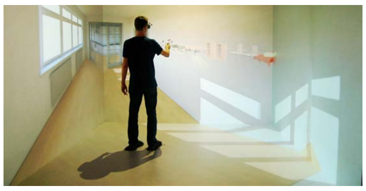


Figure 5.6: A GUI with 2D images arranged in a 1D array, placed within the 3D scene. Problematic are clipping planes and occlusion of the menu items by close objects in the scene.

5.2 INTERACTION ON THE HEYEWALL

5.2.1 Related Work

Guimbretière et al. show ways of interaction with 2D documents on a large wall display [GSW01]. Especially, digital white board functionality like annotations or zooming are discussed. Wissen presents a laser pointer interaction for a large wall display [Wis01], facilitating positioning or object dragging over large distances. Ball et al. examine physical navigation versus moving objects on a large display wall [BNB07], also using a handheld Gyration mouse. Bornik et al. work with a hybrid approach using both a conventional tablet PC for system control as well as a handheld tracked input device to work with volumetric data [BBK*06]. Weibel et al. suggest to use an *Anoto* pen on a paper on a clipboard [WPH10], as an indirect but lightweight input device. Another example is presented by Ballagas et al. who use the camera on a cell phone to enable pointer interaction [BRS05]. For each click, the screen shows a pattern that the phone uses to locate the pointer position.

Game console devices like the Nintendo *wiimote* or the Microsoft *kinect* can also be used to control image content of large displays. With an additional table next to the display wall, just mouse and keyboard or tangible tracked objects are an option, as well as a multi touch table.

An slightly dated overview of public interaction on large screens is given in our report [LB04]. There, cell phone interaction or vision systems are briefly described, enabling different types of applications.

5.2.2 Implementation

Important intended interaction methods for the HEyeWall are touch interaction which is described in the following section, an infrared laser pointer and a wireless gamepad. New applications may require novel ways of interaction and further research in that area. A kinect sensor looking down from the ceiling to recognize arm gestures may be an interesting interface in future.

5.3 TOUCH SCREEN INTERACTION

Multi touch user interfaces go beyond just multiple pointer support. The great vision is that digital objects become (almost) tangible, can be touched and manipulated very much like objects of the real world. The challenge is to provide virtual objects with physically plausible behavior, hoping to enable access to the digital world for a wider range of people with less computer experience. In that sense, an intuitive, natural behavior of the interface is usually very important.

5.3.1 Object Manipulation with Touch Screens

5.3.1.1 Related Work

In some cases of 2D interaction, touch and multi touch interactions are more sensible than mouse input. On a touch screen, the interaction takes place where the information is displayed, requiring less cognitive load than indirectly moving a mouse pointer across a dislocated screen. A major drawback is that precise manipulation is difficult because fingers and hands occlude parts of the screen. In addition, the finger tip is usually significantly larger than a pixel and a pixel precise selection is hard. Examples for touch screen installations are public displays or screens in industrial, medical or collaborative environments with special requirements. Selection may be more convenient than with a mouse. As the price for one or two pointer touch technology drops rapidly, touch support may become very common for normal desktop monitors in future.
Ringel et al. discuss touch interaction for a display wall [RBJW01] with a rear diffused illumination setup. The *magic table* is an early example of translation, rotation and scaling of photos with one or two fingers and for multiple users [LB04]. The fingers are tracked with a camera and the image is projected from above on a table.
The idea can be extended for 3D objects on a 2D screen. Hancock et al. describe multiple interaction possibilities allowing rotation and 2D translation [HCC07]. In one case, the number of touching fingers is used to select among different modes. Another technique uses areas relative to the object, where the position of the first touch selects the mode. The motivation and application is to arrange 2D documents, represented as flat boxes on a desktop.
The issue of 6DoF motion of objects is discussed by Schöning et al., who especially consider parallax effects on a stereoscopic projection screen setup [Sch08]. Similar to our observations, they state that the control of all six degrees of freedom is non-intuitive and requires complicated gestures.
Reisman et al. use a direct 6DoF manipulation with up to three fingers [RDH09]. The transformation matrix is computed by a constraint solver so that the surface stays under the three contacts. Interaction with three fingers seems to require some learning and large rotations may have to be split up into several smaller rotations. A fourth finger can be used to also control the focal length of a perspective camera.

Agarawala et al. present the *BumpTop*, a 3D box as a desktop, featuring desktop icons and documents with physically behavior [AB06]. The objects are flat boxes that can be pushed around, piled up or spread out, trying to recreate a more realistic setting similar to a real world desktop. Selections, menus and gestures are supported.
Wilson et al. show physically based 6DoF manipulation and also soft body interaction on a vision based touch screen that can also detect a palm or other tangible objects [WIH*08]. In their system, arbitrary contact shapes are realized by representative particles on the contour, each applying a force. Optical flow is used to compute the motion vectors of the particles. Another technique uses joints, similar to ropes of a fixed length. An off-center force leads to a rotation, also with just a single contact. Hancock et al. also present a physically based 6DoF interaction [HtCC09]. They use gravity and a metaphor allowing to pick up objects.
Jeff Han showed a number of well suited applications for multi touch tables at his TED talk [Han06] in 2006.
As mentioned before, tangible objects can be used on table setups, if supported by the sensing technology. Examples are the reacTable using reacTIVision [KB07], where tangible objects can be combined in order to produce sound, or the ARToolkit [KB99] which allows to track objects with an attached marker using a camera. However, if a multi touch screen is used which does not natively support detection of fiducial objects, our method may be used to recognize fixed constellations (see section 3.2.4).

5.3.1.2 Moving, Rotating and Scaling 2D Objects on Touch Screens

We implemented two different methods for multi touch object manipulation. The first uses gestures and is described in the following paragraphs, while the second uses a physically based approach with virtual springs, also suited for 3D object manipulation.
A naïve implementation for 2D object transformation can simply translate the object according to the translation of the first touch point, and use the relative angle and distance of the first two touch points to modify orientation and scale. However, unexpected or unwanted motion results. Most importantly, the object surface moves relative to the fingers, instead of sticking under them.

Basic Version. For a simple, intuitive manipulation, the necessary algorithms are much more complex than we first expected. The code below, written in *processing*, shows a much better, but still basic version for 2D object transformation.
With the code below, with one or two fingers, the first touched local object position always remains under the fingers. With more than two fingers, averages of results of all pairwise combinations are used. Whenever a finger is released or added, the local reference finger position *firstTouchPos* on the object is updated (lines 3 to 9 and 57 to 64). When fingers are moved, their new local coordinates *currentTouchPos* are computed (line 5). The new orientation is computed by the average difference angles of all pairwise combinations of two *firstTouchPos* and two *currentTouchPos* difference vectors (lines 11 to 28). Similarly, the scale is calculated (lines 30 to 44). With the updated rotation and scale, the average translation is computed for each contact (lines 46 to 53).

```
void refresh(TuioTime bundleTime) { // update translation, rotation, scale
  if (moved) {
    // update local values
    for (int i=0; i<cursorNum; i++) {
      cursors[i].currentTouchPos = global2local(cursors[i].currentTouchPosGlobal);
      if (needUpdate) { // update of firstTouchPos
        cursors[i].firstTouchPos = new PVector(cursors[i].currentTouchPos.x, cursors[i].currentTouchPos.y);
      }
    }

    // update rotation
    if (cursorNum > 1) {
      PVector sumF = new PVector(0,0);
      PVector sumC = new PVector(0,0);
      for (int r1=0; r1<cursorNum-1; r1++) for (int r2=r1+1; r2<cursorNum; r2++) { // all pairs
        // pairwise rotation
        PVector rotF = new PVector(cursors[r1].firstTouchPos.x, cursors[r1].firstTouchPos.y);
        rotF.sub(cursors[r2].firstTouchPos);
        rotF.normalize();
        sumF.add(rotF);

        PVector rotC = new PVector(cursors[r1].currentTouchPos.x, cursors[r1].currentTouchPos.y);
        rotC.sub(cursors[r2].currentTouchPos);
        rotC.normalize();
        sumC.add(rotC);
      }
      tOrientation += atan2(sumC.y, sumC.x) - atan2(sumF.y, sumF.x); // add difference angle
    }

    // update scale
    if (cursorNum > 1) {
      //float avgScaleF = 1;
      float scaleSum = 0;
      int sNum = 0;
      for (int r1=0; r1<cursorNum-1; r1++) for (int r2=r1+1; r2<cursorNum; r2++) { // all pairs
        // pairwise scale
        float scaleF = cursors[r1].firstTouchPos.dist(cursors[r2].firstTouchPos);
        float scaleC = cursors[r1].currentTouchPos.dist(cursors[r2].currentTouchPos);
        //avgScaleF *= scaleC/scaleF;
        scaleSum += scaleC/scaleF;
        sNum++;
      }
      tScale *= scaleSum/sNum; // multiply with scale difference
    }

    // update translation
    PVector t = new PVector(0,0);
    for (int i=0; i<cursorNum; i++) {
      t.add(cursors[i].currentTouchPosGlobal);
      t.sub(local2global(cursors[i].firstTouchPos));
    }
    tTranslation[0] += t.x/cursorNum; // add average position difference
    tTranslation[1] += t.y/cursorNum;
  }
  moved = false;

  /////// update of firstTouchPos
  if (needUpdate) {
    for (int i=0; i<cursorNum; i++) {
      cursors[i].currentTouchPos = global2local(cursors[i].currentTouchPosGlobal);
      cursors[i].firstTouchPos = new PVector(cursors[i].currentTouchPos.x, cursors[i].currentTouchPos.y);
    }
  }
  needUpdate = false;

  redraw();
}
```

Figure 5.7: A basic example written in processing, showing the part responsible for 2D object manipulation on a multi touch screen.

Improvements. We implemented a number of further improvements. Image indices are stored in an ordered list, so that recently touched images are drawn on top of images that were not touched for a longer time. Inertia for translation, rotation and scale has been added. We found that physically correct drag does not represent what we intend, so we decrease the drag for fast moving objects, effectively allowing flicking. We also introduce soft constraints on scale and translation: When released, objects smoothly adjust to their specified limits. In contrast, using hard constraints, it may not be obvious for a novice user that a constraint is active. When sliding freely without finger touch, objects bounce off the window border. To mimic damping and friction at a collision, we decrease the speed and add a torque. When dragging an image with a single finger off center, we also add a torque. The mentioned effects also work when zoomed in by a large factor. These improvements are illustrated in the video below.

Figure 5.8: Video (without audio) showing advanced features of moving 2D objects on a multi touch display. The objects robustly stay under the fingers when possible. Soft constraints and lower drag for flicking are additional features. Also, when clicking on an object, it comes to the front.

5.3.1.3 Physically Based Manipulation with 2D and 3D Objects on Touch Screens

Similar to the *Responsive Workbench* [FTB*00] where objects can be moved in an immersive setup with the help of virtual springs, we implemented a physically based manipulation for 3D objects, also using virtual springs that can be attached to the object. However, we transferred the method to a 2D multi touch setup and added the ability of scaling as well as visual cues for a user feedback. The temporary spring forces between each finger touch and the object surface are employed to provide an intuitive interface to translate, rotate and scale 3D objects. The method provides clear feedback to the user and has no hidden states or functionality.

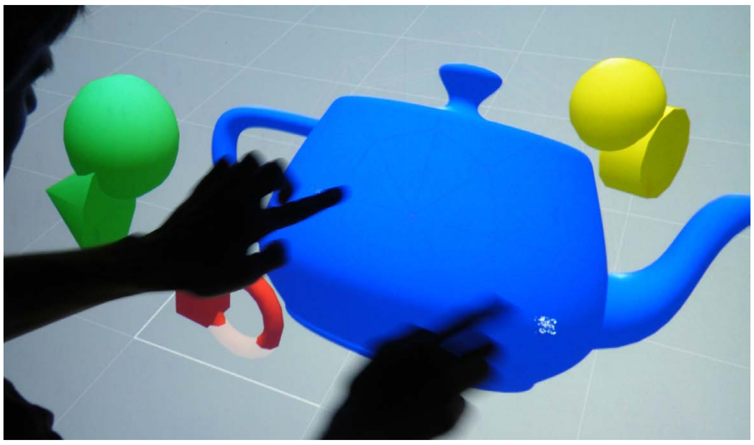


Figure 5.9: Attaching virtual springs, 3D models can be moved, rotated and scaled on this multi touch wall.

In the following description we refer to object manipulation methods as *gesture based* when a finger drag directly influences the object properties, like described above for the 2D object manipulation.

Drawbacks of Gestures with Multiple Modes. Gesture-based multi touch input methods are commonly used in practice. Usually they distinguish between different interaction modes by the number of fingers on the object. As a common example, a single touch motion may be used for 2D translation and two touch points influence 2D rotation and scaling. A third point allows both the definition of a rotation axis with the first two touch positions and the control the amount of rotation with the last finger, also using the temporal order of touches. This results in a very different behavior depending on the number of fingers. This works reasonably well for a trained user, however simply brushing over the object with a few fingers is likely not to lead to predictable motion. Such a gesture may be one of the first things an unistructed novice user tries and the feedback is not helpful to understand the system or learn how to use it better.

Physically Based Method for Moving 3D Objects. The goal is to overcome these drawbacks and at the same time to provide a general manipulation for six degrees of freedom: 3D rotation and 3D translation. As an alternative to 3D translation, 2D translation plus uniform scaling may be used. We achieve this with a physically-oriented simulation method for moving objects by attaching spring forces to surface points. The geometric configuration is shown in the figure on the side.

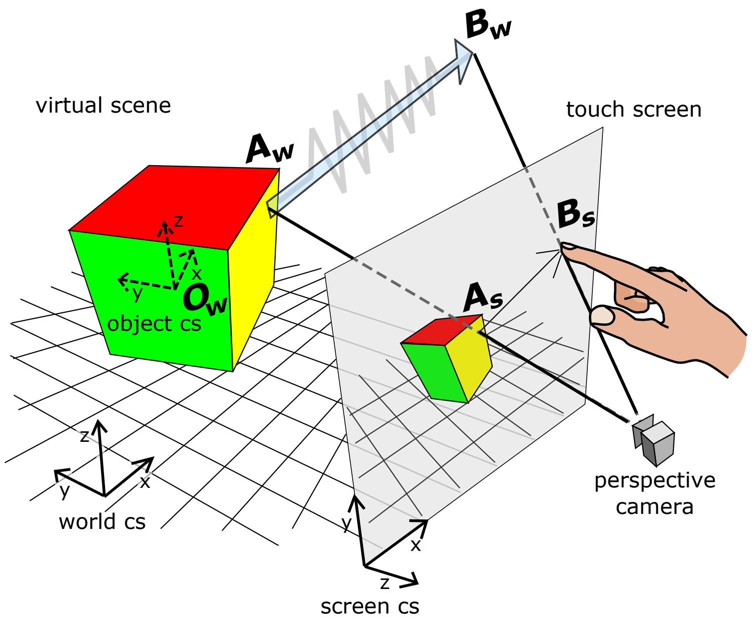


Figure 5.10: The virtual scene with object and world coordinate systems and the image on the screen with the screen coordinate system. For a finger drag from A_s to B_s a spring force from A_w to B_w is applied to the object.

Computing the end points of a spring. At the first new touch of a finger at screen position A_s, the world coordinates A_w of the *surface contact point* on the object are determined. This point A_w is then transformed to the local object coordinate system, which is used as one end point of the new spring. The z_s coordinate in the screen coordinate system is saved as well at the first contact in the local object coordinate system.
The second spring end point B_w moves directly with the finger, calculated with the updated B_s touch coordinates from the multi touch interface and the previously saved z_s value. Clearly, the goal is to decrease the distance between B_w and A_w by moving the object appropriately. Also A_w has to be recomputed in each frame because the object moves. It is the first surface contact point in the object coordinate system that is transformed into the world coordinate system.

Numerical Physics Simulation Step. In each simulation step, after updating all spring end points the formulae of Newtonian physics are evaluated numerically for each object. The equations for translation and rotation are straightforward, with the translational acceleration

$$a_t = w_t \cdot \sum \overrightarrow{A_w B_w} \tag{5.3.1}$$

with the weights $w_t = 1/\text{mass}$, and the angular acceleration

$$a_r = w_r \cdot \sum (\overrightarrow{O_w A_w} \times \overrightarrow{A_w B_w}) \tag{5.3.2}$$

with the weights $w_r = 1/(\text{moment of inertia})$. Similarly, we introduce the acceleration of the scaling factor by

$$a_s = w_s \cdot \prod \frac{\left\| \overrightarrow{O_w B_w} - \overline{\overrightarrow{A_w B_w}} \right\|}{\left\| \overrightarrow{O_w A_w} \right\|} \tag{5.3.3}$$

where $\overline{\overrightarrow{A_w B_w}}$ denotes the average force vector, and w_s is a weight to influence the scaling inertia. We consider only uniform scaling, although non-uniform scaling could easily be achieved with three independent scaling factors for each dimension. However, we understand non-uniform scaling as a form of object deformation, which is not the goal here.
Object masses, spring forces and damping (friction) coefficients can be adjusted to fit best to the applications needs.

Scaling. We differentiate between *scaling*, *moving along the z axis* and *zooming*. Scaling means to resize objects without changing their position. Moving objects along the z axis does not influence the object size in the world coordinate system but will also lead to a different size of its projected image (see figure below). Zooming is an attribute of the camera view of the scene.

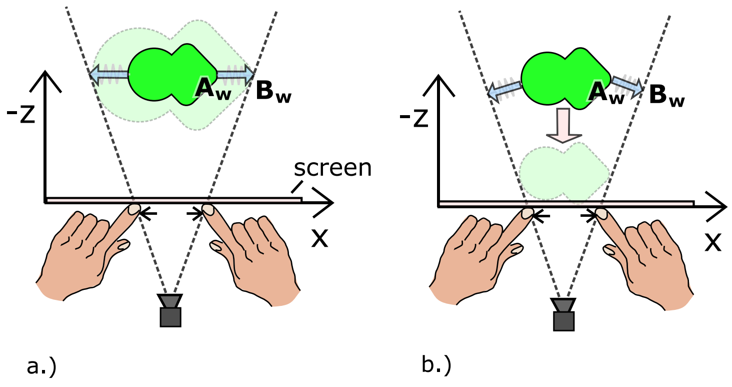


Figure 5.11: Top view in screen coordinate system. Moving the fingers away from each other enlarges the object's projected image. This can be implemented either a.) by scaling or b.) by translation in z direction. The blue arrows show the applied forces.

With the described system up to now, translation in screen coordinate z direction is not directly possible. In fact the object will move along that axis because rotation centers are not fixed, but this effect is not suitable for intentional motion in the direction perpendicular to the screen.
However, the scaling functionality can be replaced by the functionality to translate along the z axis. The computation of B_w changes in the way that z_s is chosen so that the spring length is minimal (see the figure above). Obviously this only works with a perspective camera.

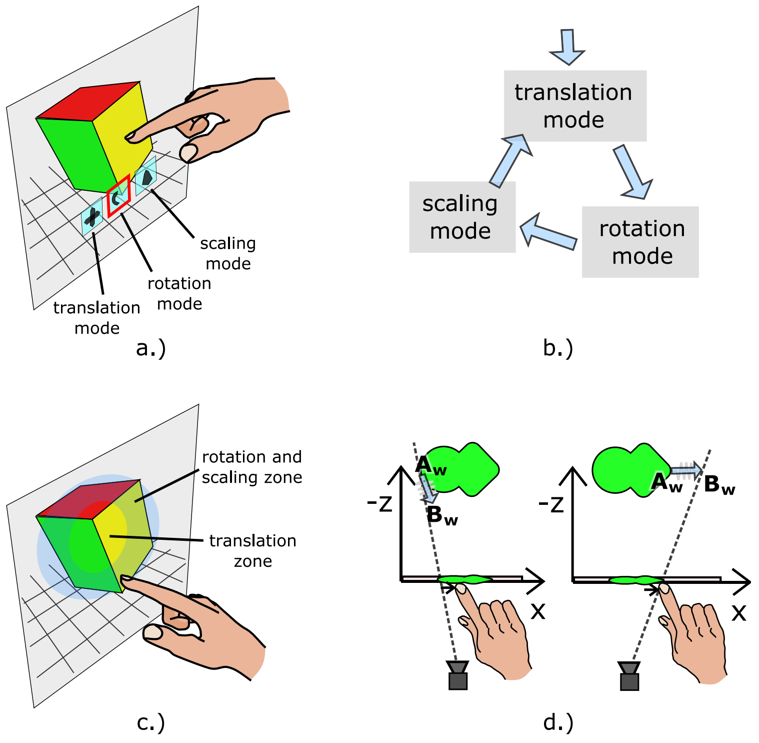


Figure 5.12: Different alternative modes that may allow a quicker and more precise control but also require some training: a.) Context menus, b.) State machine that changes the mode after each interaction, c.) Mode depends on the position of first touch relative to the object, d.) Spring forces are moved in z direction depending on relation to projected center of object.

Constraints and Modes. An advantage of gestures is a precise motion. Such a system can provides several modes with differently restricted motion, allowing e.g. to adjust rotation and translation independently. Such constraints can also be realized with the physics approach. As an example, scaling could be disallowed, or the translation could be restricted to a path. Even though such mode based interaction is less intuitive and takes more time to learn, it may perform a lot better for expert users and certain tasks.
The described scenario is tailored to first time users and intuitive control. We can easily imagine many different expert modes for a faster and finer control, using different modes with different constraints. As an example, an additional finger on the surface could indicate a mode selection of either just translation or just ro-

tation. Expert users will certainly prefer the quicker way that also can be tuned to the specific task. Although the learning time of this technique is very short, it is still significant, and the approach is not intuitive. In contrast, without using constraints or modes, a direct feedback is given with no hidden functionality.
The figure above shows a few ideas for expert methods using modes with different constraints. A global or local context menu can be used to select the current mode (a.). A state machine may be used for drag events in quick succession. The first drag causes only translation, the second only rotation, the next only scaling (b.). A visual feedback to indicate the current state seems useful, e.g. overlaid arrows showing the currently possible motion directions. The position of the first touch relative to the object could be used (c.). In the center zone, a mode could be selected only allowing rotation around the x and y axes. In the border zone, a mode for translation in x and y direction may be selected, also allowing scaling and rotation around z with two fingers. Yet another idea is to use the motion direction relative to the object. A finger drag away from object center in the screen coordinate system may result in a high translation force and little rotation, whereas a motion towards the center could lead to a high torque and little translation. Finally, scene specific constraints should be employed to reduce the number of DoF. Examples are collision detection and other effects of a scene in a physics engine, like gravity.

Properties. For both the gestures and physical based methods we can easily replace the scaling by a translation along the z axis. This may be an important choice for some applications. While one could allow for both scaling and z motion with different modes, additional rendering cues or a stereoscopic display are needed for a clear visual distinction.
We assume that each object is centered in its object coordinate system on its center of mass. With gestures we always rotate around the center of mass whereas with the physics approach it depends on the location of the surface contact points.

Physics Implementation Details. To be most flexible for testing new ideas, we implemented a basic physic simulation instead of using a physics engine. To determine the surface contact position at the first touch of a finger, we use the OpenGL depth buffer and transform the screen coordinates back into the object coordinate system. Instead of using the z buffer value of the exact touch position A_s, we find the closest z buffer value in a 15×15 pixels wide window around A_s. Thus, the picking is more robust and objects can be grabbed at the silhouettes more easily.
An advantage of using the depth buffer is that the object geometry does not have to be known outside of the drawing function. In practice, a rough bounding volume like a bounding sphere is sufficient to decide whether a new touch point belongs to a specific object.

Feedback Visualization. Feedback for the user is very important. Especially when nothing happens after a touch, e.g. due to a constraint, the user should be reassured that the touch was recognized. We achieve this by using moving glitter particles under each touch point. We also show springs rendered as lines, as well as their surface contact points.

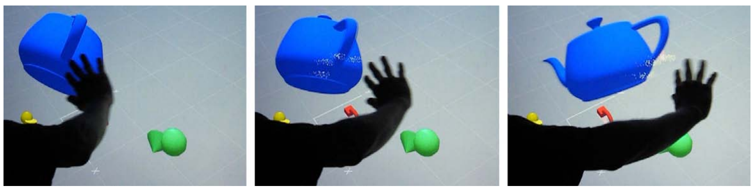


Figure 5.13: **Our method allows brushing over the object, regardless of the number of contact points. This demonstrates intuitive and error tolerant object manipulation.**

Observations. Our experience of the past years shows, that some first time users try to move objects by brushing over them with several fingers (see figure on the side). Some hardware setups are not very sensitive for light touches and may not deliver any detections. A gesture based implementation without inertia may lead to a short sudden motion. Both problems lead to an interface either ignoring the users actions, or behaving in an unexpected way. Thus, we encourage using intuitive interaction methods with constant feedback, like our presented method. When using different modes, the available degrees of freedoms should be clearly visualized.
Multiple contacts may lead to oscillations and instability of the physics simulation. We experienced problems especially for large scale factors and rotations. One idea is to dynamically remove springs that may lead to such problems.

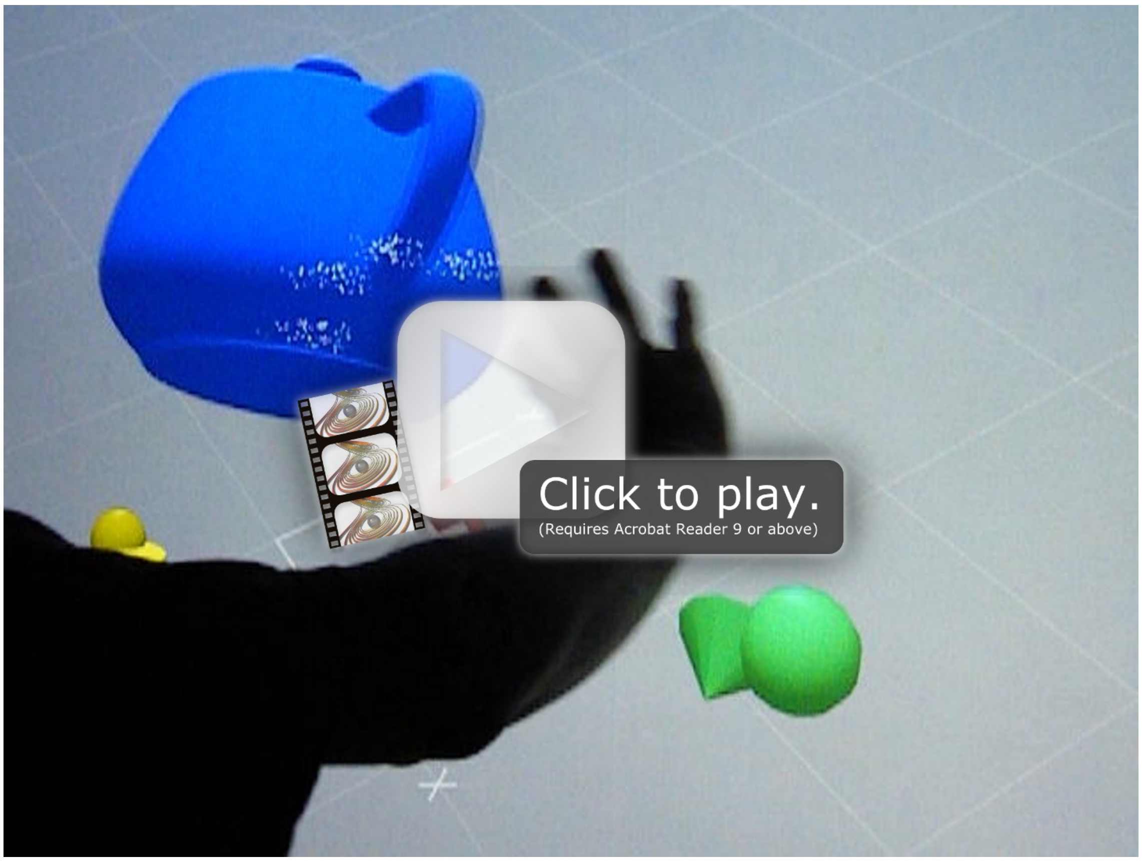


Figure 5.14: **Video (with audio) showing 6DoF multi touch interaction with 3D objects. A simple implementation with gestures as well as a physically based approach are demonstrated.**

Informal User Study. We conducted a small user study with ten computer scientists with a moderate to good knowledge of touch screens. Users had to correctly position, scale and rotate both photos in 2D as well as 3D objects in space, without any explanation or learning phase. Three interaction methods were tested. Method properties

are hard to measure subjectively, they can be largely influenced by the type of application and the chosen parameters. However, we could observe some tendencies.
The gesture approach worked especially well for people having used other multi touch systems before. As expected, a few participants were confused that with multiple finger contacts a translation was not possible any more.
The physically based method without constraints was mostly considered to be a bit easier to understand and more enjoyable to use. In terms of task completion times it performed well with the 2D photo task, with no significant difference to the gesture approach. In the 3D task it was considered to be less controllable, mainly because of unintentional simultaneous rotation and translation.
As third method we used our spring model with constraints, as an example expert mode. We employ two states or modes: first objects are only translated, any following drag in quick succession only rotates, while scaling was always enabled. This method was designed to prevent the already mentioned problems in the 3D condition, but it showed to be too difficult or unintuitive in the beginning. Participants approved the idea but would have needed an explanation and more training to master that method.
Most participants were not satisfied with the amount of pressure that was needed on the FTIR setup to generate a touch event. Also, occasional tracking errors were negatively mentioned.
An interesting finding is that it is very hard to find an actual application that requires the full 6DoF manipulation of 3D objects. Other research usually present 3D puzzles or object placement. Only Hancock et al. present a real application, a virtual sandtray for psychological therapy [HtCCI10].

Future Work. Hardware supporting force or pressure values for each touch point can be useful. An obvious idea is to influence the weights in equations 5.3.1 to 5.3.3 with regard to the amount of pressure of each finger.
Multi touch hardware and drivers providing shape information can be better employed than just using the center of a shape. Larger areas, e.g. of the palm, can be sampled and represented by multiple springs.
In some applications it will be very useful to combine the abilities of moving of objects with navigation. As an example, touches not hitting any movable objects may be used to zoom or move and rotate the camera.

Conclusion. Our contribution for 3D object manipulation on multi touch screens is a uniform approach, using a mass-spring model on a multi touch setup. In addition, we add scaling in a consistent way. For specific applications it is easily extensible or tweakable, e.g. by adding attraction forces, gravity or collision detection. Benefits are a more intuitive interface needing less explanations, a more stable behavior in case of finger tracking errors and a more enjoyable experience due to inertia.
A disadvantage compared to the gesture based method is, that finger drags influence all transformation parameters at the same time, sometimes causing necessary corrections. However, the advantages of the gesture based method can be realized in the physically based system

with modes and constraints, at the cost of intuition. We advocate against today's common practice of using the number of fingers for mode selection, as it is unintuitive and may be less robust.

5.3.2 Navigation and Travel on Touch Screens

We have used a system for indirect navigation by dragging a finger on a 2D map, thereby translating the camera of a 3D scene visible in a second window. An additional finger allows to rotate the camera by setting the view direction. Jung et al. describe an early version of the system in [JKB*08]. This approach is made to jump or teleport at a specific position on a map. However, we noticed that many users try to smoothly fly through the scene. With this approach some training is required, as interaction and 3D scene display are dislocated, not using the abilities provided by touch screens. At least, the current camera position should be rendered in the map view as a feedback. A better suited approach for this scenario is described by Edelmann et al. [EFS09], where one finger dragging in the 3D view rotates the camera without rolling and two fingers allow a 3D translation. However, to continuously move forward, the user needs a lot of physical hand motion, i.e. repeatedly perform a *zoom gesture*. For such an application, we expect that a speed control would work better, similar to how a joystick is often used for navigation. Dragging a finger up and leaving it there will result in a continuous forward motion. With a second finger, the other two translational degrees of freedom could be modified. We have not implemented this idea yet. Of course, a combination of the map based and direct method makes sense.

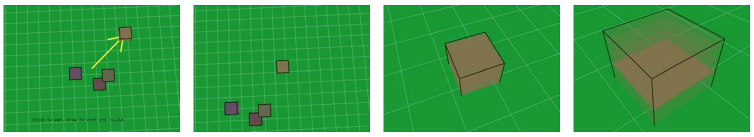


Figure 5.15: Campus Information System navigation prototype. In this sequence, the user taps on the location of interest and releases the finger, then drags the finger down, zooming in and eventually revealing the bottom floor of the building.

For a campus information system (see section 6.2.1.1), we designed a novel way for interacting with a 2.5D map. The selected hardware in 2005 only supports a single touch point. To still be able to zoom in and rotate the campus map, an interaction prototype was implemented (see images below). With a finger tap, the map smoothly pans to the center of the selected position, also showing an arrow as visual feedback (left image). A drag allows to zoom in (up/down) and rotate around the up axis with the current screen center as origin (left/right). When zooming in closer, the camera changes the pitch angle from a top view more to a side view (both center images). Further motion stops the zoom and peels the floors of the building from top to bottom (right image). This type of interaction allows for a direct navigation to a special floor of a campus building with one tap and one drag.

