UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

ALEX REIMANN CUNHA LIMA

Simulating Accommodation and Low-Order Aberrations of the Human Eye using Wave Optics and Light-Gathering Trees

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. Manuel Menezes de Oliveira Neto

Porto Alegre
October 2020

CIP — CATALOGING-IN-PUBLICATION

Cunha Lima, Alex Reimann

Simulating Accommodation and Low-Order Aberrations of the Human Eye using Wave Optics and Light-Gathering Trees / Alex Reimann Cunha Lima. – Porto Alegre: PPGC da UFRGS, 2020.

118 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2020. Advisor: Manuel Menezes de Oliveira Neto.

1. Visual simulation. 2. Low-order aberrations. 3. Partial occlusion artifacts. 4. Fourier Optics. 5. Zernike polynomials. I. Oliveira Neto, Manuel Menezes de. II. Título.

“Profound study of nature is the most fertile source of mathematical discoveries.”

— JOSEPH FOURIER

ACKNOWLEDGMENTS

I must express my very profound gratitude to my father and mother, Eden and Marlise, who have provided me through emotional, moral and financial support in my life. And I am particularly grateful to my mother; besides all the mentioned support, she played a major role in crafting the camera base device and the lens cards used in the experiments performed in this research.

I am also profoundly grateful to Manuel Menezes de Oliveira Neto, for accepting nothing less than excellence from me, but especially for all the knowledge and patience as my advisor.

Moreover, I am also grateful to Iguaçu Distribuidora de Produtos Óticos Ltda for the donation of a Snellen chart and a set of lenses used in the experiments.

Last but not least, a very special gratitude goes out to CNPq for providing funding for the work.

ABSTRACT

In this work, we present two practical solutions for simulating accommodation and low-order aberrations of optical systems, such as the human eye. Taking into account pupil size (aperture) and accommodation (focal distance), our approaches model the corresponding point spread function and produce realistic depth-dependent simulations of low-order visual aberrations (*e.g.*, myopia, hyperopia, and astigmatism). In the first solution, we use wave optics to extend the notion of Depth Point Spread Function, which originally relies on ray tracing, to perform the generation of point spread functions using Fourier optics. In the other technique, we use geometric optics to build a light-gathering tree data structure, presenting a solution to the problem of artifacts caused by absence of occluded pixels in the input discretized depth images. As such, the resulting images show seamless transitions among elements at different scene depths. We demonstrate the effectiveness of our approaches through a series of quantitative and qualitative experiments on images with depth obtained from real environments. Our results achieved SSIM values above 0.94 and PSNR above 32.0 in all objective evaluations, indicating strong agreement with the ground-truth.

Keywords: Visual simulation. low-order aberrations. partial occlusion artifacts. Fourier Optics. Zernike polynomials.

Simulação de acomodação e aberrações de baixa ordem do olho humano usando árvores de coleta de luz

RESUMO

Neste trabalho, apresentamos duas técnicas de simulação de acomodação e aberrações de baixa ordem de sistemas ópticos, tais como o olho humano. Nossos algoritmos lançam mão de determinadas informações, tais como o tamanho da pupila e a acomodação (distância focal), com o objetivo de modelar a função de espalhamento pontual (*point spread function*) do sistema, resultando na produção de simulações realistas de aberrações de baixa ordem (*p.e.*, miopia, hipermetropia e astigmatismo). Nossas simulações levam também em consideração as distâncias dos objetos que compõem a cena a fim de aplicar o borramento apropriado. A primeira técnica estende o conceito de Função de Espalhamento Pontual com Profundidade (*Depth Point Spread Function*), originalmente construída mediante o traçado de raios (*ray tracing*), que passa então a ser gerada por meio de métodos da óptica de Fourier. A segunda técnica, por sua vez, utiliza-se da óptica geométrica para construir uma estrutura de dados em forma de árvore. Esta árvore é então utilizada para simular a propagação da luz no ambiente, gerando os efeitos de borramento esperados, e de quebra soluciona o problema de artefatos visuais causados pela ausência de informação na imagem original (provocada pela oclusão parcial entre elementos da cena). Nós demonstramos a efetividade de nossos algoritmos por meio de uma série de experimentos quantitativos e qualitativos em imagens com profundidade obtidas de ambientes reais. Nossos resultados alcançaram valores de SSIM superiores a 0,94 e valores de PSNR superiores a 32,0 em todas as avaliações objetivas, o que indica uma expressiva concordância com as imagens de referência.

Palavras-chave: Simulação visual. Aberrações de baixa ordem. Artefatos de oclusão parcial. Óptica de Fourier. Polinômios de Zernike.

LIST OF ABBREVIATIONS AND ACRONYMS

CA	Chromatic aberration
CCD	Charge-coupled device
CRT	Cathode-ray tube
DPSF	Depth point spread function
DSLR	Digital single-lens reflex camera
FOV	Field of view
GPL	Geometrical path length
LGT	Light-gathering tree
MDF	Medium-density fiberboard
OPD	Optical path difference
OPL	Optical path length
NPRP	Nearest potentially reachable plane
FPRP	Farthest potentially reachable plane
PI	Plane index
PNG	Portable network graphics
PSF	Point spread function
PSNR	Peak signal-to-noise ratio
SSIM	Structural similarity
RDB	Ray distribution buffer
RGB	Red, green, blue
RGB-D	Red, green, blue - depth

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Vision is arguably our most important sense. As a process used to sense the environments through luminous stimuli, it produces a personal experience influenced by intrinsic characteristics of one's visual system. Thus, achieving faithful simulation for a given individual would require, in principle, a large amount of information from a wide variety of areas, ranging from optics and physiology to psychology and neuroscience (KRUEGER; OLIVEIRA; KRONBAUER, 2016; SCHWARTZ, 2010). However, obtaining such data tends to be impractical and this level of precision might not be justifiable.

Among the optical aberrations, the so-called *low-order* ones, which include regular astigmatism as well as both positive and negative defocus (myopia and hyperopia, respectively), are by far the most frequent cause of diminished visual acuity. They are responsible for approximately 90% of the aberration in the eye (LOMBARDO; LOMBARDO, 2010). With a considerable impact in quality of life, it is estimated that between one and two billion individuals worldwide are affected by refractive errors, but the actual prevalence depends upon the surveyed population; it varies from roughly 25% in Europe to over 80% in some Asian countries (DENNISTON; MURRAY, 2014).

Even though most patients end up using glasses or contact lenses for correcting such aberrations, a considerable number of them undergo refractive surgery as a means of correcting their sight or minimizing the errors. In light of potential risks and benefits of the surgery, it is desirable that patients could preview their post-operation sight, taking into account potential residual refractive errors, and thus make a better informed decision. This can be achieved by simulating the patient's vision before and after the surgery, using pictures and videos blurred in a controlled way. Such simulations should also help doctors and medical students in better understanding the patients' conditions.

We approach the problem of simulating vision in two forms. In the first approach, it is treated using the mathematical rigor of Fourier optics; some unfortunate aspects related to missing information due to partial occlusion, however, led us to address the theoretical aspects of a simplified version of the problem, where the scene is composed of elements that do not occlude each other. This simplification makes the problem easier because the scene is constrained to a unique depth with a convex structure, and the issue derived from different blurring methods for adjacent pixels is mitigated.

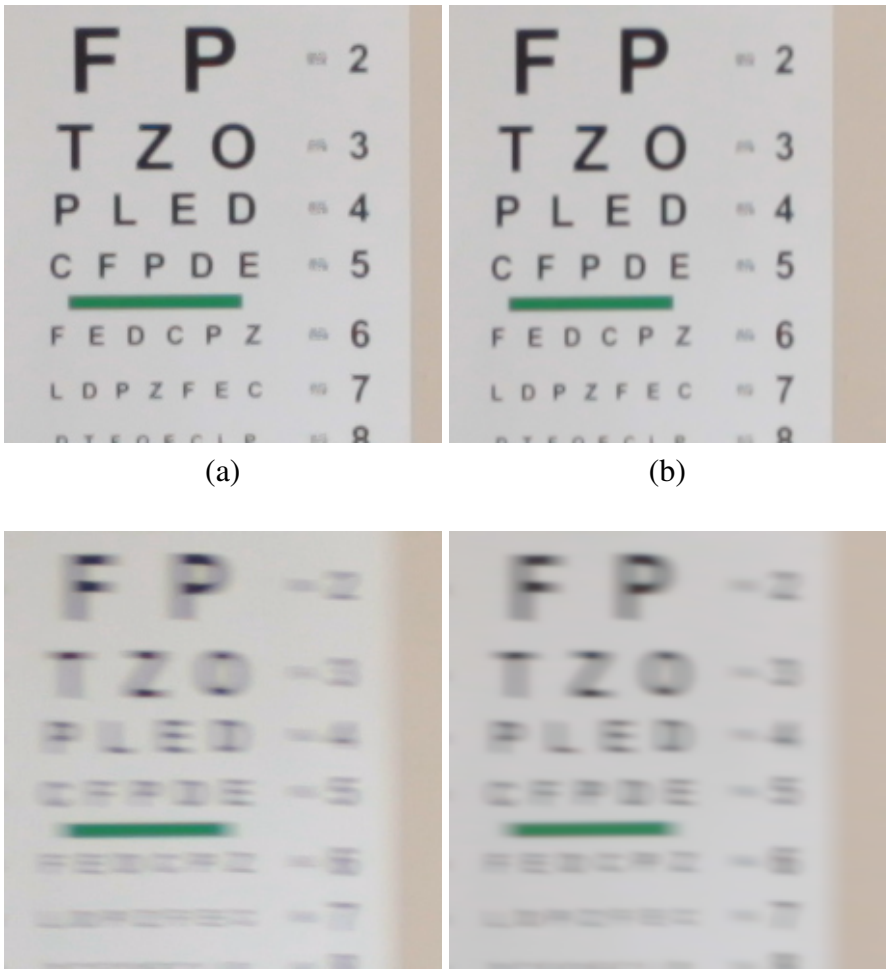
In the second approach, we overcome the inconvenience imposed by occlusions;

but in order to make this problem tractable at interactive times, some simplifications are applied to the scene such as its discretization into a set of parallel planar regions. Furthermore, the introduction of a new data structure (*light-gathering trees*) associated to such set of planes results in simulations that are able to resolve the partial occlusion problem. Nevertheless, our technique produces interactive personalized realistic simulations of how an individual would perceive real scenes considering accommodation and low-order aberrations (*e.g.*, *myopia*, *hyperopia*, and *astigmatism*), pupil size, and focal distance. By taking into account pupil size, our simulations naturally produce realistic depth-of-field effects. All these parameters can be dynamically changed during the simulation. Existing techniques that perform similar tasks are either limited to a single depth (KRUEGER; OLIVEIRA; KRONBAUER, 2016), positive defocus (XIAO et al., 2018), or lack precision when dealing with partially occluded objects (BARSKY, 2004).

Figure 1.1 illustrates a simulation result produced using our Fourier optics technique for a real scene (captured by a DSLR camera) containing an object (eye chart) at a single distance seen by an astigmatic individual. The astigmatism (S = 0 diopters, C = -1 diopters, and $\varphi = 86°$) was induced by an external lens placed in front of the camera lens. Figure 1.1a shows a portion of the original photograph captured without the external lens. Figure 1.1b shows the image obtained after applying anisotropic minification and brightness adjustment to Figure 1.1a to compensate for the use of the external lens. Figure 1.1c depicts the ground-truth astigmatic image obtained using the external lens in front of the camera. Figure 1.1d presents the simulated result produced by our Fourier optics technique taking Figure 1.1b as input. Note the similarity between the ground truth and our simulated result.

Figure 1.2 illustrates the use of our second technique for simulating accommodation and low-order aberration on a scene containing elements at various distances from the observer. Such distances are represented in Figure 1.2c. The white and blue flower, the red flower, and the game box are approximately 0.5 m, 1 m, and 2 m away from the observer, respectively. Figure 1.2a shows a simulated view of a myopic subject with 0.5 diopters and no accommodation (thus focusing 2.0 m away). Note that the texture of the game box, which is located at approximately 2 m, exhibits a relatively sharp texture. As the distance decreases towards the red, and then the white and blue flower, blurring increases. In particular, note the blurry blue petal against the sharp game box texture. The corresponding reference sub-images are shown at the bottom of Figure 1.2c, where all elements appear sharp regardless of their relative depths. Figure 1.2b shows a simu-

Figure 1.1: Simulation of astigmatism using Fourier optics. Aberration parameters: S = 0 diopters, C = -1 diopters, $\varphi = 86°$. (a) Picture taken without extra lens. Camera settings: ISO 100, exposure 1/40 s, $f = 20$ mm, $f/5$. (b) Anisotropic minification and brightness adjustment applied to (a) to compensate for the use of an external lens used to induce astigmatism. (c) Ground-truth image obtained with an external lens in front of the camera. (d) Simulated result produced by the Fourier optics technique.

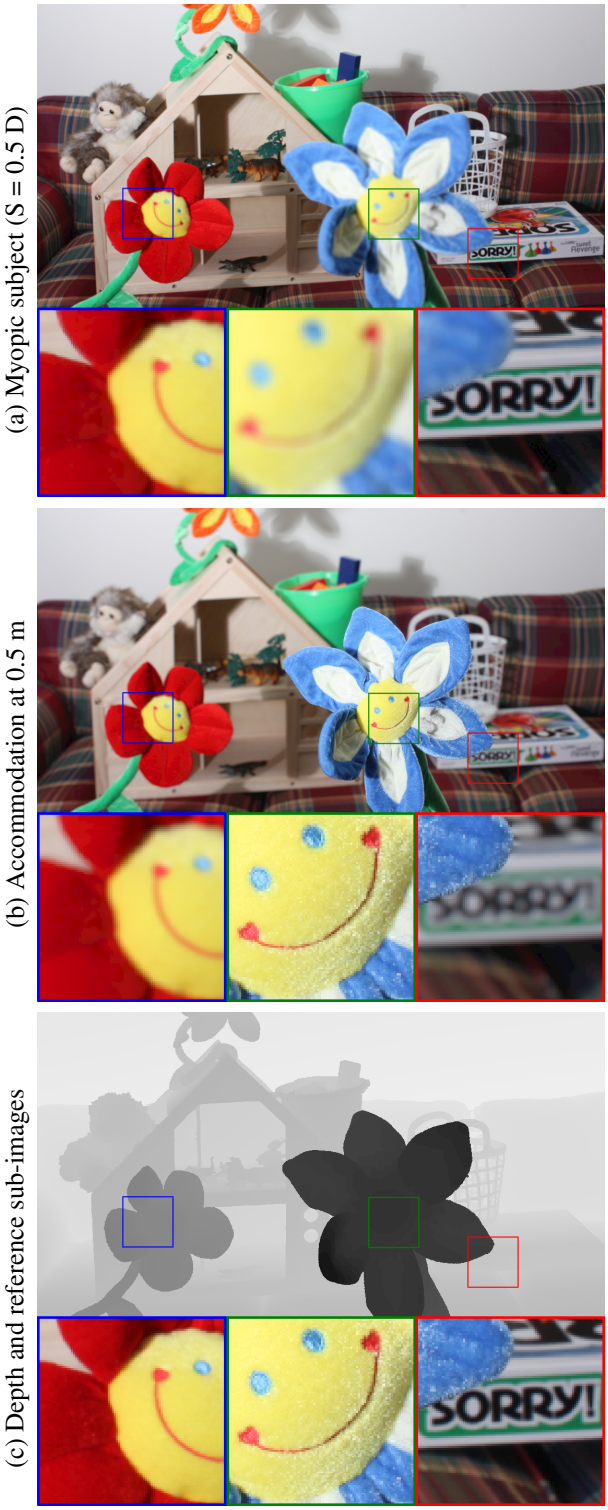


(a) (b)

(c) (d)

Source: The Authors

Figure 1.2: Simulation of accommodation and low-order aberrations using our technique. It is applied to a scene containing elements whose approximate distances to the observer are: white and blue flower, 0.5 m; red flower, 1 m; and game box 2 m. (a) Simulated view of a myopic subject with 0.5 diopters and no accommodation. Note that starting from a relatively sharp game box texture, the amount of blur progresses as the distance decreases towards the white and blue flower. Note the blurry blue petal against the sharp game box texture. The corresponding reference (sharp) sub-images are shown at the bottom of (c). (b) Simulated view of the same myopic individual this time accommodating at the white and blue flower, which now appears sharp while the game box texture becomes blurry. (c) Scene depth (top) and reference sub-images (bottom).



Source: The Authors

lated view of the same myopic individual focusing (*i.e.*, accommodating) at the white and blue flower. Such flower now appears sharp while the game box texture becomes blurry. In both examples, the red flower (located at an intermediate depth) shows some relative defocusing with respect to its reference image.

The **goal** of this work is to provide a method for interactive vision simulation considering real scenes and low-order optical aberrations and accommodation.

The **contributions** of this thesis include:

- An interactive technique for producing realistic simulations of the human vision under low-order aberrations, accommodation, and variable scene depth (Chapter 5). Our technique provides smooth transitions among scene elements located at different depths.
- A tree data structure used for light gathering that allows the handling of partial occlusions among objects in the presence of a finite pupil (Chapter 5). Such new data structure allows us to provide a practical solution to a long-standing problem related to vision simulation of real environments.
- A derivation of the Fourier transform performed by convex thin lenses geared towards the Computational Photography community (Sections 2.4 to 2.6).
- A mathematical formulation for computing the coefficients of the Zernike polynomials in order to model how one would perceive an object at an arbitrary distance while (s)he is focusing at a different distance (Chapter 4).

1.1 Thesis structure

This thesis is structured as follows. Chapter 2 reviews some geometric and wave optics concepts, including Fourier optics, required for understanding the development of the thesis. Chapter 3 discusses works closely related to ours. In Chapter 4 presents a vision simulation technique based on Fourier optics. Chapter 5 describes our second vision simulation technique, which is based a tree data structure built using geometric optics to perform light gathering. Chapter 6 describes the experiments performed to validate both techniques, as well as the obtained results. Chapter 7 summarizes this thesis contributions and presents some guidelines and ideas for future exploration.

2 BACKGROUND

This chapter presents a physics background regarding some aspects of the phenomenon of light which are necessary to understand both simulation techniques presented in this work. Light propagation itself can be explained using three different models: light rays in *geometric optics*; perturbation of the electromagnetic field in *wave optics*; and evolution of a complex probability density function in *quantum mechanics*. While the latter gives the most precise, and currently the most extensive known explanation for the light phenomenon, it adds an unnecessary layer of complexity; as a consequence, only the models of geometric and wave optics will be considered.

Section 2.1 explains the ray propagation model for light, which serves as a basis for the understanding of the bending of light and refractive errors. Section 2.2 shows in a glimpse how eyeglass prescriptions are organized. Section 2.3 tackles Zernike polynomials, which are used to mathematically model wavefront errors. Section 2.4 introduces some aspects of wave optics; together with Section 2.5, which shows the phase transformation of thin lenses, and the Huygens-Fresnel principle presented in Section 2.6, they form the basis for understanding *Fourier optics*, derived in Section 2.7. Partial-occlusion effects are described in Section 2.8.

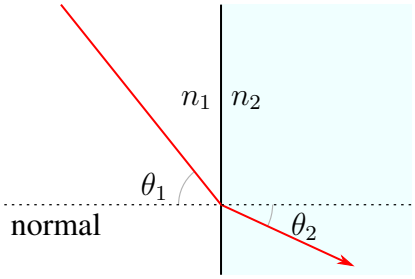
2.1 Geometric optics

In the geometric optics model, it is assumed that light in a homogeneous medium travels in straight lines called *light rays*. Besides normal propagation, light can also be absorbed, reflected, or even bent in some circumstances, which are analyzed on the next subsection. Note that, in accordance with the convention used in this work, the light source will always be illustrated on the left and the resulting image will be produced on the right; thus, light is assumed to propagate from left to right.

2.1.1 Refraction

When light rays strike the interface between different media, they are either reflected, absorbed or transmitted. In the case of transmittance, the propagation velocity of light can change due to a phenomenon that can only be properly explained by

Figure 2.1: Snell's Law. Interface between the two surfaces is represented by the vertical line, and the normal to the interface is the dashed horizontal line. The refractive indices of the left and right media are n_1 and n_2 respectively. In this example, $n_2 > n_1$. Light ray is represented by red arrow.



Source: The Authors

wave optics or quantum mechanics. The ratio between the speed of light in vacuum $c = 299{,}792{,}458\ m/s$ and its speed v in a given medium is called index of refraction, usually indicated by

$$n = \frac{c}{v}.$$

If both sides of the interface have different refractive indices (n_1 and n_2), the ray will bend according to those indices and the angle formed by the incoming ray direction and the interface normal (θ_1). This relationship (Figure 2.1) is summarized by Snell's law, which states that

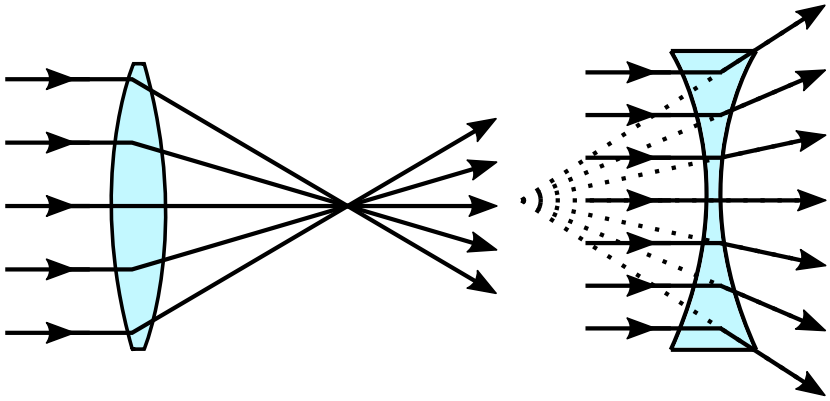
$$n_1 \sin\theta_1 = n_2 \sin\theta_2. \tag{2.1}$$

Lenses, which consist of especially designed transparent materials with curved surfaces, are devices that use the phenomenon of refraction to bend light in a controlled way and produce images. Convergent (convex) lenses focus parallel light rays coming from object space to a single point in image space (Figure 2.2a). Divergent (concave) lenses diverge parallel light rays coming from object space, giving the impression that they diverge from a single point in object space (Figure 2.2b). In the former case, a real inverted image of the object is formed; in the latter case, the image is virtual and right way up.

A matter of the utmost importance is to consider the distance between the object and the lens (S_o) and that between the object's image and the lens (S_i). They are related to each other, and can be computed by

$$\frac{1}{S_o} + \frac{1}{S_i} = (n-1)\left(\frac{1}{R_1} + \frac{1}{R_2}\right). \tag{2.2}$$

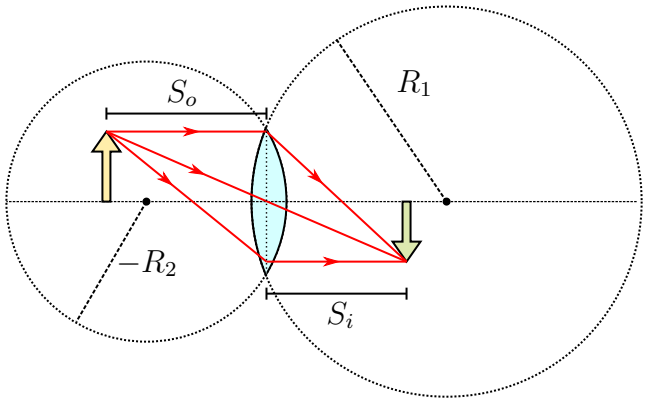
Figure 2.2: Convergent and divergent lenses. (a) Convergent lens, where parallel light rays converge to a single point on the right of the lens. (b) Divergent lens, where parallel light rays appear to diverge from a single point on the left of the lens.



(a) (b)

Source: The Authors

Figure 2.3: Lens maker relation. Distance from object to the lens is S_o. Distance from image to the lens is S_i. The radii of the intersecting spheres that form the left and right lens surfaces are R_1 and $-R_2$ respectively. The distances are measured from the central lens plane (represented by a dotted vertical line segment), since we are dealing with thin lenses.

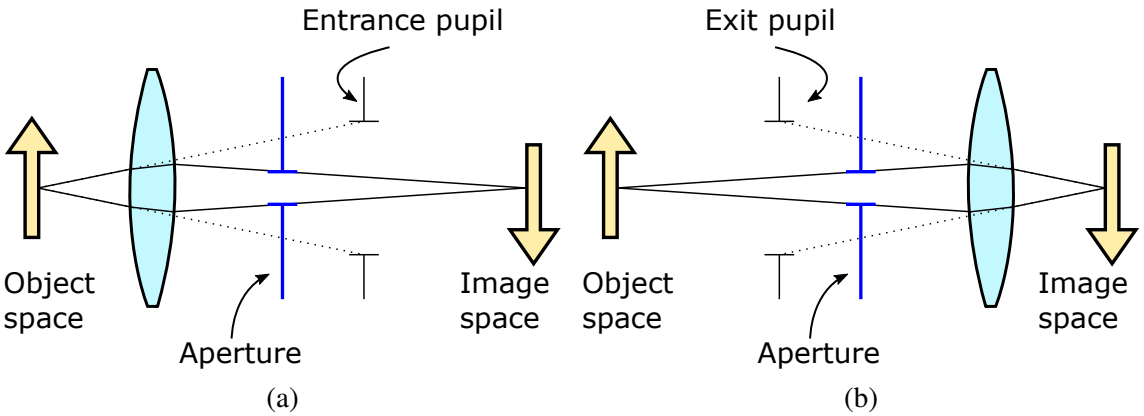


Source: The Authors

The value of S_o is always positive, while S_i is positive when the image is real (on the opposite side as the object) and negative otherwise. The radii of both surfaces of the lens are indicated by R_2 and R_1. The adopted sign convention imposes that R_1 is positive and R_2 is negative.

Every optical system has an imaginary line crossing its geometrical center (rotation axis), called *optical axis*. All light rays parallel to the optical axis crossing the lens meet at the same point at a distance f, which is the most important feature of the lens. Known as *focal distance*, it can be found by setting $S_o = \infty$ in Equation (2.2) and solving

Figure 2.4: Aperture stop, entrance and exit pupils. Entrance pupil (a) is the virtual image of the aperture stop as seen from object space. Exit pupil (b) is the virtual image of the aperture stop as seen from image space.



Source: The Authors

for $f = S_i$, yielding

$$\frac{1}{f} = (n-1)\left(\frac{1}{R_1} + \frac{1}{R_2}\right). \tag{2.3}$$

The focal distance is positive for converging lenses and negative for diverging lenses.

Magnification is another important feature of the optical phenomenon which measures the ratio between the size of the image and the size of the respective object, as indicated by

$$M = -\frac{S_i}{S_o}.$$

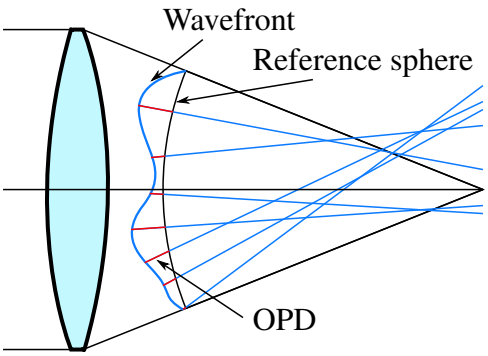
A negative magnification indicates that the image is upside-down, and that is usually the case in the applications considered in this work.

Aperture stop is the physical hole that determines the maximum conic solid angle of the bundle of light rays through an optical system. Two notable examples are the physical pupil of the human eye and the camera shutter. The image of the aperture as seen from the object plane is known as *entrance pupil* (Figure 2.4a). Conversely the image of the pupil as seen from the image plane is known as *exit pupil* (Figure 2.4b). Both are virtual images, thus their sizes do not usually match the actual aperture size due to magnification. The *entrance pupil* is closely related to the *f-number* of an optical system, given by

$$N = \frac{f}{2R}, \tag{2.4}$$

where f is the system's focal length and R is the radius of entrance pupil. It is common to indicate the f-number preceded by "$f/$". Its value tells the lens "speed", or image brightness, which is inversely proportional to the square of the f-number.

Figure 2.5: Wavefront and Gaussian reference sphere. The OPD between wavefront and reference sphere, shown in red, is known as wavefront error.



Source: The Authors

The optical path length (OPL) is obtained by summing the geometric path length (GPL) traversed by light in each medium (which is the actual distance traveled by light rays in that medium) multiplied by the refractive index of the corresponding medium (HECHT, 2002):

$$OPL = \sum_{i=1}^{m} n_i \times GPL_i.$$

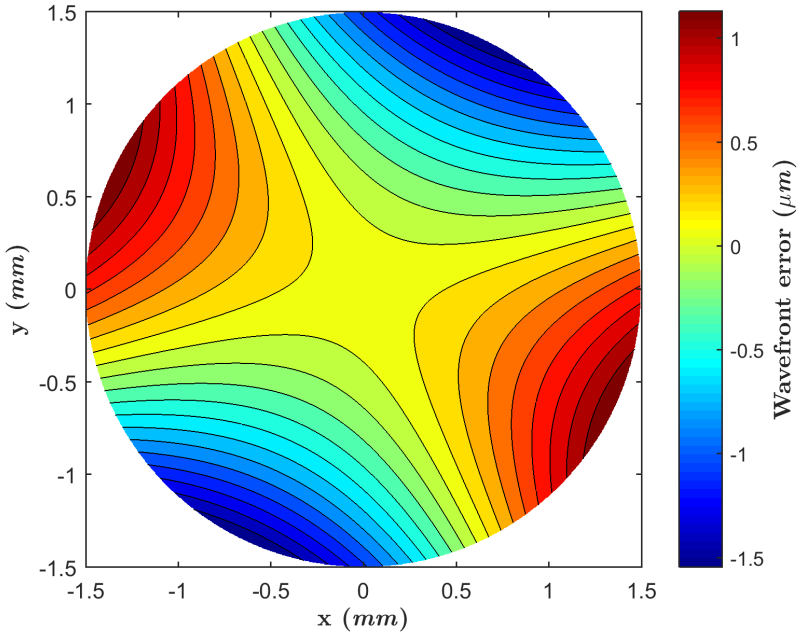
The difference between two OPL is the optical path difference (OPD), defined by

$$OPD = n_1 GPL_1 - n_2 GPL_2. \tag{2.5}$$

Besides its relevance in geometric optics, the OPD also plays a meaningful role when dealing with the wave characteristics of light, because it is directly related to the oscillation phase difference and results in important optical effects (shown in Section 2.5).

Surfaces containing points sharing the same OPL are known as *wavefronts* (Figure 2.5). When the optical system is not aberrated, the wavefronts formed in image space by incoming rays that were parallel to the optical axis in object space are shaped as a *Gaussian reference sphere* centered at the lens focal point (Figure 2.5). On the other hand, in an aberrated optical system, the same wavefronts deviate from the reference sphere; the OPD between the actual wavefront and the expected ideal reference sphere is known as *wavefront error*. The map of wavefront errors between the wavefront and the reference sphere, when both are tangent to the exit pupil, is known as *wavefront error map* (Figure 2.6).

Figure 2.6: Wavefront error map for an aberrated optical system with S = -1 D, C = 2.5 D, and $\varphi = 25°$. Exit pupil diameter is 3 mm. Each point on the map represents the wavefront deviation (in μm) from a Gaussian reference sphere centered on the ideal focal point (where the optical axis intersects the image plane).



Source: The Authors

2.1.2 Refractive errors

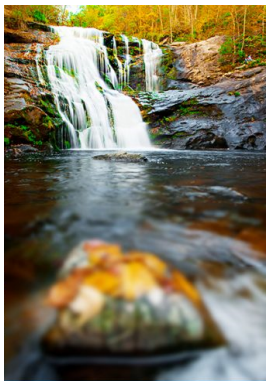
Refractive errors (or optical aberrations) comprise a class of monochromatic optical aberrations (as opposed to chromatic aberrations). They occur when a point light source is not properly focused into a single image point due to aberrations in the shape of the lens (and more specifically the human eye), causing light to bend incorrectly. This causes a divergence between the image that would be produced by an ideal system and the actual obtained image.

The two types of aberrations relevant to this work are *defocus* (Figures 2.7b and 2.7c) and *ophthalmic astigmatism* (Figure 2.7d). Among other definition methods, they are also characterized by the wavefront error, which can be described by first and second order Zernike polynomials (introduced in Section 2.3), and as such are also called *low-order aberrations*.

