# Rendering Real Time Depth Of Field Effects

Lorenzo Marsicano, 2024578

November 2023

Thesis submitted for the MSc of Game And Media Technology


Supervisor Utrecht University: Peter Vangorp
Supervisor Traverse Research: Jacco Bikker
Second supervisor: Alex Telea

# Contents

# Chapter 1

# Introduction

**Depth Of Field** is a term that indicates the range of the world that is rendered in sharp focus on an image, and it occurs naturally when light passes through an optical system, composed of one or more lenses, before reaching a sensor. An example of this can be seen in Figure 1.1.
A natural side effect of Depth Of Field is that the out-of-focus areas will have a much larger **circle of confusion**, the image of a point in the real world on the sensor, than those in the focused areas. The shape of the circle of confusion is also called **bokeh**, from the Japanese word *boke*, meaning blur.
The shape and aesthetic of the bokeh are affected both by the elements that compose a lens system and by the **aberrations** that impact how the light rays move through those elements.

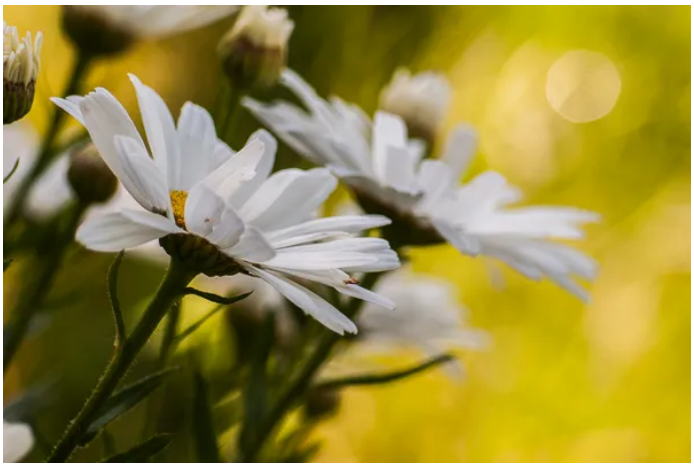


Figure 1.1: Photo with a shallow Depth Of Field: only a small portion of the world is in focus.[1]

A correct, faithful, and controlled rendering of the Depth Of Field is extremely relevant in any kind of visual media, including cinematography, photography, games, and animation. Authors and directors seek specific depth of field effects to be able to reproduce an atmosphere, tell a story, or simply shift the viewer's attention from one area of the shot to another.

(a) Kubrick had a special lens made to have everything in sharp focus. Every soldier is important in this situation.

(b) Bong Joon-Ho often uses out-of-focus areas to tell a second story.

Figure 1.2: Example of the Depth Of Field in movies[2].

Sometimes special lenses are made to have everything in sharp focus like in *Kubrick's Full Metal Jacket*, Figure 1.2a, while other times the out-of-focus areas can be used to tell a story that is parallel to the one happening in the foreground, something that *Bong Joon-Ho* does masterfully in *Memories of a murderer*, Figure 1.2b.

[1]From: `https://digital-photography-school.com/understanding-depth-field-beginners/`
[2]From `https://www.filmmakersacademy.com/depth-field-character-story/`

Other than being able to choose what is in focus, authors often choose the shapes of the out-focus highlights to convey a message or to better be able to set a scene like it has been done in Figure 1.3 by manipulating the aperture stop of the lens.

Figure 1.3: A custom bokeh shape can be obtained by placing a cardboard with a cutout of the desired shape in front of the lens.[3]

Control over the appearance of the Depth Of Field is crucial when incorporating VFX into a shot. Ideally, the goal is to replicate the same effect seen in real-life footage in the renders intended for integration. For this purpose, commercial compositing software goes a long way to try and replicate different real-life cameras for production movies. An example of such software is Foundry's Nuke[4].

## 1.1 Ray Tracing

While the concept of ray tracing has been around for a long time with its many uses in different fields including astronomy, radio signal, and optical design, its application in Computer Graphics, first suggested by Appel [1968], has a much shorter history. Whitted [1980] is one of the fundamental papers on the subject, and it is globally recognized as the base of all modern-day techniques.
Whitted's ray tracing, also known as recursive ray tracing, is an algorithm that recursively traces rays of light from the camera into the scene and computes the color of each pixel based on the interactions of the rays with the surfaces in the scene. By following the path of each ray, the algorithm can simulate different light phenomena like reflection and transmission.

[3] From `https://nicolesy.com/2021/12/29/shaped-bokeh/`
[4] `https://www.foundry.com/products/nuke-family/nuke`

Kajiya [1986], paved the way for path tracing, which solves the *rendering equation* integral with a Monte Carlo evaluator to simulate the light transport in a scene in a more physically accurate way.
Path tracing, compared to the recursive ray tracing algorithm, can be more computationally expensive, but can model global illumination more accurately. It expands on Whitted's model with indirect lighting, better sampling techniques for multiple lights, more naturally looking shadows and other features that give the final render a more photorealistic look.

## 1.2 The Pinhole Camera Model and its limitations

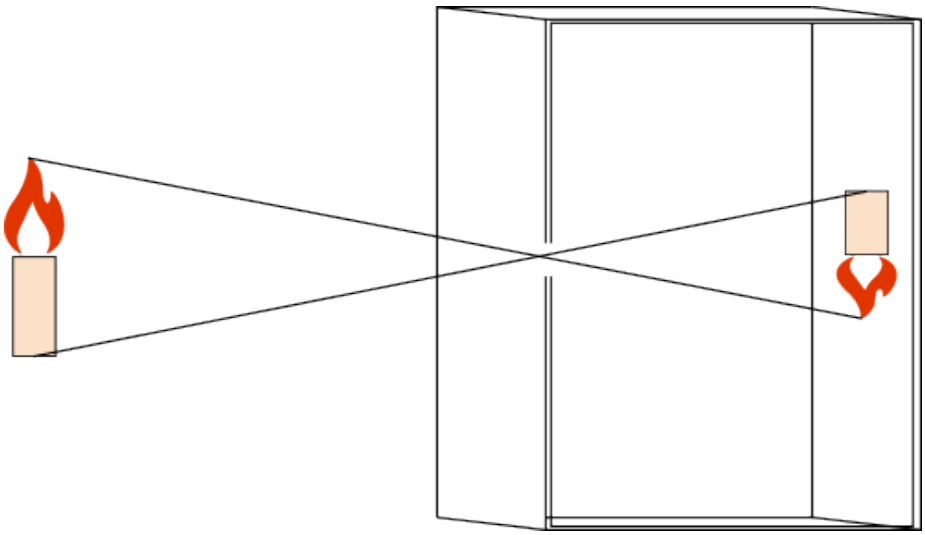


Figure 1.4: A pinhole camera.

In computer graphics is common to use a pinhole camera model both in rasterization and ray tracing.
A real pinhole camera is extremely easy to build: it is sufficient to place a light-sensitive film into a box and produce a small aperture in front of it. Light will pass through the small hole into the film which, after being exposed for a certain amount of time, will replicate what is in front of the camera.
As seen in Figure 1.4, the resulting image is flipped on the y-axis. When rendering an image through a pinhole camera in computer graphics, to avoid having to flip the final render, the virtual sensor is placed in front of the camera, as seen in Figure 1.5. Rays are then traced from the camera, through the film plane, and into the scene.

While being extremely easy to simulate, this type of camera will produce images where everything is in sharp focus, without any kind of Depth Of Field.

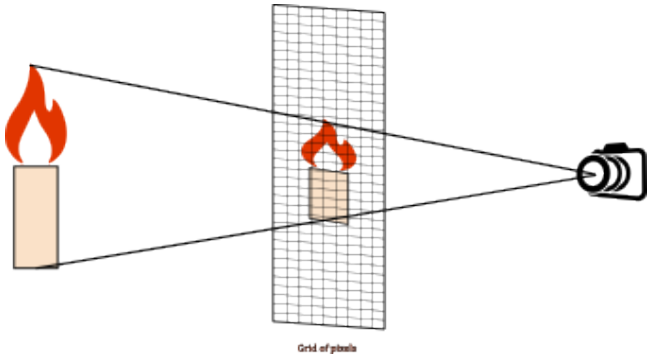



Figure 1.5: The pinhole camera model in ray tracing applications.

## 1.3 Overcoming the pinhole camera limitations

Pinhole cameras nowadays are only used for specific artistic purposes[5], as modern optical systems are now composed of many complex lens elements to account for imperfections, image stabilization and aberrations.

In computer graphics, the proposed solutions to work around the limitation of the pinhole camera started as early as Potmesil and Chakravarty [1981] and have been categorized into Object Space Approaches and Image Space Approaches by the survey from Barsky and Kosloff [2008].
**Image space approaches** are post-process techniques that add a Depth Of Field effect to an already rendered image through a classic pinhole camera, usually using blurring kernels on the whole image or in some portions of it.
This category of techniques is often used in real-time applications, thanks to their efficiency, but lack the realism that can be obtained through object space methods. The introduction of image space methods is often attributed to Potmesil and Chakravarty [1981], which laid the foundations for any subsequent work.
**Object space Approaches** on the other hand can replicate the aberrations of real lenses with very high accuracy by modeling how rays' paths are being deviated by the glass elements inside the optical system. This can be done for example by tracing rays through the lens components to physically reproduce the direction of a ray coming out of it. Distribution ray tracing by Cook et al.

[5]Pinhole cameras have been barely ever used: the concept of the Camera Obscura the pinhole camera is based on has been used mainly by painters and scientists since the 4th century B.C.

[1984], is the technique upon which many modern object space algorithms are based.
More recent papers tend to use a mixture of both screen and image space techniques and do not fit in either category, like the work from Tan et al. [2022] and Peng et al. [2022].

## 1.4 Problem Definition And Research Questions

In 2020, Asberg, Asberg [2020], defended her master's thesis on replicating realistic Depth Of Field effects using Seidel Aberrations as post-process to an already rendered image with a depth map.
In her thesis, she presented a novel approach to efficiently render both monochromatic and chromatic aberrations starting from real lens specifications to obtain authentic-looking bokeh.
To do this, and to give the end user some creative control, she reduced the lens to 7 parameters that can be changed at runtime to alter the obtained effect.
Asberg implemented her algorithm as a post-process effect, mentioning that reversing the procedure to generate primary rays for a ray tracer would work just as well, making it usable as a real-time Depth Of Field simulation.
This brings us to our first research question:

**RQ1: Can we reverse Asberg's innovative approach and expand it in order to produce primary rays for breda, Traverse Research's[6] real-time rendering framework?**

This research question was ultimately answered negatively. This reversal faced a lot of issues due to the intricate aberrations experienced by the individual rays within the optical system and the complex nature of light interactions. Those roadblocks made the reversibility of Asberg's technique non-trivial and prevented its direct application to the real-time rendering framework.
An in-depth explanation of Asberg's approach can be found in section 3.10, while a thorough analysis of its irreversibility can be read in chapter 4.

To still deliver a Depth Of Field effect, while also attempting to solve the partial visibility issue that affect any post-process effect, we turned to explore different techniques and methods to render practical DoF effect. The paper from Tan et al. [2022] introduced an interesting hybrid approach in which the missing geometry gets ray traced after a post-process effect inspired by Jimenez [2014].
This leads us to the second research question:

**RQ2: Can we implement Tan et al. [2022]'s paper in breda so that a post-process Depth Of Field can be rendered in real time, with low latency and while also improving on existing techniques for the partial visibility issue?**

[6] `https://traverseresearch.nl`

Therefore, we moved to implement Tan et al. [2022]'s paper. The complete lack of access to the code, the fact that the authors were not available to answer any of the problems that came up during development and because some sections of the paper left a lot of guess work to do this implementation was not trivial.
Those problems made it so that it was not possible to reach the same result as shown in Tan et al. [2022]. Nonetheless, part of the work was merged into the main branch of **breda**, and the final implementation still shows how the approach can actually improve on existing research.
In chapter 2, the necessary basic optics theory to understand previous work in the rendering of Depth Of Field is explained. This also includes an explanation on the aberrations that are being used by Asberg for her technique.
In chapter 3 existing techniques for rendering Depth Of Field are introduced. An important part of this chapter is dedicated to the explanation of Asberg's technique.
In chapter 4 the irreversibility of Asberg's technique is explained. First, the approach that was taken is explained, then the reason why it could not work for optical and physical limitation is expanded.
In chapter 5, Tan et al. [2022]'s paper is explained, and its implementation is discussed.
Finally, in chapter 6, a high level summary of the results of this thesis can be found, along with possible future improvements and developments.

# Chapter 2

# Basic Optics

To better understand how images on the camera sensor are formed and how the depth of field is obtained through the use of specific lenses, and therefore how we can simulate it in computer graphics, an introduction to photography and optics is needed. This chapter will highlight the necessary pieces to grasp the theory behind how the different parts of a camera lens affect the resulting shape of the highlights in the out-of-focus areas.

In chapter 3 the terms and knowledge introduced in this chapter will be used extensively. This chapter also includes a thorough explanation of the Aberration used by Asberg in her approach.

## 2.1 Lenses and Optics

### 2.1.1 Optical axis, focus distance and focal plane

Nowadays, an optical system is composed of multiple glass elements (lenses) that have the purpose of focusing the incoming light onto one or more points on the film sensor. Each of those elements lies on the **Optical axis**, (Figure 2.1).

Each lens has a, possibly fixed, **focus distance**, shown in Figure 2.2: the distance from the front of the lens at which an object will be perfectly sharp on the sensor. The plane perpendicular to the optical axis at the focus distance is called the **focal plane**.

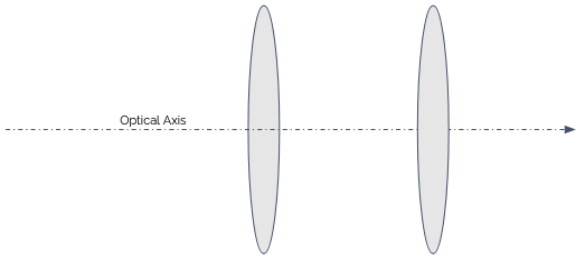



Figure 2.1: The Optical axis through two lenses.

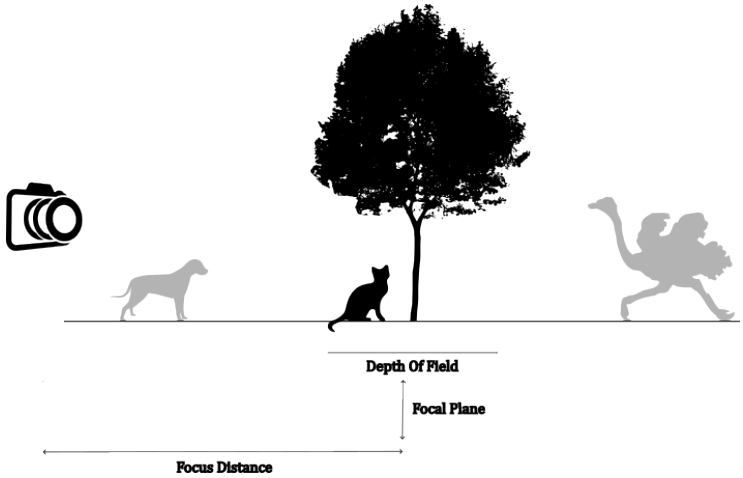



Figure 2.2: Schematic showing focus distance and depth of field.

### 2.1.2 Principal Planes and Focal Points

The **front focal point**, shown in Figure 2.3, is the point on the optical axis for which every ray that passes through it will emerge from the optical system parallel to the axis itself.
The **rear focal point**, shown in Figure 2.3, is the point on the optical axis on which rays that enter the system parallel to the axis will converge when emerging from the system.
The **Front principal plane** and the **rear principal Plane** are planes perpendicular to the optical axis at which all the refraction can be considered to happen. Refraction, in this context, refers to the bending of light as it passes through the optical system.

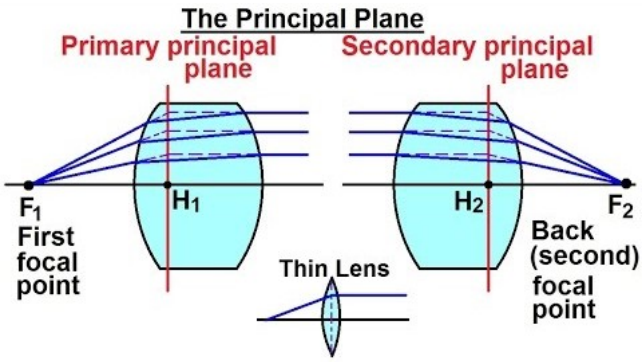



Figure 2.3: $H_1$ and $H_2$ are respectively the front and rear principal planes[1].

### 2.1.3 Focal Length

The **Focal length** of a lens is the optical distance from the point where the light converges inside the lens to the camera's sensor as shown in Figure 2.4. More specifically, it is the distance between the **rear principal plane** and the **rear focal point**, and it determines the lens's ability to focus light onto the sensor.

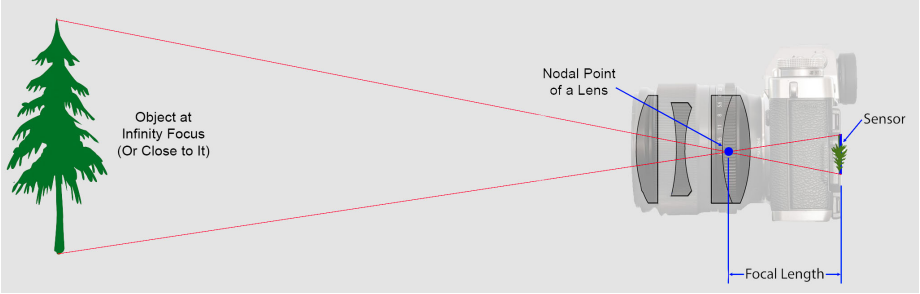



Figure 2.4: In this case, the focal length represents the distance between the sensor and rear principal plane, here called *Nodal Point of a Lens*[2]

[1] From `https://www.youtube.com/watch?v=2EUzr8fP0TA`
[2] From `https://photographylife.com/what-is-focal-length-in-photography`

### 2.1.4 Stops

Stops are elements in the lens system that limit the amount of light passing through. These could be opaque elements like the diaphragm in a camera lens or just the boundaries of the glass elements.
The **Aperture Stop** is the stop that defines how much of the incoming light reaches the sensor. This is illustrated in Figure 2.5.
In a camera, a diaphragm is an opaque element composed of 5 to 9 *"blades"* used to control the amount of light reaching the film. The size of this opening is usually defined in terms of the *"f-number"*, which is given by: $\frac{f}{D}$, where $f$ is the *focal length* and $D$ the *diameter of the aperture*.
A higher f-number means a smaller aperture size.

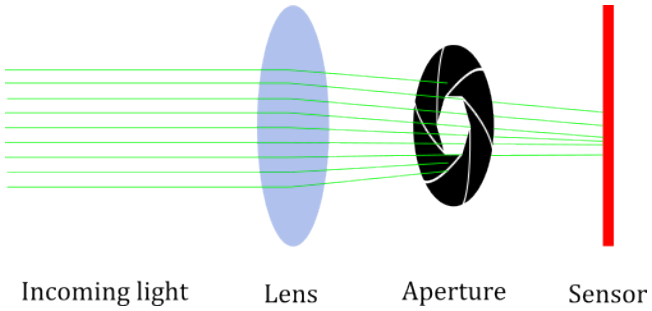



Figure 2.5: An aperture blocks incoming rays from reaching the sensor.

The image of the aperture stop is called **entrance pupil** if seen from object space and **exit pupil** if seen from image space.
Both can be seen in Figure 2.6.

Figure 2.6: The entrance pupil (on the left) and exit pupil (on the right).

### 2.1.5 Depth of field

The concept of **depth of field** is strictly related to the one of *focal length* and *aperture size*.
It is defined as the distance between the nearest and furthest objects that are in focus at the focal length, as shown in Figure 2.2.
A larger aperture (smaller *f-number*) will result in a shallower depth of field.

### 2.1.6 Bokeh and circle of confusion

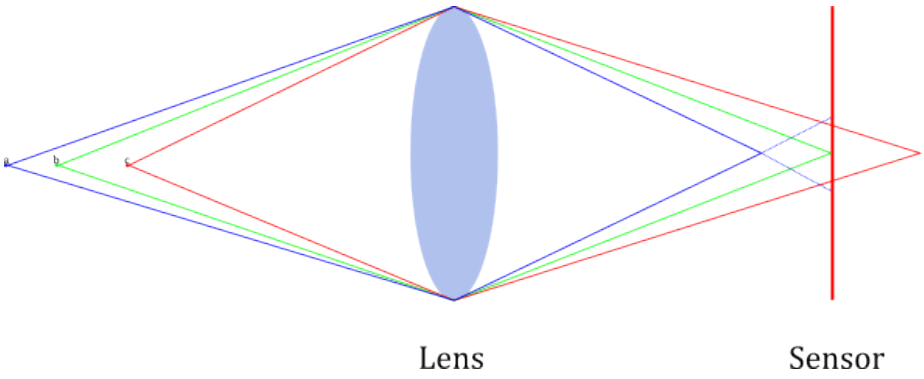



Figure 2.7: Only the green point $b$ is in focus. Light rays incoming from points $a$ (in blue) and $c$ (in red) converge to a single point in the front or the back of the film sensor.

