Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Omnidirectional Stereo Rendering of Virtual Environments

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Lukas Meindl
Matrikelnummer 1028160

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Mitwirkung: Dipl.-Ing. Dr. Anton Fuhrmann

Wien, 29.09.2015

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Omnidirectional Stereo Rendering of Virtual Environments

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Lukas Meindl
Registration Number 1028160

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Assistance: Dipl.-Ing. Dr. Anton Fuhrmann

Vienna, 29.09.2015

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Lukas Meindl
Bergenstammgasse 8 / 9, 1130 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Acknowledgements

I would like to thank my supervisor, Anton Fuhrmann, for the interesting conversations we had about the thesis topic and for all the input he provided. I would also like to thank my main supervisor, Michael Wimmer, for the valuable feedback given on the thesis.

I am grateful to my parents, grandparents and my uncle for all the support they gave me during my entire studies. I also wish to thank Ilana for her support and for being there.

I wrote this thesis as a diploma student at the VRVis Research Center in Vienna and I thank the organisation for giving me this opportunity. The competence center VRVis is funded by BMVIT, BMWFW, and Vienna Business Agency within the scope of COMET – Competence Centers for Excellent Technologies, project #843272. The program COMET is managed by FFG.

Abstract

In this thesis we discuss the use of omnidirectional stereo (omnistereo) rendering of virtual environments. We present an artefact-free technique to render omnistereo images for the CAVE in real time using the modern rendering pipeline and GPU-based tessellation.

Depth perception in stereoscopic images is enabled through the horizontal disparities seen by the left and right eye. Conventional stereoscopic rendering, using off-axis or toe-in projections, provides correct depth cues in the entire field of view (FOV) for a single view-direction. Omnistereo panorama images, created from captures of the real world, provide stereo depth cues in all view direction. This concept has been adopted for rendering, as several techniques generating omnistereo images based on virtual environments have been presented. This is especially relevant in the context of surround-screen displays, as stereo depth can be provided for all view directions in a 360° panorama simultaneously for upright positioned viewers. Omnistereo rendering also lifts the need for view-direction tracking, since the projection is independent of the view direction, unlike stereoscopic projections. However, omnistereo images only provide correct depth cues in the center of the FOV. Stereo disparity distortion errors occur in the periphery of the view and worsen with distance from the center of the view. Nevertheless, due to a number of properties of the human visual system, these errors are not necessarily noticeable.

We improved the existing object-warp based omnistereo rendering technique for CAVE display systems by preceding it with screen-space adaptive tessellation methods. Our improved technique creates images without perceivable artefacts and runs on the GPU at real-time frame rates. The artefacts produced by the original technique without tessellation are described by us. Tessellation is used to remedy edge curvature and texture interpolation artefacts occurring at large polygons, due to the non-linearity of the omnistereo perspective. The original approach is based on off-axis projections. We showed that on-axis projections can be used as basis as well, leading to identical images. In addition, we created a technique to efficiently render omnistereo skyboxes for the CAVE using a pre-tessellated full-screen mesh. We implemented the techniques as part of an application for a three-walled CAVE in the VRVis research center and compared them.

Kurzfassung

Diese Arbeit beschäftigt sich mit omnidirektionalem Stereo (Omnistereo) Rendering von virtuellen Umgebungen. Wir schlagen eine Technik vor mit der Artefakt-freie, anhand der modernen Rendering-Pipeline und GPU-basierter Tessellierung, Omnistereo Bilder für CAVE Systeme in Echtzeit erzeugt werden können.

Raumwahrnehmung in stereoskopischen Bildern wird durch horizontale Disparitäten zwischen dem linken und rechten Auge erzeugt. Konventionelles Stereo-Rendering, auf der Basis von Off-Axis oder Toe-In Projektionen, liefert korrekte Hinweisreize für die Tiefe in dem gesamten Blickfeld für eine gegebene Blickrichtung. Omnistereo Panorama Bilder, welche durch Aufnahmen der realen Welt erzeugt werden, liefern hingegen Hinweisreize für die Tiefe in alle Blickrichtungen. Dieses Konzept wurde für Rendering angepasst, wodurch mehrere Techniken entstanden sind um Omnistereo Bilder aus virtuellen Umgebungen zu erstellen. Diese Methode ist für Bildschirmsysteme die den Nutzer komplett umgeben von Relevanz, da hierbei eine gleichzeitige Raumwahrnehmung im gesamten 360° Panorama für aufrecht positionierte Beobachter erzeugt werden kann. Omnistereo Rendering kann zudem ohne Kopfrotations-Tracking verwendet werden, da die Projektion unabhängig von der Blickrichtung ist, was bei konventionellen Stereo Projektionen nicht der Fall ist. Omnistereo Projektionen liefern hingegen aber nur im Zentrum der jeweiligen Blickrichtung korrekte Hinweisreize für die Tiefe. Stereo-Disparitätsverzerrungsfehler entstehen zunehmend in der Peripherie der Blickrichtung. Eigenschaften des menschlichen visuellen Systems machen diese Fehler aber kaum bemerkbar.

Wir haben eine bestehende Object-Warping Omnistereo Rendering Technik für CAVEs verbessert, indem wir sie mit Screen-Space adaptiven Tessellierungsmethoden erweitert haben. Die Technik erzeugt Bilder ohne die zuvor wahrnehmbaren Artefakte und läuft auf der GPU in Echtzeit. Die Artefakte der ursprünglichen Technik ohne der Tessellierung wurden durch uns beschrieben. Tessellierung verringert die Kantenkrümmungs- und Texturinterpolationsartefakte, die bei großen Polygonen, aufgrund der Non-Linearität der Omnistereo Perspektive, entstehen. Der originale Ansatz der Technik basiert auf Off-Axis Projektionen. Wir haben gezeigt, dass On-Axis Projektionen auch als Basis verwendet werden können und die resultierenden Omnistereo Bilder ident sind. Außerdem haben wir eine Methode entwickelt Omnistereo Skyboxes in der CAVE mittels eines vortessellierten Vollbild-Meshes effizient darzustellen. Wir haben die Omnistereo Techniken als Teil einer Anwendung für den drei-wandigen CAVE des VRVis Forschungszentrum umgesetzt und aufgrund dieser Anwendung die Techniken miteinander verglichen.

Contents

1 Introduction **1**
1.1 Problem Statement 4
1.2 Contributions 5
1.3 Overview 6

2 Background **7**
2.1 Depth Cues 7
2.2 Immersion 8
2.3 Stereoscopic Rendering 10
2.4 Cybersickness 16
2.5 Intrusiveness 18
2.6 Perception of Omnistereo in the Human Visual System 18
2.7 Tessellation Shaders 22

3 Motivation **25**
3.1 Advantages of Omnidirectional Stereo 25
3.2 Research Aim 27

4 Related Work **31**
4.1 CAVE 31
4.2 Omnidirectional Stereo Imaging 35
4.3 Rendering Stereo Panoramas 36
4.4 Multiple-Center-Of-Projection Images 37
4.5 Omnistereo Rendering in a CAVE 39
4.6 Omnistereo Rendering in Curved Display Systems 45
4.7 Omnistereo Rendering in Domes 54
4.8 Multi-Viewer Stereo Displays 61

5 Object-Warping Based Omnistereo Projection **65**
5.1 On-Axis Omnistereo Object-Warping 65
5.2 Off-Axis Omnistereo Object-Warping 70
5.3 Discussion of Vertex-Shader Based Omnistereo Rendering 71

6 Tessellation-Based Omnistereo Projection **75**

6.1 Remedying Non-Linear Projection Artefacts 75
6.2 Adaptive Tessellation for Omnistereo Rendering 78
6.3 Omnistereo Skybox for the CAVE 81

7 Implementation and Results **85**
7.1 Setup 85
7.2 Application 87
7.3 Results 88

8 Conclusion **95**
8.1 Future Work 96

Bibliography **97**

CHAPTER 1

Introduction

In recent years, *virtual reality* (VR) technology has once again come to public attention, mainly due to the success of VR devices in home entertainment systems and the increased availability and affordability of these devices. Given *Nintendo's Wii Remote* controller and Wii balance board for the *Wii*, *Sony's PlayStation Move* controller for *PlayStation 3* and *Microsoft's Kinect* for the *Xbox 360*, all consoles of the seventh generation of video game consoles were equipped with VR input devices. New *head-mounted display* (HMD) devices have come into development, such as the *Oculus Rift*, *Sony's Project Morpheus* and *Samsung Gear VR*, which are designed to be inexpensive and for home use [1].

Huge advancements have also been accomplished regarding (3D) output devices, such as 3D television displays, 3D monitors and 3D projectors, mainly driven by the rapid spread of these technologies at cinemas and in homes. As a result of these developments, the output devices have also become more affordable. Figure 1.1 shows how a scene in reality is captured using a *3D camera* by storing separate images for the left and the right eye respectively. The captured images can, for example, be displayed with the mentioned devices. Passive or active 3D glasses are used in conjunction with these devices to view separate images with the left and right eye. These stereoscopic displays, as well as HMDs, provide greater immersion than monoscopic displays, by enabling binocular cues to the human visual system, thus offering enhanced depth perception. This principle, which is commonly used these days, for example in 3D cinemas and in 3D television, can also be used to view entirely virtual environments instead of captured real-world scenes. This is accomplished by *stereoscopic rendering*, which is the method of rendering left and right-eye images from two different perspectives, based on the left and right eye's positions in the virtual scene.

3D displays and 3D projectors, as well as the displays used in 3D theaters, only offer a correct stereoscopic effect if the viewer is located approximately central to the screen and at a certain distance. While this is an acceptable drawback when using them for entertainment purposes, it makes them not suitable for VR applications. In order to

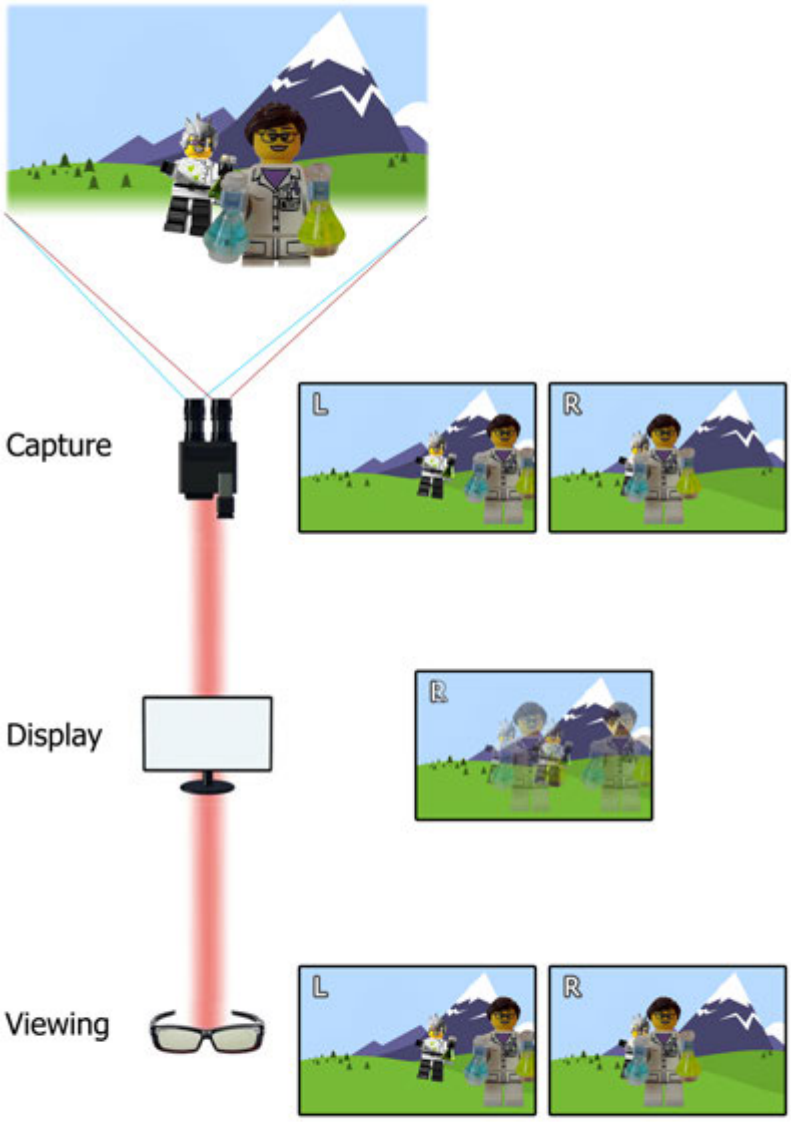

Figure 1.1: The process of capturing a scene in reality, using a 3D camera that records images for the left and right eye, to viewing the final result on a 3D display using 3D glasses.

always provide a correct view for a single viewer even if they are moving, tracking of the head position and view direction is required.

VR is researched and used in the fields of health care, training, education, automotive industry, virtual heritage, virtual archaeology, engineering, driving simulation, flight simulation and product design. For these purposes, regular commercial off-the-shelf 3D displays and 3D projectors may only provide an insufficient VR experience on their own due to the limited FOV and movement constraints. While HMDs offer free movement and often a sufficiently large field of view (FOV) at low cost, they come with a number of drawbacks, such as movement constraints, discomfort from wearing the HMD, discomfort from the proximity of the screen to the eyes and a generally higher likeliness to cause motion sickness. In addition to that, they eliminate the possibility for multiple users to share the same display and view, hinder direct communication and occlude the direct vision on other participants. Surround-screen displays, such as the CAVE (*CAVE Automatic Virtual Environment*) [2], consist of multiple 3D projectors or screens in order to provide a very large or complete FOV. Such displays suffer less from the mentioned issues but are more expensive to acquire and maintain and require more space.

In order to allow multiple users to observe a 3D object or virtual environment from multiple view points and directions simultaneously, *multi-view* stereoscopic displays have been developed. *Autostereoscopic displays*, for example, allow this without requiring any special headgear or glasses for the viewers [3]. *Lenticular displays* are autostereoscopic displays that use lenses to direct the view to a specific sub-zone of a 2D image depending on the view angle. However, this technique comes at the cost of a reduced image resolution, which is reduced further with each additionally supported view angle. Another type of autostereoscopic display, the *volumetric display*, visualises voxel data and is also viewable from multiple directions at the same time. However, the resolution of the 3D image is limited by the bandwidth that is needed to transfer the voxel data for each frame. The scene or object to be displayed also has to be modelled or scanned as voxel data. Both of these display technologies can only be used to showcase virtual objects and are not suitable for surrounding the viewer with a virtual environment. However, many VR installations are made to show large virtual scenes or immerse the user into a virtual world. For those purpose, HMDs and surround-screen displays are better suited.

HMDs and stereoscopic surround-screen displays incorporate head-tracking, which requires the entire scene to be re-rendered on changes of head position and orientation. Tracking requires extra tracking hardware and additional computations to be performed in the software before every rendered frame. Since tracking hardware is not perfectly accurate it can cause jitter. Also, since the tracking hardware and the processing of the data takes time, delays may be introduced. These two issues can cause discomfort and motion sickness. If more than one viewer is inside a surround-screen display, only the viewer who is being head-tracked will be able to view the environment correctly. Other viewers will perceive reduced, negated or reversed depth for the objects, as well as distortion errors to a varying degree based on their view direction. If there is only a single viewer who is not being tracked, the same issues arise. Therefore, being able to provide a single user, who is standing still or remains seated, with a stereoscopic view

in all view directions without any head-tracking can be of advantage.

1.1 Problem Statement

Rendering scenes stereoscopically requires knowing the viewer's position and orientation accurately, so that the stereo pairs can be rendered correctly for this viewpoint. Head-tracking is typically used in this context to determine the orientation and position of the user's head before rendering a frame. These calculations have to performed fast enough to support real-time rendering frame rates. Additionally, the head-tracking can cause jitter or introduce delays, both of which may aggravate or induce cybersickness. Creating multi-viewer VR setup that allow users to share a display is not possible with regular stereoscopic rendering. Such setups may be designed to allow users to move and look wherever they want, but stereoscopic rendering is unable to create images with satisfactory depth perception for all viewers under these circumstances. In order to provide satisfactory stereo depth cues in all directions, omnidirectional stereoscopy, also called *omnistereo*, rendering has to be used. This technique creates valid stereo images featuring correct binocular disparity in the center of the view for each direction a user might be facing, thus lifting the necessity to perform any kind of view-direction tracking. The way omnistereo images are rendered depends on the used display system and the type of screens in use. Our goal is to create an application that renders omnistereo images at real-time frame rates for the CAVE display system at the VRVis research center. A technique for rendering omnistereo projections for a CAVE display system, which is based on off-axis projections and vertex shaders, has been proposed by Simon et al. [4]. However, this technique is not independent of mesh density and can thus lead to artefacts when polygons appear large on the screen due to the non-linearity of the omnistereo perspective. Another proposed solution for omnistereo rendering in CAVEs is to render multiple views per screen [4] to approximate the projection. Vertical slices are rendered for the vertical screens of the CAVE, which are then composed into the final image. This solution is less efficient than the shader-based solution, since it requires additional geometry processing and multiple viewport rendering. Our goal is to render omnistereo images with low computational overhead compared to regular stereoscopic rendering using a technique that can be combined with regular shading and rendering. For this purpose we want to improve the vertex-shader based omnistereo rendering technique by Simon et al. [4] by remedying the occurring artefacts. These artefacts are a result of the linear interpolation and linear rasterisation performed by the programmable graphics pipeline of OpenGL and Direct3D, which was optimised and built mainly with linear perspectives in mind. Shader-based tessellation techniques have, for example, been used in the context of shadow mapping techniques [5, 6] in order to solve artefacts occurring in non-linear projections. Therefore, we intend to use tessellation techniques to resolve the aforementioned artefacts.

1.2 Contributions

The following research contributions were made in this thesis:

- An omnistereo projection for rendering omnistereo images based on on-axis projection is proposed, which lifts the necessity to recalculate the off-axis projections on every head-position change and which works analogous to the off-axis based object warping and is shown to produce equivalent omnistereo images.

- As a solution to the errors that occur in the object-warping technique due to the linear interpolation performed by the rasterisation and shader attribute interpolation, we propose *tessellation-based omnistereo projection*, which precedes object warping with a tessellation step. Two screen-space adaptive tessellation modes, a naïve approach and a sphere-based approach [7], are used in conjunction with on-axis and off-axis based object warping, which we compared with each other. We show that the technique remedies the artefacts successfully.

- We propose a simple and efficient technique to render a skybox stored in a cubemap texture in omnistereo, by using a full-screen quad and a fragment shader.

Further contributions provided in this thesis are:

- The off-axis based object-warping technique for omnistereo rendering, as originally suggested by Simon et al. [4], is discussed in detail, its geometric background is illustrated and problems that can occur, and have not been pointed out in previous work, are explained and displayed.

- A frame-rate comparison between the object-warping techniques including two different scenes with three viewpoints each and varying maximum tessellation factors set. This comparison shows that the performance of the tessellation-based omnistereo projection depends on the scene complexity but is suitable for real-time rendering and can feature a very low performance loss over regular stereo rendering.

- The design and implementation of an application featuring the omnistereo rendering techniques and rendering images for the three-walled CAVE of the VRVis research company. The application allows rendering a scene using different techniques, which can be changed at run-time for quick comparison. The modes include monoscopic and stereoscopic rendering, as well as omnistereo rendering based on an off-axis or and on-axis perspective projection, each with the option to turn tessellation on or off and to change the tessellation configuration. The application features a graphical user interface that allows configuring the rendering modes. The viewpoint inside the CAVE can either be set manually in the interface or be specified via network from another system that performs tracking, the camera position and orientation can be controlled using a gamepad.

1.3 Overview

In Chapter 2, information about stereo rendering is given and aspects of VR and the human visual system relevant in the context of omnistereo rendering are discussed. Based on the information provided in that chapter, Chapter 3 provides the motivation behind omnistereo rendering. Chapter 4 summarises the state of the art of the CAVE display system and omnistereo techniques. Omnistereo imaging based on photographs or videos and omnistereo rendering techniques used in different display system, including curved screen display systems and the CAVE display system, are mentioned. The object warping omnistereo technique is described in Chapter 5 in detail and the problems occurring with it are discussed. Our proposed improvements lead to the tessellation-based omnistereo projection technique described in Chapter 6. In this chapter we also suggest a technique for efficiently rendering omnistereo skyboxes. Chapter 7 includes the description of the application we created for rendering omnistereo techniques in the VRVis CAVE and the results of our work. Finally, in Chapter 8 we conclude the thesis and provide suggestions for future work.

CHAPTER 2

Background

2.1 Depth Cues

In order to give a user a sense of three-dimensionality for a virtual scene, VR applications make use of several depth cues for the human visual system. The following depth cues can be present in a monoscopic view [8, 2, 9]:

- Occlusion - The object occluding another object has be to be the closer one out of the two.
- Perspective - The perspective projection makes object appear smaller the further away they are. Additionally, parallel lines appear to intersect in the vanishing point of the scene.
- Height in the FOV - Objects that are perceived as resting on a surface below the horizon are perceived as being more distant if they are located higher in the FOV.
- Atmospheric - Fog, haze and other atmospheric effects can help determine the distance of an object.
- Texture and detail - Close objects typically appear in greater detail, distant objects are less detailed.
- Lighting and shadows - The way a surface reflects light and the way shadows are cast by objects into the scene help determine the shape and position of objects. Shadows are also a form of occlusion.
- Previous knowledge - Knowledge of the size of an object helps to determine the absolute depth of the object.

- Relative motion - Head motion gives hints about the relative distance of an object against its background. Objects further away appear to move more slowly than objects closer to the viewer.

Additional depth cues not present in 2D images are:

- Stereopsis - The difference between the images cast on our left and right eye's retina. This difference occurs due to the horizontal separation of the eyes - the interocular distance.
- Convergence - The eyes rotate towards the center of interest
- Accommodation - To focus at a particular depth, muscle tensions is needed to change the focal length of the eye lens

These depth cues help the human visual system estimate the distance of an object to the viewer. *Accommodation* is a depth cue that is technically difficult to provide in VR applications and therefore rarely included. Stereopsis is one of the strongest depth cues in general and the strongest one for distances up to 10 m. Stereopsis is enabled through horizontal disparity, also known as binocular disparity. The binocular disparity is the positional distance of a point in space projected onto the retina. At large distances the binocular disparity becomes too small to be an effective depth cue.

Stereoscopy is frequently used in VR to enhance depth perception by providing convergence and stereopsis depth cues. This is accomplished by rendering separate images for the user's left and right eye, each representing the respective eye's view in the VR. The two images created in the processes are called *stereo pairs*. Each image is rendered independently from separate positions, which represent the respective physical position of each eye in the virtual scene. Figure 2.1 shows how the parallax between points projected on the screen is related to the position of the point in 3D. Points behind the screen have positive, points in front of it have negative and points on the screen have exactly zero parallax. In monoscopically rendered images the projected points have zero parallax and are therefore all perceived to be located on the screen in 3D. In stereo images the virtual objects have varying parallaxes depending on their distance to the virtual eyes. The depth, and therefore the position of objects in 3D, can thereby be reconstructed in the human visual system. The parallax between stereo images for a given point in 3D also depends on the distance between the viewer and the screen and on the eye separation distance.

2.2 Immersion

Immersion is a central concept in VR. Immersion is the extent to which the feeling or belief exists that one has left the real world and is "present" in the virtual world [10]. The immersion in a VR depends on its ability to replace real world sensations with virtual ones and thus deliver a believable illusion to the user. Immersion is important for creating a credible VE, as it describes the degree to which the user *suspends disbelief*.

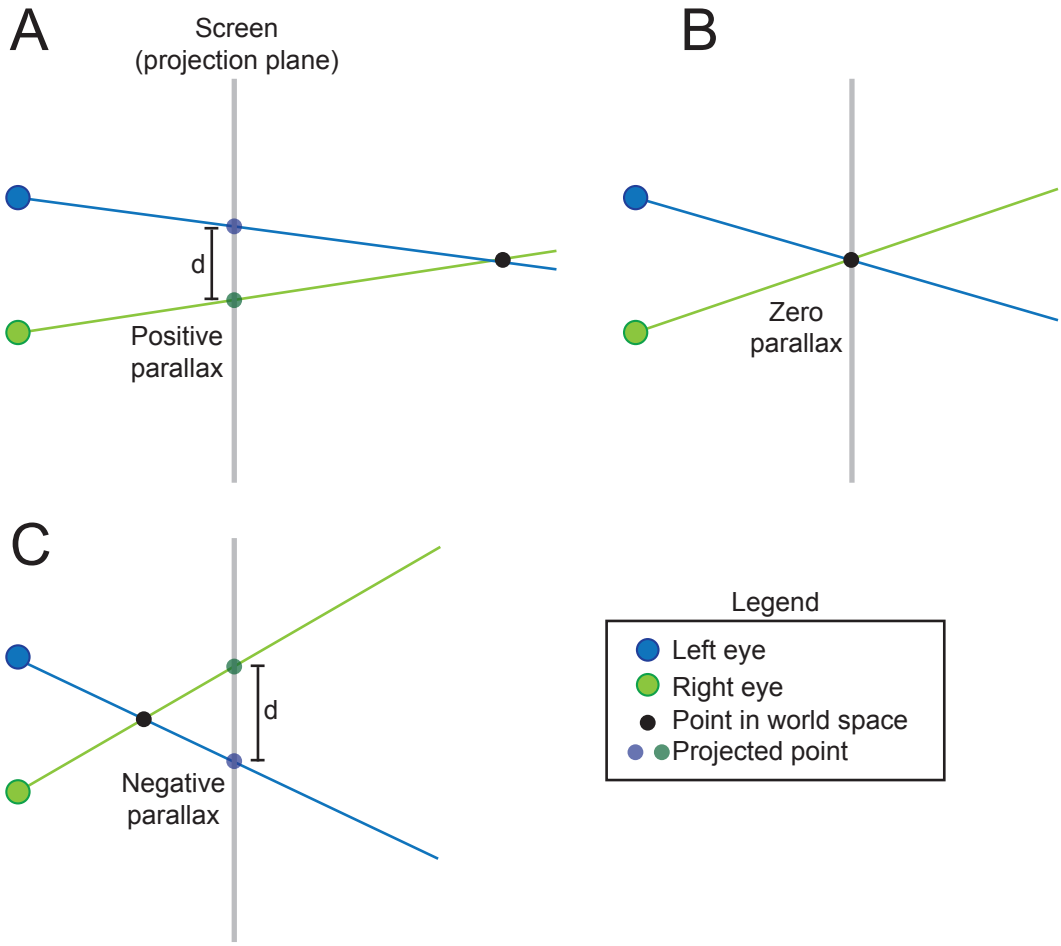

Figure 2.1: The images show points in a 3D scene being projected on the screen for the left and the right eye and their resulting parallaxes between the eye's images.

Suspension of disbelief describes the ability to suspend judgement of the implausibility of any media, a simulation or a narrative without having to force oneself. This idea is very fundamental to VR, because in the most optimal case for a VR, its user(s) will accept the artificial environment presented by the software as a real environment. Another important role in VR is played by *presence*, which describes the psychological perception of existing in the VR. This is primarily thought to be the by-product of the VR's immersive properties. Presence depends both on external and internal factors. This means, that in order to feel "present", the VR has to be immersive enough to disallow distractions or psychological ties to the external world from occurring.

Factors that support the presence are the length of exposure, social factors, system factors, pictorial realism, ease of interaction and internal factors. System factors that positively contribute to presence are, for example, stereopsis, a large FOV and

sufficiently high update rates. The FOV describes the area in which the viewer can perceive an environment from one position and orientation, defined by the angular extents of this area in degrees. For example, a display with 360° horizontal FOV completely surrounds its viewers horizontally, allowing any direction at all times to be seen using head-rotations. The human visual system only allows perceiving a FOV limited to approximately 180° horizontally and 120° vertically. Most animals also can not perceive the entire FOV, an exception being the hammer shark, which has a 360° vertical FOV. In order to convey the impression of being surrounded by a virtual environment (VE), the viewer's FOV should be covered by the environment as much as possible. A surround-screen display, such as a CAVE [2] with six screens, may offer an unlimited FOV, both horizontally and vertically, whereas HMDs typically provide a FOV of around 90 to 110°. While the entire surrounding environment can be seen with HMDs using head-rotations, only this limited FOV is visible to the viewer at a single moment.

Common consumer-grade output devices, such as computer monitors and television display devices, have a variable and comparatively low FOV. For this reason, the aforementioned displays are generally treated like a window into the virtual world. The FOV is variable because it depends on the viewer's position; increasing the distance between the viewer and the display will decrease it.

As discussed before, immersion can be broken by real world distractions or by psychological ties to the external world. A large FOV can create the impression of being surrounded by the VE and, in addition to that, allows for peripheral vision. An experiment by Prothero and Hoffman [11] was conducted to investigate if a wider FOV would encourage a higher sense of presence. The experiment was conducted with HMD's with 40° vertical FOV and 105° horizontal FOV. The participants of the experiment experienced a VR with either unrestricted FOV or a FOV limited by tanning googles. The results indicated that a higher degree of presence is reached with a higher FOV. Output displays, which offer a high FOV, such as HMDs, are therefore more adequate for VR setups. CAVEs, domes and cylindrical cinema installations can even allow fully surrounding the user, providing a horizontal and vertical FOV of 360°. Such setups create a *panorama* that can only be fully viewed by using head rotations, unlike non-surrounding low-FOV displays.

According to Peleg et al. [12] the ultimate immersive visual environment should provide three elements:

1. Binocular stereo vision
2. A complete 360° degrees view, allowing the viewer to look in any desired direction
3. Free movement

2.3 Stereoscopic Rendering

Stereo image pairs can, for example, be created with ray-tracing rendering techniques [13]. However, ray-tracing is not popular for use in VR, as it is computationally significantly

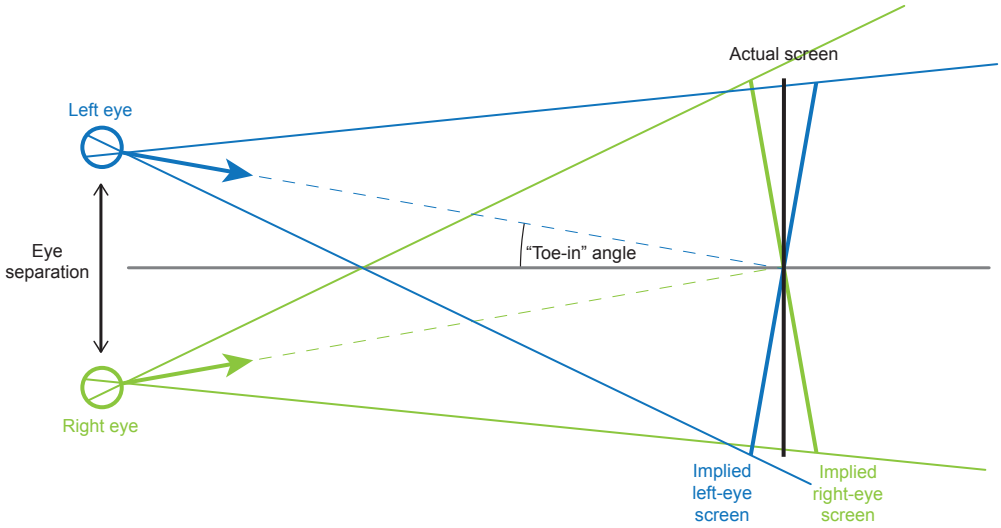

Figure 2.2: The "toe-in" method used to create stereoscopic images. The eye separation distance in the image is exaggerated for better illustration. The cameras are rotated in a way that their direction vectors meet at the focal length. The figure shows the rotated frustums and the assumed projection screens. The implied screens do not match the actual screen, resulting in incorrect parallax.

more demanding than rendering based on rasterisation and typically too slow for VR applications, which need to run at least at interactive frame rates. For planar screens, off-axis stereo projections and toe-in stereo are therefore typically used.