Figure 2.7: Effects of refractive errors on vision. (a) Emmetropic vision. (b) Myopic vision. (c) Hyperopic vision. (d) Astigmatic vision (left) compared to vision without astigmatism (right).

(a)

(b)

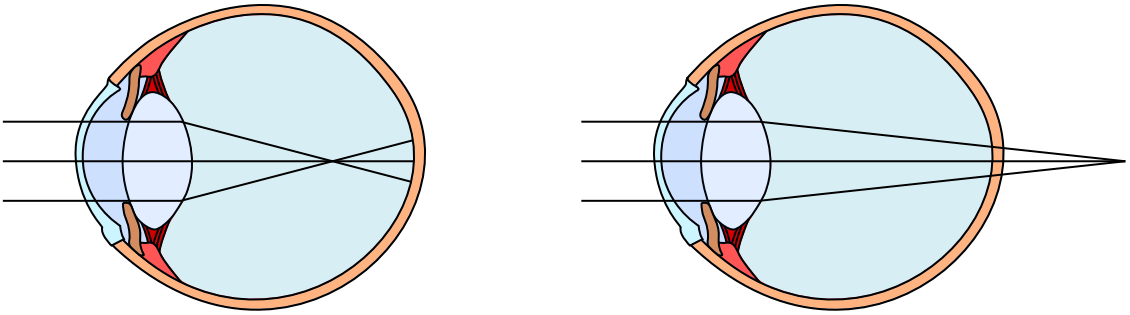


(c)

(d)

Source: (a,c,b) Lee Hung Ming eye centre; (d) Oakland Eye Care

Figure 2.8: Myopia and hyperopia in the human eye. In myopia, the focal point for parallel rays is shorter than expected, and the patient is unable to focus on distant objects. On the other hand, the focal point is longer in hyperopia, and the patient is unable focus on close objects.



(a) Myopia (nearsightedness). (b) Hyperopia (farsightedness).

Source: The Authors

2.1.3 Defocus

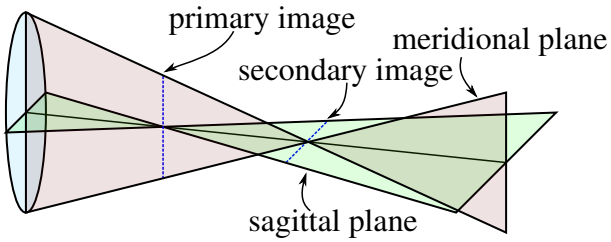
Defocus comprises myopia (nearsightedness) and hyperopia (farsightedness). It happens when the image is formed on a plane different from the focal plane. This error is easily fixed with a simple translation of the lens (or the focal plane). The mathematical formulation for this aberration stems from the OPD of a smaller circle tangent to a larger circle. Note that defocus should not be confused with *spherical aberration*, which originates from the OPD of a parabola tangent and inscribed into a circle.

Myopia (Figure 2.8a) occurs when either the eye shape is longer than normal along the optical axis or the intrinsic properties of the cornea cause light to bend more than expected. Either way, the image is formed on a plane shifted towards the lens. Hyperopia (Figure 2.8b), on the other hand, occurs when the eye's axial length is shorter than normal, or when the cornea is more planar than usual. In these situations, the light rays converge to a point beyond the retina. Both myopia and hyperopia are quantified with a single number: the *lens power* required to cause the shift between the expected image plane and the plane where image is perfectly focused. This value is defined by

$$S = \frac{1}{f},$$

and has units of m^{-1}, also known as *diopters* (D). Positive optical power denotes myopia; negative indicates hyperopia. As one should expect, in order to compensate for the aberrated vision, hyperopic lenses are prescribed to nearsighted patients and myopic lenses are prescribed to farsighted patients.

Figure 2.9: Astigmatic optical system showing the two images formed. Rays on the sagittal plane (shown in light green) meet at a point on the primary image (a vertical line segment). Conversely, rays on the meridional plane (shown in pink) meet at a point on the secondary image (a horizontal line segment). As a result, there is no single focal point.



Source: The Authors

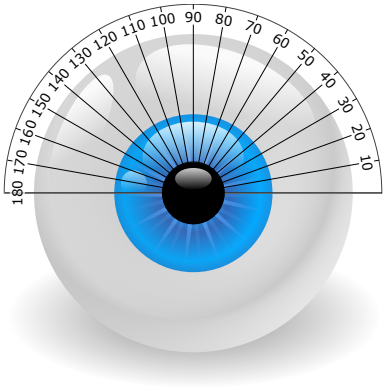
2.1.4 Ophthalmic astigmatism

Ophthalmic astigmatism is caused by an irregular curvature in the cornea (corneal astigmatism) or in the lens (lenticular astigmatism). In both cases, the shape of the eye is not curved equally in all meridians, and as a consequence, rather than a point at a single focal plane, two ellipsis-shaped spots, called *primary* and *secondary images*, are formed at different planes. An example can be seen in Figure 2.9, where these spots are represented by a vertical and a horizontal line respectively.

Due to the anisotropic nature of astigmatism, a direction is used to define the meridian perpendicular to the aberration. This is called *cylinder*, since this is the shape of a lens with zero optical power on the meridian coinciding with the cylinder axis, and maximum power on the orthogonal meridian. Thus, unlike defocus, ophthalmic astigmatism needs two numbers to be characterized: optical power (indicated by letter "C", in diopters), and axis angle (indicated by the symbol "φ", in degrees). The angle is used to define the direction of the cylinder axis, and the optical power indicates the curvature of the cylinder (always perpendicular to the axis). The eye meridians are measured according to the angle formed between the cylinder axis and the horizontal line in counter-clockwise direction, as seen in Figure 2.10.

Ophthalmic astigmatism should not be confused with another type of astigmatism, known as *oblique astigmatism*, which is a third-order aberration that occurs even on perfectly symmetrical lens when objects are off-axis. Ophthalmic astigmatism does not depend on the object's distance to the optical axis. It is usually classified according to the simultaneous occurrence of defocus on the same optical system as myopic, hyperopic, and mixed. In *mixed* astigmatism, one meridian is nearsighted and the other is farsighted. In *myopic* astigmatism, on the other hand, one meridian is nearsighted and the other is

Figure 2.10: Eye meridians used to indicate cylindrical axis. The convention used in this work states that meridians are measured counter-clockwise on both eyes as if one is viewing the patient's eye straight on and, using a clock as reference, the hour hand is on 0° at three o'clock position.



Source: Adapted from <https://openclipart.org/detail/23899>

Figure 2.11: Eyeglass prescription.

	SPH	CYL	AX
O.D.	-0.50	1.75	40
O.S.	-0.70	DS	

Source: The Authors

nearsighted or flat. In *hyperopic* astigmatism, one meridian is farsighted and the other is farsighted or flat. In all cases, it can be compensated for with a toric lens with the same cylindrical axis but opposite powers with respect to those of the eye.

2.2 Eyeglass prescriptions

Spectacle prescriptions are usually written in a form similar to the one presented in Figure 2.11. The abbreviations O.D. and O.S. stand for the Latin terms "oculus dexter" and "oculus sinister", which translate to *right eye* and *left eye*, respectively. The headlines "SPH", "CYL" and "AX" lay out the optical parameters for sphere, cylinder, and axis. *Sphere* indicates the correction in diopters used to compensate defocus. *Cylinder* tells the astigmatic maximum correction power in diopters on the meridian perpendicular to the given *axis*. The two letters "DS" written on the cylinder column of the left eye stand for *diopter sphere* and indicate absence of astigmatism on that eye.

Two different notations are used to indicate astigmatism: *plus-cylinder* and *minus-cylinder*. They differ only on the sign of cylindrical power and are remnants of the methods used for constructing lenses (either adding or subtracting a given power from a base lens). Both notations are equivalent and can be easily converted into each other through a

Table 2.1: Cylinder transposition of S = 3 D, C = 1 D, $\varphi = 150°$.

Step	Description	Operation	Current values
1	Add spherical and cylindrical powers to obtain the new spherical power.	S $\longleftarrow$ S + C	S = 4 D, C = 1 D, $\varphi = 150°$
2	Change sign of cylinder.	C $\longleftarrow$ -C	S = 4 D, C = -1 D, $\varphi = 150°$
3	Add 90° to the axis angle.	$\varphi \longleftarrow \varphi + 90°$	S = 4 D, C = -1 D, $\varphi = 240°$
4	Ensure axis angle lies in range (0°, 180°).	$\varphi \longleftarrow \varphi (\text{mod } 180°)$	S = 4 D, C = -1 D, $\varphi = 60°$

Source: The Authors

process known as *cylinder transposition*. In order to perform the conversion, the cylindrical and spherical powers should be added to obtain the new spherical power. After that, the sign of the cylindrical power is changed and 90° should be added to the cylinder axis angle; the final angle should be adjusted to stay inside the range 0° to 180°. An example of cylinder transposition for the prescription S = 3 D, C = 1 D, $\varphi = 150°$, from plus-cylinder to minus-cylinder notation, is shown in Table 2.1. The same algorithm applies to transposition from minus-cylinder to plus-cylinder notation.

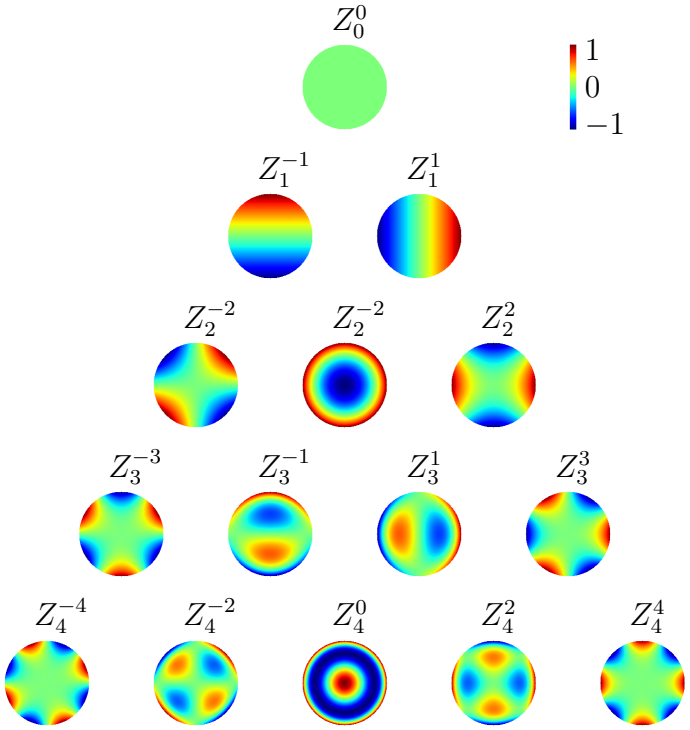
There are different approaches to measure cylinder axis angles currently in usage by optometrists and ophthalmologists. It is assumed in this work that all angles are measured as if one is viewing the patient's eye straight on and, using a clock as reference, the hour hand is on 0° at three o'clock position. Angles increase counter-clockwise and are only counted up to 180°, as shown on Figure 2.10.

2.3 Zernike polynomials

In order to describe wavefront errors, it is important to define functions that map every point on the pupil plane to the respective deviation from a perfect plane wave. The domain of such functions is commonly a dimensionless subset of $\mathbb{R}^2$, extending from -1 to 1 in both dimensions. Considering that optics is usually based on circular apertures, a special subset D of this domain is usually used instead, and it is known as *closed unit disk*. It can be mathematically described as the set of points $Q \in \mathbb{R}^2$ for which the distance to the origin $(0, 0)$ is less than or equal to one:

$$D = \{Q : |Q| \leq 1\}.$$

Figure 2.12: The first fifteen Zernike polynomials on the unit circle domain. Colors indicate the value of the function, ranging from -1 (blue) to 1 (red), as shown in the color bar.



Source: The Authors

Zernike polynomials, introduced by the optical physicist Frits Zernike in 1934, are a class of orthogonal polynomials useful for characterizing wavefront errors on the closed unit disk. Initially used in phase contrast microscopy for quantifying wavefront aberrations in circular mirrors, over the years they saw widespread usage and became "one of the most popular orthonormal polynomials over circular pupils" (DAI, 2008). They are formed by the product of a radial factor and an angular factor. Since their natural domain is the unit circle, they are more naturally expressed using polar coordinates and may be used to describe any real function on the pupil plane.

It is useful to distinguish between *even* and *odd* polynomials. Even polynomials are obtained by the expression

$$Z_n^m(\rho, \omega) = N_n^m R_n^m \cos(m\omega), \tag{2.6}$$

where ω is the polar angle (ranging from 0 to 2π) and n, m are non-negative integers. The index n indicates the highest power of ρ in the polynomial, and therefore is called *order*. The value m stands for azimuthal frequency, and denotes the frequency of the angular repeating pattern (Figure 2.12). Magnitude ρ is used to calculate the radial coefficient

R_n^m. Note that it uses an even function (cosine). Odd polynomials, on the other hand, are defined by

$$Z_n^{-m}(\rho, \omega) = N_n^m R_n^m \sin(m\omega), \tag{2.7}$$

and use an odd function (sine).

The normalization factor N_n^m is defined by

$$N_n^m = \sqrt{\frac{2(n+1)}{1+\delta_{m0}}},$$

where δ_{m0} is the Kronecker delta (its value is 1 when $m = 0$, and 0 otherwise).

Finally, the radial part is given by the polynomial

$$R_n^m(\rho) = \sum_{k=0}^{(n-m)/2} \frac{(-1)^k (n-k)!}{k!(\frac{n+m}{2}-k)!(\frac{n-m}{2}-k)!} \rho^{n-2k}.$$

Important features of Zernike polynomials include *orthogonality* over the closed unit disk and *rotational symmetry*. By the former, additions of new terms to the polynomial do not disrupt the surface (coefficients are independent). Furthermore, when in orthonormal form, the coefficients of the polynomial terms represent their standard deviations. The latter allows Zernike polynomials to be expressed as products of radial factors and functions of angle, like $R(\rho)G(\omega)$. Note that $G(\omega)$ is a continuous periodic function with period 2π; as a corollary, the coordinate system can be rotated by an angle α without changing the form of the polynomial (WYANT; CREATH, 1992). In other words,

$$G(\omega + \alpha) = G(\omega)G(\alpha).$$

There are two ways of referencing a Zernike polynomial: double-index mode, which uses indices n and m, and single-index mode, which uses index j. The formulas (DAI, 2008) to convert from from single to double-index mode are

$$n = \left\lfloor \sqrt{2j+1} + 0.5 \right\rfloor - 1, \text{ and}$$
$$m = 2j - n(n+2),$$

and the formula to convert from double to single-index mode is

$$j = \frac{n^2 + 2n + m}{2}.$$

Table 2.2: Zernike polynomials and their respective Noll indices and common names

Symbol	*Noll index*	*Name*
Z_0^0	1	Piston
Z_1^{-1}	2	Vertical tilt
Z_1^1	3	Horizontal tilt
Z_2^{-2}	4	Oblique astigmatism
Z_2^0	5	Defocus
Z_2^2	6	Vertical astigmatism
Z_3^{-3}	7	Vertical trefoil
Z_3^{-1}	8	Vertical coma
Z_3^1	9	Horizontal coma
Z_3^3	10	Oblique trefoil
Z_4^{-4}	11	Oblique quadrafoil
Z_4^{-2}	12	Oblique secondary astigmatism
Z_4^0	13	Primary spherical
Z_4^2	14	Vertical secondary astigmatism
Z_4^4	15	Vertical quadrafoil

Source: The Authors

Single-indices are also known as Noll's indices. In this text, we use mostly the double-index representation, and rely on single-index mode for certain stages of the algorithms presented in Appendix B.

Table 2.2 lists the first fifteen Zernike polynomials, as well as the common aberrations represented by them. Expanded formulas for both polar and Cartesian forms of the polynomials are shown in Table 2.3, and their respective graphs are plotted on Figure 2.12. In particular, myopia and hyperopia are modeled by Z_2^0 (defocus), while astigmatism is modeled by a linear combination of Z_2^1 and Z_2^{-1} (oblique and vertical astigmatism). The polynomials' coefficients, derived from the Seidel series (DAI, 2008), are

$$c_2^{-2} = \frac{R^2 C \sin 2\varphi}{4\sqrt{6}}, \tag{2.8}$$

$$c_2^0 = -\frac{R^2(S + C/2)}{4\sqrt{3}}, \text{and} \tag{2.9}$$

$$c_2^2 = \frac{R^2 C \cos 2\varphi}{4\sqrt{6}}. \tag{2.10}$$

As a result, defocus-only wavefront aberration is modeled as

$$W_S = c_0^0 Z_0^0, \tag{2.11}$$

and the common set of low-order wavefront aberrations (myopia, hyperopia and astigma-

Table 2.3: Zernike polynomials in polar and Cartesian forms

Symbol	*Polar Polynomial*	*Cartesian Polynomial*
Z_0^0	1	1
Z_1^{-1}	$\sqrt{4}\rho\sin\theta$	$\sqrt{4}y$
Z_1^1	$\sqrt{4}\rho\cos\theta$	$\sqrt{4}x$
Z_2^{-2}	$\sqrt{6}\rho^2\sin 2\theta$	$\sqrt{6}(2xy)$
Z_2^0	$\sqrt{3}(2\rho^2-1)$	$\sqrt{3}(2x^2+2y^2-1)$
Z_2^2	$\sqrt{6}\rho^2\cos 2\theta$	$\sqrt{6}(x^2-y^2)$
Z_3^{-3}	$\sqrt{8}\rho^3\sin 3\theta$	$\sqrt{8}(3x^2y-y^3)$
Z_3^{-1}	$\sqrt{8}(3\rho^3-2\rho)\sin\theta$	$\sqrt{8}(3x^2y+3y^3-2y)$
Z_3^1	$\sqrt{8}(3\rho^3-2\rho)\cos\theta$	$\sqrt{8}(3x^3+3xy^2-2x)$
Z_3^3	$\sqrt{8}\rho^3\cos 3\theta$	$\sqrt{8}(x^3-3xy^2)$
Z_4^{-4}	$\sqrt{10}\rho^4\sin 4\theta$	$\sqrt{10}(4x^3y-4xy^3)$
Z_4^{-2}	$\sqrt{10}(4\rho^4-3\rho^2)\sin 2\theta$	$\sqrt{10}(8x^3y+8xy^3-6xy)$
Z_4^0	$\sqrt{5}(6\rho^4-6\rho^2+1)$	$\sqrt{5}(6x^4+12x^2y^2+6y^4-6x^2-6y^2+1)$
Z_4^2	$\sqrt{10}(4\rho^4-3\rho^2)\cos 2\theta$	$\sqrt{10}(4x^4-4y^4-3x^2+3y^2)$
Z_4^4	$\sqrt{10}\rho^4\cos 4\theta$	$\sqrt{10}(x^4+y^4-6x^2y^2)$

Source: The Authors

tism), as a whole, can be computed by the linear combination

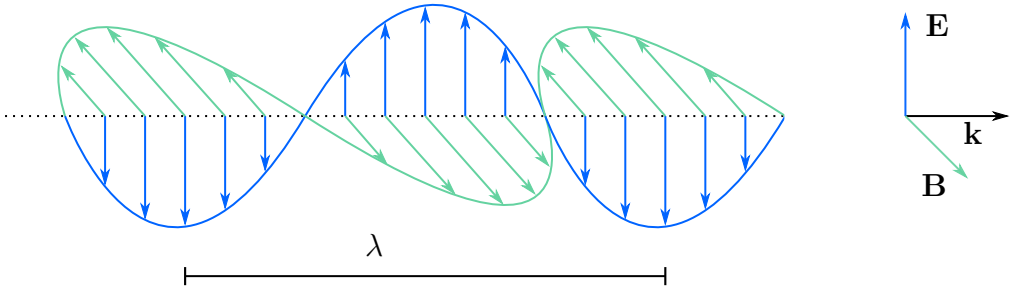
$$W_{S,C,\varphi} = c_2^{-2}Z_2^{-2} + c_2^0 Z_2^0 + c_2^2 Z_2^2. \tag{2.12}$$

2.4 Wave Optics

Electric and magnetic fields are vector fields existing throughout the entire space, and together they are known as the *electromagnetic field*. Wave optics regards light as a disturbance in the electromagnetic field caused by an accelerating charge oscillating with frequency ν, which propagates at speed c as a transverse wave. The first complete mathematical description of its nature was given in the 19$^{\text{th}}$ century by James Clerk Maxwell, as he unified and complemented a set of equations of electromagnetism discovered by various scientists.

When the medium is vacuum, these revised form of these equations (revision per-

Figure 2.13: An electromagnetic wave propagating in space. Note how the electric ($\mathbf{E}$) and magnetic ($\mathbf{B}$) fields are orthogonal to each other, and also how their values change as a sinusoidal function of position. The wavelength λ is depicted as the distance between points with the same phase.



Source: The Authors

formed by Oliver Heaviside) can be written in differential form as

$$\nabla \cdot \epsilon_0 \mathbf{E} = 0, \tag{2.13}$$

$$\nabla \cdot \mu_0 \mathbf{B} = 0, \tag{2.14}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}, \text{and} \tag{2.15}$$

$$\nabla \times \mathbf{B} = \mu_0 \epsilon_0 \frac{\partial \mathbf{E}}{\partial t}. \tag{2.16}$$

In these equations, the electric and magnetic fields are indicated by $\mathbf{E}$ and $\mathbf{B}$, respectively, time is t, and the vacuum *permittivity* and *permeability* are represented by ϵ_0 and μ_0, respectively. The fields $\mathbf{E}$ and $\mathbf{B}$ are always in phase, orthogonal to each other, and their magnitudes are related by

$$||\mathbf{B}|| = ||\mathbf{E}||/c.$$

After applying the *curl identity*

$$\nabla \times (\nabla \times \mathbf{A}) = \nabla(\nabla \cdot \mathbf{A}) - \nabla^2 \mathbf{A}$$

to Equation (2.15), substituting Equations (2.13) and (2.16) and using the relation

$$c = 1/\sqrt{\mu_0 \epsilon_0}$$

in the result, one can obtain the wave equation

$$\left(\nabla^2 - \frac{1}{c^2}\frac{\partial^2}{\partial t^2}\right) u(\mathbf{r}, t) = 0,$$

where position in space is indicated by the vector $\mathbf{r}$, and $u(\mathbf{r}, t)$ is the wave function, or *light field*, usually regarded as a proxy for the electric field. Reasons for considering the light field as a scalar rather than a vector field, which is the case for electric and magnetic fields, are given later in this section.

By setting different initial and boundary conditions, several solutions to this partial differential equation can be obtained. Two of them in particular are the *spherical wave*, given by

$$u_s(r, t) = \frac{E_0}{r} e^{j(kr-\omega t)}, \tag{2.17}$$

which represents a point-like object emitting monochromatic light from a finite distance away, and the *plane wave*, given by

$$u_p(r, t) = E_0 e^{j(kr-\omega t)}, \tag{2.18}$$

which can be regarded as the disturbance originated from a point source infinitely far away or an infinite plane source at any distance. In both cases, E_0 is the base amplitude, j is the imaginary unit $\sqrt{-1}$, ω is the angular frequency defined by

$$\omega = 2\pi\nu,$$

and k is the optical wave number defined as the number of radians per unit distance, or

$$k = \frac{2\pi}{\lambda} = \frac{2\pi\nu}{c},$$

where λ is the wavelength. The value r in the denominator of Equation (2.17) indicates that the displacement caused by spherical waves fades away as r increases.

At any point in time, the total intensity of the wave is given by the Poynting vector,

$$\mathbf{S} = \frac{1}{\mu_0} \mathbf{E} \times \mathbf{B}, \tag{2.19}$$

which is orthogonal to both $\mathbf{E}$ and $\mathbf{B}$ and indicates the direction of energy flow. This energy depends on the instantaneous value of the displacement of both electric and magnetic fields, but since they oscillate in time, the time-average is a much more relevant quantity. Before calculating the average, the light field oscillation function should be split into a time-independent and a time-dependent function by writing the scalar field u, with

arbitrary amplitude $A(r)$, as

$$u(r,t) = A(r)e^{jkr}e^{-j\omega t} = U(r)e^{-j\omega t},$$

where $U(r)$ is a time-independent complex function called phasor, defined as

$$U(r) = A(r)e^{jkr}.$$

The time-average of the intensity, derived from the instantaneous intensity relation at Equation (2.19), can be shown to be proportional to the squared magnitude of the complex phasor, since the average of the time-dependent function over a long period of time approaches $1/2$, as shown in

$$I(\mathbf{r}) = \langle ||\mathbf{S}(\mathbf{r},t)|| \rangle = \lim_{T\to\infty} \int_0^T \frac{1}{\mu_0 cT} \Re\{U(\mathbf{r})e^{-j\omega t}\}^2 \, dt = \frac{1}{2\mu_0 c}|U(\mathbf{r})|^2. \tag{2.20}$$

In this equation, the angular brackets $\langle\rangle$ represent *continuous average over time* and $\Re$ returns the real part of a complex number. This result indicates that any phase factor present in the final field $U(\mathbf{r})$ can be safely discarded without affecting the result. It also indicates that a time-independent representation is enough for eventually recovering intensity, so that Equation (2.17) and Equation (2.18) can be rewritten as the phasors

$$U_s(r) = \frac{E_0}{r}e^{jkr} \tag{2.21}$$

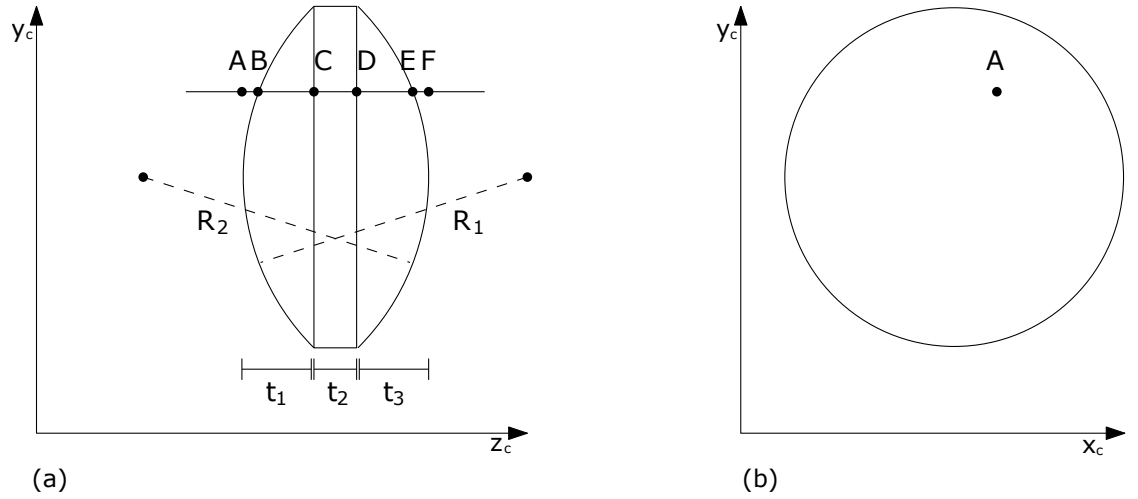
and

$$U_p(r) = E_0 e^{jkr}$$

respectively.

Finally, it is important to state that the reason for treating the light field as a scalar field rather than a vector field are given by Goodman (2005), who argues that a scalar theory of diffraction renders results very close to the vector results because only on boundaries the vector characteristics of light will matter. Since the apertures usually dealt with are far larger than the wavelength of light and considerably distant from the image plane on that same scale (a very legitimate assumption for the situations presented in this work), the boundary characteristics will not matter and the results will be consistent. It also should be emphasized that usage of electric field rather than magnetic field as a proxy for describing u is a common convention in wave optics; the magnetic field is

Figure 2.14: Optical path of a light ray traversing a thin lens. (a) Side view of the lens. The radii (R_1 and R_2) of the spherical caps are shown with dashed lines. Thickness and curvature of surfaces are exaggerated when in comparison with actual thins lens, for a better depiction of the phenomenon. (b) Front view of the lens. All points (A – F) coincide in this view because the ray is orthogonal to the x_c and y_c axes.



Source: The Authors

always present, perpendicular to the electric field, but its intensity is so much lower that it seems more natural to use the latter. Moreover, by using a scalar theory of diffraction rather than a vector treatment, the directional differences between magnetic and electric vectors are already neglected.

2.5 Phase transformation of thin lenses

This section shows, using a demonstration adapted from the one presented in Goodman (2005), how the characteristics of a thin lens regarded as an optical apparatus capable of converging light rays into a (virtual or real) focus can be translated into wave optics, where it acts as a phase transformation device. In order to simplify the analysis of light propagation, the lens has been split into three objects: a spherical cap on the left with positive curvature R_1 and maximum thickness t_1, a cylinder of thickness t_2, and a spherical cap on the right with negative curvature R_2 and maximum thickness t_3 (Figure 2.14a). An arbitrary light ray crosses each of these objects, in the given order, passing through points A, B, C, D, E, and F. By the definition of thin lenses, translation between points B and E is negligible, but phase transformation, which is orders of magnitude smaller than the lens thickness, should be considered.

Phase delay depends on the distance traveled inside the lens. For each object traversed, there is a thickness function, which depends on the (x_c, y_c) position of the intersection between the light ray and the lens (Figure 2.14b). Note that we use the c subscript in the continuous domain coordinates x_c and y_c to differentiate them from the discrete domain coordinates x and y in later sections. The thickness function $\Delta_1(x_c, y_c)$ of the first object struck by the ray gives the distance BC. It is computed by limiting the sphere equation $R_1^2 = x_c^2 + y_c^2 + z_c^2$ using the cap thickness t_1. Distance $(R_1 - z_c)$ is then subtracted from t_1, which results in

$$\Delta_1(x_c, y_c) = \sqrt{R_1^2 - x_c^2 - y_c^2} + t_1 - R_1.$$

Pulling R_1 out of the square root yields

$$\Delta_1(x_c, y_c) = R_1\sqrt{1 - \frac{x_c^2 + y_c^2}{R_1^2}} + t_1 - R_1.$$

Applying the binomial approximation (Appendix A.1) and simplifying yields

$$\Delta_1(x_c, y_c) = -\frac{x_c^2 + y_c^2}{2R_1} + t_1. \tag{2.22}$$

The second object struck by the light ray is a cylinder, which implies that the thickness CD is given by the constant function

$$\Delta_2(x, y) = t_2. \tag{2.23}$$

Like the first object, the third and last object struck by the light ray is also a spherical cap, but its radius is negative by definition. In this case, thickness DE is calculated first, and the distance $(-R_2 - z)$ is then subtracted from the sphere cap thickness t_3, resulting in

$$\Delta_3(x_c, y_c) = \sqrt{R_2^2 - x_c^2 - y_c^2} + t_3 + R_2.$$

Pulling $-R_2$ out of the square root yields

$$\Delta_3(x_c, y_c) = -R_2\sqrt{1 - \frac{x_c^2 + y_c^2}{R_2^2}} + t_3 + R_2.$$

Applying the binomial approximation and simplifying results in

$$\Delta_3(x_c, y_c) = \frac{x_c^2 + y_c^2}{2R_2} + t_3. \tag{2.24}$$

The total OPL is the sum of the OPLs computed for the three objects, which have refractive index n, added to AB and EF, which are the distances where the ray crosses the air (refractive index 1), as indicated in

$$OPL = t_1 + (n-1)\Delta_1 + n\Delta_2 + t_3 + (n-1)\Delta_3. \tag{2.25}$$

Plugging Equations (2.22) to (2.24) into Equation (2.25) yields

$$OPL = t_1 + (n-1)\left(-\frac{x_c^2 + y_c^2}{2R_1} + t_1\right) + nt_2 + t_3 + (n-1)\left(\frac{x_c^2 + y_c^2}{2R_2} + t_3\right).$$

Regrouping common terms results in

$$OPL = (n-1)\left(\frac{1}{R_1} - \frac{1}{R_2}\right)\left(-\frac{x_c^2 + y_c^2}{2}\right) + n(t_1 + t_2 + t_3). \tag{2.26}$$

Plugging Equation (2.3) in Equation (2.26) and *dropping the phase shift* $n(t_1 + t_2 + t_3)$ yields

$$OPL = -\frac{x_c^2 + y_c^2}{2f}. \tag{2.27}$$

After embedding Equation (2.27) into the argument of a phasor, it is possible to compute the light field phase transformation caused by a thin lens with focal distance f. It is the result of multiplying the incoming light field by

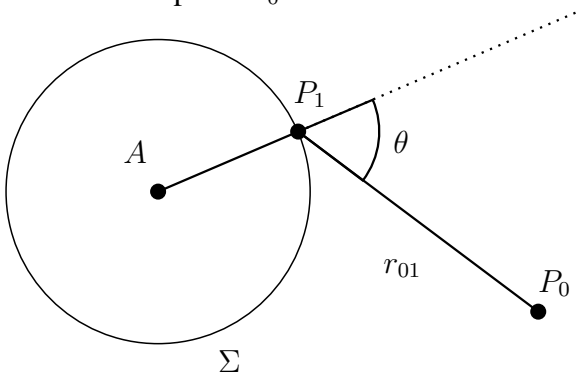
$$L_f(x_c, y_c) = e^{-jk\frac{1}{2f}(x_c^2 + y_c^2)}. \tag{2.28}$$

2.6 Huygens-Fresnel principle

When light encounters a barrier on its way, and it is no longer traveling a negligible distance as was the case considered in the previous section, the task of computing the field displacement at an arbitrary position is no longer trivial. Such cases are influenced by the phenomenon of diffraction, which takes the leading role in determining the evolution of the disturbance in space and time.

