Tampere University

Kamran Akbar

DEPTH OF FIELD GUIDED VISUALISATION ON LIGHT FIELD DISPLAYS

Master of Science in Technology Thesis
Faculty of Information Technology and Communications
Examiner: Professor Atanas Gotchev
Assistant Professor Robert Bregovic
December 2022

ABSTRACT

Kamran Akbar: Depth of Field Guided Visualisation on Light Field Displays
Master of Science in Technology Thesis
Tampere University
Master's Programme in Computing Science
Major: Human-Technology Interaction (M.Sc. Tech)
December 2022

Light field displays are capable of realistic visualization of arbitrary 3D content. However, due to the finite number of light rays reproduced by the display, its bandwidth is limited in terms of angular and spatial resolution. Consequently, 3D content that falls outside of that bandwidth will cause aliasing during visualization. Therefore, a light field to be visualized must be properly preprocessed. In this thesis, we propose three methods that properly filter the parts in the input light field that would cause aliasing. First method is based on a 2D FIR circular filter that is applied over the 4D light field. Second method utilizes the structured nature of the epipolar plane images representing the light field. Third method adopts real-time multi-layer depth-of-field rendering using tiled splatting. We also establish a connection between lens parameters in the proposed depth-of-field rendering and the display's bandwidth in order to determine the optimal blurring amount. As we prepare light field for light field displays, a stage is added to the proposed real-time rendering pipeline that simultaneously renders adjacent views. The rendering performance of the proposed methods is demonstrated on Holografika's Holovizio 722RC projection-based light field display.

Keywords: light field, light field display, angular resolution, spatial resolution, display's bandwidth, aliasing, 2D circular filter, epipolar plane images, depth of field, multi-layer depth of field rendering, tiled splatting, view synthesis, projection-based light field display.

PREFACE

This thesis was done in 3D media group at the Information Technology and Communication Faculty, Tampere University. I like to appreciate the assistance and guidance of my thesis supervisors, Assistant Prof. Robert Bregovic and Prof. Atanas Gotchev throughout my thesis. They helped me a lot in comprehending the theories related to my thesis and brainstorming methods for answering respective research questions. Also, I want to express my gratitude to my friends and colleagues for their tips and constructive criticism, and my parents who supported me throughout my life.

Tampere, 9th December 2022

Kamran Akbar

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

1D	One dimensional
2D	Two dimensional
3D	Three dimensional
4D	Four dimensional
5D	Five dimensional
7D	Seven dimensional
CoC	Circle of Confusion
CPU	Central Processing Units
DoF	Depth of Field
DSLF	Densely Sampled Light Field
EPI	Epi-polar Plane Images
FIR	Finite Impulse Response
FoV	Field of View
FP	Full Parallax
GPU	Graphics Processing Units
HPO	Horizontal Parallax Only
MSAA	MultiSample Anti-Aliasing
SMVD	Super Multi-view Display

1. INTRODUCTION

In the past few decades, media has progressed substantially, and much research has been done to improve visualization and bring more realistic experiences to humans' lives. To visualize media content in 3D, various displays have been developed. In some displays, glasses and headsets are used for 3D content visualization; however, other displays achieve the same goal without using glasses. In this thesis, we focus on glasses-free displays.

One way to visualize 3D content without glasses is to utilize the concept of light field. Light field [1] carries visual information about the real world, and it includes light rays with different directions and wavelengths, passing through every 3D point in distinct time instances [2]. A large number of light rays is needed to achieve realism of 3D content during visualization. However, producing a display that would generate the required number of rays is nowadays technically infeasible. Therefore, compromises must be made to make the production of light field displays possible. However, display simplifications will cause limitations on the quality of content shown on light field displays. In this thesis, we investigate issues caused by the Depth of Field (DoF) limitation of light field displays, and propose methods to solve these issues.

1.1 Problem Definition

In photography, DoF corresponds to the region in an image that is sharp and everything out of this region is blurred [3]. The main reason for having DoF in an image is the physical nature of the lens. In light field displays, DoF refers to a depth range around the display's screen, where virtual content is visualized with sufficient quality. The depth range depends on display's spatial and angular resolution [4], [5], [6], [7].

There is a direct relation between the smallest feature size reproducible by light field displays and its distance to the display's screen, namely the feature size increases proportionally to the distance from the display [4], [7]. Therefore, visualization of any virtual content smaller than the minimum feature size at a specific distance from the screen will cause aliasing. As illustrated in Figure 1.1, aliasing causes visual artifafacts such as double edges and color bleeding (see specifically the effects within the white-lined boxes). Consequently, aliased features need to taken care of. One way of handling aliasing is by DoF-guided rendering. More specifically, virtual points located within the display's DoF are rendered sharp, while the rest is blurred proportionally to the distance from the screen. This would effectively remove features smaller than the minimum feature size in a distance-related manner.

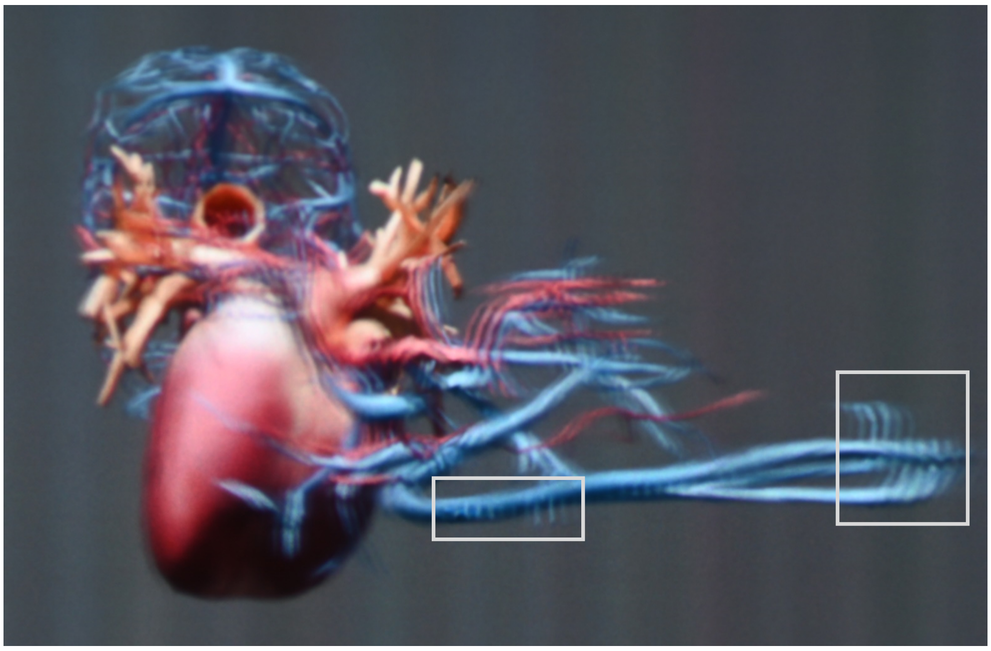


Figure 1.1. *Aliasing issue in light field displays.*

There are different approaches for DoF rendering. Path tracing [8] creates DoF effect by casting rays to the 3D scene through a thin lens model. Multi-view algorithms [9], [10], [11], [5] use several adjacent views for creating DoF in rendered images. Gathering [12], [13] and scattering [14] methods can be done in real-time. They are post-processing methods, which take the rendered image and the corresponding depth map as inputs and give an image with the desired DoF as an output. However, these methods can cause color bleeding and depth discontinuity artifacts due to the lack of information in occluded areas [15]. An alternative approach for DoF rendering is multi-layering [15], [16], [17], which divides the scene into discrete depth layers and stores geometrical information of the scene into layered images. The multi-layering technique is not a post-processing method anymore but it can be combined with scattering and gathering approaches.

In this work, for demonstration purpose, light field content is shown on Holografika's Holovizio 722RC projection-based light field displays [4], and its quantified spatial and angular resolution is adopted from [18].

1.2 Scope of the Thesis

This thesis addresses the aliasing issue in LF displays in three ways.

First, we make use of the structure of epi-polar plane images (EPI), which is inherent for rectified multiperspective views representing light fields [19]. This structure facilitates designing and applying 1D FIR filters on properly sheared EPIs from a Full Parallax (FP) Densely Sampled Light Field (DSLF) [20]. The FIR filters must be designed according to the display's angular and spatial resolution [21], [5].

Second, we aim at designing a 2D circular uniform filter to be applied directly on the FP DSLF [9], [10]. Both methods employ low-pass filters which blur features that the display cannot visualize faithfully.

The two aforementioned methods eliminate aliasing artifacts for the price of high computational cost as they work on an FP DSLF. In practice, one would be interested in a method that solves the aliasing issue in real-time. To this end, real-time DoF rendering algorithms seem good candidates for aliasing removal. The DoF rendering simulates the physical nature of the lens, and it is analogous to DoF in the light field displays [22].

Therefore, in a third approach, we study and implement the multi-layer DoF rendering with tiled splatting, as a real-time method for artifact removal [15]. We select it as its DoF rendering quality is better than the other state-of-the-art real-time DoF rendering algorithms. The method is to be further enhanced by adding Multisample Antialiasing (MSAA) for further improvement of its performance. In addition, the connection between thin lens parameters used in this method and light field display's bandwidth is to be evaluated. By utilizing this connection, a method is to be proposed to simultaneously synthesize several adjacent views with same DoF to facilitate light field rendering.

1.3 Thesis Organization

This thesis is organized as follows. In Chapter 2, we formally define light field and light field displays including different methods to parameterize the light field. We introduce the concept of DSLF, discuss projection-based light field displays, and ray propagation in those displays. In Chapter 3, we present the concept of DoF in photography, thin lens model, and light field displays DoF. Also, we look for real-time DoF rendering methods for solving aliasing artifacts in light field displays. In Chapter 4, we propose offline methods to solve the aliasing issues in light field displays by applying depth-based anti-aliasing filters on DSLF. In Chapter 5, we investigate a specific real-time DoF rendering method and describe the connection between thin lens parameters in DoF rendering and light field displays' bandwidth. Furthermore, we describe the proposed method for simultaneous synthesis of several adjacent views and discuss its limitations. In Chapter 6, we present the conclusions and state future research directions.

2. LIGHT FIELD

In this chapter, the concept of the plenoptic function and different ways for parametrizing the light field are discussed. Also, the concept of epi-polar geometry, epi-polar plane images, and DSLF are discussed. Then, light field capturing systems, projection-based light field displays, and the theory about their sampling patterns at different planes are explained.

2.1 Plenoptic Function

The concept of light field was introduced by Gershun as the plenoptic function that measures light at a particular position and direction [23]. The plenoptic function [2] describes the light's physical nature in terms of rays' intensity distribution, and is a 7D function parameterized as light rays crossing 3D points in space (x, y, z) with various directions (θ, ϕ) and wavelengths λ at a given time t as illustrated in Figure 2.1.

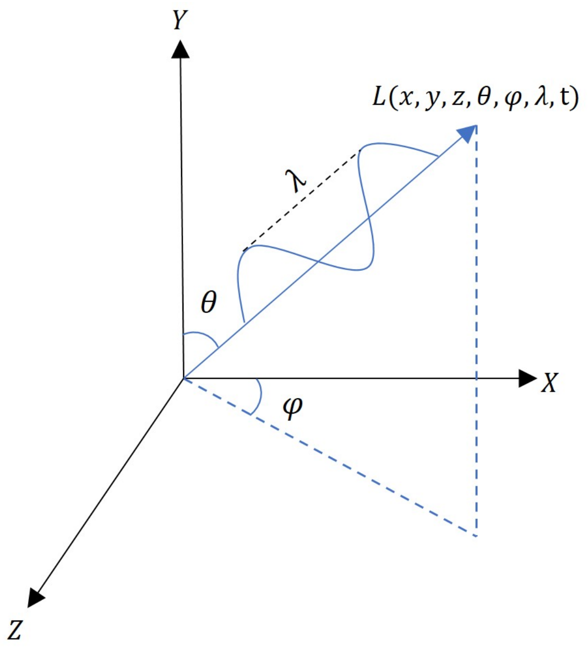


Figure 2.1. *Light ray indicating a plenoptic function's sample at a given time t passing through (x, y, z) with wavelength λ and direction (θ, ϕ) .*

2.2 Light Field Parametrization

The 7D plenoptic function is complex; however, it can be reduced to 5D by assuming a stationary scene with monochromatic illumination. Moreover, light's radiance stays constant while traveling from one point to another until it hits an opaque object or passes through another medium with a different index of refraction [1], [24]. Therefore, the plenoptic function's dimensionality can be further reduced to 4D. There are multiple ways to represent a 4D light field, e.g. pair of points and directions on a surface (space-angle parametrization) or pair of points on two parallel planes (two-plane parametrization) [1], [24].

The two-plane parametrization is denoted as $L(s, t, u, v)$, in which (s, t) and (u, v) are coordinates on each plane that the corresponding light ray travels through. In the two-plane parametrization, plane (s, t) corresponds to various viewpoints and plane (u, v) contains images of each viewpoint.

The space-angle parametrization is another way to represent the light field. This representation considers points on a plane or a curved surface with an arbitrary direction. The $L(s, t, \theta, \phi)$ notation corresponds to space-angle representation, where the light ray is originated from (s, t) with (θ, ϕ) direction. Figure 2.2 visualizes the two mentioned parametrizations.

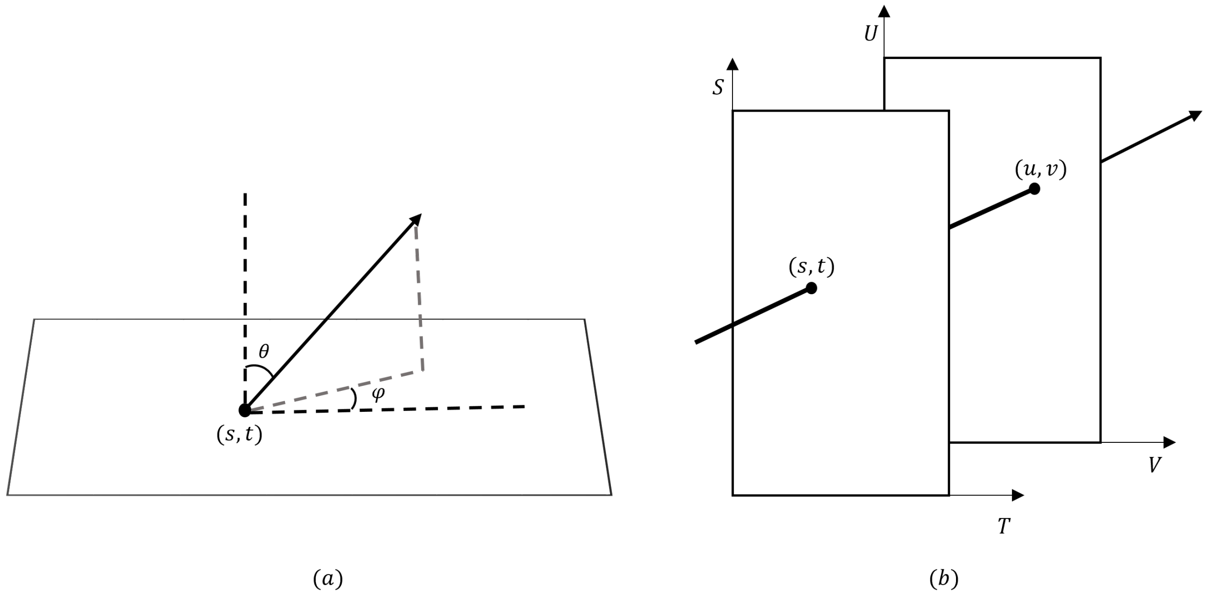


Figure 2.2. *Two different light field parametrization: (a) space-angle parametrization, (b) two-plane parametrization.*

2.3 Light Ray Propagation

The light field $L(s, t, u, v)$ can be reduced to $L(t, u, v) = L(s_0, t, u, v)$ referred as Horizontal Parallax Only (HPO) light field. To understand the mentioned assumption easily, a 4D light field is reduced to 2D, $L(t, v)$ for constant s and u, and $L(t, \theta)$ for constant s and ϕ. The relation between two-plane parametrizations and space angle parametrization is $\begin{bmatrix} t \\ v \end{bmatrix} = \begin{bmatrix} t \\ l\tan(\theta) \end{bmatrix}$ where l is a distance between the two parallel planes. The ray propagation at distance d is equivalent to denoting two light fields, as illustrated in Figure 2.3. According to the two-plane and space-angle light field parametrizations, the ray propagation can be formulated as [25] [26]

$$L_2\left(\begin{bmatrix} t_2 \\ v_2 \end{bmatrix}\right) = L_1\left(\begin{bmatrix} t_1 \\ v_1 \end{bmatrix}\right) = L_1\left(\begin{bmatrix} t_2 - \frac{d}{l}v_2 \\ v_2 \end{bmatrix}\right) \tag{2.1}$$

$$L_2\left(\begin{bmatrix} t_2 \\ \theta_2 \end{bmatrix}\right) = L_1\left(\begin{bmatrix} t_1 \\ \theta_1 \end{bmatrix}\right) = L_1\left(\begin{bmatrix} t_2 - d\tan(\theta_2)) \\ \theta_2 \end{bmatrix}\right), \tag{2.2}$$

where L_1 and L_2 refer to the first and second light field, respectively. Moreover, v_1 and v_2 stay identical after the ray propagation, but t_2 is shifted regarding the propagation distance d and the value of v_1.

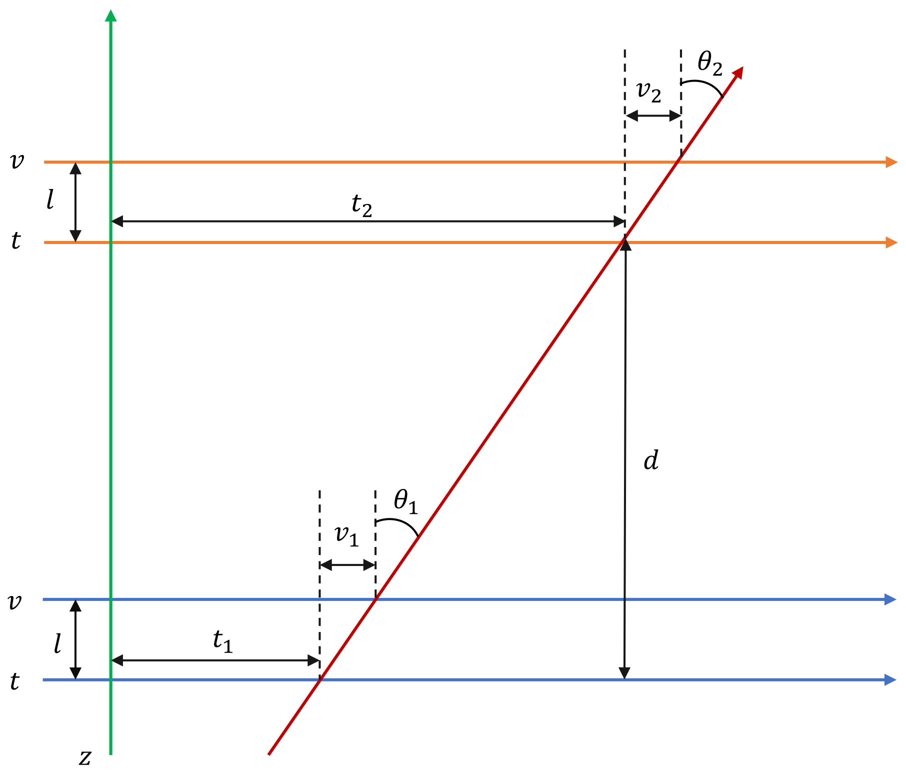


Figure 2.3. *Ray propagation at distance d.*

2.4 Epipolar Geometry

In multiple camera views, there are geometrical relations between a 3D point and its projection on each camera's image plane. The geometry that connects camera views, 3D points, and their projections on the image plane is called epi-polar geometry [27]. As depicted in the Figure 2.4, there are two pinhole cameras placed in O_1 and O_2, and the line between these two points is denoted as b, which passes cameras' image plane through e_1 and e_2 called as epi-poles. Any plane like π crossing line b intersects with the cameras' image plane in a line, which is termed as an epi-polar line. Epi-polar lines in the Figure 2.4 are I and I'. Also, every point in the π plane is mapped to the corresponding epi-polar line in each view [27]. As a result, for each 3D point P, there is a pair of points, (P_L, P_R) residing on I and I' respectively, which are equivalent in different camera viewpoints.

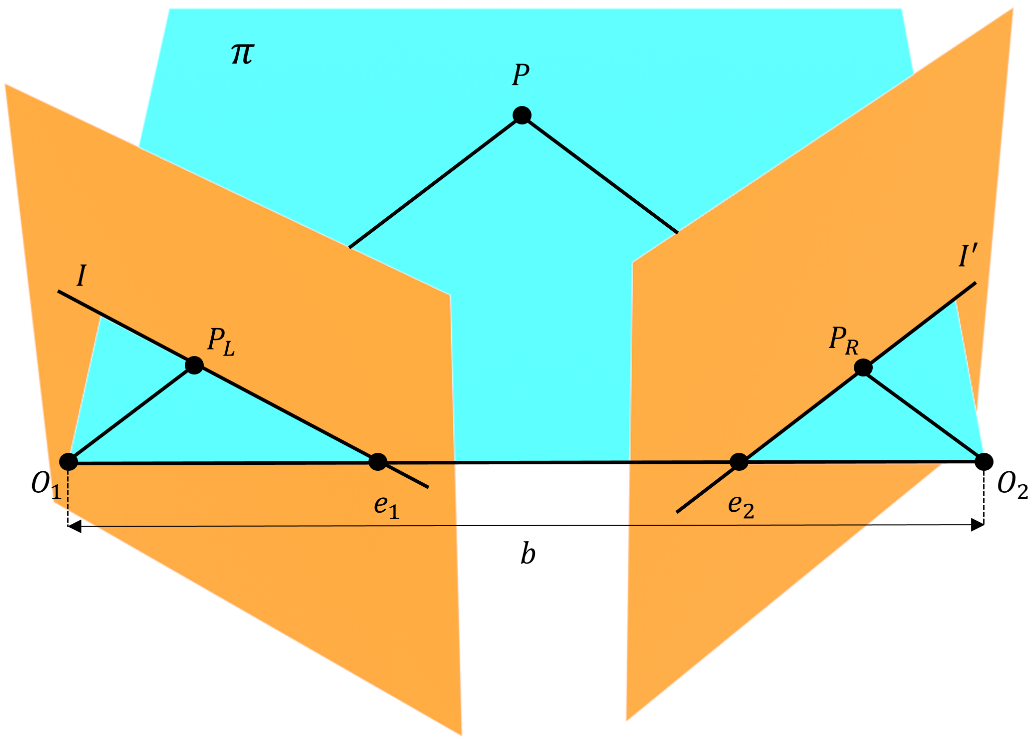


Figure 2.4. *The epi-polar geometry for two camera views.*

Camera rectification is the action of making relational rotation between camera viewpoints identity, which causes epi-polar lines be parallel to the baseline between camera viewpoints as illustrated in Figure 2.5. The distance between the projection of a 3D point P on the left rectified image plane P'_L and the right rectified image plane P'_R is denoted as disparity d. The disparity can be computed based on the camera's focal length f, the 3D point's depth z_P, and the baseline between two cameras b [27].

$$d = \frac{f \times b}{z}. \tag{2.3}$$

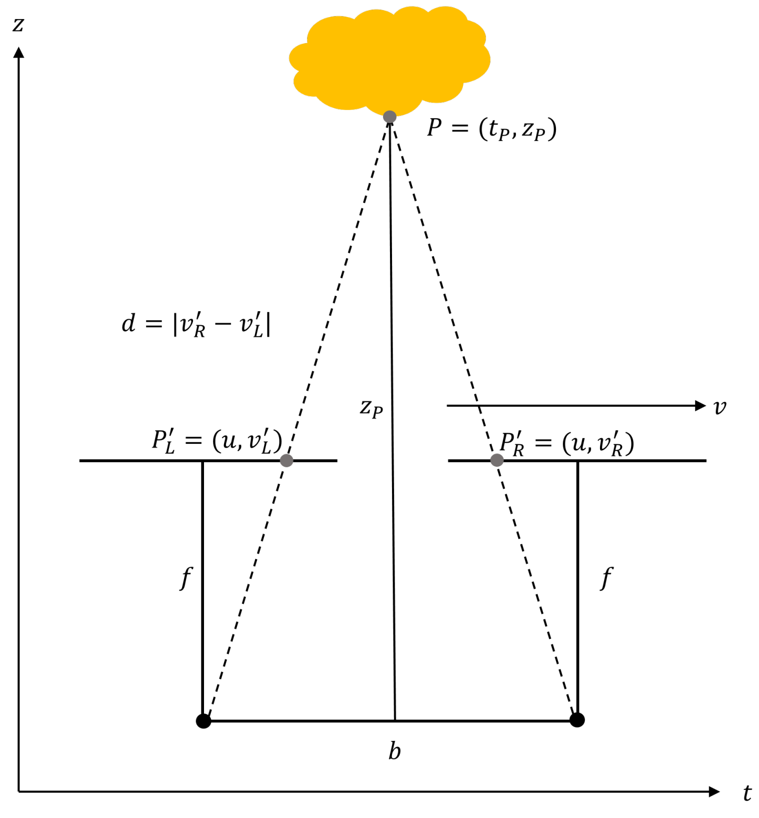


Figure 2.5. *The epi-polar geometry for two parallel camera views.*

2.5 Epipolar Plane Images

As discussed in section 2.2, light field can be simplified from a 7D function to a 4D function. For better analysis, light field representation needs to be simplified further, and epi-polar plane images (EPI) is one way for achieving this [19].

