eman ta zabal zazu

Universidad del País Vasco | Euskal Herriko Unibertsitatea

Contributions to Virtual Reality

Memoria para optar al grado de Doctor en Informática que presenta

Álvaro Segura

Dirigida por los doctores:
Alejandro García-Alonso y Julián Flórez

Universidad del País Vasco
Euskal Herriko Unibertsitatea
Donostia - San Sebastian
2018

Firma del autor______________________________

Álvaro Segura Lasa

Certificado por : ______________________________

Prof.Dr Alejandro García-Alonso Montoya (Director de la tesis)

Certified by : ______________________________

Prof.Dr. Julián Flórez Esnal (Director de la tesis)

Acknowledgements

This work was possible thanks to the opportunity given by Vicomtech to work on research projects on the subject of interactive computer graphics were the contributions of this thesis were developed. I also acknowledge the collaboration of other partner institutions and companies involved in those projects.

I would like to acknowledge the support and collaboration by people that contributed to make this thesis possible. Many thanks to colleagues who collaborated in several of the works described in this thesis: Aitor Moreno, John Congote, Jon Goenetxea, Unai Elordi, Andoni Galdos, Javier Barandiaran, Iosu Arizkuren, Iñigo Lazkanotegi.

I am grateful to my supervisors, Julián Flórez and Alex García-Alonso, for their support and encouragement, and their work reviewing texts and suggesting changes. I would also like to thank Manuel Cendoya, Amaia Bernaras and Tim Smithers who introduced me to the world of applied research.

Abstract

This thesis presents contributions to Virtual Reality organized in three topics: visual perception, immersive scenarios and ubiquitous visualization.

Visual perception in virtual reality devices differs from perception in the real world in several aspects. Stereoscopic displays can simulate the disparity of images perceived by each eye in the real world, to provide depth perception. However, this solution is limited. Correctly simulating the view from each eye of a virtual environment, so that eye convergence to virtual objects is similar to converge to equivalent real objects, requires precise knowledge of the viewing conditions of the display. Furthermore, stereo displays do not simulate the light field behavior that triggers accommodation (i.e. focus) of the eye, whereas in reality accommodation varies as we look at objects at different distances. Conventional head-mounted displays feature a pair of displays at fixed positions and thus require a fixed eye accommodation for them to be seen sharply. This thesis proposes ways to tackle these problems using variable-focus HMDs and special calibration methods. A calibration algorithm is proposed to estimate stereo projection parameters for each specific device, and calibration and control procedures are proposed to obtain expected accommodation stimuli at different virtual distances.

This thesis also proposes solutions to practical problems in **immersive scenarios**. The thesis addresses problems in diverse scenarios: construction machinery simulators, weather radar volumetric visualization and manual arc welding simulation. They demand varying degrees of immersion and special, innovative visualization solutions are proposed to fulfil their requirements.

Finally, contributions are presented for **ubiquitous visualization** scenarios where users access interactive 3D applications remotely. Here, running on networked devices using conventional web browsers is the main requirement, to the detriment of realism and immersion. The thesis follows the evolution of Web3D standards and technologies to propose original visualization solutions for volume rendering of weather radar data, e-learning on energy efficiency, virtual e-commerce and visual product configurators.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The fields of Computer Graphics and Computer Vision were categorized as image synthesis and image analysis [70]. Both fields remained separated for many years. Currently, they tend to support each other as it happens in the Augmented Reality field. Modern categorizations of these fields can be found in the literature (See James D. Foley and Hughes [44], Akenine-Möller and Haines [1]).

This thesis compiles research work from 2005 to the present. The research merges contributions from both image synthesis and analysis. However, most of the research can be classified within the classical image synthesis field. This research work will be classified in three areas:

- Stereoscopic viewing in head-mounted displays and their calibration for correct depth perception, including accommodation.
- Immersive visualization: alternative setups to provide a higher feeling of being immersed in a virtual world.
- Ubiquitous visualization: algorithms and applications of ubiquitous interactive 3D visualization where users access remote applications from anywhere with different computing devices.

The following sections describe the thesis goals and the thesis structure.

1.2 Objectives

This thesis pursues the following main goals:

- **To research in some problems found in Head-Mounted Displays**: automatic calibration and the vergence/accommodation mismatch problems. The thesis should find a calibration method to have non-subjective quantification of stereo perspective projection parameters. This requires to study how virtual 3D space is perceived in Virtual Reality devices, as opposed to how reality is seen, and what can be done to improve this perception. HMDs, by design, present a fixed focus stimulus. However, the stereo stimuli they generate make the eyes converge at different depth distances. This provokes the problem known as vergence-accommodation mismatch. An HMD with controllable variable focus should alleviate this problem. Methods to calibrate such controls are needed to provide correct focus stimuli that match the stereo depth stimuli. This problem affects, to different extents, virtually all kinds of stereo displays.

- **To find an adequate immersive solution for different real cases**: welding simulator, machinery simulator and volumetric radar data visualization. Vehicle simulators typically simulate a single vehicle type, and only high-end systems include mock-up cabs, seats and controls that feel like the real machines. A more versatile architecture should be designed to allow systems the simulation of multiple machines. How Augmented Reality can be leveraged to support this goal had to be investigated. Weather radars collect volumetric information about the atmosphere state around them. Visualization techniques that can represent this information in the most comprehensible ways had to be analyzed.

- **To provide contributions in ubiquitous visualization**. A known potential application of Virtual Reality is in transmitting learning concepts. Virtual 3D models should be integrated in systems aimed at learning. Research was needed to apply this concept for ubiquitous learning cases. On the other hand, Visual product configurators let users customize virtual products and see what they will look like when finally manufactured. Their requirement of visual realism is relatively high. Users require an ubiquitous visual product configurator that can work on the Web using advanced render techniques. Finally, the e-commerce sector has grown in presence and acceptance over the years. However, its user interfaces have seen a limited evolution, typically consisting of online catalogs displaying images and descriptions of products. There is a need to research Web3D technology and its limitations, to propose alternative user interfaces based on online Virtual Reality paradigms that provide ubiquitous solutions to make them easy to create and use.

1.3 Thesis structure

This thesis is structured in 6 chapters where this introduction is the first chapter.

Chapter 2 introduces a generic background and basic concepts that will be discussed and developed throughout the rest of the document.

The next three chapters contain the main areas of research. Each of them includes specific background and references to the state of the art in its respective area.

Chapter 3 deals with the visual perception of virtual space in HMDs. It explains the issue of *vergence-accommodation mismatch* and provides methods to calibrate the stereoscopic characteristics of HMDs and to control eye accommodation (eye focus) for best viewing.

Chapter 4 provides contributions in immersive virtual reality and Chapter 5 brings contributions in ubiquitous virtual visualization.

Finally, the main thesis conclusions are summarized, followed by possible lines of future work, in Chapter 6.

Chapter 2

Background

This chapter provides a generic background on the topics addressed in this thesis.

The three sections in this chapter introduce the core concepts most relevant to Chapters 3 to 5. Section 2.1 introduces the concept of *immersive VR visualization* and the technologies that enable it. Section 2.2 is devoted to general knowledge on the human visual system regarding binocular vision and its behaviour with immersive visualization technologies. Finally, Section 2.3 presents the point of view for *ubiquitous visualization* considered in this research. It describes standards and technologies that have been proposed over the years to support it. Using this background, Chapter 5 provides specific contributions to ubiquitous visualization.

2.1 Immersive visualization

Freina and Ott [28] review the state of the art of immersive virtual reality applied to education. In their paper they distinguish between *non- immersive* and *immersive* virtual reality. Following their definition, non-immersive VR is a simulated 3D environment that can be displayed on a standard computer screen, while immersive VR requires special devices to provide a view that surrounds the user, making him/her feel as if standing inside the virtual world. Their review focuses on two types of immersion devices: CAVE and head-mounted display (HMD), which are described in the following subsections. Then, a last subsection adds a description of the most relevant stereo display technologies, stereoscopy being a crucial part of any immersive visualization system.

In Leigh et al. [54] a related concept is introduced: *tele-immersion*. It is defined as “the integration of audio and video conferencing, via image-based modeling, with collaborative virtual reality, in the context of data-mining and significant computation”. Thus, technically, tele-immersion brings together immersive virtual reality in the form of a CAVE, with communication links that support remote multiuser work and tele-presence.

In this thesis the concept of immersive VR used does not necessarily require full 360-degree surrounding imagery. Besides HMDs, we also consider interactive systems with stereo displays of medium to large size with rendering configured to provide a perspective correct view.

2.1.1 Head-mounted displays

The first *head-mounted display*, abbreviated HMD, was created by Ivan Sutherland in 1968 [91]. He had the idea of a display that would present a perspective image that changes as he moves or looks around, just as happens when viewing the real world (see Figure 2.1). An HMD is a portable device attached to a user’s head that presents one or two small displays in front of his eyes. The HMD’s displays show a 3D view of a virtual space in front of the user.

The basic elements of an HMD are a pair of displays and a pair of optics to magnify and focus the displayed images. The displays are often small and placed at a very near distance to the eyes, so optics are used to magnify them and form images at a further virtual distance.

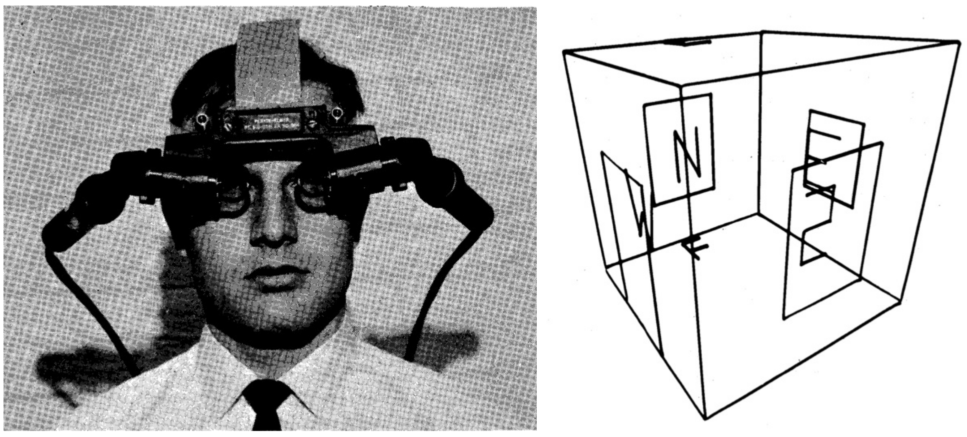


Figure 2.1: *Sutherland's original HMD (left) and the 3D view displayed in it (right), from Sutherland [91]*

Additionally, HMDs are usually coupled with tracking systems that determine the user's head orientation and/or its position and this information is used to render the 3D views from a correct point of view. Sutherland's original HMD tested two different tracking systems, one based on a mechanical arm with encoders connected to the HMD and one based on ultrasonic transmitters and receivers. Among the characteristics of an HMD, the most important visual parameters are:

Field of view (FOV) The apparent size –horizontal, vertical or diagonal– of the images presented, as seen by the user, usually expressed in degrees. A larger FOV enables a higher feeling of immersion.

Resolution The size in pixels of the images displayed to each eye.

Angular resolution The apparent size of each pixel as seen by the user, often expressed in arcminutes or as pixels per degree. This sets a limit to the smallest detail that can be represented and the image sharpness. Let us note that the human visual system's resolution is around 1 arcmin with a healthy vision.

Since the early 1990s these devices have evolved and spread. In recent years a lot of attention has been put on them focusing on the consumer market. After a Kickstarter crowdfunding campaign, in 2013 the first prototype of the *Oculus Rift* [74] was released, an innovative HMD with integrated 3-DoF head tracking (6-DoF in later versions). Unlike traditional HMDs based on separate small microdisplays, the Rift featured a single relatively large display and magnifying optics for each centred on the left and right halves of that screen. This design enabled it to have a much larger field of view, at the expense of a low angular resolution. Inspired on the Oculus Rift success, a few devices were presented by competing companies. HTC released the *HTC Vibe*, a device similar in design to the Oculus Rift with extended room tracking [39]. Similarly Sony released the *PlayStation VR* HMD in 2016. Advances in smartphones led to the appearance of very low cost accessories with which an HMD can be built. *Google Cardboard* [60] is a cardboard structure with a pair of lenses, and space for a smartphone. The displays, computation, 3D rendering and orientation tracking are all provided by the inserted smartphone, making it a low cost all- in-one head mounted display.

2.1.2 CAVE Automatic Virtual Environment

In 1992, Cruz-Neira et al. [17] presented the CAVE, a cubic immersive room in which users are surrounded by projected images of a virtual environment. A CAVE typically has semi-transparent walls (and possibly floor and ceiling) and uses rear projection to present computer-generated images on them. The user's head position and orientation is tracked by some device in order to render the 3D views for each of the walls from the user's point of view so that he gets a perspective-correct immersive view. Additionally, projection is usually stereoscopic to provide depth-perception.

Figure 2.2 depicts a very simplified view of a CAVE with three active walls. Sometimes, in the place of each shown projector there is a pair of projectors to provide dual-projection for passive stereo visualization.

In practice, CAVEs often place mirrors between projectors and screens to reduce the overall size. This is particularly useful in the projections for the floor and ceiling. With a mirror, the projection axis is bent 90° so the projector does not need to be as far from the screen.

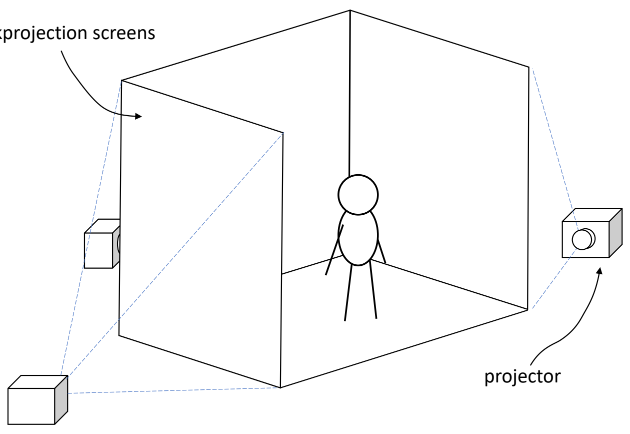


Figure 2.2: *Simplified view of a CAVE system. Actual implementations often use mirrors between projectors and screens to reduce the overall size*

The original CAVE, further described by Cruz-Neira et al. [16], had three projection walls plus the floor and used mirrors to reduce size. The floor was front-projected (thus suffering from shadows) also to limit total height. Active stereo with shutter glasses was used (see Section 2.1.3), so one projector per screen was enough. Stereo graphics rendering was performed with two off-axis projections per wall, using the estimated location of each eye inside the cube.

2.1.3 Stereoscopic displays

One of the key features of immersive visualization is stereo projection. The human visual system is binocular and its eyes capture slightly different views of the environment in front. This difference provides an important cue for depth perception and enables humans to perceive a world in three-dimensions. Since display screens are typically flat, they provide an image that is perceived as a flat surface. Stereoscopic or stereo displays implement different techniques to provide stereo vision, that is, to present slightly different images to each eye in order to provoke depth perception.

Some stereo displays use separate surfaces for the different images and require optical elements to have the eyes fuse those surfaces into a single image. Other displays present both images on the same surface and require some filter (e.g. in the form of glasses) to have each eye only see one of the images. A good review of the different techniques to provide stereoscopic projection is presented in Mendiburu [66] and more in-depth information regarding their application to stereoscopic 3D cinema can be found in Lipton [56]. Below is a list of the most relevant stereo display technologies:

- **The stereoscope**: As described by Brewster [8], this device was invented in the 19th century and is comparable in optical design to head-mounted displays but presenting paper photographs instead of video screens.
- **Anaglyphs**: Anaglyphs present stereo pairs using some colors for one eye and complementary colors for the other eye. The user wears special color filter glasses that let each eye only see one of the subimages. Actually the subimages presented have incomplete color information. For example, red-cyan anaglyphs render the image for the left eye using only its red color channel, and the image for the right eye using only the green and blue channels. A user wears glasses having a red filter on the left eye and a cyan filter on the right eye. This way the left eye sees the red channel of the left image and the right eye sees the green and blue channels of the right view. The brain finally mixes both views into a full color stereo view.

- **Shutter glasses**: Shutter glasses use time multiplexing to present stereo pairs on the same surface. The left and right views are shown alternatively switching at a high frequency and the user wears special glasses with synchronized shutters that cover the view of each eye when the view for the other eye is being presented. The switching frequency is high enough to prevent the perception of flicker.

- **Polarized light**: This technique uses light polarization to separate the left and right subimages. Often two projectors are used, each one projecting the image for one of the eyes, and each one having a complementary polarizers (linear or circular polarizers are used). Viewers have to wear relatively inexpensive passive polarizing filter glasses that allow each eye to only see one of the subimages. This projection technique requires a polarization-preserving screen so that polarized light projected into them maintains its polarization after being reflected towards viewers. A version of this technique requiring only one projector was also developed. It adds an active polarizer to the projector that switches polarization between succesive frames, which are projected alternating the left and right subimages.

- **Interference filters**: Interference filters [45] use a more advanced form of wavelength multiplexing than anaglyphs. In this case two sets of RGB primary colors are used with slightly offset wavelegths, one set for each eye. Sophisticated filter glasses are worn by viewers to separate the subimages. This technique, introduced by the Infitec company has been successful in 3D cinema under the brand *Dolby Digital 3D* [30].

- **Autostereoscopic screens**: The above technologies requires users to wear special filter glasses. Aiming at removing this requirement, some *autostereoscopic* monitors [21] were presented which could be viewed with the naked eye. The basic principle of these monitors is the placement of a special occlusion mask or lenticular layer over the screen that allows one eye to only see odd-numbered pixel columns and the other eye to only see even columns. Thus, the stereo pair has to be presented on the screen split in columns: the left view on even columns and the right view on odd columns. These displays are sensitive to observer position and movement from the designed viewing zones will produce incorrect filtering and image ghosting.

- **Head-mounted displays**: As explained in their own section, head-mounted displays present stereoscopic images by means of two small displays placed in front of the user's eyes and a pair of lens groups that provide focusing and magnification. The displays containing the lelft and right views appear superimposed in the observer's field of view.

The enumerated stereoscopy technologies have been used either in cinema to produce 3D films or in other immersive visualization applications, including industrial and leisure uses. In most applications, especially film, the intention is to provide an image with *depth*, that is, where some objects are perceived closer than others. In the cases where stereo images are shown on a screen, these objects can appear to be on the screen plane, closer, or further away. But in those applications no special interest is in making objects appear at specific distances or having specific apparent sizes. On the other hand, immersive virtual reality applications require virtual objects to appear at correct distances and preserve their sizes in order to provide good immersion.

2.2 Stereo vision and accommodation

Stereo display technologies like those described in the previous section are able to present different images to each of the viewer's eyes. How exactly the viewer perceives the displayed scene (in terms of depth and 3D shape) depend on how these images differ and how they are looked at. Moreover, when looking at a point in the real world two responses are triggered in the eye related to depth. On the one hand, the eyes converge to align with the target point in a movement called *vergence*. On the other hand, the eye's lens changes shape to focus a sharp image of the target object on the retina, a phenomenon called *accommodation*. An additional response, pupil dilation or contraction will not be taken into account in this work as it is not as relevant to depth perception. A thorough explanation of the structure of the human eye and its behaviour can be found in Gross et al. [36].

Despite their differences, an eye can be compared to a camera having as main elements a variable aperture (the iris), a focusing element (the lens) and an image projection surface (the sensor or the retina). A typical lens is characterized by its *focal length* (usually expressed in millimeters) or by its reciprocal, the *optical power* or *refractive power* (usually expressed in diopters, $1\,\mathrm{D} = 1\,\mathrm{m}^{-1}$). Using optical power has the benefit that the power of several consecutive lenses is approximately the sum of their individual powers.

Accommodation is an effect that occurs in each eye even if the other is covered. When looking at a distant object, the eye's lens is relaxed and its optical power plus the power of the cornea focus its incoming parallel rays into a point on the retina (see Figure 2.3). When looking at a near point, the eye's lens has to be deformed and thickened to focus rays into a single point. The accommodation optical power required for best focus at each moment is the accommodation demand. The eye's physiological response is the accommodation, the change in optical power from distant viewing to a shorter distance viewing. It may not be equal to the demand if this is beyond of the limit of a given eye. This demand can be expressed as an accommodation distance (the distance to the object looked at).

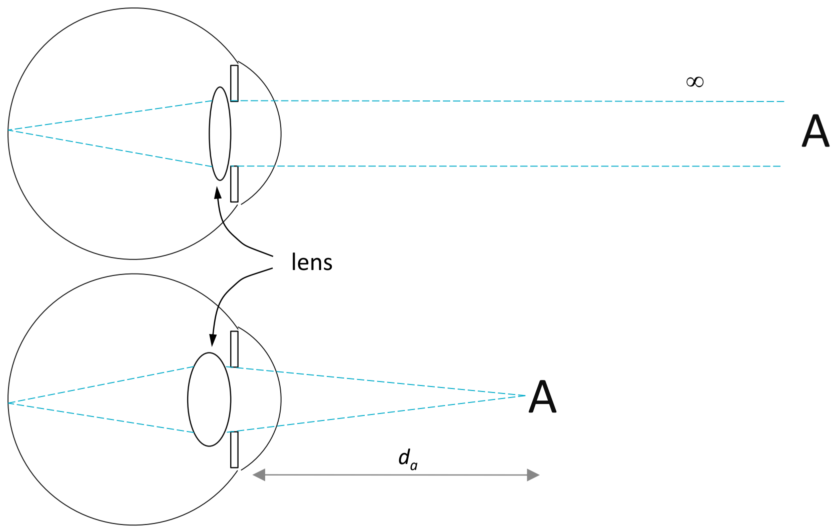


Figure 2.3: *Schematic view of accommodation: looking at a distant object (top) the lens is relaxed; looking at a near object (bottom) the lens is made thicker to accommodate (focus)*

At the same time, when one looks at an object binocularly, the eyes align with it in order to project the target point on the fovea (the central vision point on the retina). This is schematially depicted in Figure 2.4.

Typical stereo displays can trigger the eye vergence response given the disparities in the left and right presented images. But, as the display is physically at a fixed distance, blur-induced accommodation demand remains constant. Actually, the control of the vergence and accommodation responses are not independent, each controlled by retinal image disparity and sharpness, respectively. There are interdependencies so that image sharpness/blurriness can also affect vergence and stereo dis-

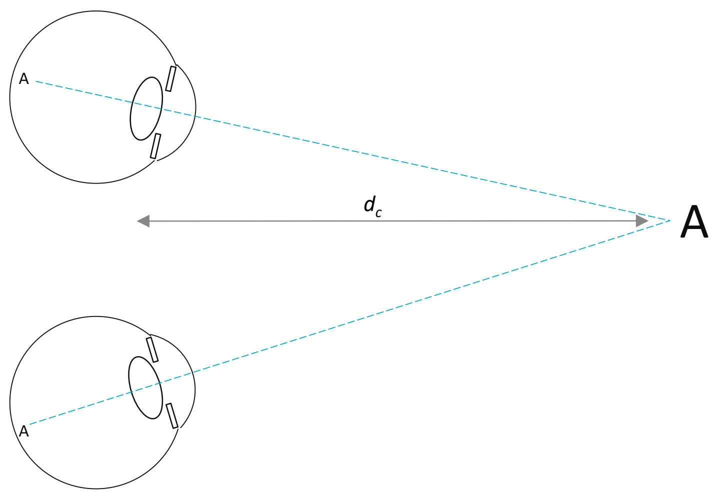


Figure 2.4: *Schematic view of the convergence needed to fixate an object at a given distane.*

parity can affect accommodation. The details of these complex interactions are out of the scope of this thesis. Detailed descriptions of these mechanisms can be found in Schor [82], Templin et al. [93].

Figure 2.5 depicts a pair of eyes watching a stereo screen displaying a small object (letter ‘A’) with positive parallax (the letter appears behind the screen). The viewer’s eyes align with their respective images of the object and thus converge at a point behind the screen. The distance to the screen d_a is where the eyes should accommodate to have a sharp vision of the object, and that is different to the convergence distance d_c.

However, as Inoue and Ohzu [41] found, the disparity in a stereo screen affects accommodation so that the eyes will try to accommodate at the convergence distance d_c. In this situation, the image of the object will be blurry because the eyes do not focus at the correct distance where the screen is located. The difference between d_a and d_c creates a conflict

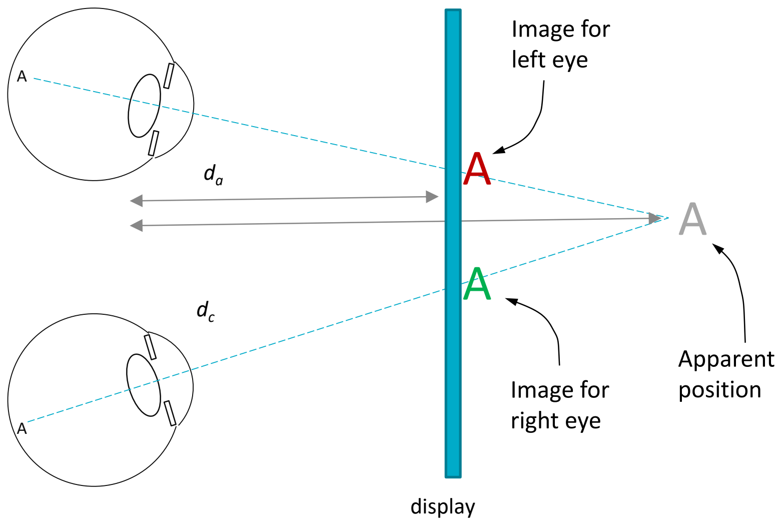


Figure 2.5: *Stereo vision of a stereo display.*

known as the *vergence-accommodation mismatch*.

In the case of viewing a virtual scene in an HMD something similar happens. See Figure 2.6. Here each eye sees a separate screen magnified by optics (one or more lenses), so the distance at which visual axes converge when looking at the displayed letter is not as straightforward to find. The paths these axes follow depend on the position of the displays with respect to the eyes and on the optical power of the lenses. Accommodation distance is also not geometrically obvious. It depends on the optical power of the optics used and the distance between the optics and each screen. This accommodation distance is fixed because those variables (lens optical power and display distances) do not change. The mismatch thus appears in HMDs as well.

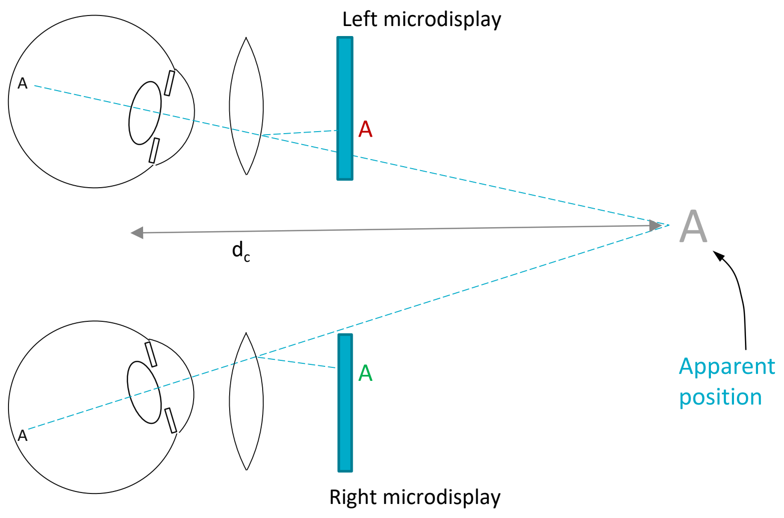


Figure 2.6: *Stereo vision in an HMD*

2.3 Ubiquitous visualization

Weiser [99] introduced the concept of *ubiquitous computing* while at Xerox PARC. He envisioned a future where computing was not done directly at workstations but in which computation would unobtrusively surround people. Similar concepts have been proposed, such as *ambient intelligence* or *pervasive computing*. Our use of the term "ubiquitous" is different. As Chapter 5 will explain, we use the term "ubiquitous visualization" to refer to visualization systems that do not require users to have a complete standalone application and all its data locally installed. Instead, ubiquitous visualization systems can be accessed remotely with little specific software on the client side. The term "ubiquitous interactive visualization" was used by Mc Lane et al. [65] in a similar way but with a different technical approach: their system was based on server-side rendering while we intend to leverage 3D graphics acceleraction on the client side. Web technology has proven to be a platform for ubiquitous delivery and execution of all kinds of applications. Clients only require a computing and display device running a Web browser. The key to ubiquitous interactive 3D visualization is the availability of 3D Web technology or Web3D, i.e. ways to display interactive 3D graphics integrated in Web content.

Web browsers initially supported hypertext-based pages (text containing links to other pages). They started to support embedded images with the release of NCSA's *Mosaic* browser [2]. Raggett [80] proposed the idea of a markup language for Virtual Reality in the Web called *VRML*. A specification of VRML (Virtual Reality Modeling Language) was written by Bell et al. [6], based on SGI's existing *OpenInventor* format. It was followed by a deeply revised version known as *VRML97* that was published as an ISO Standard [97]. VRML97 is a format able to specify virtual worlds including geometries, animations, sound, interaction and scripts to define behaviour. Later, the *X3D* (eXtensible 3D [98]) standard was presented as a successor to VRML with extended functionalities and new encodings, including an XML-based notation.

The above formats allowed the description of 3D content in standard declarative ways. Actually embedding 3D content in Web pages required Web browsers to support these formats. This was done indirectly through third-party plugins that extended the browser's functionalities (e.g. CosmoPlayer, ParallelGraphics Cortona, BS Contact). These plugins usually supported only a subset of existing web browsers and operating systems.

In 2006, a new approach to interactive 3D graphics on the Web was proposed. Vladimir Vukicevic presented a demonstration of a Mozilla Firefox extension that partially added an OpenGL context for the HTML canvas element (the canvas element already supported a 2D graphics context). The *Canvas 3D* extension [95], as it was called, exposed a subset of the *OpenGL ES* 1.1 and 2.0 APIs. With this extension, JavaScript scripts in a Web page could imperatively render 3D graphics using the available hardware acceleration. In the case of the OpenGL ES 2.0 context, rendering is based on programmable vertex and fragment shaders. In 2009, Google presented an alternative technology for accelerated graphics on the Web called *O3D* [77]. Their approach, implemented as a browser plugin, was different, based on a scene graph definition, thus allowing higher level programming.

Canvas 3D eventually evolved into *WebGL* and native support was added to all major Web browsers, on desktop and mobile platforms. It exposes the OpenGL ES 2.0 API to JavaScript in Web pages. Shading in WebGL is specified in programmable shaders written in the GLSL language, bringing a high level of flexibility and allowing complex visual effects. WebGL 1.0 was released as a standard by the Khronos Group [64] in 2011.

WebGL is a relatively low-level API and thus not convenient for many practical applications. To solve this, several JavaScript libraries have been released over the years that expose higher-level interfaces for graphics programming. Examples of these are Three.js [10] and GLGE [9]. Google's O3D, originally a native plugin, was rewritten as a JavaScript library using WebGL as rendering back end. Another interesting library is X3DOM [5], which partially implements an X3D viewer for WebGL-enabled browsers. This means X3D content can be embedded in Web pages and be visualized on any compliant browser with no need for additional plugins.