The term **circle of confusion** indicates the circle formed on the sensor by a cone of light rays that do not come in focus at a single point. This happens for points that are both behind the maximum focus distance and in front of the minimum focus distance, as shown in Figure 2.7.
Potmesil and Chakravarty [1981] gave the formula to compute the diameter of the circle of confusion for a point $u$ in object space that project a point $V_u$ on the sensor given the focal length of the lens $f$, the f-number $n$, and a point $p$ perfectly in focus which project to $V_p$ on the imaging sensor:

$$C = \frac{f}{n}\frac{|V_u - V_p|}{V_u} \tag{2.1}$$

**Bokeh** is a Japanese word that describes the shape of the circle of confusion. Since the circle of confusion diameter is larger for the highlights in the out-of-focus area of the image, it is also described as the shape of the highlights in the out-of-focus areas.
The shape of the bokeh depends on the shape of the aperture stop, as shown in Figure 2.8.

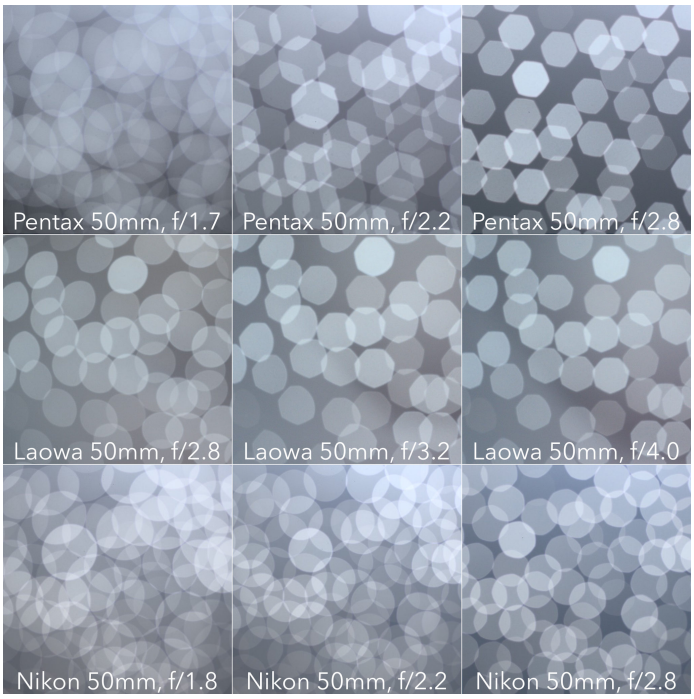



Figure 2.8: Bokeh shapes from different lenses. Lenses with different aperture shapes will result in differently shaped bokeh.[3]

### 2.1.7 Gaussian Optics and Paraxial Approximation

In optics, the **Paraxial Approximation** is a small angle approximation that can be applied when the angle produced by the incoming light with the optical axis is small enough. In this situation the following approximation is valid:

$$\tan\theta \approx \sin\theta \approx \theta \tag{2.2}$$

Gaussian Optics is a technique used to describe how light rays behave in a lens system by using the paraxial approximation. It only applies to optical systems composed of flat or spherical components.
Gaussian Optics can be used to derive certain properties of a system: *focal length*, *magnification*, *focal planes*, and *image planes*.

[3] From `https://photographylife.com`

## 2.2 Aberrations

The term *aberration* refers to a deviation of the light rays' path from the one predicted by the Paraxial Approximation.
Those deviations can arise from the inherent characteristics of the optical system, such as the shape, materials and their arrangement of its elements, or from the limitation of paraxial theory itself.
The Paraxial Approximation is a simplified model that does not take into consideration those intricacies and considers light as a ray, ignoring any aberration caused by the wavelength of the incoming light.

The five monochromatic aberrations are also known as *Seidel Aberrations* described mathematically in Seidel [1857].
The Seidel Aberrations are also known as *third-order aberrations* as they affect the image formation when expanding the power expansion of the $\sin\theta$ and $\cos\theta$ up to the third term:

$$\sin\theta = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - ...$$
$$\cos\theta = 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - ...$$

By expanding the series to even more terms, *higher-order aberrations* are obtained, and the Seidel aberrations can be considered as the sum of all contributing higher-order aberrations (Hecht [2017]).

### 2.2.1 Spherical Aberration

Spherical Aberration occurs for rays that originate on the optical axis. Those rays, leaving the point in a different direction, will pass through different points of the lens. Because of the spherical shape of the lens, not all those rays will converge into focus on a single point in the image plane as shown in Figure 2.9.

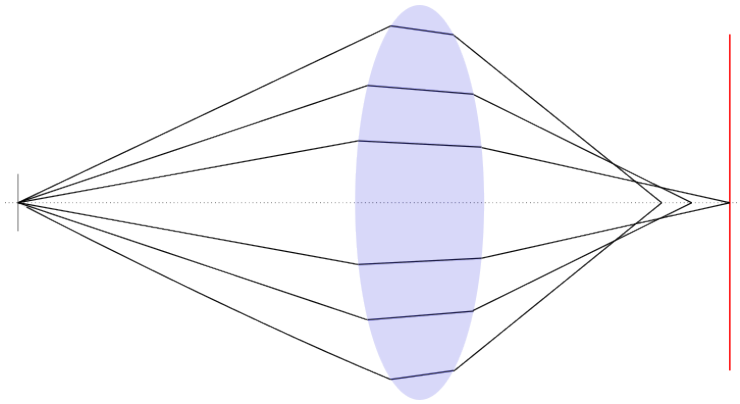


Figure 2.9: The effect of spherical aberration. Rays passing through a different part of the lens do not converge at the same distance.

### 2.2.2 Coma

Coma, also called comatic aberration, affects rays originating from a point further away from the optical axis. Even when the Spherical Aberration is corrected, those rays will come into focus at different heights on the image plane, as shown in Figure 2.10. This aberration will cause an effect that is similar to the one that can be seen from a comet, from which the name comes.

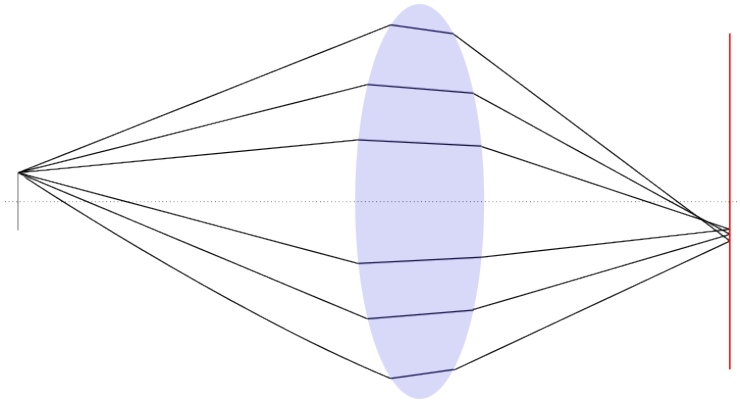


Figure 2.10: The effect of comatic aberration. Rays passing through a different part of the lens do not converge at the same height on the film plane.

### 2.2.3 Astigmatism

Astigmatism also affects rays that originate from points that are not on the optical axis. From the point of view of those rays, the lens appears tilted. Rays

that are in the plane of the tilt and rays that are perpendicular to that will pass through parts of the lens with a different profile, focussing at different distances from the lens as shown in Figure 2.11.
This aberration causes affected points to appear as a line.

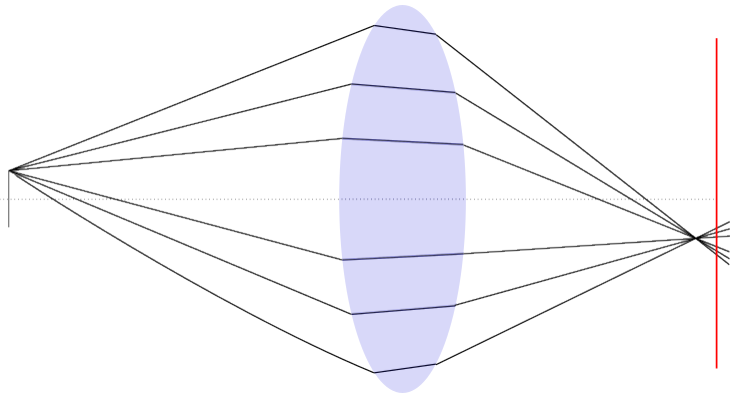


Figure 2.11: The effect of astigmatism.

### 2.2.4 Field Curvature

Even when all the incoming rays from a single point converge into perfect focus, because of the spherical shape of the lenses compared to the flat sensor, planar objects are projected as curved images, as shown in Figure 2.12.

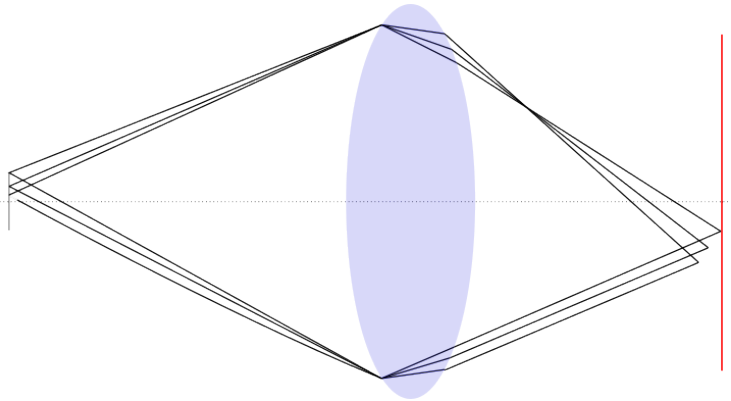


Figure 2.12: The effect of field curvature.

### 2.2.5 Distortion

The linear magnification (the ratio of the image size to the object size) is a function of the focal length, which is different for different areas of a lens, resulting in a distorted image, shown in Figure 2.13. As shown in Figure 2.14, the two kinds of distortions are *pincushion distortion* and *barrel distortion*.

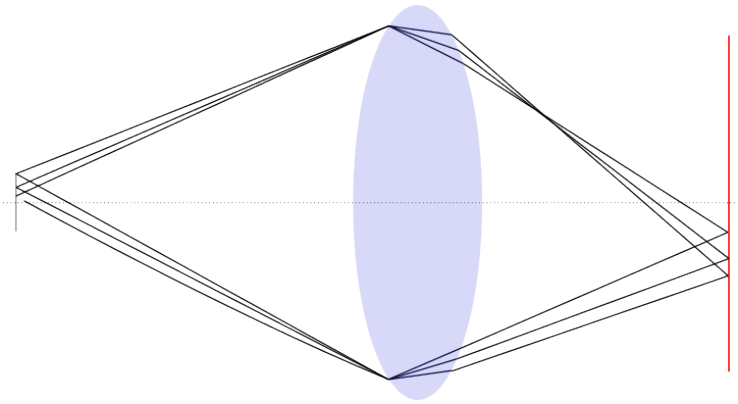


Figure 2.13: The effect of distortion.

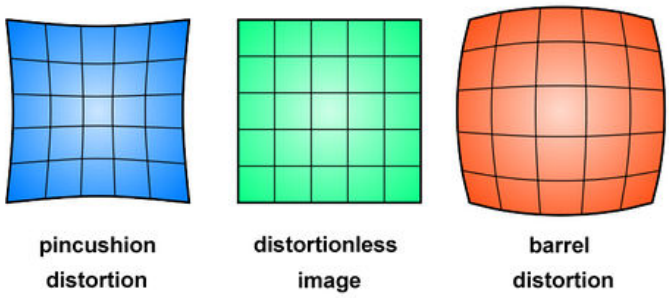



Figure 2.14: Different kinds of distortions.

### 2.2.6 Chromatic Aberration

Chromatic Aberration is caused by the inability of glass lenses to focus different wavelengths on the same point. Lenses have different focal lengths for different refractive indices, and the refractive index is dependent on the wavelength of the incoming light.
Two different kinds of chromatic aberration can be identified:

- **Longitudinal Chromatic Aberration**: Different wavelengths are focused at a different distance from the lens;
- **Transverse Chromatic Aberration**: Different wavelengths are focused at different points on the focal plane;

Figure 2.15 shows the effect of chromatic aberration.

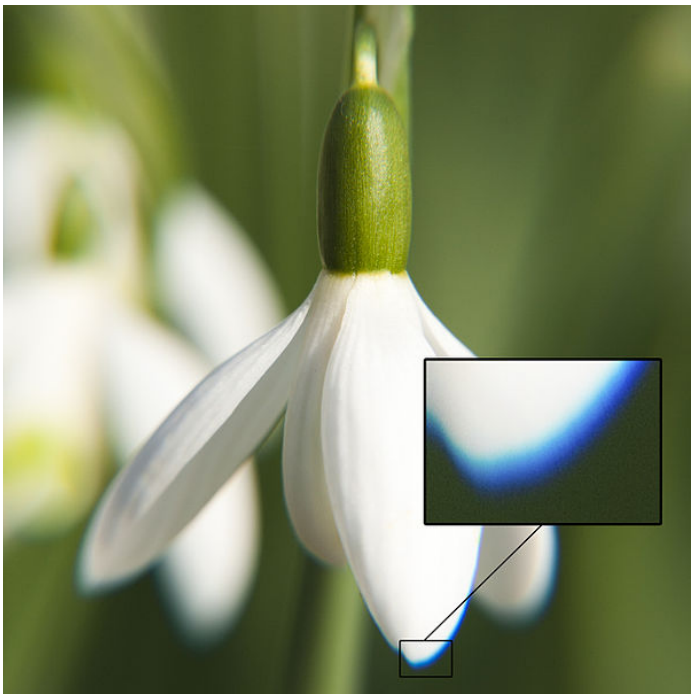


Figure 2.15: The effect of chromatic aberration.

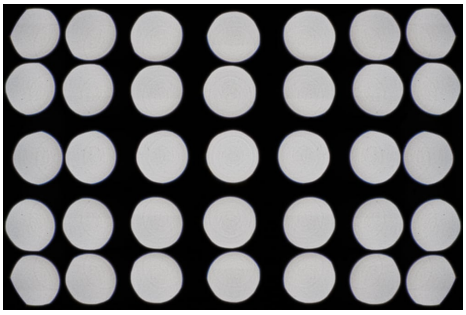
### 2.2.7 Vignetting

Vignetting is another optical effect that is not characterized as an aberration. It appears as a decrease in light intensity at the periphery of the image: light rays are blocked by the rim of one of the lenses in the system, as shown in Figure 2.16a.
Vignetting also affects the bokeh at the periphery of the image, as shown in Figure 2.16b.

A Summary of both the chromatic and monochromatic aberrations and of the vignetting effect can be found in Table 2.1.

[3] From `https://nl.m.wikipedia.org/wiki/Chromatische_aberratie`
[4] From `https://shotkit.com/guide-to-vignetting/`
[5] From `https://blog.kasson.com/nikon-z6-7/24-70-4-nikkor-s-on-z7-vignetting-and-bokeh/`

(a) Notice the darkening at the edges.[4]



(b) The effect of vignetting on the bokeh shape.[5]

Figure 2.16: The Vignetting effect.

| **Aberration** | **Description** |
|---|---|
| *Spherical Aberration* | Rays originating on the optical axis do not converge to a single point due to the lens's spherical shape. |
| *Coma* | Rays from points off the optical axis focus at different heights on the image plane, resembling a comet. |
| *Astigmatism* | Rays originating off the optical axis encounter different profiles in the lens, resulting in different focal distances. |
| *Field Curvature* | Planar objects are projected as curved images due to the difference between lens shape and a flat sensor. |
| *Distortion* | Linear magnification varies across the lens, leading to image distortion. Two common types are pincushion and barrel distortions. |
| *Chromatic Aberration* | Glass lenses cannot focus different wavelengths on the same point, causing longitudinal and transverse effects. |
| *Vignetting* | Light falloff towards the corners of the image, resulting in darker peripheral areas. |

Table 2.1: Summary of Aberrations

# Chapter 3

# Previous Work

To overcome the limits of the pinhole camera limitation, a lot of different techniques have been proposed during the last 40 years, each with its unique strengths and limitations.
This chapter will provide an overview of some of those techniques, aiming ti provide an overview of what is currently available, while highlighting their respective advantages and drawbacks.
In the upcoming sections, some of those works are presented to give an overview of what Asberg's thesis is based on and what the current state-of-the-art is. The surveyed techniques are organized according to the categorization specified by Barsky in Barsky and Kosloff [2008], going through the Object Space methods first (section 3.1, section 3.2, section 3.3, section 3.4), Image Space Methods second (section 3.5, section 3.6, section 3.7, section 3.8) and lastly some more recent techniques that do not properly fit into any of those categories, including Asberg's approach (section 3.9, section 3.10).

The latest survey on the existing techniques to simulate depth of field has been done in Barsky and Kosloff [2008]. The paper proposes a categorization in image space and object space methods: the former is more apt to be used in a real-time environment and the latter is preferred for more accurate reproduction of the desired effects.
One of the biggest issues with image space approaches is *partial occlusion*: in images captured with real cameras, foreground objects have a soft edge behind which the background can be seen. Because the information on the color behind foreground objects is simply missing when applying a depth of field effect on an already rendered image through a pinhole camera, approximations need to be made to work around this issue.
An example of this phenomena in computer graphics is shown in Figure 3.1, which compares a blurred and unblurred render of the **Sponza** scene made using **Blender**. By looking at the post-processed image it is easy to see how the out-of-focus rod expands outwards and partially cover the wall behind it. In a real image, it would be possible to partially see the hidden wall through the

non-focused colors.

(a) A render of the Sponza scene made using a pinhole camera in Blender

(b) The same render, but post-processed to add a Depth Of Field effect

Figure 3.1: Comparing an all-in-focus render and its post-processed version.

More recent approaches do not easily fit into either of those categories, like the recent paper from Tan et al. [2022], which uses a hybrid approach by first blurring a rasterized image with a classic filter and then using ray tracing to solve the partial occlusion problem.
Another interesting research area is replicating depth of field and bokeh through the use of different kinds of neural networks: Peng et al. [2022], Ignatov et al. [2020], Qian et al. [2020].

## 3.1 Distributed Ray Tracing

Cook et al. [1984] form the basis for every modern object-space technique.
Cook introduced the Distributed Ray tracing technique to address the problem

of rendering certain phenomena by distributing the rays in time instead of tracing more of them. The *"fuzzy phenomena"* that Cook addresses in his paper are depth of field, translucency, blurred reflections, penumbra, and motion blur.
The first three effects are related to the shading of a point on a surface. The intensity $I$ of the reflected light at that point is the integral over the hemisphere above the surface of an illumination function $L$ and a BRDF $R$ for the incident angle $(\phi_i, \theta_i)$ and the angle of reflection $(\phi_r, \theta_r)$:

$$I(\phi_r, \theta_r) = \int_{\phi_i} \int_{\theta_i} L(\phi_i, \theta_i) R(\phi_i, \theta_i, \phi_r, \theta_r) d\phi_i d\theta_i \tag{3.1}$$

To simulate *gloss* (the blurred reflections effect), rays are distributed over the direction of the mirror reflection and then weighted accordingly to the same distribution.
Similarly, to model *translucency*, the secondary rays are distributed about the main direction of the transmitted light according to the BTDF (Bidirectional Transmittance Distribution Function) that replaces the BRDF $R$ in Equation 3.1.
*Penumbra* is present when the intensity of light is proportional to the solid angle of its visible portion. This effect is obtained by distributing rays toward any point on the light source location.

### 3.1.1 Distributed depth of field

Cook observed that real cameras have a finite lens aperture that gives each image a finite depth of field by projecting each point in the scene as a circle of confusion.
By using the focal length $F$, and the aperture number $n$, rays can be distributed into the scene to simulate a depth of field effect.
For each point $p$ on the imaging sensor, a point on the lens is sampled using its diameter $\frac{F}{n}$ and a ray is traced through that point into the scene.

This produces a correct circle of confusion for each point, and Cook proves this by comparing it to the formula given by Potmesil and Chakravarty for the diameter $C$:

$$C = |V_D - V_P| \frac{F}{nV_D} \tag{3.2}$$

With $V_P$ being the distance of the image of a lens focused at distance $P$ and $V_D$ the distance of the image plane of a point at distance $D$ from the lens. $V_P$ and $V_D$ are given by:

$$V_P = \frac{FP}{P - F} for P > F \tag{3.3}$$

$$V_D = \frac{FD}{D - F} for D > F \tag{3.4}$$

Given a point $I$ on the image plane, the algorithm proposed by Cook traces ray inside a cone whose radius at $D$ is

$$r = \frac{1}{2}\frac{F}{n}\frac{|D-P|}{P} \tag{3.5}$$

The image plane distance from a point on the cone to a point on the axis of the cone is

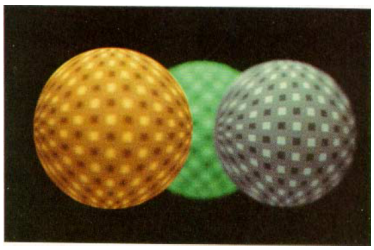
$$R = r\left(-\frac{V_P}{D}\right) \tag{3.6}$$

and it is trivial to show that

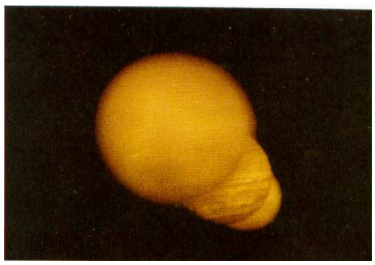
$$R = \frac{C}{2} \tag{3.7}$$

Meaning that any points on the cone have a CoC that touches the image point $I$ and points outside the cone do not affect the image point.

The integral shown in Equation 3.1 can be solved with a Monte Carlo evaluator, making the implementation of Cook's paper trivial. The results can be seen in Figure 3.2.



(a) Render from Cook et al. paper



(b) Render from Cook et al. paper

Figure 3.2: Distributed ray tracing.

To summarize, the approach introduced by Cook was very innovative and permitted to replicate many effects. The downside in this case was simply that it needed many more samples per pixel to be able to converge to a solution, making it harder to use for modern real-time application.

