Medialogy Master Thesis
Computer Graphics
Thesis: MTA-161037
August 2016

Rendering the Light Field

User Evaluation of Light Field Rendering for Head Mounted Displays using Pixel Reprojection

Anne Juhler Hansen

Jákup Klein

Dept. Architecture, Design & Media Technology
Aalborg University
Rendsburggade 14, 9000 Aalborg, Denmark

This thesis is submitted to the Department of Architecture, Design & Media Technology at Aalborg University in fulfilment of the requirements for the degree of Master of Science in Medialogy - Computer Graphics.

Contact Information:
Authors:
Anne Juhler Hansen annejuhlerhansen@gmail.com
Jákup Klein jakupklein@gmail.com

Supervisor:
Martin Kraus martin@create.aau.dk
Department of Architecture, Design
& Media Technology

Dept. Architecture, Design & Media Technology
Aalborg University
Rendsburggade 14, 9000 Aalborg, Denmark

Web : http://www.aau.dk
Phone : +45 9940 9940
E-mail : aau@aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Department of Architecture, Design and Media Technology
Medialogy, 10th Semester

Title:

User Evaluation of Light Field Rendering for Head Mounted Displays using Pixel Reprojection

Project Period:

P10 Spring 2016

Semester Theme:

Master Thesis
Computer Graphics

Supervisors:

Martin Kraus

Project no.

MTA161037

Members:

Anne Juhler Hansen

Jákup Klein

Abstract:

Light field displays have advantages to traditional stereoscopic head mounted displays, due to the fact that the vergence-accommodation conflict is not present. Rendering light fields can be a heavy task for computers due to the number of images that have to be rendered. Much of the information of the different images is repeated. We use pixel reprojection from the corner cameras, and from that the remaining images in the light field can be made.
We compare the reprojected images with non interpolated images in a user test. In most cases the users were unable to distinguish the images. In extreme cases the reprojection approach is not capable to create the light field.
Pixel reprojection is a feasible method for rendering light fields as far as quality is concerned, but render time needs to be reduced to make the method practical.

Summary

Context: The light field display allows an observer to perceive a scene at different depths and angles by placing a distance-adjusted array of microlenses in front of a display, and hereby eliminate conflicting cues which have been under suspicion of causing visual discomfort and nausea. But when rendering for a light field display, *many* 2D subimages have to be rendered from different views, as seen from an array of different cameras.

Objectives: In this study we conduct a user evaluation of light field renderings for a head mounted display by comparing images created with different methods. Images made with high-precision methods (equal to rendering one camera per subimage) are compared to images made with pixel reprojection, in order to test the users' ability to perceive a difference. The advantage of using pixel reprojection is a potential reduction of computing power necessary to render a frame, since pixel reprojection is less depended on scene geometry complexity.

Methods: We implement a method for light field rendering, where instead of rendering all virtual cameras for each subimage, we render the four corner cameras and interpolate the rest of the views using pixel reprojection in the Unity Engine. The user test was implemented as a two-interval force choice test, were participants had to perform matching-to-sample tasks.

Results: Experiments where conducted, and the images created with the two different methods were compared. The image difference revealed that we in general have a good pixel match, but the pixel reprojection method has most problems with object edges and occlusion. The user evaluation consisted of 34 test participants, and the results showed that participants were in general not able to see a difference in the images, but the method falls short when we have extreme occlusion.

Conclusions: We found that our pixel matching is good, since the test participants seem not to notice small pixel displacement. The method also has its shortcomings, since participants did notice larger areas of difference e.g. extreme occlusion, but these cases are rare, and can be avoided by using more than four corner cameras. From our results, we are able to conclude that pixel reprojection is a satisfactory method for interpolating in-between views and thereby creating sufficient images for a light field display.

Keywords: light field rendering, pixel reprojection, shader programming

Contents

Summary **I**

1 Introduction **1**

2 Related Work **2**
2.1 Vergence-Accommodation Conflict 2
2.2 The Light Field 4
2.2.1 Parameterization of the 4D Light Field 5
2.3 Light Field Displays 6
2.3.1 Head-Mounted Light Field Displays 7
2.4 Light Field Rendering 7
2.4.1 Capturing the Light Field 8
2.5 Pixel Reprojection 9
2.5.1 Summary 10

3 Implementation and Methods **11**
3.1 The Light Field Display 11
3.2 Rendering the Light Field 14
3.2.1 Pixel Reprojection 16
3.2.2 Filling the Gaps 20
3.2.3 Shader Programming 20
3.2.4 Anti-aliasing 25

4 Experiment **26**
4.1 User Test 26
4.1.1 Test Setup 27
4.1.2 Two-interval Forced Choice Test 28
4.1.3 Performance Test 29

5 Results and Analysis **30**
5.1 Performance Test 34

6 Conclusion and Future Work **36**
6.1 Conclusion 36
6.2 Future Work 37
6.2.1 Performance 37
6.2.2 Occlusion and Gaps 37
6.2.3 Stereoscopic Rendering 38

A HMD Construction **39**

B Lens Calibration **41**

C Confirming Field of View **43**

D Virtual Scaling of the Screen Size **48**

E Initial Experiment **49**

F Image Difference **51**

G One Step Interpolation **52**

Bibliography **54**

Chapter 1

Introduction

Development of head mounted displays (HMDs) has evolved increasingly during the last years, especially when looking at consumer markets and consumers' use of HMDs eg. Oculus Rift, HTC Vive, Sony PlayStation VR, etc. One of the shortcomings and challenges of traditional HMDs is the lack of 3-dimensional cues, hereunder the parallax effect and correct eye accommodation. The vergence-accommodation conflict has been under suspicion of causing visual fatigue, eye-strain, diplopic vision, headaches, and other signs of simulation sickness [1].

In the future it might be possible to eliminate visual discomfort and nausea, since a light field display can provide correct retinal blur, parallax and eye accommodation, which may balance out some of the conflicting cues which are experienced with traditional HMDs. The light field display allows an observer to perceive a scene at different depths and angles by placing a distance-adjusted array of microlenses in front of a display.

When rendering for a light field display, several 2D subimages have to be rendered from different views, as seen from an array of different cameras. Instead of rendering an array of virtual cameras, views can be interpolated from only four rendered cameras. This project investigates the feasibility of using pixel reprojection to create light field renderings, and explore the benefits and shortcomings of pixel reprojection.

A head-mounted light field display has been built and implemented, and a user evaluation of the light field images has been conducted. The goal of the experiment is to find out if users are able to perceive a difference in the light field images created with the two different methods; the full array or the four cameras.

Chapter 2

Related Work

2.1 Vergence-Accommodation Conflict

In reality the human ocular system will adapt when focus is changed between different distances, such that the point of interest remains binocularly fused. Vergence and accommodation are parameters that influence our perception of depth and focus.

Accommodation refers to the physical shape of the lens of the eye, where the eye increases optical power to maintain a clear focused image. When accommodating, the shape of the lens inside the eye changes to allow for a focused image at that distance (see Figure 2.1). Accommodation can be consciously controlled, but usually acts like a reflex. Humans can change the optical accommodation of their eyes by up to 15 diopters (the inverse of the focal length in metres), but the accommodation diversity is reduced with age [2].

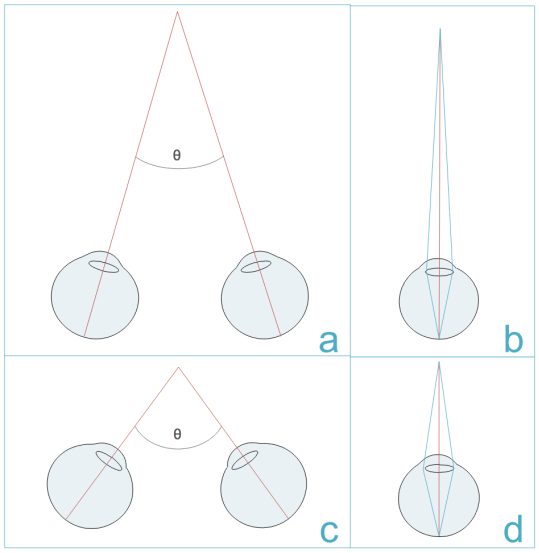

Figure 2.1: Vergence (a+c) is when the eyes move inwards (convergence) or outwards (divergence) towards a focus point. Accommodation (b+d) is the physical shape of the eye

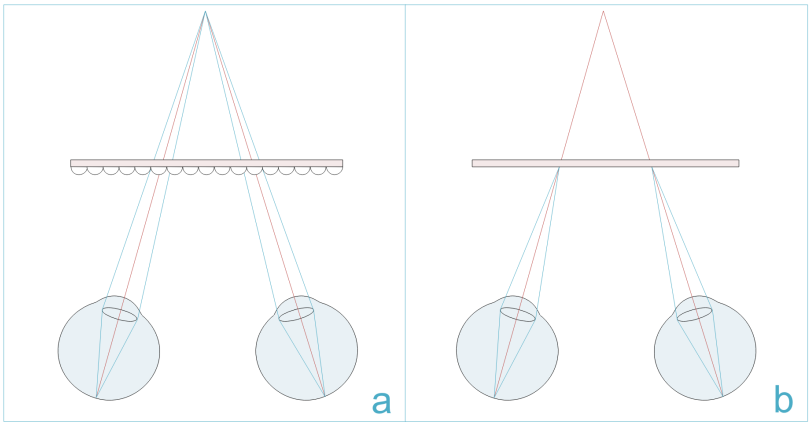

Figure 2.2: (a) Correct retinal blur as experienced through a light field display (b) Vergence-Accommodation Conflict

The vergence mechanism continually adjusts the angle between the two eyes such that features at the focus distance remain fused in the binocular vision. A pair of eyes will converge along the vertical axis, when an object in focus comes closer to the eye, or in other words, as the distance of the point of interest decreases from infinity. The eyes will diverge when the distance to a point of interest gets longer and/or goes towards infinity. The vergence and accommodation system interplay with each other in a feedback loop, since there is a secondary set of cues for both systems consisting of reciprocal signals from one another. This means that a change in visual cues will affect both system; stereo disparity drives the eyes to converge or diverge, and retinal blur prompts an oculomotor accommodation adjustment. To further strengthening the argument of these systems being very tightly coupled, Suryakumar et al. have shown that visual disparity in isolation elicits a fully comparable accommodation response to that of retinal blur [3]. The reciprocal secondary cues between accommodation and vergence serve to better coordinate the final accommodative response in natural viewing conditions [4]. However, in traditional stereo imaging where the depth is fixed, vergence towards a different distance will elicit conflicting cues between the two systems, and this has been linked to discomfort [5], visual fatigue, and reduced visual performance [1]. Research in resolving the vergence-accommodation conflict is still ongoing, and there are several proposals of solutions in both soft- and hardware [6] (see Section 2.4).

One of the consequent benefits of a light field display is that it allows natural accommodation and vergence (see Figure 2.2). Focusing at different distances simply determines which parts of the 2D image slices that are focused onto the retina. The light field images can be rendered to be perceived as if they are at natural (or unnatural) distances away from the viewer. By adjusting e.g. the field of view of each subimage camera, the depth of the optically reconstructed image will be influenced. By taking advantage of this fact, the virtual distances can correct for near- and far-sightedness of users [7], which can negate the use of glasses (or contact lenses) when wearing a HMD.

2.2 The Light Field

To understand the light field and its influence in computer graphics research, one must understand how to represent all light in a volume. The beginning of the light field and its definition can be traced back to Leonardo Da Vinci, who referred to a set of light rays as radiant pyramids [8]:

> "The body of the air is full of an infinite number of radiant pyramids caused by the objects located in it. These pyramids intersect and interweave with each other during the independent passage throughout the air in which they are infused."

Later on, the light field has been defined as the amount of light travelling in every direction through every point in space. Light can be interpreted as a field, because space is filled with an array of light rays at various intensities. This is close to the definition of the 5D plenoptic function, which describes all light information visible from a particular viewing position. This can be explained as recording the intensity of the light rays passing through the center of a pupil placed at every possible x, y, and z in a 3-dimensional volume, and at every angle θ and ϕ [8].

The plenoptic function allows reconstruction of every possible view, from every position, at every direction (see Equation 2.1).

$$P(\theta, \phi, x, y, z) \tag{2.1}$$

Since radiance does not change along a line unless it is blocked, the 5D plenoptic function can be reduced to 4D in space free of occluders [9]. The 4D light field can explain the total light intensity of each ray as a function of position and direction (see Equation 2.2).

$$P'(\theta, \phi, u, v) \tag{2.2}$$

The light intensity is given for every possible position u and v on a 2-dimensional plane, and angle θ and ϕ.

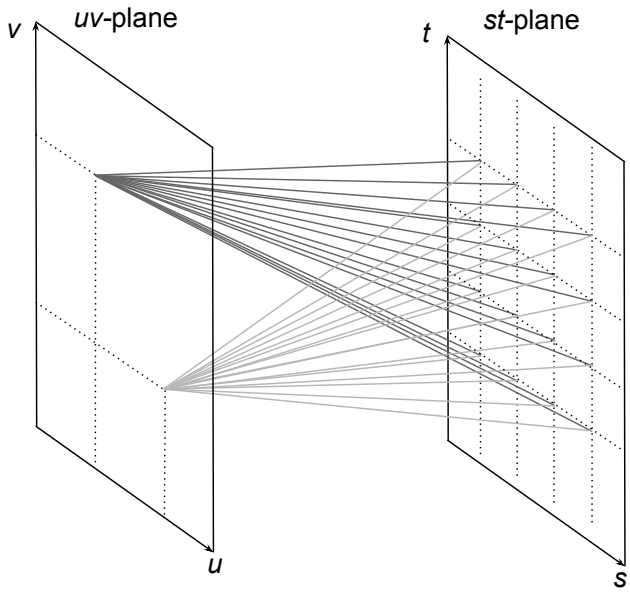

Figure 2.3: The light slab is a two-plane parameterization, where the st-plane can be thought of as a collection of perspective images of the scene, and the uv-plane corresponds to the position(s) of the observer(s).

2.2.1 Parameterization of the 4D Light Field

Levoy et al. described how a light field can be parameterized by the position of two points on two planes [9]. This parameterization is called a light slab (see Figure 2.3). A light ray enters one plane (the uv-plane) and exits another plane (the st-plane), and the result is a 2D array of images of a scene at different angles. Since a 4D light field can be represented by a 2D array of images, it has the advantage that the geometric calculations are highly efficient. The line of all light rays can simply be parameterized by the two points.

When parameterizing the light field into 2D-images, the elemental images correspond to images taken from different positions on the uv-plane, and each image represents a slice of the 4D light slab. In other words, the st-plane can be thought of as a collection of perspective images of the scene, and the uv-plane corresponds to the position of the observer.

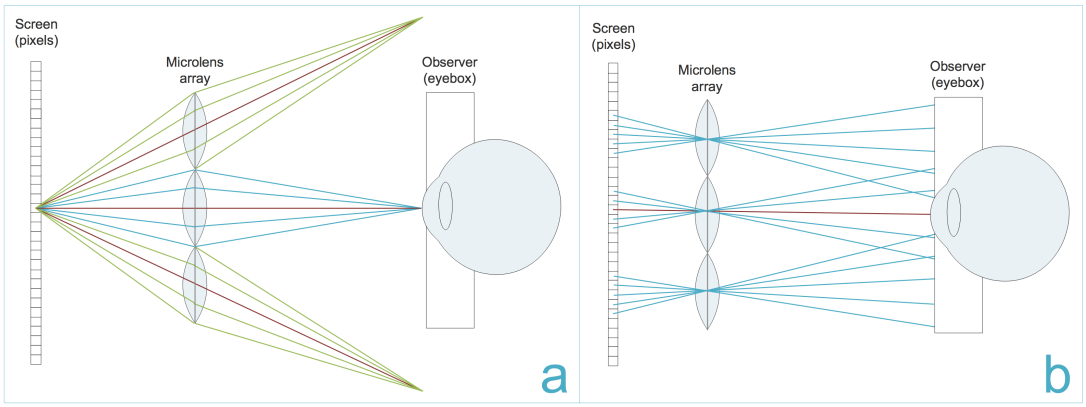

Figure 2.4: (a) Light from one pixel travels through lenses in the microlens array. Some of the light rays reach the eye, and some light rays will be bent in other directions. (b) Light from several pixels travel through the microlens array and (some) reach the eye or area of the eyebox with different incident angles. This allows the observer to focus his/her eyes while getting the corresponding light rays.

2.3 Light Field Displays

The light field can be optically reconstructed by placing a distance-adjusted array of microlenses in front of a display (see Figure 2.4). This is known as a light field display. The light field display allows an observer to integrate a correct 2D image of the light field at different depths and angles in accordance with the spatial and depth resolution that the light field contains. In other words the light field display allows an observer to accommodate and converge his/her eyes on a virtual object as if it were part the real world. Since every pixel on the screen emits light, and all lenslets in the full microlens array transmit the light in accordance with the angular information, the result will be a full light field. Depending on where the observer is looking, different subimage pixels will be used to create the view, and hence the 3-dimensional holographic effect can be experienced. The image seen through a light field display has focus cues, where the convergence point is the point in focus, and the rest of the image appears blurred just like the real world. Even a monocular experience of the light field will give appropriate depth and focus cues, since the eye will focus at a point behind the screen at the correct distance (see Section 2.1). Since distances can be virtually manipulated, the light field can be optically reconstructed to account for near- and far-sightedness of users.