Chapter 3

Stereo perception and calibration

3.1 Introduction

Despite increasing sophistication in graphics hardware and software providing a high degree of realism, a number of issues have not yet been solved for Virtual Reality displays. The human visual system provides three-dimensional space perception supported by several *cues*, i.e. stimulus features that give hints about true shape, size and depth. Among them, perspective, binocular disparity, convergence and accommodation are considered to provide the main depth cues. Immersive virtual reality addresses these issues using stereoscopic display devices. The stereo pairs displayed make the user perceive the virtual scene as existing in space. There are two main kinds of stereo displays: head-mounted displays (HMD) and stereo screens. Stereo screens can be monitors or projection screens with some filtering means to have each eye see only one of two images presented on the same surface. This work considers HMDs only.

Stereo displays suffer from an issue known as the *vergence-accommodation conflict* (VAC) [103]. The human vision adapts to the objects of interest in the real environment in two main ways: eyes rotate in order to orient their visual axes towards the object and they also change focus in order to project sharp pictures on their retinas. These actions are known as *vergence* and *accommodation*, respectively. There is a third eye response, the pupil aperture, which will not be considered in this work as it does not affect space perception. In natural binocular viewing the two main actions are synchronized: usually the focus distance to an object is the same as the distance at which visual axes intersect. However, in virtual stereoscopy the synchronization is disrupted. Stereoscopic systems render objects in a way that stimulates vergence. In this way objects can be perceived either closer or further than the display surface. However, the eyes have to focus at the display, which is placed at a constant distance. So, there is a conflict between focus and vergence that cause an unnatural view and visual fatigue, especially in near objects. This issue was studied by Hoffman et al. [38].

Stereo HMDs use two separate small screens (often microdisplays) with magnifying optics. When worn on a user's head the two optics and displays are located in front of each of the user's eyes. The stereoscopic system renders in each display a slightly different view of the virtual environment that the user perceives as three-dimensional. The distance the eyes have to focus on (accommodation distance) is not just the physical distance to the displays because the microdisplays are behind optics that alter light beams. Moreover, the convergence distance where visual axes meet is not easily identified due to the magnifying effect of the optics.

Rendering images for correct geometric perception has special requirements. Steinicke et al. [90] studied how the field of view (FOV) affects perception. They use *geometric field of view* (GFOV) to denote the FOV used for generating the perspective images, and *display field of view* (DFOV) to denote the FOV observed, i.e. the angle subtended by the display screen from the viewer's point of view. The authors state that a geometrically correct image, as if the screen was a window to the virtual world, occurs when GFOV and DFOV match. So, rendering geometrically correct 3D stereo pairs requires knowledge of the view frustum of the display for each of the eyes.

The problems we have just considered must be taken into account when a stereo system defines the frustums that will be used to render images. When the precise location of all lens elements and microdisplays are defined with precision, optical display properties can be computed by mathematical models of light transport. However, nominal parameters (such as the FOV) provided by HMD manufacturers may have small variations for each device. It has to be noted that the high magnification provided by HMD optics can transform small positioning errors of microdisplays into significant macroscopic effects that invalidate nominal values. Several calibration methods for obtaining the view parameters of HMDs have been proposed in the past (see [25], [52], [33]), but they nearly always rely on subjective impressions by the HMD wearer or are only suitable for optical see-through HMDs (see [32], [46]). We believe that users are likely to introduce uncertainties and that user-independent objective methods would help provide more repeatable and reliable information.

In our contribution we argue that to obtain a stereo view with approximately correct depth perception on a specific HMD, its characteristics should be measured using objective calibration processes. This means that a physical setup will determine with precision the characteristics of a given HMD. Gilson et al. [33] provided an objective approach but has the limitation that is does not explicitly consider the stereo nature of the perceived space.

The second contribution of this chapter tackles accommodation demand, i.e. the effort requested to the eye to focus at specific distances. In most commercial HMDs the accommodation demand is fixed. Some of them have a manually adjustable focus setting which the user can set in order to avoid wearing glasses. Then, it remains fixed. As we will see, if this mechanism could be controlled by the VR system we may have a valuable tool that will alleviate the vergence-accommodation conflict. The display system will change focus according to the actual vergence. Kramida and Varshney [50] presented a review of the state of the art on potential solutions to the VAC in HMDs. Within their classification, our system can be described as *varifocal*.

Our approach requires solving the following problems: (i) computation of the view frustums by calibration of the HMD, (ii) control of the accommodation demand.

We propose a new framework that combines all these ideas for displaying images with correctly perceived shape and size in HMD-like stereo displays simultaneously stimulating vergence and accommodation. The proposed framework includes calibration methods to objectively measure stereo view frustum parameters that will ensure correct size and distance perception. This work provides users with stereo pairs with convergence and accommodation cues. The current framework does not consider nonlinear image distortion.

The chapter is organized as follows. Section 3.2 deals with objective stereo geometry calibration. Section 3.3 addresses focus control to alleviate the vergence-accommodation conflict. Section 3.4 discusses experimental results. The chapter finishes with conclusions in Section 6.1. Throughout the text, terms such as *focus*, *accommodation* and *optical power* will be used interchangeably when their meaning is evident. When describing the hardware, the term *accommodation* will actually refer to *accommodation demand*, that is, the accommodation optical power requested to the eye in order to see a sharp image.

3.2 Objective stereo display calibration

In 3D computer graphics, images are rendered based on a set of parameters that define the projection characteristics. At least the field of view (FOV) must be defined to control perspective. A small FOV produces a view similar to a *tele* setting in a camera lens, while a large FOV creates a more perspective-distorted view similar to a *wide angle* lens. In order to create a faithful view in Virtual Reality the FOV used in rendering must match the field of view that the display appears to have from the point of view of the user wearing the HMD.

When producing stereoscopic projections, additional parameters are used to control the final effect. They should correspond to the viewing conditions. If the stereoscopic projections do not match viewing conditions, a different virtual space will be perceived, virtual distances, sizes and shapes will be distorted. These distortions have been studied by Benzeroual et al. [7] (for 3D films) and by Kelly et al. [47] (for VR environments).

In real life, when staring at a point in space, the eyes rotate such that their visual axes intersect at that point. The distance to this intersection point, the convergence distance, is the distance at which we perceive the point to be. When looking at a very distant object, visual axes remain approximately parallel. A correct stereoscopic perspective projection should have the same properties.

Modelling visual axes and their intersection in a stereoscopic *screen* only requires knowing the position of the user in front of the screen, the user's interpupillary distance and the screen dimensions. For more precision, the user's head orientation can be used to find the location of each eye relative to the screen.

In the case of a *head-mounted display* the situation becomes more complicated due to the fact that the left and right images use separate microdisplays which are seen through separate optics. The lenses of the HMD distort and magnify the image of the microdisplays. The dimensions and precise position of the microdisplays are also distorted and magnified. So, small positioning errors of each display may lead to significant misalignments affecting the stereoscopically perceived geometry. Horizontal misalignment alters depth perception as the vergences will be altered leading to a different convergence distance. And vertical misalignment may lead to double vision and no depth perception at all: the eyes can only fuse if left and right corresponding stimuli are within a small angular elevation difference. This is known as Vertical Disparity Amplitude tolerance (see Fukuda et al. [31] and Di et al. [20]).

Most VR rendering systems assume that the microdisplays of an HMD are centred in front of each of the user's eyes. So, the projection matrices that generate the images on the HMD are known to be symmetric perspective projection matrices.

However, usually the microdisplays have small offsets (see Figure 3.1). Even HMDs of the same model can have different offsets. In order to deliver a correct view in the HMD, the VR rendering system must know these offsets to compute the correct projection matrices. So, we need a simple, precise and objective calibration method that finds these data for each HMD. Figure 3.1 shows the correct view frustums that must be found.

Several calibration procedures have been proposed for different stereoscopic displays. For CAVE-like setups Ponto et al. [78] introduced a perceptual calibration procedure. The calibration of see-through HMDs has been addressed by many authors, such as Figl et al. [25], Figl et al. [26], Gilson et al. [32], Kellner et al. [46], Makibuchi et al. [61], Itoh and Klinker [42]. Indeed, correct projection in those devices used in augmented reality is crucial or otherwise virtual and real objects will appear misaligned. These methods take advantage of the see-through nature of those devices and require users to look at physical targets placed in front of the HMD. They are thus not usable in immersive HMDs (i.e. non-see-through) in which there is no view of the real world. In

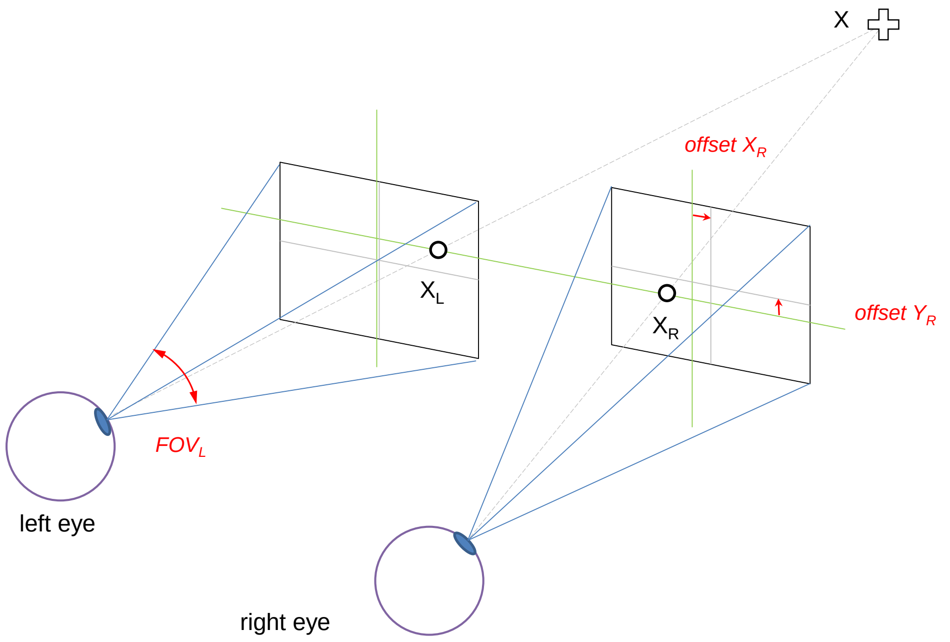


Figure 3.1: *View of the display from each eye. Each display is offset with respect to the perfectly centred position. Thus, in order to represent a point that is perceived at position* $\mathbf{X}$*, its left and right positions* $\mathbf{x_L}$*,* $\mathbf{x_R}$ *have to be drawn at vertically and horizontally offset positions in each display.*

immersive HMDs the lack of real references makes projection errors less noticeable but still a correct geometric perception requires correct projection parameters. In this case applications usually either use the device's nominal parameters or employ subjective calibration methods, such as Kuhl et al. [51], because of the lack of externally observable features (only the wearer sees the HMD's displays).

In contrast, Gilson et al. [33] proposed a method for objective calibration of immersive HMDs that uses one camera placed in the position of the user's eye. It is an evolution of their earlier method for see-through HMDs (Gilson et al. [32]). The camera captures a pattern presented in the HMD display. From analysis of the captured pattern, they compute a mapping between the camera image and the HMD display coordinates. Then the HMD is removed while keeping the camera still and a marker is placed in the space in front. The marker is tracked with an external tracking system while it moves in the space in front of the camera. By relating the projection of the marker in the camera image, its tracked position in space and the position of the displayed pattern in the same image, the method computes the theoretical projection matrix of one display of the HMD. It then proceeds with the other display to create a stereo projection pair.

We see several limitations in this approach. First, it requires a complex setup including an external high-end position tracking system and a precise positioning of the camera with respect to the HMD. Then, the method calibrates the left and right displays in sequence. Stereo depth perception depends on the relation between the apparent position of points in each display. Then, such a sequential calibration can easily distort depth percepts. The authors acknowledge that a slight modification in camera position with respect to the HMD results in a different computed view frustum. Thus, the unintended different position or orientation of the camera in front of each of the displays will lead to shifted visual axes and modified stereo distances.

The following subsection describes our proposed calibration method to obtain the parameters that allow the stereo system to compute view frustums for acceptable depth percepts.

3.2.1 New stereo view calibration procedure

Our approach shares some concepts with the one proposed by Gilson et al. but does not require an external tracking system and eliminates the uncertainty in position of the camera with respect to each of the displays, among other differences. We propose the use of a pair of rigidly attached cameras to simulate the user's eyes (*calibration cameras* or CC in Figure 3.2). This pair gets a view of the virtual space projected by the rendering system as perceived by a user. The camera pair is previously calibrated, intrinsically (each camera) and extrinsically (the right camera with respect to the left). This is computed from a set of images of a calibration pattern (a chessboard) in different positions taken with the camera pair (see Figure 3.3). This calibration step is performed only once for our CCs and is used later on for the calibration of any HMD. Its results are the matrices of intrinsic parameters ($\mathbf{K_L}, \mathbf{K_R}$) and extrinsic parameters ($\mathbf{R_R}, \mathbf{t_R}$) that will be later referenced in Equation (3.1).

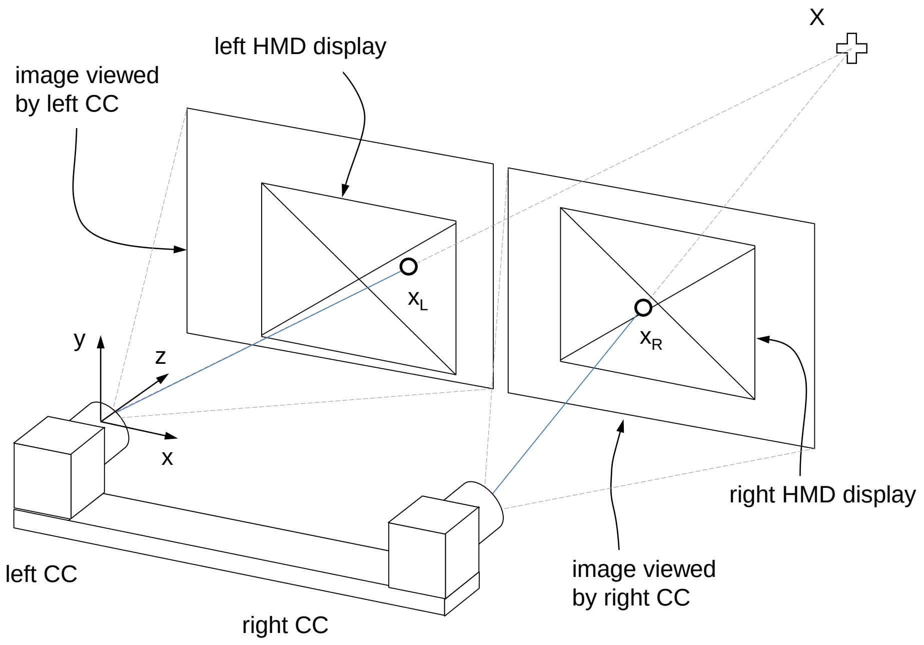


Figure 3.2: *HMD stereo projections calibration. A pair of rigidly attached CC (calibration cameras) substitute the eyes. The HMD displays' misalignment is exaggerated in the illustration.*

We have defined a simplified projection model. The rendering module of the VR system will use view frustums defined by 6 parameters. The projection model is defined by two virtual cameras separated by the

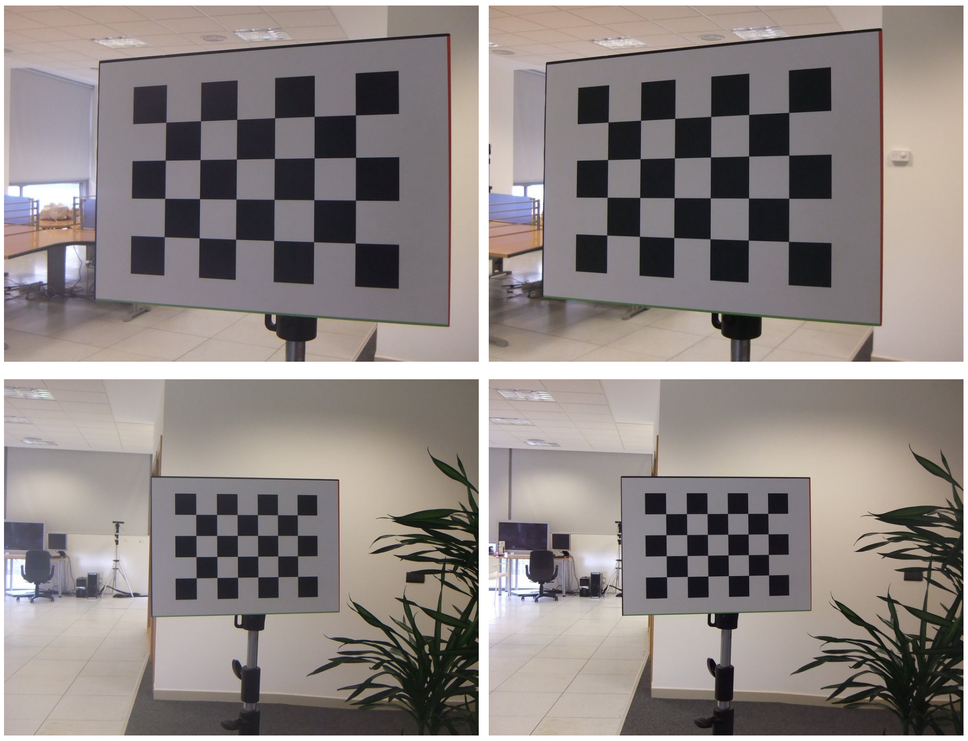


Figure 3.3: *A subset of the stereo images used for calibrating the camera pair (left camera and right camera).*

interpupillary distance. These virtual cameras have their principal axes parallel. Our *view frustum pair* (VFP) parameters are the following:

- *Interpupillary distance* or IPD. This is the separation between the two eye centres. Will be set to a fixed typical value.
- *Displays aspect ratio*. This is the ratio of width to height of the microdisplays.
- *Horizontal FOV angle of the left and right displays*. Note that these can be different, as we will see.
- *Horizontal and vertical offsets between the displays*. These parameters describe the relative displacement in both axes between the view of the microdisplays with respect to a perfectly centred

position in front of each eye. They are measured with respect to display size so that, for example, a vertical value of 0.5 would mean the right display appears shifted vertically a length equal to half the screen height.

This model does not define separate offsets for each display but a relative displacement between them. This enables a much simpler operation and still ensures depth perception.

Interpupillary distance and display aspect ratio are known in advance. However, FOV and offsets can vary for each specific HMD. These parameters must be known by the rendering module in the VR system in order to create correct virtual world projections. The calibration process we propose measures these parameters in a specific HMD. Even HMDs of the same model may have slightly different values. This fact shows the importance of using a simple and precise HMD calibration system.

Our calibration process is based on the projection of an arbitrary point in 3D space onto two different image planes, each with its own coordinate system: the HMD *displays coordinate systems* and the *calibration cameras coordinate systems*. In the equations in this section $\mathbf{x}^{\mathbf{cam}}$ will denote points projected on a CC image, and $\mathbf{x}^{\mathbf{disp}}$ will denote points projected on an HMD display. Both are in homogeneous coordinates, as they are used in projective geometry equations.

a) CC image coordinate system

Given a point $\mathbf{X}$ in 3D space and knowing the CC parameters, its projection on each CC image coordinate system is determined by Equation 3.1.

$$\mathbf{x}^{\mathbf{cam}} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X} \tag{3.1}$$

We have a pair of cameras, so there are actually two sets of parameters: $\mathbf{K_L}$, $\mathbf{K_R}$, $\mathbf{R_L}$, $\mathbf{t_L}$, $\mathbf{R_R}$, $\mathbf{t_R}$. In our case, since our extrinsic parameters take the left camera as reference, $\mathbf{R_L}$ is an identity matrix and $\mathbf{t_L}$ is a null vector. $\mathbf{K_L}$, $\mathbf{K_R}$, $\mathbf{R_R}$ and $\mathbf{t_R}$ are computed once for the CC pair and remain constant for the calibration of any HMD.

b) Display coordinate system

On the other hand, in the VR rendering system, the view frustum for each eye can be defined by a projection matrix $\mathbf{P}$. For this matrix we have chosen the form typically used in computer vision, as expressed in Equation 3.2.

$$\mathbf{P} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

This matrix should be derived from the 6 model parameters we have defined. However, we do not know their values until the HMD calibration process is completed.

Each eye sees the virtual world from a different point of view. There is no rotation (i.e. the rotation matrix is an identity matrix) because we have defined our virtual cameras to be parallel and aligned with the CC Z axis. Combining projection matrix and point of view, the projection of a point $\mathbf{X}$ in 3D space by the rendering system into one of the displays is expressed by Equation 3.3.

$$\mathbf{x^{disp}} = \mathbf{P}[\mathbf{I}_3|\mathbf{t}]\mathbf{X} \tag{3.3}$$

Again there are actually two matrices, $\mathbf{P_L}$ and $\mathbf{P_R}$ and two translation vectors $\mathbf{t_L}$ and $\mathbf{t_R}$ for the left and right displays, respectively. These subscripts will be omitted for clarity throughout this section when the specific side is irrelevant as both are processed the same way. As the equation expresses, there is no rotation as both virtual cameras have their principal axis parallel to the Z axis. And, as before, the reference system is the left eye, so $\mathbf{t_L}$ is a null vector and $\mathbf{t_R}$ is $(-\mathrm{IPD}, 0, 0)$. $\mathbf{P_L}$ and $\mathbf{P_R}$ are the unknowns that will be computed by the calibration process.

c) Mapping between display and CCimage coordinates

For each of the left and right sides, any virtual 3D point $\mathbf{X}$ is projected onto a point $\mathbf{x^{disp}}$ in the *display coordinate system* as expressed in Equation 3.3. The same 3D point is projected onto a point $\mathbf{x^{cam}}$ in the *camera image coordinate system*, as expressed in Equation 3.1. These two projected points have to be equivalent but they are in different coordinate systems so their equations cannot be combined. A mapping between these two coordinate systems is needed to solve the problem. In a low distortion environment this mapping can be approximated by a 3×3 transform matrix $\mathbf{M}$ in homogenous coordinates (i.e. a homography) as shown in (Equation 3.4).

$$\mathbf{x^{disp}} = \mathbf{M}\mathbf{x^{cam}} \tag{3.4}$$

Now we explain how this mapping $\mathbf{M}$ can be estimated. The camera images contain a view of the microdisplays (see Figure 3.2). By finding a set of corresponding points in both display coordinates and CC image coordinates, the transformation $\mathbf{M}$ can be approximated. At least 4 point correspondences are needed. Actually, since the displays are usually nearly perpendicular to the camera view orientation, the mapping is approximately equivalent to an affine transformation. In that case the minimum requirement can be simplified to 3 point correspondences and the bottom row of $\mathbf{M}$ would be $[0, 0, 1]$.

d) Computing the mapping

Our proposed calibration procedure employs an interactive process to determine this mapping. A fixed grid pattern is presented on both displays and a pair of images is grabbed with the CC cameras. Then these images are presented to a user who has to click on at least 4 specific pattern points with the mouse. We know the display coordinates of the points, as we have rendered them, and the user marks their corresponding camera image coordinates. From these point correspondences between display coordinates and camera image coordinates the algorithm computes the transform matrix **M** that relates the two images (display and camera).

e) Finding the projection parameters

Once the mapping **M** is known, we can find the elements of matrix **P** by minimization of the reprojection error as defined by the following expression.

$$\min \sum_i \left\| p(\mathbf{x}_i^{\mathbf{cam}}) - p(\mathbf{M}\mathbf{x}_i^{\mathbf{disp}}) \right\| \tag{3.5}$$

where $\mathbf{x}_i$ for $i = 1...N$ are the projections of a set of arbitrary virtual 3D points $\mathbf{X}_i$ and $p(\mathbf{x})$ is given by Equation 3.6:

$$p(\mathbf{x}) = p([x, y, z]^T) = \begin{bmatrix} x/z \\ y/z \end{bmatrix} \tag{3.6}$$

Substituting the expressions from Equations 3.1 and 3.3 Equation 3.5 expands to:

$$\min \sum_i \| p(\mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X}_i) - p(\mathbf{M}\mathbf{P}[\mathbf{I}_3|\mathbf{t}]\mathbf{X}_i) \| \tag{3.7}$$

where the only unknowns are the 4 unknown elements of matrix **P**, the rendering projection matrix. The Levenberg-Marquardt algorithm

[62] is applied to solve the minimization problem and estimate the 4 unknowns.

In each iteration the algorithm uses a set of N random points in the space in front of the viewer and visible by both eyes and projects them with the current value of the projection parameters. Their projection in display coordinates is transformed to camera coordinates using the computed mapping $\mathbf{M}$ and the distance to their projection by the cameras in pixels is used as reprojection error. The projection is initialized with a fixed set of parameters corresponding to the HMD's nominal perspective projection with no offsets or a standard guess if there is no such information. The Levenberg-Marquardt algorithm finds the projection matrix elements f_x, f_y, c_x, c_y that minimize reprojection error for each display. The process is executed for the left and right displays separately.

3.2.2 Extraction of projection parameters

The algorithm in the previous section computes a projection matrix $\mathbf{P}$ for each display. That should be sufficient as such a matrix can be converted into, for example, an OpenGL projection matrix ready for rendering graphics. But our rendering system does not use these projection matrices directly for several reasons. As the following sections will show, the projection characteristics are not fixed, but vary when varying optical power (i.e. setting different accommodation distances). For this reason the rendering engine has to continually adjust projections by interpolating values obtained in different conditions of optical power.

The computed projection matrices are affected by the orientation of the cameras with respect to the HMD. This means that the projections obtained for different optical powers will have unintended variations if the cameras are not always positioned with the exact same pose.

This is why we prefer to compute a relative, inter-display offset that is unaffected by small camera deviations.

Instead of the raw projection matrices our VR system uses the 6 parameters described at the beginning of Section 3.2.1 to recreate new projection matrices for rendering. The first two parameters (*Interpupillary distance* and *display aspect ratio* are known in advance. The remaining 4 (left and right *FOVs* and *horizontal and vertical interdisplay offsets*) are extracted from the elements of matrix $\mathbf{P}$ as follows.

The horizontal and vertical FOVs are related to the f_x and f_y elements. They can be extracted from the computed projection matrix as shown in Equations 3.8 and 3.9, where w and h are the width and height in pixels of the displays.

$$FOV_H = 2 \arctan\left(\frac{w}{2f_x}\right) \tag{3.8}$$

$$FOV_V = 2 \arctan\left(\frac{h}{2f_y}\right) \tag{3.9}$$

And the relative offsets are related to the principal point parameters c_x and c_y. They are the difference of these parameters between the left and right projections, divided by display resolution in order to make them resolution independent (Equations 3.10 and 3.11). The subscripts L and R denote the parameters for the left and right sides, respectively, which are separately obtained.

$$\text{offset}_x = \frac{c_{xL} - c_{xR}}{w} \tag{3.10}$$

$$\text{offset}_y = -\frac{c_{yL} - c_{yR}}{h} \tag{3.11}$$

The rendering engine uses the field of view value to compute a standard *OpenGL* projection matrix, and then modifies this matrix with added horizontal and vertical offsets. As we have relative offsets between displays, we apply half of each offset to the left image and half to the right image with opposite signs.

Actually, the cameras can have some optical distortion and a set of distortion coefficients are obtained as part of the camera calibration. The above camera projection equations, such as Equation 3.1 do not show this effect for simplicity but it is taken into account. We apply these distortions when computing the cost function in the minimization process.

3.3 Accommodation control

As stated in the introduction, the vergence-accommodation conflict arises due to the fixed accommodation demand created by conventional stereoscopic displays, including HMDs. We intend to alleviate this issue by providing variable accommodation demand. The goal is to vary accommodation distance in order to match the convergence distance continuously. As we will see, an additional problem is to know the actual user's convergence distance.

The idea is to be able to modify a physical variable which makes accommodation vary. Then, the relation between this variable and accommodation must be known in order to set the desired accommodation from the computer.

Fischer et al. [27] proposed eight *theoretically* feasible methods to achieve variable accommodation in HMDs. Out of them we have *implemented* and tried the following two as they seem the simplest and most feasible in practice.

1. Translation of the display: motion of the microdisplays along the optical axis by means of a precision actuator modifies their perceived optical power (Figure 3.4). A closer accommodation demand is achieved by approaching the display to the eye.

2. Use of a liquid-filled lens: One of the lenses in the optical path is an electro-optical lens, i.e. a liquid-filled lens that modifies its shape and focal length as a function of an applied electrical current (Figure 3.6).

Most of the other methods in Fischer's enumeration require too complex components, such as aspherical elements, multifocal lenses, liquid crystals or adaptive optics, and are harder to control. One of the other methods, the translation of the lens assembly, is similar to the translation of the display (they both modify the distance between the lenses and the microdisplays). This method has the disadvantage that vignetting and image clipping may appear as the lenses move. On the other hand, the survey presented by Kramida and Varshney in [50] mentions liquid lenses and translating lenses (called *sliding optics* in their paper), but not translating displays.

We will first explain the use and calibration of the first type of design, which is schematically depicted in Figure 3.4. In this case, the microdisplay for each eye can be moved back and forth along the Z axis with submillimetre precision. Each display position along this axis corresponds to one accommodation value.

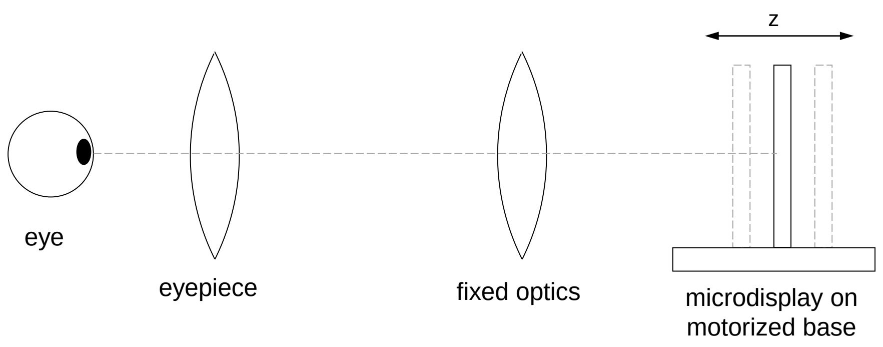


Figure 3.4: *Simplified schematic diagram of variable focus design based on movable display on a motorized base.*

Instead of directly controlling *accommodation distance*, we will control its reciprocal, the *accommodation optical power* expressed in diopters, which is more convenient (1 D = 1 m^{-1}). We define the *system optical power* as the optical power perceived by the user and that his/her eyes have to accommodate (i.e. accommodation demand). When looking at a distant point, the system optical power is zero, i.e. there is no accommodation demand. A near object located at a distance d corresponds to a negative optical power $-1/d$ forcing the eye to apply a positive accommodation power $1/d$ to compensate. System optical power is:

$$p = -\frac{1}{d} \tag{3.12}$$

where d is the distance of the displayed object.

In order to set optical power to a desired value there must be a known relation between action and effect: how optical power varies as a function of display position. One way to establish this relation would be to accurately model the theoretical optical system and determine it mathematically. Another way is to obtain the relations experimentally. We propose a calibration method to obtain this relation without detailed knowledge of the optical design.

3.3.1 Movable display calibration

The basic idea of our method is to measure several pairs of values (display position and its corresponding optical power) and fit parametric curves to them. These functions will be used by the computer to set the desired accommodation.