An important observation made by Christiaan Huygens in the 17$^{\text{th}}$ century be-

Figure 2.15: Huygens-Fresnel principle. Light is emitted by point A and the contribution of all the wavelets P_1 centered at the wavefront Σ are integrated in order to compute the field displacement at observation point P_0.



Source: The Authors

comes very useful in these cases. According to *Huygens principle*, *every point of a wavefront may be considered as a center of a secondary disturbance in form of a wavelet, and the wavefront at any later instant may be regarded as the envelope of these wavelets*. This principle is not physically correct, but it is very useful in practice. Schwartz (1987), for instance, argues that it "(...) gives the right answer for the wrong reasons", since "(...) light does not emit light; only accelerating charges emit light". Anyway, Huygens principle was a remarkable observation at the time and has been a tremendously useful tool ever since.

A later addition by Augustin-Jean Fresnel included the necessity of considering diffraction when calculating the effects of the wavelets, and this modification was so relevant that two combined ideas have been known as *Huygens-Fresnel principle* (illustrated on Figure 2.15). According to it, the field displacement at observation point P_0 can be obtained by integrating the contribution of an infinite number of spherical wavelets P_1 centered along the wavefront Σ. The integral is written as

$$U(P_0) = \frac{1}{j\lambda} \iint_{\Sigma} U(P_1) \frac{e^{jkr_{01}}}{r_{01}} \cos\theta \, d\Sigma, \tag{2.29}$$

where $\mathbf{r_{01}} = P_0 - P_1$ and θ is the angle between the wavefront normal at P_1 and $\mathbf{r_{01}}$. The factor $1/(j\lambda)$ indicates that the wavelets' phases are leading the emitter phase by 90° and experience a reduction in field amplitude inversely proportional to the wavelength. The inclination factor $\cos\theta$ also indicates a field amplitude reduction.

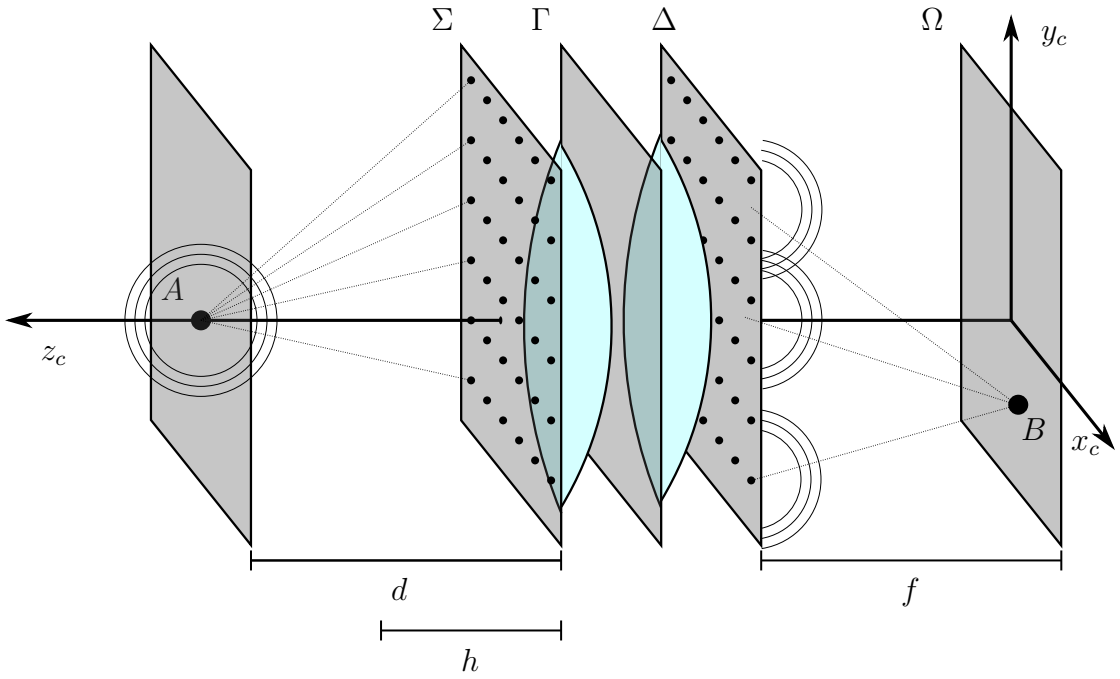
2.7 PSF generation

This section explains how an optical system generates a point spread function (PSF) out of an input point light source. The PSF is the image of the point source, and it indicates how faithfully the system reproduces objects on the image plane. In the case of a perfect optical system (which does not exist in practice), the PSF is a two-dimensional Dirac delta. An aberrated system tends to spread out the points on the image plane, causing the PSF to be blurred, but nevertheless it should be normalized in order to keep image energy intact. In signal processing jargon, the PSF is the impulse response of a focused optical imaging system, and the image produced by a lens can be obtained by the convolution of an image produced by an ideal system with the lens PSF.

Our wave propagation analysis starts on the light emitted by a point A, illustrated in Figure 2.16. A short time after the spherical wave emerges from it, the wavefront encounters an aperture at plane Σ, and after crossing a lens with focal length h, it reaches plane Γ. Afterwards, it crosses a second lens with focal length f before reaching plane Δ, and finally continues propagating towards plane Ω, where the image (PSF) is formed by the contribution of many points (only one of them, represented by B, is shown in Figure 2.16). Distance from the object A to plane Σ is d; distance from plane Δ to plane Ω is f, which is the same as the focal distance for the rightmost lens. Both lenses are considered thin lenses, so that all properties seen in Section 2.5 apply, and as a consequence the distance from plane Σ to plane Δ is zero.

The propagation of light has been split into three parts. In the first part, propagation outside the optical system from A to the plane Σ is described. The second part deals with the adaptive focus (accommodation) that takes place due to propagation inside the optical system, from plane Σ to plane Γ. Finally, in the third part, after a final phase shift caused by a second lens, from plane Γ to plane Δ — which, in the case of an incident plane wave ($d = h$), should turn the planar wavefront into a converging spherical shape centered at the origin —, the Huygens-Fresnel principle is applied to the light field in order to compute its convergence from plane Δ to the final image plane Ω, where the PSF will be registered. Two lenses have been used rather than one, in order to better explain the accommodation effects (first lens) and the image formation focus (second lens) separately, but the results are the same if a single lens combining both effects is used.

Figure 2.16: Image information in an optical system from a point-source on the optical axis using Fourier optics. Light propagates from point-source A, passing through two thin lenses between planes Σ (aperture), Γ, and Δ, and producing an image on plane Ω. Each point B on the image plane results from the superposition of an infinite number of wavelets emerging from plane Δ. The focal lengths of the left and right lenses are h and f respectively. Note that both lenses and the planes Σ, Γ, and Δ are on the same location, but shown at different depths in order to indicate the different stages of the wave propagation.



Source: The Authors

2.7.1 Point source illumination

The spherical wave emitted by point A, at a distance d from the aperture (plane Σ), can be described by the phasor at Equation (2.21). In a first approximation, which is possible because d is much larger than the aperture (paraxial approximation), the value r in the denominator is replaced by d (GOODMAN, 2005), yielding

$$U_\Sigma(x_c, y_c) = \frac{E_0}{d} e^{jkr}. \tag{2.30}$$

The same value r in the exponent cannot be approximated by d, and so it is expanded as

$$r = \sqrt{x_c^2 + y_c^2 + d^2}.$$

Pulling the value d out of the square root yields

$$r = d\sqrt{\frac{x_c^2 + y_c^2}{d^2} + 1},$$

and applying the binomial approximation to the square root results in

$$r = \frac{x_c^2 + y_c^2}{2d} + d. \tag{2.31}$$

Replacing Equation (2.31) in Equation (2.30) yields

$$U_\Sigma(x_c, y_c) = e^{jk(\frac{1}{2d}(x_c^2+y_c^2)+d)}.$$

Finally, after *dropping the phase shift*, the value obtained is

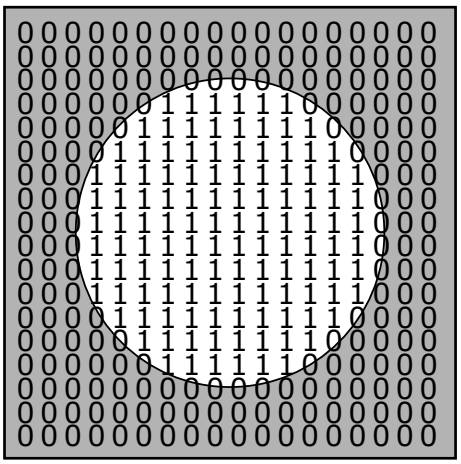
$$U_\Sigma(x_c, y_c) = e^{jk\frac{1}{2d}(x_c^2+y_c^2)}, \tag{2.32}$$

which represents the light field distribution on the aperture Σ.

2.7.2 Accommodation and aberrations

In order to calculate the light field on plane Γ, the one on Σ is multiplied by the phase transformation performed by the leftmost lens (Equation (2.28)), which has focal

Figure 2.17: The pupil function. It is characterized as a windowing function that assigns the value 1 inside the (usually circular) aperture and 0 outside.



Source: The Authors

length h, and also by the aberration function U_W and the pupil function P, yielding

$$U_\Gamma(x_c, y_c) = U_\Sigma(x_c, y_c) L_h(x_c, y_c) U_W(x_c, y_c) P(x_c, y_c).$$

When $h = d$, the phase transformation caused by the light propagation from the object to the plane is exactly canceled out by the lens phase transformation, resulting in

$$U_\Sigma(x_c, y_c) L_h(x_c, y_c) = e^{jk(\frac{1}{2d}(x_c^2+y_c^2))} e^{-jk\frac{1}{2h}(x_c^2+y_c^2)} = 1. \tag{2.33}$$

In practice, this happens when the optical system is focusing the point source A (human eye accommodation by changing the shape of the crystalline lens). In all other cases, there will be a vestigial phase transformation that can be considered an accommodation-induced defocus aberration.

When the aperture is circular, the pupil function (Figure 2.17) can defined as

$$P(x_c, y_c) = \begin{cases} 1 & \text{if } x_c^2 + y_c^2 \leq R^2, \text{or} \\ 0 & \text{otherwise.} \end{cases} \tag{2.34}$$

It provides windowing effects by setting the light field to zero outside an aperture of radius R. The shape of the aperture plays a crucial role on the diffraction effects considered in the next subsection.

Besides accommodation-induced defocus, any other lens aberration effects can also be considered at this plane by plugging the wavefront error $W(x_c, y_c)$ into the argu-

ment of the aberration and obtaining

$$U_W(x_c, y_c) = e^{-jkW(x_c,y_c)}. \tag{2.35}$$

The complex function U_Γ, also denoted by $\mathbb{P}$, is known as the *generalized pupil function*.

2.7.3 Superposition

After reaching plane Γ, the wavefront will be subject to another phase transformation on its way to plane Δ. This time, it is dictated by a non-aberrated lens with fixed focal distance f. Thus, the light field on plane Δ is

$$U_\Delta(x_c, y_c) = \mathbb{P}e^{-jk\frac{1}{2f}(x_c^2+y_c^2)}. \tag{2.36}$$

Afterwards, light will continue its journey and will eventually reach plane Ω. The image on plane Ω is then calculated as the superposition of the wavelets on the plane Δ using the Huygens-Fresnel principle shown in Equation (2.29). The resulting value of the disturbance on the image plane, calculated for every point B on that plane, is given by

$$E(\zeta, \eta) = \frac{1}{j\lambda} \iint_{-\infty}^{\infty} U_\Delta(x_c, y_c) \frac{e^{jkr}}{r}\, dx_c\, dy_c,$$

where the inclination factor $\cos(\theta)$ is assumed to be 1, and r is the distance from the wavelet at (x_c, y_c, f) to every point B on the image plane at $(\zeta, \eta, 0)$. The integration domain can safely be changed from the wavefront surface to the whole $\mathbb{R}^2$ plane because the windowing effects of the aperture have already been incorporated by the pupil function in U_Δ. Replacing the r in the exponent by the Euclidean distance results in

$$E(\zeta, \eta) = \frac{1}{j\lambda} \iint_{-\infty}^{\infty} U_\Delta(x_c, y_c) \frac{1}{r} e^{jk\sqrt{(x_c-\zeta)^2+(y_c-\eta)^2+f^2}}\, dx_c\, dy_c.$$

Using a binomial approximation to get rid of the square root yields

$$E(\zeta, \eta) = \frac{1}{j\lambda} \iint_{-\infty}^{\infty} U_\Delta(x_c, y_c) \frac{1}{r} e^{jkf} e^{j\frac{k}{2f}(\zeta^2+\eta^2)} e^{j\frac{k}{2f}(x_c^2+y_c^2)} e^{-j\frac{k}{f}(x_c\zeta+y_c\eta)}\, dx_c\, dy_c.$$

Another approximation is applied by replacing r in the denominator by f. In this case, the approximation is justified by the fact that f is much larger in magnitude than $(x_c - \zeta)$ and $(y_c - \eta)$. The disturbance obtained so far is

$$E(\zeta,\eta) = \frac{e^{jkf}e^{j\frac{k}{2f}(\zeta^2+\eta^2)}}{j\lambda f}\iint_{-\infty}^{\infty} U_\Delta(x_c,y_c)e^{j\frac{k}{2f}(x_c^2+y_c^2)}e^{-j\frac{k}{f}(x_c\zeta+y_c\eta)}\,dx_c\,dy_c. \tag{2.37}$$

Replacing Equation (2.36) in Equation (2.37) yields the final phase transformation on plane Ω, which is

$$E(\zeta,\eta) = \frac{e^{jkf}e^{j\frac{k}{2f}(\zeta^2+\eta^2)}}{j\lambda f}\iint_{-\infty}^{\infty} \mathbb{P}(x_c,y_c)e^{-j\frac{k}{2f}(x_c^2+y_c^2)}e^{j\frac{k}{2f}(x_c^2+y_c^2)}e^{-j\frac{k}{f}(x_c\zeta+y_c\eta)}\,dx_c\,dy_c.$$

The quadratic factors on (x_c, y_c) cancel out, yielding

$$E(\zeta,\eta) = \frac{e^{jkf}e^{j\frac{\pi}{\lambda f}(\zeta^2+\eta^2)}}{j\lambda f}\iint_{-\infty}^{\infty} \mathbb{P}(x_c,y_c)e^{-j\frac{2\pi}{\lambda f}(x_c\zeta+y_c\eta)}\,dx_c\,dy_c. \tag{2.38}$$

The factor $\beta = \lambda f$ shows up at various places in Equation (2.38) and can be regarded as a scaling factor. For compactness, one can group the factors outside the integral using

$$M = \frac{1}{\beta}\exp(j2\pi f/\lambda + j\pi/\beta(\zeta^2+\eta^2) - j\pi/2).$$

Aside from the factor M, the integral resembles the continuous Fourier transform of the light field at plane Γ with a scaling factor of $1/\beta$:

$$E(\zeta,\eta) = M\iint_{-\infty}^{\infty} \mathbb{P}(x_c,y_c)e^{-j\frac{2\pi}{\beta}(x_c\zeta+y_c\eta)}\,dx_c\,dy_c. \tag{2.39}$$

2.7.4 Discretization

The discretization step involves some careful choices that might affect the expected results in important ways. The pixel pitch of the generalized pupil function is arbitrary, and affects the quality of the resulting details. It is the ratio between the exit pupil diameter $2R$ and the n pixels that the diameter should correspond to.

The pixel pitch of the PSF is also arbitrary, but for our usage cases it should be

equal to the pixel pitch of the camera sensor (or any pixel discretization desired for the retina). The pixel pitch of the image is ρ. A mapping from continuous dimensions to pixels is given by the functions

$$F(u,v) = E\left(\rho u, \rho v\right) \tag{2.40}$$

and

$$U(x,y) = \mathbb{P}\left(\frac{2R}{n}x, \frac{2R}{n}y\right). \tag{2.41}$$

After discretizing the integral in Equation (2.39) using Equations (2.40) and (2.41), one obtains

$$F(u,v) = M\sum_{x=0}^{n-1}\sum_{y=0}^{n-1} U(x,y)\left[e^{-j\frac{2\pi}{\beta}\rho\frac{2R}{n}(xu+yv)}\right]\frac{(2R)^2}{n^2}.$$

Moving the constant factor out of the summation yields

$$F(u,v) = M\frac{(2R)^2}{n^2}\sum_{x=0}^{n-1}\sum_{y=0}^{n-1} U(x,y)e^{-j\frac{2\pi}{\beta}\rho\frac{2R}{n}(xu+yv)}. \tag{2.42}$$

The scaling factor α takes into account the various parameters and is defined by

$$\alpha = \frac{\beta}{\rho 2R} = \frac{\lambda N}{\rho}, \tag{2.43}$$

where N if the *f-number* (ratio between the focal length f and the exit pupil diameter $2R$). After replacing Equation (2.43) in Equation (2.42), one obtains the α-scaled discrete Fourier transform multiplied by a complex factor:

$$F(u,v) = M\frac{(2R)^2}{n^2}\sum_{x=0}^{n-1}\sum_{y=0}^{n-1} U(x,y)e^{-j\frac{2\pi}{\alpha n}(xu+yv)} = M\frac{(2R)^2}{n^2}\mathcal{F}_\alpha\{U\}.$$

When $\alpha > 1$, computing $\mathcal{F}_\alpha\{U\}$ can be accomplished by enlarging the $n \times n$ domain of U to $\alpha n \times \alpha n$ with zero-filling before calculating $\mathcal{F}\{U\}$. Otherwise, no previous enlargement of the domain should be done, but the result should be downscaled by $1/\alpha$.

Using Equation (2.20), it is possible to find the intensity of the light at every spot of the PSF with the formula

$$I(u,v) = \frac{1}{2\mu_0 c}\left|M\frac{(2R)^2}{n^2}\mathcal{F}_\alpha\{U\}\right|^2.$$

The result can be cropped as necessary, taking into account that the pixel pitch of the

resulting image corresponds to ρ.

After calculating the intensity, the PSF should be normalized in order to ensure that the energy on the original picture stays the same when using the PSF as a convolution kernel. After normalization the multipliers are eliminated, yielding

$$PSF(u,v) = |\mathcal{F}_\alpha\{U\}|^2 / \left(\sum_{\forall u,v} |\mathcal{F}_\alpha\{U\}|^2(u,v) \right), \tag{2.44}$$

which shows that *the discrete PSF is the normalized α-scaled power spectrum of the discrete generalized pupil function.*

Depending on the sensor pixel pitch, the PSF may convey visible diffraction patterns, as shown in Figure 2.18. This is a representation of the impulse response of a diffraction-limited optical system with wavelengths

$$\begin{aligned} \lambda_r &= 700\ nm, \\ \lambda_g &= 510\ nm, \text{ and} \\ \lambda_b &= 440\ nm, \end{aligned} \tag{2.45}$$

as recommended by Krueger, Oliveira and Kronbauer (2016), for the simulation of PSFs for color images. Each one, known as an *Airy pattern*, can be computed using Equation (2.44) with U representing the generalized pupil function of a plane wave, for a camera with a huge pixel density (roughly 2,888 pixels/mm). The pattern is mathematically described by

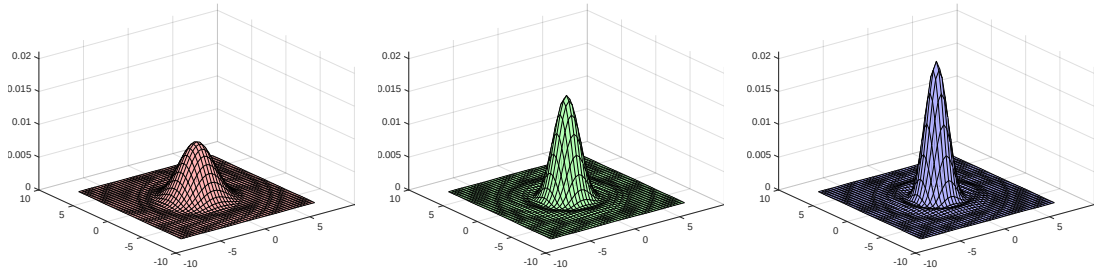
$$I(r) = E_0 \left(\frac{2J_1(\pi r/(\lambda N))}{\pi r/(\lambda N)} \right)^2, \tag{2.46}$$

where r is the radial distance from the center to each point of the pattern, E_0 is the maximum central intensity, and J_1 is the Bessel function of the first kind of order one. The first zero of this function can be shown to occur at

$$r \approx 1.22 \lambda N, \tag{2.47}$$

which is defined as the radius of the central bright circle, known as *Airy disk*.

Figure 2.18: PSF Airy patterns of a plane wave for three different wavelengths. They correspond to the usual RGB color channels, and their differences in size and intensity are notorious. The image plane where they are formed is assumed to be an idealized camera sensor with pixel density of roughly 2,888 pixels/mm. Focal length of the camera lens is set to 18 mm. The units of the sensor plane (horizontal axes) are in μm, and the vertical axis indicates the dimensionless normalized PSF weights. In an actual real CCD, the Bayer filter colors are not exactly red, green and blue. Even if they were, their corresponding PSFs would still differ slightly from those presented here, due to each band-pass filter sensitivity spectrum spanning a certain finite interval of wavelengths, rather than being limited to its peak wavelength sensitivity.



(a) $\lambda_r = 700\ nm$ (red) and $\alpha_r = 10.0649$

(b) $\lambda_g = 510\ nm$ (green) and $\alpha_g = 7.3330$

(c) $\lambda_b = 440\ nm$ (blue) and $\alpha_b = 6.3265$

Source: The Authors

2.8 Partial occlusion effects

A very challenging aspect of realistic vision simulation is dealing with partial occlusion effects. This issue is illustrated in Figure 2.19, which presents a scene with rosebuds in the foreground (occluder object) and a sunflower in the background. When the foreground is in focus (Figure 2.19a), it blocks the view of the background object. However, when the background is in focus (Figure 2.19b), there is a see-through effect throughout all the foreground, and the background sunflower becomes almost completely visible; in this case, one says that the background is partially occluded.

Figure 2.20 shows a schematic side view of this scene's structure. The background plane stands for the sunflower, and the occluding plane are the rosebuds. Note that, when focusing on the foreground plane (Figure 2.20a), a large finite region on the background reflects light rays that contribute to the formation of a single point on the image plane. This is the reason for the background to be blurry, since the colors of several points are averaged together. On the other hand, a single point on the occluding plane reflects several rays that contribute to the formation of a single point on the image plane (Figure 2.20b). This is why the foreground is sharp. In this case, there is no color mixing between background and foreground.

Figure 2.19: Partial occlusion effects. (a) When the foreground is in focus, rosebuds appear sharp and the background is blurry. (b) When the sunflower on the background is in focus, parts of it that were occluded on (a) become visible and the foreground appears to be translucent.

(a) (b)

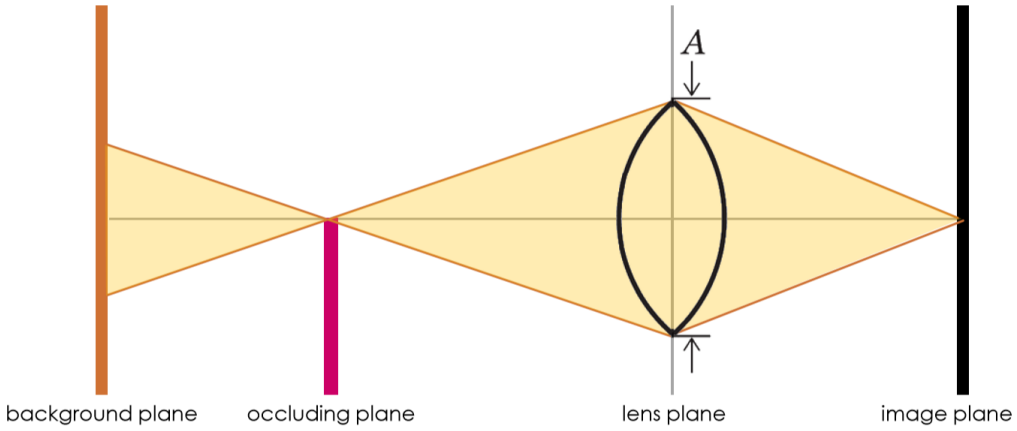
Source: Zannoli et al. (2016)

When focus is switched to the background, the situation is shown in Figure 2.20c. Now, a single point on the background is reflecting several rays that form a single point on the image plane. Yet, at the same time, a finite area on the occluding plane also reflects several rays that contribute to the formation of that very same point. This time, background and foreground colors mix together, and that is the reason for the apparent translucency of the occluding plane. In any case, it is important to notice that, if information from the entire scene is available, all these effects could be simulated using ray tracing.

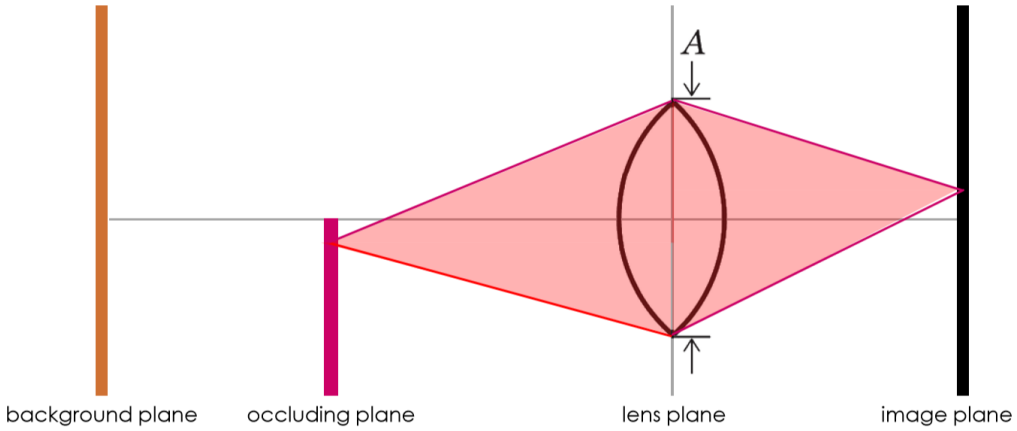
2.9 Summary

This chapter provided some background on fundamental concepts related to both geometric and wave optics that are important for understanding this thesis. This included the Zernike polynomials, which are used to characterize low-order aberrations, and a derivation of the Fourier transform produced by a lens, geared towards the Computational Photography community using the Huygens-Fresnel principle. It also presented an intuition behind partial-occlusion effects, which play a major role on the decisions leading to development of our two different simulation techniques.

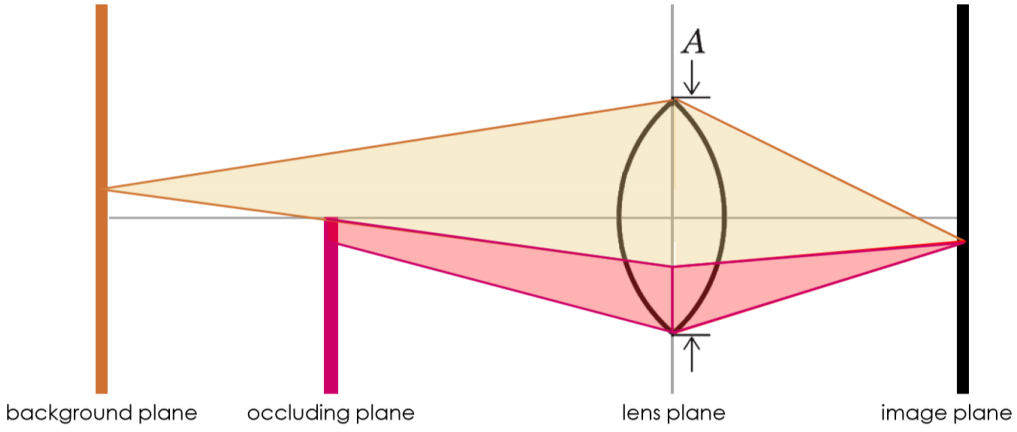
Figure 2.20: Schematic view of partial occlusion under the geometric optics model. This is a side-view representation of the scene shown in Figure 2.19. The background plane represents the sunflower and the foreground occluding plane are the rosebuds. (a) Finite area on the background emitting rays that form a single image point. (b) Single point on the foreground forming a single image point. (c) Both a single point on the background and a finite region on the foreground contribute to the formation of the same image point. (a,b) occur when the foreground plane is in focus, while (c) occurs when the background plane is in focus.



(a)



(b)



(c)

Source: The Authors

3 RELATED WORK

Vision simulation techniques that include optical phenomena like depth of field fall into two major categories: *object-based* and *image-based* algorithms. The former uses computer graphics techniques to render 3D scenes, producing accurate results as in the case of distribution ray tracing (COOK; PORTER; CARPENTER, 1984), and, more recently, in real-time simulation of human vision through eyeglasses (NIEßNER; STURM, 2012), and human eye chromatic aberration (CHOLEWIAK et al., 2017). Image-based techniques manipulate RGB-D data, leading to faster approaches and supporting the use of actual photographs as input. In turn, they tend to suffer from artifacts due to the limited amount of scene information available in a single RGB-D image. Our techniques and all the other methods discussed in this section are image-based approaches. Our new light-gathering tree data structure (described in Chapter 5) significantly minimizes the impact of missing data when producing realistic vision simulations.

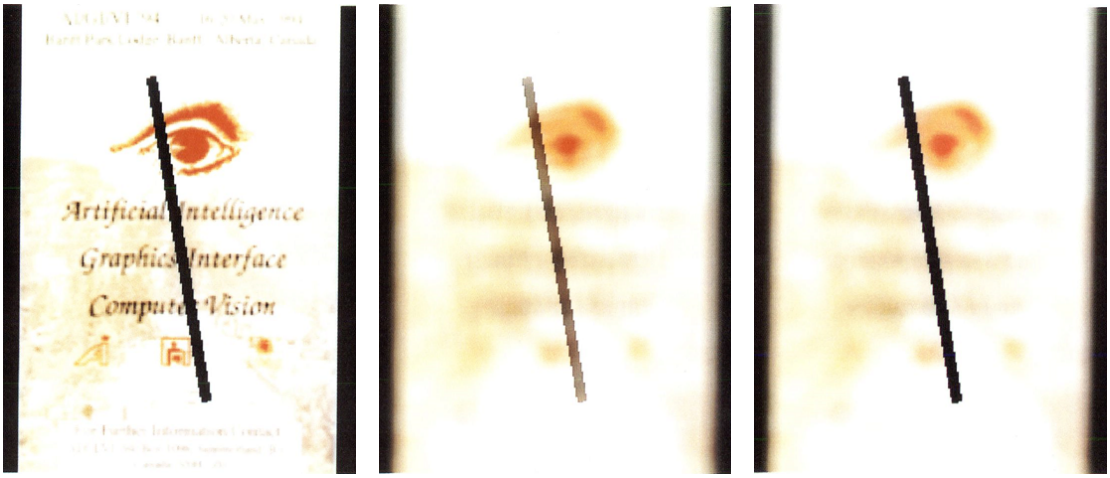
3.1 First techniques

The first developed image-space methods were able to produce a depth-dependent circle of confusion for every pixel with the goal of simulating blurriness (POTMESIL; CHAKRAVARTY, 1982). This technique, however, neglects partial occlusion effects and is prone to artifacts which can be seen on Figure 3.1b.

In an attempt to offer a solution to this problem, Scofield (1992) proposed to classify the scene objects into foreground and background fields, filtering them separately using a PSF appropriate for the distance, and finally compositing the blurred sub-images with alpha blending. This process solves the partial occlusion issue but it only allows a single level of blur per object.