The main concept of EPI is to utilize a geometrical relationship between a point in the 3D space and its projection on different camera views in the light field's view plane. Figure 2.6 illustrates the formation of EPI in the HPO light field, where the view plane is reduced to the view line. Cameras are placed through the t axis and capture the scene with an equal distance between adjacent viewpoints as depicted in Figure 2.6(a). Different scene perspectives captured by cameras along the t axis are shown in Figure 2.6(b). Then, a particular row from all camera images in t axis is chosen and stacked up together to form an EPI, as depicted in Figure 2.6(c). The dimension of a single EPI is equal to (n_{im}, n_{col}), where n_{im} is the number of cameras, referred as spatial resolution and n_{col} is the image's horizontal resolution, referred as angular resolution. The EPI of light field $L(s, t, u, v)$ can be expressed as $E_{s_0,u_0}(t,v) = L(s_0,t,u_0,v)$ by fixing $s = s_0, u = u_0$ or $E_{t_0,v_0}(s,u)$ by fixing $t = t_0, v = v_0$.

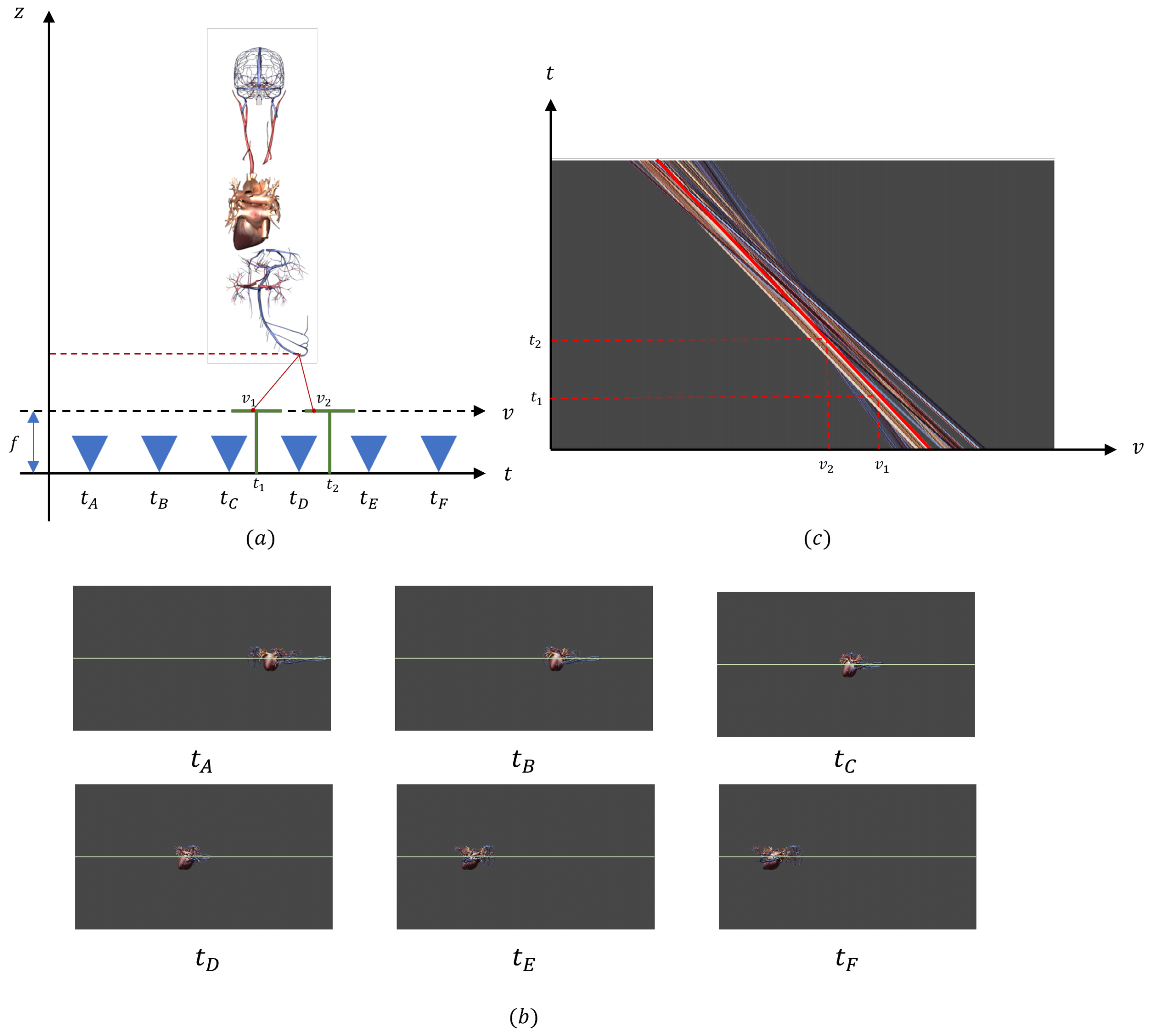


Figure 2.6. *The formation of EPI in a 3D scene. (a) 3D scene captured by multiple equidistant cameras along the t axis. (b) Images captured by cameras in various perspectives. (c) The structured EPI from a selected row in various camera images.*

The benefit of using EPI is mapping each point in the 3D scene onto a line, which makes the analysis and processing of a light field more straightforward. According to Figure 2.6(a), the camera to camera distance is $\Delta t = t_2 - t_1$ corresponding to baseline b in Equation 2.3. The distance between two points in the image plane is $\Delta v = v_2 - v_1$ and corresponds to disparity d in Equation 2.3. The relation between v and t can be expressed as

$$v = \frac{v_2 - v_1}{t_2 - t_1}(t - t_1) + v_1 = \frac{f}{z_0}(t - t_1) + v_1. \tag{2.4}$$

The slope of a line in the EPI depends on the adjacent cameras' distance, image plane resolution, and the distance of a 3D point to the view plane. As points get further away from the view plane, the slope of the corresponding line in the EPI increases and becomes vertical in infinity which refers to zero disparity.

2.5.1 Representation of EPI in Fourier Domain

Due to the EPI's regular structure, its Fourier representation is useful for further analysis [28], [29]. A scene with limited depth range will create a bow-tie shape in Fourier domain as depicted in Figure 2.7(b). The bow-tie shape in the spectral domain depends on the minimum and maximum distance of the view plane to the scene, denoted as z_{min} and z_{max}, as shown in Figure 2.7(a). Every line in the spectral domain represents a depth layer in the 3D space, and all points in the same depth are placed on a line in the spectral domain. In addition, a line in EPI parallel to the spatial spectral axis, Ω_v, indicates points in infinity with zero disparity.

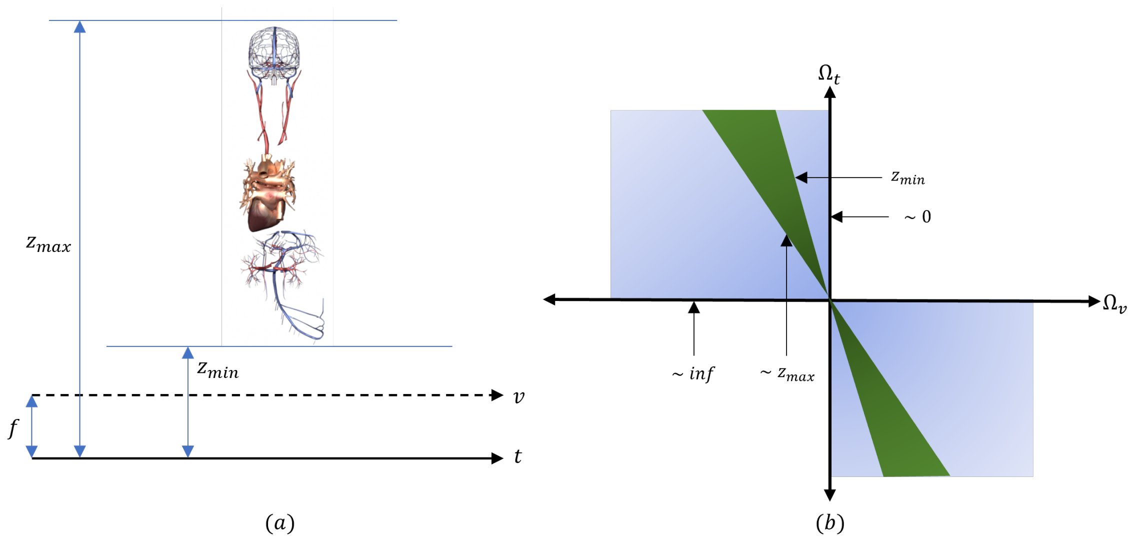


Figure 2.7. *Demonstration of EPI in Fourier domain. (a) 3D scene set up. (b) EPI in Fourier domain.*

2.5.2 Shearing Epi-Polar Plane Images

As shown in Section 2.5, zero disparity occurs at infinity. Zero disparity can also be achieved for objects at finite depth by shearing each row in the EPI along the angular axis v, by [30]

$$\delta_n = \frac{f \times (t_n - t_{\frac{n_{im}}{2}})}{z_c} \quad 0 \leq n \leq n_{im}, \tag{2.5}$$

where δ_n is the shearing amount, f is the camera focal length, t_n is the n^{th} camera viewpoint along the t axis, and z_c is the depth where zero disparity occurs. Figure 2.8(a) indicates how EPI looks after shearing, and Figure 2.8(b) illustrates sheared EPI in the Fourier domain. As seen, the bow-tie shape will rotate in a way that the line corresponding to z_c is parallel to Ω_t axis.

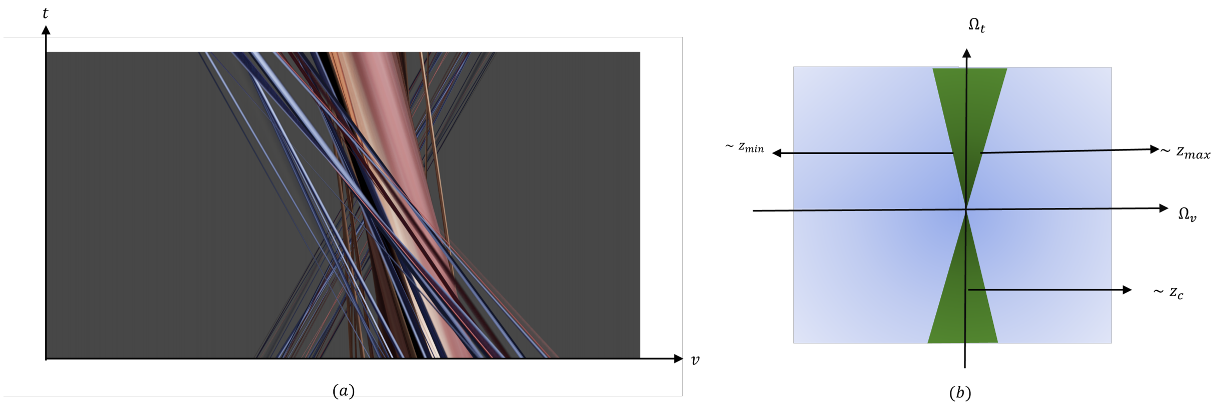


Figure 2.8. *The result of EPI shown in Figure 2.6 after shearing (a). The result of Fourier representation of EPI shown in Figure 2.7 after shearing (b).*

2.6 Densely Sampled Light Field

A densely sampled light field (DSLF) is a light field in which the absolute value of the minimum and the maximum disparities between two adjacent camera views are less than or equal to one. In this case, due to high sampling rate in the spatial domain, lines in EPI are continuous. In DSLF, novel views can be synthesized with bilinear or bicubic interpolation without introducing any substantial aliasing error during the interpolation [20].

The required sampling density of DSLF depends on several factors including the minimum distance of the scene to the camera plane and the camera resolution. Therefore, DSLF of a scene with objects closer to the view plane must have a smaller distance between adjacent views. Furthermore, DSLF can be utilized for removing and blurring details inside a scene by applying multi-dimensional filters [20].

2.7 Projection-based Light Field Displays

There are various techniques for displaying light field content including integral imaging displays [31], super multi-view displays (SMVD) [32], projection-based displays [4], and holographic stereograms [33]. In this thesis, we specifically utilize a projection-based light field display for light field content demonstration.

The projection-based light field displays are capable of reconstructing an approximation of the continuous plenoptic function out of a discrete set of rays. Projection-based light field displays are made of two main parts depicted in Figure 2.9(a). The first part is a set of projectors that count as ray generators, and the second part is the screen plane which is an optical element that acts as a discrete to continuous converter. For HPO light field displays, the holographic screen behaves as an anisotropic diffuser that diffuses a single light ray into a light beam with a narrow horizontal angle δ_x and a wide vertical angle δ_y as shown in Figure 2.9(b). In these displays, 3D objects are recreated based on ray combinations and observer position [4].

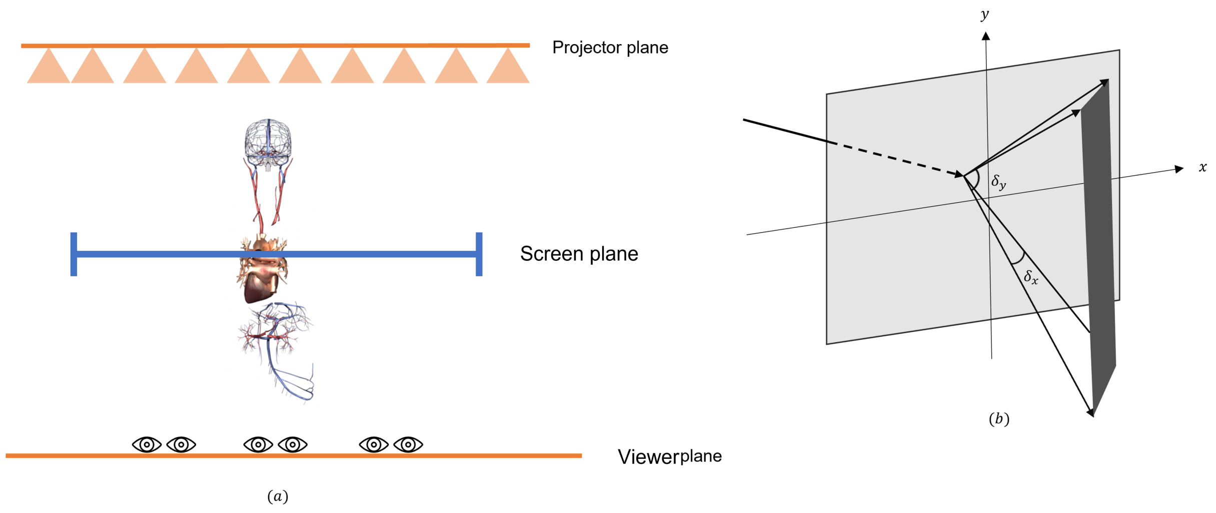


Figure 2.9. *(a) Projection-based light field display setup (b) Anisotrpic diffuser converts a light ray into a light beam.*

Projection-based light field displays do not have pixel structure as rays are generated from several projectors, and propagated rays have irregular patterns on the display's screen. Moreover, total number of rays generated by projectors define the angular and spatial resolution of the display, which manifests itself as a throughput of the display. In the end, this type of light field display can construct continuous parallax; however, they require powerful hardware to process the necessarily high amount of data.

2.8 Light Field Capture

For content visualization on light field displays, a light field needs to be captured from real-world or synthetic scenes. Light field capturing systems for real-world scenes are categorized into three main groups including devices with arrays of lenses, moving cameras, and camera arrays [34].

Devices with arrays of lenses, also referred to as plenoptic cameras, are made of cameras with a microlens arrays between the image sensor and main camera lens [35]. Devices with camera arrays are composed of multiple cameras mounted on a static rig to capture a scene with a wide baseline [36]. Devices with moving cameras are made of single or multiple cameras mounted on a motorized linear system [36]. For capturing dynamic scenes, several cameras on a static rig are used, whereas a motorized linear system is used for static DSLF capture. Capturing with a static rig produces a sparsely sampled light field; therefore, proper view interpolation is essential to avoid aliasing. Light field reconstruction methods are used for this purpose [37], [38], [39].

To capture a light field from a synthetic scene, 3D rendering software like Blender can be utilized [40]. Capturing a synthetic light field in 3D rendering software is cheaper and more straightforward because no calibration, distortion correction, and mechanical equipment are required.

2.9 Sampling Pattern

Sampling Pattern in Projection-based Displays

As discussed in the previous section, the projector plane and holographic screen are the main parts of projection-based light field displays. The projector plane has N_p projectors, and the distance between two adjacent projectors is called projector's spatial sampling rate x_p. The projector plane is placed at distance z_p from the screen plane. Each projector produces N_x rays within its field of view FoV_p. In total, all projectors in the display generate N_pN_x rays. Additionally the ray generation in the projector plane is uniformly distributed, and sampling rate in angular domain is $\alpha_p = FoV_p/N_p$. [20] [21]. In summary, the spatial and angular sampling rates at the screen and projector plane, can be expressed as (x_s, α_s) and (x_p, α_p), respectively [25], [20], [21]. Figure 2.10 illustrates the sampling rate in the projector and screen plane.

Sampling Pattern in Camera Plane

As stated in Section 2.8, light field capture happens at the camera plane, and its distance from the display screen is z_c. The camera plane has N_c cameras and each can provide N'_x rays. The spatial and angular sampling rates at camera plane can be denoted as (x_c, α_c) where $\alpha_c = FoV_c/N'_x$. Figure 2.10 indicates spatial and angular sampling rates in projector, screen, and camera planes.

As discussed in Section 2.3, the propagation of a camera ray r at a distance z from the camera plane can be parameterized as

$$\begin{bmatrix} x_z^r \\ \alpha_z^r \end{bmatrix} = \begin{bmatrix} x_c^r - z\tan(\alpha_c^r)) \\ \alpha_c^r \end{bmatrix}, \tag{2.6}$$

where (x_z^r, α_z^r) are the position and the angle of the ray r at distance z from the camera plane. Moreover, (x_c^r, α_c^r) are the position and the angle of the ray r at camera plane. Figure 2.11 (a) shows camera ray space sampling grid at $z = 0$ from the camera plane, and Figure 2.11 (b) manifests the same sampling grid at $z = z_c$ shifted by $z_c \tan(\alpha)$. Each point in the ray space sampling grid corresponds to a ray. Moreover, each sampling grid can be formulated as a sampling matrix, which can be used to resample from one sampling grid to another sampling grid [21]. The sampling matrix at the screen plane, the camera plane, and the projector plane can be defined as

$$V(x_s, \alpha_s) = \begin{bmatrix} x_s & 0 \\ 0 & \alpha_s \end{bmatrix} \tag{2.7}$$

$$V(x_c, \alpha_c) = \begin{bmatrix} x_c & 0 \\ 0 & \alpha_c \end{bmatrix} \tag{2.8}$$

$$V(x_p, \alpha_p) = \begin{bmatrix} x_p & 0 \\ 0 & \alpha_p \end{bmatrix}. \tag{2.9}$$

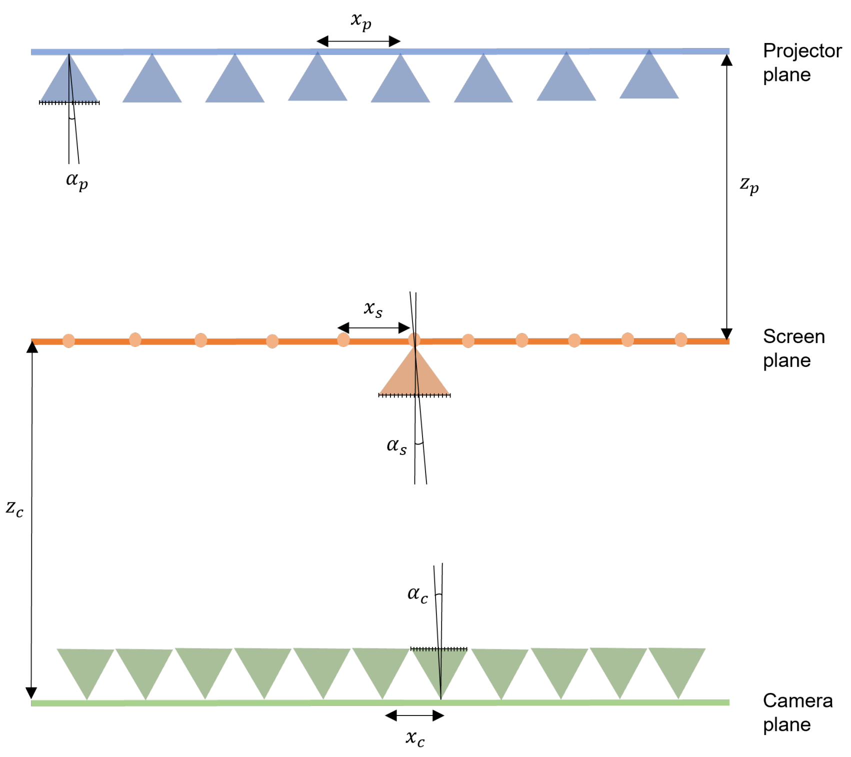


Figure 2.10. *The projection-based light field display setup illustrating angular and spatial sampling rate at projector, screen, and camera plane.*