Our calibration procedure for each of the visors in the hardware with moving displays is as follows:

1. The camera focused at infinity looks at the display through the eyepiece.

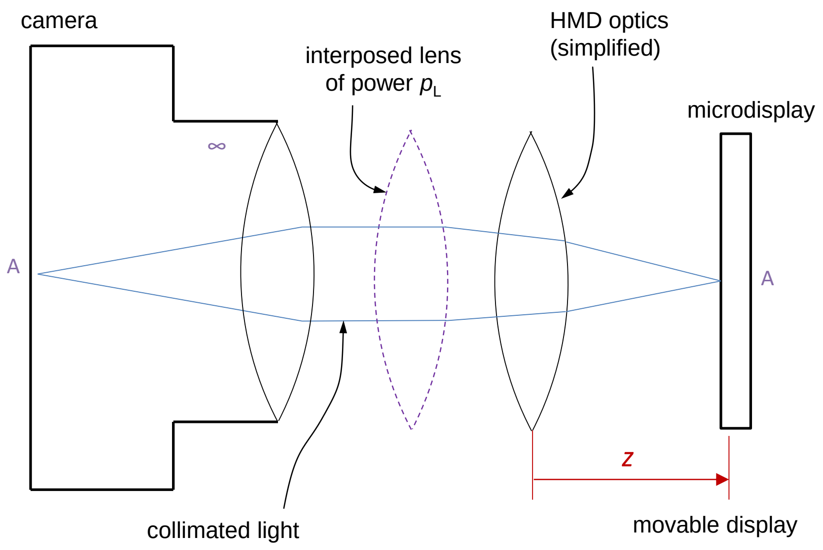


Figure 3.5: *Simplified schematic calibration setup: camera focused at infinity looking at HMD. Several lenses of known powers p_L are placed between camera and eyepiece sequentially.*

2. We move the display in z until we see the display appear sharp in the camera image. That position z corresponds to power $p = 0$.
3. Now a lens of power p_L is placed in front of the eyepiece and the display is moved in z until the display appears sharp again. The new position z corresponds to power $p = -p_L$. (see Figure 3.5).
4. Step 3 is repeated for several lenses with positive and negative powers.

The result of this procedure is a set of points (z_i, p_i). The VR system needs a continuous function $z(p)$ so that for any desired optical power it knows where to position the display.

A curve that approximates the function can be obtained by fitting a parametric model to the available data points using Levenberg-Marquardt least-squares minimization. We use third-degree polynomials which are easy to work with. This process computes the optimal parameters (a_0, a_1, a_2, a_3) in Equation 3.13.

$$z(p) = a_0 + a_1 p + a_2 p^2 + a_3 p^3 \tag{3.13}$$

On the other hand, Shiwa et al. [88] consider that, given a display placed at a distance x behind a lens of focal length f, its virtual image is perceived at a distance d given by $f\,x/(f-x)$. Following this assumption we can also model $z(p)$ as the rational function in Equation 3.14.

$$z(p) = \frac{b_0 + b_1 p}{b_2 + p} \tag{3.14}$$

with three parameters (b_0, b_1, b_2).

Our system can use either function, Equation 3.13 or 3.14.

3.3.2 Electro-optical lens calibration

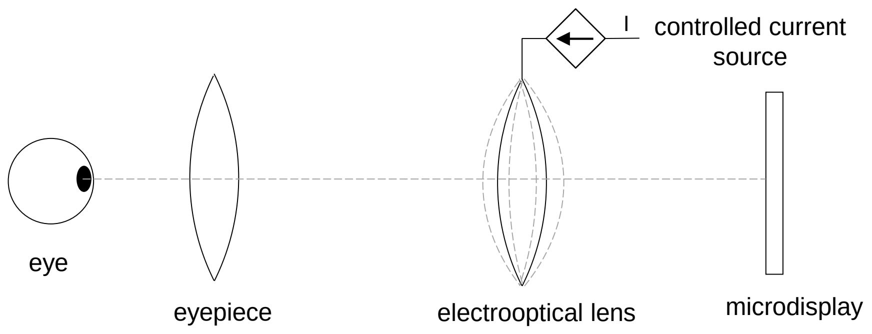


Figure 3.6: *Simplified schematic diagram of variable focus design based on electrically-tunable lenses.*

In the case of focus control based on an electro-optical lens (see Figure 3.6) the relationship between lens current intensity and accommodation optical power is needed. Electro-optical lens manufacturers provide nominal response curves for their lenses. That is, graphed curves relating applied current intensity and obtained lens focal length or optical power. It can be thought that using these manufacturer specifications for control would be enough. However, there are two drawbacks. First, the specifications define the optical power of the lens itself, but this lens is not alone in the system (the optics are composed of multiple lenses), and the combined optical power presented to the observer is different. And finally, in our tests we observed that nominal response curves are a representation of typical or average behaviour but in reality each individual lens has a notably different response.

Therefore, instead of relying on nominal data we propose a calibration procedure similar to that used for movable displays and explained in the previous section. At each step, instead of moving the display, the lens current intensity is adjusted until the display image appears sharp in the camera image. A set of points (I_i, p_i) are collected and an approximating parametric function $I(p)$ is computed. The function can be a polynomial as in Equation 3.13 that best fits the sampled points.

3.3.3 Accommodation control

Once the control curves for the specific hardware are built with the explained procedure, the system is able to set any optical power and thus an accommodation distance to the display at any moment. For instance, on a movable display setup, for a desired focus distance d the system will compute the corresponding optical power p with Equation 3.12 and will move the display to the position given by function $z(p)$.

Even if the application can vary accommodation at any given time, the question is what accommodation distance must be set at each moment. In theory, the system optical power should match the user's actual accommodation state so that whatever he looks at will appear naturally focused as in the real world. However, it is quite difficult to obtain the eye's accommodation in real time in a VR setup. We think VR systems have two options:

1. The VR system may expect that the user focuses his gaze on the most relevant object displayed. So, the VR system continuously adjusts focus to the distance to that object. The user will have a comfortable view of this object, but discomfort may appear if the user tries to stare at another object placed at a significantly different distance.

2. If the hardware is equipped with an eye tracking module with 3D point of gaze detection capability, then the VR system using our method can set the focus distance to the convergence distance of the viewer's gaze.

In this work only the first option has been implemented and tested.

3.3.4 Variable field of view adjustment

As was explained in Section 3.2, the projection, and specifically the *field of view* used by the VR rendering system has to match the view of the display. This projection can be considered constant if the focus adjustment remains unchanged. However, it was observed, as predicted, that different focus settings can alter the view of the display and require adjustments in the field of view.

For example, if focus control is based on axial display motion, when the display approaches the eye to produce closer focus demand, it obviously appears larger due to perspective. The rendering software must then adjust its FOV to the new apparent size in order to keep virtual space geometry correct. A liquid-filled lens control may also change the field of view as a lens changes its magnification when its optical power is modified, but the change in the system depends on the complete optical design.

The rendering system has to know the FOV at every focus position, which may vary continuously, so for every new value of the system optical power it needs to compute the new FOV. What is needed is a function $FOV(p)$ that models this relation.

For a particular optical power, the field of view can be computed as part of our camera-based display calibration method described in Section 3.2.1. The idea is to capture images of the displays with the focus control set to different optical powers using the modelled functions $z(p)$ or $I(p)$, depending on the hardware type. For each focus setting the FOV is calculated from the captured images using the presented technique. Then, a parametric function is fitted to the collected points (p_i, FOV_i).

We observed experimentally that the relationship between optical power and FOV is approximately linear, so a linear function will be used for this adjustment. When the VR system changes the optical power, the VR rendering system updates its FOV to the value provided by linear interpolation.

3.3.5 Variation of interdisplay offsets

The previous section explains that the field of view depends on the current optical power because the display magnification varies. Interdisplay offsets are related to the slight offset of the center of the displays with respect to the optical axis of the lenses. As magnification occurs symmetrically around the optical axis it is expected that a decentred display will appear to *move* as optical power changes. Interdisplay offsets thus vary when optical power changes.

With the small decentring and the limited FOV variations that can be expected in actual hardware, the variation of interdisplay offsets will probably be small. But it should be measured and taken into account for a better view control if noticeable effects are observed. The procedure is the same as with the FOV: the stereo calibration procedure is performed with the hardware set to several different optical powers. Then, parametric curves for $\text{offset}_x(p)$ and $\text{offset}_y(p)$ are fitted.

3.3.6 Compensation of user's ametropia

Using optical power as the controlled variable instead of accommodation distance has the added benefit that spherical ametropia or refractive error can be easily compensated. Then, users having *myopia* or *hyperopia* can use the device without wearing glasses. This simply requires adding the user's corrective power to the system optical power. The system optical power from Equation 3.12 now becomes:

$$p = -\frac{1}{d} + p_c \tag{3.15}$$

where p_c is the user's corrective power in diopters (negative for myopia and positive for hyperopia). This value has to be manually configured for each user.

3.4 Results and discussion

3.4.1 Experimental hardware setup

This work has been developed and tested on custom-built hardware that has special features needed for variable focus control. Five experimental setups were used for testing the different control designs. The hardware, while bulky and not actually wearable, is conceptually similar to a head-mounted display. It has a pair of microdisplays, one per eye, and multiple lens optics that enable focusing and magnifying the images. The setups were built in two different versions, each implementing one of the dynamic focus designs discussed in Section 3.3 (see Figures 3.4 and 3.6). In one version the microdisplays are mounted on bases with piezoelectric motors allowing them to move along the optical axis with submillimeter accuracy, while the other version includes a pair of electrically tunable lenses that can alter their optical power by varying an electrical current.

The microdisplays, as seen through the optics, do not show a noticeable distortion, so they are suitable for use with this framework which, as stated, does not take into account non-linear distortions.

All the techniques have been implemented in software for our experimental setups, including display calibration, focus control calibration and a visualization engine that uses the obtained parameters for graphics rendering and for controlling optical power via the hardware.

The following subsections present the practical application of these techniques on the actual hardware setups and the results obtained. Section 3.4.2 presents results related to the method that we proposed in Section 3.2.1 and Section 3.4.3 presents results related to the method proposed in Section 3.3.1.

3.4.2 Stereo view calibration

Field of view and stereoscopic parameters of the five setups were calibrated using the algorithms provided in Section 3.2.1. This subsection describes how the method was applied to our hardware and highlights implementation details of the intermediate steps.

As a preliminary step, the intrinsic and extrinsic parameters of the calibration camera pair (CC, see Figure 3.2) were obtained using a typical checkerboard pattern and functions from the *OpenCV* [43] computer vision library.

In the calibration process of an HMD, the first step is to capture images of the HMD displays with the CC pair. It was placed in the HMD and a pair of images were captured while the displays presented a calibration grid. Figure 3.7 shows the images of the displays captured by the left and right cameras.

The second step was to compute a mapping between CC image coordinates and display coordinates. Each image is presented to a user who has to click on four specific points of the grid. In this way we get the CC image coordinates of the four points. On the other hand, as we have

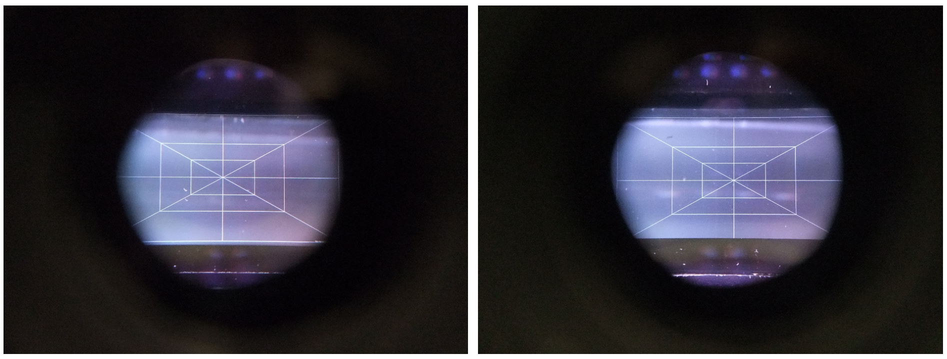


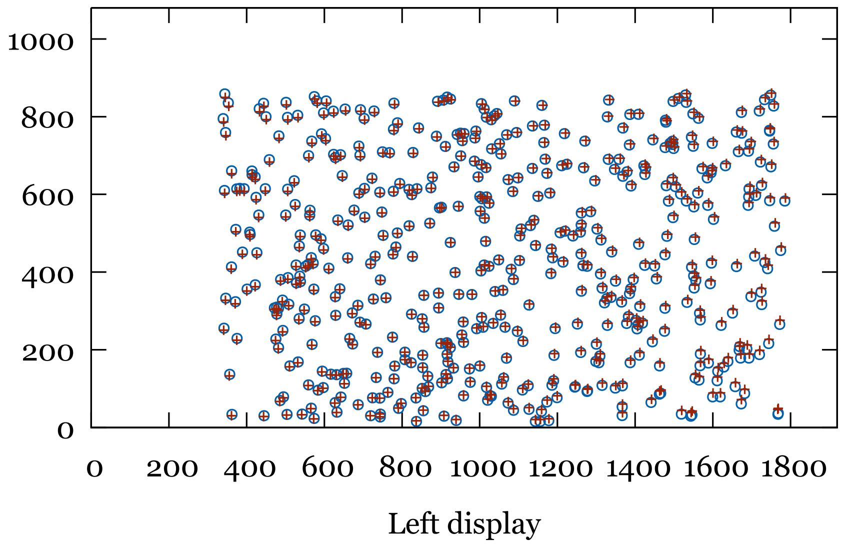
Figure 3.7: *Left and right displays showing the calibration grid as seen in the left and right CC images.*

generated the grid, we know the display coordinates of these points. So we can compute the mapping homography **M** as explained before.

Then, the left and right view projection matrices (Equation 3.2) are obtained by minimizing the reprojection error of a set of 3D points. Our software uses a set of 500 random points in 3D space. They are inside a volume that is visible by both eyes. They suitably fill the virtual space seen in the displays.

In order to verify the quality of the calibration process we perform the following test. Figure 3.8 shows the above points projected into the left and right displays by the calibrated view projection matrices (as circles). Then the mapping homography **M** is applied to these points as seen by the calibration cameras (in CC image coordinates) and the result, now expressed in the display coordinate system, is also drawn in the figure (as crosses). The figure shows a good fit between the projected points on both sides, with RMS errors of 4.2 and 3.2 pixels, respectively (4 pixels in these views are equivalent to about 0.06 degrees deviation).

In Figure 3.8 we can see that there is a small vertical disparity of any given projected point in the left and right displays. This means that our computation of the projection matrices creates a vertical displacement of the rendered images that compensates the vertical offset of the HMD displays. Horizontally there is a significant bias as well, which is due in



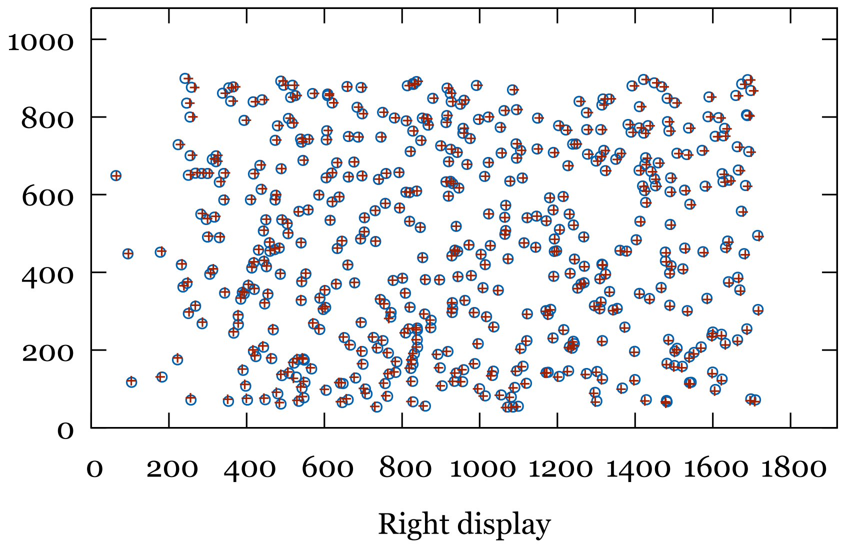


Figure 3.8: *A set of 500 random 3D points projected into the displays by the calibrated view frustums (circles) and as seen by the cameras (crosses) for the left and right sides (top and bottom, respectively) in display pixel coordinates.*

part to the mere stereo projection disparity and in part to the horizontal interdisplay offset. Two specific parameters in our view frustum pair parameters (VFP) control these offsets.

These results show that our method corrects the horizontal and vertical small misalignment of the physical displays (see Figure 3.1 in Section 3.2). The rendering software places the images in the displays correctly aligned to the user. Using the offset parameters, the VR rendering software displaces the images compensating the physical offset.

Table 3.1 shows the calibration results for the 5 hardware setups: fields of view (FOVs) for the left and right displays and interdisplay horizontal and vertical offsets. The remaining two parameters, *interocular distance* and *display aspect ratio* are known in advance to be 65 mm and 16:9, respectively. Note that the offset values are relative to the screen size so, for instance, setup 5 has a horizontal interdisplay offset of 13% of screen width and a vertical offset of -5.4% of screen height.

Table 3.1: *Left and right display FOVs and interdisplay relative offsets in each hardware setup. The* Type *column indicates if that hardware uses a tunable lens (L) or display axial motion (D) for accommodation control*

Setup no.	**Type**	**FOV-L**	**FOV-R**	**offsetX**	**offsetY**
1	L	27.1	26.5	0.039	0.032
2	L	27.5	27.0	0.007	0.044
3	L	26.6	26.5	0.031	-0.006
4	D	26.8	27.1	0.053	-0.017
5	D	25.6	25.5	0.130	-0.054

In conventional HMDs with fixed accommodation these resulting parameters would be the end of the process. But one key aspect of this work and of our hardware is the ability to control accommodation. The following subsections explain how accommodation variation affects the stereo perspective calibration and how these parameters are treated in our experimental hardware.

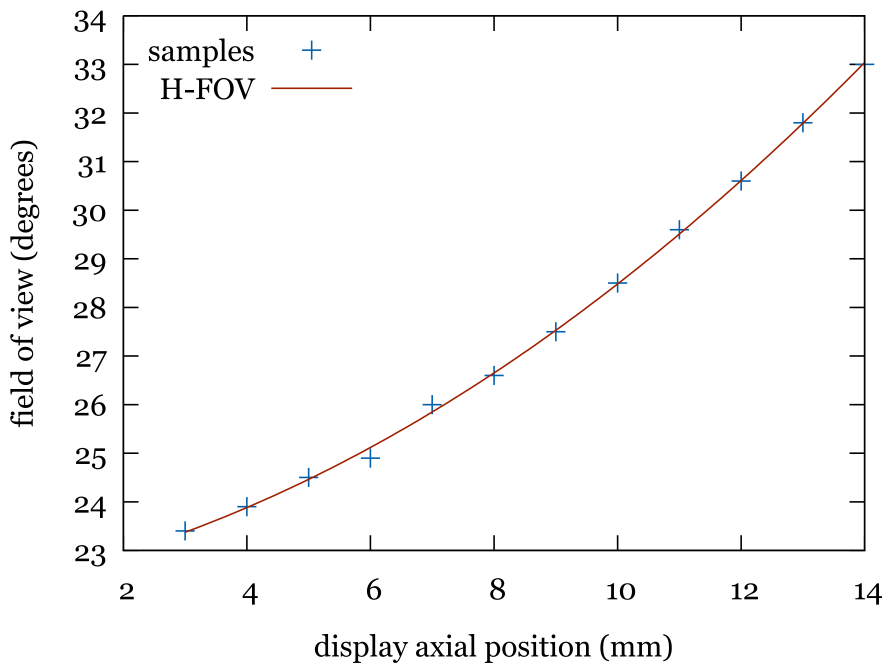


Figure 3.9: *Variation of horizontal field of view with display Z position (increasing Z means closer to the eye). The display was moved to different positions and FOV measured with the calibration camera.*

Variation of FOV with accommodation

As explained in Section 3.3.4 the view frustum, and most notably the field of view, varies with optical power. For example, Figure 3.9 shows the calibrated FOV when the display is placed at different positions along the Z axis at 1-mm steps on hardware setup 5 (based on display motion).

What the rendering system needs is the FOV as a function of optical power. Then, for each hardware setup, we repeated the calibration procedure with the hardware set at different optical powers obtaining different FOV values (The values in Table 3.1 correspond to zero optical power). From them we computed an approximating parametric function $FOV(p)$ for each case. Figure 3.10 shows a set of FOV samples at different optical powers and the corresponding fitted curve for one of the setups.

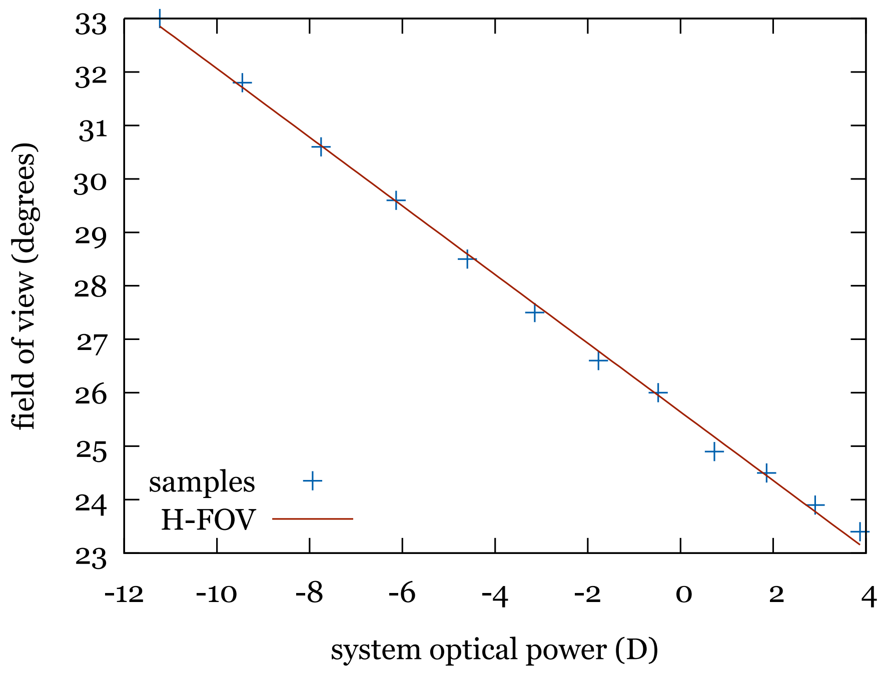


Figure 3.10: *Variation of horizontal field of view with optical power in display Z motion version.*

Variation of offsets with accommodation

Interdisplay offsets also vary with varying optical power. For this reason we also calibrated each hardware setup adjusted to different optical powers obtaining different horizontal and vertical offset values. Variations in offsets are rather small, with the worst case showing a variation of about 0.03 in vertical offset and 0.02 in horizontal offset when varying optical power 5 diopters (equivalent to moving an object from infinity to a distance of 20 cm). These variations with accommodation power can be approximated by linear functions. Figure 3.11 shows the horizontal and vertical relative offsets for one of the setups, along with approximating linear functions.

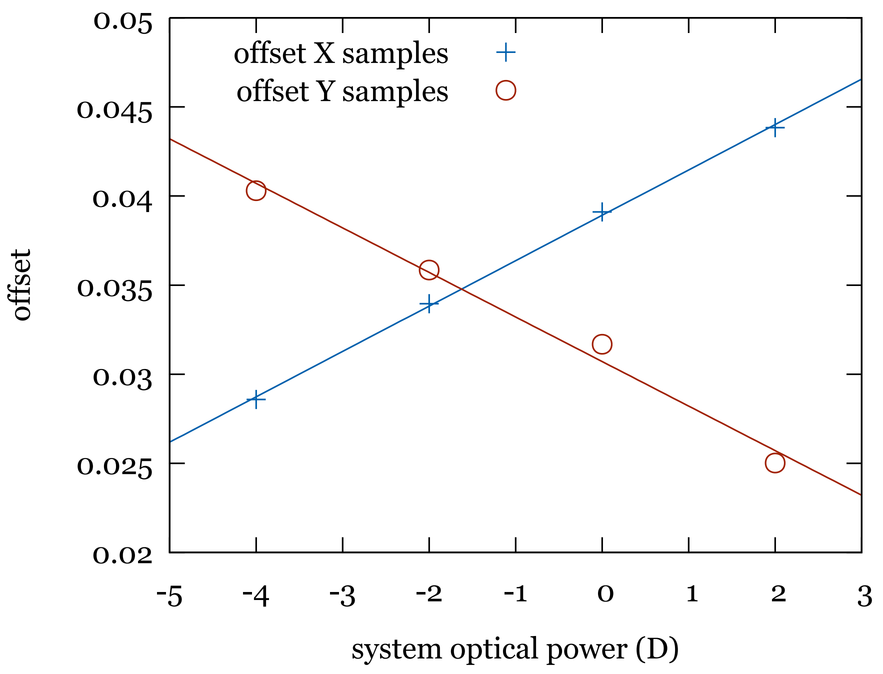


Figure 3.11: *Relative interdisplay horizontal and vertical offsets (*offsetX *and* offsetY*, respectively) at different optical powers and approximating functions for one hardware setup.*

These functions are later used by the rendering system to continuously adjust projection matrices based on these interdisplay offsets.

Robustness

Our system uses a calibrated stereo camera pair and sees both displays simultaneously, in contrast to Gilson's method which uses one uncalibrated camera and captures each display in sequence. This provides added robustness with respect to camera orientation. In Gilson's method, camera orientation errors are independent in each side so that relative offsets between the left and right views may produce unknown effects in stereo depth perception. In our method, camera orientation errors will be equal in both sides so that the relation between the views remains approximately constant.

In order to verify that our calibration is not significantly affected by small changes in position of the calibration device with respect to the HMD we performed the following test. We performed the calibration with the cameras rotated at a small varying angle around the vertical axis. And we compared the resulting calibrated offsets when that rotation is applied to the rigid camera pair and when it is applied only to the left camera.

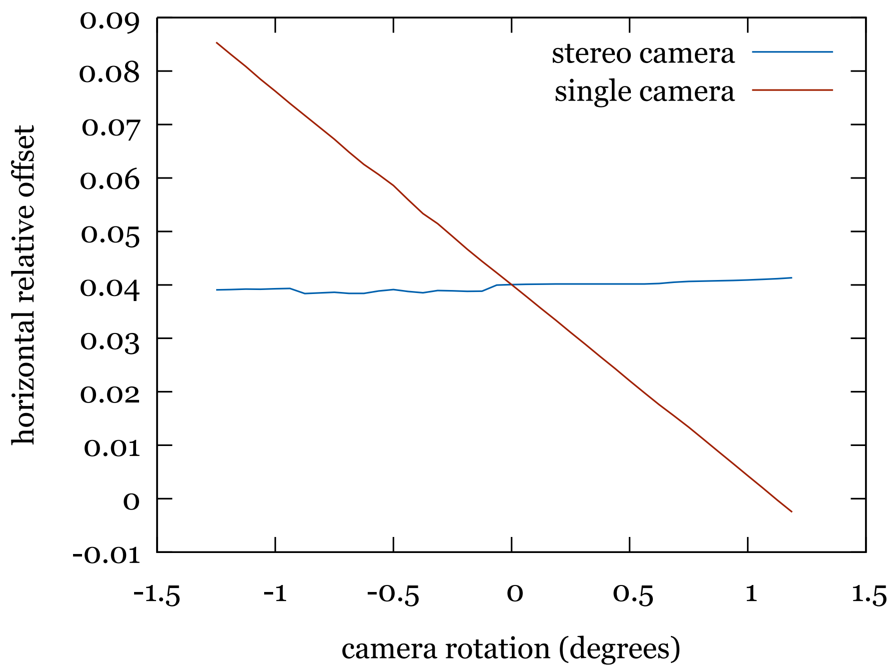


Figure 3.12: *Offsets calibrated by our stereo camera method and by a single camera method when the calibrating device changes orientation. The horizontal axis represents an added rotation around the vertical axis of the camera pair or only the left camera in each graph.*

Figure 3.12 shows the effect of camera orientation changes on the computed horizontal relative offset, which is expected to be the most affected. The graphs in the figure show the calibrated offset as a function of camera orientation, when the same orientation change is applied to both cameras (as would happen in our stereo camera method) and when it is applied to one of the cameras (simulating the effect of independent cameras, as in Gilson's method).

The graphs show that our method is robust with respect to camera orientation always producing approximately the same offset value. In contrast a method that independently measures each display is prone to producing a significantly varying offset value that will affect stereo perception in uncontrollable ways.

3.4.3 Focus calibration and control

Using the techniques from Section 3.3.1 we obtained the parametric functions for controlling optical power in both hardware setup versions (display motion and electrically-tunable lenses).

In the display motion-based hardware we sampled a set of points relating display position z to system optical power p using several lenses of known power. Figure 3.13 shows the set of (p, z) points sampled in one hardware setup and a parametric curve fitted to them. Note that in our experimental setup, the value of z position increases as the display moves closer to the eye. Some uncertainty regarding the position at which the image is the sharpest was expected. This is not a problem if a high enough number of samples are taken and used for curve fitting. The errors produced were almost always below one tenth of a diopter.

Although using third-degree polynomials seems adequate for most cases, display position control was better approximated by the rational function in Equation 3.14 that more closely models the optics. The main difference is when using the system outside the range of lens powers used in calibration. This rational function extrapolates better than polynomials.

In the hardware based on electro-optical lenses a set of samples were obtained with the same procedure. This time the samples relate input electrical current intensity I to system optical power p. Figure 3.14

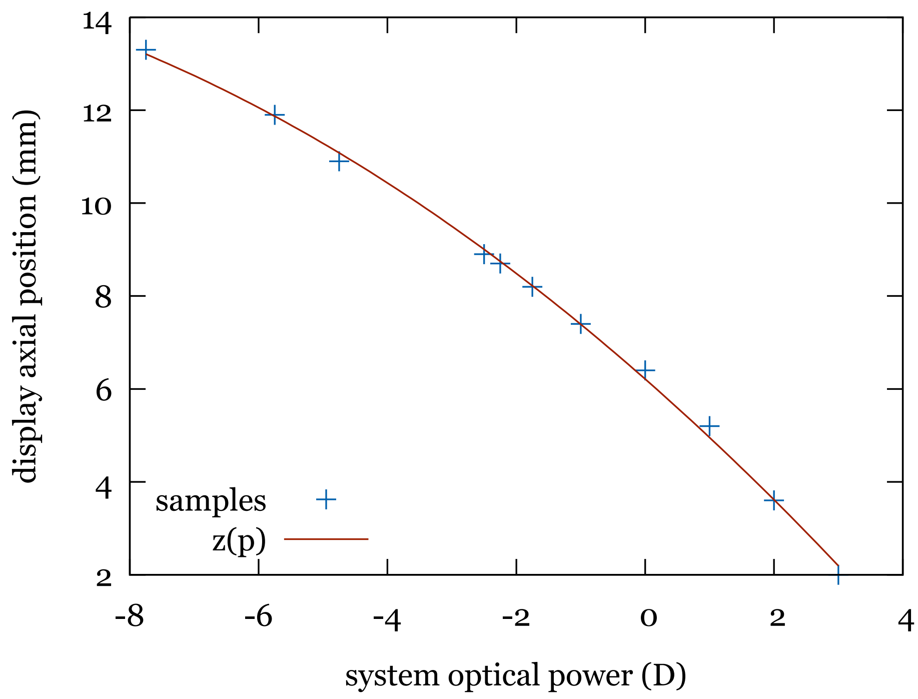


Figure 3.13: *Curve fit for optical power to display Z position.*

shows the (p, I) samples obtained in one hardware setup and a polynomial fit.