Toe-In Stereo

In order to create stereoscopic images, the "toe-in" method is often used [8]. In this method the cameras (real film cameras or virtual cameras as in rendering) are rotated so that the direction vectors meet at at the focal length, as shown in figure 2.2. It is a popular method for low-budget filming because offset/parallel cameras are more expensive and less common. It can also be used within rendering, for example when features required for off-axis rendering are not available in the rendering engine or library used. The images created with the "toe-in" method may appear stereoscopic, but are technically incorrect and may cause discomfort to the viewer. A vertical parallax is introduced by the method, which increases with the distance from the image center. Correctly rendered left and right images, however, do only have a horizontal disparity. Another way to look at the problem is to see it as a distortion in the form of a keystoning effect. Figure 2.3 shows the incorrect toe-in stereo image next to a correct one, both rendered as anaglyph images.

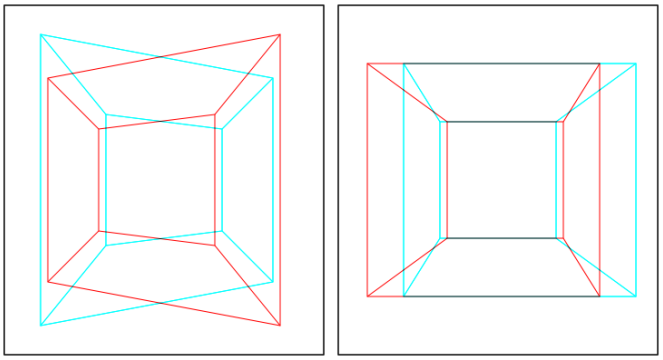

Figure 2.3: The left stereo image shows the keystoning effect caused by a toe-in stereo projection. The projections for the left and the right eye cannot be merged into a single 3D object. The right stereo image shows the correct projection. Image by Kreylos [14]

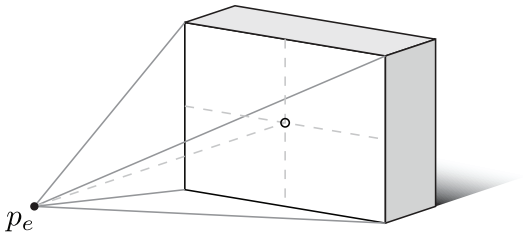

Figure 2.4: An on-axis perspective projection, with the position of the camera $_e$ projecting the screen-space origin to the centre of the screen. Image taken from Kooima [15]

Off-Axis Projection

An on-axis projection matrix projects the screen-space origin to the center of the screen [15]. The resulting image is correctly projected for viewpoints located vertically and horizontally centered relative to the display. This type of perspective projection is, for example, used to render three-dimensional (3D) video games, simulations and other 3D applications on consumer-grade display devices. In these setups no tracking of the viewer's position takes place, so that the perspective cannot be adapted to it. Therefore the real position and the assumed position typically do no match match perfectly for the viewer, but this is acceptable for most type applications. Off-axis projections project the VE correctly for viewpoints that are not centered. This can be used for monoscopic rendering and is crucial for correct stereoscopic rendering. Figures 2.4 and 2.5 show the view frustums of an on-axis and off-axis projection, respectively.

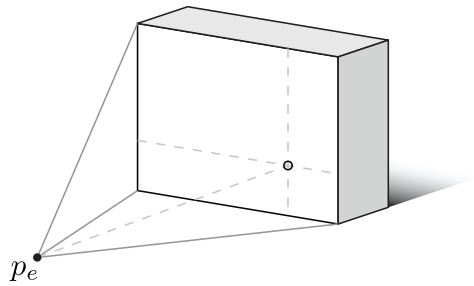

Figure 2.5: An off-axis perspective projection, with the eye position $_e$ and screen-space origin falling off center. Image taken from Kooima [15]

When rendering stereo pairs, the left and right eye positions are used to determine how the frustum of the off-axis projection has to be aligned. Unlike with on-axis projections, the frustums of off-axis projections are asymmetric and skewed. Figure 2.6 shows the frustums of the left and right eye's off-axis projection from top-down view. Using such projections is possible with OpenGL's and Direct3D's modern programmable shader pipelines, as well as with the deprecated OpenGL fixed-function pipeline. Kooima [15] discussed the mathematical computations required to create these matrices and shows how they can be used together with the fixed-function pipeline of OpenGL. The example can, with little effort, be adopted to be used in OpenGL's or Direct3D's modern programmable shader pipeline or any other rendering library or rendering engine supporting programmable shaders. The off-axis and on-axis perspective projections are planar projections and therefore suited for flat screen surfaces, which is equivalent to the projection plane in computer graphics terminology.

Figure 2.7 shows how a virtual object positioned between the viewer and the screen has to be projected onto the screen, in order to give a correct impression of depth using binocular disparity. If on-axis projections were used here instead, the resulting projections would not be in accordance to the screen both eyes are looking at and would result in a keystoning effect. When rendering stereoscopic images for VR scenarios, for example in a CAVE environment or when using a dome or a sphere as projection surface, the head of the user has to be tracked to determine the view direction and position. Figure 2.8 shows different view directions for the eye pairs. The illusion of three-dimensional depth can only be perceived correctly if the viewer's head orientation (view direction) matches the one for which the stereo pairs were rendered. In the direction perpendicular to the view direction, parallel to the line connecting the eye positions, the rendered images have no horizontal disparity, which means that there is no perceivable stereopsis. The disparity is reduced with increasing deviance from the original view direction. For this reason, this method of stereoscopy is not applicable for multiple users when using shared screens.

When rendering stereoscopically across multiple adjacent screens, for example as

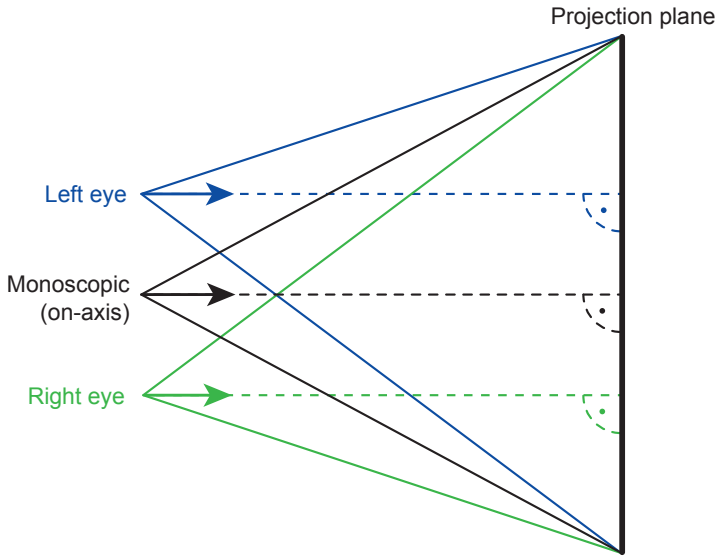

Figure 2.6: The image shows the view frustums of on-axis and off-axis (left and right eye) projections as seen from above. The mono view is rendered with an on-axis projection, while the right and the left eye's views are rendered with off-axis projections.

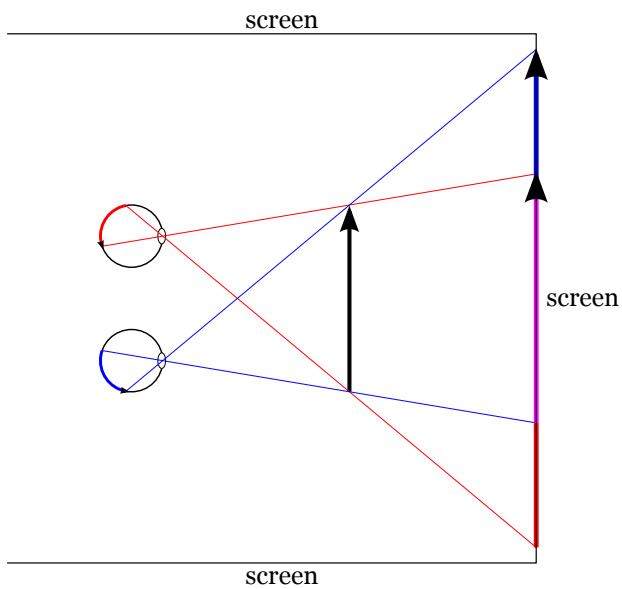

Figure 2.7: The image shows how an object has to be projected onto the projection walls to create the binocular disparity. Image used with permission from Kreylos [16]

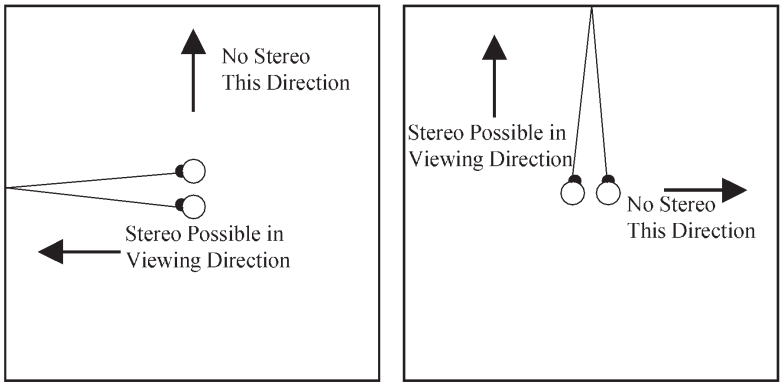

Figure 2.8: Stereo viewing is not possible in the directions perpendicular to the eye axis when using the regular perspective projection. Image taken from Peleg et al. [12]

done in CAVEs, it is necessary to use the off-axis method instead of the incorrect toe-in one in order to create visually correct images. The latter causes apparent discontinuities between the projected images on adjacent screens, which is a highly undesirable effect as it can result in gaps between the screens or in image points appearing in either of the screens. This effect also occurs if the head position is centered inside the CAVE, since the projection origins are the eye positions and not the head positions. The eye positions are offset from the head position and therefore not located at the center. Because of this it is never viable to render stereoscopically across adjacent screens with the toe-in method. Figure 2.9 shows a three-walled CAVE and the correct off-axis view frustums for a given eye position. The images projected onto the screens in this case merge without visible seams and are completely correct. For monoscopic rendering in a CAVE, if the assumed head position is not located at the center of the room, an off-axis projection is required for the same reason.

In theaters exhibiting 3D films, the viewers' heads are assumed to be positioned upright. When using linearly polarised glasses, tilting the head sideways will cause the right-eye and left-eye images to be mixed for each eye, the amount of which depends on the degree to which the head is tilted. When using circularly polarised glasses or shutter glasses, the images are shown correctly separated for each eye, but tilting the head 90° sideways removes the stereo effect and tilting the head 180° inverts the stereo effect, effectively swapping the view of the left and right eye. A CAVE, however, may contain ceiling or floor projections, which make head tilting support inevitable. Additionally, unlike in 3D film theaters, viewers can move around and significantly change their orientation. If the positions of the viewer's eyes are known, it is possible to render stereo images for any arbitrary head rotation by using off-axis projections.

When rendering stereoscopically, special care has to be taken regarding screen-space post-processing effects [17, 18]. Such effects, which are often used in monoscopic rendering for video games or VR applications are, for example, depth of field, screen-space ambient occlusion, bloom, lens flares and tone mapping. However, in stereo rendering

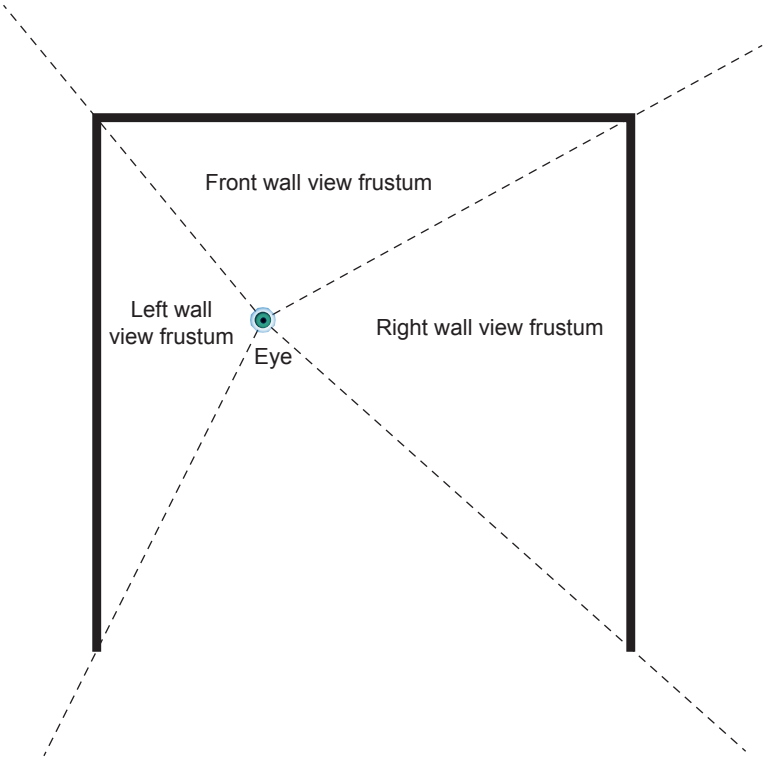

Figure 2.9: Overhead view of a CAVE setup with three walls. The view frustums of the off-axis projection are shown for a single eye position. Since no seams exist between the frustums, the transitions between the projected images are continuous.

they may look incorrect or even cause discomfort. By applying consistently for both eyes, the problem may be solved in some of the cases. Billboard rendering, which is sometimes also used for particle effects, is also affected, as the billboards are perceived flat in stereo.

2.4 Cybersickness

A problem with VR applications is that they frequently cause symptoms similar to those of motion sickness among its users [19]. These effects are exhibited both during and after the VR experience and are referred to as *cybersickness*.

Cybersickness may include one or multiple of following symptoms:

- Nausea

- Headache
- Eye strain
- Vertigo
- Disorientation

In some cases sweating, vomiting, pallor, dryness of mouth, fullness of stomach and ataxia may occur. Motion sickness and cybersickness cause the same symptoms but are not equivalent. The former can be induced by vestibular simulation alone, while visual stimulation may contribute to it, however, cybersickness can be induced without any vestibular simulation and is described as a polygenic sickness without one exact cause. The effects can last for hours after having been exposed to the VR and, additionally, after-effects can develop during this time [20]. For this reason, some aircraft bases in the USA require its pilots who used a flight simulator to not fly an aircraft in the following 12-24 hours and many VR entertainment centers require users not to drive a car in the 30-45 minutes following the exposure. The longer the exposure to a VE takes place, the longer a user needs to recover from the cybersickness effects and after-effects. People try to avoid feelings of sickness, which means that a VR system that causes cybersickness is more likely to be avoided as a whole. In addition to that, cybersickness distracts the VR users, making it more difficult to follow the original purpose of the VR application or complete tasks. These effects, and the risk of users being negatively affected in driving/flying a vehicle, are good reasons for VR developers mitigating cybersickness. Some of the factors contributing to cybersickness depend on the individual, such as gender, age, illness, whether they are sitting or standing and whether they are in control of the simulation.

There are also a number of factors related to the VR's technology and display:

- Position-tracking of the user's head or even eyes is used in VR to be able to determine the user's perspective in the VE and render it accordingly. In addition to that, some VR installations that try to display an accurate representation of the user in the virtual world, also track the user's limbs' physical positions. However, position trackers are not 100 percent accurate and have a tendency to submit unstable information, causing *jitter*. This jitter introduces constant uncontrollable movement of the user's view, which causes in dizziness and concentration loss in VR setups.
- *Lag* describes the delay between performing an action and seeing the result occurring in the VR. One very common source of lag is created by the time it takes to receive and process information from the head-tracker. When a user rotates his or her head, this lag will result in a later update of the user's view. Even though this delay might only be some milliseconds, users can find this effect very unsettling and it is especially troublesome as this can trigger some of the cybersickness symptoms.

- Another factor is *flicker*, which causes eye fatigue, is distracting and also triggers cybersickness. Flicker is more likely to be perceived with larger FOVs because of the peripheral vision's sensitivity to movement. Since VR setups try to surround the user with a VE, this is of greater concern to VR than to other applications. Higher refresh rates can help removing the perception of flicker.

Modern HMDs are light and compact and typically allow for fast head movements and rotations. Fast changes in orientation or location can lead to distracting lags, in case the tracking equipment is slower than the speed of change. In such cases the HMD users are forced to move slowly and smoothly to mitigate the issue. In CAVEs the problem remains, as head movements and rotations may introduce the same lags. However, this effect is comparatively smaller when it comes to head rotations in CAVEs, because the stereo projections only require a small alteration to generate the correct perspective [2].

2.5 Intrusiveness

The *intrusiveness* of a VR device is the degree to which it restricts the senses of its users. HMDs without see-through capability have a very high level of intrusiveness. It can isolate the user entirely from their real environment. HMDs that are partially see-through or fully see-through allow users to see the real world in addition to the virtual objects superimposing it. When using such HMDs, the intrusiveness is decreased, but, on the other hand, the immersiveness and suspension of disbelief are broken more easily. This happens due to the view of the real environment, which is only enhanced with virtual objects and not fully replaced, a process referred to as *augmented reality*.

2.6 Perception of Omnistereo in the Human Visual System

Omnidirectional stereo images, also called omnistereo images, are images that provide a correct stereo effect at the center of the view independently of the current view-direction. The stereo effect is present omnidirectionally, as all points are rendered as if the viewer is facing them, looking in their direction. The images therefore do not need to be re-rendered for changing head orientations. Simon et al. [4] suggested that it is only necessary to create accurate stereo cues around a narrow FOV in the viewing direction. In order to understand why this can be the case, a look at the human visual system and its properties needs to be taken. When both eyes are used, the human visual system has a FOV of approximately 180° horizontally and 120° vertically [21]. The area that can be seen by both eyes simultaneously, referred to as the binocular field of vision, is approximately 120° horizontally. Omnistereo techniques assume the head orientation to be aligned vertically, i.e. without a sideways tilt or a roll, in terms of principal axes. A head rotation around the vertical axis is simulated for every view direction when rendering omnistereo images, therefore the horizontal view is the most relevant one in

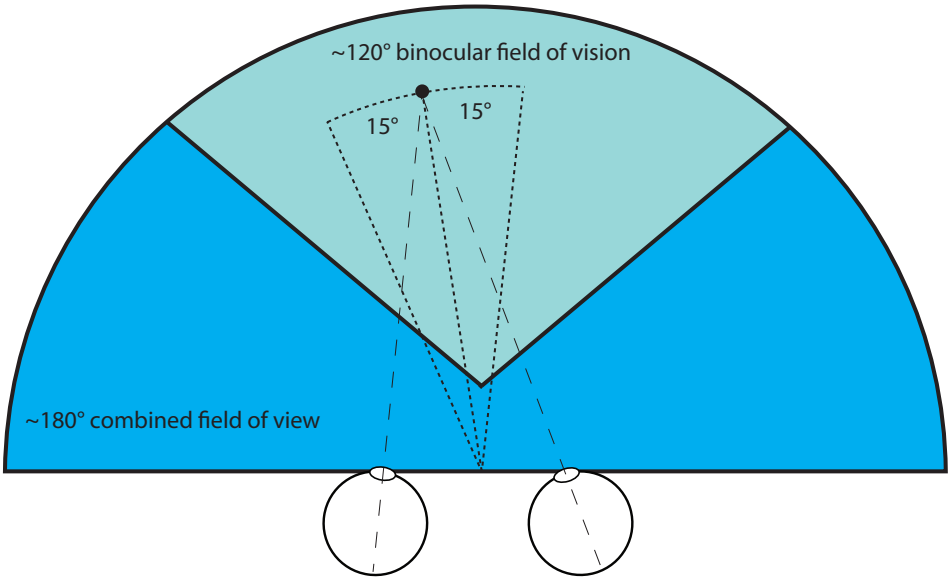

Figure 2.10: Gaze shift angles greater than approximately 15° usually make viewers turn their heads in addition to the eye movements that are done in order to focus at the point.

this context. A gaze is an intent look at something. A gaze shift is the realignment of the line of sight that is done to move the projected image of an object of interest to the central retina, where the visual resolution is the highest. Starting from the central position, eyes cannot move more than approximately 50° to the right or to the left [22]. Gaze shifts that are greater than approximately 15° usually involve a head movement in addition to the movement of the eyes, therefore limiting the rotation of the eyes. Sometimes small-amplitude gaze shifts also lead to head movements. Figure 2.10 shows the binocular FOV and the gaze shift angle.

When following moving objects with their eyes (visual tracking), people normally also move their heads. Following a moving target solely with eye movements is rarely done for significant periods of time in real life scenarios. Spectators at a tennis match are a good example for a gaze pursuit of an object, in this case the tennis ball, that involves both head and eye movements. Around the eyes' view direction, a sharp central vision is provided by the fovea. This foveal vision covers only approximately 1,5-2° of the FOV, which is approximately 0,1° of the entire FOV [23]. The peripheral view covers the much larger remaining view but has around the same amount of the optic nerve and the visual cortex available for its information. Towards the outer periphery of the vision, the images appear increasingly distorted and blurry. The visual acuity decreases the larger the angle to the central vision is, i.e. towards the outer periphery of the vision. Figure 2.11 shows this decrease depending on the location of the image on the retina. The decreased visual acuity also potentially affects large parts of the binocular FOV. In the peripheral vision, motion is perceived much faster than in the foveal vision.

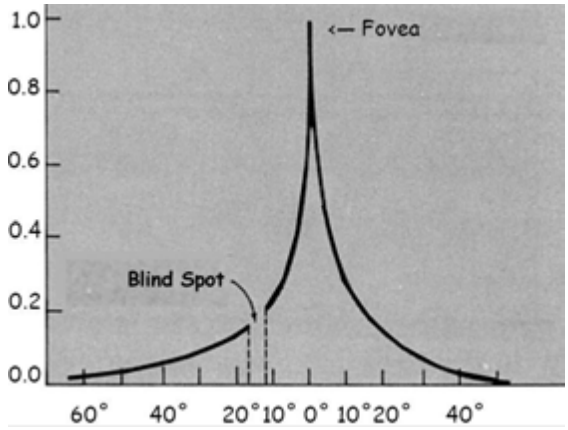

Figure 2.11: Visual acuity depending on the angular distance from the fovea. Image created by Hans-Werner Hunziker [23]

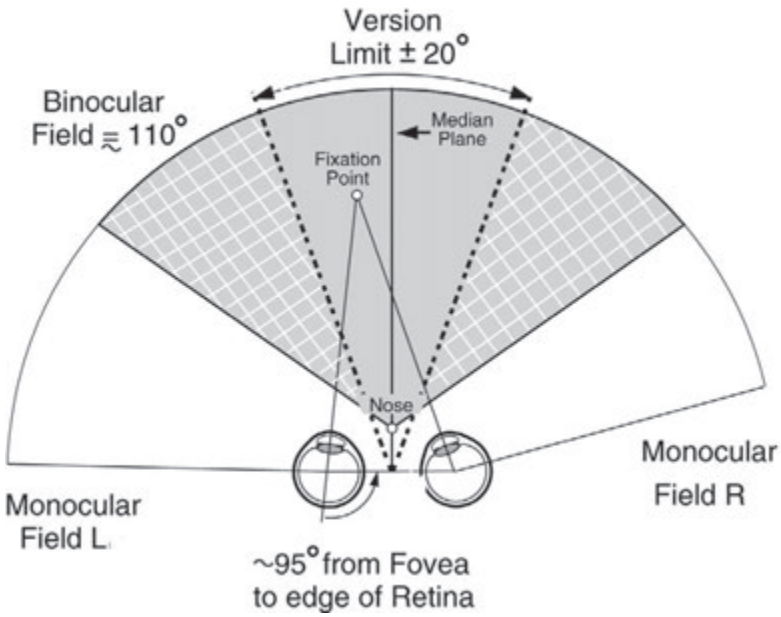

Figure 2.12: Viewers usually turn their heads when attempting to fixate points outside approximately 20° range of their head-centric median plane, which is the comfortable viewing range for eye motion (version). Image used with permission from Simon et al. [4]

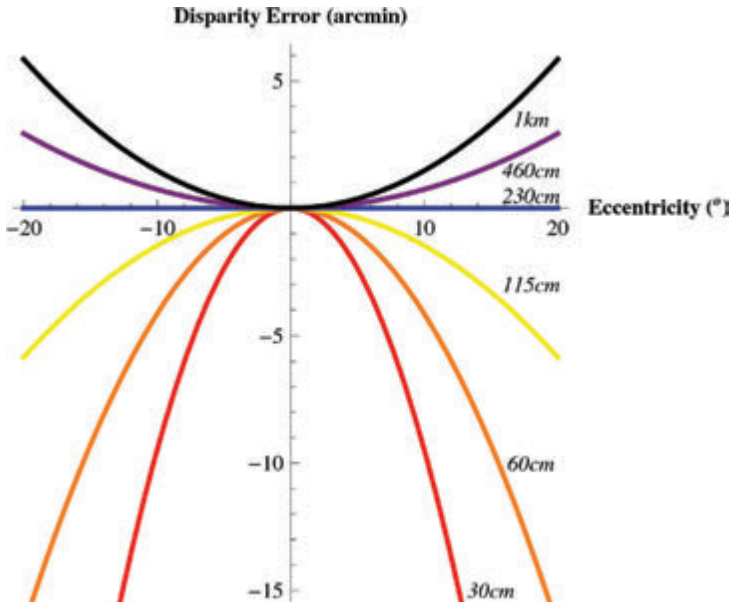

Figure 2.13: Disparity errors of an omnistereo projection for a viewer looking in a specific direction. The omnistereo projection distorts points in the periphery of the visual field. The computed disparity distortions at various distances from the observer within a 40° FOV are shown. Distortion is zero at an eccentricity of 0° and for points on the screen (blue curve). Image taken from Couture et al. [24]

When attempting to view a point outside the comfortable view range of 40°, the viewer usually turns their head toward the point, thus limiting the amount of eye rotation. Figure 2.12 shows the comfortable view range, which is an area extending from the median plane between the eyes. Detailed stereo vision is usually attained by turning the head.

Couture et al. [24] analysed the stereo disparity distortions of two omnistereo projection models. The projection models create omnistereo images for cylindrical displays for observers standing at the center. Omnistereo by design has zero distortion at eccentricity 0°. Disparity errors increase gradually away from this zero-distortion locus. Points that are located at large horizontal distance from this locus are distorted to be closer if the point is in front of the screen, and farther if it is behind the screen. Points located far away from the observer are exposed to larger depth distortions, however, these distortions are not necessarily perceivable, as the visual system measures disparity, which depends on inverse depth. Therefore, large absolute errors in the triangulated depth of a point might result in a small disparity change, which can make the error less apparent. Figure 2.13 shows the disparity errors for points at various distances from the observer within a 40° FOV.

Couture et al. [24] also examined whether the depth disparity errors are perceivable. On the one hand, geometric distortions increase with eccentricity, on the other hand the

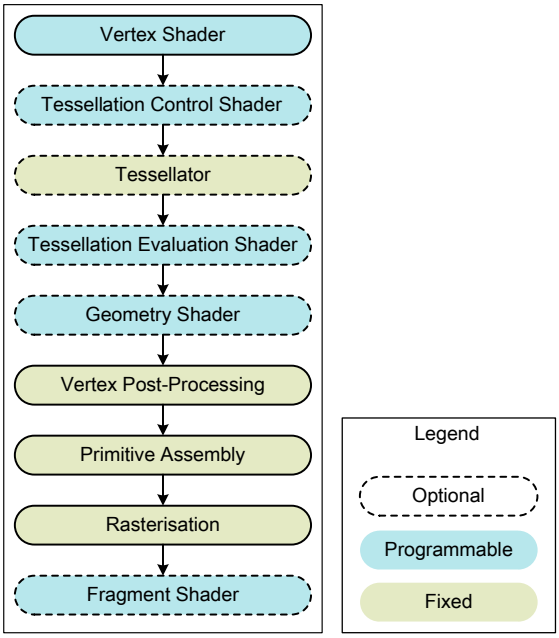

Figure 2.14: The OpenGL 4 rendering pipeline

resolution of the visual system decreases with it. Stereo acuity is the smallest detectable difference in depth that can be perceived in binocular vision. Although the human stereo acuity worsens in the periphery, the decreasing performance is not worse than expected, if considering the worsened luminance perception in the periphery [25, 26]. The remaining question is if the limits of the visual stereo acuity of the human vision outweigh the distortion errors, especially in the periphery. Based on the data from previous studies on stereo in peripheral vision, Couture et al. [24] suggested that the distortions are at or below the detection threshold for humans, as long as the rendered points are not too close to the viewer. In that case, the disparity errors may not be of significance to human observers. In their projection model, the minimum distance was 1 m, displayed as the yellow curve in Figure 2.13.

2.7 Tessellation Shaders

In OpenGL 4.0, the *tessellation stage* was added to rendering pipeline as an OpenGL *Core* feature. The rendering pipeline is shown in Figure 2.14 including the programmable shader stages. The tessellation control and tessellation evaluation shaders were introduced in OpenGL 4.0, while the geometry shader had become a core feature already in OpenGL 3.2. All shader stages, except the vertex shader, are optional. However, without a fragment shader the colour output is undefined and only the stencil and depth values will be written as usual. Both the geometry shader and the tessellation shaders

can generate new geometry based on input primitives. The geometry shader, however, has upper limits regarding the number of output vertices and is executed within a single shader. Tessellation shaders were introduced specifically to allow subdividing primitives into smaller primitives and are therefore better suited for this purpose.

The tessellation process consists of three steps. First, the tessellation control shader (TCS) determines the amount of tessellation to be applied per primitive. In this context, the continuity of adjacent edges of primitives needs to be considered. Vertices that are inserted along a triangle's edge, but are not inserted in the same manner along the edge of the neighbouring triangle, are known as *t-vertices*. If t-vertices are introduced at edges shared between primitives, gaps, breaks or seams can appear in supposedly contiguous meshes [27]. These cracks in the mesh occur around the t-vertices once a type of vertex displacement is applied to the mesh. Since object warping displaces vertices, this aspect has to be considered in this technique.

Second, the fixed-function *tessellation primitive generator* subdivides the input patch. This stage is affected by the spacing mode that can be set in the tessellation evaluation shader (TES). There are three spacing modes in OpenGL 4:

- *equal_spacing*
- *fractional_even_spacing*
- *fractional_odd_spacing*

equal_spacing leads to the edge being divided into segments of equal length. The *fractional spacing modes* provide smoother, more stable interpolation for gradually changing tessellation levels. For adaptive tessellation, the fractional modes are therefore better suited. Using equal spacing leads to perceivable "jumps" between integral tessellation levels, whereas the fractional modes factor the fractional parts of the tessellation values in, providing smooth progression.