## 3.2 A Realistic Camera Model for Computer Graphics

A few years after Cook's paper, Kolb et al. [1995] expands on their technique by tracing rays directly through a physically based representation of an optical system.

Kolb wanted to address two main issues: the computation of the geometry (correctly modeling aberrations and distortions) and radiometry (the response of

the sensor when exposed to light) of image formation.
The authors were not interested in correctly representing any aberration aberrations: chromatic or monochromatic. In their paper, models how the camera described in the paper transforms the scene radiance into the response of a pixel.

Ray tracing through a lens system is not any different from tracing rays in a scene composed of glass surfaces. For each point on the image plane, a point on the rear-most element of the system is sampled and then, for each element $E_i$, rear to front, the intersection between the ray and the lens element is computed. If the intersection is outside the $E_i$ aperture, the ray is blocked, otherwise, Snell's Law is used to compute the new direction.
Kolb derives a **thick lens approximation** and uses it to determine the exit pupil. The behavior of a thick lens is characterized by the focal points and the principal planes, and it is an approximation similar to the thin lens but where the thickness of the system becomes relevant. The principal planes can be found by tracing rays through the system or by using analytical formulas available to derive a thick lens from a collection of optical elements.

When tracing rays from the sensor to the scene through the camera, the exit pupil should be sampled instead of the camera aperture or the rearmost lens element (the one closest to the sensor) for a correct result. When sampling the former, certain rays that would pass through the lens will not be generated as the image of the aperture can be larger than the aperture itself, and when sampling the latter, rays that will not pass through will be generated and discarded as soon as they will encounter a stop.
Determining the apparent size and position from the axial point on the image plane for each potential stop by imaging the stop through the lens elements that stand in between the stop and image space is needed to find the exit pupil. The image of the image disk that subtends the smallest angle from the axial point on the image plane is the *exit pupil*. The stop corresponding to it is the *aperture stop*.

### 3.2.1 Exposure

Kolb models a simplification of the exposure process that happens in a real camera, by assuming that irradiance over a point $x'$ on the film plane is constant over a period and that the exposure time is fixed.
The following integral computes the irradiance at $x'$, $E(x')$ by integrating over the solid angle subtended by the exit pupil:

$$E(x') = \int_{x'' \in D} L(X'', x') \frac{cos\theta' \cos\theta''}{||x'' - x'||^2} dA'' \tag{3.8}$$

Which, assuming that the film plane is parallel to the exit pupil plane, becomes:

$$E(x') = \frac{1}{Z^2} \int_{x'' \in D} L(X'', x') \frac{cos^4 \theta'}{d} A'' \tag{3.9}$$

$Z$ is the distance from the film plane to the disk along the optical axis.
The weighting that differs according to the solid angle will result in variance in the irradiance over the film plane. To estimate this effect, Kolb uses two approaches:

1. **The $cos^4$ law**: Using the small angle approximation, when the exit pupil subtends a small solid angle from $x'$, $\theta'$ can be assumed constant and equal to the angle between $x'$ and the center of the exit pupil:

$$E(x') = L \frac{A}{Z^2} \cos^4 \theta' \tag{3.10}$$

2. When the solid angles are larger, Kolb describes how to use the differential form factor from a point on the film plane to a disk to estimate the variation in irradiance. An analytical solution is provided from Moon and Spencer [1981]:

$$F = \frac{1}{2} \left( 1 - \frac{a^2 + Z^2 - r^2}{\sqrt{(a^2 + Z^2 + r^2)^2 - 4r^2 a^2}} \right) \tag{3.11}$$

### 3.2.2 Sampling Strategy and noise reduction

Sampling the exit pupil is already a good step in improving the performance of the rendering.
Kolb also claims that adding importance sampling results in a noise reduction of about one percent while adding a lot of extra complexity, as each resulting value must be weighted by a factor of $\frac{\cos \theta' \cos \theta''}{||x'' - x'||^2}$.

To conclude, the resulting renders generated by the author can be seen in Figure 3.3. Depth of field is correctly simulated, as well as the effect of different focal lengths and non-linear geometric transformations like those produced by fisheye and anamorphic lenses.
Because of the assumed perfect transmission of the lenses that are part of the optical system, Kolb's model does not include the aberration produced by the deviation of the light ray discussed in section 2.2, and it ignores other camera behaviors like the opening and closing time of the shutters or the transmittance of the glass of the lens elements.

## 3.3 Rendering realistic spectral bokeh due to lens stops and aberrations

Wu et al. [2010] and Wu et al. [2012] expand on the technique introduced by Kolb et al. by removing the assumption of the perfect transmission, reproducing

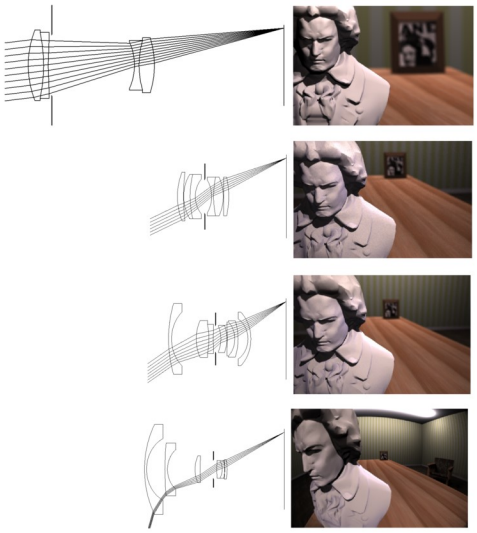


Figure 3.3: Render from Kolb et al. paper

both chromatic and monochromatic aberrations and implementing their new approach into a bidirectional path tracer.
The positions of the pupils are found by ray tracing the optical system (forward and backward), then along ray tracing, Gaussian Optics properties are used to get the pupils radii.
Tracing through the lens system is again trivial as it was presented in section 3.2: for each element of the system, compute the intersection and if the ray passes through the aperture, update its direction using Snell's Law.

For the bidirectional path tracing both a camera and a light path are needed. The authors observe that it would be impossible to connect the two paths if the first vertex of the camera path were to be placed on the exit pupil. Therefore, the two paths are created as follows:

- *Camera Sub-path*: The first vertex is placed on the entrance pupil. Its conjugate on the exit pupil is used to sample the ray from the image sensor to the pupil. The ray will be traced through the lens elements to obtain its direction leaving the system;
- *Light Sub-path*: Like in a normal bidirectional path tracer;
- *Sub-path connection*: When the two paths connect, a new ray is created that starts at the entrance pupil. The Ray is traced through the lens system to find the position on the imaging sensor it contributes to;

The 2010 paper left out the effect of dispersion from the glass lenses, which was added in the 2013 publication.

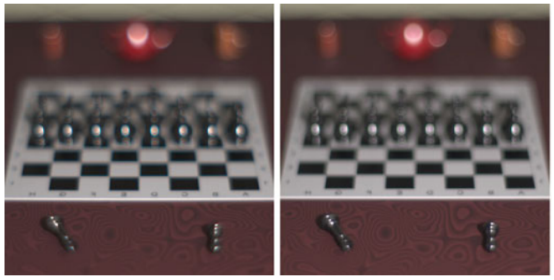
Using the glass coefficients provided by the manufacturer, the IOR of the lens is computed either with the Schott equation:

$$n^2(\lambda) = a_0 + a_1\lambda^2 + a_2\lambda^{-2} + a_3\lambda^{-4} + a_4\lambda^{-6} + a_5\lambda^{-8} \tag{3.12}$$

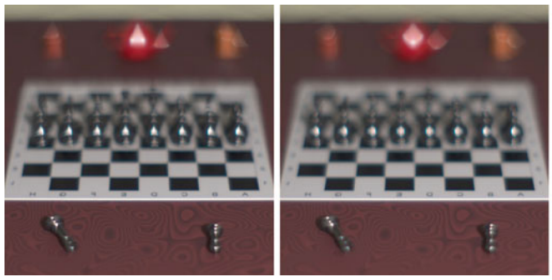
or with the Sellmeier equation:

$$n^2(\lambda) = a_0 + \frac{a_1\lambda^2}{\lambda^2 - b_1} + \frac{a_2\lambda^2}{\lambda^2 - b_2} + \frac{a_3\lambda^2}{\lambda^2 - b_3} \tag{3.13}$$

for a fixed set of wavelengths, which are then interpolated at runtime.
The authors use a Stratified Wavelength Cluster technique by Evans and McCool [1999] so that each ray carries a bundle of wavelengths instead of the trivial single wavelength per ray to increase efficiency.



(a) Rendering for two different apertures: F/1.35 and F/1.7.



(b) Rendering for a triangular and rectangular aperture.

Figure 3.4: Renders from Wu et al., 2013

Figure 3.4 shows the result of this technique for two different apertures in Figure 3.4a and for two different bokeh shapes in Figure 3.4b.

Wu et al. paper models correctly the depth of field and arbitrary bokeh shapes, but it only mentions a possible real-time implementation through NVIDIA OptiX[1] in the conclusion.

[1] `https://developer.nvidia.com/rtx/ray-tracing/optix`

In conclusion, although this model is extremely efficient and accurate, it also leaves out, like it happened in Kolb et al. [1995], unwanted reflections on the inner lens surfaces. The authors also mention that better sampling strategies could be implemented to accelerate the rendering of the bokeh.

## 3.4 Other Object Space techniques

Many other techniques that fall into this category exist and are still being developed. Some can approximate complex bokeh shapes easily while others model additional imperfections.

In Hullin et al. [2012] it is observed that the ray-lens intersection can be solved analytically and that an optical system performs a mapping from one set of directions to another.
Using the Taylor Expansion of those analytical solutions truncated to a system of polynomials, Hullin reduces the problem of tracing rays through an optical system to a set of polynomials that can be concatenated similarly to what happens in the Matrix method when concatenating multiple transfer and refraction matrices together to obtain the system matrix.
This approach describes both lens flare and the classic Seidel Aberrations correctly but ignores the diffraction effects of the glass components of the lenses.
Those polynomials are fast to solve, but not as easy to derive given a file that contains all the specifics of an optical system.

Joo et al. [2016] models aspheric lenses and, through the use of normal mapping, add scratches and manufacturing imperfections;
Aspheric lenses are generally designed to reduce or eliminate the aberrations present in spherical lenses. Since it is preferable to avoid double-precision computation on GPUs, a *bracketing-based root-finding* method is used to compute the ray-lens intersection, relying either on the *bisection method* or the *false position method*. The authors also provide an implicit representation of an aspheric lens, as the common parametric version is unsuitable for the intersection test.
The normal maps used to simulate the manufacturing imperfections are produced by simulating the grinding and polishing processes that happen when a new lens is manufactured. A texture is obtained by adding white noise to an image of concentric circles with their radii perturbed according to the tool used for the polishing process. The result of this approach can be seen in Figure 3.5.

## 3.5 A lens and aperture camera model for synthetic image generation

Probably the most important paper on applying a depth of field effect on an already rendered image is from Potmesil and Chakravarty [1981], upon which

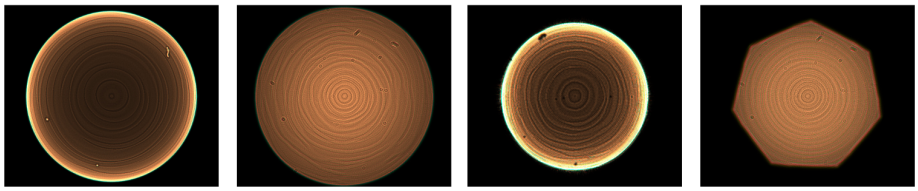


Figure 3.5: Render from Joo et al. paper

many modern techniques are built.
Potmesil and Chakravarty provided a very accurate model of the camera geometry and image formation, taking also into account any diffraction effect of the light passing through the optical system. The paper's technique uses two different steps to simulate the depth of field effect:

1. **The Hidden-Surface Processor**: using Whitted's recursive illumination model, the image is generated. If a pixel has a large variance in sampled values at its corner it gets subdivided into 2x2 squares and the sampling process is repeated. Each sample, along with the intensity and the coordinates of the pixel, contains the z-depth of the visible surface and its identification number.

2. **The Focus Processor** approximates the integration process that takes place on the film plane during the exposure from the rasterized image and the camera model. Each sample is considered a light source, and it is modeled by a delta function with a magnitude equal to the light intensity. The integral of the intensity distribution function over the circle of confusion is equal to the magnitude of the input delta function. The circle of confusion size is computed using Equation 2.1.6 and the light intensity in the circle of confusion by:

$$I(u,v) = \left(\frac{2}{u}\right)^2 [V_1^2(u,v) + V_2^2(u,v)]I_0 \tag{3.14}$$

$$I_0 = \frac{kd^2}{8F^2} \tag{3.15}$$

   Where $V_1$ and $V_2$ are the Lommel functions, for which Potmesil provides an approximation in his paper;
   The integral of the intensity distribution over the area of a pixel is the contribution of the sample point to the intensity of the pixel. This intensity is attenuated by the square of the z-depth of the point sample.

The focus processor uses the following formula to compute the pixel intensity $Q$ at pixel area $(X, X+\Delta X), (Y, Y+\Delta Y)$

$$Q(X, X+\Delta X, Y, Y+\Delta Y) = \frac{\sum_{i=1}^{N} \frac{f(x_i,y_i,z_i)}{z_i^2} q_i}{\sum_{i=1}^{N} \frac{f(x_i,y_i,z_i)}{z_i^2}} \tag{3.16}$$

With $N$ being the number of point samples in the image, $q_i$ the intensity of point sample $i$, $x_i, y_i$ the coordinates of sample $i$ in the image plane and $z_i$ the z-depth of the i-th sample.
$f$ is:

$$f(x_i, y_i, z_i) = \int_X^{X+\Delta X} \int_Y^{Y+\Delta Y} I(z_i, \sqrt{(x - x_i)^2 + (y - y_i)^2}) dy dx \qquad (3.17)$$

For several equally spaced z-depth coordinates, the focus processor computes 2D lookup tables containing the integral of Equation 2 divided by the square of the z-depth. This value is computed at each pixel where the corresponding circle of confusion overlaps and the entries outside that circle of confusion are filled with zeros.
For each input sample point, the focus processor selects the two lookup tables nearest to the z-depth of that sample. The corresponding pixel entries are interpolated and the center of this temporary table is displaced to the image coordinate of the sample point. After all samples are processed, the final color is obtained through Equation 3.5.

The result of Potmesil and Chakravarty's implementation can be seen in Figure 3.6. Potmesil implementation was a big stepping stone into the world of

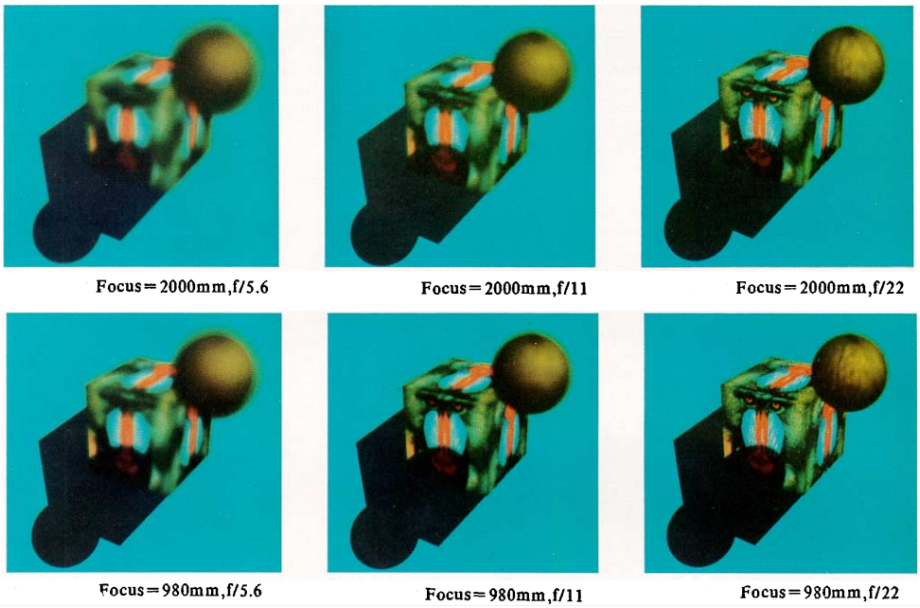



Figure 3.6: Renders from Potmesil and Chakravarty's paper.

simulated Depth Of Field, and a great starting point from all future researches.

The proposed algorithm was fast and user/artist-friendly, meaning that it was easy for the user to change its appearance by tweaking the algorithm parameters. On the other hand, the result of the blurring proposed by Potmesil is far from being physically accurate and could present some artifacts in scenes with complex occlusions or intricate geometries.

## 3.6 Blurring filters

Another classic approach to simulate the depth of field effect as a post-process is via the use of blurring filters, for example by using a Gaussian or a Box kernel. Zhou et al. [2007] uses a two-pass filter: on the first pass a vertical-only filter is applied to each pixel and the output of this pass is the input of the second horizontal-only pass.
The pixels are weighted based on an overlap function $O(r_p)$, defined as:

$$O(r_p) = \begin{cases} 0, & r_p \leq d_p \\ r_p - d_p, & d_p \leq r_p < d_{p+1} \\ 1, & r_p \geq d_{p+1} \end{cases} \tag{3.18}$$

Where $r_p$ is the radius of the circle of confusion of pixel $P$ and $d_p$ is the distance of the same pixel $P$ from the center pixel $C$.
$O(r_p)$ measures the degree of overlap of the circle of confusion of a pixel $P$ to the center pixel $C$.
The other two factors influencing the weight of the pixels in the kernel are the light intensity function $I(r_p)$ and the intensity leakage control function $L(z_p)$, which add an extra factor for pixels farther away from the camera than the focal plane to address intensity leakage. The result of this approach can be

Figure 3.7: Renders Zhou et al. paper. The focus of the right image is on the fountain.

seen in Figure 3.7. Only simple bokeh shapes can be approximated this way, by choosing accordingly shaped filters.

Mcgraw [2014] uses low-rank kernels to approximate arbitrary bokeh. The rank $r$ of a filter kernel is the rank of its matrix representation $k$: the number of its linearly independent columns.
Applying a filter of an arbitrary rank to an image $I_0$ can be done with the following equation:

$$I = \sum_{i=1}^{r} \sigma_i (I_0 * u_i) * v_i \tag{3.19}$$

Where the values of $\sigma_i, u, v$ can be determined by the singular value decomposition of $k$.
The benefit of low-rank filters over separable ones is that they can create bokeh of arbitrary shapes that can also simulate lens spherical aberration. In particular, the authors observe how bokeh with a horizontal or vertical axis of symmetry will have lower rank kernels, which gives in return a lower relative error in their approximations.
An example of the approximated bokeh can be seen in Figure 3.8.

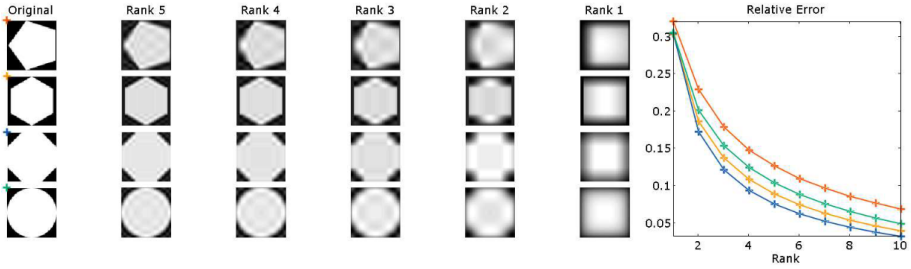



Figure 3.8: Bokeh approximated by low-rank filters. From McGraw et al. paper.

Kass et al. [2006], compute blurring by simulating the heat diffusion equation. The image intensities provide a heat distribution and the diameter of the circle of confusion is used to model the thermal conductivity.
The basic heat equation, given an input image $x(u, v)$ and a diffused output image $y(u, v)$, can be written as:

$$\gamma(u, v)\frac{\partial y}{\partial t} = \nabla \cdot (\beta(u, v)\nabla y) \tag{3.20}$$

With $\beta(u, v)$ being the heat conductivity and $\gamma(u, v)$ is the specific heat.
The authors use an Alternating Direction Implicit solution method to solve the equation to efficiently find a solution.
This approach allows for a very sharp separation of the out-of-focus areas from the in-focus ones: as soon as the conductivity drops below a certain threshold, the blurring will stop dead in its tracks. An example can be seen in Figure 3.9

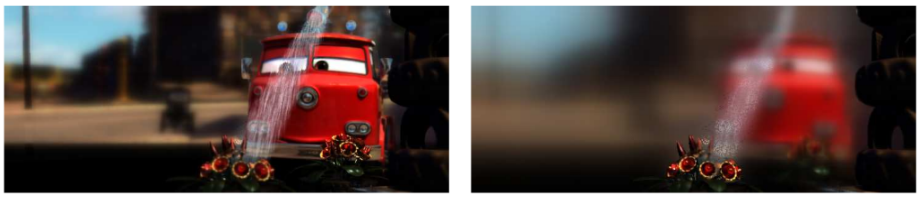


Figure 3.9: Renders from Kass et al. paper.

All of these filtering kernel approaches suffer from different issues in case of depth discontinuities. Generally speaking, in case of depth discontinuities, it is hard for a kernel to be able to classify a pixel as a background or foreground. Each technique deals with this problem in different ways, but overall they all have the worst case scenario in which the algorithm fails to blur the pixel or fallback to a much slower alternative.