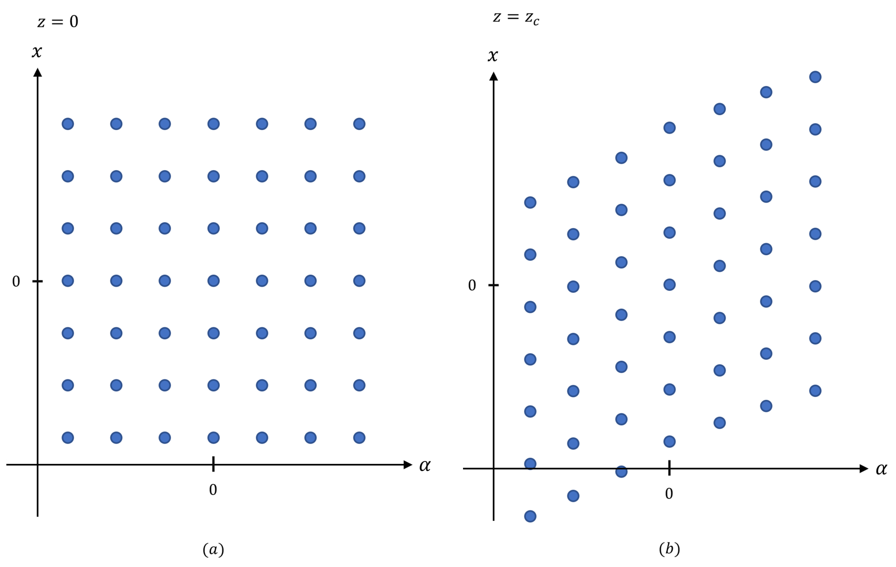


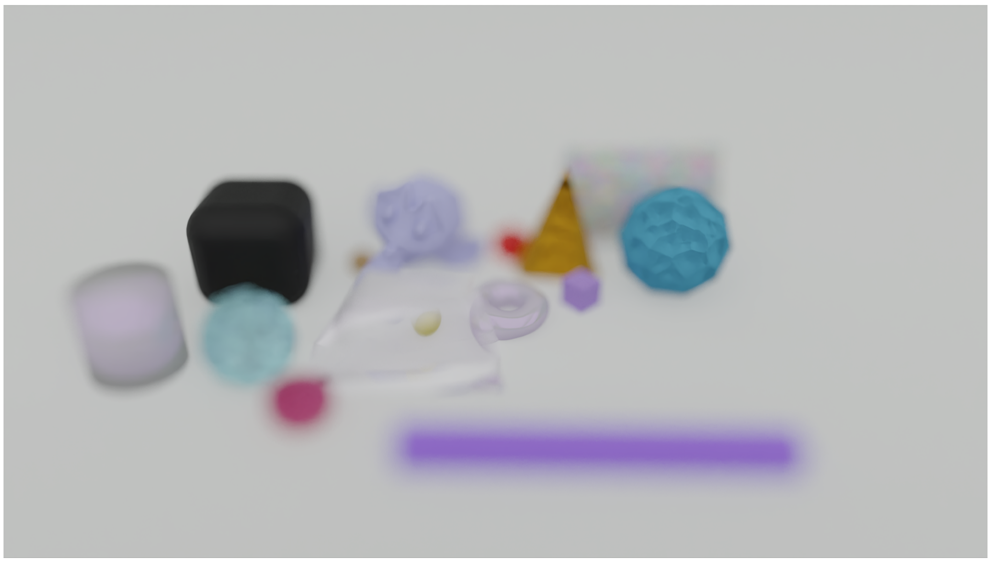
Figure 2.11. *The sampling grid of camera rays at depths (a) $z = 0$ and (b) $z = z_c$ from the camera plane.*

3. DEPTH OF FIELD

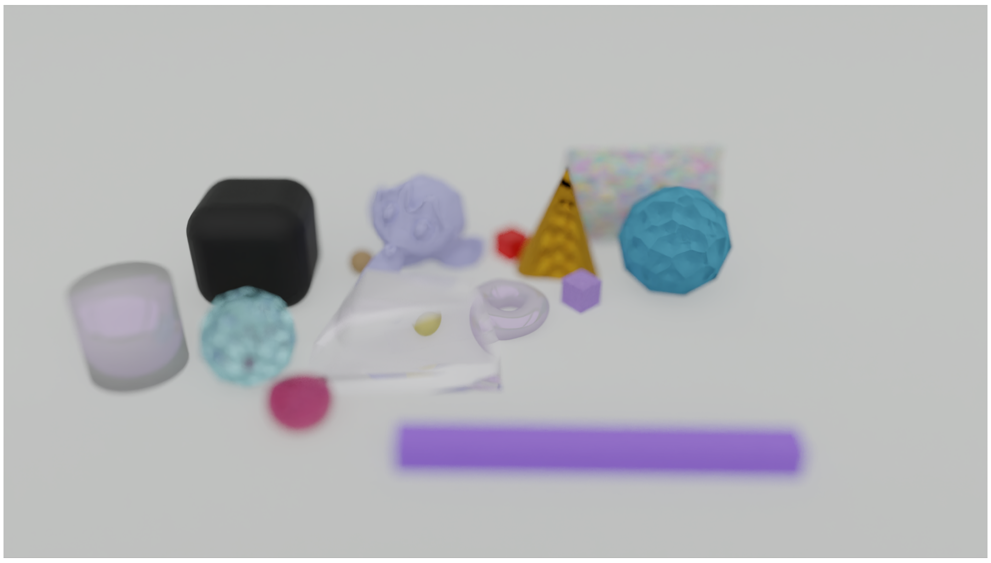
In the discussion so far, we assumed that the light field is captured with so called pinhole camera. In a pinhole camera, all rays reaching the camera sensor pass through a tiny hole. The obtained camera image is entirely sharp. Unfortunately, a pinhole camera is not practical in the real world since it requires extremely long exposure times. Therefore in practice, all real cameras have lenses that focus light rays through an aperture to the camera sensor. Since exposure time is inverse proportional to the aperture size, larger apertures typically need shorter exposure times [41].

Lenses in real cameras with finite aperture sizes focus light on the camera sensor, which creates depth of field (DoF) in the captured images, [41]. DoF is a region between the nearest and furthest objects in the scene where the captured image is approximately sharp. The range of the DoF region has an reverse relation with the aperture size; thus, the bigger the aperture size, the narrower the DoF is [41]. Figure 3.1 depicts a synthetic scene with narrow DoF (a), wide DoF (b), and no DoF (c).

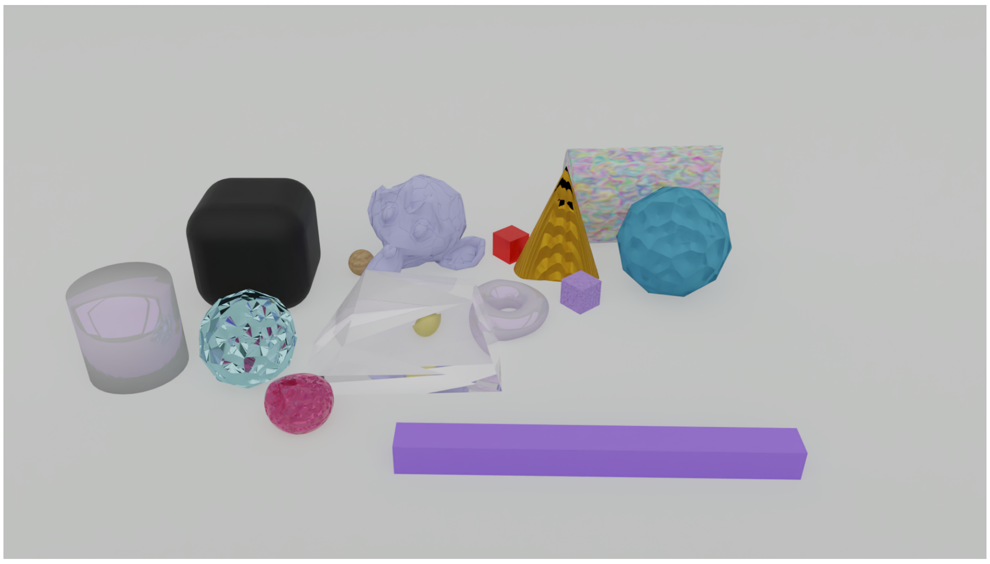
In this chapter, the concept of thin lens model and how it causes DoF in captured images are covered. In addition, the theory about DoF in light field displays is explained. Finally, various graphics algorithms for simulating the physical behavior of lenses are reviewed, and their benefits and restrictions are discussed.



(a)



(b)



(c)

Figure 3.1. *Synthetic scene with narrow DoF (a), wide DoF (b), and no DoF (pinhole camera)(c).*

3.1 Thin Lens Model

Real camera models can be simulated by the classic optic approximation called thin lens model. In such model, light rays from infinity, after passing through the lens, converge to a single point called the focal point, and its distance to the lens is denoted as focal length [41]. According to the Gaussian lens equation, an image of a point at distance z from the lens with focal length f would be created at distance z' from the lens after passing through the lens as illustrated in Figure 3.2. Consequently, if the distance between the image sensor and the lens is z'_f, z_f is considered as focal distance. The Gaussian lens equation can be expressed as [41]

$$\frac{1}{f} = \frac{1}{z_f} - \frac{1}{z'_f}. \tag{3.1}$$

Points that are not located at the focal distance are mapped to a circle on the image sensor called circle of confusion (CoC) [41]. The CoC diameter depends on the lens aperture, the point's distance from the lens, and the focal distance. By using the lens Gaussian equation, the CoC can be estimated as [41]

$$d_c = |\frac{Af(z - z_f)}{z(z + z_f)}| \times \frac{r_x}{s_w}, \tag{3.2}$$

where d_c is the CoC diameter, A is the lens aperture size, f is the focal length, z_f is the focal distance, s_w is the sensor width, r_x is the sensor horizontal resolution, and z is a point's distance to the lens. As shown in Equation 3.2, there is a reverse relation between the aperture size and the DoF range, in which a large aperture size produces a narrow DoF. Figure 3.2 illustrates the focal distance, focal length, and CoC in a thin lens model.

Figure 3.3 shows the relation between CoC diameter and the distance to the lens with 123-mm focal length and 128-mm aperture size focused at $z_f = 8m$. The aforementioned plot is asymmetrical around the focal distance, i.e. the blurring rate is not equal as shown in Figure 3.3 at distance 9 and 7 from the camera lens.

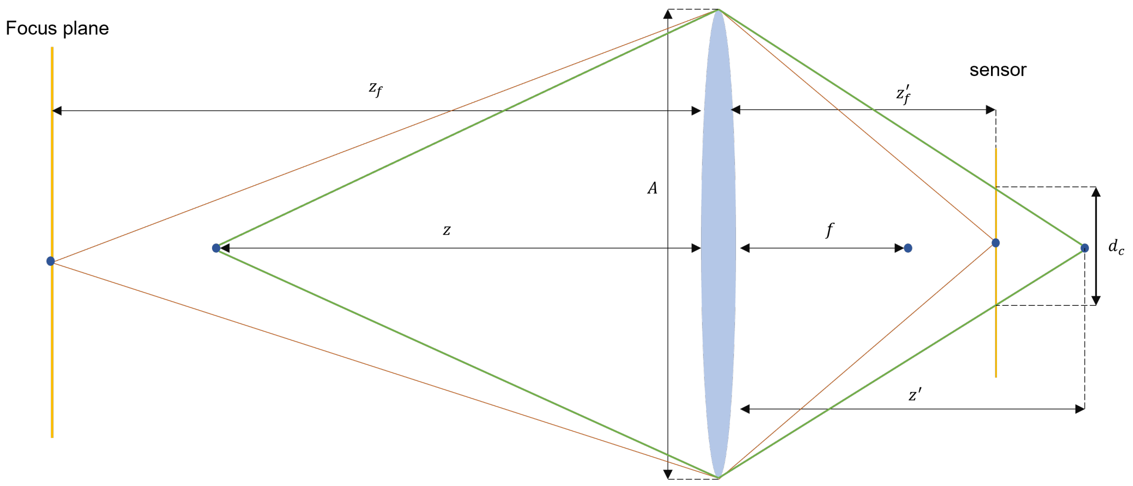


Figure 3.2. *Illustration of thin lens model using ray diagram.*

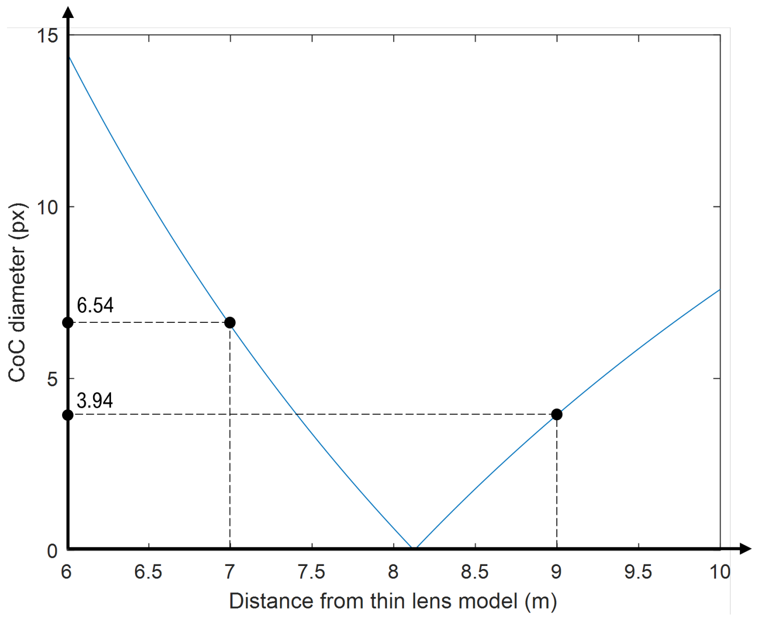


Figure 3.3. *The relation between CoC diameter and distance to the lens with focal length 123-mm and 128-mm aperture size focusing at 8m. The sensor width and resolution is 32-mm and 1280 px respectively.*

3.2 Depth of Field in Light Field Displays

DoF in light field displays is the range between the rear and the front planes located around the screen plane, where the smallest feature size that depends on the display's horizontal spatial resolution can be reproduced [4], [7]. In projection-based light field displays, the smallest feature size is proportional to the distance from the display's screen, which is formulated in Equation 3.3 and illustrated in Figure 3.4 [4].

$$p = p_0 + S\tan(\varphi). \tag{3.3}$$

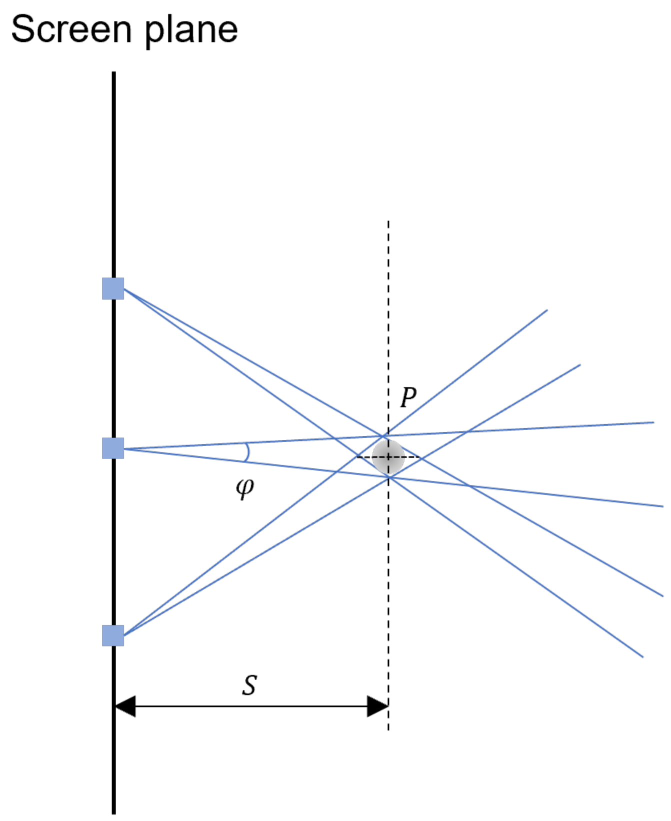


Figure 3.4. *The smallest feature size depends on the distance from the display's screen and the display's spatial and angular resolution.*

In Equation 3.3, p_0 is the smallest feature size at the light field display's screen. Moreover, φ and S are the angular resolution and the feature's distance to the screen, respectively. Furthermore, the horizontal spatial resolution of light field displays is reduced proportional to distance from the display's screen as evaluated below [4],

$$r_S = \frac{p_0}{p_0 + S\tan(\varphi)} \times r_0, \tag{3.4}$$

in which r_0 and r_S correspond to the horizontal spatial resolution at the screen plane and a plane at distance S from the display screen, respectively. Figure 3.5 shows the plot derived from Equation 3.4.

As mentioned in Section 2.9, light field displays have particular angular and spatial sampling rate at the screen level of the display. Thus, the display has a limited bandwidth and cannot reconstruct specific frequencies at a particular distance from the display's screen

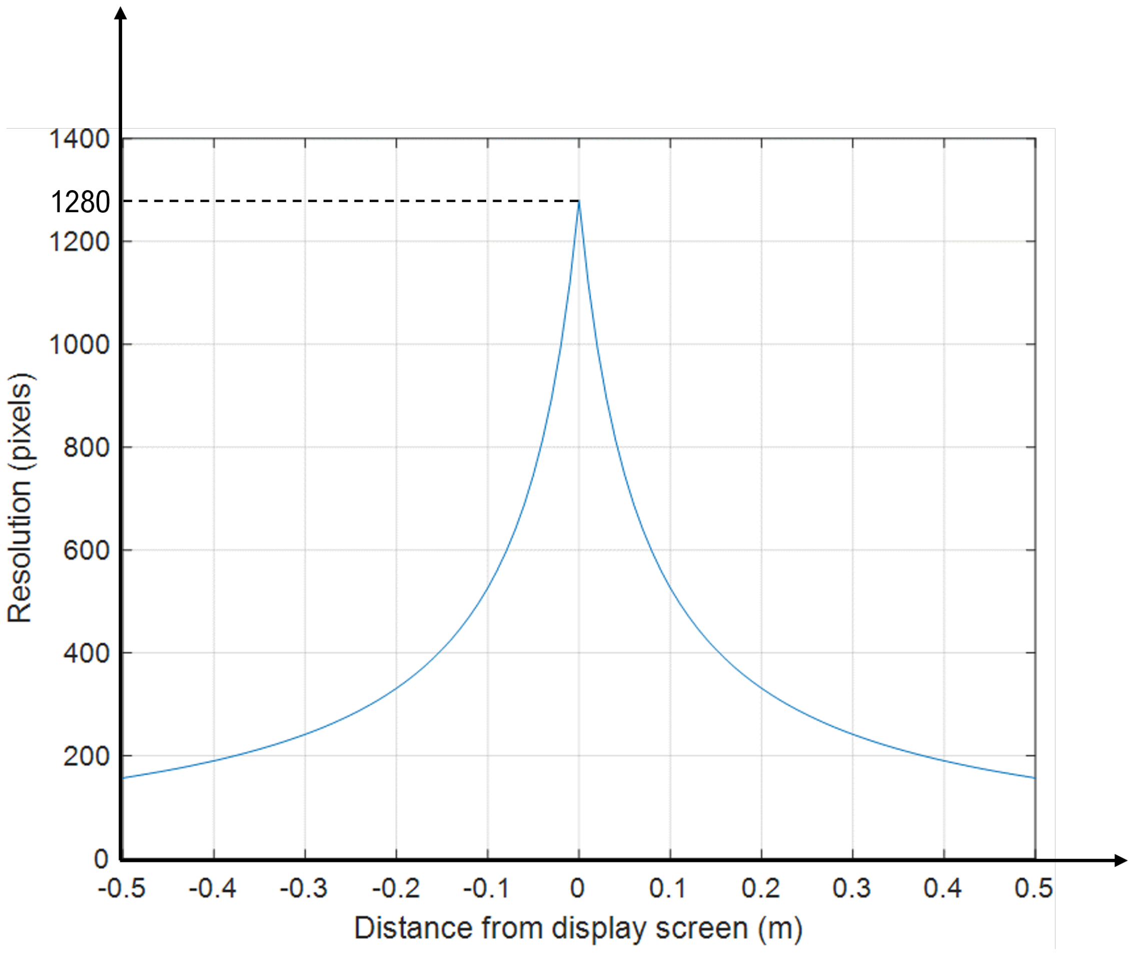


Figure 3.5. *The reduction in horizontal spatial resolution of projection-based light field displays proportional to distance from display's screen.*

[5]. The display bandwidth is the representative of its DoF which can be expressed as,

$$H(\Omega_v, \Omega_t) = \begin{cases} 1 & : \ |\Omega_v| \leq \pi/\Delta v \text{ and } |\Omega_t| \leq \pi/\Delta t \\ 0 & : \ \text{otherwise} \ , \end{cases} \tag{3.5}$$

where Δt and Δv are spatial and angular sampling rates, respectively. Moreover, Ω_t and Ω_v are the corresponding spatial and angular frequency, respectively. Figure 3.6 illustrates display DoF by utilizing Fourier domain representation of EPI, in which lines $\Omega_t \Delta t/\Delta v - \Omega_v = 0$ and $\Omega_t \Delta t/\Delta v + \Omega_v = 0$ demonstrate planes around the display's screen representing limits of the light field display's DoF [5], [6].

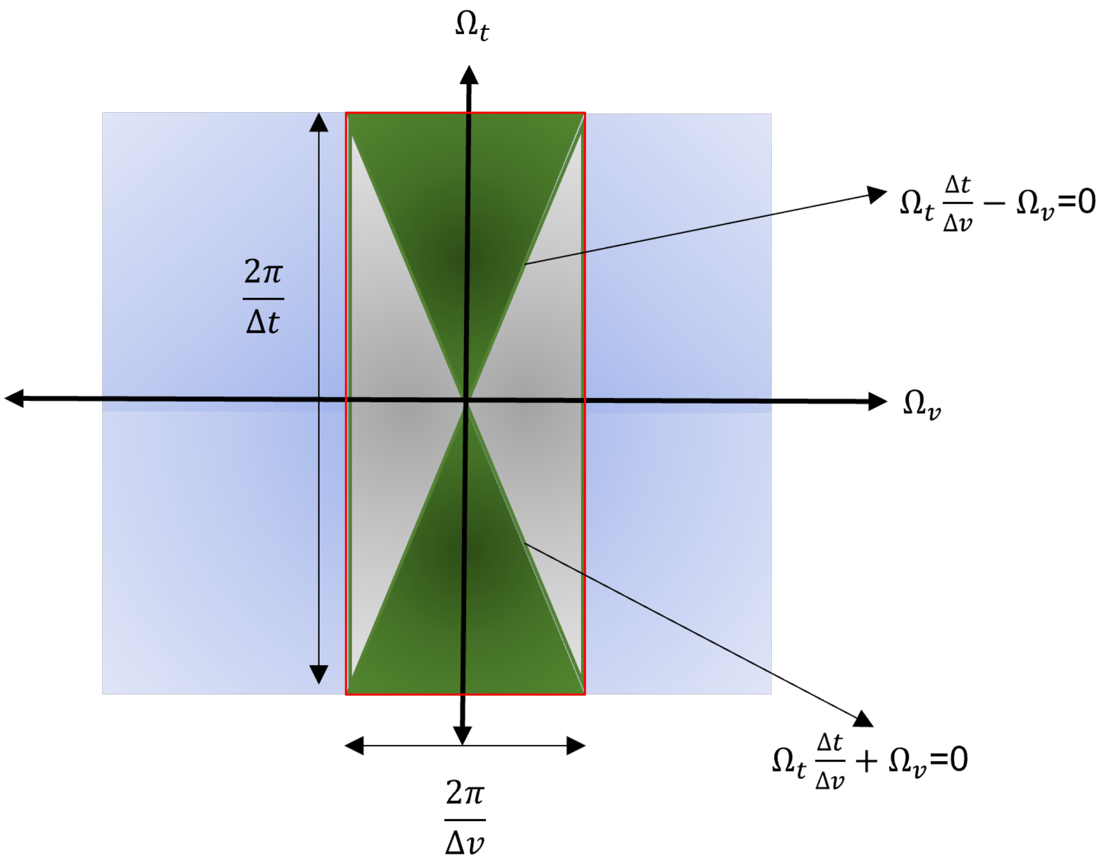

Ω_t
$\frac{2\pi}{\Delta t}$
$\Omega_t \frac{\Delta t}{\Delta v} - \Omega_v = 0$
Ω_v
$\Omega_t \frac{\Delta t}{\Delta v} + \Omega_v = 0$
$\frac{2\pi}{\Delta v}$

Figure 3.6. *The slanted lines corresponds to front and rear depth planes around the screen representing display's DoF.*

3.3 Depth of Field Rendering

As discussed in Section 3.2, proportional to distance from the screen plane, the smallest feature size that can be reproduced by the display increases. Therefore, features that are too small at a particular distance from the display's screen would be shown with aliasing and need to be blurred [4], [5]. One solution to prevent aliasing in light field displays is to modify light field data in a way that virtual content shown in the DoF region stays sharp and everything outside of that region is gradually blurred. The mentioned solution is equivalent to applying DoF to the captured light field.