Third, the TES receives the tessellation results and computes the interpolated data values based on them. When processing triangle primitives, the barycentric coordinates are accessible in the shader and can be used to interpolate the value based on the values of the triangle's original vertices.

CHAPTER 3

Motivation

In the previous chapter we discussed the generation of stereo images and several aspects that need to be considered in VR applications. Stereo rendering in conjunction with surround-screen displays provides users with a large FOV, a low amount of intrusiveness and high immersion. The stereoscopic rendering provides additional depth cues, but requires head position and direction tracking in order to generate an image with a correct perspective for a single user. If the off-axis projection is not recalculated on a change of the view-direction, the rotation of the view-direction to the left or right side by 90° results in nullified depth cues, with one eye's view being rendered closer and the other farther from the scene. When looking to the sides by 180° , the left and right eye images are swapped, effectively reversing the binocular depth cues. Therefore, an undistorted view in a conventional perspective is tied with a single view direction selected by the application, for example of a single head-tracked guide in a CAVE.

Figure 3.1 highlights the difference between off-axis and omnistereo projections. The off-axis projection features almost no binocular disparities for the points projected on the lower screen. In contrast, the omnistereo projection provides binocular disparities for all directions the viewer might be facing at that moment.

In Section 2.6 we have mentioned that omnistereo images can provide correct stereo in all view directions for the center of the view. In the periphery of the view, the omnistereo image pairs provide increasingly inaccurate stereo cues. However, we have shown that several traits of the human visual system and the likeliness of humans to rotate their heads to look at objects of interest mitigate this problem.

3.1 Advantages of Omnidirectional Stereo

One advantage of the omnistereo projection model is that it lifts the requirement of head-tracking partially or entirely, depending on the design of the VR system. Head orientation or view direction tracking is not required in any case. Head position tracking

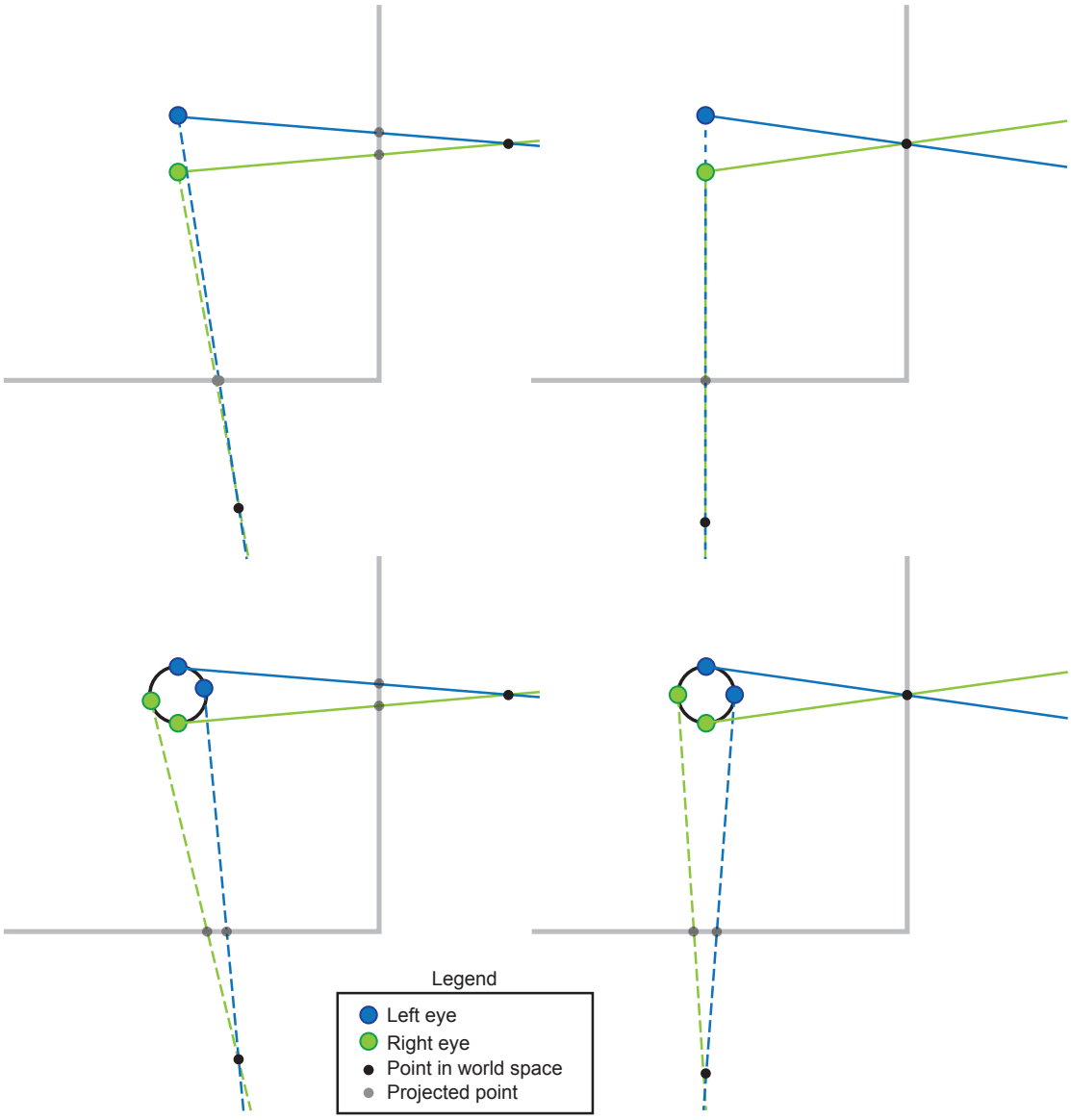

Figure 3.1: Top-down view of a CAVE. Points and their projections on the screens are shown for the left and the right eye. Top: Off-axis projection with a fixed view-direction perpendicular to the right screen. Bottom: Omnistereo projection.

is required only if the user is allowed to move away from the assumed head position to such an extent that the distortions in the projection on the screen become too large for the actual viewpoint. If the tracking is not fast enough, it can introduce delays, which in turn can cause cybersickness. Omnistereo images do not need to be re-rendered when the head is rotated, which removes this issue entirely.

Another advantage of omnistereo rendering is that it may allow multiple viewers to share a display, with some constraints. While the distortion error increases with an increasing deviation from the assumed viewpoint, it can be small enough in large-scale panoramic displays and curved screens, therefore being suited for multiple simultaneous viewers.

3.2 Research Aim

Our main motivation for omnistereo rendering is to generate omnistereo for a single user in a CAVE display system without any tracking. A constraint imposed by omnistereo rendering is that the up-axis of the viewer has to be constant, i.e., the viewer may not tilt their head. However, this is not a serious limitation for typical VR applications.

An unanswered question so far is how omnistereo pairs are to be generated. Given a display setup that is built using multiple adjacent screens, a naïve approach to achieving an omnistereo effect is to render each screen's image using a separate off-axis projection, with the view direction always set to face the respective screen. A CAVE consists of walls that serve as adjacent projection screens. In such a setup, each wall's image can be rendered with the view direction being oriented perpendicular to the projection plane. This means that, based on the head position, the assumed eye locations are positioned so that the projection is correct for a viewer facing the screen. The images are then rendered using off-axis projections based on these eye locations. In this case the stereo depth cues are indeed present in all view directions, but they are lightly diminished towards the borders between the vertical screens.

Figure 3.2 shows the left eyes' view frustums of such a setup. The frustums shown in the Figure make it apparent why the naïve approach will cause issues. The edges and planes of neighbouring view frustums do not coincide, although they would if the eye positions were identical. Thus, the parts of the virtual environment that are virtually located between the planes and inside the room are not enclosed by any of the two frustums and therefore do not appear on either of the screens, while the parts located outside the room would appear duplicated across the screens. The breaks in the transition between the views, which occur due to the large rotational difference between the view directions of the adjacent screens, make this approach not viable. This problem has nothing to do with stereo rendering itself, but rather with the use of multiple centers of projection, since each wall uses a separate center of projection [4].

Ray-tracing can be used to create stereo image pairs, as well as omnistereo images, but has considerably higher computational demands and is therefore generally not suited for usage in VR. Omnistereo images feature a non-linear perspective. This means that they cannot be produced with traditional means used in real-time rendering, such as

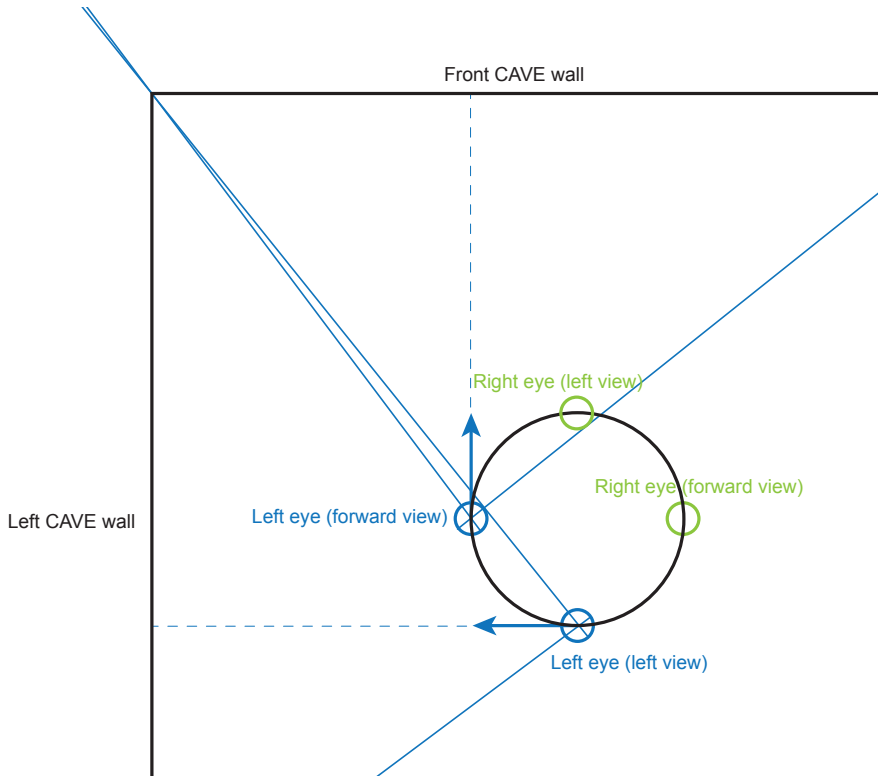

Figure 3.2: In a naïve approach to render omnistereo, separate view-directions that are perpendicular to the projection plane may be used for each CAVE wall's projection. The images shows the view frustums in such an approach for the left eye. Since the eye positions are not consistent for the respective eye, the edges of the frustums of adjacent screens do not coincide. As a result, the viewer will see apparent discontinuities between the screens.

by using a projection matrix with a homogeneous coordinate for linear perspectives. Cameras with fish eye lenses take images from a curvilinear perspective, which is a non-linear perspective as well. This perspective can be reproduced in computer graphics using a function that samples a cubemap or by approximation by the means of vertex displacement, for example, implemented inside a vertex shader. Equally to curvilinear perspectives, perspective projection can not produce correct images for display systems with curved screen, such as dome displays, unless both the viewpoint and point of projection are in the center [28]. The perspective can be approximated by using a fragment shader that computes the intersection of the view ray with the display and samples a pre-rendered cube map accordingly. However, a cubemap-based approach is not viable for omnistereo rendering, since the cubemaps are rendered from a single viewpoint, whereas the omnistereo projection is based on multiple viewpoints. Another way to render images for curved screens is by using vertex correction, which involves

casting a ray from the view point to the vertex position, determining the intersection point with the projection surface and rendering the point considering the projector's view. This can be done on the GPU in a vertex shader, but since edges are linearly rasterised in the rendering pipeline this does not work well with long edges. However, the step can be preceded by primitive tessellation to refine the mesh and achieve a better accurate approximation of the edge distortion.

All things considered, our aim is to find a technique for creating omnistereo images for a CAVE, which is implemented using traditional GPU-based rendering with the programmable rendering pipeline and rasterisation. A single-pass solution based on a vertex-shader, or potentially a tessellation shader, is preferred in order to keep the impact on the performance and changes regarding the rendering process small. Such a solution also allows adapting existing applications to render omnistereo images with small changes.

CHAPTER 4

Related Work

4.1 CAVE

The CAVE was first mentioned in 1992 by Cruz-Neira et al. [2]. The motivation that lead to the creation of the CAVE was to create a useful tool for scientific visualisation. A CAVE consists of display screens that form a complete or partial room of cubic shape. The walls, ceiling and floor of this room are used as planar projection surfaces surrounding the user who is located inside this room. An early presented version of the CAVE consisted of three wall projections and a floor projection [29]. Figure 4.1 shows a CAVE setup.

The central idea behind the CAVE was to overcome problems encountered by other virtual environment systems. In the CAVE, the binocular stereo is established by using stereo rendering on the projection screens. To accomplish a correct perspective projection and a correct stereo view, the user's head position inside the bounds of the CAVE and head rotation are tracked at all times. Figure 4.2 shows a user inside the CAVE and the view frustums of the off-axis projection based on the tracked position and direction. In a fully set up CAVE with all 4 walls, the floor and the ceiling in use as projection screens, the user is completely surrounded with projections from the virtual environment. In this setup, the user is provided with the full 360° FOV for all view points. The FOV may, however, be limited by the display hardware, such as the stereo glasses.

The cubic setup of the CAVE was chosen as an approximation to a sphere. The benefit of this setup is that it aids people standing inside the CAVE. In addition to that, off-the-shelf high resolution video projectors, as well as modern graphics libraries and hardware, are primarily specialised in projecting imagery on flat rectangular surfaces, making them perfectly suited for the CAVE's requirements.

Interaction in virtual environments often requires representing the physical body of the user, specifically the hands. Physical objects that the users are supposed to interact with might also need to be represented. The majority of HMD devices completely obstruct the user's view of the real world, restricting the senses to the computer simulation.

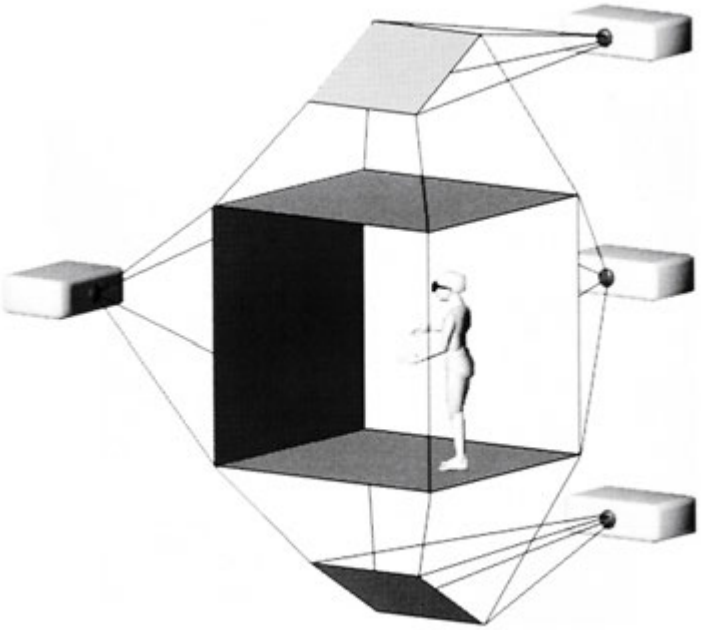

Figure 4.1: Projector setup for a CAVE

For these devices, the representation are explicit, meaning that the representations have to be simulated and rendered just like the rest of the virtual environment itself. This requires additional tracking of the body parts and the objects and the allocation of a higher computation time for rendering.

In contrast to that, the CAVE and similar surround-screen setups, offer implicit body representations because the body is visible physically and does not require additional rendering. In such setups the appearance of the body representations can not be altered. Additionally, the user's body or any other physical object in such a setup can partially or fully occlude a user's view of the virtual environment. Virtual objects that are supposed to be closer to the viewer than the occluding object, can therefore be occluded as well. This becomes a problem, for example, when the virtual object is between the user's hand and the eyes or if another user gets in the way of another user's view of an object. In such cases, where the object would have to be projected between the occluder and the viewer, the stereo collapses.

According to Cruz-Neira et al. [29], the prevalence of cybersickness in the CAVE is relatively low compared to other displays used in VR. Also, the projection planes in a CAVE do not move when the viewer changes their view angle or position, unlike HMDs. Cruz-Neira et al. [29] carried out a quantitative analysis of the effect of tracking noise and latency on the user and found out that the CAVE has a greatly minimised error sensitivity to tracking noise and latency associated with head rotation compared to HMDs.

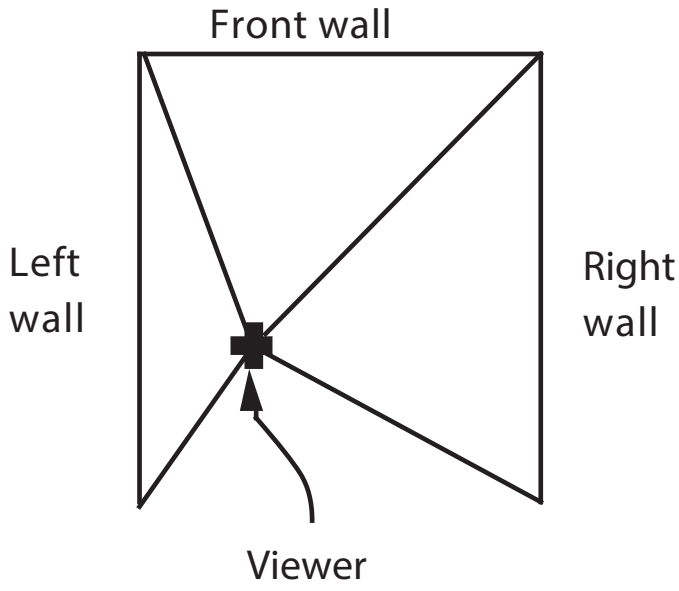

Figure 4.2: Visualisation of the frustums of the off-axis projections. The user is not positioned centrally inside the CAVE

Nevertheless, there are several shortcomings of the CAVE:

- The CAVE is relatively expensive and requires a large amount of space. Cheap wall-sized LCD screens are not available and while stereo video projectors got cheaper over the past years, the price of multiple projectors still exceeds that of many HMDs. However, if multiple viewers stand inside the CAVE, then the per-person cost could be considered cheaper. In contrast, if using HMDs, an HMD needs to be bought for each user.

- Projecting on all 6 sides of the CAVE is difficult. For example, if the floor projection is projected from above, it may cast user shadows. Rear-projections for the floor require a special floor construction, which can include having to raise the CAVE. One of the walls has to feature a human entrance and exit, which can pose a problem considering that the wall also serves as a screen.

- Internal light spillage can be caused by the CAVE's wall, ceiling and floor screens on the other screens, depending on their reflectivity and brightness. This can reduce the contrast of the images.

The first 6-sided hybrid CAVE was created by Fraunhofer-Gesellschaft [30] in 2001 [31]. The display system is called HyPI-6 (Hybrid 6-sided Personal Immersion System). The term 'hybrid' refers to the fact that the system offers both passive and active stereo projection. It was also one of very few 6-sided CAVEs at its time. The system can be driven either by a PC-multi-node-system for passive stereo or a SGI Onyx3 for active

Figure 4.3: The low-cost surround-screen display system created by Cruz-Neira et al. [32]. Image taken from Cruz-Neira et al. [32]

stereo. The dimensions of the CAVE are $2,9 \times 2,9 \times 2,7$ m. Fabric screens are used for the projection surfaces. To reduce the physical footprint of the device, mirrors are used for each screen, since rear projections are used.

According to Cruz-Neira et al. [32], the costs of acquiring surround-screen displays have generally been in the hundreds and even millions of dollars, especially if offering synchronised stereo projections across multiple screens. A typical CAVE system, consisting of three wall screens and a floor screen, can easily cost more than three-quarters of a million dollars. A large part of the cost is related to customisations that are needed to reduce the visibility of the seams between the screens and to the fact that a special room supporting the projector arrangement is required. In addition to that, expensive recurrent maintenance is required to keep systems like the CAVE operational. For these reasons, the widespread use of such systems has been limited.

The goal of Cruz-Neira et al. [32] was to create a VR display that is affordable and easier to maintain and upgrade than commercially VR displays, while providing comparable quality. They presented a low-cost and easy to maintain display system that is similar to the original CAVE but can be built based on commercial off-the-shelf components. It is modular and can be installed in a variety of locations without requiring special room conditions. The quality of the system is comparable to more expensive solutions.

A self-supporting structure that could be taken apart was used for the installation. The maximum floor dimension was restricted to approximately $14,63\ m \times 14,63\ m$ and the actual screen-bounded dimension inside the display system was approximately $3,66\ m \times 3,66\ m \times 2,74\ m$. The display consists of three wall screens. Despite having slightly larger screens than a typical CAVE, only a single projector per screen needed

to be used. A projector stand was created for each projector, so that they do not need to be attached to a ceiling or wall of the installation space. The stand has adjustable height to compensate for unevenness of the floor. The projectors featured an aspect ratio of 4:3 and provided active stereoscopic imagery. To avoid long projection distance to the screens, conversion lenses were put in front of the projectors' output to magnify the image by approximately 1,5.

Each of the three projectors projectors receives its input from a different PC. The PCs are part of a five-node graphics cluster. The fourth node acts as the master node for processing input, logic, and distributing the results to the three render nodes. The fifth node acts as a file server in the cluster to minimise maintenance and avoid software inconsistencies on the client nodes. Their system incorporated an omnidirectional treadmill for locomotion as well as tracking. Figure 4.3 shows the display system in use.

Due to the resolution, applications such as flight simulators and military command and control are therefore outside the scope of the system. Nevertheless, it can be used for architectural walk-through, engineering, science visualization, and educational applications. A disadvantage of using the magnifying lenses is that the pixel size on the screen is approximately 3,8 mm and that small aberrations occurring due to the projectors' optics are magnified. However, realistically only the trained eye can detect these with static images. A tracking system is used to acquire the user's head position and view direction to render images with a correct perspective for that user. Cruz-Neira et al. used the optical tracking system that came with the omnidirectional treadmill for this purpose.

4.2 Omnidirectional Stereo Imaging

Omnistereo methods have mainly been described two different contexts. On one hand there is the question of how to create omnistereo images based on real environments by using cameras, on the other hand there is the question of how to render omnistereo images based on virtual environments. Ishiguro et al. [33] were the first to use the term *omnidirectional stereo*, describing a method to generate omnidirectional panorama images. Their method uses a single camera that swivels about a vertical axis, consecutively taking images of a real world environment. Peleg et al. [12] described the theory of creating omnistereo images using cameras. The first method they describe is the method by Ishiguro et al. [33] where a slit camera is used, which has its film covered except for a narrow vertical slit. The slit determines a single viewing direction for the camera. The slit camera can be rotated in a way that the resulting panoramic image has a single view point. If the camera is rotated about a vertical axis directly behind the camera, the resulting viewpoints are located on a viewing circle. The circular projection projects all points in the scene with the viewer or the camera facing the point by turning towards it in the horizontal plane (i.e. rotating around the vertical axis). The projection is therefore always correct along the viewing direction but not in the peripheral vision. Figure 4.4 displays the circular projections for these two types of rotation.

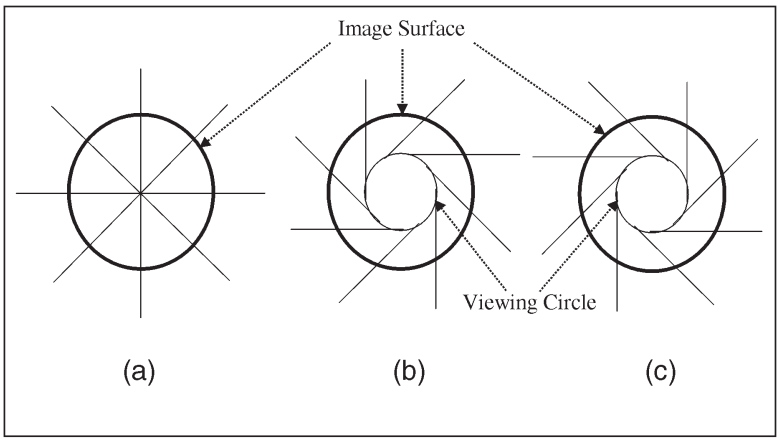

Figure 4.4: The projection from the scene to the image surface is done along the rays tangent to the viewing circle. (a) The traditional single-viewpoint panoramic image. (b) and (c) Multiple-viewpoint circular projections with inner viewing circles. Image taken from Peleg et al. [12]

Peleg et al. [12] use a camera with a right and left slit that move together with the camera. The two slits create two panoramic images enabling stereo perception in all directions. The second method describes how to create stereo panoramas with video cameras. A video camera is rotated about the vertical axis behind it, but vertical image strips are obtained from the images instead of using slits. For stereo mosaicing, two vertical strips are obtained from the left and right side of each image. Composing the narrow vertical strips approximates a circular projection on a cylindrical image surface, as can be seen in Figure 4.5. Additionally, a system using a spiral lens and another one using spiral mirrors were suggested to capture real-time movies having stereo panoramic features. However, omnistereo lenses or mirrors have not yet been developed. For this reason, and since the existing methods produced omnistereo images for static scenes only, Couture, V. and Langer, M. S. and Roy, S. [34] presented a method that produces loopable panoramic stereo videos based on dynamic scenes. The resulting omnistereo videos can be displayed up to 360° around a viewer.

4.3 Rendering Stereo Panoramas

Peleg et al. [12] described a method using conventional rendering to create omnistereo panorama images for cylindrical or spheric projection surfaces based on virtual environments and 3D models [35]. The viewing cylinder or sphere can be approximated using polygonal geometry. Each face of the geometry is rendered separately from a different viewpoint. Figure 4.6 shows how the faces of the polygonal approximation are rendered. For reasonable quality, a vertical face has to be used for each 1-2°, which accounts to approximately 200-400 faces for the entire cylinder. Stereo rendering allows controlling

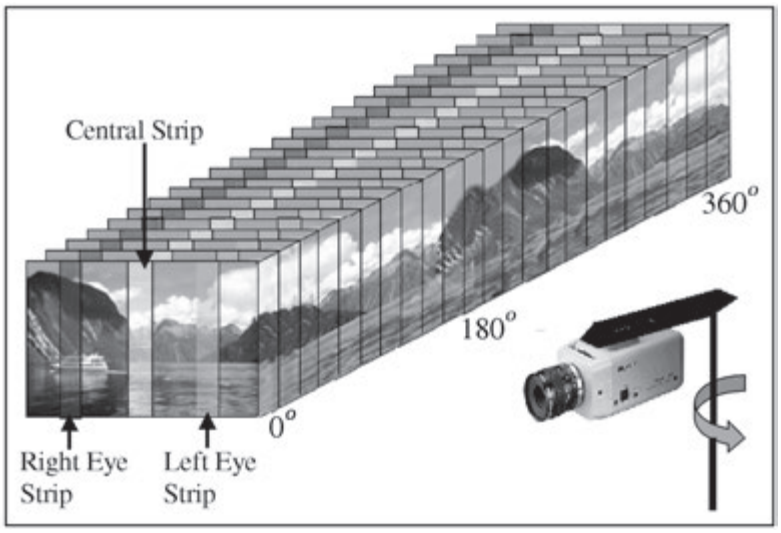

Figure 4.5: The image shows how stereo panoramas can be created using a video camera rotating about an axis behind it. The strips taken from each image are pasted together to approximate a panoramic image cylinder. Vertical strips taken from the center of the images create a monoscopic panorama, vertical strips taken with an offset from the center to the left and right of the images create right and left eye panoramas respectively, which together form a stereo panorama. Image taken from Peleg et al. [12]

binocular disparity, which makes it possible to assume an eye distance that is different from the actual eye distance of the viewer. This can be used to assume a larger eye distance for far away scenes and a smaller one for closer scenes, to make the stereo effect artificially stronger or weaker for a given scene. To view stereo panoramas on planar image surfaces as provided by flat screens, such as computer or television displays or HMDs, the panoramic image needs to be projected from the cylinder onto a plane. A central projection about the center of the cylinder, with the image plane tangent to the panoramic cylinder, should be used for this purpose and will preserve the depth perception.

Zhu, Zhigang [36] discussed multiple different omnistereo techniques for capturing omnistereo images and introduced a new catadioptric circular projection omnistereo method. The catadroptric method uses planar mirrors to enlarge the viewing circle, allowing for more compact rigs, and provides better stereo configurations. He also presented a dynamic omnistereo approach and gave a numerical analysis on omnidirectional representation, epipolar geometry and depth error characteristics.

4.4 Multiple-Center-Of-Projection Images

The issue of rendering omnistereo images based on virtual environments is essentially not an issue of stereo rendering, but rather one of creating *multiple-center-of-projection*

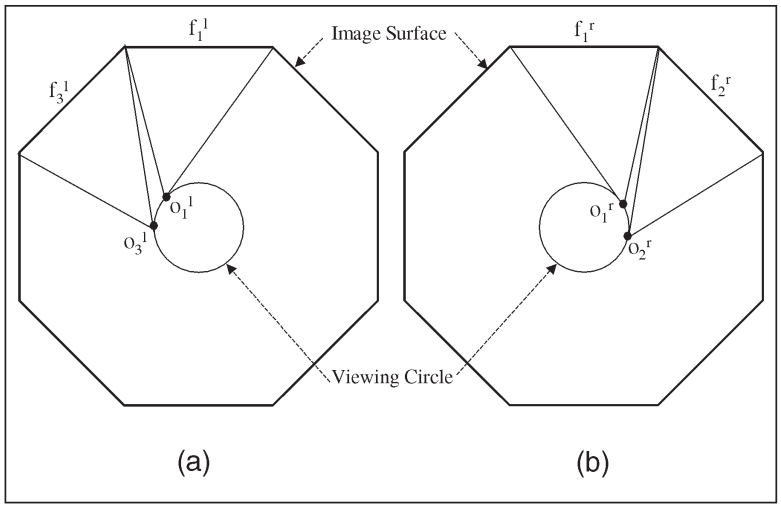

Figure 4.6: Faces of a polygonal approximation for rendering cylindrical panoramas. The faces are rendered from a central viewpoint suitable for the circular projection. Image taken from Peleg et al. [12]

images [4, 37]. A multiple-center-of-projection (MCOP) image is a single image acquired from multiple locations. The technique for creating MCOP images proposed by Rademacher and Bishop [37] allows synthesising an image from a new viewpoint based on data from multiple reference images. It combines samples from multiple viewpoints into a single image, which becomes the complete dataset. An example for a MCOP image is shown in Figure 4.7. Although the MCOP image definition encompasses a wide range of camera configurations, the work of Rademacher and Bishop is mostly based on photographic strip cameras, which consist of a moving strip of film behind a vertical slit. These cameras capture continuous image-slices of a scene are through the slit, as they move through space while the film rolls by. By these means, each vertical slice can be captured from a different view point. Images that are created by strip cameras that moves along a continuous path automatically satisfy the criteria for MCOP images. Figure 4.8 shows the process of capturing MCOP images with a strip camera.

