
Real-Time Simulation of Accommodation and Low-Order Aberrations of the Human Eye using Light-Gathering Trees

Alex R. Cunha Lima · Arthur M. Medeiros · Vitor G. Marques ·
Manuel M. Oliveira

Abstract We present a real-time technique for simulating accommodation and low-order aberrations (*e.g.*, myopia, hyperopia, and astigmatism) of the human eye. Our approach models the corresponding point spread function, producing realistic depth-dependent simulations. Real-time performance is achieved with the use of a novel light gathering tree data structure, which allows us to approximate the contributions of over 300 samples per pixel under 6 ms per frame. For comparison, with the same time budget, an optimized ray tracer exploring specialized hardware acceleration traces two samples per pixel. We demonstrate the effectiveness of our approach through a series of qualitative and quantitative experiments on images with depth from real environments. Our results achieved SSIM values ranging from 0.94 to 0.99 and PSNR ranging from 32.4 to 43.0 in objective evaluations, indicating strong agreement with the ground truth.

Keywords Real-Time Visual Simulation, Low-Order Aberrations, Partial Occlusions, Ray Tracing

1 Introduction

Vision is arguably our most important sense. It is a personal experience influenced by intrinsic characteristics of one’s visual system. Thus, achieving faithful simulations for a given individual, besides being a highly complex task, would require, in principle, a lot of information from a wide variety of areas, ranging from optics and physiology to psychology and neuroscience [12, 25]. However, obtaining such data tends to be impractical

and this level of precision might not be justifiable. We present a technique to produce real-time simulations of how a typical individual would perceive real scenes considering accommodation and low-order aberrations (*e.g.*, *myopia*, *hyperopia*, and *astigmatism*). By taking into account pupil size, our simulations naturally produce realistic depth-of-field effects. All parameters can be dynamically changed during the simulation. In contrast to our solution, previous techniques that perform similar tasks are either limited to a single depth [12], positive defocus [29], or lack precision for handling partially-occluded objects [2].

In our technique, a real scene is represented by an RGB-D image, and the scene depth is discretized as a set of distances (planes) displaced in diopters (m^{-1}). Given the vision parameters to be simulated, a novel light-gathering tree (LGT) data structure is used to accelerate the intersection of rays, leaving the viewer’s pupil, with the scene elements (*i.e.*, the RGB-D image). An LGT implicitly represents the viewer’s depth-dependent *point spread function* (PSF). A simulated view is obtained by using an LGT to compute, for each output pixel p , a weighted average of the colors in the input RGB-D image that contribute to the color of p .

Fig. 1 illustrates the use of our technique for simulating accommodation and low-order aberrations on a scene containing elements at various distances from the observer. Such distances are represented in Figure 1c. 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 1a shows a simulated view of a myopic subject with 0.5 diopters and no accommodation (thus focusing 2 m away). Note that the texture of the game box, which is located at approximately 2 m, exhibits a sharp texture. As the distance decreases towards the white and blue flower, blurring increases. In particular, note the blurry blue petal against the

Alex R. Cunha Lima, Arthur M. Medeiros, Vitor G. Marques,
Manuel M. Oliveira
Instituto de Informática, UFRGS, Porto Alegre, RS, Brazil
E-mail: {arclima, amedeiros, vitor.marques, oliveira}@inf.ufrgs.br

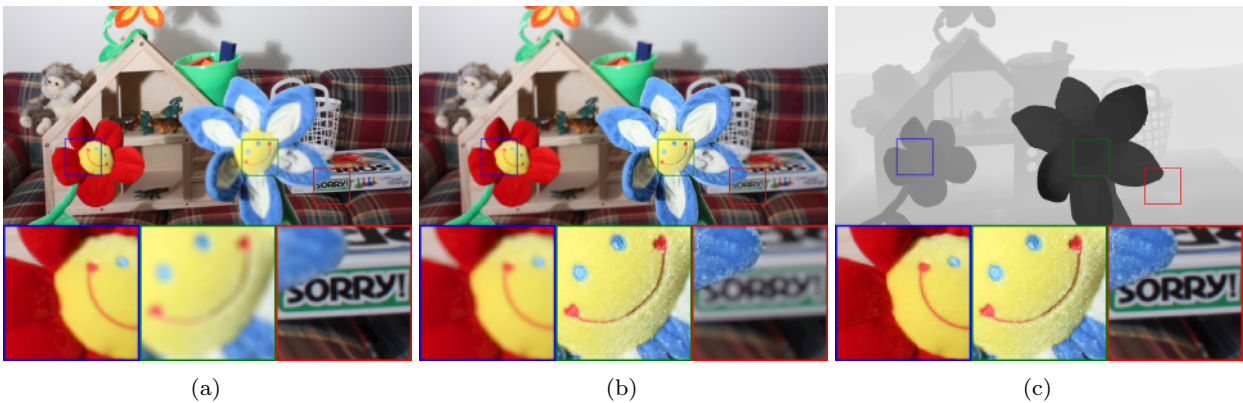


Fig. 1: Simulation of accommodation and low-order aberrations using our technique. Approximate distances of the scene objects are: white and blue flower, 0.5 m; red flower, 1 m; and game box, 2 m. (a) Simulated view of a myopic subject with 0.5 diopters and no accommodation. Starting from a relatively sharp game box texture, the amount of blur increases as the distance decreases towards the white and blue flower. Note the blurry blue petal against the sharp game box texture. The 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 appears sharp while the game box texture becomes blurry. (c) Scene depth (darker means closer) (top) and reference sub-images (bottom).

sharper game box texture. The corresponding reference sub-images are shown at the bottom of Figure 1c, with all elements sharp regardless of their depths. Figure 1b shows a simulated 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 (at an intermediate depth) shows some defocusing with respect to its reference image.

