Utrecht University

Faculty of Science

Dept. of Information and Computing Sciences

Parametrically Replicating Bokeh Using Seidel Aberrations

Author

Lynn Asberg BSc

Supervised by:

dr. ing. J. Bikker

A thesis submitted for the degree of "Master of Science"

October 1, 2020

Abstract

In this thesis we investigate a method of simulating realistic spectral depth of field in screen space. We propose the direct application of Seidel aberrations (*Von Seidel*, 1857) to incoming light rays, which is a new application of a well-known concept in optics. Our aim is to use this limited set of aberrations to offer artists intuitive control over complex lens effects without necessarily requiring lens designs. Existing implementations do require lens designs, but they are often hard or impossible to obtain. Some state-of-the-art implementations are incapable of rendering these effects altogether and resort to uniform disk- or sprite-shaped bokeh. We apply Gaussian optics theory extended with Seidel aberrations to scatter light rays to the screen in order to simulate the depth of field generated by different lens designs. In most test cases, our method approaches spectral ray tracing considerably better, both visually and in terms of quantitative image comparison metrics, than when using only Gaussian optics. In addition, we can reduce the number of non-intuitively wavelength-dependent lens parameters from seven to three, while achieving similar results. Our implementation runs in constant time for each ray, while ray tracing would run in linear time, scaling with the number of lens surfaces. When realistic depth of field is desired, our implementation can be used as a more time-efficient alternative to primary ray generation in most ray tracing frameworks.

Contents

Chapter 1

Introduction

Depth of field (DOF) is a term originating from photography, and refers to the range of in-focus distances in an image. Any object outside of this range will be visibly blurred. Depth of field is an effect intrinsic to any lens, and is frequently used as both a storytelling and artistic tool in photography, cinematography and games. It is commonly used to direct the viewer's attention by choosing which parts of the scene are in focus and which ones are blurred. An example of depth of field in a photograph is shown in Figure 1.1.

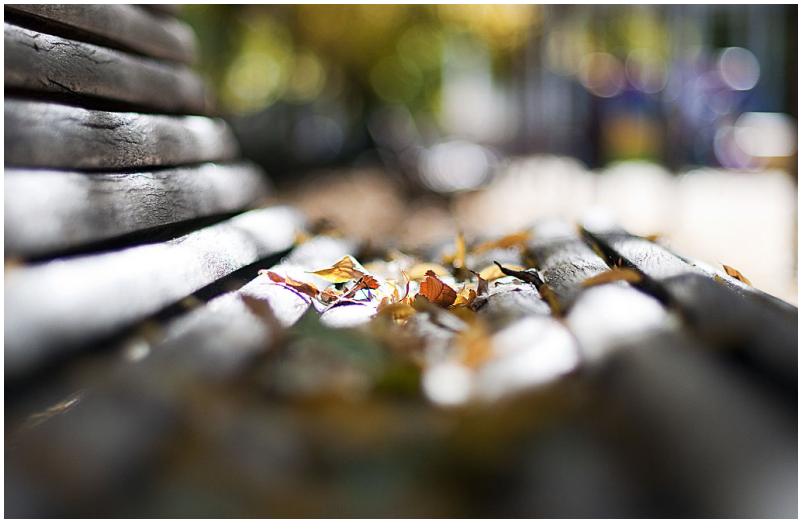


Figure 1.1: A photograph, demonstrating depth of field (image from `https://www.flickr.com/photos/rromer/5192805854`)

Because of its applicability as a storytelling and artistic tool, depth of field simulation is commonplace in computer graphics. Simulating depth of field can be done accurately by applying distribution ray tracing to lens designs (*Kolb et al.*, 1995), where many light rays are simulated as they pass through some lens system, ending up on a certain pixel on an imaging sensor. Often though, scenes are rendered using a *pinhole* camera model, which does not produce any depth of field as it leaves the whole image 'in focus'. This saves much computation time, and is generally easier to implement. Depth of field can be added to these pinhole renders in a *post-process*, which usually requires depth information for each pixel, as well as a focus distance – the distance to

the lens of objects that appear sharp – and some lens parameters. The time saved by applying a post-process instead of integrating depth of field in a ray tracer is often significant, because a large number of samples per pixel is required to render noise-free depth of field, and because of the large cost of each ray tracing sample. In addition to saving computation time, using a post-process also allows for artistic control over the depth of field after rendering has completed. This is particularly useful for computer generated imagery for films, where re-rending can take minutes per frame.

There are several challenges that need to be overcome in order to render realistic depth of field in a post-process, such as accurate separation between blurred and sharp depth layers and filling in information that is missing in the original render but would become visible when applying depth of field. An example of imperfect depth of field generated using a post-process is shown in Figure 1.2. These problems have been addressed quite successfully in several different approaches. A state-of-the-art implementation by *Abadie* (2018) is utilized in the Unreal game engine. This implementation can generate very convincing depth of field in a post-process that takes only a few milliseconds for full HD images (1920×1080 pixels) on modern hardware. But in order to achieve such performance, lens imperfections are modelled only very limitedly.

Figure 1.2: Imperfect depth of field, generated using the Unity game engine. Note that some areas are both in focus and blurred at the same time (e.g., the spark in front of the statue's head). (image from `https://docs.unity3d.com/2018.1/Documentation/Manual/PostProcessing-DepthOfField.html`)

An important aspect of depth of field is the aesthetic quality of the out-of-focus areas, which is called *bokeh*. Bokeh can vary significantly between lenses, and, especially in cinematography, lenses are often carefully sought out specifically for the unique bokeh that they produce. The imperfections – the deviations from perfectly sharp, uniformly disk-shaped bokeh – of the bokeh generated by these lenses aid in establishing the atmospheres that the story calls for. In *visual effects*, often computer generated imagery (CGI) is intermixed with footage shot with lenses that generate a particular bokeh. In order for these to blend seamlessly, it is very important to *match* the bokeh of the CGI to that of the other footage. This is often done using post-processes like ZDefocus, which is included in the popular compositing software Nuke (*Robinson*, 2013). When using these kind of post-processes, the artist usually has much control over the size of the bokeh shapes, and can control the shape and color by means of a single sprite input. This is no longer sufficient when the lens imperfections are more pronounced, as the bokeh shape can depend on the distance from the center of the image as well as its position relative to the focus point. Full ray tracing methods like that introduced by *Kolb et al.* (1995) are able to accurately simulate all bokeh imperfections, but only when fully specified lens designs are available, which is often not the case.

In order to find a solution that can be employed without the direct need of lens designs, we turn to spectral Gaussian optics and aberration theory, which allow us to describe a lens design using a small set of parameters that are a function of light wavelength. We can then calculate the pixel position at which a light ray with a certain wavelength ends up after passing through the lens without actually having to trace the ray through the whole lens system. This gives us a simplified, physically-based method for simulating depth of field. The imperfections of a lens can be expressed as *aberrations*. We simulate the five most simple aberrations, which are known as the *Seidel aberrations* (*Von Seidel*, 1857). These aberrations are well known in optics, but have never been explicitly applied and analyzed in depth of field simulation. *Hullin et al.* (2012) have previously applied aberration theory to depth of field, but only implicitly by obtaining Taylor expansions of analytically solved equations for ray parameters after passing through a lens system. We propose the explicit use of aberration theory in the context of artist-driven depth of field simulation and extensively analyze the applicability of Seidel aberrations in this context.

We use a distribution simulation method to generate depth of field by calculating the pixel coordinates for many rays per pixel, applying these Seidel aberrations. We use full sequential ray tracing of the lens, as introduced by *Kolb et al.* (1995), as our reference method. We compare the resulting images both quantitatively using RMSE and MS-SSIM and qualitatively, in order to answer the question: can aberration theory with only Seidel aberrations be used by an artist to accurately simulate depth of field and how does it compare to other methods?

As other methods we use monochromatic (single wavelength) Gaussian optics without aberrations, which serves as a basic 'disk-shaped bokeh' implementation, and a *pencil map* implementation (*Gotanda et al.*, 2015). We conclude that of the methods tested, our Seidel aberration-based method produces the best results when compared to the ray traced reference images. This is not universally true, however, as there are a few cases where a different implementation produced quantitatively more accurate results, depending mostly on the lens designs used. Our implementation takes $O(1)$ time to generate a ray, while ray tracing the lens system takes $O(n)$ time, where n is the number of lens surfaces. So our implementation can be a good alternative to ray tracing when time is of the essence or when more artistic control is desired. It can be integrated into most ray tracing frameworks as a way to generate primary rays. In order to use our method, we need to keep track of a set of parameters that depend on the wavelength and distance to the object in a non-intuitive way. We show that we can remove the distance dependence and reduce the number of lens parameters without sacrificing much of the image quality in most cases.

Chapter 2

Preliminaries

In this chapter we review fundamental concepts from the fields of photography and cinematography in general and depth of field in particular that form the basis for our research.

2.1 Cameras, lenses and photographic terminology

A photo, video or film camera creates an image by focusing light onto a light sensitive surface, which is either photographic film in analog cameras or a sensor in digital cameras. For brevity, we will refer to this surface as the *imaging sensor*. To focus the light, early cameras used a *pinhole*, which is just a tiny hole in an otherwise opaque surface in front of the sensor. A pinhole creates images that are fairly sharp all over, making focusing irrelevant and depth of field non-existent. An example of a pinhole camera photo can be seen in Figure 2.1.

Figure 2.1: A photo taken with a pinhole camera. Note that the sharpness is the same for both the church in the background and the bricks in the foreground – there is no depth of field. Image from `https://theimageflow.com/ngg_tag/pinhole-photography/`.

A downside of pinhole photography is the low amount of light that enters the camera. It is often necessary to wait for several minutes before the film or sensor has been exposed to enough light. This is due to the small size that the pinhole needs to be to create a sharp image. If we let more light in by making the pinhole larger, the image will become blurrier as light from more angles can end up on the same spot on the sensor. But if the hole is made too small, diffraction effects make the image blurrier as well. These effects are illustrated in Figure 2.2.

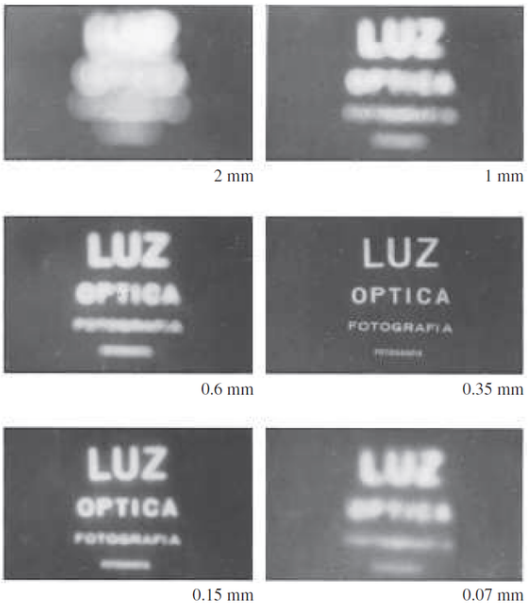


Figure 2.2: The effect of the pinhole size on the sharpness of a pinhole photograph. Larger holes blur the image by letting light from too many angles in, while smaller holes blur the image by diffraction effects (*Hecht*, 2017).

A lens can be used instead of a pinhole in order to let more light hit the sensor, without the blurring that occurs when using large pinholes. Each lens has a *focus distance*, which is the distance at which it focusses light coming from infinitely far away. Light that comes from a different point will be focussed at a different distance. The plane that goes through the focus distance and is perpendicular to the axis of the lens is called the *focal plane*. If we put the imaging sensor at this plane, only light originating from infinitely far away will be 'in focus'. Light from other distances will appear blurred.

Photographic lenses contain multiple glass elements. These elements are combined in such a way that they cancel out each other's imperfections as much as possible, in order to create a uniformly sharp image. We will refer to such a combination of glass elements as a *lens system*. An example of a cross section of a lens system is shown in Figure 2.3.

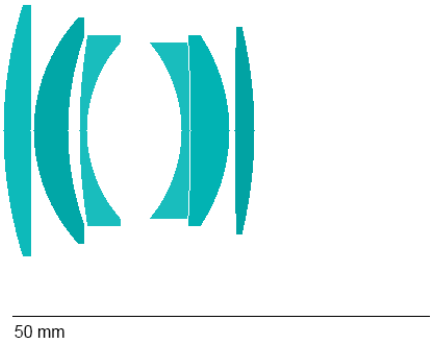


Figure 2.3: A schematic cross section of a lens system, showing the glass elements inside.

The lenses in a lens system can usually be described as spherical surfaces, the centers of which lie on a common axis called the *optical axis*. There are exceptions: there exist lenses that are not spherical (aspherical lenses) and sometimes lenses are not centered perfectly on the optical axis (e.g., in tilt-shift lenses or lenses with optical stabilization). We will however focus only on spherical, axis-aligned lens systems. Furthermore, a photographic lens usually has one or more movable sections in order to change the focus distance. As a byproduct, this can cause *focus breathing*, i.e. the field of view of a lens changing based on its focus distance. In all our simulations, however, we will simulate moving the imaging sensor and keeping the lens still in order to focus light from the desired distance.

Other than lenses, a lens system can also contain any number of *stops*, of which the *aperture stop* is one. A stop is a usually circular hole that stops some light from travelling further down the lens. The aperture stop is generally found in the middle of a lens system, and is the main light limiting stop in the system. The size of the aperture directly impacts both the blur radius of out-of-focus elements – also called the *circle of confusion* – and the amount of light that enters the lens. In order to control the aperture size, an aperture is usually made up of several *blades*, with the exact number and shape of blades varying per lens. Two examples are shown in Figure 2.4. The image of the aperture stop is called the *entrance pupil* when seen from the direction of incoming light, and the *exit pupil* when seen from the direction of outgoing light.

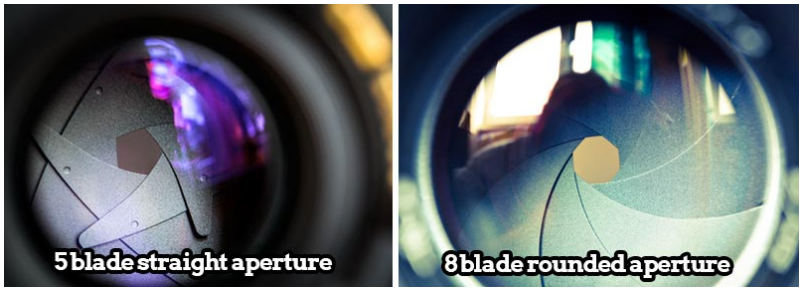


Figure 2.4: Two different apertures, with five straight blades (left) and eight rounded blades (right). Image from `https://improvephotography.com/29529/aperture-blades-many-best/`

The exact shape of the aperture is visible in the out-of-focus areas of an image, as will be discussed in the next section.

2.2 Bokeh

An important concept when talking about depth of field is *bokeh*. The term is derived from the Japanese word for *blur* or *blur quality* and denotes the aesthetic quality of the blur produced in the out-of-focus parts of an image produced by a lens (*Präkel*, 2010). The exact shape and color of bokeh can differ wildly from lens to lens, depending on the exact shape and materials of the lens elements used. It can also vary based on the position in the image. Some examples of bokeh are shown in Figure 2.5. The shape of the aperture stop generally determines the shape of the bokeh, as it is the main light limiting stop in a lens system. So if, for example, the aperture has a hexagonal shape, the bokeh will generally also resemble hexagons.

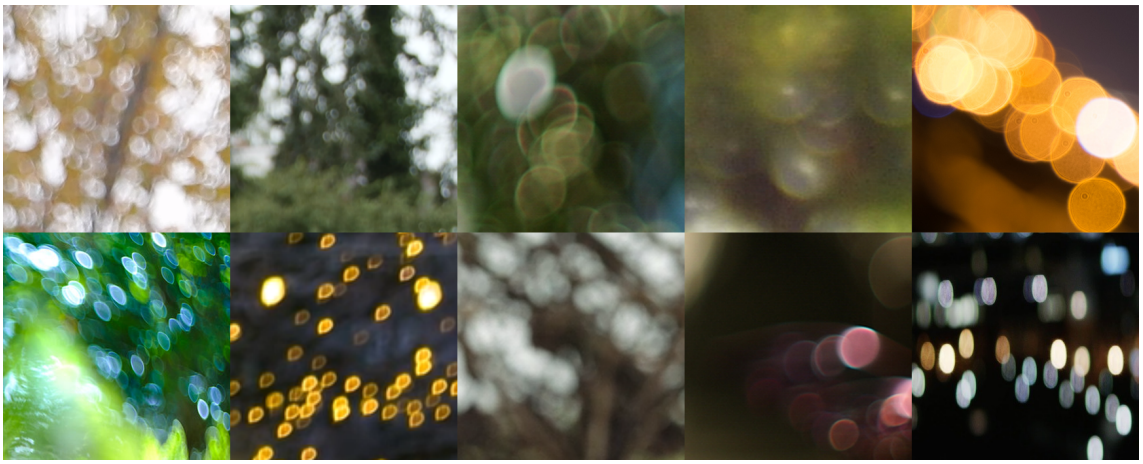
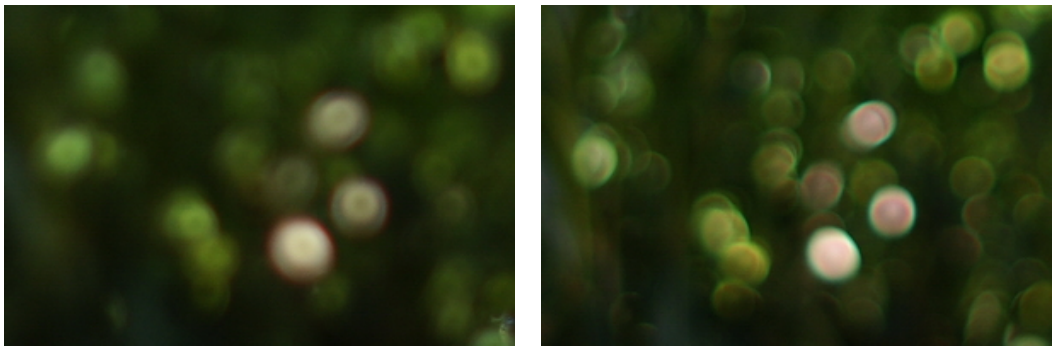


Figure 2.5: Several examples of bokeh, showing the wide variety of bokeh shapes and looks that occur in modern lenses.

Bokeh can also vary based on whether it is in the foreground (in front of the focal plane) or background (behind the focal plane), as is illustrated in Figure 2.6.



(a) Front bokeh (b) Back bokeh

Figure 2.6: Two photographs of bokeh generated by the same lens. The focus is behind (left) and in front of (right) the light source. Images from `https://photojottings.com/bokeh-sampler/`

Chapter 3

Previous work

We are now going to discuss the simulation of depth of field. As it has many application domains, this is topic that has been researched in various fields, each of which has its own requirements. In games, real-time performance is key, and realism has a lower priority. In CGI for films, realism is often the most important and longer calculation times are justified.

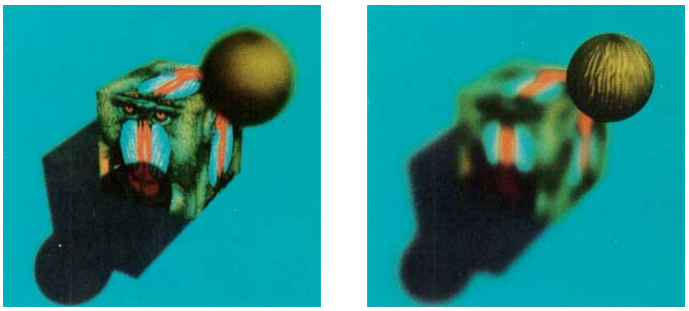
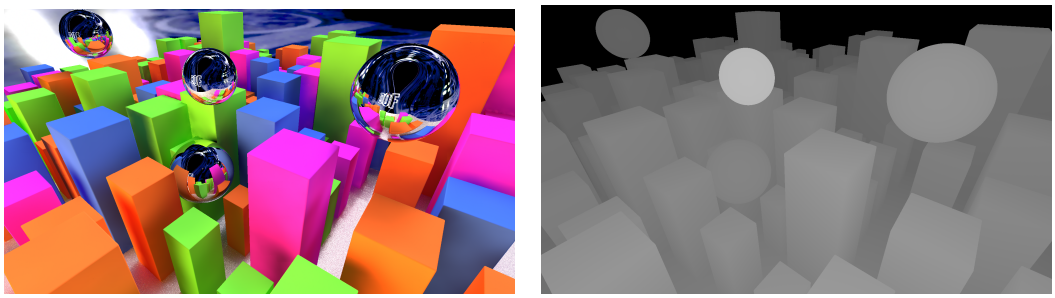


Figure 3.1: Early depth of field simulation by *Potmesil and Chakravarty* (1981), showing two renders of the same scene with different focus distances.

Depth of field simulations can be divided into two main categories: world-space and image-space methods (*Barsky and Kosloff*, 2008). World-space methods simulate a lens system inside a 3D world in one way or another, whereas image-space methods are applied as a post-process to a previously rendered image. The two have different application domains, with image-space methods mostly being used in real-time applications as they often require only a few milliseconds to run, as compared to world-space methods that can take anywhere from a tenth of a second to several minutes to run. But as this chapter will show, there is much diversity in both domains, with realistic depth of field being possible both in world-space and image-space methods. From here on out, image-space methods will be referred to as post-processes, defined as a method that has a pinhole camera render as input, usually combined with depth information, as illustrated in Figure 3.2.