The ray distribution buffer (RDB) approach provided a solution to those errors by averaging the contribution of several rays over a pixel and treating occlusion based on the ray direction (SHINYA, 1994). Before ray tracing starts, the buffers are reset to maximum distance. Every time a ray traced into the scene hits an object, the ray direction and the object's color and distance are recorded in the RDB, provided the distance is less then the previously stored for the respective direction. When tracing is finished, the buffer associated to every pixel is averaged in order to compute the final color (Figure 3.1c).

Figure 3.1: Comparison between linear filtering and ray distribution buffer techniques. (a) Sharp image input. (b) Linear filtering technique applied to input image. (c) Ray distribution buffer algorithm applied to input image.



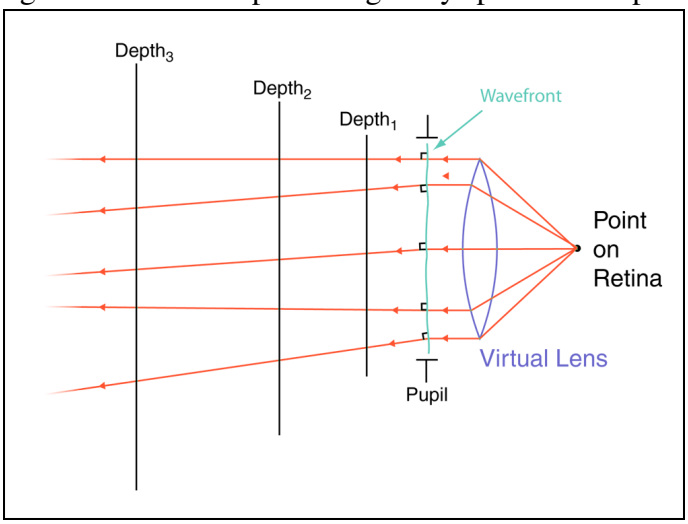
(a) (b) (c)

Source: Shinya (1994)

3.2 Vision-realistic rendering

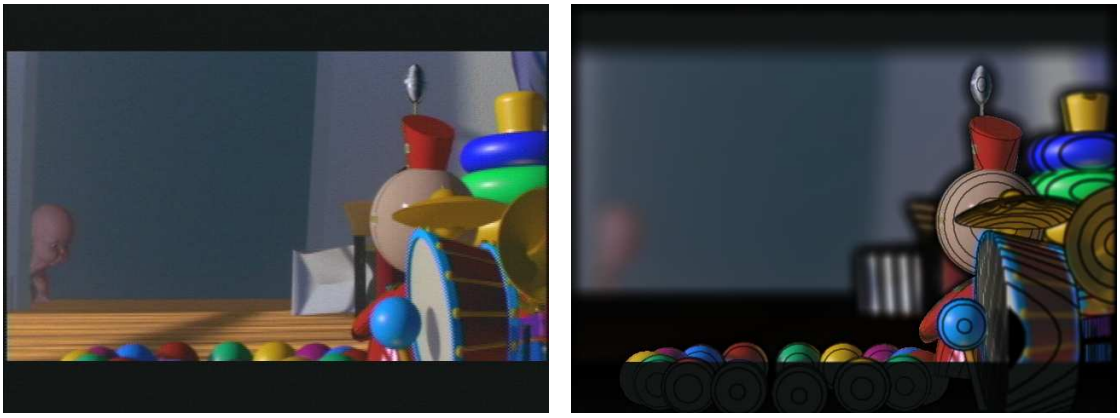
Barsky et al. use optical information from a human subject, supplied by a Shack-Hartmann aberrometer, to model a wavefront that characterizes the subject's visual system (BARSKY et al., 2002; BARSKY et al., 2003). Rays cast from a central point on a virtual retina are bent by a virtual lens and then affected by the subject's wavefront aberration before entering the scene. A set of planes regularly spaced in diopters is placed in the scene (Figure 3.2). Each such plane is associated with a histogram registering the number of rays intersecting it in different rectangular sub-regions. When normalized, each histogram is turned into a so-called *depth point spread function* (DPSF). Given an input RGB-D image, disjoint sub-images are created using pixels whose depth is closest to each plane. The sub-images are then convolved with their respective DPSFs and re-combined with alpha compositing. Unfortunately, simply compositing the convolved sub-images produces undesirable artifacts at the edges of objects (Figure 3.3). The cause of these artifacts is illustrated in 2D in Figure 3.4, where a foreground scene object (red) partially occludes a background object (blue). The scene is projected onto some image plane producing an image I (Figure 3.4a). As the image content is reprojected into the scene (Figure 3.4b), the region of the background object marked with R is missing. As this sub-image is convolved with the DPSF corresponding to its plane, the missing information is improperly treated as zero (black) producing dark regions around edges in the resulting composited image I_c (Figure 3.4c). Note that, given the missing information,

Figure 3.2: A set of planes regularly spaced in diopters.



Source: Barsky (2004)

Figure 3.3: Rendering artifacts produced by the technique of Barsky, when not properly corrected using the object identification solution. (a) Sharp input image. (b) Blurred image with border artifacts. When compared to the sharp input image (a), note how the blurred result (b) presents dark bands separating its parts on different planes.

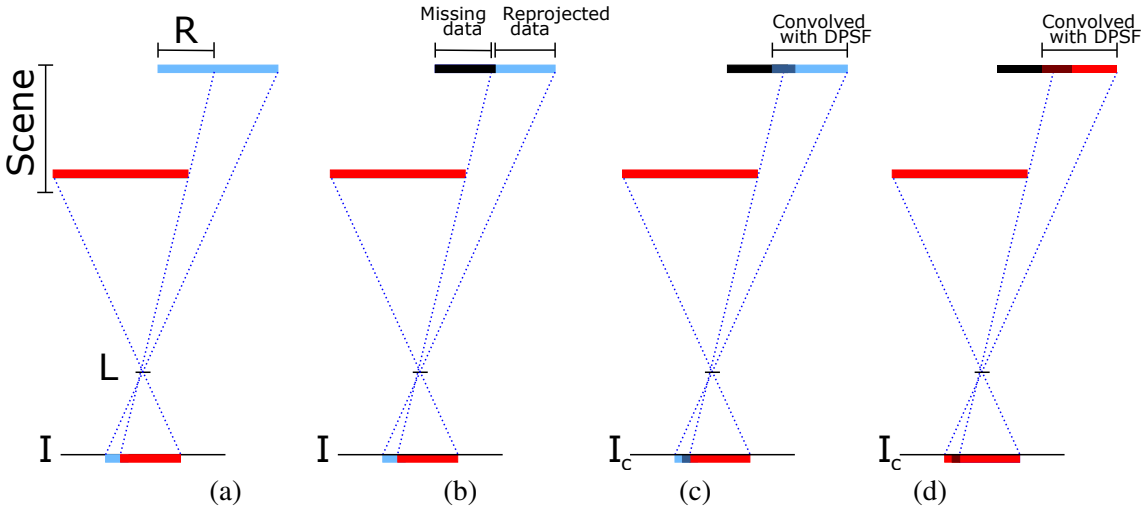


(a) (b)

Source: Barsky et al. (2003)

one should have used normalized convolution (KNUTSSON; WESTIN, 1993) instead of linear convolution. Similar artifacts result when an object spans more than one plane (Figure 3.4d). In this case, an object that should look contiguous presents dark bands separating its parts on different planes. The workaround of Barsky et al. (2002) for the case shown in Figure 3.4c is to convolve the original image with a Gaussian kernel and use some of the resulting pixels to extend the background pixels in occluded regions touched by the DPSF kernel. For the case shown in Figure 3.4d, Barsky (BARSKY, 2004) uses an object identification solution to force pixels belonging to the same object into the same sub-image, regardless of the object's depth span, which results in incorrect results. Barsky et al. (BARSKY et al., 2002; BARSKY, 2004) demonstrated their techniques using syn-

Figure 3.4: A Top-view of a scene containing a red and a blue objects located at two planes. (a) RGB-D image I corresponding to the view of a pinhole camera system at L, with region R occluded. (b) Reprojected scene from I, with missing information shown in black. (c) Planar blue sub-image convolved with its corresponding DPSF, and composited on the final image I_c. (d) Same as (c), but with a single object spanning the two planes.



Source: The Authors

thetic RGB-D images rendered using computer graphics.

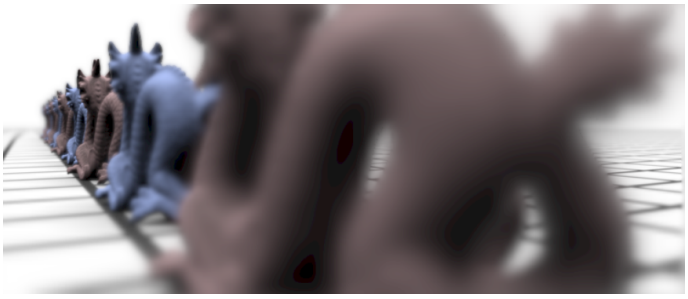
Since one's eye wavefront changes as a function of accommodation (HE; BURNS; MARCOS, 2000), the DPSFs should be re-created in the case of focus change. However, obtaining wavefront measurements for different accommodation conditions is not practical with current aberrometers. Barsky et al. (2002) try to approximate changes in focus by re-indexing the original DPSFs.

3.3 Other techniques

Some techniques related to ours employ pyramidal image processing in order to recover occlusion information (KRAUS; STRENGERT, 2007), with satisfactory results (simulation shown in Figure 3.5), although they involve depth-of-field only, and thus are not applicable to arbitrary aberrations like ours. Other methods use depth peeling in order to access occluded scene information (LEE; EISEMANN; SEIDEL, 2010; SCHEDL; WIMMER, 2012), and so only work with synthetic images or specific scene acquisition methods that register depth and color information for occluded pixels. (results shown in Figures 3.6 and 3.7 respectively).

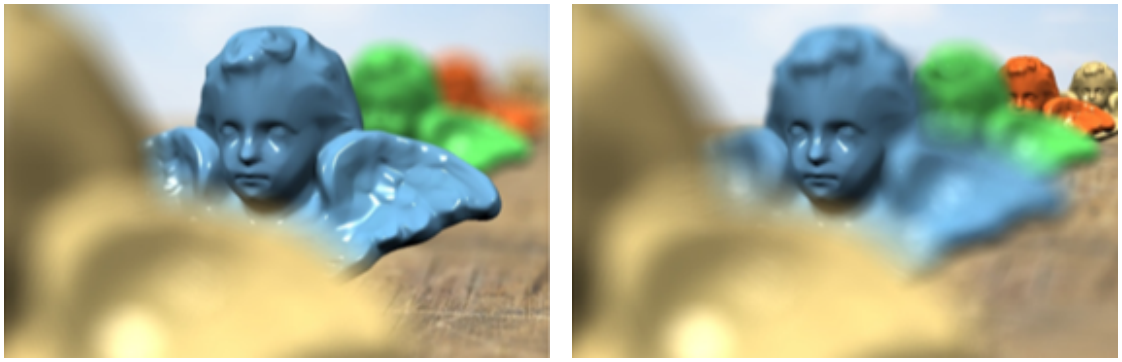
Krueger, Oliveira and Kronbauer (2016) applied Zernike polynomials to recon-

Figure 3.5: Depth-of-field rendering using pyramidal image processing for occluded information recovery.



Source: Kraus and Strengert (2007)

Figure 3.6: Real-time lens blur effects and focus control. (a) Focus on blue statue. (b) Extension to physical model allowing focus on two different distances (blue and orange statues).



(a) (b)

Source: Lee, Eisemann and Seidel (2010)

Figure 3.7: A layered depth-of-field method for solving partial occlusion.

Source: Schedl and Wimmer (2012)

struct the wavefront error resulting from low-order aberrations, using information directly available in spectacles prescriptions rather than relying on data from aberrometers. Constraining the scene to a single plane at a predefined distance, they used Fourier optics to obtain convolution kernels and perform personalized vision simulations of a planar textured surface (*e.g.*, an eye chart). Given a pupil size, the aberrated view was efficiently computed. Defocus simulations were validated against ground-truth data captured with a DSLR camera with low-order aberrations induced by the use of extra lenses placed in front of the original ones.

Xiao et al. (2018) used convolutional neural networks to obtain real-time simulation of depth of field, an important aspect for depth cues in virtual reality applications. Ziegler, Croci and Gross (2008) used several planes to evaluate the complex-valued function of electromagnetic field for light propagation inside a cone of light. Their work considers diffraction and thin lens effects on images, all using Fourier optics.

Previous vision simulation techniques are either limited to a single pre-defined depth (KRUEGER; OLIVEIRA; KRONBAUER, 2016), only handle simulation of positive defocus (XIAO et al., 2018), or require specialized data acquired from aberrometers and lack precision when dealing with partially occluded objects (BARSKY, 2004). In contrast, our more robust approach provides a solution for interactive rendering of realistic vision simulation for arbitrary depths, considers the various kinds of low-order aberrations (*e.g.*, myopia, hyperopia, and astigmatism), and properly models the propagation of light rays around obstacles.

3.4 Summary

In this chapter, we presented the works closely related to our techniques. All of them are image-based algorithms. The techniques presented include depth of field effects of Scofield (1992), ray distribution buffer (SHINYA, 1994), vision realistic rendering (BARSKY et al., 2002; BARSKY et al., 2003), personalized vision simulation of low-order aberrations of the human eye (KRUEGER; OLIVEIRA; KRONBAUER, 2016) usage of convolutional neural networks (XIAO et al., 2018), and simulations using several planes to evaluate the complex-valued function of the electromagnetic field (ZIEGLER; CROCI; GROSS, 2008).

4 SIMULATING LOW-ORDER ABERRATIONS USING WAVE OPTICS

This chapter presents a technique based on Fourier optics for simulating aberration-aware human vision restricted to scenes with a single depth. It is is based on the work done by Krueger, Oliveira and Kronbauer (2016) and also relies on low-order parameters to perform wavefront reconstruction through Zernike polynomials.

The following sections describe the method employed as well as the reasoning behind our algorithm. Section 4.1 presents the rationale behind the technique. Section 4.2 derives the formula used to simulate depth of field for objects on different distances. Section 4.3 briefly mentions the need of applying gamma decoding. Section 4.4 describes the implemented algorithm. Finally, Section 4.5 exposes an important limitation of the technique, which will only be overcome by the approach presented in Chapter 5.

4.1 Rationale of the wave optics approach

The process of producing realistic human vision simulation requires complete real-world information, such as the position and intensity of every point in the scene that contributes for the final image formation. However, obtaining such data in all is complexity is a very difficult task. A possible workaround for this issue is to use sharp images with depth information associated to every pixel; nevertheless, one still has to deal with information that will be missing due to occlusions (such issue is exemplified on Figure 4.1, where occlusion causes trees to cast shadows of missing information over the building facades).

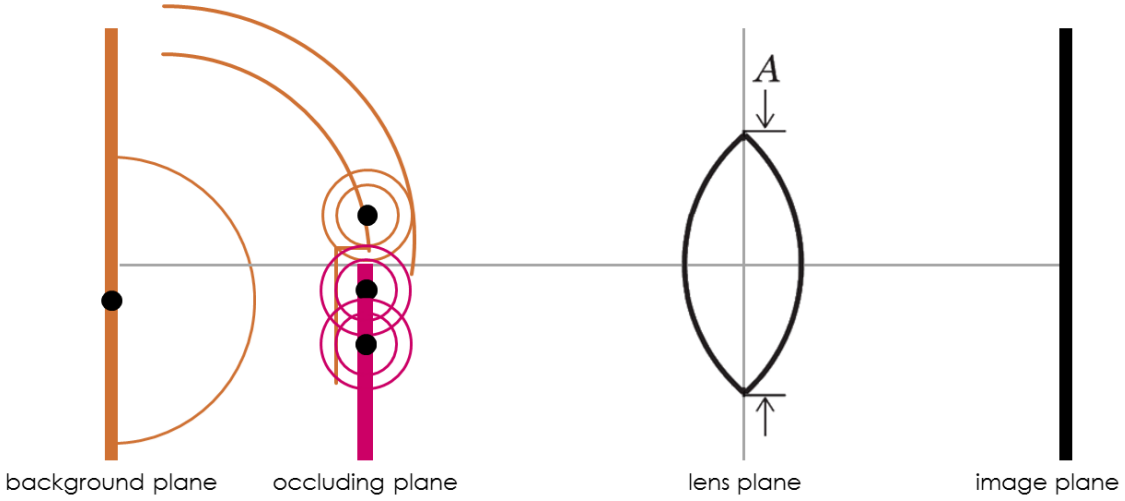
One possible way of simulating vision is using Fourier optics, a formal and rigorous tool. However, handling partial occlusions under such light propagation model is not a trivial task. As shown in Figure 4.2, when a single point on the background plane emits a spherical wave, windowing effects on the occluding plane disturb the wavefront shape, and this adds a level of significant complexity to the simulation. Besides that, the combination of the effects of wavefronts emitted from different scene points, including both occluded regions and occluders, renders the correct prediction of the results an intractable problem for real-time applications. For this reason, we will be limiting our simulations to single-depth scenes and avoid partial occlusions entirely.

The overview of our Fourier-optics single-depth algorithm is shown in Figure 4.3. Given a reconstructed wavefront error W using Zernike polynomials, the generalized

Figure 4.1: Trees occluding building facades on Lidar point cloud. Although this is not the same method we use to acquire scene information, the cause of missing information on occluded regions is the same.

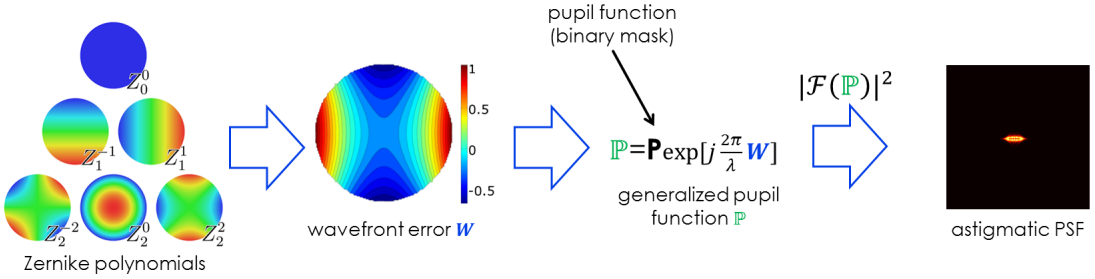
Source: (CURA; PERRET; PAPARODITIS, 2018)

Figure 4.2: Windowing effects in Fourier optics approach. Note how the wavefronts emitted from a partially occluded point on the background plane are affected by the occluder. They are combined with the wavefronts emitted from the occluding plane, and the overall light field in the lens plane results from all these effects, causing the precise simulation outcome to be difficult to compute.



Source: The Authors

Figure 4.3: Wave optics approach overview. (a) Zernike polynomials are used to build the wavefront error W, which is used as an argument for the complex generalized pupil function $\mathbb{P}$; by means of the power spectrum of that function, one obtains the PSF. (b) Simulated result is obtained by the convolution of the sharp input image with the PSF.



(a)

(b)

Source: The Authors

pupil function $\mathbb{P}$ is generated using a phasor, and the PSF is obtained by means of the power spectrum of that function. The PSF is then convolved with a sharp input image, producing the result. In this case, depth information is used to set the appropriate coefficients for the Zernike polynomials, as explained in Section 4.2.

4.2 Object position and accommodation

The key observation for performing the computation of accommodation and depth-aware PSF using Fourier optics is to notice the similarity between the coefficient corresponding to the defocus Zernike polynomial and the accommodation phase transformation. When the aberration transformation presented in Equation (2.35) is used to model defocus (shown in Equation (2.11)), it becomes

$$U_W(x_c, y_c) = e^{jkc_2^0 Z_2^0(x_c, y_c)}. \tag{4.1}$$

The relevant Zernike polynomial listed on the fifth row of Table 2.3 and its corresponding coefficient shown in Equation (2.9) are

$$Z_2^0(x_c, y_c) = \sqrt{3}(2x_c^2 + 2y_c^2 - 1) \text{ and} \tag{4.2}$$

$$c_2^0 = -\frac{S}{4\sqrt{3}}. \tag{4.3}$$

Note that the quadratic radius factor R^2 has been removed from c_2^0 because the input coordinates used here are in spatial domain rather than in the dimensionless unit disk. The cylinder power C has also been removed because we are only modeling defocus in this phase (astigmatism will be simulated in a different stage). By plugging Equation (4.2) and Equation (4.3) into Equation (4.1), it can be written as

$$U_W(x_c, y_c) = e^{jk(-\frac{S}{4\sqrt{3}})\sqrt{3}(2x_c^2+2y_c^2-1)}. \tag{4.4}$$

By disregarding the phase shift and performing some additional simplification, one obtains

$$U_W(x_c, y_c) = e^{-j\frac{k}{2}S(x_c^2+y_c^2)}. \tag{4.5}$$

On the other hand, by rewriting Equation (2.33) assuming that $d \neq h$, the cancellation does not occur anymore, resulting in

$$U_\Sigma(x_c, y_c)L_h(x_c, y_c) = e^{jk(\frac{1}{2d}(x_c^2+y_c^2))}e^{-jk\frac{1}{2h}(x_c^2+y_c^2)} = e^{-j\frac{k}{2}[\frac{d-h}{hd}](x_c^2+y_c^2)}, \tag{4.6}$$

which models a spherical wave propagation with distance d combined with accommodation h. Finally, by making Equation (4.5) to be equal to Equation (4.6) results in

$$S = \frac{d-h}{hd}, \tag{4.7}$$

which means that *we can simulate an object viewed from a distance d, while focusing at a distance h, by setting $S = (d-h)/(hd)$ diopters (an induced defocus).* This result is properly validated in Section 6.4.

4.3 Gamma correction

Human vision is more sensitive to changes in low-light intensity levels than to changes in high-intensity levels. The perception is nonlinear, and can be modeled using a power function (usually $P \propto I^{1/\gamma}$, where P measures subjective perception and I is light intensity). The value for γ is usually 2.2 and this process is called *gamma encoding*. By coincidence, the light intensity emitted by old CRT monitors is also nonlinear; it is a power function applied to the input signal that matches the inverse of human light intensity perception ($I \propto V^{\gamma}$, where V is the signal voltage). This process is called *gamma decoding*, and it also happens on newer monitors, which mimic the CRT behavior for compatibility reasons.

In order to prevent visible artifacts on lower-light intensities and to avoid wasting storage space on high intensities, color intensity data are usually gamma encoded when stored in image files. As a consequence, the consecutive tasks of reading an image file into memory and displaying it do not involve any conversion, since the gamma-encoded image will be automatically gamma-decoded by the video monitor.

However, when the task involves light manipulation before image display, it is essential to consider the nonlinearity of stored data. It is therefore necessary to apply a gamma correction to linearize the data, perform any desired manipulation, and then apply the inverse function before image output. This process is thereby crucial in the case of image blurring, and so it is used on both techniques presented in this thesis.

4.4 Implementation

The simulation of low-order aberrations using Equation (4.6) was developed in MATLAB. It adapts the PSF computation function employed by Krueger, Oliveira and Kronbauer (2016), which in turn is a modified version of the monochromatic PSF generation function described by Dai (2008). After the adaptations, it is able to generate DPSFs for arbitrary low-order aberrations, considering any focus of interest and object distance.

It is important to notice that two features of the human eye present in the original algorithm are not modeled in our simulation: Stiles-Crawford effect (non-constant directional sensitivity of cone cells) and chromatic aberration. The former was removed because it cannot be objectively validated using DSLR cameras. The latter follows the same reasoning, but its removal is also justified by the fact that the human brain adapts to

chromatic aberration (HAY; PICK; ROSSER, 1963), and so there is no need to simulate it for human perception either. We did, however, include compensation for the chromatic aberration caused by external lenses during the experiments. Such formulation is described in Section 6.3.

Our algorithm consists of two phases. In the first phase, the PSF of the optical system is computed according to the objects distances, as well as optical system parameters (*e.g.*, pupil size and focus of interest) and aberration information. The second phase involves blurring the input image using the PSF as a convolution kernel. The photographed object is assumed to be a flat surface orthogonal to the optical axis. This limitation, described on Section 4.5, led to the development of the technique described in Chapter 5.

The input image is also assumed to be in focus (sharp). The need of using a sharp in-focus image as input should be obvious, as any blurring in the final image is expected to be computed by the algorithm.

Algorithm 4.1: Generation and application of PSF using wave optics.

input : Object distance d, gazing focus distance h, sharp image I, lens aberration (S, C, φ), sensor pixel pitch ρ, camera f-number N.
output: Resulting (blurred) image B.

1 $S_f \longleftarrow S + (d-h)/(hd)$;
2 $\alpha \longleftarrow \lambda N/\rho$;
3 $W \longleftarrow \texttt{WavefrontError}(S_f, C, \varphi)$;
4 $\mathbb{P} \longleftarrow \texttt{EnlargeWithZeroFill}(P\exp(jkW), \max(1,\alpha))$;
5 $\mathsf{PSF} \longleftarrow \texttt{Normalize}(\texttt{Resize}(||\mathcal{F}\{\mathbb{P}\}||^2, \min(1,\alpha)))$;
6 $\mathsf{B} \longleftarrow (\mathsf{I}^\gamma * \mathsf{PSF})^{1/\gamma}$;

The procedure shown in Algorithm 4.1 is applied individually to each color channel by setting λ to the corresponding wavelength listed in Equation (2.45). Starting on line 1, defocus is adjusted according to Equation (4.7) to take into account the object distance and gazing focus. Then, on line 2, the scaling factor α shown in Equation (2.43) is computed. On line 3, function `WavefrontError` computes the wavefront aberration corresponding to the input data. This is accomplished by multiplying the Zernike polynomials Z_2^0, Z_2^1 and Z_2^{-1} by the corresponding coefficients indicated in Equations (2.8) to (2.10) and adding them, as shown in Equation (2.12). The generalized pupil function $\mathbb{P}$ is computed on line 4 by inserting the wavefront error function W in the argument of a phasor and multiplying it by the pupil function P. Function `EnlargeWithZeroFill` enlarges the domain of the result with zero-filling up to a factor of α, whenever $\alpha \geq 1$. On line 5, the PSF is calculated as the *power spectrum* of $\mathbb{P}$. The result is downscaled by a factor α, if $\alpha \leq 1$, and finally normalized through function `Normalize`. At last,

on line 6, the output (blurred) image is computed as the convolution of the sharp input image with the PSF. Gamma decoding and re-encoding are properly performed in order to guarantee that the convolution happens in linear color space. After the algorithm finishes, the results for the three color channels are combined forming the final blurred image. Appendix B presents the MATLAB code that implements this simulation process.

4.5 Artifacts due to missing information

Image artifacts at object borders caused by missing information due to occlusions (Figure 3.4c) are a major challenge for the technique presented in this chapter. For this reason, the approach described in this chapter deals only with single-depth images, where no information is missing due to the absence of occlusions. We address the multi-depth issue using a different technique (light-gathering trees), presented in the following chapter.

4.6 Summary

This chapter presented a technique based on Fourier optics for simulating aberrated human vision in scenes restricted to a single depth. We derived a formula used to simulate depth of field for objects on different distances. We have also shown the importance of applying gamma decoding, the internal workings of the algorithm and some important limitations of this technique.

5 SIMULATING LOW-ORDER ABERRATIONS WITH LIGHT-GATHERING TREES

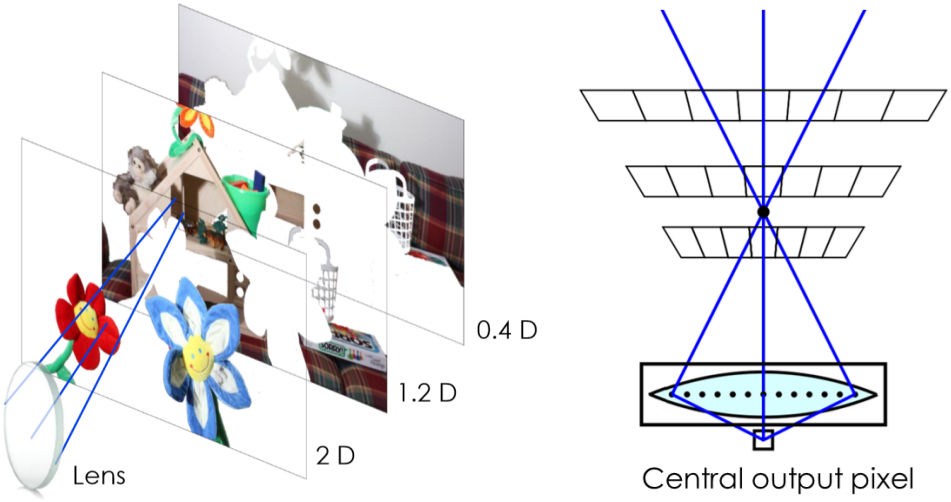
As already stated in Section 2.8, partial occlusion effects emerge naturally from a ray tracing approach, so that is the method that we will be employing in the technique presented here. Nevertheless, we still need to deal with missing information.

In order to be able to perform ray tracing in real-time, some compromises have to made. We use input images with depth information (RGB-D images) and discretize this depth into planes regularly spaced in diopters. This is shown in Figure 5.1a, where the pixels in the image presented in Figure 1.2 are classified according to the closest plane (with closeness function also operating in the dioptric domain).

Rays are cast from the lens plane and into scene (a gathering process). Note however that the ray tracing will only take place for the formation of a single on-axis point on the image plane (central output pixel shown in Figure 5.1b). For the other image points, we reuse the same process adopted for the central pixel. This is supported by the fact that the human eye has an approximately circular retinal region (fovea) with approximately 1 degree in diameter that does not exhibit significant changes in aberrations (BEDGGOOD et al., 2008). Thus, measuring the PSF only for a single on-axis point can be justified by the fact that due to the fovea's small area, the human visual system builds a mental picture of their surroundings by systematically scanning the scene. Therefore, the most prominent perceived aberrations will be those registered at the fovea. As such, we assume that the PSF is the same across the visual field, even though it tends to vary slightly with the direction of the incoming wavefront. This is known as the *isoplanatic assumption* (BEDGGOOD et al., 2008). Other researchers make a similar assumption (KRUEGER; OLIVEIRA; KRONBAUER, 2016), albeit implicitly.

The following sections describe the technique in more details. Section 5.1 presents the rationale behind the technique. Section 5.2 explains the construction of the LGTs. Section 5.3 covers the usage of the LGTs. Section 5.4 describes some important optimizations that allow the technique to run in interactive times. Section 5.5 shows the algorithm used to compute ray directions for low-order aberrations. Finally, Section 5.6 offers a very high-level idea of the algorithm.

Figure 5.1: Plane-discretized scene and the isoplanatic assumption. (a) Pixels are classified into planes according to their dioptric distance to each plane. (b) Rays are only cast for the formation of a single on-axis output pixel, and reused for the formation of all other points (isoplanatic assumption).