Strip cameras have been used extensively, for example, in aerial photography. The *strip cameras* we refer to in this work are digital strip cameras that capture each pixel-wide vertical slice from a different center of projection. An MCOP image is an extension of a conventional two-dimensional image and is characterised by having a set of cameras contributing to it instead of only a single one. Therefore, its individual pixels or sets of pixels are acquired by different parameterised cameras. Four conditions have to be met by MCOP images: the camera locations have to lie on a continuous curve or a continuous surface instead of being an unorganised set of points, samples captured from multiple cameras are not blended together - i.e. each pixel is acquired by a single camera -, viewing rays vary continuously across neighbouring pixels and two neighbouring pixels must either correspond to the same camera or to neighbouring cameras, therefore assuring

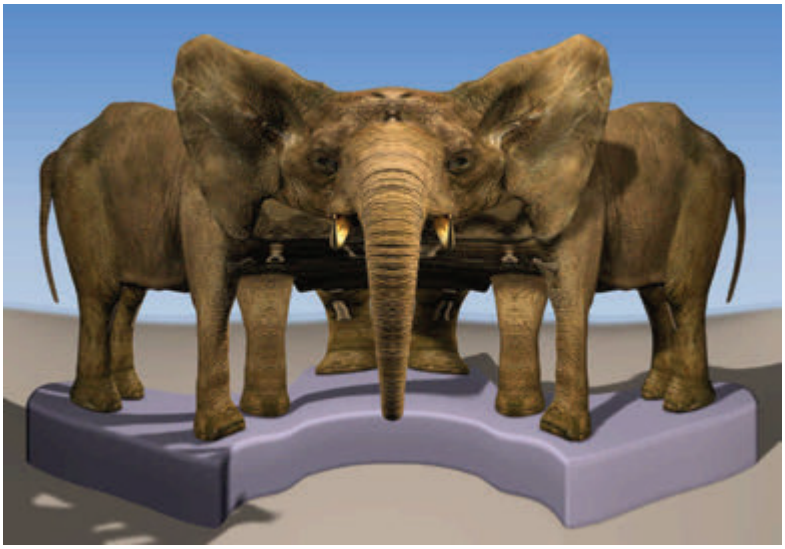

Figure 4.7: An example multiple-center-of-projection image of an elephant. Image taken from Rademacher and Bishop [37]

a smooth transition from camera to camera when traversing pixels. Sampling issues in MCOP image rendering can occur if, for example, the viewing rays of the cameras capturing a nearly-convex object approach grazing angles with the object. In contrast, if sampling every point at near-normal angle to the objects, the acquired samples are sufficient everywhere.

The process of creating MCOP images with slit cameras is analogous to that of creating an omnistereo image for a vertical screen. The latter involves moving a virtual or real camera along a viewing circle while capturing vertical slices. However, this is not true when it comes to omnistereo rendering for non-vertical screens, such ceiling or floor screens in a CAVE, which require a different approach. The specific data structure of MCOP images, which saves camera-related data together with the two-dimensional image data, is not relevant in the context of omnistereo images, as only the image itself is relevant. It should also be noted that the sampling issues related to the camera's view rays occur in a similar fashion in omnistereo techniques that render vertical image-slices to approximate the omnistereo image.

4.5 Omnistereo Rendering in a CAVE

Max [38] described a way to create omnistereo images for an OMNIMAX dome (now called IMAX Dome) using ray tracing. Although this approach is not feasible for real-time rendering, it provided the descriptive process for creating such images. Simon et al. [4] presented two techniques to render omnistereo images for the CAVE in real-time.

When using off-axis projections, the tracked physical eye positions of the viewer in

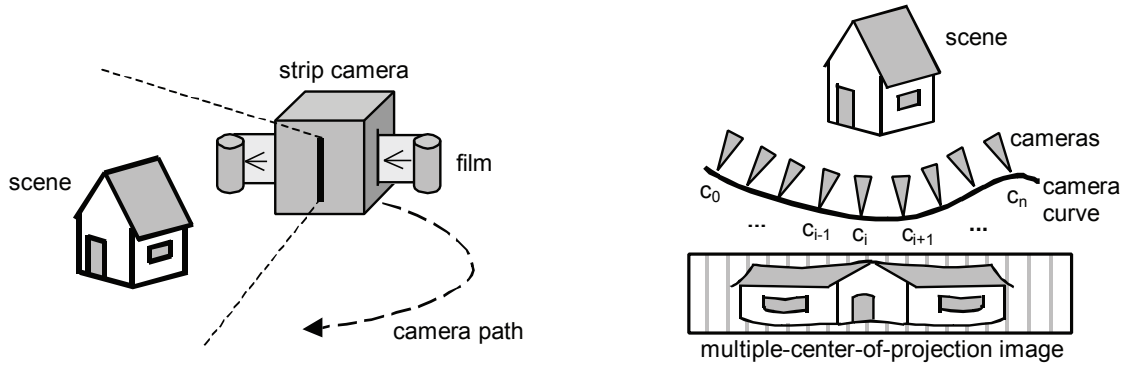

Figure 4.8: The left image shows a strip camera, consisting of a moving strip of film located behind a vertical slit. The right image shows a discrete number of image-slices being captured along a curve. The single MCOP image captured in this way allows seeing three sides of the house simultaneously. A similar configuration was used for Figure 4.7. Images taken from Rademacher and Bishop [37]

the display system coincides with the two *centers of projection* (COPs) of the stereo image projected on the screen. The resulting view on the virtual environment is therefore completely accurate and provides a correct binocular disparity, as it is re-rendered for every new head position and rotation. Conversely, omnistereo projections are only correct in the center of the visual field for every view direction, as the accuracy degrades in the periphery. Given two perspective projections based on COPs defined by the viewer's the left eye and right eye positions, the resulting location of a projected object in screen space will generally differ. This difference is relevant for the binocular disparity and therefore for the perceived depth. Since the two COPs depend on the viewer's head position and view direction, it is impossible to create stereoscopic images for multiple different view positions and directions at the same time in one stereo image while providing correct depth information in the entire visual field. However, it is possible to do so accurately, if not perfectly, as shown by Simon et al. [4]. An explanation why the properties of the human visual system make these imperfections of such an omnistereo projection acceptable is given in Chapter 3.

Figure 4.9 shows a top-down view of two lines and their perspective projections on a single screen. The projection is based on a single viewing position and view direction, whereas the view direction is perpendicular to the screen. A number of perceived locations of *model points* along the lines are shown. The human visual system of the viewer reconstructs the 3D location of points in the two-dimensional images based on the depth information provided in the stereo images. In this figure's setup the reconstructed location of the points are correct because the viewer's head position and direction coincide with those used for the projection. The rays from the left and right eye positions to the projected positions of points are shown. Since the viewer's actual eye locations are identical to the locations used in the projection, the projection is entirely correct. This

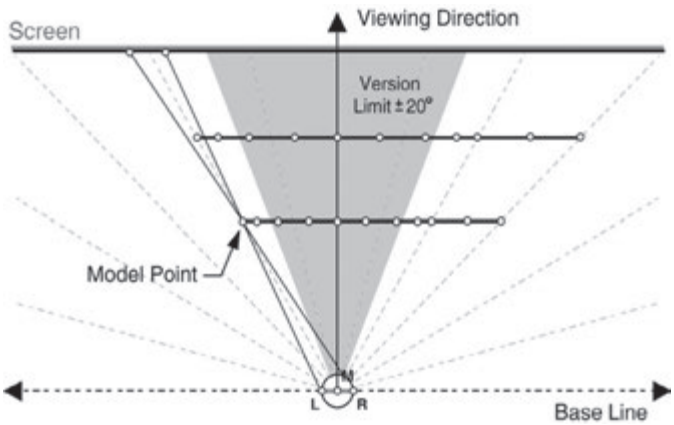

Figure 4.9: Top-down view of two lines being projected on a screen with a regular off-axis projection. The human visual system can correctly reconstruct the positions of the model points along the original line, based on the stereo images created with this projection. Image used with permission from Simon et al. [4]

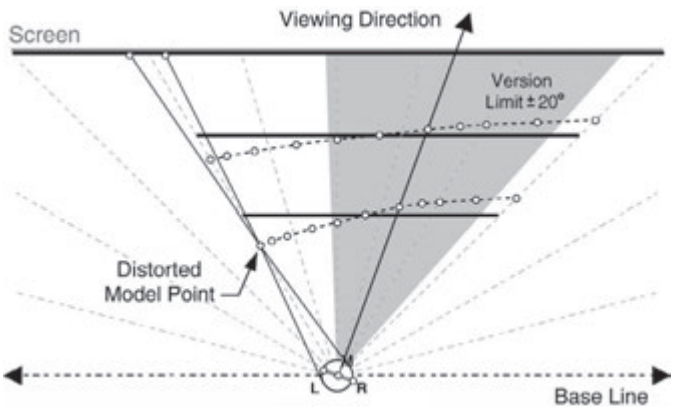

Figure 4.10: The identical image is shown on the screen as in the setup of Figure 4.9 but the view direction of the viewer has changed. Without having re-rendered the image, the depth of the model points can not be correctly reconstructed anymore. The model points are seen distorted, even along the view direction. Image used with permission from Simon et al. [4]

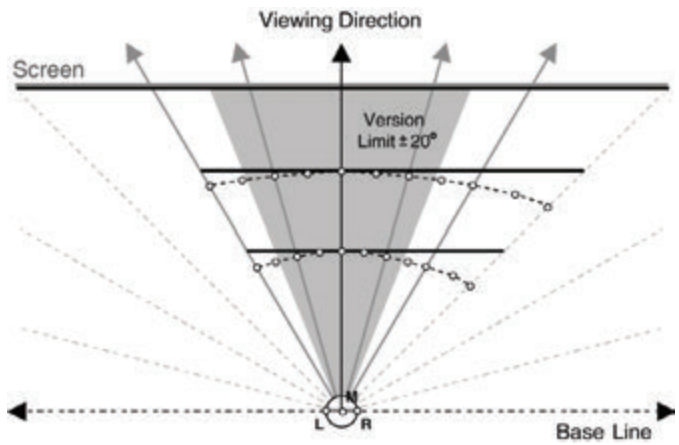

Figure 4.11: An omnistereo image is shown on the screen to a viewer with the same view direction as in Figure 4.9. No distortion is perceived in the center of the view, along the view direction, but distortion is perceived increasingly towards the periphery. Image used with permission from Simon et al. [4]

means that the depth can be reconstructed correctly from the image, as shown on the figure by the fact that the perceived model points are indeed located on the lines at the correct depth, since all eye ray intersections meet along the lines. Figure 4.10 shows the identical projection on the screen, the image on the screen is therefore unchanged. However, in this figure the viewer's eye locations do not coincide with those of the projection, since the viewer's head is rotated. In this case, the depth of any point on the screen can not be reconstructed correctly in 3D based on the 2D images, which causes the model to be perceived distorted. This can be seen by the fact that the reconstructed positions of the models points are situated along a dashed curve, whereas their actual positions are situated on the solid lines. In addition to that, it should be noted that the location of the model points is incorrectly perceived even along the view direction.

Figure 4.11 shows circular projections of the model point on the screen and the perceived distortion of the model point positions for a given view direction. In the omnidirectional circular projection, the screen space positions are only saved when the head's view direction is the same as the direction to the model point. In the center of the view there is no distortion, the model points are perceived along the line. However, a distortion occurs in the peripheral vision of the view direction. It should be noted that no distortion occurs in the peripheral view when using an off-axis projection, as long as the view direction coincides with the one used for the projection, as shown in Figure 4.9. Additionally, it can be seen that the distortion is relatively low around the view direction, increasing towards the periphery. Figure 4.12 shows the model's distortion in another view direction based on the same omnistereo images on the screen. Again the distortion is non-existent along the view direction and similarly increases towards the periphery.

Circular projections have been suggested as way to acquire omnistereo images. The

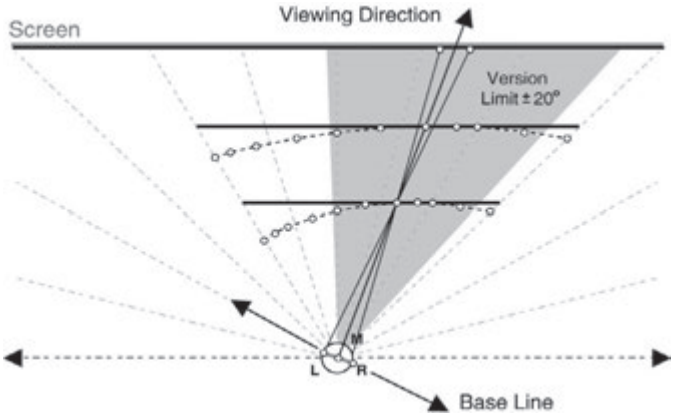

Figure 4.12: The same omnistereo image as in Figure 4.11 is shown on the screen. The viewer's view direction is changed as in Figure 4.9. No distortion occurs along the view direction, but distortion occurs towards the periphery of the view. Image used with permission from Simon et al. [4]

images can, for example, be rendered by adapting a ray-tracing program to implement the circular projection. Since this is not suitable for real-time rendering, Simon et al. [4] suggested two methods that can be rendered at real-time frame rates. The first technique works analogously to the multi-view rendering technique originally suggested by Peleg et al. [12]. It can be seen as an improvement to the naïve approach of rendering omnistereo images as discussed in Chapter 3. The naïve approach only uses one view direction per screen, using a regular linear projection (off-axis projection). If using multiple view directions per screen, and thus multiple COPs, the circular projection can be approximated with increasing accuracy. Raising the number of view directions per screen leads to a reduction of the error and thereby also to a reduction of the extent of the discontinuities between the views. The angular differences between the views are decreased as well. In the naïve approach, in order to have the views face the walls of the CAVE, the view is rotated 90° around the vertical axis each time. In the multiple COPs per view approach, a discrete number of views is rendered per screen. Each view uses a separate off-axis projection. The result of each view is a rendered vertical image slice. By combining these image slices, an omnistereo image is formed. The primary question surrounding the multi-view approach is how many views are needed to generate an omnistereo image without visible artefacts. The resolution of the virtual environment display system and the pixel size play a role in this matter. Simon et al. [4] suggest that the gap and the overlap errors between adjacent slices should be reduced below half a pixel in order to remove most visible artefacts. For a pixel size of 1-2mm, it was suggested to use 15° views to achieve pixel errors below half a pixel in common view conditions (objects between ½ screen distance and infinity). Simon et al. were able to confirm this approximation with their implementation of the technique.

The second method pre-warps the meshes of all models for each screen. The vertices

of the meshes are displaced so that the standard off-axis projection creates an omnistereo image for that screen. This pre-warping technique, called *object warp*, is appropriate for real-time rendering on the GPU. Pre-warping of meshes has also been used by Wartell et al. [39] for controlling stereo distortion, which occurs when deliberate incorrect (under-/overestimated) eye-separation distances are used for the viewer. In object warping, the projection of a vertex is created from a fixed *projective viewpoint* using a standard off-axis perspective projection. This projective viewpoint can be different from the assumed position of the viewer, which defines the circular projection. The projective viewpoint and the assumed position of the viewer define how the vertices of the object's mesh are warped. The objects are warped in order to appear on screen as if they were seen in the center of the view by the viewer, while facing the direction of the object, yet they are projected from the projective viewpoint using a regular perspective projection. The eye positions of the omnistereo projection are always located along the viewing circle for each object's direction. The object has to be warped for the left and the right eye separately for each omnistereo image pair and object proportions are maintained in the warping. The *eye ray* is the ray that starts from the eye position and passes through the original vertex position. The point of intersection of the eye ray and the projection plane is relevant for the calculation of the warping, as the warped vertex position has to lie on the ray that starts from the projective viewpoint and goes through this intersection point. This ensures that the warped vertex's position in screen space, once projected from the projective viewpoint, will be seen from the eye position as expected. By using relationships of similar triangles in the geometry of this setup, Simon et al. [4] derived a simple and fast solution to calculate the warped position. The method can be combined with any algorithm based on regular perspective projections. The geometrical background of this object warping technique, as well as an implementation of the method, are described in detail in Section 5.2.

Simon et al. [4] implemented both the multiple-view method and the object-warp technique. They were used in an i-Cone™ display system and in a four-walled CAVE. According to Simon et al., the technique allowed their displays to be used as true multi-user displays, providing users with the ability to look around and browse the virtual environment freely. They conducted an informal user evaluation with 50 users, most of whom are expert users of immersive visualisation systems, which confirmed that the omnistereo images provided a seamless image with correct stereoscopy in all view-directions. One of their i-Cone™ displays consisted of four projectors, each providing a images covering a horizontal FOV of 60° coverage, providing a total FOV of 240°, and consisting of four 15° views. The rendering time per projector was reported to be twice as long as with conventional stereo. The panoramic virtual environments were shown to groups of 3 to 10 people. The users were asked to compare the stereo impression of a virtual environment rendered in omnistereo with that of the same scene rendered in regular stereo with a fixed view direction. For the omnistereo rendering, users reported a better quality of stereo, a wider FOV and that they had looked around more in it. No artefacts, such as visible seams, were observed and the only negative remark was about slower rendering speeds. The four-walled CAVE consisted of three vertical walls and one

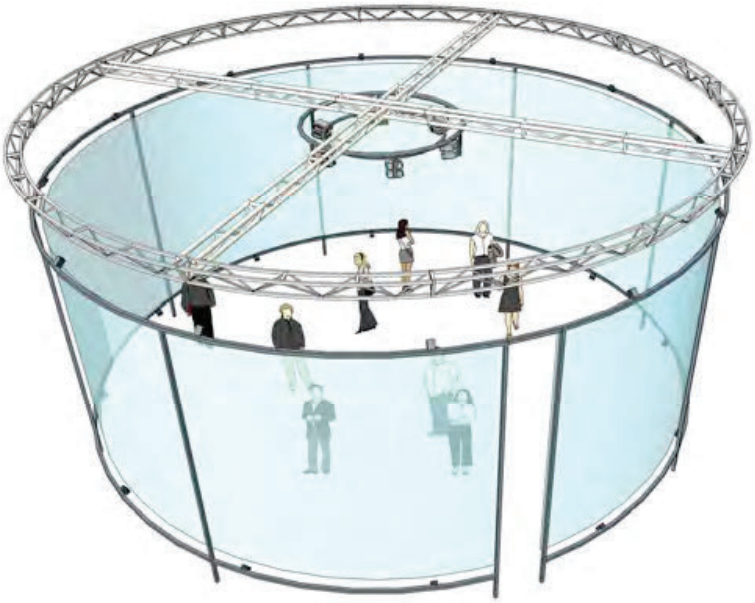

Figure 4.13: The AVIE omnistereo VR theater with multiple users inside. Image taken from McGinity et al. [41]

floor and the test scene consisted of a car interior as seen from the driver's position. The object-warping took place on the GPU inside a vertex shader and had a rendering overhead of less than 20°. Objects closer than ½ the distance to the display were reported to be visually distorted, objects further away appeared acceptable. Additionally, objects underneath the viewer were difficult to view. An open question that was raised by the authors was how the lighting, in particular the specular highlights, should be handled during the omnistereo rendering.

4.6 Omnistereo Rendering in Curved Display Systems

Generating stereo images for planar projection surfaces is well known. However, stereoscopic rendering for non-planar projections, such as cylindrical and spherical projections, is still a challenge [40]. Creating an omnistereo rendering method for such surfaces is an additional challenge, since multiple centers of projection have to be used therein. Omnistereo techniques have been developed for full 360° cylindrical displays, as well as for conical displays.

AVIE

A 360° VR cinema called *AVIE* was described by McGinity et al. [41] in 2007. It combines real-time omnistereo projection with surround audio and marker-less motion

tracking for up to 20 users; Figure 4.13 shows the general setup. It was designed to display both video content and real-time computer graphics. According to the McGinity et al., the AVIE was the first only-360° cylindrical stereo VR theater. Important goals of the project were: a high level of immersion, support of up to twenty simultaneous users and allowing physical activity and group interaction while staying low-cost. A cylindrical screen surface was chosen for the cinema because the authors saw it as advantageous for projecting omnistereo images. The display uses a total circumferential resolution of around 7500 pixels and polarising filters are used to separate right and left eye images. In total, twelve projectors and seven PC's are used in the setup. Due to the cylindrical projection surface, techniques using a distortion mesh and blend textures had to be used to render seamless, distortion-free images across the entire surface. Tracking is performed using twelve infra-red cameras, twenty infra-red flood lights and a cluster of four Linux operated PC's that run the tracking algorithms in parallel, providing tracking in real-time. The tracked movements and gestures act as the primary human-computer interface. The omnistereo rendering of the AVIE is based on the work by Simon et al. [4]. Due to this, the tracked head positions and orientations of viewers are not used to change the projection. Instead, the view point is assumed to always be located at the center of the cylinder and the omnistereo is supposed to allow users to freely move and look around wherever they choose. The image distortions that arise from the discrepancies between the viewer's position and the assumed viewpoint are less perceptible in a cylindrical displays than they are in a composite planar systems, such as a CAVE. McGinity et al. [41] stated that this is the case since the distortions are continuous over the whole screen in a cylindrical display. According to observations by McGinity et al. based on the experiences of more than a hundred visitors of the theater, the omnistereo images can be observed comfortably from any position inside the AVIE theater. However, they mentioned that moving viewers perceive motion in the image when there is none, which is due to the lack of motion parallax. The effect is most evident when there are distinct foreground and background objects in a scene.

i-Cone™

The i-Cone™ is a panoramic curved screen display system for virtual environments [42]. It features a horizontal FOV of up to 360° horizontal FOV. The screen of the display is a conical section with an opening angle of 5°. Before this system, curved screen systems have not been commonly used for this purpose. According Simon and Gobel [42], this is due to the lack of workspace for standing participants and due to bad acoustics. A typical curved screen installation used a three-channel setup providing a horizontal FOV of 160° and no floor projection. Their approach optimises screen geometry and projector placement allowing to create a front projection curved screen system that has an extended workspace for standing participants. A motivation for using curved screens is given by the fact that they distribute the geometrical errors from the untracked viewing position smoothly over the display area. This makes them inherently better suitable for big audiences than non-curved screens. The display system can include floor projections and spatial audio. The geometry of the i-Cone™ screen is described

Figure 4.14: The AVIE used as part of a mining training simulator. Image taken from McGinity et al. [41]

as a conic section with 5° opening angle, Figure 4.15 shows a four channel display of this type. One of the installed display systems had a height of 2,8 m and a radius of 3,3 m and a horizontal FOV of 240°. Due to the curved surface, the CRT projectors used in the system require distortion correction, which was handled statically by geometric adjustment using physical marking of the screens with reference points. The projected images from the four projectors are blended to create a seamless view. Two different hardware setups were used for the image generation: a cluster of PC's including four dedicated graphics cards, rendering to a resolution of 6200×1460 pixels, and a twelve-processor four-pipe SGI Onyx2 system rendering at a resolution of 5600×1320 pixels. Neither of these two systems offered floor projections, although the possibility to add such projections was mentioned by Simon and Gobel.

Omnistereo rendering

Simon and Beckhaus [43] developed an omnistereo projection technique for the i-Cone™ display system. The technique allows rendering omnistereo images in real-time, providing stereo vision for all directions without head-tracking for multiple participants simultaneously. The approach used by Simon and Beckhaus is based on the circular projection technique by Peleg and Ben-Ezra [35], which is mentioned in Section 4.2. While perspective projection forms an image by rays converging in a single viewpoint, the circular projection forms it using rays that are tangential to a viewing circle. A comparison of these techniques is shown in Figure 4.16. In the circular projection, the baseline for every view direction is perpendicular to the viewing direction. Therefore, the resulting omnistereo images provide a parallax for all viewing directions. The real-time rendering approach approximates the circular projection stereo image by rendering slices. The amount of slices required for rendering a sufficient approximation depends on the approximation error of the sliced perspective image compared to the true circular projection.

Figure 4.15: The 240° i-Cone™ display system. Image used with permission from Simon and Gobel [42]

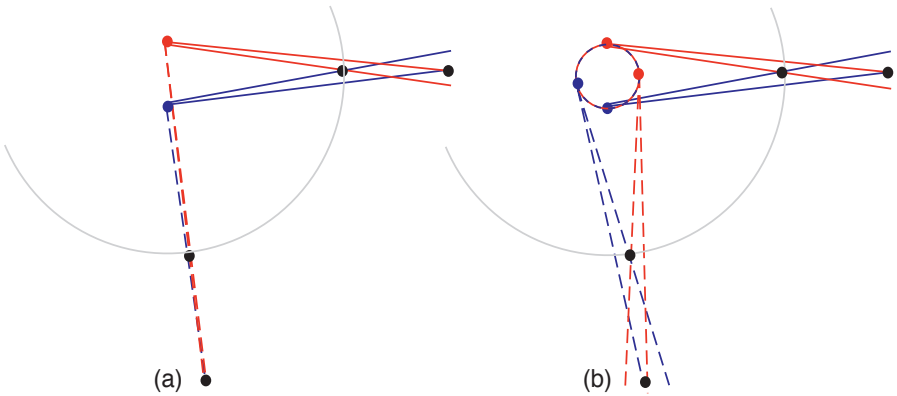

Figure 4.16: Comparison of the perspective projection (a) and the circular projection (b). Image used with permission from Simon and Beckhaus [43]

For the i-Cone™ display, four slices per projector were used, resulting in 15° horizontal FOV coverage per slice. The reported overhead for this setup rendering on an SGI Onyx2 for a typical scene is less than 50%. A vertex-shader based method called *object warping* has been suggested as an alternative method to render omnistereo images [4]. Models can be pre-warped in order to achieve omnistereo rendering using regular perspective projections thereafter.

McGinity et al. [41] implemented the multi-view and the object-warping omnistereo

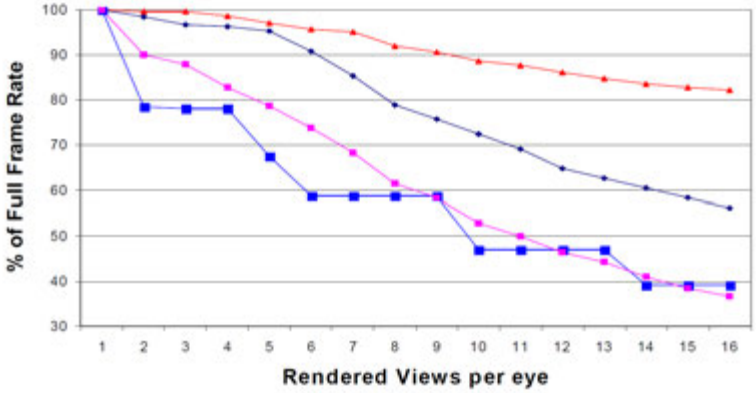

Figure 4.17: Benchmark showing the decrease in frame-rate depending on the number of views. Four different scenes of varying complexity were tested. Image taken from McGinity et al. [41]

techniques that had been suggested by Simon et al. [4]. The object warping method was implemented as a CPU and GPU version. For the multi-view method, the horizontal FOV of each view may not be larger than 11° in order to maintain an acceptable continuity between the views. The AVIE projectors cover 66° horizontal FOV each, therefore six to eight views per eye need to be rendered per projector. Figure 4.17 shows a chart displaying the frame rate reduction for different scenes and different amounts of rendered views. The CPU object-warp method is more efficient and can be rendered in less batches but requires special attention when using particle systems or calculations done inside vertex shaders. The GPU object-warp method, however, can be integrated into existing vertex shaders. The AVIE was used to display the following projects:

- Stereo panoramic still images using slide shows
- Panoramic videos captured with a panoramic video camera
- Stereo panoramic video, which is a composition of computer generated stereo images and still panoramic photographs. The project *Hampi* featured still panoramas of archaeological sites in India that were combined with pre-rendered CG characters.
- Mining Training Simulator - a simulator for the UNSW School Of Mining Engineering, shown in Figure 4.14
- TVisionarium - a prototype of an immersive televisual data-mining application

Lorenz and Döllner [44] presented a real-time rendering method for non-planar projections with a single center and straight projection rays [45]. Their goal was to provide an optimal and consistent image quality at real-time frame rates. Their method removes the need for image re-sampling, approximating the projection by a set of perspective

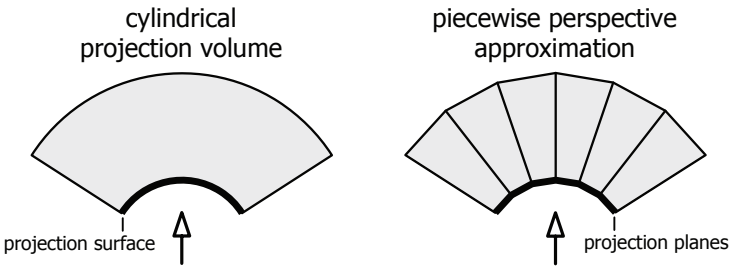

Figure 4.18: Top-down view of cylindrical projection volumes. The left image shows the ideal volume, the right image shows a piecewise approximation using perspective projections. Image taken from Lorenz and Döllner [44]

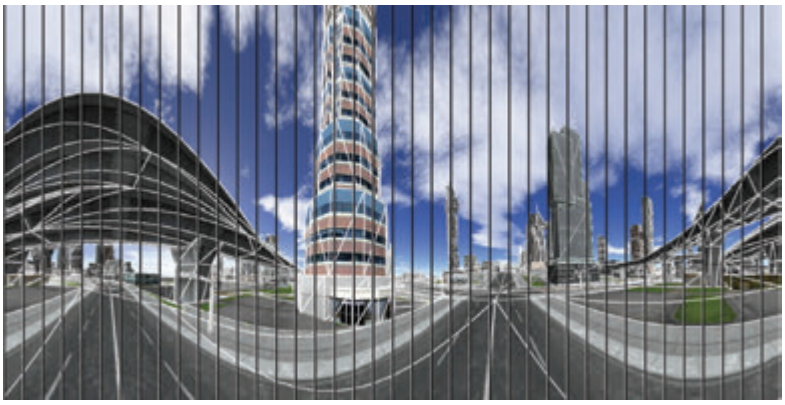

Figure 4.19: The image diplays the rendering of a scene using the piecewise perspective projection method. The curved projection volume is split into narrow rectangular vertical slices in order to approximate an omnistereo projection. Image taken from Lorenz and Döllner [44]

projection pieces. Each piece itself can provide optimal image quality. The combination of the pieces creates the desired projection. The piecewise perspective projection method is possible because a cylindrical projection only uses a non-planar projection in the horizontal direction, whereas in the vertical direction a regular perspective projection is used. Therefore, changing the horizontal edge length of each slice rendered with a perspective projection allows controlling the quality of the approximation. The method splits the curved projection volume into narrow rectangular vertical slices, as visualised in Figure 4.18. The scene's primitives are replicated on-demand and rendered in those projection pieces, in which they were determined to be visible. Therefore, the method is independent of the input mesh density regarding the quality of the projection. Figure 4.19 displays how a virtual environment is rendered using slices, based on the piecewise perspective projection method.

The original implementation ran on the GPU inside a geometry shader by using

transform feedback. Since then, graphics hardware with tessellation shaders have become available, which allows having a single-pass implementation of piecewise perspective projections.

A considerable drawback of the technique is the high geometry processing overhead. The vertex processing that happens after the primitive replication create a bottleneck for the rendering performance, especially in scenes with a high amount of polygons. Compared to an image-based projection, the piece-wise projection method turned out to be significantly faster in some setups, but significantly slower in others. The performance of the methods depends on the amount of slices, the amount of primitives in the scene and the targeted image size. However, the piece-wise projection method provided a better image quality than the image-based method in all of the comparisons.