The first depth of field post-process was introduced by *Potmesil and Chakravarty* (1981), who pioneered depth of field research in the context of CGI. They introduced a method for calculating the diameter of the *circle of confusion* (the disk over which the light of a point light source is spread on the screen), even taking into account effects like diffraction, and proposed a post-process for a raster image (Figure 3.1).



(a) (b)

Figure 3.2: The usual information provided to a depth of field post-process: a pinhole camera render (left) and a depth map (right).

3.1 Uniform depth of field simulations

In this section we give an overview of depth of field simulation methods that produce uniform depth of field and produce simple bokeh (at most aperture-shaped).

Accumulation buffer

Haeberli and Akeley (1990) introduced the accumulation buffer, where the scene is rendered separately from multiple viewpoints, the results of which are averaged to form the final image. An advantage of this method is that it can be applied to any rendering algorithm, and is therefore applicable to both rasterization and ray tracing. The accumulation buffer is rarely used nowadays, mainly because of the large amount of time it takes to converge to a smooth image. If a low number of samples is used, this method produces banding artifacts.

Gathering

A widely used depth of field post-process is gathering, which is a method to determining which circle of confusion overlap which pixels. When using gathering, every pixel checks which of its surrounding pixels – usually in some kernel – contribute to it. If the circle of confusion of a surrounding pixel overlaps the sampling pixel, the color is added to a sum, which is later normalized by dividing it by the number of contributing samples. Gathering is highly parallelizable, making it ideal for GPU implementation, which is why it is often used in real time applications (*Riguer et al.*, 2004).

Separable blur

One method of accomplishing an out-of-focus effect is to apply a blurring filter, such as a Gaussian filter. Image blurring can be done efficiently by using a separable filter. A separable filter can be written as the product of two one-dimensional filters, which allows for a separate horizontal and vertical convolution pass, reducing the algorithmic complexity from $O(n^2)$ time to $O(n)$ time. Because of this separable nature, not every blur shape can be accomplished using a separable filter. But it turns out that it is possible to approximate convex (*Moersch and Hamilton*, 2014) and even arbitrary (*McGraw*, 2015) bokeh shapes quite well using multiple passes. *McGraw* (2015) uses low-rank linear filtering to approximate an arbitrary kernel as a sum of separable kernels. The higher the number of kernels (the filter rank), the closer the resemblance to the original kernel. A comparison of low-rank linear filtering to gathering is shown in Figure 3.3.

(a) (b)

Figure 3.3: An image blurred with a hexagonal bokeh shape using sparse gathering (left) and low-rank linear filtering (right) (*McGraw*, 2015).

With separable blur approaches it is not trivial to generate variable-sized bokeh. Conventionally, a uniform kernel is applied over the entire image. *Riguer et al.* (2004) and *Garcia* (2017) achieve variable sharpness by linearly interpolating between the blurred and sharp version of the input image, as illustrated in Figure 3.4. *Moersch and Hamilton* (2014) introduce a method to generate variable-sized bokeh using a modified separable approach where the first (horizontal) blur pass exports not a 2D but a 3D image, supplying the second (vertical) pass with the extra information needed. Their approach does however require a convex bokeh shape with a constant intensity.

Figure 3.4: An example of interpolation between the blurred and sharp version of an image using a single-size separable filter. Note the clearly visible transition between the blurred and sharp versions on the ground behind the tree. (*Garcia*, 2017)

3.2 Approximating imperfections

All the implementations discussed so far only render disk-shaped bokeh or at most emulate aperture shapes. This is because these implementations depend on simplified lens simulations, at most approaching Gaussian optics and the thick lens approximation – these will be discussed in detail in Chapter 4. Here we will discuss Seidel aberrations and optical vignetting, which approximate the imperfections introduced by real lenses.

3.2.1 Seidel aberrations

Von Seidel (1857) first introduced a set of five aberrations, now known as the *Seidel aberrations*, which build upon Gaussian optics to better approach the real-life behaviour of lenses. The Seidel aberrations are *spherical aberration*, *coma*, *astigmatism*, *field curvature* and *distortion*. Each of these describes a particular departure from a 'perfect' lens – as described by Gaussian optics – and changes the bokeh in a particular way. The further from the optical axis light enters a lens, the larger the aberrations will be. The Seidel aberrations are the five lowest-order aberration terms, and will be introduced mathematically in Section 4.2.3. In Figure 3.5, a set of light sources is rendered at two different distances – at the focus distance d and at $0.54d$ – with no aberrations, and in Figure 3.6 the same set of light sources is rendered but with the five Seidel aberrations applied. Note that spectral effects are not simulated in these figures.

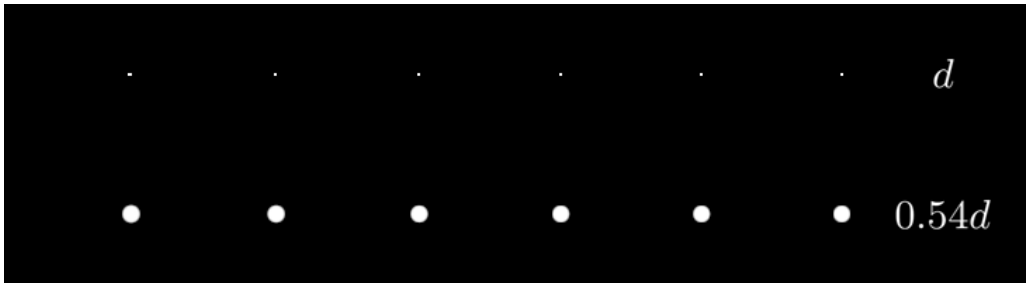


Figure 3.5: A set of six light sources rendered without aberrations. The points have varying distances from the optical axis, with the rightmost point being on the optical axis. The points are rendered twice, with different distances from the lens: at the focus distance d (top) and at $0.54d$ (bottom).

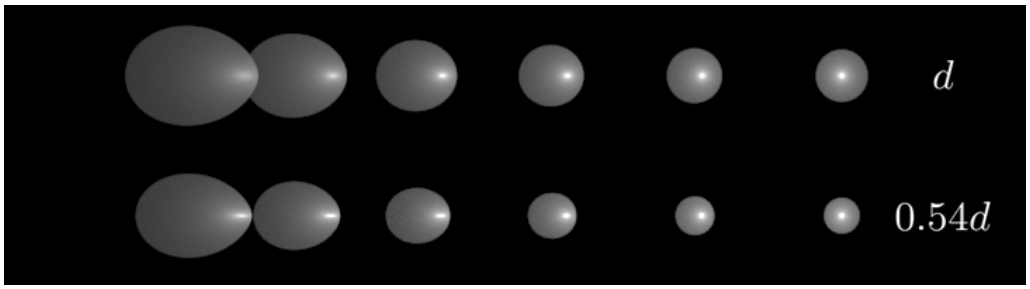
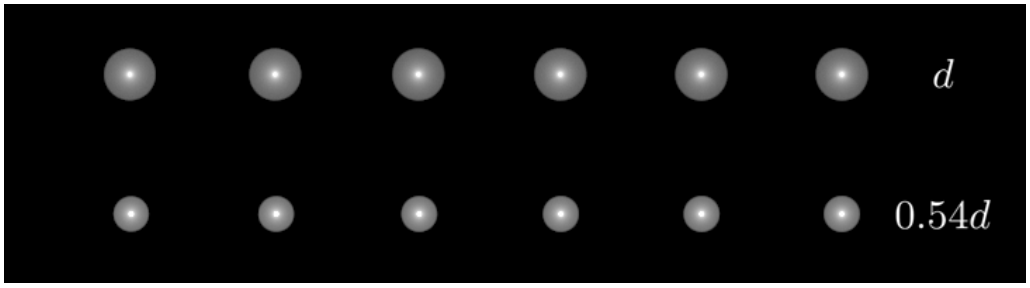


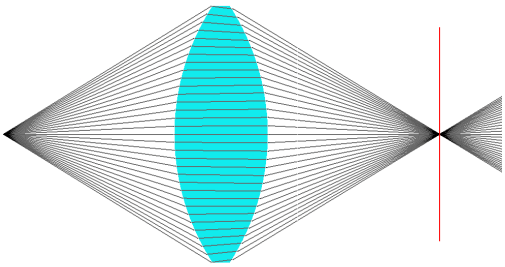
Figure 3.6: The same set of light sources, with the five Seidel aberrations applied.

Each of the Seidel aberrations is associated with a distinct change in appearance of the bokeh, which is explained and illustrated below.

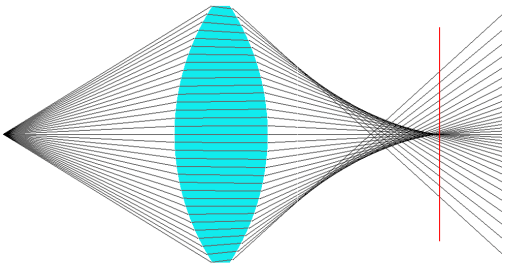
Spherical aberration A spherical lens surface does not focus all incoming light into a point, but along a section of the optical axis, which causes spherical aberration. This affects all rays, also when originating from a point on the optical axis (Figure 3.7).



(a) Bokeh rendered at d and at $0.54d$ with spherical aberration



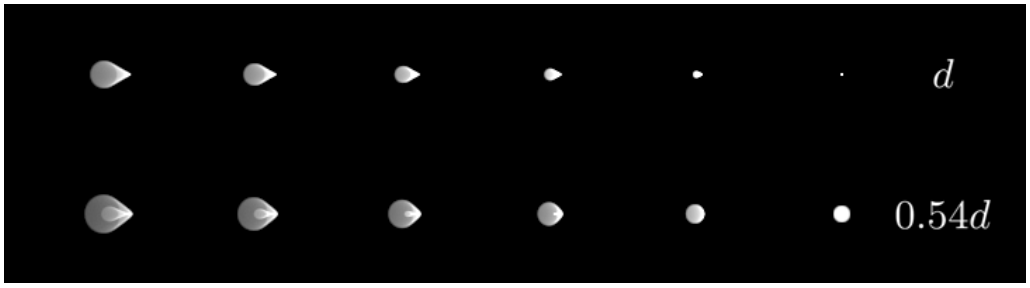
(b) Schematic, no aberrations



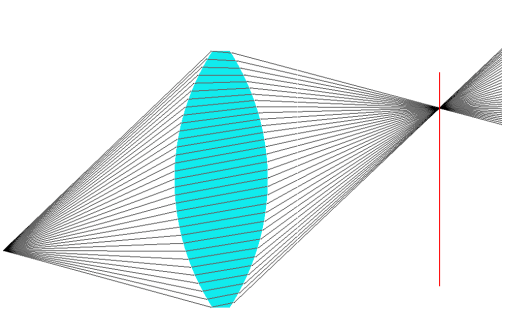
(c) Schematic, spherical aberration only

Figure 3.7: Spherical aberration

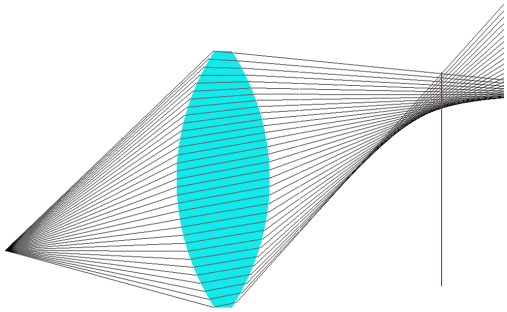
Coma Even if spherical aberration is eliminated, different parts of the lens still do not focus the incoming light onto a single point. This causes point sources to appear to have a tail (coma) like a comet. Coma only affects rays originating from a point away from the optical axis (Figure 3.8).



(a) Bokeh rendered at d and at $0.54d$ with coma



(b) Schematic, no aberrations

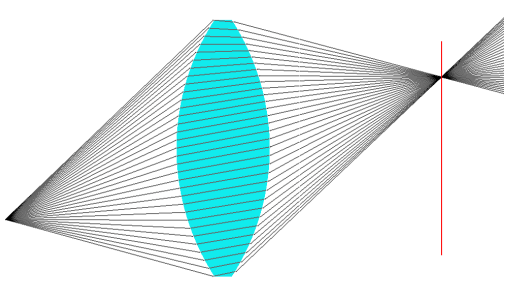


(c) Schematic, coma only

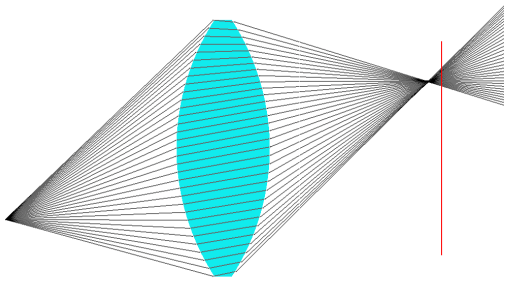
Figure 3.8: Coma

Astigmatism From the point of view of rays originating from a point away from the optical axis, the lens appears to be tilted, which causes astigmatism. This makes the focal length of the lens a function of the angle between the incoming light ray and the apparent long axis of the lens (Figure 3.9).

(a) Bokeh rendered at d and at $0.54d$ with astigmatism



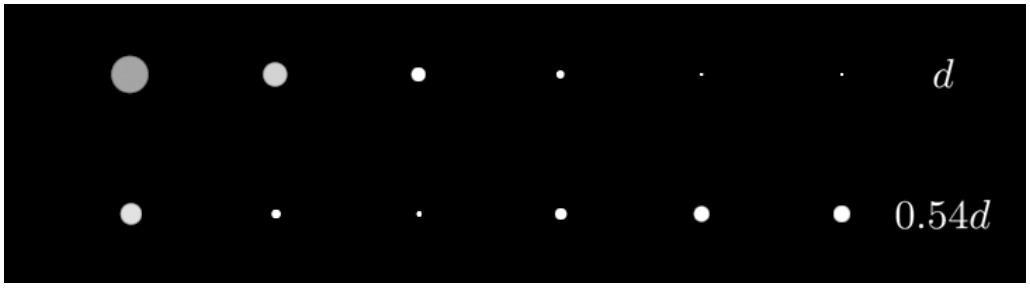
(b) Schematic, no aberrations



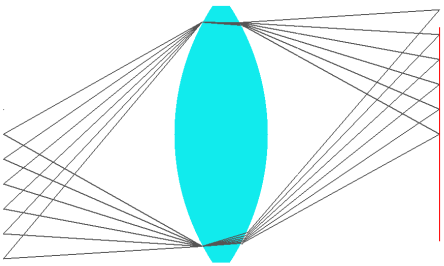
(c) Schematic, astigmatism only

Figure 3.9: Astigmatism

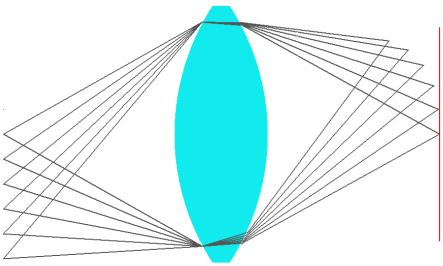
Field curvature The surface along which lie the focussed points of light sources – which themselves lie on a flat plane – is generally not a flat plane, but a curved surface. This is described by field curvature (Figure 3.10).



(a) Bokeh rendered at d and at $0.54d$ with field curvature



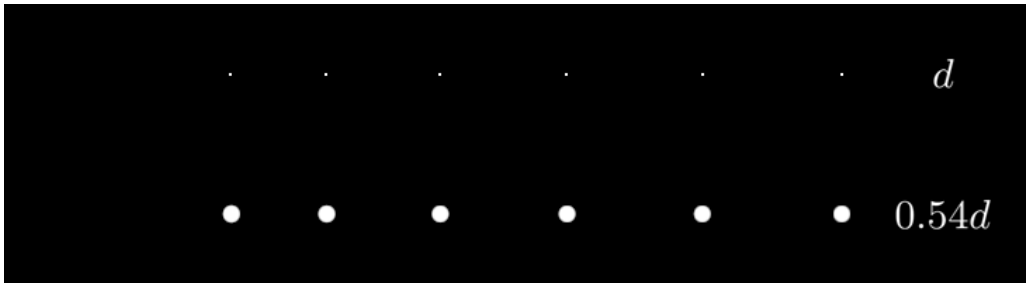
(b) Schematic, no aberrations



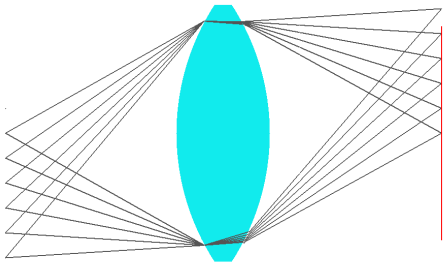
(c) Schematic, field curvature only

Figure 3.10: Field curvature

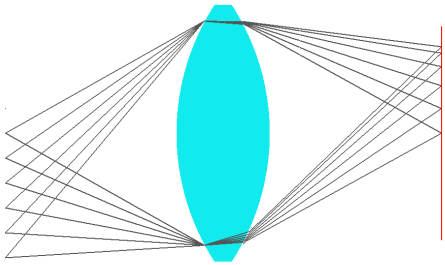
Distortion The distance from the optical axis of the image point is generally not linearly proportional to that of the source light point, which is described by distortion (Figure 3.11).



(a) Bokeh rendered at d and at $0.54d$ with distortion



(b) Schematic, no aberrations



(c) Schematic, distortion only

Figure 3.11: Distortion

3.2.2 Optical vignetting

With aberrations, we can calculate how a lens changes the ray vector. But that is only one part of the solution: the ray can also be blocked inside the lens and never reach the imaging sensor (*Gotanda et al.*, 2015). This effect is called *optical vignetting*. If light enters the lens off-axis, a portion of it may be blocked internally from fully travelling through the lens. The further we are from the optical axis, the larger this effect will be. The visual result on DOF is a darkening of off-axis pixels as less light reaches them, and cut-off bokeh shapes. An example is shown in Figure 3.12.

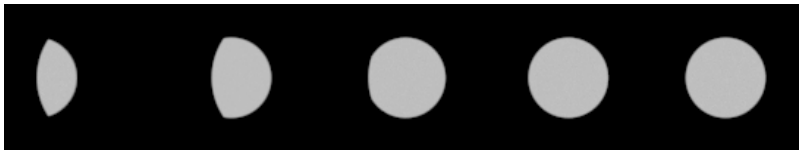


Figure 3.12: An example of optical vignetting. The right-most light source is on the optical axis.

3.3 Approaching realistic bokeh

So far, we have only discussed depth of field simulation methods that produce uniform bokeh with simple polygonal shapes. If we want to produce bokeh that more accurately approaches that

produced by real life lenses, we need more accurate simulations.

3.3.1 Sprites

Sprite-based approaches are post-processes that use a technique called *scattering*, in which a sprite of a bokeh shape is rendered to each pixel, scaled with the circle of confusion size of that pixel and with an alpha value inversely proportial to its area (*Demers*, 2004). Currently, sprite-based methods yield the largest amount of creative control and flexibility, as any input image can be used as bokeh sprite. An example of a sprite-based approach is Nuke's ZDefocus[1] (*Robinson*, 2013), which is widely used in the visual effects industry. It is possible to procedurally generate various bokeh shapes within a sprite-based approach, such as polygonal aperture shapes. Procedural generation allows for much control over the exact shape, as well as animated aperture changes (*Gotanda et al.*, 2015; *Abadie*, 2018).

If implemented accurately, a sprite-based approach is accurate for one point in space at most: the point at which the bokeh shape which is used as the sprite was captured.

3.3.2 Pencil maps

Gotanda et al. (2015) introduced *pencil maps*, which are pre-computed textures showing light paths as they emerge from some lens system, with the distance between the lens and the light source on one axis, and the on-screen distance from the center of the bokeh shape on the other, as in Figure 3.13. Making use of the rotational symmetry, we can take a vertical slice of the pencil map and map the pixels to a disk to obtain a bokeh sprite at various distances from the lens. This process is theoretically able to capture all axial aberrations.

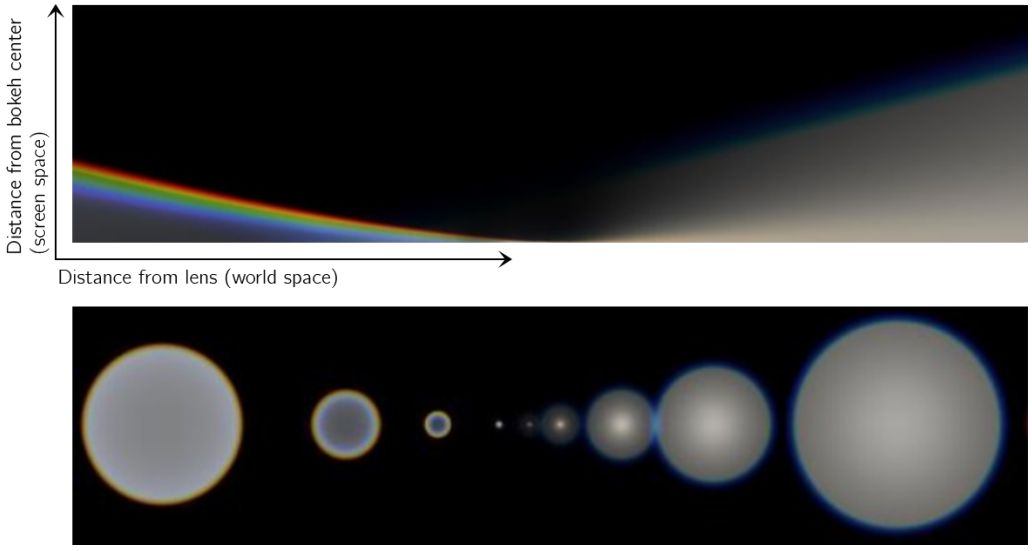


Figure 3.13: A pencil map (top) and bokeh sprites generated from vertical slices taken from it (bottom) (*Gotanda et al.*, 2015).