The virtual reality application loads all curves for the specific target hardware and uses them for control.

Control based on display Z motion is more repeatable although its response is slower. For example, changing focus power from zero to -8 D (equivalent to changing accommodation distance from infinity to 12.5 cm) requires the display to approach about 7 mm which the actuator will take around 0.3 seconds to perform. That is not really much slower than the eye's accommodation response but does not permit frequent sudden changes in accommodation demand. It should be noted that the relation between accommodation distance and accommodation power is not linear, so, if a virtual object approaches at constant speed, the system will have to move the display in increasingly larger steps in each

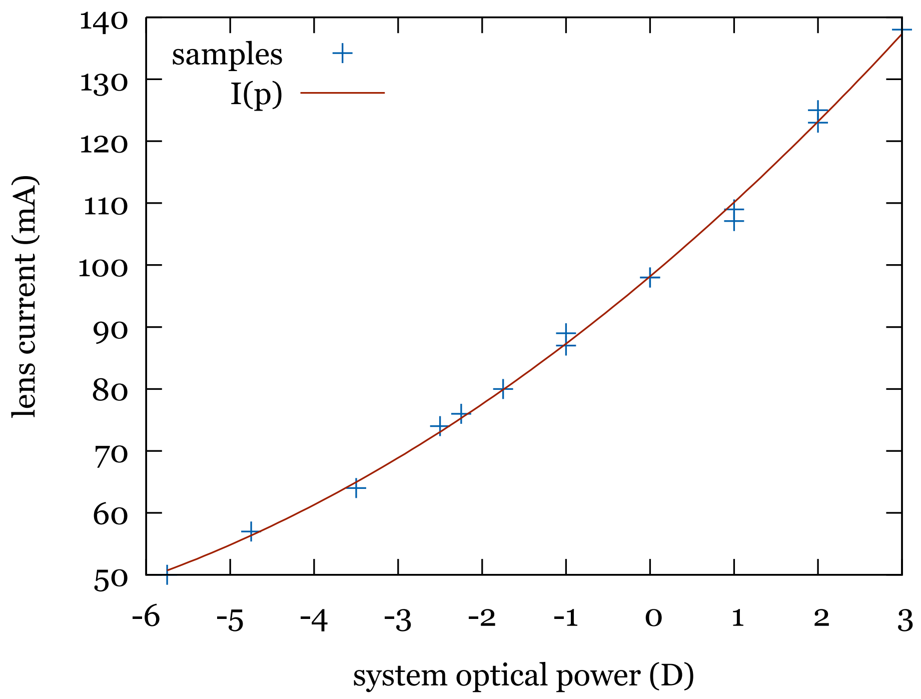


Figure 3.14: *Curve fit for optical power to current relation.*

frame.

Control based on electro-optical lenses is more silent and has an apparently faster response. The problem in this setup is repeatability. Due to unknown factors (possibly the effect of increasing temperature), the response curve of these tunable lenses may vary. Thus after some time in use there may be a difference of several tenths of a dioptre for the same current. Another drawback is related to the fact that existing tunable lenses are quite small and so the system needs a complex optical design to let all light pass through the lens without reducing observable field of view.

Obviously the system has a limited range of possible optical powers. This range depends on the physical construction, electro-optical lens power range or display motion range. Additionally, as system optical power includes a term for correcting the user's spherical ametropia, large corrections will lead to a reduced range for focus distances. For example, our motion platform can move from 0 to 16 mm. In Figure 3.13 we can see that that displacement corresponds to a range of about -10 D to +4 D. For a person with normal vision we can virtually approach an object to 10 cm (1/10 D = 0.1 m). But if the person has -6 D myopia, the nearest focus distance will be 25 cm as only -4 D can be used to control focus distance.

3.4.4 Subjective test

The stereo view parameters and control curves obtained through our calibration methods were applied to the rendering engine on our hardware and tested subjectively by viewers. The virtual scene displayed alternated between an object slowly moving back and forth from a distance of 15 cm to 2 meters away and a static object with sudden changes in distance between near (40 cm) and far (6 meters). The application continuously set the accommodation demand to match the distance to this moving object.

We tested the system with and without the interdisplay offset correction applied. When 3D images were displayed without offset correction users were unable to fuse and experienced uncomfortable double vision. When offset correction was applied, users reported mostly comfortable stereo viewing. This reveals that our microdisplays are in fact slightly misaligned, both horizontally and vertically, and that the offset correction works acceptably. However, sometimes virtual objects at very far distances were hard to see by some users which suggests there might be a slight divergence induced (visual axes not parallel but slightly diverging). Nevertheless, offset values obtained from calibration often needed small modifications until vision was correct.

Users with different refractive error (only myopia and hyperopia) tried the system after configuring their ametropia. They could comfortably see a sharp picture of objects at different virtual distances, both near and far. The relation between vergence and accommodation thus seemed to be correct. Finally, in order to check the effectiveness of accommodation distance control, we added intentional mismatches. First, the virtual object was placed 6 meters away but accommodation distance set to 40 cm. And then the opposite was performed, a near object with a far accommodation was shown. In both cases users were not able to see a fused and sharp object. This is in accordance with the study of So et al. [89]. These results show the benefit of variable accommodation hardware to achieve a wide range of virtual distances.

3.5 Conclusion

This chapter presents a practical approach to presenting stereo 3D images with correctly perceived size and depth on HMD-like displays in a way that reduces the vergence-accommodation conflict. The approach consists of a variable accommodation immersive display and methods to calibrate its projection parameters and its focus control. One central idea in this chapter is that accommodation variations imply variations in projection parameters. That is why accommodation control and stereo projection are intertwined and have to be handled together.

Experimental results show that the approach is acceptable and permits comfortable vision of static or moving virtual objects at a wide range of distances. Vergence and accommodation cues appear to match adequately. There was still some manual correction needed in the computed interdisplay offsets which may require further investigation.

Future work includes research to overcome the main limitations of the current approach, as well as to improve the quality of calibrations (see Section 6.3, Future work). First, the display calibration procedure could be automated by using computer vision algorithms to avoid manual user intervention and could be further modified to take into account nonlinear distortions. Finally, we would consider the use of an eye tracking system to sense the user's 3D gaze point and control accommodation based on the distance to that point.

3.6 Contributions

The research described in this chapter produced the following contributions:

- A calibration procedure for stereo head-mounted displays. It produces projection parameters including field-of-view (horizontal and vertical) and interdisplay offsets for correct stereo rendering.
- Tests of control mechanisms for variable focus in HMDs including moving displays and electrically-tunable lenses using stereo rendering with the calibrated parameters.
- A calibration procedure for the variable focus mechanisms that let optical power be set at runtime to the equivalent of desired focus distances.

These contributions were published in Segura et al. [84].

3.7 Publications

The research described in this chapter is supported by the following publication:

- Álvaro Segura, Javier Barandiaran, Aitor Moreno, Iñigo Barandiaran, Julián Flórez. *Improved virtual reality perception with calibrated stereo and variable focus for industrial use* (2017). International Journal on Interactive Design and Manufacturing.

Chapter 4

Immersive Visualization Scenarios

This chapter describes contributions in several practical use case scenarios where immersive Virtual Reality visualization systems were researched and developed. These contributions were developed in the context of applied research projects.

The following sections present different use cases. Each section has the same structure. First, the use case objective is explained. Then a discussion of alternative solutions is provided which includes references to state-of-the-art systems and techniques. Our proposed architecture is then described. And finally, a discussion of the outcome of the system is presented, including pros and cons, and the effectiveness of the solution.

Section 4.4 summarizes contributions of this chapter and Section 4.5 enumerates related publications.

The following use cases are presented:

- **Machinery simulators**: Training simulators of construction machinery.
- **Weather radar data visualization**: Visualization of weather radar data integrated in 3D topographic maps.

- **Welding simulation**: First person simulation of manual welding for basic skills training.

4.1 Machinery simulators use case

4.1.1 Objective

In the applied research project VAR-TRAINER, partially funded by the European Commission (EC-Contract no. COLL-CT-2003-500452), we sought to create an interactive simulator aimed at training in the safe use of construction machinery. Contrary to existing solutions focused on one single machine, the project intended to create a flexible platform for different types of machinery.

This research was initiated to develop an experimental framework with the following main requirements:

- A versatile system able to simulate different machines. Four machines were selected, including vehicles and non-vehicles:
  - Hydraulic excavator
  - Dumper
  - Mast climb platform
  - Work goods lift
- Use as many real physical elements as possible instead of fake gaming controls (i.e. actual excavator cab, joysticks, pedals and controls).
- Create and render virtual models to provide information to trainees.
- Simulation of motion to provide a realistic feeling.

4.1.2 Discussion of alternatives

First person simulators of vehicles typically use a virtual visualization type that suits their intended complexity and degree of immersion. The most used set-ups are the following:

- Projection screens. Either one large central screen or several screens, often three, for an extended field of view. In some cases the projection is stereoscopic and requires special filter glasses.
- Immersive head-mounted display. An HMD coupled with a suitable head tracking system can provide total 360-degree immersion.
- Standard monitor. Used in lower-end PC-based simulators, these provide very little immersion into the virtual simulation.

Additionally, some experimental web-based simulators of excavators had been presented such as Robert Lipman's VRML simulator [55]. Regarding simulation and visualization realism, this simulator was very simple. But given the primitive state of Web 3D technology at the time, it was quite complete. Lipman's VRML simulator allowed controlling several machines at a construction site, including an excavator, and featured a heightmap-based ground excavation simulation algorithm.

Existing simulators of construction machinery are typically built for specific machines (e.g. hydraulic excavator simulators, crane simulators, etc.). However, the VAR-TRAINER project aimed at creating a versatile system that could simulate different machines.

Simulators of excavators have been developed in the past. Freund et al. [29] describe a simulator of hydraulic excavators and construction machines based on *latest* (as of 2001) virtual reality technology. Other researchers have focused on the mathematical simulation models. For example, Borinara Park's PhD thesis [76] focuses on the digging process simulation and González et al. [35] focus on the 3D simulation of an excavator.

Ni et al. [72, 71] developed an excavator simulator based on a real commercial excavator cab mounted on a moving platform. In their setup, the five windows of the driving cab were replaced by five LCD screens of approximately matching size. This way they achieved a fully immersive view of the environment. We initially considered a similar setup. However, this solution was discarded as it would be too tied to one specific machine and driving cab geometry and it would not use any augmented reality.

A European Commission-funded project, SHE [81] ("Training Simulator of Hydraulic Excavator", Esprit LTI Project 28957) developed a simulator of a hydraulic excavator. This project used a motion platform on which a mock-up driving cab was mounted. For visualization this simulator could use either an immersive HMD or a projection screen. The HMD together with a magnetic 6-DOF head tracker provided full immersion into the virtual environment. However, there was no sight of the real excavator cab, pedals and joysticks. The projection screen, on the other hand, allowed the user to see the cab controls but it provided very little immersion as it was a single non-stereo screen placed a few metres in front of the platform.

As the existing above-mentioned visualization technologies did not meet our requirements we investigated the feasibility of a novel alternative. We decided to use a novel simulation solution: a system based on augmented reality on an HMD and chroma-key. Our idea solved the mentioned problems and is a more adequate and original way of merging real controls with virtual environments and machines.

4.1.3 Architecture

A simulation system was designed and developed for visualization in immersive simulators. It was based on immersive augmented reality: a video-see-through HMD with a chroma-key module. The idea is that the near environment elements (cab interior and controls) are real, while the environment around the vehicle is virtual. The HMD has two cameras placed in front of the user's eyes, looking forward, and capture what is in front of the user. An optical tracking system supported by an active IR-emitting marker on the user's head, estimates its position and orientation. The cab windows and all surfaces through which the outside could be seen are covered with a blue film. The visual effect of this setup can be seen in Figure 4.1.

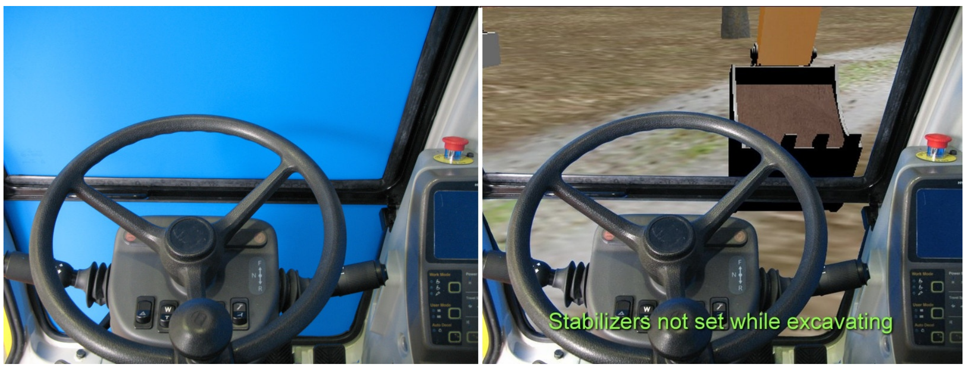


Figure 4.1: *Simulated illustration of the chroma-key concept. Left: a view of the excavator cab with blue covered windows. Right: the view after chroma-key substitution of blue areas with virtual environment*

The software architecture designed to achieve these effects is depicted in Figure 4.2. The *Simulator* module at the left contains the core physics simulation algorithms. The *Visualization* module at the centre takes input from the simulator, the cameras and tracking system, and sends output to the HMD.

The visualization module processes the image from the cameras and substitutes the blue areas with a virtual environment rendered in real time using the point of view provided by the head tracking system. Our mixed reality setup presents three layers of information to the user. These layers are, from back to front:

- The virtual construction site scenario and the virtual part of the vehicle being the virtual background of the scene.

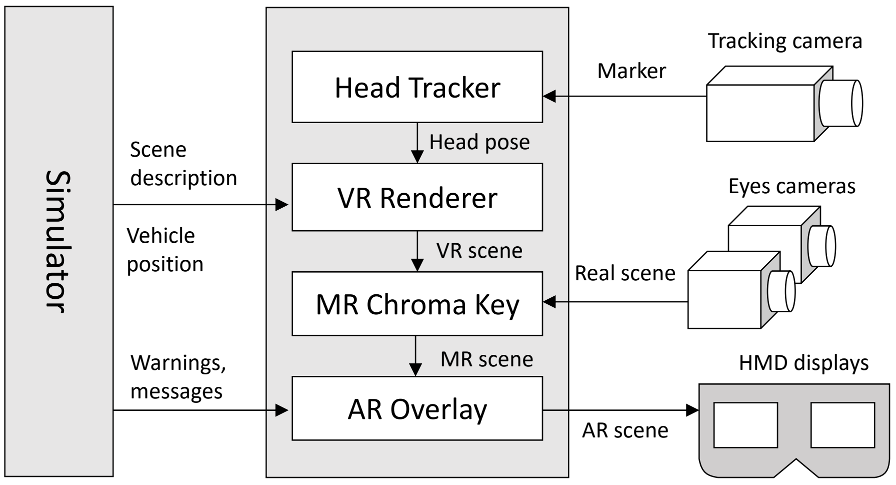


Figure 4.2: *Architecture for simulator visualization*

- The real parts of the cabin in front of the virtual scene, i.e. the real foreground.
- The augmented messages on top of the real foreground to display notifications or to provide hints.

Once we have a composed image for each eye, they are augmented with the additional text that will inform and help the trainee with messages and hints to properly carry out the assigned exercise. Special care must be taken in the integration of text (2D) in a stereoscopic visualization system, otherwise unpleasant depth perception effects show up.

In order to support the four types of machines selected, two interchangeable enclosures were prepared. For both vehicles (excavator and dumper) a real excavator driving cab had its windows covered with blue film. For the non-vehicle simulators, a large cabinet-like box was built, with all its internal walls covered with the same blue film, as seen in Figure 4.3. The driving cab and the blue box were mounted on a steward motion platform when in use.

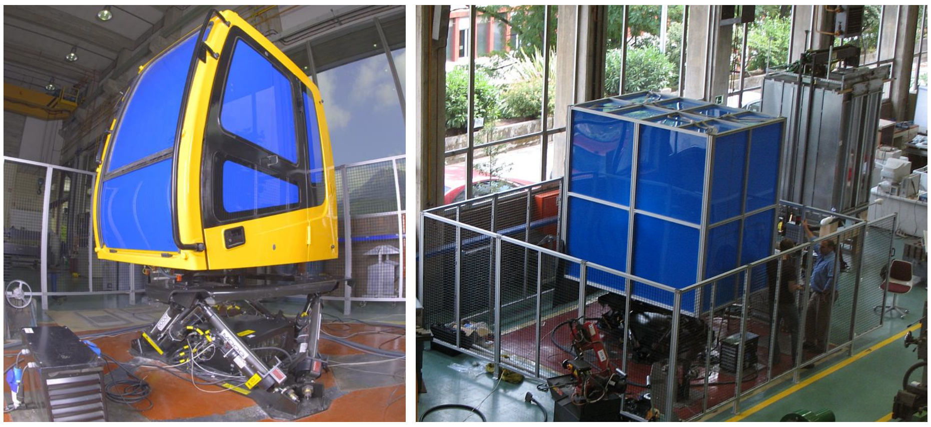


Figure 4.3: *Left: Excavator cab, Right: Blue box for non vehicle environments*

A special video see-trough HMD was developed for this simulator. The HMD can be seen in Figure 4.4. The device is based on a commercial *eMagin Z800* HMD on which a pair of digital *FireWire* cameras were mounted to provide the view of the real world. Additionally, an active marker featuring infrared LEDs was mounted at the top of the head straps to provide head tracking. Both the modified HMD and the tracking system were built by researchers from the Fraunhofer IGD institute.

The actual chroma-key process (the filtering of blue image areas) is performed in a GPU fragment shader, which also applies a simplified compensation of optical distortion to the camera images. Pixels from incoming camera images are substituted with the 3D rendered views of the virtual environment if their (r, g, b) components satisfy the condition in Equation 4.1:

$$\frac{b}{g} > k_{bg} \text{ and } \frac{b}{r} > k_{br} \text{ and } b > k_b \tag{4.1}$$

The conditions attempt to filter blue areas corresponding to the blue films regardless of illumination intensity. The equation depends on three parameters (k_{bg}, k_{br}, k_b) that have to be tuned to best filter

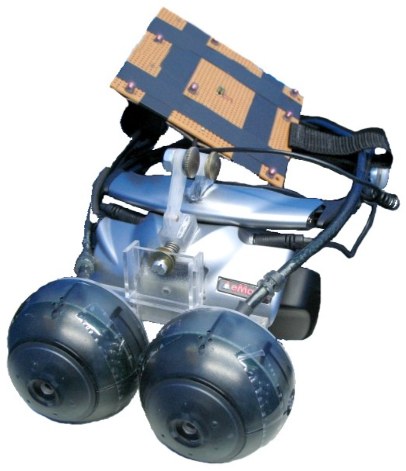


Figure 4.4: *Experimental see-through HMD developed in the project. Two cameras are mounted in front and an active marker for pose tracking is seen at the top.*

the blue areas. Two sets of parameter values were finally used, one for the excavator cab and another for the blue box, since the illumination characteristics were slightly different and a single set was not adequate.

As explained by Steinicke et al. [90], a correct geometry perception of a virtual scene requires the *display field of view* (DFOV, the angle subtended by the display in the viewing conditions), and the *geometric field of view* (GFOV, the FOV used to generate the perspective images shown) to have the same value. In this case, where two views are mixed both of their GFOV values have to match the display's DFOV. The displays in the eMagin Z800 HMD have a diagonal field of view of 40 degrees, so this FOV value is used in the 3D rendering engine for perspective projection. The mounted cameras have a higher field of view as well as some optical distortion. The images captured by the cameras are thus processed to compensate distortion and to reduce their field of view to match that of the virtual scene.

Additionally, an interactive editor for preparing the simulation environments was developed. This was an interactive graphical application where a user can design scenes, modify the shape of the ground using sculpting-like modifiers, add roads, add buildings under construction with configurable features, define special areas, place automated vehicles and characters, etc.

4.1.4 Results and discussion

The AR visualization architecture developed was unique and original. In conventional AR a real background is captured by a camera and virtual objects and superimposed over it. The opposite occurs in our case: the environment is virtual and the real elements are superimposed. In fact, a third layer is overlaid on top of the mixed view containing textual information with notifications and warnings to the user.

Professionals from the construction sector examined and tested the simulator. The system was shown to associations participating in the project specification (some of them working on risk prevention), and was then presented at the German BAUMA Construction Fair. The impressions were generally positive and a good potential for training was seen by professional visitors.

The degree of immersion is limited by several factors. The eMagin Z800 HMD displays have a field of view of 40 degrees and has some space under and around the them through which the user can partially see the real environment.

The quality of the chroma-key effect suffered from some issues which had an impact on realism and immersion. Examples of the actual resulting mixed reality images as presented to the HMD user can be seen in Figure 4.5. The images show a highly aliased or staircase-like boundary between the real and virtual parts (i.e. the camera image and the 3D rendered image). This is due to the cameras applying chroma subsampling to reduce communication bandwidth.

The mounted cameras use a 4:1:1 subsampling scheme meaning that every four pixels in a row share the same chrominance value. Therefore

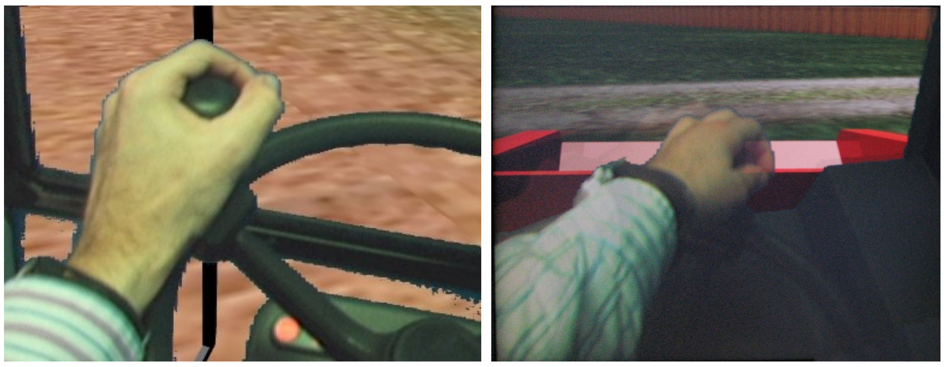


Figure 4.5: *Chroma-key result as presented to the user in the HMD displays. Left: driving an excavator, Right: driving a dumper*

the blue filtering shader effectively acts at a lower resolution than the pixel resolution of the cameras.

Additionally, under certain conditions the chroma-key effect suffered as a result of the non-uniformity of blue areas. This was mainly due to variations in illumination and especially to specular highlights. In the excavator cab the blue window were back lit by the external ambient light which provided a highly uniform color. In the blue box fluorescent lamps were installed in its ceiling to illuminate the machine's controls and fences. These lamps produced soft specular highlights in some areas, particularly when looking up. These specular highlights were not filtered by the chroma-key shader. They modify the original chroma color, greatly altering its hue and saturation. As a consequence, the filtering process did not behave properly.

Finally, the head tracking system is the element of the setup that made the testers more uncomfortable. When they did some very aggressive movements or moved out of tracked area, they noticed that the image got frozen, because the tracking system lost the user's point of view. After some training, the users learned where they can move their head, and this problem was reduced. In the platform setup, this effect is dramatically increased because the users are standing up and they tend to walk in all directions, causing frequent tracking failures. In this case, it is more difficult to teach the user to stay in the tracked area.

The tracking system has an inherent instability (even with the user at an optimal position) that should be limited to tolerable ranges. The inclusion of image stabilizer filters can improve the user experience, but in this case, no filter additions were needed, as the achieved tracking stability is enough.

4.2 Weather radar data visualization use case

4.2.1 Objective

As part of a research project on visualization of weather information, this work had the goal of creating a new interactive and immersive 3D viewer for weather radar volumes. Weather information is collected by discrete weather stations (containing a number of sensors) and weather radars. All this information is georeferenced as it belongs to specific points on a geographical region. The objective of this work was to put together the digital terrain model of the region (in the form of a *digital elevation model* or DEM) and the dynamic weather radar information collected by a radar and provide an immersive view with navigation based on natural interaction.

The immersive experience should be shared simultaneously by a group of technicians.

4.2.2 Background

A weather radar scans its surroundings in successive sweeps. Beginning at a low angular elevation, the radar performs a scan by rotating 360° around the vertical axis. At each direction a ray is emitted and a number of echoes are measured, each corresponding to a distance. This way atmospheric information about a conic surface is gathered. The radar then proceeds step by step performing scans at different elevation angles. All scanned cones together form the scanned volume of the radar (see Figure 4.6).

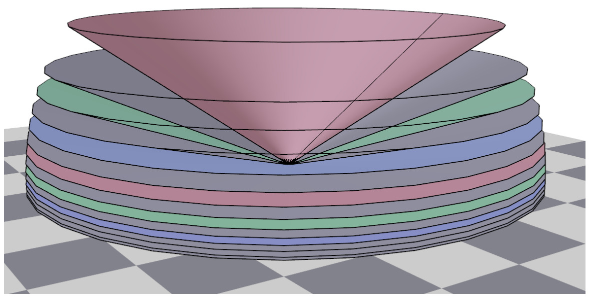


Figure 4.6: *Weather radar scan volume.*

Radar scans are usually displayed as 2D images. The most typical representations are the *PPI* (plan position indicator), that displays the data of one of those cones corresponding to a single elevation angle, and the *CAPPI* (constant altitude plan position indicator), which combines data from several cones and builds a virtual planar section at a given altitude.

4.2.3 Discussion of alternatives

Different visualization technologies were considered for this application. To achieve an immersive experience a stereoscopic display was needed using one of the techniques described in Section 2.1. The most immersive solution was based on head-mounted displays but that was not well suited to our research goal as using a HMD does not allow a group discussion. Among the stereo display technologies, projection is what can achieve the largest shared experience.

For interaction the alternatives were conventional wired input devices (mice and keyboards) or more advanced sensors enabling natural interaction. Magnetic tracking sensors and hand-held gaming controls were considered. Finally, interaction experiences were carried out using a commercial wireless input device.

CAVE is a projection technology that was considered. Compared to projection screens, it has the added benefit of providing a virtual environment that surrounds the user. However, it was discarded because a CAVE provides room for a smaller audience, the surrounding experience is not required, and its cost is much higher. Furthermore, a CAVE requires a huge laboratory volume that cannot be shared with other research systems.

4.2.4 Architecture

The system implemented in this work was based on rear projection on a translucent screen and gesture-based interaction using low cost gaming devices. Stereo images were provided by two projectors each projecting the image for one eye, and having linearly polarizing filters in front of their lenses. Users had to wear complementary polarized glasses to view the 3D scene.

A weather radar scans its surroundings by emitting a beam while turning along 360 degrees of azimuth and capturing the reflected signal. It performs this at several elevation angles in order to scan a volume. The geometry of the weather radar data is approximated by a set of concentric cones, centred at the radar location. These cones correspond to the different elevation angles scanned. The visualization is performed by applying a texture image with an opacity channel to each of those cones.

Raw radar scan data consists of matrices of reflectivity values, one per elevation. Our software applies a transfer function to convert these scalar values to RGBA pixel colors and obtain the final texture images for visualization.

The data used in this work came from the radar at Mount Kapildui, in the Basque Country, Spain. A digital model of the Basque Country

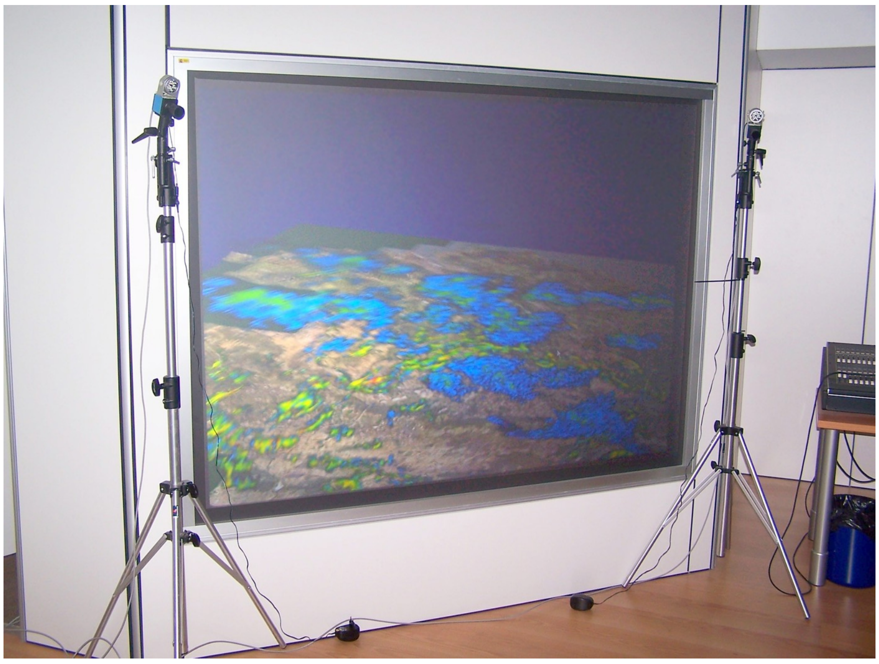


Figure 4.7: *Immersive stereo viewer of weather radar data (ambient light has to be dimmed for a better experience)*

was obtained from public data and stored in a hierarchical elevation grid using a PagedLOD node of the *OpenSceneGraph* rendering library [75].

The interaction model was based on the *Nintendo Wiimote* and *Nunchuk* remote control devices. It consisted of gestures assigned to different commands such as *play*, *stop*, *rewind*, *fast forward* and *restart*, apart from navigation in the 3D world.

Stereoscopic visualization required the rendering of the environment using two virtual cameras into two full-screen images sent to the two separate projectors. It has to be noted that perspective and depth perception depend on the virtual camera parameters and on the viewing conditions. That is, the same images projected on smaller screens or viewed from a different distance produce different perception of depth and size. As explained in Section 3.2 rendering projection parameters have to match the viewing conditions for a correct perception (this includes most notably the screen field of view).

In this case projection parameters were chosen for a standard viewer position in front of the screen at a distance of two meters. Stereo rendering parameters have to be chosen correctly so that disparities on the screen correspond to the expected disparities of objects at specific distances. Objects at very far distances should produce an on-screen disparity equal to the interocular distance. If we assume the screen to be significantly larger than the user's inter-eye distance and the observer to be centred in front of the screen, the following approximate model is valid. The horizontal field of view of a screen of width w as seen by a centred observer at a distance d is given by Equation 4.2.

$$\mathrm{fov}_H = 2 \arctan \frac{w}{2\,d} \tag{4.2}$$

The perspective projections used for rendering must have the computed field of view and the view for each eye must have a horinzontal offset corresponding to half the inter-eye distance.

The setup was similar to that of 3D cinema. However, a virtual reality experience was required, which 3D cinema does not provide, and this is why viewing conditions were considered. One limitation of this setup is that the view is only perspective-correct for a user at the predefined position. If the user moves, some distortion of the virtual space will appear. As shown in Figure 4.8, observers A and B have a different view frustum and FOV. As we use the theoretical frustum of observer A for rendering, observer B will have a slightly distorted perceived view.