In computer graphics, the act of simulating a thin lens model is called DoF rendering. There are various methodologies for applying DoF in the rendering process, including path-tracing [8], gathering [12], [13], scattering [14], multi-view rendering [9], [10], [11], [5], and multi-layering [15], [16], [17].

DoF rendering with path-tracing can be achieved by tracing rays in pixel and aperture sampling domains to determine the final color of a pixel. Each pixel's surrounding represents the lens aperture in the aperture sampling [8], [42]. To render an image with high DoF quality, it is necessary to sample a large number of rays in both pixel and aperture sampling domains, which is time-consuming. To reduce the sampling rate in the two domains, there is a method that considers whether the sampling region is in-focus or out of focus, and adapts the sampling rate accordingly [43]. Since, path-tracing approaches are slow, utilizing them for real-time application is not suitable.

On the contrary, gathering and scattering methods in DoF rendering, which start with a pinhole camera image and a depth map of the rendered scene are appropriate for real-time applications. In the gathering approach, a pixel color value is computed by accumulating surrounding pixels within the pixel's CoC [12], [13]. Applying this approach naively can cause artifacts like color bleeding (intensity of the in-focus region bleeds into the out-of-focus region) and depth discontinuity (the out-of-focus region in front of the in-focus region is blurred with sharp edges) [15]. More advance filtering can prevent color bleeding by dividing the depth map into discrete layers. Also, depth discontinuity can be partially solved by approximating occluded areas [44], [45].

Moreover, in scattering methods, a pixel color intensity is distributed over its CoC. Therefore, each pixel is rendered as it is semi-transparent, and multiple color values from scattered pixels are assigned to it. The final color value is calculated by accumulating color of those pixels [14], [46]. To achieve a better quality with scattering methods, scattered pixels need to be sorted regarding their depths and advance data structures should be utilized. [15], [46]. However, sorting is challenging in the case of performance; thus, fast sorting algorithm is crucial for performance improvement [15].

Deep learning can be adopted in post-processing approaches to generate DoF [47], which is mainly utilized to focus on regions that are in front. Additionally, there are image-processing approaches like [48] that use optimization algorithms to approximate occluded pixels values. Also, anisotropic diffusion is another approach for producing DoF in images [49].

Most post-processing DoF rendering methods cannot render partial occlusion properly due to the lack of information in occluded regions. Moreover, occluded regions that are in front become semi-transparent in the neighborhood of in-focus regions after applying DoF. One solution for this problem is to divide the scene geometry into discrete number of layers, store the projection of each area into a multi-layer image, and utilize those images to solve the partial occlusion issues. The multi-layering approach can be combined with ray-tracing [16], gathering [17] and scattering [15] approaches, and each method has its own advantages and disadvantages. Merging multi-layering with ray-tracing generates DoF with a good quality; however, it is slow in cases with a high number of samples per pixel [16]. Integrating multi-layering with gathering is quite fast but the quality is not as good as in other methods [17]. Finally, combining multi-layering with scattering produces the result with average speed and average quality in comparison with other two multi-layering methods [15].

Multi-view DoF rendering is another approach for creating DoF by using several adjacent views. For instance, one method computes light field by image warping around a single view in order to render DoF [11]. Another method processes light field in the frequency domain to generate images following light field display's bandwidth [5]. Several other methods use adjacent views according to lens aperture for rendering DoF [9], [10]. In multi-view algorithms, to have an acceptable DoF without any artifacts, a large number of views must be used which is computationally demanding.

The focus of this thesis is to adopt multi-view methods, such as [5] and [10] which apply depth-based anti-aliasing filters on DSLF to remove artifacts introduced by light field display's limited DoF. As a real-time approach, we utilize a multi-layer DoF rendering with tiled splatting [15] for the same purpose. The result of first two multi-view DoF rendering methods is used as a ground truth for comparing the result of real-time algorithm.

4. MULTI-VIEW DOF RENDERING USING DEPTH-GUIDED ANTI-ALIASING FILTERS

As discussed in Section 3.2, light field displays have limited DoF due to their restricted bandwidth, which causes drop in the displays' spatial resolution by increasing the distance from the display's screen. Therefore, light field displays are not able to present content with too small details at far distance from the screen without aliasing.

This chapter proposes two offline multi-view DoF rendering methods that utilize depth-guided anti-aliasing filters to solve the aliasing issue in light field displays [5], [10]. Both algorithms adopt DSLF to create the DoF effect. In both cases, the filter cut-off frequency is determined based on the resampling ratio from the ray space in the camera plane to the ray space in the screen plane. The first method is based on filtering the 4D light field with a 2D circular FIR filter. Second method uses EPI as a light field representation with each EPI being filtered with a 1D FIR filter. Figure 4.1 illustrates the pipeline of both multi-view DoF rendering methods.

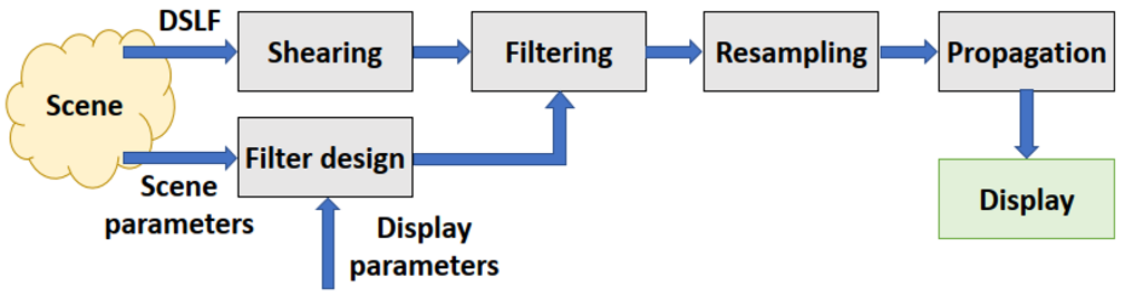


Figure 4.1. *The pipeline of multi-view DoF rendering using depth-guided anti-aliasing filters.*

4.1 HPO and FP DSLF Rendering

As discussed in Section 2.6, in a DSLF, the disparity between adjacent views must be less than or equal to one. Capturing a DSLF can be done in different ways as discussed in Section 2.8. One technique is to use 3D rendering engines like Blender [40] for rendering a synthetic DSLF. Figure 4.2(a) and 4.2(b) demonstrate the horizontal and vertical parallax configurations for rendering a DSLF and Table 4.1 shows the parameters' values used for rendering example in this thesis. To shear the rendered images in Blender, the built-in camera's shift property is utilized [50], and the applied shifting amount in the horizontal and vertical directions can be expressed as [51]

$$shift_t = \frac{c_t}{D_t} \tag{4.1}$$

$$shift_s = \frac{c_s}{D_s}, \tag{4.2}$$

where (c_t, c_s) are the camera position in the camera plane, and (D_t, D_s) corresponds to the display width and height.

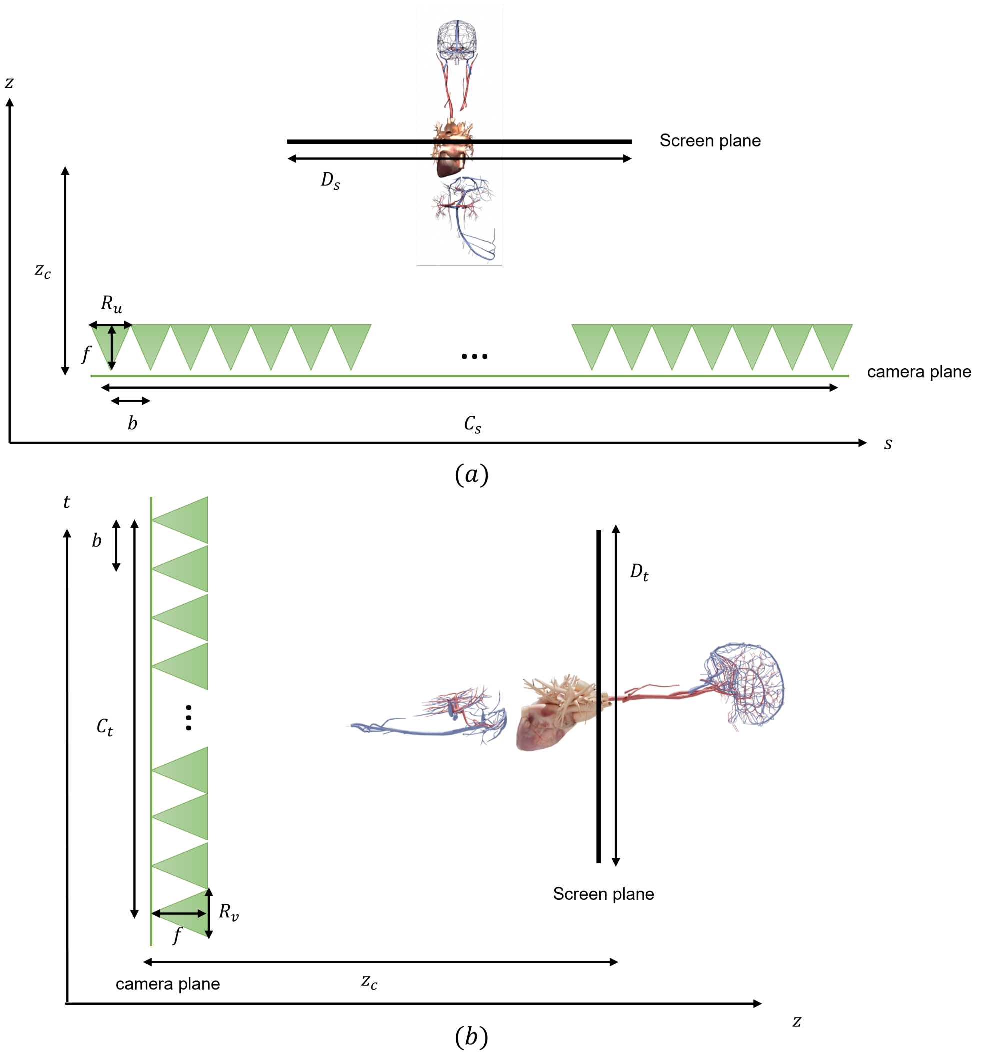


Figure 4.2. *(a) Horizontal and (b) vertical parallax configurations for rendering a DSLF in Blender.*

Name	Symbol	Value
Horizontal and vertical distance between adjacent views [mm]	b	4
Display width [mm]	D_t	1560
Display height [mm]	D_s	880
Distance to display [mm]	z_c	3000
Horizontal resolution [px]	R_u	1280
Vertical resolution [px]	R_v	720
Sensor width [mm]	s_w	32
Number of horizontal views	C_t	661
Number of horizontal views	C_s	221

Table 4.1. *Configuration parameters' values for rendering a DSLF in Blender.*

4.2 Shearing Sampling Grid in Ray Space from Camera to Screen Plane

As referred in Sections 2.9 and 3.2, light field displays have particular bandwidth in both spatial and angular domain at the screen plane, which determines displays' DoF; therefore, they are not able to show content with too small details at a specific distance because of insufficient sampling rate at the screen level, which causes aliasing. To solve the aliasing issue, one solution is to render a DSLF with high angular and spatial resolution in the camera plane, resample, and propagate from the camera plane to the screen plane by ray propagation discussed in Section 2.3. Consequently, the post-processed light field follows the displays' bandwidth. Figure 4.3 and Equations 4.3 and 4.4 illustrate the camera sampling grid and its basis vectors before and after the ray propagation.

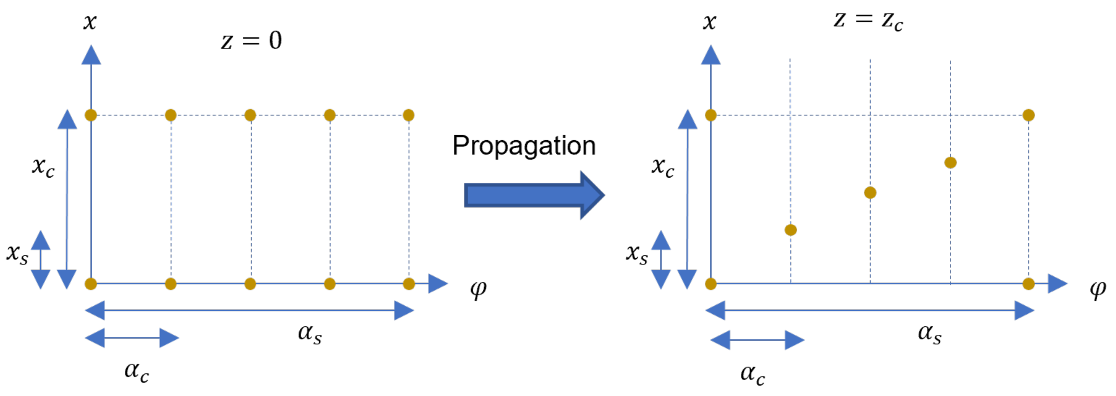


Figure 4.3. *Camera sampling grid before and after propagation from camera plane to screen plane.*

$$V(x_c, \alpha_c) = \begin{bmatrix} x_c & 0 \\ 0 & \alpha_c \end{bmatrix} \tag{4.3}$$

$$V(x_c, \alpha_c, z_c) = \begin{bmatrix} x_s & 0 \\ \alpha_c & \alpha_s \end{bmatrix}. \tag{4.4}$$

Here α_c in the sampling grid $V(x_c, \alpha_c, z_c)$ can be omitted because $\alpha_c << \alpha_s$, which will simplify this filter design. As a result, the new sampling grid $V'(x_c, \alpha_c, z_c)$ can be expressed as

$$V'(x_c, \alpha_c, z_c) = \begin{bmatrix} x_s & 0 \\ 0 & \theta_c \end{bmatrix}, \tag{4.5}$$

where θ_c is the angle between two adjacent view points in the camera plane as shown in Figure 4.4. As the first basis vectors in both sampling grids, $V(x_s, \alpha_s)$ and $V'(x_c, \alpha_c, z_c)$ are equal, the resampling factor from $V(x_s, \alpha_s)$ to $V'(x_c, \alpha_c, z_c)$ is $K = \left\lceil \frac{\alpha_s}{\theta_c} \right\rceil$.

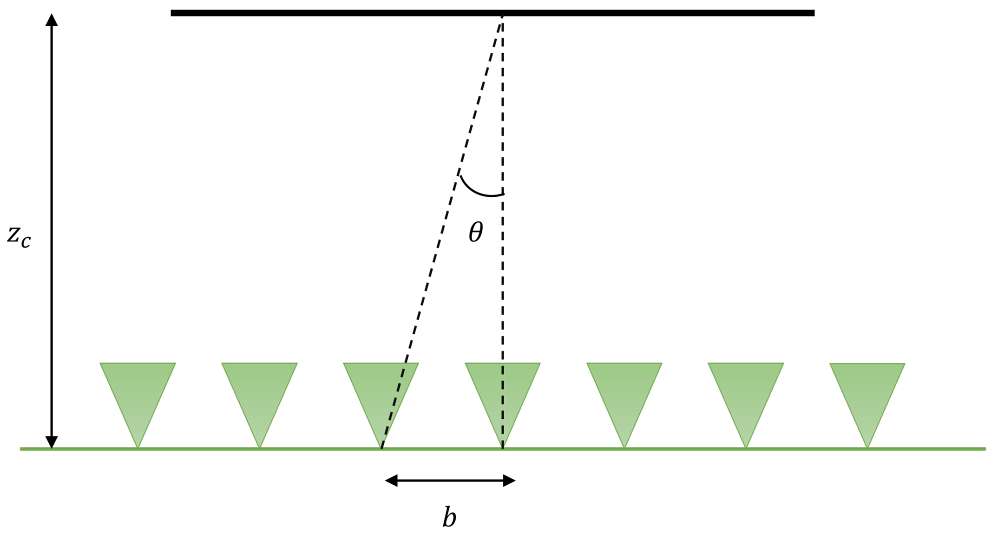


Figure 4.4. *The angle between two adjacent cameras depending on the distance between the camera and the screen plane.*

4.3 Aperture Filtering

After evaluating the decimation factor, namely, the ratio $K = \left\lceil \frac{\alpha_s}{\theta_c} \right\rceil$, light field in the camera plane needs to be decimated in the spatial domain. Therefore, an anti-aliasing low-pass filter is required to be designed. This approach applies a 2D uniform circular filter over 4D light field. The corresponding impulse response can be formulated as

$$h_{s_0,t_0}(s,t) = \begin{cases} 1 & : \sqrt{(s-s_0)^2 + (t-t_0)^2} \leq r \\ 0 & : \text{otherwise} \,, \end{cases} \tag{4.6}$$

where (s_0, t_0) is the camera position in the camera plane and r is the filter radius determined by half of the decimation factor K, computed in Section 4.2. The computed impulse response is applied over 4D light field $L(s,t,u,v)$ at camera position (s_0,t_0) as shown in Equation 4.7 and Figure 4.5.

$$L_f(s_0,t_0,u,v) = \frac{1}{M} \sum_s \sum_t h_{s_0,t_0}(s,t) L(s,t,u,v). \tag{4.7}$$

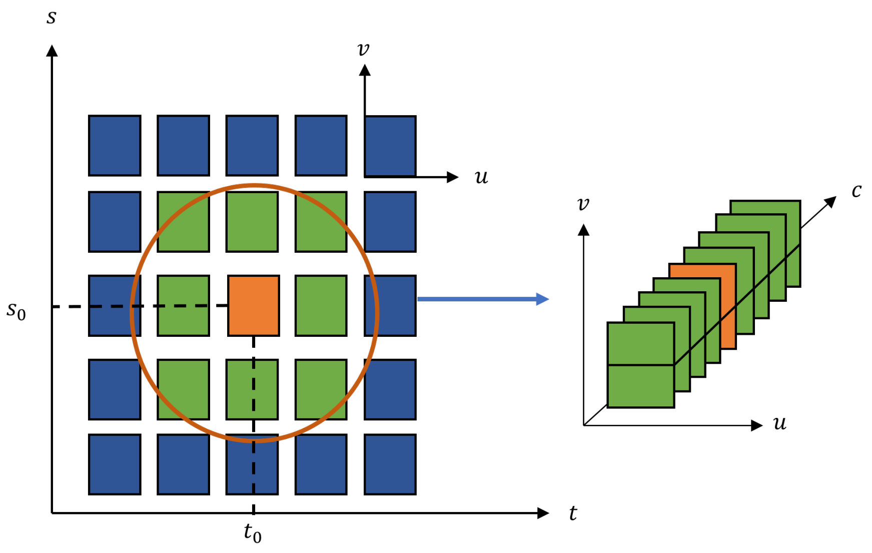
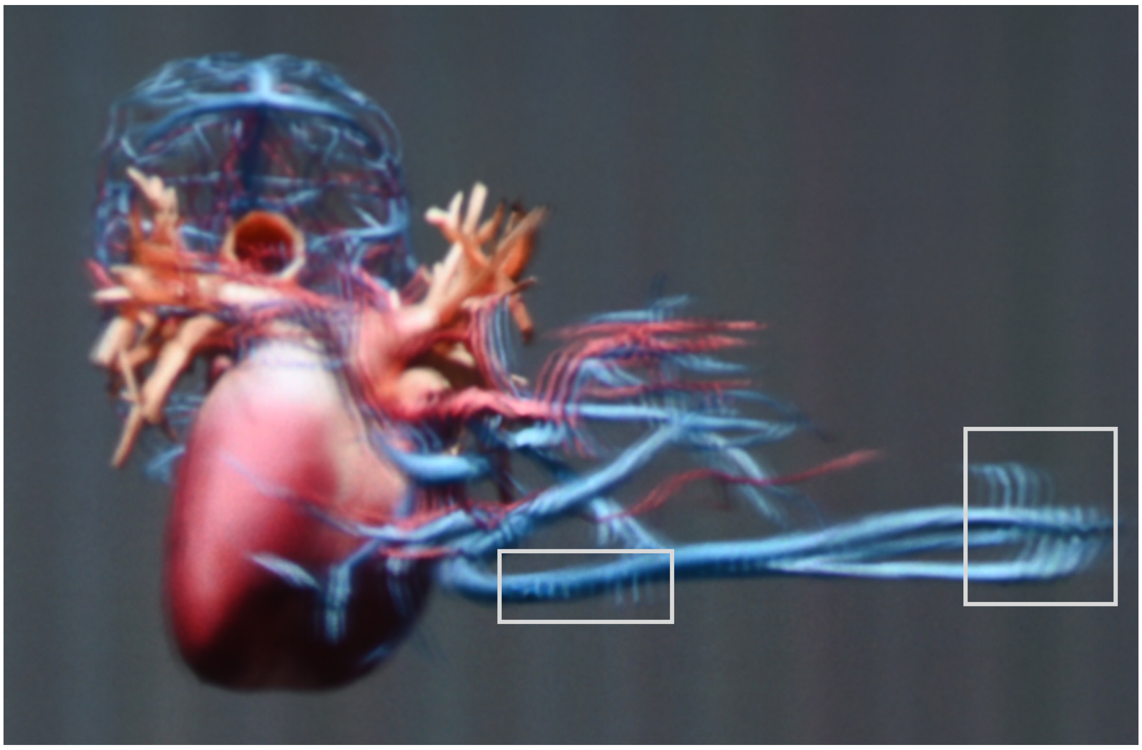