5.3.3 Gesture and Handwriting Recognition

In some applications it can be useful to use zoom and pan gestures to control the view and manipulate objects, while also allowing a handwritten character recognition. An example application was implemented, enabling to furnish a room on a map view. Drawing a previously trained stroke inserts a new object, while also allowing zooming and panning the map or moving and rotating the objects. Details can be found in the bachelor's thesis of Georg Kapeller [Kap09].

5.3.4 Public Touch Interface Design - Lessons Learned

We have noticed, that the best way to introduce a user to a new user interface, is to demonstrate it first and then let the user try, encouraging or helping out if necessary.
In an unattended setup however, the task is more complicated. One idea is to show a video of someone using the interface. For our touch screens, when idle, we show a moving hand that shows possible interactions, both indicating that it is an interactive and touchable screen at all, as well as motivating passers-by to also try the system. A clear feedback of possible or unexpected interactions is important to keep the user frustration level low. A difficult problem is, that many touch technologies exist that all have a different way of sensing objects. To use the display in an optimal way, the user should be informed what to do. This is especially true for using one or multiple fingers or the flat hand, or the necessary amount of pressure required. Feedback on constraints is very important. If the user performs an action that is not allowed, e.g. moving an object outside a specific area, the user may not be aware of the constraint, and without additional feedback he may be left in the dark why the action is not performed. Instead, some feedback should indicate that the desired action is recognized but not allowed. An example is our 2D object translation that builds up more and more resistance and moves the object back to a valid position when releasing the fingers. Another example is *Ripples* by Wigdor et al. [WWC*09], showing a trembling line to indicate a strong present force.

5.4 MISCELLANEOUS

5.4.1 IO Device Library

While the Davelib allows rendering of 3D scenes on many VR setups, the remaining issue for portability is input device handling. To solve the problem, we designed the *IO Device Library* which is is still in an experimental stage.
The idea is that a Lua script is responsible to connect the input and output devices with the program logic. It can be adapted to each setup specific device configuration. With the help of the LuaSocket module, the script can interface with both local and remote devices via network on its own, without any help of the main C++ application. Devices that do not already provide a network interface need a relay server to send and receive data via the network. We recommend an abstract way, trying to use the intended meaning rather than the direct input values. An example is a webcam that should be used as motion detector. A program running on the machine where the camera is connected computes the amount of motion and sends this value via network to the Lua script. Another example is a car control

as illustrated in the figure below. The developer must take care of exposing application states or functions. Important Davelib states should also be accessible by the script for a flexible control, mainly the tracking and projection matrices.

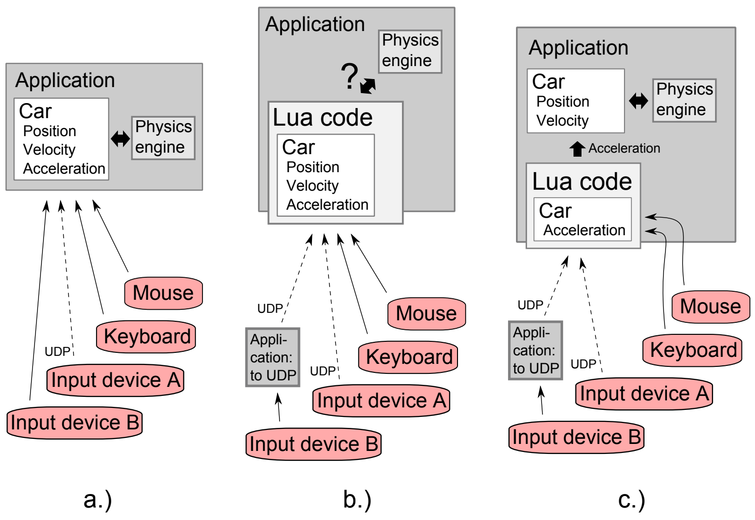


Figure 5.16: IO device library design considerations with the example of a car control. a) A common approach where input device support is hard coded in the application. b) A Lua script is used to process input devices and to compute the new position of the car. This complete control by the script may not always work well, collision detection and physics are probably better handled in the main application. c) Suggested method, where only the object parameters like the car acceleration are set by the script.

Note that we suggest to treat mouse and keyboard input in a special way for the following two reasons: First, an application running on a single PC should not require other programs just to feed it with the basic input options, and second, such programs globally capture keyboard input and could easily serve as key loggers. The disadvantage of this approach with special treatment is that different frameworks like GLUT or SDL have different key mapping definitions that need to be transformed into a common definition for the Lua script. This is necessary, as it makes sense to use the same script e.g. for a common navigation among several programs that may use different frameworks. Following this example, all Davelib and OpenSG programs in the DAVE using a flight navigation could use the same script.
Another advantage of using scripting is a much lower inhibition threshold for a new developer to start adapting the software. Finding and changing a few lines in a short script is much easier than recompiling the whole application. A disadvantage may be a less comfortable debugging during development.

5.4.2 System Control

To start and stop programs, the user interface is the most visible, but it is just a small part. It is not easy to make the underlying system work well in practice. However, it often seems to be taken as granted and not relevant for research. We use a small wireless handheld device in combination with a web browser interface for system control (see Figure 2.26). At the moment, Python scripts work in the background to execute the respective commands.

Remote Configuration. To ease configuration and development at the HEyeWall, a mouse and keyboard forwarding application is used, allowing to move the mouse cursor over the whole tiled display. When the cursor is moved across a border it jumps to the respective position on the next computer. Only the computer currently showing the mouse pointer receives keyboard input. For the DAVE this idea is less useful as not all screens are visible from the controlling PC. Instead, a remote connection must be opened for each of the eight PCs.

5.4.2.1 Peripheral Device Control

Turning the DAVE on consists of switching a physical power switch for the tracking system LEDs and the sound amplifier, and clicking on a link on a web interface to switch on the projectors and the tracking system. A set of Python scripts is used to execute respective commands. When using the DAVE, a volume control for the applications is useful. We wrote a command line tool to get and set the current master volume in percent by a string. We use this to scale an image representing a volume bar on the web interface, showing the current volume. This image as well as an empty bar on the right side are implemented as server side image maps, so that a click at the desired position at the bar sets the volume. The modified projectors can be controlled via a serial interface that can also be used via the web interface. It is important to be able to send control commands from within the DAVE to correct a wrong projector state, sometimes needing to read the on screen information.

5.4.2.2 Starting and Stopping Applications

To run applications and calibration programs in a VR environment like the DAVE, we use Python and batch or shell scripts. Over the years we had several different script setups, all with different drawbacks. The current scripts provide functionality to look at the console output of the programs, which is very useful for debugging or development. However, one of the scripts needs to be modified each time a new application is added. With a single error in the script, the whole system breaks. In an additional html file, the link is added to start the application. A web server, Python and Linux or *Cygwin* are the requirements. We are in the process of writing an application control system without these requirements and drawbacks. The requirements are listed below.

Requirements. Different setups for local and remote control on different operating systems should be supported, ideally without an installation procedure. Examples are the DAVE, the HEyeWall, the rear projection wall, the multi touch tables and a demo CD for desktop setups, that can be distributed e.g. for advertisement. This includes possibilities to start different tracking systems or switch the display to a special stereo mode, depending on what is required by each individual applications. Also some applications require environment variables to be set. In some cases, multiple programs must be launched in the correct order or with a delay. A very useful feature is a logging mechanism to somehow be able to see console outputs of the program, that often contain valuable debugging information. In many setups, we only use one application as the main application that should be given all resources, like the graphics hardware. Thus, all other applications that could take up such resources should be stopped automatically upon starting the next application. For some applications, the same program must be executed on each PC, while for others like OpenSG, server and client applications have different names and parameters.

Implementation. We have already partially implemented such an application, the *VRAppStarter*. It acts as a web server for a universal way to send the control commands. We plan to use the same configuration file for both specifying location and startup parameters and arguments as well as for generating a graphical interface with a website automatically. In that way, to add a new application, only a single file needs to be modified. Our new approach is more user friendly, because unknown commands are ignored and an error can only affect a single parameter or application. By running an instance on each machine, remote execution is possible. Starting an application on one machine makes it possibly to relay the command to a cluster, where the same or a different executable command may be executed.

Useful Ideas for Future Work. Another useful feature for operators is an always visible status display of the system components. A number of small green dots is enough to indicate that everything works as expected. Each machine could check all its crucial devices and update its status every few seconds. In case of the DAVE, this includes whether all machines can be reached via network, if the sound card and joystick are detected and if the tracking server and the file server are running.

5.4.2.3 Unattended Kiosk Systems

Special considerations must be taken into account for devices and software that shall run 24/7 in a public space. The hardware and operating system should be capable of automatically starting up after a power cut and even reboot in case of a failure. For our experimental purposes, we use standard Windows, which is not well suited for such setups. To prevent update and popup windows appearing, we set our application window to the topmost window order. The access to the PC is physically blocked and we restrict touch input to just work with the desired software, not with the operating system. We implemented *Watchwolf*, a watch dog application for the purpose of surveying the application status. The application must regularly report to Watchwolf, otherwise it is assumed that the program is crashed. In that case it is killed and then restarted automatically. Also, the current status can be send by email. A final option is to automatically reboot the PC either at night or when repeatedly detecting a problem.
For the user, a good feedback is essential, as there is usually nobody around who could be asked. Bad examples are far too often present. Examples are slow or long processing or network access or update of the screen without user feedback, that e.g. a click was actually recognized and is being processed. It is also relevant how important the system is for the user. If an advertisement system does not work well or is out of order, this is often more acceptable than consequences

of a failing information system or ticket vending machine. Too often, vending machines serve as negative examples, not allowing to go back one step, not providing a good explanation, or imposing unclear limitations on the type of accepted payment. Especially if there is no alternative to the machine, the user frustration level can easily rise. In conclusion, it always should be considered that a kiosk system may fail. As a consequence, important systems often have a telephone hotline for support or to report a broken system.

5.4.3 The Ideal Interface

Which interface is ideal and will be used in the future? Researchers hope that with 3D interaction in a VE, tasks can be accomplished better. However, many aspects show that it is very difficult to actually find a useful application for the suggested solution. There is evidence that with current technology, a 2D interface is more effective for many tasks and that reducing the degrees of freedom often helps [BKL*09]. However, there seems to be the assumption, especially by non-experts, that 3D interfaces are or will be superior. An often mentioned example is the movie "Minority Report", where the main character stands in front of a huge holographic display and interacts by waving around with his arms. Many people having seen that movie seem to assume that once such technology is available, everyone will want to use that kind of interface. A critical view quickly reveals that such an interface seems much less ergonomic than a standard desktop setup. This was even examined scientifically [BIB*09]. Bowman et al. provide an interesting list of suggestions for future research in 3D user interfaces [BF05], [BCF*08].

5.4.4 Usability Studies

To verify that a newly developed interface performs better than existing methods, usability studies are carried out. It is very difficult to design them in order to get representative and useful observations. Even then, often the findings are hard to generalize, i.e. for another application or setup. Also, probably due to simplicity, a travel technique is evaluated e.g. by the task to move along a maze or corridor as quick as possible. If this really was the intended application, users could be greatly assisted for such a task. Instead, test scenarios should resemble closer to the target application. Usually such studies are conducted at one research lab with the available and suitable hardware. There are many variables that are likely to have impact on the results, such as the latency and accuracy of a tracking system and display hardware. The performance may be very dependent of the exact implementation and choice of parameters. A sensible idea to improve the expressiveness of such investigations is to collaborate with multiple research groups at different locations to first develop independent solutions and then collect the best ideas. Finally, the same concept and variations may be tested on all setups of the different locations. However, this requires a large effort. A way to realize a similar idea is a public contest or challenge, as sometimes seen in the context of a conference.

6 Applications

Having discussed all necessary technologies, both our implemented applications as well as concepts and ideas are presented. Not for all applications a fully immersive setup is the best choice. Robertson et al. discuss the advantages of non-immersive Virtual Reality already in 1993 [RCM93]. We thus split the applications into two groups, according to how well they are suited for either non-immersive or immersive setups. Some are just promising conceptual ideas, others are mature applications. The insights and lessons learned are a valuable source of information for designing new applications. A brief overview on the areas of VR applications is given first.

6.1 RELATED WORK

Interactive exploration and visualization of data is already a successful application of VR and is used to gain insight into, understand or verify the data. This may be an important part in a decision process. Annotation, modification or even creation of data are interesting but more challenging options. The following subsections list common areas of application.

6.1.1 Large High Resolution Screens.

Especially for large amounts of data, a large number of available pixels is useful. Applications, where scrolling is often necessary on a desktop setup, may profit from the simultaneous overview of the data while still showing the details.
Examples for such large amounts of data can be found in *medical imaging*, *visual analytics* and *data mining*.

Colocated and Remote Collaboration. A large display may ease collaborative situations compared to a desktop setup. Interaction of multiple users with equal rights may be accomplished e.g. with multi touch systems.
Video conferencing has the potential to save a large amount of time, money and environment pollution. The acceptance of these systems is not very high. Prototypes exist trying to avoid current problems, like enabling eye contact.

Mapping and Charts. Maps, charts and graphs profit from a large number of pixels. In case of static information, printed versions may still be far superior in terms of resolution.
Geographic information systems (GIS) may be visualized. For 3D GIS, AR applications are developed for inspection on site.

Command, Control and Surveillance. Oil and gas, power, homeland security, traffic control and facility control are examples that may require large displays.

Consulting and Sales. For a sales conversation, a multi touch table setup may be useful to show all options and results of options to the client, e.g. for cars and bank contracts. It is a collaborative setting that in addition should convey competence and innovation to the customer.

6.1.2 Immersive Setups.

The CAVE was developed with the hope to find a much better way to visualize scientific 3D data, such as simulation results. Teleconferencing and medical visualization were also among the early ideas but showed to be less successful in immersive setups. At the time, the hope was to better understand abstract data when being immersed in it. Experience shows that virtual scenes representing the real world are much more successful.

Entertainment and Games. Multi screen setups, stereoscopic setups and head tracking are slowly making their way to the gaming market. However, such setups are still only used by a small number of enthusiasts. 3D input devices however have gained a lot of attention with the console market, mainly the *Wiimote* by Nintendo, the *PlayStation Move* controller by Sony and the *kinect* by Microsoft.
Simulators for motor bike and car racing, skiing etc. sometimes come with a motion platform and are installed in some game halls.
Virtual communities exist, even though they are usually not used with immersive setups.

6.1.3 Applications for Both Immersive and Non-Immersive Setups

Some areas of applications may profit from a variety of different advantages on both types of setups, as listed below.

Biology and Chemistry. Bioinformatics, computational biology and computational chemistry, comparative genomics and molecular docking and rendering are used.

Scientific Visualization. Vector data such as flow simulations can be analyzed. Interactive placement of clipping planes or virtual smoke particles help the visualization. Flow visualization from computational fluid dynamics (CFD) [PPF*].
In astronomy and astrophysics or atmospheric modeling and prediction, a 3D visualization may be advantageous.

Architecture and City Planing. Indoor and outdoor architecture as well as urban design are obvious and well suited applications.

Engineering and Manufacturing. Virtual assembly tests and visualization of simulation data are important. Industrial design or vehicle prototypes may be visualized. Assembly and maintenance instructions often employ AR rather than VR technology.

Psychological Experiments. Simulations are used to study humans and human behavior, both for basic research as well as in order to optimize systems and products. An example is the automotive sector, e.g. testing visibility and reachability with new dash board designs.

Therapeutic Applications. Exposure therapy is used to treat phobias or PTSD (post traumatic stress disorder). As an example, veterans can be faced with critical situations of a battle to process traumatic events in a controlled way. VEs are used for rehabilitation and may increase the motivation of patients, e.g. when exercises are hidden in games.

Education and Training. Dangerous or expensive tasks can be trained in a simulator. Medical doctors, safety and military staff,

pilots or even astronauts can use simulators to prepare for the real situation.
Flight simulators are a well known application for such safe and in the long run more economic training.
Mathematical geometry and electromagnetic fields can interactively be explored and manipulated in an immersive setup, hoping to better understand the effects which may otherwise be hard to imagine.

Medical Visualization. Medical visualization and editing may help to prepare, plan and guide operations. Teleoperations allow experts from around the world to perform critical tasks.
A 3D rendering of medical data in a CAVE was mentioned as a possible application. However, experts usually prefer visualization on 2D monitors, often looking at 2D slices of a volume rather than using 3D volume visualization.

Cultural Heritage and Archeology. Current sites of interest may be 3D scanned or past sites may be reconstructed. The inspection and presentation may profit from an immersive setup.

Virtual Tourism. A 3D visualization may be used for travel planing or general information. The next step is virtual tourism, replacing real travel by inspection of sights on the display.

Demonstration and Public Relations. VR and related display technology is often used to show innovation and to present products in a good way.

Arts. An early example of art for a CAVE that was first shown in 1995 is *Crayoland* [Pap98], showing a non photorealistic world with colorful hand drawn textures.

3D Formats. A big issue for 3D applications is still the 3D format to describe the content. There is no single format that is suitable for all kinds of applications. New technological developments should also be reflected by the 3D format. Formats that support many features require more complex processing, so that other formats may be chosen in favor of efficiency. X3D and Collada were developed as universal formats. X3D is an XML format and can easily be extended for special purposes. In that case, at least the compliant parts can be read by other applications and no new format is necessary.

6.2 NON-IMMERSIVE APPLICATIONS

6.2.1 Touch Screen Applications

Touch screens for a PC are a relatively recent way of interaction to most people. Some screens offer support for multiple pointers at a time. A sensible, intuitive and effective use of multi touch input is a challenge.

6.2.1.1 Public Information Systems

We worked on single touch and multi touch screen applications intended for unattended information screens in public. This work was done at the HITLabNZ.

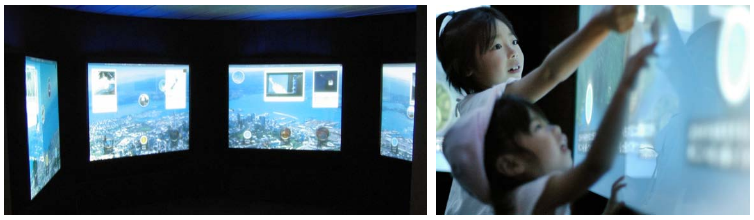


Figure 6.1: New Zealand pavilion at the World Exhibition 2005 in Japan. Left: The inner part consisted of five large multi touch screens. Right: Fascinated users playing with the touch screen. Images courtesy of HIT Lab NZ, University of Canterbury.

Aichi. We have realized several information systems intended for use in public. For the New Zealand pavilion of the World Exhibition 2005 (jap. *Aichi bampaku*) in Japan, we developed a touch screen setup and presentation software providing information on New Zealand. A rear diffused illumination setup is used as described in section 2.3.2.1, unfortunately with the screens spaced further apart than we would have liked.

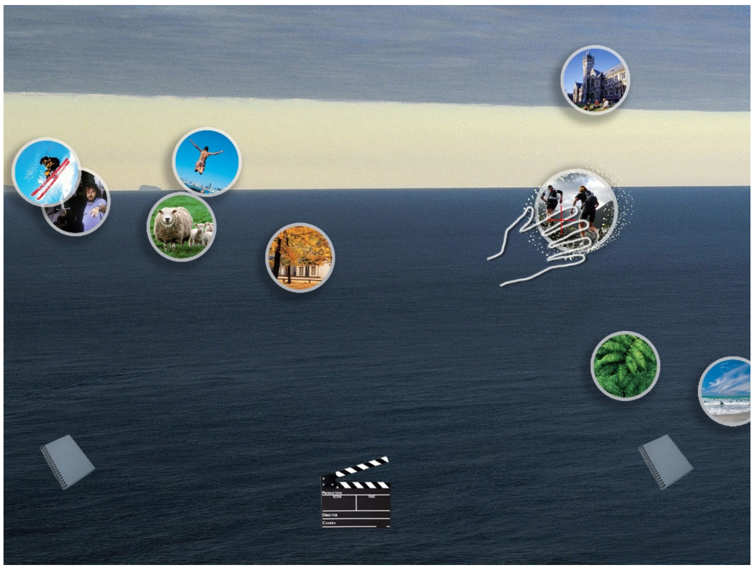


Figure 6.2: Screenshot in idle mode. When nobody touches for some 10 seconds, an animation of a moving hand indicates how and where to drag the icons. The icons move from left to right on curved paths, thereby moving up and down. They have inertia and can be flicked across the screens. Background images changed from time to time, professional media was shown with video clips or slide shows, depending on the tool the icon is dragged to. Image courtesy of HIT Lab NZ, University of Canterbury.

The screens show icons floating from left to right across the linked screens. Users can touch and drag an icon around. Dragging an item to an audio or video symbol triggers a sound or video display. We added inertia for a more enjoyable experience, as well as glitter particles for visual feedback of a recognized touch. A recognized grab or drag is indicated by an animated enlargement of the icon, its change of motion and by repelling forces to other icons. It is possible to push other icons away and off their original trajectory. After a few seconds of inactivity, an artificial hand outline is superimposed, demonstrating how to drag an item to one of the media players. This is used to make clear that the installation is interactive and encourages to directly copy the visual instructions. The icons also move up and down along a wave like trajectory in order to use the whole screen and to attract attention while making sure even small children can reach any icon once in a while.
It may not be obvious that screens are interactive touch screens, there should

be several visible hints encouraging people to interact. A clear touch feedback should be given so that the users know whether the touch is recognized. Objects with physics behavior are enjoyable to use.

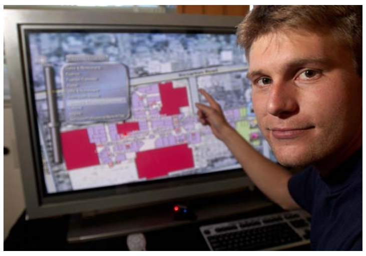


Figure 6.3: Shopping mall information system in 2004. This screen shows a zoomable map with a shop index and life bus data for public transport. Image courtesy of HIT Lab NZ, University of Canterbury.

Mall Information System. We looked at the potentials of electronic mall information systems and installed a test setup in a shopping mall for two weeks. Furthermore, a survey was conducted to help to decide on useful contents on such systems.

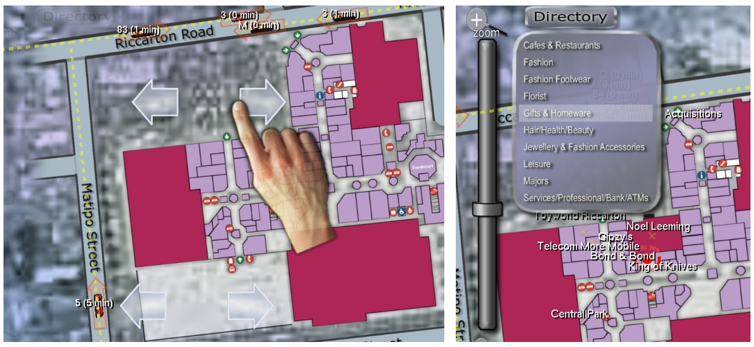


Figure 6.4: Zoomable map with real time bus information. A moving image of a hand illustrates possible interaction and animates passers-by to try themselves (left). Zooming closer, shop names are displayed, also allowing to only show the names of a certain category of shops (right).

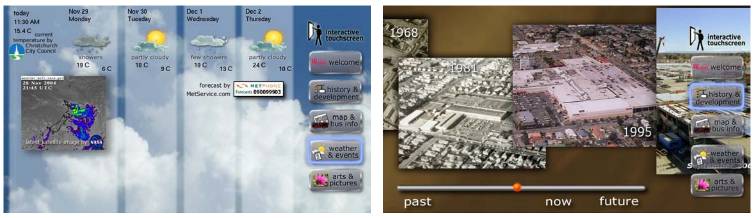


Figure 6.5: Left: The current weather and radar satellite imagery as well as a forecast. The information is automatically updated by downloading and parsing data from three different web servers. Slowly moving clouds are shown in the background. Right: Photos of the historical development of the mall. More photos can be seen by panning the photos or dragging the slider on the time line.

Again, for an idle screen we use an animated photo of a hand showing what to do. In addition, we use a pictogram and the text *touch screen* to show its capabilities. We also use the sparkling glitter particles as a touch feedback, in addition to an auditory signal generated by the hardware. A selected menu button is highlighted with a blue glow. For burn in prevention on the plasma screen, the whole content very slowly and imperceptibly moves around the screen by a few pixels. The content consists of the following pages:

- An interactive zoomable map, featuring realtime bus information, a search function and a shop index.
- Weather information with the current situation, the forecast and a satellite image, collecting the data from three different websites.
- Historic information of the mall, realized by photos arranged at a time line with a slider.
- Fun applications: a Breakout clone and an image warping of a webcam image.

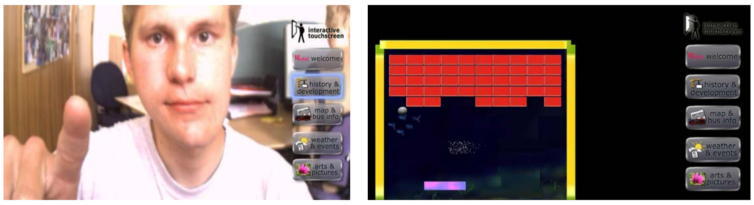


Figure 6.6: Games. On the left, a life stream from a webcam can be distorted, using a deformable mass spring grid that slowly recovers to its original state. The right shows a breakout game. Both games were eventually not used in the mall, as kids may have blocked the screen by playing the games.

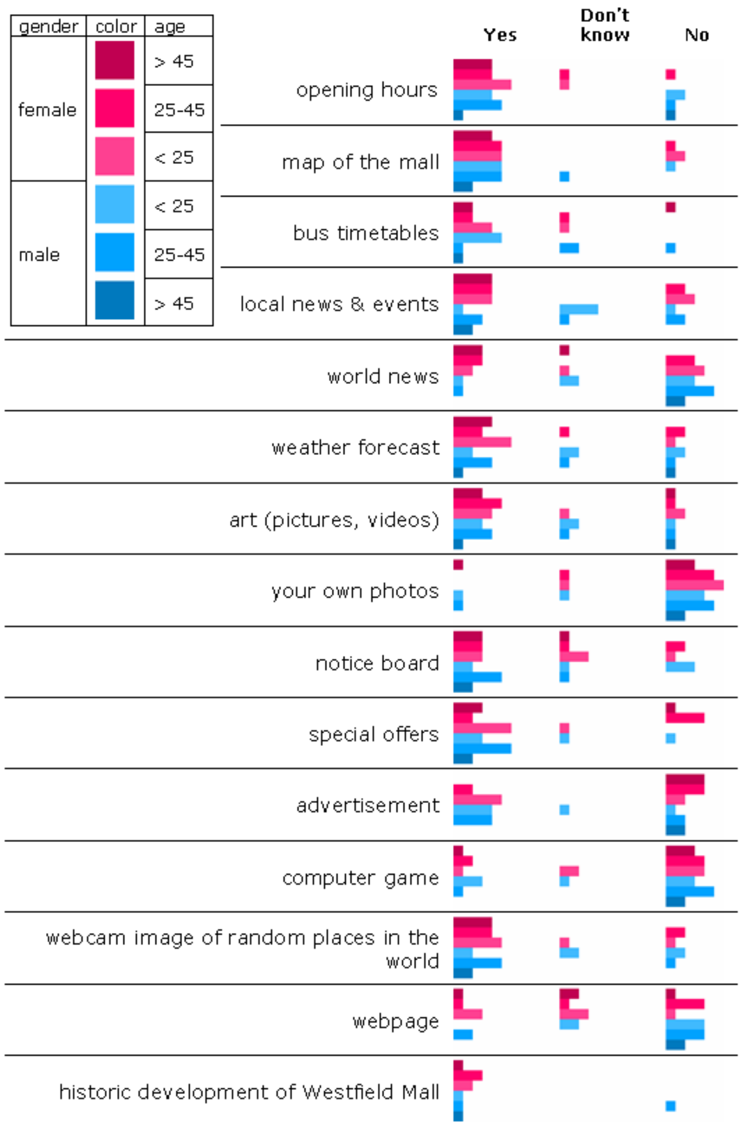


Figure 6.8: Results of a survey in the mall, asking people if they would like to see the respective content on the information screen.

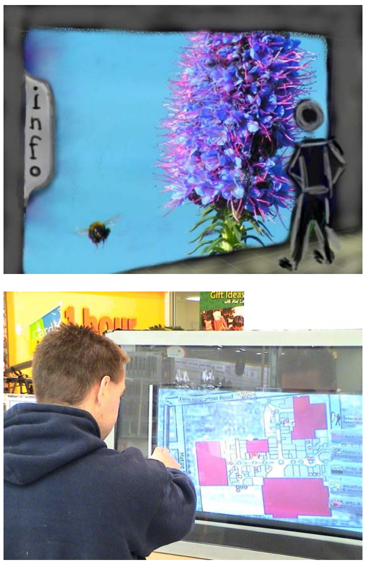


Figure 6.7: An early concept with a large screen (top) and the actual installation with a much smaller screen (bottom).

Results and insights from the survey in 2005 are, that people did not expect the screen to be an interactive a touch screen, older people think they cannot use it, and in general people do not expect to find useful information on such a screen. Nowadays, many people have powerful personal mobile devices and thus, this concept of information screens looses importance for some of the provided information.

Campus Information System. We also thought of installing a campus information system with an interactive map as the main component. The motivation is that compared to a static, printed map, the screen has the potential to always show up to date information, more meta data and an event calendar.

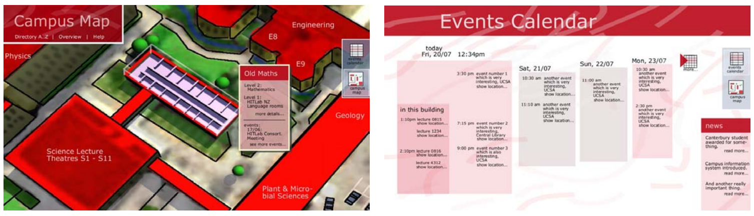


Figure 6.9: Campus Information System sketches.

6.2.1.2 Multi Touch Photo Table

A well suited demo application for multi touch screens is a photo table, where objects can be moved, rotated and scaled. Also, other multimedia objects like videos or PDF files or any application window can be used. Rotating the content is especially important for a table setup, with users standing on any side of the table. The algorithms for interaction are described in section 5.3.1.

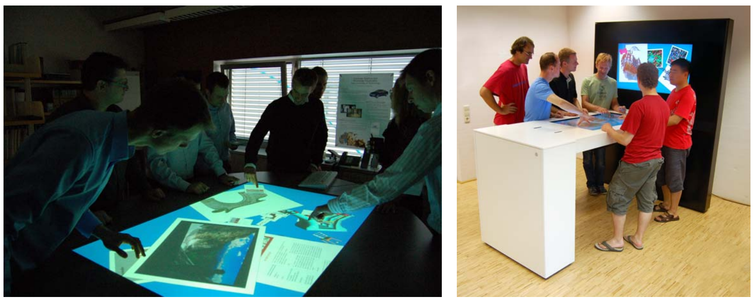


Figure 6.10: Our 2D and 3D object manipulation demo software on two different multi touch tables.

Rotating Arbitrary Windows. As today's operating systems do not support a rotation of any application window (except for the Compiz window compositor for Linux), we partly implemented a concept using a VNC or RDP connection to a remote application [USOF09]. Thus, we render the application window content as a texture within our own application. Also, it is possible to transform the touch input and forward it to the application, transforming the input coordinates accordingly. We have not completed the implementation for remote application windows yet.

Hardware related sensing errors and tracking errors lead to unwanted effects. Users may not use the screen correctly, e.g. touching with the whole hand when a single finger should be used. Soft constraints are helpful for users to understand that an action is recognized but not permitted. The photo table application can be used to both play around as well as demonstrate a useful application.

6.2.1.3 Interactive Floor Projection in a Public Space

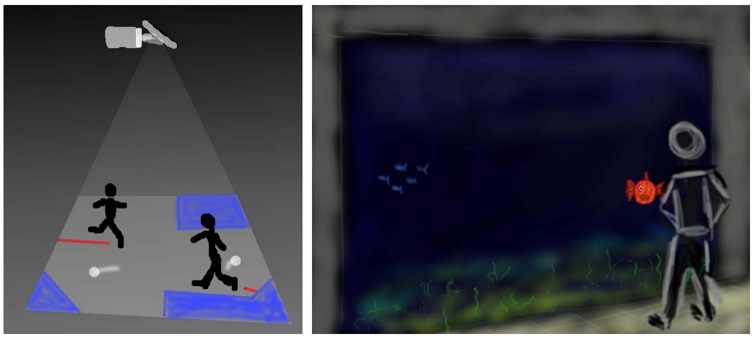


Figure 6.11: Left: Sketch of a game using a floor projection. Players try to reduce the free gray area, in that balls can move, by walking over the map and thereby building walls. Unfinished walls (red) are broken by the balls and the player may be penalized. Right: Sketch of an interactive, virtual aquarium. The fish may react on passers-by by following, inspecting or fleeing from them.

We had the idea for an interactive game with a floor projection, which has not been implemented yet (left image above). The same principle is now used in advertisement. When somebody walks over an image, it reacts with some image effect, e.g. by burning away to show the next image, or with showing water ripples.