Light field display technology is being researched in several areas: 3D-displays [10], light field projection [11], and holography [12]. Commercial products, like the Leia 3D display, are already on the market, and claim to give holographic imagery with content appearing to come out of a conventional liquid crystal display (LCD) and showing the parallax effect with head movement without the need of any glasses. Likewise does the head mounted light field stereoscope [13] not use a microlens array, but creates the light field via stacked liquid crystal panels and hereby emphasizes that light field renderings can be shown with different technologies.

2.3.1 Head-Mounted Light Field Displays

Head-Mounted Displays are still struggling with being heavy and having big and bulky optics [14]. Most HMDs do not account for the vergence-accommodation conflict (see Section 2.1), and they suffer from low resolution and a low field-of-view (FOV). Since light fields consist of more information than usual 2D images, light fields can improve on some of the limitations of traditional fixed-focus HMDs.

With the benefits from using microlenslet arrays in HMDs, Lanman and Luebke have shown that a light field display can be integrated into a HMD, which can both minimize the size of HMDs and potentially allow for much more immersive VR solutions compared to the fixed focus displays used in most common HMDs [7]. Lanmanand Luebke have created near-eye light field displays with a thickness of 1 cm. [15], and Shaulov et al. demonstrated that ultracompact imaging optical relay systems based on microlenslet arrays can be designed with an overall thickness of only a few millimetres [16].

We have also previously implemented a Near-eye Light Field Display [17] and examined the perceived accommodation range and spatial resolution. We evaluated the performance of test participants using standardized visual acuity tests and compared several accommodation ranges. We found that the best visual acuity scores were obtained when the distance between viewer and object were below 2m, and that as of today the most limiting factor of the Light Field Display is the resolution.

2.4 Light Field Rendering

One of the first times light fields were introduced into computer graphics was by Levoy et al. in 1996, where they used image based rendering to compute new views of a scene from pre-existing views without the need for scene geometry [9]. The technique showed a real-time view of the light field, where it was possible to see a scene with correct perspective and shading, and with the option of zooming in and out. When zooming in, the light samples disperse throughout the array of 2D slices, so the perceived image is constructed from pieces from several elemental

images.

Davis et al. have created a system for interactively acquiring and rendering light fields using a camera being waved around an object [18]. They present a new rendering algorithm that triangulates the captured viewpoints and is specially designed for the unstructured and dense data of the light field. Using direct light field rendering, Jeoung et al. have introduced an image-based rendering method in the light field domain, which attempts to directly compute only the necessary samples, and not the entire light field, to improve rendering in terms of complexity and memory usage [19].

Light field technology is competing with other technologies that are trying to display some of the same effects but with different advantages and short comings. Foveated rendering is a technique where the image resolution is not uniform across the image, and where the abilities of the human peripheral vision can be taken advantage of. The technique can be used to create retinal blur, which is the blurred perception of objects outside the center of gaze (and therefore in the peripheral vision).

Gupta et al. worked on tracking and predicting eye gaze accurately with the objective of improving interactivity using eye-gaze information by enabling foveated rendering or simulate retinal blur [20]. Since the method also can be used to accelerate graphics computation, Guenter et al. performed a user study on foveated 3D graphics. Their method tracks the user's gaze point and from that renders three image layers around it at progressively higher angular size but lower sampling rate. They state: "The result looks like a full-resolution image but reduces the number of pixels shaded by a factor of 10-15" [21],

Reducing complexity is highly desired when working with light fields, and (re)construction of overlapping views is a good place to start, since this is where the light field contains a lot of redundant information.

Much of the data is repetitive, especially when looking at a scene placed at infinity, where all subimages are created from parallel light rays. Instead of creating a virtual camera or capturing an individual subimage for each elemental image, interpolation can be used to reduce the computational effort.

2.4.1 Capturing the Light Field

In light field photography a 2D representation of the 4D light field can be captured and then sampled into a 2D image with a specific focus plane within the limits of the stored light field. The light field can be captured in several ways; either with an array of cameras [22, 23], by moving a camera forward and backward [24], or by using a plenoptic camera containing an array of microlenses [25].

The first hand-held plenoptic camera that captures the 4D light field in one photographic exposure was created by Ng et al. [25]. The 4D light field is reconstructed into a 2D image in software post-capture, and can compute sharp photographs focused at different depths. In other words this method creates a

synthetic aperture, that expands editing possibilities in post production by eliminating limitations related to a fixed aperture.

Interpolation strategies for optimizing resolution with light field photography are also being explored. Georgeiv et al. [26] have created an interpolation method that creates a better resolution in the final light field photograph by virtually increasing the amount of views to be more than the amount of microlenslets.

Naimark et al. created a stereo image capture rig, that captures a pair of stereo images [27]. From that a synthetic scene with depth could be calculated using cross dissolve. Since the light field gives a high accuracy of sampling it is possible to triangulate points into a point cloud, which provides the ability of tracking objects and semi-reconstruct objects and scenes 3-dimensionally.

This is one of the reasons why light field technology has potential benefits in the field of visual effect (VFX). Since light field photography essentially captures depth, it can be used to redefine previous methods (e.g. chroma keying) and develop new approaches (e.g. depth screen removal). Depth screen removal is one example of a new and improved technique for the VFX workflow, where the volumetric data from the light field can be used to disperse the object of interest from the background. The depth can among things be used to create semi-automated segmentation and rotoscoping [28].

VR is already exploring the use of live-action footage, e.g. with the use of the Ricoh Theta, which is an omnidirectional camera that with two fish-eye lenses captures 360° with a single shot. The captured images overlap, and can therefore be stitched together, taking every photo from that single point of view. Similar solutions include the Jaunt, the Nokia Ozo, and the GoPro Odyssey, but a 360° spherical image will though only create a flat panorama in VR, and will get no 3D and parallax effect.

The Lytro Immerge is a new light field solution for cinematic virtual reality (VR), with a configurable dense light field camera array, that is declared to have six degrees of freedom. With six degrees of freedom the solution claim to allow virtual views to be generated with precise visual perspectives in a seamless capture that requires no stitching [29]. The future might bring light field live-action footage to the VR platform, and therefore the motivating force to research light field renderings and evaluating it through user testing is both interesting and relevant for future studies and implementation.

2.5 Pixel Reprojection

Pixel reprojection is about reprojecting data (e.g. pixel color value) from one image to another. Geometrically valid pixel reprojection techniques have been studied by Kang, who states "If the depth value at each pixel is known, then the change in location of that pixel is constrained in a predictable way" [30]. The traditional approach for generating virtual views of a scene is to render a

3-D model, but with image-based rendering techniques new views can be created using pixel reprojection from source images onto the target image, and thereby the method only relies on simple interpolation calculations. Another benefit is that the cost of rendering is independent of the scene complexity.

Pixel reprojection can be used when rendering video, where data reprojection can exploit the natural temporal coherence between consecutive frames by caching expensive intermediate shading calculations performed at each frame, and then reuse this data when rendering subsequent frames [31]. In other words, temporal anti aliasing can be created by matching pixels from the current frame with pixels from the last frame, and using that information in-between views can be calculated.

Pixel reprojection can therefore also be used as a tool to optimize shaders, since reusing data between consecutive frames can accelerate real-time shading [32] [33]. The spatio-temporal coherence of image sequences has been exploited for several rendering systems [34][35] (e.g. for global illumination), and in general pixel reprojection is studied in the field of computer graphics.

2.5.1 Summary

We have investigated related work in the light field area, and found that Light field displays and renderings for light field displays are gaining interest, but still a lot of research needs to be done to understand the problems and benefits that arise with new technology.

By studying previous work of light field renderings, we find that by parametrization of the 5D plenoptic function we can reduce it to a 4D light field, which can be represented by a 2D array of images. The light field can be optically reconstructed using an array of microlenses, but first all subimages have to be rendered.

When previously implementing a Near-eye Light Field Display, the perceived accommodation range and spatial resolution where examined. In this project we re-implementing the physical Light Field Display with better alignment methods (and thereby reducing the need of small rotational and translational alignment changes done in software (see Appendices A, B, and C)). More importantly we want to focus on rendering the light field using pixel reprojection, and evaluate whether or not test subject are able to notice a difference in the image quality.

Chapter 3

Implementation and Methods

Our approach is to render only the four corner cameras of the subimage array, and then interpolate between these four views in order to create all subimages of the light field. We want to implement the interpolation of the subimages in the light field with the use of pixel reprojection, while maintaining correct perspective and shading, and investigate where short-comings of the interpolation might occur.

A user evaluation of the light field images will be conducted with the goal of finding out if users are able to perceive a difference in the light field images created with the full array of virtual cameras and our method using four cameras and pixel reprojection.

3.1 The Light Field Display

The head mounted near-eye light field display is constructed using an array of lenses (a Fresnel Technologies #630 microlens array) in front of a similar size adjusted array of rendered images (see Figure 3.1). The #630 microlens array has a focal length of 3.3 mm and a physical lenslet size of 1×1 mm, which determines the subimage array size and the number of pixels in each subimage.

Based on research by Lanman and Luebke [7], each of the lenslets in the microlens array can be seen as a simple magnifier for each of the subimages in the array. Depicting the individual lenslets as a thin lens is though only an approximation, since the lenslets are influenced by parameters of a thick lens; curvature, its index of refraction and its thickness. Since we are working with precision on micrometre scale, there are many sources of error, and therefore our approach is based on a thin lens model, but certain parameters are determined by observations (see Appendices A, B, C, and D).

The lens separation d_l can be found using the Gaussian thin lens formula (see Equation 3.1) where d_l is the distance between the lens and the display (with $0 < d_l \leq f$), f is the focal length, d_0 is the distance to the virtual image, and d_e is the eye relief.

$$\frac{1}{f} = \frac{1}{d_l} - \frac{1}{d_0 - d_e} \Leftrightarrow d_l = \frac{f(d_0 - d_e)}{f + (d_0 - d_e)} \tag{3.1}$$

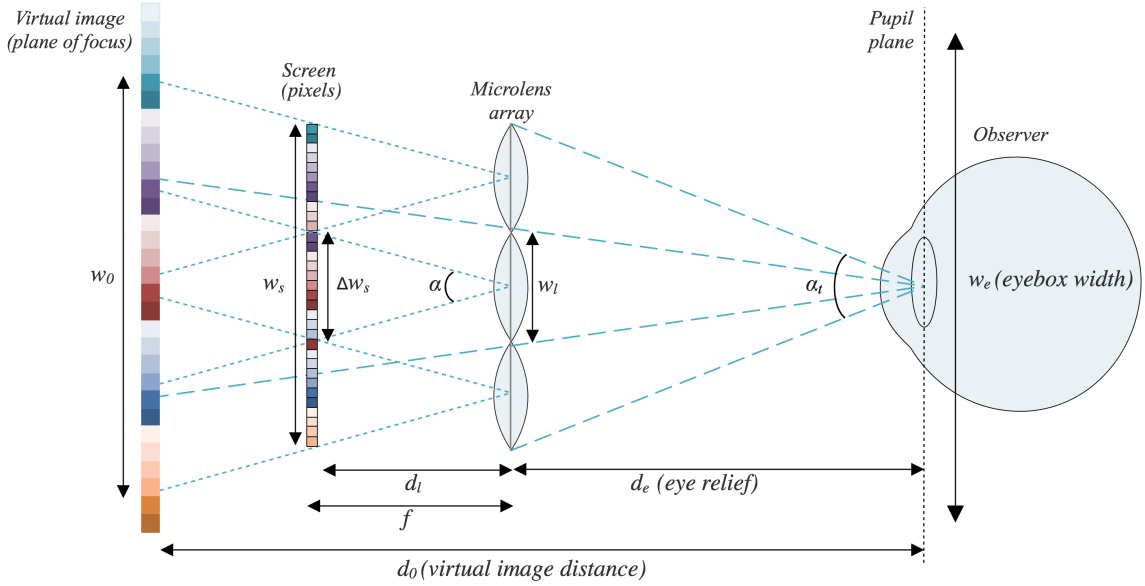

Figure 3.1: An observer sees the light field by looking through a microlens array in front of a screen, where each lens covers one subimage. Rays from different subimages enter the pupil of the observer, and the light field is experienced as one image, where the light samples disperse throughout the array of subimages. When focus (vergence and accommodation) is changed, the perceived image will be constructed from rays from other subimages.

The lens separation d_l is one of the parameters in the formula with the greatest impact on the perceived image, since the microlens array should be placed at a distance $0 < d_l \leq f$. With $f = 3.3$ mm the lens separation should be $d_l \approx 3.29$mm or in other words just below the focal length $f = 3.3$ mm. With an eye relief of 35mm and d_0 set to 1 meter, then the lens separation $d_l = 3.2888$ mm.

The lens separation was manually adjusted to the best possible alignment $d_l \approx$3.29mm using a 3D printed spacer (see Appendix A). Since the microlens array has a thickness of 3.3mm, it had to be turned with the flat side up, which might cause sources of error, since it is difficult to confirm the distance $d_l = 3.2888$mm (see Appendix B).

The magnification factor can be used to calculate the field of view, since it tells us the magnification of the image on the screen to the image plane at d_0. With $f = 3.3$mm and $d_0 = 1000$mm the magnification factor is M=293.42 (see Equation 3.2 [7]), where w_0 is the width of the virtual image at the plane of focus, and w_s is the width of the microdisplay.

$$M = \frac{w_0}{w_s} = \frac{d_0 - d_e}{d_l} = 1 + \left(\frac{d_0 - d_e}{f}\right) \tag{3.2}$$

The FOV is either limited by the extent of the lens (lens-limited magnifier) or it is limited by the dimensions of the display (display-limited magnifier). The lens-limited magnifier is influenced by $\frac{w_l}{2d_e}$, whereas the display-limited magnifier is influenced by $\frac{Mw_s}{2d_0}$, and since our FOV only can be limited by the lens (see Equation 3.3), we can then calculate the FOV for each of our virtual cameras in the array.

Field of view α (from the lens) per camera:

$$\alpha = 2 \arctan\left(\frac{\Delta w_s}{2d_l}\right) \tag{3.3}$$

The FOV per rendered camera is then calculated to be 17.28°. When confirming the FOV, we could though conclude, that a FOV of 19.86° gave a sharper image (see Appendix C). In the end we decided to work with the FOV = 19.86°, and we suspect the thin lens equation to be the source of error, since the approximation is not good enough.

Since a microlens array can be interpreted as a set of independent lens-limited magnifiers, the total field of view α_t from the viewer's eye can be found using the array width $N_l w_l$, and the eye relief d_e. N_l is the number of lenses, and w_l is the lens width. The total FOV α_t should then given by [7]:

$$\alpha_t = 2 \arctan\left(\frac{N_l w_l}{2d_e}\right) \tag{3.4}$$

The vertical FOV for 15 lenses is calculated to be $FOV_v = 24.2°$ and the horizontal FOV for 8 lenses is $FOV_h = 13.0°$ (see Equation 3.4).

We can also calculate the maximum spatial resolution N_p, by using the distance to the virtual image d_0, the FOV α, the magnification factor M and the pixel pitch p. When calculating with the used FOV = 19.86° and a pixel pitch p calculated to be 0.012 mm for both vertical and horizontal axis (with a resolution of 1280×720 and the screen size 15.36mm×8.64mm).

Lanman and Luebke [7] state that the maximum spatial resolution N_p is given by:

$$N_p = \left(\frac{2d_0 \tan(\alpha_t/2)}{Mp}\right) \tag{3.5}$$

We get a maximum spatial resolution of 121x64 px (see Equation 3.5), but since α_t is expanded by the number of lenses N_l, and part of the rendered subimages are repeated across some or all of the elemental images, this repetition will reduce the perceived spatial resolution. Also, since the virtual cameras are quadratic, we either will have to cut off the top and bottom to fill the 15x8 ratio

of the screen, or we will show the complete quadratic view plus extra views of the light field on the sides.

In ray optics focus appears at an image point, which is the point where light rays from the scene converge. The point is on the focus plane when it is in perfect focus. If the point is not on the focus plane the point will form a circle due to the light converging either in front of or behind the image plane. This is called the circle of confusion. Due to the circle of confusion the focus plane is the only section of a scene being in focus. Since the size of the circle of confusion decreases (approaching zero) when a point approaches the focus plane, then any circle of confusion below the lowest level of detail that the system is able to distinguish will appear to be in focus. On a screen the smallest distinguishable detail is the pixel, so if the circle of confusion is equal to or smaller than to one pixel, the point shows the highest focus resolution that it can.

The circle of confusion c_0' is therefore dependent on the optical characteristics that determine how the size of the circle of confusion changes over distance d_0'. Additionally the circle of confusion depends on the screen since the circle can not be smaller than a single pixel (see Equation 3.6 [7]). Note that the circle of confusion being calculated is not the circle of confusion on the image plane but rather on the focus plane, which changes over distance d_0'.

$$c_0' = \max\left(\left(\frac{d_0' - d_0}{d_0 - d_e}\right) w_l, \left(\frac{d_0' - d_e}{d_l}\right) p\right) \tag{3.6}$$

The depth of field is the area surrounding the focus point that appears to be in focus due to the circle of confusion being smaller than the smallest distinguishable detail (pixel pitch p). Two factors affect the circle of confusion: actual circle of confusion from the lens, and the smallest detail possible due to pixel pitch (see Figure 3.2). As long as the optical circle of confusion is smaller than a pixel the point appears to be in focus. In our setup the depth of field stretches from 24.8cm and continues to infinity.