When implemented accurately, the resulting bokeh is correct for light sources along the optical axis (center of the image), and gets more and more incorrect the further away from it we get.

[1]Internally, ZDefocus does not render sprites but works by separating the scene into a number of depth layers, each of which are convoluted with a scaled version of the bokeh shape image. This helps with accurate depth sorting. The output image will be identical, provided that there are enough depth layers.

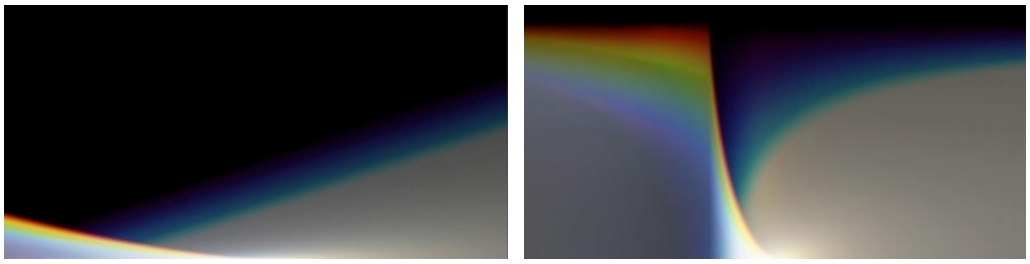


Figure 3.14: A normal pencil map (left) and its normalized version (right) (*Gotanda et al.*, 2015).

Pencil maps can be optimized by normalizing the height of each pixel to the circle of confusion size, which makes it possible for the pencil map to be used with any circle of confusion radius as this is no longer encoded in the pencil map itself. It also increases the precision around the focus point (where the circle of confusion is small), as more information is saved in otherwise black pixels. An example of a pencil map and its normalized version are illustrated in Figure 3.14.

3.3.3 Ray tracing

In 1984, *Cook et al.* (1984) introduced distribution ray tracing, which uses multiple rays to numerically integrate the integrals involved in motion blur, soft shadows and depth of field. In order to generate depth of field, the pinhole camera position is replaced by a 2D camera aperture which is sampled randomly. Effects like partial bokeh occlusion and circular bokeh shapes are achieved. Distribution ray tracing is still a widely used method, as it requires little extra resources to implement in an existing ray tracer and generates good results. *Cook et al.* (1984) used a thin lens model and Gaussian optics to generate camera rays, producing purely disk-shaped bokeh. *Kolb et al.* (1995) first described a monochromatic physically-based camera model and lens system implemented in a distribution ray tracer, that included multiple lens elements, as well as stops and apertures. This implementation produces realistic lens aberrations. Ray tracing a lens system is also possible as a post-process when using screen-space ray tracing. This can be done effectively for generating depth of field, as demonstrated by *Lee et al.* (2010).

Recently, 3D animation films such as Pixar's *Inside Out* have started using models of real life lens systems to increase realism[2]. A lens system can be fully ray traced quite easily, but doing so efficiently requires a good sampling strategy. *Wu et al.* (2013) observe that the straightforward way of generating a ray by connecting a random point on the image plane to a random point on the lens closest to the image plane is hardly efficient, as the majority of these rays are blocked by the rims of the successive lens elements. Instead, they calculate the entrance and exit pupils of the lens and place random points on those pupils. Additionally, an efficient method of rendering spectral effects is introduced. Their approach is capable of efficiently generating realistic spectral aberrations, as illustrated in Figure 3.15.

[2] `https://www.studiodaily.com/2015/07/pixars-inside-modeled-real-world-filmmaking-tools/`

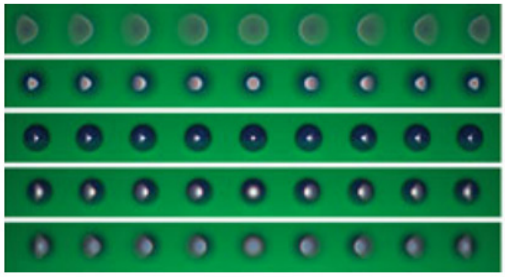


Figure 3.15: Simulated bokeh by *Wu et al.* (2013), showing realistic spectral aberrations.

Hullin et al. (2012) derive polynomial-based approximations for the origin and direction vectors of rays going through a complex lens system, in order to efficiently generate spectral aberrated bokeh, as well as lens flares. This is done by analytically solving the intersection of a ray with all surfaces and obtaining the Taylor series of the resulting equations. Terms of high orders can be truncated as desired to reduce the complexity of the equations. Because of this, their approximations take into account both lower (Seidel) and higher order aberrations. An advantage of using polynomials is that they are cheap to evaluate on most hardware.

These polynomial approximations were further explored by *Schrade et al.* (2016) (Figure 3.16), who fitted polynomial approximations to brute-force samples taken by ray tracing a lens system. Their implementation also supports anamorphic lenses. These approximations are then stored in a light field data structure, which can be quickly transformed for refocusing. They note a problem that is intrinsic to realistic lens modelling: it can be hard to obtain an exact description of the elements and materials used in a lens system, and even harder to obtain information about the specific lens coatings that were used.

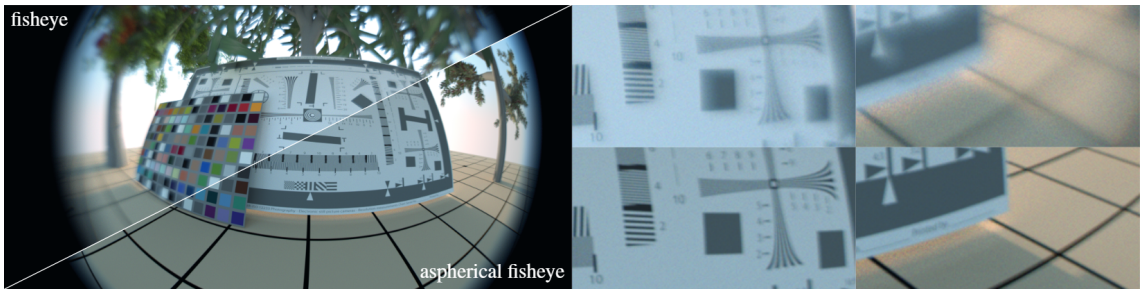


Figure 3.16: Fisheye (large field of view) lens simulations from *Schrade et al.* (2016).

In addition to an efficient ray intersection method for aspheric lenses, *Joo et al.* (2016) introduce a texture-based approach to model manufacturing imperfections as well as tiny dirt marks found in real life lenses. The imperfections introduced by grinding and polishing, which produce so-called 'onion ring bokeh', are approximated by applying a normal map to certain lens elements, yielding realistic bokeh shape textures (Figure 3.17).

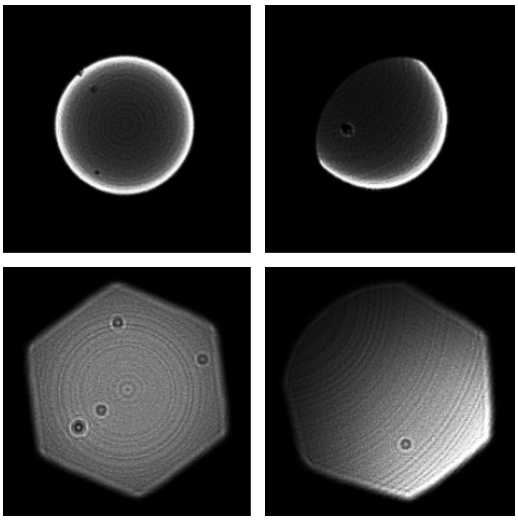


Figure 3.17: Bokeh shapes generated by modelling manufacturing imperfections using normal maps and dirt marks using textures in a ray traced lens system (*Joo et al.*, 2016)

Our method will approximate spectral aberrated bokeh taking into account Seidel aberrations, which will make our implementation similar to that of *Hullin et al.* (2012). We will propose a more artist-focused approach and analyze the applicability of Seidel aberration in the context of depth of field generation specifically.

Chapter 4

Optics and aberration theory

In order to model a lens system, we need to represent it as a number of variables, which we can calculate using Gaussian optics and aberration theory. In this chapter we will go over the math we need for this, with the end goal of calculating the coordinates at which a light ray hits a camera's imaging sensor after traversing the lens system.

4.1 The thick lens approximation

The most basic form of optics uses the *thin lens approximation*. In this approximation, a lens is represented by a single, flat surface at which all light refraction occurs. A more accurate approximation is the *thick lens approximation*, which also takes into account the thickness of the lens. Here, the lens is represented by two flat surfaces at which the refraction occurs, with a certain distance (thickness) between them. These surfaces are known as the *principal planes*. We define the principal plane at which the light enters the lens as the *front principal plane*, and the one where light exits the lens as the *rear principal plane*.

In our implementation, we can simplify the lens system by approximating it as one thick lens. In order to calculate the lens magnification later on, we need to calculate the positions of the principal planes[1], as well as the effective focal length of the lens system. One method to do this is the matrix method (*Hecht*, 2017). This method is particularly useful to us because of its linear nature, which makes it easy to work with when dealing with different lens systems with a variable number of glass elements. We will not go into the specifics here, but merely outline the equations used in our implementation. The full derivation can be found in *Hecht* (2017).

There are two matrices, the *refraction matrix* $\mathfrak{R}_i$ and the *transfer matrix* $\mathfrak{T}_i$, which are defined as

$$\mathfrak{R}_i = \begin{pmatrix} 1 & -\mathfrak{D}_i \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -(n_{i+1} - n_i)/r_i \\ 0 & 1 \end{pmatrix}, \tag{4.1}$$

where $\mathfrak{D}_i$ is the *power* of the ith refractive surface, n_i and n_{i+1} are the refractive indices before and after the ith refractive surface, respectively, and r_i is the radius of curvature of the ith refractive

[1] In our implementation we only really need the front principal plane. But the rear principal plane was used to calculate the field of view of the camera, which we used to automatically stretch the input image such that it filled the field of view for any lens system.

surface; and

$$\mathfrak{T}_i = \begin{pmatrix} 1 & 0 \\ d_{i,i+1}/n_{i+1} & 1 \end{pmatrix}, \tag{4.2}$$

where $d_{i,i+1}$ is the distance between the ith and $i+1$th surfaces. With these matrices, we can calculate the *system matrix* $\mathfrak{A}$ as

$$\mathfrak{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \mathfrak{T}_k\mathfrak{R}_k\mathfrak{T}_{k-1}\mathfrak{R}_{k-1}...\mathfrak{T}_2\mathfrak{R}_2\mathfrak{T}_1\mathfrak{R}_1, \tag{4.3}$$

where k is the number of refractive surfaces in the lens system. Now, using element a_{12} of the system matrix we can find the effective focal length as

$$f = -\frac{1}{a_{12}}, \tag{4.4}$$

the position of the front principal plane as

$$z_{\text{fpp}} = z_{\text{first surface}} + \frac{1 - a_{11}}{-a_{12}}, \tag{4.5}$$

and the position of the rear principal plane as

$$z_{\text{rpp}} = z_{\text{last surface}} + \frac{a_{22} - 1}{-a_{12}}. \tag{4.6}$$

We have now reduced the lens system to two planes and a focal length, which will be very useful later on.

4.1.1 Entrance and exit pupils

As introduced previously, the aperture stop is the main light limiting stop in a lens system. When seen through either side of the lens system, we instead see the *image* of the aperture stop, which is called the *entrance pupil* when seen from the direction of the incoming light, and the *exit pupil* when seen from the outgoing light direction. The entrance and exit pupils are useful for sampling and are also used in equations to be introduced later in this chapter.

The positions of the entrance and exit pupil planes are calculated by using a *chief ray*. This is any ray that intersects the center of the aperture stop and makes a nonzero angle with the optical axis. For the entrance pupil plane, a chief ray is traced from the center of the aperture stop backwards through the lens system, and then intersected with the z axis to find

$$z_{\text{pupil}} = \bar{\mathbf{O}}_{\mathbf{c}z} - \frac{\bar{\mathbf{O}}_{\mathbf{c}x,y}}{\bar{\mathbf{D}}_{\mathbf{c}x,y}}\bar{\mathbf{D}}_{\mathbf{c}z}, \tag{4.7}$$

where $\bar{\mathbf{O}}_{\mathbf{c}}$ and $\bar{\mathbf{D}}_{\mathbf{c}}$ are the origin and direction vectors, respectively, of the corresponding chief ray after tracing it through the lens system. The subscripts $_z$ and $_{x,y}$ indicate the z-component and the distance from the z-axis, respectively. The path can also be reversed in order to find the position of the exit pupil plane; then the chief ray is traced forwards through the lens system.

The radii of the pupils can be calculated with a *marginal ray* – a ray that touches the edge of the aperture stop and therefore also the edges of the entrance and exit pupils – as

$$r_{\text{pupil}} = \left[\bar{\mathbf{O}}_{\mathbf{m}} + \frac{z_{\text{pupil}} - \bar{\mathbf{O}}_{\mathbf{m}z}}{\bar{\mathbf{D}}_{\mathbf{m}z}}\bar{\mathbf{D}}_{\mathbf{m}}\right]_{x,y}, \tag{4.8}$$

where $\bar{\mathbf{O}}_{\mathbf{m}}$ and $\bar{\mathbf{D}}_{\mathbf{m}}$ are the origin and direction vectors, respectively, of the marginal ray at the same side of the lens system as the pupil.

4.2 Gaussian optics and aberrations

The equations we used so far are all *paraxial approximations*, meaning that they are only valid in the paraxial region – the region very close to the optical axis. Optics, when using the paraxial approximation, is also known as *Gaussian optics*. The further away from the optical axis we move, the larger the error in calculated values gets, when using the paraxial approximation. In order to improve our calculations, we can add *aberrations* to values calculated using paraxial approximations. Our goal is to calculate the imaging sensor plane coordinates at which an incident light ray ends up after moving through a lens system. To achieve this, we will first calculate these coordinates purely using Gaussian optics, and then add some aberrations, specifically the Seidel aberrations.

We assume that lens systems are symmetrical along the optical axis, which we will define as the z-axis. Vectors are either 2D vectors along a plane perpendicular to the z-axis or 3D vectors. 2D vectors are notated as $\mathbf{V}$, while 3D vectors, adding a z-coordinate, are notated as $\bar{\mathbf{V}}$ (note that the x and y components of $\bar{\mathbf{V}}$ are equal to those of $\mathbf{V}$). Most equations and definitions in this section come from *Born et al.* (1999). More extensive explanations and derivations for everything discussed here can be found in *Born et al.* (1999).

4.2.1 Gaussian image plane coordinates

In Figure 4.1, a schematic representation of a lens system is shown. Our first goal is to use Gaussian optics to calculate $\mathbf{P}_{\mathbf{1}}^{*}$, the Gaussian image plane coordinates.

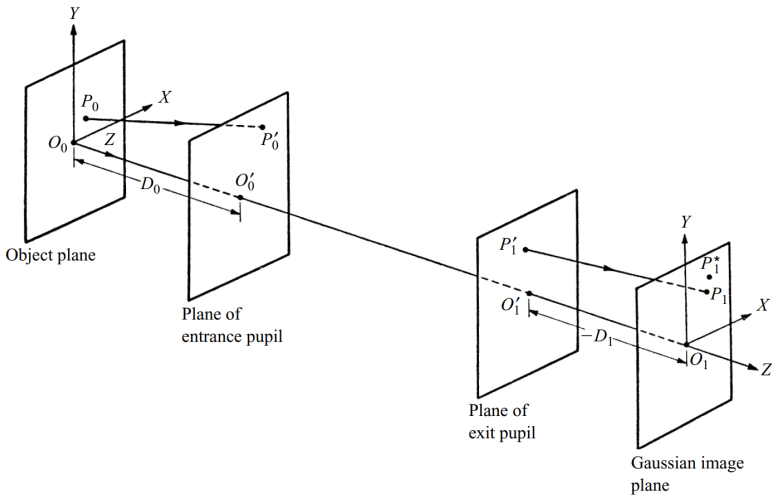


Figure 4.1: A schematic representation of a lens system (*Born et al.*, 1999)

We start out with the light source (object) plane coordinates $\mathbf{P_0}$, which we can calculate such that the input image exactly fills the screen, as

$$\mathbf{P_0} = \text{sensor width} \times \frac{D}{z_{\text{sensor}} - z_{\text{rpp}}} \mathbf{P_p}, \tag{4.9}$$

where $\mathbf{P_p}$ are the pixel coordinates, normalized such that $-0.5 \leq \mathbf{P_p}_x \leq 0.5$. We then randomly sample the exit pupil to find the exit pupil plane coordinates $\mathbf{P'_1}$, taking into account the exit pupil radius and, if desired, the aperture shape. We then simply find the entrance pupil plane coordinates $\mathbf{P'_0}$ as

$$\mathbf{P'_0} = \frac{1}{M'} \mathbf{P'_1}, \tag{4.10}$$

where M' is the lateral magnification between the entrance and exit pupils, or

$$M' = \frac{\text{exit pupil radius}}{\text{entrance pupil radius}}. \tag{4.11}$$

Then, we can find the Gaussian image plane coordinates simply as

$$\mathbf{P_1^*} = M \mathbf{P_0}, \tag{4.12}$$

where M, the lateral magnification between the object and image planes, is calculated as[2]

$$M = -\frac{f}{x_0} = -\frac{f}{z_{\text{light source}} + z_{\text{fpp}} - f}. \tag{4.13}$$

We can obtain $z_{\text{light source}}$ from the pixel depth value D – which we assume to be the distance between the pinhole camera and the light source – by

$$z_{\text{light source}} = \sqrt{D^2 - |\mathbf{P_0}|^2}. \tag{4.14}$$

Finally we need to add the aberration vector $\mathbf{\Delta P_1}$, which we will calculate later on, to obtain

$$\mathbf{P_1} = \mathbf{P_1^*} + \mathbf{\Delta P_1}. \tag{4.15}$$

4.2.2 Imaging sensor plane coordinates

We have now found the Gaussian image plane coordinates, but these are only equal to the imaging sensor plane coordinates if the light source is at precisely the focus distance. In the general case, we need to find the ray between $\bar{\mathbf{P}}_\mathbf{1}$ and some other point $\bar{\mathbf{Q}}$ to obtain the imaging sensor plane coordinates, as illustrated in Figure 4.2.

[2]From lecture slide 9 at `https://ocw.mit.edu/courses/mechanical-engineering/2-71-optics-spring-2009/video-lectures/lecture-5-thick-lenses-the-composite-lens-the-eye/MIT2_71S09_lec05.pdf`.

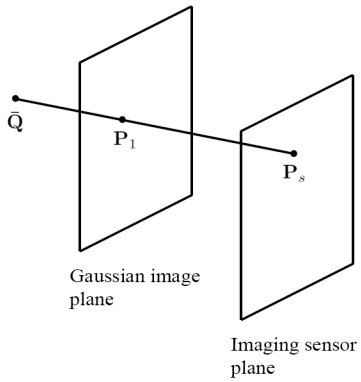


Figure 4.2: In order to find the imaging sensor coordinates $\mathbf{P_s}$, we need to construct a ray from some point $\bar{\mathbf{Q}}$ and the Gaussian image plane coordinates $\bar{\mathbf{P}}_\mathbf{1}$.

We can find a point $\bar{\mathbf{Q}}$ by looking at the *wavefront* of the outgoing light, that intersects the Gaussian image plane at $\mathbf{P_1}$. $\bar{\mathbf{Q}}$ is the intersection point of the ray from $\bar{\mathbf{P}}'_\mathbf{1}$ to $\bar{\mathbf{P}}_\mathbf{1}$ with the wavefront. The ray intersects both the wavefront and its Gaussian approximation, which allows us to use the simpler Gaussian wavefront. In Gaussian optics, the wavefront is a perfect sphere centered at $\bar{\mathbf{P}}^*_\mathbf{1}$ that also intersects the center of the exit pupil. We can therefore calculate $\bar{\mathbf{Q}}$ as

$$\bar{\mathbf{Q}} = \bar{\mathbf{P}}^*_\mathbf{1} + \frac{\bar{\mathbf{P}}'_\mathbf{1} - \bar{\mathbf{P}}^*_\mathbf{1}}{|\bar{\mathbf{P}}'_\mathbf{1} - \bar{\mathbf{P}}^*_\mathbf{1}|} R_w, \tag{4.16}$$

where R_w is the radius of the spherical wavefront, which can be calculated as

$$R_w = |\bar{\mathbf{P}}^*_\mathbf{1} - z_{\text{exit pupil}}\hat{\mathbf{z}}|, \tag{4.17}$$

where $\hat{\mathbf{z}}$ is the unit vector along the positive z-direction. In order to find the sensor plane coordinates $\mathbf{P_s}$, we set up a ray starting at $\bar{\mathbf{P}}_\mathbf{1}$ and ending at $\bar{\mathbf{Q}}$, with direction vector

$$\bar{\mathbf{D}} = \bar{\mathbf{P}}_\mathbf{1} - \bar{\mathbf{Q}}. \tag{4.18}$$

We can then find the sensor plane coordinates as

$$\bar{\mathbf{P}}_\mathbf{s} = \bar{\mathbf{P}}_\mathbf{1} + d_{\text{image}\to\text{sensor}} \frac{\bar{\mathbf{D}}}{\bar{\mathbf{D}} \cdot \hat{\mathbf{z}}}, \tag{4.19}$$

where $d_{\text{image}\to\text{sensor}}$ is the distance between the Gaussian image plane and the imaging sensor plane. In order to calculate this distance, we first calculate the z-coordinate of the Gaussian image plane as

$$\bar{\mathbf{P}}_{\mathbf{1}z} = z_{\text{exit pupil}} - D_1, \tag{4.20}$$

where D_1 is the negative distance between the exit pupil plane and the Gaussian image plane, as illustrated in Figure 4.1. We can calculate D_1 as

$$D_1 = MM'D_0 = MM'(\bar{\mathbf{P}}_{\mathbf{0}z} + z_{\text{entrance pupil}}), \tag{4.21}$$

where D_0 is the distance between the object plane and the plane of the entrance pupil. Combining Equations 4.20 and 4.21, we get

$$\bar{\mathbf{P}}_{\mathbf{1}z} = z_{\text{exit pupil}} - MM'(\bar{\mathbf{P}}_{\mathbf{0}z} + z_{\text{entrance pupil}}). \tag{4.22}$$

To calculate $\bar{\mathbf{P}}_{\mathbf{s}z}$, the z coordinate of the sensor plane, we can use the same formula, but using the focus distance instead of z_{object}, and using constant values for the exit pupil and entrance pupil plane positions, indicated with $_{550}$ as these values are calculated using 550 nm light[3]:

$$\bar{\mathbf{P}}_{\mathbf{s}z} = z_{\text{exit pupil, 550}} - M_{550}M'_{550}(\text{focus distance} + z_{\text{entrance pupil, 550}}). \tag{4.23}$$

Here, M_{550} and M'_{550} are calculated again using Equations 4.13 and 4.11, again using constant values obtained using 550 nm light.