## 3.7 Sprite Rendering approaches

This is another post-process technique used to render a sprite of a bokeh shape onto the blurred image, transforming it based on the size of its circle of confusion.
Jeong et al. [2022] uses this approach to render a bokeh obtained through ray tracing a thick lens approximation onto a blurred image. Figure 3.10 shows the pipeline of their approach.
At first, the highlights are extracted from a multi-layered pinhole render, then visibility sampling is performed using a novel *forward batch visibility sampling*, where the visibility of a highlight is only related to rays in the cone formed by the highlight and the lens. Similar and close-by highlights are also clustered together to improve the performances.
The visibility map is computed either using this rasterized approach or an image-based ray tracing technique for complex scenes to avoid actual geometry passes. Finally, the parameters needed to render the bokeh are computed and stored in a look-up table. At runtime, those parameters are used to render the bokeh over the highlight through a series of transformations that allows control of the appearance of the bokeh when influenced by focus length, object depth, spectral dispersion, lens aperture, distortion and optical vignetting.
Despite the good quality of the results, this approach suffers from many limitations: because of the multiple rasterization passes, it is extremely inefficient in case of excessive highlights; temporal coherence is not guaranteed by the highlight extraction technique, which is also view-independent, making the final result different from the reference implementation chosen by the author.
Abadie [2018] gave a presentation at SIGGRAPH on the depth of field implementation used in Unreal Engine. It is considered the state-of-the-art of depth of field simulation in real-time rendering (Ignatov et al. [2020], Asberg [2020]) and uses a scattering & gathering approach to correctly simulate DoF and bokeh

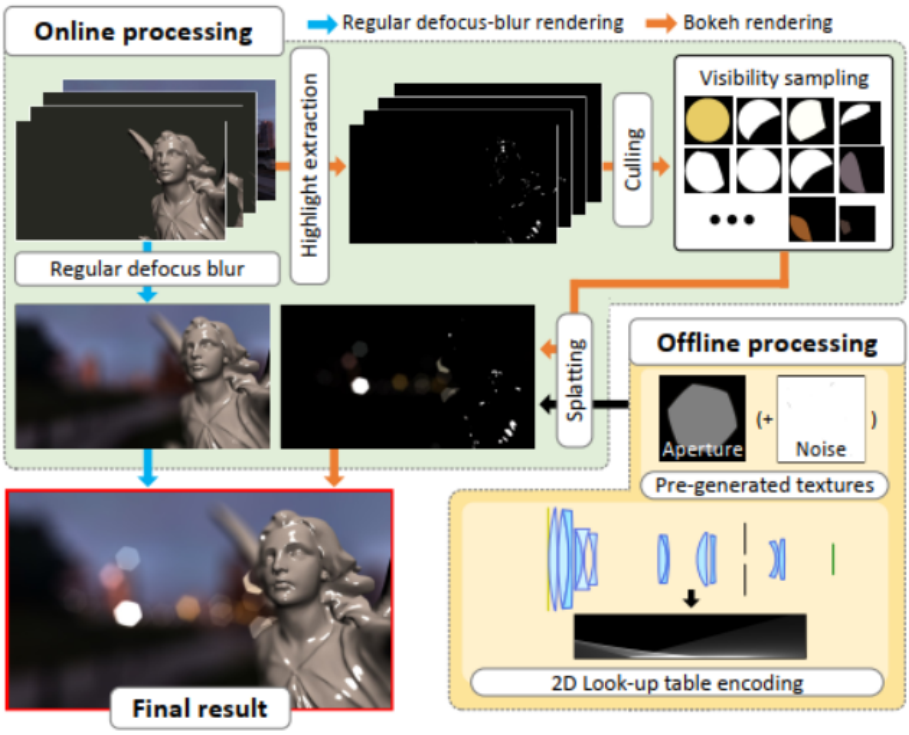



Figure 3.10: From Jeong et al.

effects.
The background and foreground are blurred separately at half of the resolution of the original image. The kernel uses a mix of scattering and gathering and adaptive density sampling to correctly simulate occlusion, suppress noise and obtain partial occlusion.
For the partial occlusion issue, when the kernel meets a background pixel for which its neighbor is a foreground one, the opposite pixel in the kernel is sampled to mirror its color.
Together with a procedurally generated aperture shape, this approach allows for a correct real-time implementation of the depth of field and bokeh effect, tackling the known issues (partial occlusion, undersampled highlights among many others) by combining novel and known techniques.

## 3.8 Pencil Maps

Gotanda et al. [2015] precomputes the light paths through the lens for several wavelengths and stores them in a texture that can be seen in Figure 3.11.

Those textures have on the horizontal axis the distance from the light source in object space and on the vertical axis the distance from the center of the bokeh. Since the information on those textures is sparse, Gotanda et al. normalize them to reduce the amount of information that needs to be passed to the GPU. A normalized version of the pencil map can be seen in Figure 3.12. By taking

Figure 3.11: Pencil map from Gotanda et al., 2013

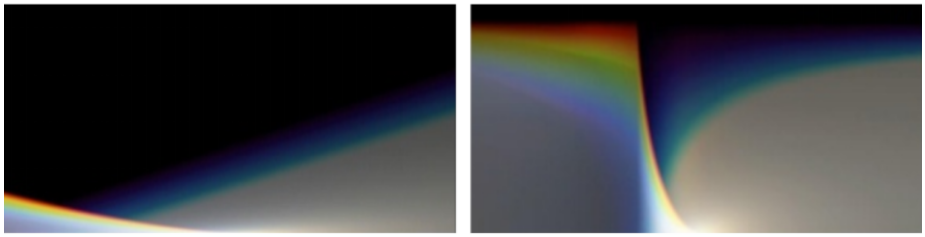


Figure 3.12: Pencil map (on the left) and its normalized version (on the right) from Gotanda et al., 2013

vertical slices out of the pencil map, bokeh that simulates chromatic aberration can be produced, as seen in Figure 3.13, and by using lookup tables that store the distance from the center of the bokeh, arbitrary bokeh shapes can be approximated, as shown in Figure 3.14. This approach produces correct bokeh

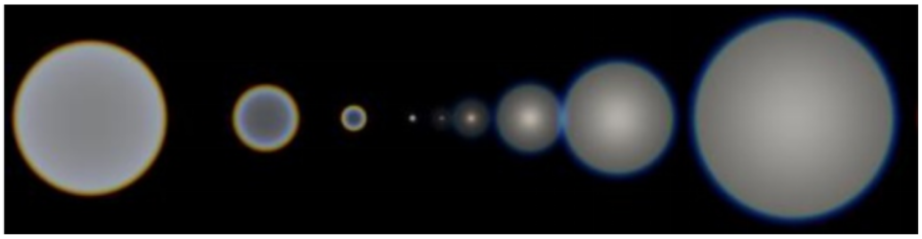


Figure 3.13: Bokeh obtained by a slice from a pencil map. From Gotanda et al., 2013

for light sources along the optical axis, and decrease in precision the further away we get from it.

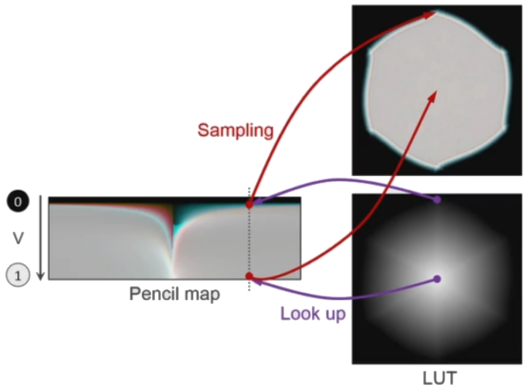



Figure 3.14: A LUT can be used to obtain arbitrary bokeh shapes. From Gotanda et al., 2013

## 3.9 Neural Rendering of depth of field

Peng et al. [2022] uses a neural network to study and correct post-processed depth of field effects.
A classic post-process effect is first applied to the image and its disparity map, then an error map is used to replace artifacts with the blurring effect coming from a neural renderer.
The neural renderer is decomposed into two subnetworks: *Adaptive Rendering Network* ARNet and *Iterative Upsampling Network* IUNet. The pipeline of their system can be seen in Figure 3.15.

The approach presented in the paper allows the neural renderer to preserve the bokeh shape of the original classical render and to produce blurs of larger sizes, which was one of the biggest limiting factors for the neural rendering approach to depth of field.
The technique used by the "classical renderer" is a scattering approach in which each pixel is scattered to its neighbor areas where the distance between them is less than the blur radius of the pixel.
The blur radius is given by:

$$r = K|d - d_f| \tag{3.21}$$

where $d$ is the disparity of the pixel, $d_f$ the disparity of the focal plane and $K$ is a user-defined blur parameter.

Error maps are generated by studying the error at depth discontinuity from a thick lens approximation. The neural renderer will be trained to predict error maps from an image to replace artifacts happening in that areas with the result of the neural renderer.
The first of the two subnets, ARNet, downsamples the image and outputs an error map and a bokeh shape, while the second one, IUNet, upsample that bokeh

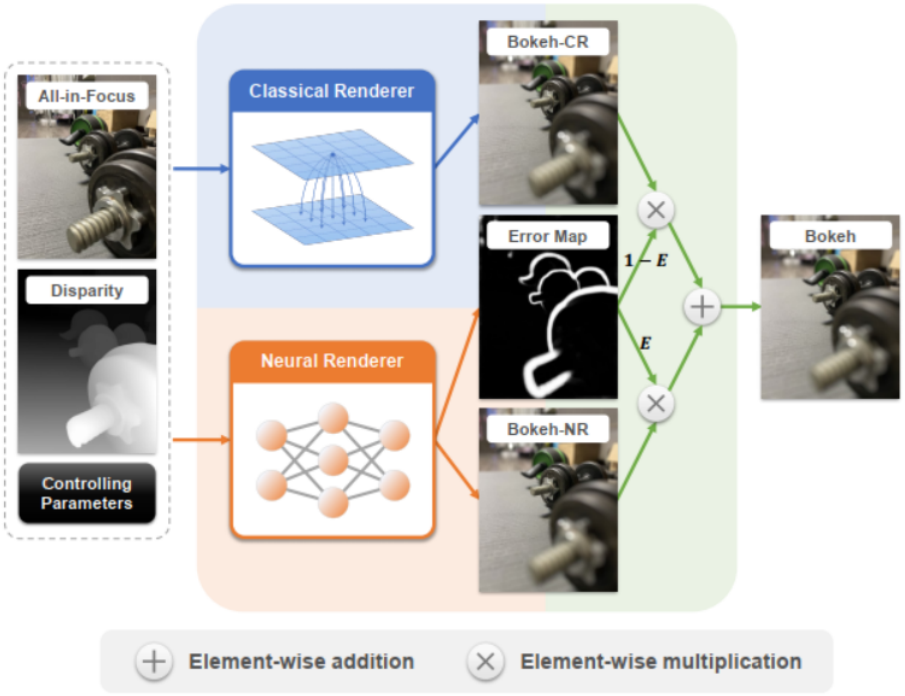



Figure 3.15: From Peng et al.

shape by a factor of 2 until the original resolution is reached.
This approach has some issues when highlights that happen to lie at the boundary of the error map. In those cases it produces a bokeh that might look too different from the others in the scene. It also works only on HDR images, and for scene with many bright lights, the simple gamma correction might not be enough.

## 3.10 Parametrically Replicating Bokeh Using Seidel Aberrations

Asberg [2020] thesis was defended at Utrecht University to conclude the Game And Media Technology M.Sc. in 2020.
The method implemented by Asberg can render very convincing Seidel and chromatic aberrations to an already rendered image that also has a depth map accompanying it.
To do so, a real camera is reduced to seven wavelength-dependent parameters that are used to compute the direction of a light ray coming from the image, through the lens and to the imaging sensor.

### 3.10.1 Preprocessing step

When a lens specification file is loaded, a preprocessing step will compute the wavelength-dependent parameters for 64 different wavelengths, which will be stored in a lookup table for future reference.
The lens is reduced first to a thick lens approximation through the use of the System Matrix. A thick lens approximates an optical system to a focal length and to the rear and front principal planes, where refraction is assumed.
The System Matrix of a lens is the result of the product of a refraction matrix $R_i$ and a transfer matrix $T_i$ for each glass element that is part of the optical system. Using the formulation from Hecht [2017], the two matrices are defined as:

$$R_i = \begin{bmatrix} 1 & -D \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -\frac{(n_{i+1}-n_i)}{r_i} \\ 0 & 1 \end{bmatrix} \tag{3.22}$$

With $D$ being the *power* of the i-th refractive surface, a measure of the degree to which a lens converges or diverges light, $r_i$ being the curvature radius of the i-th refractive surface, and $n_i$ its index of refraction;

$$T_i = \begin{bmatrix} 1 & 0 \\ -\frac{d_{i,i+1}}{n_{i+1}} & 1 \end{bmatrix} \tag{3.23}$$

With $d_{i,i+1}$ being the distance between the i-th and the i+1th surface.
The system matrix $A$ for an optical system of k elements is then defined as:

$$A = T_k R_k T_{k-1} R_{k-1} ... T_2 R_2 T_1 R_1 = \begin{bmatrix} a_1 1 & a_1 2 \\ a_2 1 & a_2 2 \end{bmatrix} \tag{3.24}$$

From the system matrix, the effective focal length can be obtained:

$$f = -\frac{1}{a_{12}} \tag{3.25}$$

And the position of the front and rear principal plane:

$$z_{fpp} = z_{firstsurface} + \frac{1 - a_{11}}{-a_{12}} \tag{3.26}$$

$$z_{rpp} = z_{lastsurface} + \frac{a_{22} - 1}{-a_{12}} \tag{3.27}$$

The other two pairs of parameters that are needed are the position and the radius of the entrance and exit pupils.
To find the position, a *chief ray* is traced through the lens system, from the sensor towards image space for the exit pupil plane and backward for the entrance pupil one.

A *chief ray* is any ray that passes through the center of the aperture and starts from an off-axis point in the object.
The position of the pupil plane is then:

$$z_{pupil} = O_{\mathbf{c}z} - \frac{O_{\mathbf{c}x,y}}{D_{\mathbf{c}x,y}} D_{\mathbf{c}z} \tag{3.28}$$

With the subscript $_{x,y}$ corresponding at the distance from the $z$-axis and $O_c, D_c$ being the origin and direction of the vector.
To find the radii of the pupils, a *marginal ray* is traced through the lens system. A marginal ray is a ray that originates from a point on the optical axis and passes through the edge of the aperture stop.
The radius is then obtained by:

$$r_{pupil} = \left[ O_m + \frac{z_{pupil} - O_{\mathbf{m}z}}{D_{\mathbf{m}z}} D_{\mathbf{m}} \right]_{x,y} \tag{3.29}$$

Similarly to what has been done for the position of the exit pupil, its radius can be found by reversing the process.

At this point all the wavelength-dependent parameters needed to apply the post-process effect have been found:

- Effective Focal length $f$;
- Position of the front and rear principal planes $z_{fpp}, z_{rpp}$;
- Position of the entrance and exit pupil $z_{en\_pupil}, z_{ex\_pupil}$;
- Radius of the entrance and exit pupil $r_{en\_pupil}, r_{ex\_pupil}$;

### 3.10.2 The aberration vector and the Seidel Coefficients

The aberration vector $\mathbf{\Delta p}$ is used to find the position of a point in object space on the Gaussian image plane, according to the Seidel aberrations explained in section 2.2.
Computing the aberration vector is not straightforward. Asberg references Born's book on Optics, Born et al. [2000], for the full derivation.
First, two normalized coordinate systems must be defined: one on the object plane and one on the image plane, such that within Gaussian Optics, the coordinates of the two planes, respectively $\mathbf{p_0}$ and $\mathbf{p_1}$, are the same: $\mathbf{p_0} = \mathbf{p_1}$.
Then two pairs of units of length are defined: the first in the object and image planes, $l_0, l_1$ and the second in the entrance and exit pupil planes $\lambda_0, \lambda_1$, such as:

$$\frac{l_1}{l_0} = M \tag{3.30}$$

$$\frac{p_1}{p_0} = M' \tag{3.31}$$

By normalizing the object and image plane coordinates, $\mathbf{p_0}, \mathbf{p_1}$ are defined:

$$\mathbf{p_0} = C\frac{\mathbf{P_0}}{l_0} \tag{3.32}$$

$$\mathbf{p_1} = C\frac{\mathbf{P_1}}{l_1} \tag{3.33}$$

Considering that $\mathbf{P_0}$ is a point on the image being processed in object space, $\mathbf{P_1}$ the corresponding point on the sensor and $C$ a constant which, assuming $n_0 = 1$ and defining $\lambda_0 \equiv 1, l_0 \equiv 1$, can be defined as $C = \frac{1}{D_0}$.
Finally:

$$\mathbf{\Delta p} = \mathbf{p_1} - \mathbf{p_0} = \frac{1}{D_1}(\mathbf{P_1} - \mathbf{P_1^*}) \tag{3.34}$$

Where $\mathbf{P_1^*}$ is the coordinate of the Gaussian Image Plane for $\mathbf{P_0}$. The process to compute $\mathbf{P_1}$ and $\mathbf{P_1^*}$ will be explained in the next section.
The aberration vector, with the Seidel Coefficients, is then obtained by a power series expansion up to the third order.

$$\mathbf{\Delta p} = \begin{bmatrix} B\rho^3 \sin\theta - 2Fy_0\rho^2 \sin\theta\cos\theta + Dy_0^2\rho\sin\theta \\ B\rho^3\cos\theta - Fy_0\rho^2(1+2\cos^2\theta) + (2C+D)y_0^2\rho\cos\theta - Ey_0^3 \end{bmatrix} \tag{3.35}$$

Where $y_0 = \mathbf{p_{0y}}$, $\rho, \theta$ are polar coordinates of a sampled point on the exit pupil plane and $B, C, D, E, F$ the Seidel coefficients that control the contribution of the Seidel aberrations.

### 3.10.3 The Seidel Coefficients

Born et al. provides simplified formulas to compute the Seidel coefficients, simplified by assuming that only spherical and rotationally symmetric surfaces are being considered:

$$\begin{aligned}
B &= \frac{1}{2}\sum_i h_i^4 K_i^2 \left(\frac{1}{n_i s_i'} - \frac{1}{n_{i-1}s_i}\right) \\
C &= \frac{1}{2}\sum_i (1 + h_i^2 k_i K_i)^2 \left(\frac{1}{n_i s_i'} - \frac{1}{n_{i-1}s_i}\right) \\
D &= \frac{1}{2}\sum_i h_i^2 k_i K_i (2 + h_i^2 k_i K_i) \left(\frac{1}{n_i s_i'} - \frac{1}{n_{i-1}s_i}\right) - K_i \left(\frac{1}{n_i^2} - \frac{1}{n_{i-1}^2}\right) \\
E &= \frac{1}{2}\sum_i k_i (1 + h_i^2 k_i K_i)(2 + h_i^2 k_i K_i) \left(\frac{1}{n_i s_i'} - \frac{1}{n_{i-1}s_i}\right) - \frac{1 + h_i^2 k_i K_i}{h_i^2} \left(\frac{1}{n_i^2} - \frac{1}{n_{i-1}^2}\right) \\
F &= \frac{1}{2}\sum_i h_i^2 K_i (1 + h_i^2 k_i K_i) \left(\frac{1}{n_i s_i'} - \frac{1}{n_{i-1}s_i}\right)
\end{aligned} \tag{3.36}$$

Where:

- $n_i$ is the refractive index of the medium that follows the $i$th surface;
- $-s_i$ is the distance between the object plane and the $i$th lens surface along the optical axis;
- $s'_i$ is the distance between the $i$th lens surface and its Gaussian image plane along the optical axis;
- $-t_i$ is the distance between the plane of the entrance pupil and the $i$th lens surface along the optical axis;
- $t'_i$ is the distance between the $i$th lens surface and the plane of the exit pupil along the optical axis;
- $K_i$ is calculated from the *Abbe relations*:

$$K_i = n_{i-1}\left(\frac{1}{r_i} - \frac{1}{s_i}\right) = n_i\left(\frac{1}{r_i} - \frac{1}{s'_i}\right) \tag{3.37}$$

  Which can be rewritten to find $s'_i$:

$$s'_i = \frac{r_i s_i n_i}{r_i n_{i-1} + s_i(n_i - n_{i-1})} \tag{3.38}$$

  and $t'_i$, by rewriting the other Abbe relation $L_i$ as described in Born's book:

$$t'_i = \frac{r_i t_i n_i}{r_i n_{i-1} + t_i(n_i - n_{i-1})} \tag{3.39}$$

- $h_i$ and $k_i$ are abbreviations introduce to simplify the equations:

$$\begin{aligned} h_1 &= \frac{s_1}{t_1 - s_1} \\ h_{i+1} &= \frac{s_{i+1}}{s'_i} h_i \end{aligned} \tag{3.40}$$

$$\begin{aligned} k_1 &= \frac{t_1(t_1 - s_1)}{n_0 s_1} \\ k_{i+1} &= k_1 + \sum_{j=1}^{i} \frac{d_j}{n_j h_j h_{j+1}} \end{aligned} \tag{3.41}$$

### 3.10.4 Applying the Seidel aberrations

With all the necessary components in place to calculate the aberration vector, it can be effectively integrated with Gaussian Optics to determine the imaging sensor point corresponding to every pixel in the image undergoing the post-process effect.
Starting from $\mathbf{P_0}$, the coordinates of the pixel of the image in object space, a random sample the exit pupil to find $\mathbf{P'_1}$ is performed. The entrance pupil

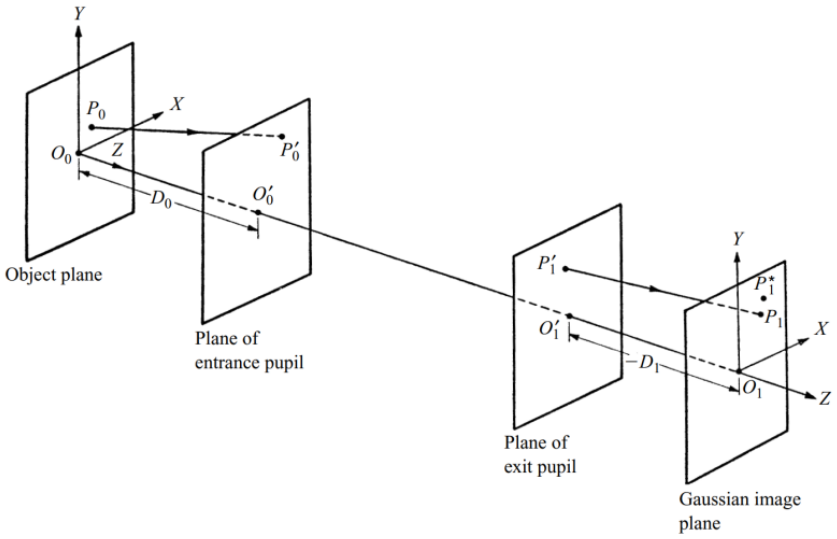



Figure 3.16: Lens system schematic from Born et al. [2000]

coordinates are found using the lateral magnification between the entrance and exit pupils $M' = \frac{r_{expupil}}{r_{enpupil}}$:

$$\mathbf{P'_0} = \frac{1}{M'}\mathbf{P'_1} \tag{3.42}$$

The Gaussian Image Plane coordinates $\mathbf{P^*_1}$ is where the imaging sensor would have to be placed in order to have $P_0$ in perfect focus in a system without aberrations. In order to find those coordinates, the need the lateral magnification between the object and the image planes $M = -\frac{f}{x_0} = -\frac{f}{z_{lightsource}+z_{fpp}-f}$. $z_{lightsource}$ need to be computed by using the depth information stored in the depth map.
$\mathbf{P^*_1}$ is then found by multiplying $M$ with $\mathbf{P_0}$:

$$\mathbf{P^*_1} = M\mathbf{P_0} \tag{3.43}$$

The aberration vector $\mathbf{\Delta P_1}$ s then summed to it to find the Gaussian Image plane coordinates of $\mathbf{P_0}$:

$$\mathbf{P_1} = \mathbf{P^*_1} + \mathbf{\Delta P_1} \tag{3.44}$$

Figure 3.16 shows the various steps needed to obtain $\mathbf{P_1}$

To find the coordinates on the imaging sensor, a ray originating on the exit pupil is traced towards a point $\mathbf{Q}$, as shown in Figure 3.17.