3.2 Rendering the Light Field

Through the Unity engine, a virtual image is rendered for every lenslet that is within the bounds of the microdisplay, so the light field will be perceived as one holographic image with focus cues. Each subimage (or elemental image) is rendered to a portion of the microdisplay; optimally 15mm × 8mm out of 15.36mm × 8.64mm to utilise most possible of the spatial resolution (see Appendix C). The center of a subimage should be calibrated to correspond to the center of the lenslet, and the virtual camera array should form a grid that would ideally be spaced with the same distance as that between each lenslet (1mm × 1mm). Any spacing is usable, as long as the relationship follows the physical lens-spacing in both axes.

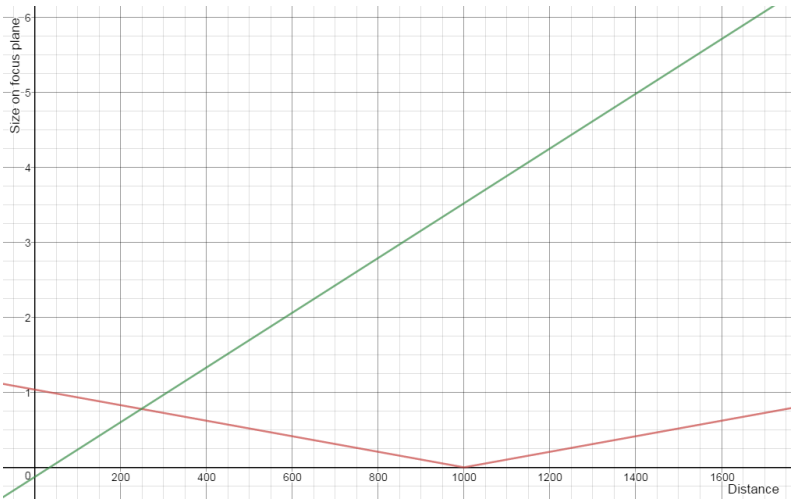

Figure 3.2: The red line shows the computed circle of confusion while the green line shows the smallest detail (pixel) that is possible to show due to pixel pitch p (see Equation 3.6).

Scaling the grid spacing in the scene essentially scales the virtual world size accordingly. For our rendering engine we increase this grid by a factor of 1000 to move the world further away from the nearest possible camera clipping plane. As already mentioned, object distances can be adjusted to correct for near- and far-sightedness (see Section 2.1).

The light field is computed by extracting the two-dimensional slice from the 4D light field (see Figure 3.3). Since the perceived image is constructed from pieces from several subimages, we need to render all these subimages in an array corresponding to the dimensions of our microlens array.

The secure and reliable solution would be to render 15×8 different virtual cameras, where each camera has the same alignment as the lenslets. We refer to this as a light field image created with virtual cameras, we consider this the golden standard and compare our approach to this method in tests. Our approach is to render only the four corner cameras of the subimage array, and then interpolate between these four views by using pixel reprojection to create the subimages in-between the four corner cameras.

Our method can be outlined by several steps (see Figure 3.4):

1. First we render the four corner cameras to separate render textures. The depth is saved in the alpha channel.

2. Then a shader calculates the in-between images on the x-axis on the top and bottom row (note: the images between the top and bottom row are not used and are, because the are empty, filled with an average colour of the available information from the four corner cameras (see Section 3.2.2)).

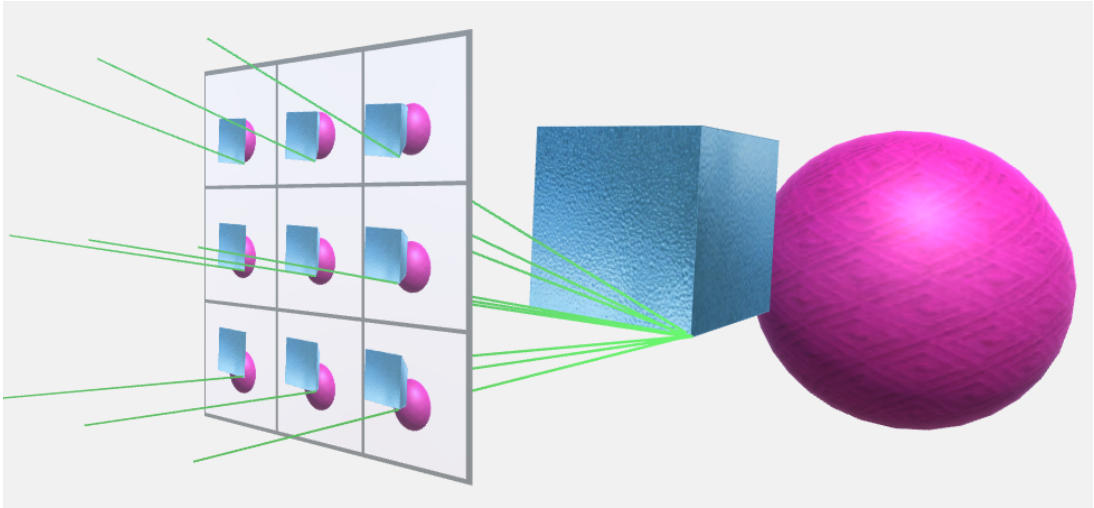

Figure 3.3: A 4D light field can be ssen as a collection of images of a scene, where the focal points of the cameras are all on a 2D plane.

3. The x-axis result is saved to a render texture, and from that another shader calculates the in-between images on the y-axis.

4. Again the result is saved to a render texture, and anti-aliasing is done using super sampling in a third shader.

5. At last the image is scaled to fit the display output by a fourth shader.

3.2.1 Pixel Reprojection

Pixel reprojection involves the redistribution of information from a set of input pixels to a set of output pixels. To capture an image of a scene consisting of vertices in a 3D volume (world space) the vertices must be transformed to the camera's space (camera/eye space), where a 2D image with perspective distortion within near and far plane can be generated (see Figure 3.5).

The interpolation of the subimages is accomplished by using pixel reprojection, where the pixels from the corner images are copied to the corresponding place in the interpolated subimage. To achieve this the pixel must be placed back to the 3D world and be "captured" to the interpolated subimage (see Figure 3.6). Here the view space must be projected to the image plane that will be displayed on the screen. The view space renders through a camera centered in the origin, hence view space is also referred to as camera space or eye space. The input pixel energy must be redistributed to the output pixel based on the exact overlap between these pixels. Not having a correct calculated image for one or more of the four corner cameras is a good way of debugging, since flaws in the corner views will always produce incorrect in-between views.

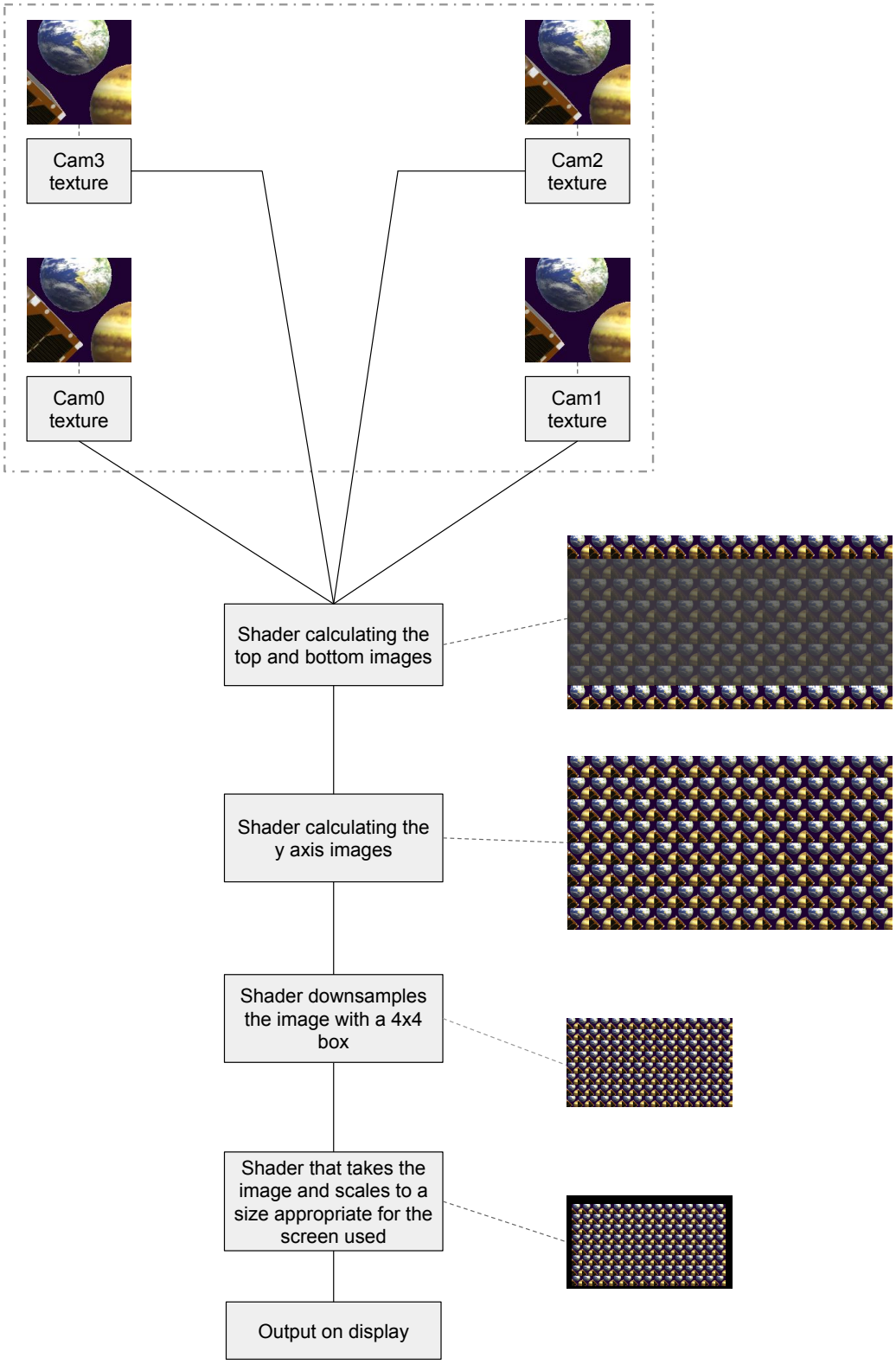

Figure 3.4: Flowchart of the complete process. Note that the step where the top and bottom row are calculated portions of the texture that are not used have been grayed out.

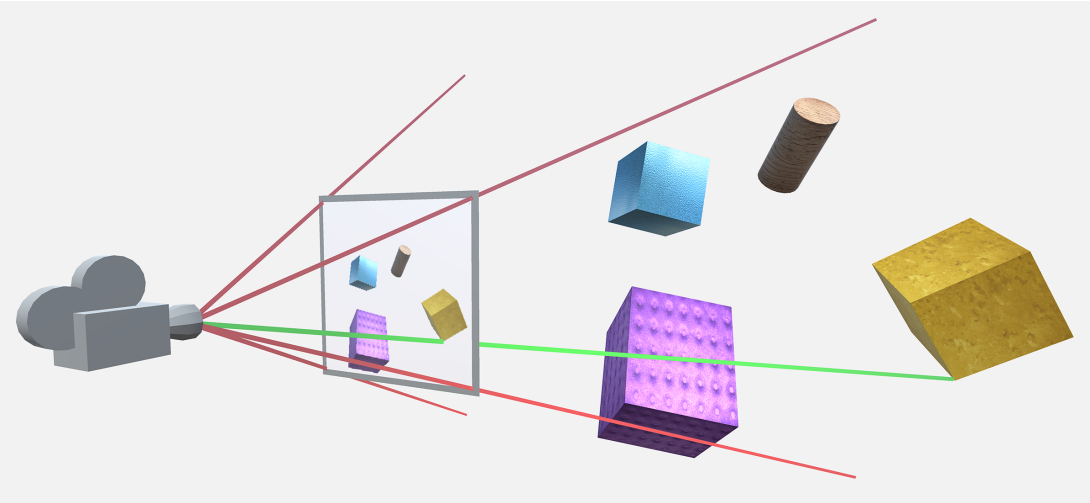

Figure 3.5: 2D projection: The vertices are transformed to eye space where a 2D image with perspective distortion is generated. The geometry in the scene is projected onto the 2D plane, and using that information the image can then be calculated.

The transformation goes back and forward between the projection plane (the 2D generated image) and the eye/camera space (the 3D scene with the camera as the center). All sub-views (interpolated cameras) have an individual position in world space and need to do a transformation between these spaces in order to generate the interpolated subimages. If the projection plane for one camera and the transformation in relation to the other camera is known, then the pixels can be reprojected to the other camera. Finally the view space is projected onto the 2D screen, where near and far clipping plane are obtained via frustum culling (clipping), and the clip coordinates are transformed to the normalized device coordinates.

Our transformation depends on the x-coordinates on both the projecion plane, x_p, and in eye space, x_e, as well as the near clipping plane n and the z-position z_e in eye space (see Equation 3.7).

$$\frac{x_p}{x_e} = \frac{-n}{z_e} \tag{3.7}$$

The clip coordinate system projects all vertex data from the view space to the clip coordinates by comparing x_{clip}, y_{clip}, and z_{clip} with w_{clip} (which are [x,y,z,w] in clipping space). Any clip coordinate vertex that is less than a certain $-w_{clip}$ or greater than a certain w_{clip} will be discarded, and then the clipping occurs.

The x-coordinate of eye space, x_e is mapped to x_p, which is calculated by using the ratio of similar triangles (see Equation 3.8).

$$x_e = -\frac{x_p \cdot z_e}{n} \tag{3.8}$$

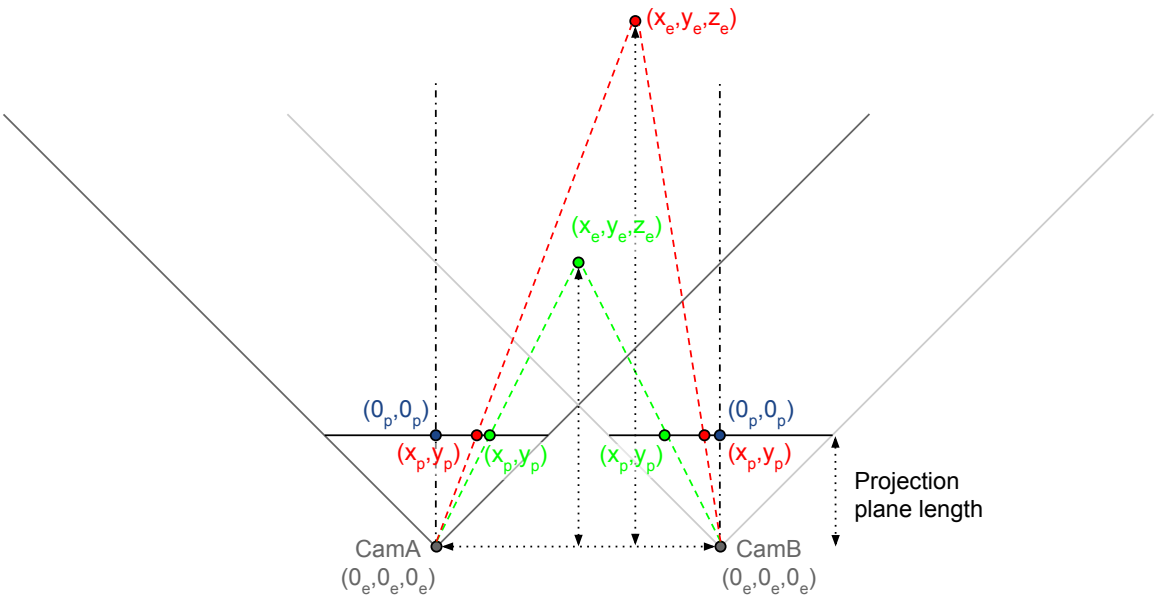

Figure 3.6: Pixel reprojection shown betwen two cameras (CamA and CamB). A pixel can be reprojected from a corner camera (CamA), to an interpolated camera (e.g. CamB). The cameras have coordinates in the 3-dimensional eye space, whereas the projection plane is 2-dimensional.

Likewise the transformation from eye/camera space to the projection plane is influenced by the position in eye space x_e, the position on the projection plane x_p, the near clipping plane n, and the depth z_e (see Equation 3.9).

$$x_p = -\frac{n \cdot x_e}{z_e} \tag{3.9}$$

We need the depth information of the scene to effectively interpolate between the images. Using perspective projection the relation between z_e and z_n is non-linear with high precision at the near plane and little precision at the far plane. We need to account for the non-linear relationship (from vertex position in object space to vertex position in clip space) to get correct distances and depth in a normalized [0,1] range [36].

The transformation from the projection plane to the eye space requires the depth from the eye space. The depth is saved from the corner cameras into the (unused) alpha channel. It was found through experimentation that a 24 bit texture (8 bits per channel) was not sufficient to give accurate depth information, if, however, a 32 bit (float32) texture was used, the problem was negated.