We can now calculate the distance between the two as

$$d_{\text{image}\to\text{sensor}} = \bar{\mathbf{P}}_{\mathbf{s}z} - \bar{\mathbf{P}}_{\mathbf{1}z}. \tag{4.24}$$

4.2.3 The aberration vector

When calculating the aberrations, it is useful to first define two normalized coordinate systems on the object and image planes, such that within Gaussian optics the object and image plane coordinates $\mathbf{p_0}$ and $\mathbf{p_1}$ are equal:

$$\mathbf{p_0} = \mathbf{p_1}. \tag{4.25}$$

For this we will define two units of length l_0 and l_1 in the object and image planes, respectively, such that

$$\frac{l_1}{l_0} = M. \tag{4.26}$$

Similarly, we define two units of length λ_0 and λ_1 in the entrance and exit planes, respectively, such that

$$\frac{\lambda_1}{\lambda_0} = M'. \tag{4.27}$$

We can then define $\mathbf{p_0}$ and $\mathbf{p_1}$ by normalizing the object and image plane coordinates as

$$\begin{aligned} \mathbf{p_0} &= C\frac{\mathbf{P_0}}{l_0} \\ \mathbf{p_1} &= C\frac{\mathbf{P_1}}{l_1}, \end{aligned} \tag{4.28}$$

[3]This is approximately the wavelength which has the highest perceived luminance, which is why we chose it for these calculations.

Where C is a constant defined by

$$C = \frac{n_0 l_0 \lambda_0}{D_0} = \frac{n_1 l_1 \lambda_1}{D_1}, \tag{4.29}$$

which is chosen in order to simplify calculations later on. n_1 is the refractive index of the image space. We can safely assume that $n_0 = 1$, and we will define $\lambda_0 \equiv 1$ and $l_0 \equiv 1$. Equation 4.29 then reduces to

$$C = \frac{1}{D_0}. \tag{4.30}$$

We can then rewrite Equation 4.15 as

$$\mathbf{P_1} = \mathbf{P_1^*} + D_1 \mathbf{\Delta p}, \tag{4.31}$$

where

$$\mathbf{\Delta p} = \mathbf{p_1} - \mathbf{p_0} = \frac{n_1 \lambda_1}{D_1}(\mathbf{P_1} - \mathbf{P_1^*}) = \frac{1}{D_1}(\mathbf{P_1} - \mathbf{P_1^*}). \tag{4.32}$$

We can do a power series expansion in order to calculate this new aberration vector $\mathbf{\Delta p}$. The terms of this expansion up to the third order are also known as the Seidel aberrations, and are

$$\mathbf{\Delta p} = \begin{bmatrix} B\rho^3 \sin\theta - 2Fy_0\rho^2 \sin\theta\cos\theta + Dy_0^2\rho\sin\theta \\ B\rho^3\cos\theta - Fy_0\rho^2(1+2\cos^2\theta) + (2C+D)y_0^2\rho\cos\theta - Ey_0^3 \end{bmatrix}, \tag{4.33}$$

where $y_0 = \mathbf{p_0}_y$, ρ and θ are the polar coordinates of the sampled point on the exit pupil plane and B, C, D, E and F are the five Seidel coefficients, each of which controls the contribution of one Seidel aberration: B controls spherical aberration, C controls astigmatism, D controls field curvature, E controls distortion and F controls coma. Equation 4.33 assumes that $\mathbf{p_0}_x = 0$. So for the general case, we need to do some rotations in order to accurately calculate $\mathbf{\Delta p}$.

4.2.4 Calculating the Seidel coefficients

The five Seidel coefficients can be calculated using the following equations from *Born et al.* (1999):

$$\begin{aligned}
B &= \frac{1}{2}\sum_i h_i^4 \frac{b_i}{r_i^3}(n_i - n_{i-1}) + h_i^4 K_i^2 \left(\frac{1}{n_i s_i'} - \frac{1}{n_{i-1}s_i}\right), \\
C &= \frac{1}{2}\sum_i h_i^4 k_i^2 \frac{b_i}{r_i^3}(n_i - n_{i-1}) + (1 + h_i^2 k_i K_i)^2 \left(\frac{1}{n_i s_i'} - \frac{1}{n_{i-1}s_i}\right), \\
D &= \frac{1}{2}\sum_i h_i^4 k_i^2 \frac{b_i}{r_i^3}(n_i - n_{i-1}) + h_i^2 k_i K_i (2 + h_i^2 k_i K_i)\left(\frac{1}{n_i s_i'} - \frac{1}{n_{i-1}s_i}\right) - K_i \left(\frac{1}{n_i^2} - \frac{1}{n_{i-1}^2}\right), \\
E &= \frac{1}{2}\sum_i h_i^4 k_i^3 \frac{b_i}{r_i^3}(n_i - n_{i-1}) + k_i (1 + h_i^2 k_i K_i)(2 + h_i^2 k_i K_i)\left(\frac{1}{n_i s_i'} - \frac{1}{n_{i-1}s_i}\right) - \frac{1 + h_i^2 k_i K_i}{h_i^2}\left(\frac{1}{n_i^2} - \frac{1}{n_{i-1}^2}\right), \\
F &= \frac{1}{2}\sum_i h_i^4 k_i \frac{b_i}{r_i^3}(n_i - n_{i-1}) + h_i^2 K_i (1 + h_i^2 k_i K_i)\left(\frac{1}{n_i s_i'} - \frac{1}{n_{i-1}s_i}\right),
\end{aligned} \tag{4.34}$$

which are sums over the i refractive surfaces in a lens system. For simplicity we will only use lenses that consist of purely spherical surfaces. That eliminates the first terms in the equation, as $b_i = 0$ for spherical surfaces. Equation 4.34 then reduces to

$$
\begin{aligned}
B &= \frac{1}{2}\sum_i h_i^4 K_i^2 \left(\frac{1}{n_i s_i'} - \frac{1}{n_{i-1} s_i}\right), \\
C &= \frac{1}{2}\sum_i (1 + h_i^2 k_i K_i)^2 \left(\frac{1}{n_i s_i'} - \frac{1}{n_{i-1} s_i}\right), \\
D &= \frac{1}{2}\sum_i h_i^2 k_i K_i (2 + h_i^2 k_i K_i) \left(\frac{1}{n_i s_i'} - \frac{1}{n_{i-1} s_i}\right) - K_i \left(\frac{1}{n_i^2} - \frac{1}{n_{i-1}^2}\right), \\
E &= \frac{1}{2}\sum_i k_i (1 + h_i^2 k_i K_i)(2 + h_i^2 k_i K_i) \left(\frac{1}{n_i s_i'} - \frac{1}{n_{i-1} s_i}\right) - \frac{1 + h_i^2 k_i K_i}{h_i^2} \left(\frac{1}{n_i^2} - \frac{1}{n_{i-1}^2}\right), \\
F &= \frac{1}{2}\sum_i h_i^2 K_i (1 + h_i^2 k_i K_i) \left(\frac{1}{n_i s_i'} - \frac{1}{n_{i-1} s_i}\right).
\end{aligned}
\tag{4.35}
$$

There are still some variables here that we need to define. First are the refractive indices and distances, which are illustrated in Figure 4.3:

- n_i is the refractive index of the medium that follows the ith lens surface,
- n_{i-1} is the refractive index of the medium that precedes the ith lens surface,
- $-s_i$ is the distance between the object plane and the ith lens surface along the optical axis,
- s_i' is the distance between the ith 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 ith lens surface along the optical axis, and
- t_i' is the distance between the ith lens surface and the plane of the exit pupil along the optical axis.

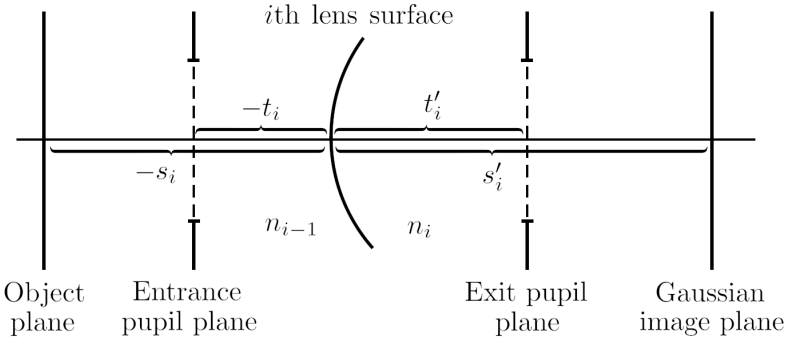


Figure 4.3: An illustration of a single surface in a lens system.

Since the Gaussian image formed by the first i surfaces of the system is the object for the $(i+1)$th surface, and also the exit pupil of the ith surface is the entrance pupil of the $(i+1)$th surface, we

can define

$$\begin{aligned} s_{i+1} &= s_i' - d_i, \\ t_{i+1} &= t_i' - d_i, \end{aligned} \tag{4.36}$$

where d_i is the distance between the poles of the ith and the $(i+1)$th surface.

K_i can be calculated from the *Abbe relations* (*Born et al.*, 1999) by

$$K_i = n_{i-1}\left(\frac{1}{r_i} - \frac{1}{s_i}\right) = n_i\left(\frac{1}{r_i} - \frac{1}{s_i'}\right), \tag{4.37}$$

where r_i is the radius of curvature of the ith surface. r_i is positive when the surface is convex towards light incident from the negative z-direction[4]. If we rewrite Equation 4.37 we can find the following equation for s_i':

$$s_i' = \frac{r_i s_i n_i}{r_i n_{i-1} + s_i(n_i - n_{i-1})}. \tag{4.38}$$

Similarly, we can rewrite the other Abbe relation

$$L_i = n_{i-1}\left(\frac{1}{r_i} - \frac{1}{t_i}\right) = n_i\left(\frac{1}{r_i} - \frac{1}{t_i'}\right), \tag{4.39}$$

to find

$$t_i' = \frac{r_i t_i n_i}{r_i n_{i-1} + t_i(n_i - n_{i-1})}. \tag{4.40}$$

Finally we need to calculate the "abbreviations" h_i and k_i, which are introduced to simplify the equations. h_i is calculated by

$$\begin{aligned} h_1 &= \frac{s_1}{t_1 - s_1}, \\ h_{i+1} &= \frac{s_{i+1}}{s_i'} h_i, \end{aligned} \tag{4.41}$$

as long as $\lambda_0 = 1$, which we have defined before. Finally, k_i is calculated as

$$\begin{aligned} k_1 &= \frac{t_1(t_1 - s_1)}{n_0 s_1}, \\ k_{i+1} &= k_1 + \sum_{j=1}^{i} \frac{d_j}{n_j h_j h_{j+1}}. \end{aligned} \tag{4.42}$$

Now that we know how to calculate everything needed to find the imaging sensor plane coordinates $\mathbf{P_1}$. In the end, we need to keep track of the following parameters in our implementation:

- The five Seidel coefficients B, C, D, E and F,
- The focal length f,

[4] As an example, the lens surface illustrated in Figure 4.3 has a positive curvature.

- The entrance and exit pupil positions $z_{\text{entrance pupil}}$ and $z_{\text{exit pupil}}$,
- The entrance and exit pupil radii,
- The front and rear principal plane positions z_{fpp} and z_{rpp}.

All these parameters depend on the wavelength of the light[5], and the Seidel coefficients also depend on the distance between the light source and the entrance pupil (to find s_1 and t_1). In the next chapter we will discuss our implementation, and how we keep track of these changing parameters.

[5]Note that the rear principal plane position is only used to scale the input image. As such, a constant value is desired, and the wavelength dependence can be ignored.

Chapter 5

Implementation

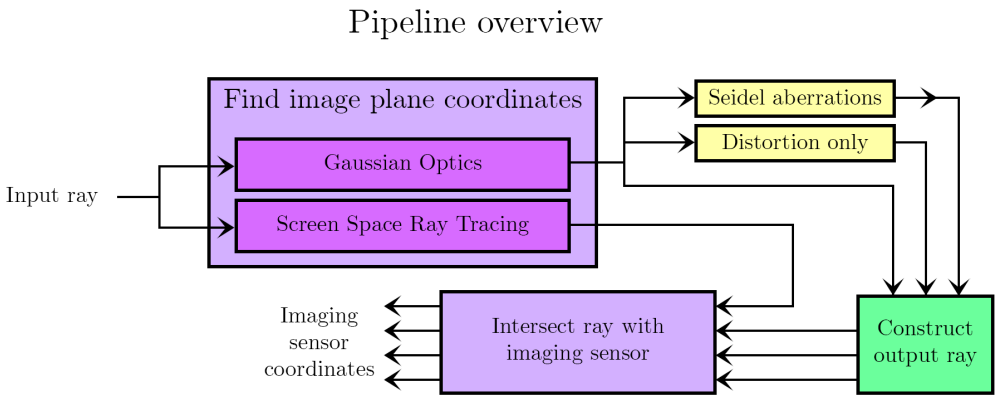


Figure 5.1: A schematic overview of the implementation

In order to test the accuracy of the Seidel aberrations, we have implemented a pipeline that takes as input an incoming light ray, the wavelength of the ray and a description of a lens system, simulates how the lens system transforms the ray, and outputs the point at which the transformed ray hits a certain imaging sensor. A schematic overview of the pipeline is shown in Figure 5.1. We chose to trace rays from object to image space and not the other way around – even though this is not the way it is usually done in ray tracers – because the existing literature on Seidel aberrations uses this assumption as well. The implementation should yield similar results if we reverse the order of the lens surfaces or trace the rays in the other direction.

5.1 Ray transformation and DOF methods

We use five different methods in order to simulate the ray transformation and spectral effects, which we will refer to by the shortened names in brackets:

1. Screen Space Ray Tracing (*SSRT*)
2. Gaussian optics (*Gaussian*)
3. Gaussian optics + Seidel aberrations + primitive optical vignetting (*Seidel*)

4. Gaussian optics + Distortion + primitive optical vignetting (*Distortion + vignetting*)
5. Gaussian optics + pencil map (*Pencil map*)

Screen Space Ray Tracing (*SSRT*): The incoming light ray is traced through the lens system using spectral ray tracing, using the algorithm described by *Kolb et al.* (1995). Only refractions are simulated, and it is assumed that 100% of the rays pass through each lens, as simulating the effects of anti-reflective lens coatings that exist on real life lenses is beyond the scope of this thesis.

In order to assess the validity of the straightforward ray tracing algorithm as a reference implementation, we compared the simulation of a certain lens description with photographs taken using the actual lens the description was based on. The results are shown in Figure 5.2. Visible effects that are missing from the SSRT simulation include artifacts introduced by manufacturing defects and/or dirt and grime on the lenses, diffraction patterns at the edges of the bokeh. Additionally, in the left four images, the bokeh shape seems to be slightly warped, which may be caused by small differences between the real lens and the separately obtained lens description. In general, though, SSRT seems to resemble the photographed bokeh well, which makes it a good choice for a reference implementation.

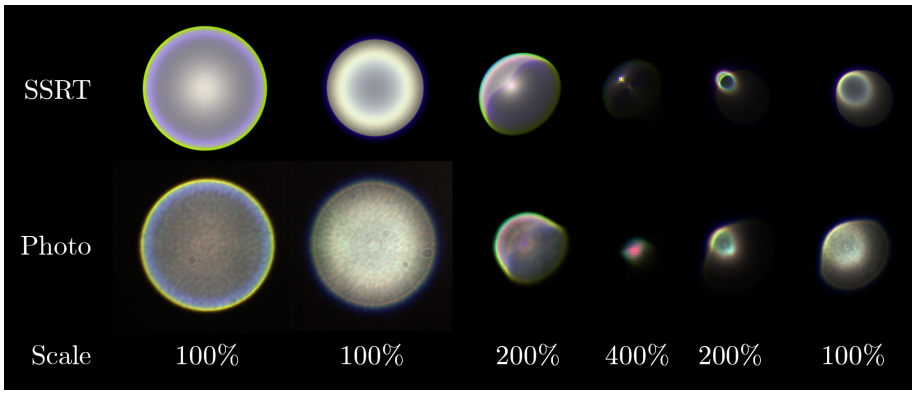


Figure 5.2: A selection of photos of bokeh captured using a physical camera and their simulated analogs, generated using SSRT.

Gaussian optics (*Gaussian*): The incoming light ray is transformed using the math introduced in Sections 4.1, 4.2.1 and 4.2.2, using purely the Gaussian (paraxial) approximations and not including any aberrations. This method does not use any spectral rendering, and represents all methods that produce perfectly uniform disk-shaped bokeh.

Gaussian optics + Seidel aberrations + primitive optical vignetting (*Seidel*): The incoming light ray is again transformed using Gaussian optics, but Seidel aberrations are applied as well, as described in Section 4.2.3. This method is fully spectral, and adds a primitive form of optical vignetting, by checking if the ray passes through the aperture of the first lens surface.

Gaussian optics + distortion + primitive optical vignetting (*Distortion + vignetting*): Similar to the *Seidel* method, this method adds upon the Gaussian optics method, but does not fully implement Seidel aberrations. Only the distortion aberration is added, as well as primitive optical vignetting. This method does not use any spectral rendering.

Gaussian optics + pencil map (*Pencil map*): This final method also builds upon the Gaussian optics simulation, but adds crude spectral effects to it when rendering to the screen by multiplying the sampled pixel color associated with the ray with a color looked up in a pencil map, as described by *Gotanda et al.* (2015). This method was chosen because it aims to add realism to pure Gaussian optics, but in a simplified way.

For the *SSRT*, *Gaussian*, *Seidel* and *Distortion + vignetting* methods, each input ray is assigned a random wavelength, the associated CIEXYZ color of which is then looked up in a table (described in more detail in Section 6.1). This color is then multiplied component-wise by the sampled RGB pixel color, the result of which is plotted to the screen at the calculated pixel coordinates. This is applied with the *Gaussian* and *Distortion + vignetting* methods as well even though they do not use any spectral effects, because this makes sure that the output color will converge to the same result when the input color is the same. This is done in a similar way when using the *Pencil map* method, as it is already taken care of by filling the pencil map in the first place.

The incoming ray is generated by starting at a certain object point and then randomly sampling the entrance pupil. This significantly reduces the number of rays that do not fully travel the lens and therefore decreases the number of samples necessary for convergence.

As our implementation is built on tracing a single ray at a time, occlusion effects other than optical vignetting are not simulated.

5.2 Lens system

During the initialization stage, a model of a lens system is loaded from a ZEMAX file, which is a widely used file format used in the optical design application Zemax OpticStudio[1]. We use this to obtain the centers, radii and apertures of the spherical lens surfaces, as well as the identifiers of the used glass types. These identifiers are used to look up coefficients to be used to calculate the refractive index n of the glasses at a certain wavelength λ, using one of two formulas, depending on the available coefficients (A_i, B_i, C_i):

$$\begin{aligned} n &= \sqrt{1 + \frac{B_1\lambda^2}{\lambda^2 - C_1} + \frac{B_2\lambda^2}{\lambda^2 - C_2} + \frac{B_3\lambda^2}{\lambda^2 - C_3}}, \\ n &= \sqrt{A_0 + A_1\lambda^2 + A_2\lambda^{-2} + A_3\lambda^{-4} + A_4\lambda^{-6} + A_5\lambda^{-8}}. \end{aligned} \tag{5.1}$$

The first of these equations is known as the *Sellmeier equation*, the second one is the dispersion formula used in the HOYA glass catalog[2]. The coefficients are obtained from `https://refractiveindex.info/`[3] (*Polyanskiy*, 2018).

Using equations from Section 4.1, the entrance pupil radii and positions, as well as the front and rear principal plane positions and the focal length are calculated for a set of 64 different wavelengths and stored in a lookup table. The same is done for the Seidel coefficients, which are calculated for 64 different distances to the lens for each of the 64 wavelengths using the Equations in Section 4.2.4.