To have a correct 3D perception from any view point, a tracking system was required to find the location of the user's head. We experimented with a magnetic 6-DoF tracker for this purpose. Once the user's position was known, the virtual camera was moved accordingly. In the original CAVE system (see Cruz-Neira et al. [16]), a more complex system was used that estimated the location of each of the user's eyes to

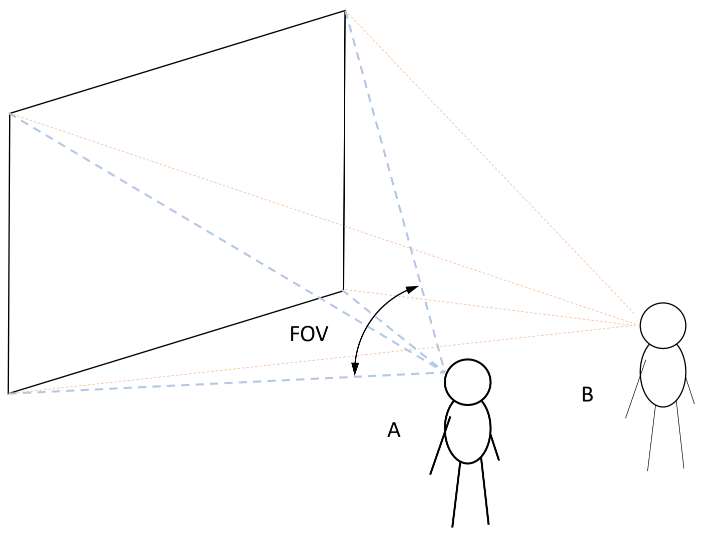


Figure 4.8: *Screen view frustum for different users*

compute a correct frustum for each wall. Since the possible positions of a user in our setup were more limited a simpler model was acceptable.

4.2.5 Results and discussion

The system described above provides a display solution for weather radar data integrated with terrain geometry. The stereo projection system is well suited to this domain for several reasons: the large screen size provides a good, comfortable view of the topography extension, and the stereoscopic effect enables improved perception of the volumetric nature of rain and cloud data. The projection system is not as immersive as the CAVE, but it is much more affordable.

The magnetic head tracking tested did slightly improve the freedom of motion of the user in front of the screen and produced a noticeable effect when moving sideways. However, it had drawbacks: there was a short delay in the sensor readings so that the reaction of the projected image to the user movements was a bit slow, the sensor range was limited to around one meter from the magnetic source, and there were wires for the sensor. The system was finally used without user tracking as the improvements were not sufficient with respect to the added limitations.

These 3D images deliver more insight about the atmospheric situation than the standard solutions described at the beginning of this section. A 3D render of the scanned volume transmits more complete information on the shape of the atmospheric phenomena, particularly in the vertical dimension. And stereo projection improves noticeably the perception of these volumetric data, helping to differentiate the different layers that otherwise appear overlapped.

By displaying together ground geometry and radar reflectivity data, some radar echoes are easily identified as *ground clutter*. Ground clutter is caused by radar beams being reflected by the surrounding ground (i.e. mountains) instead of by atmospheric content. Figure 4.9 shows the high radar signal around a mountain range. Our software added special filters to remove these fake echoes by subtracting a background radar range image taken on a clear day.

Using advanced volume rendering techniques was not a goal of this research. Instead of rendering the different scanned slices separately, a more advanced approach would be to apply volume ray casting (also known as ray marching) techniques as explained in Section 5.1. In fact, it must be noted that in reality radar beams do not travel in straight lines. Due to refraction related to variable atmospheric density, beams are slightly curved downwards. This effect is only significant at very long distances and has been ignored in this implementation.

Interaction control based on mouse and keyboard to interact with the application is unnatural in a large immersive setup like the one presented. The natural interaction system based on gesture recognition via a gaming control proved to be a good approach in this group immersive

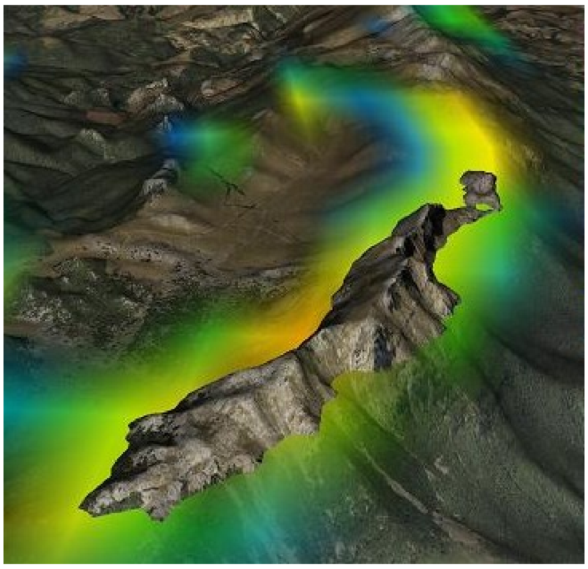


Figure 4.9: *Ground clutter*

view domain. Simple gestures allow the meeting leader to move around the virtual environment and to control time (i.e. display previous or next radar scans and see how atmospheric phenomena evolve).

4.3 Manual welding simulator use case

4.3.1 Objective

Manual welding workers require training using potentially hazardous equipment. It also requires spending great amount of resources including energy and materials. A vocational training centre that offered welding courses was particularly interested in testing virtual ways to train in shielded metal arc welding.

This type of welding is characterised by the use of consumable electrodes. The welder uses an electrode holder to grasp a shielded electrode initially about 30 cm long. As welding progresses, the electrode material is melted and deposited so that its length continuously decreases. The welding technique consists of first initiating the electric arc by quickly touching the workpiece with the tip of the electrode. Then, the electrode tip must be moved along the intended weld line on the workpiece maintaining a gap and at a relatively constant speed and orientation.

A simulator of the process should be able to train in the dexterity required to move the tool correctly maintaining a nearly constant gap. This means that as the electrode is consumed, its tip comes closer to the electrode holder and the welder has to compensate this by bringing the tool continuously closer to the workpiece.

The simulator must be as immersive as possible in order to let trainees feel as if they were performing actual welds. Manual welding is performed just by moving the welding tool (i.e. the electrode holder) in space, so the simulator must support natural input in the same way. Additionally, tactile feedback would be a desired feature. Finally, in order to have a chance to be used in practice in vocational centres, the solution should be relatively cost-effective.

4.3.2 Discussion of alternatives

Different technological approaches were considered, including virtual and augmented reality. Few welding simulators were found in the literature. The most advanced simulator found was *VRSim* [24, 79]. VRSim es a simulator of GMAW welding processes, not disposable electrode welding as was our target. The simulator is based on a mixed reality system comprising an immersive HMD and a 3-DoF haptic device. A real welding torch was attached to the haptic device which provided tactile feeling to the user (i.e. contact with the work piece). Head and hand tool position tracking was provided by a hybrid ultrasonic/inertial tracking system and tool orientation was also tracked by gimbals.

Haptic feedback is a positive added value but it adds too much complexity and cost. In our case with disposable electrodes, the tip of the electrode is not a t a fixed position on the tool. As the electrode is consumed, it becomes shorter, and its tip moves towards the hand-held tool (the electrode holder). This situation makes a design with touch feel especially difficult.

An alternative design was that of *CS-WAVE* [18], in which trainees performed simulated welds directly on a flat screen that acted as a metal work piece (see Figure 4.10). The hand tool is tracked by an ultrasonic 6-DoF tracking system. In this design there is no need for a head tracking, and tactile feedback is natural because the physical screen surface is the virtual work piece. Nevertheless, the immersion achieved is relatively low and the work pieces are limited to completely flat surfaces.

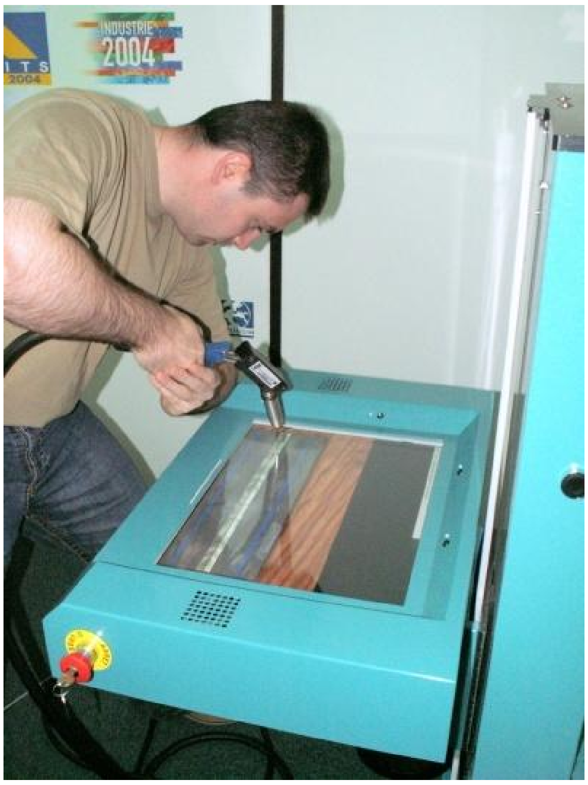


Figure 4.10: *The CS-WAVE training welding simulator.*

Another interesting system, *sMIG*, is presented in White et al. [102]. It is intended as a low cost simulator of MIG (Metal Inert Gas) welding for beginner students. Similarly to CS-WAVE this simulator uses a flat screen as a display device, but in this case it is a stereoscopic LCD monitor which provides depth perception. Head and tool tracking are provided by a commercial optical tracker.

4.3.3 Architecture

The diagram in Figure 4.11 depicts the architecture of the simulator visualization and interaction systems. A magnetic 6-DOF tracker with two sensors estimates the position and orientation of the user's head and the welding tool in space. The PC receives this information, runs the simulation and renders a 3D view from the point of view of the user.

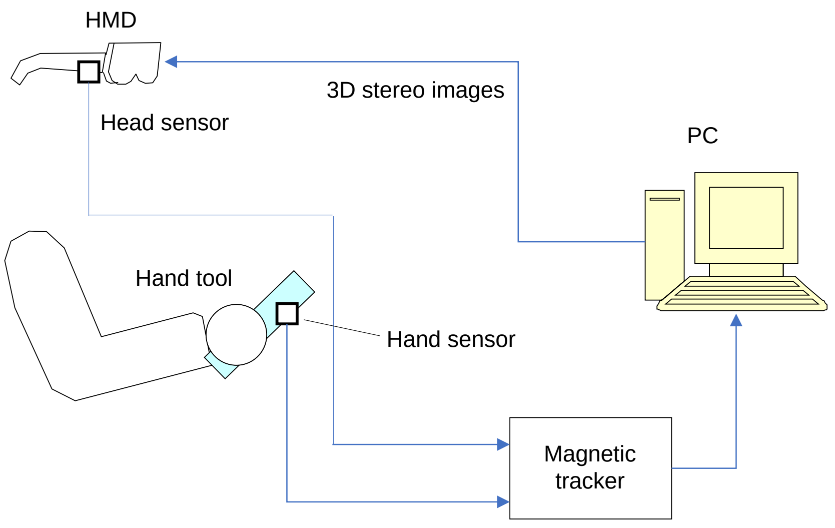


Figure 4.11: *Architecture diagram of our experimental welding simulator.*

The tracker provides pose information about its two sensors with respect to its magnetic field emitter. Two constant affine transformation

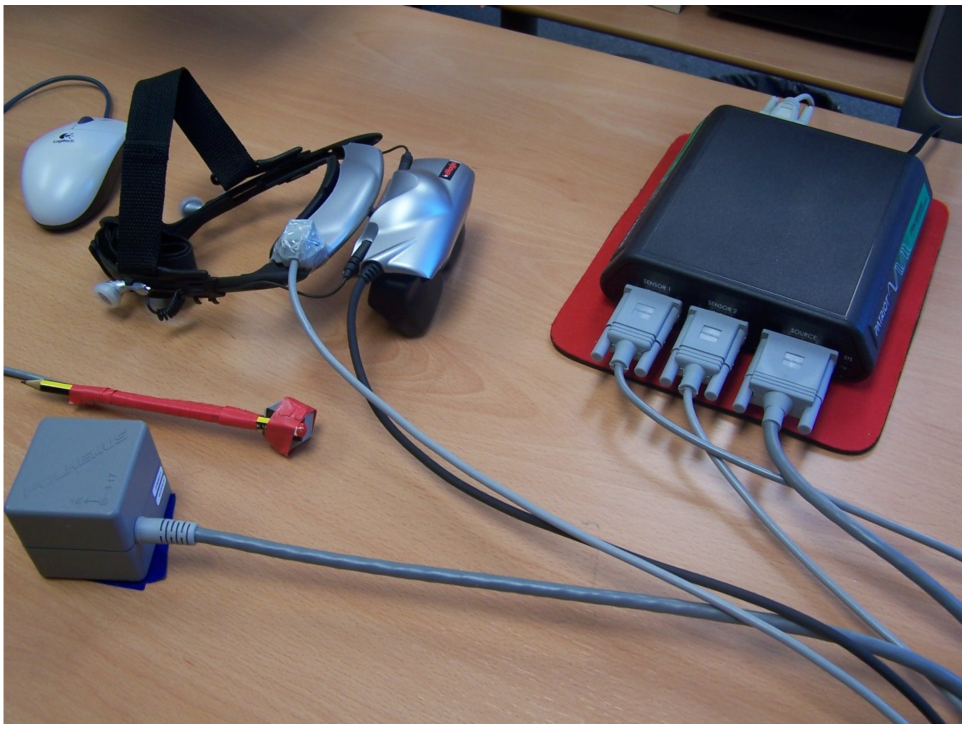


Figure 4.12: *Hardware for the experimental manual welding simulator. The Z800 HMD with a magnetic sensor attached, a hand-held stick with a magnetic sensor (simulating the welding electrode holder) and the magnetic field emitter and control box.*

matrices are applied to those poses to compute the tool and tool tip position, and the user's point of view. Figure 4.12 shows the hardware used in our experimental simulator, including dual-sensor magnetic tracker and head-mounted display.

In this application, having depth perception is particularly important. Virtual objects are at near distances and the user has to approach the tip of the virtual electrode to nearly touch the work pieces. The Z800 HMD uses one single input with a frame-sequential stereo signal. That is, even and odd frames alternate between the left and right images. The generation of this alternating signal was handled by the graphics hardware.

The 3D view rendered represented the user's hand, the welding tool (electrode holder and electrode), the work piece, and the weld, as seen in Figure 4.13. Additional effects included sparks (small drops of molten metal that are ejected and might get stuck to the piece) and the electric arc. The weld bead, as it progressed, was formed by stacked ellipsoids. A visual simulation of the cooling of hot metal was performed by altering the emissive component of the metal materials.

4.3.4 Results and discussion

This research showed that a simulator can benefit significantly from the use of immersive virtual reality. Manual welding requires particular dexterity and depth perception that is hard to simulate without an immersive stereoscopic display and without 6-DoF tracking of a hand tool. Magnetic tracking performed very well, providing position and orientation of the user's head and hand in near real-time. It has to be noted that such a technology is affected by magnetic metals in the environment which distort the magnetic field generated by the source. In the short distances involved in our lab setup the distortions were acceptably low. But we noticed significant distortions if the hardware was placed very close to metal structures (the same sensor was discarded in the machinery simulators described in Section 4.1 for this reason after some testing).

One key aspect that was important and particularly challenging was the welding simulation model. The physics involved is very complex and had to be simulated in real time, so approximations have to be made. Our simulator implemented quite simple simulation algorithms based on empirical observations to run in real time. Other simulators described in the literature employed more complex models based on neural networks or heat transfer simulation via finite differences that provided more realistic results. They were limited to a fixed set of simple work piece shapes (a flat plate or two plates at a right angle). Our solution permitted the use of arbitrary geometries.

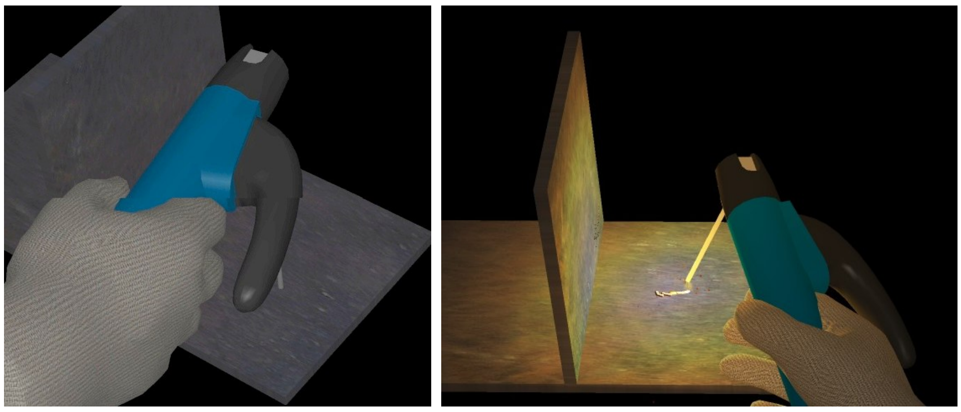


Figure 4.13: *Virtual images of the 3D view presented to the user of the welding simulator.*

We argue that immersive virtual reality based on HMDs is a good visualization solution for a simulator of this kind. Even if full 360-degree immersion is not essential (welders do not care what is behind them), the ability to approach the workpiece and to look at it from different angles is best provided by HMDs. What a simulator like ours lacks is a complete tactile feedback. *VRSim*'s simulator integrates a haptic device to provide touch feeling, at the expense of a much higher cost. We can argue that touch feeling is not essential because a correct weld with a shielded electrode is made without touching the piece (there has to be a gap for the electric arc). But touching, particularly among novices, is frequent. And in some conditions the electrode can become solidly stuck to the work piece. We skipped these effects to achieve a simpler a more affordable solution but this indeed reduces realism.

4.4 Contributions

We summarise in this section our contributions in the field of immersive visualization. We consider immersive visualization that which goes beyond a conventional computer screen and input devices and partially immerses the user in a virtual environment. Contributions present novel visualization systems applied in different application scenarios. We can conclude that immersive visualization requires different approaches in different applications.

Two types of visualization systems have been introduced and researched. Contributions include novel HMD-based Augmented Reality, and stereo projection based Virtual Reality.

Contributions in each of the application scenarios are described below.

One key contribution for immersive visualization was our mixed reality visualization system. As Section 4.1 explains we combine features of virtual and augmented reality in an innovative way.

This architecture uses available physical machine controls. This saves work because the machine controls are not modelled in the virtual environment. Even more, the user feels more comfortable with real controls and real cab environment. Additionally, the learning experience is safe, as all operations (driving, moving loads, climbing to heights, etc.) are virtual experiences.

The VAR-TRAINER project (see Section 4.1) proved the feasibility of implementing versatile simulation cores which can be adapted to different machines and run different hardware platforms. Four machines were integrated into the platform but more could be simulated on the same simulation and visualization core.

Section 4.1 shows that our immersive visualization architecture is easily adapted to different immersive hardware environments. The system supports this innovative mixed reality system as well as autostereoscopic screens (with RGB-Z input), dual back-projection with passive polarized glasses and immersive HMDs with magnetic head trackers.

These contributions were published in Segura et al. [86] and Segura et al. [85].

The interactive, immersive 3D viewer (see Section 4.2) provided a novel combination of volumetric weather data with a digital model of the terrain. This was published in Segura et al. [87], Moreno et al. [68], Goenetxea et al. [34]

Finally, a new immersive simulator of manual welding was introduced. It made a novel use of relatively low cost VR devices (head-mounted display and magnetic trackers) together with real time simulation and graphics to create an immersive interactive welding experience.

4.5 Publications

The research described in this chapter is supported by the following publications in international journals and conferences:

- Alvaro Segura, Aitor Moreno, Gino Brunetti, Thomas Henn, *Interaction and ergonomics issues in the development of a mixed reality construction machinery simulator for safety training* (2007). Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 4566 LNCS, pp. 290-299.

- Alvaro Segura, Aitor Moreno, Gino Brunetti, Thomas Henn, *Visualization for an augmented reality construction machinery simulator* (2007). 5th International Industrial Simulation Conference, Delft. 2007, ISC 2007, pp. 324-328.

- Jon Goenetxea, Aitor Moreno, Luis Unzueta, Andoni Galdós, Alvaro Segura, *Interactive and stereoscopic hybrid 3D viewer of radar data with gesture recognition* (2010). Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6076 LNAI (PART 1), pp. 213-220.

- Alvaro Segura, Aitor Moreno, Igor García, Naiara Aginako, Mikel Labayen, Jorge Posada, Jose Antonio Aranda, Rubén García

De Andoin, *Visual Processing of Geographic and Environmental Information in the Basque Country: Two Basque Case Studies* (2009). GeoSpatial Visual Analytics: Geographical Information Processing and Visual Analytics for Environmental Security, pp 199-207. Springer, Dordrecht, Netherlands.

- Aitor Moreno, Andoni Galdos, Andoni Mujika, Alvaro Segura, *Visual analytics of multi-sensor weather information: Georeferenciation of Doppler weather radar and weather stations* (2014). In IVAPP 2014 - Proceedings of the 5th International Conference on Information Visualization Theory and Applications.

Chapter 5

Ubiquitous Visualization Scenarios

This chapter addresses contributions in Virtual Reality visualization in ubiquitous systems. We consider ubiquitous visualization systems those that do not require the user to be at the same location where the complete application software and data are stored and running. This is similar to the concept used by Mc Lane et al. [65], where the authors wanted a remote access to graphics intensive applications across the Web and called it "ubiquitous interactive visualization". But our approach is different because they implemented server-side rendering and we intend to render on the client. These applications can be used anywhere without a specific software application at hand. This ubiquity is achieved using Web technology that can be accessed anytime, anywhere with a variety of desktop or mobile devices.

The technologies that enable real time 3D graphics rendering on the web are often collectively referred to as *Web3D*. The key technology that marked a turning point in the Web3D landscape was *WebGL* [64]. Introduced in 2011, WebGL is a JavaScript API that brings the functionality of the *OpenGL ES 2.0* API to the Web. As such, it permits writing software that runs in Web browsers to render real time 2D or

3D graphics on a programmable shader pipeline. The API is now implemented in all major browsers using the available graphics accelaration hardware. This is a step forward from earlier approaches that required installation of browser plug-ins and which usually did not support low level shaders.

The following sections present use cases of ubiquitous visualization. They are structured as in Chapter 4: First, the use case objective is presented, then a discussion of technical alternatives, our proposed architecture, and finally a discussion of the implementation and results obtained.

The following use cases are presented:

- **Weather radar visualization**: Volume rendering of 3D radar scans.
- **Energy-efficient building**: Interactive 3D buildings to support e-learning on energy efficiency.
- **Renewable energy learning**: 3D wind generators in renewable energy maintenance e-learning courses.
- **E-commerce**: First person environments as interfaces to e-commerce.

5.1 Weather radar visualization use case

5.1.1 Objective

This work on weather visualization involved the processing and rendering of volumetric data. The source data for this use case is a collection of volume data sets acquired by a Doppler weather radar. This data is the same used in Section 4.2. The objective of this work was to implement 3D volume rendering suitable for the radar data on the Web (desktop or mobile). One of the main challenges was the fact that the volume data is scanned in circles at several angular elevations and thus the data is not arranged as a uniform voxel 3D array (see previous Section 4.2).

5.1.2 Discussion of alternatives

There are different ways in which the radar data can be displayed. As explained in Section 4.2, this data is usually displayed in 2D representations such as the PPI and the CAPPI, which miss the three dimensional nature of the scans. One simple alternative is the one described in Section 4.2, where the data for each elevation is applied as a texture image to a cone with equivalent aperture. Although the general 3D shape of the scanned volume is approximately represented, this technique leaves gaps between the elevations.

An early attempt at displaying radar data together with data from georeferenced automated weather stations was developed using *Google Earth Plugin*. The textured cones were exported as files in COLLADA format. The viewer code loaded the cones corresponding to the selected date and placed them on the geographic location of the radar.

More advanced visualization involves using true volume rendering techniques. These techniques take as input a volume dataset consisting of an array of voxels and render images interpolating space between samples.

There are different techniques for rendering voxel fields. One way is to first extract one or more isosurfaces (surfaces at which the volume has a specific scalar value) and then render them with a conventional mesh rendering engine (see Westermann et al. [100]). Extracting an isosurface is most commonly done with the *marching-cubes* algorithm (Lorensen and Cline [57]). This technique is suitable for solid objects or segmented organs in medical images, for example. However, weather radar data does not have a clearly defined outer surface.

We tried this isosurface by marching cubes approach with a radar dataset. The first step required is to resample the space in a uniform cartesian array. The main issue in this step is the selection of a sampling density. Because the source data is in spherical coordinates, its density is highly variable: samples near the center are much closer together than samples in the farthest regions. Then, if a density is selected similar to that of the outer regions, the central region will lose detail. And viceversa, if a high density is selected to correctly sample the center, the outer regions will have redundant sampling and the dataset will occupy a huge amount of memory. The second issue is the selection of a scalar value for which to extract an isosurface. Figure 5.1 shows isosurfaces extracted for a value at the boundary of rain. The shape of rain falling from water masses in the atmosphere can be somewhat perceived.

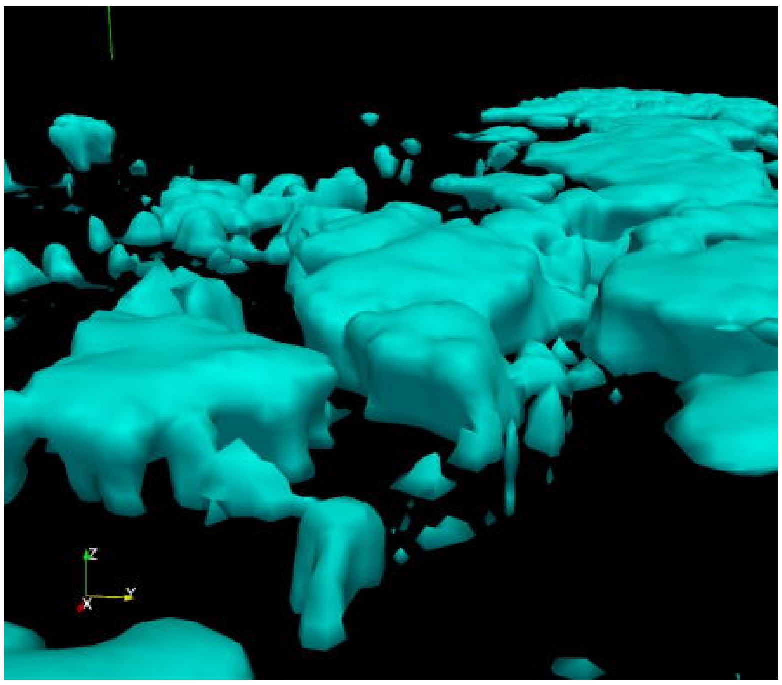


Figure 5.1: *Isosurfaces from a radar scan generated with the marching cubes algorithm after cartesian resampling*

Other rendering techniques for voxel data are based on the concept of *ray marching*, also known as *volume ray casting*. In a way similar to ray tracing, rays are shot from the camera to each pixel in the image. These rays are then advanced step by step to collect samples from the volume. The samples along the path of each ray are finally combined to compute a pixel color. The advance along the ray needs to be done in small steps if enough depth resolution is desired, so that a detailed enough representation is produced. This can be time consuming when image resolution is high. In some situations a variable step size can be used, starting with a large step and reducing it as the ray is traversed. A well-known optimization technique uses *distance fields* as explained by Zuiderveld et al. [104] or other precomputed structures like in Westermann and Sevenich [101] to traverse large steps where the volume is empty. In our case, where there is no clear distinction between solid and void, and where the volume space is not arranged in cartesian voxels, this is not practical.

Volume ray casting is computationally expensive but lends itself to parallelization. Use of GPUs for parallel computation in ray casting has been used in the past, as Marques et al. [63] describes.

5.1.3 Architecture

Our solution was a direct volume rendering based on the ray marching technique implemented on GLSL shaders. For each pixel in the image a ray is shot and traversed in fixed steps. At each ray position the volume is sampled and a color and opacity are computed by a transfer function. The final pixel color is thus a combination of all colors and opacities computed along the ray. When sampling the volume at an arbitrary point in space the value is trilinearly interpolated from its 8 surrounding samples. This makes the volume appear as a contiuous 3D field.

One key aspect of our method is that it perfoms the rendering al-

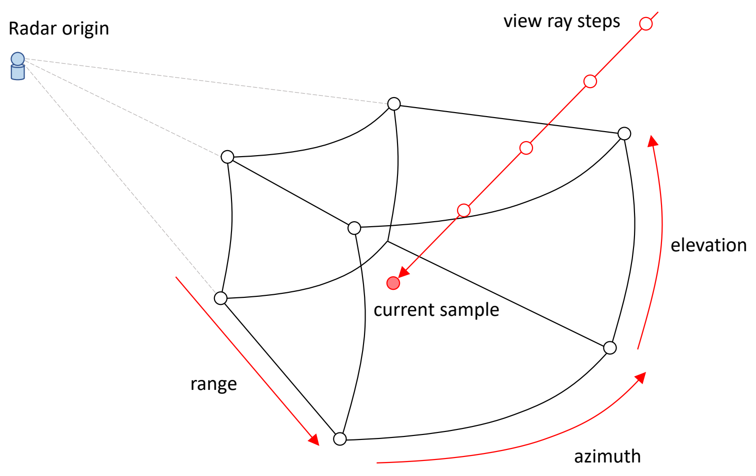


Figure 5.2: *Illustration of a voxel in spherical coordinates and the view ray step by step sampling*

gorithm directly in the polar scans obtained by the radar. To do so, the ray cartesian coordinates at each step are converted to spherical coordinates (Equations 5.1-5.3) and then interpolated from the elevations above and below them. Finding the surrounding azimuth angles is straighforward because azimuth resolution is uniform. On the other hand, as elevations are not uniform, finding the above and below slices needs iteration and branching.

$$r = \sqrt{x^2 + y^2 + z^2} \tag{5.1}$$

$$\varphi = \arctan(y, x) + \pi \tag{5.2}$$

$$\theta = \arccos(z/\varphi) \tag{5.3}$$

The volume data is stored as a single 2D grayscale texture image. The image contains the different elevation slices as tiles forming a texture atlas (See Figure 5.3). This is done because WebGL 1.0 does not support 3D textures. To interpolate the volume samples, a point in two slices has to be sampled. In each slice the (azimuth, range, slice index) triple has to be converted to a texel (u, v) coordinates coordinates.