The multi-view omnistereo method suggested by Simon and Beckhaus [43] can be combined with the piece-wise perspective projection technique by Lorenz and Döllner [44]. This would allow rendering the view-slices in a single pass, increasing the performance of the method.

Trapp et al. [40] adapted both an image-based rendering technique for non-planar surfaces and the geometry-based mesh-refinement technique by Lorenz and Döllner [46] to output stereoscopic images. The techniques were then compared. In addition, their goal was to create omnistereo panoramas with the techniques. The image-based method uses normal-based image warping. A cube-map texture is dynamically generated to capture the virtual environment surrounding the scene's camera. The cube-map faces are rendered with parallel projections, since off-axis projections toe-in cameras would lead to artefacts or missing stereo disparity. A projection function is used to sample the cube map using computed normal vectors. The method runs on the GPU, implemented in a fragment shader program using a post-processing pass. A full-screen quad is used for the rendering. A naïve approach for stereo rendering is the creation of two cube maps using two rendering passes. Stereo rendering is then performed by computing two non-planar projections. Trapp et al. [40] proposed to create two cube map textures within a single rendering pass to avoid unnecessary state changes. Figure 4.20 gives an overview of the image-based stereo rendering method.

The geometry-based approach uses geometry shaders to apply view-dependent tessellation with limited edge length per primitive to all primitives of the virtual environment. This dynamic mesh refinement is necessary to ensure sufficient vertex density on the screen. Each primitive is emitted twice using separate projections, directing them to two different layers. The geometry-based approach only needs a single scene evaluation to output the stereo image pair. Furthermore, it supports omnistereo rendering. They concluded that the geometry-based approach has a number of advantages as compared to the image-based one, such as providing better quality, better performance and allows for more functionality with respects to stereo rendering. In addition, the geometry-based approach can be used to create full 360° omnistereo panoramas, whereas the image-based approach is not suited to create omnistereo images.

Hernández et al. [47] implemented a multi-slice rendering approach for omnistereo rendering in the AVIE. The viewer's head position for all their implemented techniques

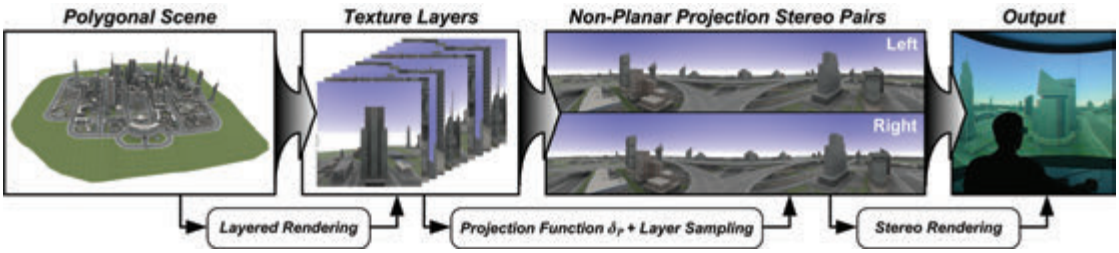

Figure 4.20: Overview of the implementation of the image-based stereo rendering for non-planar projections. Layered rendering of the cube maps is used to create image representations of the virtual environment. These images are then sampled using a projection functions to generate the stereo image pairs. Image taken from Trapp et al. [40]

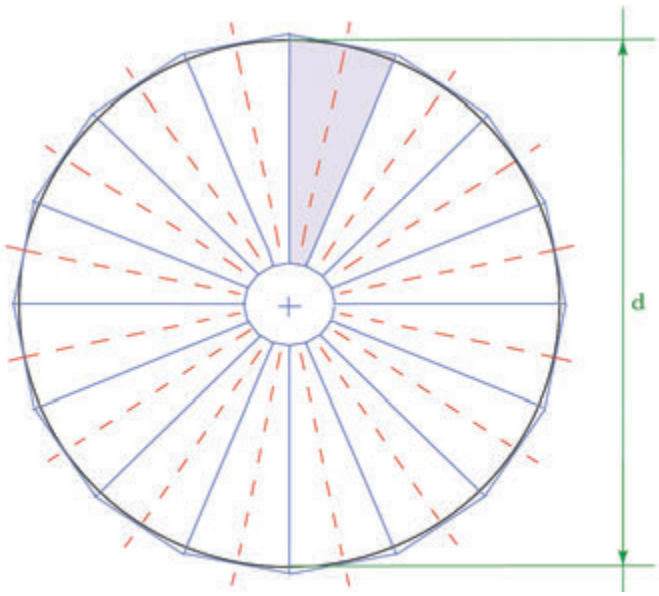

Figure 4.21: Top-down view of the multi-slice approach for omnistereo rendering in a cylindrical display. Image taken from Hernández et al. [47]

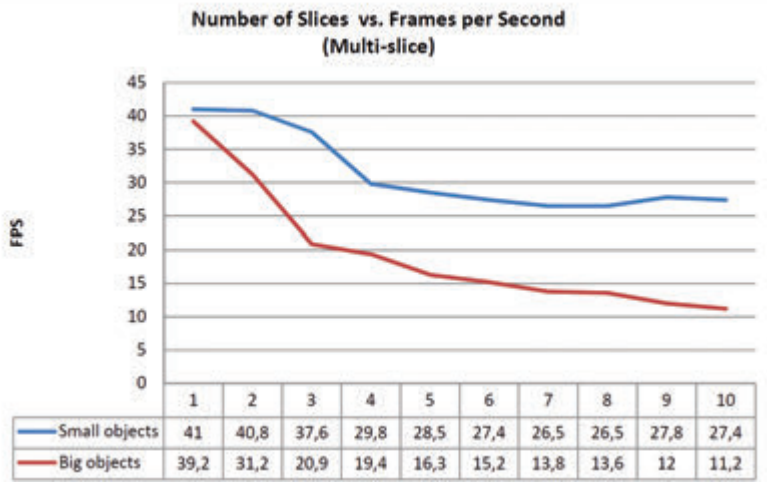

Figure 4.22: Benchmark of FPS depending on the amount of slices for the omnistereo approximation. Objects that appear large on the screen affect the FPS differently than objects which appear small. Image taken from Hernández et al. [47]

was assumed to be located in the center of the cylinder and to be static. Figure 4.21 shows a top-down view of the omnistereo projection of the multi-slice approach. Figure 4.22 shows a benchmark of the frames per second (FPS) depending on the amount of slices per view. The GPU used for the benchmark was a Nvidia(R) 330M. The scene consisted of 24.000 textured triangles, which were rendered separately for each eye resulting in a total of 48.000 triangles per stereo pair. The benchmark was done in two modes, one with small objects, i.e. objects that do not span over a large distance on the screen, and on with big objects. Objects that span across multiple view-slices will not be affected by the frustum-culling as often, therefore affecting the rendering performance more than objects that appear small on the screen. Amounts of slices that allow achieving an acceptable approximation of the omnistereo projection without visible artefacts typically reduce the rendering performance significantly, as compared to a single-view rendering.

Since the multi-slice approach did not provide sufficient performance, Hernández et al. [47] implemented a vertex-shader based omnistereo approach for rendering to cylindrical screens. Figure 4.23 shows how the vertices are projected on the screen in their shader-based approach. Instead of using a perspective projection matrix, their method transforms the vertices directly into clip space using a conversion implemented in the vertex shader. The vertex positions are converted from the Cartesian coordinates to a cylindrical coordinate system inside the shader for this purpose. Given that vertices that are outside the view frustum of the non-omnistereo projection could be inside the view frustum for the omnistereo projection, the view frustum culling was modified to cover a wider FOV depending on the interocular distance. Moreover it should be noted that other than with the multi-slice approach, the interpolation can become an issue with the shader-based approach when polygons cover a large horizontal FOV on the

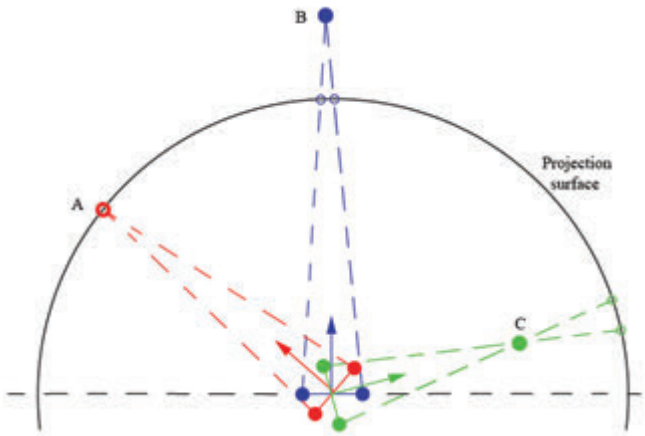

Figure 4.23: The right image shows how points are projected onto the screen in the shader-based omnistereo rendering for cylindrical displays. Image taken from Hernández et al. [47]

screen. This occurs due to the interpolation done between the vertices being linear, which is correct for linear projections but wrong for the non-linear projection performed by the shader. The results of the shader-based method showed a higher accuracy than the multi-slice approach, a seamless projection and no artefacts.

4.7 Omnistereo Rendering in Domes

Hemispherical domes, which can be small personal domes, such as the iDome, or large domes, such as used in planetariums, offer an alternative to the CAVE and cylindrical screen systems [48, 49, 50]. Historically their use has been restricted to large planetariums, primarily for public education in astronomy. The planetariums that had the ability to present real-time digital graphics only projected the imagery on a small portion of the dome using a single projector. In later developments, multiple projectors were used, achieving full dome projections. The movies shown in such setups featured fisheye images. With the advancement of graphic cards technology, the content was not limited to movies and images any longer and real-time interactive graphics could be shown, opening possibilities for VR applications. Planetariums commonly use multiple digital projectors. This enables a higher maximum resolution for the combined result of the projected images. However, in a single person dome this is a less attractive option due to space constraints and difficulties in arranging the projectors without interfering with the viewer. Additionally, using multiple projectors requires careful alignment and edge blending across overlapping projection regions, which can be problematic.

In comparison to cylindrical displays and CAVES, domes offer a similarly large FOV and can feature stereoscopic projections. A disadvantage of CAVEs is that they have corners that are difficult to hide and may make viewers aware of the projection sur-

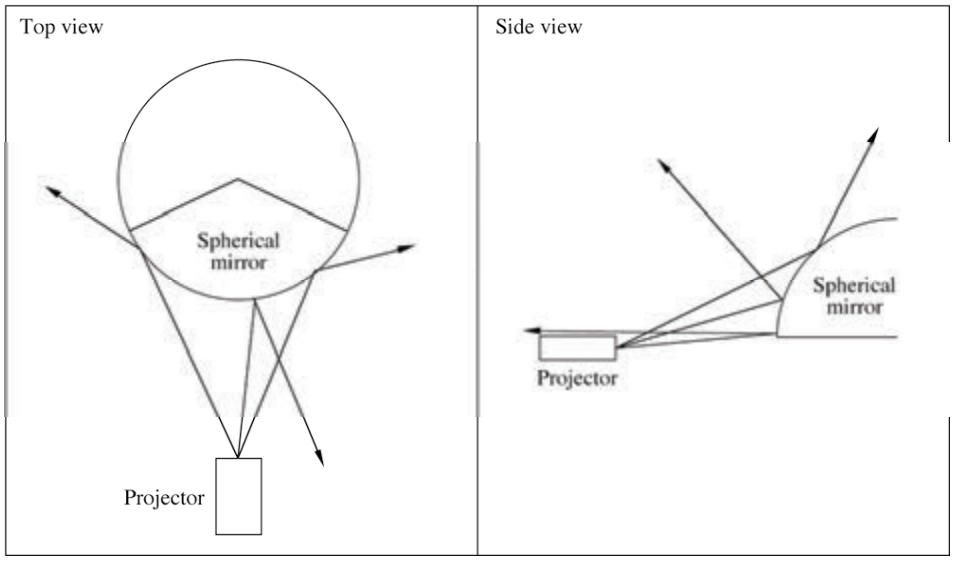

Figure 4.24: Representation of the path of rays from a projection source, reflected off a spherical surface. Image taken from Bourke [48]

face, which can reduce the illusion of being somewhere else if not hidden well enough. Cylindrical displays and domes have seamless projections. Cylindrical displays can wrap around the viewer providing a complete 360° horizontal FOV, but they generally have a limited vertical FOV.

In order to make domes less expensive, smaller domes of around 10 m in diameter, as well as even smaller easily transportable, inflatable domes of around 5 m in diameter have been developed. A single projector with a fisheye lens, which was located in the center of the dome, was employed in such systems. The disadvantages regarding the use of fisheye lens projectors are the lower resolution and brightness, as well as chromatic aberration. Nevertheless, they are easier to manage and cheaper than multi-projector installations. Bourke [48] introduced an alternative projection system that reduces the cost of dome projection while maintaining a similar quality by using spherical mirror projections. A spherical mirror can reflect light from a rectilinear frustum, as produced by regular consumer-grade projectors, over almost the whole surface of the dome. Just like a fisheye lens, the spherical mirror scatters the light across the surface of the hemisphere. Figure 4.24 displays the path of the rays before and after the reflection from the mirror. Different placements of the projector and mirror are possible, such as placing the projector close to the rim of the dome, in the middle of the dome or placing the mirror at the base of a vertically mounted truncated dome. Alternative geometries have been proposed, such as using a convex mirror to project into cylindrical environments or using polyhedral spaces. Benefits of using the described setup instead of a fisheye lens include:

- The projection hardware can be located away from the center of the dome. As

this is generally a location that provides a large tolerance to viewpoint deviations in respect to the resulting view distortion, this can be of advantage, especially for single person domes.

- A fisheye lens can typically only be fitted to a very narrow range of projectors. By having the projector and the optics separated, it is possible to choose from a wider selection of projectors. However, a good depth of focus has to be provided by the projector, due to variations in path lengths when using a mirror.

- The coverage on the dome can be controlled. However, neither the mirror-based system nor the fisheye lens projectors can cover the whole dome. The latter due to pixel efficiency reasons.

- Higher resolutions and complete coverage can be achieved using multiple projectors and mirrors. For example, a dual mirror and projector system would have a single blending zone in the middle of the dome.

- In contrast to fisheye lenses, a good quality mirror does not cause chromatic aberration.

- Mirrors are less expensive than fisheye lenses for projectors

Images projected onto the dome surface need to be warped to look correct and undistorted. A warping mesh is used for this purpose, which is created by finding the point on the projector frustum for any point on the dome. Bourke [50] describes the creation of the distortion as being relatively straightforward. The mesh used for projection calibration can be seen in Figure 4.27. This makes precise image warping possible and can be done in real-time on modern graphics cards.

The iDome, developed by the iCinema Centre, is a single person dome display system that uses spherical mirror for the projection instead of a fisheye lens. The hemispherical surface completely surrounds the viewer [50]. The mirror is placed behind the dome and can be almost completely hidden. Figure 4.25 shows the projection setup of the iDome. Applications for the iDome include astronomy visualisation, driving simulators, exploration of molecular datasets (as shown in Figure 4.26), architectural visualisation and virtual heritage.

Bourke [49] described the most common omnidirectional displays to be cylindrical screens. Since omnidirectional projections give a single user the ability to view direction in stereo without requiring head-tracking, an immediate consequence is that multiple participants can potentially be supported by such omnistereo screens with each one looking in different directions. The stereo view is only correct for one viewing position, which is the case both when using planar displays as projection surfaces, and when using cylindrical displays. For the latter, the viewing position is typically located at the center of the cylinder. The region in which the distortion and parallax errors, which are induced by moving away from this optimal viewing position, are not an issue is larger for the cylindrical displays. However, not even a single user located at the optimal viewing

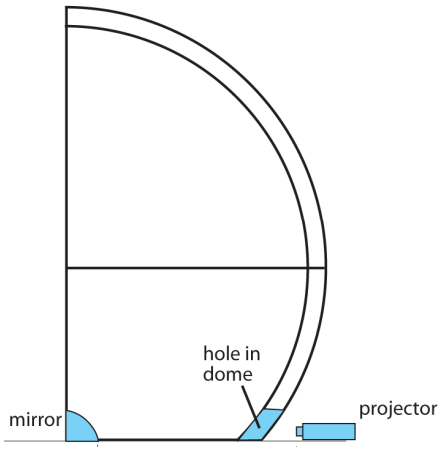

Figure 4.25: An illustration of the projection geometry of the iDome using a mirror for projection. According to Bourke [50], not all projectors meet the focus requirement for this setup. Image taken from Bourke [50]

Figure 4.26: Example usage of the iDome: The exploration of molecular datasets. Image taken from Bourke [50]

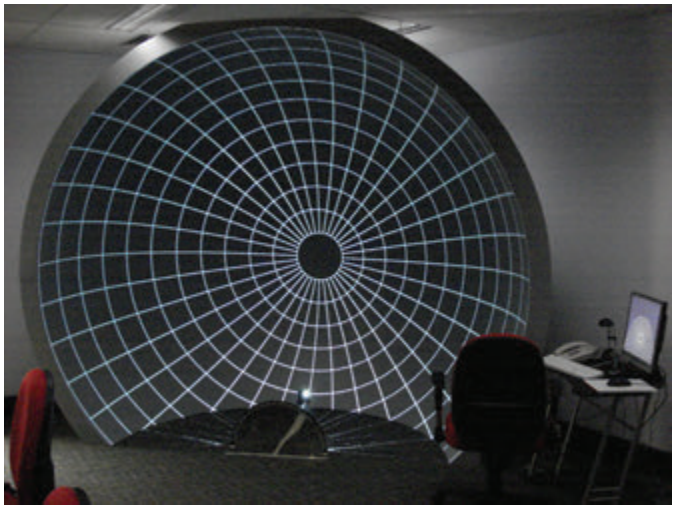

Figure 4.27: The polar mesh used for projection calibration. Image taken from Bourke [50]

position of the omnistereo displays is presented with completely correct stereo images. Towards the edges of the observers' FOV the parallax of the images is lower in reality, as well as in regular stereoscopic rendering. This is not the case when using omnistereo projections. For example, the parallax of an imagery at 90° of the viewer's view direction would be 0. However, Bourke [49] describe this as not being an issue in general and that it does not induce serious eye strain because of the limited FOV imposed by the stereoscopic glasses. Despite the fact that the observer might see imagery in their peripheral view, stereoscopic sensations are not perceived in that region. The stereoscopic glasses do not wrap around or block the imagery in that region, therefore the degree of immersion is unaltered. A simple way to create stereo fisheye images for a dome is to horizontally offset two standard fisheye projections and rotate the view direction so that the zero parallax occurs at the correct distance along the view direction, as shown in Figure 4.28. This is analogous to the toe-in stereo rendering approach for planar screens. An issue with this approach is that for a 180° FOV the edges between the two cameras do not match. In order to solve this, it has been suggested to over-render the fisheye and rotate the images. However, a persistent problem is the incorrect parallax information that occurs when the viewers looks away from the central view direction. Views perpendicular to the central view direction have no parallax information. The problem are analogous to the problems occurring with planar toe-in stereo projections. Off-axis fisheye projections, as shown in Figure 4.29, are most commonly used to provide users of single-user domes with an undistorted view within the dome. The user does not have to be located in the center of the dome to get a correct view. The stereoscopic fisheye projection images created by the off-axis and toe-in methods provide satisfactory depth cues only for the "forward" view direction. Objects that are located horizontally towards the edges have different scaling, if comparing the left and the right eye's views, and a decreased parallax.

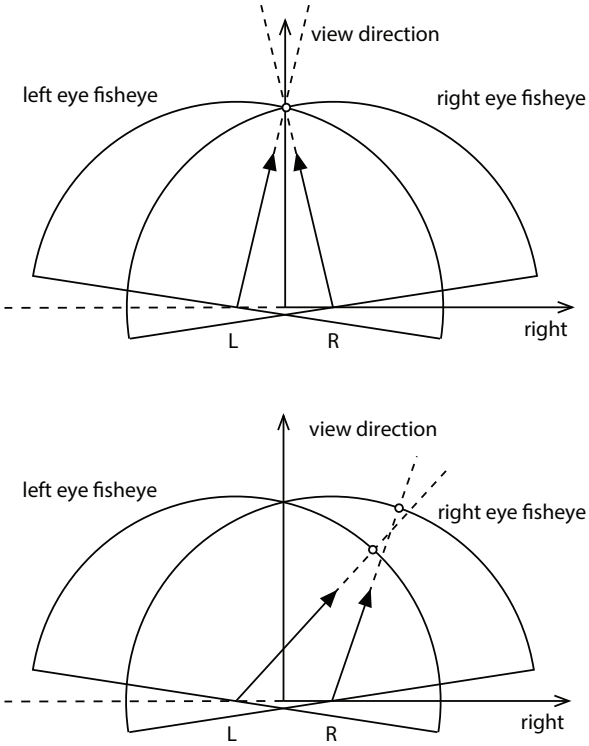

Figure 4.28: Toe-in stereo fisheye projections. Image taken from Bourke [49]

However, for observers looking forward the difference in scale and the reduced parallax are not critical because this region lies out of the focus. Peripheral cues are given in a satisfactory way even when using omnistereo projections.

Bourke [49] suggested a way to create omnidirectional stereoscopic fisheye images. The method meets the requirements of both smaller personal domes such as the iDome and directional planetariums. Directional planetariums are planetariums in which the seating and dome orientation is laid out in a way that all of the audience is essentially looking in a similar direction. The omnistereo fisheye images are created by mimicking how the viewer rotates their head around the vertical axis (the up axis) when looking around in the hemispherical dome. This eye-axis rotation defines the eye positions for any possible view direction. This allows maintaining the correct parallax information for any localised area on the display. At the middle of the FOV, the approach gives correct results but it degrades away from the middle. However, the reduction of the FOV, which happens due to the stereoscopic eye-wear, diminishes this error. The geometry for the view rays of the omnistereo projection, as seen from top-down, is displayed in Figure 4.30. The techniques and algorithms were tested in an iDome using anaglyph techniques, as well as with frame sequential stereo using shutter glasses. Figure 4.31 shows a comparison of the off-axis stereo fisheye projection and the omnistereo fisheye projection. The images are rendered from the same view point; the left and right eye

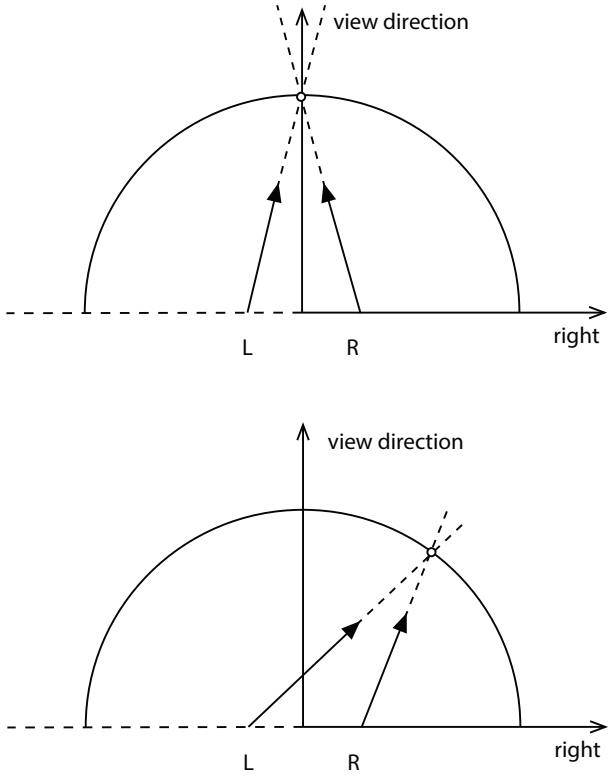

Figure 4.29: Off-axis stereo fisheye projections. Image taken from Bourke [49]

image are superimposed. The omnistereo projection is the only type of projection that provides acceptable depth perception irrespective of the viewing direction of the observer.

The calculations of the omnistereo pairs is done in view space coordinates. The origin of this coordinate system represents the center of the viewer. The coordinates are converted to polar coordinates, in which the computations take place. Bourke [49] suggested that for content that is rendered offline a modification of a ray-tracer can be used. For real-time rendering, a different technique is required due to the computational intensity of ray-tracing methods. Analogous to the techniques for cylindrical screens, described in Section 4.6, one way to accomplish this is to render vertical slices that are composed into a final image. Another way is to use a vertex shader that adjusts the geometry, which allows rendering the scene in a single camera render pass. As this results in a non-linear projection, the interpolation done on the GPU hardware will be incorrect, as it linearly interpolates between vertices. Vertices that stretch across large distances in screen space can therefore require tessellation in order to create a sufficiently correct omnistereo effect. Bourke [49] stated that this tessellation needs to be performed on the CPU resulting in significant amounts of geometry that have to be transferred to the GPU. This is, however, not the only solution since tessellation can be now done on the GPU using tessellation shaders. Potentially, this may remove the drawback that arises

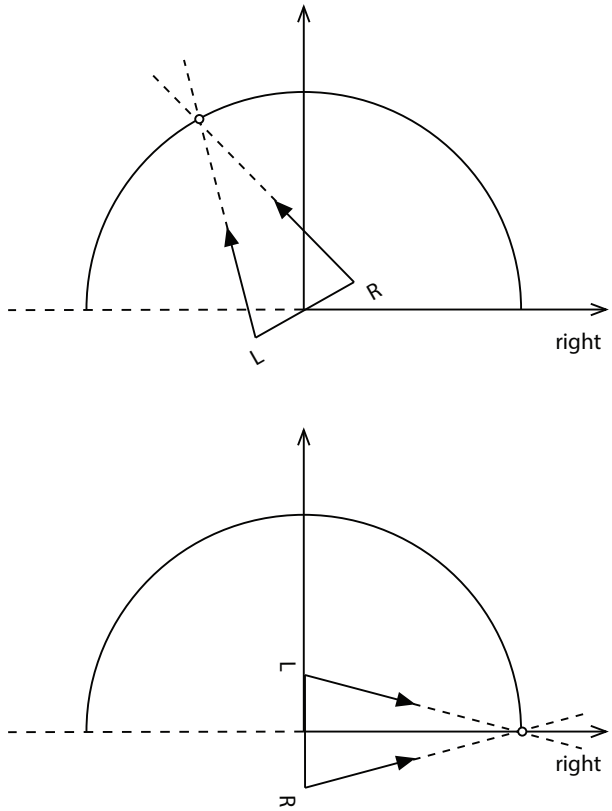

Figure 4.30: Top-down view of the view rays for the rotated eyes in the omnistereo projection. Image taken from Bourke [49]

from having to transfer the tessellated geometry. Our omnistereo rendering approach for CAVE displays, which will be described in Chapter 7, for example, runs in a single pass on the GPU and adaptively tessellates the geometry for the omnistereo projection solving the mentioned issues for planar screens.

4.8 Multi-Viewer Stereo Displays

As mentioned in Chapter 3, regular stereoscopic rendering creates a correct perspective only for a single head-tracked viewer. To provide a correct stereoscopic projection for two or more tracked viewers simultaneously, multi-viewer stereo displays have been proposed by Fröhlich et al. [51]. Using this system, the users can operate and collaborate in the same interaction space. The display was originally designed for two to six users. This can be achieved based on shutters only. An alternative is to use shuttered LCD-projectors combined with polarised stereo glasses. This combination requires half the shutter frequency and provides double brightness, as displayed in Figure 4.32.

The authors describe the combination of LC-shutter and polarisation as being the

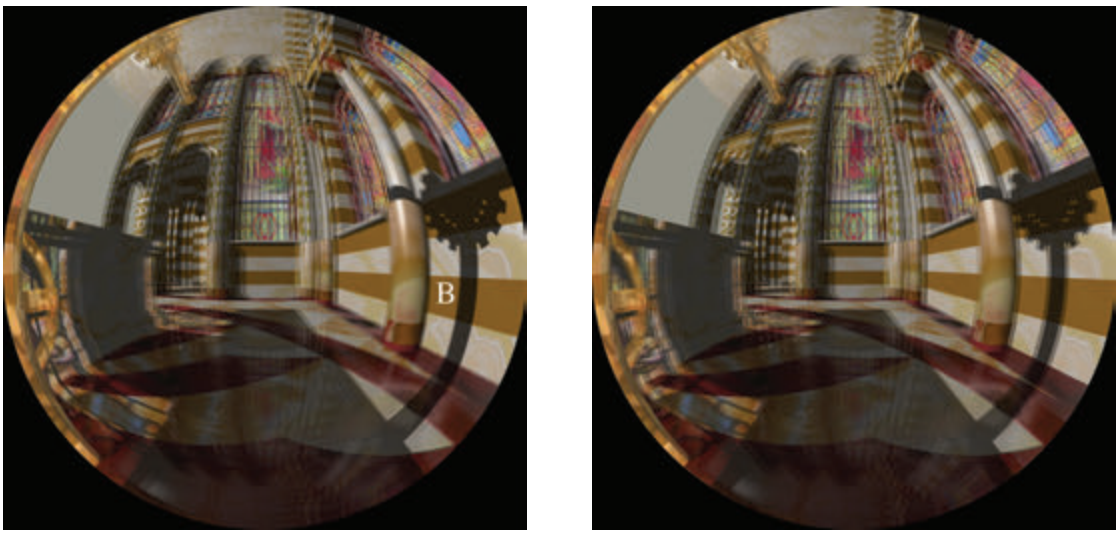

Figure 4.31: Left: Off-axis stereo projection with a lack of parallax towards the sides, e.g. at position B. Right: Omnidirectional stereo fisheye pair, in which the horizontal parallax is equally present at the sides and the center of the screen. Image taken from Bourke [49]

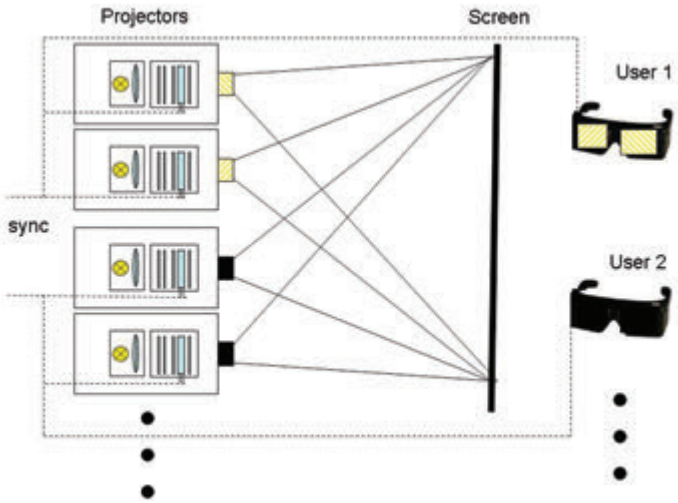

Figure 4.32: A combination of a LC-shutter stereo system with a polarised stereo system. When the stereo images for user 1 are shown, the other users' images are not projected. Image taken from Fröhlich et al. [51]

most promising setup of those they have tried. Nevertheless, even if using this technique, four-user setups have noticeable disadvantages. At higher shutter frequencies, the images can appear too dark, since the light which is emitted by the projection system is distributed over the amount of views. Visible flickering can occur at lower shutter frequencies. Therefore, the maximum feasible amount of users is very limited. Pross et al. [52] presented a multi-view stereo display based on pulsed LED light sources of a set of multiple LCD projectors. This display technology optimises the multi-view stereo technique towards higher brightness and minimises crosstalk by allowing high frequency switching. Instead of using mechanical shutters in front of the projectors, only one of the LED projectors emits light at a time. Nevertheless, neither of these techniques is suitable for larger amounts of simultaneous viewers.