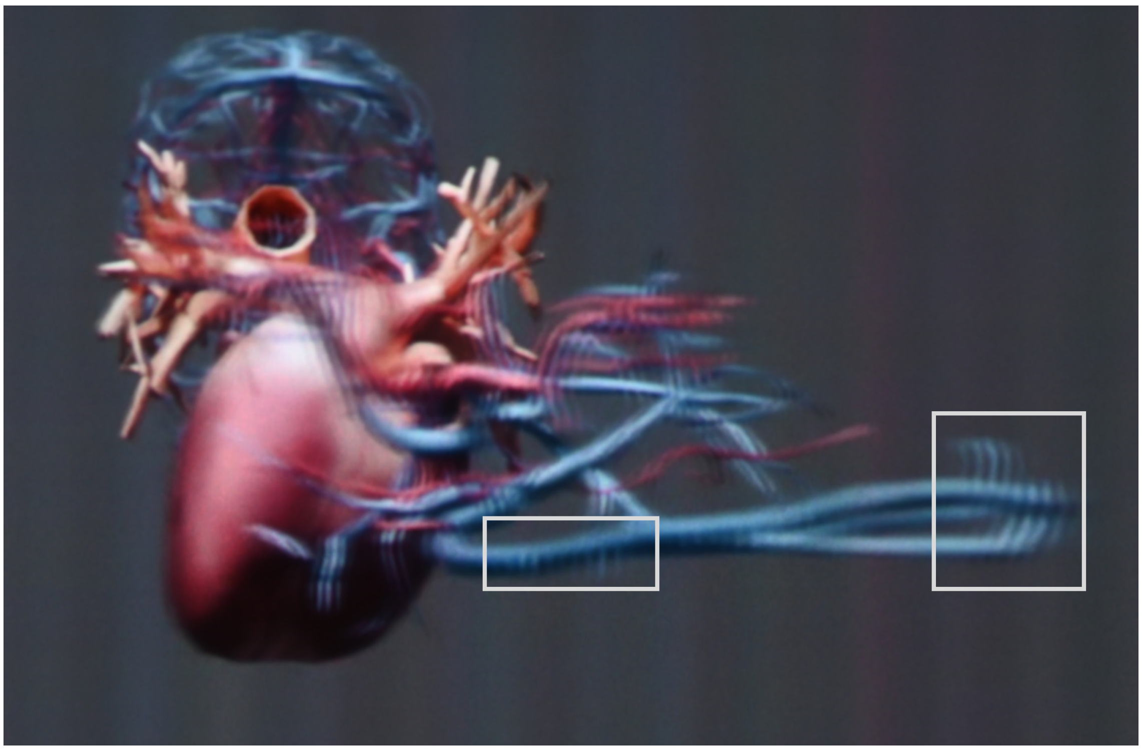
Figure 4.5. *The process of accumulation and averaging over camera viewpoints residing in a circular proximity of the camera* (s_0,t_0) *with a radius* r*.*

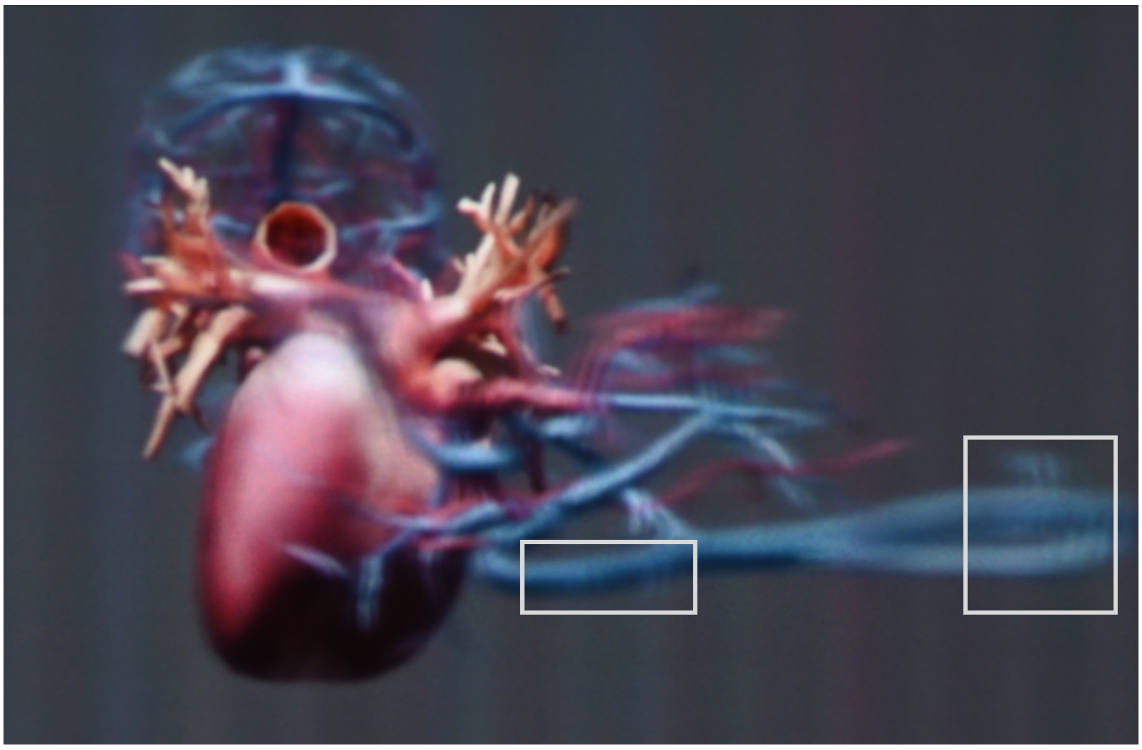
Result

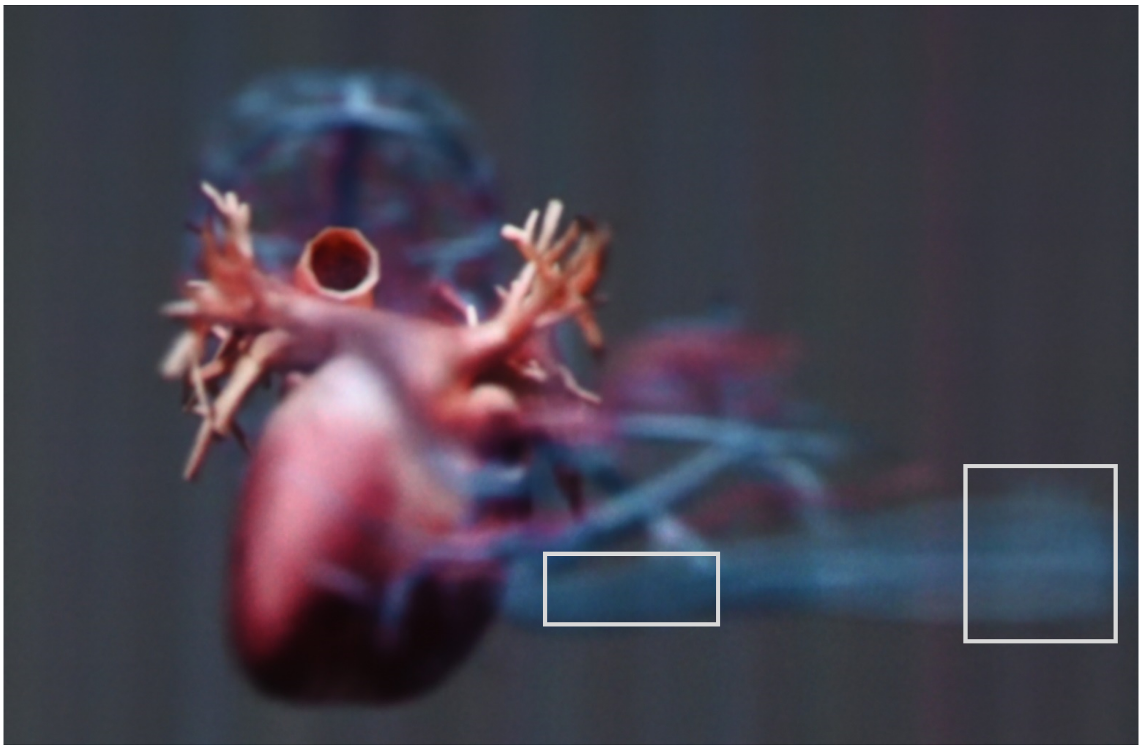
Figure 4.6 shows the result of filtering an FP light field with a 2D uniform FIR circular filter with radii 1, 4, 7, and 16. The filtered light field is visualized on Holografika's Holovizio 722RC projection-based light field display. The aperture filtering method has pros and cons. For instance, rendering or capturing the FP DSLF is slow due to high time complexity, and applying 2D filters on DSLF is both time-consuming and memory-demanding. However, the filtered light field follows the display's bandwidth and prevents the aliasing.



(a) *Result of aperture filtering with* $r = 1$


(b) *Result of aperture filtering with* $r = 4$


(c) *Result of aperture filtering with* $r = 7$



***(d)** Result of aperture filtering with $r = 16$*

***Figure 4.6.** Visualization of a filtered FP light field with a 2D uniform FIR circular filter with different radii on a projection-based light field display.*

4.4 FIR Filtering in EPI Domain

The next method for anti-aliasing is to apply 1D FIR low-pass filters on each EPI derived from FP or HPO light field. The windowing technique is used to design filters' impulse response $h(n, f_c)$ which can be expressed as

$$h(n, f_c) = w(n)h_{id}(n, f_c) \text{ for } \frac{-N}{2} \leq n \leq \frac{N}{2}, \tag{4.8}$$

where $w(n)$ is the window function. In this thesis, the Hamming and Gaussian windows are used that are defined as

$$w_{hamming}(n) = \begin{cases} 0.54 + 0.46\cos(\frac{2\pi n}{N'}) & : \frac{-N'}{2} \leq n \leq \frac{N'}{2} \\ 0 & : \text{otherwise} \end{cases} \tag{4.9}$$

$$w_{gaussian}(n) = \begin{cases} e^{\frac{-1}{2}(a\frac{n}{(N'-1)/2})^2} & : \frac{-(N'-1)}{2} \leq n \leq \frac{(N'-1)}{2} \\ 0 & : \text{otherwise}\ , \end{cases} \tag{4.10}$$

and $h_{id}(n, f_c)$ is the impulse response of the ideal sinc filter

$$h_{id}(n, f_c) = \begin{cases} 2f_c \mathrm{sinc}(2\pi f_c) & : \text{ for } n \neq 0 \\ 2f_c & : \text{ for } n = 0, \end{cases} \tag{4.11}$$

where the cut-off frequency has inverse relation with decimation factor

$$f_c = \frac{1}{K}. \tag{4.12}$$

The filter order N needs to be equal or greater than the decimation factor K to have a proper filter performance with the preferable filter order being $N = 4K$. The filtered light field is obtained by sequential applying FIR filters on vertical and horizontal EPIs which can be expressed as

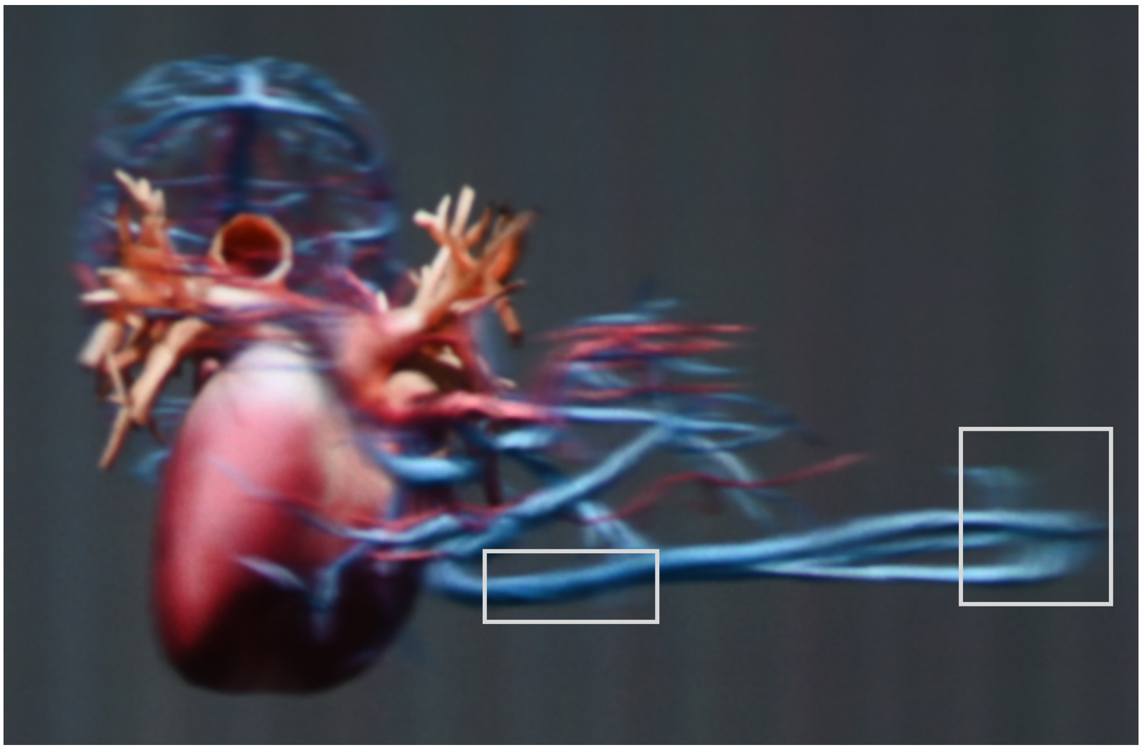
$$E^{(h)}_{s_0,u_0}(t, v_0) = E_{s_0,u_0}(t, v_0) * h(t, f_c) \text{ for } \forall v_0 \in v \tag{4.13}$$

$$E^{(v)}_{t_0,v_0}(s, u_0) = E^{(h)}_{t_0,v_0}(s, u_0) * h(t, f_c) \text{ for } \forall u_0 \in u, \tag{4.14}$$

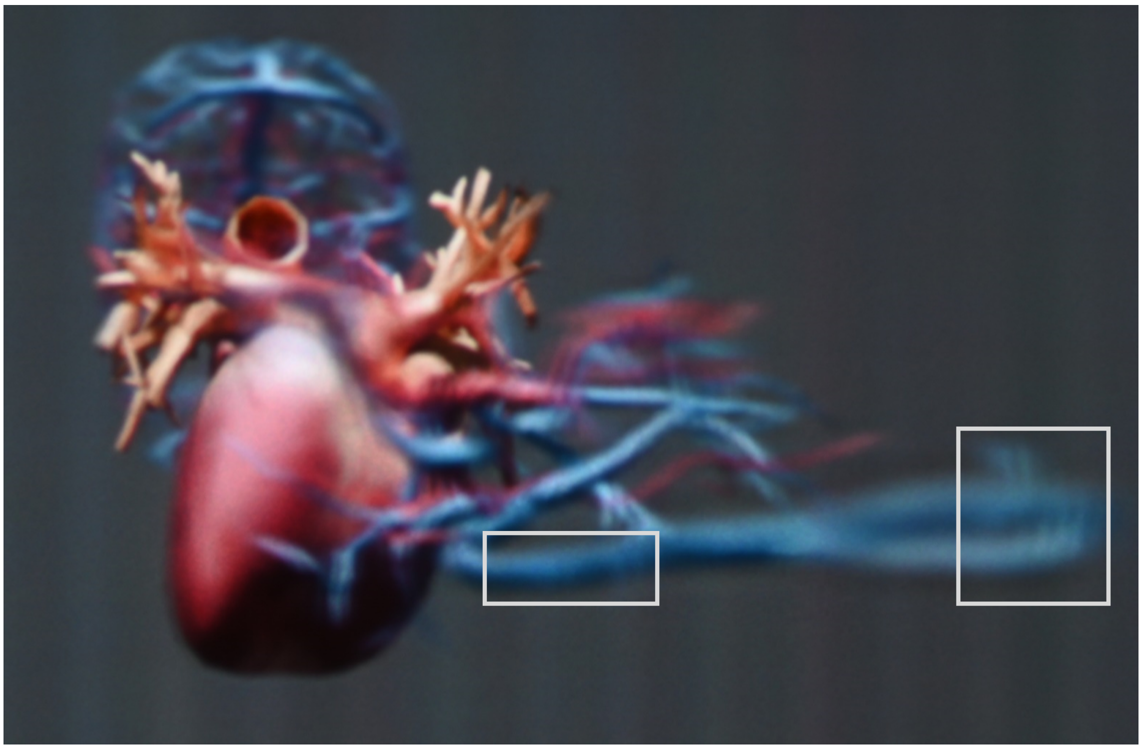
where $E^{(h)}_{s_0,u_0}(t, v_0)$ is the horizontal EPI after applying FIR low-pass filter and $E^{(v)}_{t_0,v_0}(s, u_0)$ is the vertical EPI after filtering the filtered horizontal EPI.

Result

Figure 4.7 shows the result of filtering HPO and FP light field with Gaussian window function by order 52. Filtering HPO light field does not blur dominant horizontal features because there is no information on vertical parallax. However, this issue can be solved by using FP light field, and the final result is blurred naturally. Figure 4.8 illustrates the result of HPO light field being filtered with filter orders 52 and 78 by Gaussian window function. Moreover, Figure 4.9 visualizes the filtering result of HPO light field with Gaussian and Hamming window function by order 52. Although the performance of this method is great, it cannot be utilized in real-time applications because rendering or capturing FP and HPO DSLF is time-consuming. The filtered light field is visualized on Holografika's Holovizio 722RC projection-based light field display.

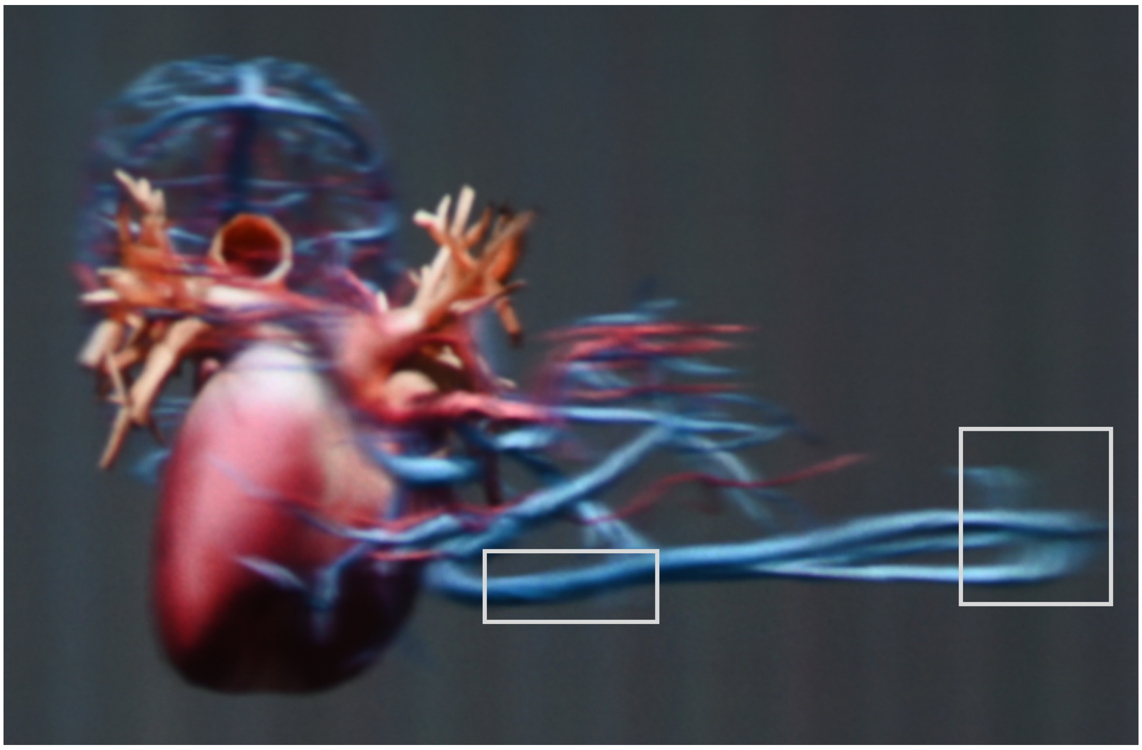


***(a)** Result of filtering HPO light field with Gaussian window function*

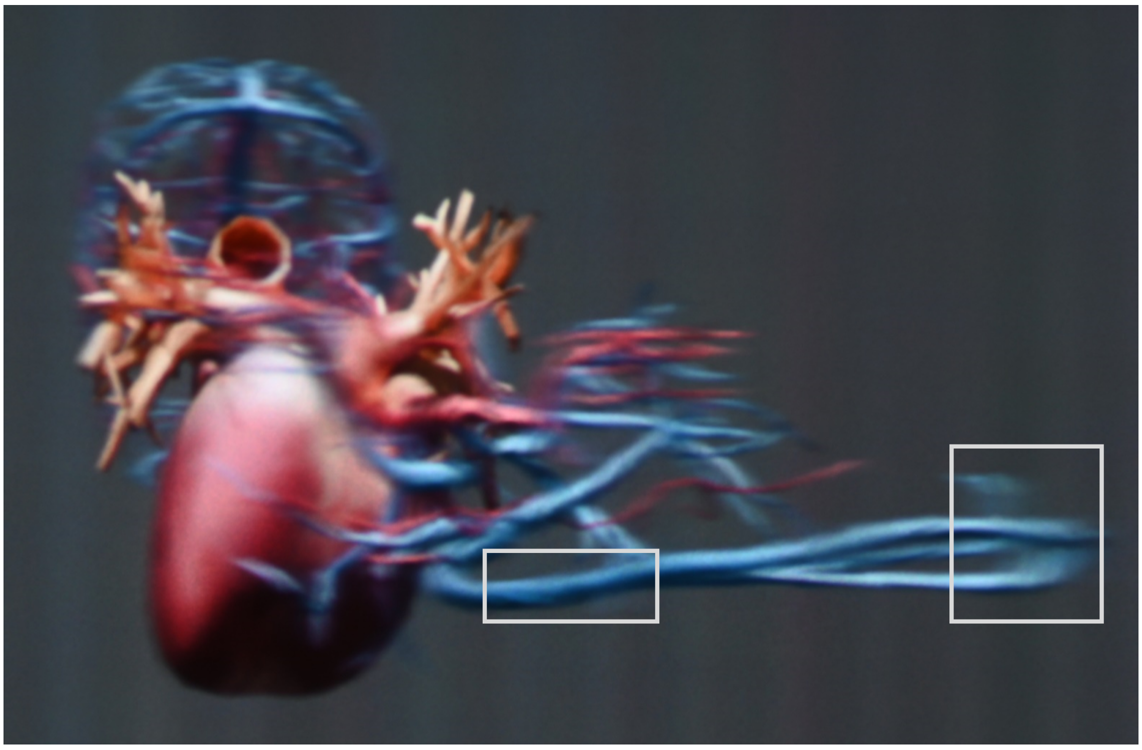


***(b)** Result of filtering FP light field with Gaussian window function*

***Figure 4.7.** The difference in the result of HPO and FP light field after applying 1D FIR low-pass filters with Gaussian window function by order 52.*