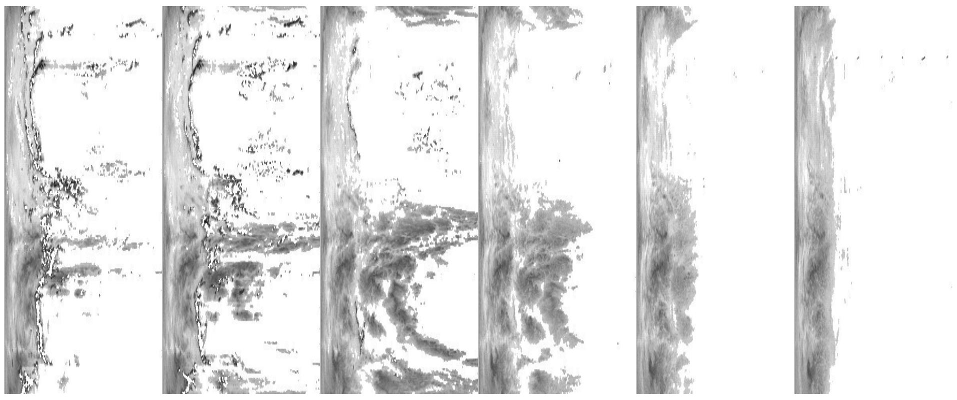


Figure 5.3: *Texture atlas for radar reflectivity data*

5.1.4 Results and discussion

The ray casting technique produces smooth images of volume scans. The software needs to specify a region of space to render as an axis-aligned bounding box. If the box is adapted to contain the whole radar scan, little detail can be seen. The region where a radar scan contains information lies relatively near the ground, but spans hundreds of kilometers horizontally. In order to have zooming and panning control that allows detailed viewing, we applied a scaling and translation to the radar data when sampling it in the fragment shader. Figure 5.4 shows a volume render of the central region around the radar. A conical hole seems to appear due to the absence of data above the highest elevation angle..

The performance of this technique depends mainly on two parameters: image resolution (i.e. number of pixels to render) and ray sampling step size. One limitation of WebGL 1.0 GLSL shaders is that loops

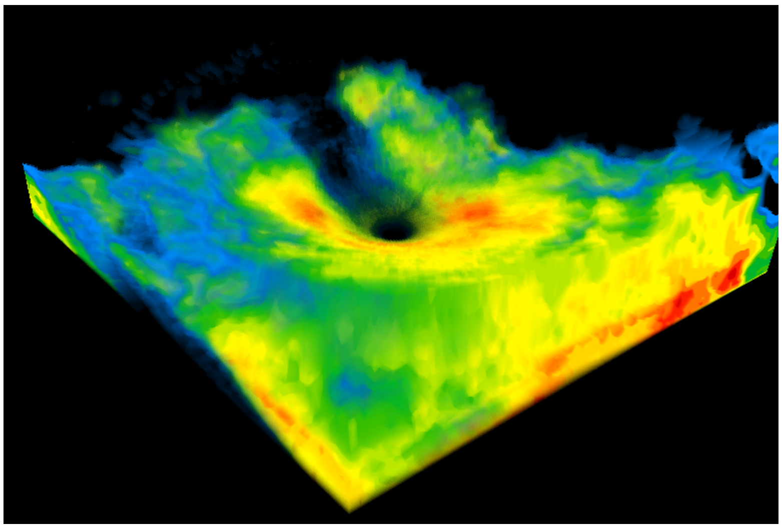


Figure 5.4: *Volume render of radar scan (central area)*

must have a compile-time constant number of iterations. Modifying the number of steps thus required recompiling the fragment shader. In some browsers this recompilation took a significant time (tens of seconds) due to optimizations applied by the ANGLE subsystem.

The proposed Web3D-oriented algorithm has been tested on networked computers. The computers do not need any additional specific software to visualize the weather data. Standard web browsers are used without any additional plug-in.

Mobile hardware has been tested as well. Their limited GPU computational power slows down the rendering process. However, images are generated, although at a lower rate. Special parameters (resolution and ray sampling step) have to be turned down to use the application on mobile devices effectively.

5.2 Energy-efficient building use case

5.2.1 Objective

The context of this use case was energy efficiency in buildings. Balaras et al. [4] consider buildings are responsible for over 40% of energy consumption. To address this issue, project EnEf (*510198-LLP-1-2010-1-IT-LEONARDO-LMP*) was set up aiming at creating online accessible learning content to create awareness on the subject of energy efficiency and to transmit basic knowledge. The idea was to design and publish conventional Web-based e-learning lessons and to complement them with 3D models and simulations illustrating the presented concepts.

The e-learning course covered topics of energy losses in buildings, building isolation and energy generation.

The main part of the platform is a quite conventional e-learning course. The contents were developed by experts in the field and integrated in the ILIAS Learning Management System. ILIAS is a SCORM-compliant LMS. The lessons are structured in the following seven modules:

- Concepts: general concepts of energy, heat transfer and building
- Legal Framework: both National and European regulations
- Marketing, including economic aspects and building energy certification
- Four modules for the execution of refurbishment
  - Facade Insulation: thermal insulation of external walls
  - Glazing: window designs and materials
  - Installations: energy consuming installations such as heating, air conditioning, water, and renewable energy sources for buildings such as photovoltaic solar panels, biomass

- Flat roofs: insulation of roofs

5.2.2 Discussion of alternatives

Since its inception e-learning has taken advantage of advances in ICT technology to assist in learning in many areas, including higher education and lifelong learning. E-learning usually consists of learning materials (textual, graphical and multimedia) and self-assessment tests in specialised Web sites. There have been examples of e-learning systems on energy-related topics such as Energy-Wise (Desai [19]).

E-learning courses sometimes contain multimedia elements (i.e. embedded videos or animations) that help understand some technical topics. But the use of embedded real time 3D simulations is uncommon. Conventional Web technology that forms the basis of e-learning platforms was not designed to support complex real time graphics and computations and the use of Web3D standards in learning content has been scarce. In Álvaro Segura et al. [59] we can see an example of integration of interactive 3D models in SCORM-compliant e-learning content, in that case about wind generators.

On the other hand, building simulation software has been available for decades. A comparison of 20 recent packages can be found in Crawley et al. [14] and Crawley et al. [15] describes one of them in depth. The tools described in that study are complex systems aimed at engineers designing new buildings and many include a geometry modelling component in addition to the simulations for different aspects: heat transfer, ventilation, HVAC, lighting, photovoltaic generation, etc. Additionally, some administrations offer software designed to calculate compliance of new buildings to national regulations on energy demand. For example, Spain provides the LIDER [67] software for this purpose.

The system described in this work is not an engineering tool; it is a simplified interactive simulation of a specific building for illustrative purposes accompanying an e-learning course. It is an attempt at joining e-learning, simulation and virtual reality on the Web. In fact, we are witnessing how more types of applications are moving from the desktop PC to the Web as noticed in Taivalsaari et al. [92]. In a comparable approach Google presented in 2010 Body Browser, an interactive visualization of the human body for educational purposes. The main difference with our approach is that it only visualizes an inert body whereas our approach tries to present a living building.

It was not the aim of the project to create full-fledged energy and heat transfer simulation systems.

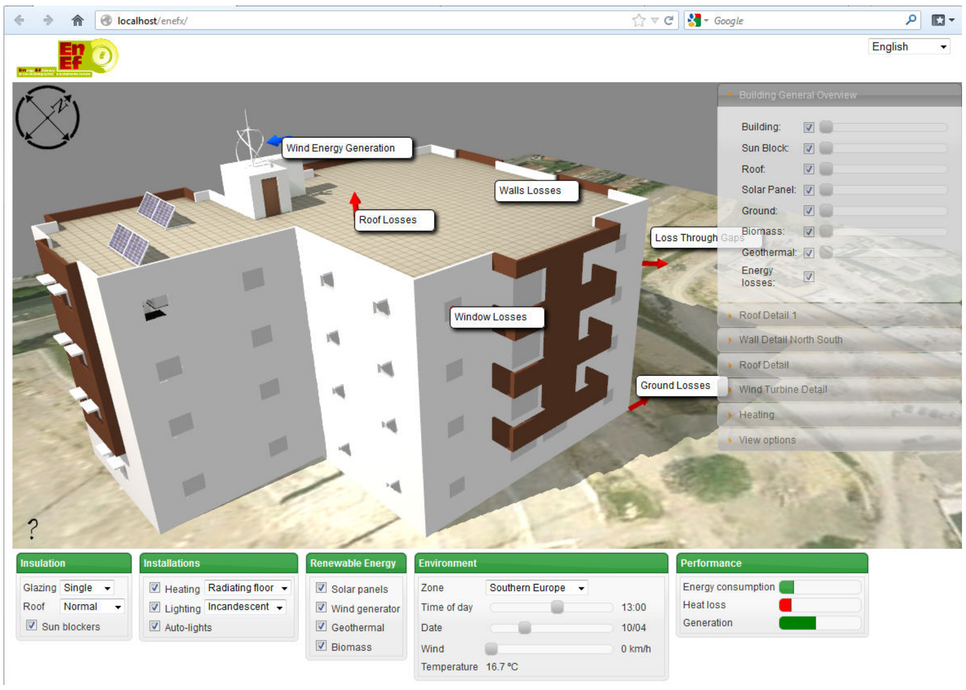
5.2.3 Architecture

The VR application designed and implemented was based on WebGL and used the GLGE library [9]. The system consisted of a computational model, a visualization system and a user interface. Figure 5.5 shows the main view.

Visualization

In our system, a custom XML file contains a definition of a scene and several *subscenes*, each specifying a set of visible objects, a set of labels and a camera position. The objects were stored separately as COLLADA files [48]. By hiding occluding parts these subscenes highlight different parts of the building such as a general external view, a detail of a room in a corner with sectioned walls, the roof and the basement. Labels identify relevant parts in each scene and are located in 3D space with a line pointing at the associated point. Figure 5.6 shows some of these subscenes depicting the isolation layers of different walls. When activating these subscenes, the rest of the building was hidden and the camera approached them.

By dragging the mouse on the view, the user can freely rotate the point of view. The mouse wheel moves the camera forward or backward



Insulation
Glazing: Single
Roof: Normal
☑ Sun blockers

Installations
☑ Heating: Radiating floor
☑ Lighting: Incandescent
☑ Auto-lights

Renewable Energy
☑ Solar panels
☑ Wind generator
☑ Geothermal
☑ Biomass

Environment
Zone: Southern Europe
Time of day: 13:00
Date: 10/04
Wind: 0 km/h
Temperature: 16.7 °C

Performance
Energy consumption
Heat loss
Generation

Figure 5.5: *Interactive 3D building model*

in the direction of view. And when clicking on any of the subscenes in the list, the camera starts a transition from the current position to the selected subscene's point of view.

The complete scene represents an urban building with a flat roof. Sun light is important in this application and so the current Sun position is taken into account and projected shadows are rendered accordingly. Emphasis was put into calculating the correct position for the time and date, and to make the light source colour be affected by its elevation in order to mimic real Sun light.

Interaction

The screen is divided in three parts with different interaction mechanisms: i) the virtual 3D view, ii) the views/scenes menu, and iii) the control panel at the bottom. The first two affect the visualization:

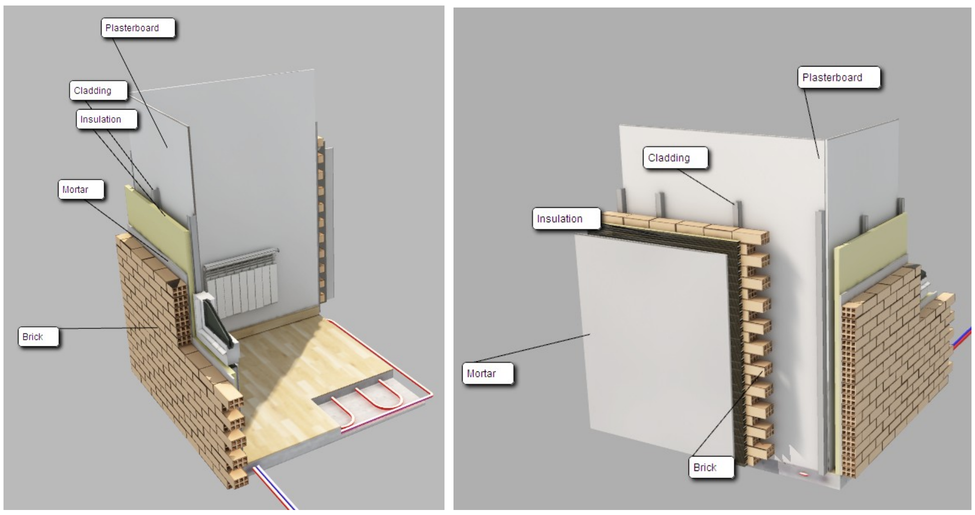


Figure 5.6: *Wall isolation layers*

rotation and movement of the point of view with the mouse, selection of different scenes and transparency of the main objects.

The last part, the control panel at the bottom contains a user interface that affects the simulation model. The panel has the following sections with interaction controls:

- Environment: sets external conditions for the simulation including geographical location, time of day, date of year and wind speed.
- Insulation: allows selection of roof insulation type, window glazing and window sun blockers.
- Installations: controls heating and lighting, both their type and their activation
- Renewable energy: adds renewable energy sources to the building including photovoltaic solar panels, a wind generator, biomass and geothermal energy.

Computational model

The tool presented is an interactive visualization that allows users to explore the building and see energy related elements in place. But as a simulation tool the building has some behaviour that is governed by a computational model. This model takes as input the variables controlled by the interaction controls listed above and calculates an internal state. Any change in any variable from the user interface triggers an immediate calculation of its effect. The calculations need to be simple enough to keep the system responsive and animation smooth. The state variables calculated and represented are energy consumption, heat losses and renewable energy generation, shown in the graphical bars at the bottom right of the UI.

The environment controls are used to calculate the current Sun position. The user can select among four different locations in Europe. For each one, the tool knows its geographical coordinates and the average monthly temperatures. From the current time of day, day of year and geographical coordinates, the software calculates the Sun position in the local horizontal coordinate system (azimuth and elevation).

If there is a wind mill present, the model calculates its power generation based on the current wind speed and air density. We assume a 5 kW generator, a typical power for vertical axis urban wind turbines. The turbine has a cut-off speed of 20 m/s, so if the wind is stronger than that the rotor will stop and no power will be generated.

If solar panels are present, the model calculates energy generation based on the current Sun position with respect to the panel (cosine of the angle between the panel normal and the Sun direction). The available panel is currently fixed in a South orientation and 45 deg elevation giving a maximum power at noon. For biomass energy a fixed contribution is added if present. Geothermal, on the other hand, is considered a source of heating and is only taken as a reduction in power consumption when heating is enabled.

Installations such as heating and lighting add to the power consumption when enabled. The user can select between incandescent and low power illumination which have different power contributions.

Finally, heat losses are calculated from the selected insulations (roof types, glazing type), the external air temperature and wind speed. A rough approximation is used to account for the different factors based on the convection laws (proportional to temperature difference and to wind speed).

5.2.4 Results and discussion

The 3D virtual model of a building described above was included as a complement to an e-learning course published online using the *ILIAS* learning management system. The course offered theoretical contents about energy efficiency in the building sector, including sections on thermal insulation and losses, surface materials for walls and roofs and energy generation installations for buildings. The simulation implemented a added an enhanced illustration to the theoretical contents. Users could see in virtual reality a building in which they could interactively experiment with different options related to energy consumption, loss and generation, and they could explore the building and observe the effects of changes in external conditions. This cannot be done with traditional illustrations or even animated videos.

The technologies used, WebGL and HTML5, were effective and proved to be a good choice. Nevertheless, WebGL was still relatively new and required specific browsers and recent graphics hardware. We did encounter a few users having problems to access the system due to old hardware or outdated graphics drivers. This is no longer an issue today.

5.3 Renewable energy learning use case

5.3.1 Objective

This use case also applies VR to enhance e-learning courses. In this case, the context is renewable energies and wind turbines in particular. In many technical areas, interactive simulation and Virtual Reality can provide a much more effective learning experience than just text and images, as pointed out by Huk and Floto [40].

An e-learning course was developed to support the initial stages of learning about wind generators, their parts, their operation and maintenance tasks. To support learning VR models were designed to complement textual and graphical explanations. Two kinds of 3D content were selected: non-interactive animated sequences that can convey the general structure of a wind generator and its main parts, and interactive 3D models where a user can modify working conditions and see the effects in the behavior of the generator.

This research provides a reference that highlights the advances achieved in more recent use cases described in this chapter.

5.3.2 Discussion of alternatives

At the time this work was carried out, WebGL had not yet appeared and there was no standard way to embed real time 3D graphics in Web applications that browsers could display by themselves. The available standard data formats for specifying 3D scenes for the Web were the Web3D Consortium standards *VRML* (Virtual Reality Modeling Language, [97]) and *X3D* (eXtensible 3D [98]). But Web browsers did not have built-in support for them or any other way to render 3D graphics using hardware acceleration. Loading and displaying 3D models based on any of those standards was done via installable third party browser plug-ins or extensions such as Java applets. In addition to standard formats for 3D content, there were proprietary formats or imperative APIs for creating scenes without an associated declarative format.

Table 5.1 lists browser extensions available at the time which enabled embedding interactive 3D graphics in Web pages. The “Format” column indicates what type(s) of content the viewer could load and display. The “Platform” column indicates the supported browser(s) in the case of plug-ins or the extension type (such as Java applet or Flash) otherwise. The “Scripting” columns indicate whether the viewer supported scripting to define behaviour in the 3D scenes. Scripting can be *internal*, where the model includes scripts that define animations or responses to user actions within the 3D environment (such as clicking on 3D objects), while *external* means that scripts in the Web page containing the model can communicate with the embedded 3D scene and control it. The label “VRML (+)” for the *Cortona 3D* viewer means that this viewer supports VRML with extended features.

Table 5.1: *3D viewers Web browser plug-ins and their supported 3D formats and scripting support*

Viewer	**Format**	**Platform**	**Scripting (internal)**	**Scripting (external)**
Cortona 3D	VRML(+)	Firefox, IE	Yes	Yes
Flux/Vivaty	VRML, X3D	Firefox, IE	Yes	IE only
BS Contact	VRML, X3D	Firefox, IE	Yes	?
Blaze 3D	Flash 3D	Java	ActionScript	-
Kaon	Proprietary	Java	?	-
Anfy3D	-	Java	Java API	Yes
Life 3D	COLLADA	Firefox, IE	-	-
Adobe Reader	U3D in PDF	Firefox, IE	JavaScript	-
Papervision 3D	COLLADA	Flash	ActionScript	-

Both VRML and X3D are standard specifications that allow the definition of 3D models and scenes together with their behaviour. They base scenes on a hierarchy of nested *nodes* (such as transform groups, primitive shapes, polygon meshes, textures, materials, etc.) with properties called *fields*. X3D was presented as a successor to VRML adding extended functionality in the form of new node types. In addition to the original VRML language syntax, X3D supports a new encoding based on XML [96].

Adding basic behaviour and interaction to VRML or X3D models can be done declaratively in the by associating sensor nodes (representing user input or clock time) to interpolation nodes and those to properties of scene nodes. This enables simple animations to play, possibly triggered by user actions. But more complex behaviour such as that needed in physical simulations requires scripting.

5.3.3 Architecture

The architecture finally included non-interactive 3D animations and an interactive 3D model of a wind turbine. Both kinds were based on the same 3D model of a virtual wind generator. The non-interactive animation used offline rendered animations with a predefined camera path. They were embedded in the relevant lessons as digital videos. For the interactive 3D model, the VRML specification was used and interactive environments were embedded in course content using the *Cortona 3D* viewer. Both, the interactive and non-interactive parts were based on the same 3D model of a wind generator modeled in *3ds Max* as depicted in Figure 5.7. A *texture baking* technique was used to apply illumination and ambient occlusion to the model in a way that would be visible in the interactive simulation.

The simulated wind turbine model was fully contained in VRML files. Even the user interface controls were built with 3D objects in the 3D world and simulation behaviour was written in embedded script nodes. Figure 5.8 shows a screenshot of the virtual simulated windturbine as it appeared in the online course.

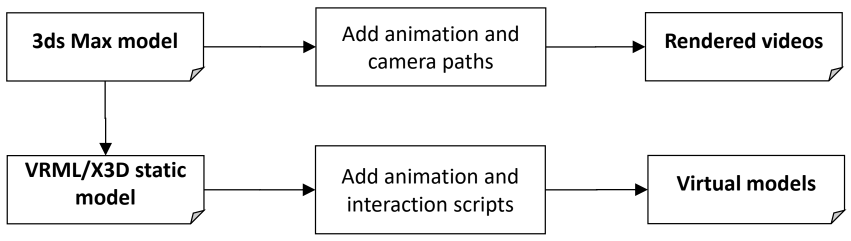

3ds Max model
Add animation and camera paths
Rendered videos
VRML/X3D static model
Add animation and interaction scripts
Virtual models

Figure 5.7: *Development process*

The panel at the left bottom of the figure contains a minimial user interface that was built with 3D shapes. It included interactive controls to change the wind direction and speed, and to change the blades pitch using sliders. An additional slider allowed varying the opacity of the nacelle to make its internal parts visible.

5.3.4 Results and discussion

Two techniques were used in this project to embed virtual content in e-learning modules: interactive real time graphics and non-interactive 3D rendered videos. It can be argued that all content could have been made using the same real time graphics technology. However, given the technology available at the time, offline rendering into videos offered much better animation and illumination capabilities and was finally selected for some scenes. Using current Web3D technology all content would be implemented as interactive real time graphics.

5.4 E-commerce use case

5.4.1 Objective

This work aimed at investigating the use of ubiquitous 3D graphics as an interface to e-commerce applications. Over the years, e-commerce

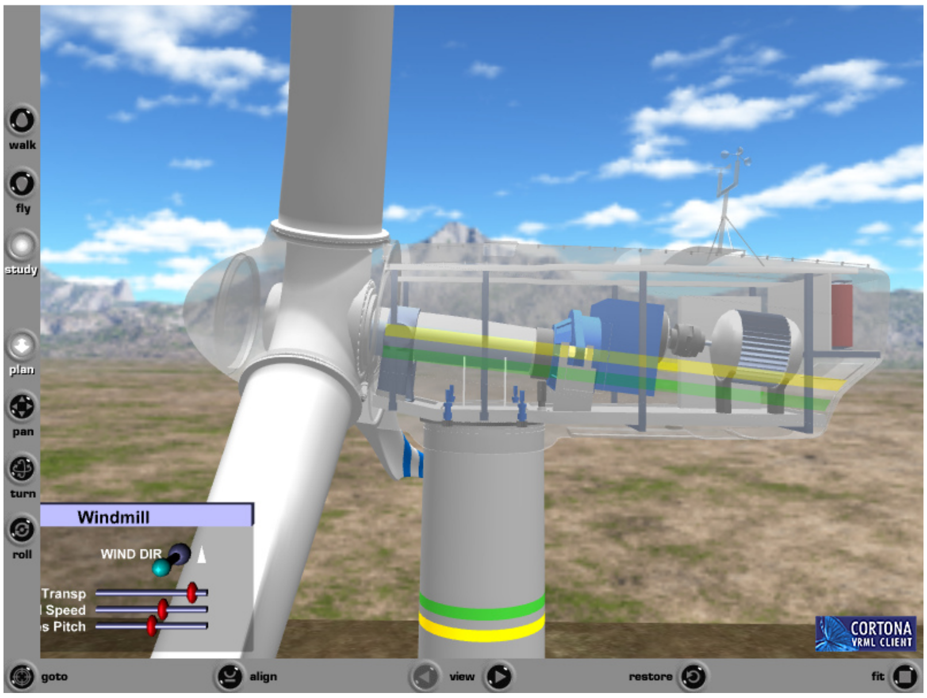


Figure 5.8: *Interactive wind turbine simulation*

applications have experienced a huge growth and have overcome initial trustworthiness and some reluctance from potential buyers (see Egger [22]). E-commerce applications typically use conventional web pages containing searchable catalogs of products and fill-in forms as user interfaces. This is in contrast to traditional shopping where customers browse shops walking around in a specially decorated and distributed space. We intended to apply latest plug-in-free 3D rendering on the Web to simulate the experience of browsing retail shops.

The application required two parts, one for the shop manager and one for the customers. The first part is intended to provided means for the manager to create and populate retail spaces, and to analyse customer activity.

5.4.2 Discussion of alternatives

Conventional electronic commerce is well established nowadays and generates a considerable amount of Internet traffic and business. E-commerce sites provide the means for customers to find products of their interest and actually purchase them. An e-commerce Web site acts as an online electronic catalogue where users can look up a product, read a description and see a picture of it, and finally buy it. The experience is completely different from that of visiting a local modern shop designed with visitor comfort in mind, walking around, browsing and examining objects exhibited, and buying them.

Virtual Reality interfaces to e-commerce have seen some attempts with little success. As explained by Chittaro and Ranon [11], a VR store has the advantage that it can be closer to the real-world shopping experience, more familiar to the customer. One such attampt was *Kinset* [49], aimed at creating and publishing 3D virtual stores. However, Kinset consisted of applications (one for creators and one for visitors) that users had to manually install and run and is thus far from our definition of ubiquitous. Later, *VirtuyMall* [94] was released. Virtuy relied on the *Unity Web Player* browser plug-in to present interactive virtual stores in a 3D mall which display their products with information and allow actual purchasing. The user is presented a third person view that shows his/her 3D avatar from behind as it navigates the environment. The shops in Virtuy are 3D models, but products for sale in them are not, being simple flat images rendered as billboards (i.e. 2D sprites) on the shelves. Representing the user's own avatar helps in the estimation of sizes and enables social interaction functionalities when other users are displayed as well.

An additional feature of our work was the interactive creation of the virtual environments. We identified a few existing systems aimed at the design and decoration of homes. *HomeStyler* [3] was a user-friendly tool to model and furnish homes aimed at casual planning of home construction or reforms. HomeStyler was based on Adobe's *Flash* technology, a browser plugin now in significant decline [69]. Due to the limitations of the technology, editing was made on a top view of the virtual home and previews of the virtual home only displayed a pseudo-3D orthographic rendering from one of eight possible viewpoints (i.e. the view could only be rotated in 45° steps).

As we can see, existing systems with comparable functionalities have used either specific PC applications, application-specific browser plugins or general purpose browser plugins. At the time this research started, the WebGL standard had just been released and seemed an opportunity to create a purely Web-based system without plugins. But being WebGL a relatively low-level imperative interface, some support library was considered necessary for practical development. Several high level JavaScript libraries to manage and render 3D scenes with WebGL had appeared: *GLGE* [9], *Three.js* [10], *X3DOM* [5], to name a few. Among them we chose GLGE due to its balanced design (not too low level and not too high level), its modeling and rendering capabilities and its good ray intersection functionality (crucial as later explained).

5.4.3 Architecture

The system design was based on a client-server architecture. The main application runs in a Web server containing static assets such as core content (HTML, JavaScript scripts, CSS) and the application data including 3D models of the shops and of the products on sale. Additionally, a Web service is exposed to let the client side query the catalog of available products and to perform purchases. Data about the products (such as current price and availability) might not be stored in the application server, but in the actual retail company infrastructure. To get this information, the server can relay specific requests to a remote server made available by said company through SOAP Web service requests. This architecture is depicted in Figure 5.9.

The client interface is a Web page that contains an embedded virtual scene of one or more shops the user can enter. Two different profiles are defined and assigned to each user, *customer* and *administrator*. Upon logging-in, the user profile determines the functionalities presented and the mode of navigation in the virtual environment. On the one hand, customers that enter a shop are intended to walk around as they would in a real shop. So, a first-person view and walking navigation controls are implemented, including mechanisms for ground following and collision detection in order to not traverse walls, for example. Administrators, on the other hand, need more freedom and are given a fly-mode style of navigation.

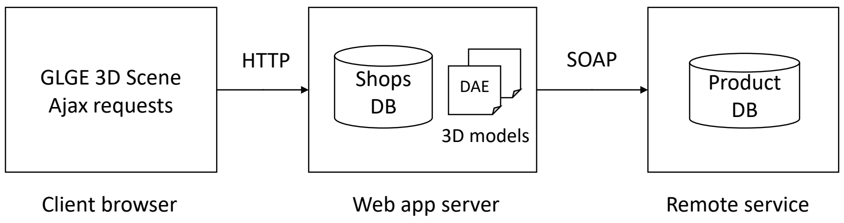


Figure 5.9: *Data and communications architecture*

Administration profile functionality

The administration part of the application allows managers to create and furnish new shops. A new environment can be created in several ways. The user can import a scene already modelled with a 3D modelling tool such as *Blender* or *Autodesk's 3ds Max* or can create the shop geometry interactively. The second case allows the design of the floor plan layout of the shop and presents the user four options with user-editable parameters:

Rectangular Creates a rectangular layout based on three parameters: width, depth and height.

L-shaped Creates a layout with the shape of an 'L' with parameters for length, depth, short length, short depth and height.

Free-form layout A 2D editor is presented that allows the user to draw the shape of the layout interactively using the mouse.

After the floor plan shape is defined, a 3D view of the empty shop is presented, having only floor, walls and ceiling. The user can then furnish the shop with decorative objects and place products for sale around the environment. We decided that a first person navigation mode would be cumbersome and then implemented an unrestricted flight navigation. That is, the camera can be moved in space with the mouse and can go through walls. In this editing mode a panel at the right shows a catalog of available objects, both decoration and products, that can be dragged with the mouse and dropped into the 3D scene. This panel can be seen in Figure 5.10.

Newly added objects can be moved around to a new position in the environment. Upon selection with a mouse click, a blue manipulator handle appears around the object (See Figure 5.11). Objects dropped into the scene or dragged around with heir handle will always stay on top of existing objects (typically shelves or similar). When dragging, a ray is shot from the viewpoint through the mouse position and its intersection with the scene is computed. This ensures objects stay on some support and do not float in the air.

Objects can also be rotated around a vertical axis. Right-clicking on an object and selecting the rotation option shows a red manipulator handle appears. Dragging this handle with the mouse rotates the object.