$\mathbf{Q}$ is the intersection point of a ray originating from $\mathbf{P'_1}$ directed towards $P_1$ with the *wavefront of the outgoing light*.

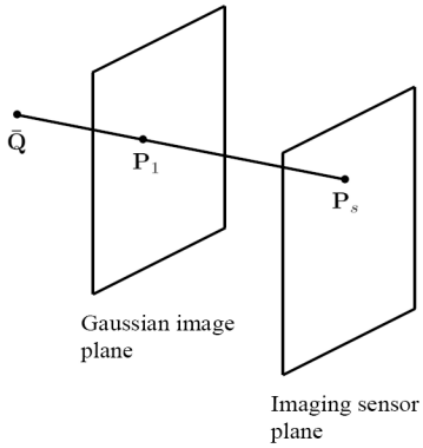



Figure 3.17: The ray that intersects the imaging sensor starts from point **Q**. From Born et al. [2000]

Since the ray intersects both the wavefront and its Gaussian approximation, a simplified *Gaussian Wavefront* can be used. The *Gaussian Wavefront* is a perfect sphere centered at $\mathbf{P_1^*}$ that also intersects the center of the exit pupil:

$$\mathbf{Q} = \mathbf{P_1^*} + \frac{\mathbf{P_1'} - \mathbf{P_1^*}}{|\mathbf{P_1'} - \mathbf{P_1^*}|} R_w \tag{3.45}$$

With $R_w = |\mathbf{P_1^*} - \mathbf{z_{expupil}}\hat{\mathbf{z}}|$ being the radius of the spherical wavefront. Finally, imaging sensor coordinate can be determined by tracing a ray from $\mathbf{P_1}$ with a direction $\mathbf{D} = \mathbf{P_1} - \mathbf{Q}$. The distance of the imaging sensor from the Gaussian Image Plane can be computed, so the coordinates $P_s$ can be found without actually computing the intersection:

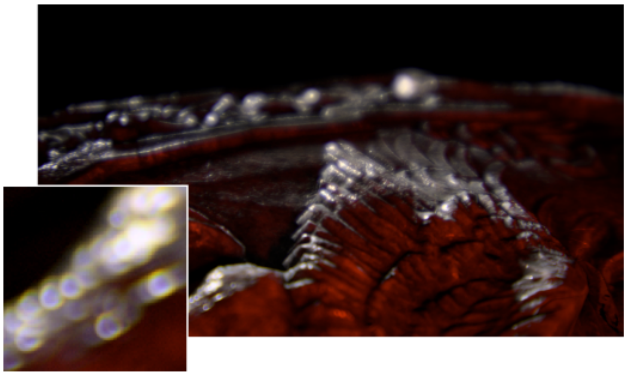
$$\mathbf{P_s} = \mathbf{P_1} + d_{image->sensor} \frac{\mathbf{D}}{\mathbf{D} \cdot \hat{\mathbf{z}}} \tag{3.46}$$

$d_{image->sensor}$ can be computed by first computing the z-coordinate of the sensor plane and of the Gaussian image plane.

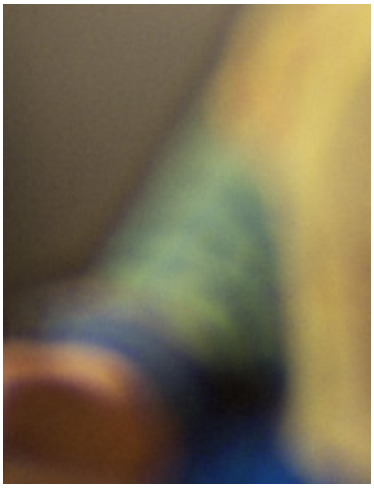
### 3.10.5 Implementation and results

Asberg's thesis can reproduce convincing bokeh with both chromatic and monochromatic aberrations. An example of the result can be seen in Figure 3.18a.
Her renders were compared to a Screen Space Ray Tracing, implemented according to Kolb et al. [1995], to the pencil map approach by Gotanda et al. and to a variation of her original approach using only Gaussian optics (no aberrations) for a specific set of lenses. In most cases, using RMSE and MS-SSIM, the

Seidel Aberration approach was the most similar to the ground truth, with the advantage that the rendering time does not increase with the number of lens elements in the system.



(a) A render made using Asberg implementation. Notice both the chromatic and monochromatic aberrations in the bokeh



(b) Notice the thick dark border in this image. The information on the background color is missing.

Figure 3.18: Results from Asberg's thesis.

Even reducing the number of parameters from seven to just three, which were observed to mostly influence the bokeh shape, results in a good approximation of the bokeh. Unfortunately, the partial occlusion is still an issue. In Figure 3.18b, a strong border behind the silhouette of the in-focus subject can be seen.

## 3.11 Summary

In conclusion, there are many approaches to tackle the limitations of the pinhole camera model, each with its unique trade-offs and considerations.
When choosing between any of those techniques, the developer needs to keep in mind that there is always a trade-off between efficiency, aesthetic quality, physical plausibility and ease of use from the user.
While an ideal post-process effect would be able to replicate photorealistic Depth Of Field from real-life lens specification in real-time, such an astonishing technique might still not be suitable for certain production, where the technical artist needs to have a fine control on the shape of the bokeh in each frame, for example when applying VFX to an existing real-life footage.

When implementing a Depth Of Field effect into a rendering engine, one must also keep in mind each approach's limitations. Whether grappling with the partial visibility of all image-space techniques or addressing the complexities of depth discontinuities in filtering approaches like Zhou et al. [2007], Mcgraw [2014] or Kass et al. [2006] or the limited realistic models of object-space approaches like the thin lens approximation shown in Cook et al. [1984] or the choice of not studying more complex light-glass interaction like in Wu et al. [2010], Wu et al. [2012] and Kolb et al. [1995], it's essential to acknowledge the nuances of different methods.

More modern approaches like those using neural networks (Qian et al. [2020]) or hybrid techniques like Asberg [2020], Jeong et al. [2022], Gotanda et al. [2015] and Abadie [2018] are extremely promising and produce good results, although the ever-advancing hardware landscape keeps this research area far from saturation.

In conclusion, Asberg approach is interesting as it is an image space effect that produces physically accurate bokeh, replicating both chromatic and monochromatic aberration.
Despite the challenge of partial visibility, the implementation is straightforward and adaptable to any image with an attached depth map. While certain limitations persists, including the difficulty for the user to manually tweak the effect to achieve the desired result, the rendered aberrations enhance and increase the realism of every render, marking a significant advancement in DoF rendering.

# Chapter 4

# Irreversibility

This chapter will cover what reversing Asberg's technique meant for the purpose of this thesis, the challenges met during this research and the answer to the first research question.

## 4.1 Reversing Asberg [2020]

The Research Question 1 (RQ1) was the following:

**RQ1: Can we reverse Asberg's innovative approach and expand it in order to produce primary rays for breda, Traverse Research's[1] real-time rendering framework?**

In her thesis, Asberg proposes a way to render a post-process photorealistic Depth Of Field starting from an image, its accompanying depth map and a file that specifies the position and the characteristics of every element in an optical system. This innovative approach uses Gaussian Optics, Seidel Aberrations theory and the characteristics of light as a wave to compute, for each pixel of the rendered image, its aberrated position on the imaging sensor without tracing any ray through the actual optical system, except for the preprocess step that computes the camera parameters.
Inverting this process in order to produce a distribution of rays coming out of the lens and into the scene, means, for every pixel of the imaging sensor, sampling one or more of the possible aberrated rays that would reach this pixel to trace it back into the scene.

A positive answer to the first research question would allow the use of the same Seidel Aberrations theory to generate rays for a real-time ray tracer, rendering a photorealistic Depth Of Field effect with both chromatic and monochromatic aberrations and little overhead on the rendering time.

[1] `https://traverseresearch.nl`

The idea was to get the equations that Asberg introduced in her thesis but solving them for this new objective, arriving at the following final procedure:

1. For each pixel of the imaging sensor:
2. Sample the entrance pupil to find $P'_0$, find the conjugate point on the exit pupil $P'_1$ using the lateral magnification $M'$: $P'_1 = M'P'_0$;
3. Find the Gaussian Image Plane coordinates (GIP) $P^*_1$. In this scenario the GIP coordinates are the coordinates of all the point in object space world that would be in perfect focus in the pixel we are tracing rays from: $P^*_1 = MP_1$, with $M$ being the lateral magnification between the image plane and the object plane. In this scenario the image plane would be approximated with the focus distance;
4. Using the precomputed aberration vector $\Delta P_1$ to find the aberrated coordinates of the current pixel on the GIP;
5. Find the point $Q$ on the Gaussian Wavefront (like explained in section 3.10): $Q = P^*_1 + \frac{P'_0 - P^*_1}{|P'_1 - P^*_1|} R_w$;
6. Finally trace the rays that originates in $P_1$ and with direction $Q - P_1$ into the scene;

Mathematically, this approach worked, and the Helmholtz reciprocity principle ensures that the light ray can be evaluated from both direction.
In the following section, the reason why this reversed process cannot work will be explored, but it is important to note how this was implemented successfully until the introduction of wavefront optics. In fact, it is perfectly possible to find the Gaussian Image Plane coordinates of a point from either side of the lens, but the wavefront optics is strongly dependent on the complex interaction between glass and light, therefore inverting this path is not as trivial as it originally thought.

## 4.2 Challenges in Reversing Asberg's Approach

The Seidel coefficients are usually computed by expanding the wavefront of the light passing through the optical system into a series of mathematical terms. To obtain the values of those terms for a given system, a specific set of rays, arranged in a two-dimensional fan pattern, is traced through it and the deviation from an ideal reference point is used to compute the Seidel coefficients.

The Helmholtz reciprocity principle, as introduced in von Helmholtz [1867], describes how a ray of light and its reverse ray match an optical path, meaning that an optical path can be evaluated in any direction. While the Helmholtz reciprocity principle applies to each individual ray within the two-dimensional fan, it cannot be applied to the fan as a whole. In other words, an optical path

can be evaluated in any direction using the reciprocity principle, but the entire fan of rays cannot be reversed as a single entity.

When trying to reverse the path by tracing rays from the sensor to the world, the goal is trying to reverse the path of the bidimensional fan of rays that converged onto a particular point of the sensor. However, each individual ray has been aberrated differently by the optical train according to the original direction and angle of entry into the optical system.
The Seidel aberrations represent a global property of the system, characterizing how the wavefront is displaced by the system overall, and to apply the reciprocity principle it would be necessary to trace the rays locally through all the surfaces that are part of the system.
To make it clear, the case of a lens exhibiting spherical aberration can be considered. Light rays passing through the lens from the world to the sensor will converge at different points on the latter, as shown in Figure 2.9. Reversing the direction of the light through the system will not produce the same effect, as aberrations will act differently on it, by causing different focal points and a distinct final image formation.

Sampling all possible paths that terminate at a specific point on the sensor and subsequently weighting them with a suitable Probability Distribution Function (PDF) was considered.
Unfortunately building such a Probability Distribution Function presents its challenges. During the traversal of the optical system, information about the original path is lost due to multiple complex events like diffractions, reflections, and scattering. Since a PDF assumes a known distribution of paths, even if such an approximation could be built, this would likely result in unreliable outcomes.

## 4.3 The answer to RQ1

In conclusion, the answer to the first research question is negative.
The approach proved to be non-reversible due to the unique aberration effects experienced by the individual rays that comprise the wavefront and the complex nature of light interactions within an optical system.
An existing alternative to apply a Depth Of Field effect that simulates the aberration of real lenses without ray tracing them is to use polynomial optics, like in the papers from Hullin et al. [2012] and Hanika and Dachsbacher [2014].
Furthermore, it is not excluded that a non-trivial way of using Seidel Aberrations and Wavefront Optics to generate a distribution of primary rays for a real-time ray tracer exists, but it was decided to not investigate this further, as it was not the main scope of this thesis anymore.

# Chapter 5

# A Hybrid Approach

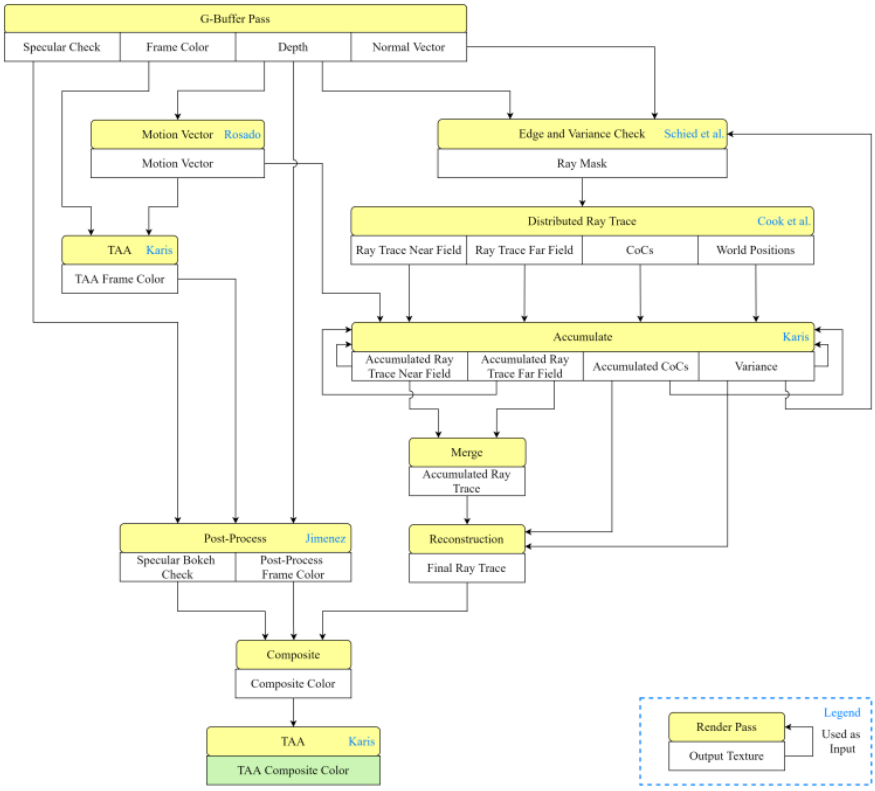



Figure 5.1: The pipeline for the hybrid rendering from Tan et al.

In chapter 4, the issues with reversing Asberg's innovative DoF approach were explored, and the first research question was answered negatively due to

complex optical phenomena.
In this chapter we shift our focus to the second research question to explore different innovative Depth Of Field techniques by implementing the techniques introduced in the paper by Tan et al. [2022] within **breda** ray tracing framework.
This chapter describes the paper's original approach and a report on **breda**'s implementation, highlighting the issues encountered and the needed guesswork and workarounds that were necessary due to ambiguities in the original paper and the lack of a reference implementation.

In Tan et al. [2022], the authors address the partial visibility problem that affects every post-process technique by adding a ray tracing pass to their pipeline. Despite not reaching the state-of-the-art results from Abadie's implementation in Unreal Engine Abadie [2018], it is nonetheless an interesting approach as it is the first that implements ray tracing as part of a post-process Depth Of Field effect.

Their approach is divided into two macro steps: a post-process effect that uses a filtering kernel to blur the image and apply the depth-of-field effect to it, and a ray tracing step using a ray mask, that sends a different number of rays where the probability of finding a partially visible edge are higher: at the edges of foreground objects and in areas with high luminance variation. Figure 5.1 shows the pipeline of the hybrid renderer.

The screen space blurring technique is inspired by Jimenez [2014], with some adjustments to account for the ray tracing pass.
Jimenez [2014] introduced a scatter-as-you-gather approach to produce the blurred image from the rasterized image downscaled to half resolution. section 5.0.1 introduces the steps that Tan et al. [2022] uses to apply the post-process Depth Of Field filter while section 5.0.2 shows how ray tracing is applied to solve the partial visibility problem through a ray tracing mask.
**breda**'s implementation of the paper is discussed in section 5.1.

### 5.0.1 Post-process Filtering

For the post-process step, both Jimenez [2014] and Tan et al. [2022] use three separate passes to apply the Depth Of Field effect using the gathering approach: a *Prefilter Pass*, the *Main Filter Pass*, and a *Post Filter* or *Reconstruction Pass*. Before these steps, a *tiling and dilate pass* is performed to compute the closest depth and maximum Circle of Confusion (CoC) for each tile in the image from the depth map.

The scatter-as-you-gather approach introduced by Jimenez classifies the samples of the main filter in foreground and background using the nearest depth stored in the tile from the tiling pass and performs an additive alpha blending average for each of the two layers.
The sample classification is continuous, as shown in Figure 5.2: a distance of 0

means that the sample is in the foreground, while a distance of 100 means it is in the background.

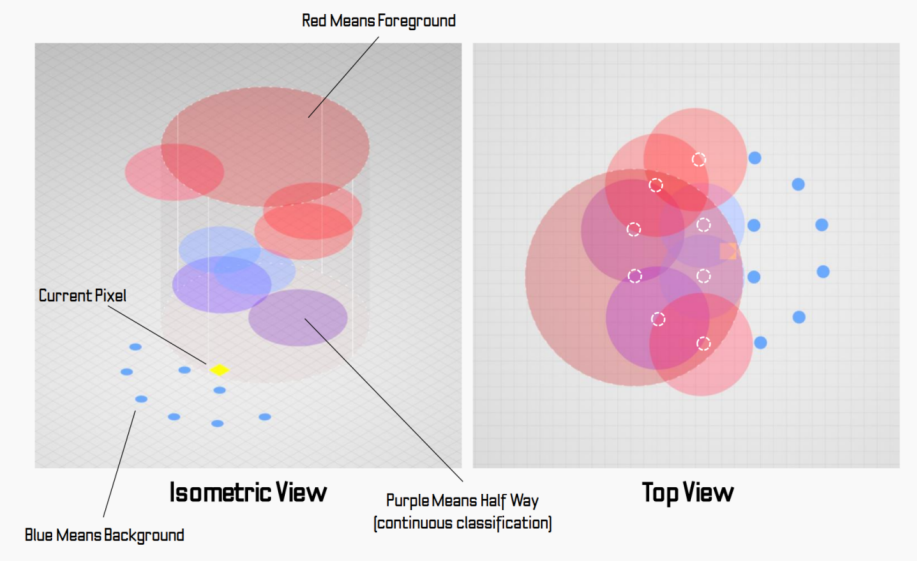



Figure 5.2: Scatter-as-you-gather approach as shown in Jimenez's presentation.

In the *Prefilter Pass*, the color buffer is downsampled using a 9-taps bilateral filter scaled to 1/8 of the maximum CoC of the pixel's tile. This bilateral filter is a non-linear filter that preserves edges while smoothing the image. The size of the kernel is capped to the diagonal of one pixel: $\sqrt{2}$. Additionally, the depth buffer is downsampled to half resolution by taking the maximum depth for each 2x2 tile.
The prefilter pass also includes a *presort pass*, which categorizes each pixel as either a background or foreground pixel based on a comparison with the closest depth in each tile. While not a strict foreground/background classification, this process produces weights for both the background and foreground. These weights are then multiplied with the alpha value that will be used for the additive alpha blending in the main pass.