[1] `https://www.zemax.com/`

[2] `http://www.hoya-opticalworld.com/english/technical/002.html`

[3] The whole dataset of coefficients and other glass data can be found at `https://github.com/polyanskiy/refractiveindex.info-database`.

5.2.1 Influence of lens parameters

The lens parameters (focal length, pupil radii and positions and principal plane positions) have various uses in the simulation, and impact the end result in different ways. The lateral magnification M (Equation 4.13) depends on the focal length and front principal plane position. Similarly, the lateral magnification between the entrance and exit pupil planes M' (Equation 4.11) depends on the entrance and exit pupil radii. M directly impacts the circle of confusion radius. The entrance pupil position is used to find D_0 (as illustrated in Figure 4.1) and the exit pupil position is used to find $\bar{\mathbf{P}}_{\mathbf{1}z}$ and $\bar{\mathbf{P}}_{\mathbf{s}z}$, which, together with M and M', are used to calculate $d_{\text{image}\to\text{sensor}}$ (Equation 4.24). This further impacts the circle of confusion size and general bokeh shape, as it is used to obtain the sensor plane coordinates (Equation 4.19). The rear principal plane value is only used to calculate the size of the input image such that it fills the field of view of the lens. For this an average value for the lens is used, and therefore the wavelength dependence of this parameter can be ignored.

Because the influence of the lens parameters is mainly concentrated in $d_{\text{image}\to\text{sensor}}$ and M, we will generate renders where only those variables are wavelength-dependent. As M depends on f and z_{fpp}, and $d_{\text{image}\to\text{sensor}}$ scales linearly with M, we will distill the lens parameters to only three: f, z_{fpp} and $\frac{1}{-M}d_{\text{image}\to\text{sensor}}$. We use $-M$ instead of M as this will later give some additional insight in the values of this last parameter.

5.3 Sensor position

Each method requires a fixed sensor plane position. We chose not to use the same sensor plane position for all methods, as the perceived focus distance (the distance at which the circle of confusion is actually smallest) can differ between methods, depending on the specific aberrations of the different lenses. So for a specified focus distance, we must calculate the image sensor plane position.

To find this location for the *SSRT* method, we trace rays originating at the focus distance on the optical axis through the lens and find the weighted average over the whole color spectrum and entrance pupil (weight = luminance) of the distance at which they come into focus. This position will be used as the sensor distance.

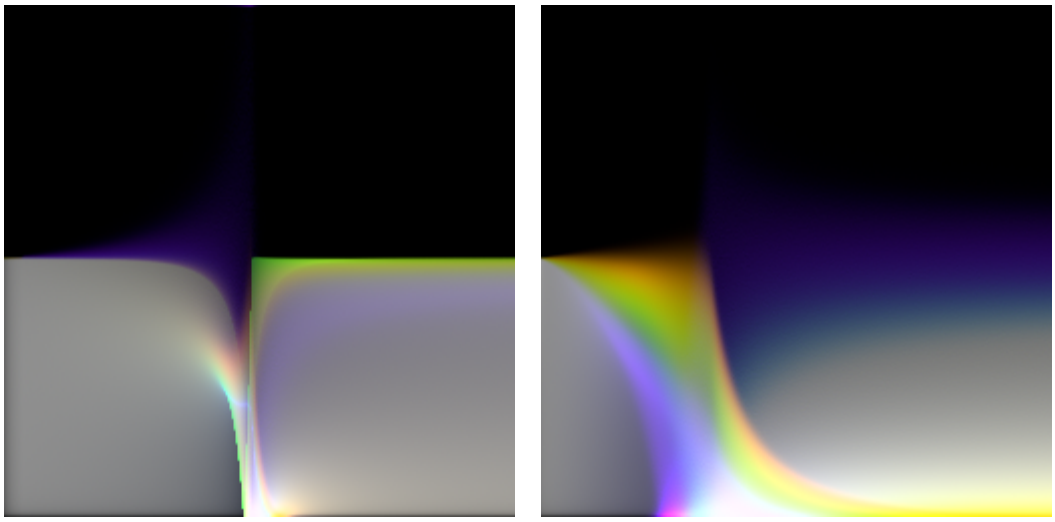
For the *Seidel* method we obtain a similar weighted average, but trace only paraxial rays, as we should use Gaussian optics only.

The other methods can directly use the specified focus distance, as there are no aberrations present that can change the perceived focus distance.

5.4 Filling the pencil map

We generate a 256×256 pixel pencil map, where the x-coordinate is the distance between the light source and first lens surface and the y-coordinate is the distance from the optical axis where the light ray intersects the imaging sensor. For each distance, we use stratification to uniformly sample at 512 wavelengths and 512 entrance pupil coordinates[4]. These samples are plotted to the pencil map such that precisely 99% of the luminance is stored in the bottom 128 pixels. This way, we can account for both the 'sharp' bokeh shape in the bottom 128 pixels and for any leakage, which has room in the top 128 pixels. Two examples of pencil maps are illustrated in Figure 5.3.

[4]We only need to do 2D calculations as the system is rotationally symmetric, so we only need to spread these samples over one axis of the entrance pupil.



(a) Nikkor 50mm f/1.8 (b) Single lens

Figure 5.3: Two examples of pencil maps as generated by the program, using two different lenses (processed for visibility)

When sampling the pencil map, we linearly interpolate between the four closest values based on the light source's distance to the lens and the position on the entrance pupil.

5.5 Importance sampling

To ensure fairly uniform noise in the final image, we calculate the number of samples to be taken separately for each pixel. Each pixel i gets assigned a weight equal to

$$W_i = \mathrm{CoC}_i \times l_i, \tag{5.2}$$

where l_i is the pixel's luminance as calculated using Equation 6.1, and CoC_i is the pixel's circle of confusion, which is estimated by taking a single sample. It may be necessary to clamp CoC_i in order to reduce the impact of pixels with a very large circle of confusion, and to keep $W_i > 0$ even when $\mathrm{CoC}_i = 0$.

In our implementation, each frame adds a specified number of samples N over the entire image, and all frames are added together to the final output image, which is then normalized by diving it by the number of frames. For each frame, the number of samples S_i to be taken for pixel i is determined as

$$S_i = \mathrm{floor}(\alpha_i W_i) + \begin{cases} 1, & \text{if rnd}() < \alpha_i W_i - \mathrm{floor}(\alpha_i W_i) \\ 0, & \text{otherwise} \end{cases}, \tag{5.3}$$

where rnd() is a function that returns a random value between 0 and 1. This way, the overall sample distribution converges to the calculated distribution over time. α_i is a scaling factor that

we set to

$$\alpha_i = NW_i \Big/ \sum_k W_k, \tag{5.4}$$

so that the total number of samples over the whole image is equal to N. Because the number of samples differs per pixel, we need to normalize the output color. We do this by multiplying it by

$$\frac{1}{S_i} \times \begin{cases} (\alpha_i W_i)^{-1}, & \text{if } \alpha_i W_i < 1 \\ 1, & \text{otherwise} \end{cases}. \tag{5.5}$$

An alternative way to normalize the output would be to keep track of the cumulative number of samples per pixel and only normalize the output when the final image is saved, as

$$1 \Big/ F \sum_{f=0}^{F} S_{i,f}, \tag{5.6}$$

where F is the total number of frames.

Chapter 6

Experiment setup

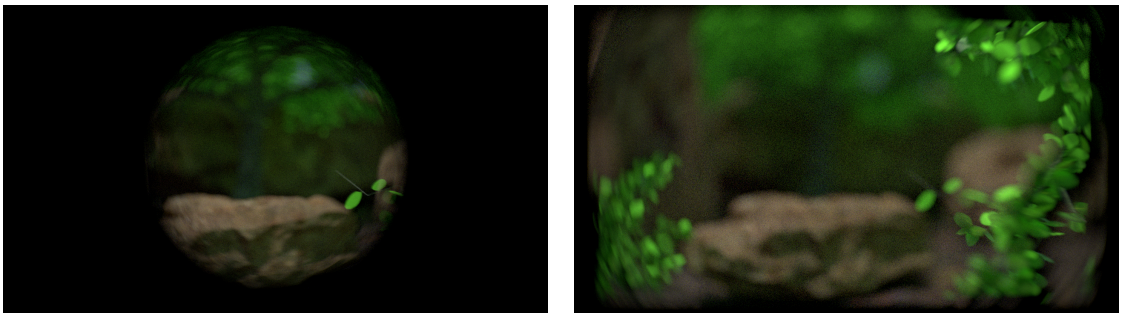
Our goal is to assess the applicability of the Seidel algorithm in artistic DOF simulation. We will do this in three parts:

1. Compare the rendered images qualitatively by visual inspection,
2. Compare the rendered images generated by Seidel, SSRT and the other reference implementations quantitatively using RMSE (root mean square error) and MS-SSIM (multi-scale structural similarity index measure),
3. Examine the variation in the pre-calculated values (Seidel coefficients and lens parameters) and assess the applicability of the implementation as an artistic process.

We will render three test scenes with seven different lens systems, using all five DOF methods (SSRT, Seidel, Distortion + vignetting, Gaussian and Pencil map). Each render contains a total of 10^{10} samples for an image of 1280×720 pixels, which averages out to around $10,850$ samples per pixel. This ensures that the noise is low enough as not to impact the RMSE and MS-SSIM calculations that we will do in any significant way[1].

We chose to use a sensor width of 35mm, which is the most common width of both film and image sensors. With an exception for the Petzval lenses: as these produce a very small image, as can be seen in Figure 6.1, we chose to use a sensor width of 15mm for these lenses.

[1] Because pencil maps are not importance sampled and as 99% of the luminance is stored in the bottom half of the pencil map, around 50% of the pencil map samples sample a black value and therefore do not contribute to the final result, bringing the effective sample count closer to $5,000$ per pixel, increasing the noise as compared to the other methods. Additionally, with the SSRT, Seidel and Distortion + vignetting DOF methods, some samples are lost due to optical vignetting. The percentage of lost samples varies a lot between lenses and increases from around 0% at the center to higher values near the edge of the frame.



(a) 35mm sensor width (b) 15mm sensor width

Figure 6.1: Comparing the image generated when using the Petzval lens with two different sensor widths (35mm and 15mm)

We choose to compare images using both RMSE and MS-SSIM, as each has their own advantage. MS-SSIM is a good method to compare structural similarities between images, and outperforms the simpler RMSE method in this. But since we cannot simply use the available accurate color difference functions – as discussed in see Section 6.1 – for MS-SSIM, we chose to also calculate the RMSE to get more insight in the color similarities between renders. To calculate the MS-SSIM between pairs of images, we used code by *Forst* (2015) and convert the images to grayscale. Note that this implementation downsizes the input images to 340×340 pixels, so shapes smaller than about 2 to 3 pixels may not contribute to the end result.

If our Seidel method is to be useful in an artistic context where the original lens designs are not available, it must be doable to set parameter values by hand. This does not seem feasible if there are seven lens parameters that all depend on the wavelength in different ways, as well as five Seidel coefficients that depend on both the wavelength and the distance to the entrance pupil. As was concluded from the evaluation of the influence on the end result of the lens parameters in Section 5.2, M and $d_{\text{image}\to\text{sensor}}$ have the greatest influence on the final bokeh shape. This allows us to reduce the lens parameters to only three: f, z_{fpp} and $\frac{1}{-M}d_{\text{image}\to\text{sensor}}$. Taking this into consideration, we will evaluate four different simplifications to the parameters that may make this manual tweaking more feasible:

1. Seidel coefficients fixed and all lens parameters fixed (*Fixed Everything*),
2. Seidel coefficients only dependent on wavelength (distance set to 10^3m) and
   (a) No lens parameters fixed (*Fixed Distance*),
   (b) All lens parameters fixed, but with wavelength-dependent values for f, z_{fpp} and $\frac{1}{-M}d_{\text{image}\to\text{sensor}}$ only (*Fixed Most*),
   (c) All lens parameters fixed (*Fixed Parameters*).

We will use these simplifications to render the shield scene using all seven lenses. We will then compare these renders using RMSE and MS-SSIM, as well as by visual inspection.

6.1 Color spaces

As spectral effects on aberrations are a large part of this research, it is essential that we choose accurate methods to generate and evaluate colors. For this reason, we use the CIE 1931 XYZ color space (*Smith and Guild*, 1931), which is an additive color space defined to match human perception as closely as possible. Every sample that is taken is first given a random wavelength

between 360 and 830 nm, which is then converted to an XYZ color using a table of CIE 1931 XYZ color matching function values[2], using linear interpolation between the two nearest values.

To calculate the error between two colors, the CIEDE2000 color difference function is used[3], which is designed to be as perceptually-uniform as possible, meaning that the visual difference between two colors with a certain color difference value will be the same no matter where on the spectrum the colors reside. In order to use the CIEDE2000 function, the CIEXYZ colors are first converted into the CIELAB color space using the D65 illuminant.

Additionally, we choose to use a grayscale MS-SSIM implementation, so the color images needed to be converted to grayscale. For this, we first calculated the luminance as

$$\text{lum}_\alpha = 0.27r + 0.67g + 0.06b. \tag{6.1}$$

But to make sure that the luminance cannot exceed 1, as we need to store it in an 8-bit PNG image, we set

$$\text{lum}_\beta = 1 - 2^{-\text{lum}_\alpha}. \tag{6.2}$$

This makes sure that the luminance never exceeds 1, even when lum_α does. This way, any detail in high-luminance areas is retained when saved as an 8-bit PNG image. We did some testing to see how MS-SSIM values are impacted when using lum_β instead of lum_α. We filled a window of 10×10 pixels with random noise, and calculated the SSIM value of this window compared to iteratively more randomly offset versions, using both lum_α and lum_β. The results of a few passes of this are plotted in Figure 6.2.

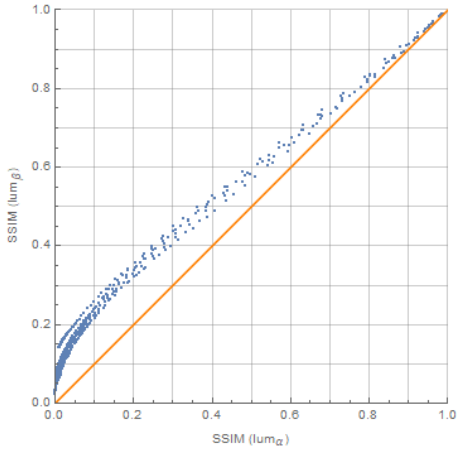


Figure 6.2: SSIM calculations, comparing lum_α and lum_β by calculating the SSIM using both methods on the same 10×10 pixel input window.

The SSIM values are different between the methods, but the ordering is preserved, which allows us to safely compare images using lum_β.

[2]The table can be found at `https://github.com/mocabe/CIE-1931-XYZ-Color-Space-Standard-Observer-Color-Matching-Functions`.

[3]The implementation of the CIEDE2000 color difference function that was used can be found on `https://github.com/gfiumara/CIEDE2000`.

Finally, we convert lum_β into an 8-bit sRGB value by applying a gamma correction[4]. A resulting image can be seen in Figure 6.3.

(a) Color (CIEXYZ) (b) Black and white luminance

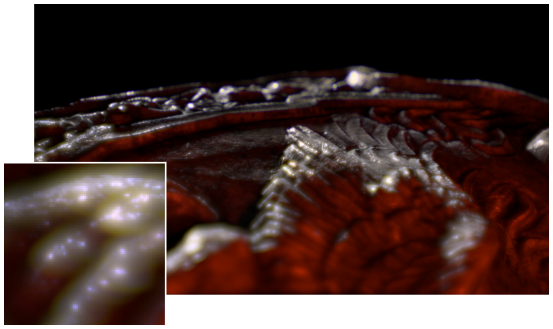
Figure 6.3: A rendered image and its luminance black and white version.

6.2 Lens choices

Seven lenses were selected to compare, all of which are very different in design and thus in the image they produce. All lens designs were obtained in ZEMAX file format from `http://www.lens-designs.com/`. Below, each selected lens is listed, together with a schematic cross-section and an SSRT render of one of the test scenes. A characteristic part of the render is enlarged to illustrate the lens's unique characteristics.

1. 50mm f/4.5 Double Gauss design from 1897[5] (Figure 6.4). A simple, primitive lens design that produces strong aberrations.

(a) Schematic cross-section



(b) Rendered image

Figure 6.4: Double Gauss lens

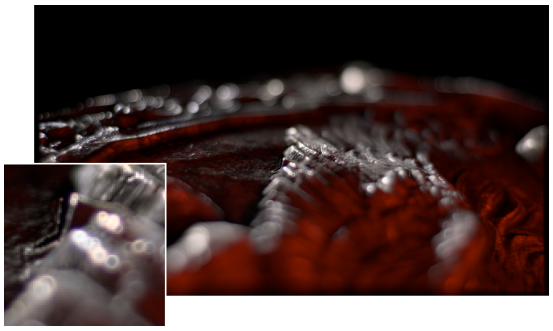
2. Nikkor 50mm f/1.8 AI-s[6] (Figure 6.5). This prime lens was introduced in 1980 and is still considered a very good, sharp lens. (Note: Nikkor is the name used for lenses produced by Nikon.)

[4]The formula used to apply the gamma correction can be found at `https://en.wikipedia.org/wiki/SRGB` under *Specification of the transformation.*

[5]Based on U.S. Patent 583336, figure 3

[6]More information on the Nikkor 50mm f/1.8 AI-s is available at `https://web.archive.org/web/20170111074403/http://www.nikkor.com:80/story/0060`

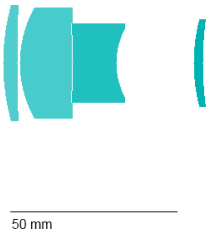
(a) Schematic cross-section



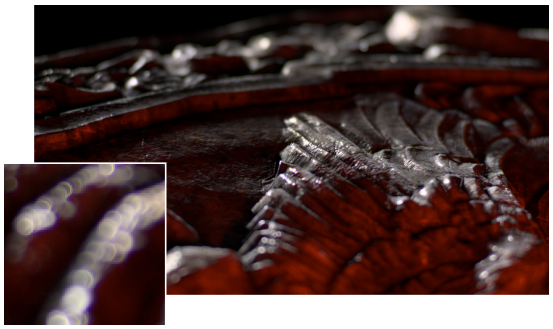
(b) Rendered image

Figure 6.5: Nikkor 50mm lens

3. Nikkor 135mm f/4 (originally known as NIKKOR-QC 13.5cm f/4)[7] (Figure 6.6). This lens was first produced in 1946 and was one of the first lenses produced by Nikon. The lens produces some clearly visible aberrations, but no major ones.



(a) Schematic cross-section



(b) Rendered image

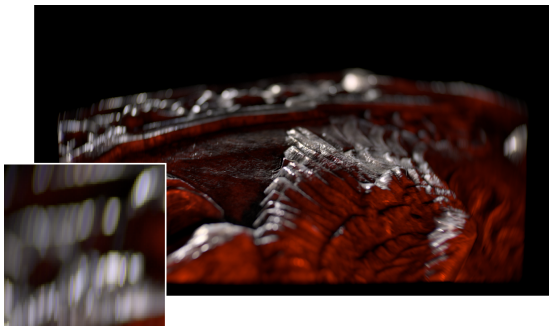
Figure 6.6: Nikkor 135mm lens

4. A 25mm f/2.2 Petzval lens[8] (Figure 6.7). This type of lens was popular in the 19th century, and is still popular for the extreme 'swirley bokeh' effect, caused by a very strong Petzval aberration. This particular design stems from 1937.

[7] More about the NIKKOR-QC 13.5cm f/4 can be read at `https://web.archive.org/web/20160317213020/http://nikkor.com/story/0043/`

[8] Based on U.S. Patent 2158202

(a) Schematic cross-section

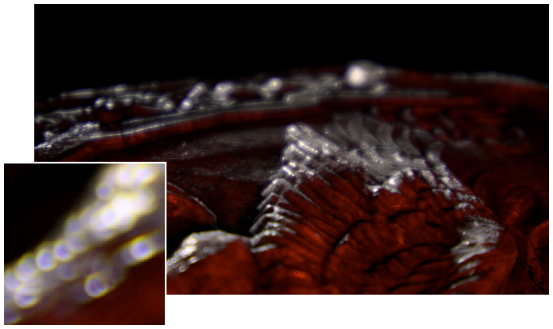


(b) Rendered image

Figure 6.7: Petzval 25mm lens

5. Nikon Pikaichi 35mm f/2.8[9] (Figure 6.8). This lens was made for cheap compact cameras and was produced starting in 1982. The lens has a low optical quality and produces many aberrations.

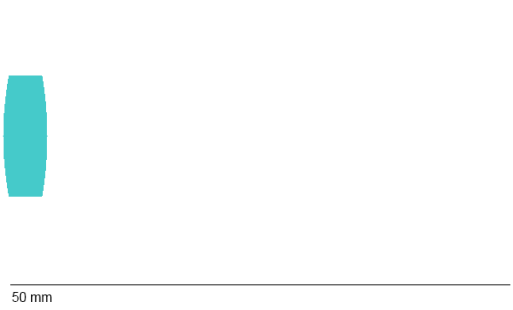
(a) Schematic cross-section



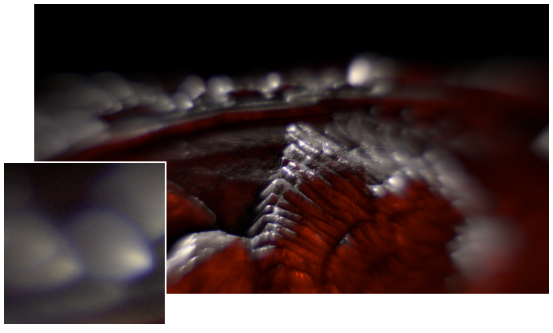
(b) Rendered image

Figure 6.8: Pikaichi 35mm lens

6. A single lens (Figure 6.9), producing large aberrations. This one has a focal length of around 34mm and an f-stop of around f/3.