(a) (b)

Source: The Authors

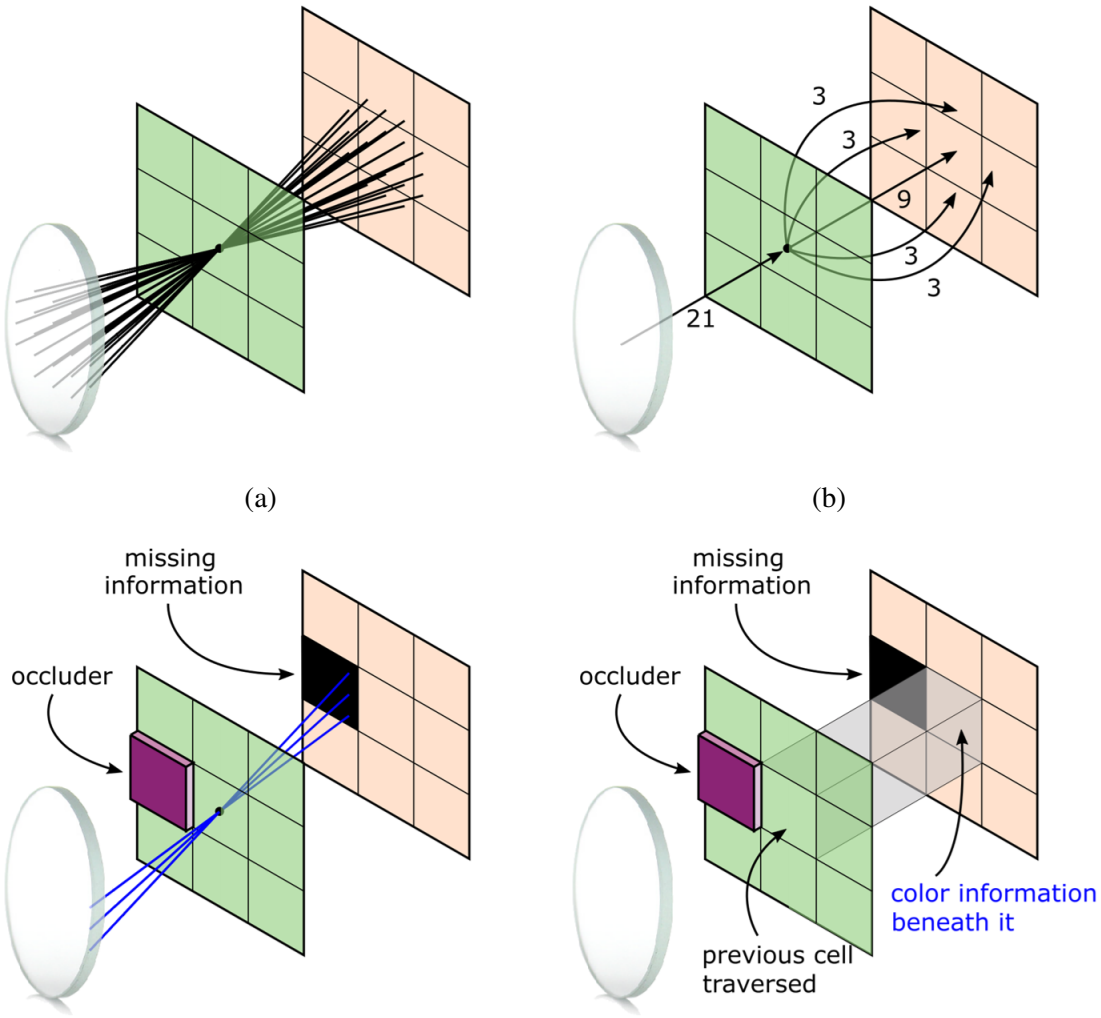
5.1 Rationale of the light-gathering trees approach

The intuition behind using a tree to emulate ray-tracing is shown on the sequence illustrated in Figure 5.2. If a large amount of rays were be cast into the scene as shown in Figure 5.2a, a considerable amount of them would end up traversing the same plane-pixels (or cells). We replace such groups of rays by arrows, as shown in Figure 5.2b, where weights indicate the number of rays represented by each arrow. This hierarchical sequence of arrows define a tree structure (a *light-gathering tree*). It is now evident that ray tracing should be equivalent to traversing this tree it from root to each one of the leaves. Furthermore, this tree can be precomputed and reused for all the other image points due to the isoplanatic assumption.

If an occluder is present at a certain plane and a group of rays would end up on a region behind it where information is missing (Figure 5.2c), dark artifacts would be produced for the reasons previously discussed. We overcome this issue by accessing color information beneath the previous traversed cell (Figure 5.2d) and using that as the missing color. It should be emphasized, however, that although we are illustrating this issue using rays, this process takes place during runtime, when the tree has already been created (it is being used to emulated the ray tracing process).

The tree's nodes are defined based on the scene content, subject's low-order aber-

Figure 5.2: Intuition behind using a tree for performing a gathering process. (a) Several rays cast into scene. (b) Tree structure where weights on arrows indicate number of rays in bundles. (c) Occluder causing missing information. (d) Color information used to fill occluded pixels.

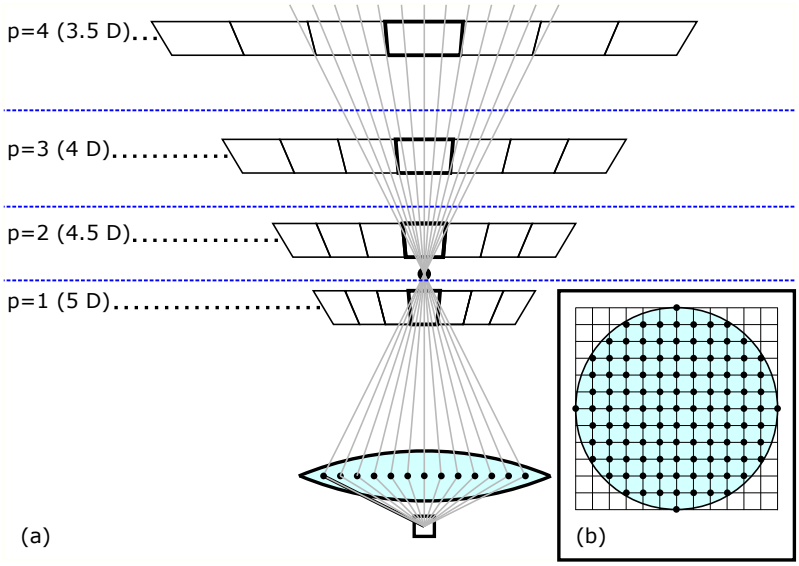


(a) (b)

(c) (d)

Source: The Authors

Figure 5.3: 2-D representation of the light-gathering tree concept. (a) The scene space is discretized using a set of planes placed at distances corresponding to several dioptric powers covering the intended simulation depth range. Scene sampling rays emerging from the sensor pixel are bent by a lens and converge to a focus (black dot) along the optical axis of the lens, and diverge after passing through it. (b) A regular grid defines the sampling positions on the lens.



Source: The Authors

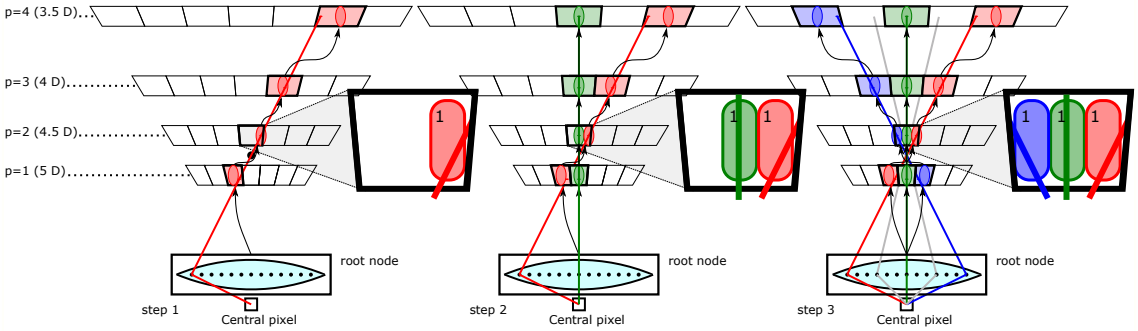
rations, focal distance, and pupil size. Rather than casting rays into the scene in the traditional way during runtime, we traverse the tree structure and sample the data required to simulate the subject's vision.

5.2 Light-gathering tree construction

Human blur discrimination is approximately linear in diopters. Thus, we place planes at distances corresponding to the several dioptric powers along the range of distances to be used for vision simulation (Figure 5.3a). As such, the spacing between adjacent planes increases from planes closer to the viewer to far away ones. Inter-plane boundaries (shown as dashed blue lines in Figure 5.3a) are positioned half-way consecutive planes, subdividing the space and classifying objects as belonging to each plane subspace. Thus, the spacing between them are small for planes closer to the viewer and large for far away ones. The number of planes, as well as the distances of the closest and farthest planes (in diopters) are user-supplied parameters.

A light-gathering tree (LGT) is defined by nearest and farthest plane distances, number of planes, focal distance, pupil size, number of traversal rays, and low-order

Figure 5.4: Three rays cast into a virtual scene (not shown) and the tree nodes built during each step of the process. Colors red, green, and blue are used for discriminating between first, second and third rays (and the nodes created by them), respectively. Arrows indicate parent-child relationship between nodes. A node intersected by n rays coming from different parent nodes is considered as n different nodes, but a node intersected by k rays coming from the same parent node is considered as single node (see Figure 5.8). The value inside each node indicates the number of intercepted rays. The root note is shown as a black rectangle containing a light-blue lens.



Source: The Authors

aberration parameters (S, C, φ). Thus, for a given subject the nodes of an LGT may need to be updated when the focal distance changes, as this may lead to a change in accommodation (*i.e.*, a change in the dioptric power of the crystalline lens). Updating the nodes of an LGT can be done at interactive rates. For instance, for a typical LGT with 14 planes and a million rays it takes approximately one second on a Core i5 2.8 GHz CPU using nonoptimized C# code.

A grid of cells is laid over each plane, representing a bijection between cells and input image pixels (Figure 5.3a). Each plane corresponds to a level of the LGT. Cells crossed by the optical axis, indicated by a thick outline, are called *center cells*. Figure 5.3b shows a regularly spaced grid over a disk representing the subject's pupil. The grid crossings are the starting positions of rays traced into the scene towards a point along the optical axis. In Figure 5.3 such point is indicated by a large dot between planes 1 and 2.

The process of building an LGT is illustrated in Figure 5.4, where only three rays are shown for simplicity. We call *tree width* the maximum number of plane cells spanned horizontally or vertically by any level of a given LGT. Thus, the width of the LGT shown in Figure 5.4 is 5. When the first ray is cast (step 1), one tree node is created inside each cell traversed by the ray. Nodes are all linked from parent to child. Each node stores the following information: number of intercepted rays, cell position, and parent node's cell

position. All positions are stored as 2D coordinates relative to the center cell. Note that the root node is on the pupil plane. In the example shown in Figure 5.4, every node is traversed by a single ray, causing the ray count on all nodes to be one. Figure 5.8 depicts a situation where a node is intercepted by three rays, all from the same parent node, leading to node reuse.

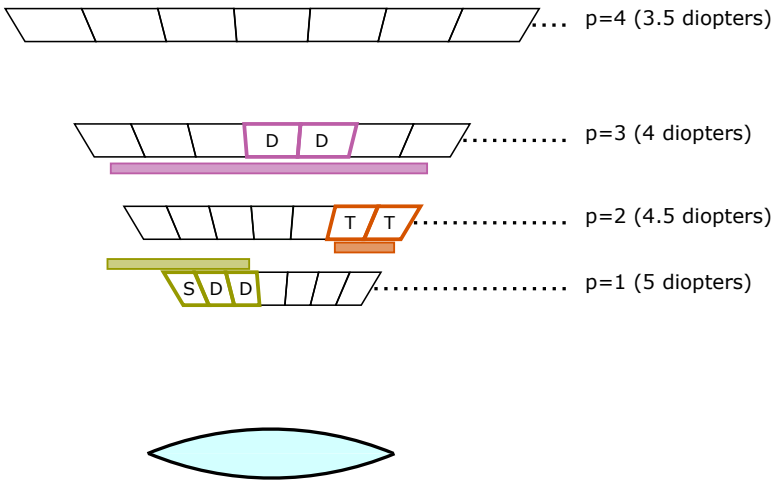
The intended simulated blurred image results from the gathering process described in Section 5.3, where the contributions of each portion of the scene hit by rays are integrated into the final color for the output pixels.

5.3 Light-gathering tree usage

The use of an LGT requires three supporting maps, all with the same dimensions as the input image: a *plane index* (PI) map, a *nearest potentially reachable plane* (NPRP) map, and a *farthest potentially reachable plane* (FPRP) map. The PI map stores, for each pixel of the input RGB-D image, the index of the closest scene-discretization plane to that pixel. Both NPRP and FPRP maps are constructed from the PI map, by considering at each cell of the PI map a neighborhood of size LGT width $\times$ LGT width. The NPRP map stores in each of its cells the smallest plane index in the neighborhood centered at its corresponding cell in the PI map. Likewise, the FPRP map stores in each of its cells the largest plane index in its corresponding neighborhood in the PI map. When considering neighborhoods on the PI map, only cells inside the map are considered. Figure 5.6 shows examples of PI, NPRP, and FPRP maps computed for a simple scene containing three objects (color rectangles) shown in Figure 5.5, whose LGT width is 5.

The usage of the LGT constructed for the scene in Figure 5.5 is illustrated in Figure 5.7. By following the tree from its root towards the leaves, one emulates a beam of rays cast into the scene. Step 1 is performed when the leftmost branch is taken. The first (leftmost) red node, corresponding to plane $p = 1$, indicates that the pixel immediately to the left of the center cell should be analyzed (Figures 5.7a and 5.7b). The corresponding (third) entry on the plane index map (Figure 5.6a) shows the presence of an object on plane $p = 1$, issuing a stop condition because the ray has hit an obstacle. The pixel's output color is updated by accumulating the color stored in the input image pixel indicated by the current node multiplied by some weight (the reciprocal of the number of rays leaving the parent node) (Figure 5.7b). In this example, the parent node is the root node from which three rays have been traced. Thus, each of these rays has a weight of $1/3$.

Figure 5.5: Top-view of a simple scene with three objects (blocks). Pixels are colored according to the object nearest to them. Labels on the pixels indicate the optimization to be applied: Two-plane (T), single-plane (S), or default algorithm (D).



Source: The Authors

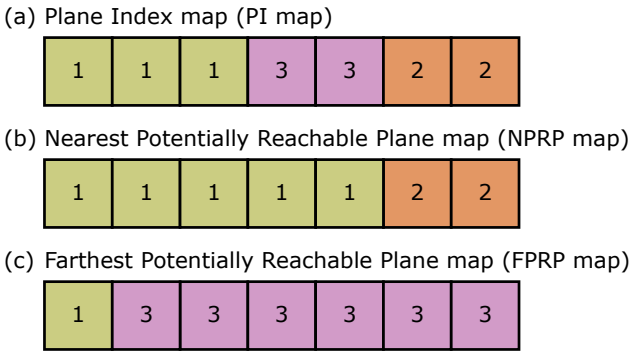
Step 2 corresponds to a beam of rays cast through the center cell, which is accomplished by following the green branch of the LGT (Figures 5.7a and 5.7c). The stop condition will only be met at the third level of the tree, when it is verified that the number on the fourth entry of the plane index map (Figure 5.6a) is equal to the current plane ($p = 3$). Once again, the colors of the reached pixels in the image are averaged, weighted, and added to the color of the output pixel.

Step 3 is defined by the blue nodes (Figures 5.7a and 5.7d). Similarly to the previous step, the nodes are followed until the third level. However, this time the value in the third entry of the plane index map shows "1", which is closer to the observer (root node) than the current plane ($p = 3$). This indicates that information about the exact hit location is missing, and it is the source of partial occlusion artifacts common in image-based vision simulation methods discussed in Chapter 3. Our technique addresses this situation by using the parent node's position (fourth element of the plane index map) instead, resulting in a pink color contribution. That is the reason why we store the parent's index in each node. As in the previous steps, the returned color is multiplied by the current node's weight (1/3) and accumulated into the final pixel color (Figure 5.7d).

5.4 Runtime optimizations

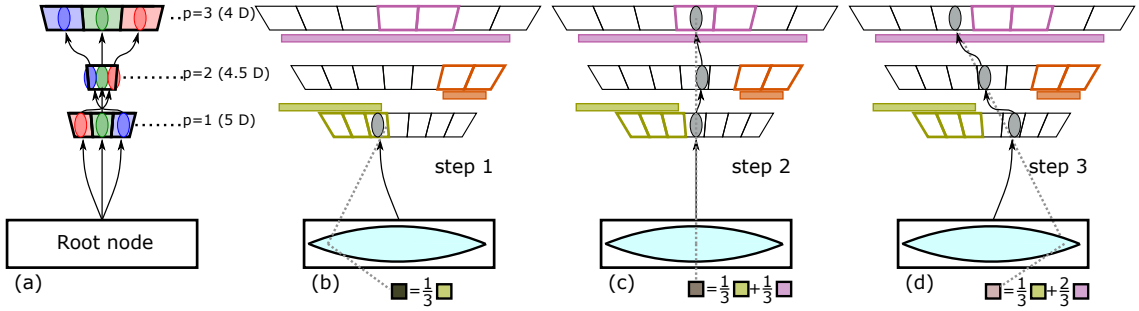
The algorithm described in Section 5.3 is adequate for modeling the actual spreading of the light beams over the scene, but the number of nodes traversed can be very large

Figure 5.6: Maps used for defining which planes to look at during LGT traversal. They have the same resolution as the input image and guide the traversal of the associated LGT. This example was created for a hypothetical 7-pixel image representing the scene depicted in Figure 5.5. (a) Map showing which plane each pixel belongs to. (b) and (c) Maps used to decide which (optimized) algorithm to use on each input pixel.



Source: The Authors

Figure 5.7: Light-gathering tree usage example. (a) The tree used in the example. (b,c,d) The three steps taken to fully produce the output pixel. Each one of them will return a color in the scene (read from the input color image), multiplied by a weight. The final output color, shown in (c), is the color that will be shown in the final blurred image.

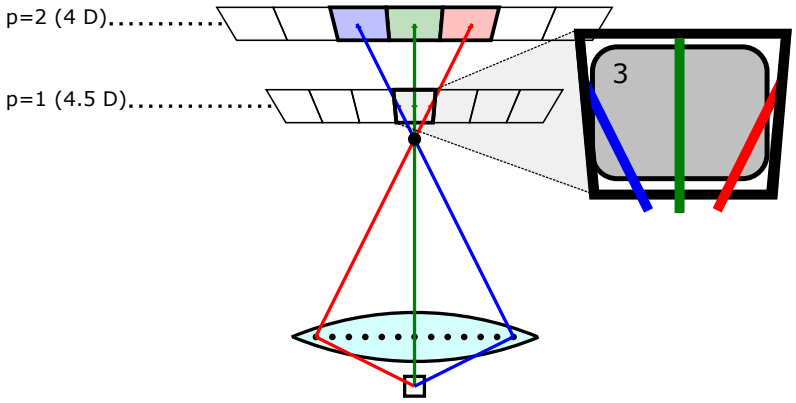


Source: The Authors

depending on the tree configuration. It is possible to significantly reduce the number of LGT nodes that need to be visited for a given ray by considering only a subset of the LGT layers. Combining a complete LGT (*i.e.*, a tree built with all planes/layers) with smaller two-layer trees built for pairs of adjacent planes, one can perform vision simulation in real-time. An LGT considering only adjacent planes a and b is called an a-b tree. Figure 5.8 shows an example of a 1-2 tree. Once an LGT is created, a-b trees are automatically created for each pair of adjacent planes. During runtime, when computing the color of a given pixel of the output image, if its nearest and farthest intersected planes (stored in the NPRP and FPRP maps) are adjacent, we select the corresponding a-b tree instead of the full tree to speed up the simulation process.

If the closest and farthest planes are the same, as shown in the first entry of the

Figure 5.8: Example of a 1-2 tree. Its central cell (node) is intercepted by several rays, all from the same parent node.



Source: The Authors

maps in Figures 5.6b and 5.6c, one can apply an even further optimization and get rid of the trees entirely. In this case, we use the (normalized) number of rays that hit each LGT cell on the plane as the weight to be multiplied by the respective pixel color. The output pixel color is obtained by summing all these contributions. This is equivalent to locally applying a convolution kernel.

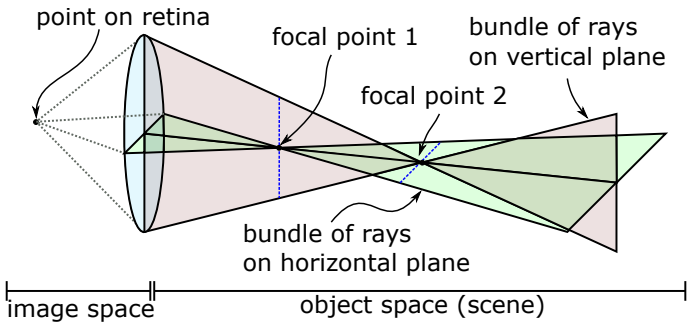
We call these optimizations *two-plane* and *single-plane* optimizations, respectively. In Figure 5.5, pixels where two-plane optimization is applied are indicated by the letter T, and those where single-plane optimization applies are indicated by the letter S. The ones for which the complete LGT is applied are marked with D.

5.5 Determining ray directions

So far, we have shown simplified examples, using a single point of interest for eye accommodation. The actual algorithm supports two focal points in order to simulate astigmatism (Figure 5.9). We adopt a left-handed coordinate system, as shown in Figure 5.10. The cylinder axis lies on the xy plane making an angle φ with the horizontal axis (Figure 5.10d). For each ray, its starting position (x_s,y_s,0) and direction (Δ_x,Δ_y,1) are obtained using Equations (5.1) and (5.2):

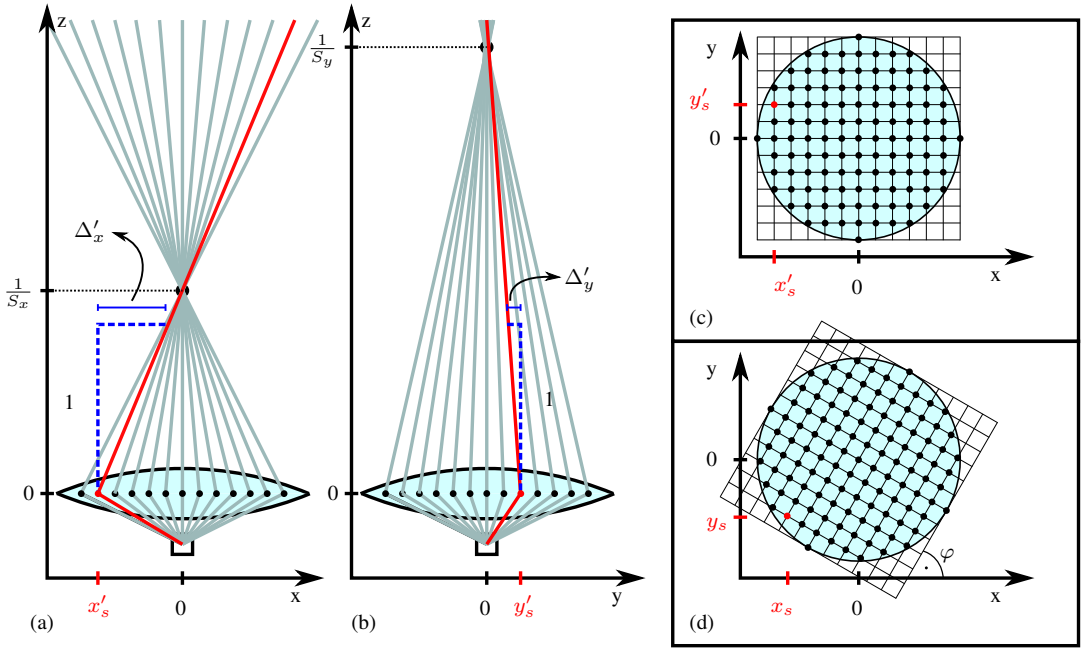
$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} x'_s \\ y'_s \end{bmatrix}, \tag{5.1}$$

Figure 5.9: Illustration of an astigmatic optical system with $\varphi = 0°$. A bundle of rays on the horizontal plane is related to focal point 1 at distance $1/S$, while a bundle of rays on the vertical plane is related to focal point 2 at distance $1/(S + C)$.



Source: The Authors

Figure 5.10: Ray casting for an astigmatic optical system using a left-handed coordinate system. Each ray is defined by a starting position (x_s,y_s,0) and direction (Δ_x,Δ_y,1), given by Equations (5.1) and (5.2), respectively. (a) Top view (xz plane) of the scene. (b) Side view (yz plane) of the scene. (c) Back view (xy plane) of the scene when $\varphi = 0$. (d) Back view (xy plane) of the scene when $\varphi \neq 0$.



Source: The Authors

$$\begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} \Delta'_x \\ \Delta'_y \end{bmatrix}, \tag{5.2}$$

where (x'_s,y'_s,0) and $(\Delta'_x, \Delta'_y, 1)$ correspond to the ray starting position and direction for the case of an axis-aligned grid (*i.e.*, $\varphi = 0°$) (Figure 5.10c). The values of Δ'_x and Δ'_y are computed as

$$\begin{aligned} \Delta'_x &= -\frac{x'_s}{1/S} = -x'_s \cdot S \text{ and} \\ \Delta'_y &= -\frac{y'_s}{1/S_{pC}} = -y'_s \cdot S_{pC}, \end{aligned} \tag{5.3}$$

where $S_{pC} = S + C$ is the sum of the spherical (S) and cylindrical (C) powers.

5.6 General LGT algorithm

The general algorithm used for simulation using LGTs consists of two different stages. During the offline stage, the following sequence of steps is taken for each possible tree (considering all $k-$plane optimizations):

- Trace rays from pupil plane into virtual scene.
- Group rays into weighted beams.
- Build a tree from beams (root node on lens plane).

The runtime stage, for instance, executes these two main steps for each output pixel:

- Choose tree based on planes nearby screen area.
- Follow every path from root to leaf.

The collection of output pixels will then form the result of the simulation.

5.7 Summary

This chapter presented our second and more robust technique, used to simulate accommodation and low-order aberrations of the human eye in scenes with multiple depths and partial occlusions. We showed the definition, construction and usage of a tree data

structure that allows us to visualize the scene under low-order aberrations at interactive rates. We also showed how we provide a solution for artifacts (due to missing information) by caching the parent's tree node.

6 EXPERIMENTS AND RESULTS

This chapter presents experimental results related to the theoretical aspects introduced in Chapter 2, as well as validation and results of the two techniques presented in this work (Chapters 4 and 5). All pictures in the experiments were captured using a Canon Rebel T6 DSLR camera.

Section 6.1 shows a comparison of the Airy pattern predicted by the theory and the one obtained with the camera. Section 6.2 describes the support device we have crafted to use in the camera experiments on the subsequent tests. Section 6.3 discusses some adjustments required to allow comparisons of our results with ground-truth images. Section 6.4 presents validation for the formula deduced in Section 4.2 relating objection position, gazing focus and defocus aberration, achieving structural similarity (SSIM) values above 0.93 and peak signal-to-noise ratio (PSNR) above 31.0. Section 6.5 presents a quantitative evaluation of low-order aberrations produced by both techniques (implemented as a MATLAB script) using SSIM and PSNR metrics. Section 6.6 shows the results of the LGT technique (implemented as a Unity compute shader) applied to real scenes with various depths, acquired from a publicly available dataset. Section 6.7 addresses some of the limitations and issues in our techniques.

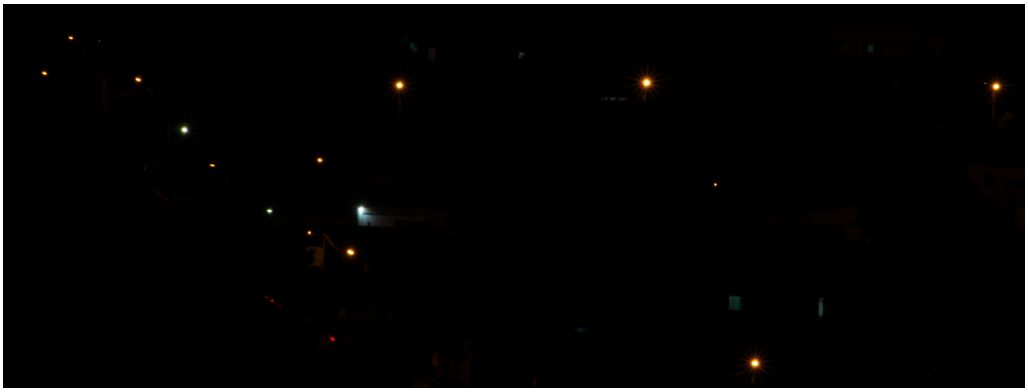
The results of these experiments show that both techniques produce quite realistic simulations of accommodation and low-order aberrations, achieving SSIM values above 0.94 and PSNR above 32.0 in all objective evaluations.

6.1 Airy pattern validation

In this section, we perform a comparison of the Airy pattern predicted by theory (Section 2.7.4) with the PSF of a point source captured by the camera, in order to verify the correctness of the scaling factor α derived in Equation (2.43). We have employed the method shown in Trantham and Reece (2015) to measure the Airy pattern generated by some far away mercury-vapor and sodium-vapor street lights visible at night. Given the large distance from the lights to the camera, they can be considered as being at infinity, and treated as point-like light sources.

For comparison, we have generated the PSF for infinitely distant point light sources by modeling a plane wavefront, using the strongest peak on the line spectrum of both elements in the green region as their corresponding wavelengths. Due to its higher reso-

Figure 6.1: Night time picture of town landscape with mercury-vapor and sodium-vapor lamps displaying the Airy pattern. Picture is down-cropped to 4,549×1,705 pixels, followed by 25% downscaling. Camera settings: ISO 100, exposure 15 s, f = 300 mm, $f/45$.



Source: The Authors

Table 6.1: Comparison of photographed, simulated and computed diameter of Airy disk.

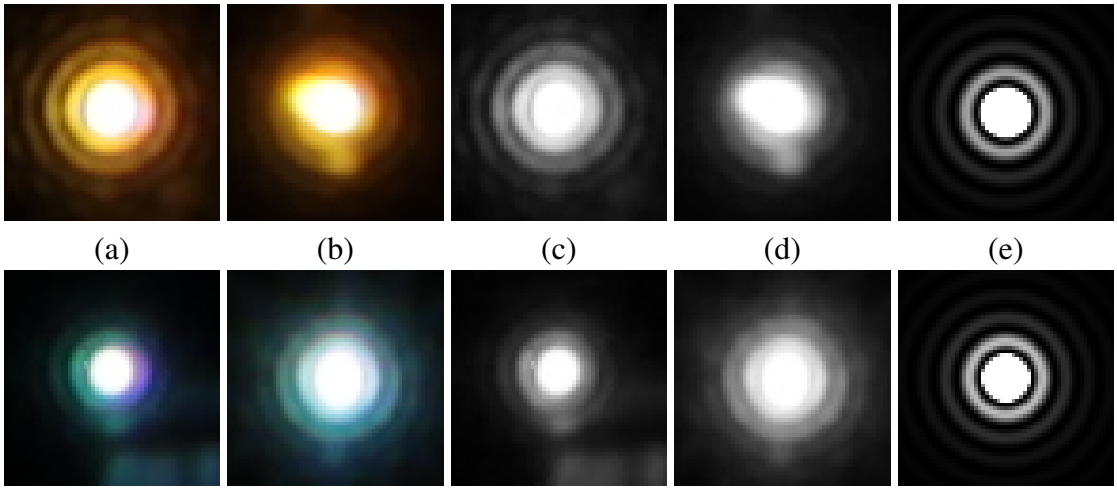
		Airy disk diameter (px)		
Element	λ (nm)	*Picture*	*Simulation*	*Predicted*
Sodium	589	16 ± 0.5	16 ± 0.5	15.509
Mercury	546.1	14 ± 0.5	14 ± 0.5	14.379

Source: The Authors

lution on the Bayer filter pattern, only the green color channel from the pictures is used. The wavelengths values used to simulate the corresponding PSFs are λ = 589 nm and λ = 546.1 nm (JENKINS; WHITE, 2001), since they are the most prominent lines found in the frequency spectrum of, respectively, Sodium-vapor and Mercury-vapor lamps (Figure 6.2).

The camera's RAW file metadata informs that uncropped pictures are 5,344 pixels wide. Along with the known sensor width of 22.3 mm (obtained from the camera's specifications), this results in a pixel pitch of 4.17 μm. Applying the pixel pitch to Equation (2.43), it is possible to compute the factor to be used to scale the PSF obtained through the Fourier optics approach, and compare it to the expected value given by Equation (2.47). The comparison of the generated PSFs with ground-truth pictures is shown on Figure 6.2 and objectively measured on Table 6.1, demonstrating that the radii of both the photographed disk and the one generated through Fourier optics using the scaling factor are in agreement with the radius expected by Equation (2.47) within a margin of error of $\pm$0.5 pixel.

Figure 6.2: Details of Airy pattern seen on the PSF of point light source shown in Figure 6.1. (a) and (b) Sodium-vapor lamps, with respective green channel isolated in (c) and (d), respectively. (e) Our simulation of point-source impulse response for $\lambda = 589$ nm. (f) and (g) Mercury-vapor lamps, with respective green channel isolated (h) and (i), respectively. (j) Our simulation of point-source impulse response for $\lambda = 546.1$ nm.



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Source: The Authors

6.2 Camera holding device

With the goal of ensuring precise lens positioning during the tests, a support device was crafted out of polystyrene and medium-density fiberboard (MDF) sheets, with a tight-fitting area for inserting the camera. A *tripod quick release plate* was attached to the camera to ensure steadiness. The device's surface, made out of MDF, was designed to fit the plate without gaps, in an effort to prevent looseness and conserve camera positioning between shots even when the camera is removed and later reattached to the device.