Visitor profile functionality

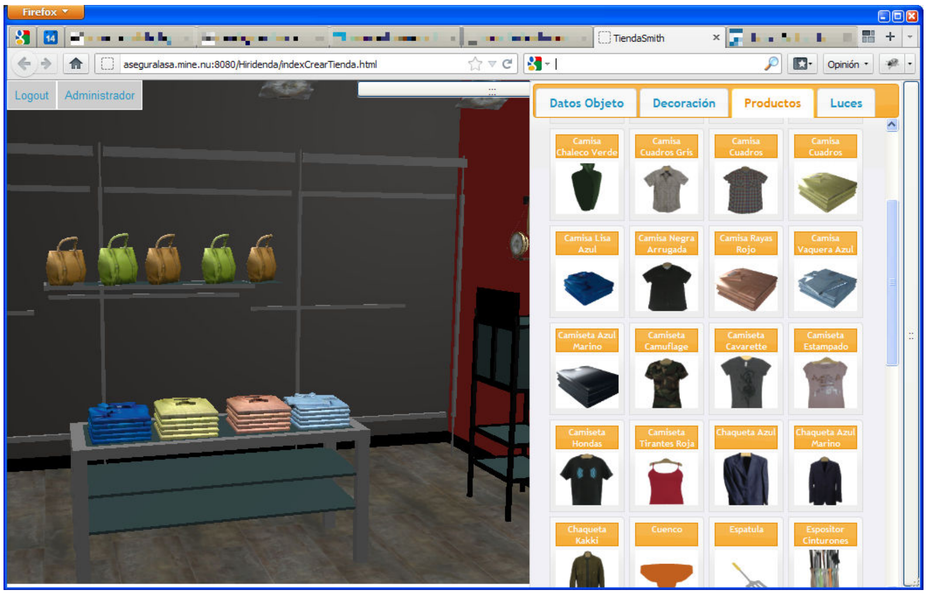


Figure 5.10: *Product placement interface*

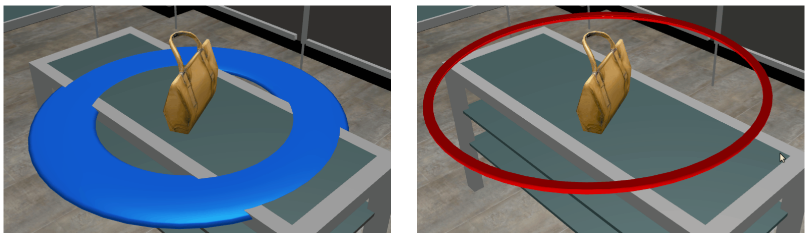
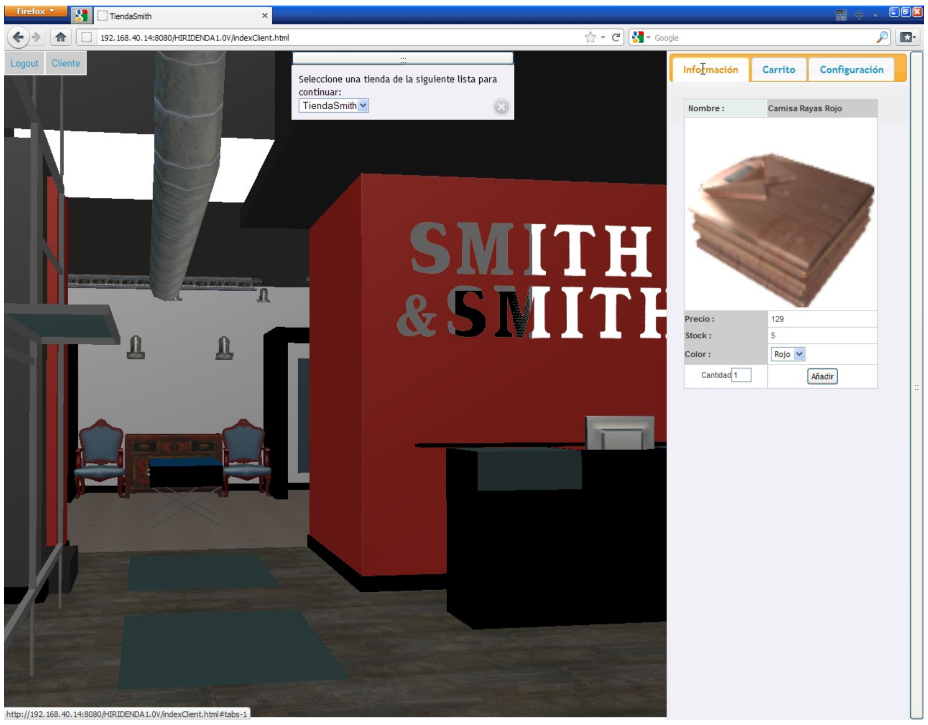


Figure 5.11: *Interactive manipulators: translation (left) and rotation (right)*

Visitors to the virtual shops log into the application and are presented a first-person view of the interior, as seen in Figure 5.12. Interaction is simpler than in the case of adiminstrators as visitors must not be able to modify the environment. As explained earlier, navigation in this case simulates walking with ground following (the camera stays a fixed height above the ground or stairs) and collision detection (the camera does not move through walls or tall obstacles). Either the keyboard or the mouse can be used to navigate. The user can walk in the direction of the view, can rotate around a vertical axis and can look up or down (±90°). Collision detection is performed by tracing a ray towards the user's forward direction and finding its closest intersection point with the environment geometries. Only if the distance to this point is less than a threshold, forward walking is allowed.



192.168.40.14:8080/HIRIDENDA1.0V/indexClient.html

Seleccione una tienda de la siguiente lista para continuar: TiendaSmith

Nombre : Camisa Rayas Rojo

Precio : 129

Stock : 5

Color : Rojo

Cantidad 1

http://192.168.40.14:8080/HIRIDENDA1.0V/indexClient.html#tabs-1

Figure 5.12: *Virtual shop*

The user can walk around to browse the shop and can select products on display by clicking on them. Upon selecting a product from the scene, information about it is displayed in the panel at the right, including description, price, optional parameters (such as color or size) and availability. And the selected product can be added to the virtual shopping cart with a corresponding button.

User behaviour analysis

One last functionality implemented intended to enable shop administrators to monitor visitors' activity. All users movements around the virtual space are recorded. Whenever the user moves, his position and view orientation are stored together with a timestamp. Discrete actions such as selecting or purchasing a product are also stored with a timestamp. All this information is ultimately collected by the application server and stored anonymously.

The manager can access the logs of the latest visitors and play back their sessions from the perspective of the visitor. Additionally, the manager can see the complete trajectory of one or several visitors superimposed on a top view of the shop. To depict more clearly the areas where visitors spend most time, we implemented this using a *heat map* representation. As shown in Figure 5.13.

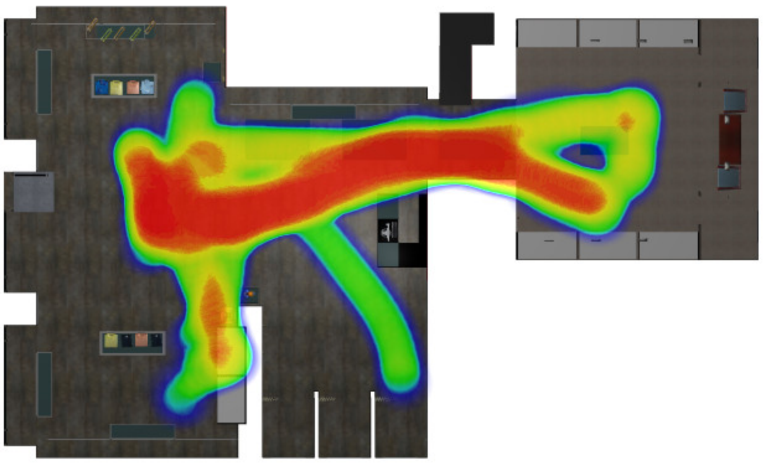


Figure 5.13: *Heat map of user movements around the environment*

A heat map is generated by tracing all users trajectories with a circular kernel on a grayscale image. The kernel adds to the previous image contents as it advances. Areas where one or more users spent more time gets a higher value in the map. Finally, the grayscale map is converted to a color image using a color transfer function and superimposed on a top view image of the shop environment.

5.4.4 Results and discussion

The work carried out in this scenario brought the idea of a first person VR environment to Web-based e-commerce. Unlike previous systems that relied on specific PC applications or browser plugins, this system is usable with just any modern browser supporting WebGL.

The intention of the management functionality was to enable editing of new environments by non-experts in 3D modelling. That is why the editing options were intentionally limited. A full 3D editing tool allows any change to the scene, even placing objects floating in space or rotating them around any axis. But in the real world, we can only place objects on top of other objects and usually can only rotate objects, which have a defined "up" direction, around a vertical axis. We intended to emulate these restrictions of the real world to make editing more intuitive. This reduces the available degrees of freedom and helps ensure a logical result. If a real editing tool would allow changes to 6 degrees of freedom per object, here we only let three free degrees of freedom. In the case of objects attached to walls, such as paintings or lamps this is further reduced because rotation around the vertical axis is blocked and objects follow the orientation of the wall.

Several people from our department, but unrelated to the project, were presented a demo of the platform prototype. While they see the value of being able to create the whole VR environment in the same Web platform, they admit that the visual quality achievable is relatively low. The real shop modelled with an external tool and imported is much more visually rich that the one generated parametrically. Performance (i.e. frame rate and perceived smoothness) is considered good but suffers slightly in the case of the detailed model due to the ray intersections computations needed for ground following and collision detection against a large number of meshes and polygons.

A few pleople from the e-commerce sector tried the prototype as well. Upon testing the system, they found the concept interesting but expressed concerns about the process of updating the catalogue with newly available products. Products need associated 3D models and a skilled designer is required to create them. This may enable a business model in which a studio is hired every time new objects must be displayed. However, this would be a problem for small businesses. Simpler methods of 3D model creation show up as one of the major limitations to the success of these systems

5.5 Visual product configurators use case

In this final section we describe an early application use case due to its interest as an innovative solution that highlights the evolution of technologies and standards over the last years.

Product configuration systems are software tools that assist customers, sales staff and design engineers to easily assemble and customize product solutions. They allow the user to setup a specific solution for a client, make sure that the solution can be built, and automatically provide a quote. Some companies with well structured product lines already benefit from computer programs providing that functionality via fill-in forms linked to databases of configurable options and lists of constraints. These programs may also generate a simple sketch of the custom product, giving the client a suggestive idea of its appearance as it is being modified.

5.5.1 Objective

The objective of this research was to create interactive product configuration systems with a focus on visual fidelity. That is, applications that given a set of configurable parameters produce virtual models that given a faithful representation of what the product, once manufactured will look like. In particular, we targeted the configuration of lifts as a specific use case. The goal was to highlight the aesthetics and appearance of a finished lift interior in all possible configurations, as opposed to the availability of material catalogs.

An additional requirement of this research was that a complete configuration system had to be a desktop application and a less capable version had to be ubiquitous (online accessible). And these two versions had to be in sync and up-to-date: if new functionalities or configuration options were added or changed in one version, it should be easy and quick to update the other version accordingly. Finally, it was desired that the final system did not impose hard requirements on specific high-end graphics hardware and that it could be used (though suboptimally) even with software-based rendering (without GPU support).

5.5.2 Discussion of alternatives

At the time this work was carried out, standards for 3D graphics on the Web were limited to VRML and a recently released but not widespread X3D. An initial very simple configuration demonstrator was developed using VRML and scripts embedded in a Web page. The page had a few UI controllers to modify some parameters (just lift dimensions and materials) and the scripts made use of the VRML EAI (External Authoring interface) to build a 3D virtual lift from scratch based on the parameters values. The demonstrator served as a first proof of concept of parametric object construction. But it was soon noticed that the capabilities of VRML or even X3D were not sufficient for the required visual realism. Particularly, the representation of mirrors and reflective materials could not be achieved with the required quality. And moreover, it would have been hard to maintain this development in sync with the desktop configuration application.

Several alternatives to VRML existed (see Table 5.1), all based on browser plugins or Java applets and using a declarative scene description architecture.

To have more control over the rendering would require a custom graphics engine. The only way at the time to achieve lower level control of graphics generation or even hardware acceleration was through native code in browser plugins. Plug-ins are extensions to a browser's functionalities implemented in native languages such as C or C++. Plugins are embedded in Web pages and can usually communicate with JavaScript scripts in the same page using a per-browser specified API. The two main technologies available at the time were the *Netscape Plugin API* (NPAPI), supported in *Netscape Navigator*, and Microsoft's *ActiveX*, supported in Microsoft's *Internet Explorer*.

As long as native code can be executed using any of the above technologies, the software can directly use any available API to draw on the target window. In order to take advantage of hardware graphics acceleration the two main available APIs were *OpenGL* and *Direct3D*. These APIs allowed imperative rendering over a fixed-function pipeline, including multiple rendering passes.

5.5.3 Architecture

The integration technology selected for a research demonstrator was Microsoft's ActiveX and the graphics rendering was based on OpenGL. Microsoft Internet Explorer was by far the dominant browser at the time and ActiveX provided a relatively simple way to expose an API to JavaScript scripts. An ActiveX control presents a window (corresponding to a rectangular area in the Web page) on which the software draws a virtual 3D scene using OpenGL functions. This solution enabled an easy synchronization between the desktop version and the Web version of the configurator as most of the parametric modelling and rendering code could be shared.

Modular architecture

In order to manage and render the virtual scene we designed and implemented a *scene graph* structure. A diagram of the configuration system architecture is depicted in Figure 5.14. The different boxes represent software modules, and the sheet-like rectangles represent data files, while the arrows indicate what modules invoke or use what modules or files.

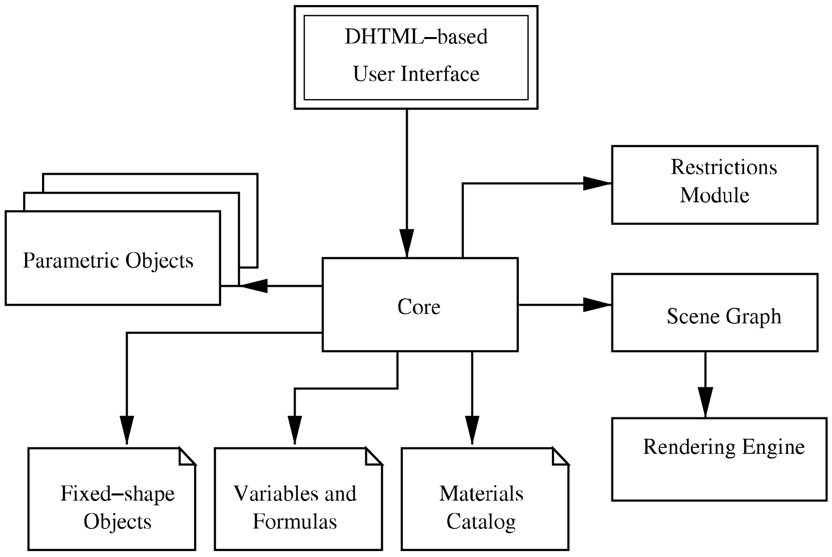


Figure 5.14: *Visual configuration system architecture*

The following is a description of the differnt modules that formed the architecture of this visual configuration system:

- **Core**: Contains the main logic and exposes an API to the user interface in the Web page.

- **Parametric objects**: These modules have the capability to construct 3D geometries for parametric objects in the scene. That is, objects whose geometry and appearance depend on the value of a set of parameters. Several categories of parametric objects exist, such as walls, ceilings/illumination, control panels and hand rails.

- **Fixed-shape objects**: These are solid geometries such as push buttons, digital displays and hand rail fixtures.

- **Variables and formulas**: This file contains the a list of mathematical expression needed to compute the variables needed in building the lift geometry and are used by the parametric objects modules. Some variables have a constant value while others have an expression value as a function of other variables. For example, the width of each module of a modular ceiling is written as a function of the lift main dimensions.

- **Materials catalog**: This file contains a list of surface material appearance definitions to be used by the parametric objects. Along with typical shading parameters such as Lambertian diffuse colour and Blinn specular shininess, these definitions included a *reflection* factor to be used by the rendering engine in recursive reflection effects.

- **Restrictions module**: This module contains the product configuration logic provided by the manufacturing company. It ensures that the current combination of construction parameters is allowed. Basically, it contains the configuration state defined by a set of parameter values. For each parameter it can provide a list of available values (e.g. available surface materials for a specific wall type, available door distributions or possible ceiling illumination designs based on the selected lift class). When setting a new value for a configuration parameter, this module returs a set of parameters and their values that automatically change as a consequence.

- **Scene graph**: This module contains a generic object-oriented scene graph structure that the core and the different parametric object modules use to build the virtual 3D scene representing the lift. This structure is a tree of nodes of different types. The nodes can represent geometries (polygon meshes), light sources or transformations.

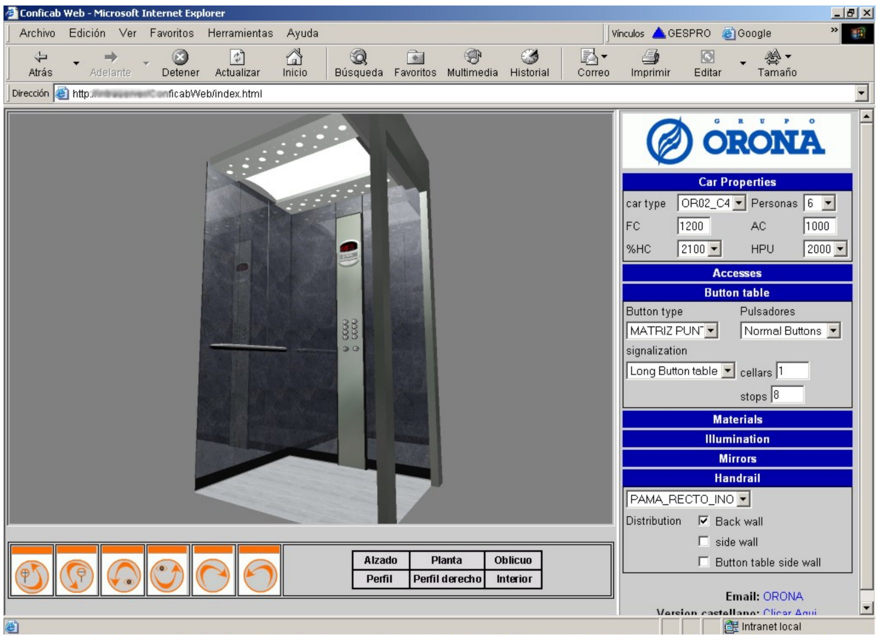
- **Rendering engine**: This module is in charge of rendering an image the scene described by the scene graph using the current camera view point. Two rendering engines were actually implemented, one for real-time visualization based on OpenGL and one based on an external ray tracer for added realism off-line (this one only available in the desktop version). Our OpenGL engine implemented some special features that will be explained below.

- **User interface**: The ubiquitous version of the configuration application had a dynamic HTML-based user interface synchronized with the configuration logic. The user interface displayed an interactive 3D view of the curent state of the lift and a control panel for interactive configuration. Upon any change by the user, scripts in the page queried the restrictions module about the available values for things like materials, sizes or optional elements and the user interface adapted accordingly.

The user is initially presented a front view of a lift built with default parameters. The user can then change any construction or style parameter through the user interface controls. Any change immediately triggers a reconstruction of the lift model based on the new set of parameter values. The 3D view can be interactively rotated and translated by dragging it with the mouse or by selecting one of the predefined views.

Figure 5.15 shows the main user interface of the lift configuration system in Microsoft's Internet Explorer Web browser.

The rendering engine was based on the OpenGL fixed-function pipeline that was readily available at the time of development. Yet, it achieved very convincing visual effects to represent the actual appearance of the lifts. The main challenges in this use case were the presence


Conficab Web - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Vínculos GESPRO Google
Atrás Adelante Detener Actualizar Inicio Búsqueda Favoritos Multimedia Historial Correo Imprimir Editar Tamaño
Dirección http:[redacted]ficabWeb/index.html
GRUPO ORONA
Alzado Planta Oblicuo
Perfil Perfil derecho Interior
Intranet local

Car Properties

car type	OR02_C4	Personas	6
FC	1200	AC	1000
%HC	2100	HPU	2000

Accesses

Button table

Button type	Pulsadores
MATRIZ PUN	Normal Buttons
signalization	
Long Button table	cellars 1
	stops 8

Materials

Illumination

Mirrors

Handrail

PAMA_RECTO_INO

Distribution ☑ Back wall ☐ side wall ☐ Button table side wall

Email: ORONA

Version castellano: Clicar Aqui

Figure 5.15: *Lift configurator in a Web browser*

of mirrors and partially reflecting surfaces, the presence of polished metals such as stainless steel, and the effect of spot lights at the ceiling.

Mirror reflections

Several alternative methods exist to simulate the effect of planar mirrors or mirror-like surfaces in rendered images [1]. They usually require a specifically designed render loop involving more than one rendering pass. One of the alternatives is based on the use of textures. A virtual camera is placed in an inverted position on the other side of the mirror plane and the scene is rendered into an off-screen image. That image is then applied as a texture map on the mirror surface with an appropriate projective texture transform when rendering the real scene. This method has the advantage that the texture map can be post-processed in order to produce effects such as blurred reflections. But the often limited texture resolution lowers the quality of the reflection making pixels visible.

The approach we selected, inspired in the ideas presented by Nielsen [73], maintains a high resolution across the whole image. Basically, it consists of a multi-pass algorithm in three stages. A first pass marks all mirror surfaces in the stencil buffer. Then for each existing mirror an inverted scene is rendered over the marked areas by applying a special reflection transform matrix to the vertices and after establishing a clipping plane on the mirror surface. Finally, the real scene is painted including a blended shape on the mirror surface in case it is not a perfect reflector. Equation 5.4 shows the mirror matrix we use for the general case of an arbitrary reflection plane containing the point $\vec{p}$ and having $\vec{n}$ as surface normal.

$$M(\vec{p},\vec{n}) = \begin{bmatrix} n^2 - 2n_x^2 & -2n_xn_y & -2n_xn_z & 2n_x(\vec{p}\cdot\vec{n}) \\ -2n_yn_x & n^2 - 2n_y^2 & -2n_yn_z & 2n_y(\vec{p}\cdot\vec{n}) \\ -2n_zn_x & -2n_zn_y & n^2 - 2n_z^2 & 2n_z(\vec{p}\cdot\vec{n}) \\ 0 & 0 & 0 & n^2 \end{bmatrix} \quad (5.4)$$

An important feature of our engine, apart from the implementation of the above algorithm, is that it allows placing reflecting surfaces in the scene in a relatively easy way. The scene graph itself keeps a list of reflection planes (each specified as a point and a normal) and materials have a reflection factor property. In order to put a reflective shape, one simply has to assign it a material with nonzero reflection factor (1 to make it a perfect mirror) and add its containing plane to the list of reflection planes. Usually real-time implementations of mirrors, on the other hand, are ad-hoc hard-coded demonstrations, but do not address the general case.

The number of allowed mirrors in a scene is not explicitly limited, and the only practical limitation is an increasing performance penalty with every reflecting surface added. The system handles inter-reflections between facing mirrors up to a given level of recursion. In Figure 5.16

the effect of recursive reflections is shown. There is only one lift control panel in the scene and two mutually facing mirrors. The control panel seen at the left is reflected first in the mirror facing the control panel and then in the mirror next to the control panel.

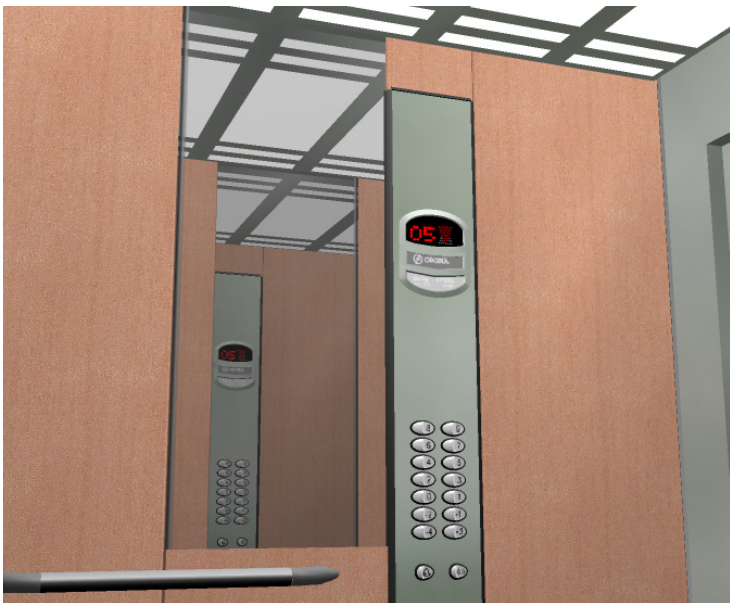


Figure 5.16: *Recursive mirror reflections*

View-independent environment mapping

Environment or reflection mapping is a technique –actually a set of techniques, as explained by James D. Foley and Hughes [44]– that can improve the perceived quality of a rendered image by simulating the reflection of the environment on the surface of an object of any shape. The technique makes an assumption that is known to be incorrect but the resulting effect is visually satisfactory. Reflection mapping uses an image of the environment surrounding an object contained in one or several texture maps and for each pixel in the object a reflected vector is calculated based on the surface normal and a source texel is selected from it. There are several techniques to perform this calculation depending on how the map is represented.

Spherical environment mapping is the simplest technique, requiring only one texture image. In its classical form, the mapping between points on the sphere and texture coordinates is made with a set of standard formulas and as a result produces view dependent reflections.

Nevertheless, we considered view-dependent environment mapping insufficient for our requirement of high-quality rendering and a view-independent calculation is therefore required. One solution is traditionally thought to be the use of cube maps through some graphics API extension and special hardware support. However, in the context of a Web application, we cannot rely on the availability of such advanced features on the client machine. We thus implemented a software-based view-independent spherical environment reflection technique. A related computationally inexpensive technique was presented in Heidrich and Seidel [37] but requires two texture images instead of one.

Our system uses non-standard equations that reduce the distortions on the sphere map and makes it more intuitive and easier to prepare. The method calculates at each vertex the reflected vector $\vec{r} = (x, y, z)$ given the camera position and the surface normal, then converts this vector to spherical coordinates (r, θ, ϕ) (Equation 5.5) and calculates the corresponding texture coordinates by considering θ and ϕ the polar coordinates of a vector on the texture map, and converting it to cartesian coordinates (Equation 5.7).

$$\theta = \arctan_2(y, x) \tag{5.5}$$

$$\phi = \frac{1}{4} + \frac{1}{2\pi} \arctan \frac{z}{\sqrt{x^2 + y^2}} \tag{5.6}$$

$$\begin{cases} u = \frac{1}{2} + \phi \cos\theta \\ v = \frac{1}{2} + \phi \sin\theta \end{cases} \tag{5.7}$$

It may look too complex to be computed for each vertex in real time, but the whole process can be simplified through the following expressions:

$$\begin{array}{ll} m = \frac{1}{\sqrt{x^2+y^2}} & \left\{ \begin{array}{l} u = \frac{1}{2} + \phi m x \\ v = \frac{1}{2} + \phi m y \end{array} \right. \\ \phi = \frac{1}{4} + \frac{1}{2\pi} \arctan mz & \end{array} \quad (5.8)$$

The effect of this technique can be seen in Figure 5.17 (it is more apparent when rotating the view, which cannot be appreciated in a still image). Environment mapping was applied to metallic surfaces including polished stainless steel walls and chromed steel bars. The environment map was intentionally blurred to give the apearance of fuzzy reflections seen in an actual lift walls.

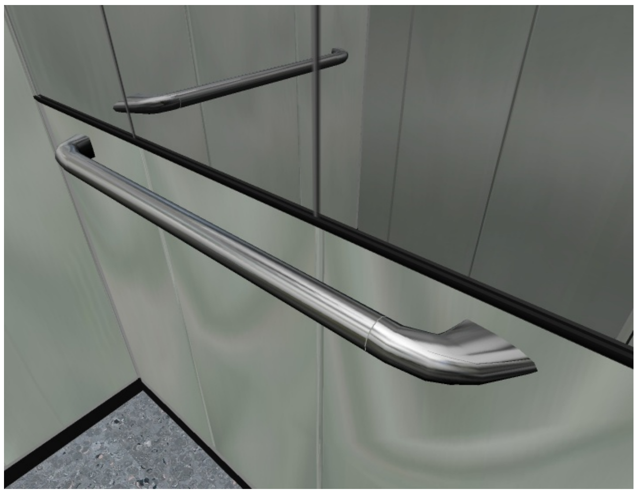


Figure 5.17: *Environment mapping reflection applied to chromed steel handrails and polished steel walls*

Improved shading

Gouraud shading and Phong shading are widely used interpolation methods to render a polygon mesh. In the Gouraud shading methods, lighting and reflection calculations are only performed at the vertices and the resulting values are then interpolated across the polygons. Smoothly varying gradients of diffuse illumination are acceptably handled by this simplification, but specular highlights and the lighting produced by spotlight lamps are more problematic. It is obvious that specular highlights can be completely missed or distorted by Gouraud shading for polygons whose screen areas are greater than the highlight areas. One solution is to render a high-resolution mesh, however, the Web context requires to keep mesh sizes low.

On the other hand, Phong shading does all calculations per pixel using interpolated normals so that even small highlights are represented. Nowadays, in WebGL or other programmable shader environments it is conventional to use Phong shading in interactive applications. But at the time, using the fixed-function pipeline and supporting software-only rendering that was not feasible and Gouraud shading was the norm.

We therefore implemented real time triangle subdivision to efficiently obtain high quality highlights from Gouraud shading and standard hardware rendering systems [12]. Our rendering engine recursively subdivided triangles *on-the-fly* until all their edges were smaller than a threshold.

The subdivision method initially implemented was *planar*: it inserted new smaller triangles within each original triangle keeping its plane and normal (for each edge, a new vertex is inserted at its mid point). We then tried to create an experimental real-time subdivision that would create the new sub-triangles following the shape of curved surfaces to enhance the appearance of low polygon-count models. In this case, the new vertex inserted per edge is shifted outward or inward of the triangle depending on the normals at both edge vertices. The result of this method can be seen in Figure 5.18 where rendering in the two left-most images was done with flat shading in order to highlight the triangle sizes.

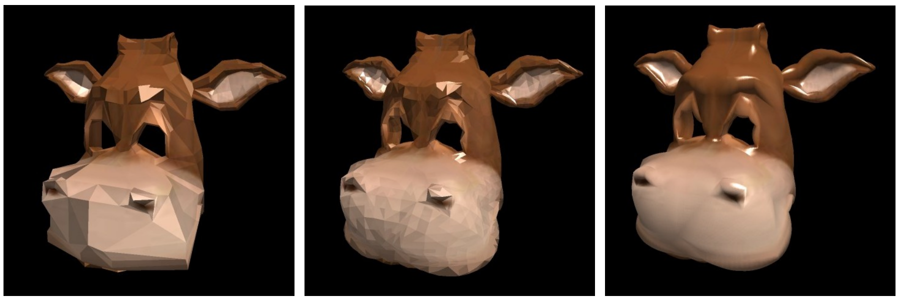


Figure 5.18: *Real-time subdivision: (left) low polygon count model, (center) online subdivided model, (right) subdivided model with smooth shading*

5.5.4 Results and discussion

The research and development described in this section demonstrated the concept of visual product configurators. It has also highlighted how dramatically the graphics technology landscape has evolved over the years. The work was carried out before WebGL existed and before programmable shaders became mainstream. Algorithms had to be provided adapted to the situation of the time.

The *ubiquity* of the solution was partial as it only supported Windows-based PCs with the Internet Explorer browser (the vast majority at the time) and required an initial download and installation of the configurator ActiveX control. However, this download was relatively transparent: it was triggered automatically on the first visit to the Web page, and it only downloaded the core of the software. The rest of the software modules and data files were downloaded on demand and the system was kept up-to-date. Nevertheless, deploying custom ActiveX controls or browser plug-ins has fallen out of use nowadays and running native code is considered a security risk.

Certainly, with the availability of WebGL a similar configuration system could be developed today following the same high-level logic but relying on JavaScript and WebGL to implement geometry processing and 3D rendering. It would then be truly ubiquitous.

5.6 Contributions

This section summarises our contributions in the field of ubiquitous virtual reality visualization.

A notable contribution was our method for volume rendering on WebGL. It proved the feasibility of rendering volume datasets on the Web without plugins, a functionality previously only found on specialised desktop applications. Our paper Congote et al. [13] has been cited numerous times. Furthermore, the direct rendering of radar data in polar coordinates is unmatched.

Our work on the EnEf project showed the value of simplified virtual reality simulations as illustrations for learning a subject like energy efficiency. Typical e-learning content contains text, images and sometimes videos. We contributed an interactive simulated building model that served as a highly-enhanced illustration and was compatible with the web platform containing the e-learning content. This work was published in Álvaro Segura et al. [58].

A similar contribution to e-learning was presented in Álvaro Segura et al. [59] aimed at wind turbine maintenance technicians. We contributed a methodology to embed 3D VR simulations in SCORM learning content, both interactive and non-interactive.

Another contribution was an architecture for Web-based virtual reality acting as an interface to e-commerce. Where e-commerce typically uses web pages containing catalogs and forms as interfaces we presented a first person 3D environment. Additionally, the system tracks users' movements around the virtual environment providing information about the areas of higher interest and where more relevant items should be placed. This was published in Elordi et al. [23].