Accurate depth is crucial for the pixel reprojection method to work. This also means that no anti-aliasing can be performed on the four corner cameras due to

the fact that while the smoother edges of a anti-aliased image are more esthetically pleasing, they would no longer match the depth map, resulting in artifacts in the reprojected images. Anti-aliasing can still be achieved by supersampling after all pixel reprojection calculation are finished (see Section 3.2.4).

3.2.2 Filling the Gaps

There are cases where pixel reprojection will not yield a full image, but rather an image with gaps. This is because in some cases objects will occlude other objects in such a way that when the camera is being reprojected, information is missing. The effect can be seen, when an interpolated image is comprised of the pixels from two corner cameras, but because of the way the objects are placed in the scene, there are spots where the depth and colour are unknown (see Figure 3.7). The size of the "shadow" depends on the distance from the camera to the occluding object and the distance between the camera that captures the scene and the pixel reprojected "camera".

The problem is that the information needed is not available, so there is no easy way to find the correct information to fill the holes. The only easy solution would be to include more cameras to the scene (5 cameras, one in the center and one in each of the four corners, would eliminate many cases). When the information is not available then the holes must be filled with something that hides them as well as possible. One could use the pixels on the edge of the hole and use them to fill it out. This could however lead to visible pixel "streaks". Another solution is the use an average colour of the available information. In this project the missing pixel values were filled by using the subpixel position where the pixel value from the corner images were read, and then using the mean of these values, the pixel value could be created. The result is that the hole is filled with colours that are present in the scene and there are no pixel "streaks" (see Figure 3.8).

3.2.3 Shader Programming

The pixel reprojection requires a number of calculations to be made for every pixel. Because of the large amount of calculations and the repeatability of the calculations, a shader would be a good tool. In this case a cg shader was used in the Unity Engine.

The interpolation is implemented as an image post-processing effect. Unity works with image effects as scripts attached to a camera to alter the rendered output, and the computation is done in the shader scripts [37]. The rendered images can be accessed via render textures, which are created and updated at runtime [38].

There are disadvantages of using a shader for the pixel reprojections. It is easy to find out where a (known) pixel on one subimage would be reprojected to

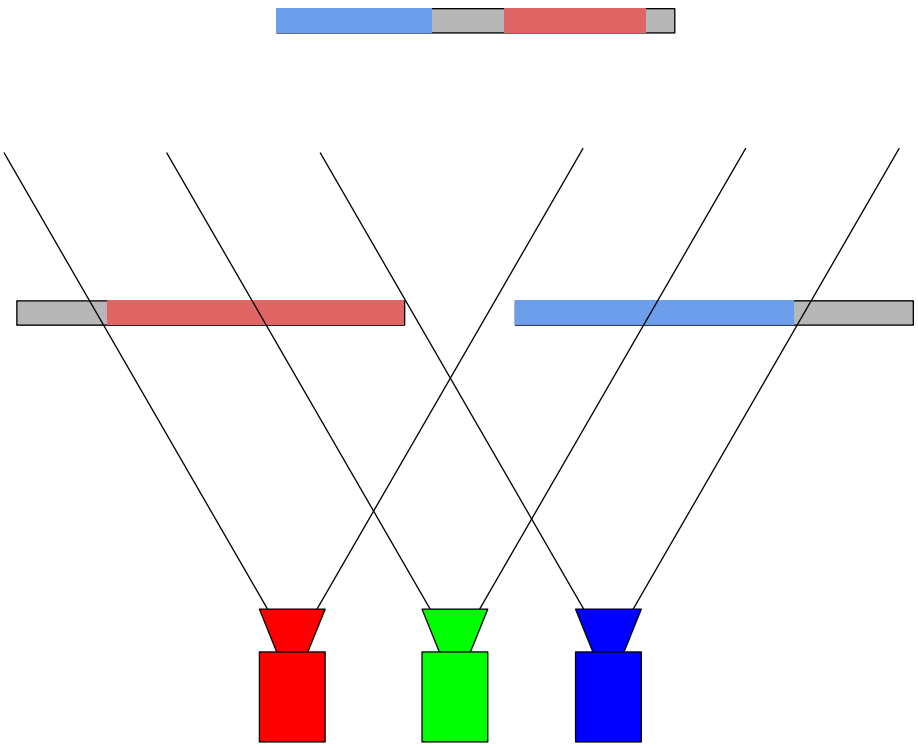

Figure 3.7: Example of a scene where the corner cameras (red and blue) do not have the necessary information to create an in between camera (green). The colour of the rectangles indicates what camera can see them. Red indicates that the red camera can see it, blue indicates that blue can see it and grey indicates that neither red or blue can see it.

on another subimage. It is not as simple to go the other way since there is no depth map for that subimage (because it is not one of the corner images).

One alternative to a shader is to use a compute shader. Compute shaders are capable of both random read and write. In this project random write would make it possible to make the pixel reprojection calculation from each of the corner pixel to all subimages, and write the color value directly to the relevant pixel.

Our method runs in a loop, where all calculated pixel values start with an output value of [0.0,0.0,0.0,2.0] (see Figure 3.9). If the depth is initially smaller than 2.0 (which it will always be), then the color for the actual fragment position is chosen from the camera with the lowest depth.

If the scene being captured by the cameras is placed at infinity, then all subimages will show the same image. In this case any position on any subimage should be filled with the colour from a corner camera at the same position. If, however, the scene contains elements placed closer to the cameras, then there will be a difference between the position of those points in the two projection

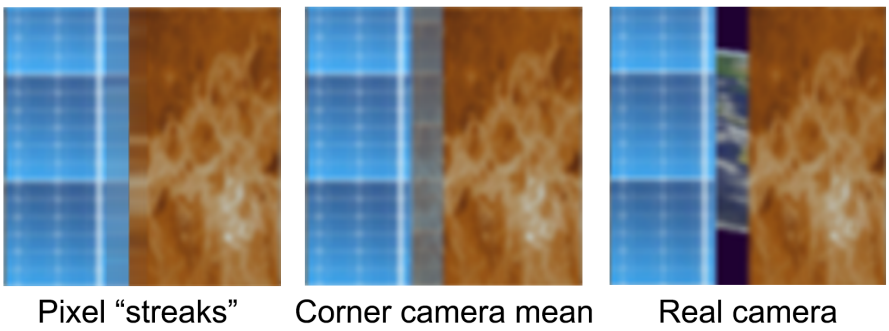

Figure 3.8: On the left the hole is filled with the edge pixels and repeated until the hole is full. In the center the hole is filled by averaging the pixel values in the corresponding position on the corner cameras. On the right is the image as it should look.

planes. This distance is greatest at the near clipping plane and zero at infinity (see Equation 3.9) where x_e is increased in order to offset it to a new camera position, the position x_p would depend on the depth z_e.

Knowing this and the two subimages' position relative to each other, one only needs to check all possibilities. Given any subimage position the same position is checked in the camera subimages, then the neighbours are checked until the maximum disparity is reached. Whether left or right neighbours are checked depends on where the corner camera is relative to the pixel reprojected "camera". The variables that affect maximum disparity is the distance to the closest object (or rather the cameras near clipping plane) and the distance between the corner camera and the point where the pixel reprojected "camera" is placed and the cameras field of view. None of reprojected pixels are necessarily a perfect match, but one should be the closest, and within 0.5 pixels.

It is important to note that the disparity is not one-dimensional but two-dimensional since the images are two dimensional. Here the x- and y-axis can be combined into one or the two axis can be calculated separately. The difference between the "one step" and the "two step" approaches would be computation time. If the "two step" approach is used, then one axis would be calculated first and saved to a texture, after this the other axis would be calculated. If the "one step" approach is used, then the number of times that values have to be written to textures would be reduced. One would, however, need to consider the added calculation (and logic) necessary to find the correct pixel in one step (see Appendix G). The two step approach was used in this project due to debugging purposes where errors would easily be located because the different stages of the program would be inspected separately.

The sub-pixel offset needs to be corrected since the subimages are a result of (up to) all four corner cameras. The offset will be slightly different for each

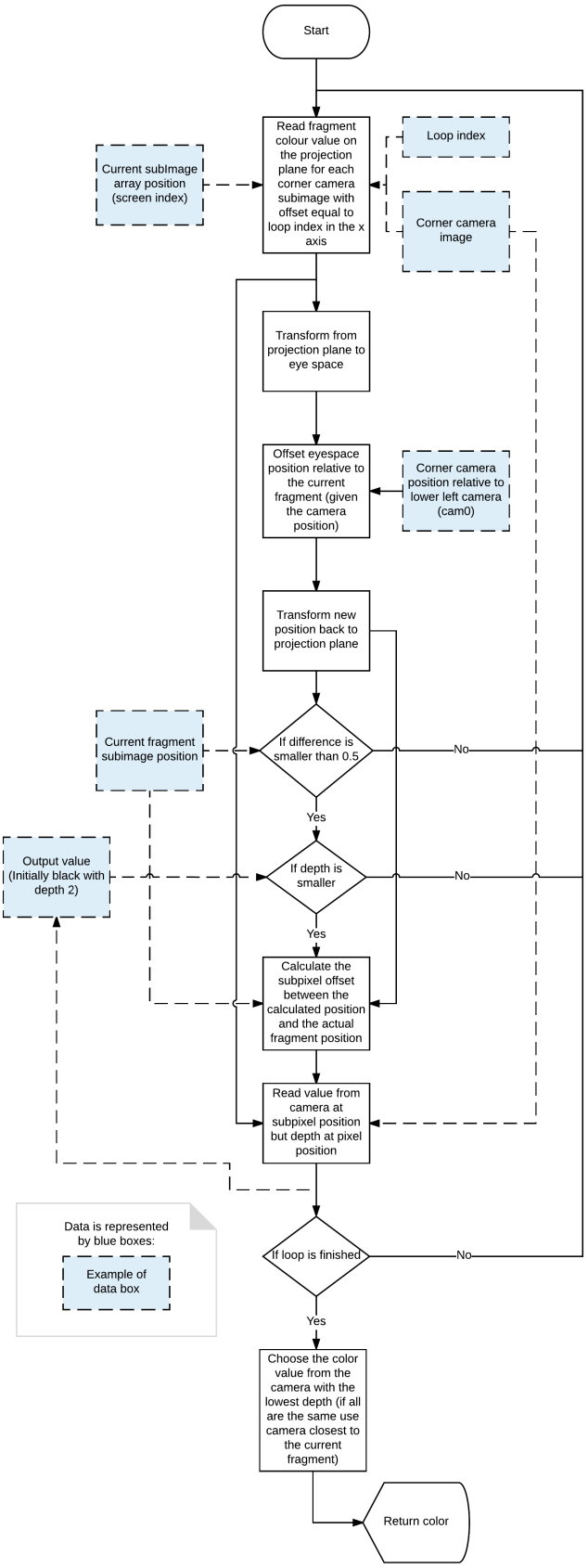

Figure 3.9: Image reprojection flowchart

camera due to their different positions. The closest pixel match is up to 0.5 pixels off. This offset is known. By subtracting the closest pixel match with the offset a point is found that is close to be a perfect match. In our setup this point is between two pixels a corner camera image, and then a color value is found by interpolation.

The pixel values lend themselves quite well to interpolation, but this is not the case with the depth map. The depth map can easily be interpolated on surfaces. The edges will, however, be smudged since the the difference between the two depths is so large. The result is three neighbouring pixels where one will have the depth of the background, one will have the depth of an object and the middle pixel will have a depth that is somewhere in between. The simple solution is not to interpolate the depth while interpolating the colour values. This will result in an image where the pixel values and depth values do not match completely but are quite close to correct. The downside is that the pixel values are interpolated while the depth isn't, meaning that the edge of an object can go beyond the edge in the depth map, effectively spilling colour to the neighbouring objects.

The interpolation of the corner images was split into three steps (before these steps the depth map is saved to the alpha channel of the corner cameras). The first step interpolated the subimages on the x-axis between the corner images (see Figure 3.4). The result was an image where the top and bottom row were filled with subimages. The rest of the image was filled with the mean of the colour values from the corner cameras at that subimage position. This is because nothing was there and it was therefore seen as a hole (since information was missing). The next step interpolated the values from the two rows of subimages and thus filled the remainder of the image with subimages. Because of the two step interpolation special care had to be made to avoid looking beyond the boundaries of the images in question. This is achieved by checking that the position is within the image, but after this step the sub pixel correction is performed. This process utilises linear interpolation of two pixels. If this interpolation is performed sufficiently close to the edge of the image, then the interpolation will be made between one pixel that resides on the images and one that does not. The problem was solved by "clamping" all textures, clamping means that the edge pixels will be repeated if a position is requested that is beyond the boundaries of the image. This is however not the case in the texture that has the top and bottom row filled with subimages. The edges of the texture are clamped but this is not the case between the two rows, thus the edge of the subimages facing the middle was repeated to achieve the same effect as the edges of the texture.

The two previous steps were performed on textures that were four times larger than the output (screen) size because of super sampling. The next step reduces the size by taking the mean of several pixels, thus minimising the "staircase" effect.

The last step in the interpolation was scaling. The reason for this step is that the previous steps are relying on that each subimage has a resolution in whole

pixels. This is, however, not the case as one millimeter on the screen occupies $\approx 83\frac{1}{3}$ pixels/mm, but due to scaling this number was $\approx 80\frac{1}{3}$ pixels/mm. The subimages had therefore a size of 81px×81px and were scaled down to size in the scaling shader.

3.2.4 Anti-aliasing

When rendering to a near eye light field, anti-aliasing must be used [7]. The problem with paring pixel reprojection and anti-aliasing is that anti-aliasing smooths out hard edges, but pixel reprojection requires the pixel colours to stay on the pixel position in order to have an accurate depth for that point and still maintain the correct colour. The depth can not be interpolated since that would result in edges being averaged between objects that are not at the same depth. This would result in pixels where the depth lies between objects in the scene effectively creating new objects. This would not show up in the corner images, but as soon as the camera view is moved from the original position.

The solution is to apply anti-aliasing after all pixel reprojection is completed. To achieve this super sampling is needed where an image with a resolution much higher than the intended output image is rendered, then the image size is reduced to its intended resolution by combining the pixel values of the high resolution image.

In this project a 4× (double width and double height) resolution image was used. When reducing the size, several methods can be used. One example is a 2×2 box where the mean of a box of 2×2 pixels is found and used in the lower resolution image. This will improve the look of the edges by reducing the "staircase" effect. The effect is, however, still strong. Another method is to use a 4×4 box where the outermost ring will overlap with the outermost ring in the neighbouring pixels [39]. In this project we used the 4×4 box to obtain a higher quality image.

Chapter 4

Experiment

4.1 User Test

This experiment aims at statistically comparing if subjects can discriminate between the image created with virtual cameras (VC), created with 120 (8×15) virtual cameras in Unity Engine, and the interpolated image (IC), created with our pixel reprojection method. The 120 camera image was created by capturing the camera views to individual render textures (162 px $\times$ 162 px) and combining them to a larger render texture that would fit 15×8 subimages, these were super sampled and scaled to the appropriate screen size. Essentially the same method as when interpolating the images just without the interpolation. 5 different scenes were tested (see Appendix F) in a total of 10 different image tests (5 shown with the light field display and 5 single-image (position [4;8] out of [8;15]) on a computer monitor.

The 5 different scenes were designed to test different rendering scenarios, and how the difference in geometry influence our rendering method. The different scenes include various numbers of objects, shapes, sizes and textures.

1. Scene with many objects occluding each other
2. Scene with few objects occluding each other
3. Scene with semi-few object occluding each other
4. Scene with curved texture
5. Scene with extreme occlusion (objects 12 cm away from the camera).

Image 5 was intentionally designed to fail the test. The image was created to push the boundaries of the method with the presumption that the test participants were able to notice a difference between IC and VC. It is also important when designing an experiment to account for participant frustration. If the test participants are never sure if their answers are correct or not, they can get frustrated, and decide that the task is impossible and then start answering randomly [40]. This would bias the results and should particularly be avoided.

4.1.1 Test Setup

The test took place at Universitarium, an experimentarium in Aalborg organised by Aalborg University, UCN, Tech College, and Aalborg Municipality. Universitarium is targeted at young children but they are accompanied by adults. Both adults and children took part in the test.

Results from 34 test participants are in the experiment, since some samples have been removed due to test participants having bad sight. Since the objects in the scenes were within 12 cm to 6 m, then test participants with nearsightedness were fit, but participants with farsightedness would bias the results, since farsightedness does not allow participants to accommodate on objects that are close. Therefore most test participants had normal vision or corrected to normal vision, and a few test participants were in the (glasses/contact lenses strength) range -1.75 to +0.50, but do not have corrected to normal vision. All samples were independent. 16 female and 18 male participants attended and their age ranged from 9 to 67 years (some participants refused to answer the age question).

The test participants were asked to sign a consent form (if they were underage, their guardian would sign). They were explained that they would be shown a reference image and then two other images - one of which was the same at the reference image - and that they should identify it. They were explained how to use the controller that they used to switch between the images. They were given the choice to put headphones on for auditory feedback, when they pressed the same button more than once or when they tried to choose the reference image. If they declined, then the facilitator would give the feedback. They first took the test with the light field HMD, and after that they took the same test where the center subimage of the light field was shown. Finally they filled out a questionnaire. The order of the images was randomised.