CHAPTER 5

Object-Warping Based Omnistereo Projection

In the following two sections, object-warping based omnistereo projections for the CAVE are described. One method is based on off-axis projections and one is based on a single COP and can be used with an on-axis projection. The object-warping method, which is used to generate omnistereo images by modifying the vertex positions before the perspective projection, has originally been suggested by Simon et al. [4], whose work is described in Section 4.5. The original object-warping, discussed in Section 5.2, was proposed in combination with off-axis projections. In Section 5.1 we show that the method can also be based on-axis projection, leading to the same omnistereo projection images as a result. In the course of the following sections, we discuss these object warping methods and attempt to illustrate the geometric background in greater detail than in the original paper, specifically by providing several corner-cases. The previously undescribed issues that arise due to the non-linearity of the perspective are described in detail and are shown. These artefacts result from the fact that the approximation is only correct at the final vertex position, whereas the edge between the vertices is interpolated linearly, which is an incorrect form of interpolation for the omnistereo perspective.

5.1 On-Axis Omnistereo Object-Warping

In this technique, the projective viewpoint is located at the center of the CAVE, and a regular on-axis perspective projection is used for the final projection. In object warping methods, the vertices of the model's mesh are warped before a specific projection takes place. The position of each vertex in the vertex shader program is changed in a way so that the perspective projection, which is applied thereafter, projects the vertex to the screen-space position of the desired omnistereo projection. The method in this section is based on the original object warping method suggested by Simon et al. [4] but uses on-

axis instead of off-axis projections. One benefit of being able to use on-axis projections is that this may lift the necessity to calculate new off-axis projection matrices for each screen whenever the head position changes, which also requires updating the shader attribute on the GPU each time. In case the used graphics library or engine does not support off-axis projections out-of-the-box, this technique may be better suited as well. Additionally, this method can be considered to be easier to illustrate and understand, since it features a single projective viewpoint located at a central fixed position, as opposed to the method based on off-axis projections.

Figure 5.1 shows the geometric construction of object-warping in a CAVE, as seen from a top-down view. The warping is based on a given vertex position V and the front-wall projection from the left eye's viewpoint $E(left)$ is shown. The object warping for the right eye's view and for the other wall screens works analogously and is therefore not shown. In the displayed arrangement, the head position O is located coinciding with the center of the CAVE, and thus with the on-axis projection's COP CC. $OtoV$ depicts the line that passes through O and V. The respective left or right eye's position, which is the position from which we want to actually see the vertex, has to be calculated for each vertex to achieve stereo in each view direction. Therefore the head position needs to be rotated towards the vertex's, which allows deducing the respective eye position knowing the eye separation distance. In the view-space coordinate system, the left or right eye's position can be determined by flipping the OtoV vector respectively to the left or right in the xz-plane. The distance of the eye from the head position O is ½ of the eye separation distance. The resulting eye position, the head position and the COP all lie on a line in this arrangement, as can be seen in Figure 5.1. Based on the calculated position of the eyes, the line $EtoV$ from the eye position to the original vertex position can be created. This line intersects the image plane at the point I. We define the *warp line* as the line onto which a vertex has to be moved in order to be projected to the omnistereo projection's screen-space position. Based on the intersection point I and the COP, the warp line can be created. Using a perspective projection from the projective view point, a warped point along this line is projected to its final position.

The exact position along the line should be chosen carefully, as it can affect depth sorting in the final image if objects occlude each other. The depth is defined by the distance of the object to the viewer and is determined by the view space z-coordinate in OpenGL and Direct3D. There is no intuitively correct solution to the problem of finding the right position but the depth order has to be correct in regard to the omnistereo eye position of the warped vertex.A naïve approach to define the depth order is, for example, to use the distance from the projection center or the eye position to V as the distance from the eye to V', thus retaining the original order. Alternatively, a line, parallel to the line connecting the eye positions and passing through the projection center, can be used to find an intersection by letting it pass through V and intersecting it with the warp line, defining the new position of the vertex. Figure 5.3 and Figure 5.2 display how V' can be determined using this approach. When using this method, similar triangles can be used to simplify the calculation: The triangle spanned by I, O and the eye position is similar to the one spanned by I, V' and V. Using similar triangles was first suggested by Simon

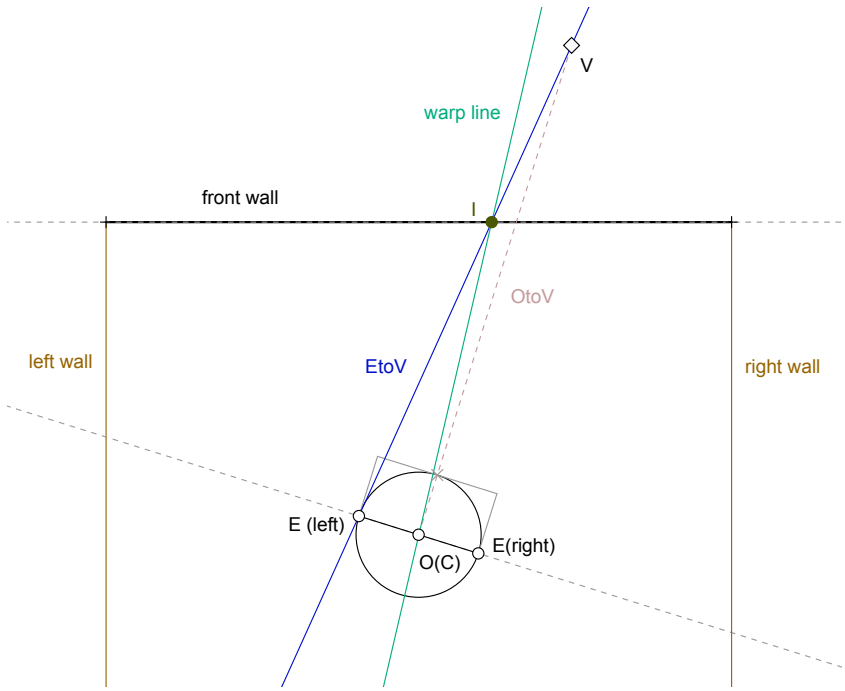

Figure 5.1: The geometry of the omnistereo object-warp technique for a CAVE, using an on-axis perspective projection from the CAVE's center. Eye positions are defined separately for each vertex, positioned perpendicularly to the line passing through the head position and the vertex. The *warp line* is constructed based on an intersection point on the image plane.

et al. [4] for omnistereo rendering based on off-axis projections, which is explained in Section 5.1.

Figure 5.2 shows that the similar-triangle approach still works in extreme cases. The shown case features a possible position for vertices resulting in the warp line being parallel to the projection plane, in which case the position of the point I is undefined. In case it is nearly parallel, the point I is defined but possibly located at very large, up to near-infinity distance from the projection center, which introduces numeric errors. Solutions for V' that depend on the calculation of the point I in the Cartesian coordinate system may therefore not always yield the desired result, whereas the similar-triangle based solution does.

Omnistereo warping for a central viewpoint using an on-axis projection by means of Figure 5.3 can be expressed in two equations and be further simplified. In this arrangement, the head position and the COP coincide and are located at the origin $(0, 0, 0)$ in view space. As previously described, the eye position can be retrieved trivially in view space by flipping the unit vector pointing from the head position to the vertex position and scaling it by ½ the eye separation distance and adding the result to the head position. In addition to the similar triangle relation mentioned previously, another

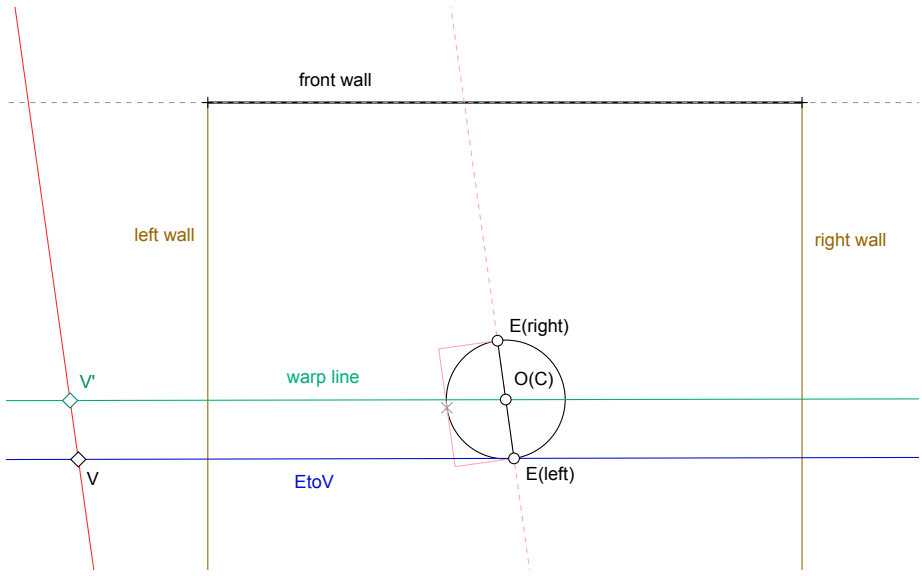

Figure 5.2: In some arrangements the position of the vertex (V) can cause the intersection point I to be undefined or located in near-infinity distance from the eye. A solution can be established that also provides a correct solution for such arrangements can be established by means of similar triangle relationships.

relation of similar triangles can be introduced, which allows further simplification of the calculation. The point resulting from the orthogonal projection of the eye position onto the projection plane is called E_p and the resulting point of the same projection done for the vertex position is called V_p. The triangle consisting of E_p, the eye position and I is similar to the triangle V, V_p and I. The respective eye position will be referred to as E and the COP (coinciding with O) as C. The following length-based ratios using similar triangle relationships can be constructed:

$$\frac{EC}{VV'} = \frac{EI}{VI} = \frac{EE_p}{VV_p} \tag{5.1}$$

The vector from the COP to V is known. In order to retrieve V' based on V, the vector from COP to V needs to be scaled by a factor and added to V. By means of the additional similarity that we established using the orthogonally projected points, the calculation of the factor f can be reduced to a single dimension, namely the depth, defined by the z-coordinate in view space. The distance of the projection plane from the center of the CAVE is denoted by d_{pp}. The following equation is based on a right-handed coordinate system:

$$f = \frac{V.z - (-d_{pp})}{(-d_{pp}) - E.z} = -\frac{V.z + d_{pp}}{E.z + d_{pp}} \tag{5.2}$$

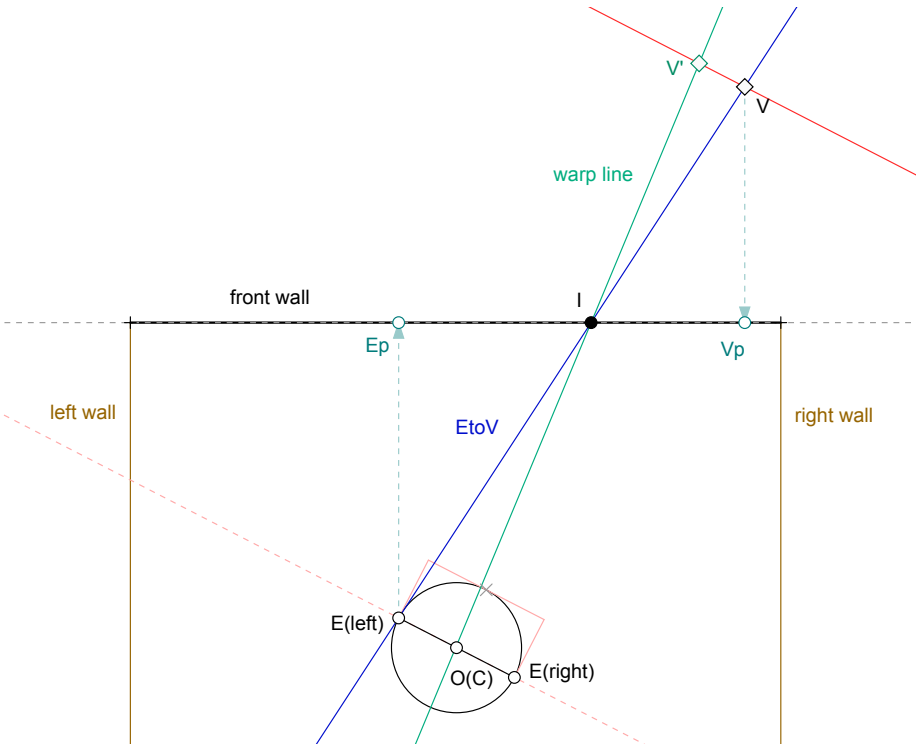

Figure 5.3: In order to find a solution for point V' along the warp line, a line, which is parallel to the line passing through the desired viewpoint and the COP C, can be drawn passing through V.

V' can be calculated as follows:

$$V' = V + f * \overrightarrow{CE} \tag{5.3}$$

The presented method allows warping an object for a fixed central view position. In order to support view positions different from the central one, the head position and the projection center have to be separated in the arrangement. In order to retain the triangle similarity in this changed setup, we construct a line parallel to the line EC, which passes through the respective eye position and the projection center. It can be noted that EC does not pass through the head position and the other eye position in this arrangement, since the head position has been moved away from the COP. Figure 5.4 shows this new setup. The equations 5.1, 5.2 and 5.3 can be used in this setup as well.

We implemented both the solution based on similar triangles and a method that defined the warp line vector based on the point I. The latter method produced frequent artefacts in our high-polygon virtual environment for triangles outside the view frustum, which is a result of the mentioned numerical instability of I. The introduced artefacts could be described as "triangle clutter", in which numerous triangles are stretched at apparent randomness over the screen, with their arrangement and dimensions changing

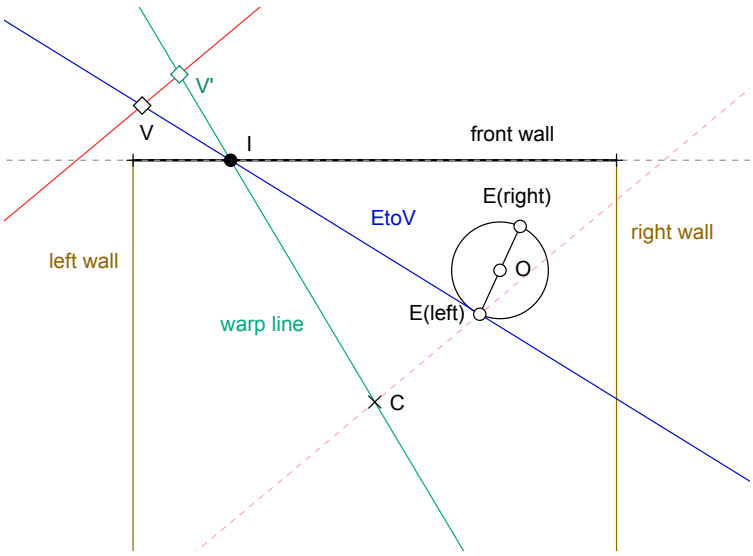

Figure 5.4: The changed geometrical setup for off-center view positions using an on-axis projection matrix.

whenever the view position is changed. This issue can be mitigated by falling back to an estimation for the point or to the original vertex position whenever I is undefined or at near-infinity distance. Since such a fall-back can cause further complications, we recommend only using the similar-triangle based solution.

5.2 Off-Axis Omnistereo Object-Warping

This section describes the object warping method as originally suggested by Simon et al. [4] in detail. The difference to the method presented in the previous section is that this one involves an off-axis projection of the vertex instead of an on-axis projection. The projective viewpoints do not have fixed positions, but vary for each eye and CAVE wall and depend on the head position in the CAVE. The view direction that defines the view positions for the off-axis projection can either be fixed per-screen or fixed across all screens. Simon et al. [4] suggested to use, for example, the direction perpendicular to the front wall's screen as view direction for all screens. The following figures and formulas are based on this suggestion. The described view-direction and the resulting projective viewpoints serving as COPs ($C_{(left)}$ an $C_{(right)}$) are shown in Figure 5.5. It should be noted that any arbitrary view direction, resulting in different eye positions used as projective viewpoints, could be used as basis. Similarly, a single head position could also be used as projective viewpoint, requiring only small changes to the calculations.

The arrangement of this setup is shown in Figure 5.5. $C_{(left)}$ and $C_{(right)}$ are the left and right eye's respective COPs, which are located at the origin $(0, 0, 0)$ in view space for the left and right eye's view respectively. The desired per-vertex view positions

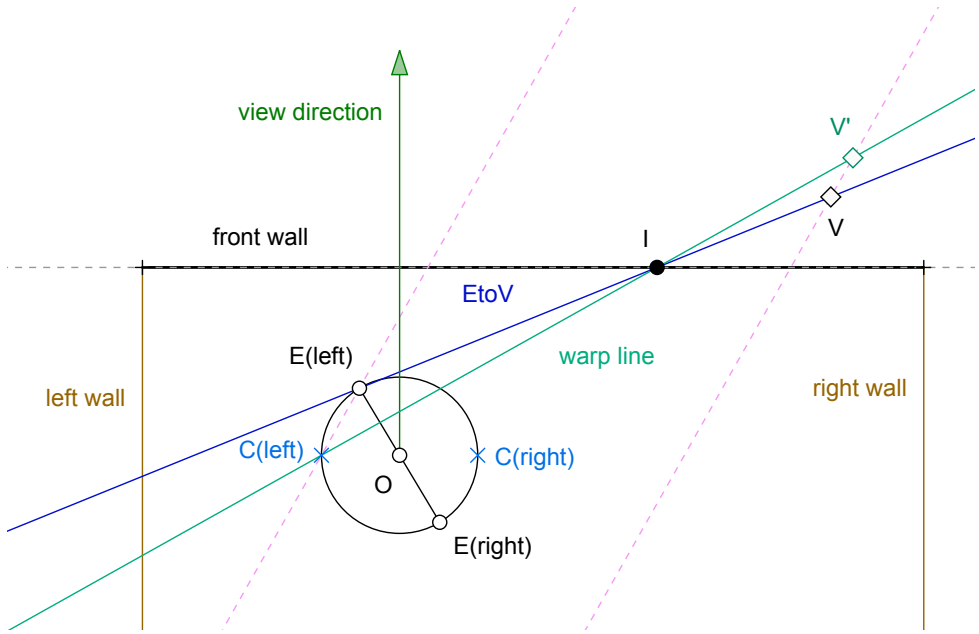

Figure 5.5: Omnistereo object warping based on an off-axis projection. The left and right eye COPs ($C_{(left)}$ an $C_{(right)}$) are defind by the head position and the fixed view direction, which is perpendicular to the front wall's screen.

are retrieved in the same way as in the previous method. The relationships shown in Equation 5.1 are equally true in this setup. Therefore, based on the right-handed coordinate system we can calculate the scaling factor for the vector VV' as shown in Equation 5.2 with a small change to d_{pp}. In the on-axis based method, d_{pp} denotes the distance from the center of the CAVE to the screen, which is the on-axis location of the COP. Since the COP's position in off-axis projections depends on the head position and view-direction, the distance to the screen d_{pp} needs to be calculated separately for each screen. Equation 5.3 can then be used to calculate V'.

5.3 Discussion of Vertex-Shader Based Omnistereo Rendering

We described two methods for performing object-warping of meshes in order to create omnistereo images. Both were implemented in our application using GLSL vertex shaders and generate the same omnistereo images for a given head position, providing a stereo effect in all view directions. Special attention should be given to frustum culling techniques, as they may erroneously cull faces when using the techniques. This can be solved by modifying the frustum size to encompass any polygons that may be visible after the object-warping, or by turning frustum culling off entirely.

In the work by Simon et al. [4], the question of performing the lighting calculations

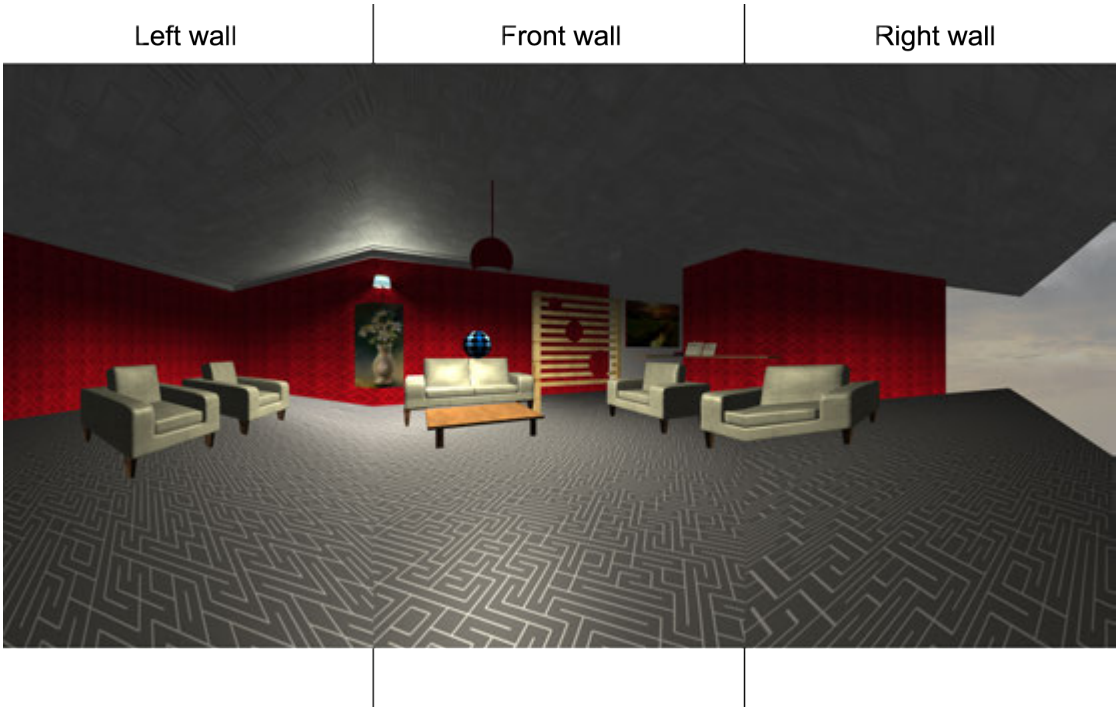

Figure 5.6: A living room scene rendered in omnistereo using the per-vertex object warping technique. The left, middle and right wall's images for the CAVE setup are shown. Due to the wide spacing of vertices on the floor, texture discontinuities are visible between the screen.

in omnistereo rendering was raised. The shading of a scene provides a strong cue for the depth and shape of objects. While the position of the viewer does not influence diffuse shading, it affects specular shading and thus the location and size of the specular highlights. Therefore, the specular shading differs for the left and the right eye. Experiments conducted by Adams and Elder [53] have shown that the presence of highlights affects shape perception and that misaligned highlights are no longer perceived as specularity. We decided to base the lighting calculations on the new eye position of the omnistereo projection, which is calculated on a per-vertex basis. For this purpose, the light and vertex positions used in the shading calculations can be translated into this eye position's view space before lighting calculations take place. The resulting specular highlights are, analogous to every aspect of omnistereo projection, only correct in the center of the view.

Figure 5.6 shows one of our test scenes, a living room, rendered using the vertex-shader based omnistereo projection technique. The images for the left, middle and right CAVE wall are combined into a single image in this figure. The living room's floor, walls and ceiling are composed of relatively large polygons leading to artefacts. The texture of the living room's floor features discontinuities across contiguous screens and

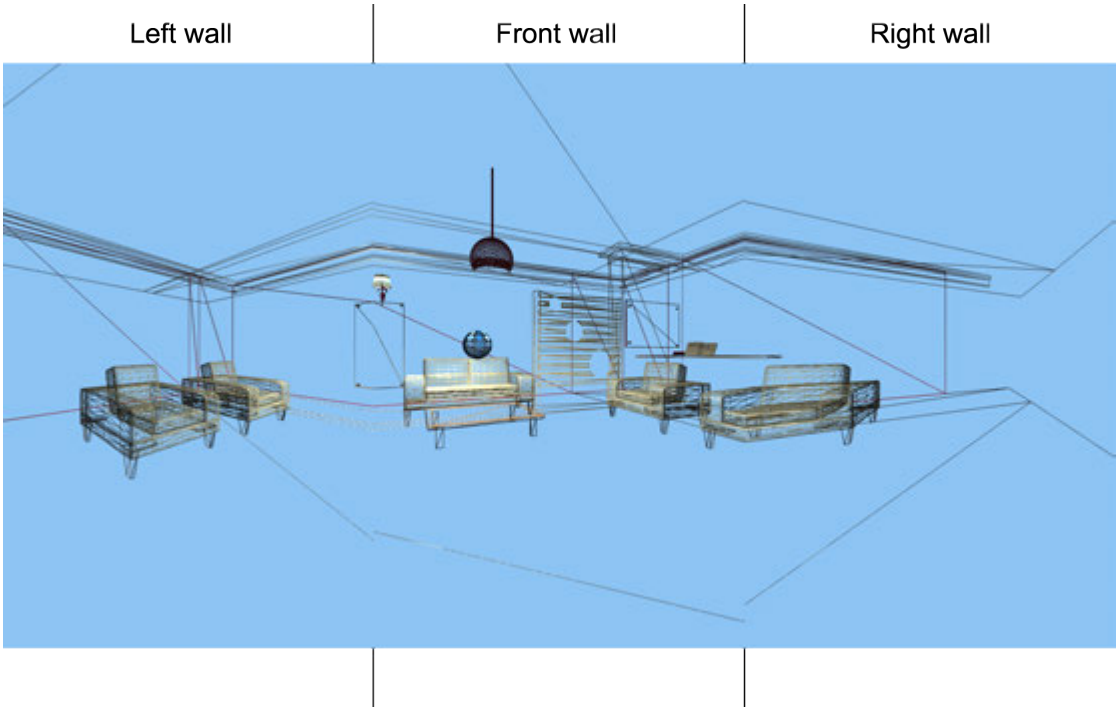

Figure 5.7: A wireframe rendering of the projection shown in Figure 5.6. The discontinuities of the floor polygon's edge across the different screen images is visible.

the floor's pattern appears displaced around the vertical line where the screens join. Edges of polygons can be affected analogously, featuring discontinuities at the screen edges. Figure 5.7 shows a wireframe rendering of the same projection, in which the discontinuity of the polygon edge is apparent. Polygons that feature these errors appear even worse in stereo, as they provide incorrect depth cues and may even introduce a vertical parallax. Furthermore, when small objects are in front of a larger polygon, an erroneous partial or complete occlusion, or lack thereof, may occur, which leads to a discrepancy in the stereo images that cause discomfort when viewed.

CHAPTER 6

Tessellation-Based Omnistereo Projection

In this chapter we suggest a solution for solving the issues arising in the object warping technique, mentioned in Chapter 5. Tessellation-based techniques are discussed as a solution to remedy the artefacts. Additionally, a novel omnistereo skybox rendering technique using a pre-tessellated mesh is described.

6.1 Remedying Non-Linear Projection Artefacts

The reason why artefacts appear in object-warp based omnistereo rendering is because the vertex-shader based implementation only provides correct results at the vertex position. The rendering pipeline then interpolates linearly between the non-linearly projected vertices. As described in Chapter 3, the omnistereo perspective is a non-linear perspective. Figure 6.1 shows a correct omnistereo projection of a cube. The straight edges of the cube become curved in the image due to the omnistereo projection. However, the rasterisation and the interpolation are performed linearly by the rendering pipeline and can therefore not produce curved lines.

During the rasterisation step, the primitives are broken down into fragments based on their sample coverage. For each fragment, the output data values from the previous stage are computed using interpolation. GLSL has three different modes of interpolation [54]:

- *flat* - The value will not be interpolated, instead the value from the *provoking vertex* of the primitive is taken.
- *noperspective* - The value will be linearly interpolated in screen space.
- *smooth* - The value will be interpolated in a perspective-correct way, which is the default mode of interpolation.

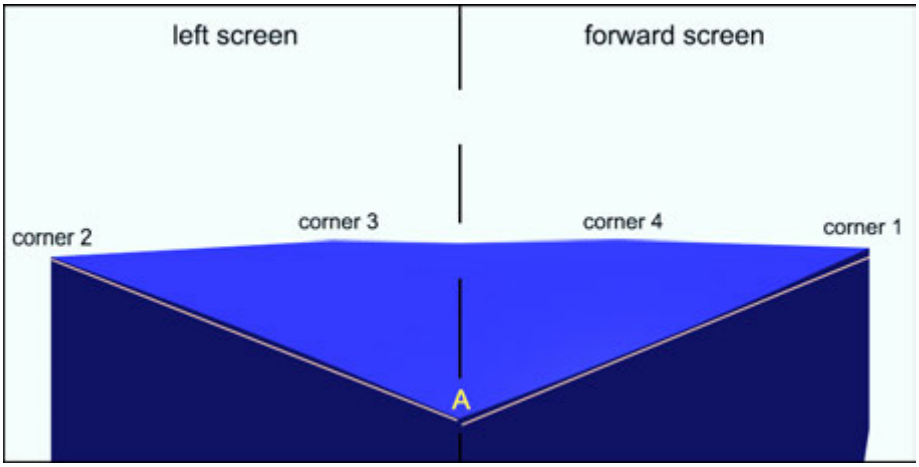

Figure 6.1: A blue cube rendered for the front and left wall of a CAVE using omnistereo projection. The corners of the top face are marked, so that A, located at the CAVE screen's border, is not mistaken as one of the corners. Two yellow lines were added to the image to improve visibility of the two edge curvature created by the non-linear projection.

None of these modes can provide correct interpolation based on non-linearly projected vertices. However, the interpolated output data values, such as texture coordinates and the position in view or world space, are needed for lighting calculations and texturing in the fragment shader and are typically interpolated using one of these modes.

Theoretically, a correct non-linear interpolation could be achieved for each fragment by performing the interpolation explicitly in the fragment shader. This approach is computationally expensive and leads to complications when using the OpenGL or Direct3D graphic APIs. In these two APIs, the only way to directly receive a non-interpolated value from the vertex shader is by using the *flat* output qualifier in OpenGL's GLSL shaders or the *nointerpolation* interpolation modifier in Direct3D [55]. However, when using this qualifier, all fragments inside a primitive receive their input value from a single vertex, called the *provoking vertex* (or *leading vertex*). This vertex is determined by the *provokeMode* and the *primitive type* [56]. The values of multiple or all vertices of the primitive associated with the fragment are therefore not accessible from within the fragment shader. It is possible to achieve non-linear interpolation in the fragment shader nevertheless, for example, by transforming from the warped space back into linear space, as done by Lloyd [57] in the context of logarithmic perspective shadow map rendering, followed by a composition of a new result based on the desired form of interpolation.

An additional problem that still needs to be solved is that the primitive rasterisation occurs in a linear fashion leading to incorrect sample coverage. Fragments supposed to be outside the fragment may be erroneously rasterised as part of the primitive and fragments supposed to be outside may be erroneously handled as covered ones. This occurs because non-linear projections cause lines to become curved in the projected

image, resulting in increased or reduced coverage of the primitive along an edge. The rendering pipeline only supports linear projections and therefore provides no solution for non-linear primitive rasterisation.