(a) Schematic cross-section



(b) Rendered image

Figure 6.9: "Single" 34mm lens

[9]More about the Nikon Pikaichi 35mm f/2.8 at `https://web.archive.org/web/20160627194908/http://nikkor.com/story/0033/`

7. A 243mm f/4.1 zoom lens (Figure 6.10). From searching online this one is probably modelled after the Pentax 60-250mm f/4 lens. This is a modern design; the Pentax lens was introduced in 2007 and is still sold today. It produces very sharp images with little aberrations, though some are still visible.

(a) Schematic cross-section

(b) Rendered image

Figure 6.10: "Zoom" 243mm lens

6.3 Test images

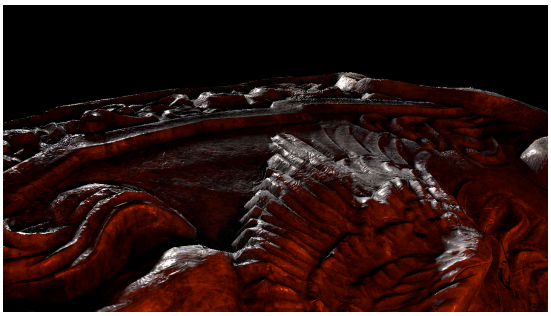
Three test images were produced, each with their own characteristics to test the applicability of our implementation in different scenarios. We have the *forest* scene which has a relatively constant illumination and no strong highlights, but does have much detail in the leaves of the trees near the edge of the image. The *shield* scene does have many strong white highlights in the reflections of the metal shield, both in the foreground and background, which should produce clear bokeh shapes. Finally, the *skyline* scene also has many strong highlights, but they are colored in many different hues.

The images were rendered with Cinema 4D using a pinhole camera model and with all anti-aliasing disabled in order to generate a valid surface color and depth value for each pixel. For calculating the depth values, the distance from the camera point to the pixel was used, ignoring any refractions or reflections.

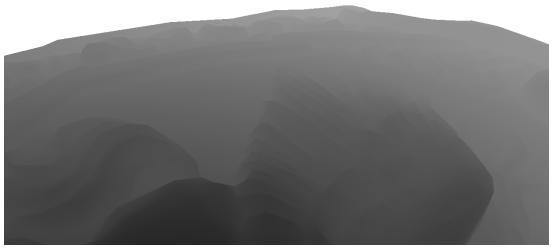
For the Nikkor 135mm and Zoom lenses, a modified 'tele' version of each image was produced, because the circles of confusion would otherwise become too large to work with. For the *forest* and *skyline* scenes this was done by merely multiplying the depth map by 3, while a separate image rendered using a tele lens model was created for the *shield* scene. The images are shown in Figure 6.11.

(a) Forest scene RGB

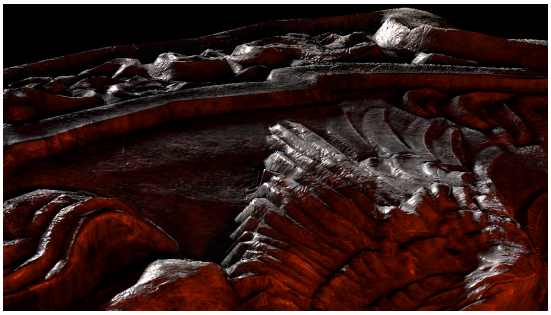
(b) Forest scene depth



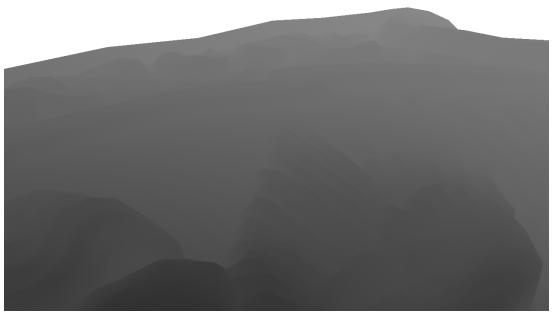
(c) Shield scene RGB



(d) Shield scene depth

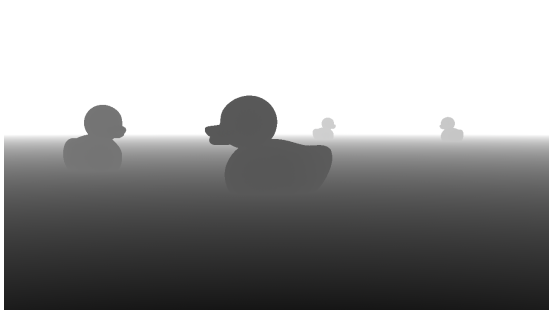


(e) Shield tele scene RGB



(f) Shield tele scene depth

(g) Skyline scene RGB



(h) Skyline scene depth

Figure 6.11: The test images together with their depth maps. The depth maps are processed for visibility.

Chapter 7

Results

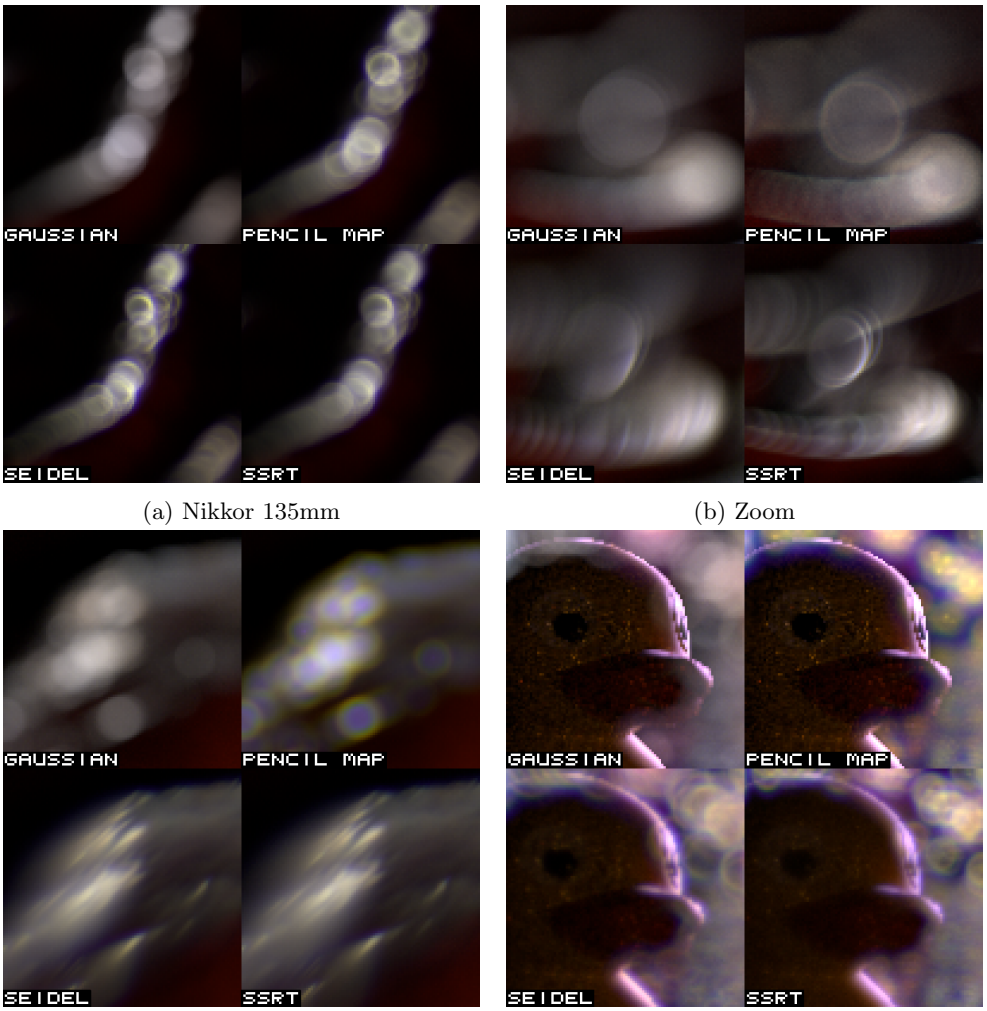
In this chapter, we present both our qualitative and quantitative results and give each some explanation and interpretation.

7.1 Qualitative analysis

In Figure 7.1, cropped regions of renders using some lenses are shown, using all four methods[1]. In every render, Seidel DOF approaches the reference image best. Especially so for the Single and Double Gauss lenses (Figures 7.1c and 7.1d, respectively), which exhibit the strongest aberrations. Seidel DOF accurately captures the slight softening effects visible with the Double Gauss lens and the streaking blurring generated by the Single lens, while both Gaussian DOF and Pencil map DOF do not.

Generally, the more the bokeh produced by a lens resembles a uniform disk, the smaller the differences between Seidel, Gaussian and Pencil map DOF get – as is expected, as uniform bokeh means that the aberrations added in the Seidel simulation are small and that the Pencil map is fairly uniform. But even in those cases, such as with the Nikkor 135mm (Figure 7.1a) and Zoom (Figure 7.1b) lenses, Seidel improves over those methods by simulating non-uniform effects like the bright edge at the right side of the bokeh generated using the Zoom lens.

[1] We do not include *Distortion + vignetting* here, as it would mostly resemble Seidel, but without spectral effects.



(a) Nikkor 135mm

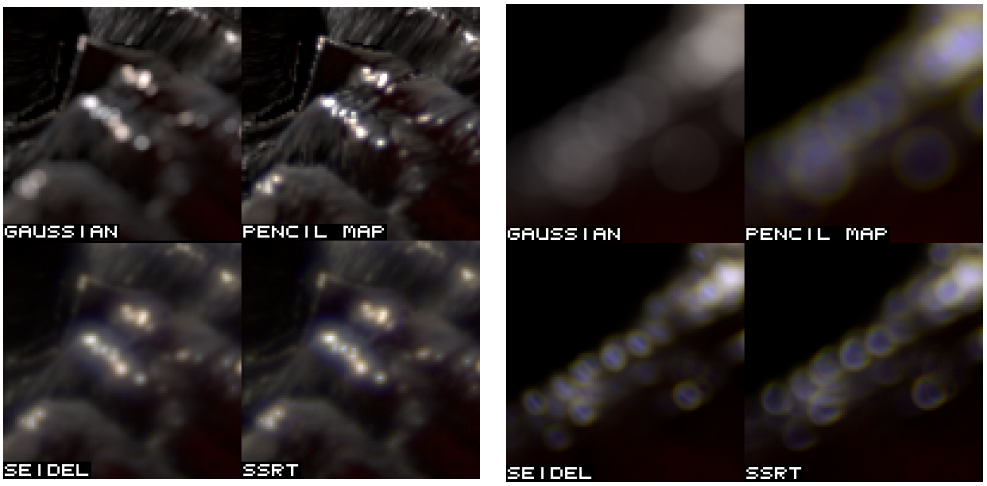
(b) Zoom

(c) Single

(d) Double Gauss

Figure 7.1: Cropped regions rendered using various lenses.

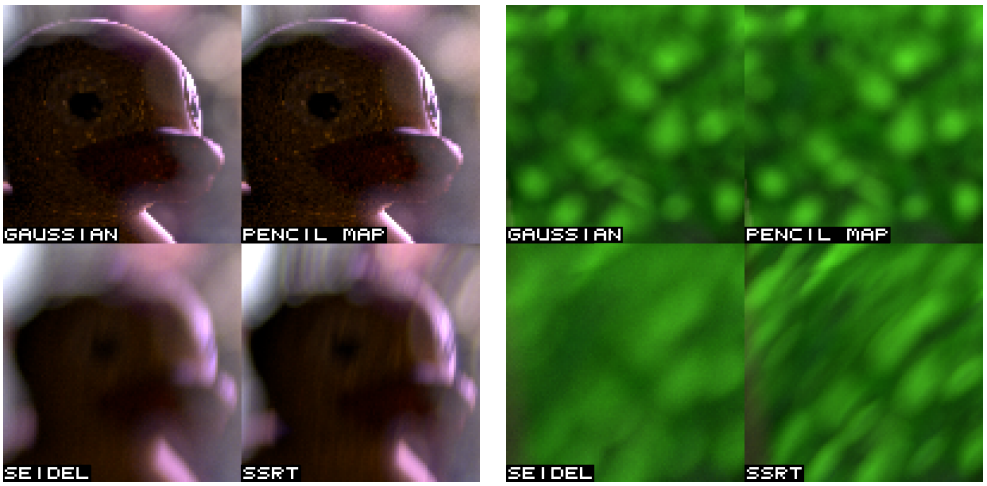
There are some visible dissimilarities between Seidel and SSRT DOF. The bokeh size is not always correct, such as in Figures 7.1b and 7.1d. In both cases, the Seidel bokeh is clearly too big when compared to SSRT. In Figure 7.2, two regions rendered using the Pikaichi 35mm lens are shown, one close to the optical axis and one further from the optical axis. Close to the optical axis (Figure 7.2a), Seidel DOF resembles SSRT DOF very well, and no large dissimilarities catch the eye. But further from the optical axis (Figure 7.2b), some effects are obviously missing from the Seidel simulation. These are most likely produced by higher order effects and not captured by the Seidel aberrations. Note, though, that Seidel DOF approaches the reference much better than both Gaussian and Pencil map DOF. The differences between the methods are similar when using the Petzval lens, as illustrated in Figure 7.3. Seidel DOF captures the swirly bokeh effect better than Gaussian and Pencil map DOF, though there are disparities, in this case likely caused by the simplified optical vignetting implementation, as this effect was determined to be particularly strong with the Petzval lens (Figure 6.1).



(a) Close to the optical axis

(b) Further from the optical axis

Figure 7.2: Two regions of the shield scene, rendered using the Pikaichi 35mm lens.

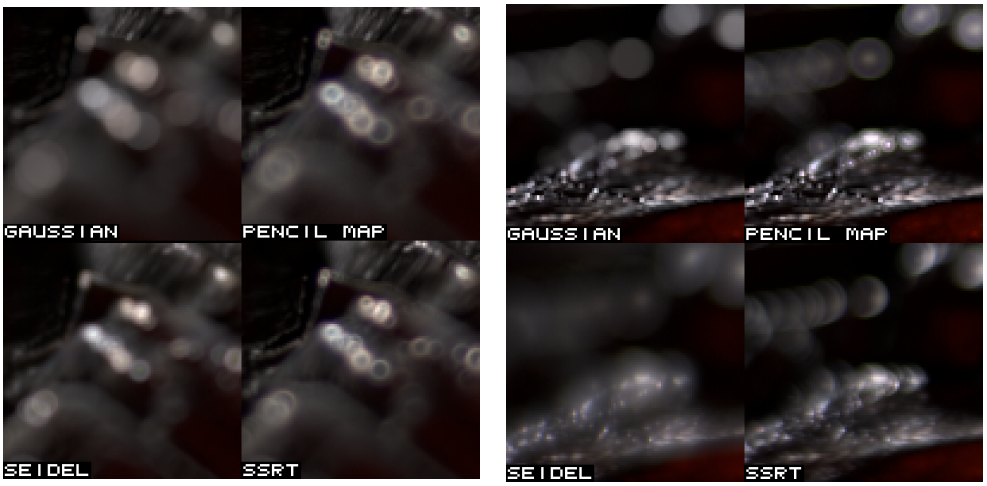


(a) Skyline scene

(b) Forest scene

Figure 7.3: Two renders made using the Petzval lens.

Figure 7.4 shows two regions of the shield scene rendered using the Nikkor 50mm lens, and illustrates the limitations of both Pencil map DOF and Seidel DOF. Visually, Pencil map DOF approximates the SSRT reference very well close to the optical axis, but further away it becomes very obvious that radial aberrations are not included in this simulation. And while Seidel DOF does include these aberrations, it is visually significantly different from the SSRT reference. This could again be a limitation of using only the five Seidel aberrations, as the bokeh looks off both near and far from the optical axis.



(a) Close to the optical axis

(b) Further from the optical axis

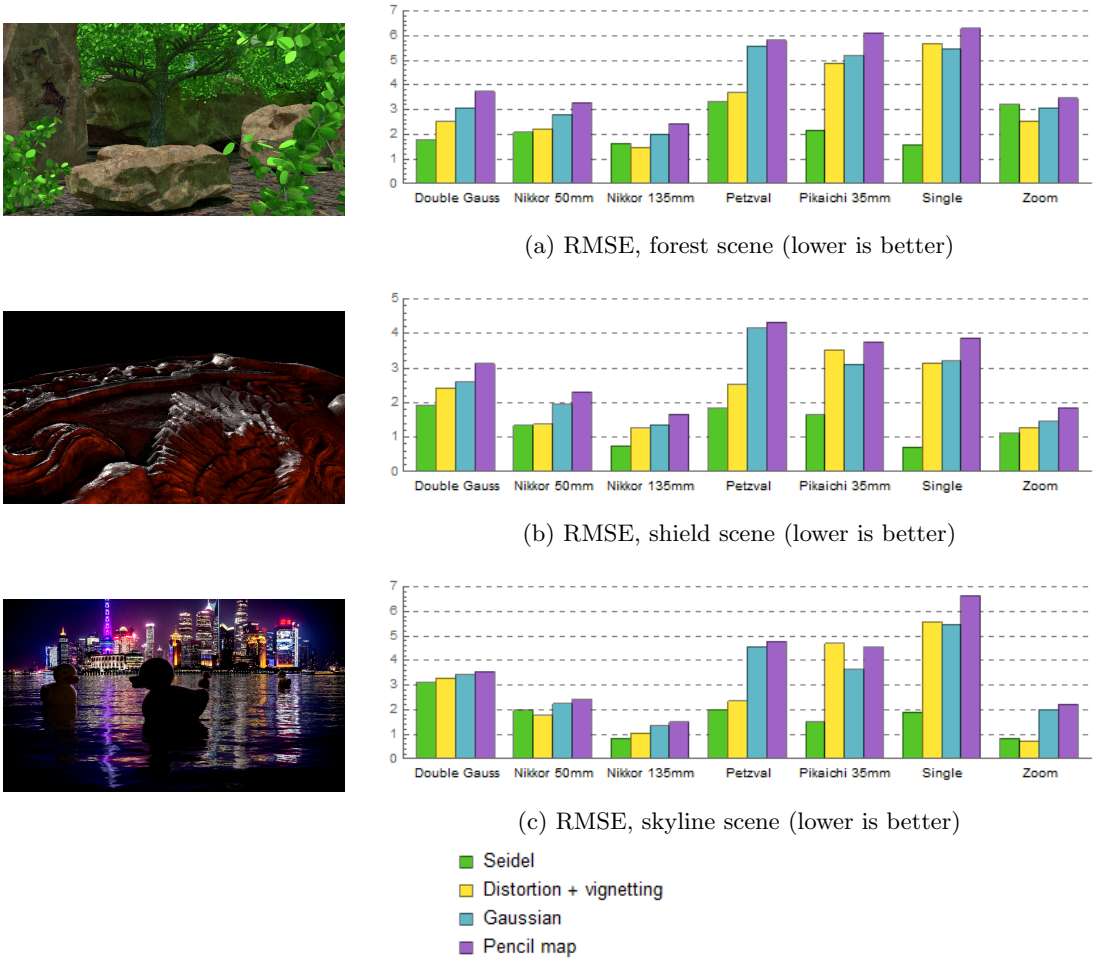
Figure 7.4: Two regions of the shield scene, rendered using the Nikkor 50mm lens.

7.1.1 Discussion

The Seidel DOF method approaches the reference method better than Gaussian and Pencil map DOF. This was expected, as adding Seidel aberrations brings us closer to reality. Though the Seidel DOF implementation is successful with most of the tested lens designs, it was not so universally. The dissimilarities are most severe with the Nikkor 50mm lens, where Seidel DOF is not able to capture the most significant aberrations, as visible in Figure 7.4. This may be due to a lack of higher order aberrations. Our simplified optical vignetting implementation is probably to blame for the dissimilarities between Seidel and SSRT DOF with the Petzval lens, as visible in Figure 7.3.

7.2 Quantitative comparison

For each scene and lens combination, the Seidel, Distortion + vignetting, Gaussian and Pencil map DOF renders were compared to the reference SSRT renders. The results are plotted in Figures 7.5 and 7.6, which show the RMSE and MS-SSIM values of these comparisons, respectively. Note: all renders that used the Petzval lens used a smaller sensor width (15mm instead of 35mm) as this lens produces a very small image, as discussed in Chapter 6. But it turned out this was not enough: there were still small black edges present in SSRT, Seidel and Distortion + vignetting DOF images, but not in Gaussian or Pencil map images. As comparing these images would increase the RMSE and MS-SSIM values significantly, we chose to crop the Petzval renders to 90% size in each dimension (from 1280×720 to 1152×648 pixels).