Controller

An xBox 360 controller was used as an input device for the test participants. The screen was black when they took the HMD on. They would then start the test by pressing the A, B, X or Y button, and the reference image would be shown. They could then press the left trigger to view one image or the right trigger to view the other image. They could also press the A, B, X or Y button to view the reference image again. They could view the different images as many times as they saw fit. When they had made a decision they would press both bumpers to choose the currently shown image (see Figure 4.1). If the current image at that time was the reference image, the controller would vibrate to indicate that the image could not be chosen.

Figure 4.1: The controller used as the input device for the test. The same image was used to explain the controls to the test participants.

4.1.2 Two-interval Forced Choice Test

A forced choice test is one that requires the test participants to identify a stimulus by choosing between a finite number of alternatives. We chose the 2-interval forced choice test where test participants must choose one of two alternatives with no neutral alternatives listed. The test participants were asked to solve several matching-to-sample tasks, where the standard stimuli (the sample or reference) is shown together with two other stimuli (the comparison stimuli), and then the test participants are requested to choose the comparison stimulus that most closely matches the reference.

The experiment was conducted as a delayed matching-to-sample, where the test participants were first shown a reference image, and then after the sample was removed two stimuli were presented sequentially. The inter-stimulus interval (ISI), which is the break between two stimuli, should be minimum 250 ms to help prevent temporal integration and masking effects. The time spent looking at each stimuli, the inter-trial interval (ITI), should be longer than the ISI [40].

With two possible choices shown sequentially this is referred to as a two-interval forced choice (2-IFC) procedure. If the test subjects can do no better than a random guess, then the test has been passed, meaning that we can con-

clude that the test participants experience no difference between VC and IC.

The 2-IFC tasks are:

1. The reference image is shown.
2. Two visual stimuli are presented in random order (visual stimulus and reference image can be revisited as many times as the test participant desires and the delay interval ISI is properly implemented).
3. The test participant chooses one of the two visual stimuli.

This test is passed if the probability for test participants to incorrectly identify IC as VC is greater than 19% with a confidence level of 95%. This corresponds to the commonly used threshold of test participants guessing incorrectly minimum 25% of at least 100 trials and complies with true hypothesis testing where the probability of incorrectly rejecting the null hypothesis is less than 5% [41].

The probability mass function for the number i of incorrect answers [42]:

$$f(i|n, p_{null}) = \frac{n!}{x!(n-i)!} p^x (1 - p_{null})^{n-x} \tag{4.1}$$

where p_{null} is the probability of IC incorrectly identified as VC, i is the number of incorrect answers and n is the number of trials. From the probability mass function we can find the critical number i_c which is the minimum amount of test participants that need to incorrectly identify the IC image to be the best match to the the reference image (VC) [42].

$$i_c(n, p_{null}) = min\{i| \sum_{j=i}^{n} f(j; n, p_{null}) < 0.05\} \tag{4.2}$$

With 34 test participant the critical number $i_c = 11$.

4.1.3 Performance Test

The implementation of pixel reprojection made in this project was made in order to see the possibilities (and limitations) of this approach. It was, however, not optimised and thus the performance should not be comparable with alternative methods, a performance test was nonetheless performed. The test is a comparison of two methods to render light fields; one is pixel reprojection, the other uses 120 cameras in the Unity Engine. The image size and position on the viewport/screen is adjusted such that it matched the subimages in the other method (making it compatible with the HMD). We refer to this method as "Direct to Screen" (DS).

The data collected is the time it takes to render a frame, 1000 samples are collected. The frame rate will in some cases be low on start but will stabilise. Data capture is started after 100 frames to avoid these outliers. The data is stored in an array during the test and written to a file after all data is captured.

Chapter 5

Results and Analysis

The results from the experiment show that with $i_c = 11$ at least 11 of the test participants would have to choose our image, IC, to match the reference image, VC, in order for the test to be passed. With 10 different image tests (5 shown with the light field display and 5 on a computer monitor) we see that image 1-4 passed the test (see Table 5.1 and Figure 5.2). When looking closely at the images (see Figure 5.1), we can find small mistakes in the IC images, and it is especially easy to notice the difference between VC and IC on image 5. Image 5 was designed to show the inadequacy of our method. Holes in the interpolated images are created when occluded objects need to be shown on the screen, when this happens no information is available and a hole is created. We expect our test participants to notice the difference in IC and therefore will choose VC to match the reference image (VC).

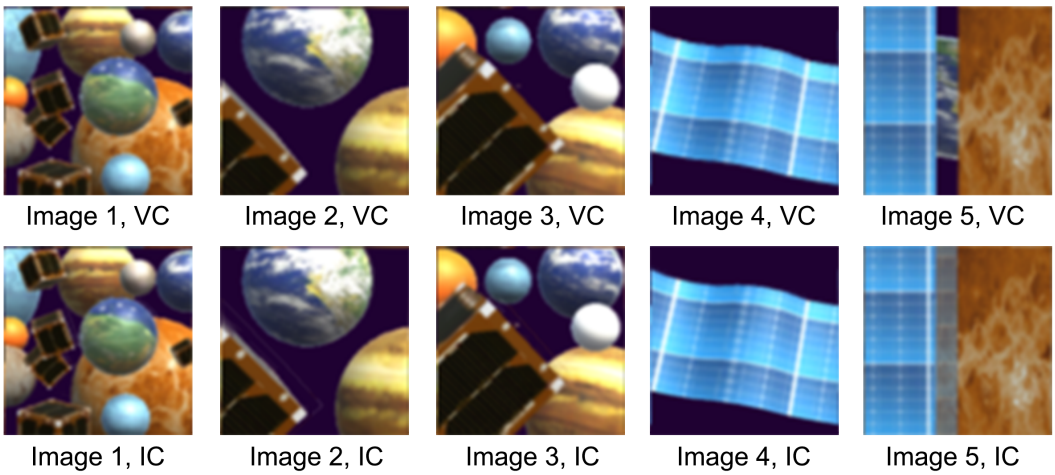

Figure 5.1: Image samples from the center virtual camera of the five different scenes (position $[4, 8]$ out of $[8, 15]$). When looking closely we can see small mistakes in the IC images, and especially Image 5 have a large difference.

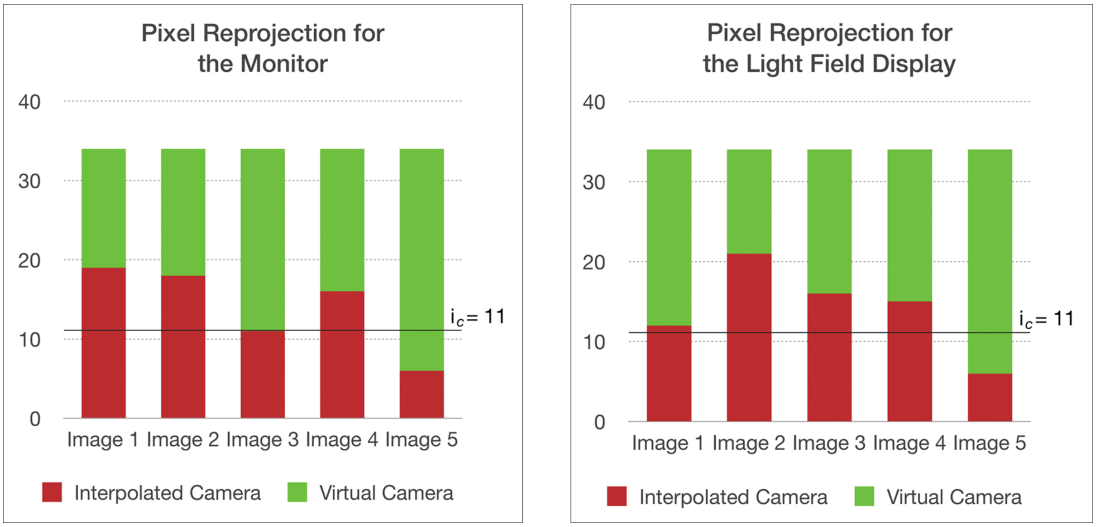

Figure 5.2: When 11 or more test participants choose IC, we can conclude that the test participants can do no better than a random guess, and therefore that they do not see a differenec between VC and IC.

In our setup with 15×8 cameras (120 cameras total) and a inter-camera-distance matching our microlenses ($1mm \times 1mm$) we were pressing the boundaries of the method when objects were only 12 cm away from the cameras. Areas (holes) that are invisible to the corner cameras (the occluded regions) becomes larger when the objects are close to the camera, but small problems can occur at any distance. With higher disparity, the occlusion will also be more extreme.

With only 6 test participants choosing IC for image 5 the critical number $i_c = 11$ was not reached and therefore the tests failed. We can then conclude that our method is inefficient when we have extreme occlusion, since participants are able to notice the difference. When subtracting IC from VC and taking the absolute value, we can see the difference between the two images (see Figure 5.3).

We have found the difference between all VC and IC images (see Appendix F), and analysed the pixel difference for the red, green and blue colour channel individually. A complete pixel match will be shown as black [0], and the pixel differences is normalized and therefore in the range [0;1] (for our test images our difference is between [0;0.6] (see Table 5.2). The worst maximum pixel value is found in image 5 in the green channel, and as already mentioned, we have extreme occlusion in image 5, which is also why this is the image with the worst result. We do though also find relatively high peak values in image 3, where we have a maximum difference value of 0.4471 in the green channel (in other words there is a pixel value difference of 44.71%). But since the maximum values are only peaks of the whole image, it is also interesting to look at the overall difference in the images.

Image no.	$\mathbf{i_{lfdisplay}}$	**LF Display**	$\mathbf{i_{monitor}}$	**Monitor**
Image 1	12	PASSED	19	PASSED
Image 2	21	PASSED	18	PASSED
Image 3	16	PASSED	11	PASSED
Image 4	15	PASSED	16	PASSED
Image 5	6	FAILED	6	FAILED

Table 5.1: Image 1 to 4 all have values equal to or above the critical minimum $i_c = 11$, and therefore we can conclude, that the test participants were not able to notice a difference in the images. Image 5 did not pass the test, since only six test participants chose our IC image, and this is below $i_c = 11$.

Image no.	**Min (R)**	**Max (R)**	**Min (G)**	**Max (G)**	**Min (B)**	**Max (B)**
Image 1	0	0.3569	0	0.3725	0	0.3569
Image 2	0	0.3686	0	0.3804	0	0.3725
Image 3	0	0.3922	0	0.4471	0	0.3882
Image 4	0	0.2157	0	0.3020	0	0.2902
Image 5	0	0.5451	0	0.6431	0	0.4549

Table 5.2: The minimum value is always 0, meaning that for all pixels, the smallest difference we find is equal to zero, and thereby an exact pixel match. The maximum value differs for the different images, with the highest peaks in image 5. That signifies that some pixels are not well reprojected, and the result is that some pixels of IC does not match VC.

The mean pixel value of the difference image will as previously mentioned be 0 (completely black) if we have an exact pixel match for all pixels in the image (range is still normalized to [0;1]). In general we see the largest mean in image 5 and the smallest mean in image 4 (see Table 5.3). A low mean difference does though not necessarily indicate that more test participants will choose IC, but it gives us a general idea of how similar/different the images IC and VC are.

We can see that our method has a small image difference, and the difference is largest around the edges of objects (see Figure 5.4), and/or when we have extreme occlusion and data simply is not available. We can also see small pixel value differences in textures, but in general we have many black or dark pixels, and thereby a good pixel match. We see that the pixel value difference is not equally spaced on the whole image, and therefore we can also look at the pixel difference median (see Table 5.4).

The median value of 0.0078 can be seen repeated several times, and is equiv-

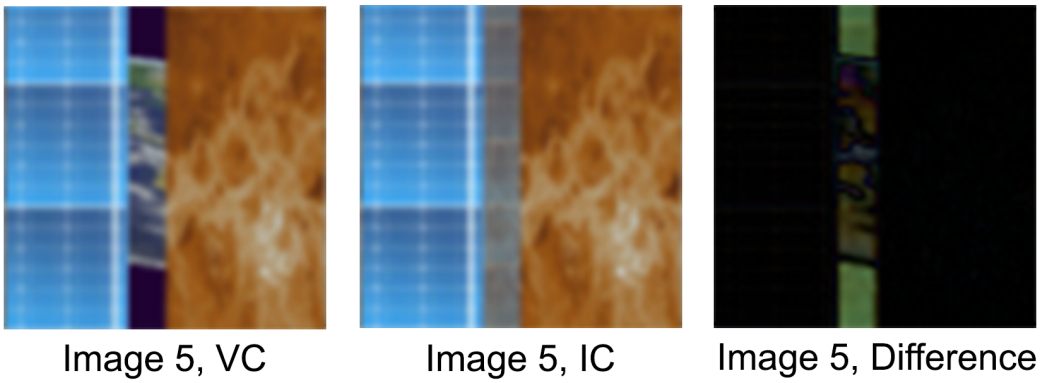

Figure 5.3: Example of the shortcomings of our pixel reprojection method; When objects are occluded from the corner cameras they can not be reprojected and therefore we are missing information. Left image shows what the image should look like. Center image shows our method with mean corner camera colors used for filling holes. Right image shows the difference. NOTE: Picture *Image 5, Difference* has enhanced brightness and contrast for printing

alent to a pixel difference of 0.78% or 2 in the range [0;255] (since we have an 8-bit image per colour channel). The median is in general low, meaning that the difference of the images for minimum half of the pixels have a color change of maximum 1.18%. As already mentioned the pixel value difference is not equally spaced on the image, and colours used in the scene will affect the mean and median difference of the images: were dark and bright colours meet (especially edges and occlusion), we can expect to see a larger pixel difference. Our pixel matching is good, though not perfect, but since the test participants seem not to notice small pixel displacement but rather notice larger areas of difference (e.g. extreme occlusion), we can conclude that pixel reprojection is a sufficient method for interpolating views in-between corner cameras. The results indicate that the method is satisfactory for both light field renderings and single-image renderings shown on a screen. The test participants were in general not able to see a difference between our image IC and the 120 camera image VC, but when objects get really close to the cameras, shortcomings of our method will be noticed, since we are missing camera information due to extreme occlusion. In our setup the extreme occlusion happens when objects are app. minimum 12 cm away from the camera(s), but since our depth of field stretched from 24.8 cm and continues to infinity, then we have created a scenario that in all cases is inadvisable.

Image no.	**Mean (R)**	**Mean (G)**	**Mean (B)**
Image 1	0.0133	0.0121	0.0155
Image 2	0.0098	0.0091	0.0116
Image 3	0.0150	0.0133	0.0164
Image 4	0.0099	0.0084	0.0089
Image 5	0.0236	0.0245	0.0215

Table 5.3: A low or high mean difference does not necessarily indicate whether or not test participants will choose IC, but it gives us a general idea of how similar/different the images IC and VC are.

Figure 5.4: Example of the short comings of our pixel reprojection method; edges and textures can have a small pixel value difference. NOTE: Picture Image 5, Difference has enhanced brightness and contrast for printing

5.1 Performance Test

The performance test consisted of two parts. One compared the super sampled pixel reprojection to the DS method (see Section 4.1.3). The other compared DS to pixel reprojection without super sampling.

The test was performed on an Asus K53SV with memory upgraded to 16GB.

Mean render time of super sampled pixel reprojection (mean 0.1870 sec, sd 0.1577) was significantly larger than DS (mean 0.0184 sec, sd 0.0027) (twosample t-test, $p = 2.2 \times 10^{-16}$).

Mean render time of (non super sampled) pixel reprojection (mean 0.1794 sec, sd 0.1511) was significantly larger than DS (mean 0.0184 sec, sd 0.0027) (twosample t-test, $p = 2.2 \times 10^{-16}$).

The difference between pixel reprojection and DS is around a factor 10. The DS method has a quite high frame rate of ≈ 54.3 fps and could be considered

Image no.	**Median (R)**	**Median (G)**	**Median (B)**
Image 1	0.0078	0.0078	0.0118
Image 2	0.0078	0.0039	0.0078
Image 3	0.0078	0.0078	0.0118
Image 4	0.0078	0.0039	0.0039
Image 5	0.0078	0.0078	0.0078

Table 5.4: Table with the median difference value of the three channels in the five images.

usable, although rendering is only performed for one eye. The pixel reprojection method on the other hand has a completely unusable framerate of ≈ 5.35 fps, and super sampling doesn't appear to a considerable influence on the frame rate (≈ 5.57 fps without super sampling). Note that all renderings were made for one eye, and stereo should be used in any production model.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Our approach was to render only the four corner cameras of the subimage array, and then interpolate between these four views in order to create all subimages of the light field. We have implemented the interpolation of the subimages in the light field with the use of pixel reprojection, while maintaining correct perspective and shading, and investigated where shortcomings of the interpolation occur.

4 out of 5 images passed the test, meaning that test participants were not able to notice a difference between the image created with 120 cameras and our image created with 4 cameras and pixel reprojection. The results were applicable for both images rendering for a light field display, but did also pass the test on a computer monitor.

Image 5 was deliberately designed to fail the test in order to find the short comings of the pixel reprojection method. In our setup we have problems when points are invisible to the corner cameras, and we therefore are missing information to create the in-between views. Missing data creates noticeable gaps (extreme occlusion) when objects are e.g. approximately 12 cm away from the camera(s), but since our depth of field stretched from 24.8 cm and continues to infinity, then we can argue that this scenario in all cases is inadvisable. In other words, the general problem is not extreme occlusion but points that are invisible from the corner cameras, but should not from in-between "cameras". Since our in-between views are created only from the corner cameras, then our in-between views will have gaps whenever the corner cameras have invisible points.