For each pixel, the *Main Pass* applies an 81-tap filter with 4 rings of samples, as shown in Figure 5.3.
In the implementation by Jimenez [2014], the main filter has 48 taps distributed on 3 rings, and the normalized alpha of the foreground is used to perform the additive alpha blending. Tan et al. [2022] found the result unsatisfactory and decided to increase the number of samples in the filter and to use a normalized weighted sum of foreground and background contribution for the post-process color $v_p$ using the following formula:

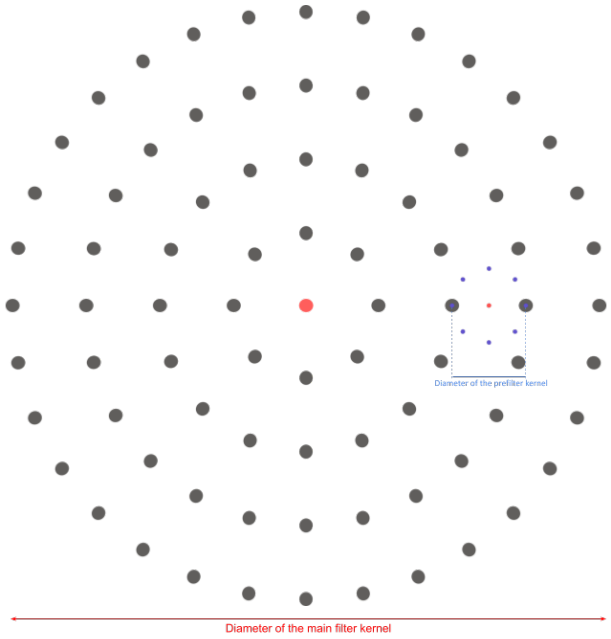



Figure 5.3: The 9-tap bilateral filter that fills the gaps in samples left by the 81-tap main filter.

$$v_p = \frac{v_f + v_b}{\sum_{i=1}^{81} D(0,i)\dot{s}ampleAlpha(r_i)} \tag{5.1}$$

Where *sampleAlpha* is the function used in the prefilter, $D(0,i)$ being the comparison of the CoC of sample $i$ to its distance to the center tap of the kernel, $r_i$ being the CoC radius of sample $i$ and $v_f$ and $v_b$ representing the accumulated color for the foreground and background.
Another difference from the original implementation is the added gathering of the proportion of samples with high specular values for each pixel, used later on to composite the ray-traced texture over the post-processed one.

Finally, in the *Postfilter Pass*, to fight undersampling, a 3x3 median filter by Smith [1996] is applied to the half resolution buffer to reduce noise efficiently using hardware $min3$ and $max3$ functions.

### 5.0.2 Ray Tracing missing geometry

The Partial Occlusion problem arises when objects in the scene partially occlude each other or have complex shapes or edges. In those situations, when

the entire image is blurred through the filtering kernel, the blurred pixels of the foreground object may overlap with the in-focus pixels of the background shape, but since the information of the part of the background that was hidden from the foreground is missing in rasterized images, often those kinds of effects try to find a local solution using neighboring pixels or by extending the foreground. An example has already been shown in Figure 3.1.
To tackle this problem, Tan et al. [2022] implements an edge detection pass from which a ray mask is built. The ray mask is then used to selectively trace the scene and fill in the missing geometry information for regions of interest. This adaptive ray tracing approach allows the results in a more accurate and visually appealing Depth Of Field effect.

To build the ray mask, the half-resolution GBuffer goes through a Gaussian filter, to reduce noise, and a 5 x 5 Sobel convolution kernel, which gives an estimation of how extreme an edge is.
Both depth and normal derivatives are used: the former captures the separation between overlapping objects, while the latter can detect variation in the orientation of the primitive faces within the objects themselves.

With $\delta_d$ and $\delta_n$, being respectively the magnitude of the derivative of depth and surface normals in the 5 x 5 tile, the per-pixel output $x_n$ is the following:

$$x = (\delta_d + \delta_n) \cdot s, s \in [0, 1] \tag{5.2}$$

$$x_n = saturate(1 - \frac{1}{x + 1}) \tag{5.3}$$

With $s$ being a user-specified scaling factor.
To shoot a different number of rays per pixel, based on the variance in luminance (Schied et al. [2017]), a per-pixel ray count $x_f$ is also computed using a temporally-accumulated variance estimate $\sigma^2$:

$$x_f = saturate(x_n + \sigma^2 \cdot 100000) \cdot m \tag{5.4}$$

The large weight that multiplies $\sigma^2$, 100000, has been chosen empirically by the authors of the paper, $m$ is the maximum number of rays per pixel to be shot.

The rays are shot on half resolution using Cook et al. [1984] Distributed Ray Tracing technique to sample the aperture of a thin lens.
Samples are accumulated into different textures based on their distance from the current focus distance. Reprojection of both the near and far-field is also used to account for movement. For the former, velocity vectors computed as in Rosado [2008] are used, while for the far field, the average far-field world position must be computed first, as it could happen that under low ray counts, there might be no far field hit at all.
The ray-traced near and far-field are then merged into a single texture by blending between them using the hit ratio $h$ of foreground hit over the number of rays being shot:

$$color = lerp(far, near, h); \tag{5.5}$$

From this merged texture, an estimated variance texture is computed following Schied et al. [2017].
The estimated variance for each pixel $\sigma_i^2$ is then used to direct more rays toward regions that might contain more noise than others.
Finally, before the compositing step, the ray-traced texture undergoes a spatial reconstruction step, in which the texture is median filtered, and then a circular kernel scaled to the average circle of confusion size for that frame is used to gather the surrounding colors' contribution of neighboring pixels.
The variance estimates are used to blend between the accumulated color and one of the target pixels. The variance estimates $\sigma^2$ is multiplied by a large weight (the authors choose 2000 based on observations of various renderings) and then clamped:

$$b = clamp\left(\sigma^2 \cdot 2000, 0, 0.9\right) \tag{5.6}$$

### 5.0.3 Compositing

In the end, the ray-traced, the post-processed, and the unblurred colors are blended to produce the final depth-of-field image.
The zone of focus is defined as the range of z-values in which the size of the circle of confusion of the pixels is less than the pixel "size" $\sqrt(2)$ and can be derived from the thin lens equation:

$$\frac{a \cdot f \cdot d}{\left(a \cdot f + \sqrt{2}\,(d - f) \cdot \frac{w_s}{w_i}\right)} \leq z \leq \frac{a \cdot f \cdot d}{\left(a \cdot f - \sqrt{2}\,(d - f) \cdot \frac{w_s}{w_i}\right)} \tag{5.7}$$

The variables include $a$ for the aperture diameter, $f$ for the focal length, $d$ for the focus distance, $w_s$ for the sensor's width in meters, and $w_i$ for the image's width in pixels.
In the region of focus, the sharp rasterized image is applied. For objects in the near-field and their silhouettes, the upscaled ray-traced color is favored over the post-processed one to ensure proper handling of semi-transparency.
In the case of objects in the far-field, a blending of the upscaled post-processed color with the upscaled ray-traced one is employed. The hit ratio is used for interpolation between the two, with the caveat that blending is performed only when the hit ratio $h$ is less than or equal to 0.3, aiming to minimize blur discontinuities and tiling artifacts. When the hit ratio exceeds this threshold, the post-process color is applied.

### 5.0.4 Paper's Results

The authors of the paper implemented this approach using the Falcor API from NVIDIA, achieving relatively interactive frame rates without delving too deep into optimization.

Two different scenes were tested for the results: *Pink Room* and *Bistro Exterior*, each with a different number of max rays per pixel $m$. The spent on the ray tracing step naturally increases with the number of rays per pixel, but their screenshots clearly show how their approach (c) closely matches the ground truth of a distributed ray tracer (d) and improves on the reconstructed background of both Unreal Engine 4 DoF implementation (b) and Jimenez post-process effect (a), as shown in Figure 5.4.

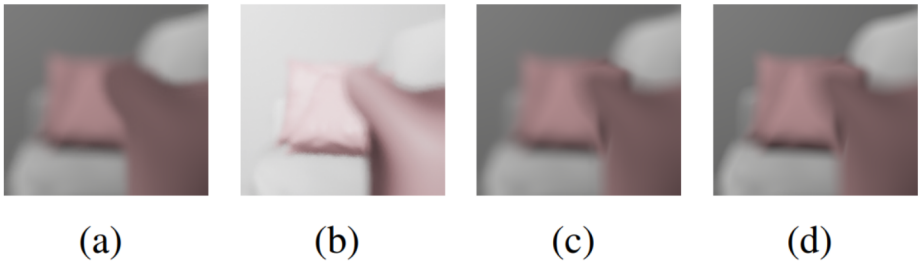



Figure 5.4: Comparison of different Depth Of Field implementation, from Tan et al. paper

However, it is essential to acknowledge some existing problems in their implementation. Tiling artifacts arise from both the post-process effect and a tiling of the ray-mask, some ghosting during movement (likely due to TAA) and the presence of noise when only one sample per pixel is being used. Despite all of this, the approach is interesting and worth investigating further.

## 5.1 Implementation in breda

**breda** is a Rust-based rendering framework developed by **Traverse Research**. It facilitates rapid prototyping of novel rendering techniques by providing a wide range of libraries for tasks such as scene parsing, ray tracing, and applying post-processing effects.

It was decided to implement the Depth Of Field effect as part of the *wavefront-path-tracer* mode of *breda-ray-tracer* application. The latter is a hybrid renderer that employs a rasterizer followed by Ray Tracing for tasks like shadow rendering and reflections.
The choice to work on the wavefront path tracer was made because it was easier to match the output of the ray tracing pass with the full-resolution unblurred texture produced by **breda**, but this does not preclude compatibility with the hybrid renderer, which will only require changing the ray tracing shader of the Depth Of Field pipeline.

Our implementation differs from the one of Tan et al. [2022]'s paper because of ambiguities in the paper and lack of both communication from the authors and access to an existing implementation. The following sections will describe the implementation of the paper in **breda**, highlighting the challenges met during its development and the differences from the reference implementation.
A pipeline of our implementation can be found in Figure 5.5.

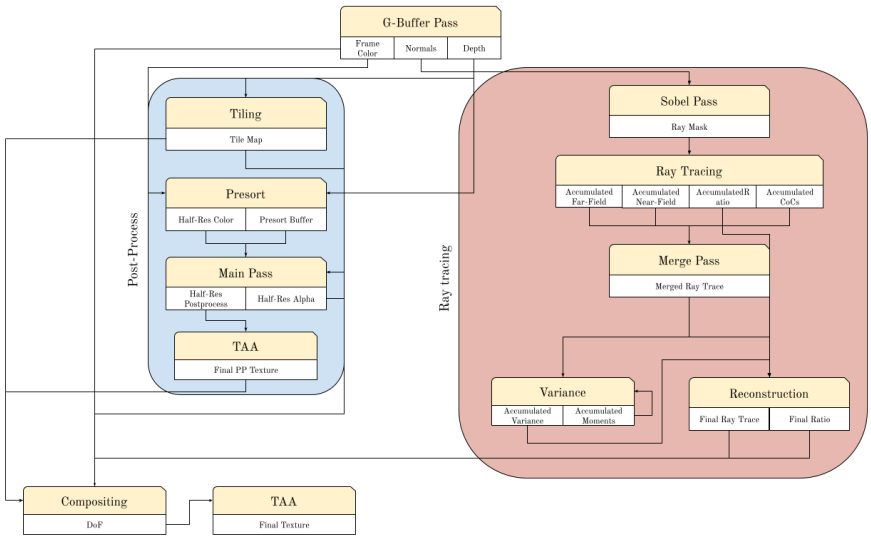


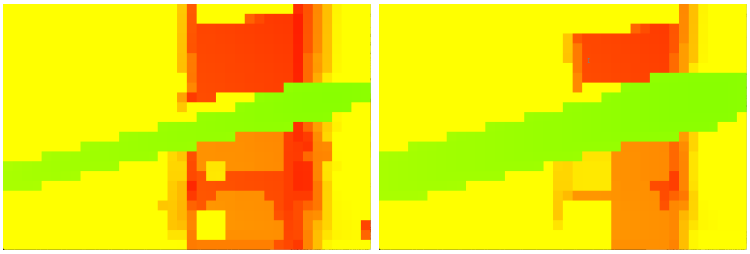

Figure 5.5: The pipeline for our implementation of the hybrid DoF rendering

### 5.1.1 Post-process

The post-process step is a reimplementation of Jimenez [2014], as introduced in the previous section. The objective is to weigh each pixel both as background or foreground, and then use the main filter pass to blend between background and foreground and produce the final result.

It begins by tiling the jittered depth map, each tile of 32x32 pixels, by using separate vertical and horizontal passes for performance optimizations. This pass produces a float2 texture containing, for each tile, the nearest depth and the diameter of the largest Circle of Confusion (CoC) among its pixels.

The tiled map is then dilated to smooth out artifacts that might have been introduced at tile boundaries: each tile looks for its immediate neighbors and stores the nearest depth and biggest CoC diameter within all the tiles in a 3x3 window. Figure 5.6 shows a screenshot of the tiled depth Figure 5.6a and its dilated version Figure 5.6b.
The red channel of the texture stores the closest depth of the tile, while the green channel stores the diameter of the largest CoC.



(a) (b)

Figure 5.6: The depth map tiled into 32x32 tiles and its dilated version.

As suggested by Jimenez, presorting, downsampling, and prefiltering of the full-resolution image can be combined into a single pass.
The presort pass takes the farthest z-depth in a 2x2 window around the sampled pixel and compares it with the nearest z-depth from the dilated map. The function shown in Listing 5.1.2 produces a weight for both the background and foreground classification of the pixel. The return value represents the likelihood that the pixel belongs to the foreground (depthCmp.x) or the background (depthCmp.y).

```
float2 depthCmp2(DofConstants constants, float depth,
    float closestDepth) {
    float d = (constants.focusDistance * 1.5) * (
        depth - closestDepth);
    float2 depthCmp;
    depthCmp.x = smoothstep(0.0, 1.0, d);
```

```
        depthCmp.y = 1.0 - depthCmp.x;
        return depthCmp;
    }
```

The result of this operation is then multiplied by a computed alpha value for the additive alpha blending average in the main filter pass that is obtained by the radius of the circle of confusion of that tile and stored into a presort texture, shown in Figure 5.7. This texture encodes in the red channel the diameter of the CoC relative to the furthest z-depth in a 2 x 2 window around the current pixel, in the green and blue channels the precomputed weight for the background and the foreground respectively of the current pixel.
As mentioned previously, this step is needed to reduce the computational load of the main blurring pass.

The prefilter step is needed to fill in the pixels of the original image that will

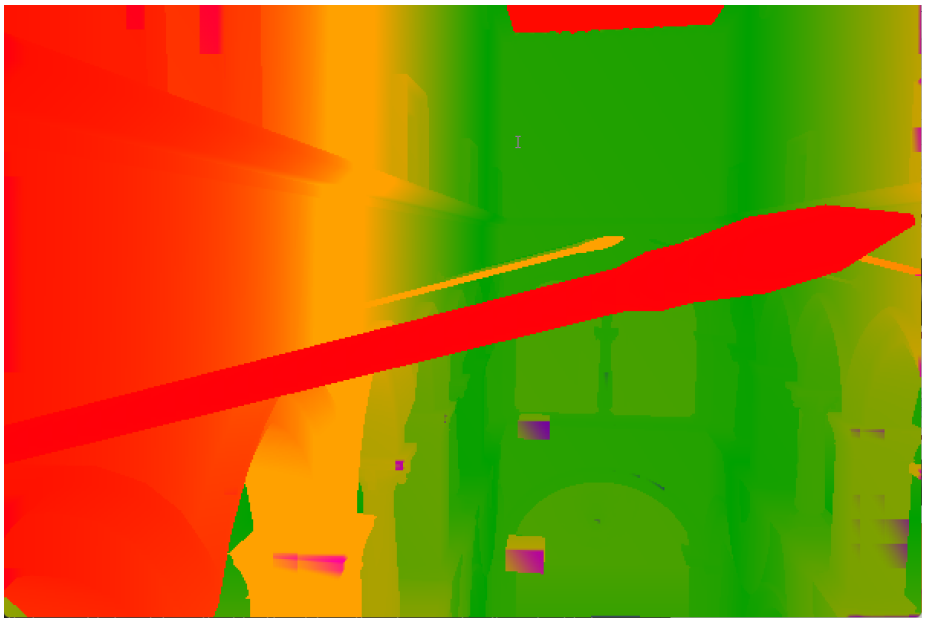


Figure 5.7: The presort texture generated by the presort pass.

be left unsampled by the kernel of the main pass as previously shown in Figure 5.3. As suggested in Tan et al. [2022] the size of the prefilter kernel has a diameter of $\frac{1}{8}$ of the diameter of the maximum circle of confusion of the tile instead of the one suggested by Jimenez of $\frac{1}{6}$, as the number of taps of the main filter has been increased from 48 to 81 for better quality of the filtering despite performances. The prefilter applies depth bilateral weighting. The minimum size of the prefilter radius is capped at the radius of a single pixel.
To fight the aliasing issues that might arise when using a flat kernel, Jimenez suggests using a Karis average (Karis [2013]) during prefilter that trades tem-

poral stability for Bokeh contrast.
This step is also responsible for downsampling the original texture to half-resolution using bilinear sampling. Figure 5.8 shows the downsampled texture that has been through the prefilter pass, ready for the main filter that finalizes the Depth Of Field effect.

The main pass uses the increased size of the filter suggested by Tan et al.

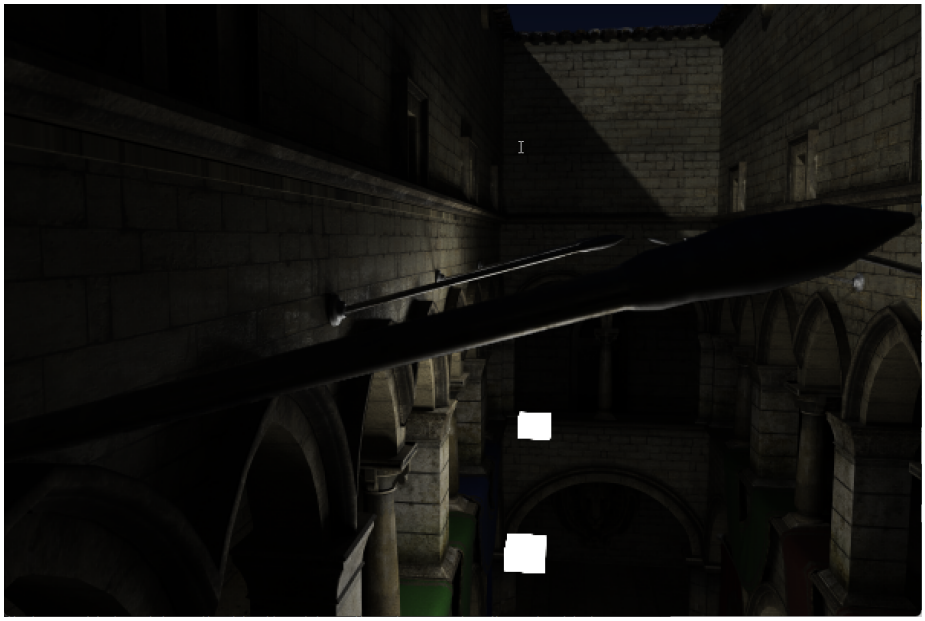


Figure 5.8: The downsampled texture generated by the prefilter step.

[2022] of 81-tap distributed across four rings scaled using the maximum circle of confusion size within the pixel's tile.
For each sample collected by the kernel, its presort weights are sampled by using point sampling and a randomized offset, and its Depth Of Field spread is computed by the code shown in Listing 5.1.1.

```
float spreadCmp(float offsetCoc, float sampleCoc, float
    pixelToSampleUnitsScale) {
     float spread = saturate(pixelToSampleUnitsScale *
         sampleCoc - offsetCoc + 1.0);
     return offsetCoc <= 1.0 ? pow(spread,
         kDofSpreadToePower) : spread;
}
```

The spread value returned by this function represents how much each sample should contribute to the background and foreground blurring.

The samples are accumulated into two float4 registers for the background and the foreground. The fourth component of each register stores the accumulated weight.
An additive alpha value is computed based on the accumulated alpha value of foreground samples which is then used to linearly interpolate between accumulated foreground and background to obtain the post-processed image shown in Figure 5.9. The accumulated alpha is also stored in a separate texture to be used in the compositing step.
The result of the post-process step gets stabilized using **breda**'s TAA pass.

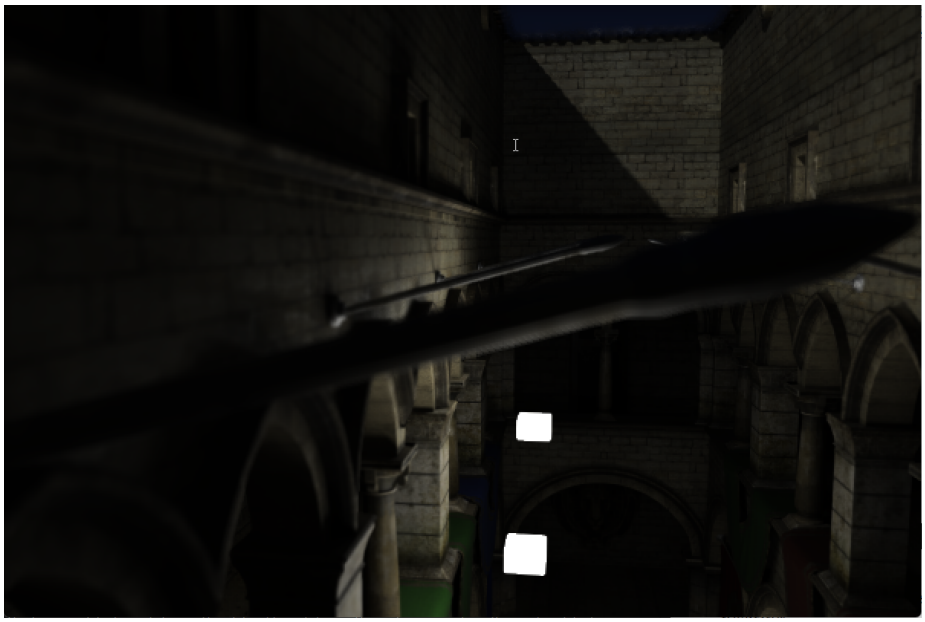


Figure 5.9: The final result of Jimenez's post-process technique before compositing.