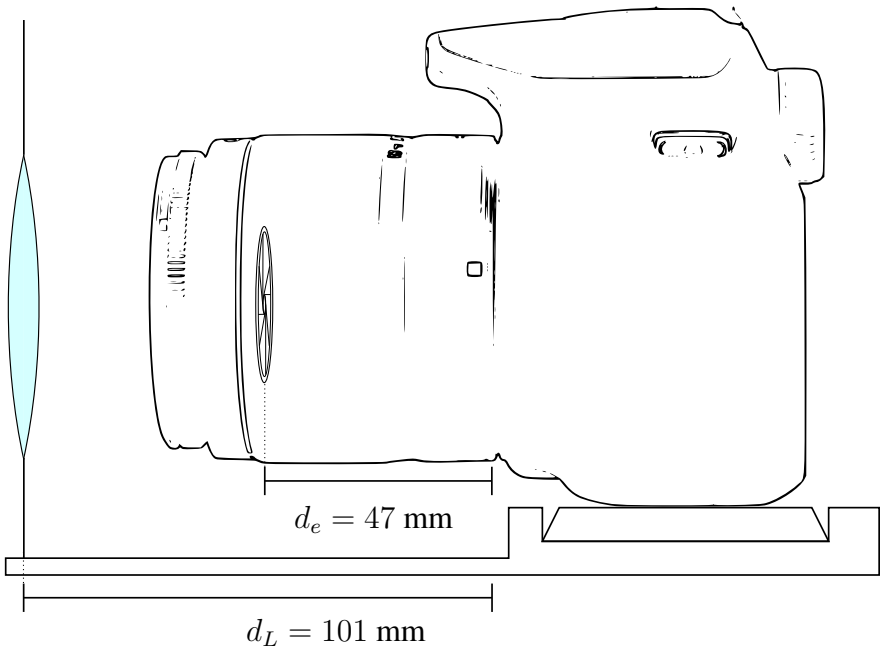
For each lens, a cardboard sheet was cut into a rectangular card and a round aperture was carved out from its middle. The lens was glued to the aperture. A slot was cut out of the polystyrene structure, right in front of the device and orthogonal to the camera optical axis, to be used for inserting the lens cards. The distance from the camera lens mount to the lens slot was set to 101 mm (Figure 6.4).

Figure 6.3: Camera mounted on custom supporting device. (a) Back view of device with camera pointing to Snellen chart. (b) Lens inserted into the device slot.

(a) (b)

Source: The Authors

Figure 6.4: Camera holding device scheme. Distance from the entrance pupil to the camera zoom lens fixation base (d_e) was measured to be 47 mm. Distance from the external lens to the camera zoom lens fixation base was fixed at d_L = 101 mm.



Source: The Authors

6.3 Optical power and image adjustments

Given the distance v from the camera lens to the external lens, known as *vertex distance*, the resulting anisotropic magnification due to astigmatism is given by

$$M_{\varphi} = \frac{1}{1 - vS} \quad \text{and} \quad M_{\varphi^{\perp}} = \frac{1}{1 - vS_{pC}}, \tag{6.1}$$

where M_{φ} and $M_{\varphi^{\perp}}$ are, respectively, the magnification factors along the directions that make angles φ and $\varphi + 90°$ with the horizontal axis, and S and S_{pC} are the spherical and the sum of spherical and cylindrical powers as discussed in Section 5.5. In the absence of astigmatism, the magnification is isotropic, with $M_{\varphi} = M_{\varphi^{\perp}}$. The *effective optical power* is then obtained as $S'_{\varphi} = SM_{\varphi}$ and $S'_{\varphi^{\perp}} = SM_{\varphi^{\perp}}$.

Image magnification may introduce incorrect values due to interpolation. Thus, when comparing our results to ground-truth, rather than magnifying a smaller dimension to match a larger, we downscale the larger to match the smaller. One should note, however, that magnification is a function of vertex distance and vanishes when $v = 0$. Thus, magnification and its compensation have only been used for the sake of the validation experiment that uses an external lens. This is not a stage of the techniques themselves, which are geared towards the simulation of scenes as seen by a naked eye.

Likewise, brightness adjustment is required to compensate for some amount of light that is reflected/absorbed by the extra lens, effectively not reaching the camera sensor. The images captured with the extra lens tend to be darker than ones captured without it. To perform brightness adjustment, for each different external lens, a small white patch is taken from the same area in images captured with and without the additional lens. The ratio between the average intensities from the darker and brighter patches was used to modulate the images simulated with our technique, making them exhibit brightness similar to the ground-truth images. This is important when performing quantitative comparisons using metrics such as SSIM and PSNR (Figures 6.5 to 6.8 and Tables 6.2 to 6.5).

Chromatic aberration due to the external lens is given by

$$S''_{\varphi c} = \frac{S'_{\varphi}(\mu_c - 1)}{\mu_y - 1} \quad \text{and} \quad S''_{\varphi^{\perp} c} = \frac{S'_{\varphi^{\perp}}(\mu_c - 1)}{\mu_y - 1},$$

where $S''_{\varphi c}$ and $S''_{\varphi^{\perp} c}$ are the resulting aberrated powers (in diopters) for wavelength λ_c, S'_{φ} and $S'_{\varphi^{\perp}}$ are the effective optical powers due to the vertex distance v, μ_c is the lens refractive index for wavelength λ_c, and $\mu_y = 1.5085$ is the reference refractive index, which is

usually on the yellow region of the spectrum. For our experiments, we used the following indices of refraction for red, green, and blue, respectively: $\mu_r = 1.4998$, $\mu_g = 1.5085$, and $\mu_b = 1.5152$, which were obtained from an online refractive index database (POLYANSKIY, 2019), and correspond to wavelengths λ_r = 700 nm, λ_g = 510 nm, and λ_b = 440 nm (Equation (2.45)).

6.4 Object defocus compensation using an extra lens

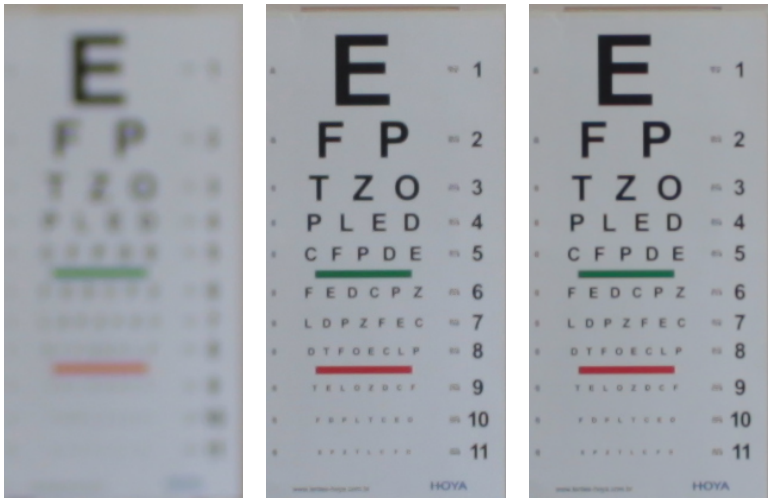
This experiment shows that observing an object at a certain distance d while simultaneously focusing at a distance h can be simulated by a single lens with optical power given by Equation (4.7). On each of the four performed tests, we took a picture of an eye chart positioned in front of the camera with distance h. We used the camera auto focus to make sure it was focusing exactly on the eye chart. We locked the focus and took a sharp picture of the eye chart. We then moved the chart to a distance d from the camera and took another picture, which appears out of focus. Then, we put an extra lens in front of the camera in order to allow it to focus again on the chart at distance d from the camera, and took another picture.

The results of this experiment are shown in Figures 6.5 to 6.8. The SSIM and PSNR values shown in the captions measure the similarity between lens-corrected defocus images (column b) and ground-truth in-focus images (column c). In all cases, they indicate quantitatively that the deduced formula is indeed correct. The small qualitative divergence (slight defocus) observed in Figure 6.8b is due to imprecision on measurements occurring in a low-tolerance and low depth-of-field region in the vicinity of the external lens.

6.5 Quantitative evaluation

For the quantitative evaluation, we took a set of pictures from two eye charts using the DSLR camera with extra lenses placed 54 mm in front of the camera's original lens to induce low-order aberrations. The charts were placed 7.0 m (approximately 23.96 feet) away from the camera. We used external lenses with various spherical (S) and cylindrical (C) powers, as well as astigmatism axes (Tables 6.2 to 6.4). The acquired ground-truth images (JPEG, 5,184×3,456 pixels) were compared against our simulations for the corresponding low-order aberrations using the SSIM and PSNR objective metrics. Since the

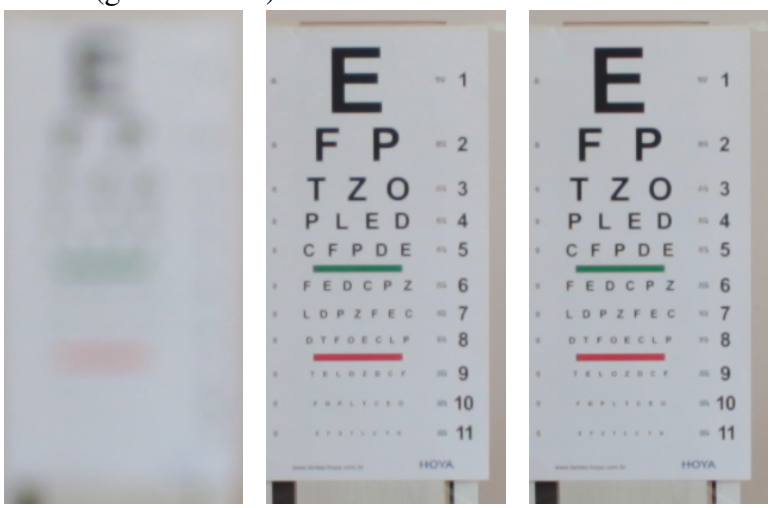
Figure 6.5: Induced myopia corrected with extra lens. Object distance d = 6,047 mm. PSNR = 37.2334 and SSIM = 0.9664. (a) Camera focus at f = 943 mm (myopia). (b) Camera focus at f = 943 mm. Hyperopic extra lens with S = -1.00 D. (c) Camera focus at $f = d$ = 6,047 mm (ground-truth).



(a) (b) (c)

Source: The Authors

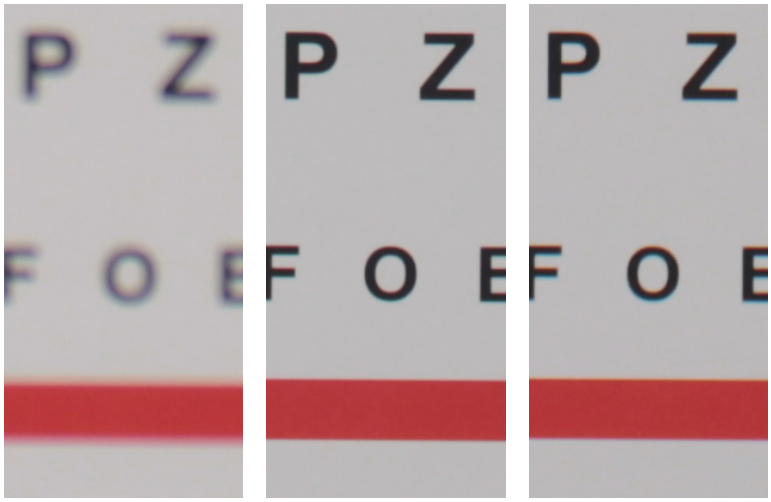
Figure 6.6: Induced myopia corrected with extra lens. Object distance d = 6,047 mm. PSNR = 37.6544 and SSIM = 0.9616. (a) Camera focus at f = 469 mm (myopia). (b) Camera focus at f = 469 mm. Hyperopic extra lens with S = -2.50 D. (c) Camera focus at $f = d$ = 6,047 mm (ground-truth).



(a) (b) (c)

Source: The Authors

Figure 6.7: Induced hyperopia corrected with extra lens. Object distance d = 864 mm. PSNR = 36.2875 and SSIM = 0.9491. (a) Camera focus at f = 6,047 mm (hyperopia). (b) Camera focus at f = 6,047 mm. Hyperopic extra lens with S = +1.00 D. (c) Camera focus at $f = d$ = 864 mm (ground-truth).



(a) (b) (c)

Source: The Authors

Figure 6.8: Induced hyperopia corrected with extra lens. Object distance d = 414 mm. PSNR = 31.2921 and SSIM = 0.9367. (a) Camera focus at f = 6,047 mm (hyperopia). (b) Camera focus at f = 6,047 mm. Hyperopic extra lens with S = +2.25 D. (c) Camera focus at $f = d$ = 414 mm (ground-truth).

(a) (b) (c)

Source: The Authors

eye charts are planar, these experiments take advantage of the single-plane optimization.

The use of an external lens introduces changes to the camera's optical system, resulting in changes in magnification, brightness, and chromatic aberrations of the captured ground-truth images. Such changes need to be compensated for in our synthesized results using the methods described in Section 6.3 for proper comparisons.

6.5.1 Objective validation

We validated our technique by performing some quantitative evaluation of low-order aberrations (myopia, hyperopia, and astigmatism) with and without considering chromatic aberrations.

Figure 6.9a shows three pictures of an eye chart captured with the following camera settings: ISO 100, exposure 1/40 s, f = 20 mm, and $f/5$. Figure 6.9b shows the pictures in Figure 6.9a after anisotropic minification and brightness correction. Such minification was performed using MATLAB's interpolation function `interp2`. Figure 6.9c shows the ground-truth pictures captured by placing a lens whose parameters are described in the corresponding rows of Tables 6.2 and 6.3. Figures 6.9d and 6.9e show the simulated results produced by both of our techniques (Fourier optics and LGT respectively) using Figure 6.9b as input and not taking chromatic aberration into account. Note how similar they are to the corresponding ground-truth images. Tables 6.2 and 6.3 show SSIM and PSNR values for the results shown in Figures 6.9d and 6.9e respectively. SSIM values are above 0.94 and PSNR values are above 32.4 for all simulated results, both with or without considering chromatic aberration (CA). The metric values obtained when considering CA were just slightly higher than without considering it. The results are visually indistinguishable. Including CA in the simulations does not seem to improve the results to justify its additional computation. Thus, in this thesis, we only show pictures of simulated results without considering CA.

Figure 6.10a shows pictures of an eye chart captured under a shorter exposure. The camera settings are: ISO 100, exposure 1/125 s, f = 18 mm, and $f/5$. Again, the two images were taken by placing extra lenses 54 mm in front of the original camera lens. The spherical (S) and cylindrical (C) powers, as well as the corresponding φ cylinder axis angle of the extra lens used to capture each picture of this experiment are shown in Tables 6.4 and 6.5. Figure 6.10c shows the ground-truth image captured by placing a lens with parameters described in the corresponding row of Tables 6.4 and 6.5. Figures 6.10d

Figure 6.9: Inducing low-order aberrations (hyperopia and astigmatism) by placing external lenses in front of a camera's original lens (v = 54 mm). Camera settings: ISO 100, exposure 1/40 s, f = 20 mm, $f/5$. (a) Picture taken without extra lens. (b) Anisotropic minification and brightness adjustment applied to (a). (c) Ground-truth image obtained with an external lens in front of the camera. (d) Simulated results produced by the Fourier optics technique. (e) Simulated results produced by the LGT technique. Lens' parameters, SSIM and PSNR values for comparison of (c), (d) and (e) are in Tables 6.2 and 6.3.

Source: The Authors

and 6.10e show our simulated results. Again, note how similar they are to the corresponding ground-truth images. Tables 6.4 and 6.5 show the corresponding SSIM and PSNR values, which are higher than 0.98, and 42.2, respectively, indicating strong agreement with the ground-truth.

6.6 Qualitative evaluation

For the qualitative experiments, we use a set of RGB-D images whose depth ranges cover several diopters. Thus, the simulations discussed in the section use combinations of complete LGTs and a-b trees, as well as the process in single-plane optimization. The set of RGB-D images was obtained from a stereo online dataset that offers 23 color

Table 6.2: Extra lens parameters, SSIM and PSNR values for the simulation results of Fourier optics approach in Figure 6.9. Results are shown both with (CA) and without (NCA) considering chromatic aberration. "Row" is the row number in Figure 6.9.

	Lens			*CA*		*NCA*	
Row	*S*	*C*	φ	*SSIM*	*PSNR*	*SSIM*	*PSNR*
1	0 D	-1 D	86°	0.959	34.318	0.959	34.296
2	0 D	-0.5 D	82°	0.944	32.465	0.944	32.445
3	-0.25 D	-2.25 D	69°	0.953	36.500	0.951	36.430

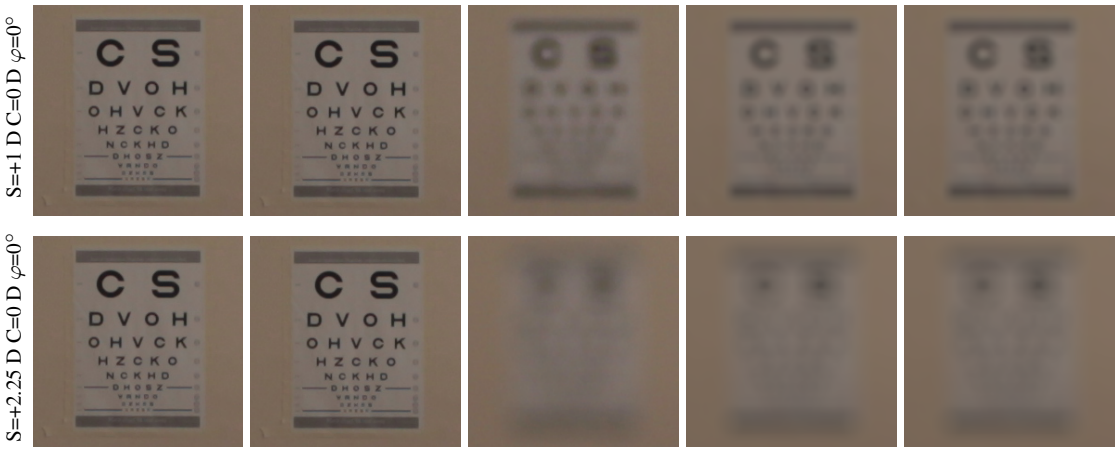
Source: The Authors

Table 6.3: Extra lens parameters, SSIM and PSNR values for the simulation results of LGT approach in Figure 6.9. Results are shown both with (CA) and without (NCA) considering chromatic aberration. "Row" is the row number in Figure 6.9.

	Lens			*CA*		*NCA*	
Row	*S*	*C*	φ	*SSIM*	*PSNR*	*SSIM*	*PSNR*
1	0 D	-1 D	86°	0.959	34.426	0.959	34.401
2	0 D	-0.5 D	82°	0.942	32.434	0.942	32.411
3	-0.25 D	-2.25 D	69°	0.952	36.382	0.950	36.312

Source: The Authors

Figure 6.10: Inducing low-order aberrations (myopia) by placing external lenses in front of a camera's original lens (v = 54 mm). Camera settings: ISO 100, exposure 1/125 s, f = 18 mm, $f/5$. (a) Picture taken without the extra lens. (b) Minification and brightness applied to (a). (c) Ground-truth image obtained by placing an external lens in front of the camera. (d) Simulated results produced by the Fourier optics technique. (e) Simulated results produced by the LGT technique. Lens' parameters, SSIM and PSNR values for comparison of (c), (d) and (e) are in Tables 6.4 and 6.5.



Source: The Authors

Table 6.4: Extra lens parameters and SSIM and PSNR values for the simulation results of Fourier optics approach shown in Figure 6.10. Results are shown both with (CA) and without (NCA) considering chromatic aberration. "Row" is the row number in Figure 6.10.

	Lens			*CA*		*NCA*	
Row	*S*	*C*	φ	*SSIM*	*PSNR*	*SSIM*	*PSNR*
1	+1 D	0 D	0°	0.986	42.013	0.987	42.165
2	+2.25 D	0 D	0°	0.991	42.979	0.991	43.078

Source: The Authors

Table 6.5: Extra lens parameters and SSIM and PSNR values for the simulation results of LGT approach shown in Figure 6.10. Results are shown both with (CA) and without (NCA) considering chromatic aberration. "Row" is the row number in Figure 6.10.

	Lens			*CA*		*NCA*	
Row	*S*	*C*	φ	*SSIM*	*PSNR*	*SSIM*	*PSNR*
1	+1 D	0 D	0°	0.986	42.220	0.987	42.370
2	+2.25 D	0 D	0°	0.991	43.204	0.991	43.078

Source: The Authors

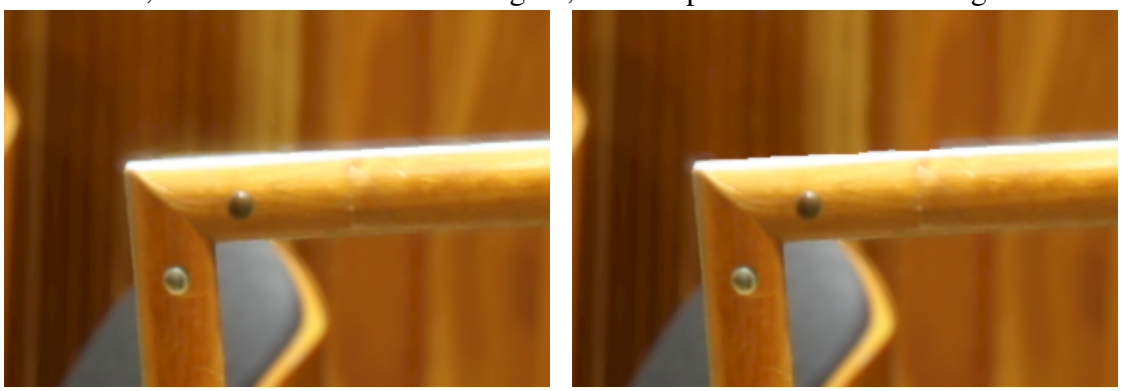
images with corresponding disparity maps, which can be converted to depth information (SCHARSTEIN et al., 2014). The per-pixel depth values expressed in meters were computed as:

$$Z = \frac{b \times f/(d + dpp)}{1000}, \tag{6.2}$$

where b is the camera baseline, f is camera's focal length in pixels, d is the pixel disparity value, and dpp is the x-difference of principal points (SCHARSTEIN et al., 2014). All these values are available in the files accompanying each image in the dataset. One should note that some of the depth values computed by this procedure are not properly aligned to the color pixels or do not correspond to a valid distance. In those cases, we manually adjusted the depth map using the distance from objects that were correctly registered and roughly correspond to the same depth. Early tests have shown that even slightly misplaced depth values can result in noticeable artifacts, such as the introduction of light or dark auras around objects, similar to the one shown in Figure 6.11.

Figure 1.2 demonstrates the use of our technique for simulating the view of a myopic subject (0.5 D) focusing at scene objects located at different depths. In Figure 1.2a the subject is focusing on the game box, causing the white and blue flower to appear blurry. In Figure 1.2b the focus has moved to the white and blue flower, making the game box to look defocused. Figure 6.12 illustrates the view of a hyperopic subject (-0.3 D) observing the same scene showing in Figure 1.2. Note how closer objects appear blurrier than far away ones

Figure 6.11: Light aura around handrail due to misplaced depth values. (a) Aura is present due to wrong depth values. (b) After fixing some depth values on the top-left border of the handrail, the aura is absent on that region, but still present on the other regions.



(a) (b)

Source: The Authors

Figure 6.13 shows a scene containing an Adirondack chair. The input color and depth images are shown in Figures 6.13a and 6.13b. The scene's plane index map is illustrated in Figure 6.13c, where each color indicates a different plane numbered from 0 (closest) to 13 (farthest) according to Table 6.6. Figures 6.14a and 6.14b compare the results of myopic simulations for 1.5 D and 0.75 D, respectively. These correspond to focusing at the two armrests (red and blue insets), which are located approximately at 0.67 m and 1.33 m from the observer. Please note that only one armrest appears in focus in each image. The white book is closer to the farthest armrest in diopters and, as such, its image appears sharper in Figure 6.14b compared to its appearance in Figure 6.14a.

Figure 6.15 illustrates the combined simulation of myopia and astigmatism (S = 1D, C = 3 D, $\varphi = 20°$). Note that the anisotropic blurriness on the book cover and on the mug handle (red inset) is more pronounced at 110°, a direction perpendicular to φ. Along such direction, the dioptric power is given by S+C = 4 D.

Figure 6.16 shows a scene containing a backpack on the foreground (approximately 0.91 m from the viewer) and a wardrobe and a broom on the background (approximately 1.54 m from the viewer), both presenting high and low-frequency content. Figure 6.17a illustrates the simulated view of a myopic subject with S = 1.1 D, thus focusing on the backpack (see red and green insets), while the background looks blurry (blue inset). Figure 6.17b shows another simulation, this time for a myopic subject with S = 0.65 D, thus focusing on the broom (blue inset), while the backpack appears blurry (red and green insets). Figure 6.17b simulates the view of a subject with myopia and astigmatism (S = 0.65 D, C = 0.45 D, and $\varphi = 90°$).

Table 6.6: Plane indices and their encoded colors.

0	1	2	3	4	5	6
7	8	9	10	11	12	13

Source: The Authors

Figure 6.18 shows a scene where various objects span several planes in terms of depth range (Figure 6.18c), and illustrates how our technique can produce realistic simulations without exhibiting inter-plane artifacts. Figure 6.19a simulates the view of an observer focusing on the moose puppet (blue inset) (0.52 m away), causing other elements to go increasingly out of focus as the distance from the viewer increases (green and red insets). Figure 6.19b illustrates the view of the scene when the focus is on the back of the green bucket located 0.87 m away from the observer (red inset). Finally, Figure 6.19c simulates the view of an individual with myopia and astigmatism (S = 1.10 D, C = 0.85 D, and $\varphi = 75°$).

Our project website provides some supplementary materials[1], including a video[2] captured in real time, illustrating the use of our technique. They also provide a user interface that can be used to explore high-resolution versions of the results shown in this work.

6.7 Discussion and Limitations

Our technique assumes a constant PSF across the entire visual field (*isoplanatic assumption*), even though it should slightly vary according to the direction of the incoming wavefront relative to the optical axis. It also does not take into account any of the high-order aberrations, which can be represented by a linear combination of Zernike polynomials.

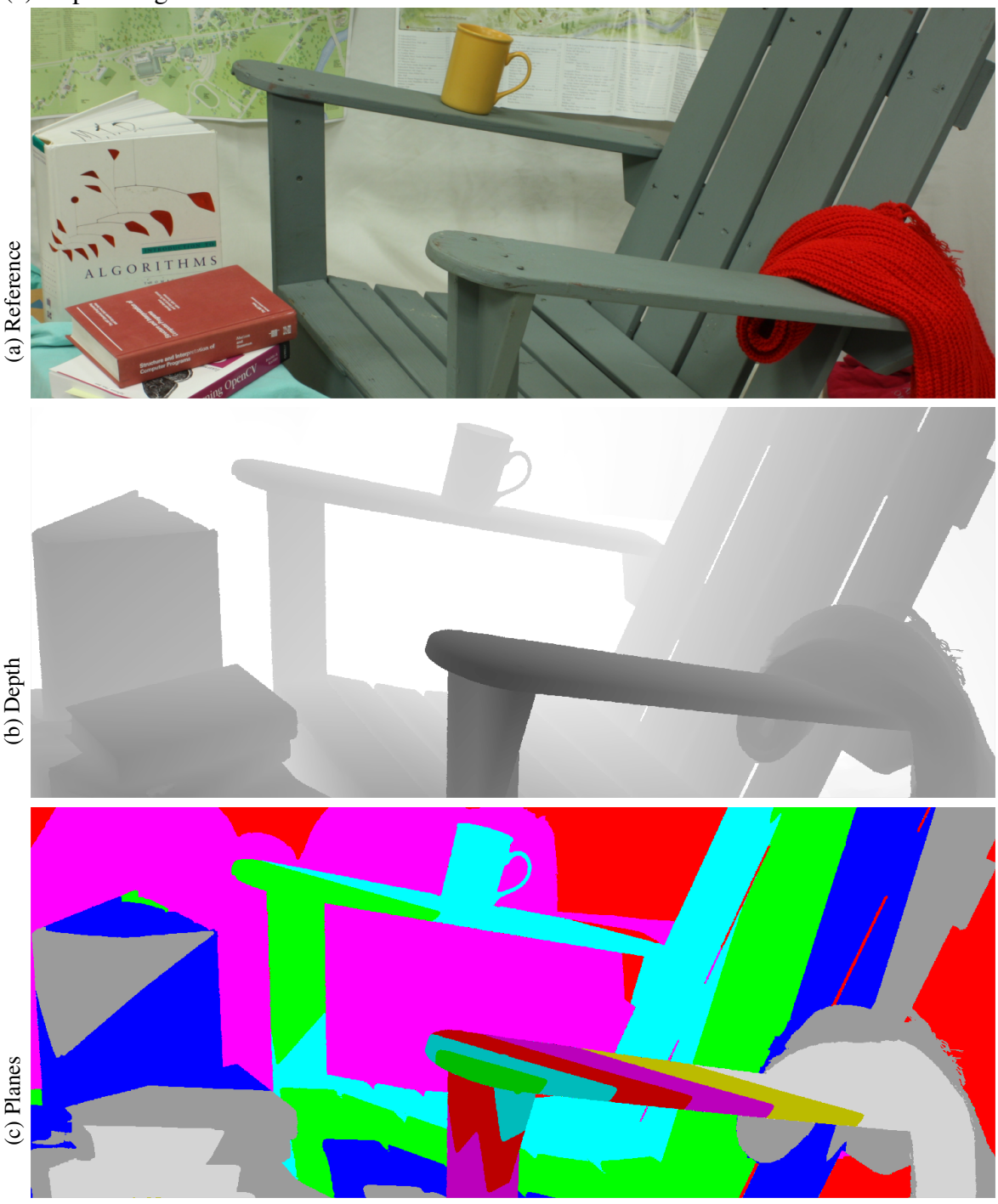
Wavefront errors are a function of accommodation, meaning that when a subject changes focal distance, aberrations might change as well (HE; BURNS; MARCOS, 2000). Remarkably, defocus is not affected because it is already determined by the change in the focus of interest. Astigmatism, on the other hand, might be affected. He et al. report a wavefront error of roughly 0.5 μm for each of the astigmatism coefficient in the Zernike polynomials (HE; BURNS; MARCOS, 2000). Our techniques disregard such minor effects.

[1] Supplementary materials: http://www.inf.ufrgs.br/~oliveira/pubs_files/VS/SM/
[2] Video: http://www.inf.ufrgs.br/~oliveira/pubs_files/VS/SM/VS_video.mp4

Figure 6.12: Simulated view of a hyperopic subject (-0.3 D). Note how closer objects appear blurrier than far away ones. The pairs of sub-images compare simulated (left) and original (right) patches.