Vertex-shader based omnistereo rendering is greatly affected by this limitation whenever a primitive's edge spans across a large distance in screen space. When two vertices of an edge are projected on different adjacent CAVE walls, this can cause a discontinuity between the screens, making the issue become even more apparent. Objects that are small or have a high polygon density are less likely to cause any of the described problems but may nevertheless be at risk of doing so when viewed very close-up. However, it is not unusual for virtual environments to contain big objects with low polygon-density, such as planar, man-made structures like streets, floors or façades, especially in the context of architectural 3D rendering, simulations, VR applications and video games. It should be noted that for vertical screens, only the horizontal extent of a polygon affects its exposure to non-linearity issues, since the projection along the vertical axis is linear.

A general solution for non-linear rasterisation has not yet been found [58, 59]. Point-based rendering, adaptive subdivision, and ray-casting have been used as possible solutions.

The problem of non-linear rendering has also been discussed in the context of several shadow mapping techniques. Dual Paraboloid Shadow Maps feature non-linear vertex data, which are linearly interpolated in the rasterisation step performed by the hardware [5]. This leads to distortion and bending of shadows when polygons are large with respect to the light, which in turn can be solved with tessellation. Osman et al. [5] showed that moving some of the work from the vertex shader to the fragment shader eliminates the need for tessellation for shadow receivers. However, tessellation is still required for shadow casters in their technique. For this purpose, the necessary amount of tessellation is determined based on the distance of shadow casters to the light. During the shadow rendering, the tessellation parameters are then adjusted so that it is ensured that the rendered triangles do not exceed a given size. A similar solution was used in the context of Rectilinear Texture Warping (RTW) for efficient generation of adaptive shadow maps [60]. In this technique, accurate shadows can only be created with non-linear rasterisation, as the warping maps introduce non-linear distortions. However, due to the linear rasterisation, visual artefacts can occur in the output along the shadow edges. To overcome this issue, they used tessellation shaders to adaptively tessellate triangles based on their screen-space size after projecting them using RTW. The tessellation factors are based upon the edge length, such that no edge is above a pre-defined value in length.

It should be noted that a tessellation factor determined by the edge's length in screen-space will only lead to optimal results if the tessellation itself is performed perspectively correct. If the tessellation is based on barycentric interpolation of view, world or clip space coordinates, as it is typically the case, edges oriented at a skew angle to the camera will be tessellated uniformly in the respective coordinates' space but non-uniformly in screen space. However, for the purpose of alleviating the problems caused by the linear interpolation, the tessellation should be uniform in screen-space.

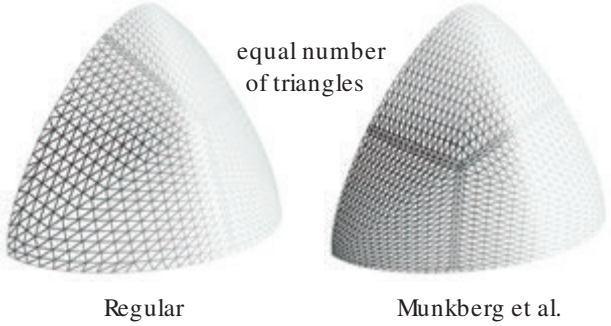

Figure 6.2: Comparison between tessellation techniques based on a PN-displaced triangle. The screen-space uniform tessellation places more vertices closer to the camera and provides a better distribution of vertices in screen space. Image taken from Munkberg et al. [6]

Munkberg et al. [6] described how to tessellate triangles considering the perspective distortion applied during the projection. This form of tessellation is based on perspective-correct interpolation. A disadvantage of the technique is that clipping is required, since triangles that are partially behind the camera can cause artefacts. The rendering pipeline clips triangles to the view frustum before the perspective division, avoiding such problems. Another issue is that triangles outside the view frustum with one or two vertices in front of the near plane will be over-tessellated. Since the tessellation step occurs before the clipping performed by the rendering pipeline, additional clipping needs to be performed in the shaders before the tessellation step. The clipping only needs to be done for the original input triangles but is nevertheless computationally expensive. A comparison of regular tessellation and the perspective-correct tessellation by Munkberg et al. [6] can be seen in Figure 6.2.

Lloyd [57] proposed logarithmic perspective shadow maps, which require logarithmic rasterisation causing planar primitives to be curved. In order to avoid errors caused by the linear rasterisation, a brute-force rasterisation is performed in the fragment shader. Bounding quads are created for each triangle to achieve a correct coverage of the fragments. When it is determined that a fragment falls outside the triangle, it is discarded in the fragment shader. Creating the bounding quads was determined to be a bottleneck in their implementation. This could be improved using the tessellation or geometry shaders, which are now available on GPU hardware for this purpose, but the logarithmic rasterisation would nevertheless be considerably slower than linear rasterisation.

6.2 Adaptive Tessellation for Omnistereo Rendering

The aforementioned techniques for solving non-linear rasterisation problems can also be used to solve the non-linear rasterisation and interpolation problems in the omnistereo projection. Screen-space adaptive tessellation techniques offer an efficient solution,

whereas screen-space uniform tessellation and the brute-force rasterisation require a computational demand that can be considered too high for real-time rendering. In order to efficiently remedy the artefacts occurring in object-warp based omnistereo rendering, we propose to combine existing screen-space adaptive tessellation approaches with the object-warping technique.

In Section 2.7 we described the tessellation shaders present in modern rendering pipelines. We also mentioned how interpolation and subdivision are handled in regard to adaptive tessellation, which is a suitable tessellation technique for the purpose of solving non-linear rasterisation problems and is also the technique we use to improve omnistereo rendering techniques. When using adaptive tessellation to solve the artefacts produced by object warping, the vertex shader only takes care of performing transformations. In this setup, the object warping itself takes place in the TES, to where it can be moved without mentionable changes.

A method for determining the tessellation levels in the TCS has to be chosen. A naïve and fast approach is to use the view-space distance of the vertices as basis. However, primitives with adjacent edges might be tessellated differently if only the vertex positions are taken into account, leading to visible seams in the result as mentioned earlier. Thus, a better approach is to calculate the tessellation level for each edge separately. However, this does not take the viewport size and the projection parameters into account. The size of an edge in screen space is decisive for the quality of the omnistereo projection. Thus, a screen-space adaptive tessellation approach is better suited. The exact position of the vertices after the object warping cannot be known during the determination of the tessellation levels. An off-axis projection can, however, provide a rough approximation for the post-object-warp screen space positions. A COP and view direction for the off-axis projection has to be chosen for each eye. We suggest using either the assumed head position as COP for both of the eyes or off-axis projections with a view direction perpendicular screen, resulting in a separate projection for each eye.

We combined the object-warping techniques with two existing screen-space adaptive methods [27, 7] for determining the tessellation levels. Both of the methods first transform the vertices into screen-space coordinates. This is done inside the shader, by transforming the world-space vertex into clip space, performing a perspective division and multiplying the x and y coordinates of the resulting NDC space position with the viewport width and height respectively. We used two parameters for the tessellation: A target edge length, which defines the desired screen-space edge length we want to achieve after tessellation, and a maximum tessellation level, which defines an upper limit for the tessellation of each edge. When determining the tessellation levels for the vertical CAVE wall screens, thus excluding the ceiling and floor screens, only the edge lengths in the x-direction should influence the tessellation levels, since the vertical edge length does not affect the quality of the omnistereo images. For the ceiling and floor screens, the length in both the x- and y-direction is relevant, since the projection is non-linear in both directions.

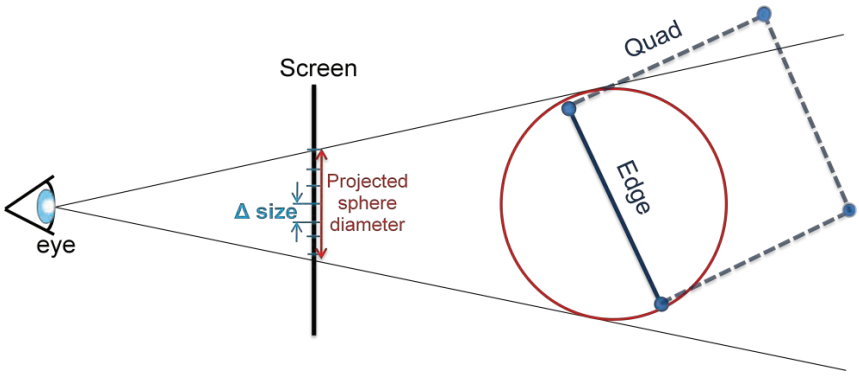

Figure 6.3: Adaptive screen-based tessellation based on bounding spheres. Image taken from Cantlay and NVIDIA Corporation [7]

Edge-based screen-space adaptive tessellation

The first method evaluates the tessellation levels in a straight-forward way based on the edge information, as originally suggested by Doggett and Hirche [27]: The distance between the screen-space vertex positions of each edge is calculated and then divided by the targeted edge length. The result is clamped between 1 and the maximum tessellation level. This is done for each edge to determine all of the triangle's tessellation levels. The TES then determines the position and attributes of the new vertices using barycentric interpolation without considering perspective correctness. A downside of this approach is that it leads to over-tessellation of triangles that are partially or fully outside the view frustum [6]. Additionally, it does not take the perspective projection distortion into account and will under-tessellate edges at a skew-angle to the view direction.

Sphere-based adaptive tessellation

As second method we used sphere-based adaptive tessellation, suggested by Cantlay and NVIDIA Corporation [7]. Originally intended for terrain rendering, the sphere-based screen-space adaptive tessellation method by Cantlay and NVIDIA Corporation [7] computes the tessellation level by conceptually fitting a sphere around the edge in screen space, as shown in Figure 6.3. The sizes of the triangles resulting from this technique are generally uniform in screen space. First, the edge length in world or view space is calculated. The result is used as the diameter of the sphere. The midpoint of the edge transformed into view space constitutes the center of the sphere. A second point is determined by adding the radius of the sphere to the x-coordinate of the sphere center. Both points are then transformed into screen space and the edge's tessellation level is determined analogously to the previously mentioned method. In both methods, the additional rendering complexity is dependent on the complexity of the scene, the current viewpoint and the hardware used.

During our informal evaluation we noticed that edges, whenever edges are not tessellated sufficiently, the error associated with them becomes more apparent when they are visible on multiple adjacent screens, since the tessellation levels may differ between the screens and discontinuities can therefore be induced. In some situations, it can therefore be desirable to tessellate primitives that are potentially visible on multiple screens by the highest calculated tessellation level for any potential screen. This may be needed if the scene may not be tessellated enough due to hardware constraints or if this type of artefact frequently occurs in a specific scenes in the used method. We implemented the approach in the TCS and tested it with our tessellation methods, but although it fixes the problem sufficiently for static viewpoints, it leads to popping during camera movement and overall has a high impact on the rendering performance. A better solution is to ensure that a method is used that provides sufficient tessellation for all objects and that the hardware allows high enough tessellation levels. In this case the achieved quality of the omnistereo projection can be good enough to make the tessellation differences across the screens imperceptible, even if the tessellation levels for polygons may vary between the screens. If the maximum tessellation level and target edge length for the sphere-based tessellation method was chosen accordingly for our test scene, no such issues appeared and the performance was significantly better than if using lower settings and determining the highest tessellation across the screens.

We found that the sphere-based tessellation method generally produced better tessellation quality than the naïve method in conjunction with the omnistereo projection technique. The determined tessellation levels for perspectively distorted edges lead to better screen-space tessellation and it does not suffer from over-tessellation issues regarding vertices lying outside the frustum. As a result, a better approximation of the non-linear omnistereo projection could be achieved at equal frame rates. A comparison between the original method and the method combined with adaptive tessellation is shown in Section 7.3.

6.3 Omnistereo Skybox for the CAVE

Skyboxes and skydomes are used to efficiently render skies and distant backgrounds, such as buildings or mountains, in real time. A skybox is a cube with the environment mapped onto each side of the cube, stored as six square textures or six regions of a single texture. A skydome is based on the same principle but uses a sphere or hemisphere instead of the cube. Skyboxes and skydomes therefore offer information for every view direction.

Skyboxes can be rendered either based on a textured cube, which has each corner position located at infinite or near-infinite distance, or by sampling a cubemap, such as provided by OpenGL [54]. Rendering the skybox using a distant cube is possible with the tessellation-based omnistereo projection technique that we proposed. In doing so, the skybox can be rendered like any other scene geometry. However, since the maximum tessellation level is limited by the hardware and may be additionally limited by the parameters of the technique to prevent over-tessellation, it may be necessary to pre-tessellate the cube geometry of the skybox in order to attain the desired quality of the

projection.

The second way of rendering a skybox is to render it using a full-screen quad and a cubemap. A cubemap is a texture consisting of six 2D images and offers the same mipmapping and filtering capabilities as other OpenGL textures. If the feature is supported, OpenGL allows enabling the *GL_TEXTURE_CUBE_MAP_SEAMLESS* capability to seamlessly filter cubemaps. In order to display the information of the skybox stored in the cubemap, a full-screen quad can be rendered in conjunction with a fragment shader that samples the cubemap. In OpenGL, the full-screen quad should cover the range $[-1, 1]$ in x- and y-direction. The z-coordinate of the input vertices should be set to -1 and the w-coordinate 1, so that the coordinates can be used directly for the calculation of the lookup vectors. The full-screen quad's vertex positions represent Normalized Device Coordinates (NDCs) and can therefore be used as the clip-space output position (*gl_Position*) in the vertex shader without requiring any transformations or projections. However, for the purpose of serving as vertex position, the z-coordinate of the vertex has to be set to 1, so that the skybox appears in the background as desired. To determine the lookup vector, the unaltered vertex coordinates are first transformed into view space using the inverse projection matrix. The resulting position can be interpreted as a view direction in view space. This direction vector is then transformed into world space to serve as a lookup vector for the cubemap texture, for example by transforming it using the transpose of the upper-left 3×3 submatrix extracted from the view matrix. Finally, the lookup vector is interpolated and used in the fragment shader to sample the cubemap texture. If the skybox is rendered as the last part of the scene, the values in the depth buffer can be used for z-culling, which rejects pixels early on to improve the performance. While this technique works for off-axis and on-axis perspective projections, it can not be used for omnistereo rendering.

Thus, we propose a novel technique to efficiently render omnistereo skyboxes using a pre-tessellated full-screen quad. The tessellated mesh is created on the CPU with the vertices of its triangles laid out in the range $[-1, 1]$ in the x- and y-direction, while the z-coordinate stays fixed at -1 and the w-coordinate at 1. The clip-space output position is set in the same fashion as in the aforementioned method. The vertex positions are then transformed so that they represent coordinates of this screen in the CAVE. For this purpose, the coordinates are transformed into a view space coordinate system that has its position at the center of the CAVE and the view direction perpendicular to the CAVE wall. This can be done simply by multiplying the x-, y- and z-coordinates of the input vertex position with the distance from the center to the respective CAVE wall. If the screen is not of square dimensions, the respective coordinate has to be modified by the aspect ratio. In our implementation, the y-coordinate therefore had to by divided the aspect ratio. By transforming the assumed head position into this space, the eye positions for the omnistereo projection can be determined, as described in Section 5.1. The view-space lookup vector is defined as the vector from the eye position to the screen's view-space vertex. The view-space lookup direction can subsequently be transformed into world space, for which the transpose of the upper-left 3×3 submatrix extracted from the view matrix was used in the previous method. Since only the orientation

of the view matrix affects this transformation, any view matrix with a view-direction perpendicular to the screen can be used for this purpose. Finally, the unnormalised lookup vector can be retrieved as interpolated vector in the fragment shader to sample the cubemap.

The tessellated screen-space quad's mesh used in the omnistereo skybox technique needs to be tessellated horizontally depending on the screen's resolution, in order to provide a visually acceptable approximation. Analogous to regular omnistereo projection, vertical tessellation is not necessary for vertical CAVE walls and has no effect on the result. The closer the assumed head position is located to the CAVE wall, the more vertical slices are required due to the larger degree of distortion. From our observations, approximately 1 vertical slice per 30 horizontal pixels is sufficient to provide accurate results, even if the assumed head position is located very closely to one or multiple CAVE's walls. For the floor and ceiling screens of a CAVE display system, both horizontal and vertical tessellation is necessary.

CHAPTER 7

Implementation and Results

7.1 Setup

An application was created to test and display our techniques and to be able to quickly compare them at run-time. It was written in C++ and initially relied on Ogre [61] version 1.9, as we did not require tessellation shaders in the beginning. The version of the Ogre library in use was modified by us to provide stereo rendering capabilities based on OpenGL quad buffers. Our latest version of the application uses a version of the *Ogre3D* engine from the development branch, as it features tessellation capabilities in its OpenGL3 renderer. The tessellation capabilities are a requirement in order to to remove artefacts in the omnistereo projection. We used the graphical user interface (GUI) library *CEGUI* [62] to display and handle the GUI in the application. The editor *CEED*, which is a part of the CEGUI project, was used to create the XML-based layouts that our GUI is based on. The shaders were written in *GLSL* [54]. The application loads scenes based on the *DotScene* file format of the *Ogre3D* engine, which allows importing scenes from Blender and Autodesk Maya. The viewpoint and direction in the scene, as well as the rendering configuration for the different modes, can be controlled in the application using a mouse, keyboard and the GUI or alternatively by using an a gamepad featuring an XBox360 controller layout.

Our project was created for the CAVE of the VRVis research center. This CAVE consists of three walls (front, left and right) providing a projection with a horizontal FOV of 270°. The room for the CAVE has a quadratic floor area with 3,3 m side length; the projection screen height is 2,9 m. Unlike the original CAVE proposed by Cruz-Neira et al. [2], this CAVE uses forward projections. Figure 7.1 shows the projection and positioning of the LCD projectors in the installation. Dell S500 projectors are used, providing a resolution of 1280 × 720 pixels with a 16:9 ratio and a refresh rate of 120Hz each. The combined resolution across all screens is 3840 × 720, and active shutter glasses are used for stereoscopic vision. Typical CAVEs, as well as the original CAVE proposed by Cruz-Neira et al. [2] and the low-cost three-wall CAVE suggested

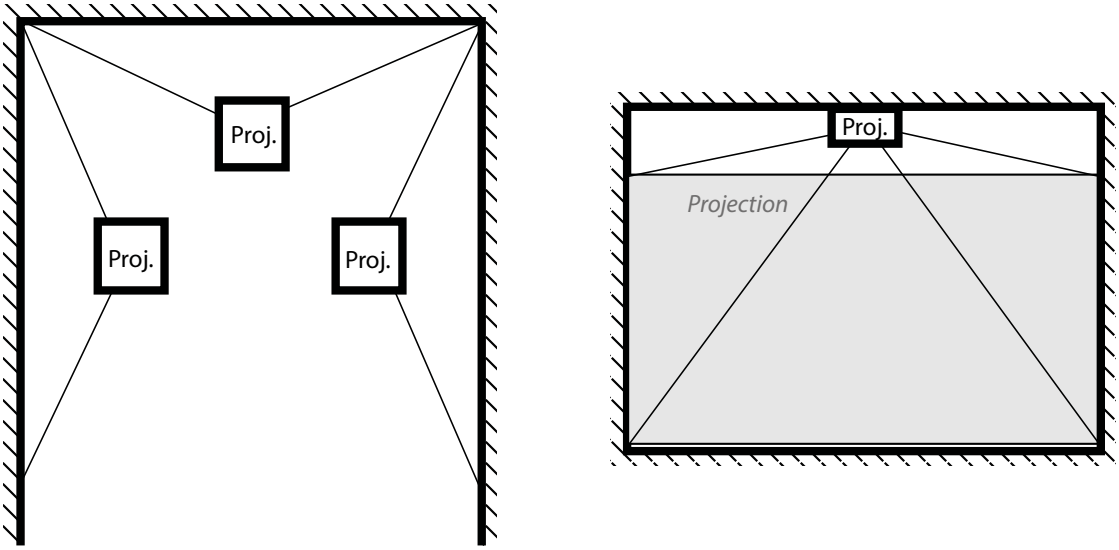

Figure 7.1: Floor plan (left) and elevation (right) of the CAVE setup at the VRVis. Copyright ©2012-2015 VRVis

by Cruz-Neira et al. [32] that were described in Section 4.1, use synchronized PC clusters for rendering. However, this CAVE was designed so that a single PC can render for the entire CAVE at real-time frame rates. An Nvidia Quadro K5000 GPU with multiple outputs is used for this purpose and since no PC clusters are required and commercial off-the-shelf projectors are used, this setup is relatively low-cost compared to typical CAVEs. Thus, it could, for example, be used to make the low-cost CAVE presented by Cruz-Neira et al. [32] even less expensive.

The Nvidia Mosaic™ multi-display technology is used to span the application across all projectors, which allows rendering to the display system synchronously and without tearing artefacts. This mode is used in conjunction with active stereo rendering.

In order to render non-anaglyph stereoscopic images, quad-buffer rendering is typically used. It enables double-buffering for rendering from each eye's viewpoint, using a separate front and back buffer per eye. Whereas OpenGL supports quad buffering natively since version 2.0 [63] (2004), Direct3D had not supported stereoscopic rendering natively until version 11.1 (2012). This version of Direct3D is only available as a complete version on Windows 8 or higher. A back-ported version of Direct3D 11.1 is available for Windows 7, however it lacks the quad buffer capabilities entirely [64]. For this reason, OpenGL has been chosen as graphics API for projects developed for the VRVis CAVE, including this application.

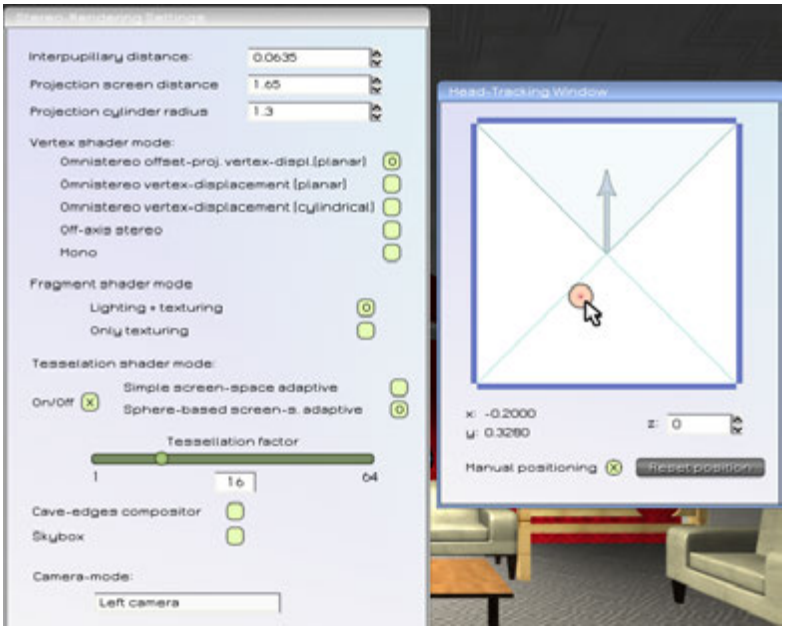

Figure 7.2: The GUI overlay of the application. The left window allows changing between shaders, adjust eye separation distance, CAVE dimensions and other settings. The right window allows configuring the head position in the CAVE in all three dimensions and to switch on/off the automatic network-acquired positioning.

7.2 Application

A fixed head position is used in several omnistereo techniques described in Chapter 4. In those cases, the projection is created so that multiple users can share the same display and view. The view position in such setups is typically assumed to be at the center of the display system at all times to minimise the distortion error, created due to the deviation from the actual view position, across all viewers. Since our application is created primarily for single-viewer usage, we wanted to be able to change the viewpoint freely. With this in mind we created our application to allow either manually setting the head position in our application's GUI or receiving the head position via the Open Sound Control (OSC) protocol. We used the *oscpack* [65] library for receiving the packages from the local network and unpacking them. We have successfully tested this setup using head-tracking provided by a second computer in connection with a Microsoft Kinect [66]. Figure 7.2 shows the GUI of the application, consisting of two GUI windows allowing to adjust the mode and parameters of the omnistereo rendering techniques.

It is possible to implement the object-warping omnistereo techniques in view space or world space. Some of our original implementations were created based on the world space coordinate system. However, since omnistereo techniques generally assume that the viewer's head is always in the upright position, the majority of calculations, such as the calculation of the eye position and intersections of lines, can be reduced to a two-

dimensional problem in view space. If the calculations are performed in world space, they involve a higher level of complexity, which leads to a higher likelihood for numeric errors to occur and also negatively affects the rendering performance. Since using world space coordinates also offers no noteworthy advantages, we decided to implement all of our techniques based on the view space coordinate system.

We did not implement an omnistereo technique using multi-view rendering featuring multiple COPs per screen with one per view using vertical slices, because of the lower performance and, if an insufficient amount of views is used, the lower quality provided. These drawbacks are discussed in Chapter 4 in the context of the mentioned omnistereo rendering techniques.

7.3 Results

Based on our application we tested the rendering techniques in multiple test scenes, using various tessellation configurations and different head positions in the CAVE. We compared the images rendered using regular off-axis stereo projection with those of the object-warping omnistereo technique and the tessellation-based technique. The comparison shows how the tessellation-based technique manages to remove the artefacts in the object warp based images. Our application was also used to compare the frame rates between the mentioned rendering methods.

Tessellation-based omnistereo projection

As discussed in Section 5.3, the vertex-shader based technique using object-warping for omnistereo rendering can cause noticeable artefacts. Figure 7.3 shows omnistereo images rendered with and without using tessellation being performed before the per-vertex operations. The omnistereo images rendered with the off-axis and on-axis based object-warping techniques show no differences and were rendered without significant difference in speed. Figure A and Figure B show the omnistereo images for the left, front and right CAVE walls as anaglyph images. Figure A, which is generated with the technique without tessellation, features the discussed artefacts stemming from the linear interpolation. Along the edge between the images for left and the front wall, inter-screen discontinuities are visible at the floor and at the connection between the wall and the ceiling. On the left wall's image, the painting is fully concealed by the wall in one eye's image, although it is supposed to be in front of the wall for both images, and the lamp shade is partially concealed. The floor texture and the ceiling section are rendered with wrong depth cues. None of the artefacts appear in Figure B, which features images generated with the object warp technique preceded by a tessellation step.

For comparison with regular stereoscopic off-axis projections, we also rendered the scene from the same head position with the view direction being set perpendicular to the front wall's screen, which is equal to the naïve approach mentioned in Chapter 3. As can be seen in the anaglyph images in Figure 7.4, the stereo disparity is only present in the front wall's image and is barely present in the left and right wall's images. Furthermore,

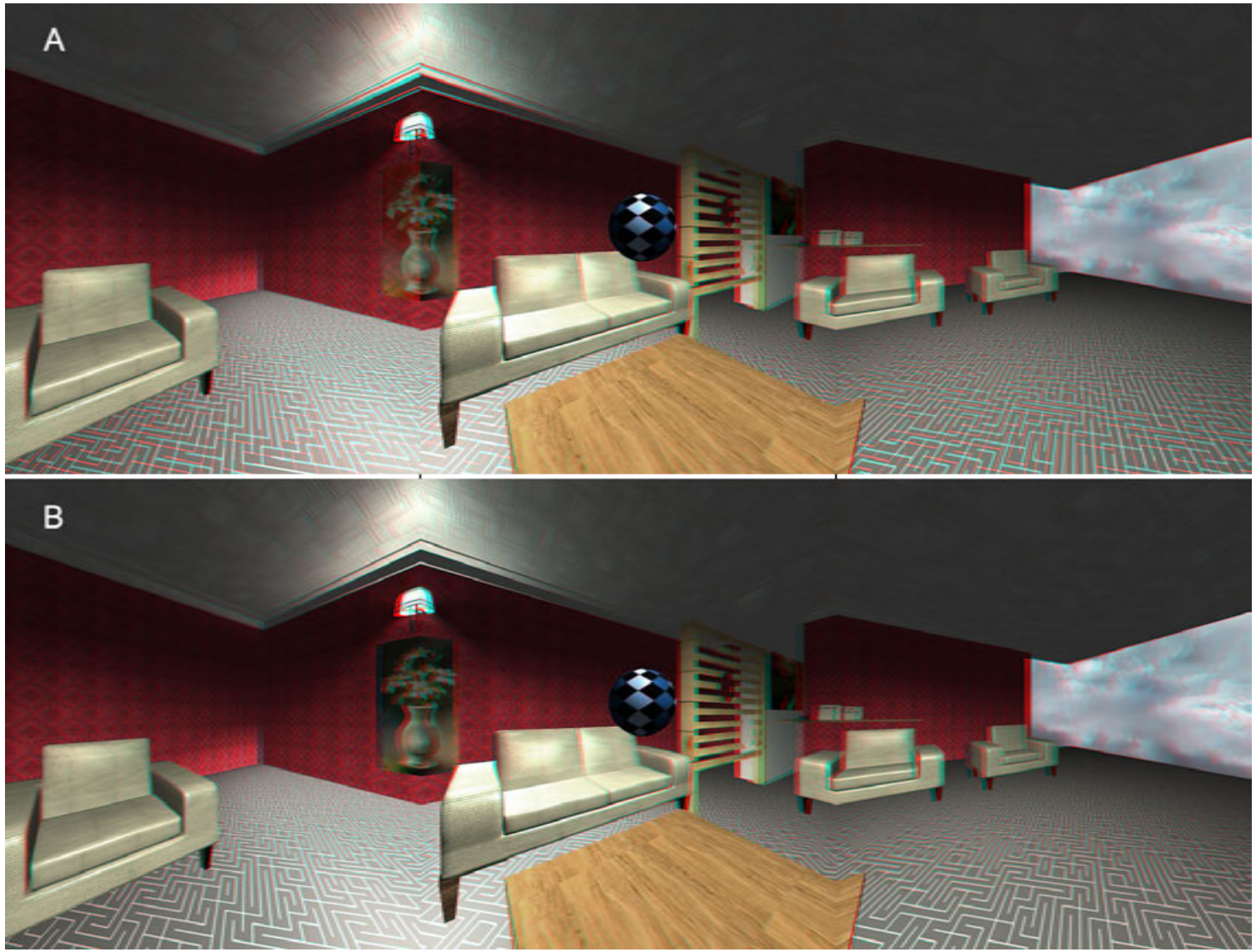

Figure 7.3: Omnistereo images rendered without tessellation (Figure A) and with tessellation (Figure B).

the chair legs and the floor on the left image were rendered featuring a vertical parallax in stereo, since one eye is closer to the CAVE's wall than the other. While this image is correct for a viewer facing the front wall's screen, viewers facing the left or right wall or facing any other direction that is not perpendicular to the front wall are not able to perceive correct stereo depth cues in the images and are exposed to an incorrect vertical parallax.

Informal testing showed that adding the tessellation step to the rendering process of the living-room test scene resulted in a frame-rate decrease of approximately 10-40% depending on the view point and scene complexity. One way to increase rendering performance when using tessellation is by performing back-face culling [67] and view-frustum [68] culling inside the tessellation shader. Regular back-face culling, if enabled, occurs during the *primitive assembly* stage, as shown in Figure 2.14. Implementing culling in the TCS allows discarding primitives before performing any tessellation on them, as well as before determining the tessellation levels for the primitive. Adding back-face culling to the shader reduced the impact of the tessellation on the rendering performance, resulting in a frame-rate decrease of 15% compared to rendering without tessellation. The regular OpenGL back-face culling was enabled at all times during these comparisons. In the shader-based back-face culling, a small epsilon was used to avoid erroneous culling, considering that the vertex displacement happening during the object-warping in later steps may change the orientation of a face. When using this method, we could not find occurrences of erroneously culled primitives in the scene. When using the sphere-based tessellation method in conjunction with back-face culling, the rendering frame rate was therefore decreased by approximately 15% while removing all visible issues from the interpolation and rasterisation.