6.2.2 Computational Photography

A relatively new name for certain applications of digital photography is the field of *Computational Photography*. We mostly use the term for applications where multiple photos with different settings are combined to overcome hardware restrictions of a camera. This includes the lens, the aperture, the imaging chip but also the camera position. Most applications in this field would greatly benefit from direct camera control, but on commercial photo cameras, options for customization of capture processes are quite limited. However, there is a small community that reverse engineers firmware by experimentation and analyzing updates. The *Canon Hack Development Kit (CHDK)* provides control for many operations. It even allows scripting and is available for many of the point and shoot models by the company. A more scientific project is the *Frankencamera* [ATP*10], a mobile camera hardware platform with an open access, especially made to enable and encourage research in Computational Photography.

6.2.2.1 Image Alignment

A common application are photo stacks. They can be used for high dynamic range (HDR) images, focus stacks or noise reduction. While such stacks require exactly aligned images and are usually supposed to be captured with a tripod, this requirement is a major drawback for the practical usage, as most photographers do not always carry a tripod with them. For handheld panoramas or exposure brackets, multiple images should be overlayed so that in overlapping regions, pixels fit perfectly. Due to camera movement, when shooting without a tripod and a panoramic head allowing to rotate the camera around the nodal point, no simple and perfect alignment is possible. In addition, with slightly moving objects like tree branches or clouds, visible ghosting artifacts may result. With a following automatic photometric estimation and correction, this may even lead to further errors in the processing chain. Thus, we looked at possibilities to warp image stacks acquired with a handheld camera to produce aligned images for subsequent image stack algorithms.

Automatic Optical Flow Based Image Alignment. Following the idea of Kang et al. [KUWS03], we also tested automatic image alignment using optical flow. Small misalignments in textured areas can be corrected well using optical flow. However, erroneous matches may also be produced, especially in low textured regions, leading to visible artifacts. Further constraints are necessary for smooth results also allowing sharp edges at depth discontinuities. Such regularization may be achieved with the TV-L^1 energy functional, as described in [ZPB07] by Zach et al. We have not implemented this yet.

Figure 6.12: Panoramic stitching with a slight camera motion. Without correction, ghosting artifacts appear in the grass area (left image). The optical flow is computed (center image) and used to warp one image to eliminate ghosting (right image).

Larger scene changes are not handled very well and lead to visible artifacts. In the example below, local rotation occurs. To correct this, rotation should be included in the model that is used for regularization.

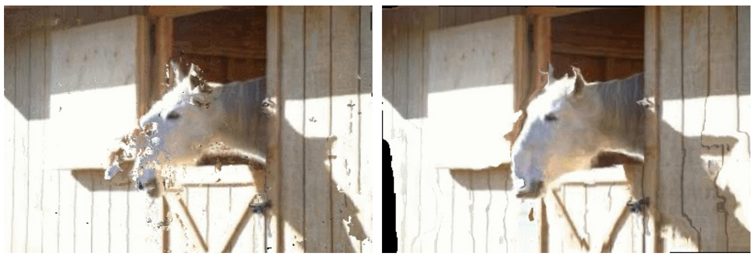


Figure 6.13: Source images or blending without warp!! To align exposure bracketed images for high dynamic range fusion, our simple local model is not good enough in this case. The horse head rotated significantly and errors occur with both of our approaches (photos by [Sze06]).

A different application using the same optical flow computation is to produce a motion blurred image from two sharp input images. This is useful for stop motion animations or for animated renderings, where physically correct computation motion blur increases the rendering time by a large amount.
More details can be found in the master project of Markus Rum-

pler [Rum09].
Automatic local image alignment works well for most parts of the image, when using images with a similar dynamic range. To avoid very visible outliers, a regularization is necessary. Exposure stacks are not well suited for automatic optical flow based alignment.

Manual Image Alignment with Freeform Deformation. To avoid problems that occur with the automatic approach described above, we also implemented several free-form deformation algorithms and an editor to manually align image stacks.
Currently, only a simple planar projection is used. For panoramas, a spherical projection should be implemented. A radial lens distortion is included, allowing us to calibrate and reuse lenses. For alignment, the image opacity can be changed, the colors of an image can be inverted and an edge filter can be applied. These visualization options greatly simplify alignment of differently exposed images. In contrast to existing solutions, the program shows the deformation in real time, a simple mouse drag adds a new control point and the result is immediately visible, allowing a very fast registration. The most important points for future improvements are an automatic fine tuning of control points as well as a spherical projection for panoramas.

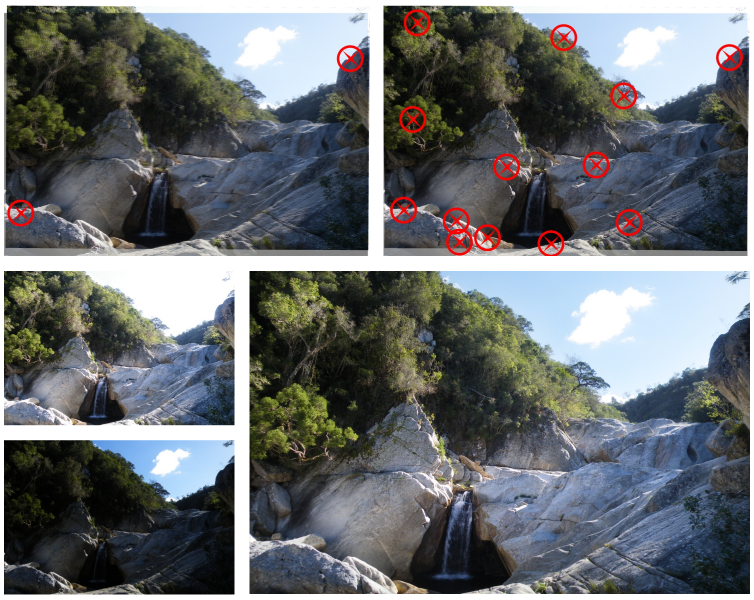


Figure 6.14: Photo Distorter. Top row: Rotation, translation and scale are adjusted, defined by two control points. Some ghosting remains (left) vs. free-form deformation with more control points (right). The two small images on the bottom left are the original input images, the bottom right shows the final tone mapped result using the free-form alignment.

The tool is available as open source under the name PhotoDistorter. More details can be found in the bachelor's thesis of Georg Marius [Mar10].
A real time feedback of the image alignment is valuable and allows a fast manual registration and simultaneous quality control.

6.2.2.2 Motion Blur Magnification

Another idea for using multiple aligned images is what we call *motion blur magnification*. An example is a photograph in daylight, taken with a point and shoot camera that cannot close the aperture far enough for a desired long time exposure. Instead, multiple photos or a video can be taken even without a tripod and the motion can be analyzed to then blur the reference image along the local motion direction. In that way, an effect similar to a long time exposure can be achieved. This is only a concept, we have not implemented it yet. However, to show the idea, we prepared the following mockup images.

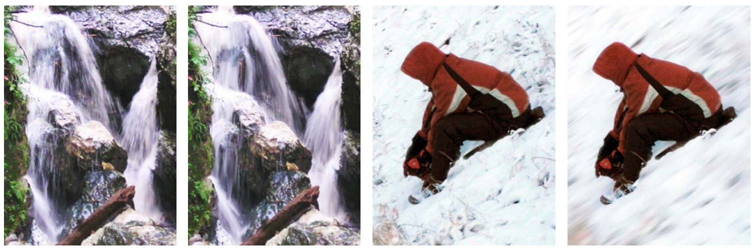


Figure 6.15: Mockup images demonstrating the idea of Motion Blur Magnification. The left image of each pair is the original image, the right image shows the possible result when combining multiple shots to reproduce an similar effect as a long time exposed image.

6.2.2.3 Face Distortion Correction

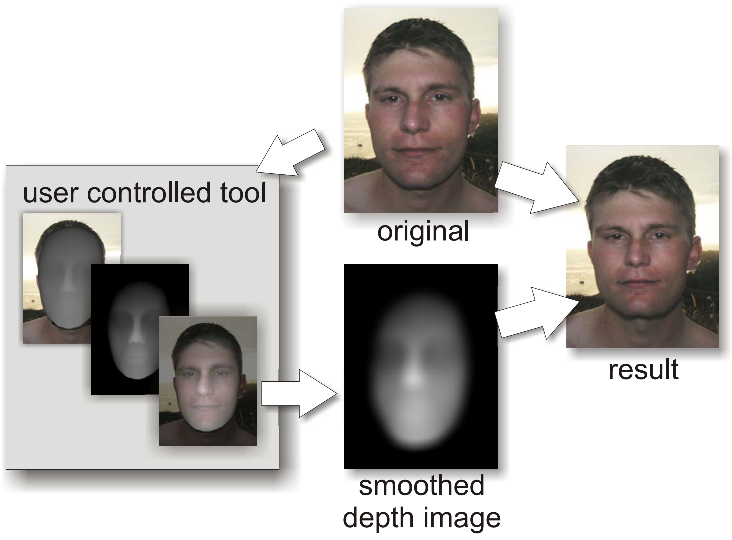


Figure 6.16: Face Distortion Correction for unflattering close up shots.

For close up photos of a face, well known features like the nose appear in an unfavorable way distorted due to perspective. A photographer can usually step back and choose a better suited focal length to not run into this problem. However, web cams or mobile cameras for video calls are usually placed relatively close to the face. To mostly correct this effect in software, we suggest the following system.
A face detection fits a 3D head to the image so that with a known focal length of the lens an estimated depth is obtained. The depth map is blurred and its magnitude and gradient is used to warp the image. We think that especially mobile video calls may benefit from this idea, as mobile devices are usually hand held at a comfortable distance, resulting in notable distortions. With face tracking an automatic distortion correction may be applied. Our prototype implementation uses a manual registration of the 3D head model.

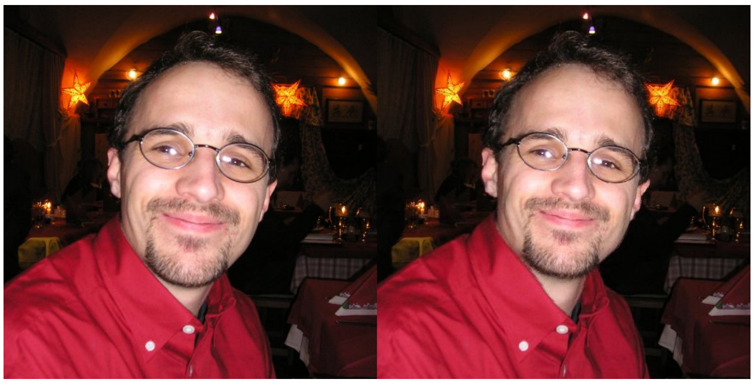


Figure 6.17: Original photo (left) and corrected version (right). Note especially the size of the nose.

6.2.2.4 Zoomable Images

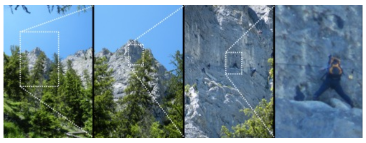


Figure 6.18: A common way of visualizing large zoom amounts, especially useful for static images like print media.

Nowadays it is easily possible to interactively display gigapixel images. Zooming by a large factor and panning the image is an important application, considering that computer displays can only show a small fraction of the data at the same time. For the application of photographically acquired images, the creation requires the combination of many tiles and is still a long or expensive process. On the other hand, the increasing optical zoom ranges of consumer cameras of over 30× nowadays allow consumers to easily take high resolution zoomed in inset photos together with a wide angle photo of the same scene. Often, there are some obvious points of interest like a distant village or mountain top. It is not so important to acquire the whole image in a high resolution. Only little work has been done to appropriately display the high resolution photo as an inset. Automatic stitching is possible, but the transition from high resolution content to low resolution content may be annoyingly apparent, especially when the resolution difference is high. When gradually alpha blending over a large area, the transition is less visible, but a lot of the high resolution information is lost and ghosting artifacts may result. To overcome these problems, we introduce a different, novel approach by gradually attenuating the maximum image frequency. We achieve this with a Gaussian blur filter with an exponentially increasing standard deviation across the transition.

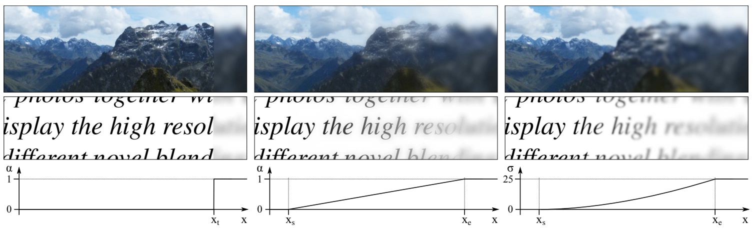


Figure 6.19: Two sample images with transitions from the high to the low resolution content. From left to right: hard transition, standard linear blending (alpha blending) and our method. The plots below show the alpha values for blending in the left and center and the standard deviation (in pixels) used for the Gaussian blur on the right. The images have a width of 1024 pixels.

Alignment and Detail Generation. At the moment a manual alignment of the images is used. An automatic alignment is possible, an approximate scale can be extracted from the focal length stored in the EXIF information of the photos. We avoid possible problems with different exposure or white balance by replacing the low frequency content of the high resolution image by the low resolution image. We do this by blurring both images with a large radius and then adding the difference to the high resolution image. With the steps described above, we can handle hand held photos with different exposure and white balance settings.
To enlarge the area with high resolution information, we implemented a simple detail transfer (see Figure 6.22). We compute a good cross correlation based match of random position pairs and copy high resolution patches. However, we recommend to use a more sophisticated approach like the one described in [FJP02] by Freeman et al. to get more plausible results.

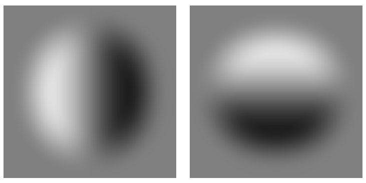


Figure 6.20: Gabor convolution kernels used for local frequency analysis.

Transition. To help understand the present image frequencies across a transition, we looked at sine kernel responses with varying frequencies (see below). As kernel we use the sine function multiplied with a Tukey window function. Similarly, we generate a rotated version by 90 degrees, resulting in the *Gabor* kernels shown on the side. For each pixel in each row in the image, we add the absolute values of the convolutions with each kernel. The figure below shows the input images in the first row and the result in the second row, with kernel dimensions of $\lceil\lambda\rceil \times \lceil\lambda\rceil$. For the example images, we use a high resolution image and generate a low resolution version by applying a large Gaussian blur. Note that this is an approximation, as common upscaling algorithms will produce slightly different results.

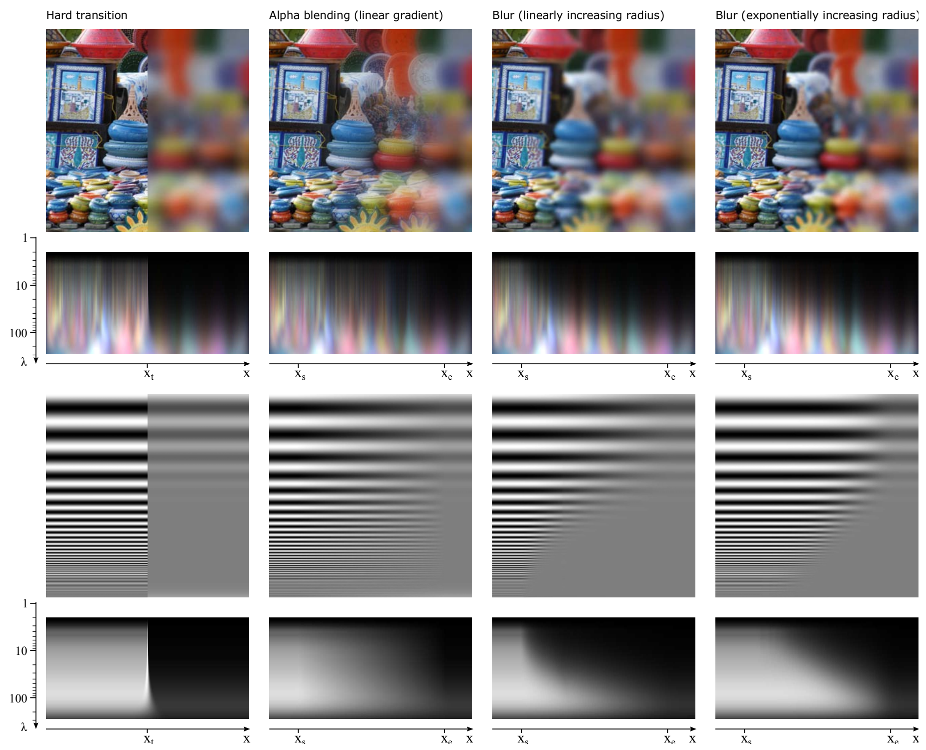


Figure 6.21: Frequency analysis for the market image and for an image with a sine with exponentially increasing frequency. The rightmost column shows our suggested method described below.

Transition with Alpha Blending. As expected, the linear alpha blending frequency response in the figure above shows, that high frequencies are present almost across the entire transition and then quickly drop to zero on the right side, leading to a relatively hard transition in the frequency domain. Especially in the middle of a gradual transition, ghosting appears, i.e. edges look foggy.

Transition with Frequency Blending. Our approach is to gradually suppress high frequency details across the transition. We use a Gaussian blur with an exponentially growing standard deviation (with a kernel radius of 2.5 times the standard deviation). We first tried a linearly growing radius and found, that the blur increases too sudden and high frequencies present drop to quickly at the start (left side) of the transition. We found that an exponentially growing standard deviation works better (see both right columns in the figure above).

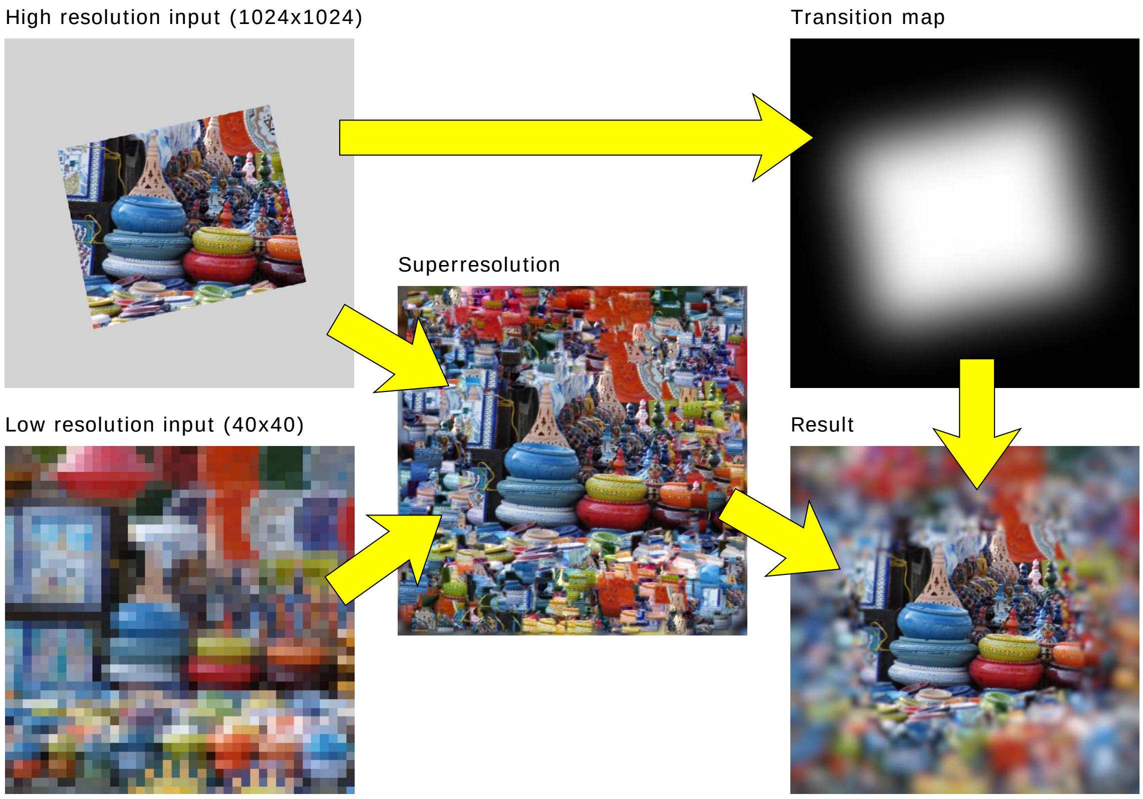

High resolution input (1024x1024)
Transition map
Superresolution
Low resolution input (40x40)
Result

Figure 6.22: Transition generation workflow. Aligned and photometrically corrected low resolution and high resolution images are the input images on the left. The center shows and intermediate result where high resolution patches have been copied to areas with similar low frequency content. The transition map is derived from the blurred alpha mask of the high resolution input image and controls the amount of blur. The final result is shown on the bottom right.

For our images in sRGB space we approximately linearize with a gamma value of 2.2 and transform the result back appropriately. To compute the required standard deviation of the Gaussian from the magnification factor, we found that a factor of 0.5 produces a blur that is similar to the magnified low resolution content, i.e. $\sigma = 0.5 \cdot magnification$.
Realtime visualization software often uses hardware magnification with linear interpolation. For a large magnification, the transition between the Gaussian blurred image and the linearly interpolated

image will be slightly visible. Also, the high resolution inset is in general not aligned to the low resolution coarse pixel grid. To avoid a visible seam, we alpha blend the final result with a smooth gradient over a small distance.

Comparison to Irregular Masks. In the photo zoom project, Eisemann et al. suggest to use irregular masks [EESM10]. The idea is to get a sharp transition at object boundaries and hence avoid the ghosting artifacts. We implemented an algorithm producing similar irregular masks to further analyze this solution. We also use a cost function which is a constant plus a weighted edge value. The edge value is computed from the maximum absolute sobel egde value of each color channel, setting low edge values below a threshold to zero. We manually tuned the parameters to get a good mask. The figures below show the generated mask and zoomed crops respectively. While the idea works well in the cropped region in the figure below, the cropped regions in Figure 6.24 show ghosting at the cars and a visible edge caused by a roof on the high resolution region boundary.

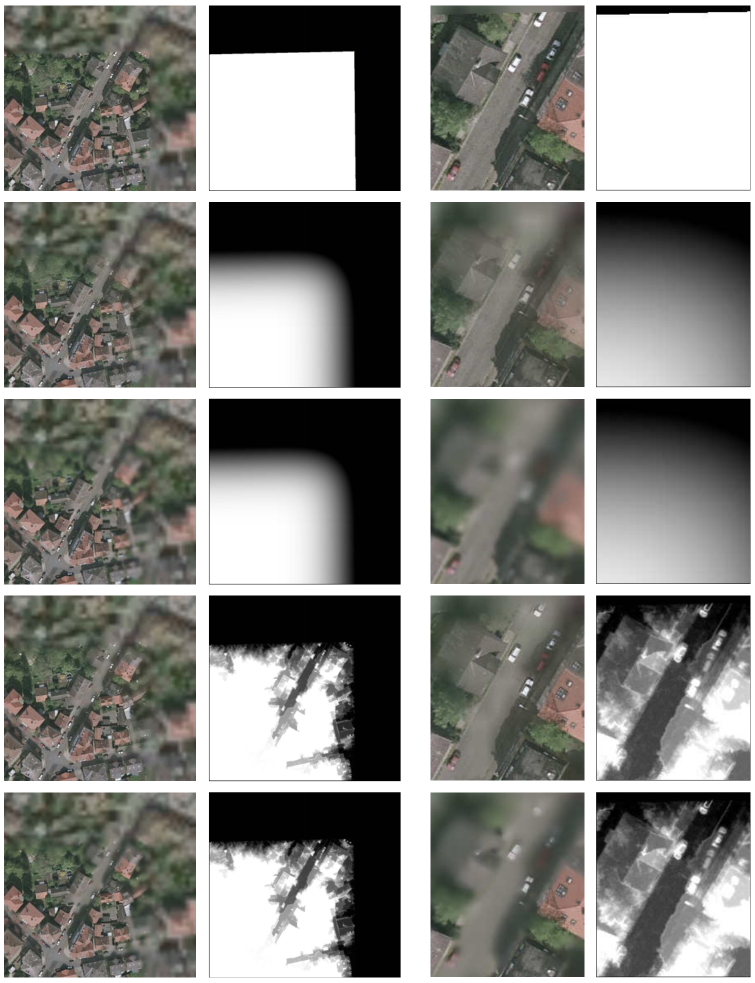


Figure 6.23: Aerial image of a city. The left columns shows the complete image with the respective maps, the right columns show a zoomed region close to the center. From top to bottom: hard transition, linear blending with smooth transition, our method with exponentially increasing blur, linear blending with irregular mask, blur with irregular map. The image is provided by the city of Braunschweig.

We also tried to use the same mask as an input to our algorithm. To avoid blurring with a large kernel over masked out details, we could multiply the Gaussian weight with the mask. Instead, we used the GIMP Focus Blur plugin which behaves similarly. While the results do not look very convincing, it may help at a coarser level to hide the regular, usually rectangular border of the high resolution content.

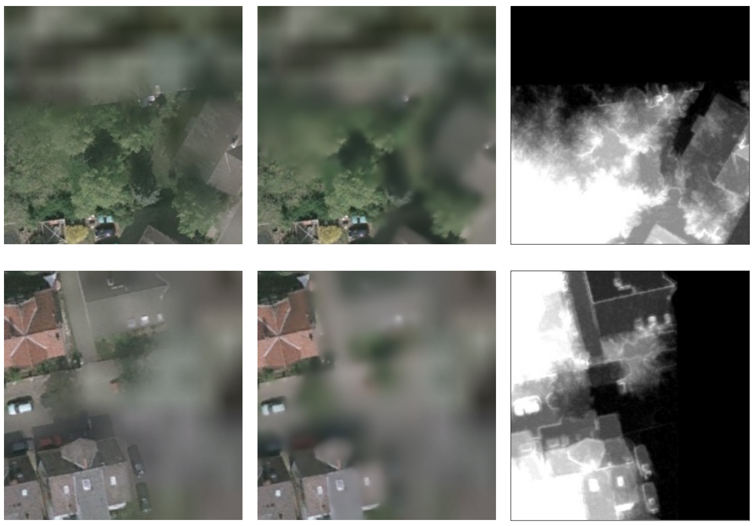


Figure 6.24: Two more closeup views, comparing irregular masks/maps (right) for linear blending (left) and blur (center). Artifacts are visible in all cases. Top left: hard edge of a roof. Top center: some trees are only partially very blurry. Bottom left: cars at the bottom suffer from ghosting. Bottom center: only some parts of the house are very blurry.

We successfully avoid ghosting artifacts completely. The image below shows an interactive example with two blended high resolution insets. The first has a zoom factor of 5.6, the second has an additional zoom factor of 3.3 (18.4 in total, i.e. 18×18 high resolution pixels correspond to one low resolution pixel). As the images are scaled down but still have about 1.5 megapixels, this is only visible at a high magnification, please use the electronic version of this document and zoom in. Common current PDF viewers seem to only interpolate using the nearest neighbor, leading to pixel artifacts at high magnification levels. Even when explicitly searching for the transition, it is not easy to find. For reference and to ease inspection, the right image visualizes the position of the three images by inverting the color of the middle image.
We also display our zoomable images on our multi touch setups (see section 6.2.1.2). To prevent images accidentally being scaled too large, we use a maximum scale factor. We extend this by adapting the maximum scale factor to the dimensions of the smallest zoom region, thus allowing to zoom in far enough.

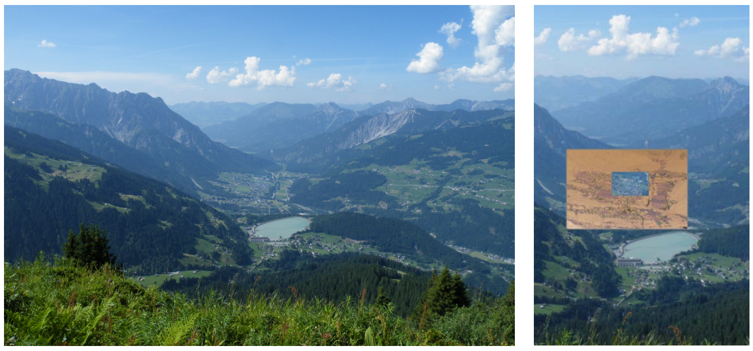


Figure 6.25: Interactive zoomable image. The location of the high resolution images is shown in the crop on the right. Note that for better compression and unknown magnification algorithm of your PDF viewer, in this example we do not blend the outer border.

We have not yet considered multiple partially overlapping images with different resolutions.
For more details, please refer to our publication [LF11a].
As much high resolution information is lost for gradual transitions, the high resolution region should be extended before with super resolution approaches.

6.2.3 Visualization and Rendering

6.2.3.1 Preview Visualization for 3D Scanner

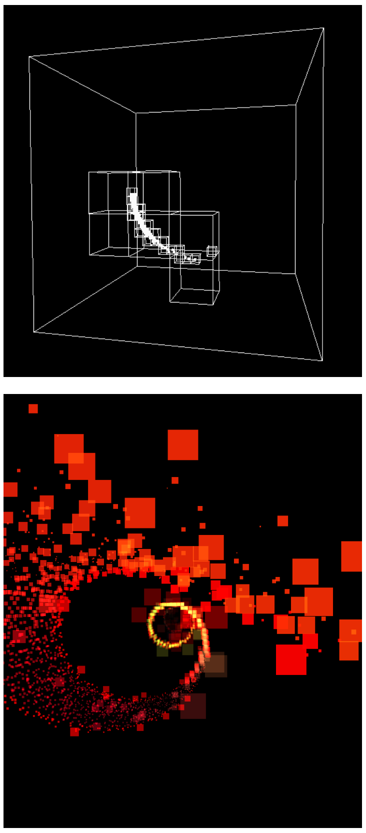


Figure 6.26: Top: Dynamic octree with bounding volumes of different LoDs. Bottom: a test with colored particles.

For the *scene scanner* by *IRL*, we conducted early research in 2005. The 6 DoF tracked handheld 3D scanner is equipped with a laser line scanner for depth and a color camera for texture. By swiping the scanner over static objects, they are scanned from that point of view. In a postprocessing step, all recorded data is transformed into a high quality point cloud or surface. However, during scanning, a fast preview is useful so that the operator can see which parts of the object are already scanned and which are still missing. Also, visual feedback on the quality or resolution of the surface is practical. The research problem is to find a way to show a preview with interactive frame rates, even after many minutes of scanning. Also, the scanning volume is not known in advance and should adapt to the measured points.
Our solution is a dynamic octree with a constant maximum number of points. It can expand its size dynamically when necessary. Also, resolution can be decreased by merging branches, allowing to decrease the number of points. One idea is to decrease resolution of old branches first. The figures on the side show a test case with an ever growing spiral.

Sample selection. To reduce the amount of input data, only a few evenly spaced samples of the scan line are chosen. Their resolution in world space is estimated with the maximum distance between scan line and temporal neighbors, using points of several scan lines. At a depth discontinuity a few samples in its neighborhood will be assigned a low resolution, encouraging the user to scan this area again. Vertices that are far away from others of the same scan line are considered as outliers and are ignored.

Point Cloud Data Structure. Grid based and unstructured hierarchies have been considered. A self growing, regular cube grid is chosen as data structure to provide fast update of existing points as well as Level of Detail (LoD) and culling support. Information in such a cubic cell consists of

- The 3D position quantized to a mm grid
- The average color
- The average normal, if available
- The minimum resolution
- A 3D array of pointers to a subdivided structure of that cell

Point Cloud Management. When adding a point outside of current boundaries, the maximum dimensions grow automatically so that a new top level is inserted containing the old one as a sub level. An update of existing information is performed if the new information has a higher resolution than the existing one. To preserve a high frame rate, the number of subdivided cubes is limited. High resolution information is therefore deleted over time, preferably at random positions far away from recently scanned areas.

Display. For display, depending on the position of the camera, suitable levels of detail are chosen and culling is performed for objects outside the viewing frustum. When traversing the hierarchy, invisible cubes are ignored (hierarchical view frustum culling) and depending on the projected size, just a certain level of detail is rendered (detail culling) while reducing aliasing artifacts at the same time. Only the GL_POINT primitive is used, as more sophisticated splatting is more expensive and was hard to implement at the time. Another interesting problem is camera placement. We envisioned several possible ideas, including a viewpoint from the handheld scanner, a viewpoint from an estimated head position of the user, and a viewpoint from above for a top view.
Point clouds are very well suited for interactive LoD algorithms.

6.2.3.2 Video Surveillance: Autovista

Video surveillance operators are faced with the problem, that more and more camera images are available and they cannot look at all images simultaneously. In the *Autovista* project (Advanced Unsupervised Monitoring and Visualization of Complex Scenarios) we looked at how to help the operators. With our project partners, we use Computer Vision algorithms to analyze each image in real time and automatically signal unusual events. Also, person detection is performed. Our task in visualization is to show an overview of the situation, including the results and details on request. A 3d model is of the facility is used to show the information in its spatial context. At a distant view, an overview is possible with a simplified rendering of high level information. Zooming in, camera images are visible that are projected back onto the 3D model geometry, especially facilitating the task of following a person from one camera image to the next. To be able to simultaneously display the large amount of information, it is helpful to have a display with many pixels available. Thus, we implemented this application for our high resolution Graz HEyeWall.