### 5.1.2 Ray Tracing

The Ray Tracing step is used to fill in the missing geometry at the edges of the foreground objects.
The different passes that are part of the ray tracing step were described in Tan et al. [2022], but the text left a lot of guesswork to do, and it has some ambiguities to it.
For this reason, the re-implementation of the ray tracing step is probably quite different from a possible reference implementation, and the following sections will highlight which of the proposed passes did not deliver what was promised in the paper.

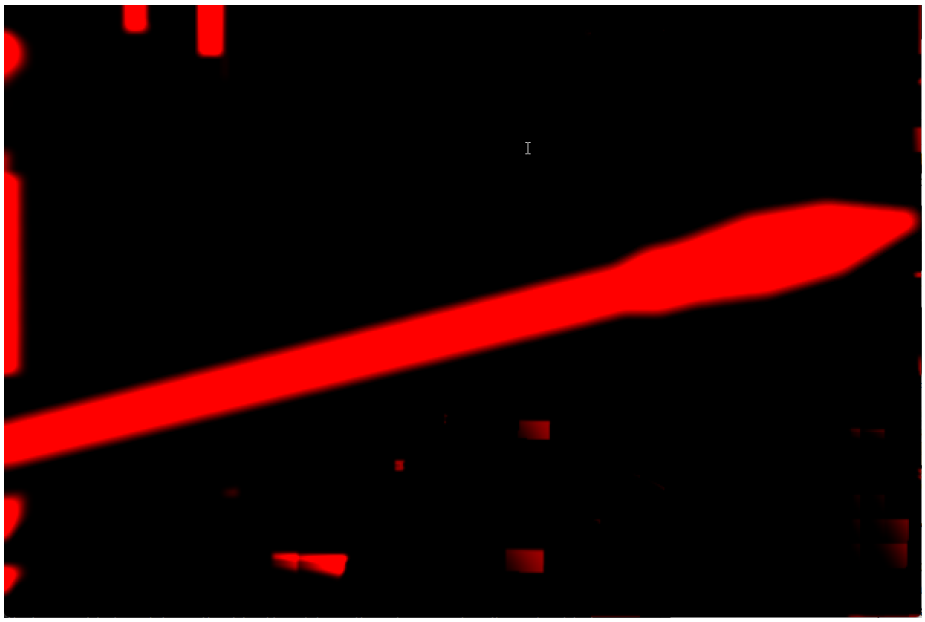


Figure 5.10: The additive alpha computed on the foreground's 4th component.

All Ray Tracing steps are executed at half-resolution and the resulting textures are then upscaled to full-resolution before the compositing step.

### Ray mask generation

This chapter, in the original paper, is not clear enough on how the ray mask is generated. Before discussing the ambiguities of this process, it will be introduced as-is to be able to understand it properly.

Following the instructions found in Tan et al. [2022], an edge mask is first generated using a 5 x 5 Sobel pass on the depth and the normal buffer.
In addition to this, because **breda**'s depth map is jittered at each frame, some sort of temporal stabilization of the ray mask is applied by looking at the previous frame. If the pixel at the current frame is not part of an edge, but it was in the previous frame and the accumulation is valid (i.e. no movement or changes in the depth of field parameters), the current pixel will still be added to the ray mask of the current frame.
This is needed to avoid pixels being added and removed from the edge mask at each frame because of the jittered depth map.

The ray mask containing the number of rays to be traced per pixel $x_f$ is com-

puted as follows:

$$x = (\delta_n + \delta_z) \cdot s \tag{5.8}$$

with $\delta_n$ and $\delta_z$ being the magnitude of the derivative of the Normal map and Depth map outputted by the Sobel pass and $s$ a user-defined value to scale down the result;

The result gets then normalized:

$$x_n = saturate\left(1 - \frac{1}{x + 1}\right) \tag{5.9}$$

The number of rays per pixel $x_f$ is then computed by complementing the normalized value with a temporally-accumulated variance estimate $\sigma^2$ to aim more rays toward the areas of the scene with higher variance:

$$x_f = saturate\left(x_n + \sigma^2 \cdot 100000\right) \times m \tag{5.10}$$

In this formula, 100000 is a value chosen experimentally by the authors of the paper to boost the weight of the variance map and $m$ is the maximum budget of rays to be shot per pixel for a single frame.
Figure 5.11 shows the texture in which the number of rays per pixel is being stored.

In their paper, the authors claim to be tracing at least one ray per pixel in the near field. The meaning of this is not entirely clear. We ended up using $x_f$ as computed above without accounting for the near field.
We also tested the way it was suggested in the paper by checking if the pixel's world space depth was less than the focus distance $d$: $z_i \leq d$. If this is the case, we *clamp* $x_f$ in the range $[1; m]$ so that every pixel in the near field will be traced. Otherwise, we use the value of $x_f$ as previously explained. This resulted in improved partial visibility, as shown in Figure 5.12, but it created some very visible discontinuities on bright bokeh, as shown in Figure 5.13.
We tried tweaking the composite step to prefer the post-processed image rather than the ray-traced when the luminance of a sample was higher than a certain threshold, but the issue persisted.
We believe that this result could be improved with a better compositing strategy than the one discussed in the original paper, but for lack of time, we ended up preferring slightly worse see-through edges than discontinuities in bright bokeh.

The way the number of rays to be traced per pixel is computed in the paper causes most of the pixels to tend towards having $m$ rays per pixel. This will rapidly decrease performance with values of $m$ higher than 5, as in complex scenes with a lot of background and foreground objects, an entire ray mask with 10 or more rays per pixel will be quite slow to trace.

Figure 5.11: The ray-mask. The red component stores $x_f$.

It was also impossible to use the magnitude derived from the depth map. This increases quite rapidly the further we get from the camera, and after a certain distance, every flat surface will be considered as an edge. Probably a better technique to detect edges involving both depth and normals would improve the resulting ray mask, and help increase the performances of the entire effect.

Because of lack of time, it was not possible to investigate this further and find a solution or a better technique to generate a ray mask. Therefore, it was decided to only use the normal map, with $m = 5$ for all our tests.

### Raytracing step

The ray mask is then used in the ray tracing step. Rays are generated using the same technique described in Cook et al. [1984], in which an aperture, its size set by the user according to the desired Depth Of Field effect, is sampled to generate a distribution of rays going into the scene as shown in Figure 5.14. The user specifies the f-stop $N$, which is then converted into the aperture diameter $A$ by

$$A = f/N \tag{5.11}$$

With $f$ being the current focal length.

Rays will be directed towards the edges identified in the previous step, and because we are sampling an aperture rather than using a classic pinhole camera

(a) Detail of the partially occluded rod using $x_f$ as computed in the paper.

(b) Detail of the partially occluded rod using at least one ray for every foreground pixel.

Figure 5.12: Better partial occlusion with at least one sample per pixel in the near field.

(a) Detail of a bright bokeh shape using $x_f$ as computed in the paper even for foreground pixels.

(b) Discontinuities in bright bokeh shapes when using one ray for every foreground pixel.

Figure 5.13: Strong discontinuities when using at least one sample per pixel in the near field.

model, some rays of this distribution will hit the edge itself, while others will get past it and gather information about the geometry hidden behind the foreground. The shaded colors are stored in a **near-field** and a **far-field** texture according to whether the distance from the camera was less than or greater than the current focus distance set by the user. The ratio of **near-field** hit over the number of rays shot per pixel is also stored in a separate texture, and it will be used to merge the two fields in the next step. Both the color textures and the ratio are temporally accumulated. During this step, the average circle of confusion size per pixel is also stored in the alpha channel of the respective texture, as it will be needed for the spatial reconstruction step.

The **far-field** and the **near-field** are then merged into a single texture by linear interpolating between them, utilizing the accumulated hit ratio $h_i$. A hit ratio of 0 implies that all the rays intersect with the **far-field**, and for that pixel, there is no near-field data. Conversely, a hit ratio of 1 indicates that all the rays intersect with the foreground, leaving no information about what lies behind it. You can see the **far field** texture, the **near-field** texture, and the final merged result in Figure 5.15.

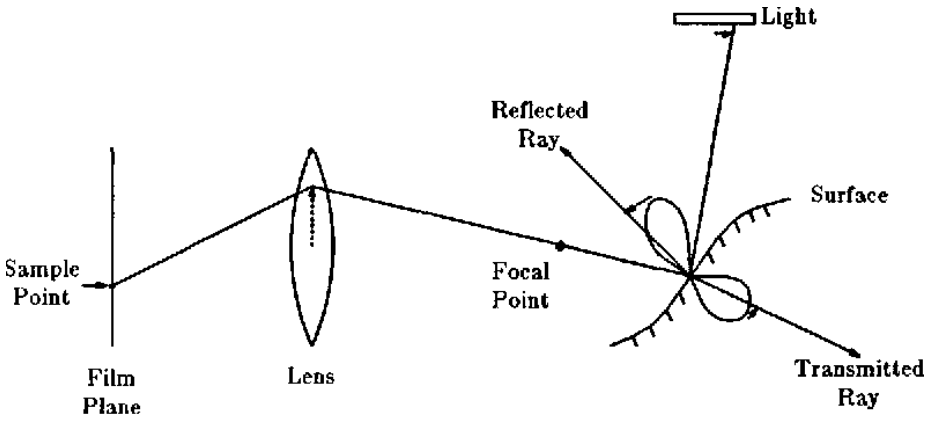



Figure 5.14: How rays are traced using Cook's technique, as seen in his paper.

### Computing color moments and variance map

The variance of a pixel $\sigma_i^2$ mentioned during the creation of the ray mask is used to target areas with high variance with a greater number of rays. This is computed using the same approach explained in Schied et al. [2017] based on the first and second raw moments of color luminance $\mu_{1_i}$ and $\mu_{2_i}$ for each pixel $p_i$:

$$\mu_{1_i} = luminance(p_i)$$
$$\mu_{2_i} = \mu_{1_i} \cdot \mu_{1_i}$$

The first and second raw moments of color luminance $\mu_{1_i}$ and $\mu_{2_i}$ are used to compute the variance of a pixel $\sigma_i^2$. The first moment $\mu_{1_i}$ is the average color luminance of the pixel $p_i$, while the second moment $\mu_{2_i}$ is the square of the first moment.
The variance of a pixel $\sigma_i^2$ is used to target areas with high variance with a greater number of rays. This is computed using the same approach explained in Schied et al. [2017] based on the first and second raw moments of color luminance $\mu_{1_i}$ and $\mu_{2_i}$ for each pixel $p_i$.
In summary, $\mu_{1_i}$ and $\mu_{2_i}$ are not the mean and variance, but they are related to them. The first moment $\mu_{1_i}$ is equivalent to the mean, while the second moment $\mu_{2_i}$ is related to the variance.

The color moments are temporally accumulated into $\mu_1'$ and $\mu_2'$ and the variance is computed:

$$\sigma_i^2 = \mu_{2_i}' - \mu_1^{'2}{}_i \tag{5.12}$$

The resulting variance map, shown in Figure 5.16 undergoes a Gaussian blur before being used for both the spatial reconstruction and the ray mask generation steps.

(a) The Near Field Texture

(b) The Far Field Texture

(c) The result of merging the far and near-field texture by linear interpolating between the two using the hit ratio $h$.

Figure 5.15

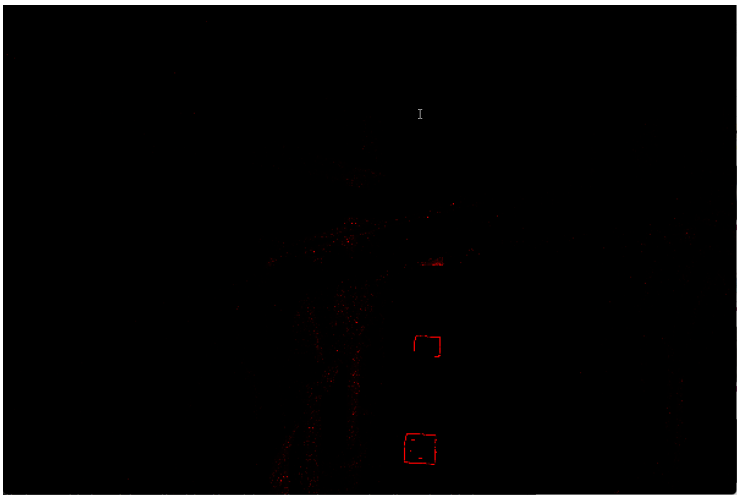
**Spatial Reconstruction step**

The last step before compositing is a Spatial Reconstruction one, in which a circular kernel scaled by the size of the average accumulated Circle of Confusion is used to collect color and hit ratio information from surrounding samples of the merged texture after it passes through a Median9 filter from Smith [1996]. Samples with a circle of confusion size larger than the distance from the center of the kernel are discarded, as their CoC does not overlap with the current pixel. The color and hit ratio of the center pixel of the kernel is linearly interpolated with the accumulated ones by clamped variance estimates

$$b = clamp(\sigma^2 \cdot 2000, 0, 0.9) \tag{5.13}$$

The scaling of $\sigma^2$ by 2000 will blur every pixel with a small variance value and the specific value has been chosen by the authors based on observations.

The relevant paragraph of Tan et al. [2022]'s paper is particularly ambiguous. The weights of the filter are never really mentioned, but since it is supposed



(a) $\sigma^2$

(b) $\sigma^2 * 100000$, as used in the ray-mask creation step.

Figure 5.16: The variance map computed on the merged texture.

to do a light blurring of the merged texture it was decided to use a Gaussian weighting.
The main issue with how this step is described is related to using the median9 filtered texture in the spatial reconstruction: since the merged texture is fully black outside the ray mask, the median9 filter reduces the edges of the ray-traced areas as it samples the black pixels as well. The spatial reconstruction step then does the same again, enhancing those dark edges and creating a very strong contrast where the traced pixels meet the non-traced ones, as seen in

Figure 5.17.

Figure 5.17: Dark Edges in the final images caused by the use of the filtered texture in the spatial reconstruction step.

After a few failed attempts at mitigating those edge artifacts by trying to detect them and reverting to the non-filtered texture, it was decided to use the merged, non-filtered, texture directly.
This leaves some dark edges because of the spatial reconstruction, but they are less noticeable.

### Compositing

Finally, the post-processed texture, the full-resolution unblurred image, and the ray-traced one are composited to obtain the final result.
Unfortunately, the compositing step is quite vague in the original paper, leaving a lot of space for interpretation, which made it very hard to reach the same result shown in Tan et al. [2022].
The texture containing the alpha values from the post-process step and the post-processed texture itself first undergo a Median9 filter as explained in Smith [1996]. The post-processed image and the ray-traced texture are then upscaled to full-resolution, while the alpha texture is left at half-resolution as suggested in Jimenez's presentation to allow for proper blurry foreground over the focused background.
To obtain the final post-processed color from Jimenez's approach, first, a delta of the depth value for pixel i $\delta_i$ is computed based on the current focus distance and the difference between the z-depth of the current pixel and the nearest depth in the pixel's tile from the dilated tile map.

The foreground factor is obtained by linearly interpolating between the largest circle of confusion from the dilated tile map and the size of the circle of confusion of the z-depth of the current pixel using the previously computed delta.
Finally, a combined factor is calculated by interpolating the size of the current Circle of Confusion with the foreground factor, utilizing the alpha obtained

during the post-process step. This combined factor is subsequently employed to interpolate between the samples from the original full-resolution rasterized image and the half-resolution result of the post-process step.
Then, the half-resolution ray-traced image is linearly interpolated with the final post-processed image using that combined factor.

```
float alpha = bnd.halfResAlpha.sampleLevel2D<float>(
    linearSampler, uv, 0);
float depthDelta = saturate(1.0 - focusDistance * (
    sampledDepth - nearestDepthInTile));
float fgFactor = lerp(largestCocInTile, depthCoc,
    depthDelta);
float combinedFactor = saturate(lerp(depthCoc,
    fgFactor, alpha));
float3 color = lerp(fullResColor, postProcessedColor,
     combinedFactor);
```

In the code shown above, sampledDepth is the depth of the current pixel, depthCoc is its diameter of the Circle of Confusion at that depth, nearestDepthInTile and largestCocInTile are, respectively, the red and green channel of the dilated tile map.

Utilizing the thin lens equation, the range of z-depth values for which the size of the circle of confusions is smaller than a certain size $c$ is determined, indicating that the pixel will be considered in focus:

$$z_{near} = \frac{a \cdot f \cdot d}{\left(a \cdot f + c\,(d - f) \cdot \frac{w_s}{w_i}\right)}$$
$$z_{far} = \frac{a \cdot f \cdot d}{\left(a \cdot f - c\,(d - f) \cdot \frac{w_s}{w_i}\right)}$$

In this case, $c$ has been chosen to have a value of $0.2mm$. This is the size of the Circle of Confusion at which a point looks *in focus* for most people watching an image at the *near distance for distinct vision* (Jacobson et al. [2001]) of $25cm$

For all the pixels with z-depth $z_i$ such as $z_{near} \leq z_i \leq z_{far}$ the full-resolution non-post-processed color is applied.
For all the pixels with a z-depth that is less than $z_{near}$, the ray-traced image is applied. This makes it so that all the detected edges of near-field objects will show what is partially hidden behind them.
For far-field geometry, the post-processed color is blended with the ray-traced one using the hit ratio $h_i$. The linear interpolation happens only for hit ratios higher than 0.3 to minimize blending artifacts and retain the Ray-Traced semi transparencies.

```
float ratio = smoothstep(0.0, 0.3, h);
float3 finalColor = lerp(rtColor, ppColor, ratio);
```

Just like for the post-processed image, this final result gets stabilized using **breda**'s TAA pass.
Figure 5.18 shows the final result that the newly implemented Depth Of Field module in **breda** sends back to the caller.

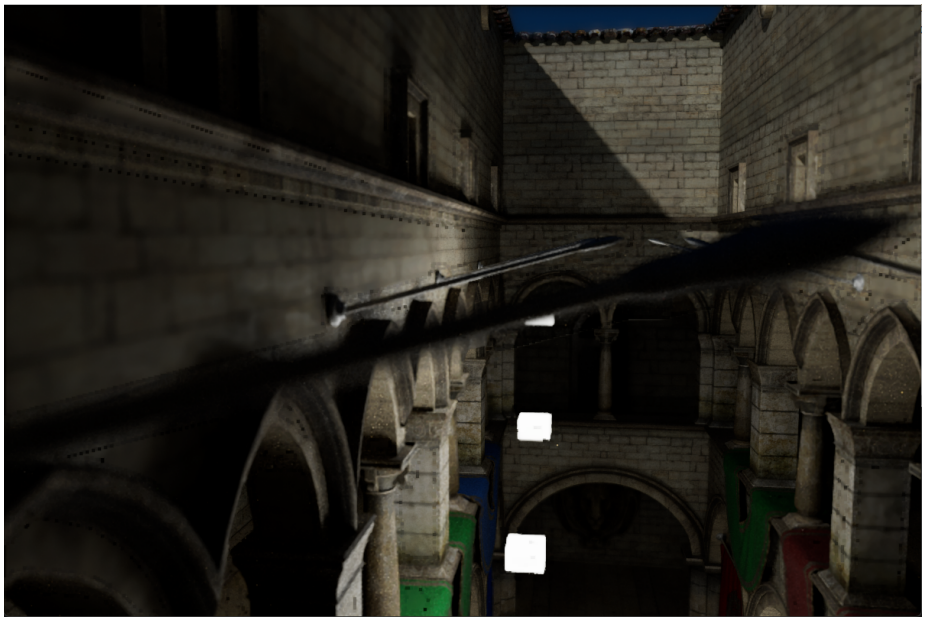


Figure 5.18: Final result sent back from the Depth Of Field module to the caller. Notice the darker pixel on the edges of the ray traced image.

### 5.1.3 Other ambiguities and limitations

Because this technique had to be implemented as a working algorithm in **Traverse Research** rendering framework, there have been a few problems in integrating it with the existing code.
The original paper from Tan et al. [2022] mentions that bokeh appearing on pixels with a high specular value coming from the GBuffer is handled differently in the compositing step by favoring the post-process over the ray trace color for samples with high specular values.
**breda** does not include this information in the output of its renderer as it implements PBR materials, so this distinction could not be made. There might be other assumptions that can be made to improve the merging heuristic.
Being able ao implement this heuristic would also improve the quality of the render when generating the ray mask with at least one sample per pixel, decreasing the discontinuity artifacts shown in Figure 5.13.

In the paper's introduction, the authors mention that the same split between near and far fields that happens in the ray tracing step also happens in the

post-process pass. As shown in Figure 5.1, that is not mentioned again in the paper: not in the pipeline description, not in the post-process section, and not in the compositing paragraph. It was decided to ignore that sentence and make the post-process step output just a single final texture.