We have successfully re-implemented a head-mounted light field display using a distance-adjusted array of microlenses, and improved upon the calibration methods of the physical design of the HMD. We have focuses on pixel reprojection through shader programming and found that pixel reprojection can be used to lower the amount of cameras needed to render the 4D light field. There are several benefits of using a light field display over a traditional HMD; in particular the light field can avoid the vergence-accommodation conflict and can also correct for near- and farsightedness. Development of light field displays and efficient rendering of the light field is highly desired, and the technology is gaining

interest in several areas. There are though still limitations related to the technology. Our light field display with a resolution of 1280px×720px and the screen size 15.36mm×8.64mm (0.012 mm/px), we get a maximum spatial resolution of 121×64px and each subimage size under each lenslet is only $\approx 80\frac{1}{3}$ pixels/mm. In other words, the resolution is low and the quality is still not good enough to be usable for more than prototyping. Furthermore the frame rate achievable with pixel reprojection is too low to be usable.

6.2 Future Work

Future development would require higher resolution displays, but we can conclude that our pixel reprojection method is applicable on higher resolution images. With a pixel offset error of maximum 0.5 px, the pixel error percentage will only lower with higher resolution images.

We have shown that the pixel reprojection method creates acceptable images for light field renderings, but the method needs optimization before being applicable in real-time scenarios. The performance test showed that the framerate (≈ 5.35px) is far from usable, and needs to be drastically optimized before being useful.

6.2.1 Performance

Using pixel reprojection to render light fields is comparable in quality to the DS method. This can though not be said about the performance, if pixel reprojection is being used in a realistic user scenario improvements have to be made.

Currently pixel reprojection is too slow to be usable in a real time scenario. This project sought to test the possibility of using pixel reprojection to render light fields. Shaders were used in order to utilise the GPU, but the code is not optimised. If pixel reprojection is to be a feasible method to render light fields, then the issue of performance needs to be addressed.

When pixel reprojecting the color value of fragments is found by testing a line of pixels of the corner cameras. Currently these pixels are tested one by one. To decrease render time one could use a method to find the correct pixel without using brute force.

6.2.2 Occlusion and Gaps

Extreme occlusion is one of the shortcomings of our method, since the experiment showed that test participants notice when gaps are filled with a mean corner camera value due to no real camera information is available. This problem could be solved by using extra corner cameras eg. a fifth camera could be placed in the middle of area covered by the four corner camera, and this camera would in many

cases contain some of the unavailable information, that none of the four corner cameras can see. Rendering more cameras than four (but less than eg. 120), is easy to implement in this method, and four cameras is just an example of how few cameras we can be used to create fairly good results.

6.2.3 Stereoscopic Rendering

Stereoscopic rendering simulates the natural stereoscopy of the human eyes by rendering two images representing the view of each eye. For this project there are two possibilities, squeezing the two views in to a single image and let the HMD stretch the images out and display them on the correct display. The downside is that the resolution is effectively cut in half, considering how low the resolution is for each lenslet, another solution is needed. The alternative is the use quad buffering, where the left and right image is rendered in full resolution.

Adding stereo to the HMD is a relative simple task, but is crucial for making a usable HMD.

Appendix A

HMD Construction

The HMD was constructed using a plastic VR headset for mobile devices, the screens from a Sony HMZ-T2, a Fresnel Technologies 630 microlens array and custom designed 3D printed parts (see Figure A.1).

The Fresnel Technologies 630 microlens array is a rectangular conventional lens array that comes in an overall size of 152 × 152 mm. It has a thickness of 3.3 mm and has 100 lenslets per sq. cm, with a lenslet spacing of 1 mm.

Before laser cutting the microlens array in smaller pieces that could fit in front of the screens (app. 1 × 1.5 cm) the lenses where checked for optical defects (see Figure A.2).

The precise optical center of the lenses in the array was not known. We knew that the distance between the optical center of the lenses and the screen should be the optimal distance calculated (see Section 3.1), but this distance would be close to impossible to achieve. The distance should be $0 < d_l f$, where the lens to screen distance is d_l and the focal length of the lens is $f = 3.3$.

The problem was solved by 3D printing spacers of different sizes (see Figure A.3). The spacer that was just small enough to produce a sharp image was used, due to the fact that the calculated d_l was so close to the focal length of the lens (see Section 3.1). This however meant that d_l was close but not the calculated value $d_l = 3.2888$ mm.

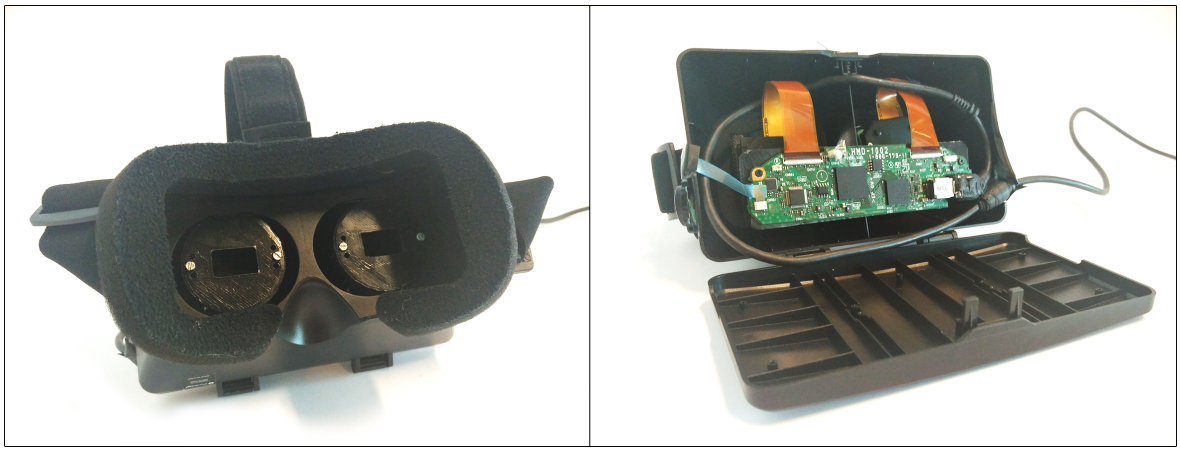

Figure A.1: The headset with the Sony HMZ-T2 screens, the Fresnel Technologies microlens array and custom designed 3D printed parts

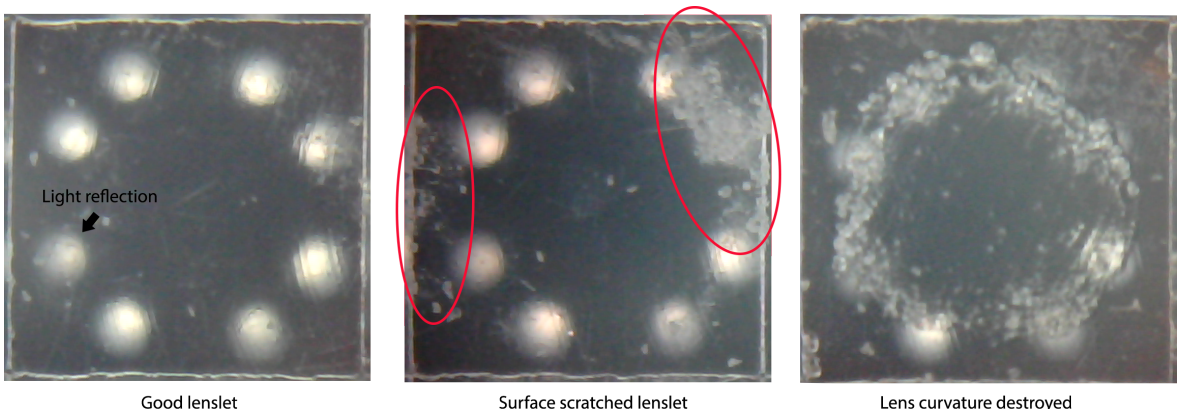

Figure A.2: The lenslet quality was checked before cutting out the pieces used in the HMD. From left we see a good lens, a lens with a few surface scratches, and a lens where the curvature has been destroyed.

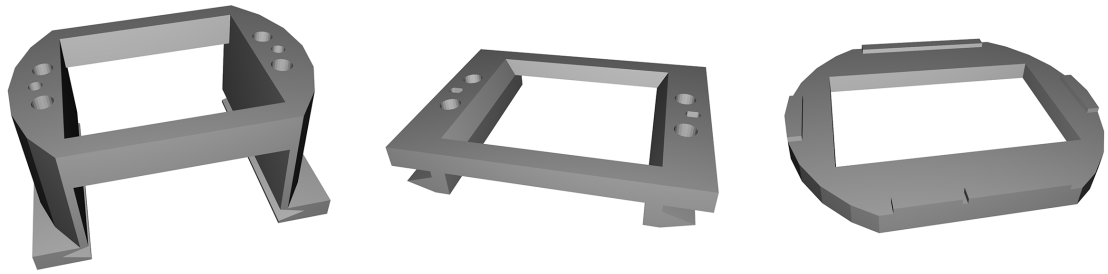

Figure A.3: 3D printed parts from left: Eye relief distance adjustment, vertical slider (to allow for vertical adjustment of the screen) and the lens separation (d_l) spacer. The lens separation spacer was printed several times and the best fit was chosen.

Appendix B

Lens Calibration

The lenslet array must be placed in front of the screen so that the lenses and the subimages align. This is especially true for the rotation since the position of the subimages can easily be adjusted in the shader. This is, however, not the case for rotation.

The alignment was achieved by placing a microscope directly above the screen. The image from the microscope was then sent to a Processing sketch that placed a (green) grid on top of the sub-images (see Figure B.1, top).

The image will appear distorted if the rotation alignment is off by several degrees (see Figure B.1, center). If the alignment, however, is close to correct the image will appear correct.

The lenslet array was placed on top and another grid (red) was shown that had the same position and rotation but was scaled to accommodate the larger squares (see Figure B.1, bottom). The lenslet array was held in place by sticky fix while fine tuning the position and rotation. Once the lenses aligned with the grid, they were glued in place with epoxy resin.

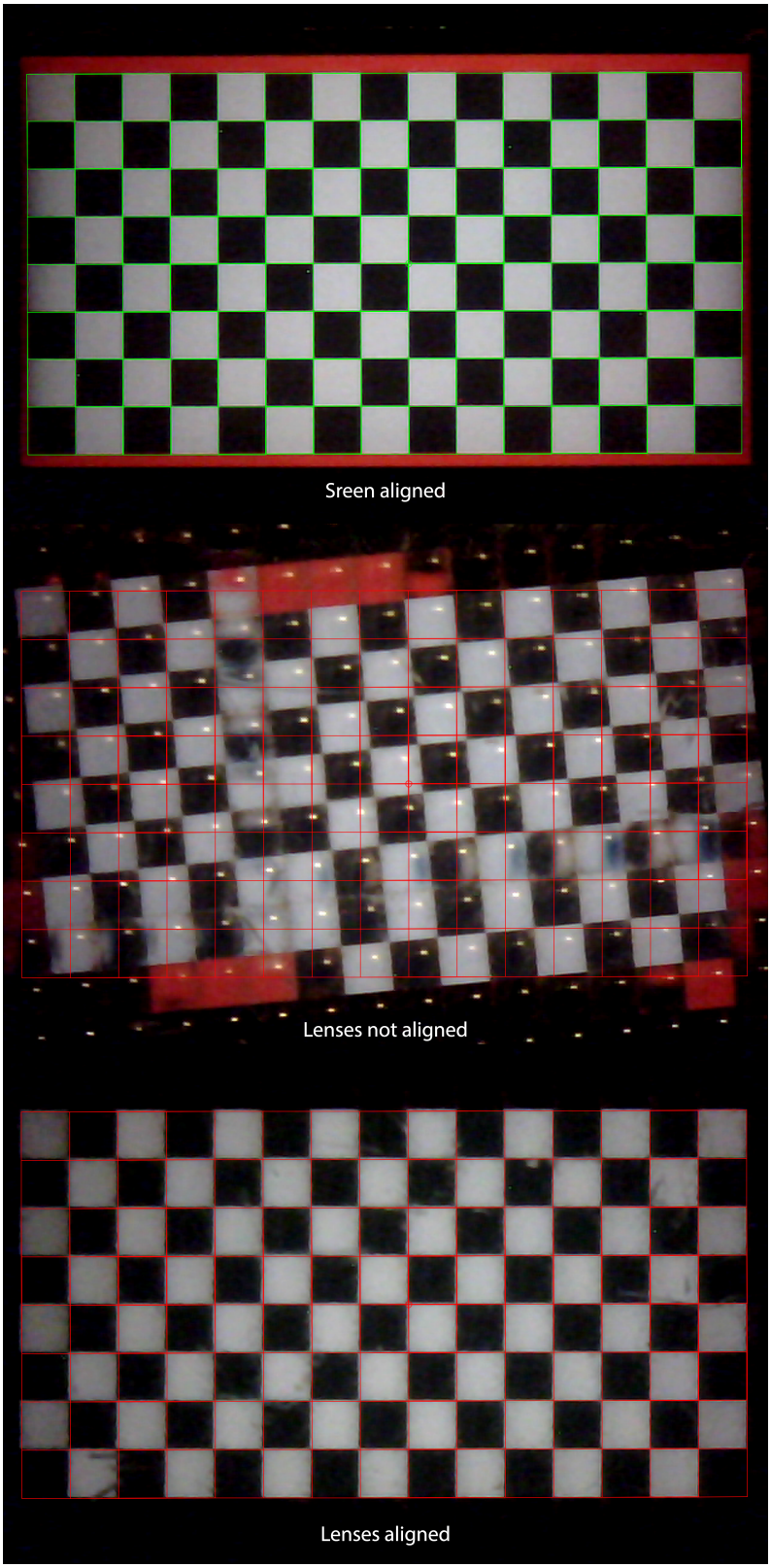

Figure B.1: Aligning the subimages with the lenslets. Top Image shows a grid superimposed over the subimages in the screen. Middle image shows lenses misaligned with the subimages. Bottom image shows the correct alignment of the subimages and lenses

Appendix C

Confirming Field of View

d_l was used to find the field of view of the images on the screen and, by extension, the field of view of the cameras in Unity Engine. A mismatch of these fields of view would result in erroneous perception of depth when looking at the light field. To combat this problem the field of view was measured with the chosen 3D printed spacer.

The image of the screen was changed so that one subimage was one colour (green) and the rest of the screen was another colour (red). The lens above the green subimage was surrounded by black tape to avoid confusion (see Figure C.2). The screen was then placed directly under a camera so that the lens in question was in the middle of the image coming from the camera. This alignment was achieved by sending the image from the camera to a processing sketch that has a small circle overlaid in the center of the image. The position of the screen was marked, at this point the image from the lens was completely green. If the screen was moved to the left or right (or up/down), at some point the image would change from green to red. Between the two extremes the edge between the two colours would appear. The point where this line was in the center of the lens was marked on the left and right side (see Figure C.1). If the distances between the edges and the center were not the same, then the alignment of the subimage relative to the lens was off. This was corrected until the distance was the same on both sides. Once this was achieved, the length was measured. With this length and the distance between the camera and the lenslet array the angle could be calculated. As expected the angle was close but not equal to the field of view calculated and the cameras in Unity Engine adjusted accordingly.

To confirm that the field of view measured was correct, a test was devised. Two ellipses were placed in the scene, one was in the center of the light field, the other was placed next to the first at an angle. If the ellipses parent object was scaled, the scale and position of the ellipses would change but they would appear to have the same position and scale in the light field but the depth would change (see Figure C.3). A camera was looking down at the screen with the same angle as the second ellipse relative to the center of the light field. If the ellipse appeared to be in the same place when the parent object was scaled, then the FOV was correct (see Figure C.4).

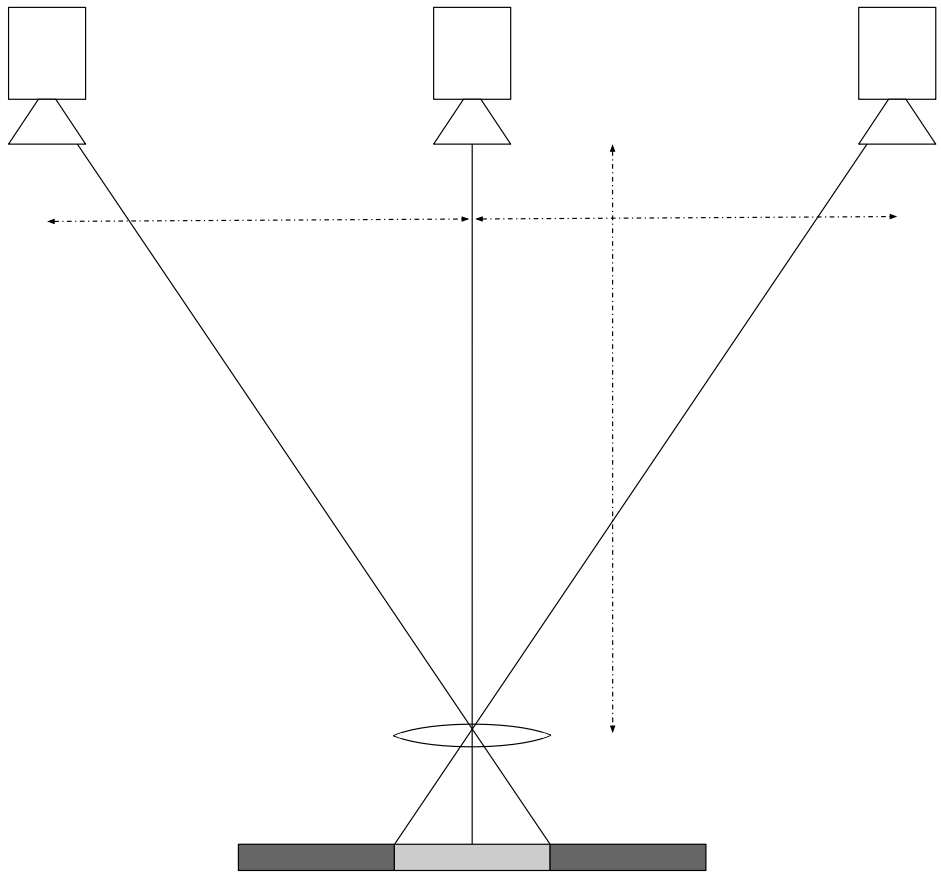

Figure C.1: The camera is moved to find the FOV of the subimage.