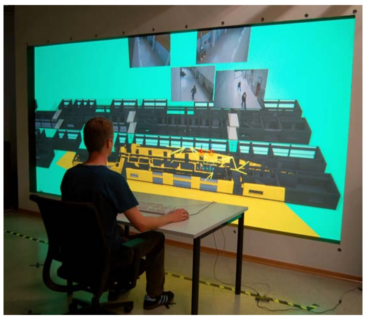


Figure 6.27: AUTOVISTA: A CCTV surveillance system with a 3D model to provide spatial context.

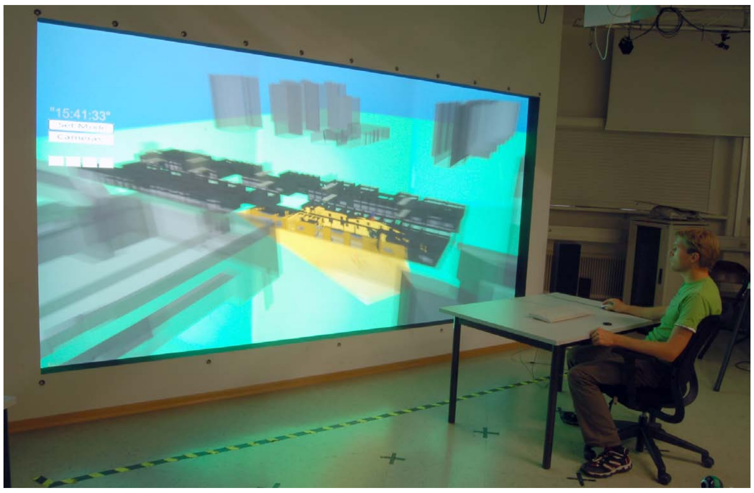


Figure 6.28: An operator looking at an overview of the whole scene.

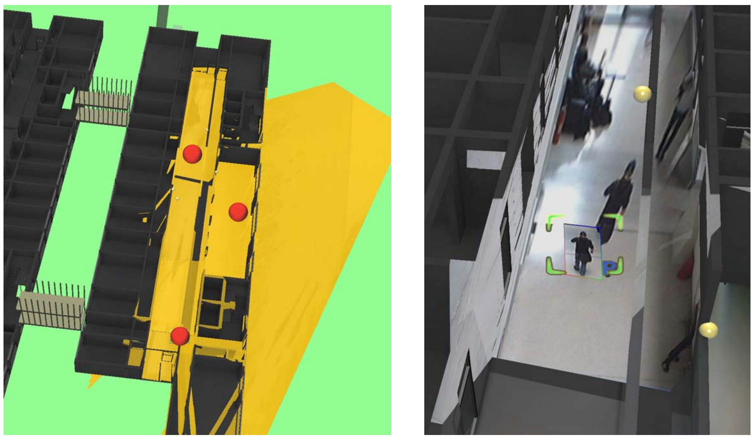


Figure 6.29: Different levels of detail are shown. Left: The buildings are abstract and semi transparent. The surveyed areas are marked in yellow and red dots show detected persons. Right: The buildings are detailed and camera images are projected from the camera positions (yellow spheres) back onto the geometry. A detected person is additionally drawn with a rectangular billboard. When zooming, the the levels of detail blend continuously.

One challenge for scaling the prototype system with few cameras to a large system is the limited video bandwidth. We started development of video servers, each connected to about a dozen cameras, capable of dynamically scaling down several video streams to the currently necessary resolution and sending DXT1 compressed streams via UDP to the rendering machine. We also implemented frame rate limitations to further reduce the bandwidth. Unfortunately, the compression and high bandwidth network transmission was never finished, as in the course of the project, only previously recorded data by few cameras was used. For an unoccluded view of multiple levels of a building, we experimented with explosion views, i.e. separating the levels and moving them away from each other. This successfully avoids overlaps but also makes it harder to track people across levels. For more details, please refer to our publications [LSHF09] and [RSW*11].
In combination with facility control, we see a great potential to facilitate the task of the operators. Challenges remain to be solved, both in Computer Vision algorithms as well as high performance video transmission. The

realization of some good ideas is prohibited by law for use in public spaces, in order to prevent misuse.

6.2.3.3 Realtime Raytracer

A simple GLSL raytracer was implemented to both showcase the Davelib and also to use the framework for educational purposes in the lecture on *photo realism in computer graphics*. With a working framework and the primary ray already set up, students can easily test and experience ray tracing in real time, implementing ray intersection algorithms for several primitives, as well as caring about shading and lighting. In the meanwhile, NVIDIA released OptiX [PBD*10], an interesting and powerful framework as an alternative for such an application. In section 6.3.5 the usage in immersive setups is described.

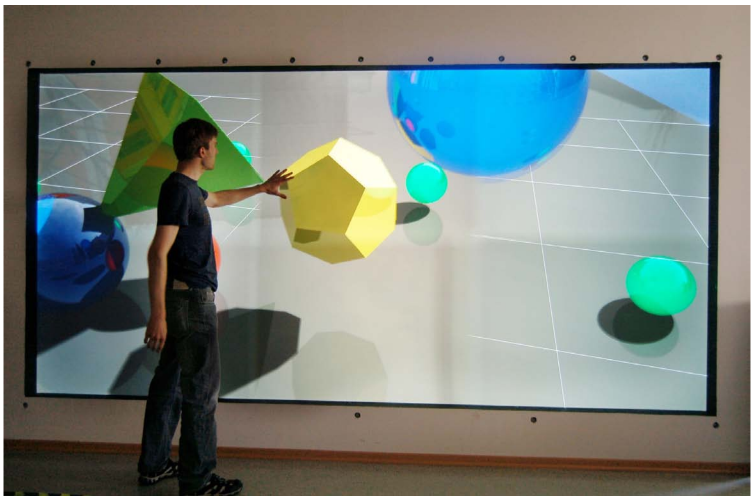


Figure 6.30: A simple realtime raytracer on the HEyeWall, here still with a few clipping artefacts due to the calibration matrix.

The concept of raytracing is well suited for educational purposes. A fundamental understanding of the algorithms and problems is conveyed to students in a fun way of an own implementation.

6.2.3.4 Miscellaneous

Here, a few further visualization applications are presented that are not described in detail.

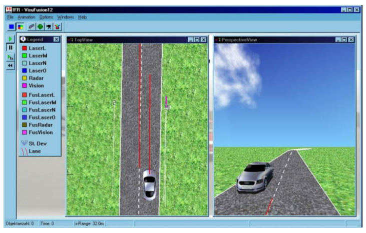


Figure 6.31: 3D visualization for autonomous navigation with the TU Braunschweig.

Visualization of Sensor Data: Visufusion. In a project for autonomous navigation of cars with the TU Braunschweig in 2001, we helped with the 3D visualization of previously recorded sensor data.

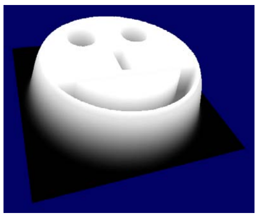


Figure 6.32: Interactive height field visualization for illustrative purposes.

Height Field Visualization. The real time visualization of height fields is a special module used in the presentation viewer (see section 6.2.4.3, used for illustrating image processing results.

Wavelets on a Sphere. Motivated by results with sperical wavelets by Schröder et al. [SS95], we also implemented a subdivision of an icosahedron with triangles. Our intentions are especially targeted to wavelet based image processing of spherical textures. The image below shows the application of blurring a texture, useful e.g. for generating a blurred version of an environment map that can be used in a shader.

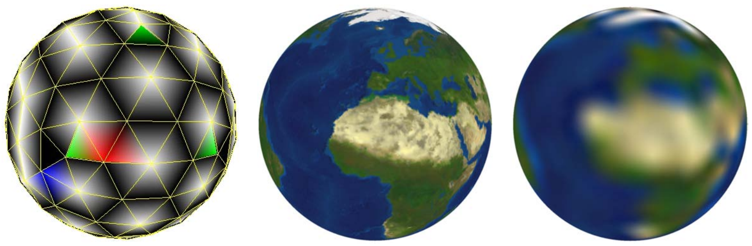


Figure 6.33: Icosahedron subdivision for spherical wavelets on a sphere. Visual debugging and testing of finding neighbor vertices (left), loading an image texture (middle) and applying a blur by setting wavelet coefficients of a higher level to zero (right).

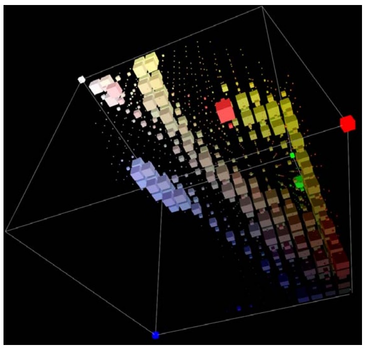


Figure 6.34: Interactive 3D rgb histogram viewer.

3D RGB Histogram. To visualize the color distribution of an image in RGB space, this interactive 3D histogram viewer was implemented. The size of the cubes depends on the number of pixels within that color range. This tool was used to analyze images and gain insights to adapt a segmentation algorithm for the 3D scanner already mentioned in section 3.3.3.

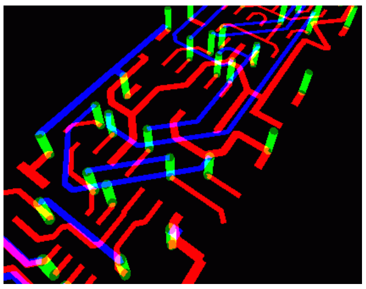


Figure 6.35: Interactive illustration of multi layer circuit layouts.

3D Visualization of PCB Designs. Printed circuit board designs often involve multiple layers. This viewer helps to better understand the complex 3D structure of the layout.

Random Dot Stereograms. To better understand the principles of random dot stereograms, we implemented it ourselves.

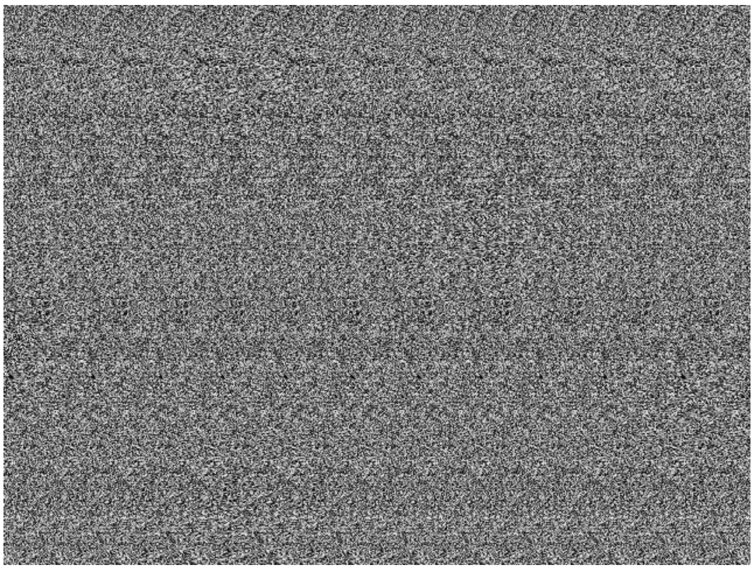


Figure 6.36: Random dot stereogram. Input images are a depth image and a pattern. For the output, the pattern is repeated with a horizontal spacing influenced by the depth image. When viewing the image cross eyed or parallel, the 3D structure of a sine surface is perceived.

Score Four. To measure the influence of different types of displays for complex 3D structures, we picked and implemented the *Score Four* game and extended it with free placement of the tokens.

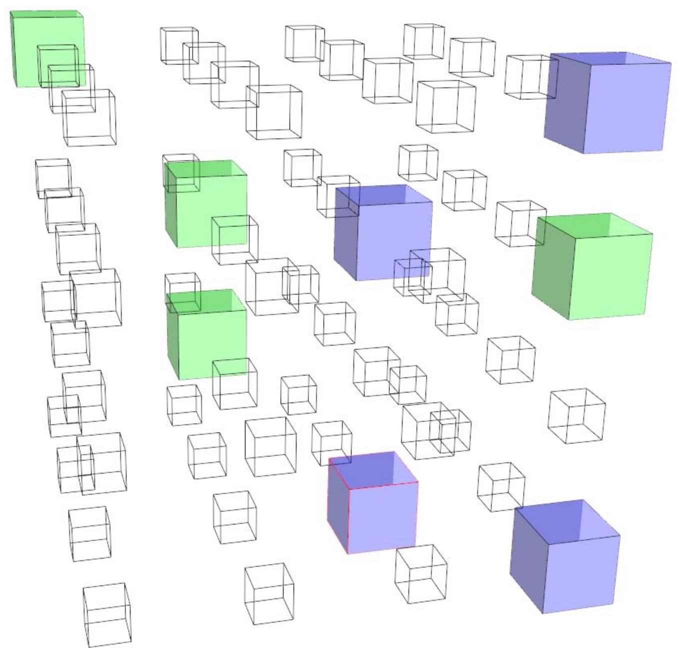


Figure 6.37: A variant of the Score Four game with free placement in 3D. This network game for a monoscopic screen was written to compare performance of players with and without head tracking. We found that head tracking was used much like a rotation input device, so instead of head tracking an automatic back and forth rotation has a similar effect to provide a good understanding of depth by motion parallax.

6.2.4 Browsing, Organizing and Viewing

6.2.4.1 Multimedia Database Organizer

Browsing and finding specific photos within a large amount of photos can take a long time. We looked at options how to improve this situation, with the potential to use this for other multimedia files as well.
This project started in 2005, in the mean time, other free and commercial solutions became available. PhotoMesa provides an innovative user centered zoomable interface. Adobe Photoshop Lightroom targets professional users, especially facilitating batch processing of shootings. Google's *Picasa* is especially useful to also publish photo collections in the internet. *Google Maps* shows georeferenced photos on a 2D map, while Microsoft *Streetview* and *Photosynth* register photos in 3D space. Interestingly, little work was done in research at the time. As an example from 2006, in *Understanding Photowork* [KSRW06], Kirk et al. analyze what tasks users perform with their photos as well as their workflow, with the goal to better understand how to support them. A few further papers discuss new user interface ideas, like *photo magnets* [CRB10] for structured and unstructured exploration,

also allowing to find similar images or group photos with a magnet metaphor.
To find a known photo in a private photo collection, vague information is usually known, like a time range, a location, etc. We intended to include a filtering mechanism to allow to increasingly restrict the displayed number of images. Also, we opted for a zoomable, mouse oriented interface, allowing visual browsing by displaying many thumbnails, i.e. maximizing the number of pixels used to display images instead of GUI elements.
Another important requirement is a simple way to add new meta information during browsing, facilitating future searches. Our idea is to easily define new types of information using categories or key-value pairs with text or numerical 1D and 2D values. Some may only make sense in a special context. Different rating criteria for the technical or artistical quality may be defined by a user. Some photos are probably found a lot faster by tagging them with a category, like *animal*. In some special situations, it might make sense to e.g. specify the number of people visible in the photo. In a browsing session later on, the photos of that session might be arranged in 2D with a horizontal position given by the time and using the number of people for the vertical axis. Another scenario may be first selecting and sorting out photos for a specific slide show and then showing the photos in the desired order. For an easy reuse of meta data, backup and copying of folders, we store the data in one xml file per directory, with a quick binary cache in the background for meta information and small thumbnails. The goal is a scalable system, allowing to see all of the user's photos at once. Note that a lot of useful information can usually quickly be extracted from EXIF tags of the photos. When loading new photos, a major performance increase is to use the EXIF thumbnail for the low resolution display, if available. This allows to quickly browse new files and folders. Some information may also be generated or estimated, e.g. heuristic algorithms might be used to automatically detect and tag an outdoor photo with a blue sky. Such an external program could easily work with the xml database in that folder, which provides a simple interface for such specialized applications.

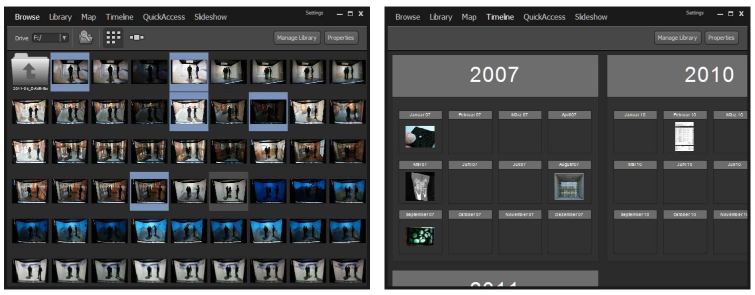


Figure 6.38: Zoomable folder view (left). Image loading is done in the background, so that zooming and panning is always fluent. If possible, the small preview image in the EXIF header of a JPEG file is loaded, a major speed up. A hierarchical calendar view (right) which seems to be less useful. A time line may be a better approach.

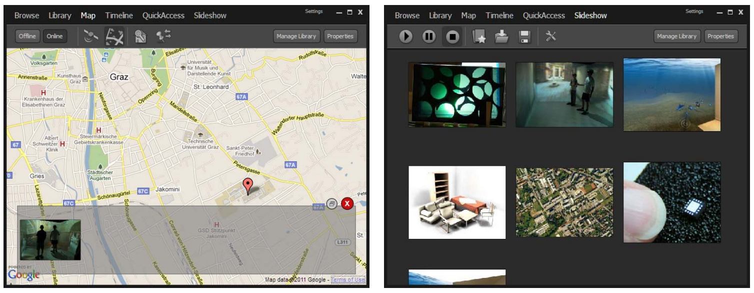


Figure 6.39: The Google maps API was included for geotagging photos (left). In the slide show view, the order of the images can easily be rearranged by dragging with the mouse (right).

To display arbitrary 1D and 2D values, both positioning the photos at or near the respective value as well as a sorted view may make sense. Potential problems of the first approach are either large gaps or many photos very close together. Some important values deserve special treatment, notably the geographic location. Even today, only few cameras feature automatic *georeferencing*. However, it is often easy for the user to place a group of photos on a map, also specifying an uncertainty radius. An example are holiday photos. A quick annotation by selecting the photos, moving them to the country on a map and selecting a large uncertainty radius of hundreds of kilometers, has the potential to greatly speed up a future search. Of course, the location may be refined later on.

The idea of quickly browsing and finding photos is illustrated in the following example. The user selects a view, like the time axis, then drags with the mouse to select a time range. Changing to the map view, only the selected photos are shown. Again, by dragging with the mouse, the user selects a region on the map, thereby restricting the amount of photos drastically. After changing to a sorted view, the few remaining images may be browsed by looking at the thumbnails. In some cases, it is desirable to add photos with additional selection criteria. The solution is a filter graph, allowing both *and* and *or* rules. A filter graph represents a special view or set of photos that may be saved for future use, with automatically updating its content when the information in the database changes.

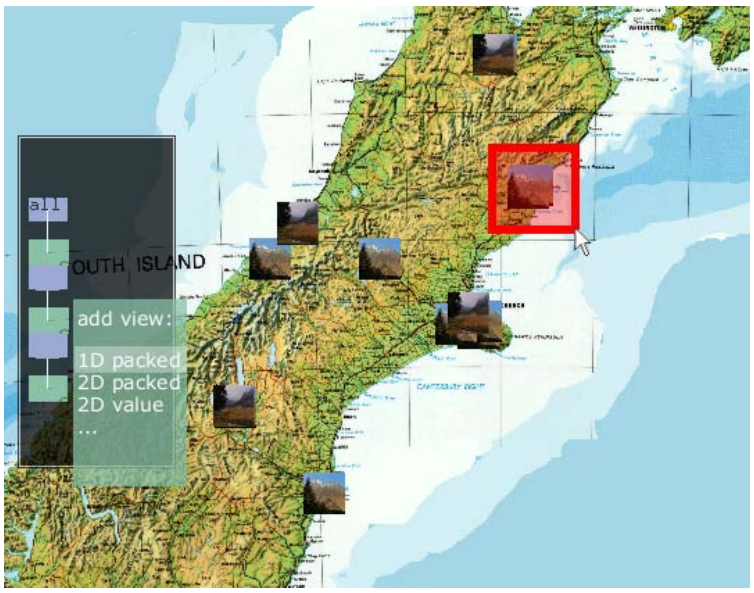


Figure 6.40: Early interface prototype for a fast, zoomable photo browser. To quickly find a photo, the user can subsequently add filters to a filter graph (left) and use a drag (red box) to define the filtered values, e.g. dragging a rectangle on the map to restrict the selection to all geotagged photos within that area.

Browsing such collections profits from a high resolution display and a touch interface for collaboration, thus it is well suited for the HEyeWall or the multi touch table.

Such a flexible, zoomable interface makes it possible to allow a much better browsing and searching than what a conventional file system offers. File *systems should natively support storage of self defined meta data so that this concept can be used effectively in a general way.*

6.2.4.2 Gigapixel Image Viewer

Another project idea of special interest for the high resolution HEyeWall is an image viewer for large image data, like map datasets or gigapixel panoramas. We have not implemented such an application yet. Now, Microsoft's Deep Zoom viewer allows this on a destop PC, but it cannot easily be used on the HEyeWall.

6.2.4.3 Presentation Viewer

The first version of our slide presentation viewer dates back to 2002, where slide transitions with Microsoft PowerPoint were not smooth, rendered with very few frames per second. Thus, we implemented a hardware accelerated slide viewer. Large textures were supported by automatically generating tiles at the maximum texture resolution of many graphics cards at the time at 256×256 pixels.
When showing a new slide, the next slide is preloaded in the background, only keeping the current slides in a texture cache. For an overview, zooming out to a thumbnail view and a quick selection is possible with a mouse drag, as shown in the image below. A few 2D and 3D slide transition effects are the main feature. Support for showing textured 3D models on a slide with the .obj format, .avi videos and high resolution 2D insets (see section 6.2.2.4) was added. For a series of images, like exported PowerPoint slides, a default slideshow configuration file can be generated. Lacking a graphical editor, changes have to be made manually with a text editor.

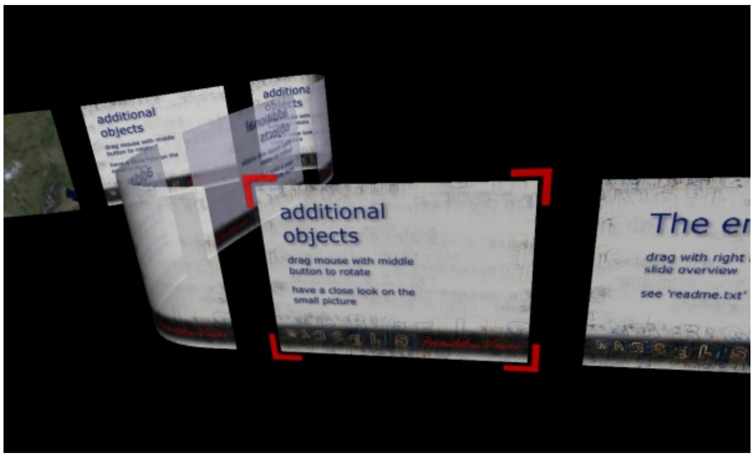

additional objects
drag mouse with middle button to rotate
have a close look on the small picture
The en
drag with right
slide overview
see 'readme.txt'

Figure 6.41: An OpenGL based presentation viewer with 3D slide transition effects. Here, an overview is shown to jump to another slide, i.e. during questions at the end of a talk.

As the code is rather dated, we are in the process of creating a new version using x3d and instantreality. There, slides can be freely arranged in a 3D scene and templates are available, helping to quickly generate such information. One useful new feature will be a preview of the next slide on a laptop display, while showing the current slide on the projector.
Instead of a commonly used 1D organization of slides, a logical and interesting way is to use more dimensions and a hierarchy. Zooming and

panning can serve for a better overview, using the human cognitive mapping capabilities.

6.2.4.4 Museum Artifact Interaction

In museums, very small artifacts of an exhibition can not be seen well or from all sides because of precautions for safety and conservation. However, by placing a monitor next to the real artifact that shows a rendering of a 3D scan, visitors can see all details. Additionally, annotations may also be overlayed. Each visitor should be able to rotate the object in an easy way. A mouse is probably not the best interface, and even a trackball may seem too daunting or complicated for people who do not use personal computers. One of our ideas is to build a physical prop, resembling the artifact, that the visitors can turn in order to rotate the object on the screen.
After some time during the development of a camera based solution using natural features on the prop, it became more and more clear that still a lot of effort would be necessary for a good working system. Also, there are always potential problems with occlusion, a limited field of view of the camera, lighting issues, dirty or dusty lenses, etc. We thus moved to a different solution with a three axis accelerometer and a custom interface as described in section 2.3.5. Measuring the direction of gravity, two degrees of freedom are obtained. We implemented a direct mode, applying the angles to the object orientation directly, and a mode where the rotation speed can be controlled.
The main problem is the missing absolute value of the rotation around the gravitational axis. Especially when the virtual object and physical prop rotation are off by about 180 degrees, this leads to a motion opposite to the expected one. To overcome these problems, gyroscopes can be added to better estimate relative rotations around the gravitational axis. The correct absolute value may be determined with the help of an optical system that only has to work once in a while to correct the absolute value, while the other systems always deliver relative data with fast update rates. For more details, please refer to our publication [HSLF07].
Trying to develop a tangible interface that is easy to use, we have not found an optimal solution yet. For the intended application to rotate a 3D object on the screen, a touch screen may be better suited.

6.2.5 Arts and Education

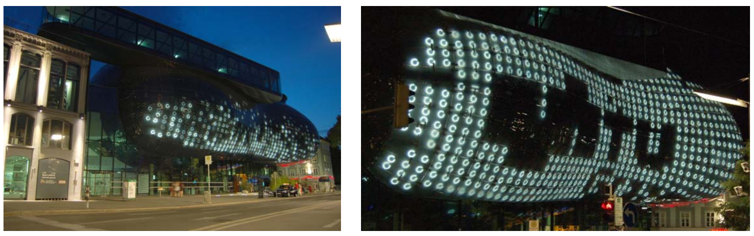


Figure 6.42: The BIX media fassade of the Kunsthaus Graz is the largest display we used so far, in this case for artistic purposes. It has only 930 pixels in total.

6.2.5.1 Rapid Prototyping Framework for Innovative User Interfaces

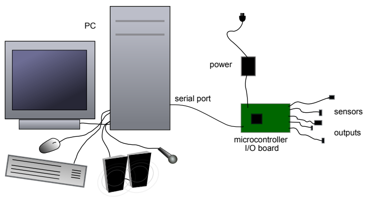


Figure 6.43: Hardware components being addressed by the IO interface educational framework.

For a one week workshop in 2005 on *rapid prototyping of innovative user interfaces*, we developed an educational framework with software available in both C++ and processing. For multimedia applications we added support for sound input with frequency analysis and output, video rendering and access to custom external hardware components. The self developed IO interface provides 8 analog inputs, 7 digital inputs, 1 analog (PWM) output and 6 digital outputs, that can easily be manipulated by software. Thus, a number of hardware sensors could easily be used, e.g. photo transistors, accelerometers, motors, pressure sensors, buttons or potentiometers.
At the same time, development for the very similar *Arduino* platform started. It provides open source hardware and software and now there are many extensions and compatible clones available. The hardware can be self made but also bought for a reasonably good price, making it a good choice for prototyping nowadays. Many software examples are provided by the community. Related, earlier projects are the *I-Cube System* [Mul95] or *Phidgets* [GF01].

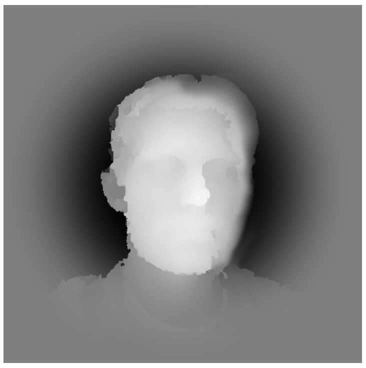


Figure 6.44: Kinect depth map processing for customizing products.

6.2.5.2 Product Customization

For the *MetaDesigner* project, we looked at possibilities for users to customize product geometry. One of the many ideas is to use the geometry of the user's face, obtained by a scan with a kinect sensor and post processing with a combination of a few simple filters. This depth map can be used as a surface offset on a product.

6.3 IMMERSIVE 3D APPLICATIONS

In this section, applications are presented that fit well to our immersive VR setups.

6.3.1 Lighting Tool for Precomputation of Static Scenes

Usually, in architectural applications most geometry is static. When also restricting the options to static lighting and (mostly) diffuse textures, a complex global illumination can be computed in a preprocessing step. The amount of incident light is baked to a light map, in addition to the color texture map. To compute such a light map, we use Autodesk Maya. However, this process is manual and in some situations it does not work: The software just crashes and there is no way to get a result. Also, reflective or refractive materials that are realized by a shader in the interactive visualization, cannot easily be included in the same way in Autodesk Maya, resulting in a lot of work or wrong lighting.
To avoid these problems, we implemented a tool that computes the view from each point of view of a texel, using the average brightness as light map value. We start with black maps and switch to the new maps after computing the intensities for each texel of the scene. In few iterations, a good looking solution is computed. Note that with each iteration, the maps become brighter. This can easily be addressed by using float values and tone mapping. As the rendering engine is

directly used to compute the incoming light, all effects and shaders work automatically. For example, glass, mirrors and other transparent or reflective surfaces are handled correctly. As an example, sun light reflections of a window or mirror are visible and even caustics are produced on a diffuse surface.

A technical problem is the near clipping plane, as closer geometry is simply not rendered. Without handling this case, light may leak through walls, producing well visible artifacts. In other occasions, light from close surfaces may miss, resulting in too dark patches. We solve this by rendering backsides of polygons with a reserved color (plain pink). Now, only valid pixels are used and their brightness is averaged. To increase performance, the hemisphere or hemicube is approximated by a wide angle perspective camera view of the scene.

Another issue is automatic texture coordinate generation for the light maps. Often, in 3D models the same textures are reused for similar models or for tiling. For light maps, a separate light map for each surface is necessary. The object size in the world coordinate system should influence the size of the light map. For an efficient arrangement, a texture atlas can be used, which we have not implemented yet. To avoid interpolation errors on light map patch edges, a few pixels space are left as padding to the next patch. After each iteration, colors are spread into the gaps with an iterative morphological algorithm.

Finally, the low speed of this approach is an issue. However, all speed improvements from the rendering engine can be used, and most of the computation can be distributed to several computers, only needing a synchronization after each iteration.

An interesting possible future extension is to store the incoming light for each texel not as a single scalar, but instead a few terms with spherical harmonics to approximate the incoming light on the hemisphere. In that case, diffusely reflecting materials can be realized. An efficient data representation and compression will probably be essential.

Using the rendering engine for precomputation of light maps solves a few problems. To achieve reasonable computation times, many common ideas for acceleration can be used. This approach may be especially interesting for fully automatic lighting computation, e.g. for clients who want to inspect their own 3D model in the DAVE and upload the model without lighting the day before.

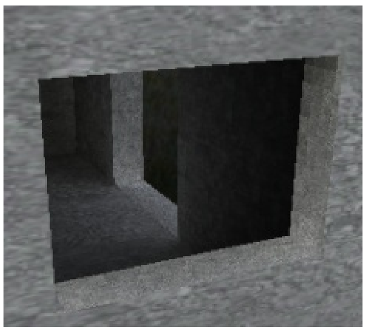


Figure 6.45: Result of our hardware accelerated global illumination tool for static scenes. It is robust and always works when the model can be loaded. A preview of intermediate solutions is available as it is iteratively refined. By using the same rendering technology as for the interactive display, all lighting and materials are included correctly, such as an environment map for image based lighting and shader materials.

6.3.2 Interactive Illumination for Dynamic Scenes.

In the last years, screen space approximations for global illumination became popular. Already in 2003, we also came up with this idea. At the time, the computation had to be done on the CPU. A recent implementation is presented by McGuire in [MOBH11]. Such screen buffer based approximations may lead to visible differences at edges in the DAVE.

The amount of artifacts at the edges in the DAVE due to the approximations may be negligible. It makes sense to adapt and test such algorithms for dynamic scenes in the DAVE.

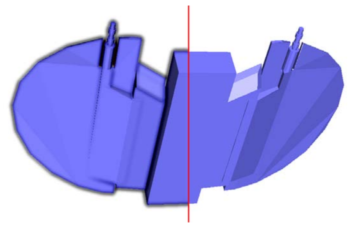


Figure 6.46: An attempt of screen space ambient occlusion (AO) in 2003 (left) compared to normal rendering (right). The AO pass had to be done on the CPU back then.

6.3.3 Architecture

Arguably the best applications for a CAVE are architectural visualizations. The large angle of view in a CAVE is perfect for an immersive impression of the site. An interesting observation is that a beautiful 3D model may well be the most important factor for a convincing application. User interaction, navigation, technology etc. can be great and innovative, but will hardly impress if the wrong model or scene is used.

6.3.3.1 Indoor Architecture

One of the most successful DAVE demos is a detailed model of an apartment with precomputed lighting. Additionally, a few furniture items can be inserted via a menu with the joystick. These inserted objects do not influence the lighting, e.g. they do not cast a shadow. However, we baked a precomputed ambient occlusion into their textures for a more believable lighting of the object.