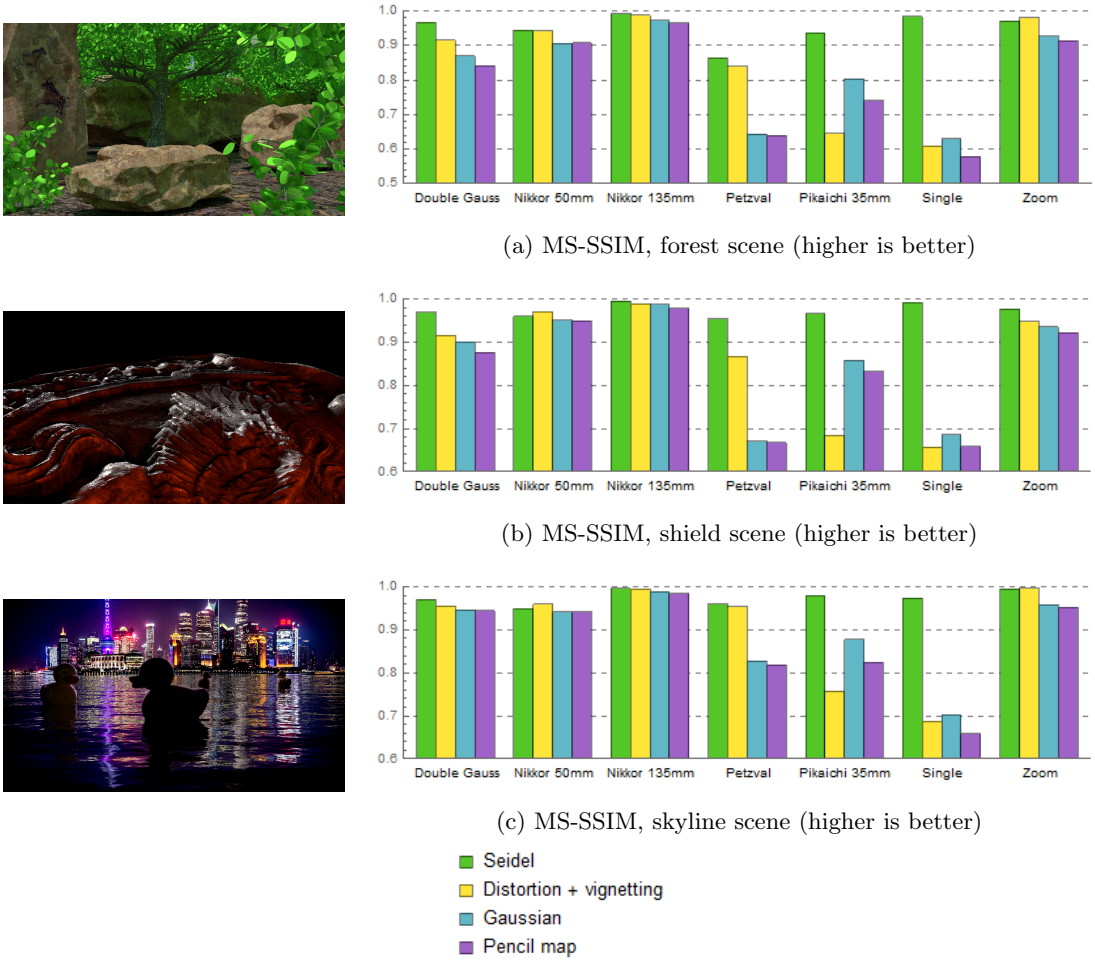


(a) RMSE, forest scene (lower is better)

(b) RMSE, shield scene (lower is better)

(c) RMSE, skyline scene (lower is better)

Figure 7.5: RMSE comparison of the four depth of field simulation methods against the SSRT reference method, using seven different lenses and three different scenes.



(a) MS-SSIM, forest scene (higher is better)

(b) MS-SSIM, shield scene (higher is better)

(c) MS-SSIM, skyline scene (higher is better)

Figure 7.6: MS-SSIM comparison of the four depth of field simulation methods against the SSRT reference method, using seven different lenses and three different scenes.

Evidently, Seidel DOF performs best in virtually all cases. There is a handful of cases where Distortion + vignetting DOF outperforms Seidel DOF, mostly with the Nikkor 50mm and Zoom lenses, but also once with the Nikkor 135mm lens (RMSE, forest scene). When using the Zoom lens on the forest scene, Seidel actually finished third in terms of RMSE: even Gaussian DOF achieves a result that is more similar to the reference image – though all RMSE values here are fairly similar.

In all cases where Distortion + vignetting DOF outperforms Seidel DOF, the differences are relatively small. The RMSE values are between 9.8% and 21.1% lower, and the MS-SSIM values are between 0.2% and 1.1% higher. Overall, Seidel DOF approaches the reference images more consistently than the other methods, as shown by the mean values and standard deviations of the RMSE and MS-SSIM values, which are plotted in Figure 7.7.

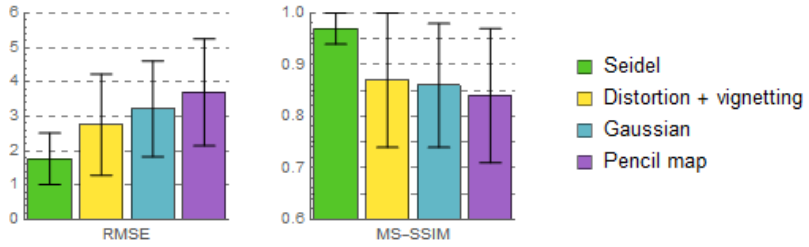
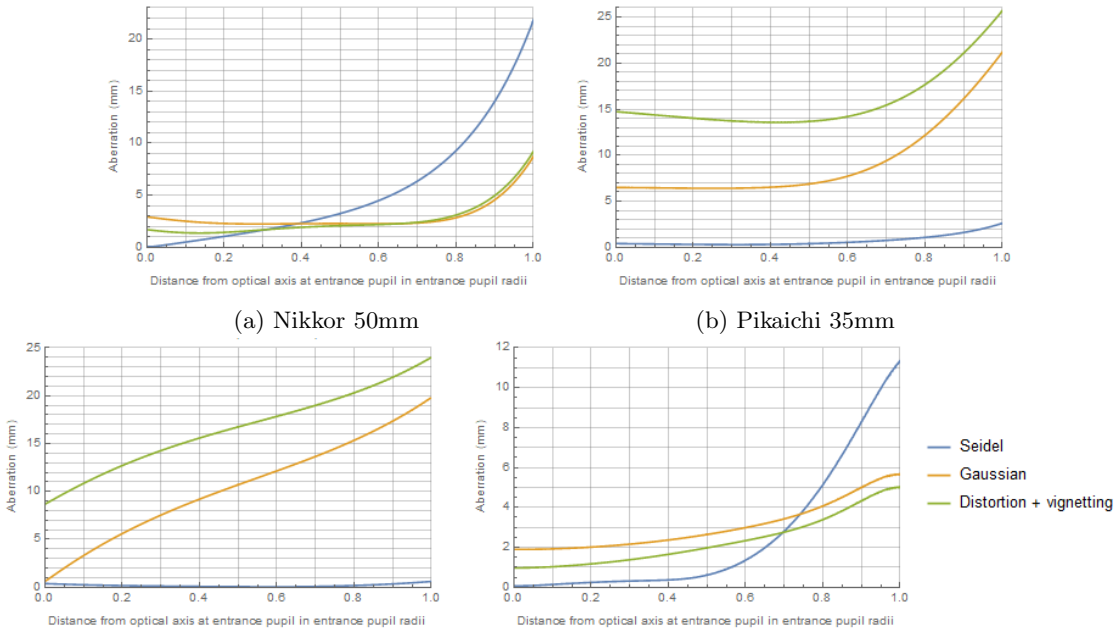


Figure 7.7: Mean values and standard deviations of the RMSE (lower is better) and MS-SSIM (higher is better) values of all four methods.

7.2.1 Aberration distances

A light ray with a wavelength of 550 nm originating from a distance of 2 meters was traced through four lenses using both SSRT (reference) and the three other methods (Seidel, Gaussian and Distortion + vignetting), at different distances to the optical axis. The aberration distances – the distances between the sensor coordinates of the reference method and of the other methods – were then computed, and are plotted in Figure 7.8. The lens is focused at 0.88 meters, so this light source would produce back bokeh.



(a) Nikkor 50mm

(b) Pikaichi 35mm

(c) Single

(d) Zoom

Figure 7.8: The aberration distances for four lenses, using a light source of 550 nm at 2 meter distance, while the lens is focused at 0.88 meters away. The aberration distance is the distance between the sensor coordinates of the currently used method and the reference method (SSRT).

Using Seidel DOF, the aberration is always (nearly) 0 at the optical axis, and then starts to grow. Note that slight discrepancies may be introduced here by the different ways of calculating the sensor position for each method, as described in Section 5.3. With Gaussian DOF, the aberration

is more constant, while also generally growing when moving away from the optical axis. Distortion + vignetting DOF behaves more like Gaussian DOF than Seidel DOF, and sometimes has larger aberrations than either. With the Single and Pikaichi 35mm lenses, Seidel performs really well, with very low aberrations overall. With the Nikkor 50mm and Zoom lenses, it starts out performing really well near the optical axis, but quickly diverges, making the other two methods more accurate in terms of aberrations.

7.2.2 Runtime

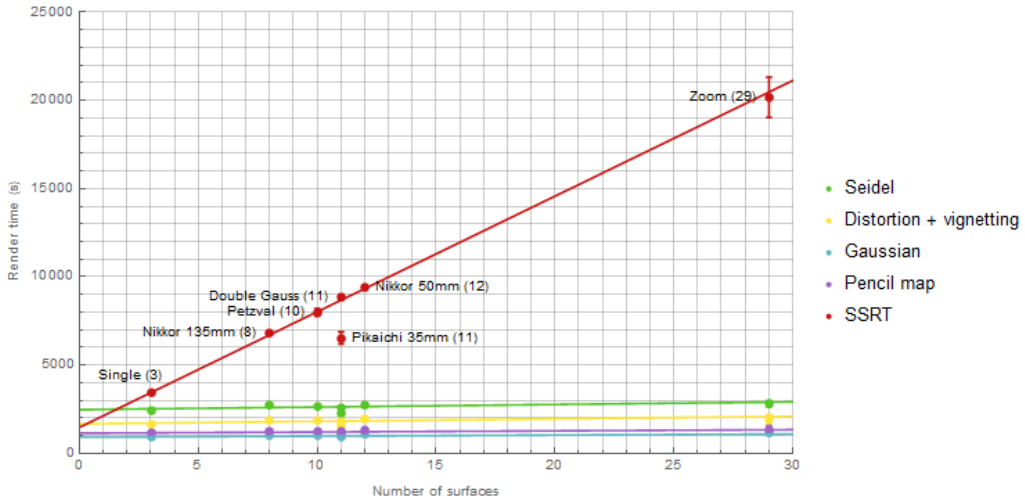


Figure 7.9: Render times in seconds for the different lenses, plotted against the number of lens elements, and with linear relations fitted to the data.

The time to render an image using each of the five methods is plotted in Figure 7.9. This includes both the initialization and simulation steps, but the render times can be reasonably interpreted as the simulation time, as the time the initialization step takes up is negligible – it is only executed once and is completed in under a second, while the whole render times range between 4,400 and 20,000 seconds.

As each DOF method was used to render three separate images, the mean render times were used, together with error bars showing the standard deviations. To the data of each DOF method, a linear relation was fitted using σ^{-2} as weights (where σ is the standard deviation). With SSRT DOF, the render time seems to increase linearly with the number of surfaces in the lens, while the render time is approximately constant for all other methods. The only data points that do not seem to adhere to the linear relations are those from SSRT, Seidel and Distortion + vignetting DOF using the Pikaichi 35mm lens. This can be explained by the high number of samples that do not pass through the lens, and therefore exit the simulations early. The percentage of passing samples when using SSRT DOF is approximately 69% for the Pikaichi lens, while for other lenses it lies between 86% and 100% ($93\% \pm 6\%$).

As the graph in Figure 7.9 confirms, the SSRT algorithm runs in $O(n)$ time, where n is the number of lens surfaces. This can be explained by the fact that each ray has to intersect every surface when ray tracing the lens. The other algorithms run in $O(1)$ time, due to all coefficients being calculated at the initialization step.

7.2.3 Discussion

The quantitative results show that Seidel DOF approaches the reference method better than Gaussian and Pencil map DOF, just as with the qualitative results. What we did not expect was for Gaussian DOF to outperform Pencil map DOF, as the latter should add at least some aberrations to the former. This may be because of the pencil map normalization that we used – the circle of confusion size was dictated by Gaussian optics and not by the pencil map. This could be improved by e.g. creating a formula to calculate the circle of confusion, based on the lens system. Such modifications could make Pencil map DOF a better competitor to Seidel DOF and make the comparisons a bit more fair.

Calculating the imaging sensor positions differently for each method could have impacted the results more than anticipated. This decision was made based on the look of the final image rather than the math behind them.

7.3 Variation in coefficients and parameters

7.3.1 Seidel coefficients

Besides being a function of the wavelength, the Seidel coefficients are also a function of the distance to the entrance pupil. In Figures 7.10 and 7.11, the Seidel coefficients of the Single and Zoom lenses, respectively, are plotted against the distance to the entrance pupil. The dependence on wavelength is taken into account by rendering the coefficients for many different wavelengths using the CIEXYZ color associated with those wavelengths.

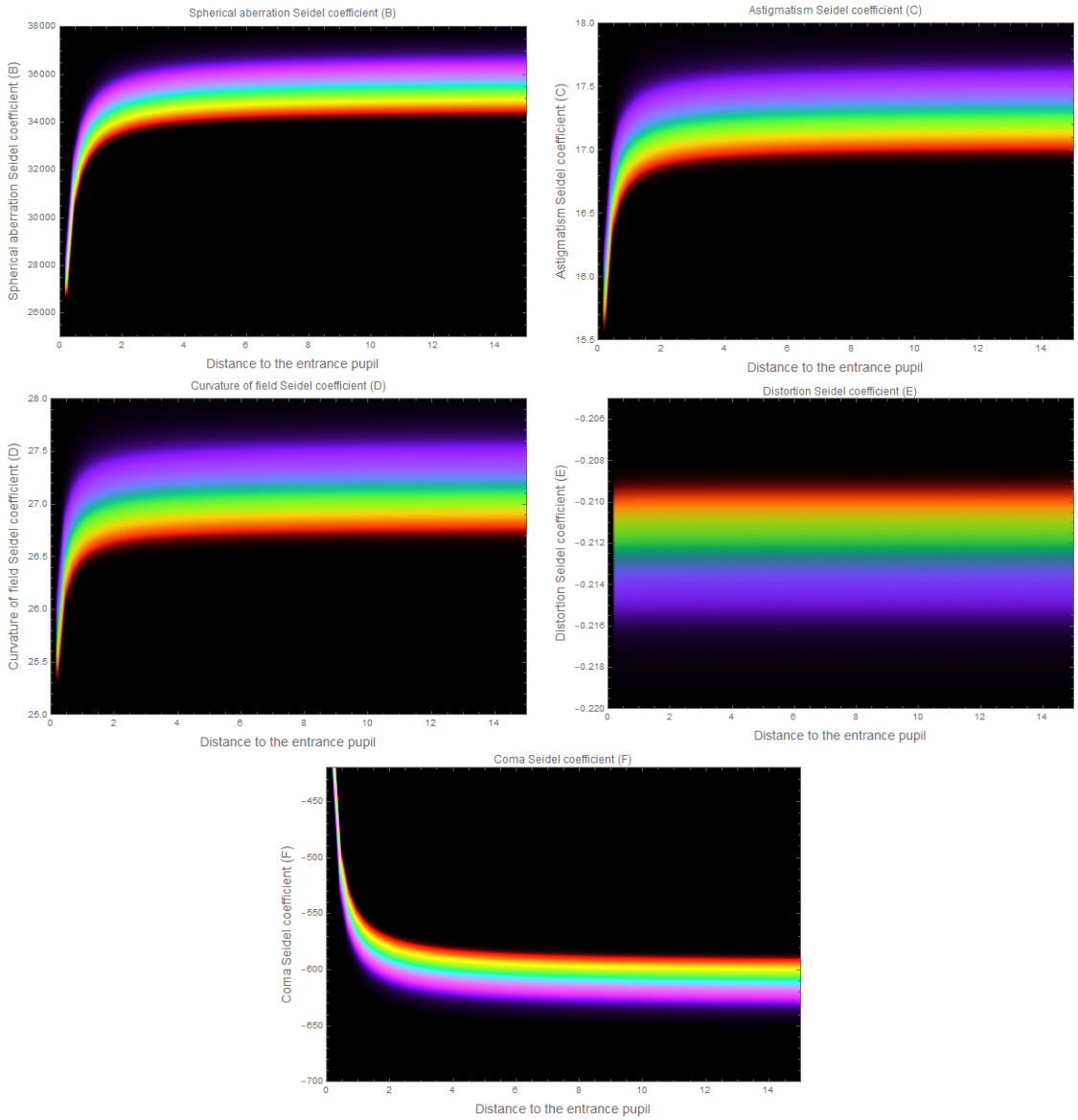


Figure 7.10: Plots of the five Seidel coefficients using the Single lens. As the coefficients are a function of both the distance to the lens and the wavelength of the light, we chose to render the graphs using the CIEXYZ colors associated with each wavelength and sampling many wavelengths for each distance.

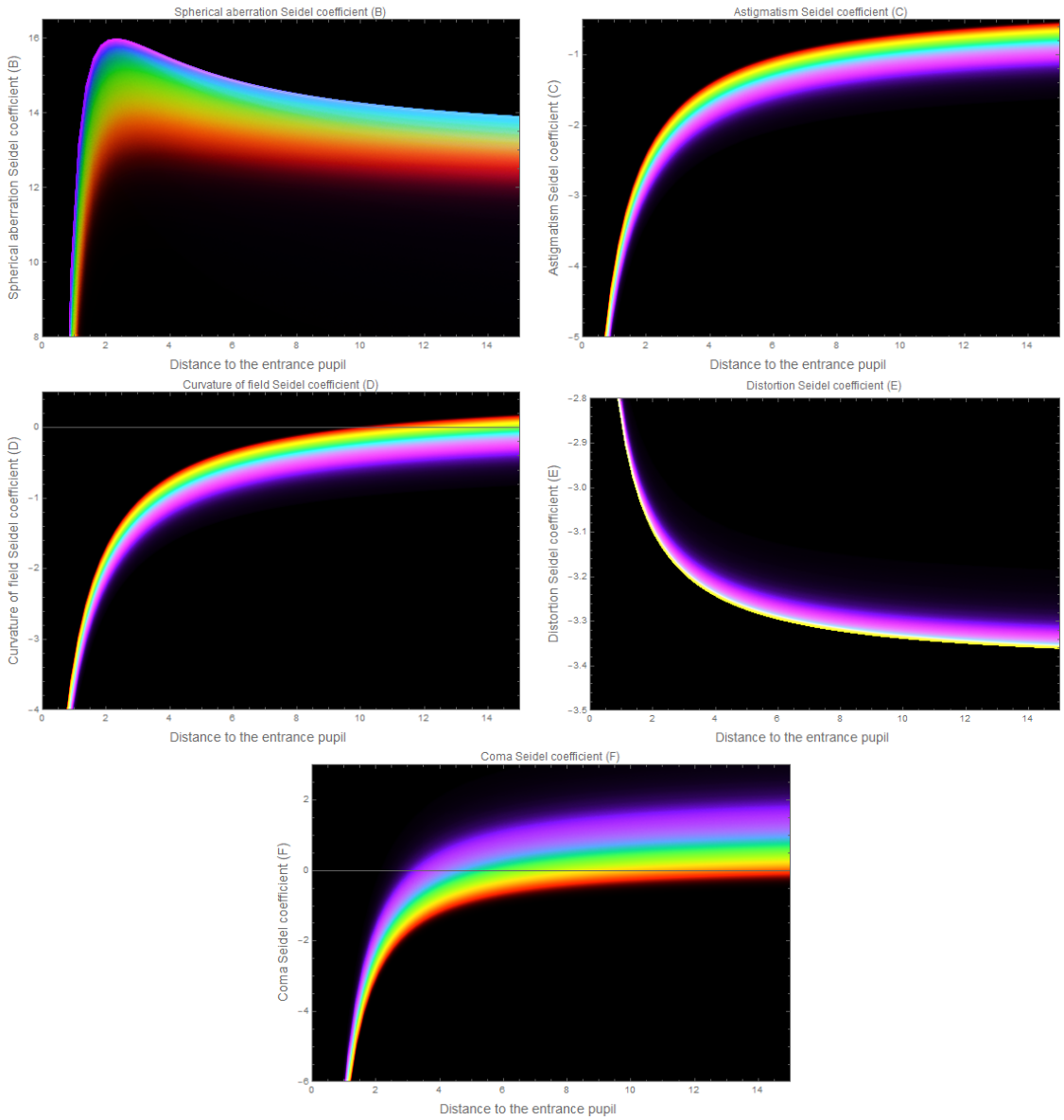
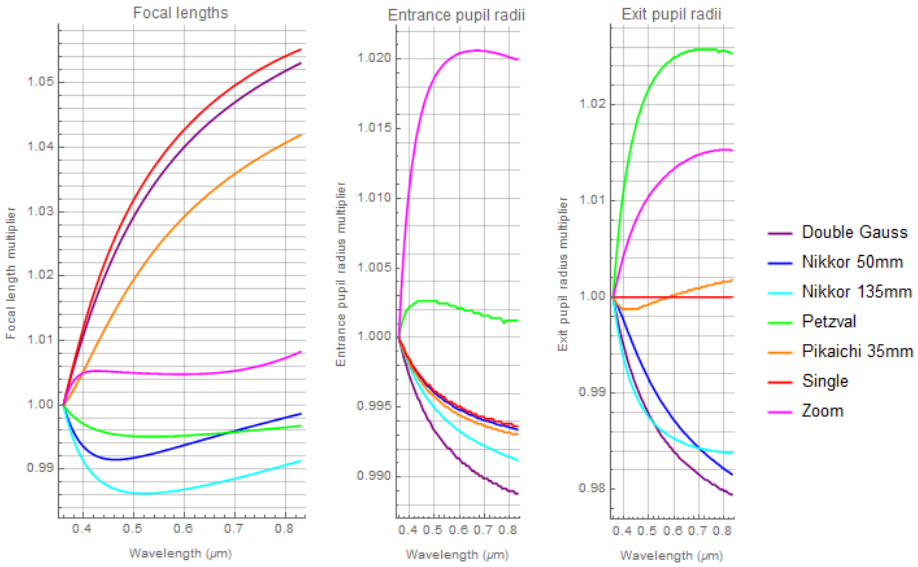


Figure 7.11: Plots of the five Seidel coefficients using the Zoom lens. As the coefficients are a function of both the distance to the lens and the wavelength of the light, we chose to render the graphs using the CIEXYZ colors associated with each wavelength and sampling many wavelengths for each distance.