Figure 7.5 shows a frame-rate comparison of different rendering configurations measured in two test scenes with two different camera viewpoint locations each. The first test scene features a small living room scene with several objects of varying polygon density. The second test scene is larger and features an urban area consisting of objects with low polygon-density. The measurements were taken while rendering the left eye's images for three CAVE wall screens at 500×500 pixel resolution each and without rendering a skybox. An AMD Radeon 7970 GPU was used, and the hardware anti-aliasing setting of Ogre was set to *FSAA 8x*. The viewpoint 1 of the scenes was positioned in a way that the images capture a large portion of the scene. Viewpoint 2 was positioned in centrally in the scenes and in close proximity to several objects. The performance of the vertex-shader based omnistereo technique was insignificantly different from the regular off-axis projection rendering, according to the measurements. The omnistereo projection with tessellation was rendered in three different configurations, each with a different maximum tessellation factor set for the sphere-based tessellation technique. Based on informal evaluation of the results using the test scenes and varying view points and head positions, we determined that in our setup a maximum tessellation factor of 16 was sufficient to remove the object-warping artefacts in the tessellation-based rendering technique, as higher values did not further improve the quality of the images. Additionally, the frame rate comparison showed that higher maximum tessellation values did not

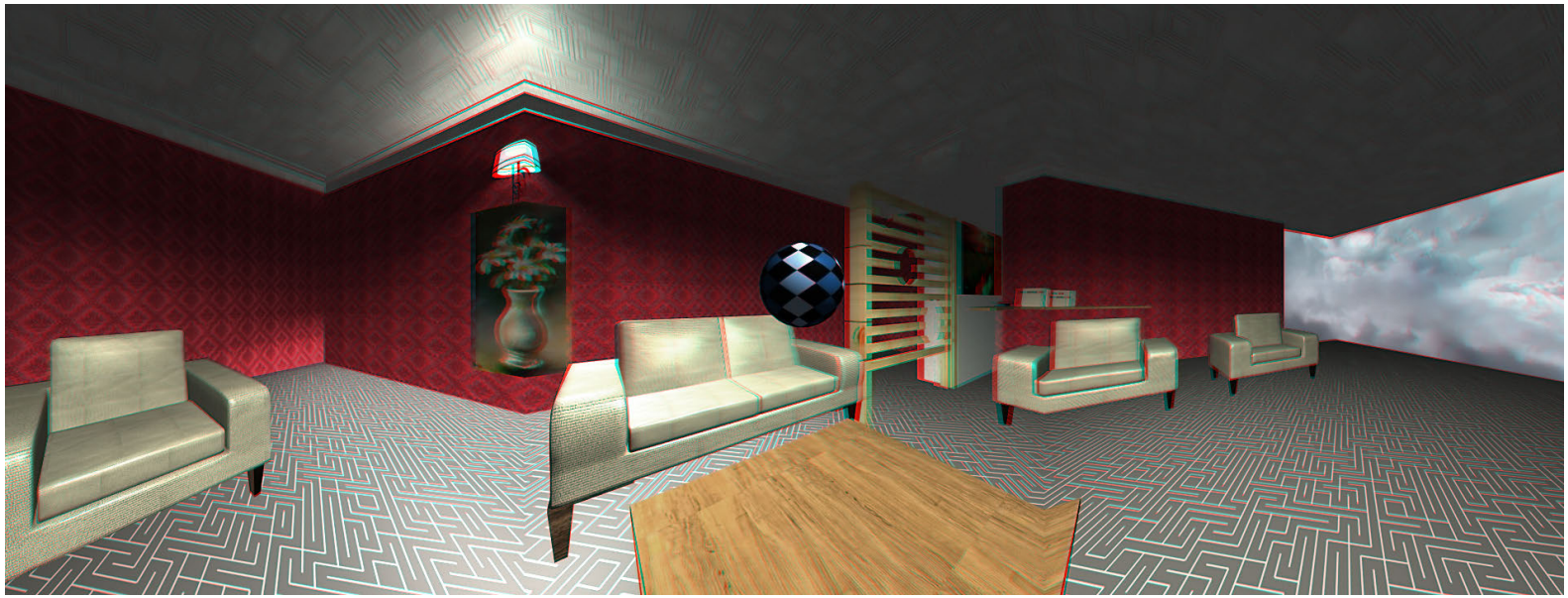

Figure 7.4: Off-axis perspective projections with the view-direction perpendicular to the front wall's screen.

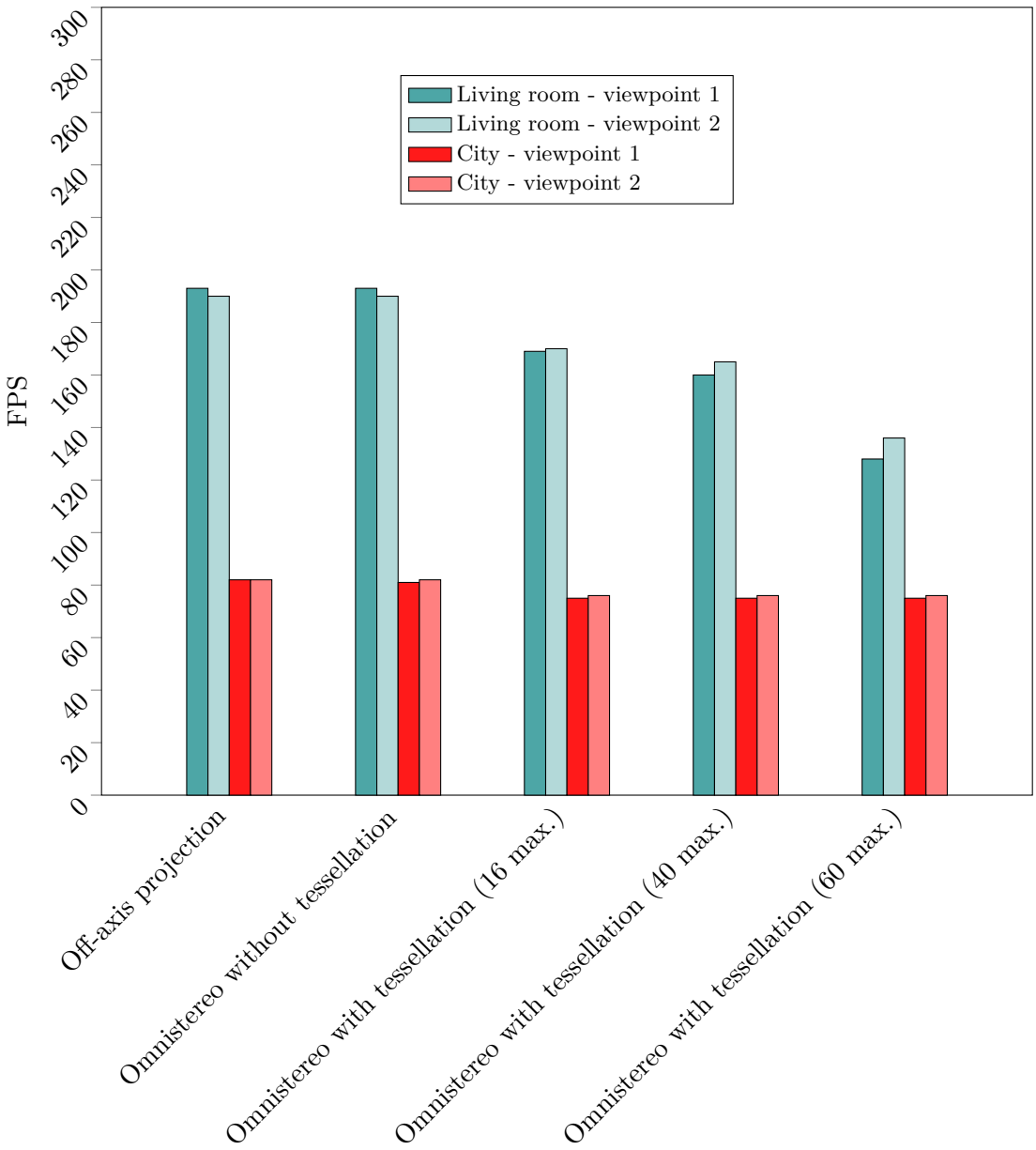

Figure 7.5: FPS Benchmark of different rendering configurations

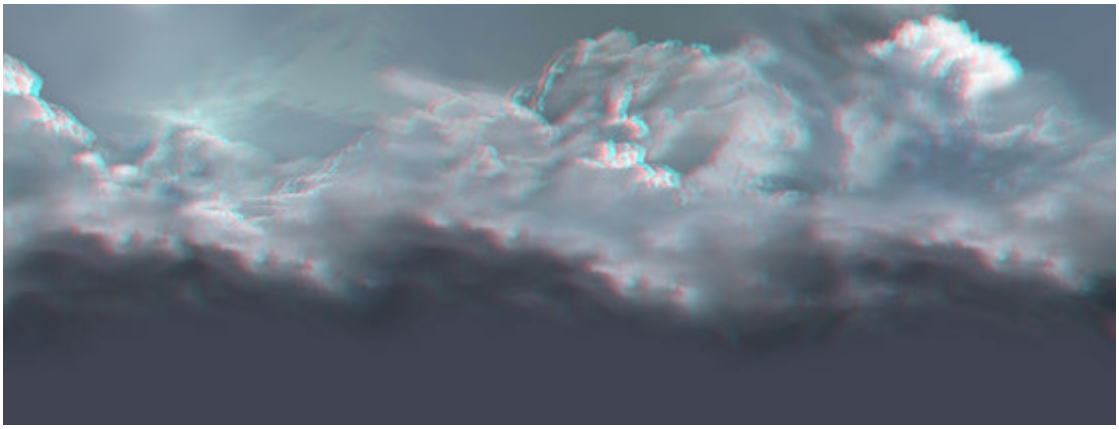

Figure 7.6: An omnistereo skybox rendered for the front, left and right CAVE walls. Stereo disparity is present in all view-directions and the images are seamless across the screens.

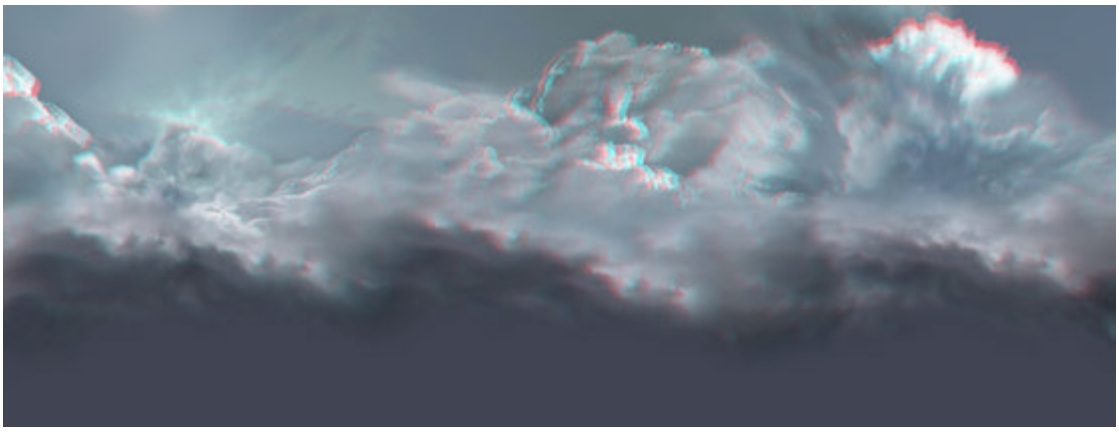

Figure 7.7: An off-axis projection skybox rendering with the view direction perpendicular to the front wall's screen. The stereo disparity becomes increasingly incorrect, as the view-direction changes horizontally away from a horizontal center of the image.

impact the frame rate in the city scene, but significantly reduced the frame rate in the living room scene, since it is more densely populated with objects and the objects are closer to the viewer.

Full-screen-quad based omnistereo-skybox

Figure 7.6 shows an omnistereo skybox rendering for the left, front and right CAVE wall as anaglyph image. The omnistereo skybox images feature stereo depth cues in all view directions without containing seams between the screens. Arbitrary head positions inside the CAVE are supported by the technique. Figure 7.7 shows a skybox rendering

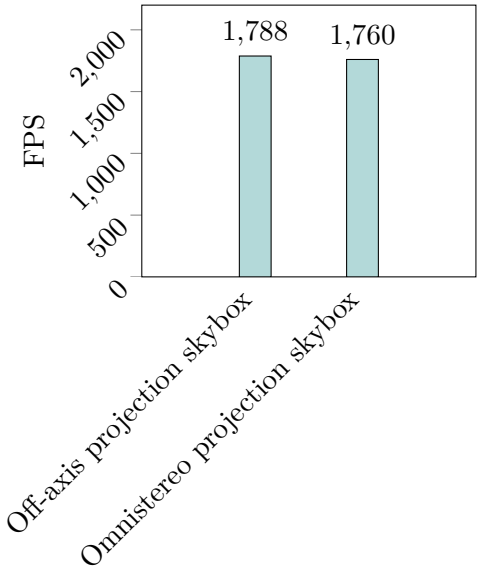

Figure 7.8: Frame rate comparison of the omnistereo skybox technique with the regular off-axis projection based technique

from the same viewpoint using an off-axis perspective projection, with the view direction perpendicular to the front screen. The advantages of rendering skyboxes in omnistereo over rendering them in regular stereo are analogous to those of rendering meshes using omnistereo projections instead of regular stereo projections, which were discussed in Section 7.3.

Figure 7.8 shows a comparison of the frame-rates between the two methods. The omnistereo and off-axis projection skybox was rendered for one eye's view for three CAVE wall screens with a resolution of 500×500 pixels each, using an AMD Radeon 7970 GPU.

CHAPTER 8

Conclusion

The research goal of this thesis was to create an efficient technique for rendering virtual environments in a CAVE providing stereo depth cues in all view directions. Important aspects of VR and the reason why omnistereo images can be perceived without noticing the distortion in the periphery were discussed, forming the motivation for using such omnistereo rendering techniques. A state of the art of CAVEs, omnistereo imaging and omnistereo rendering was presented. The mathematical background for the off-axis based object-warping technique, as originally proposed by Simon et al. [4], and for an on-axis projection based object-warping technique were provided, as well as illustrations showing the geometry behind it and potentially problematic corner-cases. We implemented an application that renders omnistereo images for a CAVE display system using the object-warping techniques. Artefacts that occur due to the non-linearity of the omnistereo projection were described and illustrated in the thesis. Our proposed solution to these problems is to precede the object-warp based omnistereo rendering technique with a screen-space adaptive tessellation step, while keeping the performance impact on the frame rate minimal. Using this approach, artefact-free rendering of omnistereo images can be achieved, which can be used in combination with regular shading techniques and renders at real-time frame rates.

The tessellation-based omnistereo rendering technique, which combines adaptive tessellation with per-vertex object-warping, provides a great improvement to regular stereoscopic rendering with only a few drawbacks. We have shown that by implementing back-face culling in the tessellation shader, the impact of the tessellation step can be greatly reduced, which can be further improved upon by adding a frustum-culling technique tailored to the specific requirements of the non-linear projection. Additionally, we created an efficient skybox rendering technique that can be used in conjunction with omnistereo rendering. We also presented the CAVE display system created by the VRVis, which was used for our project and can render to all of its three CAVE walls with a single PC and GPU, making it a system that is more affordable and simpler to maintain than typical CAVEs.

8.1 Future Work

Future work regarding shader-based frustum culling for our omnistereo technique is necessary. Efficient frustum-culling implementations are already known, but a method determining the smallest frustum dimension that includes all visible polygons, while considering the vertices moved into visibility due to object-warping, has to be a found. As discussed in Chapter 2, Couture et al. [24] analysed the disparity distortion of omnistereo in cylindrical displays. Analogous analysis is required for omnistereo in CAVE display systems. Additionally, a formal evaluation needs to be conducted to determine to which extent disparity errors in the periphery are perceived by users in CAVE displays when observing omnistereo projections. A question that remains open in this context, is how far a viewer can move their head position from the assumed view position without distortion errors becoming apparent and how the dimensions of the CAVE, proximity to the screen and resolution influence this. Although it can be crucial to provide free movement to the user in a VR installations, some installations constrain the user's movement to a narrow area. That can be the case, for example, when using treadmills for locomotion, which may limit the user to the center even if moving. Furthermore, we discussed omnistereo rendering for surround-screen displays such as hemispherical, cylindrical and conical display systems as well as for CAVEs. Less established, experimental display systems, such as *image projection with convex mirrors* [69] and *multi-projection system with hybrid screen* [70], could profit from omnistereo projections as well.

Research is required to determine if a large-sized CAVE display allows for multi-viewer usage once omnistereo projections are rendered. This may be possible if the viewers are not spread out too far from the assumed viewpoint of the omnistereo projection. If this is feasible, it might allow for collaborative work using the shared display or for viewing the virtual environment in groups with a guide, teacher or trainer who controls the view. Also, in Section 2.3 we discussed how some rendering effects do not work well or need to be changed in stereo rendering. Another question that remains open, is how omnistereo rendering affects these, especially compared to regular stereo rendering, and if the same solutions as for regular stereo rendering can be applied.

Bibliography

[1] L. Avila and M. Bailey, "Virtual Reality for the Masses," *IEEE computer graphics and applications*, no. 5, pp. 103–104, 2014.

[2] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart, "The cave: Audio visual experience automatic virtual environment," *Commun. ACM*, vol. 35, no. 6, pp. 64–72, 1992.

[3] N. A. Dodgson, "Autostereoscopic 3d displays," *Computer*, no. 8, pp. 31–36, 2005.

[4] A. Simon, R. C. Smith, and R. R. Pawlicki, "Omnistereo for panoramic virtual environment display systems," in *Virtual Reality, 2004. Proceedings. IEEE*, 2004, pp. 67–279.

[5] B. Osman, M. Bukowski, and C. McEvoy, "Practical implementation of dual paraboloid shadow maps," in *Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames*. ACM, 2006, pp. 103–106.

[6] J. Munkberg, J. Hasselgren, and T. Akenine-Möller, "Non-uniform fractional tessellation," in *Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, ser. GH '08. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2008, pp. 41–45.

[7] I. Cantlay and NVIDIA Corporation, "DirectX 11 terrain tessellation," http://developer.download.nvidia.com/assets/gamedev/files/sdk/11/TerrainTessellation_WhitePaper.pdf, 2011, (last accessed on 29th April 2015).

[8] P. Bourke, "Calculating Stereo Pairs," http://paulbourke.net/stereographics/stereorender, 1999, (last accessed on 12th December 2014).

[9] L. Harrison, D. McAllister, and M. Dulberg, "Stereo computer graphics for virtual reality," *SIGGRAPH'97, Course Notes*, vol. 6, 1997.

[10] W. Sadowski and K. M. Stanney, "Presence in virtual environments," in *Handbook of virtual environments : Design, implementation and applications*. L. Erlbaum Associates Inc., 2002, pp. 791–806.

[11] J. D. Prothero and H. G. Hoffman, "Widening the field-of-view increases the sense of presence in immersive virtual environments," *Human Interface Technology Laboratory Technical Report TR-95*, vol. 2, 1995.

[12] S. Peleg, M. Ben-Ezra, and Y. Pritch, "Omnistereo: panoramic stereo imaging," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 3, pp. 279–290, 2001.

[13] P. Bourke, "Creating correct stereo pairs from any raytracer," http://http://paulbourke.net/stereographics/stereorender, 2001, (last accessed on 12th December 2014).

[14] O. Kreylos, "Good stereo vs. bad stereo," http://doc-ok.org/?p=77, 2012, (last accessed on 12th December 2014).

[15] R. Kooima, "Generalized Perspective Projection," http://csc.lsu.edu/~kooima/pdfs/gen-perspective.pdf, 2009, (last accessed on 3rd March 2014).

[16] O. Kreylos, "Homepage of Oliver Kreylos," http://idav.ucdavis.edu/~okreylos/, 2014, (last accessed on 3rd March 2014).

[17] NVIDIA Corporation, "NVIDIA GPU programming guide, version 2.5.0." https://developer.nvidia.com/nvidia-gpu-programming-guide, pp. 70–73, 2006, (last accessed on 8th June 2015).

[18] H. Grasberger and R. Habel, "Introduction to Stereo Rendering," http://www.cg.tuwien.ac.at/research/publications/2008/Grasberger_2008_ISR, 2008, (last accessed on 24th February 2015).

[19] J. J. LaViola, Jr., "A discussion of cybersickness in virtual environments," *SIGCHI Bull.*, vol. 32, no. 1, pp. 47–56, 2000.

[20] T. A. Stoffregen, M. H. Draper, R. S. Kennedy, and D. Compton, "Vestibular adaptation and aftereffects," in *Handbook of virtual environments : Design, implementation and applications.* L. Erlbaum Associates Inc., 2002, pp. 773–790.

[21] R. Kalawsky, *The Science of Virtual Reality and Virtual Environments*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1993.

[22] J. Tresilian, *Sensorimotor Control and Learning.* Palgrave Macmillan, July 2012, p. 245–246.

[23] H. Hunziker, *Im Auge des Lesers: vom Buchstabieren zur Lesefreude ; foveale und periphere Wahrnehmung.* Transmedia, 2006.

[24] V. Couture, M. S. Langer, and S. Roy, "Analysis of disparity distortions in omnistereoscopic displays," *ACM Trans. Appl. Percept.*, vol. 7, no. 4, pp. 25:1–25:13, Jul. 2010.

[25] R. F. Hess, F. A. A. Kingdom, and L. R. Ziegler, "On the relationship between the spatial channels for luminance and disparity processing," *Vision research*, vol. 39, no. 3, pp. 559–568, 1999.

[26] M. S. Banks, S. Gepshtein, and M. S. Landy, "Why is spatial stereoresolution so low?" *The Journal of Neuroscience*, vol. 24, no. 9, pp. 2077–2089, 2004.

[27] M. Doggett and J. Hirche, "Adaptive View Dependent Tessellation of Displacement Maps," in *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*, ser. HWWS '00. New York, NY, USA: ACM, 2000, pp. 59–66.

[28] R. Kooima, D. Roberts, and M. SubbaRao, "Real-time Digital Dome Rendering Techniques and Technologies," in *Proceedings of IPS2008*. International Planetarium Society, 2008.

[29] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti, "Surround-screen Projection-based Virtual Reality: The Design and Implementation of the CAVE," in *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '93. New York, NY, USA: ACM, 1993, pp. 135–142.

[30] Fraunhofer-Gesellschaft, "HyPI-6: 6-Sided-Cave," http://web.archive.org/web/20030203023821/http://www.vr.iao.fhg.de/6-Side-Cave/index.en.html, 2001, (last accessed on 25th April 2015).

[31] Fraunhofer IAO, "Virtual Reality Lab (VR Lab) mit HypI-6 (6-Wand-CAVE)," http://www.iao.fraunhofer.de/lang-de/geschaeftsfelder/engineering-systeme/277-virtual-reality-lab.html, 2015, (last accessed on 25th April 2015).

[32] C. Cruz-Neira, D. Reiners, and J. P. Springer, "An affordable surround-screen virtual reality display," *Journal of the Society for Information Display*, vol. 18, no. 10, pp. 836–843, 2010.

[33] H. Ishiguro, M. Yamamoto, and S. Tsuji, "Omni-directional stereo for making global map," in *Computer Vision, 1990. Proceedings, Third International Conference on*, 1990, pp. 540–547.

[34] Couture, V. and Langer, M. S. and Roy, S., "Panoramic stereo video textures," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1251–1258.

[35] S. Peleg and M. Ben-Ezra, "Stereo panorama with a single camera," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 1. IEEE, 1999.

[36] Zhu, Zhigang, "Omnidirectional Stereo Vision," in *Proc. of ICAR'01*, 2001, pp. 22–25.

[37] P. Rademacher and G. Bishop, "Multiple-center-of-projection images," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques.* ACM, 1998, pp. 199–206.

[38] N. Max, "Computer graphics distortion of IMAX and OMNIMAX projection," in *Nicograph '83, Conference Proceedings*, Oct 1983, pp. 137–159.

[39] Z. Wartell, L. F. Hodges, and W. Ribarsky, "Balancing fusion, image depth and distortion in stereoscopic head-tracked displays," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques.* ACM Press/Addison-Wesley Publishing Co., 1999, pp. 351–358.

[40] M. Trapp, H. Lorenz, and J. Döllner, "Interactive Stereo Rendering for Non-planar Projections of 3D Virtual Environments - With a Comparison of Image - and Geometry-based Approaches," in *GRAPP 2009 - Proceedings of the Fourth International Conference on Computer Graphics Theory and Applications, Lisboa, Portugal, February 5-8, 2009*, 2009, pp. 199–204.

[41] M. McGinity, J. Shaw, V. Kuchelmeister, A. Hardjono, and D. D. Favero, "AVIE: a versatile multi-user stereo 360° interactive VR theatre," in *Proceedings of the 2007 workshop on Emerging displays technologies: images and beyond: the future of displays and interaction*, ser. EDT '07. ACM, 2007.

[42] A. Simon and M. Gobel, "The i-Cone - a panoramic display system for virtual environments," in *Computer Graphics and Applications, 2002. Proceedings. 10th Pacific Conference on*, 2002, pp. 3–7.

[43] A. Simon and S. Beckhaus, "Omnidirectional Stereo Surround for Panoramic Virtual Environments," in *ACM SIGGRAPH 2003 Sketches &Amp; Applications*, ser. SIGGRAPH '03. New York, NY, USA: ACM, 2003, pp. 1–1.

[44] H. Lorenz and J. Döllner, "Real-time Piecewise Perspective Projections," in *GRAPP 2009 - International Conference on Computer Graphics Theory and Applications.* INSTICC Press, February 2009, pp. 147–155. [Online]. Available: http://cgs.hpi.uni-potsdam.de/publications/Public/2009/LD09

[45] H. Lorenz and J. Döllner, *High-Quality Non-Planar Projections Using Real-time Piecewise Perspective Projections* , ser. Communications in Computer and Information Science. Springer, 2010, vol. 68, pp. 45–58.

[46] H. Lorenz and J. Döllner, "Dynamic Mesh Refinement on GPU using Geometry Shaders," in *Proceedings of the 16-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2008*, February 2008. [Online]. Available: http://cgs.hpi.uni-potsdam.de/publications/Public/2008/LD08

[47] R. D. Hernández, A. Hardjono, and F. J. A. Cerdá, "Rendering stereographic 3d images in cylindrical spaces," diploma thesis, Polytechnic University of Valencia, University of New South Wales, September 2010, https://riunet.upv.es/handle/10251/13760.

[48] P. Bourke, "Using a spherical mirror projection into immersive environment," *Graphite, ACM Siggraph, Dunedin, Nov/Dec*, 2005.

[49] P. Bourke, "Omni-Directional stereoscopic fisheye images for immersive hemispherical dome environment," in *Proceedings of the Computer Games & Allied Technology 09*, 2009, pp. 136–143.

[50] P. Bourke, "iDome: Immersive Visualisation," http://paulbourke.net/dome/iDome/iDomePosterA1.pdf, 2013, (last accessed on 24th February 2014).

[51] B. Fröhlich, J. Hochstrate, J. Hoffmann, K. Klüger, R. Blach, M. Bues, and O. Stefani, "Implementing multi-viewer stereo displays," in *Proc. Int'l Conf. in Central Europe on Computer Graphics and Visualization (WSCG 2005)*, 2005.

[52] A. Pross, R. Blach, M. Bues, R. Reichel, and O. Stefani, "Optimization of a multi-view system based on pulsed LED-LCD projectors," in *Proc. SPIE, Stereoscopic Displays and Applications XXIII*, vol. 8288, 2012.

[53] W. J. Adams and J. H. Elder, "Effects of specular highlights on perceived surface convexity," *PLoS Comput Biol*, vol. 10, no. 5, 05 2014.

[54] Khronos Group, "The OpenGL® Shading Language - Language version 4.40," https://www.opengl.org/registry/doc/GLSLangSpec.4.40.pdf, 2004, (last accessed on 29th April 2015).

[55] Microsoft, "Reference for HLSL - Struct Type," https://msdn.microsoft.com/en-us/library/bb509706%28VS.85%29.aspx, 2015, (last accessed on 18th May 2015).

[56] Khronos Group, "The OpenGL® Graphics System: A Specification (Version 4.4 (Core Profile) - March 19, 2014)," https://www.opengl.org/registry/doc/glspec44.core.pdf, 2014, (last accessed on 18th May 2015).

[57] D. B. Lloyd, "Logarithmic perspective shadow maps," Ph.D. dissertation, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 2007.

[58] J.-D. Gascuel, N. Holzschuch, G. Fournier, and B. Peroche, "Fast non-linear projections using graphics hardware," in *ACM Symposium on Interactive 3D Graphics and Games*, Feb 2008. [Online]. Available: http://maverick.inria.fr/Publications/2008/GHFP08

[59] V. Popescu, P. Rosen, L. Arns, X. Tricoche, C. Wyman, and C. M. Hoffmann, "The general pinhole camera: Effective and efficient nonuniform sampling for visualization," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 16, no. 5, pp. 777–790, 2010.

[60] P. Rosen, "Rectilinear texture warping for fast adaptive shadow mapping," in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ser. I3D '12. ACM, 2012, pp. 151–158.

[61] Ogre development team, "Ogre homepage," http://www.ogre3d.org, 2015, (last accessed on 29th April 2015).

[62] CEGUI Team, "CEGUI homepage," http://cegui.org.uk/, 2015, (last accessed on 29th April 2015).

[63] Khronos Group, "The OpenGL Graphics System: A Specification," https://www.opengl.org/documentation/specs/version2.0/glspec20.pdf, 2004, (last accessed on 25th April 2015).

[64] Microsoft, "Platform Update for Windows 7," https://msdn.microsoft.com/en-us/library/jj863687.aspx, 2012, (last accessed on 25th April 2015).

[65] Bencina, R., "oscpack," http://www.rossbencina.com/code/oscpack, 2015, (last accessed on 29th April 2015).

[66] Microsoft, "Kinect for Windows," https://www.microsoft.com/en-us/kinectforwindows/, 2015, (last accessed on 29th April 2015).

[67] B. Bilodeau and AMD, "Direct3D 11 tutorial: Tessellation," http://amd-dev.wpengine.netdna-cdn.com/wordpress/media/2012/10/Direct3D%2011%20Tessellation%20Tutorial.ppsx, 2010, (last accessed on 29th April 2015).

[68] E. Persson and AMD Graphics Products Group, "ATI Radeon™ HD 2000 programming guide," http://amd-dev.wpengine.netdna-cdn.com/wordpress/media/2012/10/ATI_Radeon_HD_2000_programming_guide.pdf, 2007, (last accessed on 29th April 2015).

[69] Hashimoto, N. and Ishiwata, Y. and Sato, M., "Surrounding image projection with convex mirrors," in *Proceedings of the 3rd International Universal Communication Symposium*, ser. IUCS '09. ACM, 2009, pp. 146–149.

[70] S. Jeong and N. Hashimoto, "Immersive multi-projector display on hybrid screens with human-scale haptic interface," *IEICE Transactions on Information and Systems*, vol. 88, no. 5, pp. 888–893, 2005.