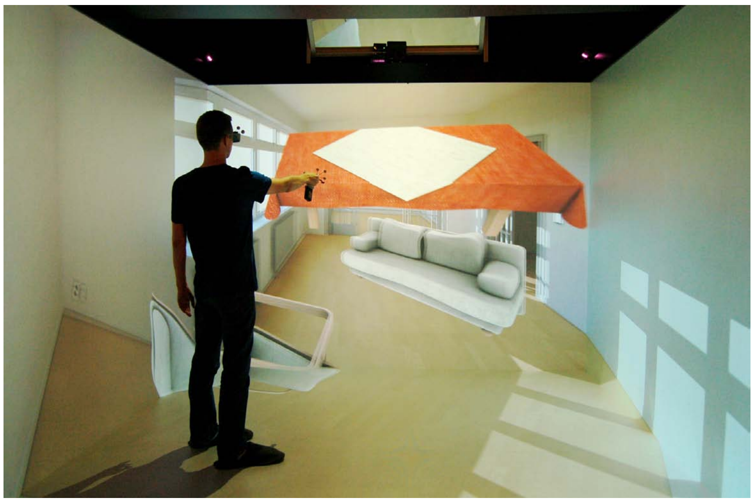


Figure 6.47: Interactive furniture arrangement in a virtual apartment.

The inserted objects can be moved around and interact with each other and with the walls. For that purpose, a very coarse geometry of the objects and the apartment is manually replicated in order to speed up the collision detection, ignoring many of the details used for the visualization. The NVIDIA physics engine is used to compute collision responses and is very fast.
The furniture can be freely rotated to allow exploration of the interface and possibilities. Already positioned furniture is easily pushed away by accident while placing another furniture item. This is more fun than actually useful for the task of arranging furniture. In the latter case, the standard orientation of furniture items should respect their natural up axis and a reduced or disabled collision reaction between the items seems helpful.
The push of a button on the joystick toggles between travel and manipulation modes. For novice users, this is sometimes not clear and a better feedback should be added. Also, novice users do not understand why an object jumps up when they place it slightly in the floor by mistake. A damping of that motion, a visual feedback or an object motion with physics provide alternatives that may be better to understand. The non-linear translation of objects works very well.
A simple exploration of virtual models of sights is also an interesting application. Such virtual tourism can be tested with the 3D model of the National Library. This application does not only save a visitor to physically travel to the location that may be on the other side of the world, but also allows inspection from positions that are not publicly accessible on the real site.

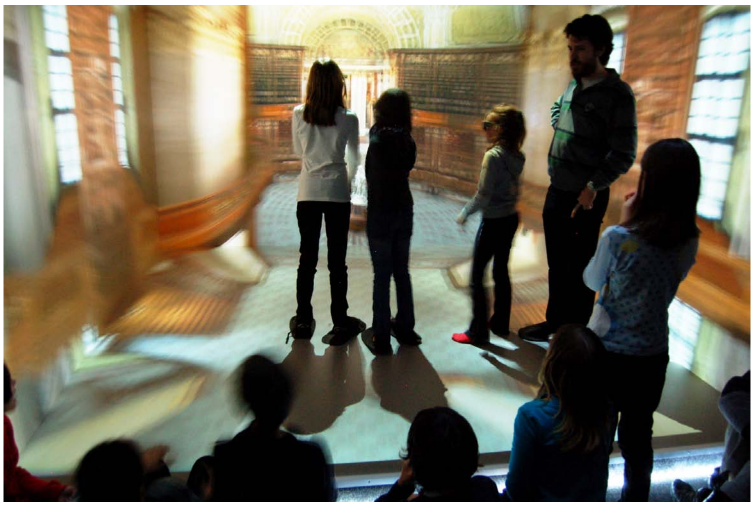


Figure 6.48: Kids flying through the State Hall of the Austrian National Library.

Levels in professional 3D games are often modeled with a lot of effort by skilled artists. Showing such models in the DAVE is far more impressive than to use our self made models. An early example is the *Quake III Arena* level viewer. We downloaded and modified it to be able to fly through the levels in the DAVE. An unwanted effect that we could not easily change is that the collision response of the viewer reflects the motion vector when hitting a wall, even when hitting at a slight angle. This may lead to an awkward feeling, especially when traveling at a high speed.
Even worse, the collision reaction is computed for the origin of the DAVE, regardless of the viewer position in the DAVE. With the current implementation, the user cannot navigate through a wall with the joystick, but physically walk through it within the limits of the DAVE. Better would be to push the model away in that case, not allowing a penetration. Another problem with collision detection is that a novice can quickly get or at least feel trapped. For these reasons, we often just allow a free motion, also through walls.
Collision detection in the DAVE should include the current user position with respect to the origin and should be modified to push the model away in case a user causes a collision by physical walking.
Later, Id Software released the game source code to the public domain and we ported the game with all the interactions as well. Interestingly, many visitors with an age of 20–30 years especially ask for such a game port. Already in 1996 and later, Paul Rajlich worked on clones of the similar games Doom and Quake II that ran in the CAVE at NCSA.

House Modeler. The 3D preview of planned architecture is an interesting application, but the cost and effort of manually generating a 3D model are often a problem. Still today, not always a complete 3D model is available for new buildings. A fully automatic generation from the 2D plans is not yet possible, they must be interpreted by a human observer.
To facilitate a semi automatic model generation from 2D plans, we designed the concept of the *House Modeler*. The Generative Modeling Language (GML) [Hav05] is used to allow easy subsequent changes,

like resizing a window or moving a wall. Especially cost effective may be a model range that can be reused, such as a limited choice of models for prefabricated houses. The available options may be offered, even colors and materials may be selected appropriately and their cost can be displayed. To show the model with a nice lighting in the DAVE without further manual work, the lighting tool described in section 6.3.1 was developed. A few different lighting situations may be precomputed and selected for interactive viewing in the DAVE.
This idea sounds good but there is little interest. Especially vendors of prefabricated houses have build many similar houses that can be visited by potential customers. Materials are shown in large halls for comparison, which is still much better than looking at an image of that material.
While the immersive preview is valuable in our opinion, it is unclear how much money customers are willing to spend. If the architect planning a new building has a tool that can export a 3D model, an automatic workflow can be established in order to reduce manual work and reduce the cost.

Yachts and Cruise Liners. For the verification of planed passenger cabins on a cruise liner, we obtained a textured 3D model. While the designer complained about a few details that were apparently wrong in the supplied 3D data, it made clear how convincing and realistic the impression is. We observed this with several other 3D scenes: Errors in the model may be overlooked on a normal 2D monitor, but in the DAVE they are often obvious at the first inspection.
The visualization of yacht interior design is a current project with the hope to help selling the product.

6.3.3.2 Buildings and City Planning, Exterior Architecture

Using available data from a city, we automatically generate a 3D model. Within a week, the city of Braunschweig model with 200km^2 was converted in 2004.

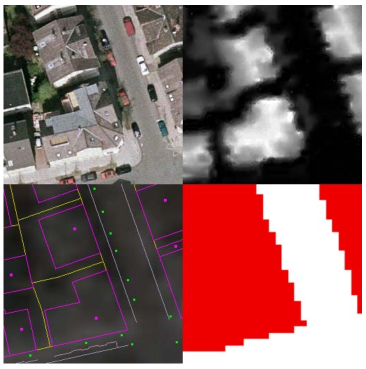


Figure 6.49: Available data for the city. Top left: orthophoto, top right: LIDAR scan (processed height data), bottom left: cadastral map, bottom right: soil usage map.

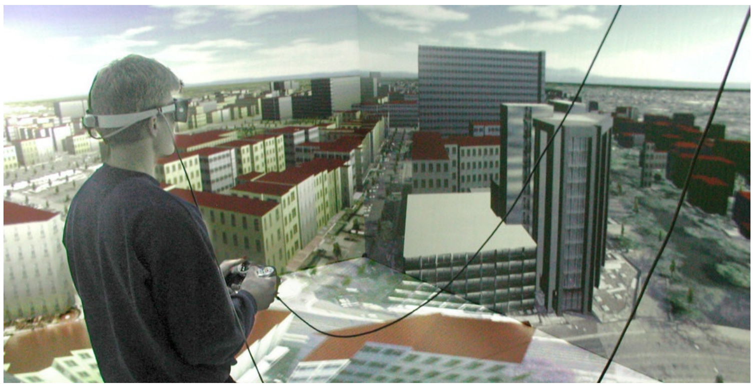


Figure 6.50: An automatically generated 3D city model of 200km^2 of Braunschweig in the first DAVE at TU Braunschweig. The data illustrated in the figure above served as input. The building on the right is one of the few detailed buildings that were modeled manually.

We identified a number of applications for a 3D city model. However, most of them do not require an immersive environment:

- Information and navigation (maps, tourism, navigation systems, user interface for GIS)
- City planning, architecture and real estate business

- Physical and statistical analyses and simulations (radio wave propagation, flooding, fire, air flow, traffic noise)
- Military and security (training, emergency coordination)
- Multimedia (computer games, entertainment, movies, advertisement)

For these applications, traditional city geographic information system (GIS) data needs to be transferred into a 3D GIS database. In a case study with the data from the city of Braunschweig, our major achievements are a unified access to data from various sources, automatic model generation and interactive display through a number of culling and LoD algorithms. For more information, please refer to [LF04a] and [LF04b].

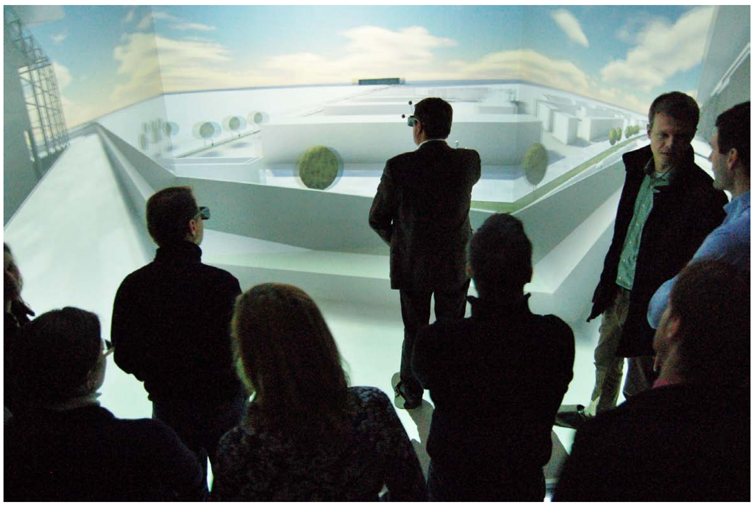


Figure 6.51: Frankfurt exhibition center, a business example for verification of a planned site using VR.

We see a great potential for 3D city models. Cost for conversion or acquisition and maintenance as well as non standardized data formats are major problems. Google and Microsoft both solve the problem by investing a lot of money and own acquisition.

6.3.4 Drawing, Volume Data Visualization and Interaction

6.3.4.1 Volumetric Painting

For 3D sketching and painting, we developed a volumetric painting program for the DAVE, based on a voxels, cubic volume elements in a regular grid. Each voxel contains a color and transparency value, similar to a 2D pixel based painting program.

Cursor Position and Visualization. The spray position where the paint appears should not directly be set to the position of the joystick tracking target, as in that case the hand and joystick are inside the generated volume, a case where occlusion is not handled correctly. Instead, we choose a position some 20cm into the pointing direction. To provide correct occlusion, the cursor visualization should be included in the voxel visualization code. However, we have not implemented it that way out of performance reasons.

Sound Feedback and Tangible Input Device. For additional feedback, we added playback of a prerecorded spray sound when pushing the trigger. We also planned to use a physical spray can as prop, but found that already the joystick works well enough.

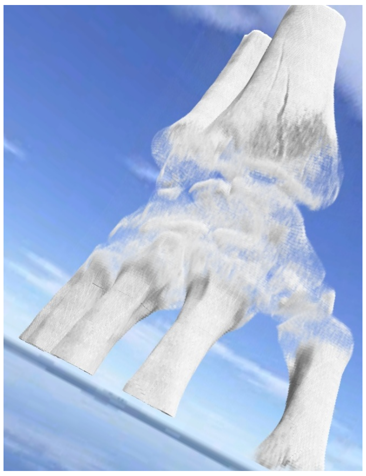


Figure 6.52: Volume visualization of a computer tomography of a fractured radius (top right), loaded in the volumetric paint program. In contrast to conventional volumetric imaging, the creases are made clearly visible by the lighting algorithm.

Lighting. When we first tested spraying with a plain color, we soon noticed that the depth impression in the DAVE did not work well, as in that case, larger areas of the screen have a single color. This gives no clues for stereoscopic fusion and the screens may be perceived instead. Adding a texture or lighting would help. We implemented a very rough approximation of a global illumination, slowly updating for random voxels in the background. A lighting value is stored for each voxel and set according to the transparency values of a few voxels above and to the side. The more of these voxels are transparent, the lighter gets the result.

Rendering Speed. The drawing volume is the whole DAVE so that all positions can be reached without additional travel. To save memory and rendering time, the volume is split into smaller cubes that are sorted in depth and rendered individually. Empty cubes are not rendered at all, a very simple form of empty space skipping. With our rather basic implementation with a constant step distance for ray casting, filling larger volumes of the DAVE significantly slows down rendering and becomes unusable. The application was written in 2006, now much faster algorithms such as OptiX by NVIDIA may be used on faster hardware to allow a larger volume to be filled while still achieving high frame rates. See section 6.3.5 for details on the ray setup.

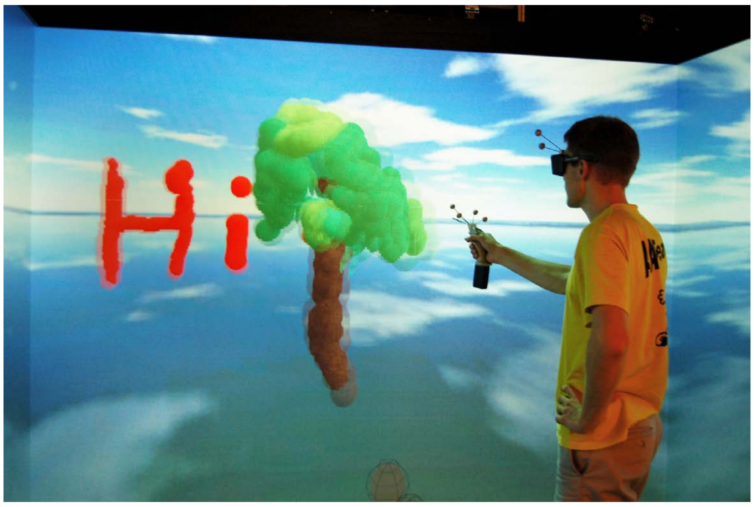


Figure 6.53: Volumetric painting in the DAVE.

A future goal is to be able to draw, paint and sketch volumetric sceneries and models, e.g. a nature scene with a landscape. Non-photo realistic rendering should be considered, as well as testing different types of primitives, e.g. constructing the scene by just using spheres. *Large screen areas with a plain color lead to a loss of the stereoscopic impression. Due to occlusion issues, virtual content should not be rendered at the joystick position but e.g. some 20cm in the pointing direction.*

6.3.4.2 Direct Manipulation of Triangle Meshes

A few specialized tools were developed to manipulate triangle meshes and height maps. One tool enables pushing and pulling of a surface with dynamic retesselation, another tool allows drawing smooth pipes and the last one allows to generate and texture 2.5D landscapes. To select a point on the surface, the ray from the average eye position through the joystick tip is used as bearing. As a feedback, the surface colors around the cursor is changed to represent a cursor.

6.3.4.3 Modeling and Modifying Meshes

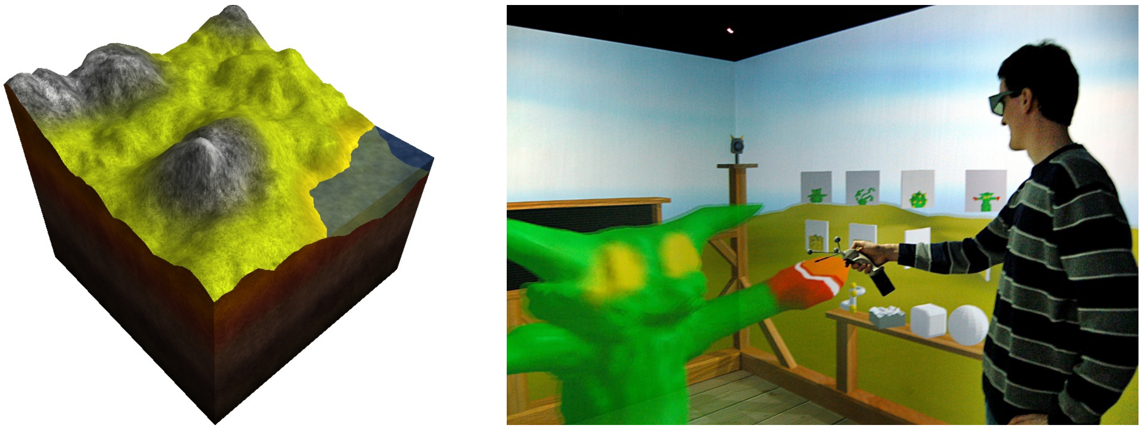


Figure 6.54: Triangle mesh modeling tools for the DAVE. Left: The result of a 2.5D terain. Left: Sculpting is performed by surface extrusion and dynamic retesselation. On the right side, a set of tools can be selected or previously saved models can be loaded.

Using the surface deformation tool, it is rather hard to create a model as intended by the user. When pushing the joystick button, the mesh moves into the relative direction of the joystick motion. However it does not feel like a very direct way of grabbing the surface, but rather acts like drawing a surface offset. The landscape tool is useful, but in that case an immersive environment has little advantage over a desktop setup. The idea of subsequently scaling up the landscape to fly over it is not realized yet. More details can be found in the master's thesis of Lukas Fraser [Fra10].

Instead of sending a new mesh to the clients, only mesh modification commands are send via TCP. According to the developer, in some cases the same commands on the same data on equivalent hardware lead to different results, i.e. comparing two lengths with an almost equal value caused problems. The problem is now mostly avoided in practice by adding a small constant to one of the values. In consequence, robustness is reduced when relying on consistently replicated data when distributing the same commands. A system tolerating errors and maybe slowly synchronizing redundant data may help to allow recovery in case of an error.

3D sculpting may be more suited on a setup with haptic feedback. Often needed functions are mapped to joystick buttons, while other commands are executed by walking up to icons on the side and clicking on them, a well working concept. Obstacles are placed at the screen positions, restricting the physical user motion successfully in a natural way.

6.3.4.4 Generative Modeling

A couple of interactive applications using the Generative Modeling Language (GML) [Hav05] were ported to the DAVE using the Davelib. Thus, parameters of previously programmed models can be modified by picking and dragging sliders in 3D space, next to the model. Some values may significantly slow down the evaluation of the program, so they should be restricted to a sensible range to maintain high enough frame rates.

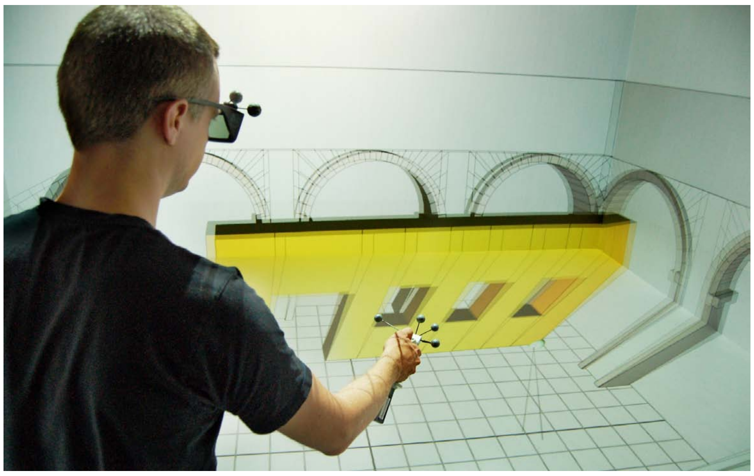


Figure 6.55: Generative modeling of walls. The user aims with the joystick at the small green sphere, resizing the wall by dragging. Note that for the user, the black line visible on the floor appears to start at the joystick position.

6.3.5 Raytracing

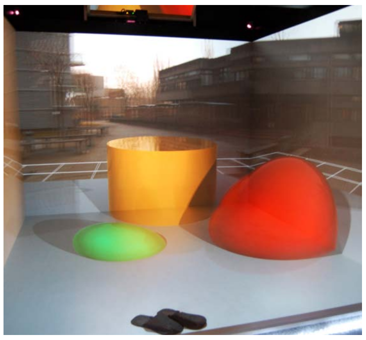


Figure 6.56: Photo of the same raytracing framework as in section 6.2.3.3 in the DAVE, with a shader implemented by a student as an exercise for a related lecture.

Brute Force Raytracer We implemented a framework for a simple GLSL raytracer that we also use for teaching (see also section 6.2.3.3). It fully uses the Davelib and runs on all of our platforms and setups.

Ray Setup. To be able to easily use the Davelib configuration and calibration matrix, we came up with a technique similar to volume rendering, allowing navigation and head tracking to work as usual. One point of the ray for each pixel is given by the camera center, i.e. the respective eye position. We compute that position from the inverse of the OpenGL modelviewmatrix multiplied with the eye position in the DAVE coordinate system (see lines 1 to 9 in the listing below). The second point of the ray is computed by the fixed function OpenGL pipeline. A box as a proxy geometry is rendered around the camera position (see lines 16 to 23). The vertex transformation takes geometric setup and calibration into account as usual via the OpenGL projection matrix. The position in world coordinates *worldPos* is set in the vertex shader and is automatically interpolated for the fragment shader by the fixed function pipeline.

```
glGetDoublev(GL_MODELVIEW_MATRIX, modelViewMatrix);
calcInvModelViewMatrix(modelViewMatrix, invModelViewMatrix);

float eyeWorldPos[3]; // includes camera movement (and tracking and stereo offset)
for (int i=0; i<3; i++)
  eyeWorldPos[i] = invModelViewMatrix[0+i]*eyeDaveCSx
          + invModelViewMatrix[4+i]*eyeDaveCSy
          + invModelViewMatrix[8+i]*eyeDaveCSz
          + invModelViewMatrix[12+i];

// set shader variables
GLint locCamPos = glGetUniformLocationARB(wus->shaderProgram, "cameraPosition");
if (locCamPos >= 0) glUniform3fARB(locCamPos, eyeWorldPos[0], eyeWorldPos[1], eyeWorldPos[2]);
else printWarning("shader variable view_position");

// draw cube faces as proxy geometry
glBegin(GL_QUADS);
 glVertex3f(-10 + camera.x, 10 + camera.y, 10 + camera.z); // +y
 glVertex3f( 10 + camera.x, 10 + camera.y, 10 + camera.z);
 glVertex3f( 10 + camera.x, 10 + camera.y, -10 + camera.z);
 glVertex3f(-10 + camera.x, 10 + camera.y, -10 + camera.z);
  ... // 5 more quads
glEnd();
```

Figure 6.57: OpenGL code fragment to setup raycasting or raytracing using the Davelib.

For the code above, the inverse of the model view matrix is needed. Usually, it only consists of rotation and translation. In that case, the following fast and simple code is sufficient:

```
// this works for rotation and translation matrices only
// matrices in OpenGL order
// mv : input matrix
// imv : results will be in here
void calcInvModelViewMatrix(GLdouble mv[16], GLdouble inv_mv[16]) {
  // transpose 3x3 rotation part
  inv_mv[ 0] = mv[ 0];
  inv_mv[ 1] = mv[ 4];
  inv_mv[ 2] = mv[ 8];
  inv_mv[ 4] = mv[ 1];
  inv_mv[ 5] = mv[ 5];
  inv_mv[ 6] = mv[ 9];
  inv_mv[ 8] = mv[ 2];
  inv_mv[ 9] = mv[ 6];
  inv_mv[10] = mv[10];

  // multiply negative translation with new rotation matrix
  inv_mv[12] = -mv[12]*inv_mv[ 0] - mv[13]*inv_mv[ 4] - mv[14]*inv_mv[ 8];
  inv_mv[13] = -mv[12]*inv_mv[ 1] - mv[13]*inv_mv[ 5] - mv[14]*inv_mv[ 9];
  inv_mv[14] = -mv[12]*inv_mv[ 2] - mv[13]*inv_mv[ 6] - mv[14]*inv_mv[10];

  inv_mv[ 3] = 0;
  inv_mv[ 7] = 0;
  inv_mv[ 4] = 0;
  inv_mv[15] = 1;
}
```

Figure 6.58: Fast 4×4 matrix inversion for a matrix only representing rotation and translation.

The respective GLSL vertex and fragment shaders are shown below.

```glsl
varying vec3 worldPos;

void main(void) {
  vec3 Pos = gl_Vertex.xyz * 1.0;
  gl_Position = gl_ModelViewProjectionMatrix * vec4(Pos,1.0);
  worldPos = Pos;
}
```

Figure 6.59: GLSL fragment shader for a raycasting or raytracing setup in the DAVE.

```glsl
1  uniform samplerCube environment;
2  uniform vec3 cameraPosition;
3  varying vec3 worldPos;
4
5  const int SPHERE_NUM = 15;
6  const int MAT_NUM = SPHERE_NUM;
7  const int LIGHT_NUM = 3;
8  const int REFL_NUM = 2;
9
10 uniform vec3 spherePos[SPHERE_NUM]; // spheres
11 uniform float sphereRadius[SPHERE_NUM];
12 uniform vec4 matColor[MAT_NUM]; // materials
13 uniform float matReflectivity[MAT_NUM];
14 uniform vec3 lightPos[LIGHT_NUM]; // lights
15 uniform vec3 lightColor[LIGHT_NUM];
16
17  ...
18
19 // returns t, where intersectionPosition = rayStartPos + t*dir
20  float intersectRaySphere(int sphereIndex, vec3 dir) {
21    ...
22   return t;
23 }
24
25 vec3 getRayColor() {
26    ...
27   for (int i=0; i<SPHERE_NUM; i++) {
28     float nt = intersectRaySphere(i, dir);
29     if ((nt >= 0.001) && (nt < t)) { // is intersection closer to a previously computed intersection?
30       t = nt;
31       reflDir = reflect(dir, normal);
32        ...
33     }
34   }
35    ...
36   return col;
37 }
38
39 void main(void) {
40   // initialize primary ray
41   vec3 rayStartPos = cameraPosition;
42   vec3 rayDir = worldPos − rayStartPos;
43
44   // trace a few reflections
45   for (int i=0; i<REFL_NUM; i++) {
46     col += ...
47   }
48
49   gl_FragColor = vec4 (col, 1.0);
50   gl_FragDepth = 0.5+0.5 * depth; // transform into depth range
51 }
```

Figure 6.60: Important parts of a fragment shader of the brute force raytracer written in GLSL.

As the distance to the hit object is available in the shader, we can assign it to the depth buffer (see lines to), so other OpenGL objects occlude and intersect correctly with ray traced objects. However, these OpenGL objects will not appear in raytraced reflections. When writing to the depth buffer, it is also important to take the clipping planes into account, as depth values out of range result in the pixels not being rendered.

Raytracer with CUDA. We ported our CUDA pathtracer to the DAVE using parts of the Davelib. To render at a sufficient framerate, the scene is simple and the number of rays is kept low. Thus, lots of noise is visible.

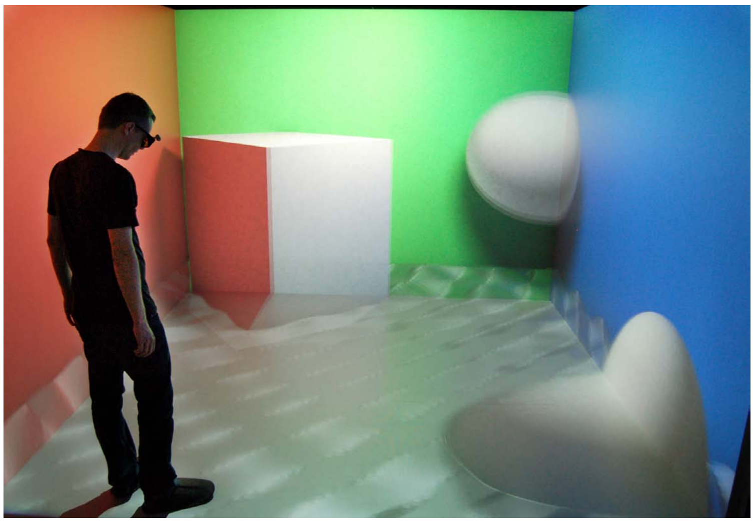


Figure 6.61: A pathtracer implemented with CUDA, ported to the DAVE with the Davelib. Note the color bleeding, especially visible at the left side of the box. To achieve a high enough framerate, the quality is low and a lot of noise is visible, that this photo does not show as the camera exposed over a few frames.

Raytracing is close to being ready for use in immersive displays with a moderate model size on affordable hardware. The head tracked view requires a high update rate, compared to other applications where interactive rendering with progressive refinement already shows great results. With few samples and a good reconstruction filter such as the one described in [BEM11] in a postprocessing step, the strong noise can be avoided, taking a partially blurred image into account.

6.3.6 Shader Materials

In addition to diffuse materials, we implemented a few reflecting materials. To get shaders for such materials work correctly in the DAVE, head tracking needs to be considered. The setup of the viewing vector is very similar to the ray tracing explained above (see section 6.3.5). In instantreality, the transformation matrices are already inverted. The ray computation could be done in JavaScript, however there is an undocumented predefined shader variable *OSGCameraposition* that is set to the camera position in world coordinates, including head tracking and stereo offset. Note that when using a 3D model, its vertex coordinates must be given in world coordinates, i.e. no additional transformations must be applied.

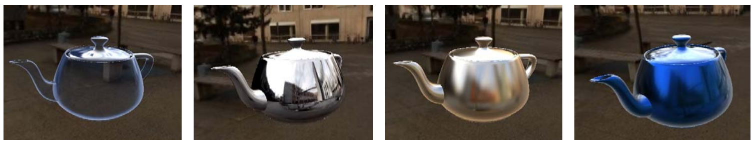


Figure 6.62: A few shader materials that also work correctly in the DAVE with stereo and head tracking. As an example, a mirror reacts on head motion and shows correct depth. To include head tracking, the shaders and their setup need to be adapted. Reflections are only approximated with an environment map computed for one point of view. A second diffuse version of this map ca be used to obtain a diffuse reflection without a loss of performance.

```glsl
uniform samplerCube Environment;
uniform samplerCube EnvironmentDiffuse;
uniform sampler2D fresnel_gradient;

varying vec3 vNormal;
varying vec3 vViewVec;
uniform vec3 porcelain_color;

void main(void) {
  vec3 normal = normalize(vNormal);

  float  v = dot(normalize(vViewVec), normal);
  float  angle = acos(v) * 2.0/3.1415926;

  float   reflectivity   = texture2D(fresnel_gradient, vec2(angle,0.0)).x;

  // reflection  vector
  vec3 reflVec =  reflect (-vViewVec, normal);
  vec3 refl  = textureCube(Environment, reflVec).xyz;
  float reflLum = refl.x + refl.y + refl.z;
  refl  = vec3(reflLum, reflLum, reflLum);

  vec3  reflDiffuse  = (dot(vec3(0.0, 1.0, 0.0), normalize(reflVec))*0.5+0.8) * porcelain_color;

  // mix diffuse and specular
  vec3 color  = reflDiffuse  + refl  *  reflectivity ;

  gl_FragColor = vec4 (color,  1.0);
}
```

Figure 6.63: GLSL fragment shader for a porcelaine material. Two cubemaps and one 1D array are precomputed and used as textures.

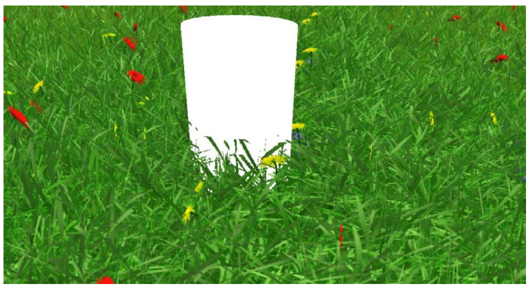


Figure 6.64: We ported the shader from *Instant Animated Grass* [HWJ07] to GLSL and adapted it to correctly work with instantreality in the DAVE. Similar to parallax mapping, the grass is rendered just a single quad, a ray casting is performed in the fragment shader, that even allows correct handling of depth, which is demonstrated here with the white cylinder. Horizontal and vertical slices of grass textures are used and slightly distorted over time to create a wind effect.

6.3.7 Scanned Scenes

An important application is to show a real place in a virtual environment, e.g. for inspection, virtual tourism, city planning etc. Manual modeling is often too much effort. Instead, scanned point clouds may be acquired by merging laser scans from multiple points, or by structure from motion approaches that use multiple camera images. It is still an active research topic to generate a textured surface with few polygons out of such a point cloud.

6.3.7.1 Point Clouds

A straightforward idea is to directly visualize the point cloud. With one OpenGL point primitive per vertex, the visualization is very fast. Unfortunately, the point size does not vary with the distance, leading to holes in close surfaces and too large points in the distance. One solution is to use shaders to display a sphere for each point (see also section 4.2.5.2). The vertex shader generates a billboard quad of sufficient size, the fragment shader fills and shades the necessary pixels to display the sphere. This can also be done using a texture lookup for a fast computation. Note that in general, the sphere projection is not circular. We convert the point data and store it in a binary format for fast loading.