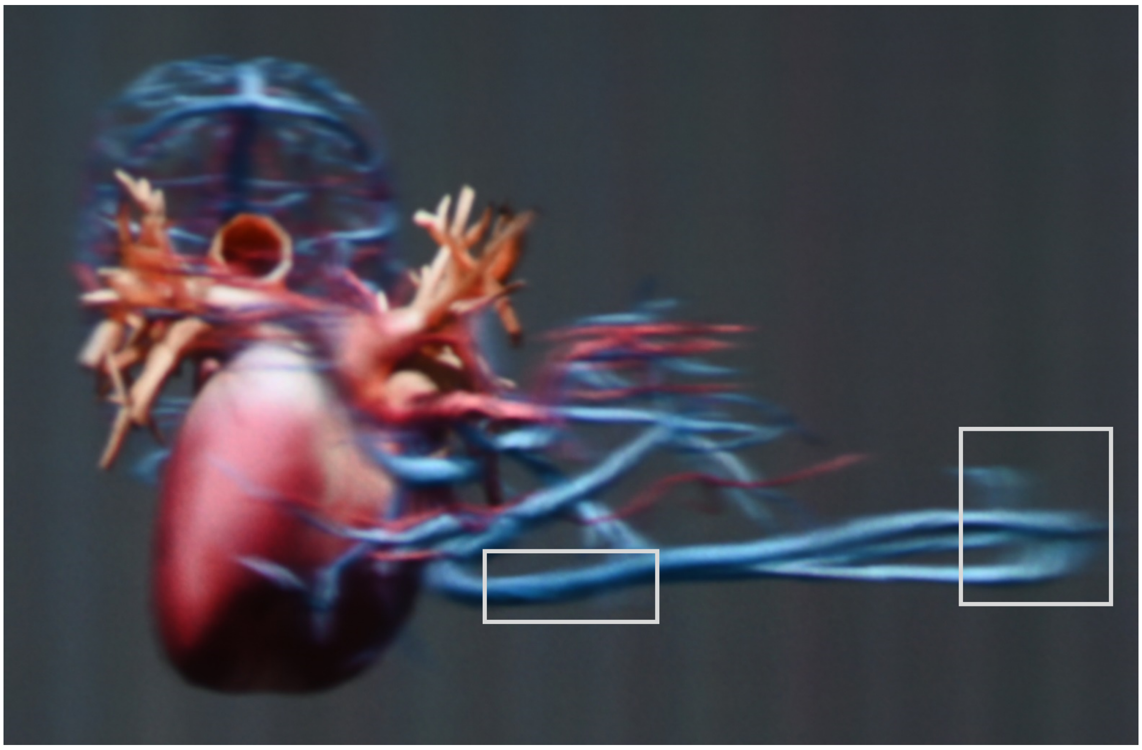


***(a)** Result of filtering HPO light field with filter order 52*

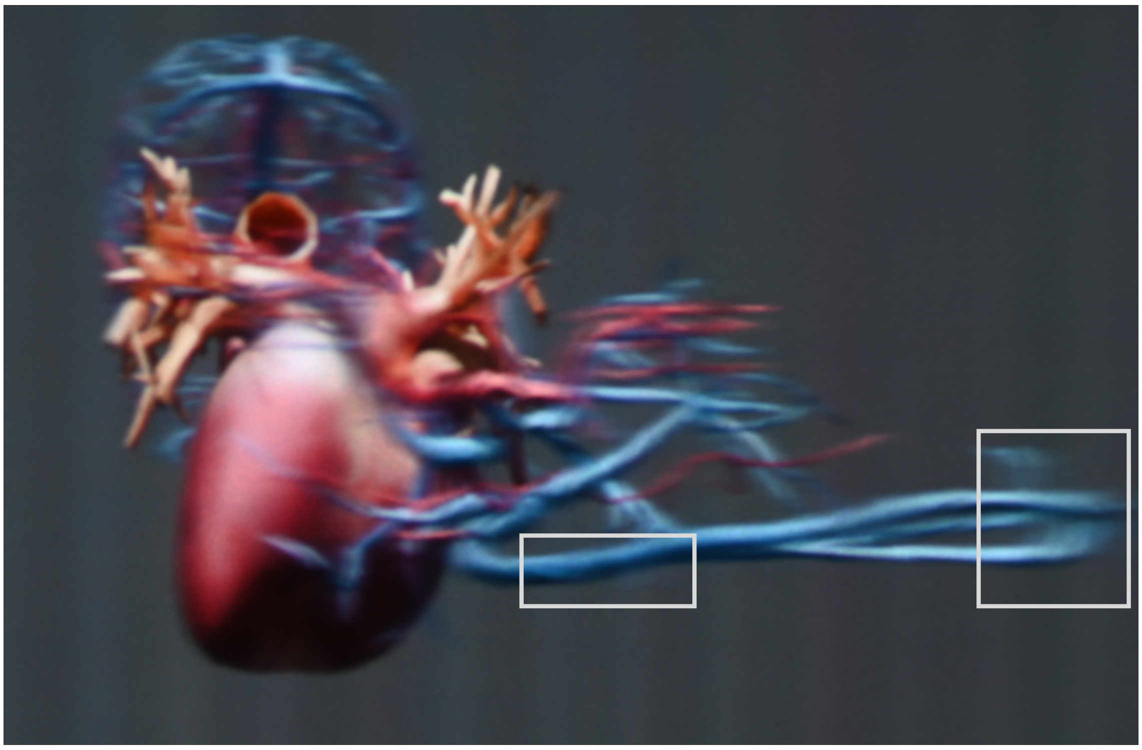


***(b)** Result of filtering HPO light field with filter order 78*

***Figure 4.8.** The result of filtering FP light field with filter orders 52 and 78 by Gaussian window function.*



(a) *Result of filtering HPO light field with Gaussian window function*



(b) *Result of filtering HPO light field with Hamming window function*

Figure 4.9. *The result of filtering HPO light field with Gaussian and Hamming window function by order 52.*

5. REAL-TIME MULTI-LAYER DOF RENDERING WITH TILED SPLATTING

This chapter proposes a real-time DoF rendering solution for eliminating aliasing issues in light field displays. The chosen underlying algorithm is called *multi-layer DoF rendering with tiled splatting* [15]. In this chapter, initially, the relation between the thin lens model utilized in the real-time DoF rendering approach and the display's bandwidth is derived based on Section 4.2. Next, each stage of the algorithm's pipeline is explained, and results of the proposed DoF rendering algorithm are shown. As the aliasing in rasterization affects the quality of DoF rendering, an MSAA step has been added to the pipeline to improve the final quality of DoF rendering. Based on the implemented algorithm, a real-time method is developed to synthesize multiple adjacent views with DoF in order to facilitate light field rendering for light field displays. Finally, limitations of the multi-view synthesis method are discussed.

5.1 Relation between Thin Lens Parameters and Display's Bandwidth

The large aperture can be simulated with arrays of pinhole cameras positioned along the aperture axis as shown in Figure 5.1 in which each pixel in the final viewpoint is the average of ray samples from cameras inside the aperture [1]. The aperture size in DoF rendering algorithms determines the amount of blurriness. Additionally, the decimation factor computed in Section 4.2 refers to the number of cameras positioned inside the aperture. Therefore, the relation between the aperture size and the decimation factor can be expressed as

$$A = K * b, \tag{5.1}$$

where A and b are the aperture size and the distance between adjacent cameras in mm, and K is the decimation factor.

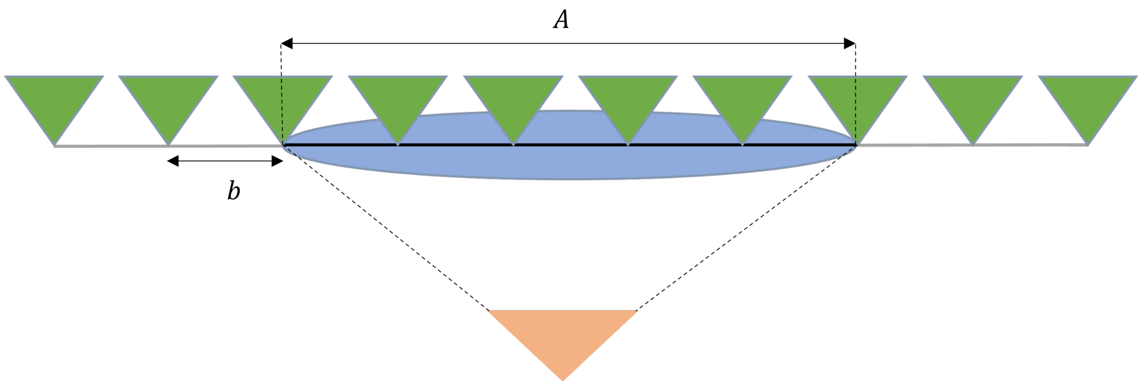


Figure 5.1. *The relation between the aperture size and the decimation factor computed in Section 4.2.*

5.2 Algorithm Pipeline

This section describes the pipeline of multi-layer DoF rendering with a tiled splatting, which generates DoF in real-time without artifacts discussed in Section 3.3, e.g. depth discontinuity and color bleeding. The pipeline stages, explained in the following subsections are: generation, reduction, tiling, sorting and accumulation. Figure 5.2 illustrates the pipeline of the implemented algorithm including the multi-view synthesis and MSAA improvement stages.

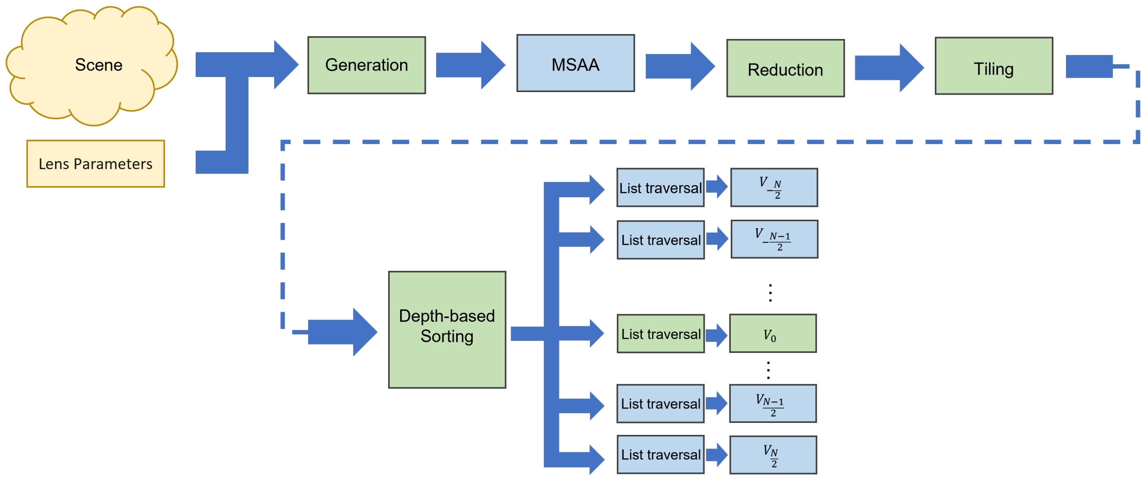


Figure 5.2. *The pipeline of multi-layer DoF rendering with tiled splatting. The original algorithm is depicted in green and the additional stages are shown in blue.*

Multi-Layer Data Structure Generation

The main purpose of this stage is to generate a multi-layer data structure of the scene which stores scene data like depth and color into several layers. This stage can be done in two steps. The first step is to render the multi-layer image by using the depth peeling algorithm as explained in Appendix A.1. The second step is to find areas with depth discontinuity which should be stored in layered textures due to occurrence of partial occlusions. To detect depth discontinuity, a disocclusion map has been used which is explained in Appendix A.3. Figure 5.3 illustrates the result of multi-layering technique using depth peeling, where each layer shows the region occluded by the previous layer.

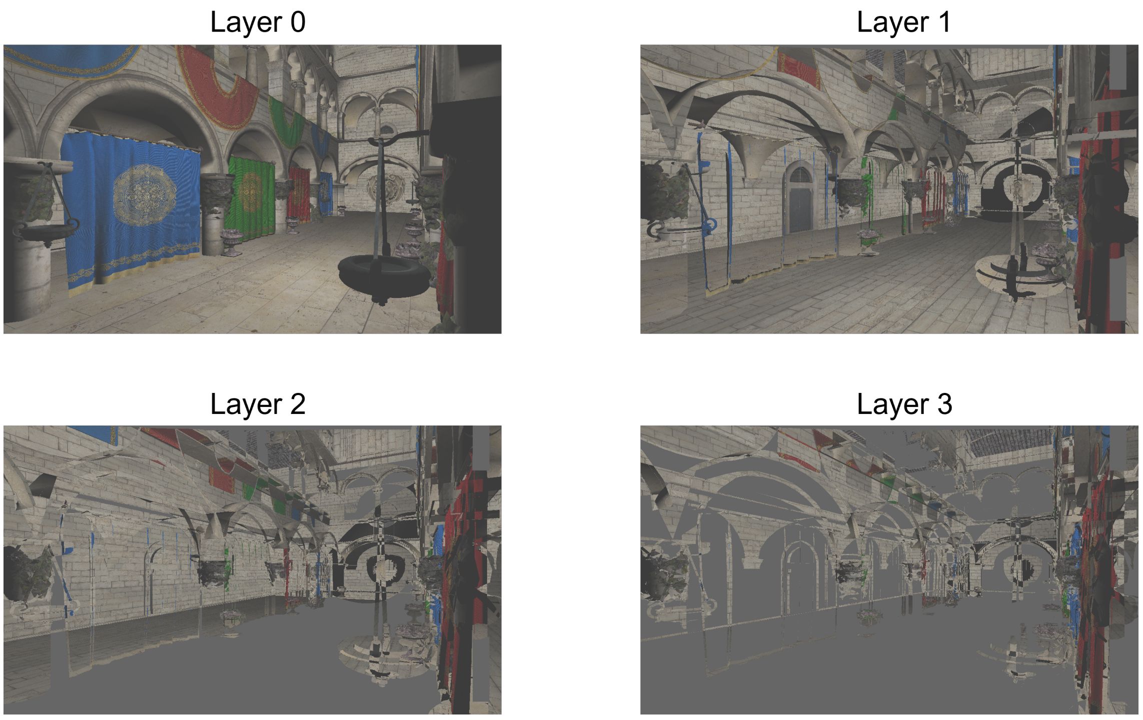


Figure 5.3. *The result of multi-layering method using depth peeling which facilitates the color estimation of occluded areas.*

Fragment Reduction

This stage of the pipeline is optional and its purpose is to decrease the amount of data by merging pixels. According to Figure 5.4, four adjacent pixels are merged if their depth and luminance are similar and their distance to the focus plane is substantial. The CoC of merged pixels can be expressed as

$$r'_{p_i} = \begin{cases} r_{p_i} + \frac{\sqrt{2}}{2} & : \text{merged} \\ r_{p_i} & : \text{otherwise} , \end{cases} \tag{5.2}$$

where r'_{p_i} is the radius of i^{th} pixel's CoC after merging four adjacent pixels which might or might not happen, and r_{p_i} is the radius of i^{th} pixel's CoC before the merge. Afterwards, pixels' information like color, depth, and screen coordinate are stored into an inhomogeneous linked list as described in Appendix A.2. The downside of this method is that the final CoC gets larger and it causes over-blurring [15].

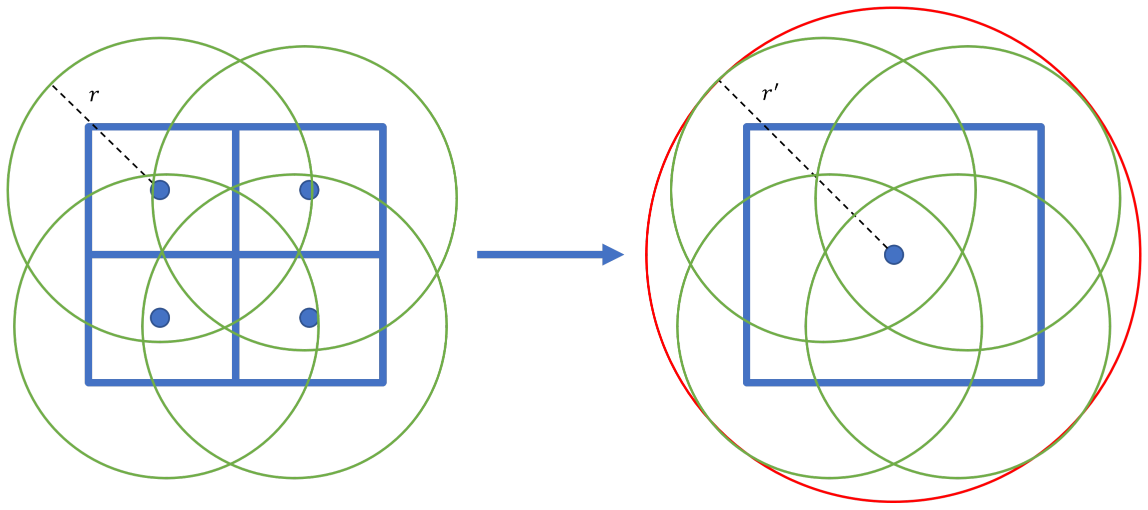


Figure 5.4. *Merging pixels based on their color, depth, and their distance to the focus plane. The red circle in the right image is considered as CoC after merging.*

Tiling and Depth-based Sorting

To compute each pixel's blur more efficiently, the framebuffer is partitioned into tiles. Furthermore, each pixel in the framebuffer is examined with 9 surrounding tiles whether its corresponding CoC overlaps. If the CoC of a pixel overlaps with a tile, a copy of that pixel's information is stored into the tile. Therefore, each tile is a set of pixels which can be expressed as

$$\begin{aligned} t_{n_v,n_u} = \{(p_{i,j}, z_{i,j}) : &|i - N_v \times n_v| < r^c_{i,j} \\ &\vee |j - N_u \times n_u| < r^c_{i,j} \\ &\vee ||(i,j),(n_v,n_u).(N_v,N_u)|| < r^c_{i,j} \\ &\vee (n_v - 1, n_u - 1).(N_v,N_u) < (i,j) < (n_v,n_u).(N_v,N_u)\}, \end{aligned} \tag{5.3}$$

where t_{n_v,n_u} is the $(n_v, n_u)^{th}$ tile and (N_v, N_u) is the total number of tiles in the horizontal and vertical direction. Moreover, $p_{i,j}$ and $z_{i,j}$ are the color and the distance to the camera which is stored in a pixel at the coordinate (i, j) in the framebuffer. Finally, $r^c_{i,j}$ is the radius of CoC at the pixel coordinate (i, j) in the framebuffer with respect to Equation 3.2. This is followed by sorting pixels in each tile based on their distance to the camera that is essential for the next stage. Bitonic sort is the utilized sorting algorithm, which is clarified in Appendix A.4. Figure 5.5 illustrates the partitioning, the overlapping condition check, and the copying procedure into tiles.

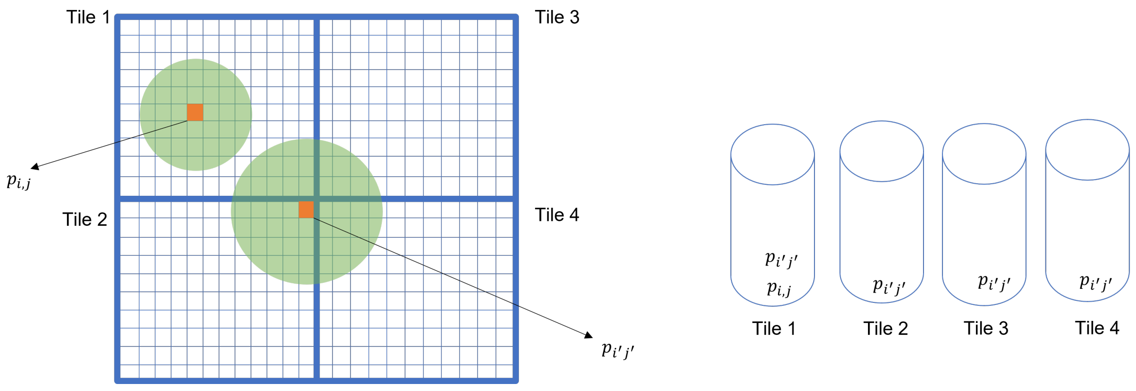


Figure 5.5. *The framebuffer is partitioned into several tiles, and each tile stores a copy of a pixel's information when its CoC overlaps with the tile.*

List Traversal

In the last stage, pixels' information stored in each tile are accumulated with Alpha-blending as discussed in Appendix A.5. To calculate the color of a pixel at the coordinate (i, j) in the framebuffer, the tile in which the pixel is located need to be computed as below

$$(n_v, n_u) = \left\lfloor \frac{(i, j)}{(N_v, N_u)} \right\rfloor , \tag{5.4}$$

where (n_v, n_u) is the location of a tile at which the pixel with coordinate (i, j) belongs to. Next, the distance of the coordinate (i, j) to each pixel's coordinate (i', j') in the tile t_{n_v,n_u} is computed and compared with their CoC radius. If the calculated distance is less than the CoC radius, the pixel's color is used in Alpha-blending and its respecting weight is

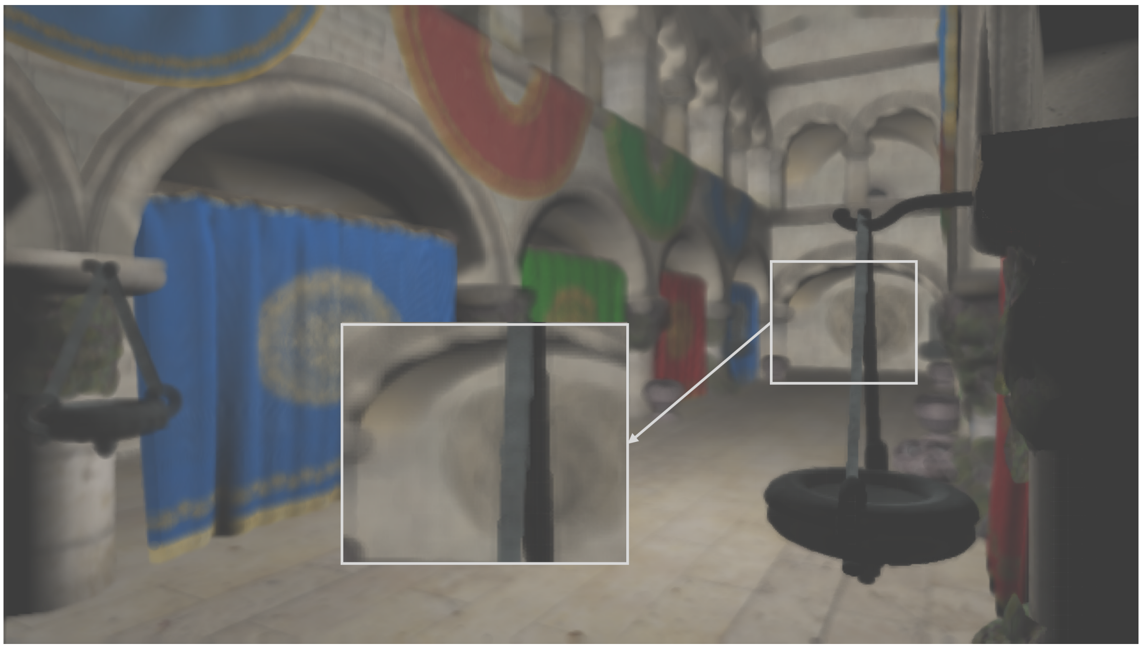
$$P = \{p_{i',j'} : ||(i, j), (i', j')|| < r^c_{i',j'})\} \tag{5.5}$$

$$\gamma_{i',j'} = \begin{cases} \sum_{k=0}^{3}(1 - \frac{1}{(r^c_{i',j'})^2})^k & : \text{ merged} \\ 1 & : \text{ otherwise} , \end{cases} \tag{5.6}$$

where $\gamma_{i',j'}$ is the weight of a pixel at coordinate (i', j'), which depends on the pixel's CoC radius and the fact of being merged. Here, P is a set of pixels which are used in the Alpha-blending algorithm.

Result

Figure 5.6 shows the result of multi-layer DoF rendering with tiled splatting. In Figure 5.6 (a), the focus is in front and there is no color bleeding from focused area into out of focus region. The focus in Figure 5.6 (b) is in the rear region and the out of focus area in the front is blurred and semi-transparent without depth discontinuity artifact.