In the closing chapters of Tan et al. [2022] it is mentioned that the final result suffers from some tiling artifact from the ray-tracing step. Tiling in the ray-tracing step is never mentioned during the rest of the paper. It is most likely an optimization technique similar to the tiling pass that happens in the post-process step, but it was decided to not try to implement the tiled version as it was unclear which and how the tiling should be applied.

Finally, the problem with the creation of the ray-mask discussed in the relevant section made it hard to achieve good framerates. While the idea of using the intensity of the edges together with the variance map is a strong one on paper, when implemented the way it was explained caused artifacts in the form of chunks of pixels being toggled on and off each frame and too many rays to be traced per pixels.

### 5.1.4 Results and Comparisons

All the measurements have been taken on a Razer Blade 2022 laptop with an Nvidia GeForce RTX 3060. Being a laptop we often had problems with throttling during testing, as the GPU would overheat very fast, hence the timings shown in this section might not be representative of the actual results.

Two different scenes have been used for testing: *Sponza* with a close-up of one of the rods to show the partial visibility around the edges (*Sponza FG*) and with a wider shot to highlight the background blurring (*Sponza BG*), and *Bistro* showing both background and foreground objects. The selected focus distance $d$, F-number $N$ and focal length $f$ for each scene have been selected to show off different parts of the Depth Of Field effect and can be seen in Table 5.1.

| | $f$ | $N$ | $d$ |
|---|---|---|---|
| *Sponza BG* | 70 | 3.4 | 3.7 |
| *Sponza FG* | 75 | 1.0 | 9.5 |
| *Bistro* | 50 | 3.5 | 6.0 |

Table 5.1: The parameters chosen for each scene

**breda**'s implementation is not a 1:1 reimplementation of the paper from Tan et al. [2022] because of lack of access to their code, the workarounds that had to be implemented to fill in missing information from the text and because of how **Traverse Research**'s framework is structured.

Despite this, the implementation of the post-processed Depth-Of-Field from Jimenez [2014] has been merged into the main branch of the codebase and its results are satisfactory, both in terms of performance, shown in Table 5.2, and on looks, shown in Figure 5.19.

(a)

(b)

Figure 5.19: Final result of Jimenez's approach implemented in **breda**

The ray-tracing part however still has improvements needed both in terms of performance and rendering quality for actual usage in commercial renders. The timings shown in Table 5.2 show how the algorithm is not ready to be used in production. The timings in Tan et al. [2022] are much better, with FPS that range from **90 to 120**, against ours, which range from **10 to 30**, but they do not state the resolution at which those timings were taken and we could not use the same scenes because of how **breda** handles materials: we were not able

| | Sponza (FG) | Sponza (BG) | Bistro |
|---|---|---|---|
| *Tiling (combined)* | 123.8$\mu s$ | 101.4$\mu s$ | 114.6$\mu s$ |
| *Dilate* | 45.9$\mu s$ | 68.7$\mu s$ | 47.5$\mu s$ |
| *Downsampling + presort* | 140.4$\mu s$ | 166.8$\mu s$ | 136.7$\mu s$ |
| *Main* | 381.6$\mu s$ | 393.2$\mu s$ | 341.5$\mu s$ |
| **Total Post-Processing** | **691.7**$\mu s$ | **730.1**$\mu s$ | **640.3**$\mu s$ |
| *Sobel pass* | 143.2$\mu s$ | 154.0$\mu s$ | 138.2$\mu s$ |
| *Dilate edges* | 71.0$\mu s$ | 69.4$\mu s$ | 61.2$\mu s$ |
| *Ray Tracing* | 18815.6$\mu s$ | 22997.6$\mu s$ | 16821.3$\mu s$ |
| *Merge* | 10.9$\mu s$ | 772.1$\mu s$ | 14.3$\mu s$ |
| *Spatial Reconstruct* | 147.8$\mu s$ | 154.1$\mu s$ | 144.2$\mu s$ |
| *Median9 (Combined)* | 58.0$\mu s$ | 60.5$\mu s$ | 60.5$\mu s$ |
| *Compositing* | 119.4$\mu s$ | 122.8$\mu s$ | 128.3$\mu s$ |
| **Total** | **20.057**ms | **25.061**ms | **18.558**ms |

Table 5.2: The mean timing of each shader pass for each scene with $m = 5$ and a resolution of 1200x800

to convert the *Modern Living Room* scene materials in a format that could be parsed by our renderer. Being able to test their implementation against ours would have also helped to understand if the time difference is not just a hardware issue.
Artifacts at the edges of the ray-traced textures are especially obvious when foreground objects are too close to the camera and a different compositing strategy is needed to better blend the post-processed texture with the ray-traced and the full-resolution one.
The ray-traced texture also does not completely match with the full resolution unblurred texture that is used as a starting point by the Depth of Field module. Despite all of this, Figure 5.20 shows a detail of *Sponza* where we can see the geometry that is hidden behind the out-of-focus rod in Jimenez's approach.
It is possible that with a smarter blending technique than the one presented in Tan et al. [2022], the final result can get closer to the reference of Cook's distributed ray tracing.
It might be worth trying splitting the post-processed texture into far and near fields like it is currently being done for the ray-traced texture, and compositing them separately together with the far and near ray-traced texture could result in a better final depth of field effect.

It is also quite important to notice that, since the ray mask is used in the composite step, the issues with its generation might have also impacted the final quality of the Depth Of Field. In Figure 5.21, Figure 5.22 and Figure 5.23, *Sponza BG*, *Sponza FG* and the *Bistro* scene can be seen in all three Depth of Field rendering technique mentioned above. Even with higher values of maximum rays per pixel $m$, our hybrid approach does not match Cook et al. [1984]'s implementation. Therefore, it was decided to keep a low value to have

(a) Our implementation (b) Jimenez Post-Processing

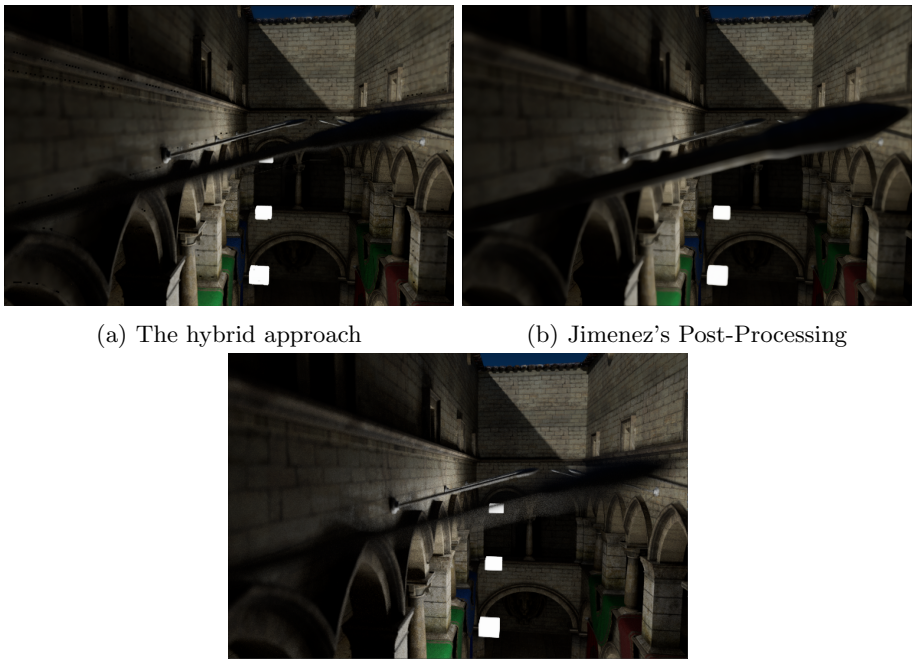
(c) Cook's distributed ray tracing

Figure 5.20: Detail of the same scene with the same camera settings rendered with 3 different Depth Of Field techniques.

(a) The hybrid approach (b) Jimenez's Post-Processing

(c) Cook's distributed ray tracing

Figure 5.21: Comparison of the Sponza BG scene with different Depth Of Field rendering techniques.



(a) The hybrid approach

(b) Jimenez's Post-Processing

(c) Cook's distributed ray tracing

Figure 5.22: Comparison of the Sponza FG scene with different Depth Of Field rendering techniques.

better framerates.

(a) The hybrid approach

(b) Jimenez's Post-Processing

(c) Cook's distributed ray tracing

Figure 5.23: Comparison of the Bistro scene with different Depth Of Field rendering techniques.

# Chapter 6

# Discussion and Future Work

### 6.0.1 Future work

Now that an implementation of the paper for **breda** exists, there are a few possible optimizations that could be used to extend the original code and improve performance and image quality.

#### Jimenez Post-Process Optimizations

Jimenez, in his presentation, suggests that the tiles generated from the dilation pass could be processed differently according to the difference between the minimum and the maximum depth in each tile. If the difference between the minimum and maximum depth is equal to zero, then the algorithm converges to returning the prefilter color, while if it is between 0 and a certain threshold, a simpler average of the sampled colors will be returned.
Another possible technique suggested by Jimenez is the possibility of using different numbers of rings according to the size of the tile's largest CoC.
Both of those optimizations are a trade-off between performance and accuracy which need to be tested and tweaked to produce a satisfactory result, but if well tuned might increase the speed of the post-process step as several tiles could be processed much faster than the others.

#### Matching the Ray Tracing pass with the rasterized output

As mentioned at the beginning of the previous section, the implementation has been done to match the output of the **wavefront-path-tracer** library included in **breda**. This made it possible to reuse the ray tracing code from the reference path tracer to match the result of the ray-tracer pass from the Depth Of Field pipeline.
Traverse Research's real product is the hybrid renderer, which has a much higher

frame rate, and it is being demo-ed to clients and conferences, therefore the code of the newly added Depth Of Field crate has been written in such a way that only the ray tracing shader needs to be changed to match the output of the hybrid renderer.

#### Replacing the ray-mask generation technique

This is one of the biggest issues with the original paper right now. While this implementation proved that the approach works and can provide good results, the ambiguities in the ray-mask generation process made it very hard to replicate the results from the original implementation.
A different edge detection technique, with a better heuristic for the number of rays to be traced per pixel will greatly increase the quality and the performance of this algorithm.

#### Generic optimizations

While the code was not written with a complete disregard for speed, fast shader passes were not the main scope of the implementation. There are certainly some optimizations that can be implemented, like splitting the Sobel pass in a vertical and horizontal one like it has been done for the tiling pass of the post-process effect, or computing MIP maps of various textures beforehand (like for the depth) to sample directly from a downsampled version.
The original paper achieved relatively real-time framerates on low ray count, claiming that their code was also not optimized because the framework they used was very limiting. With attention to the code, we believe that real-time framerates can be achieved even on higher ray count.

### 6.0.2 Conclusion

The goal of this work was to find a way to reverse Asberg's innovative approach to produce a distribution of primary rays for *Traverse Research*'s real-time renderer: **breda**.

The original research question was the following:

**RQ1: Can we reverse Asberg's innovative approach and expand it to produce primary rays for breda, Traverse Research's[1] real-time rendering framework?**

A positive answer to RQ1 would have provided **breda** with a real-time, photo-realistic, and customizable depth of field effect with little overhead on the total rendering time. In chapter 4 we delved into the intricacies of optical simulations and the challenges associated with reversing Lynn Asberg's approach. As discussed, the unique aberration effects experienced by individual rays and the

[1] `https://traverseresearch.nl`

complex interactions within optical systems make the reversal of this approach much harder than originally planned. The Helmholtz reciprocity principle could not be applied because of the fundamental limitations imposed by the individual ray behavior within the optical system. Each ray, as it passes through various optical elements, encounters distinctive aberrations, altering its path in a highly individualized manner. Attempting to reverse this complex, non-uniform behavior for the entire fan of rays proved to be a daunting task, far more intricate than originally anticipated.

While not completely ruling out the possibility of an approach based on Asberg's work, time constraints led us to focus on other research directions more closely aligned with the thesis's goals.

After the negative answer to RQ1, the aim of the thesis moved to RQ2 to explore different innovative Depth Of Field techniques by implementing the techniques introduced in the paper by Tan et al. [2022] within **breda** ray tracing framework.

**RQ2: Can we implement Tan et al. [2022]'s paper in breda so that a post-process Depth Of Field can be rendered in real-time, with low latency while also improving on existing techniques for the partial visibility issue?**

Our implementation of Tan et al. [2022], while promising, encountered several limitations partially caused by the lack of a reference implementation. The unavailability of the paper's source code restricted our ability to fine-tune and optimize the implementation fully, and the lack of communication on the authors' part left a lot of guesswork to do, like on the ray-mask generation step, the Spatial Reconstruction step, and in the compositing pass.
These limitations, along with other complications arising from how Traverse Research's rendering framework is structured, revealed that reaching a working solution that could be used on actual real-time rendering engines might need some extra work and research, especially regarding the performances and the output of the spatial reconstruction step and the generated ray-mask.

However, our implementation still demonstrates the potential of such a technique regarding the issue of having realistic partially visible occlusions in post-process effects with a small overhead on the framerate.
It is worth continuing to investigate such a technique, as it can also be used to solve the same problem of missing geometry on the edges of applying motion blur as a post-process effect. Access to the original source code, if possible, will help to better understand how certain steps, like the compositing or the spatial reconstruction ones, were implemented and tweaked to obtain the paper's results.
With more time available, it would be certainly beneficial to keep studying this approach, finding better ways to perform edge detection to create a ray mask, finding ways to reduce the number of rays per pixel, and efficiently blur the

ray-traced texture to match the post-process color.

But is also equally interesting to continue the research into the possibility of using complex topics from the field of optics to create a new approach to rendering Depth Of Field by applying the Seidel Aberration theory that Asberg researched for her thesis. As we said in chapter 4, we are not excluding that such a technique exists, and it would be quite impressive if a way to apply the aberration theory to generate primary rays without tracing the entire lens could be found.

In conclusion, this research shows the irreversibility of Asberg's approach using conventional means, the limitations faced in implementing Tan et al.'s paper, and the potential for future advancements in depth-of-field simulations.

The incredibly smart technique introduced by Asberg in Asberg [2020] and the innovative approach of Tan et al. in Tan et al. [2022] show that there is still a lot of room for improving the realism of Depth of Field simulations in computer-generated images, both for online and offline rendering.
As new techniques will be developed and new hardware will allow for more ray-tracing that will not impact the framerates, we will soon get even closer to having realistic optic simulation on real-time applications.

# Bibliography


Guglielmo Abadie. A life of a bokeh. https://epicgames.ent.box.com/s/s86j70iamxvsuu6j35pilypficznec04, 2018. [Online; accessed 17-February-2023].

Arthur Appel. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, AFIPS '68 (Spring), page 37–45, New York, NY, USA, 1968. Association for Computing Machinery. ISBN 9781450378970. doi: 10.1145/1468075.1468082. URL https://doi.org/10.1145/1468075.1468082.

Lynn Asberg. Pametrically replicating bokeh using seidel aberration. https://lynnasberg.nl/thesis.pdf, 2020.

Brian Barsky and Todd Kosloff. Algorithms for rendering depth of field effects in computer graphics. 01 2008.

M. Born, E. Wolf, and A.B. Bhatia. *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*. Cambridge University Press, 2000. ISBN 9780521784498. URL https://books.google.nl/books?id=oV8OAAAAIAAJ.

Robert L Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 137–145, 1984.

Glenn Evans and Michael D. McCool. Stratified wavelength clusters for efficient spectral monte carlo rendering. In *Graphics Interface*, 1999.

Yoshiharu Gotanda, Masaki Kawase, and Masanori Kakimoto. Real-time rendering of physically based optical effects in theory and practice. In *ACM SIGGRAPH 2015 Courses*, SIGGRAPH '15, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336345. doi: 10.1145/2776880.2792715. URL https://doi.org/10.1145/2776880.2792715.

Johannes Hanika and Carsten Dachsbacher. Efficient monte carlo rendering with realistic lenses. *Computer Graphics Forum*, 33(2):323–332, 2014. doi: https://doi.org/10.1111/cgf.12301. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12301.

E. Hecht. *Optics*. Pearson Education, Incorporated, 2017. ISBN 9780133977226. URL https://books.google.nl/books?id=ZarLoQEACAAJ.

Matthias B. Hullin, Johannes Hanika, and Wolfgang Heidrich. Polynomial optics: A construction kit for efficient ray-tracing of lens systems. *Comput. Graph. Forum*, 31(4):1375–1383, jun 2012. ISSN 0167-7055. doi: 10.1111/j.1467-8659.2012.03132.x. URL https://doi.org/10.1111/j.1467-8659.2012.03132.x.

Andrey Ignatov, Jagruti Patel, and Radu Timofte. Rendering natural camera bokeh effect with deep learning. pages 1676–1686, 06 2020. doi: 10.1109/CVPRW50498.2020.00217.

Ralph E. Jacobson, Norman Axford, Sidney Ray, and Geoffrey G. Attridge. *Manual of Photography: Photographic and Digital Imaging*. Butterworth-Heinemann, USA, 9th edition, 2001. ISBN 0240515749.

Yuna Jeong, Seung Youp Baek, Yechan Seok, Gi Beom Lee, and Sungkil Lee. Real-time dynamic bokeh rendering with efficient look-up table sampling. *IEEE Transactions on Visualization and Computer Graphics*, 28(2):1373–1384, 2022. doi: 10.1109/TVCG.2020.3014474.

Jorge Jimenez. Next generation post processing in call of duty: Advanced warfare. https://www.iryoku.com/next-generation-post-processing-in-call-of-duty-advanced-warfare, 2014. [Online; accessed 23-july-2023].

Hyuntae Joo, Soonhyeon Kwon, Sangmin Lee, Elmar Eisemann, and Sungkil Lee. Efficient ray tracing through aspheric lenses and imperfect bokeh synthesis. *Computer Graphics Forum*, 35, 2016.

James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20 (4):143–150, aug 1986. ISSN 0097-8930. doi: 10.1145/15886.15902. URL https://doi.org/10.1145/15886.15902.

Brian Karis. Real shading in unreal engine 4. 2013. URL https://api.semanticscholar.org/CorpusID:14922512.

Michael Kass, Aaron Lefohn, and John D Owens. Interactive depth of field using simulated diffusion on a gpu. 2006.

Craig Kolb, Don Mitchell, and Pat Hanrahan. A realistic camera model for computer graphics. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, page 317–324, New York, NY, USA, 1995. Association for Computing Machinery. ISBN 0897917014. doi: 10.1145/218380.218463. URL https://doi.org/10.1145/218380.218463.

Tim Mcgraw. Fast bokeh effects using low-rank linear filters. *The Visual Computer*, 31, 05 2014. doi: 10.1007/s00371-014-0986-6.

P. Moon and D. E. Spencer. *The photic field*. 1981.

Juewen Peng, Zhiguo Cao, Xianrui Luo, Hao Lu, Ke Xian, and Jianming Zhang. Bokehme: When neural rendering meets classical rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Michael Potmesil and Indranil Chakravarty. A lens and aperture camera model for synthetic image generation. In *International Conference on Computer Graphics and Interactive Techniques*, 1981.

Ming Qian, Congyu Qiao, Jiamin Lin, Zhenyu Guo, Chenghua Li, Cong Leng, and Jian Cheng. BGGAN: Bokeh-glass generative adversarial network for rendering realistic bokeh. In *Computer Vision – ECCV 2020 Workshops*, pages 229–244. Springer International Publishing, 2020. doi: 10.1007/978-3-030-67070-2_14. URL https://doi.org/10.1007%2F978-3-030-67070-2_14.

Gilberto Rosado. Motion blur as a post-processing effect. In *GPU Gems 3*, GPU Gems. Nvidia, 2008.

Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R. Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. Spatiotemporal variance-guided filtering: Real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics*, HPG '17, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450351010. doi: 10.1145/3105762.3105770. URL https://doi.org/10.1145/3105762.3105770.

P.L. Seidel. *Ueber die Theorie der Fehler, mit welchen die durch optische Instrumente gesehenen Bilder behaftet sind, und über die mathematischen Bedingungen ihrer Aufhebung*. Abhandlungen der Naturwissenschaftlich-Technischen Commission bei der Königl. Bayerischen Akademie der Wissenschaften in München. Cotta, 1857. URL https://books.google.nl/books?id=f8rVPgAACAAJ.

Jeremy I Smith. Implementing median filters in xc4000e fpgas. 1996.

Yu Wei Tan, Nicholas Chua, Nathan Biette, and Anand Bhojan. A hybrid system for real-time rendering of depth of field effect in games. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 2022. doi: 10.5220/0010839800003124. URL https://doi.org/10.5220%2F0010839800003124.

H. von Helmholtz. *Handbuch der physiologischen Optik*. Number v. 1 in Added t.-p.: Allgemeine encyklopädie der physik ... hrsg. von G. Karsten. IX bd. Voss, 1867. URL https://books.google.nl/books?id=E3EZAAAAYAAJ.

Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, jun 1980. ISSN 0001-0782. doi: 10.1145/358876.358882. URL `https://doi.org/10.1145/358876.358882`.

Jiaze Wu, Changwen Zheng, Xiaohui Hu, Yang Wang, and Liqiang Zhang. Realistic rendering of bokeh effect based on optical aberrations. *The Visual Computer*, 26:555–563, 06 2010. doi: 10.1007/s00371-010-0459-5.

Jiaze Wu, Changwen Zheng, Xiaohui Hu, and Fanjiang Xu. Rendering realistic spectral bokeh due to lens stops and aberrations. *The Visual Computer*, 29, 01 2012. doi: 10.1007/s00371-012-0673-4.

Tianshu Zhou, Jim X. Chen, and Mark Pullen. Accurate Depth of Field Simulation in Real Time. *Computer Graphics Forum*, 2007. ISSN 1467-8659. doi: 10.1111/j.1467-8659.2007.00935.x.