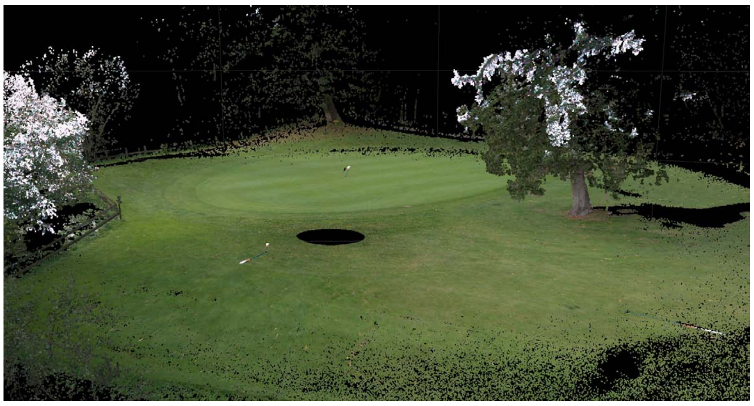


Figure 6.65: Point cloud viewer, showing a LIDAR scanned scene of a golf court near Graz. The original full resolution scan contains 50 million points, only 10% random samples are used here for visualization. The scanner was placed on a tripod near the center of the image (black hole). Close to the scan location, the point density is very high. Scan shadows are visible e.g. at the right side, where geometry was occluded by the tree. To obtain a color, photos can be registered to the scan. The bright areas in the trees show sky color due to a misregistration.

Point clouds are well suited for large models, as clustering, simplification of data and using multiple levels of detail is easy to implement. Similar to our work, immersive rendering and analysis was presented by Kreylos et al. [KBK08]. Wimmer and Scheiblauer present *instant points*, a fast out of core rendering system for large point clouds [WS06] and also show selection and editing [SW11]. Their system streams data from the hard disc via a low level access to achieve optimal performance for out of core rendering.
Our 3D city model (see section 6.3.3.2) also contains a 2.5D LIDAR point cloud that is rendered out of core. As the points are arranged in

a regular grid, we implemented both a quad mesh as well as vertical lines for visualization.
To maintain a fast rendering for a continuously growing number of points, we implemented a simplification on the fly with a dynamic octree, see section 6.2.3.1.
A combination of triangles (e.g. for buildings) and point clouds (e.g. for plants) may be interesting.

6.3.7.2 Image Based Rendering

To avoid the complex process of surface generation from point clouds or problems with holes in point cloud visualization, we also tried image based view interpolation. Multiple photos from different viewpoints are registered and the best ones are selected to generate a new viewpoint. Many approaches exist, see [SCK07] for a comprehensive overview.
We designed a view interpolation allowing free motion for the DAVE. However, so far we only implemented a version allowing 1D translation, using the rear projection wall with head tracking. While seeing a stereo image, moving the head sideways also gives a parallax effect. For this experiment we only use two photos as input and precompute the optical flow from one image to the next and reverse. Using the two images we inversely warp the original images and mix both colors to obtain the result.

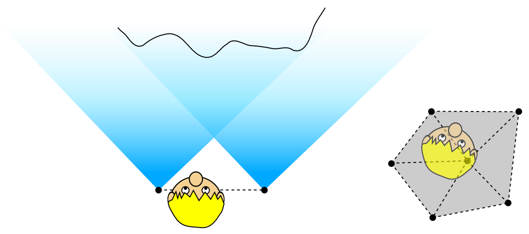


Figure 6.66: Image based rendering with generated virtual viewpoints. The input images are taken from the positions marked by the black circles. The left figure shows the simple 1D case with only two input photos. For each eye, the closest point on the connecting line is computed to obtain a warping or interpolation factor. The right figure shows a set of 360° photos. Virtual viewpoints can be generated in the shaded area, using an interpolation of three photos for each eye.

The figures above illustrated the idea for the 1D and 2D cases. For the 3D extension, the space is divided in tetrahedrons, using four images for interpolation, allowing free motion of the user within the volume. With 360 degree panoramic images recorded at multiple locations, good representative images can be selected to build a tetrahedral mesh. To render an image from an arbitrary point of view, the four images representing the vertices of the containing tetrahedron are used for view interpolation. For each edge in that tetrahedral network the optical flow is precomputed in both directions. As we have not managed to quickly capture 360 degree panoramas and automatically register them in space, we have not implemented this extension yet.

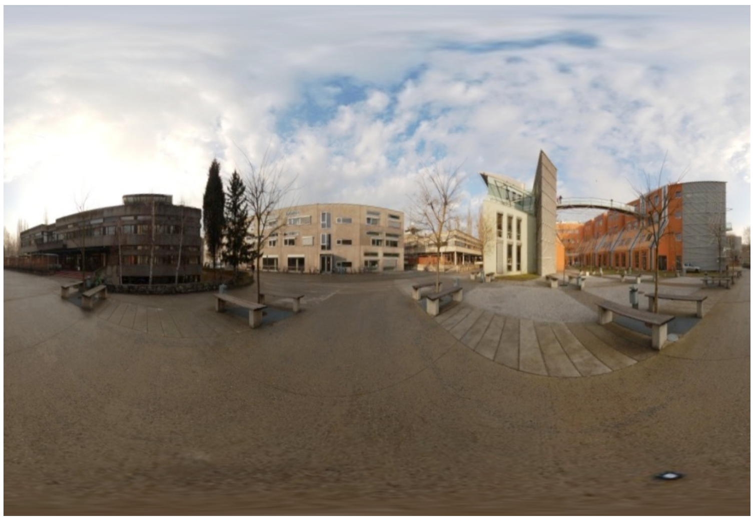


Figure 6.67: One of the 360 panoramas acquired for viewmorph, here in a cylindrical mapping.

View interpolation of wide angle or panoramic photos taken from many positions has a great potential to quickly visualize small complex real world scenes. An automatic image registration is an essential part of the workflow.

6.3.7.3 Surface Mesh Based Rendering

A comparable solution is to generate a densely meshed surface and use view dependent texturing. The result of a semi-automatic scan using *Autodesk Photofly*, shown in the image below, demonstrates the strengths and weaknesses of this technique. Especially for reflecting or untextured areas, no geometry is generated at all, resulting in holes. We expect that the view interpolation has less visible artifacts. About 40 photos were used for this model.

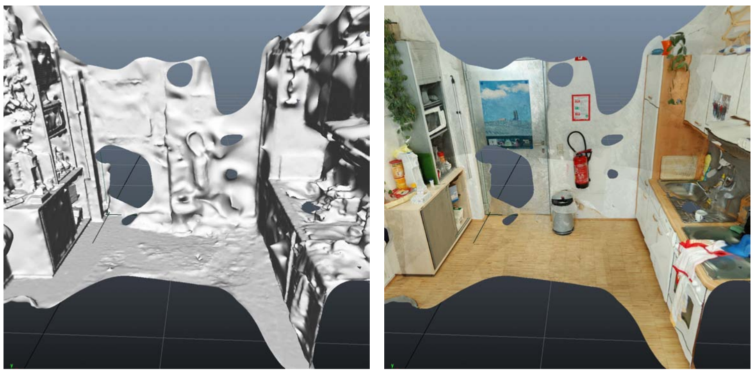


Figure 6.68: A photographically acquired scene with about 40 hand held photos. The automatic reconstruction failed and a number of manual reference points had to be specified to obtain this result using *Autodesk Photofly*.

Better results can be achieved, obtaining a dense reconstruction in real time using video images [ND10] or by using additional depth sensing capabilities, e.g. of the kinect sensor. *KinectFusion* is a project that aims for real time scanning with the kinect.

6.3.8 Simulation Result Visualization

Vector data in 3D space, like for fluid and gas simulation, is not easy to visualize. Just drawing arrows is not enough, as the arrows in the front occlude vectors further away and there is simply too much data to get an overview. A clipping volume around the user's head is very helpful. Further ideas are presented in [FRK11]. Also, transparency may help to hide arrows that are less important.
Another idea is interactive placement of smoke or particles to visualize a flow. This method is used in real wind tunnels but can also be applied for visualizing a simulation. In that case, a particle source is interactively placed. The particles have a short life time and are transported along the vector field. A well working example is presented in [SBK06], [BRKE*11], using spheres and tubes for visualization.
Similarly, rendering of molecules shows a high depth complexity. Stereoscopic viewing gives a much better impression than a monocular view. Different views are implemented and the application was ported to OpenSG and also used on the multi touch table.

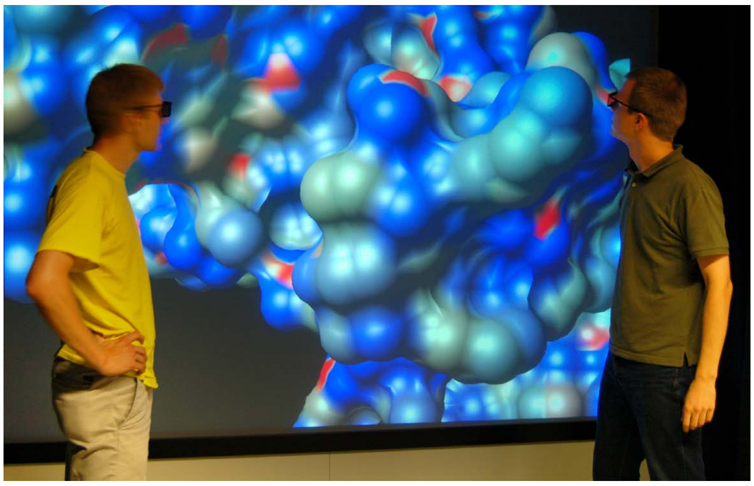


Figure 6.69: The *bio browser* for molecular visualization is a good example to show the advantage of stereoscopic rendering.

6.3.9 Games

The adaption of the game Quake III Arena is already mentioned in section 5.1.1.4. Unfortunately, a few issues remain. The navigation is rather hard to learn and use. Even though the user has more options for control, i.e. independent viewing and shooting directions, it is much harder to play the game in the DAVE compared to the desktop setup. Instead, we prefer to show the level viewer to the visitors of the DAVE. With the current implementation, the joystick index needs to be specified in a configuration file. This number may change with every reboot of the server.

6.3.9.1 Aquatic World: Virtual Fish

Aquatic World is a DAVE and desktop application showing behavioral animated fish. The fish can hunt or flea, eat, or explore. Some can even form a school of fish. A ship wreck and a few slowly moving plants are placed on the seafloor. Unfortunately, collision avoidance for the fish is not implemented yet. An issue that we fixed late was the use of the standard OpenGL fog, using the distance into the medium to reproduce a scatter effect. However, the used value is just the part of the distance perpendicular to the viewing direction, so that the visibility is infinite looking 90 degrees to the side. We fixed this by computing the Euclidean norm in a fragment shader.

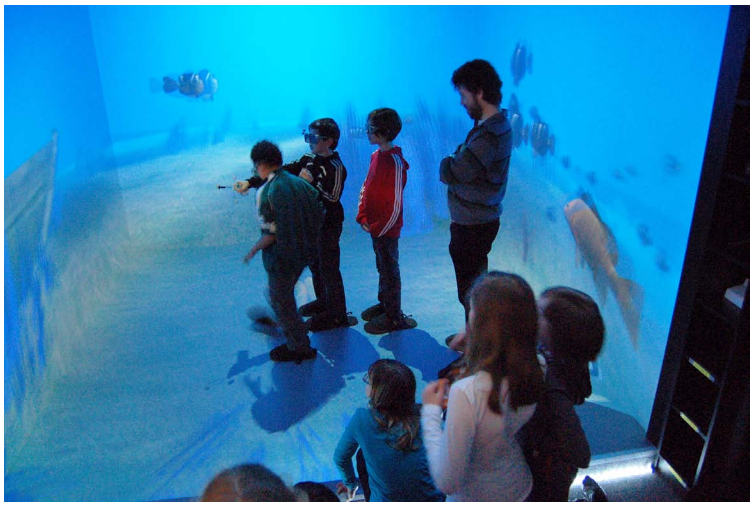


Figure 6.70: *Aquatic World* with virtual animated fish.

More details can be found in the bachelor's thesis of Martin Kandlhofer [Kan07].
Fog or similar effects should be realized using the euclidean distance in a custom shader to avoid artifacts. Animated objects make the scene more interesting. Already a simple soundscape or music in the background helps to convey a certain mood.

6.3.9.2 PPRacer

This free game, a successor of *Tux Racer*, was ported to the DAVE with the help of the Davelib. Changes included to start the game at once without clicking through menus, removing head up displays, and correcting the size of the sky box. Interestingly, the sky box was initially very small but since the depth buffer was not used, this was not noticeable in a 2D window. This example shows that there are situations where code was not written with stereoscopic display in mind. This is also a problem for OpenGL drivers trying to generate stereoscopic images. With the help of the Davelib the correct projection matrix is set and the important game states are synchronized over the network. The tracked head position is used to control the character, a penguin sliding down a slope (see section 5.1.1.4). When lying down, the eye come close to the floor projection screen. In that case, the pixels appear large and the accommodation error gets so high that it is uncomfortable to use and quickly leads to eye strain. A better pose is to lean on the elbows.
For more details, please refer to our publication [SRO*08].
We also used this application with minor modifications for a BCI experiment (see section 6.3.10.1).
Applications that were not written with immersive or stereoscopic rendering in mind usually need adaption. A too small skybox is a common example.

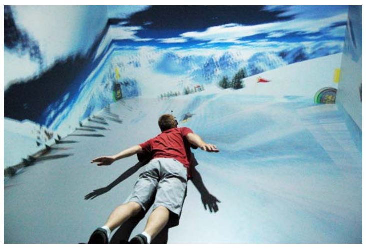


Figure 6.71: PPRacer, an open source game ported to the DAVE with the Davelib. The penguin is controlled by the user's head position.

6.3.9.3 Glider

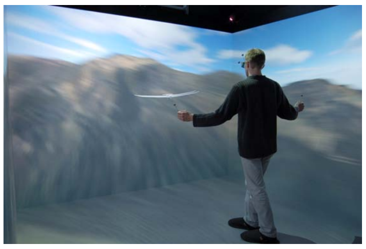


Figure 6.72: Virtual glider simulation, optimized for quick learning and easy usage.

A virtual glider simulation was developed, that allows users to fly over a landscape. The focus is an easy way to control the glider. The user's arms represent the wings and are tracked by a marker on a stick in each hand. The user can move the arms similar to kids playing airplane. Lowering the right arm and lifting the left arm leads to a curve flying to the right. When stepping forward, the glider goes down, stepping back leads to an upward motion (see also section 5.1.1.4). The simulation is very error tolerant and not physically correct, in order to avoid stalls and crashes and to achieve a short learning phase. For a good feedback, a 3D model of the glider is displayed that executes the steering commands. An automatically following camera keeps the glider visible. Yaw and pitch are applied but not rolling, so that the user does not loose balance.
The application is realized with OpenSG and features a dynamic terrain engine with height maps and satellite imagery data of a large area in Tyrol. The height values are exaggerated by 100%, so that the mountains look more interesting. The terrain engine handles level of detail display and dynamic loading or deletion of the tiles. At the moment, the data is loaded by the master PC and distributed over the network, resulting in occasional glitches. Instead, each PC could also load the data from the local drive for a faster access.
An idea to transform this application into a game is to use rings in the air as waypoints and fly a course through these rings.
For more details, please refer to our publication [SRO*08].
Large data sets that are dynamically loaded should not be distributed via the scene graph. Instead each machine should locally load the data upon request by the scene graph system. The control of the glider is greatly simplified by using a following camera. As expected, an attempt with the viewpoint in the plane (without actually rendering the plane) showed to be harder to control.

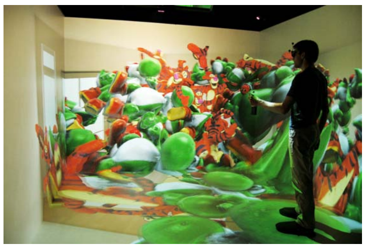


Figure 6.73: Playing with the physics engine with many objects. Disabling gravity is a fun additional feature.

6.3.9.4 Physics Engine

Skill games with balancing and stacking objects or simple shooting games can quickly be realized using a physics engine. For a short time, it is fun to experiment with a few objects even without game elements like rules or a task.
Even simple skill games are an interesting application that may also be used for educational purposes.

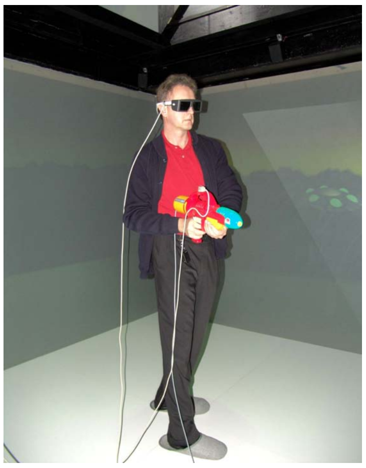


Figure 6.74: Playing *Davender*, a game for the DAVE. Space crafts .

6.3.9.5 Shooting Games

An early simple game called *Davender* shows space crafts flying around the center. The player simply has to shoot those vessels before they come too close by pointing a pistol and pressing the trigger. We modified a water pistol and connected a sensor for the trigger and added LEDs for a weapon heat feedback. With each shot, the weapon heats up and only slowly cools down. When it is too hot, firing is temporarily disabled.
A new, similar game was made up with a more friendly setting. Balloons slowly rise from below and must be shot before they reach a critical height. This game can also be played on the HEyeWall, when unwrapping the scene from a cylinder to a planar wall. Balloons can be popped by tapping on the screen. In that way, multiple players can play simultaneously on multiple devices. With small additional rules,

an interesting collaborative or competitive game can be made. As an example, each user could have an own color of balloons. Shooting balloons of the wrong color could be penalized. We are in the process of implementing such a game with instantreality.

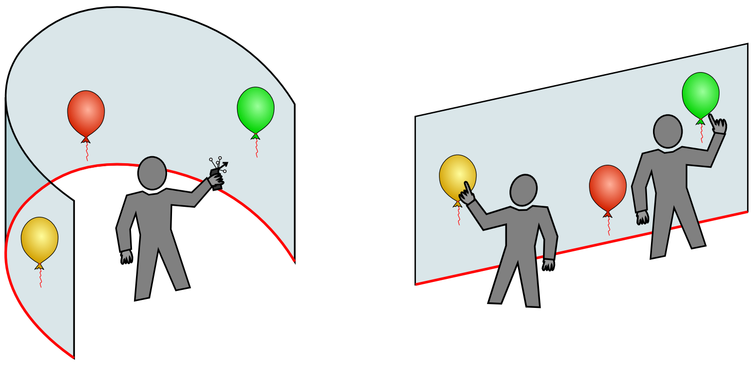


Figure 6.75: A concept of a balloon shooting game with the geometry for the DAVE (left) and the HEyeWall (right). Here, multiple people play the game simultaneously on two different setups. New balloons are spawned on the red line.

6.3.10 Psychological Experiments and Simulations

For user studies and psychological experiments, real world tests are sometimes dangerous, take a long time to setup or are expensive. In some cases, it is very hard to control the situation for reproducible testing conditions. In a number of such situations the DAVE may be used, allowing circumvention of the problems above. For each individual case it must be decided to what extend results from a test in the DAVE are applicable to real world scenarios.

6.3.10.1 Brain Computer Interface in the DAVE

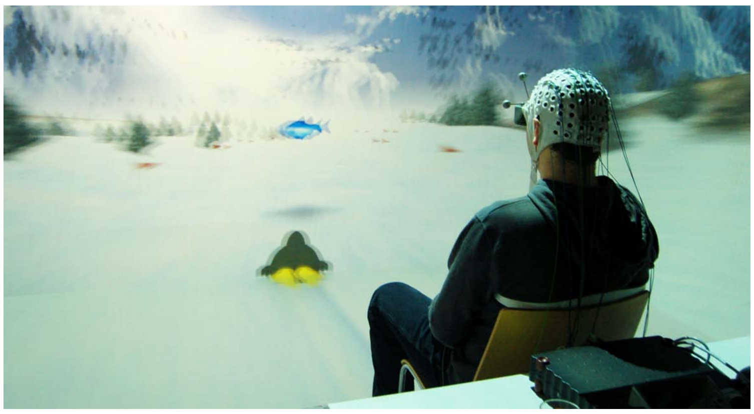


Figure 6.76: A modified version of the PPRacer linux game. The test subject controls a penguin sliding down a snowy slope, collecting fish.

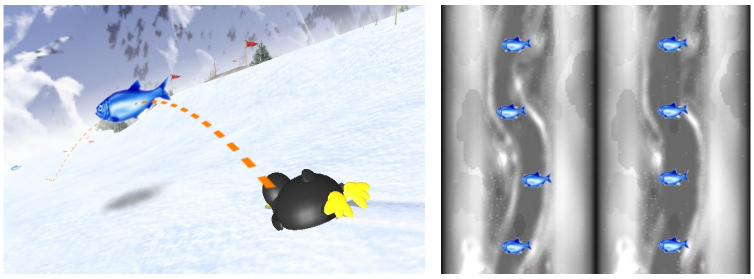


Figure 6.77: We modified the game and moved the fish above ground. Using the Brain Computer Interface (BCI), the jump is controlled by thought to collect the fish. The BCI signal has a delay, so subjects must start concentrating on a previously learned activity a couple of seconds in advance. Two custom levels are shown, without (left) and with (right) additional steering with a gamepad.

Another related experiment [LSFP07] was conducted earlier using the National Library model (also see Figure 2.32).

6.3.10.2 Navigation Tasks

For the Psychology Department, we helped setting up a stereo projection with our 3D tracking system and provide software and configuration for showing 3D models. One task is to test brain activities during a navigation task through a maze [KLFN10]. Another experiment uses the 3D model of the campus Inffeldgasse. The advantage is here that tests can also be done in the real environment and compared with the virtual counterpart. For the latter project we decided to use instantreality. In that way, we hope to be able to quickly adapt to changes or new similar tasks in the future. The missing functionality is added by plugins and JavaScript code. As examples, the requirements include a log file of the user position, a parallel port signal when approaching a sign (for synchronization with EEG records) and disabling the navigation for a few seconds at the start. Also, the psychologists wanted to place and exchange signs on their own to plan and modify the test conditions. Most requirements were easy to fulfill, while others took a lot of time.

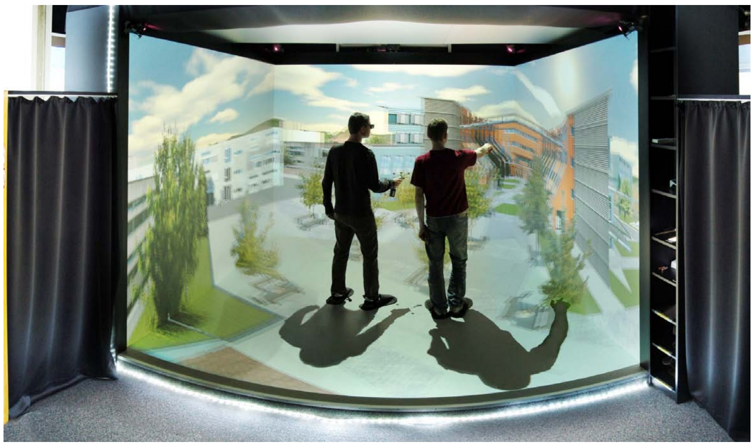


Figure 6.78: Model of the campus, also used in psychological experiments.

This is our most complex self made model up to now. We also implemented a workflow for the conversion of the model, as the model is improved from time to time. After exporting the model with the

correct settings from Maya to a VRML file, we transform this to X3D using an optimizing tool by instantreality. We then modify the texture paths with an additional tool. To include shaders and be able to reuse instanced objects, we then apply an XSLT transformation to replace the specified nodes. The last two steps may be fused into a single step in future.
For more details, please refer to our publication [KLFN10].
In such experimental setups, scripting is very well suited, as small modifications are often easy to achieve.

6.3.10.3 Efficient Signage - Imitate

One research project where the DAVE is used is called *Imitate*. Signage placement and indirect guidance by lighting is tested with the example of a train station and the connecting public transport system. Experiments can be repeated with the same conditions easily, opposed a real train station where conditions may quickly change. It is still an open question, how useful the results will be. It is unclear how well walking distances can be judged. Ideas include a physical treadmill, walking in place [SUS95] where the user lifts the feet alternately in order to navigate, or travel by joystick while moving the camera up and down corresponding to virtual steps, similar to some first person shooter games.

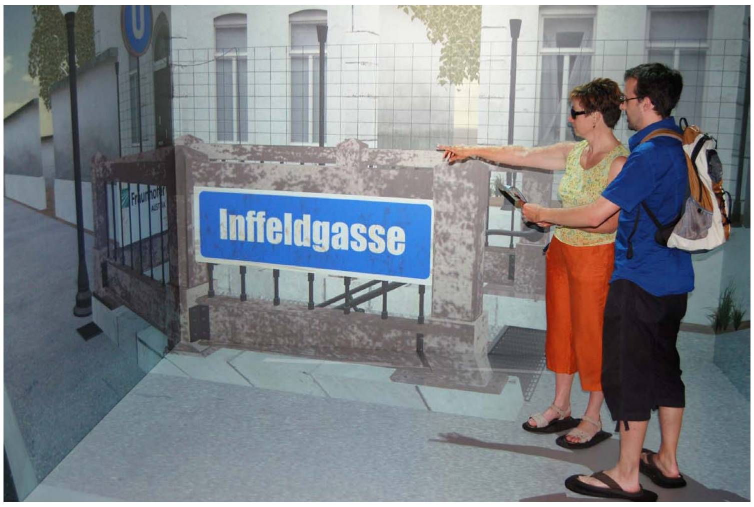


Figure 6.79: A concept model, illustrating the *Imitate* project in that signage placement is verified.

6.3.10.4 Vehicle Simulation

Unfortunately, testing cockpits or dash boards of virtual vehicles in a CAVE seems not to be such an ideal application. Due to the vergence-accommodation conflict, very close objects may lead to serious eye-strain and the depth perception is not as convincing.

Aircraft Visualization. An aircraft designer provided us with the geometry of e-Go, a not yet built one person plane. One interesting aspect for the designer is, how large the cockpit feels.
Due to accommodation problems, close objects are usually not well suited for the DAVE. To minimize eye strain, they should be placed close to the screens if possible.

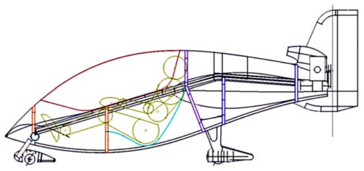


Figure 6.80: The e-Go one person aircraft fuselage from the side. (modified from image in public domain by Tony Bishop)

Car Simulator. The Institute of Automotive Engineering at TU Graz plans to construct a car simulator. If the funding and room requirements can be met, the plan is to mount parts of a real car on a hexapod motion platform. A wide angle autostereoskopic display is envisioned for visualization (see section 2.2.5.2). The goal is to be able to develop driver models and run experiments for emergency assistance systems with many test subjects. In a real car, such tests are lengthy to prepare and expensive, so that e.g. only a handful of people can be tested. In the simulator, ten times more test subjects are feasible, much better representing different types of drivers. Our part is to realize the display and visualization.

6.3.11 Visitors

An important task of universities is to educate the general public by spreading knowledge and research results, also demonstrating the necessity of research and encouraging scientific interest, especially for children. The DAVE is a tool making this particularly simple. Almost all visitors are amazed by its capabilities and realism.
With an estimated number of around 2000 visitors in the past five years, we gained a considerable amount of feedback, e.g. on our interaction methods that we progressively refine to adjust to the user's demands. A majority of visitors are first time users. Thus, it is important to provide interaction that is very easy and quick to learn and describe.

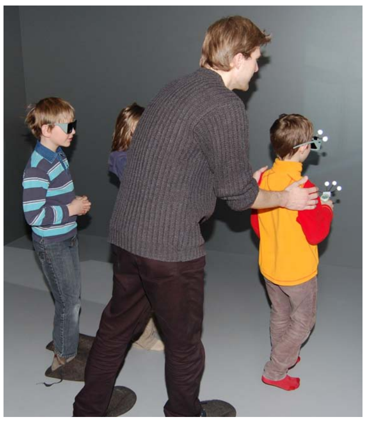


Figure 6.81: Some visitors, especially younger children, may loose balance and may need support.

Children. Young children under the age of about five years tend to start crying very quickly. We have not investigated further for reasons. Also, young children typically need to hold the glasses which are too big for them. With many school classes visiting, usually with an age from eight to ten years old, we realized that the kids become very excited and active. It is important to calm them down, stopping them from accidentally walking into the physical screen. For tough situations, we either quit the demo or turn off the display, i.e. we switch to an application that shows a black image on all projectors.

Safety. For several reasons, we only allow access to two or three people at a time, while the rest of the group can watch from behind. Fewer people are easier to take care of. Often, a user has to be stopped from accidentally walking or poking through the screen, or hitting a neighbor with the joystick while navigating. In some cases users may loose balance. Especially when flying through a wall for the first time, children may fall backwards out of reflex. Users that report dizziness are encouraged to sit down on a chair or go outside.
About once a year, a carbon stick on the joystick or the glasses breaks because people accidentally drop a device. In that case, the stick needs a replacement and the target must be recalibrated.

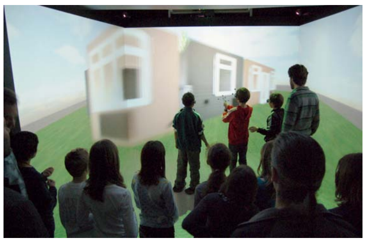


Figure 6.82: *KinderUni* educational workshop for children. We usually allow only three visitors to enter the DAVE.

Health. For the DAVE or other setups where glasses are passed around, there is a danger of spreading a viral or bacterial eye infection called *pink eye*. However, in our case the danger is low, as the virus or bacteria can survive on the equipment for a maximum of five days. We only occasionally disinfect our equipment, as frequently such a time passes between demos. However, for frequented places like 3D cinemas, this is a serious issue and glasses should be disinfected after each use.

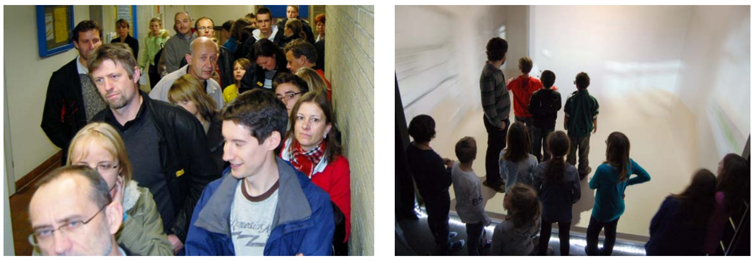


Figure 6.83: Only a single user can control and correctly view the 3D scene in the DAVE. As minimum time, one minute per visitor makes sense. With many visitors this leads to long waiting times, as here at the *Lange Nacht der Forschung*.

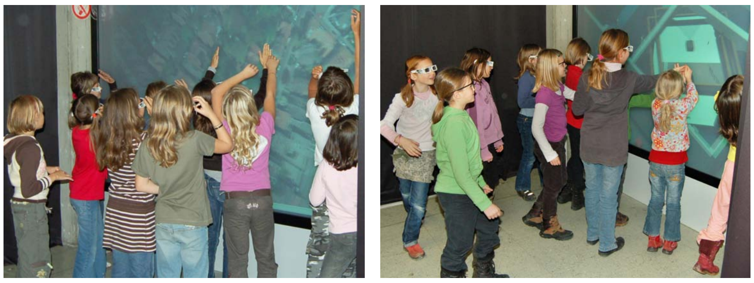


Figure 6.84: The children instinctively test if eyes tell the truth, but no haptic feedback is available.

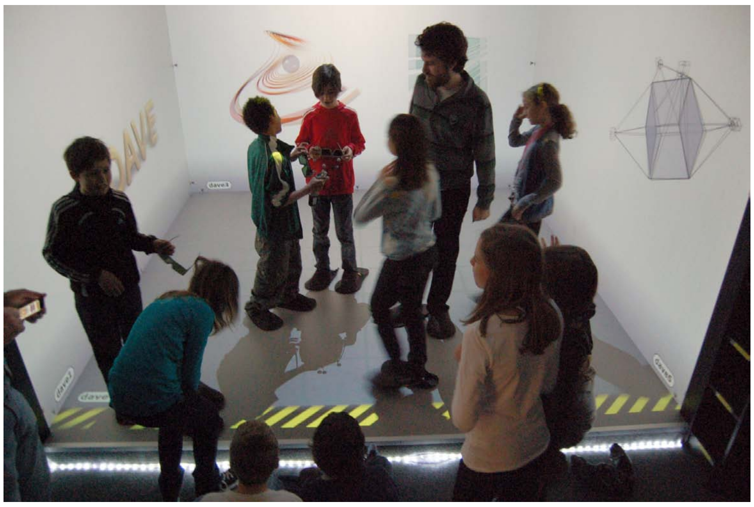


Figure 6.85: We have a step that is also marked by black yellow stripes. Later we added a room light in form of an LED strip. This enables basic orientation in the DAVE even when the projectors show black, while it does not influence projection contrast or the tracking system notably.

To avoid injury and broken equipment, only few visitors should be allowed in the DAVE. Operators should stand close to the principal user, ready to quickly grab and stop him from walking through the screen.