(a) *The Sponza scene with DoF in front.*

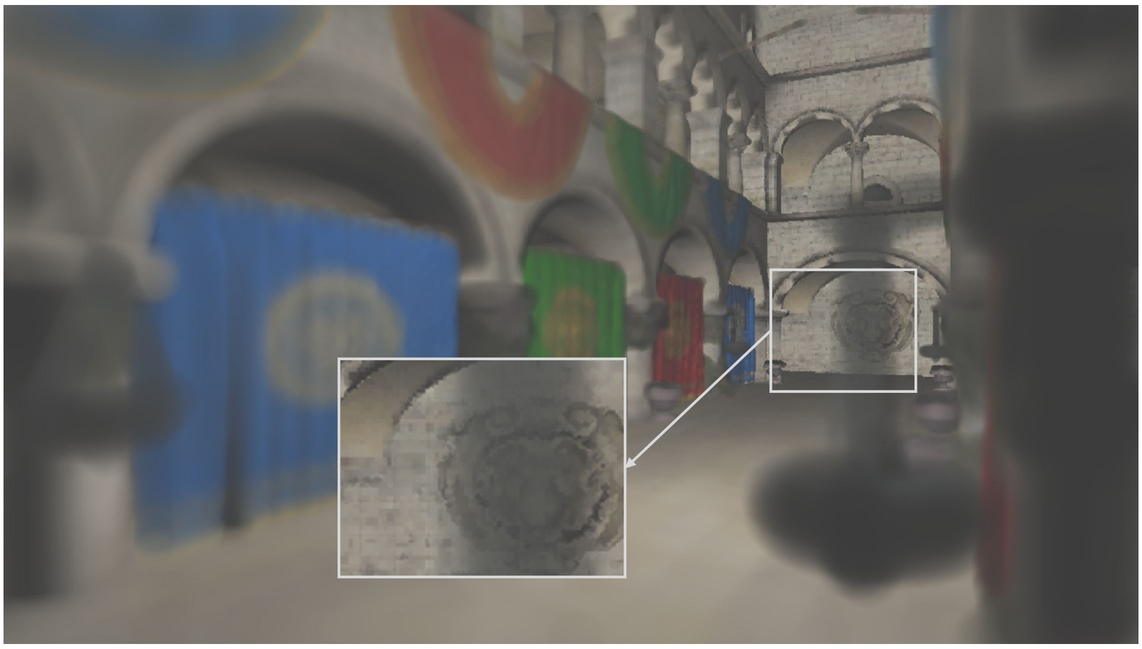
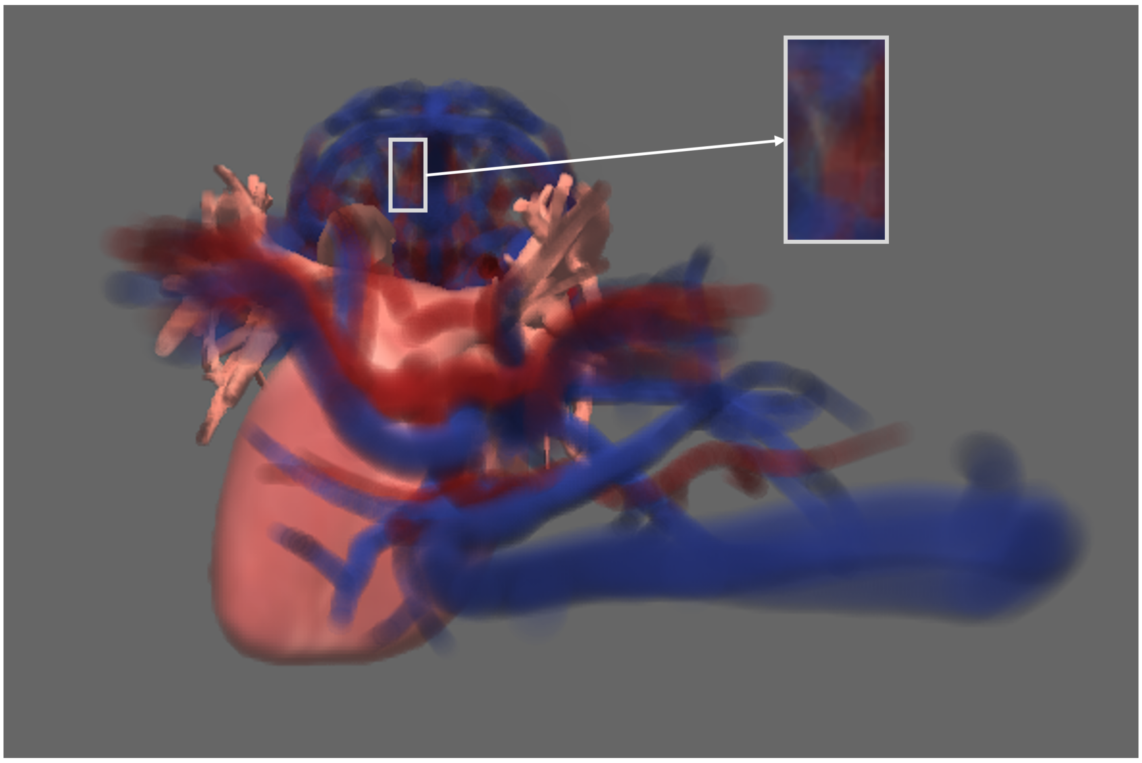

(b) *The Sponza scene with DoF in back.*

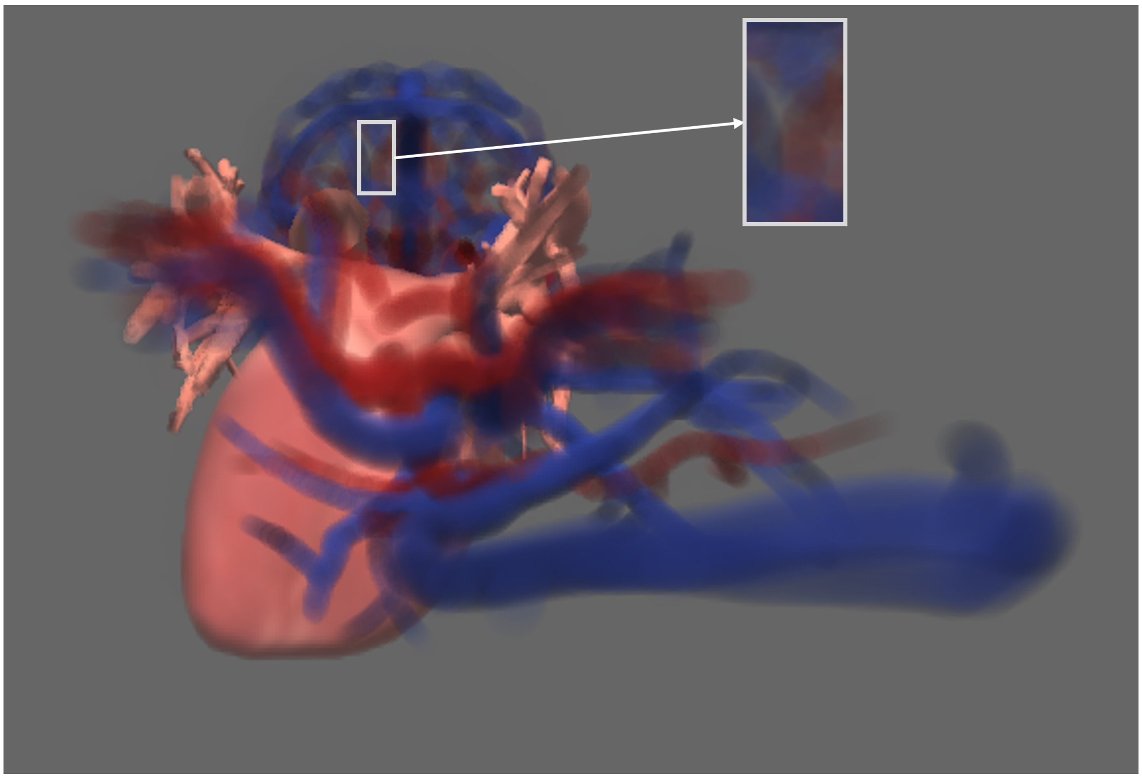
Figure 5.6. *The result of multi-layer DoF rendering with tiled splatting.*

5.3 Improvement with MSAA

The multi-layering technique in the first stage of the pipeline, generation, utilizes rasterization [52]. The rendering result of rasterization may have jagged edges which can influence the DoF rendering result. Therefore, rasterization could cause aliasing issue, which can be smoothed with anti-aliasing methods. MSAA is the anti-aliasing method we have used because it is simple, fast, and implemented in OpenGL. More explanation about MSAA is given in Appendix A.6. Figure 5.7 (a) shows the DoF rendering result without applying MSAA where the color of the specified area is not blurred properly. However, the same area in Figure 5.7 (b) is blurred more due to utilization of MSAA.



(a) *DoF rendering without MSAA.*



***(b)** DoF rendering with MSAA (8 samples per pixel).*

***Figure 5.7.** The comparison between applying MSAA or not in multi-layer DoF rendering with tiled splatting.*

5.4 Novel View Synthesis

The real-time DoF rendering algorithm should take into account light field display's bandwidth in order to avoid aliasing. Moreover, content visualization on light field displays requires properly pre-processed light field. Therefore, proposing a method that renders simultaneously several adjacent views following display's bandwidth will facilitate light field preparation. To synthesize novel views, first, the main view V_0 is rendered, and the recentered disparity between each novel view V_i and the main view V_0 is computed based on the distance between them and their distance to the focus plane. Next, alpha-blending in the list traversal stage is used to determine the blur and render adjacent views with same DoF as mentioned in Section 5.2, which follows display's bandwidth. Listing 1 describes that each novel view is synthesized based on alpha blending in the list traversal stage. Figure 5.8 illustrates that each novel view is generated from the main view V_0.

Algorithm 1 View Synthesis Algorithm

```
procedure ViewSynthesis($b$, $f$, $P$, $p$)
    for $p_i$ in $P$ do
        set $disp$ to $\frac{b \times f}{p_i.z} - \frac{b \times f}{z_{focus}}$
        set $p_i.uv$ to $p_i.uv + (disp, 0)$
        if $||p_i.uv - p.uv|| <$ COC($p_i.z$, $f$) then
            do Alpha Blending
        end if
    end for
end procedure
procedure MAIN($N$)
    set $baseline$ to $b$
    set $focal$ to $f$
    $P$ is all pixels in the tile
    $p$ is the pixel, color should be evaluated.
    for $i \leftarrow 0, n$ do
        set d to $(i - \frac{N}{2}) \times baseline$
        ViewSynthesis($d$, $f$, $P$, $p$)
    end for
end procedure
```

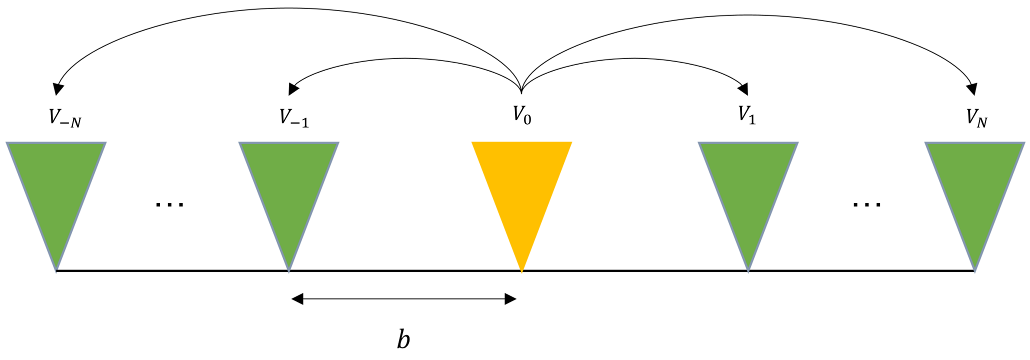


Figure 5.8. *The generation of novel views according to the middle view produced by DoF rendering algorithm.*

5.5 Result and Limitations

In the multi-view DoF rendering mentioned in Chapter 4, DSLF is required to apply depth-based anti-aliasing filters for blurring and removing content that cannot be reproduced by the light field display. Therefore, multi-view DoF rendering using DSLF is slow and time-consuming. Furthermore, the proposed real-time approach is fast due to the usage of GPU-based multi-layering technique for solving depth discontinuity in DoF rendering. However, the computed CoC for each pixel is limited because tiling in GPU uses shared memory which its size is restricted. Thus, some features will not be blurred or removed thoroughly.

Adjacent view synthesis in a real-time DoF rendering approach would decrease the light field generation workload. Figure 5.9(a) shows a synthesized view with the implemented method and compares it with Figure 5.9(b), a view rendered without additional synthesis stage. Deriving from Figure 5.9, there is no visible difference between them. In this case, the distance between the synthesized view V_1 and the main view V_0 is 32-mm.

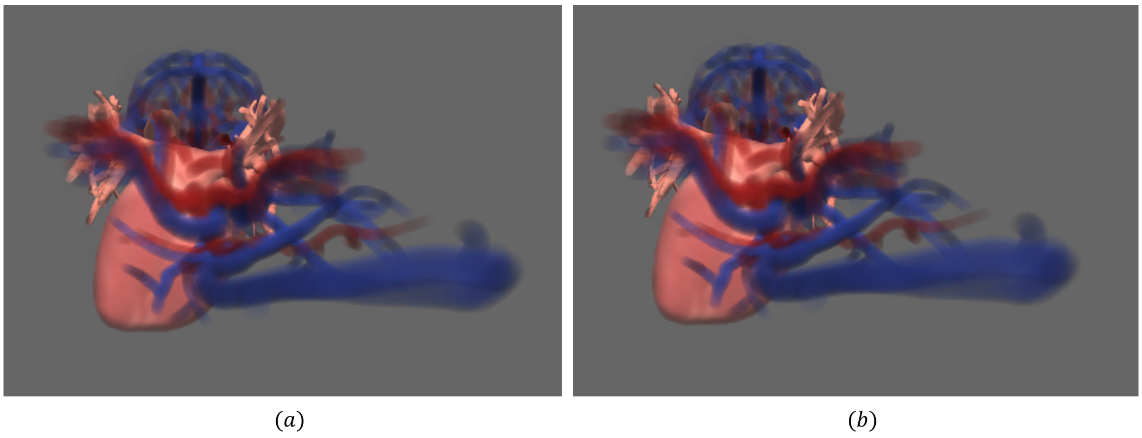


Figure 5.9. *(a) The synthesized view V_1 with 32-mm distance from the main view V_0. (b) The view rendered without utilization of synthesis stage.*

Figure 5.10 shows the relation between the time and the number of rendered views for the implemented real-time DoF rendering method with and without multi view synthesis stage in the algorithm's pipeline. Consequently, adding multi-view synthesis in the pipeline decrease the rendering time in compare with a pipeline without this stage.

The view synthesis limitation is the maximum distance between the generated view and the main view, which should be less than or equal to half of the aperture size. In order to render a novel view without artifacts, the amount of disparity must be less than CoC at all distances from the camera sensor, as illustrated in Figure 5.11. Otherwise, the alpha-blending generates novel views with incorrect pixels' color which creates artifacts.

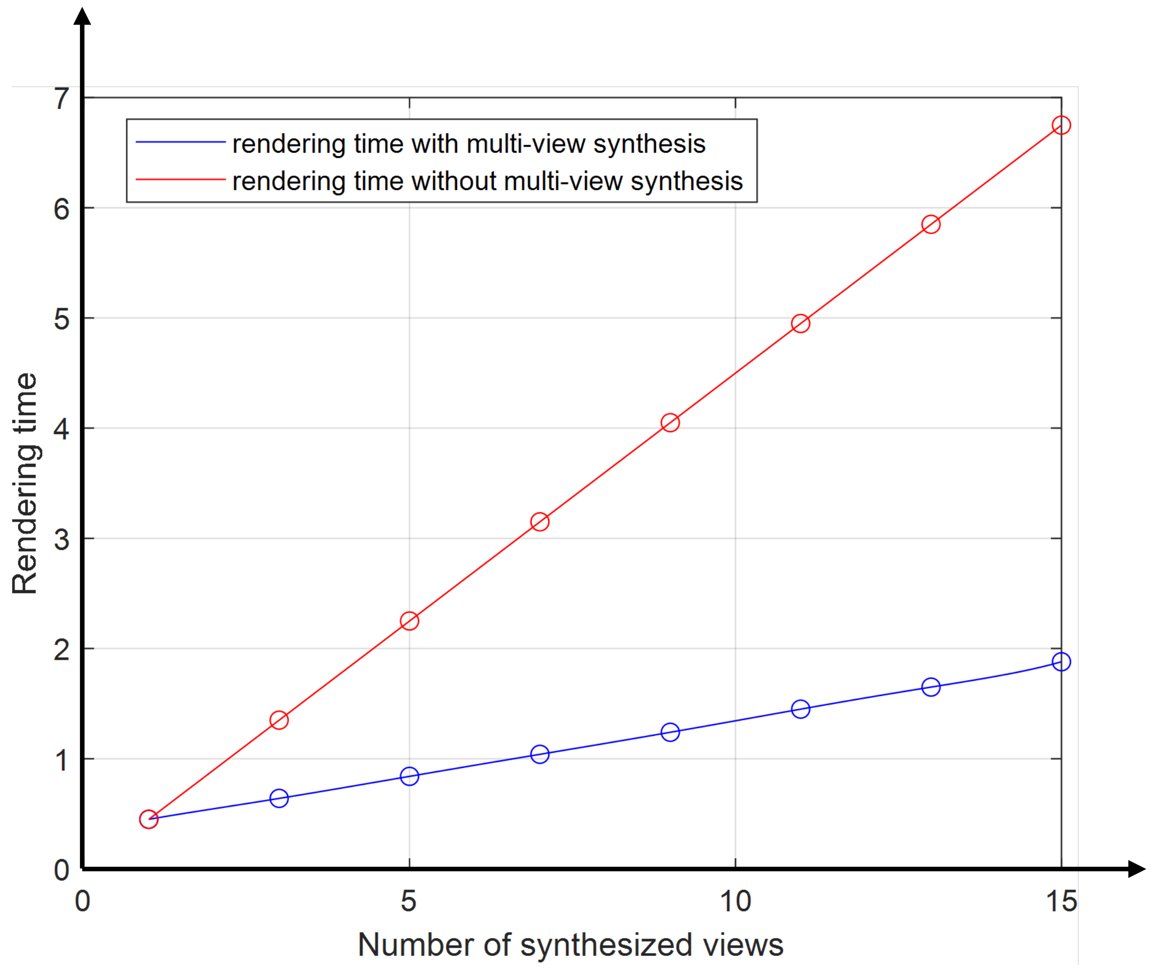


Figure 5.10. *The relation between the time and the number of rendered views with and without the synthesis stage*

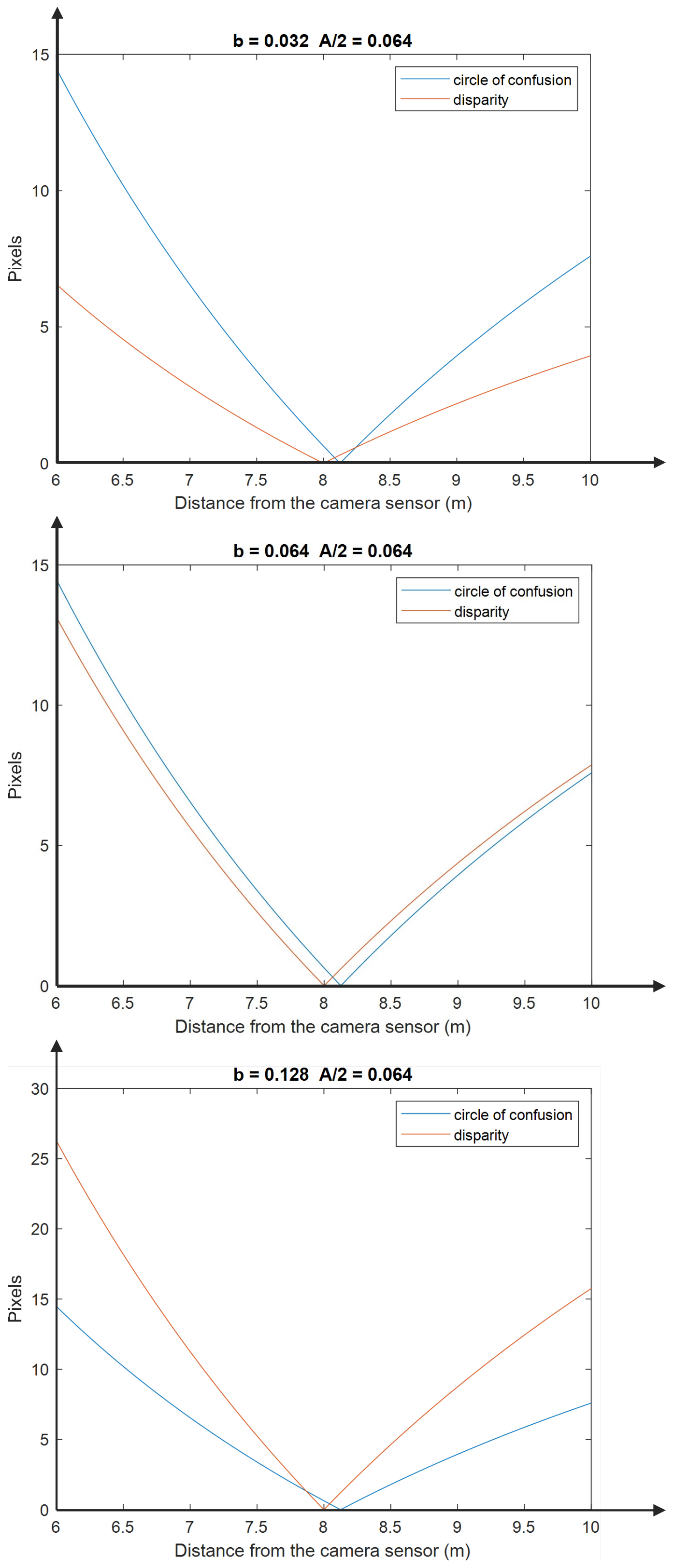


Figure 5.11. *The CoC and disparity graph with baseline = 0.032, 0.064, and 0.128 mm.*

6. CONCLUSIONS AND FUTURE WORK

Light field displays have limited DoF, which means the display's spatial resolution would decrease by increasing the viewing distance. As a result, displays cannot show content with too small details at a far distance from the screen properly and aliasing occurs. To mitigate this issue, this thesis proposed three DoF rendering algorithms. The first two methods uses DSLF rendering and depth-based anti-aliasing filters, and the final method adopts real-time multi-layered DoF rendering using tiled splatting. Furthermore, the concepts of DoF in the thin lens model and DoF rendering in computer graphics have been reviewed, and the analogy between the DoF in the thin lens model and light field displays has been investigated.

In the first two methods, an FP and HPO DSLF are rendered. Afterward, a resampling factor for matching the spatial and angular frequency of the rendered DSLF and the display is evaluated. For the first method, a 2D uniform circular filter is designed whose radius corresponds to the resampling factor, and it is applied to a local neighborhood of each image in the FP DSLF. In the second method, several FIR filters are designed and applied to the EPI's spatial domain obtained from HPO or FP DSLF.

The final implemented method is real-time multi-layer DoF rendering with tiled splatting, which is appropriate for real-time applications. This thesis has established a connection between lens parameters in this method and the display's bandwidth. Also, the rendering result of this algorithm is improved by MSAA. To facilitate light field rendering in the proposed real-time method, a stage is added to the main algorithm's pipeline for simultaneously synthesize several adjacent views. However, the baseline size in the light field rendering is limited and this limitation is estimated in this thesis.