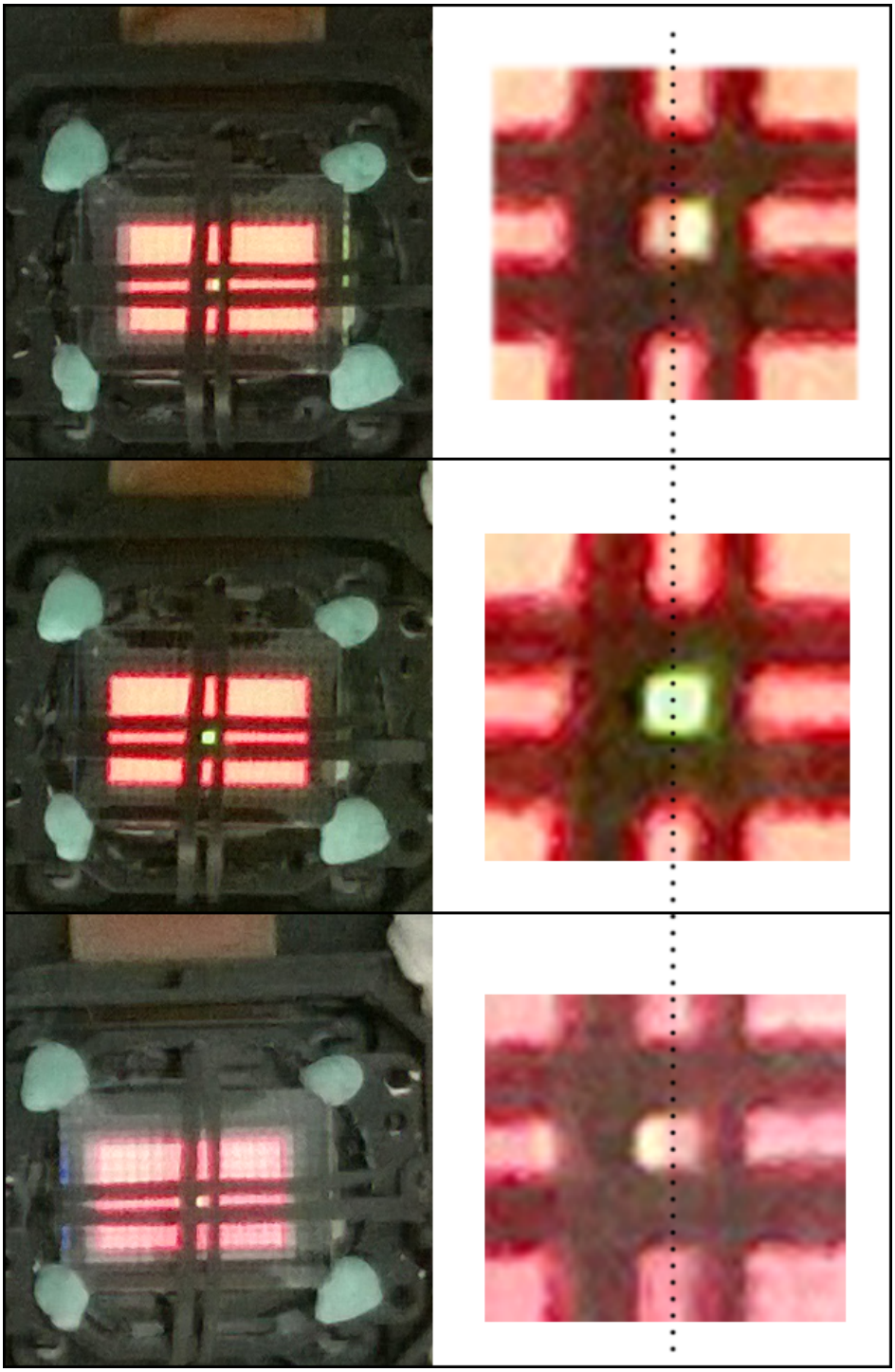

Figure C.2: Finding the FOV of the subimages.

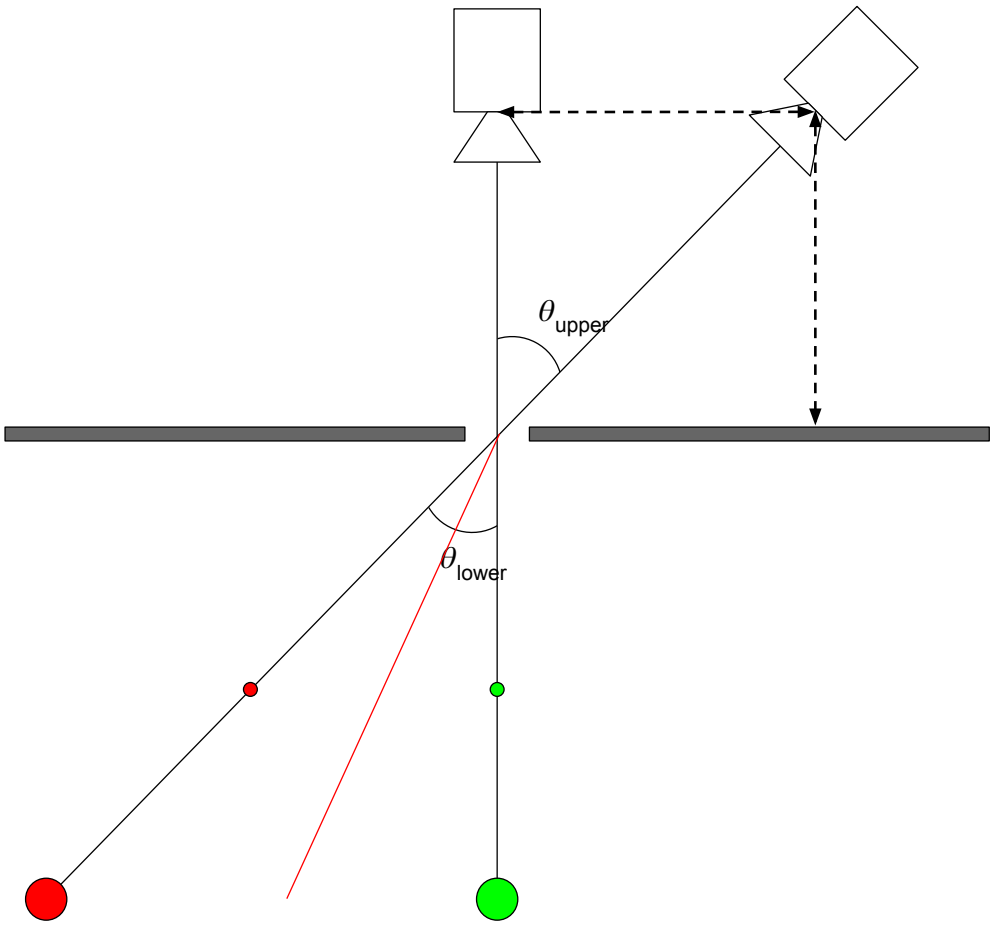

Figure C.3: Confirming that the FOV is correct; if Θ_{lower} and Θ_{upper} are the same, then the angled camera will see both red spheres. If the angles are not the same, it won't see both spheres (it will follow another line e.g. the red line).

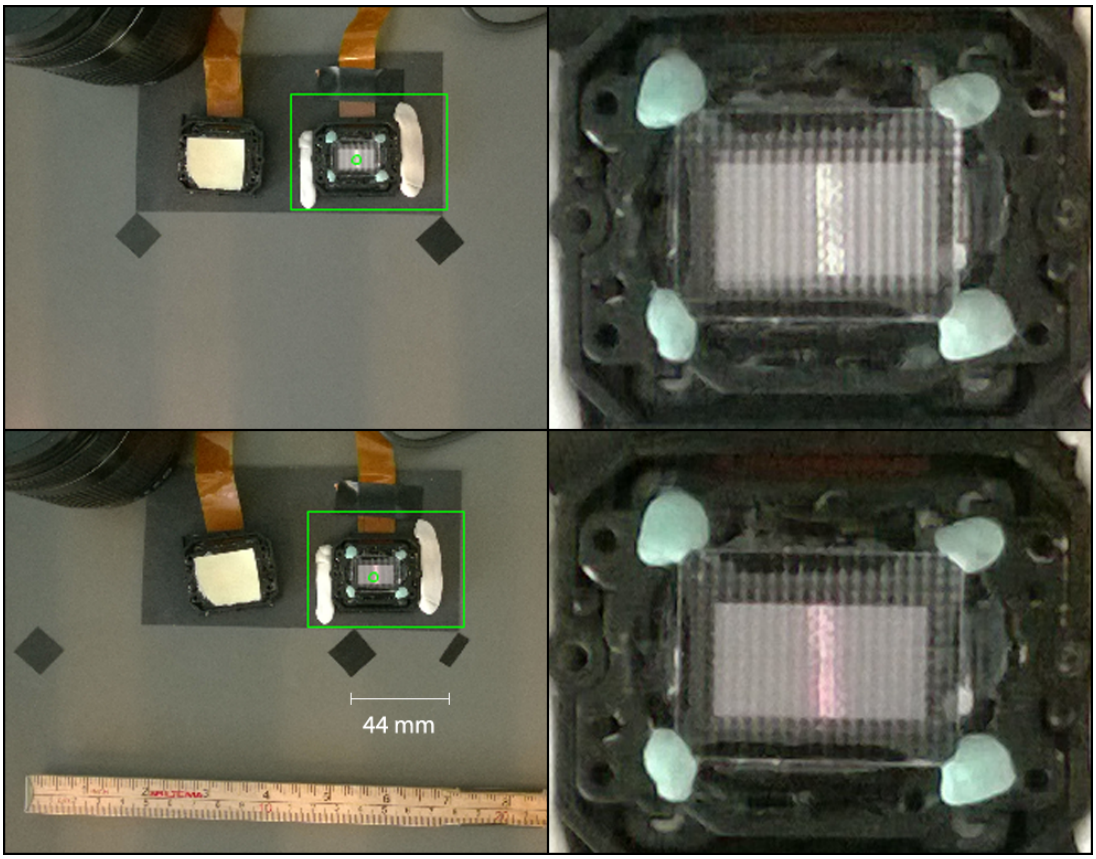

Figure C.4: Confirming the field of view of the light field; the camera should be able to see the thin green sphere, when the camera looks straight down, and the thin red sphere when the camera was angled with θ. When moving the spheres closer or further away in the virtual scene and scaling the spheres accordingly, the spheres should look consistent on the light field display.

Appendix D

Virtual Scaling of the Screen Size

When the software was at a stage where a light field could be produced, it was discovered that the image wasn't sharp. It was found that if the size of the subimages was scaled down the image became sharp. First thought was that the implemented scaling was off, but this was not the case (see Figure D.1). Another possibility is that the lenses are not the advertised $1mm \times 1mm$ [43] but slightly smaller (at least the small section that was used for the hmd).

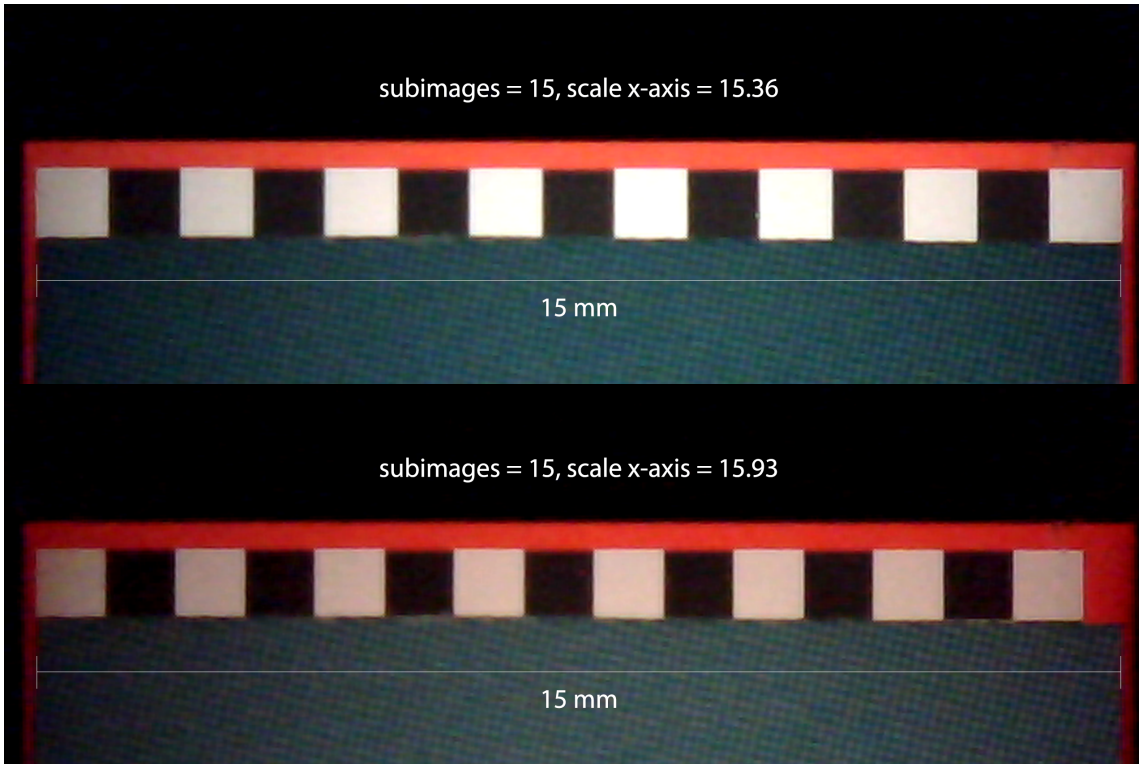

Figure D.1: Scaling of the screen size. Scale x-axis is the size of the screen. If the screen size is set to be larger than real life then the subimages will be smaller than $1mm \times 1mm$. The reason for this slightly backwards approach was the fact that the adjust was already implemented and could be used to solve this problem.

Appendix E

Initial Experiment

An experiment was done early in the project. The experiment took place at the Create building at AAU (Rendsburggade 14, Aalborg). The test was very similar to the test used in this project, but the problem with it was that the test participants were not given the option to look at the reference image after they had proceeded to view the other images. This meant that they had to remember the reference image with which many had trouble with. Another test was therefore devised where the test participants had the option to view the reference image at any time. Below are the results of the first test. Note that in this test when there were occlusion holes, they were filled with black, rather than an averaged colour from the corner cameras. Super sampling was not utilised in this test (see Figure E.2). The images used in this test had different textures than the ones used in the final test (see Figure E.1). The change was made to fit with Univeritariums theme which was space exploration.

There were 40 test participants (8 female, 32 male), age ranging from 21 to 28 years, samples were independent. With 40 test participant the critical number $i_c = 13$. The 2-IFC tasks were:

1. The reference image is shown.
2. Two visual stimuli are presented in random order (visual stimuli can be revisited as many times as the test participant desires and the delay interval ISI is properly implemented).
3. The test participant chooses one of the two visual stimuli.

Image no.	$\mathbf{i_{lfdisplay}}$	**LF Display**	$\mathbf{i_{monitor}}$	**Monitor**
Image 1	25	PASSED	13	PASSED
Image 2	20	PASSED	17	PASSED
Image 3	15	PASSED	17	PASSED
Image 4	18	PASSED	11	FAILED
Image 5	8	FAILED	2	FAILED

Since reference image can not be revisited, this experiment might test memory instead of similarity between VC- and IC-images.