6.3.11.1 Education and PR

It is very hard to convey the immersive experience in the DAVE with words or 2D images, we encourage everyone to try it out in person. To reach many people and gain new users in industry and research, we have a regular visiting hour once a week. A nicely looking flyer was designed and lots of information is provided on our homepage. This includes our video publication in 2008 [LSF08].
For larger groups of more than about ten visitors, we split up the groups. While ten people are in the DAVE, we explain the technology and show demos to the rest of the group on the rear projection wall.

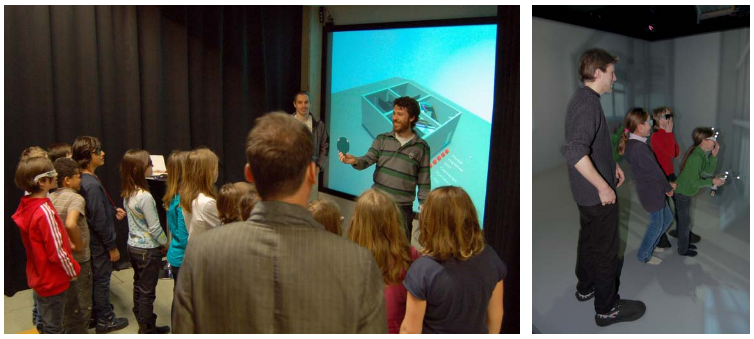


Figure 6.86: *KinderUni* educational workshop for kids. We usually present different technologies for stereoscopic viewing and explain and show head tracking. Instead of just presenting, we let them try different effects themselves.

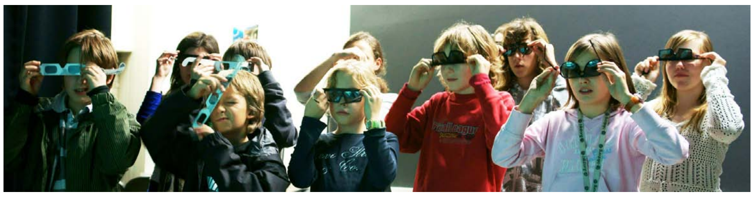


Figure 6.87: In this experiment, the children try out what happens when the polarized glasses are turned. We also ask them to move around to experience the effect of missing head tracking or motion parallax.

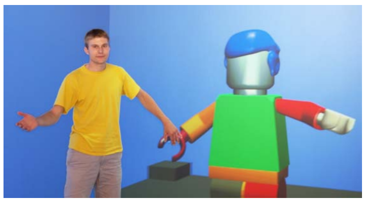


Figure 6.88: While this photo looks good for 2D media, it is misleading for non-experts. The user as shown here has no haptic feedback, sees a distorted view and lacks of 3D glasses. Often, photos are edited, e.g. to add a 3D object that extends over the screen borders.

6.3.11.2 TV cameras, filming, photography for publication and press

It is hard to convey the 3D impression with 2D media. To avoid double images or flickering that would result from a not synchronized filming of a time interlaced stereo image, we usually switch to a monoscopic display. For projectors showing the color channels sequentially, the exposure time should be set to a multiple of the frame duration to avoid color deviations. For a correct perspective for the camera, we sometimes attach the head tracking target to the camera. Filming a user in such a way, the user sees a distorted mono image and may have difficulties with interaction tasks like picking, but the resulting photo or video looks better. For one scene in the DAVE movie, we used a tracked moving camera to film the first person's view of a user walking through an appartment.

6.3.12 Lessons learned

We have implemented and ported many programs using the Davelib, OpenSG and instantreality. New insights came with almost each application or even just different models in the same application.

Suitability for the DAVE. To judge whether an application is well suited for the DAVE, several questions come to mind.
Is a wide angle of view required? For looking at a small object from outside, a single stereo wall might be sufficient.
Is there a reasonable amount of three dimensional structure? Objects too far away will not give a good 3D impression, e.g. a solar system in its real dimensions.
Are objects very close? Too close 3D objects are not well suited, like a dense set of vectors or a tight cockpit. We identified the vergence-accommodation conflict as a major technical problem of CAVE technology today. Another tricky part for displaying close objects is correct occlusion. In a CAVE for multiple users or for close objects, occlusion is wrong, as people and the own body always cover the screen. A solution may be an additional see-through HMD.
Is the rendering technology capable of displaying the data in the necessary quality? The rendition capabilities of colors and dynamic range are rather limited.
Is the interaction suitable? Some applications may not work well without correct haptic feedback. An example are tasks requiring precise input, where a mouse is particularly good at. Even if the tracking is precise enough, it is very hard to move the hand precisely in mid air. Further issues concerning correct haptic feedback remain.

Negative Example. A negative example of an application for the DAVE is free form sketching. Close objects lead to eyestrain, occlusion is not correctly handled and imprecise input render such application unusable.

Misperceptions. An often discussed problem of the CAVE is a distorted perception of size. However, we did not strongly experience such deficiencies ourselves. For the application of indoor architecture we discussed and tested the usage of an HMD instead of a CAVE. A small field of view, latency and heavy gear on the head lead to a worse impression. In contrast to the DAVE, with the HMD we did not immerse the same way, in particular we did not feel capable of judging the size of a room. We suspected that the main cause is the small field of view, but a quick experiment looking through a pair of cardboard tubes also restricting the field of view by a similar amount showed that this is not the (only) reason. Among experts this phenomenon is still an open question.

Central Media Distribution. OpenSG and instantreality normally distribute all content via network to the render clients. When large amounts of data need to be loaded dynamically during usage, this concept is not so well suited. When flying over a large terrain as in the glider application, data for the new tiles must be loaded and distributed over the network. Blocking can easily occur, which is

unpleasant for the user's experience. A better way is that all clients load the necessary data locally.
Another example is high resolution video playback on the HEyeWall with instantreality. For a FullHD (1920×1080) video, the maximum frame rate is about 1 Hz. We assume that the main application decodes the video stream and sends the new texture image in an uncompressed or weakly compressed format over the network. Much more efficient is the approach that every PC loads and decompresses the video frame on its own. For maximum performance, the video may be cropped to only the necessary region for each individual PC in a preprocessing step. While OpenSG is not designed for such an approach, special code can be written to be able to locally access resources. In InstantReality, to our knowledge this is not possible at the moment.

7 Conclusion

The achievements of the work described in the previous chapters are discussed and set in relation to the goals stated in the introduction. Limitations and areas for future work are given.

7.1 ACCOMPLISHMENTS

7.2 CONTRIBUTIONS AND BENEFIT

This work presents three major contributions on the way to fulfilling the goals mentioned in the introduction.

First, many improvements and work on the DAVE allow immersive exploration and limited modification of 3D worlds even for novice users without computer knowledge. The software framework Davelib is fully rewritten to provide a lean way to port applications to VR setups, a cost effective optical 3D tracking is realized, usability for operators is improved significantly and a number of different applications are implemented to showcase the DAVE.

Second, the HEyeWall Graz introduces a new concept for a large seamless display. The potential of multi touch hardware and interaction on the HEyeWall and on other setups is demonstrated.

Third, in the field of Computational Photography, a few specialized applications are explored. Image alignment methods for image stacks and multi resolution images are presented.

The DAVE had about 2000 visitors in the past five years, and our multi touch setups were shown on a number of exhibitions and fairs, demonstrating that our research is not purely theoretical but that practical demands are also met.
When building the DAVE in Graz, we had hoped to receive requests for use from other research groups or the industry. We suspected that the existence of the DAVE may have been largely unknown, so we sent flyers to each institute, invested a lot of time in weekly demonstrations and arranged for further publications in the university news. Unfortunately, very little feedback was received from our efforts. The same is true for our efforts to collaborate with architects and the faculty of architecture.
Many potential users may be afraid that a VE is very complex to use and program. The main purpose of the Davelib is to simplify the process of porting existing OpenGL applications into a VR environment. Using our library, a minimal amount of code changes is required in such an application. In contrast, most other VR frameworks require the application to adapt to *their* functionality. Nevertheless, development of new applications using the Davelib makes perfect sense, especially for prototypes or experimental software that is sometimes hard to realize in large all-in-one VR frameworks.
A number of multi touch applications were tested. But are they really useful or just fun and games? In fact, it is not easy to find a new multi touch application that does *not* work without such an interface. But considering multiple simultaneous users on a single large screen, multi touch makes perfect sense, as no input device has to be handed over from one user to the next. However, this is still a very special

scenario. We found that for our own every day work, we do not go to that room and power on the multi touch setup. It may be different if the hardware was already integrated in our desktop monitors or the office also had a large screen. The currently best suited applications for such multi touch setups are presentation, e.g. as attraction on an exhibition, and selling, e.g. a collaborative configuration of a car or a bank contract with clients and salesmen.
Within the last decade, many people moved from analog to digital photography. However, the potential is hardly used. Work is presented to help organize and browse databases in a quick way. Multi resolution images are presented, a simple idea which has received surprisingly little interest yet. To circumvent physical limitations of photo cameras, image stacks can be used e.g. for HDR acquisition. We present image alignment with free-form deformation that consequently allows use these applications for the common case of not having a tripod at hand.

7.3 LIMITATIONS

7.3.1 Feasibility and Effort

From the researcher's point of view, many applications for the presented technologies are possible today, but feasibility is another important factor for the actual usage. VEs are still rare and for many applications, immersive visualization does not add a sufficient benefit to justify the costs and effort for a VR setup. Application development and maintenance is a considerable part. These may be the reasons why VR became somewhat out of fashion after a hype with unfulfilled expectations one to two decades ago. For example, virtual tourism or archaeological reconstructions and exhibitions of a past time are great but rare applications for immersive VR environments.
Also, multi touch setups need special development to create intuitive usage. Today they are hardly used for everyday purposes, but mostly to show off the innovation. Applications that do not work without multi touch hardware are rare and often constructed for the purpose of demonstrating the technology itself. While they may be less used for productive work, they may enable or enhance social interaction and activities.

7.3.2 Technical Limitations

7.3.2.1 DAVE

Display Hardware Limitations. For stereoscopic displays, the vergence-accommodation conflict poses a major technical challenge that remains to be solved. In commonly used setups for immersive screens with two different images per eye, objects at a distance with a significantly different accommodation than the screen lead to serious discomfort or eye strain. So far, only volumetric or holographic displays can achieve a correct effect by attenuating or emitting light at the respective position in 3D space. Many restrictions, large hardware and rendering efforts and a high price are common drawbacks of these displays. We suspect that the vergence-accommodation conflict may also be the cause for reports on mismatch of the perceived size in VR.

For both immersive and non-immersive visualization, a major challenge is the introduction of HDR imaging to the consumer market. Interestingly, this seems to receive rather little attention by the industry. While LCDs with LED matrix backlights are a good approach for monitors, a feasible solution for HDR projection is less clear. Digital cameras usually capture images with a bit depth of more than 8 bits per color channel already. The hardware interfaces and the software in between cameras and displays is the limiting factor, the operating system is an example.

The photo below shows us developers, mentally comparing the real Austrian National Library State Hall to our 3D model. While the amount of perceived geometric details and materials is certainly different, the most striking differences is probably the dynamic range.

Figure 7.1: Contemplating developers in the real National Library State Hall.

Haptics and Acceleration. Taking a step back, what other applications would be really useful in a virtual environment? A great achievement would be a sports simulator, allowing effective and safe training. Teaching, practicing and feedback are expensive and take a long time, not only for competitive athletes but also for recreational athletes. In addition, practicing might be dangerous and often stops people from trying at all. In fact, a few simulators already exist which are useful to some extent, like a golf simulator with a real club and a virtual ball. In contrast, some other sports are too challenging. Examples are: kite surfing, wind surfing, wave surfing, skiing, mountain biking, kayaking, pole vaulting or dancing. There even are a few specialized haptic simulators, like for swimming and kayaking, but they just allow the user to move in a similar fashion to the real motion. Their haptic feedback is extremely limited and nowhere near the real experience, not allowing the user to learn and practice in the simulator. Instead, it is merely suited for training the muscles. We have little hope that haptic feedback will soon improve up to a level in order to fulfill the requirements for these mentioned applications.

Valuable Improvements. The optical tracking system hardware and software can be updated. Especially a hardware synchronization of the cameras and an increased amount of light will help to make tracking more robust in case of fast motion. Another option is to replace parts or the complete system by a now affordable commercial alternative.
The implementation of the IO device library based on Lua scripting is the last part to complete a mostly setup independent program development for VEs. A deeper understanding of Lua will be necessary to complete the library for a simple development of efficient software. With instantreality, a similar idea can now be realized with the help of plugins.
The completion of the implementation of the *VRAppStarter* application for system control will replace a grown set of confusingly nested scripts in order to reduce maintenance overhead and improve usability and flexibility for the operators.

7.3.2.2 HEyeWall

Frequency Split Display. A working prototype for the novel idea of a scalable seamless high resolution display was implemented. In order to allow a shadow free touch interaction for multiple users, a rear projection is necessary. Hotspots are inevitable and result in view dependent brightness changes. An even stronger reduction of hotspots with lower gain factors may lead to an intolerable loss of brightness. Wide angle projection increases problems with hotspots, especially noticeable with the lens of the large projector. A photometric compensation makes only sense for a single head tracked user, applying a view dependent correction. This requires additional head tracking, a new calibration model, i.e. multiple calibrations from different registered views and imposing the restriction of a single user. Finally, the large projector intensity should match the intensity of a tiled projector, i.e. the Ansi lumen of the large projector should be similar to the sum of the Ansi lumens of all tiled projectors. This reduces a feasible scalability of the setup.
A more promising but still partial solution seems to be a projector array with large overlaps. To reduce color non-uniformity, a fast automatic photometric and colorimetric state-of-the-art calibration seems helpful. New light sources like Laser & LED hybrids for DLP projectors may lead to a much better color consistency and solve the problem of colorimetric differences in that way.

Multi Touch Interface. The current FTIR multi touch hardware on the HEyeWall allows acceptable input for tapping or short drags, but works less reliably for extended drags or more complex interaction. Possible improvements are a more homogeneous silicone coating, hot mirrors in front of each projector lens, synchronized pulsed LED lighting or the replacement of the lighting setup by line laser modules and matching narrow bandwidth filters for the cameras. The additional use of a low powered infrared laser pointer should be considered for larger scale interactions. Input with a kinect sensor may be an interesting area for further research.

7.4 FUTURE WORK

Computer Graphics. One big remaining challenge in rendering algorithms for VR is real time global illumination. This is also important for the game industry, and continuous efforts are made to come closer to a unified solution. In that context, more realistic materials will be important.

Hardware. To improve the level of immersion or presence, reducing tracking latency and the vergence-accommodation conflict, an increased field of view and display resolution as well as HDR capabilities may be even more important than a photo realistic rendering.

VR Frameworks. A next step is the development of cross setup applications. Good software engineering is necessary for a flexible and easy incorporation of the diverse input hardware. A commonly accepted configuration for the displays of VEs will help to spread such applications. *instantreality* is a partial solution, using the X3D file format as scene description. Closed source code as well as unfinished implementation and documentations leave the user dependent on the support by the few developers.
With better support of the tools that architects use, model export and lighting workflows will become much more affordable. We expect architectural previews to eventually become a killer application for VR.

Appendix A

Publications and Collaborations

This thesis is partially based on the publications and talks listed in this appendix.

A.1 PUBLICATIONS

[GLB05] Grasset R., Lancelle M., Billinghurst M.: *Mining Data 3D Visualization and Virtual Reality: Enhancing the Interface of LeapFrog3D*. Tech. rep., HITLabNZ, 2005. Technical report.

[HSLF07] Havemann S., Settgast V., Lancelle M., Fellner D. W.: 3D-powerpoint - towards a design tool for digital exhibitions of cultural artifacts. In *VAST* (2007).

[KLFN10] Kober S., Lancelle M., Fellner D. W., Neuper C.: Cortical correlate of spatial presence in 2-D and 3-D interactive virtual reality: An EEG study. In *RAVE-10, Barcelona, Spain* (2010). Talk.

[LB04] Lancelle M., Billinghurst M.: *Public Interactive Large Screen Displays*. Tech. rep., HITLabNZ, 2004. Technical report.

[LF04a] Lancelle M., Fellner D. W.: *Automatische Generierung und Visualisierung von 3D-Stadtmodellen*. Master's thesis, Technische Universität Braunschweig, 2004. Master's thesis.

[LF04b] Lancelle M., Fellner D. W.: Current issues on 3D city models. In *Image and Vision Computing New Zealand* (2004). Poster.

[LF11a] Lancelle M., Fellner D. W.: Smooth transitions for large scale changes in multi-resolution images. In *16th International Workshop on Vision, Modeling, and Visualization (VMV)* (2011).

[LF11b] Lancelle M., Fellner D. W.: Soft edge and soft corner blending. In *Workshop "Virtuelle & Erweiterte Realität" (VR/AR)* (2011).

[LH03] Lancelle M., Havemann S.: Silhouettenbasiertes 3D-Scannen, 2003. Bachelor's thesis.

[LOUF06] Lancelle M., Offen L., Ullrich T., Fellner D. W.: Minimally invasive projector calibration for 3D applications. In *Dritter Workshop Virtuelle und Erweiterte Realität der GI-Fachgruppe VR/AR* (2006).

[LSF08] Lancelle M., Settgast V., Fellner D. W.: Definitely Affordable Virtual Environment. In *IEEE Virtual Reality Conference* (2008). Video.

[LSHF09] Lancelle M., Settgast V., Havemann S., Fellner D. W.: Spatially coherent visualization of image detection results using video textures. In 33rd *Workshop of the Austrian Association for Pattern Recognition (AAPR/OAGM)* (2009).

[RSW*11] Roth P. M., Settgast V., Widhalm P., Lancelle M., Birchbauer J., Brändle N., Havemann S., Bischof H.: Next-generation 3D visualization for visual surveillance. In 8th *IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)* (2011). Poster.

[SRO*08] Steiner M., Reiter P., Ofenböck C., Settgast V., Ullrich T., Lancelle M., Fellner D. W.: Intuitive navigation in virtual environments. In 14th *Eurographics Symposium on Virtual Environments, Eindhoven, Netherlands* (2008). Poster.

A.1.1 Conference Talks

This is a list of conference or workshop talks on own work that are not covered by the list of publications.

- Multi Touch, Winter Augmented Reality Meeting (WARM) 2010
- "The HEyeWall - a 4m x 2m multitouch screen" , Winter Augmented Reality Meeting (WARM) 2009
- DAVE/VR at CGV, Wintergraph 2006

A.1.2 Lectures

- Photo Realism in Computer Graphics, lecture, summer term 2010, with Thomas Schiffer
- Simulation and Animation in Computer Graphics, lecture, summer term 2009, with Volker Settgast and Eva Eggeling
- Simulation and Animation in Computer Graphics, lecture, summer term 2008, with Volker Settgast
- Introduction to Photography Artifacts and Computational Photography, annual talk from 2007 to 2011
- Educational workshops on Virtual Reality were held from 2007 to 2011 within the *KinderUni* and *JuniorUni* children's university programs.

A.2 SUPERVISING ACTIVITIES AND COLLABORATIONS

A.2.1 Supervised Student Theses

The following list summarizes the successfully completed bachelor, diploma and master theses and master projects supervised by the author. Some results of this work are partially used as an input for this thesis.

[Fal11] Falkensteiner A.: Reflective facade display, 2011. Master project, co-supervised with Sven Havemann.

[FF09] Feuerstein M., Fetzel R.: Multitouch screen, 2009. Bachelor's thesis.

[Fra10] Fraser L.: 3d painting in the dave, 2010. Master's thesis, co-supervised with Volker Settgast.

[KÖ7] Köstinger M.: Motion capturing in the dave, 2007. Bachelor's thesis.

[Kan07] Kandlhofer M.: Aquaticworld, 2007. Bachelor's thesis, co-supervised with Volker Settgast.

[Kap09] Kapeller G.: Multitouch sketch and gesture recognition, 2009. Bachelor's thesis.

[Mar10] Marius G.: Image warping for hdr stitching, 2010. Bachelor's thesis.

[Rum09] Rumpler M.: Local image alignment, 2009. Master project.

[SR08] Steiner M., Reiter P.: Virtual glider, 2008. Bachelor's thesis, co-supervised with Volker Settgast.

A.2.2 Collaborations

Much work presented in this thesis was realized in collaboration. The following list contains collaborators with significant contributions to the respective projects. It is organized by the chapters in which they are described in most detail.

2 – Hardware Devices. Animations on BIX media fassade: Volker Settgast, Lukas Daum, Claus Degendorfer, Lukas Fraser, Harald Grabner, Bernhard Hohmann, Ferdinand R. Knapitsch-Scarpatetti-Unterwegen, Christian Kurz, Robert Lanner, Martin Mörth, Jürgen Oswald, Philipp Reiter, Markus Rumpler, Jörg Schlager, Markus Steinberger, Marc Steiner, Martin Strobl, Philipp Michael Wagner. DAVE construction and system: Sven Havemann, Armin Zink né Hopp, Lars Schimmer, Volker Settgast, Torsten Techmann. Brain Computer Interface: Robert Leeb, Volker Settgast. HEyeWall Graz construction and system: Sven Havemann, Lars Schimmer, Volker Settgast. Stereo rear projection wall: Sven Havemann, Lars Schimmer, Torsten Techmann. Car Simulator Concept: Volker Settgast. 3D optical tracking camera housing: Torsten Techmann. FTIR multi touch prototype: Richard Fetzel, Markus Feuerstein. HEyeWall projector mounting: Sven Havemann.

3 – Input: Optical Tracking. Optical 3D tracking: Hyosun Kim, Martin Köstinger, Lars Offen.

4 – Output: Image Rendering. Davelib: Dominique Gunia, Sven Havemann. Projector calibration: Lars Offen, Torsten Techmann, Torsten Ullrich. Frequency split display: Dieter Fellner, Sven Havemann.

5 – User Interaction. Navigation by pose: Christian Ofenböck, Reinhold Preiner, Marc Steiner. Navigation with BCI: Robert Leeb. Joystick travel in the DAVE: Volker Settgast. Object manipulation in the DAVE: Volker Settgast. 2D multi touch object manipulation: Volker Settgast. Gesture and handwriting with multi touch: Georg Kapeller.

6 – Applications. Aichi mulit touch information system: Michael Herchel, David Thompson. Optical flow based image alignment: Markus Rumpler. Alignment with free-form deformation: Georg Marius. Autovista video surveillance: Sven Havemann, Bettina Könighofer, Ullrich Krispel, Peter Roth, Volker Settgast, Philip Weber. Sensor fusion visualization: Jan Becker. Score four game: Fran̗ois Bérard. Multimedia database organizer: Thomas Friedl. Museum artifact interaction: Sven Havemann, Volker Settgast. Rapid prototyping framework electronics: Marilyn Lim. MetaDesigner: Christoph Schinko, Sven Havemann. Architectural visualization: Volker Settgast. Mesh manipulation in the DAVE: Lukas Fraser. Generative modeling in the DAVE: Sven Havemann, Wolfgang Thaller. Brute force raytracer: René Zmugg. Cuda raytracer: Thomas Schiffer. Shader materials: Volker Settgast. Sphere particle shader: Andreas

Halm. Image based rendering: Ullrich Krispel. Bio browser: Andreas Halm, Lars Offen. Quake III arena game port: Christian Ofenböck, Reinhold Preiner, Volker Settgast. Aquatic world: Martin Kandlhofer. Glider: Philipp Reiter, Volker Settgast, Marc Steiner. Physics engine integration: Volker Settgast. Psychological experiments: Sylvia Kober, Robert Leeb, Volker Settgast. Imitate: Volker Settgast. Dave demos and workshops: Wolfgang Scheicher, Lars Schimmer, Volker Settgast, Wolfgang Thaller.

References

[AAM05] Amemiya T., Ando H., Maeda T.: Phantom-drawn: direction guidance using rapid and asymmetric acceleration weighted by nonlinearity of perception. In *Proceedings of the 2005 international conference on Augmented tele-existence* (New York, NY, USA, 2005), ICAT '05, ACM, pp. 201–208.

[AB06] Agarawala A., Balakrishnan R.: Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. In *Proceedings of the SIGCHI conference on Human Factors in computing systems* (New York, NY, USA, 2006), CHI '06, ACM, pp. 1283–1292.

[AF01] Ahmed M., Farag A.: Non-metric calibration of camera lens distortion. In *Image Processing, 2001. Proceedings. 2001 International Conference on* (oct 2001), vol. 2, pp. 157 –160 vol.2.

[ASLP10] Amici S. D., Sanna A., Lamberti F., Pralio B.: A wii remote-based infrared-optical tracking system. *Entertainment Computing 1*, 3-4 (2010), 119 – 124.

[ATP*10] Adams A., Talvala E.-V., Park S. H., Jacobs D. E., Ajdin B., Gelfand N., Dolson J., Vaquero D., Baek J., Tico M., Lensch H. P. A., Matusik W., Pulli K., Horowitz M., Levoy M.: The frankencamera: an experimental platform for computational photography. *ACM Trans. Graph. 29* (July 2010), 29:1–29:12.

[Aus03] Austin C.: Magic carpet, 2003.

[BBK*06] Bornik A., Beichel R., Kruijff E., Reitinger B., Schmalstieg D.: A hybrid user interface for manipulation of volumetric medical data. In *Proceedings of the 3D User Interfaces* (Washington, DC, USA, 2006), 3DUI '06, IEEE Computer Society, pp. 29–36.

[BC09] Baudisch P., Chu G.: Back-of-device interaction allows creating very small touch devices. In *Proceedings of the 27th international conference on Human factors in computing systems* (New York, NY, USA, 2009), CHI '09, ACM, pp. 1923–1932.

[BCF*08] Bowman D., Coquillart S., Fröhlich B., Hirose M., Kitamura Y., Kiyokawa K., Stuerzlinger W.: 3d user interfaces: New directions and perspectives. *Computer Graphics and Applications, IEEE 28*, 6 (nov.-dec. 2008), 20 –36.

[BEM11] Bauszat P., Eisemann M., Magnor M.: Guided image filtering for interactive high-quality global illumination. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering (EGSR)) 30*, 4 (June 2011), 1361–1368.

[BF05] Bowman D. A., Fröhlich B.: New directions in 3d user interfaces. In *Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality* (Washington, DC, USA, 2005), VR '05, IEEE Computer Society, pp. 312–.

[BGZ*06] Bimber O., Grundhofer A., Zeidler T., Danch D., Kapakos P.: Compensating indirect scattering for immersive and semi-immersive projection displays. In *Proceedings of the IEEE conference on Virtual Reality* (Washington, DC, USA, 2006), VR '06, IEEE Computer Society, pp. 151–158.

[BI08] Bimber O., Iwai D.: Superimposing dynamic range. In *ACM SIGGRAPH Asia 2008 papers* (New York, NY, USA, 2008), SIGGRAPH Asia '08, ACM, pp. 150:1–150:8.

[BIB*09] Bérard F., Ip J., Benovoy M., El-Shimy D., Blum J. R., Cooperstock J. R.: Did "minority report" get it wrong? superiority of the mouse over 3d input devices in a 3d placement task. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part II* (Berlin, Heidelberg, 2009), INTERACT '09, Springer-Verlag, pp. 400–414.

[BIS01] Buoguila L., Ishii M., Sato M.: Scaleable spidar: a haptic interface for human-scale virtual environments. In *Haptic Human-Computer Interaction*, Brewster S., Murray-Smith R., (Eds.), vol. 2058 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2001, pp. 182–193.

[BIWG08] Bimber O., Iwai D., Wetzstein G., Grundhöfer A.: The visual computing of projector-camera systems. In *ACM SIGGRAPH 2008 classes* (New York, NY, USA, 2008), SIGGRAPH '08, ACM, pp. 84:1–84:25.

[BKH97] Bowman D. A., Koller D., Hodges L. F.: Travel in immersive virtual environments: An evaluation of viewpoint motion control techniques. In *Proceedings of the 1997 Virtual Reality Annual International Symposium (VRAIS '97)* (Washington, DC, USA, 1997), VRAIS '97, IEEE Computer Society, pp. 45–.

[BKH98a] Bowman D., Koller D., Hodges L.: A methodology for the evaluation of travel techniques for immersive virtual environments. *Virtual Reality 3*, 2 (June 1998), 120–131.

[BKH98b] Bowman D. A., Koller D., Hodges L. F.: A methodology for the evaluation of travel techniques for immersive virtual environments. *Journal of the Virtual Reality Society 3* (1998), 120–131.

[BKL*09] Bowman D., Kruijff E., LaViola J., Poupyrev I., Stürzlinger W.: 3d user interfaces, 2009. Short Course.

[BKLP04] Bowman D. A., Kruijff E., LaViola J. J., Poupyrev I.: *3D User Interfaces: Theory and Practice*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.

[BMY05] Brown M., Majumder A., Yang R.: Camera-based calibration techniques for seamless multiprojector displays. *IEEE Transactions on Visualization and Computer Graphics 11* (March 2005), 193–206.

[BNB07] Ball R., North C., Bowman D. A.: Move to improve: promoting physical navigation to increase user performance with large displays. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2007), CHI '07, ACM, pp. 191–200.

[BR04] Baar J. v., Raskar R.: Flexible calibration of multiple projectors for low-cost curved screen displays. In *International Conference on Artificial Reality and Telexistence (ICAT)* (2004).

[BRKE*11] Bussler M., Rick T., Kelle-Emden A., Hentschel B., Kuhlen T.: Interactive particle tracing in time-varying tetrahedral grids. In *Proc. Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)* (2011), pp. 71–80.

[BRS05] Ballagas R., Rohs M., Sheridan J. G.: Sweep and point and shoot: phonecam-based interactions for large public displays. In *CHI '05 extended abstracts on Human factors in computing systems* (New York, NY, USA, 2005), CHI EA '05, ACM, pp. 1200–1203.

[BS11] Bruder G., Steinicke F.: Perceptual evaluation of interpupillary distances in head-mounted display environments. In *Workshop "Virtuelle & Erweiterte Realität" (VR/AR)* (2011).

[CB61] Comeau C. P., Bryan J. S.: Headsight television system provides remote surveillance. In *Electronics, Nov. 10* (1961), pp. 86–90.

[CCD*08] Cignoni P., Corsini M., Dellepiane M., Ranzuglia G., Vergauwen M., Van Gool L.: Meshlab and arc3d : photo-reconstruction and processing 3d meshes, 2008.

[CNSD*92] Cruz-Neira C., Sandin D. J., DeFanti T. A., Kenyon R. V., Hart J. C.: The cave: audio visual experience automatic virtual environment. *Commun. ACM 35*, 6 (1992), 64–72.

[CNSD93] Cruz-Neira C., Sandin D. J., DeFanti T. A.: Surround-screen projection-based virtual reality: the design and implementation of the cave. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1993), ACM, pp. 135–142.

[CPS*97] Czernuszenko M., Pape D., Sandin D., DeFanti T., Dawe G. L., Brown M. D.: The immersadesk and infinity wall projection-based virtual reality displays. *SIGGRAPH Comput. Graph. 31* (May 1997), 46–49.

[CRB10] Chen Y.-X., Reiter M., Butz A.: Photomagnets: supporting flexible browsing and searching in photo collections. In *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction* (New York, NY, USA, 2010), ICMI-MLMI '10, ACM, pp. 25:1–25:8.

[Cro92] Crone R. A.: The history of stereoscopy. *Documenta Ophthalmologica 81* (1992), 1–16.

10.1007/BF00155009.

[DCC97] Darken R. P., Cockayne W. R., Carmein D.: The omni-directional treadmill: a locomotion device for virtual worlds. In *Proceedings of the 10th annual ACM symposium on User interface software and technology* (New York, NY, USA, 1997), UIST '97, ACM, pp. 213–221.

[DD95] DeMenthon D. F., Davis L. S.: Model-based object pose in 25 lines of code. *International Journal of Computer Vision 15* (1995), 123–141.

[DDS*09] DeFanti T. A., Dawe G., Sandin D. J., Schulze J. P., Otto P., Girado J., Kuester F., Smarr L., Rao R.: The starcave, a third-generation cave and virtual reality optiportal. *Future Generation Computer Systems 25*, 2 (2009), 169 – 178.

[DDSD03] Décoret X., Durand F., Sillion F. X., Dorsey J.: Billboard clouds for extreme model simplification. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 689–696.

[dHGP08] de Haan G., Griffith E. J., Post F. H.: Using the wii balance board^{TM} as a low-cost vr interaction device. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2008), VRST '08, ACM, pp. 289–290.

[DHL01] Davide F., Holmberg M., Lundström I.: *Virtual Olfactory Interfaces: Electronic Noses and Olfactory Displays*. IOS Press, Amsterdam, 2001.

[DLR*09] DeFanti T. A., Leigh J., Renambot L., Jeong B., Verlo A., Long L., Brown M., Sandin D. J., Vishwanath V., Liu Q., Katz M. J., Papadopoulos P., Keefe J. P., Hidley G. R., Dawe G. L., Kaufman I., Glogowski B., Doerr K.-U., Singh R., Girado J., Schulze J. P., Kuester F., Smarr L.: The optiportal, a scalable visualization, storage, and computing interface device for the optiputer. *Future Gener. Comput. Syst. 25* (February 2009), 114–123.