The real-time method proposed in this thesis is only suitable for synthetic scenes. Thus, one option for future studies is to develop a real-time anti-aliasing algorithm for light field displays that can be applied on real-world scenes. Since DSLF has been used to remove the aliasing in the first two methods, the other option for further research is to use a sparsely sampled light field instead of DSLF as the input of depth-based anti-aliasing filtering methods. Furthermore, a user study can be done on human participants to comprehend which result is preferable for them, e.g. whether they prefer over-blurred scene or a scene with small aliasing and more visible details.

A. APPENDIX

In the Appendix, several algorithms and data structures that are implemented in the multi-layer DoF rendering using tiled splatting are overviewed. These algorithms and data structures are parallelized GPU implementations.

A.1 Depth Peeling

In the depth-peeling algorithm, the frame buffer's render target is a texture array in which the geometry shader multiplies each geometrical primitive to the number of rendering layers. Afterward, each fragment is rendered into the respective layer of the texture array corresponding to its distance from the camera origin [53], [54]. Listing 2 and 3 demonstrate the pseudo-code of the single-pass and multi-pass depth-peeling algorithm.

Algorithm 2 Multi-pass depth peeling algorithm

```
procedure GEOMETRYSHADER($tri$)                              ▷ First shader pass
    emit $T_t(tri)$ to $layer0$
end procedure
procedure FRAGMENTSHADER($x, y, z$)
    return $S(x, y, z)$
end procedure
procedure GEOMETRYSHADER($tri$)                              ▷ Second shader pass
    emit $T_t(tri)$ to $layer1$
end procedure
procedure FRAGMENTSHADER($x, y, z$)
    if $z > Z_t[0][x, y, z] + \Delta z$ then
        return $S(x, y, z)$
    else
        discard the fragment
    end if
end procedure
```

Algorithm 3 Single-pass depth peeling algorithm

```
procedure GEOMETRYSHADER($tri$)
    for $i \leftarrow 1, n$ do
        emit $T_t(tri)$ to $layer_i$
    end for
end procedure
procedure FRAGMENTSHADER($x, y, z$)
    if $layer_{id} == 0$ then
        return $S(x, y, z)$
    else
        $L \leftarrow layer_{id} - 1$                              ▷ Previous layer
        if Previous then
            $C \leftarrow (x, y, z)$                               ▷ Comparison texel
        else if Reproject then
            $C \leftarrow (x_{t-1}, y_{t-1}, z_{t-1})$             ▷ Comparison texel
        end if
        if $z_C > Z_t[L][x, y, z] + \Delta z$ then
            return $S(x, y, z)$
        else
            discard the fragment
        end if
    end if
end procedure
```

A.2 Real-time Non Homogeneous Linked List

In Graphics Processing Units (GPU), dynamic memory allocation is not as trivial as it is in Central Processing Units (CPU). Therefore, the generation of dynamic data structures is different in GPU. For instance, to implement a 2D linked list in GPU by utilizing OpenGL API [52], [55], two buffers are necessary to be initialized. The first buffer corresponds to data in the linked list and the second buffer tracks the number of elements in each linked list node as illustrated in Figure A.1. The size of the first buffer is equal to the number of nodes multiplied by the maximum element numbers each node can have [56].

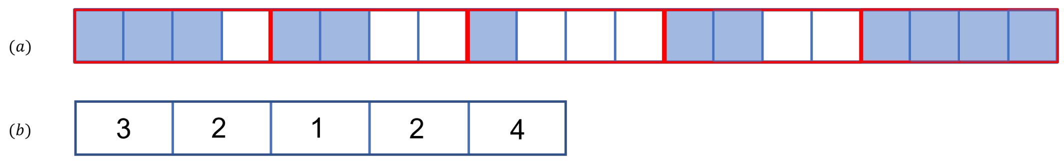


Figure A.1. *The real-time non-homogeneous linked list generation in GPU. (a) The data buffer initialized with maximum buffer size 20 with 5 nodes and each one can have maximum 4 elements. (b) The buffer that tracks the number of elements in each linked list node.*

A.3 Disocclusion Map

The disocclusion map is a texture showing areas where depth discontinuity occurs, and it is computed by thresholding the depth map or applying high-pass filters like Sobel and Laplacian. The Figure A.2 depicts the disocclusion map acquired from the depth map.

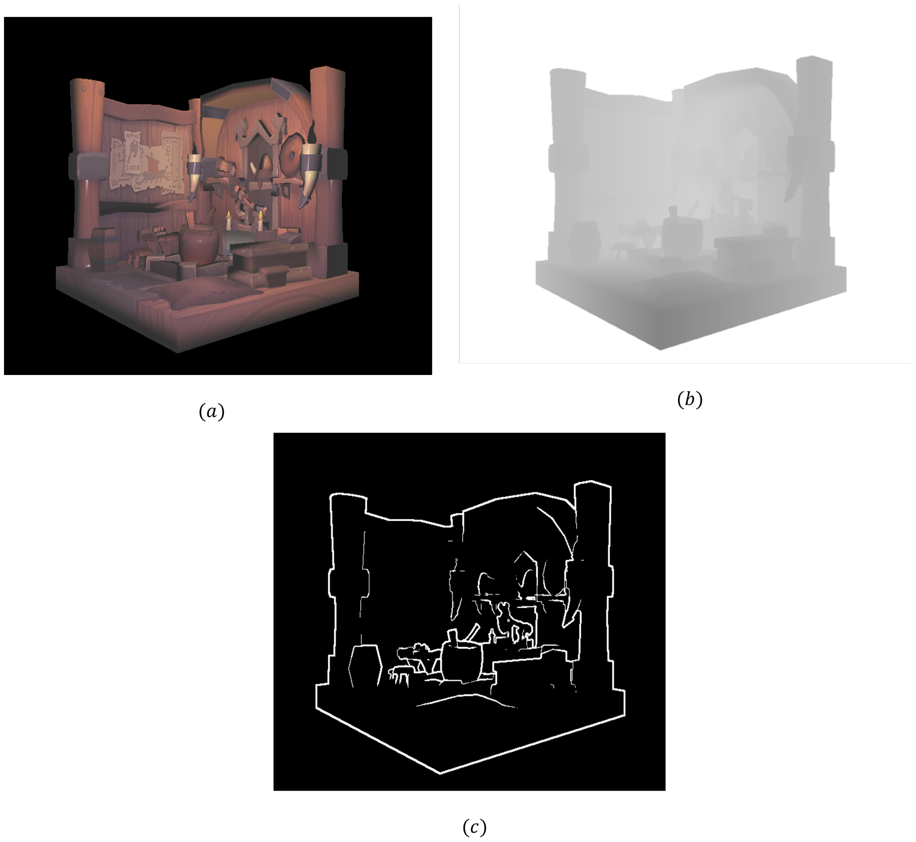


Figure A.2. *(a) Color image and (b) the depth map of the rendered scene. (c) Disocclusion map obtained from the depth map.*

A.4 Bitonic Sort

Bitonic sort is a sorting algorithm with $O(n \log^2(n))$ complexity order, which is suitable for parallel implementation [57]. Moreover, the number of elements in Bitonic sort should be 2^n; otherwise, the algorithm will fail. In practice, the set can be extended with zero elements in order to reach a set with 2^n size [57]. Listing 4 and 5 show the Bitonic sort algorithm.

Algorithm 4 Bitonic Sort

```
procedure BITONIC-SORT(arr[], lowIndex, count, direction)
    if count > 1 then
        k ← (count / 2)
        bitonic-sort(arr[], lowIndex, k, 1)
        bitonic-sort(arr[], lowIndex, lowIndex + k, 1)
        bitonic-merge(arr[], lowIndex, count, direction)
    end if
end procedure
```

Algorithm 5 Bitonic Merge

```
procedure BITONIC-MERGE(arr[], lowIndex, count, direction)
    if count > 1 then
        k ← (count / 2)
        for i ← lowIndex, lowIndex + k do
            if direction == (arr[i] > arr[i + k]) then
                swap arr[i] with arr[i + k]
            end if
        end for
        bitonic-merge(arr[], lowIndex, k, direction)
        bitonic-merge(arr[], lowIndex + k, k, direction)
    end if
end procedure
```

A.5 Alpha Blending

Alpha blending is an algorithm useful for merging several pixels' color based on their opacity to create transparent or semi-transparent appearance [58], [59]. Alpha blending for n fragments can be formulated as,

$$A_n = \begin{cases} \alpha_n \prod_{i=1}^{n-1}(1-\alpha_i) & : i \geq 1 \\ 1 & : i = 0 \end{cases} \tag{A.1}$$

$$C_n = C_{n-1} + A_n c_n \tag{A.2}$$

$$C_n = c_0 + \sum_{j=1}^{n} \alpha_j \prod_{i=1}^{j-1}(1-\alpha_i)c_j, \tag{A.3}$$

where Equation A.3 computes the color C_n after blending with each fragment's color c_j, which is multiplied by the weight of all prior fragments $\alpha_i, 1 \leq i \leq j$. The Listing 6 illustrates the alpha blending procedure.

Algorithm 6 Alpha Blending

```
procedure ALPHA-BLENDING(frag[], count)
    $\alpha_{dest} \leftarrow 1.0$
    for i $\leftarrow$ 0, count do
        color $\leftarrow$ frag[i].rgb
        $\alpha \leftarrow$ frag[i].a
        $\alpha \leftarrow \alpha_{dest} \times \alpha$
        color $\leftarrow \alpha \times$ color + color
        $\alpha_{dest} \leftarrow \alpha_{dest} - \alpha$
    end for
end procedure
```

A.6 Multi-sample Anti-Aliasing

Multi-sample anti-aliasing (MSAA) is a spatial anti-aliasing technique, which improves the rasterization rendering result. Each pixel in rasteriztion has a single sampling point defining the final pixel color value. Therefore, aliasing may occur if the spatial sampling resolution is not enough for rendering small details [60]. However, MSAA uses multiple sampling points per pixel. Each sampling point can have a different color value, and the final pixel color is evaluated as the average of all sampling points [60]. The Figure A.3 illustrates the difference between a pixel color with a single sampling point and several sampling points. Also, Figure A.4 shows the rendering result before and after applying MSAA, where jagged lines in rasterization get smooth. In OpenGL, MSAA can be implemented by using multi-sample textures and frame buffers [52].

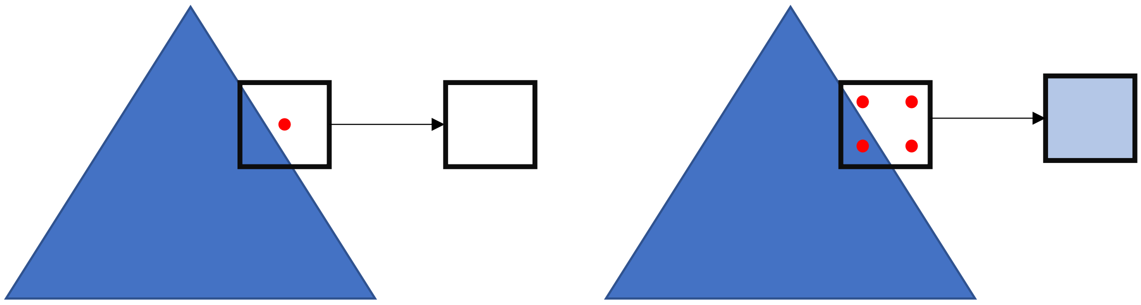
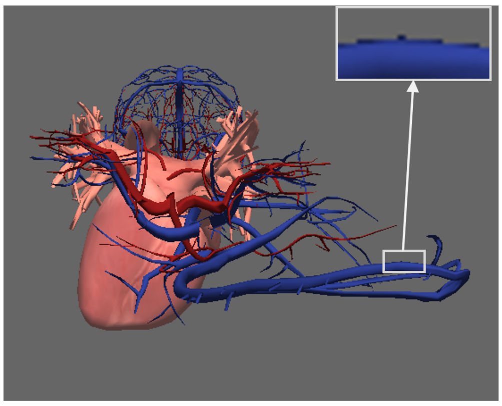


Figure A.3. *The pixel color after rasterization with (a) single sampling point and (b) 4 sampling points per pixel.*


(a)

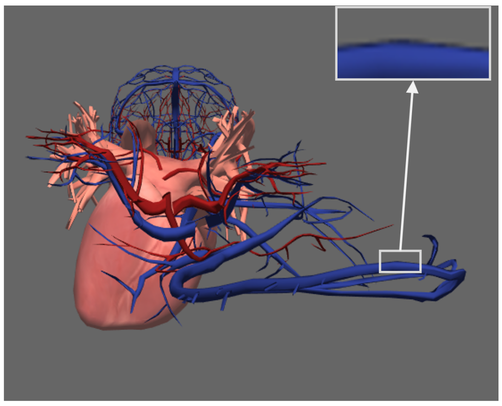

(b)

Figure A.4. *The rasterization rendering result (a) without MSAA and (b) with MSAA (8 samples per pixel). The jagged edges have disappeared after using MSAA.*

REFERENCES

[1] M. Levoy and P. Hanrahan, “Light field rendering,” *ACM*, vol. 30, Sep. 2002.

[2] E. H. Adelson and J. R. Bergen, “The plenoptic function and the elements of early vision”, *Computational Models of Visual Processing*, pp. 3–20, 1991.

[3] N. Salvaggio, *Basic Photographic Materials and Processes*. Taylor & Francis, 2009, 480 pp.

[4] T. Balogh, P. T. Kovács, Z. Dobrányi, *et al.*, “The holovizio system - new opportunity offered by 3d displays,” *D Display*, p. 11, Apr. 2008.

[5] M. Zwicker, W. Matusik, F. Durand, H. Pfister, and C. Forlines, “Antialiasing for automultiscopic 3d displays”, in *ACM SIGGRAPH 2006 Sketches on - SIGGRAPH ’06*, Boston, Massachusetts: ACM Press, 2006, p. 107.

[6] P. Wang, S. Xie, X. Sang, *et al.*, “A large depth of field frontal multi-projection three-dimensional display with uniform light field distribution”, *Optics Communications*, vol. 354, pp. 321–329, Nov. 2015.

[7] C.-G. Luo, X. Xiao, M. Martínez-Corral, C.-W. Chen, B. Javidi, and Q.-H. Wang, “Analysis of the depth of field of integral imaging displays based on wave optics”, *Optics Express*, vol. 21, no. 25, p. 31 263, Dec. 2013.

[8] R. L. Cook, T. Porter, and L. Carpenter, “Distributed ray tracing,” *ACM SIGGRAPH Computer Graphics*, vol. 18, no. 3, pp. 137–145, Jan. 1984.

[9] P. Haeberli and K. Akeley, “The accumulation buffer: Hardware support for high-quality rendering,” in *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH ’90, New York, NY, USA: Association for Computing Machinery, Sep. 1990, pp. 309–318.

[10] A. Isaksen, L. McMillan, and S. J. Gortler, “Dynamically reparameterized light fields”, in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH ’00*, Not Known: ACM Press, 2000, pp. 297–306.

[11] X. Yu, R. Wang, and J. Yu, “Real-time depth of field rendering via dynamic light field generation and filtering”, *Computer Graphics Forum*, vol. 29, no. 7, pp. 2099–2107, Sep. 2010.

[12] “Chapter 28. practical post-process depth of field,” NVIDIA Developer. ().

[13] J. Mulder and R. van Liere, “Fast perception-based depth of field rendering,” Jan. 2000, pp. 129–133.

[14] M. Potmesil and I. Chakravarty, “A lens and aperture camera model for synthetic image generation,” *ACM SIGGRAPH Computer Graphics*, vol. 15, no. 3, pp. 297–305, Aug. 1981.

[15] L. Franke, N. Hofmann, M. Stamminger, and K. Selgrad, “Multi-layer depth of field rendering with tiled splatting,” *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 1, no. 1, 6:1–6:17, Jul. 2018.

[16] S. Lee, E. Eisemann, and H.-P. Seidel, “Depth-of-field rendering with multiview synthesis,” *ACM Transactions on Graphics*, vol. 28, no. 5, pp. 1–6, Dec. 2009.

[17] K. Selgrad, C. Reintges, D. Penk, P. Wagner, and M. Stamminger, “Real-time depth of field using multi-layer filtering,” in *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games*, ser. i3D ’15, New York, NY, USA: Association for Computing Machinery, Feb. 2015, pp. 121–127.

[18] P. T. Kovács, R. Bregović, A. Boev, A. Barsi, and A. Gotchev, “Quantifying spatial and angular resolution of light-field 3-d displays,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 7, pp. 1213–1222, Oct. 2017, Conference Name: IEEE Journal of Selected Topics in Signal Processing.

[19] R. C. Bolles, H. H. Baker, and D. H. Marimont, “Epipolar-plane image analysis: An approach to determining structure from motion”, *International Journal of Computer Vision*, vol. 1, no. 1, pp. 7–55, Mar. 1987.

[20] R. Bregovic, E. Sahin, S. Vagharshakyan, and A. Gotchev, *Signal Processing Methods for Light Field Displays*. Springer, 2019.

[21] R. Bregović, P. T. Kovács, and A. Gotchev, “Optimization of light field display-camera configuration based on display properties in spectral domain,” *Optics Express*, vol. 24, no. 3, pp. 3067–3088, Feb. 2016, Publisher: Optica Publishing Group.

[22] B. Barsky and T. Kosloff, “Algorithms for rendering depth of field effects in computer graphics,” Jan. 2008.

[23] A. Gershun, P. H. Moon, and G. Timoshenko, *The light field*. Cambridge, Mass.: Massachusetts Institute of Technology, 1939, OCLC: 38645018.

[24] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, “The lumigraph,” in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH ’96, New York, NY, USA: Association for Computing Machinery, Aug. 1996, pp. 43–54.

[25] R. Bregovic, P. Kovács, T. Balogh, and A. Gotchev, “Display-specific light-field analysis,” Jun. 2014, p. 911 710.

[26] H. Zhu, Q. Wang, and J. Yu, “Light field imaging: Models, calibrations, reconstructions, and applications”, *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 9, pp. 1236–1249, Sep. 2017.

[27] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* West Nyack: Cambridge University Press, 2004, OCLC: 1044713766.

[28] J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum, “Plenoptic sampling,” in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH ’00, USA: ACM Press/Addison-Wesley Publishing Co., Jul. 2000, pp. 307–318.

[29] C. Zhang and T. Chen, “Spectral analysis for sampling image-based rendering data,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, pp. 1038–1050, Dec. 2003.

[30] T. Suzuki, K. Takahashi, and T. Fujii, “Sheared EPI analysis for disparity estimation from light fields,” *IEICE Transactions on Information and Systems*, vol. E100.D, no. 9, pp. 1984–1993, 2017.

[31] G. Lippmann, “Epreuves reversibles. photographies integrals”, *undefined*, 1908.

[32] Y. Takaki, K. Tanaka, and J. Nakamura, “Super multi-view display with a lower resolution flat-panel display,” *Optics express*, vol. 19, pp. 4129–39, Feb. 2011.

[33] J. T. McCrickerd and N. George, “Holographic stereogram from sequential component photographs,” 1968.

[34] M. Levoy, “Light fields and computational imaging,” *Computer*, vol. 39, no. 8, pp. 46–55, Aug. 2006.

[35] R. Ng, M. Levoy, M. Bredif, *et al.*, “Light field photography with a hand-held plenoptic camera”, p. 11, Feb. 2005.

[36] Y. Xu, K. Maeno, H. Nagahara, and R.-i. Taniguchi, *Mobile camera array calibration for light field acquisition*, Jul. 2014.

[37] S. Vagharshakyan, R. Bregovic, and A. Gotchev, *Light field reconstruction using shearlet transform*, Sep. 2015.

[38] Y. Gao, R. Bregovic, R. Koch, and A. Gotchev, *DRST: Deep residual shearlet transform for densely sampled light field reconstruction*, Mar. 2020.

[39] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, *NeRF: Representing scenes as neural radiance fields for view synthesis*, Aug. 2020.

[40] B. Foundation. “Blender.org - free and open 3d creation software”, blender.org. ().

[41] M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, Sep. 2016, 1270 pp.

[42] E. Lopez. “Depth of field in path tracing”, Medium. (Apr. 2018).

[43] C. Soler, K. Subr, F. Durand, N. Holzschuch, and F. Sillion, “Fourier depth of field”, *ACM Transactions on Graphics*, vol. 28, no. 2, pp. 1–12, Apr. 2009.

[44] B. A. Barsky, M. J. Tobias, D. R. Horn, and D. P. Chu, “Investigating occlusion and discretization problems in image space blurring techniques”, p. 11, Sep. 2003.

[45] M. Kraus and M. Strengert, “Depth-of-field rendering by pyramidal image processing”, *Computer Graphics Forum*, vol. 26, no. 3, pp. 645–654, Sep. 2007.

[46] T. J. Kosloff, M. W. Tao, and B. A. Barsky, “Depth of field postprocessing for layered scenes using constant-time rectangle spreading”, p. 8, May 2009.

[47] A. Ignatov, J. Patel, and R. Timofte, “Rendering natural camera bokeh effect with deep learning”, in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Seattle, WA, USA: IEEE, Jun. 2020, pp. 1676–1686.

[48] B. Zhang, B. Sheng, P. Li, and T.-Y. Lee, “Depth of field rendering using multilayer-neighborhood optimization”, *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 8, pp. 2546–2559, Aug. 2020.

[49] M. Bertalmio, P. Fort, and D. Sanchez-Crespo, “Real-time, accurate depth of field using anisotropic diffusion and programmable graphics cards”, in *Proceedings. 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004.*, Thessaloniki, Greece: IEEE, 2004, pp. 767–773.

[50] “Cameras — blender manual.” (), [Online]. Available: `https://docs.blender.org/manual/en/latest/render/cameras.html`.

[51] S. Moreschini, F. Gama, R. Bregovic, and A. Gotchev, “Civit datasets: Horizontal-parallax only densely-sampled light-fields”, *European Light Field Imaging Workshop*, p. 4, 2019.

[52] G. Sellers, R. S. W. Jr, and N. Haemel, *OpenGL Superbible: Comprehensive Tutorial and Reference*. Addison-Wesley Professional, Jul. 2015, 1804 pp., Google-Books-ID: Nwo0CgAAQBAJ.

[53] M. Mara, M. McGuire, D. Nowrouzezahrai, and D. Luebke, “Deep g-buffers for stable global illumination approximation”, p. 15, 2016.

[54] C. Everitt, “Projective texture mapping”, p. 11.

[55] “OpenGL - the industry standard for high performance graphics.” (), [Online]. Available: `https://www.opengl.org/` (visited on 08/07/2022).

[56] J. C. Yang, J. Hensley, H. Grün, and N. Thibieroz, “Real-time concurrent linked list construction on the GPU”, *Computer Graphics Forum*, vol. 29, no. 4, pp. 1297–1304, Aug. 2010.

[57] K. E. Batcher, “Sorting networks and their applications”, in *Proceedings of the April 30–May 2, 1968, spring joint computer conference on - AFIPS ’68 (Spring)*, Atlantic City, New Jersey: ACM Press, 1968, p. 307.

[58] G. Thomas, *Advanced Visual Effects with DirectX 11: Compute-Based (GPU) Particle Systems*. 2014.

[59] A. R. Smith, “Image compositing fundamentals”, p. 8, Aug. 1995.

[60] A. Fridvalszky and B. Tóth, “Multisample anti-aliasing in deferred rendering”, *Eurographics 2020 - Short Papers*, pp. 21–24, 2020.