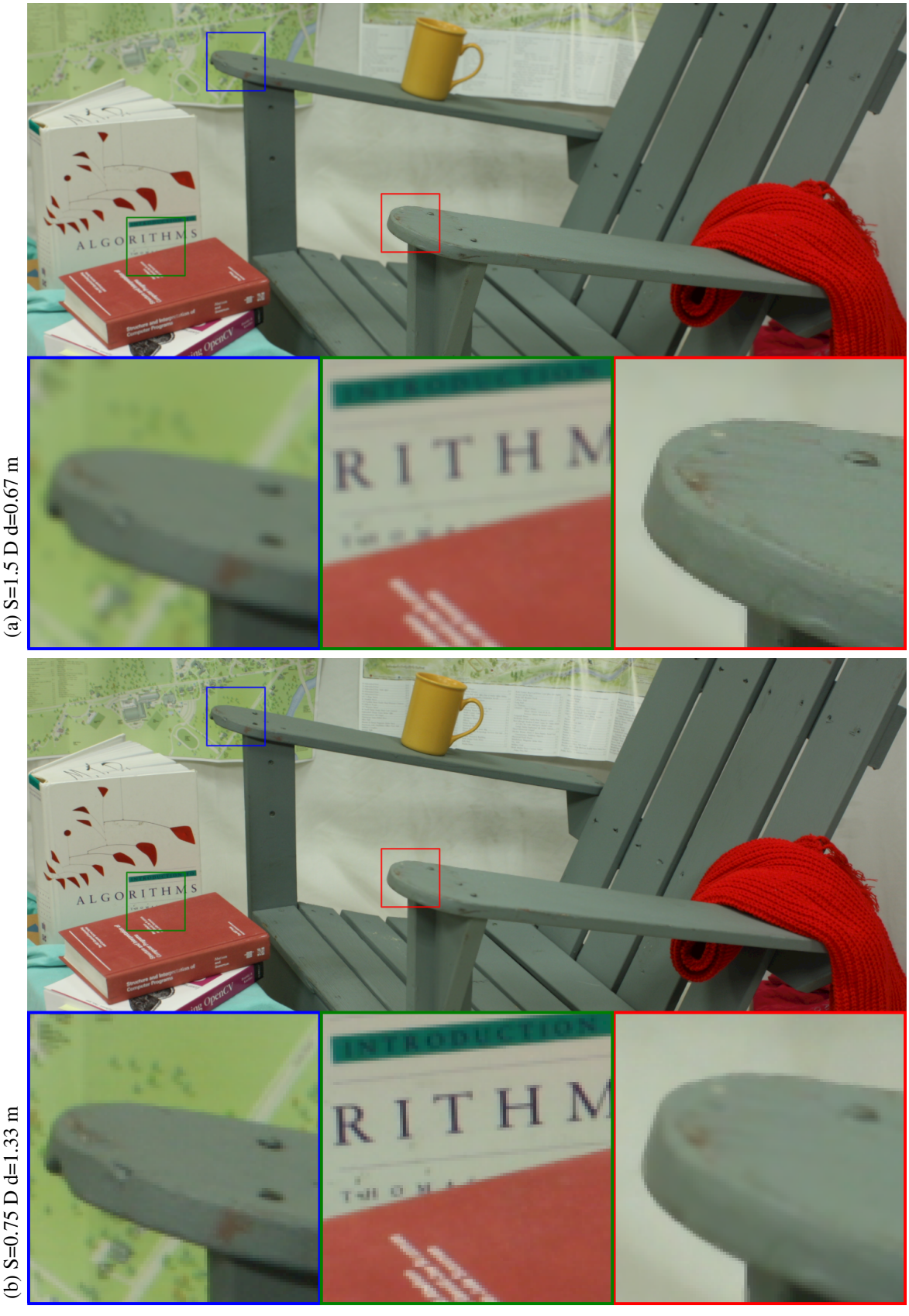
Source: The Authors

Figure 6.13: Adirondack chair. (a) Reference image. (b) Field discretization plane set. (c) Depth image.



Source: The Authors

Figure 6.14: Myopic simulations of Figure 6.13. Myopic simulations for 1.5 D (a) and 0.75 D (b), which correspond to focusing at the two armrests (red and blue insets), which are located approximately at 0.67 m and 1.33 m from the observer. The white book (green inset) is closer to the farthest armrest and, therefore, appears sharper in Figure 6.14b.



Source: The Authors

Figure 6.15: Myopic and astigmatic simulations of Figure 6.13. Simulation of myopia and astigmatism: S = 1 D, C =3 D, $\varphi = 20°$.

Source: The Authors

Figure 6.16: Backpack and broom on the background. (a) Reference image of an Backpack and a broom. (b) Depth image. (c) Field discretization plane set.

Source: The Authors

Figure 6.17: Myopic and astigmatic simulations of Figure 6.16. A backpack (0.91 m from the viewer) and a wardrobe and a broom on the background (1.54 m from the viewer). (a) Simulated view of a myopic subject with S = 1.1 D. (b) Simulated view of a myopic subject with S = 0.65 D. (c) Simulation of myopia and astigmatism (S = 0.65 D, C = 0.45 D, $\varphi = 90°$).

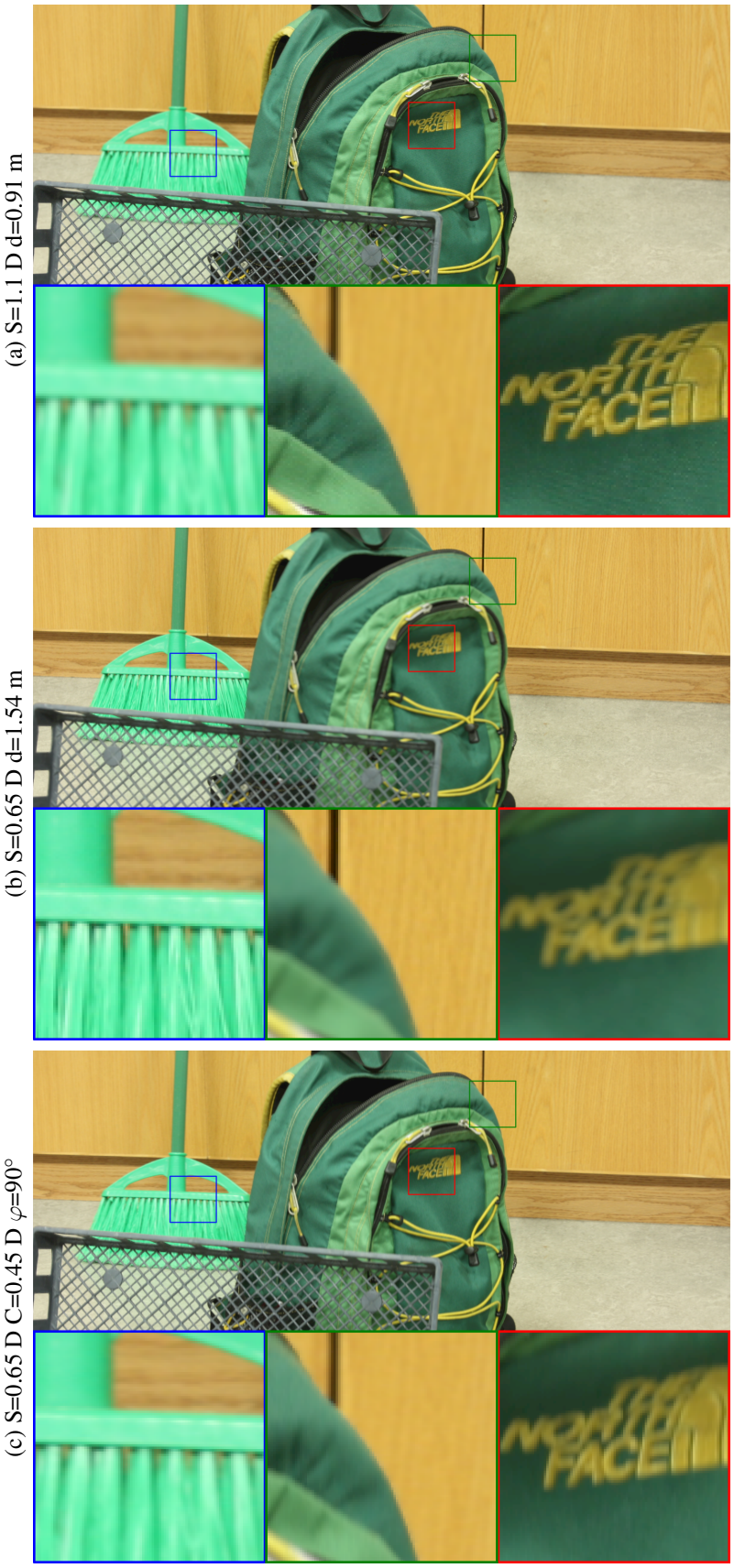
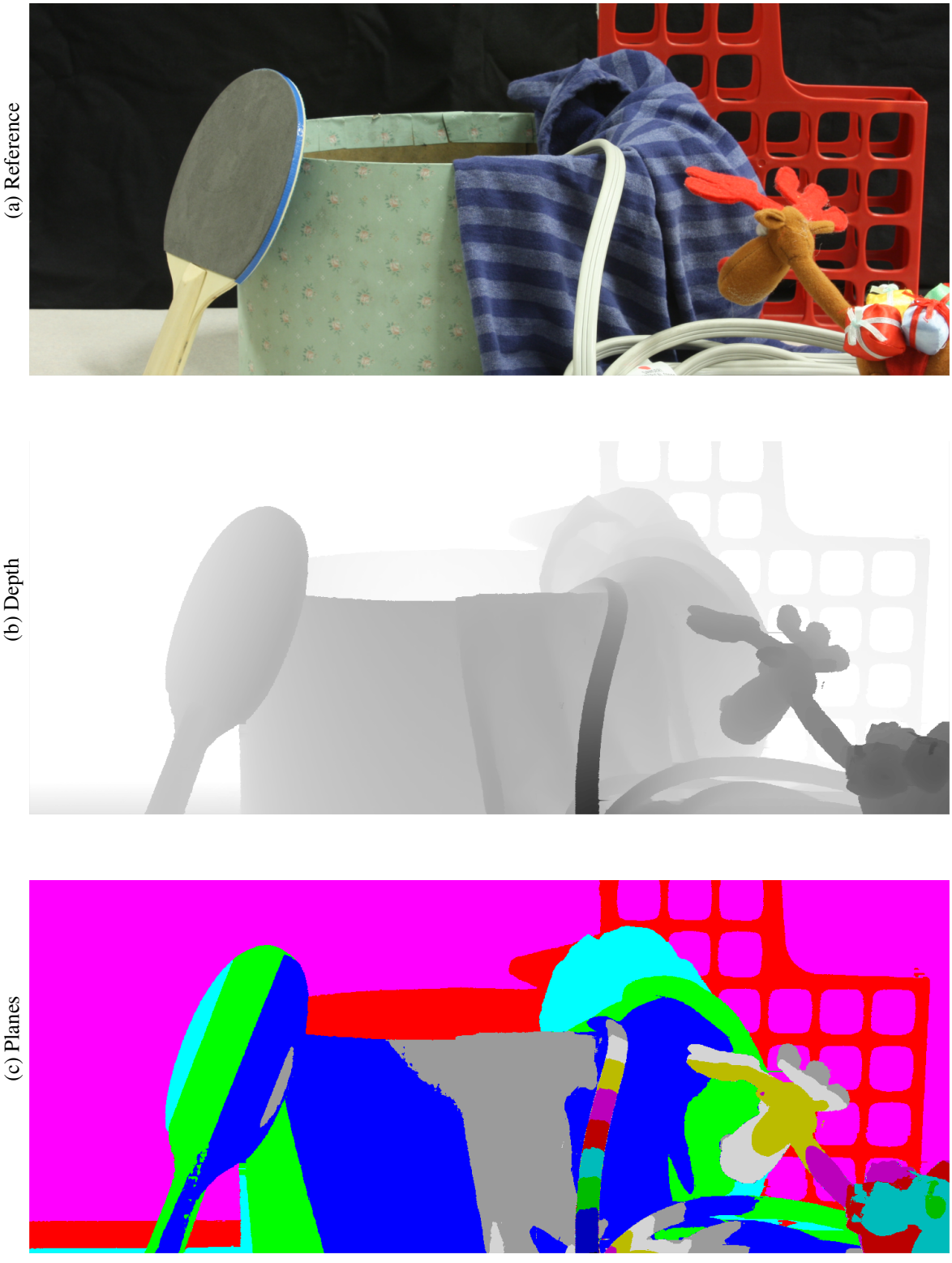
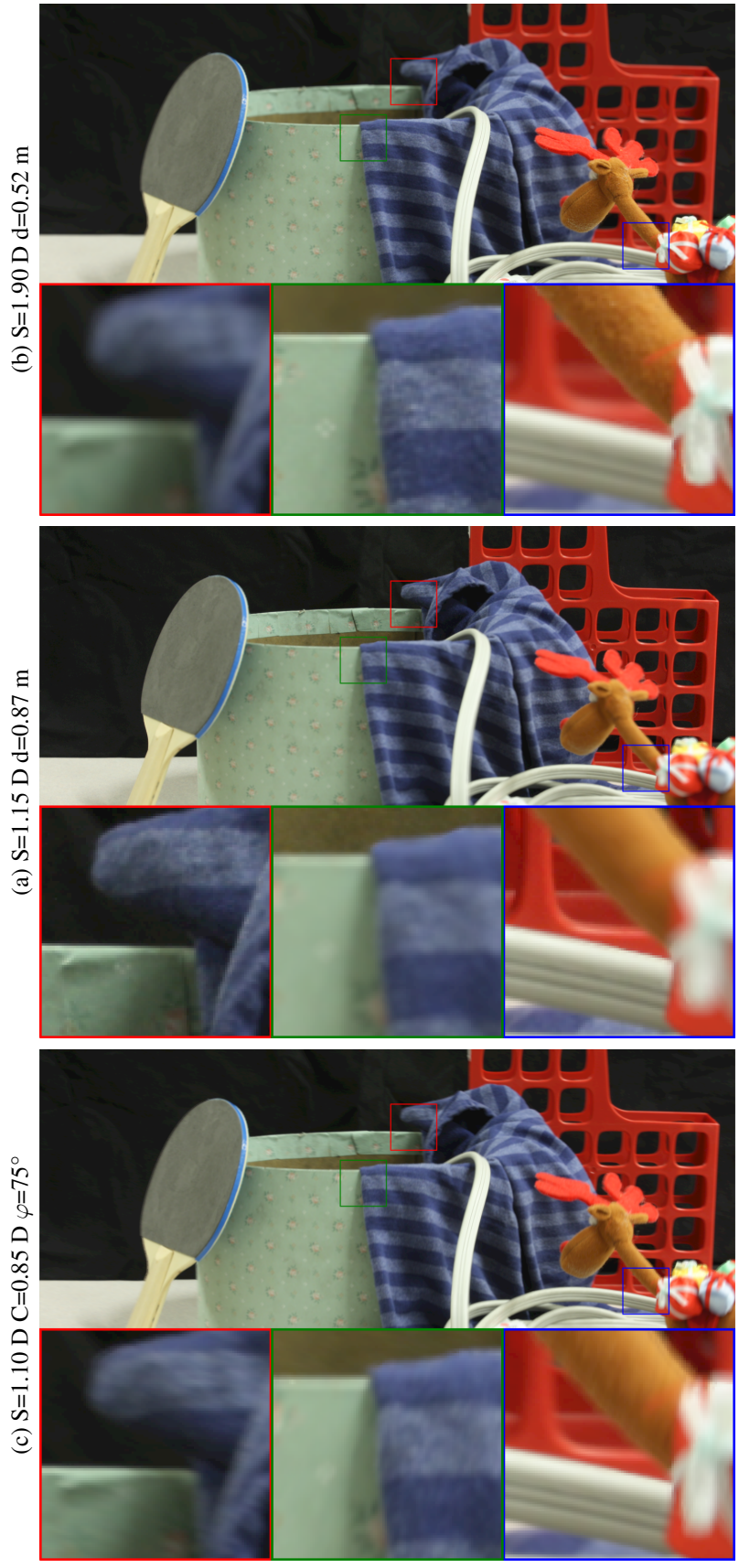


Figure 6.18: Scene with most objects spanning several planes in terms of depth range. (a) Reference image. (b) Depth image. (c) Field discretization plane set.



Source: The Authors

Figure 6.19: Myopic and astigmatic simulations of Figure 6.18. (a) View of a subject is focusing on moose puppet (0.52 m away). (b) View of a subject is focusing on the bucket back rim (0.87 m away). (c) Simulated view of a subject with myopia and astigmatism (S = 1.10 D, C = 0.85 D, $\varphi = 75°$).



Source: The Authors

The disocclusion (missing information recovery) technique we employ involves color extending and assumes that the color pattern around the occlusion border will persist across the occluded interior. Deep learning techniques could be used in order to attempt a more elaborated guess, but nevertheless they fall into the same category, in the sense that it is impossible to recover missing information with 100% accuracy.

6.8 Summary

This chapter described some experimental results of the techniques and formulas presented in this thesis. We performed an Airy pattern measurement to test a scaling factor predicted by theory in Fourier optics. We also validated the formula used in our wave optics technique, and performed various quantitative evaluation experiments for both techniques. A set of qualitative evaluation experiments were done to test the LGT technique. The results showed that both techniques produce quite realistic simulations of accommodation and low-order aberrations.

7 CONCLUSIONS AND DISCUSSION

We presented two practical solutions for simulating accommodation and low-order aberrations of the human eye, considering real scenes. The first technique, using Fourier optics, included the derivation of a formula for computing the equivalence between defocus aberration and the combination of object position and gazing focus distance. However, the lack of information due to partial occlusions among objects in arbitrary scenes precludes the use of such technique in general, limiting its use to scenes containing flat or concave depth values. This led to the development of our second method.

Our second technique is based on a new data structure called *light-gathering tree* (LGT), built from an RGB-D image and low-order aberration parameters (S, C, φ), focal distance, and pupil size. The use of an isoplanatic assumption and a set of auxiliary maps (PI, NPRP, and FPRP) leads to a light data structure that only needs to store the paths of rays traced through few tree nodes.

We validated the results of our techniques using quantitative and qualitative approaches. Quantitative validation was performed against ground-truth data (captured using a DSLR camera coupled with external lenses) for single-depth scenes using metrics such as SSIM and PSNR. For astigmatic optical configurations, our results achieved SSIM and PSNR values above 0.94 and 32.4, respectively. In the case of defocus-only, the SSIM and PSNR values are above 0.98 and 42.2, respectively. Such results indicate a strong agreement with the ground-truth images. Overall, the results obtained showed that the geometric optics approach does produce results compatible with wave optics in the human vision simulation domain. The only obvious divergence are on the diffraction patterns, but they were only observed on the Airy pattern measurement because an extremely long focal length was used, and that is obviously unrealistic for human vision simulation. Qualitative evaluation was performed using RGB-D images of real scenes as input.

Our techniques can be used in eye care areas where realistic human vision simulation is important. This includes providing doctors with concrete representations of how their patients see the world; explaining the benefits of refractive surgery to patients, contrasting their current vision with the corrected one, considering potential residual errors; and as training tool for medical students.

7.1 Future work

An interesting direction for future exploration is to consider microscopic environments by designing LGTs that can handle diffraction effects using Wigner functions (LUIS, 2007) to represent rays using the Huygens-Fresnel principle. The use of a separable bokeh technique (NIEMITALO, 2011; GARCIA, 2017) could further improve the algorithm's performance. However, this would probably be limited to myopia and hyperopia, since the astigmatic bokeh is not circularly symmetrical. Finally, one could implement higher-order aberrations replacing ray direction determination by sampling along the wavefront normals.

REFERENCES

BARSKY, B. et al. Introducing vision-realistic rendering. In: DEBEVEC, P.; GIBSON, S. (Ed.). **Proc. 13th Eurographics Workshop on Rendering**. Aire-la-Ville, Switzerland: Eurographics Association, 2002. p. 1–7.

BARSKY, B. A. Vision-realistic rendering: Simulation of the scanned foveal image from wavefront data of human subjects. In: **Proceedings of the 1st Symposium on Applied Perception in Graphics and Visualization**. New York, NY, USA: ACM, 2004. (APGV '04), p. 73–81. ISBN 1-58113-914-4. Available from Internet: <http://doi.acm.org/10.1145/1012551.1012564>.

BARSKY, B. A. et al. Investigating occlusion and discretization problems in image-based blurring techniques. In: HALL, P. W. P. (Ed.). **Vision, Video, and Graphics, VVG 2003, University of Bath, UK, July 10-11th, 2003, Proceedings**. Aire-la-Ville, Switzerland: The Eurographics Association, 2003. p. 97–102.

BEDGGOOD, P. et al. Characteristics of the human isoplanatic patch and implications for adaptive optics retinal imaging. **Journal of Biomedical Optics**, v. 13, n. 2, p. 1 – 7 – 7, 2008.

CHOLEWIAK, S. A. et al. Chromablur: Rendering chromatic eye aberration improves accommodation and realism. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 36, n. 6, p. 210:1–210:12, nov. 2017. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/3130800.3130815>.

COOK, R. L.; PORTER, T.; CARPENTER, L. Distributed ray tracing. In: **Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques**. New York, NY, USA: ACM, 1984. (SIGGRAPH '84), p. 137–145. ISBN 0-89791-138-5. Available from Internet: <http://doi.acm.org/10.1145/800031.808590>.

CURA, R.; PERRET, J.; PAPARODITIS, N. A state of the art of urban reconstruction: street, street network, vegetation, urban feature. **CoRR**, abs/1803.04332, 2018. Available from Internet: <http://arxiv.org/abs/1803.04332>.

DAI, G. **Wavefront Optics for Vision Correction**. P.O. Box 10, Belligham, Washington 98227-0010, USA: Society of Photo Optical, 2008. (SPIE Press monograph). ISBN 9780819469663.

DENNISTON, A.; MURRAY, P. **Oxford Handbook of Ophthalmology**. 3. ed. Oxford, UK: Oxford University Press, 2014. ISBN 9780199679980. Available from Internet: <http://oxfordmedicine.com/view/10.1093/med/9780199679980.001.0001/med-9780199679980>.

GARCIA, K. Circular separable convolution depth of field. In: **ACM SIGGRAPH 2017 Talks**. New York, NY, USA: ACM, 2017. (SIGGRAPH '17), p. 16:1–16:2. ISBN 978-1-4503-5008-2. Available from Internet: <http://doi.acm.org/10.1145/3084363.3085022>.

GOODMAN, J. W. **Introduction to Fourier Optics**. 3. ed. 4950 S. Yosemite Street, F2, #197, Greenwood Village, CO 80111: Roberts & Company Publishers, 2005. 491 p.

HAY, J. C.; PICK, H. L.; ROSSER, E. Adaptation to chromatic aberration by the human visual system. **Science**, American Association for the Advancement of Science, v. 141, n. 3576, p. 167–169, 1963. ISSN 00368075, 10959203. Available from Internet: <http://www.jstor.org/stable/1710699>.

HE, J.; BURNS, S.; MARCOS, S. Monochromatic aberrations in the accommodated human eye. **Vision Research**, v. 40, n. 1, p. 41–48, 2000. ISSN 0042-6989. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S004269899900156X>.

HECHT, E. **Optics**. Addison-Wesley, 2002. (Pearson education). ISBN 9780805385663. Available from Internet: <https://books.google.com.br/books?id=7aG6QgAACAAJ>.

JENKINS, F.; WHITE, H. **Fundamentals of Optics**. 4. ed. 1333 Burr Ridge Parkway Burr Ridge, Il 60521: McGraw-Hill Higher Education, 2001.

KNUTSSON, H.; WESTIN, C.-F. Normalized and differential convolution: Methods for interpolation and filtering of incomplete and uncertain data. In: **CVPR**. New York City, USA: [s.n.], 1993. p. 515–523.

KRAUS, M.; STRENGERT, M. Depth-of-field rendering by pyramidal image processing. **Comput. Graph. Forum**, v. 26, p. 645–654, 09 2007.

KRUEGER, M. L.; OLIVEIRA, M. M.; KRONBAUER, A. L. Personalized visual simulation and objective validation of low-order aberrations of the human eye. In: CAPPABIANCO, F. A. M. et al. (Ed.). **Electronic Proceedings of the 29th Conference on Graphics, Patterns and Images (SIBGRAPI'16)**. São José dos Campos, SP, Brazil: IEEE, 2016. p. 64–71. ISSN 2377-5416. Available from Internet: <http://gibis.unifesp.br/sibgrapi16>.

LEE, S.; EISEMANN, E.; SEIDEL, H.-P. Real-time lens blur effects and focus control. **ACM Transactions on Graphics (Proc. of SIGGRAPH)**, v. 29, p. 65:1–65:7, July 2010. Available from Internet: <http://graphics.tudelft.nl/Publications-new/2010/LES10a>.

LOMBARDO, M.; LOMBARDO, G. Wave aberration of human eyes and new descriptors of image optical quality and visual performance. **Journal of Cataract & Refractive Surgery**, Elsevier, v. 36, n. 2, p. 313–331, Feb 2010. ISSN 0886-3350. Available from Internet: <https://doi.org/10.1016/j.jcrs.2009.09.026>.

LUIS, A. Complementary huygens principle for geometrical and nongeometrical optics. **European Journal of Physics**, v. 28, p. 231–240, 2007.

NIEMITALO, O. Blog, **Circularly symmetric convolution and lens blur**. 2011. <http://yehar.com/blog/?p=1495>.

NIEßNER, M.; STURM, R. Real-time simulation and visualization of human vision through eyeglasses on the gpu. In: **Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry**. [S.l.]: ACM, 2012. p. 195–202.

POLYANSKIY, M. N. **Refractive index database**. 2019. <https://refractiveindex.info>. Accessed on 2019-01-23.

POTMESIL, M.; CHAKRAVARTY, I. Synthetic image generation with a lens and aperture camera model. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 1, n. 2, p. 85–108, April 1982. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/357299.357300>.

SCHARSTEIN, D. et al. High-resolution stereo datasets with subpixel-accurate ground truth. In: JIANG, X.; HORNEGGER, J.; KOCH, R. (Ed.). **GCPR**. Münster, Germany: Springer, 2014. (Lecture Notes in Computer Science, v. 8753), p. 31–42.

SCHEDL, D.; WIMMER, M. A layered depth-of-field method for solving partial occlusion. **Journal of WSCG**, v. 20, n. 3, p. 239–246, jun. 2012. ISSN 1213-6972. Available from Internet: <https://www.cg.tuwien.ac.at/research/publications/2012/schedl-2012-dof/>.

SCHWARTZ, M. **Principles of Electrodynamics**. Dover Publications, 1987. (Dover Books on Engineering). ISBN 9780486654935. Available from Internet: <https://books.google.com.br/books?id=dCQiejCy1kcC>.

SCHWARTZ, S. H. **Visual perception: A clinical orientation**. 4. ed. New York, NY, USA: McGraw-Hill Medical Pub. Division, 2010.

SCOFIELD, C. 2 1/2-d depth-of-field simulation for computer animation. In: KIRK, D. (Ed.). **Graphics Gems III (IBM Version)**. San Francisco: Morgan Kaufmann, 1992. p. 36–38. ISBN 978-0-12-409673-8. Available from Internet: <http://www.sciencedirect.com/science/article/pii/B978008050755250018X>.

SHINYA, M. Post-filtering for depth of field simulation with ray distribution buffer. In: **Proceedings of Graphics Interface '94**. Toronto, Ontario, Canada: Canadian Human-Computer Communications Society, 1994. (GI '94), p. 59–66. ISBN 0-9695338-3-7. ISSN 0713-5424. Available from Internet: <http://graphicsinterface.org/wp-content/uploads/gi1994-8.pdf>.

TRANTHAM, K.; REECE, T. J. Demonstration of the airy disk using photography and simple light sources. **American Journal of Physics**, v. 83, n. 11, p. 928–934, 2015.

WYANT, J. C.; CREATH, K. Basic wavefront aberration theory for optical metrology. In: SHANNON, R. R.; WYANT, J. C. (Ed.). **Applied Optics and Optical Engineering**. [S.l.]: Academic Press, 1992. XI, p. 1–53.

XIAO, L. et al. Deepfocus: Learned image synthesis for computational display. In: **ACM SIGGRAPH 2018 Talks**. New York, NY, USA: ACM, 2018. (SIGGRAPH '18), p. 4:1–4:2. ISBN 978-1-4503-5820-0. Available from Internet: <http://doi.acm.org/10.1145/3214745.3214769>.

ZANNOLI, M. et al. Blur and the perception of depth at occlusions. **Journal of Vision**, Association for Research in Vision and Ophthalmology Inc., v. 16, n. 6, 1 2016. ISSN 1534-7362.

ZIEGLER, R.; CROCI, S.; GROSS, M. Lighting and occlusion in a wave-based framework. **Computer Graphics Forum**, v. 27, n. 2, p. 211–220, 2008. Available from Internet: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2008.01118.x>.

APPENDIX A — MATHEMATICAL BACKGROUND

A.1 Binomial approximation

Binomial approximation is a mathematical tool used to approximate powers of the binomial $(1+x)$, where x happens to be a small number. Whenever $|x| < 1$ and $|\alpha x| \ll 1$, where $x, \alpha \in \mathbb{C}$, one can truncate the Maclaurin series of the binomial to the second term, yielding the approximation

$$(1+x)^{\alpha} = 1 + \alpha x + \frac{1}{2}\alpha(\alpha-1)x^2 + \frac{1}{6}\alpha(\alpha-1)(\alpha-2)x^3 \ldots \approx 1 + \alpha x.$$

As a corollary, $\sqrt{1+x} \approx 1 + x/2$.

APPENDIX B — MATLAB SOURCES

This appendix provides the source code for the approach described in Chapter 4, along with the adjustments described in Section 6.3. It consists of a setup code, listed in Appendix B.1, which is used to define the optical system properties, perform any necessary adjustments in the image (chromatic aberration, magnification, and brightness changes) and call the function `gen_PSF`, defined in Appendix B.2. Both of these algorithms issue calls to the other helper functions, listed in Appendices B.3 to B.15.

In case one does not wish to simulate effects caused by external lenses, the variable `bpar.extralens(1).present` should be set to 0, and `bpar.ab_S`, `bpar.ab_C` and `bpar.ab_angle` should be adjusted according to the low-order aberration parameters (S, C, φ) respectively.