For both lenses, the distance dependence decreases as the distance becomes greater, which happens more quickly with the Single lens than with the Zoom lens. The wavelength dependence is similar across the whole distance range, but does vary a lot between the two lenses and between the different coefficients.

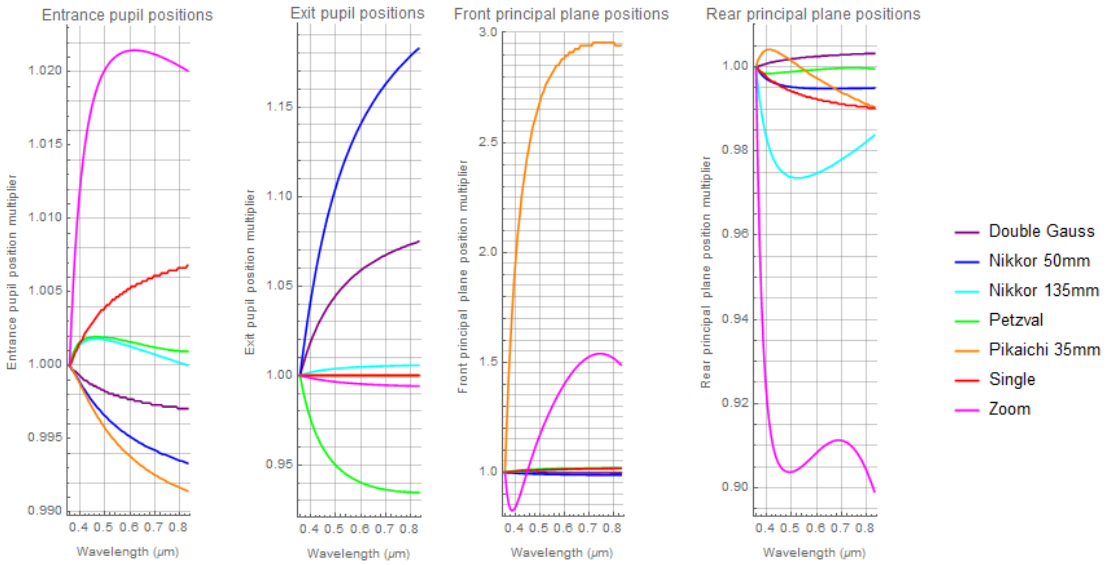
7.3.2 Lens parameters

In Figure 7.12 and 7.13, the lens parameters are plotted against the light wavelength. Each parameter is divided by its initial value (at $\lambda = 0.360$nm), in order to show the different growth rates between the various lenses. Some parameters vary a lot, while others seem to stay within a few percent of the initial value throughout the visible spectrum. The largest difference can be seen in the front principal plane position, which increases to almost three times the initial value with the Pikaichi lens. Together with the focal length, this impacts the lateral magnification (as discussed in Section 5.2.1).



(a) Focal lengths (b) Entrance pupil radii (c) Exit pupil radii

Figure 7.12: Focal lengths, and entrance and exit pupil radii, divided by the initial value (at $\lambda = 360$ nm).



(a) Entrance pupil plane positions

(b) Exit pupil plane positions

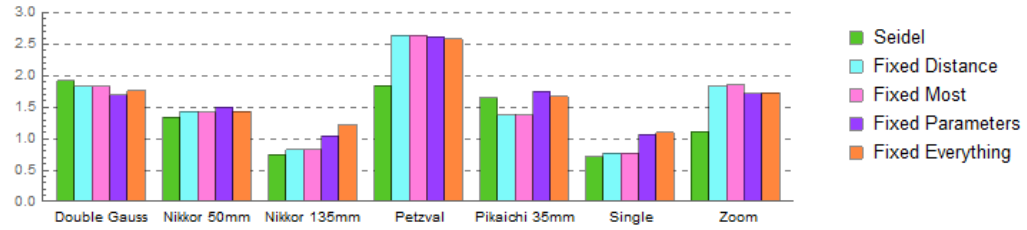
(c) Front principal plane positions

(d) Rear principal plane positions

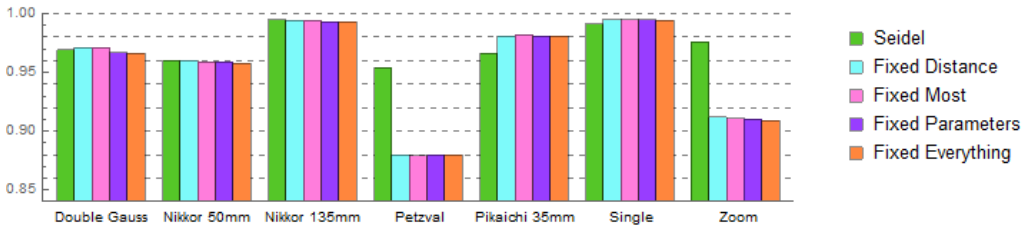
Figure 7.13: Entrance and exit pupil plane positions, and front and rear principal plane positions, divided by the initial value (at $\lambda = 360$ nm).

7.3.3 Simplified parameters

In Figure 7.14, the RMSE and MS-SSIM values are calculated between the SSRT reference renders and the renders made using Seidel with the parameter simplifications as discussed in Chapter 6. Also included are the results of the full Seidel implementation, as comparison. In Figure 7.15, the values of $\frac{1}{-M} d_{\text{image}\to\text{sensor}}$ are plotted. Interestingly, the curves seem to generally resemble those of f, which are plotted in Figure 7.12a, even though f itself is not used in the calculation of $\frac{1}{-M} d_{\text{image}\to\text{sensor}}$.



(a) RMSE, shield scene (lower is better)



(b) MS-SSIM, shield scene (higher is better)

Figure 7.14: RMSE and MS-SSIM comparison of the various Seidel simulation methods, using seven different lenses and the shield scene.

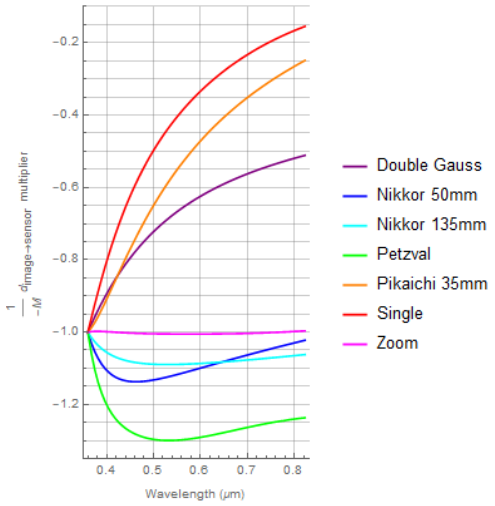
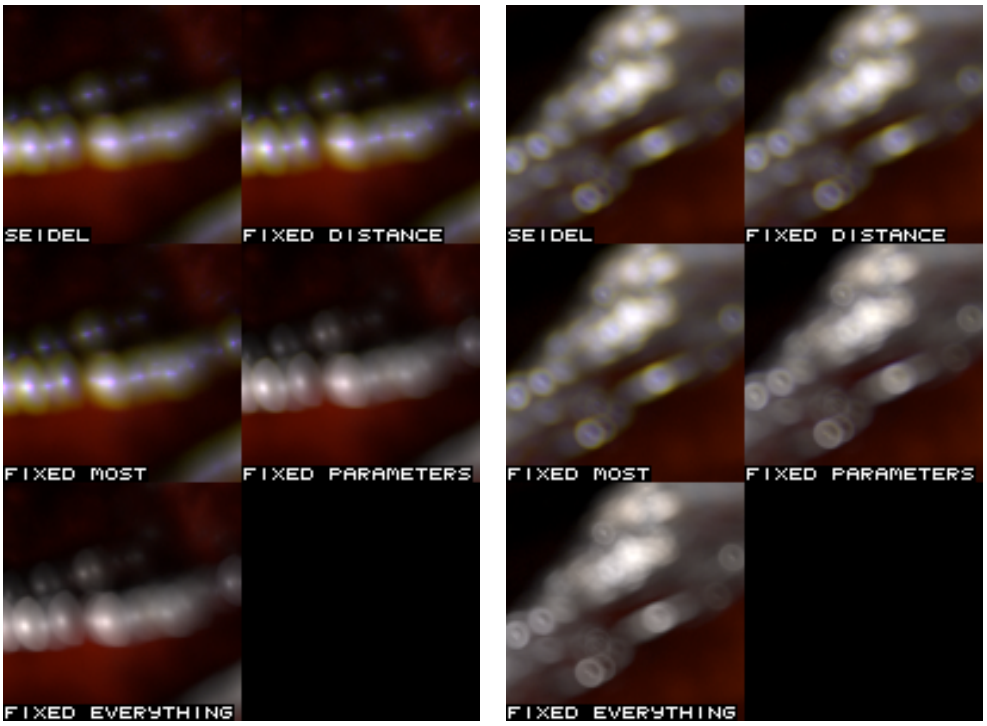


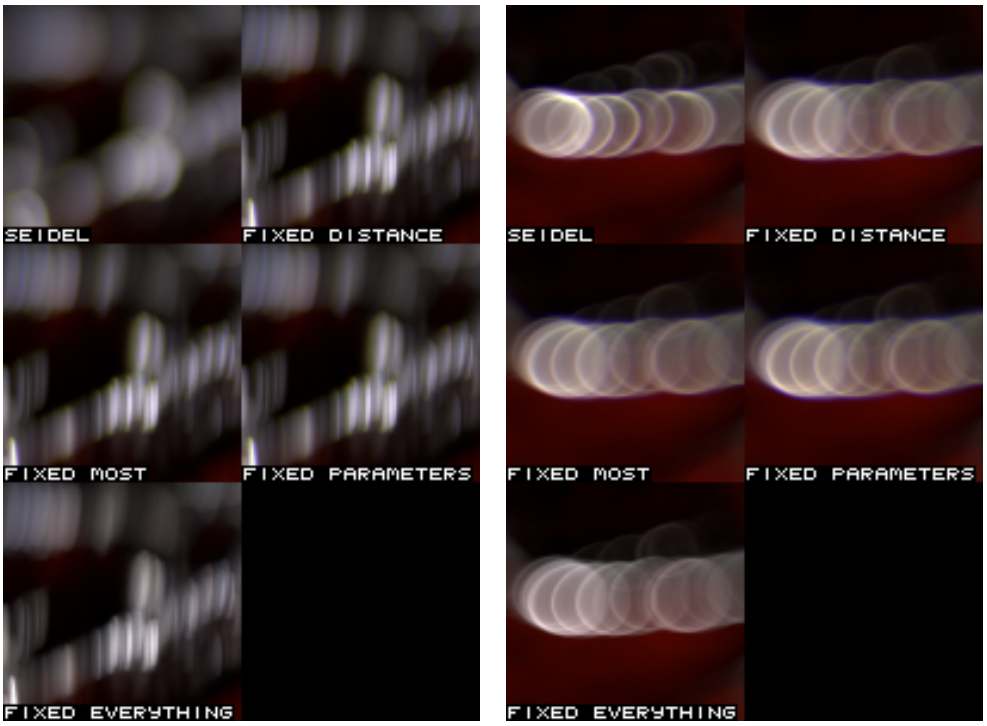
Figure 7.15: Values of $\frac{1}{-M} d_{\text{image}\to\text{sensor}}$ for all lenses.

The rate of success of the simplifications differs per lens, but the results are fairly similar overall. The Seidel implementation mostly outperforms the simplified implementations when using the Petzval or Zoom lenses. However, the differences between the simplifications are remarkable when doing visual inspection. In Figure 7.16a and 7.16b, a region of the renders made using the Double Gauss and Pikaichi 35mm lenses, respectively, is shown for the original Seidel renders and the parameter simplifications. Both *Fixed Distance* and *Fixed Most* accurately retain the color present in the *Seidel* renders, while spectral effects are mostly absent with *Fixed Parameters*. The *Fixed Everything* renders have no spectral effects, as is expected due to the full removal of wavelength dependence.



(a) Double Gauss (b) Pikaichi 35mm

Figure 7.16: A region of the shield scene, rendered using the Double Gauss and Pikaichi 35mm lenses.



(a) Petzval (b) Nikkor 50mm

Figure 7.17: A region of the shield scene, rendered using the Petzval and Nikkor 50mm lenses.

From the simulations with the Petzval lens, as illustrated in Figure 7.17a, it is clear that the parameter simplifications are not always sufficient. Here problems arise because the Seidel coefficients are too dependent on the distance to the entrance pupil to remove the distance dependence altogether. The same is true for the Nikkor 50mm lens, as illustrated in Figure 7.17b. The circle of confusion size is clearly a bit smaller in the Seidel method as compared to the simplified variants. Similarly, the circle of confusion size is reduced when using the simplifications with the Zoom lens, as illustrated in Figure 7.18.

Figure 7.18: A region of the shield scene, rendered using the Zoom lens.

7.3.4 Discussion

With the simplified lens parameters we still get pretty good results. But it does come with a cost, as the distance dependence of the Seidel coefficients cannot always be ignored. It is still hard to gauge how useful Seidel DOF is when the lens design is unknown and parameters must be set by hand. We simplified the lens parameters, but did not test how intuitive they might be to an artist trying to match bokeh to existing footage. There are indications that there exist correlations between the lens parameters f and $\frac{1}{-M} d_{\text{image}\to\text{sensor}}$ as their graphs look alike, this may enable us to make some more simplifications – but for now, even our simplified parameters may be too much to handle.

Chapter 8

Conclusions and future work

In this thesis, we have introduced a novel application of Seidel aberrations, a well-known concept in optics theory, that has previously only been implicitly implemented in depth of field simulation. The Seidel DOF method nearly always outperforms other methods when comparing to ray traced reference images in terms of RMSE and MS-SSIM (Figure 7.7). Near the optical axis, the distance aberration on the imaging sensor as compared to ray tracing is nearly 0, but increases when the distance to the optical axis increases (Figure 7.8). For some lenses, the aberration exceeds that of the other methods (Gaussian, Distortion + vignetting). Seidel takes a constant time per ray, just like the Gaussian and Pencil map methods, while ray tracing takes linear time. The time saved by using our method over ray tracing can be significant, especially when using lenses with many surfaces (for example the Zoom lens, which has 29 surfaces).

Seidel outperforms the other methods most when a lens's bokeh does not resemble a uniform disk (Single, Pikaichi 35mm, Double Gauss and Petzval). If the bokeh is more uniformly disk-shaped (Nikkor 135mm, Zoom), the differences between the methods are smaller, but Seidel still is the most accurate. For most tested lenses, the most important aberrations are simulated accurately using Seidel. An exception is the Nikkor 50mm lens, which would benefit from adding higher order aberrations. Similarly, a more accurate simulation of optical vignetting could improve results with the Petzval lens, where this effect is especially pronounced and a major factor in generating the lens's distinctive swirly bokeh.

If we remove the distance dependence of the Seidel coefficients and reduce the number of lens parameters from seven to three (f, z_{fpp} and $\frac{1}{-M}d_{\text{image}\to\text{sensor}}$) we still get good results for most lenses, making the implementation more friendly to manually setting parameters, which makes it more conceivable that the Seidel method could be used in an artistic context where the original lens designs are not available. But for some lenses the distance dependence is too strong to be fully ignored. However, this should not be a problem when the image does not have a large depth range, as the fixed distance can be set to any number. In our implementation, we set the distance to 10^3 meters, while the depth values of the test image were closer to 1 meter.

Improvements can still be made in the simplification of the Seidel algorithm, in order to allow for simpler and more intuitive parameters. The same goes for the Seidel coefficients: still, all five coefficients have their own wavelength dependence. There may exist correlations between the wavelength dependence of the different coefficients, or perhaps the dependence can be modelled with a simple curve. It would also be interesting to see if wave effects like diffraction, as well as the effects of coatings and internal reflections could be added using similar parameterizations as we used in our method.

Similar results may also be obtained by combining aspects of multiple methods: perhaps we only really need wavelength dependence for axial parameters, which would allow for the usage of pencil maps combined with a monochromatic simulation of the four off-axis Seidel aberrations, making life even easier when trying to match bokeh to footage taken with unknown lenses.

In conclusion, Gaussian optics extended with Seidel aberrations forms a solid basis for depth of field simulation. If extended with elements like hole filling it could successfully be used as a post-process for simulating accurate depth of field. Our implementation simulates rays travelling from object space → screen space, but should yield similar results the other way around. Reversing the direction to screen space → object space would allow the use of our method to generate primary rays in any existing ray tracer.

Figure 8.1: Accurate depth of field simulation can be applied to any image with depth information, no matter the medium!

Bibliography

Abadie, G.
2018. A life of a bokeh. https://epicgames.ent.box.com/s/s86j70iamxvsuu6j35pilypficznec04.

Barsky, B. A. and T. J. Kosloff
2008. Algorithms for rendering depth of field effects in computer graphics. In *Proceedings of the 12th WSEAS international conference on Computers*, volume 2008. World Scientific and Engineering Academy and Society (WSEAS).

Born, M., E. Wolf, A. B. Bhatia, P. C. Clemmow, D. Gabor, A. R. Stokes, A. M. Taylor, P. A. Wayman, and W. L. Wilcock
1999. *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*, 4 edition. Cambridge University Press.

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

Demers, J.
2004. Depth of field: A survey of techniques. *Gpu Gems*, 1(375):U390.

Forst, J.
2015. Image SSIM. https://github.com/darosh/image-ssim-js.

Garcia, K.
2017. Circular separable convolution depth of field. In *ACM SIGGRAPH 2017 Talks*, Pp. 1–2.

Gotanda, Y., M. Kawase, and M. Kakimoto
2015. Real-time rendering of physically based optical effects in theory and practice. In *ACM SIGGRAPH 2015 Courses*, Pp. 1–14.

Haeberli, P. and K. Akeley
1990. The accumulation buffer: Hardware support for high-quality rendering. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, Pp. 309–318, New York, NY, USA. ACM.

Hecht, E.
2017. *Optics*. Pearson Education, Incorporated.

Hullin, M. B., J. Hanika, and W. Heidrich
2012. Polynomial optics: A construction kit for efficient ray-tracing of lens systems. In *Computer Graphics Forum*, volume 31, Pp. 1375–1383. Wiley Online Library.

Joo, H., S. Kwon, S. Lee, E. Eisemann, and S. Lee
2016. Efficient ray tracing through aspheric lenses and imperfect bokeh synthesis. In *Computer Graphics Forum*, volume 35, Pp. 99–105. Wiley Online Library.

Kolb, C., D. Mitchell, and P. Hanrahan
1995. A realistic camera model for computer graphics. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, Pp. 317–324, New York, NY, USA. ACM.

Lee, S., E. Eisemann, and H.-P. Seidel
2010. Real-time lens blur effects and focus control. *ACM Transactions on Graphics (TOG)*, 29(4):1–7.

McGraw, T.
2015. Fast bokeh effects using low-rank linear filters. *The Visual Computer*, 31(5):601–611.

Moersch, J. and H. J. Hamilton
2014. Variable-sized, circular bokeh depth of field effects. In *Proceedings of Graphics Interface 2014*, Pp. 103–107.

Polyanskiy, M. N.
2018. Refractive index database. `https://refractiveindex.info`.

Potmesil, M. and I. Chakravarty
1981. A lens and aperture camera model for synthetic image generation. *ACM SIGGRAPH Computer Graphics*, 15(3):297–305.

Präkel, D.
2010. *The Visual Dictionary of Photography.* AVA Publishing SA.

Riguer, G., N. Tatarchuk, and J. Isidoro
2004. Real-time depth of field simulation. *ShaderX2: Shader Programming Tips and Tricks with DirectX*, 9:529–556.

Robinson, S.
2013. ZDefocus. `https://www.foundry.com/products/nuke`.

Schrade, E., J. Hanika, and C. Dachsbacher
2016. Sparse high-degree polynomials for wide-angle lenses. In *Computer Graphics Forum*, volume 35, Pp. 89–97. Wiley Online Library.

Smith, T. and J. Guild
1931. The C.I.E. colorimetric standards and their use. *Transactions of the Optical Society*, 33(3):73–134.

Von Seidel, P. L.
1857. Über 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.*

Wu, J., C. Zheng, X. Hu, and F. Xu
2013. Rendering realistic spectral bokeh due to lens stops and aberrations. *The Visual Computer*, 29(1):41–52.