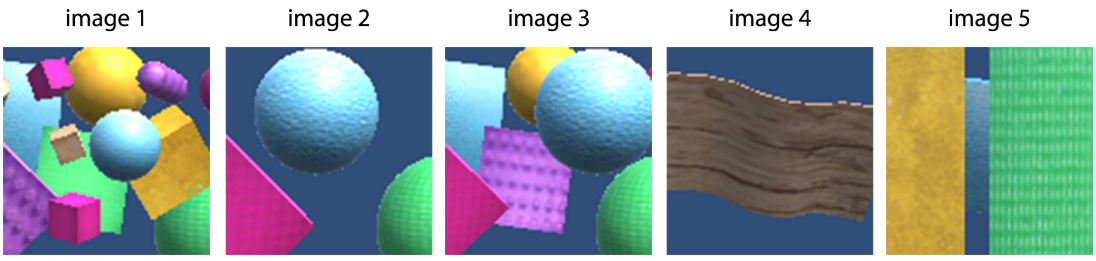

Figure E.1: Image samples from the center virtual camera of the five different scenes (position $[4, 8]$ out of $[8, 15]$)

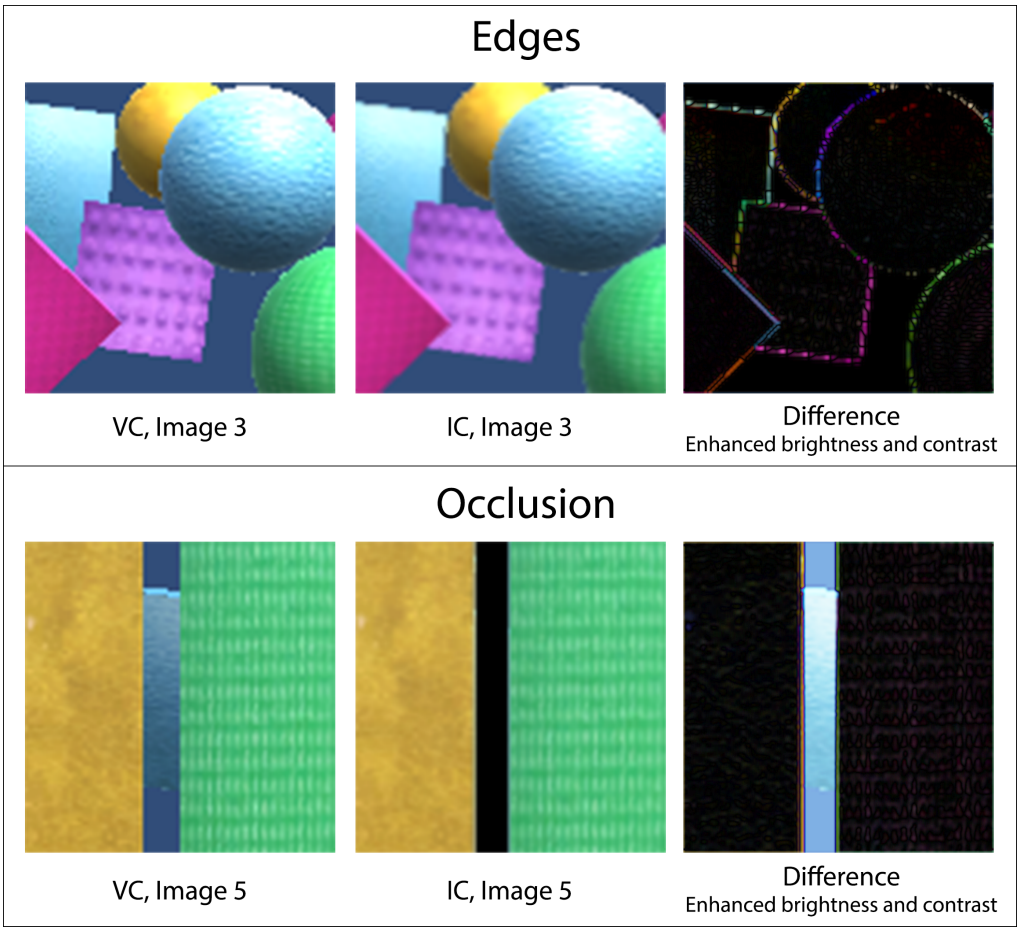

Figure E.2: Example of the short comings of our pixel reprojection method; edges are off by 1 pixel and occlusion happens at an extremely close distance (12 cm). NOTE: Pictures of *Difference* have enhanced brightness and contrast

Appendix F

Image Difference

We have evaluated the pixel colour difference by subtracting IC from VC and taking the absolute value. A complete pixel match will be shown black, and the difference can be seen for the RGB-values (see Figure F.1). The compared images have the same resolution as rendered for the light field display (1215 × 648px).

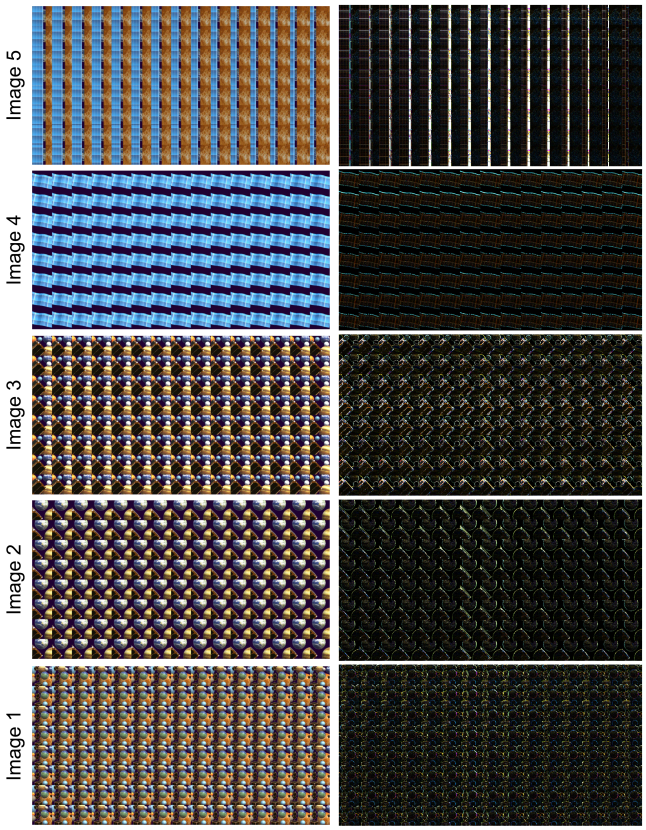

Figure F.1: Left: VC light field. Right: The image difference (the absolute value of subtracting IC from VC). OBS: Brightness and contrast have been enhanced for printing.

Appendix G

One Step Interpolation

In this project a "two step" approach was used when interpolating between the corner cameras, where the first step was to interpolate the top and bottom row (the x-axis). After this the pixels in between the two rows were interpolated (the y-axis). These two steps can be combined to one step. The interpolation works by starting from its current subimage position. This position is read from the corner camera, the pixel is reprojected. If, after the reprojection, the position is within 0.5 pixels, the correct pixel is found. Then the next step can begin (sub pixel offset) if not then the next pixel is tested. In the "two step" approach this pixel would be offset by one on either the x- or y-axis (see Figure G.1). In the one step approach a direction has to be calculated. This is done by using the subimage index on the x- and y-axis. From this and the known position of the corner cameras the slopes between the cameras and the subimage of interest are found.

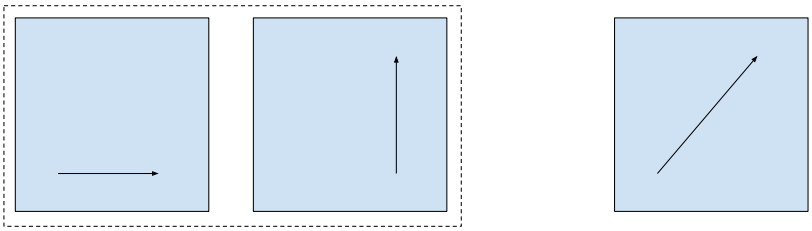

Figure G.1: Left: Direction of pixel lookup in the two step approach. Right: The two steps combined to one step.

When using this method it is important to consider the slope when moving to the next pixel; if the slope is small one should move in the x-axis, if the slope is larger than 1 the y-axis should be used (see Figure G.2). If the wrong axis was used then each increment would be more than one pixel and would in many cases result in skipping the correct pixel.

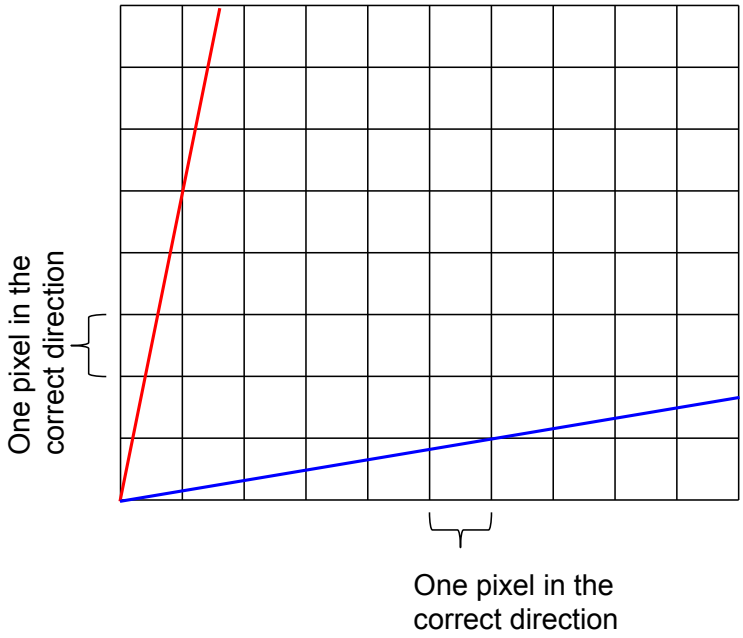

Figure G.2: Difference in small and large slope. One had to increment with the correct axis.

Bibliography

[1] David M Hoffman, Ahna R Girshick, Kurt Akeley, and Martin S Banks. Vergence–accommodation conflicts hinder visual performance and cause visual fatigue. *Journal of vision*, 8(3):33, 2008.

[2] W Neil Charman. The eye in focus: accommodation and presbyopia. *Clinical and experimental optometry*, 91(3):207–225, 2008.

[3] Rajaraman Suryakumar, Jason P Meyers, Elizabeth L Irving, and William R Bobier. Vergence accommodation and monocular closed loop blur accommodation have similar dynamic characteristics. *Vision research*, 47(3):327–337, 2007.

[4] Clifton M Schor, Jack Alexander, Lawrence Cormack, and Scott Stevenson. Negative feedback control model of proximal convergence and accommodation. *Ophthalmic and Physiological Optics*, 12(3):307–318, 1992.

[5] Takashi Shibata, Joohwan Kim, David M Hoffman, and Martin S Banks. Visual discomfort with stereo displays: Effects of viewing distance and direction of vergence-accommodation conflict. In *IS&T/SPIE Electronic Imaging*, pages 78630P–78630P. International Society for Optics and Photonics, 2011.

[6] Gregory Kramida and Amitabh Varshney. Resolving the vergence-accommodation conflict in head mounted displays.

[7] Douglas Lanman and David Luebke. Near-eye light field displays. *ACM Transactions on Graphics (TOG)*, 32(6):220, 2013.

[8] Edward H Adelson and James R Bergen. The plenoptic function and the elements of early vision. *Computational models of visual processing*, 1(2): 3–20, 1991.

[9] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42. ACM, 1996.

[10] Gordon Wetzstein, Douglas Lanman, Matthew Hirsch, and Ramesh Raskar. Tensor displays: compressive light field synthesis using multilayer displays with directional backlighting. *ACM Trans. Graph.*, 31(4):80, 2012.

[11] Matthew Hirsch, Gordon Wetzstein, and Ramesh Raskar. A compressive light field projection system. *ACM Transactions on Graphics (TOG)*, 33(4): 58, 2014.

[12] Andrew Jones, Ian McDowall, Hideshi Yamada, Mark Bolas, and Paul Debevec. Rendering for an interactive 360 light field display. *ACM Transactions on Graphics (TOG)*, 26(3):40, 2007.

[13] Fu-Chung Huang, Kevin Chen, and Gordon Wetzstein. The light field stereoscope: immersive computer graphics via factored near-eye light field displays with focus cues. *ACM Transactions on Graphics (TOG)*, 34(4):60, 2015.

[14] JP Rolland and Hong Hua. Head-mounted display systems. *Encyclopedia of optical engineering*, pages 1–13, 2005.

[15] Douglas Lanman and David Luebke. Supplementary material: Near-eye light field displays. *ACM Transactions on Graphics (TOG)*, 32(6):220, 2013.

[16] Vesselin Shaoulov, Ricardo Martins, and Jannick P Rolland. Compact microlenslet-array-based magnifier. *Optics Letters*, 29(7):709–711, 2004.

[17] Jon Aschberg, Jákup Klein, and Anne Juhler Hansen. Experimental evaluation of the perceived accommodation range of a near-eye light-field display. 2015.

[18] Abe Davis, Marc Levoy, and Fredo Durand. Unstructured light fields. In *Computer Graphics Forum*, volume 31, pages 305–314. Wiley Online Library, 2012.

[19] Young Ju Jeong, Hyun Sung Chang, Yang Ho Cho, Dongkyung Nam, and C-C Jay Kuo. Efficient direct light field rendering for autostereoscopic 3d displays.

[20] Kushagr Gupta, Suleman Kazi, and Terry Kong. Dynamic eye gaze tracking for foveated rendering and retinal blur.

[21] Brian Guenter, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. Foveated 3d graphics. *ACM Transactions on Graphics (TOG)*, 31(6):164, 2012.

[22] Bennett S Wilburn, Michal Smulski, Hsiao-Heng K Lee, and Mark A Horowitz. Light field video camera. In *Electronic Imaging 2002*, pages 29–36. International Society for Optics and Photonics, 2001.

[23] Hartmut Schirmacher, Li Ming, and Hans-Peter Seidel. On-the-fly processing of generalized lumigraphs. In *Computer Graphics Forum*, volume 20, pages 165–174. Wiley Online Library, 2001.

[24] Cha Zhang and Tsuhan Chen. Light field capturing with lensless cameras. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III–792. IEEE, 2005.

[25] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. Light field photography with a hand-held plenoptic camera. *Computer Science Technical Report CSTR*, 2(11), 2005.

[26] Todor Georgiev, Ke Colin Zheng, Brian Curless, David Salesin, Shree Nayar, and Chintan Intwala. Spatio-angular resolution tradeoffs in integral photography. *Rendering Techniques*, 2006:263–272, 2006.

[27] Michael Naimark, John Woodfill, Paul Debevec, and Leo Villareal. Immersion'94. *Interval Research Corporation image-based modeling and rendering project from Summer*, 1994.

[28] Jon Karafin. Light field imaging: The future of vr-ar-mr - part 4, 11 2015. URL `https://www.youtube.com/watch?v=_PVok9nUxME`. Presented by the VES Vision Commitee. Presentation by Jon Karafin, Head of Light Field Video for Lytro, followed by a QA with all presenters moderated by Scott Squires, VES. [Accessed June 1, 2016].

[29] https://www.lytro.com/press/releases/lytro-immerge-the-worlds-first-professional-light-field-solution-for-cinematic-vr. Accessed June 1, 2016.

[30] Sing Bing Kang. Geometrically valid pixel reprojection methods for novel view synthesis. *ISPRS journal of photogrammetry and remote sensing*, 53 (6):342–353, 1998.

[31] Pitchaya Sitthi-amorn, Jason Lawrence, Lei Yang, Pedro V Sander, Diego Nehab, and Jiahe Xi. Automated reprojection-based pixel shader optimization. *ACM Transactions on Graphics (TOG)*, 27(5):127, 2008.

[32] Diego Nehab, Pedro V Sander, Jason Lawrence, Natalya Tatarchuk, and John R Isidoro. Accelerating real-time shading with reverse reprojection caching. In *Graphics hardware*, volume 41, pages 61–62, 2007.

[33] Vlastimil Havran, Cyrille Damez, Karol Myszkowski, and Hans-Peter Seidel. An efficient spatio-temporal architecture for animation rendering. In *ACM SIGGRAPH 2003 Sketches & Applications*, pages 1–1. ACM, 2003.

[34] Stephen J Adelson and Larry F Hodges. Generating exact ray-traced animation frames by reprojection. *IEEE Computer Graphics and Applications*, 15(3):43–52, 1995.

[35] Takehiro Tawara, Karol Myszkowski, and H-P Seidel. Exploiting temporal coherence in final gathering for dynamic scenes. In *Computer Graphics International, 2004. Proceedings*, pages 110–119. IEEE, 2004.

[36] http://beta.unity3d.com/talks/Siggraph2011_SpecialEffectsWithDepth_WithNotes.pdf. Accessed June 1, 2016.

[37] http://docs.unity3d.com/Manual/WritingImageEffects.html. Accessed June 1, 2016.

[38] http://docs.unity3d.com/Manual/class-RenderTexture.html. Accessed June 1, 2016.

[39] Martin Kraus. Quasi-convolution pyramidal blurring. *Journal of Virtual Reality and Broadcasting*, 6(6), 2009.

[40] Douglas W Cunningham and Christian Wallraven. *Experimental design: From user studies to psychophysics*. CRC Press, 2011.

[41] Suzanne P McKee, Stanley A Klein, and Davida Y Teller. Statistical properties of forced-choice psychometric functions: Implications of probit analysis. *Perception & Psychophysics*, 37(4):286–298, 1985.

[42] Martin Borg, Stine Schmieg Johansen, Dennis Lundgaard Thomsen, and Martin Kraus. Practical implementation of a graphics turing test. In *International Symposium on Visual Computing*, pages 305–313. Springer, 2012.

[43] *HIGH QUALITY FRESNEL LENSES IN A VARIETY OF SIZES FOCAL LENGTHS*. Fresnel Technologies, 2014.