B.1 Setup code

```
% These constants are the wavelengths in nanometers in vacuum for each of the RGB
% channels, plus an extra 'White' channel centered on yellow frequency to use in
% grayscale simulations
R_nm = 700;
G_nm = 510;
B_nm = 440;
W_nm = 550; % "White" wavelength is mean value of human sensitivity

bpar.iterate_pupil_pixels           = 1;        % Whether to iterate over better size in
                                                % pixels for pupil. If not iterating,
                                                % the size in pixels is fixed
bpar.sensor_pixel_scale             = 1;        % Set <> 1 if you wish to have different
                                                % sized sensor pixels
bpar.use_wave_optics                = 1;
bpar.num_components                 = 3;        % RGB are the three components
bpar.f                              = 18;       % This is the camera zoom lens focal
                                                % length in mm
bpar.Fnum                           = '5.0';    % This is the camera F number
bpar.frame                          = 112;      % This is the number of pixels in the
                                                % frame minus one
bpar.brightness                     = 0;        % Brightness to use
bpar.ignore_vertex_distance         = 0;
bpar.new_mode                       = 1;
bpar.apply_chromatic_aberration     = 1;        % If 1, then chromatic aberration will
                                                % be applied to external lens
bpar.psf_magnification              = 1;        % (internal use) the magnification to be
                                                % applied to PSF before returning it to
                                                % caller
numerical_validation_with_linear    = 0;

% These field should be eliminated
bpar.doapply                        = 0;        %
bpar.scale                          = 0;        %
bpar.entrance_pupil_pos             = 47;       % distance (in mm) from camera lens
                                                % mount to entrance pupil
bpar.gaze_objdist                   = 0;        % distance (in mm) from camera lens
                                                % mount to object to be observed
bpar.gaze_focus                     = 0;        % distance (in mm) from camera lens
                                                % mount to gazing focus point
```

```
bpar.extralens_overall_astig_angle  = 0;

bpar.ab_S = 0;                                  % S (spherical) in diopters
bpar.ab_C = 0;                                  % C (cylinder) in diopters
bpar.ab_angle = 0;                              % axis angle in degrees

bpar.extralens(1).present           = 0;
bpar.extralens(1).thin_S            = 0;
bpar.extralens(1).thin_C            = 0;
bpar.extralens(1).astig_angle       = 0;

bpar.extralens(1).pos               = 101;      % distance (in mm) from camera lens
                                                % mount to extra lens back surface
bpar.extralens(1).thickness         = 0;        % extra lens center thickness (in mm)
bpar.extralens(1).n                 = 1.5;      % extra lens refractive index
bpar.extralens(1).raw_back_x_power  = 0;        % extra lens back surface power in x
                                                % direction
bpar.extralens(1).raw_back_y_power  = 0;        % extra lens back surface power in y
                                                % direction
bpar.extralens(1).raw_front_x_power = 0;        % extra lens front surface power in x
                                                % direction
bpar.extralens(1).raw_front_y_power = 0;        % extra lens front surface power in y
                                                % direction

bpar.extralens(2).present           = 0;
bpar.extralens(2).thin_S            = 0;
bpar.extralens(2).thin_C            = 0;
bpar.extralens(2).astig_angle       = 0;

bpar.extralens(2).pos               = 111;      % distance (in mm) from camera lens
                                                % mount to extra lens back surface
bpar.extralens(2).thickness         = 0;        % extra lens center thickness (in mm)
bpar.extralens(2).n                 = 1.5;      % extra lens refractive index
bpar.extralens(2).raw_back_x_power  = 0;        % extra lens back surface power in x
                                                % direction
bpar.extralens(2).raw_back_y_power  = 0;        % extra lens back surface power in y
                                                % direction
bpar.extralens(2).raw_front_x_power = 0;        % extra lens front surface power in x
                                                % direction
bpar.extralens(2).raw_front_y_power = 0;        % extra lens front surface power in y
                                                % direction

MAX_EXTRA_LENSES = 2;

% RGB_nm is a 3-element vector containing the wavelength
% in nanonmeters for each color channel. If number of components is not 3,
% it is assumed to be one, indicating grayscale image processing
if (bpar.num_components == 3)
    bpar.RGB_nm = [R_nm, G_nm, B_nm];
else
    bpar.RGB_nm = [W_nm, W_nm, W_nm];
end

FL_crop_dx =  0;
FL_crop_dy =  0;

%================================================
% Set up experiment for astigmatism
%================================================

bpar.apply_chromatic_aberration     = 0;

bpar.gaze_objdist = 6490 + 25;
bpar.gaze_focus = 6490 + 25;

% Base path for all input images (JPG and DNG)
bpar.base_path = 'E:/ufrgs/experiments/2019_02_13/';

bpar.extralens(1).present = 1;              % If 1 then extra lens is present
bpar.extralens(1).thin_S =  0;
bpar.extralens(1).thickness = 0;            % extra lens center thickness (in mm)

crop_x = 2620;
```

```
crop_y = 1538;

crop_w = 113;
crop_h = 113;

% Base file name for to-be-blurred image (image A)
bpar.fname_A = 'IMG_0197';

bpar.f = 20;                        % This is the camera zoom lens focal length in mm
bpar.brightness = 0;

bpar.extralens(1).thin_C = -2.25;
bpar.extralens(1).thin_S = -0.25;
bpar.extralens(1).astig_angle = 69;

FL_crop_dx =  5;        % cropping adjustments due to magnification affecting
FL_crop_dy = -5;        % position of off-axis objects

bpar.fname_C = 'IMG_0196';

crop_w = 113 * 2;
crop_h = 113 * 2;

%----------------------------------------
% Setup phase finished. Start the process
%----------------------------------------

gt_crop_dx = FL_crop_dx;
gt_crop_dy = FL_crop_dy;

border_x = crop_w;
border_y = crop_h;

% Load the images
[meta_info, raw] = simply_load_dng(strcat(bpar.base_path, bpar.fname_A, '.dng'));
clear raw;
warning off MATLAB:imagesci:png:libraryWarning
full_image_A = unapply_gamma(im2double(imread(strcat(bpar.base_path, bpar.fname_A, ...
    '.jpg'))));
full_image_C = unapply_gamma(im2double(imread(strcat(bpar.base_path, bpar.fname_C, ...
    '.jpg'))));

% Convert images to grayscale if using only one channel
if bpar.num_components == 1
    full_image_A = rgb2gray(full_image_A);
    full_image_C = rgb2gray(full_image_C);
end

if strcmp(meta_info.Model, 'Canon EOS Rebel T3')
    bpar.sensor_pixels              = 4278;
    bpar.sensor_width               = 0.0222;
elseif strcmp(meta_info.Model, 'Canon EOS Rebel T6')
    bpar.sensor_pixels              = 5344;
    bpar.sensor_width               = 0.0223;
else
    error('UNKNOWN CAMERA')
end

% Defocus added to wave optics in order to simulate observing light-emitting object at
% distance 'd', while focusing at distance 'f'
bpar.wave_defocus = 0;

% If object or gazing have been set (not zero), then we compute the actual values for
% defocus and plane
if (bpar.gaze_objdist ~= 0) && (bpar.gaze_focus ~= 0)

    % proper relative distance and conversion to meters
    gf = (bpar.gaze_focus - bpar.entrance_pupil_pos) * 0.001;
    % proper relative distance and conversion to meters
    od = (bpar.gaze_objdist - bpar.entrance_pupil_pos) * 0.001;

    % This variable is used by wave optics. This formula is based on our derivation in
    % the dissertation
```

```
    bpar.wave_defocus = (od - gf) / (od * gf);

    clear gf;
    clear od;
end

num_extralenses = 0;

for lens_i = 1:MAX_EXTRA_LENSES
    if (bpar.extralens(lens_i).present == 1)
        num_extralenses = num_extralenses + 1;
    end
end

magnification_angle = 0;
x_magnification = 1;
y_magnification = 1;

if num_extralenses > 0
    for lens_i = 1:MAX_EXTRA_LENSES
        if bpar.extralens(lens_i).present == 0
            continue;
        end

        sin_theta = sin(deg2rad(bpar.extralens(lens_i).astig_angle));
        cos_theta = cos(deg2rad(bpar.extralens(lens_i).astig_angle));

        % ------------------------------------------------------------------------
        % Compute the magnification caused by extra lens
        %
        extralens_distance = (bpar.extralens(lens_i).pos - bpar.entrance_pupil_pos) ...
            * 0.001;
        if bpar.extralens(lens_i).thickness == 0
            extralens_raw_x_power = bpar.extralens(lens_i).thin_S;
            extralens_raw_y_power = bpar.extralens(lens_i).thin_S + ...
                bpar.extralens(lens_i).thin_C;
            x_power_factor = 1 / (1 - extralens_distance * extralens_raw_x_power);
            y_power_factor = 1 / (1 - extralens_distance * extralens_raw_y_power);

            % Formula derived to apply a rotated scaling
            this_x_magnification = (1 * x_power_factor);
            this_y_magnification = (1 * y_power_factor);
            if (this_x_magnification ~= this_y_magnification)
                if magnification_angle ~= 0
                    error('Competing astigmatism angles')
                end
                magnification_angle = bpar.extralens(lens_i).astig_angle;
            end
            if bpar.extralens(lens_i).thin_C ~= 0
                if bpar.extralens_overall_astig_angle ~= 0
                    error('Competing astigmatism angles')
                end
                bpar.extralens_overall_astig_angle = bpar.extralens(lens_i).astig_angle;
            end
            x_magnification = x_magnification * this_x_magnification;
            y_magnification = y_magnification * this_y_magnification;
            clear extralens_rot_raw_thin_x_power;
            clear extralens_rot_raw_thin_y_power;
        end
    end

    clear sin_theta;
    clear cos_theta;

    % ------------------------------------------------------------------------
    % Each channel will have its own chromatic-aberration-dependent S
    if bpar.num_components == 3
        for i_lambda = 1 : bpar.num_components
            bpar.nd_S(i_lambda) = 0;
            bpar.nd_C(i_lambda) = 0;
            for lens_i = 1 : MAX_EXTRA_LENSES
                if bpar.extralens(lens_i).present == 1
                    extralens_distance = (bpar.extralens(lens_i).pos ...
```

```
                        - bpar.entrance_pupil_pos) * 0.001;
                    ca_S = f_CalcChromaticDiopter(bpar.extralens(lens_i).thin_S, ...
                        i_lambda, 0, bpar);
                    bpar.nd_S(i_lambda) = bpar.nd_S(i_lambda) + f_CalcNewDiopter( ...
                        ca_S, extralens_distance * 1000);
                    ca_C = f_CalcChromaticDiopter(bpar.extralens(lens_i).thin_C, ...
                        i_lambda, 0, bpar);
                    bpar.nd_C(i_lambda) = bpar.nd_C(i_lambda) + f_CalcNewDiopter( ...
                        ca_C, extralens_distance * 1000);
                end
            end
        end
    else
        for i_lambda = 1 : 3
            bpar.nd_S(i_lambda) = 0;
            bpar.nd_C(i_lambda) = 0;
            for lens_i = 1 : MAX_EXTRA_LENSES
                if bpar.extralens(lens_i).present == 1
                    extralens_distance = (bpar.extralens(lens_i).pos ...
                        - bpar.entrance_pupil_pos) * 0.001;
                    bpar.nd_S(i_lambda) = bpar.nd_S(i_lambda) + f_CalcNewDiopter( ...
                        bpar.extralens(lens_i).thin_S, extralens_distance * 1000);
                    bpar.nd_C(i_lambda) = bpar.nd_C(i_lambda) + f_CalcNewDiopter( ...
                        bpar.extralens(lens_i).thin_C, extralens_distance * 1000);
                end
            end
        end
    end
    clear ca_S;
    clear ca_C;
else
    x_magnification = 1;
    y_magnification = 1;
    for i_lambda = 1 : 3
        bpar.nd_S(i_lambda) = 0;
        bpar.nd_C(i_lambda) = 0;
    end
end

% ------------------------------------------------------------------
% Positive lens causes magnification (m > 1)
% We should shrink the PSF and shrink the ground truth blurred image
if x_magnification > 1
    PSF_x_scale = 1 / x_magnification;
    gtruth_x_magnification = 1 / x_magnification;
    input_image_x_magnification = 1;
% ------------------------------------------------------------------
% Negative lens causes demagnification (m < 1)
% don't do this!! We should enlarge the PSF and shrink the already-convolved image
% We should shrink the yet-to-be convolved image
elseif x_magnification < 1
    PSF_x_scale = 1;
    gtruth_x_magnification = 1;
    input_image_x_magnification = x_magnification;
% No magnification at all
else
    PSF_x_scale = 1;
    gtruth_x_magnification = 1;
    input_image_x_magnification = 1;
end

% ------------------------------------------------------------------
% Positive lens causes magnification (m > 1)
% We should shrink the PSF and shrink the ground truth blurred image
if y_magnification > 1
    PSF_y_scale = 1 / y_magnification;
    gtruth_y_magnification = 1 / y_magnification;
    input_image_y_magnification = 1;
% ------------------------------------------------------------------
% Negative lens causes demagnification (m < 1)
% don't do this!! We should enlarge the PSF and shrink the already-convolved image
% We should shrink the yet-to-be convolved image
elseif y_magnification < 1
```

```
    PSF_y_scale = 1;
    gtruth_y_magnification = 1;
    input_image_y_magnification = y_magnification;
% No magnification at all
else
    PSF_y_scale = 1;
    gtruth_y_magnification = 1;
    input_image_y_magnification = 1;
end

%--------------------------------------------------------------------
% Perform input image downscaling if necessary
if (input_image_x_magnification ~= 1) || (input_image_y_magnification ~= 1)
    image_A_is_resized = 1;

    resized_image_A = apply_anisotropic_magnification(full_image_A, magnification_angle,
     ...
        input_image_x_magnification, input_image_y_magnification, 0, 0);
    resized_image_A = max(0,min(resized_image_A,1));
else
    image_A_is_resized = 0;
    resized_image_A = full_image_A;
end

%--------------------------------------------------------------------
% Perform ground-truth image downscaling if necessary
if (gtruth_x_magnification ~= 1) || (gtruth_y_magnification ~= 1)
    image_C_is_resized = 1;

    resized_image_C = apply_anisotropic_magnification(full_image_C, magnification_angle,
     ...
        gtruth_x_magnification, gtruth_y_magnification, 0, 0);
    resized_image_C = max(0,min(resized_image_C,1));
else
    image_C_is_resized = 0;
    resized_image_C = full_image_C;
end

%% auto brightness code - these coordinates lead to a small white patch on the image
bright_sample_A = imcrop(resized_image_A,   [crop_x - 340, ...
                                               crop_y, ...
                                               64, 64]);
bright_sample_C = imcrop(resized_image_C,   [crop_x + gt_crop_dx - 340, ...
                                               crop_y + gt_crop_dy, ...
                                               64, 64]);

for i_lambda = 1 : bpar.num_components
    channel_brightness_scale = mean2(bright_sample_A(:, :, i_lambda)) / ...
        mean2(bright_sample_C(:, :, i_lambda));
    resized_image_C(:, :, i_lambda) = resized_image_C(:, :, i_lambda) * ...
        channel_brightness_scale;
end

image_A_x = crop_x;
image_A_y = crop_y;

cropped_image_A = imcrop(resized_image_A, [image_A_x - border_x, ...
                                           image_A_y - border_y, ...
                                           crop_w + 2 * border_x, ...
                                           crop_h + 2 * border_y]);

image_C_x = image_A_x;
image_C_y = image_A_y;


im_U = apply_gamma(imcrop(full_image_A, [crop_x, ...
                                           crop_y, ...
                                           crop_w, crop_h]));

cropped_image_C = imcrop(resized_image_C, [image_C_x + gt_crop_dx, image_C_y + ...
    gt_crop_dy, crop_w, crop_h]);

w_out = gen_PSF(0, bpar);
```

```
403
404 %-------------------------------------------------------------------
405 % Resize the PSF and compute OTF for each channel
406 w_OTF = zeros(size(cropped_image_A));
407 g_OTF = zeros(size(cropped_image_A));
408 for i_lambda = 1 : bpar.num_components
409     unresized_PSF = w_out.PSF(:, :, i_lambda);
410     if (PSF_x_scale ~= 1) || (PSF_y_scale ~= 1)
411         resized_PSF = apply_anisotropic_magnification(unresized_PSF, magnification_angle
    , ...
412             PSF_x_scale, PSF_y_scale, 0, 0);
413         resized_PSF = max(0,min(resized_PSF,1));
414     else
415         resized_PSF = unresized_PSF;
416     end
417     PSF = zeropad_newsize(resized_PSF, size(cropped_image_A));
418     PSF = PSF / sum(sum(PSF));
419     w_OTF(:, :, i_lambda) = psf2otf(PSF);
420 end
421
422 %-------------------------------------------------------------------
423 % Convolution in spacial domain is the same as multiplication in frequency domain
424 w_convolved_rgb = zeros(size(cropped_image_A), 'double');
425 for i_lambda = 1 : bpar.num_components
426     w_convolved_rgb(:, :, i_lambda) = ifft2(fft2(cropped_image_A(:,:,i_lambda)) .* ...
427         w_OTF(:, :, i_lambda));
428 end
429
430 g_convolved_rgb = zeros(size(cropped_image_A), 'double');
431 for i_lambda = 1 : bpar.num_components
432     g_convolved_rgb(:, :, i_lambda) = ifft2(fft2(cropped_image_A(:,:,i_lambda)) .* ...
433         g_OTF(:, :, i_lambda));
434 end
435
436 %-------------------------------------------------------------------
437 % Remove borders and apply gamma encoding
438 linear_im_A = (imcrop(cropped_image_A, [border_x, border_y, crop_w, crop_h]));
439 im_A = apply_gamma(linear_im_A);
440 linear_im_BW = imcrop(w_convolved_rgb, [border_x, border_y, crop_w, crop_h]);
441 im_BW = apply_gamma(linear_im_BW);
442
443 linear_im_C = cropped_image_C;
444 im_C = apply_gamma(linear_im_C);
445
446 if image_A_is_resized == 1
447     total_images = 4;
448     sp1=subplot(1, total_images, 1);
449     imshow(im_U);
450     title('[Original] Sharp input image');
451 else
452     total_images = 3;
453 end
454 sp2=subplot(1, total_images, total_images - 2);
455 imshow(im_A);
456 if image_A_is_resized == 1
457     title('[Resized] Sharp input image');
458 else
459     title('[Original] Sharp input image');
460 end
461 sp3=subplot(1, total_images, total_images - 1);
462 imshow(im_BW);
463 title('WO-Blurred image');
464 sp5=subplot(1, total_images, total_images);
465 imshow(im_C);
466 if image_C_is_resized == 1
467     title('[Resized] Ground truth');
468 else
469     title('[Original] Ground truth');
470 end
471 if numerical_validation_with_linear == 1
472     w_SSIM = ssim(linear_im_BW, linear_im_C)
473     w_PSNR = psnr(linear_im_BW, linear_im_C)
474 else
```

```
    w_SSIM = ssim(im_BW, im_C)
    w_PSNR = psnr(im_BW, im_C)
end
```

B.2 PSF generation function

```
%----------------------
% gen_PSF(in_rgb, bpar)
%----------------------
% Generate a PSF
function out = gen_PSF(in_rgb, bpar)
    % Convert the F-Number from string to integer
    Fnumber = str2double(bpar.Fnum);

    % D is the pupil diameter in mm
    D = bpar.f / Fnumber;

    % pupil_radius is the pupil radius in mm
    pupil_radius = D/2;

    % final_side is the number of pixels on each dimension of the final PSF image.
    % We add one because bpar.frame should always be even, and so final_side will
    % always be odd.
    % We want odd pixel dimensions because this way we can center the convolution kernel
    % to produce
    % something similar to a dirac delta when the wavefront is aberration-free
    final_side = bpar.frame + 1;

    % Q_alpha is a 3-element vector containing, for each channel, the alpha scaling
    % factor needed
    % to obtain the desired PSF resolution.
    Q_alpha = [0, 0, 0];

    % pupil_frame is a 3-element vector containing, for each channel, the number of
    % pixels in each dimension
    % of the general pupil function domain minus one (even).
    pupil_frame = [0, 0, 0];

    % out.pupil_frames is a copy of pupil_frame that is exported to the user
    out.pupil_frames = [0, 0, 0];

    % pupil_scaled_frame is a 3-element vector containing, for each channel, the number
    % of pixels in each dimension
    % of the alpha-zero-padded general pupil function domain minus one (even).
    % We pad zeros around the pupil domain in order to obtain the desired frequencies in
    % the Fourier Transform
    % that computes the PSF.
    pupil_scaled_frame = [0, 0, 0];

    % sensor_pixels is the horizontal number of pixels in the camera sensor. This value
    % can be obtained from the web (https://www.digicamdb.com/specs/canon_eos-rebel-t6/,
    % https://www.digicamdb.com/specs/canon_eos-rebel-t3/).
    sensor_pixels = bpar.sensor_pixels * bpar.sensor_pixel_scale;

    % sensor_width is the horizontal width of the camera sensor in meters. This value
    % can be obtained from the web.
    sensor_width = bpar.sensor_width;

    % out.expected_unit contains the units for every pixel in the final PSF
    out.expected_unit = 10^(6) * sensor_width / sensor_pixels;

    % out.actual_units is a 3-element vector containing, for each channel, the computed
    % units (in nanometers) for each pixel in the final PSF. Ideally all 3 channels
    % should have the same units, but this is not currently possible because (1) our FFT
    % function does not allow arbitrary frequency multipliers and (2) our imresize
    % function does not allow non-integer dimensions.
```

```
    % These values should be as close as possible to out.expected_unit.
    out.actual_units = [0, 0, 0];

    % Calculating alpha (Q)
    % alpha = sensor_pixels * lambda * Fnumber / sensor_size
    for i_lambda = 1 : bpar.num_components
        Q = sensor_pixels * (bpar.RGB_nm(i_lambda) * 10^(-9)) * Fnumber / sensor_width;
        [pf, pfs] = best_pupil_pixel_size(bpar, Q);
        unit = 1 / ((pfs + 1) / (pf + 1));
        unit = unit * bpar.RGB_nm(i_lambda) * Fnumber * 10^(-3);
        Q_alpha(i_lambda) = Q;
        pupil_frame(i_lambda) = pf;
        pupil_scaled_frame(i_lambda) = pfs;
        out.actual_units(i_lambda) = unit;
        out.pupil_frames(i_lambda) = pf + 1;
    end
    lin_srgb = zeros(size(in_rgb));
    OTF = zeros(size(in_rgb));
    out.PSF = zeros([final_side, final_side, 3]);
    for i_lambda = 1 : bpar.num_components
        lambda = bpar.RGB_nm(i_lambda) * 10^(-3);
        k      = (2*pi) / lambda;
        pupf = pupil_frame(i_lambda);
        [X, Y]  = meshgrid((-1:2/pupf:1),(1:-2/pupf:-1)); % from -1 to 1, steps of 2/n
        r       = sqrt(X.^2 + Y.^2);            % calulate radius
        coeff = [0; getZernikeFromPrescription( pupil_radius, ...
                bpar.nd_S(i_lambda) + bpar.wave_defocus + bpar.ab_S, ...
                bpar.nd_C(i_lambda) + bpar.ab_C, ...
                bpar.extralens_overall_astig_angle + bpar.ab_angle)];
        WavefrontPhaseError = my_ZernikeSurface(pupf,X,Y,r,coeff);
        if i_lambda == 1
            out.WE_R = WavefrontPhaseError;
        elseif i_lambda == 2
            out.WE_G = WavefrontPhaseError;
        elseif i_lambda == 3
            out.WE_B = WavefrontPhaseError;
        end
        S = exp( 1i * k * WavefrontPhaseError);
        S(r > 1) = 0;                                % circular aperture
        oldside = pupf + 1;
        new_side = pupil_scaled_frame(i_lambda) + 1;
        if new_side > oldside
            S = zeropad_newsize(S, [new_side, new_side]);
        end
        PSF = fft2(S);                               % amplitude impulse response
        PSF = PSF .* conj(PSF);                      % square magnitude
        PSF = real(fftshift(PSF));                   % real part
        if new_side < oldside
            PSF = imresize(PSF, [new_side, new_side]);
        end
        if final_side > new_side
            PSF = zeropad_newsize(PSF, [final_side, final_side]);
        elseif final_side < new_side
            PSF = zerocut_newsize(PSF, [final_side, final_side]);
        end
        smallPSF = PSF / sum(sum(PSF));              % Scale so that PSF sums to unity.
        out.PSF(:, :, i_lambda) = smallPSF;
    end

    if (bpar.num_components == 1)
        OTF(:, :, 2) = OTF(:, :, 1);
        OTF(:, :, 3) = OTF(:, :, 1);
    end

    % from -1 to 1, steps of 2/n
    [out.X, out.Y]  = meshgrid(linspace(-1,1,bpar.frame+1),linspace(-1,1,bpar.frame+1));
    unit = (bpar.frame + 1) / 2;

    out.X = out.X * unit;
    out.Y = out.Y * unit;
end
```

B.3 Zernike coefficients generation

```
%-------------------------------------------------
% getZernikeFromPrescription(pupil_r, S, C, theta)
%-------------------------------------------------
% Zernike coefficients generation
function [coeffs] = getZernikeFromPrescription(pupil_r, S, C, theta)
    coeffs = zeros(20,1);
    R = pupil_r;
    % c3 = y-astigmatism     2,-2
    % c4 = defocus           2, 0
    % c5 = x-astigmatism     2, 2
    syms c3;      %   microns        +1.05
    syms c4;      %   microns        -3.55
    syms c5;      %   microns        -0.98
    % S in diopters
    % C in diopters
    % t in radians
    % R in mm
    T_rad = deg2rad(theta);
    eqn1 = c3 == (R^2*C*sin(2*T_rad)) / (4*sqrt(6));
    eqn2 = c4 == - ( (R^2*(S+(C/2)))  / (4*sqrt(3)) );
    eqn3 = c5 == (R^2*C*cos(2*T_rad)) / (4*sqrt(6));
    tmp1 = solve(eqn1, c3);
    tmp1 = vpa(tmp1);
    tmp1 = double(tmp1);
    tmp2 = solve(eqn2, c4);
    tmp2 = vpa(tmp2);
    tmp2 = double(tmp2);
    tmp3 = solve(eqn3, c5);
    tmp3 = vpa(tmp3);
    tmp3 = double(tmp3);
    c3 = tmp1;
    c4 = tmp2;
    c5 = tmp3;
    coeffs(3) = c3;
    coeffs(4) = c4;
    coeffs(5) = c5;
end
```

B.4 Wavefront phase error surface generation

```
%---------------------------------
% my_ZernikeSurface(nn, X, Y, r, z)
%---------------------------------
% This function calculates the wavefront based on a set of Zernike
% coefficients z to get frame size (nn+1)x(nn+1).
function [S] = my_ZernikeSurface(nn, X, Y, r, z)
    terms = length(z)-1;
    Theta = atan2(Y, X);
    S = zeros(nn+1);
    for i = 0:terms
        [n, m] = single2doubleZ(i);
        if (m == 0)
            pa = sqrt(n+1);
        else
            pa = sqrt(2*(n+1));
        end
        coef = pa;
        Surf = zeros(nn+1);
        for s = 0:(n-abs(m))/2
            c1 = n-s;
```

```
            c2 = (n+m)/2-s;
            c3 = (n-m)/2-s;
            Surf = Surf + (-1)^s*factorial(c1)/factorial(s) ...
                / factorial(c2)/factorial(c3)*power(r, n-2*s);
        end
        if (m < 0)
            Surf = Surf.*sin(abs(m)*Theta);
        elseif (m > 0)
            Surf = Surf.*cos(m*Theta);
        end
            S = S + z(i+1)*coef*Surf;
    end
    S(r > 1) = NaN;
end
```

B.5 Zernike's single to double-index conversion

```
%-------------------
% single2doubleZ(jj)
%-------------------
% This function converts single->double index in Zernike polynomials
% Source: [DAI, G. Wavefront Optics for Vision Correction]
function [n, m] = single2doubleZ(jj)
    n = floor(sqrt(2*jj+1)+0.5)-1;
    m = 2*jj-n*(n+2);
end
```

B.6 Fitting pupil size in pixels

```
%-------------------------------
% best_pupil_pixel_size(bpar, Q)
%-------------------------------
% Find closest integers that generate the best approximation for pupil frames
function [pupil_frame, pupil_scaled_frame] = best_pupil_pixel_size(bpar, Q)
    attempts = 360;
    pupil_frame = 112 * 1 + attempts * 2;
    try_pupil_frame = pupil_frame + 2;
    best_diff = 1000;
    if bpar.iterate_pupil_pixels ~= 1
        attempts = 1;
        try_pupil_frame = 112 + 2;
    end
    for i = 1:attempts
        try_pupil_frame = try_pupil_frame - 2;
        new_side_float = (try_pupil_frame + 1) * Q;
        new_side_ceil = ceil(new_side_float);
        new_side = floor(new_side_float);
        if (bitand(new_side_ceil, 1) == 1)
            new_side = new_side_ceil;
        end
        this_diff = new_side / new_side_float;
        if this_diff < 1
            this_diff = new_side_float / new_side;
        end
        if this_diff < best_diff
            best_diff = this_diff;
            pupil_frame = try_pupil_frame;
            pupil_scaled_frame = new_side - 1;
```

```
        end
    end
end
```

B.7 Matrix dimensions splitting

```
%----------------
% matsplit(A,dim)
%----------------
% matsplit
% version 1.3.0.0 (568 Bytes) by Matthew Eicholtz
% available at https://www.mathworks.com/matlabcentral/fileexchange/48439-matsplit
function varargout = matsplit(A,dim)
    %MATSPLIT Split matrix elements into separate variables.
    %   VARARGOUT = MATSPLIT(A) returns each element of the array A in a
    %   separate variable defined by VARARGOUT.
    %
    %   VARARGOUT = MATSPLIT(A,DIM) only splits the matrix in one dimension. If
    %   DIM=1, each column vector is assigned to an output variable. If
    %   DIM=2, each row vector is assigned to an output variable.
    %
    %   MRE 11/12/14 (last updated 11/13/14)
    if nargin==1
        varargout = num2cell(A);
    else
        varargout = num2cell(A,dim);
    end
end
```

B.8 Cropping a given surface

```
%----------------------------
% zerocut_newsize(W, newsize)
%----------------------------
% This function zero-pads the frame to be alpha times as large.
function S = zerocut_newsize(W, newsize)
    [p, q] = size(W);
    [n, m] = matsplit(newsize);
    S = W((p-n)/2+1:(p+n)/2,(q-m)/2+1:(q+m)/2);
end
```

B.9 Zero filling a given surface

```
%----------------------------
% zeropad_newsize(W, newsize)
%----------------------------
% This function zero-pads the frame to be alpha times as large.
function S = zeropad_newsize(W, newsize)
    oldsize = size(W);
    n = oldsize(1);
```

```
    m = oldsize(2);
    p = newsize(1);
    q = newsize(2);
    pn = int32(p - n);
    if (bitand(pn, 1) == 1)
        y1 = (p-n+1)/2;
        y2 = (p+n-1)/2;
    else
        y1 = (p-n)/2+1;
        y2 = (p+n)/2;
    end
    qm = int32(q - m);
    if (bitand(qm, 1) == 1)
        x1 = (q-m+1)/2;
        x2 = (q+m-1)/2;
    else
        x1 = (q-m)/2+1;
        x2 = (q+m)/2;
    end
    if ndims(W) == 2
        S = zeros(p, q);
        S(y1:y2,x1:x2) = W;
    else
        S = zeros(p, q, 3);
        S(y1:y2,x1:x2, :) = W;
    end
end
```

B.10 Gamma encoding

```
%--------------------
% apply_gamma(in_rgb)
%--------------------
% Gamma Correction
function rgb = apply_gamma(in_rgb)
    rgb = in_rgb.^(1/2.2);
end
```

B.11 Gamma decoding

```
%----------------------
% unapply_gamma(in_rgb)
%----------------------
% Gamma Correction
function rgb = unapply_gamma(in_rgb)
    rgb = in_rgb.^(2.2);
end
```

B.12 Applying anisotropic magnification

```
%-------------------------------------------------------------------
% apply_anisotropic_magnification(in_rgb, angle, x_scale, y_scale, x_add, y_add)
%-------------------------------------------------------------------
% Apply anisotropic magnification
function rgb = apply_anisotropic_magnification(in_rgb, angle, x_scale, y_scale, x_add,
    y_add)
    % extract image dimensions
    [nr nc channels] = size(in_rgb);
    % create a grid running from -0.5 to 1.0 in both directions
    x = linspace(-0.5, 0.5, nc);
    y1 = linspace(-0.5, 0.5, nr);
    y = flipud(y1);
    [X, Y] = meshgrid(x, y);
    % compute sine and cosine for the reverse angle
    % (we use reverse angle here because we have to simulate how a camera would view the
    % rotation and magnification in a way similar to how we compute all the reverse
    % transformations when creating a view camera matrix in OpenGL)
    sin_theta = sin(-deg2rad(angle));
    cos_theta = cos(-deg2rad(angle));
    % Rotate (this is a standard rotation algorithm)
    P_r = X*cos_theta+Y*sin_theta; Q_r = -X*sin_theta+Y*cos_theta;
    % Scale (we use the reciprocal of scaling because this is a reverse (camera)
    % transformation
    P_sr = P_r * (1 / x_scale); Q_sr = Q_r * (1 / y_scale);
    % Unrotate (this is the inverse matrix of a standard rotation)
    P = P_sr*cos_theta - Q_sr*sin_theta + x_add / nc; Q = P_sr*sin_theta + Q_sr * ...
        cos_theta + y_add / nr;
    % apply interp2 to all channels
    rgb=zeros(size(in_rgb));
    if channels == 3
        rgb(:,:,1) = interp2(X,Y,in_rgb(:, :, 1),P,Q,'linear',0);
        rgb(:,:,2) = interp2(X,Y,in_rgb(:, :, 2),P,Q,'linear',0);
        rgb(:,:,3) = interp2(X,Y,in_rgb(:, :, 3),P,Q,'linear',0);
    else
        rgb = interp2(X,Y,in_rgb,P,Q,'linear',0);
    end
end
```

B.13 Computing chromatic aberration diopters

```
%-------------------------------------------------------
% f_CalcChromaticDiopter(D_lens, i_lambda, lambda, bpar)
%-------------------------------------------------------
% Compute diopters for chromatic aberration
function [N] = f_CalcChromaticDiopter(D_lens, i_lambda, lambda, bpar)
    mu_r = 1.4998;
    mu_g = 1.5085;
    mu_b = 1.5152;
    mu_y = 1.5085;
    if (bpar.num_components == 3) && (bpar.apply_chromatic_aberration == 1)
        if i_lambda == 1
            n_t = mu_r;
        elseif i_lambda == 2
            n_t = mu_g;
        elseif i_lambda == 3
            n_t = mu_b;
        end
        N = (n_t - 1) * D_lens / (mu_y - 1);
    else
        N = D_lens;
    end
end
```

B.14 Computing vertex-distance adjusted diopters

```
%-----------------------------
% f_CalcNewDiopter(D_lens, vd)
%-----------------------------
% Recompute diopters considering vertex distance
function [N] = f_CalcNewDiopter(D_lens, vd)
    d = vd; % vertex distance in mm
    N = D_lens / (1 - (d * 0.001) * D_lens);
end
```

B.15 Loading DNG image files

```
%--------------------------
% simply_load_dng(filename)
%--------------------------
% Read a .DNG file
function [meta_info, raw] = simply_load_dng(filename)
    warning off MATLAB:imagesci:tiffmexutils:libtiffWarning
    warning off MATLAB:imagesci:tiffmexutils:libtiffErrorAsWarning
    warning off MATLAB:imagesci:tifftagsread:badTagValueDivisionByZero
    t = Tiff(filename,'r');
    offsets = getTag(t,'SubIFD');
    setSubDirectory(t,offsets(1));
    raw = read(t);
    close(t);
    clear t;
    meta_info = imfinfo(filename);
end
```