[Dod04] Dodgson N. A.: Variation and extrema of human interpupillary distance in stereoscopic displays and virtual reality systems. In *Proc. SPIE 5291* (2004), pp. 36–46.

[Dol10] Dold C.: *Retrospective and prospective motion correction for magnetic resonance imaging of the head*. PhD thesis, Graz University of Technology, 2010.

[Dow87] Dowling J.: *The Retina*. Belknap Press, 1987.

[DQ] Degreen, Quilbert:.

[DU02] Dorfmüller-Ulhaas K.: *Optical Tracking - From User Motion To 3D Interaction*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, 2002.

[Dun04] Dunn A.:, 2004.

[DVC07] Damera-Venkata N., Chang N.: Realizing super-resolution with superimposed projection. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on* (june 2007), pp. 1 –8.

[EESM10] Eisemann M., Eisemann E., Seidel H.-P., Magnor M.: Photo zoom: high resolution from unordered image collections. In *Proceedings of Graphics Interface 2010* (Toronto, Ont., Canada, Canada, 2010), GI '10, Canadian Information Processing Society, pp. 71–78.

[EFS09] Edelmann J., Fleck S., Schilling A.: The dabr – a multitouch system for intuitive 3d scene navigation. In *3DTV CON - The True Vision* (2009).

[Ell94] Ellis S. R.: What are virtual environments? *IEEE Comput. Graph. Appl. 14* (January 1994), 17–22.

[EPO] Epoch, european network of excellence in open cultural heritage.

[FDGB08] Férey N., Delalande O., Grasseau G., Baaden M.: A vr framework for interacting with molecular simulations. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2008), VRST '08, ACM, pp. 91–94.

[FJP02] Freeman W. T., Jones T. R., Pasztor E. C.: Example-based super-resolution. *IEEE Comput. Graph. Appl. 22* (March 2002), 56–65.

[FKC*05] Fels S., Kinoshita Y., Chen T.-P. G., Takama Y., Yohanan S., Gadd S., Takahashi A., Funahashi K.: Swimming across the pacific: A vr swimming interface. *IEEE Comput. Graph. Appl. 25* (January 2005), 24–31.

[FLL*93] Fairchild K., Lee B., Loo J., Ng H., Serra L.: The heaven and earth virtual reality: Designing applications for novice users. In *Virtual Reality Annual International Symposium, 1993., 1993*

IEEE (sep 1993), pp. 47 –53.

[FMM*08] Fitzmaurice G., Matejka J., Mordatch I., Khan A., Kurtenbach G.: Safe 3d navigation. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2008), I3D '08, ACM, pp. 7–15.

[FRE03] Fernandes K. J., Raja V., Eyre J.: Cybersphere: the fully immersive spherical projection system. *Commun. ACM 46* (September 2003), 141–146.

[Fri99] Frisch K. J.: The fully immersive cave. In *in H.-J. Bullinger & O. Riedel, eds, '3. International Immersive Projection Technology Workshop, 10./11. May 1999, Center of the Fraunhofer Society Stuttgart IZS* (1999), Springer, pp. 59–63.

[FRK11] Freitag S., Rausch D., Kuhlen T.: Visualizing acoustical simulation data in immersive virtual environments. In *Workshop "Virtuelle & Erweiterte Realität" (VR/AR)* (2011).

[FSG98] Fuhrmann A., Schmalstieg D., Gervautz M.: Strolling through cyberspace with your hands in your pockets: head directed navigation in virtual environments. In *Proceedings of the 4th EUROGRAPHICS Workshop on Virtual Environments* (1998), pp. 216–227.

[FTB*00] Fröhlich B., Tramberend H., Beers A., Agrawala M., Baraff D.: Physically-based manipulation on the responsive workbench. In *In IEEE VR 2000* (2000), pp. 5–11.

[GB08] Grundhöfer A., Bimber O.: Real-time adaptive radiometric compensation. *IEEE Transactions on Visualization and Computer Graphics 14* (January 2008), 97–108.

[GF01] Greenberg S., Fitchett C.: Phidgets: Easy development of physical interfaces through physical widgets. *Proceedings of the UIST 2001 14th Annual ACM Symposium on User Interface Software and Technology* (2001), 209–218.

[Gre] Grey P.: Spectral-firefly.

[GSW01] Guimbretière F., Stone M., Winograd T.: Fluid interaction with high-resolution wall-size displays. In *Proceedings of the 14th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2001), UIST '01, ACM, pp. 21–30.

[Han05] Han J. Y.: Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2005), ACM, pp. 115–118.

[Han06] Han J.: Jeff han demos his breakthrough touchscreen. Talk, Technology, Entertainment, Design (TED) 2006, Aug. 2006.

[Hav05] Havemann S.: *Generative Mesh Modeling*. PhD thesis, Institute of Computer Graphics, Braunschweig Technical University, 2005.

[HCC07] Hancock M., Carpendale S., Cockburn A.: Shallow-depth 3d interaction: design and evaluation of one-, two- and three-touch techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2007), CHI '07, ACM, pp. 1147–1156.

[HCS*06] Harville M., Culbertson B., Sobel I., Gelb D., Fitzhugh A., Tanguay D.: Practical methods for geometric and photometric correction of tiled projector. In *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop* (Washington, DC, USA, 2006), CVPRW '06, IEEE Computer Society, pp. 5–.

[HEB*01] Humphreys G., Eldridge M., Buck I., Stoll G., Everett M., Hanrahan P.: Wiregl: a scalable graphics system for clusters. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 129–140.

[Hei60] Heilig M. L.: Stereoscopic-television apparatus for individual use, 1960. U.S. Patent No. 2,955,156.

[Hei62] Heilig M. L.: Sensorama simulator, 1962. U.S. Patent No. 3,050,870.

[HGAB08] Hoffman D. M., Girshick A. R., Akeley K., Banks M. S.: Vergence-accommodation conflicts hinder visual performance and cause visual fatigue. *Journal of Vision 8*, 3 (2008), 33.

[HHN*02] Humphreys G., Houston M., Ng R., Frank R., Ahern S., Kirchner P. D., Klosowski J. T.: Chromium: a stream-processing framework for interactive rendering on clusters. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 693–702.

[HJA04] Hogue A., Jenkin M., Allison R.: An optical-inertial tracking system for fully-enclosed vr displays. In *Proceedings of the 1st Canadian Conference on Computer and Robot Vision* (Washington, DC, USA, 2004), CRV '04, IEEE Computer Society, pp. 22–29.

[HKMA07] Höllerer T., Kuchera-Morin J., Amatriain X.: The allosphere: a large-scale immersive surround-view instrument. In *Proceedings of the 2007 workshop on Emerging displays technologies: images and beyond: the future of displays and interacton* (New York, NY, USA, 2007), EDT '07, ACM.

[HNH08] Hay S., Newman J., Harle R.: Optical tracking using commodity hardware. In *Mixed and Augmented Reality, 2008. ISMAR 2008. 7th IEEE/ACM International Symposium on* (September 2008), pp. 159 –160.

[HS95] Hartley R. I., Sturm P. F.: Triangulation. In *Proceedings of the 6th International Conference on Computer Analysis of Images and Patterns* (London, UK, 1995), CAIP '95, Springer-Verlag, pp. 190–197.

[HS08] Hoskinson R., Stoeber B.: High-dynamic range image projection using an auxiliary mems mirror array. *Optics Express 16*, 10 (May 2008), 7361–8.

[HSHF10] Hoskinson R., Stoeber B., Heidrich W., Fels S.: Light reallocation for high contrast projection using an analog micromirror array. *ACM Trans. Graph. 29* (December 2010), 165:1–165:10.

[HtCC09] Hancock M., ten Cate T., Carpendale S.: Sticky tools: full 6dof force-based interaction for multi-touch tables. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces* (New York, NY, USA, 2009), ITS '09, ACM, pp. 133–140.

[HtCCI10] Hancock M., ten Cate T., Carpendale S., Isenberg T.: Supporting sandtray therapy on an interactive tabletop. In *Proceedings of the 28th international conference on Human factors in computing systems* (New York, NY, USA, 2010), CHI '10, ACM, pp. 2133–2142.

[HTNS09] Hoshi T., Takahashi M., Nakatsuma K., Shinoda H.: Touchable holography. In *ACM SIGGRAPH 2009 Emerging Technologies* (New York, NY, USA, 2009), SIGGRAPH '09, ACM, pp. 23:1–23:1.

[HWJ07] Habel R., Wimmer M., Jeschke S.: Instant animated grass. *Journal of WSCG 15*, 1-3 (Jan. 2007), 123–128. ISBN 978-80-86943-00-8.

[IdFC07] Ierusalimschy R., de Figueiredo L. H., Celes W.: The evolution of lua. In *Proceedings of the third ACM SIGPLAN conference on History of programming languages* (New York, NY, USA, 2007), HOPL III, ACM, pp. 2–1–2–26.

[INK*11] Izadi S., Newcombe R., Kim D., Hilliges O., Molyneaux D., Kohli P., Shotton J., Hodges S., Davison A., Fitzgibbon A.: Kinectfusion: Real-time dynamic 3d surface reconstruction and interaction, 2011.

[IRA07] Interrante V., Ries B., Anderson L.: Seven league boots: An new metaphor for augmented locomotion through large scale immersive virtual environments. In *In Proceedings of IEEE Symposium on 3D User Interfaces (3DUI* (2007), IEEE Computer Society.

[JJK*04] Jeong D. H., Jeon Y. H., Kim J. K., Sim S., Song C. G.: Force-based velocity control technique in immersive v.e. In *Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia* (New York, NY, USA, 2004), GRAPHITE '04, ACM, pp. 237–241.

[JKB*08] Jung Y., Keil J., Behr J., Webel S., Zöllner M., Engelke T., Wuest H., Becker M.: Adapting x3d for multi-touch environments. In *Web3D '08: Proceedings of the 13th international symposium on 3D web technology* (New York, NY, USA, 2008), ACM, pp. 27–30.

[JSCH09] Jeong D. H., Song C. G., Chang R., Hodges L.: User experimentation: an evaluation of velocity control techniques in immersive virtual environments. *Virtual Real. 13* (February 2009), 41–50.

[Kab76] Kabsch W.: A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A 32*, 5 (September 1976), 922–923.

[Kab78] Kabsch W.: A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A 34*, 5 (1978), 827–828.

[KB99] Kato H., Billinghurst M.: Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd IEEE and ACM International*

Workshop on Augmented Reality (Washington, DC, USA, 1999), IEEE Computer Society, pp. 85–.

[KB07] Kaltenbrunner M., Bencina R.: reactivision: a computer-vision framework for table-based tangible interaction. In *Proceedings of the 1st international conference on Tangible and embedded interaction* (New York, NY, USA, 2007), TEI '07, ACM, pp. 69–74.

[KBBC05] Kaltenbrunner M., Bovermann T., Bencina R., Costanza E.: Tuio - a protocol for table based tangible user interfaces. In *Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005)* (Vannes, France, 2005).

[KBG09] Kuhlen T., Beer T., Gerndt A.: The vista virtual reality toolkit. In *5th High-End Visualization Workshop, Baton Rouge, Lousianna* (2009).

[KBK08] Kreylos O., Bawden G. W., Kellogg L. H.: Immersive visualization and analysis of lidar data. In *Proceedings of the 4th International Symposium on Advances in Visual Computing* (Berlin, Heidelberg, 2008), ISVC '08, Springer-Verlag, pp. 846–855.

[KOM*03] Kaiser E., Olwal A., McGee D., Benko H., Corradini A., Li X., Cohen P., Feiner S.: Mutual disambiguation of 3d multimodal interaction in augmented and virtual reality., 2003.

[KRK03] Kresse W., Reiners D., Knöpfle C.: Color consistency for digital multi-projector stereo display systems: the heyewall and the digital cave. In *Proceedings of the workshop on Virtual environments 2003* (New York, NY, USA, 2003), EGVE '03, ACM, pp. 271–279.

[KSRW06] Kirk D., Sellen A., Rother C., Wood K.: Understanding photowork. In *Proceedings of the SIGCHI conference on Human Factors in computing systems* (New York, NY, USA, 2006), CHI '06, ACM, pp. 761–770.

[KUWS03] Kang S. B., Uyttendaele M., Winder S., Szeliski R.: High dynamic range video. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 319–325.

[LB04] Letessier J., Bérard F.: Visual tracking of bare fingers for interactive surfaces. In *Proceedings of the 17th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2004), UIST '04, ACM, pp. 119–122.

[LC87] Lorensen W. E., Cline H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), SIGGRAPH '87, ACM, pp. 163–169.

[LFKZ01] LaViola Jr. J. J., Feliz D. A., Keefe D. F., Zeleznik R. C.: Hands-free multi-scale navigation in virtual environments. In *Proceedings of the 2001 symposium on Interactive 3D graphics* (New York, NY, USA, 2001), I3D '01, ACM, pp. 9–15.

[LHH*09] Love G. D., Hoffman D. M., Hands P. J., Gao J., Kirby A. K., Banks M. S.: High-speed switchable lens enables the development of a volumetric stereoscopic display. *Opt. Express 17*, 18 (Aug 2009), 15716–15725.

[LHKR10] Lanman D., Hirsch M., Kim Y., Raskar R.: Content-adaptive parallax barriers: optimizing dual-layer 3d displays using low-rank light field factorization. *ACM Trans. Graph. 29*, 6 (2010), 163:1–163:10.

[LIH07] Lambooij M. T. M., IJsselsteijn W. A., Heynderickx I.: Visual discomfort in stereoscopic displays: a review. Woods A. J., Dodgson N. A., Merritt J. O., Bolas M. T., McDowall I. E., (Eds.), vol. 6490, SPIE, p. 64900I.

[Lip08] Lippmann M. G.: Épreuves réversibles. photographies intégrales, 1908.

[LK03] Loomis J. M., Knapp J. M.: Visual perception of egocentric distance in real and virtual environments. 21–46.

[LKG*03] Llamas I., Kim B., Gargus J., Rossignac J., Shaw C. D.: Twister: a space-warp operator for the two-handed editing of 3d shapes. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 663–668.

[LSFP07] Leeb R., Settgast V., Fellner D., Pfurtscheller G.: Self-paced exploration of the austrian national library through thought. *International Journal of Bioelectromagnetism 9*, 4 (2007), 237–244.

[Mat05] Mathieu H.: The cyclope: A 6 dof optical tracker based on a single camera. In *2nd INTUITION International Workshop, Paris, France, 24-25 Nov 2005* (nov 2005).

[McC45] McCollum H. J. d. N.: Stereoscopic television apparatus, 1945. U.S. Patent No. 2,388,170.

[McD10] McDonald J.: Nvidia 3d vision automatic: Best practices guide. NVIDIA whitepaper, 2010.

[MEK06] Mehling M., Eberhardt B., Kaufmann H.: *Implementation of a Low Cost Marker Based Infrared Optical Tracking System*. Master's thesis, Hochschule der Medien, Fachhochschule Stuttgart, 2006.

[Min95] Mine M. R.: *Virtual Environment Interaction Techniques*. Tech. rep., Chapel Hill, NC, USA, 1995.

[MOBH11] McGuire M., Osman B., Bukowski M., Hennessy P.: The alchemy screen-space ambient obscurance algorithm. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics* (New York, NY, USA, 2011), HPG '11, ACM, pp. 25–32.

[MS02] Majumder A., Stevens R.: Lam: luminance attenuation map for photometric uniformity in projection based displays. In *Proceedings of the ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2002), VRST '02, ACM, pp. 147–154.

[MS04] Majumder A., Stevens R.: Color nonuniformity in projection-based displays: analysis and solutions. *Visualization and Computer Graphics, IEEE Transactions on 10*, 2 (march-april 2004), 177 –188.

[MS05a] Majumder A., Stevens R.: Perceptual photometric seamlessness in projection-based tiled displays. *ACM Trans. Graph. 24* (January 2005), 118–139.

[MS05b] Majumder A., Stevens R.: Perceptual photometric seamlessness in projection-based tiled displays. *ACM Transactions on Graphics* (2005), 118–139.

[Mul95] Mulder A.: The i-cube system: Moving toward sensor technology for artists. *Proceedings of the 6th International Symposium on Electronic Art, Montreal, QC, Canada* (1995).

[Mul98] Multigen: Smartscene, 1998. Video Clip, Discovery Channel's NextStep program.

[MVL00] Mulder J. D., Van Liere R.: Enhancing fish tank vr. In *Proceedings of the IEEE Virtual Reality 2000 Conference* (Washington, DC, USA, 2000), VR '00, IEEE Computer Society, pp. 91–.

[ND10] Newcombe R. A., Davison Andrew J.: Live Dense Reconstruction with a Single Moving Camera. *CVPR* (2010).

[Nee62] Needham J.: *Science & Civilisation in China*, vol. IV:1. Cambridge University Press, 1962.

[NNHY06] Nakaizumi F., Noma H., Hosaka K., Yanagida Y.: Spotscents: A novel method of natural scent delivery using multiple scent projectors. In *Proceedings of the IEEE conference on Virtual Reality* (Washington, DC, USA, 2006), VR '06, IEEE Computer Society, pp. 207–214.

[NSS*06] Ni T., Schmidt G. S., Staadt O. G., Livingston M. A., Ball R., May R.: A survey of large high-resolution display technologies, techniques, and applications. In *Proceedings of the IEEE conference on Virtual Reality* (Washington, DC, USA, 2006), VR '06, IEEE Computer Society, pp. 223–236.

[NVH*04] N S., V H. P., H H., R K., A Y., M K., K S.: Cyberdome: Pc clustered hemi spherical immersive projection display. *6th International Conference on LAVAL Virtual Reality* (2004).

[NVI11] NVIDIA: Nvidia 3d vision automatic: Stereo unprojection. NVIDIA whitepaper, 2011.

[Opt] Optics E.: Electronic imaging resource guide.

[Pap98] Pape D.: Crayoland. In *ACM SIGGRAPH 98 Electronic art and animation catalog* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 116–.

[PBD*10] Parker S. G., Bigler J., Dietrich A., Friedrich H., Hoberock J., Luebke D., McAllister D., McGuire M., Morley K., Robison A., Stich M.: Optix: a general purpose ray tracing engine. In *ACM SIGGRAPH 2010 papers* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 66:1–66:13.

[PBWI96] Poupyrev I., Billinghurst M., Weghorst S., Ichikawa T.: The go-go interaction technique: non-linear mapping for direct manipulation in vr. In *Proceedings of the 9th annual ACM symposium on User interface software and technology* (New York, NY, USA, 1996), UIST '96, ACM, pp. 79–80.

[PFC*97] Pierce J. S., Forsberg A. S., Conway M. J., Hong S., Zeleznik R. C., Mine M. R.: Image plane interaction techniques in 3d immersive environments. In *Proceedings of the 1997 symposium on Interactive 3D graphics* (New York, NY, USA, 1997), I3D '97, ACM, pp. 39–ff.

[Pho] Autodesk project photofly.

[PK07] Pintaric T., Kaufmann H.: Affordable infrared-optical pose-tracking for virtual and augmented reality. In *Proceedings of T̈rends and Issues in Tracking for Virtual EnvironmentsẄorkshop, IEEE VR 2007, Charlotte, NC, USA, March 2007* (2007).

[PK08] Pintaric T., Kaufmann H.: A rigid-body target design methodology for optical pose-tracking systems. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2008), VRST '08, ACM, pp. 73–76.

[PKG*07] Peterka T., Kooima R. L., Girado J. I., Ge J., Sandin D. J., DeFanti T. A.: Evolution of the varrier autostereoscopic vr display: 2001-2007. In *Stereoscopic Displays and Virtual Reality Systems XIV, SPIE, San Jose, California* (2007).

[Pow94] The powerwall, 1994. The University of Minnesota, Silicon Graphics, Inc., Ciprico, Inc. and IBM Storage Products Division.

[PPF*] Pobitzer A., Peikert R., Fuchs R., Schindler B., Kuhn A., Theisel H., Matkovic K., Hauser H.: On the Way Towards Topology-Based Visualization of Unsteady Flow. pp. 137–154.

[Pro] Prosilica: Spectral-ec750.

[PS05] Pavlovych A., Stuerzlinger W.: A high-dynamic range projection system. In *Photonic Applications in Biosensing and Imaging, Proceedings of SPIE 5969, 2005. SPIE-The International Society for Optical Engineering* (2005).

[PV11] Pavlik R., Vance J.: Vr jugglua: A framework for vr applications combining lua, openscenegraph, and vr juggler. In *Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)* (2011), IEEE Virtual Reality.

[PWS09] Pirchheim C., Waldner M., Schmalstieg D.: Deskotheque: Improved spatial awareness in multi-display environments. In *VR* (2009), IEEE, pp. 123–126.

[Ras00] Raskar R.: Immersive planar display using roughly aligned projectors. In *Proceedings of the IEEE Virtual Reality 2000 Conference* (Washington, DC, USA, 2000), VR '00, IEEE Computer Society, pp. 109–.

[RBJW01] Ringel M., Berg H., Jin Y., Winograd T.: Barehands: implement-free interaction with a wall-mounted display. In *CHI '01 extended abstracts on Human factors in computing systems* (New York, NY, USA, 2001), CHI EA '01, ACM, pp. 367–368.

[RBY*99] Raskar R., Brown M. S., Yang R., Chen W.-C., Welch G., Towles H., Seales B., Fuchs H.: Multi-projector displays using camera-based registration. In *Proceedings of the conference on Visualization '99: celebrating ten years* (Los Alamitos, CA, USA, 1999), VIS '99, IEEE Computer Society Press, pp. 161–168.

[RCM93] Robertson G. G., Card S. K., Mackinlay J. D.: Non-immersive virtual reality. *IEEE Computer 26* (1993), 81–83.

[RDH09] Reisman J. L., Davidson P. L., Han J. Y.: A screen-space formulation for 2d and 3d direct manipulation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology* (New York, NY, USA, 2009), UIST '09, ACM, pp. 69–78.

[RKW01] Razzaque S., Kohn Z., Whitton M. C.: Redirected walking, 2001.

[RMWW94] Rushton S., Mon-Williams M., Wann J. P.: Binocular vision in a bi-ocular world: new-generation head-mounted displays avoid causing visual deficit. *Displays 15*, 4 (1994), 255 – 260.

[RS01] Reitmayr G., Schmalstieg D.: An open software architecture for virtual reality interaction. In *Proceedings of the ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2001), VRST '01, ACM, pp. 47–54.

[RS05] Reitmayr G., Schmalstieg D.: Opentracker: A flexible software design for three-dimensional interaction. *Virtual Real. 9* (December 2005), 79–92.

[RSS*02] Razzaque S., Swapp D., Slater M., Whitton M. C., Steed A.: Redirected walking in place. In *Proceedings of the workshop on Virtual environments 2002* (Aire-la-Ville, Switzerland, Switzerland, 2002), EGVE '02, Eurographics Association, pp. 123–130.

[RVSS10] Ricolfe-Viala C., Sánchez-Salmerón A.-J.: Correcting non-linear lens distortion in cameras without using a model. *Optics and Laser Technology 42* (2010), 628–639.

[RWF98] Raskar R., Welch G., Fuchs H.: Seamless projection overlaps using image warping and intensity blending. In *in Fourth International Conference on Virtual Systems and Multimedia*

(1998).

[SBJ*10] Steinicke F., Bruder G., Jerald J., Frenz H., Lappe M.: Estimation of detection thresholds for redirected walking techniques. *IEEE Trans. Vis. Comput. Graph. 16*, 1 (2010), 17–27.

[SBK06] Schirski M., Bischof C., Kuhlen T.: Interactive particle tracing on tetrahedral grids using the gpu. In *Proceedings of Vision, Modeling, and Visualization (VMV)* (2006).

[Sch08] Schöning J.: Multi-touch surfaces: A technical guide. *IEEE Tabletops and Interactive Surfaces* (Oct. 2008).

[SCK07] Shum H.-Y., Chan S.-C., Kang S. B.: *Image-Based Rendering*. Springer, 2007.

[SCP95] Stoakley R., Conway M. J., Pausch R.: Virtual reality on a wim: interactive worlds in miniature. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1995), CHI '95, ACM Press/Addison-Wesley Publishing Co., pp. 265–272.

[SCS01] Sukthankar R., Cham T.-J., Sukthankar G.: Dynamic shadow elimination for multi-projector displays. In *Proceedings of IEEE Computer Conference on Computer Vision and Pattern Recognition* (2001), vol. 2, pp. 151 – 157.

[See09] Seetzen H.: *High dynamic range display and projection systems*. PhD thesis, University of British Columbia, 2009.

[SGvR*03] Schirski M., Gerndt A., van Reimersdahl T., Kuhlen T., Adomeit P., Lang O., Pischinger S., Bischof C.: Vista flowlib - framework for interactive visualization and exploration of unsteady flows in virtual environments. In *Proceedings of the workshop on Virtual environments 2003* (New York, NY, USA, 2003), EGVE '03, ACM, pp. 77–85.

[Sha68] Shaw B.: *The Light of Other Days*. Analog, 1968.

[SHS*04] Seetzen H., Heidrich W., Stuerzlinger W., Ward G., Whitehead L., Trentacoste M., Ghosh A., Vorozcovs A.: High dynamic range display systems. *ACM Transactions on Graphics, Proc. SIGGRAPH 2004 23*, 3 (2004), 760–768.

[SKO09] Schöning J., Krüger A., Olivier P. (Eds.):. *Multi-touch is Dead, Long live Multi-Touch. ACM International Conference on Human Factors in Computing Systems (CHI-2009), Workshop on Multi-touch and Surface Computing, April 4 - March 9, Boston,, MA, United States* (2009), ACM Press, New York, NY, USA.

[SL98] Shaw J., Lantz E.: Dome theaters: Spheres of influence. TiLE Proceedings, 1998.

[SMD*01] Sandin D., Margolis T., Dawe G., Leigh J., DeFanti T.: The varrier auto-stereographic display. In *SPIE, San Jose, California* (2001), vol. 4297.

[SMP05] Svoboda T., Martinec D., Pajdla T.: A convenient multicamera self-calibration for virtual environments. *Presence: Teleoper. Virtual Environ. 14* (August 2005), 407–422.

[SPDAM*02] Serolli Pinho M., Dias L. L., Antunes Moreira C. G., González Khodjaoghlanian E., Pizzini Becker G., Duarte L. M.: A user interface model for navigation in virtual environments. *Cyberpsychology and Behavior 5*, 5 (2002), 443–449.

[Spea] Spectral-fluorescent. de.academic.ru.

[Speb] Spectral-halogen. www.tauchfunzel.de.

[Spec] Spectral-ir-led. Osram.

[Sped] Spectral-uhp-projector. AVForums.

[SS95] Schröder P., Sweldens W.: Spherical wavelets: efficiently representing functions on the sphere. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1995), SIGGRAPH '95, ACM, pp. 161–172.

[SS97] Salisbury J., Srinivasan M.: Phantom-based haptic interaction with virtual objects. *Computer Graphics and Applications, IEEE 17*, 5 (sept.-oct. 1997), 6 – 10.

[SSKS03] Silverman N. L., Schowengerdt B. T., Kelly J. P., Seibel E. J.: Engineering a retinal scanning laser display with integrated accommodation depth cues. vol. 34, SID, pp. 1538–1541.

[SSSF04] Schowengerdt B. T., Seibel E. J., Silverman N. L., Furness T. A.: Stereoscopic retinal scanning laser display with integrated focus cues for ocular accommodation. Woods A. J., Merritt J. O., Benton S. A., Bolas M. T., (Eds.), vol. 5291, SPIE, pp. 366–376.

[SUS95] Slater M., Usoh M., Steed A.: Taking steps: the influence of a walking technique on presence in virtual reality. *ACM Trans. Comput.-Hum. Interact. 2* (September 1995), 201–219.

[Sut65] SUTHERLAND I. E.: The ultimate display. In *Proceedings of the IFIP Congress* (1965), pp. 506–508.

[Sut68] SUTHERLAND I. E.: A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I* (New York, NY, USA, 1968), AFIPS '68 (Fall, part I), ACM, pp. 757–764.

[SW11] SCHEIBLAUER C., WIMMER M.: Out-of-core selection and editing of huge point clouds. *Computers & Graphics 35*, 2 (Apr. 2011), 342–351.

[SWW03] SEETZEN H., WHITEHEAD L. A., WARD G.: A high dynamic range display using low and high resolution modulators, 2003. Society for Information Display (SID) Digest.

[Sze06] SZELISKI R.: Image alignment and stitching: a tutorial. *Found. Trends. Comput. Graph. Vis. 2* (January 2006), 1–104.

[THS*01] TAYLOR II R. M., HUDSON T. C., SEEGER A., WEBER H., JULIANO J., HELSER A. T.: Vrpn: a device-independent, network-transparent vr peripheral system. In *Proceedings of the ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2001), VRST '01, ACM, pp. 55–61.

[TMHF00] TRIGGS B., MCLAUCHLAN P. F., HARTLEY R. I., FITZGIBBON A. W.: Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice* (London, UK, 2000), ICCV '99, Springer-Verlag, pp. 298–372.

[Tsa92] TSAI R. Y.: Radiometry. Jones and Bartlett Publishers, Inc., USA, 1992, ch. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, pp. 221–244.

[Ume91] UMEYAMA S.: Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell. 13* (April 1991), 376–380.

[USOF09] ULLRICH T., SETTGAST V., OFENBÖCK C., FELLNER D. W.: Desktop integration in graphics environments. In *Virtual Environments 2009; Joint Virtual Reality Conference of EGVE - ICAT - EuroVR* (2009), Hirose M., Schmalstieg D., Wingrave C. A., Hishimura K., (Eds.), Eurographics Association, pp. 109–112. doi: 10.2312/EGVE/JVRC09/109-112.

[Val07] VALLI A.: Natural interaction white paper, 2007.

[Vic72] VICKERS D. L.: *Sorcerer's apprentice: head-mounted display and wand*. PhD thesis, 1972.

[WB95] WELCH G., BISHOP G.: *An Introduction to the Kalman Filter*. Tech. rep., Chapel Hill, NC, USA, 1995.

[WBV*99] WELCH G., BISHOP G., VICCI L., BRUMBACK S., KELLER K., COLUCCI D.: The hiball tracker: high-performance wide-area tracking for virtual and augmented environments. In *Proceedings of the ACM symposium on Virtual reality software and technology* (New York, NY, USA, 1999), VRST '99, ACM, pp. 1–ff.

[WCL03] WALLACE G., CHEN H., LI K.: Color gamut matching for tiled display walls. In *Proceedings of the workshop on Virtual environments 2003* (New York, NY, USA, 2003), EGVE '03, ACM, pp. 293–302.

[WIH*08] WILSON A. D., IZADI S., HILLIGES O., GARCIA-MENDOZA A., KIRK D.: Bringing physics to the surface. In *Proceedings of the 21st annual ACM symposium on User interface software and technology* (New York, NY, USA, 2008), UIST '08, ACM, pp. 67–76.

[Wis01] WISSEN M.: Implementation of laser-based interaction technique for projection screens. 31–32.

[WPH10] WEIBEL N., PIPER A. M., HOLLAN J. D.: Hiperpaper: introducing pen and paper interfaces for ultra-scale wall displays. In *Adjunct proceedings of the 23nd annual ACM symposium on User interface software and technology* (New York, NY, USA, 2010), UIST '10, ACM, pp. 407–408.

[WRM*08] WAGNER D., REITMAYR G., MULLONI A., DRUMMOND T., SCHMALSTIEG D.: Pose tracking from natural features on mobile phones. In *ISMAR* (2008), pp. 125–134.

[WS06] WIMMER M., SCHEIBLAUER C.: Instant points. In *Proceedings Symposium on Point-Based Graphics 2006* (July 2006), Eurographics, Eurographics Association, pp. 129–136.

[WWC*09] WIGDOR D., WILLIAMS S., CRONIN M., LEVY R., WHITE K., MAZEEV M., BENKO H.: Ripples: utilizing per-contact visualizations to improve user interaction with touch displays. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology* (New

York, NY, USA, 2009), UIST '09, ACM, pp. 3–12.

[YGH*01] Yang R., Gotz D., Hensley J., Towles H., Brown M. S.: Pixelflex: a reconfigurable multi-projector display system. In *Proceedings of the conference on Visualization '01* (Washington, DC, USA, 2001), VIS '01, IEEE Computer Society, pp. 167–174.

[YKNT03] Yanagida Y., Kawato S., Noma H., Tetsutani N.: A nose-tracked, personal olfactory display. In *ACM SIGGRAPH Sketches & Applications* (2003).

[Zha99] Zhang Z.: Flexible camera calibration by viewing a plane from unknown orientations. In *in ICCV* (1999), pp. 666–673.

[ZPB07] Zach C., Pock T., Bischof H.: A duality based approach for realtime tv-l1 optical flow. In *Proceedings of the 29th DAGM conference on Pattern recognition* (Berlin, Heidelberg, 2007), Springer-Verlag, pp. 214–223.

[ZWAY08] Zhou J., Wang L., Akbarzadeh A., Yang R.: Multi-projector display with continuous self-calibration. In *Proceedings of the 5th ACM/IEEE International Workshop on Projector camera systems* (New York, NY, USA, 2008), PROCAMS '08, ACM, pp. 3:1–3:7.

Index

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am ……………………………   ………………………………………………..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

……………………………   ………………………………………………..
date   (signature)