Our last contribution was a concept for a parametric visual product configuration system. It allows the presentation of a virtual model, relatively faithful to reality, whose shape and appearance depends on a set of user selectable features and parameters. It was published in

Segura et al. [83].

5.7 Publications

The research described in this chapter is supported by the following publications:

- John Congote, Álvaro Segura, Luis Kabongo, Aitor Moreno, Jorge Posada, Oscar Ruiz. *Interactive visualization of volumetric data with WebGL in real-time* (2011). Proceedings of the 16th International Conference on 3D Web Technology, Web3D 2011, Paris, pp. 137-145.

- Álvaro Segura, Aitor Moreno, Petra Müsebeck, Sybille Hambach. *Integrating 3D Virtual Reality Simulations in Reusable e-learning Courses* (2009). Proceedings of e-Learning Baltics 2009, Rostock.

- Álvaro Segura, Sabine Delaitre, Igor García Olaizola. *Interactive 3D simulation tool for Energy Efficiency in Buildings* (2012). Proceedings of the International Workshop on Energy Efficiency for a More Sustainable World EEMSW2012, Ponta Delgada.

- Unai Elordi, Álvaro Segura, Jon Goenetxea, Aitor Moreno, Jon Arambarri. *Virtual reality interfaces applied to web-based 3D E-commerce* (2012). ASME 2012 11th Biennial Conference on Engineering Systems Design and Analysis, Nantes, ESDA 2012, 1, pp. 341-350.

- Álvaro Segura, Iosu Arizkuren, Iñaki Aranburu, Iñaki Telleria. *High Quality Parametric Visual Product Configuration Systems Over the Web* (2005). Proceedings of the 10th International Conference on 3D Web Technology, Web3D 2005, Bangor, (pp. 159–167).

Chapter 6

Conclusions and future work

This thesis provides contributions in several areas of virtual reality visualization. They are presented in Chapters 3, 4 and 5 and correspond to the following areas:

- Stereoscopic viewing in head-mounted displays and their calibration for correct depth perception, including accomodation.

- Immersive visualization: alternative setups to provide a higher feeling of being immersed in a virtual world.

- Ubiquitous visualization: algorithms and applications of ubiquitous interactive 3D visualization where users access remote applications from anywhere with different computing devices.

Each chapter provides background, an extended description of the contributions and related publications to these contributions. Section 6.1 briefly describes the achievements of the presented research. Section 6.2 enumerates contributions. Finally, Section 6.3 proposes future work.

6.1 Conclusions

Virtual reality and visualization are broad areas encompassing different modes of presentation of virtual objects and environments. Not all applications have the same requirements and it is not feasible or sensible to apply all the highest complexity techniques and devices to every use case. Common to all cases considered in this thesis is the requirement of real-time 3D computer graphics rendering with some degree of realism. We consider that in the cases of immersive visualization, stereoscopic rendering is mandatory.

6.1.1 Stereo perception and accommodation calibration

Chapter 3 addressed the issues related with stereoscopic perception, particularly in the case of head-mounted displays and the *vergence/accommodation mismatch* problem. A new architecture was proposed for calibration and control of variable accommodation in HMDs. However, even if techniques for accommodation variation can be found in the literature, actual applications using it are very rare. That chapter stressed how the correct perception of depth in virtual reality requires a synchronization of the stereo view vergence and accomodation stimuli. But since variable accommodation would make HMDs much more complex, this is nearly always ignored, and most applications can tolerate the small mismatch provoked by a fixed accommodation.

The HMD calibration methods proposed in Chapter 3 have two separate parts: one for perspective and stereoscopy, and one for accommodation. In this way, a variable-accommodation HMD can provide matched vergence and accommodation stimuli. The first part is also applicable to conventional HMDs with fixed accommodation.

6.1.2 Immersive visualization

Chapter 4 presents results of applied research for application-specific use cases of immersive visualization systems. The different cases presented show how different hardware setups can be deployed tailored at the needs of the different cases, from HMDs to stereo projection screens.

The manual welding simulator presented represents a classic approach to virtual reality, a stereo HMD with head and hand tracking. On the other hand, the machinery simulator uses a custom HMD in a highly innovative way to provide an alternative augmented reality experience.

Additionally, Section 4.3 presents an immersive viewer system for weather radar data based on stereo projection. The system used dual back projectors and passive stereo with polarized filters on the projectors and the users' glasses. The degree of immersiveness is lower in this setup compared to an HMD as only part of the field of view in front of us forms a window into the virtual world. The goal of the research was to represent the volumetric radar data together with the surrounding topography, mixing Geographic Information Systems (GIS) and meteorology. The size of the projection and its stereoscopic nature let observers perceive the three-dimensional shape of atmospheric phenomena and their relation to the underlying terrain, which is hard to see in a conventional computer screen.

6.1.3 Ubiquitous visualization

Chapter 5 presents application use cases for ubiquitous visualization. This area is growing in importance and today more and more applications are used remotely through Web browsers without having to install them locally. The systems described in this chapter highlight the evolution of Web-based 3D graphics standards and technologies over the last years. From application-specific browser plugins, through VRML scenes to WebGL applications, the capabilities of Web 3D applications have seen huge improvements. The use cases described made use of the available technologies at the time their research took place. The evolution of these technologies has gone from declarative scene description languages (e.g. VRML and X3D) suported by a few well-known browser plugins, to modern graphics programming using programmable shaders.

We can conclude that the future of Web-based 3D visualization is WebGL (versions 1.0 or 2.0) and its successors. They provide increasing access to low-level features of the end user's hardware. For most application development, new JavaScript libraries will continue to let developers build 3D environments declaratively or in a scene-graph way (as the simulated 3D building and the virtual e-commerce described in Sections 5.2 and 5.4). For specific uses with specialised rendering requirements, lower level software and custom fragment shaders will be necessary (as in the weather radar volume rendering system). In cases of complex graphics requirements (e.g. volume rendering), legacy systems tended to use server-side rendering and a complex network communication mechanism to present the produced images to the client. However, the capabilities of client-side rendering today make those architectures unnecessary.

It is important to note that the context and data used in the weather radar rendering system and in the immersive radar viewer of the previous chapter are the same. This is an example of situations and data that can be presented in different ways for different use cases and requirements. For higher-end visualization Section 4.2 proposed an immersive projection setup which requires quite specialized hardware. On the other hand, for an ubiquitous use, Section 5.1 shows WebGL used in a novel way to create an innovative volume rendering technique. Direct radar volume rendering had not been done that way before, as far as we know.

6.2 Contributions

Some contributions provided by this thesis have been presented at the end of the central chapters on immersive visualization applications and ubiquitous visualization applications. This Section summarizes all contributions.

6.2.1 Stereo perception and accommodation calibration

- Method for calibration of HMD stereo view parameters
- Method for calibration of accommodation in variable focus HMDs
- Method for control of accommodation of variable focus HMDs

6.2.2 Immersive visualization

- A Mixed Reality visualization architecture based on chroma-key with a custom HMD having a pair of forward-looking cameras attached.
- An architecture for machine simulators that uses real machine cabs and controls.
- A versatile arcitecture for the simulation of hydraulic excavators, dumpers and aerial work platforms.
- A novel combination of volumetric weather radar data with a digital terrain model in a large stereo projection setup for an immersive experience.
- An immersive simulator of manual welding based on relatively low-cost VR devices (head-mounted display and magnetic trackers) together with real time simulation and graphics.

6.2.3 Ubiquitous visualization

- A method for real-time volume rendering on the Web without plugins using WebGL.
- An algorithm for rendering volumetric weather radar data in WebGL, using directly the reflectivity data arranged in spherical coordinates.
- Interactive 3D building simulation for learning about energy efficiency in an online e-learning course.
- An interactive 3D model of a wind turbine for learning about renewable energy in an online e-learning course.
- A Web-based 3D visual product configurator for lifts.
- Methods for rendering scenes containing mirrors and semi-reflective surfaces including recursive reflections and a new simple spherical environment reflection mapping technique.

6.3 Future work

The calibration and control system for head-mounted displays described in Chapter 3 dealt with classic design HMDs based on microdisplays and relatively narrow fields of view. In that case and given the actual devices used for experimentation, the little optical distortions were ignored and the view of the displays was considered approximately frontal. An interesting work would be to extend the system to the recent trend of wide field-of-view HMDs based on a single large screen. Lee and Hua [53] proposed a calibration method that estimates distortion parameters but did not explicitly consider stereo vision. They calibrated each eye separately. The HMDs they used for experimentation have relatively low FOVs. So, there are challenges in this area that require research.

Additionally, the accommodation control method used in this thesis was based on the position of the *main* object shown in front of the user. This is certainly a limitation as the user might look somewhere else. A possible better solution would be to set the physical accommodation distance to match where the user is actually looking. Such an approach requires the use of an eye tracking system that can fit in the HMD and which can estimate the user's gaze convergence distance. Our research group made some early experiments in this regard with our own gaze tracking but results were still not satisfactory.

For a more complete emulation of real-world vision in VR, another aspect to simulate would be the actual blurring of objects closer or further from the focus distance (i.e. a depth of field simulation). Developments and testing are necessary to evaluate the perceived impression of this effect and the most appropriate techniques to implement it. This should take into account a balance between physical accuracy and computational complexity.

Regarding ubiquitous visualization, one recent technology that deserves attention is *WebVR*. This standard API enables the use of WebGL 3D graphics on modern immersive HMDs. This will bridge the gap between immersive VR and ubiquitous visualization. Future work will try to bring applications like those described in this thesis to the immersive world enabled by new devices. This should expand the range of supported display devices to access an application, from the desktop and mobile devices used now, to advanced personal displays.

A line of work that has already started is the introduction of 3D visualization and interaction technologies in the manufacturing industry, including immersive and ubiquitous. Our team created some applications on visualization of manufacturing machines, processes and automation systems. Coupled with real-time communication systems and data storage and analysis, these can enable ubiquitous *digital twins*.

Bibliography

[1] Akenine-Möller, T. and Haines, E. (2002). *Real-Time Rendering, 2nd ed.* A. K. Peters.

[2] Andreessen, M. (1993). NCSA Mosaic technical summary. *National Center for Supercomputing Applications*, 605.

[3] Autodesk (2011). Autodesk Homestyler at `http://www.homestyler.com`. http://www.homestyler.com.

[4] Balaras, C. A., Gaglia, A. G., Georgopoulou, E., Mirasgedis, S., Sarafidis, Y., and Lalas, D. P. (2007). European residential buildings and empirical assessment of the Hellenic building stock, energy consumption, emissions and potential energy savings. *Building and Environment*, 42(3):1298 – 1314.

[5] Behr, J., Eschler, P., Jung, Y., and Zöllner, M. (2009). X3DOM: a DOM-based HTML5/X3D integration model. In *Proceedings of the 14th International Conference on 3D Web Technology*, pages 127–135. ACM.

[6] Bell, G., Parisi, A., and Pesce, M. (1995). The Virtual Reality Modeling Languag: Version 1.0 Specification. `http://paulbourke.net/dataformats/vrml1/`.

[7] Benzeroual, K., Allison, R. S., and Wilcox, L. M. (2010). Distortions of Space in Stereoscopic 3D Content Karim. In *SMPTE*

International Conference on Stereoscopic 3D for Media and Entertainment, volume 1100.

[8] Brewster, D. (1856). *The Stereoscope - Its History, Theory and Construction.*

[9] Brunt, P. (2010). GLGE, at `http://www.glge.org/`.

[10] Cabello, R. (2010). Three.js, at `https://threejs.org/`.

[11] Chittaro, L. and Ranon, R. (2002). New directions for the design of virtual reality interfaces to e-commerce sites. In *Proceedings of the Working Conference on Advanced Visual Interfaces - AVI '02*, page 308.

[12] Cho, Y., Neumann, U., and Woo, J. (1996). Improved Specular Highlights With Adaptive Shading. In *Proceedings of the 1996 Conference on Computer Graphics International*, page 38.

[13] Congote, J., Segura, A., Kabongo, L., Moreno, A., Posada, J., and Ruiz, O. (2011). Interactive visualization of volumetric data with WebGL in real-time. In *Web3D 11 Proceedings of the 16th International Conference on 3D Web Technology*, volume 326, pages 137–146. ACM Press.

[14] Crawley, D., Hand, J., Kummert, M., and Griffith, B. (2008). Contrasting the capabilities of building energy performance simulation programs. *Building and Environment*, 43:661–673.

[15] Crawley, D., Lawrie, L., Winkelmann, F., Buhl, W., Huang, J., Pedersen, C., Strand, R., Liesen, R., Fisher, D., Witte, M., and Glazer, J. (2001). EnergyPlus: creating a new-generation building energy simulation program. *Energy and Buildings*, 33:443–457.

[16] Cruz-Neira, C., Sandin, D. J., and DeFanti, T. A. (1993). Surround-screen projection-based virtual reality. *Proceedings of*

the 20th annual conference on Computer graphics and interactive techniques - SIGGRAPH '93, pages 135–142.

[17] Cruz-Neira, C., Sandin, D. J., DeFanti, T. a., Kenyon, R. V., and Hart, J. C. (1992). The CAVE: audio visual experience automatic virtual environment. *Communications of the ACM*, 35(6):64–72.

[18] Da Dalto, L., Balet, O., Benus Jr, F., and Steib, D. (2008). CS WAVE: Learning welding motion in a virtual environment. Technical report.

[19] Desai, A. (2010). Energy Wise E-learning Suite, at `http://www.energy-wise.eu`.

[20] Di, Z., Xinzhu, S., Peng, W., Duo, C., and de Bougrenet de la Tocnaye, J. (2015). Comparative visual tolerance to vertical disparity on 3D stereo versus lenticular autostereoscopic displays. *Journal of Display Technology*, (c):1–1.

[21] Dodgson, N. A. (2005). Autostereoscopic 3D displays. *Computer*, 38(8):31–36.

[22] Egger, F. N. (2001). Affective Design of E-Commerce User Interfaces : How to Maximise Perceived Trustworthiness. *CAHD: Conference on Affective Human Factors Design*, pages 317–324.

[23] Elordi, U., Segura, A., Goenetxea, J., Moreno, A., and Arambarri, J. (2012). Virtual reality interfaces applied to web-based 3D E-commerce. In *ASME 2012 11th Biennial Conference on Engineering Systems Design and Analysis, ESDA 2012*, volume 1.

[24] Fast, K., Gifford, T., and Yancey, R. (2004). Virtual training for welding. *ISMAR 2004: Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, (Ismar):298–299.

[25] Figl, M., Birkfellner, W., Hummel, J., Ede, C., Hanel, R. A., and Bergmann, H. (2003). Calibration of an optical see through head mounted display with variable zoom and focus for applications in computer assisted interventions. In Galloway, Jr., R. L., editor, *Proceedings of SPIE - The International Society for Optical Engineering*, volume 5029, pages 618–623.

[26] Figl, M., Ede, C., Birkfellner, W., Hummel, J., Hanel, R., and Bergmann, H. (2005). Design and automatic calibration of a head mounted operating binocular for augmented reality applications in computer aided surgery. In Galloway, Jr., R. L. and Cleary, K. R., editors, *Progress in Biomedical Optics and Imaging - Proceedings of SPIE*, volume 5744, pages 726–730.

[27] Fischer, R. E., Reiley, D. J., Pope, C., and Peli, E. (1997). Methods for improving depth perception in HMDs. In *Proceedings of the Workshop on The Capability of VR to meet Military Requirements*, number December, pages 5–9.

[28] Freina, L. and Ott, M. (2015). A literature review on immersive virtual reality in education: State of the art and perspectives. *11 th International Scientific Conference eLearning and Software for Education*, pages 133–141.

[29] Freund, E., Rossmann, J., and Hilker, T. (2001). Virtual reality technologies for the realistic simulation of excavators and construction machines: from VR-training simulators to telepresence systems. In *Proc. SPIE Mobile Robots XV and Telemanipulator and Telepresence Technologies VII*, pages 4195–4205.

[30] Fuchs, A. (2007). Dolby Does 3D.

[31] Fukuda, K., Wilcox, L. M., Allison, R. S., and Howard, I. P. (2009). A reevaluation of the tolerance to vertical misalignment in stereopsis. *Journal of vision*, 9(November):1.1–8.

[32] Gilson, S. J., Fitzgibbon, A. W., and Glennerster, A. (2008). Spatial calibration of an optical see-through head-mounted display. *Journal of neuroscience methods*, 173(1):140–6.

[33] Gilson, S. J., Fitzgibbon, A. W., and Glennerster, A. (2011). An automated calibration method for non-see-through head mounted displays. *Journal of neuroscience methods*, 199(2):328–35.

[34] Goenetxea, J., Moreno, A., Unzueta, L., Galdós, A., and Segura, Á. (2010). Interactive and stereoscopic hybrid 3D viewer of radar data with gesture recognition. In Graña, M., Corchado, E., and García-Sebastian, M., editors, *Lecture Notes in Computer Science*, volume 6076 LNAI, pages 213–220. San Sebastian.

[35] González, M., Luaces, A., Dopico, D., and Cuadrado, J. (2009). A 3D Physics-Based Hydraulic Excavator Simulator. In *ASME-AFM 2009 World Conference on Innovative Virtual Reality*, pages 75–80, Chalon-sur-Saône. ASME.

[36] Gross, H., Blechinger, F., and Achtner, B. (2008). The Human Eye. In Gross, H., editor, *Handbook of Optical Systems*, volume 4. Wiley-VCH.

[37] Heidrich, W. and Seidel, H.-P. (1998). View-independent environment maps. In *Proceedings of the ACM Siggraph/Eurographics workshop on Graphics hardware*, pages 39–ff. ACM Press.

[38] Hoffman, D. M., Girshick, A. R., and Banks, M. S. (2008). Vergence-accommodation conflicts hinder visual performance and cause visual fatigue. *Journal of Vision*, 8(3):1–30.

[39] HTC (2016). HTC Vibe at `https://www.vive.com`.

[40] Huk, T. and Floto, C. (2003). Computer-Animations In Education: The Impact Of Graphical Quality (3D/2D) and Signals. In

Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education, Chesapeake, USA.

[41] Inoue, T. and Ohzu, H. (1997). Accommodative responses to stereoscopic three-dimensional display. *Applied optics*, 36(19):4509–4515.

[42] Itoh, Y. and Klinker, G. (2014). Interaction-free calibration for optical see-through head-mounted displays based on 3D Eye localization. In *2014 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 75–82. IEEE.

[43] Itseez (2015). The OpenCV Computer Vision Library.

[44] James D. Foley, Andries van Dam, S. K. F. and Hughes, J. F. (1996). *Computer Graphics, principles and practice, 2nd ed.* Addison-Wesley.

[45] Jorke, H., Simon, A., and Fritz, M. (2008). Advanced stereo projection using interference filters. *2008 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video, 3DTV-CON 2008 Proceedings*, pages 177–180.

[46] Kellner, F., Bolte, B., Bruder, G., Rautenberg, U., Steinicke, F., Lappe, M., and Koch, R. (2012). Geometric calibration of head-mounted displays and its effects on distance estimation. *IEEE transactions on visualization and computer graphics*, 18(4):589–96.

[47] Kelly, J., Burton, M., and Pollock, B. (2013). Space perception in virtual environments: Displacement from the center of projection causes less distortion than predicted by cue-based models. *ACM Transactions on Applied Perception*, 10(4):1–23.

[48] Khronos (2008). COLLADA - Digital Asset Schema Release 1.5.0, at `https://www.khronos.org/files/collada_spec_1_5.pdf`.

[49] Kinset (2007). Kinset: the Second Life for shopping, at `http://www.readwriteweb.com/archives/kinset_like_second_life_for_shopping.php`.

[50] Kramida, G. and Varshney, A. (2015). Resolving the Vergence-Accommodation Conflict in Head Mounted Displays. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–16.

[51] Kuhl, S. A., Thompson, W. B., and Creem-regehr, S. H. (2008). HMD calibration and its effects on distance judgments. In *APGV '08 Proceedings of the 5th symposium on Applied perception in graphics and visualization*, number 212, pages 15–22. ACM.

[52] Kuhl, S. a., Thompson, W. B., and Creem-Regehr, S. H. (2009). HMD calibration and its effects on distance judgments. *ACM Transactions on Applied Perception*, 6(3):1–20.

[53] Lee, S. and Hua, H. (2013). A robust camera-based method for optical distortion calibration of head-mounted displays. In *2013 IEEE Virtual Reality (VR)*, pages 27–30. IEEE.

[54] Leigh, J., Johnson, A. E., DeFanti, T. A., Brown, M., Ali, M. D., Bailey, S., Banerjee, A., Benerjee, P., Jim, C., Curry, K., Curtis, J., Dech, F., Dodds, B., Foster, I., Fraser, S., Ganeshan, K., Glen, D., Grossman, R., Heiland, R., Hicks, J., Hudson, A. D., Imai, T., Khan, M. A., Kapoor, A., Kenyon, R. V., Kelso, J., Kriz, R., Lascara, C., Liu, X., Lin, Y., Mason, T., Millman, A., Nobuyuki, K., Park, K., Parod, B., Rajlich, P. J., Rasmussen, M., Rawlings, M., Robertson, D. H., Thongrong, S., Stein, R. J., Swartz, K., Tuecke, S., Wallach, H., Hong Yee, W., and Wheless, G. H. (1999). A review of tele-immersive applications in the CAVE research network. *Virtual Reality, 1999. Proceedings., IEEE*, 2(Vr):180.

[55] Lipman, R. and Reed, K. (2000). Using VRML in construction industry applications. *Proceedings of the fifth symposium on Vir-*

tual reality modeling language (Web3D-VRML) - VRML '00, pages 119–124.

[56] Lipton, L. (1982). *Foundations of Stereoscopic Cinema.*

[57] Lorensen, W. E. and Cline, H. E. (1987). Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169.

[58] Álvaro Segura, Delaitre, S., and Olaizola, I. G. (2012). Interactive 3D simulation tool for Energy Efficiency in Buildings. In *Proceedings of the International Workshop on Energy Efficiency for a More Sustainable World*, São Miguel, Portugal. University of Azores.

[59] Álvaro Segura, Moreno, A., Müsebeck, P., and Hambach, S. (2009). Integrating 3D Virtual Reality Simulations in Reusable e-learning Courses. In *Proceedings of e-Learning Baltics 2009*, Rostock, Germany. University of Rostock.

[60] MacIsaac, D. (2015). Google Cardboard: A virtual reality headset for $10? *The Physics Teacher*, 53(2):125.

[61] Makibuchi, N., Kato, H., and Yoneyama, A. (2013). Vision-based robust calibration for optical see-through head-mounted displays. In *Proceedings of ICIP 2013*, pages 2177–2181. IEEE Comput. Soc.

[62] Marquardt, D. W. (1962). An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441.

[63] Marques, R., Santos, L. P., Leškovský, P., and Paloc, C. (2009). GPU Ray Casting. In Coelho, A., Cláudio, A. P., Silva, F., and Gomes, A., editors, *17º Encontro Português de Computação Gráfica*, pages 83–91, Covilhã, Portugal. En Anexo.

[64] Marrin, C. (2011). *WebGL Specification at `http://www.khronos.org/webgl/`*. Khronos WebGL Working Group.

[65] Mc Lane, J., Czech, W., Yuen, D., Knox, M., Greensky, J., Kameyama, M., Wheeler, V., Panday, R., and Senshu, H. (2008). Ubiquitous interactive visualization of 3-D mantle convection through web applications using java. In *Proceedings of the International Symposium on Visual Computing*, volume 2009, pages 1011–1021, Las Vegas.

[66] Mendiburu, B. (2012). 3D cinema technology. In *Handbook of Visual Display Technology*, pages 1843–1858. Springer.

[67] MinisterioFomento (2017). Herramienta unificada LIDER-CALENER (HULC), at `http://https://www.codigotecnico.org/index.php/menu-recursos/menu-aplicaciones/282-herramienta-unificada-lider-calener.html`.

[68] Moreno, A., Galdós, A., Mujika, A., and Segura, Á. (2014). Visual analytics of multi-sensor weather information: Georeferenciation of Doppler weather radar and weather stations. In *IVAPP 2014 - Proceedings of the 5th International Conference on Information Visualization Theory and Applications*.

[69] Newman, J. (2015). The Agonizingly Slow Decline Of Adobe Flash Player. https://www.fastcompany.com/3049920/the-agonizingly-slow-decline-of-adobe-flash-player.

[70] Newman, W. M. and Sproull, R. F. (1973). *Principles of Interactive Computer Graphics*. McGraw-Hill.

[71] Ni, T., Zhang, H., Yu, C., Zhao, D., and Liu, S. (2013). Design of highly realistic virtual environment for excavator simulator. *Computers and Electrical Engineering*, 39(7):2112–2123.

[72] Ni, T., Zhao, D., and Ni, S. (2009). Visual system design for excavator simulator with deformable terrain. *2009 IEEE International Conference on Mechatronics and Automation, ICMA 2009*, pages 724–728.

[73] Nielsen, K. H. (2000). Real-Time Hardware-based Photorealistic Rendering. Master's thesis, Technical University of Denmark.

[74] Oculus (2012). Oculus Rift - Virtual Reality Headset for 3D Gaming, at `https://www.oculus.com`.

[75] Osfield, R. (2004). OpenSceneGraph, at `http://www.openscenegraph.org/`.

[76] Park, B. (2002). *Development of a Virtual Reality Excavator Simulator: a Mathematical Model of Excavator Digging and a Calculation Methodology*. PhD thesis, VirginiaTech.

[77] Paul, R. (2004). Google joins effort for 3D Web standard with new plugin, API. *Ars Technica*.

[78] Ponto, K., Gleicher, M., Radwin, R. G., and Shin, H. J. (2013). Perceptual calibration for immersive display environments. *IEEE Transactions on Visualization and Computer Graphics*, 19(4):691–700.

[79] Porter, N., Cote, A., Gifford, T., and Lam, W. (2006). Virtual Reality Welder Training. *Journal of Ship Production*, 22(3):126–138.

[80] Raggett, D. (1994). Extending WWW to support platform independent virtual reality. In *Proceedings of the 5th Joint European Networking Conference (JENC5)*, volume 464, page 2.

[81] Romary, D. (2004). Antycip SA Delivers Hydraulic Excavator Simulator. `https://web.archive.org/web/`

`20041206073039/http://www.antycip.com:80/fr/images_db/HydraulicExcavator_SIMU_UK.pdf`.

[82] Schor, C. (1999). The influence of interactions between accommodation and convergence on the lag of accommodation. *Ophthalmic & physiological optics : the journal of the British College of Ophthalmic Opticians (Optometrists)*, 19(2):134–50.

[83] Segura, Á., Arizkuren, I., Aranburu, I., and Telleria, I. (2005). High quality parametric visual product configuration systems over the web. In *Proceedings of the 10th International Conference on 3D Web Technology*, pages 159–167, Bangor.

[84] Segura, Á., Barandiaran, J., Moreno, A., Barandiaran, I., and Flórez, J. (2017). Improved virtual reality perception with calibrated stereo and variable focus for industrial use. *International Journal on Interactive Design and Manufacturing.*

[85] Segura, Á., Moreno, A., Brumetti, G., and Henn, T. (2007a). Visualization for an augmented reality construction machinery simulator. In *5th International Industrial Simulation Conference 2007, ISC 2007*, pages 324–328.

[86] Segura, Á., Moreno, A., Brunetti, G., and Henn, T. (2007b). Interaction and ergonomics issues in the development of a mixed reality construction machinery simulator for safety training. *Lecture Notes in Computer Science including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*, 4566:290–299.

[87] Segura, A., Moreno, A., Garcia, I., Aginako, N., Labayen, M., Posada, J., Aranda, J. A., and De Andoin, R. G. (2009). Visual Processing of Geographic and Environmental Information in the Basque Country: Two Basque Case Studies. In de Amicis,

R., Stojanovic, R., and Conti, G., editors, *GeoSpatial Visual Analytics: Geographical Information Processing and Visual Analytics for Environmental Security*, pages 199–207. Springer, Dordrecht, Netherlands.

[88] Shiwa, S., Omura, K., and Kishino, F. (1996). Proposal for a 3-D display with accommodative compensation: 3DDAC. *Journal of the SID*, pages 255–261.

[89] So, R. H. Y., Wong, W. S., Yip, R., Lam, A. K. C., and Ting, P. (2011). Benefits of Matching Accommodative Demands to Vergence Demands in a Binocular Head-Mounted Display: A Study on Stereo Fusion Times. *Presence: Teleoperators and Virtual Environments*, 20(6):545–558.

[90] Steinicke, F., Bruder, G., and Kuhl, S. (2011). Realistic perspective projections for virtual objects and environments. *ACM Transactions on Graphics*, 30(5):1–10.

[91] Sutherland, I. E. (1968). A head-mounted three dimensional display. *Proceedings of the AFIPS '68 (Fall, part I)*, pages 757–764.

[92] Taivalsaari, A., Mikkonen, T., Anttonen, M., and Salminen, A. (2011). The Death of Binary Software: End User Software Moves to the Web. pages 17–23.

[93] Templin, K., Didyk, P., Myszkowski, K., Hefeeda, M. M., Seidel, H.-P., and Matusik, W. (2014). Modeling and optimizing eye vergence response to stereoscopic cuts. *ACM Transactions on Graphics*, 33(4):1–8.

[94] Virtuy (2008). Virtuymall at `http://www.virtuy.com`.

[95] Vukicevic, V. (2007). Canvas 3D: GL power, web-style, at `https://web.archive.org/web/20110717224855/http://blog.vlad1.com/2007/11/26/canvas-3d-gl-power-web-style/`.

[96] W3C (1998). *Extensible Markup Language at https://www.w3.org/TR/REC-xml.*

[97] Web3D (1997). *VRML Specification ISO/IEC 14772-1 at https://web.archive.org/web/19990128143519/http://www.vrml.org:80/home.html.*

[98] Web3D (2004). *Extensible 3D (X3D) Specification ISO/IEC 19775 at http://www.web3d.org/documents/specifications/19775-1/V3.0/index.html.*

[99] Weiser, M. (1991). The Computer for the 21st Century. *Scientific American*, pages 94–104.

[100] Westermann, R., Kobbelt, L., and Ertl, T. (1999). Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces. *The Visual Computer*, 15(2):100–111.

[101] Westermann, R. and Sevenich, B. (2001). Accelerated Volume Raycasting Using Texture Mapping. In *Proceedings of the Conference on Visualization '01*, VIS '01, pages 271–278, Washington, DC, USA. IEEE Computer Society.

[102] White, S. A., Prachyabrued, M., Chambers, T. L., Borst, C. W., and Reiners, D. (2011). Low-cost simulated MIG welding for advancement in technical training. *Virtual Reality*, 15(1):69–81.

[103] Yang, S. and Sheedy, J. E. (2011). Effects of Vergence and Accommodative Responses on Viewer's Comfort in Viewing 3D Stimuli. In Woods, A. J., Holliman, N. S., and Dodgson, N. A., editors, *Proc. SPIE 7863, Stereoscopic Displays and Applications*, pages 78630Q–78630Q–13.

[104] Zuiderveld, K. J., Koning, A. H. J., and Viergever, M. A. (1992). Acceleration of ray-casting using 3-D distance transforms. In *Proc.SPIE*, volume 1808, pages 1808– 1820.