The **contributions** of this work include:

- A real-time technique for producing vision realistic simulations of real scenes considering low-order aberrations and accommodation (Section 4);
- A tree data structure for accelerating vision-realistic rendering that also handles partial occlusions among objects in the presence of a finite pupil (Section 4).

2 Related Work

Vision simulation techniques fall into two major categories: *depth-of-field* and *visual aberrations simulations*.

2.1 Depth-of-Field Simulation

Potmesil and Chakravarty used wave optics to simulate defocus and depth of field (DOF) by means of an integration process [21]. Their algorithm leads to visual artifacts at occlusion borders due to *partial occlusions*. Such artifacts are avoided with the use of distribution ray tracing [6], which shoots rays starting at various

points on the camera aperture. The ideas from both techniques were later united in an algorithm that classifies scene objects into foreground and background fields, filters them separately using PSFs for the corresponding distances, and finally composites the blurred sub-images [26]. Later, the ray distribution buffer (RDB) approach [27] addressed the partial occlusion artifacts by averaging the contributions of several rays over a pixel and treating occlusion according to ray direction. Kraus and Strengert used GPU-based pyramidal interpolation of scattered pixel data to interpolate missing color and depth values [11]. Schedl and Wimmer addressed the partial occlusion problem in depth-of-field simulations using depth peeling to access occluded scene information [24]. More recently, Xiao et al. [29] used convolutional neural networks to obtain real-time simulations of DOF. All techniques discussed so far were applied to synthetic images only. In contrast, we demonstrate our technique on real scenes.

2.2 Visual Aberration Simulation

Krueger et al. [12] used Zernike polynomials to reconstruct the wavefront error resulting from low-order aberrations, using information directly available in spectacle prescriptions. 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 planar textured surfaces (*e.g.*, eye charts). Cholewiak et al. [5] used ray tracing to simulate defocus and low-order aberrations in synthetic images.

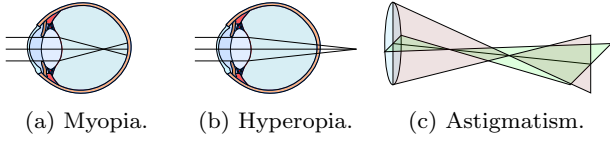


Fig. 2: Myopia, hyperopia, and astigmatism.

Lee et al. [13] used depth peeling to access occluded scene information to simulate optical aberrations also in synthetic images. In contrast to these solutions, our technique simulates low-order aberrations on real scenes containing objects at multiple depths and represented by single RGB-D images.

Barsky et al. used optical information from a human subject, supplied by a Shack-Hartmann aberrometer, to model a wavefront that characterizes the subject’s visual system [1, 3]. 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 a synthetic scene. A set of planes regularly spaced in diopters is placed in the scene. For a given scene, disjoint sub-images are created using pixels whose depth is closest to each plane. The sub-images are then convolved with PSFs computed for each plane and re-combined with alpha compositing (similar to [26]). Simply compositing the convolved sub-images produces undesirable artifacts at occlusion borders. In contrast, our technique does not require information from aberrometers, works on real scenes, and handles occlusion borders.

3 Background

Refractive errors cause light to not be properly focused on one’s retina. The two types of refractive errors relevant to this work are *defocus* and *ophthalmic astigmatism*. They are also characterized by wavefront errors and described by first and second order Zernike polynomials [28], and as such are called *low-order aberrations*.

Defocus comprises *myopia* (nearsightedness) and *hyperopia* (farsightedness). Myopia makes the eye unable to focus at distant objects. Hyperopia, in turn, prevents the eye from focusing at close objects. In both cases, the cone of light entering the eye projects itself on the retina as a circular region (Fig. 2 (a) and (b)), causing blurred vision.

Ophthalmic astigmatism results from the shape of the eye not being equally curved in all meridians. As a consequence, a point is imaged as two spots formed at different planes (Fig. 2c). Due to its anisotropic nature, characterizing astigmatism requires two other numbers in addition to the spherical power S : an *cylindrical power* C (in diopters), and an *angle* φ (in degrees), measured

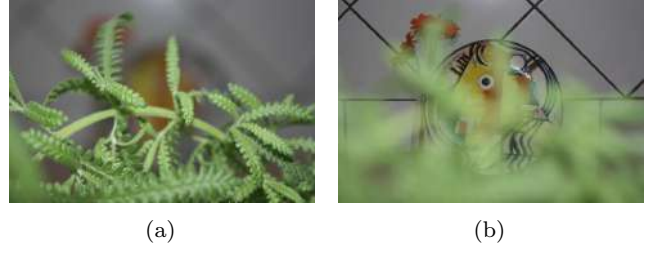


Fig. 3: Partial occlusion effects. (a) An in focus opaque foreground object blocks part of the blurry background. (b) When the background is in focus, the foreground object appears translucent.

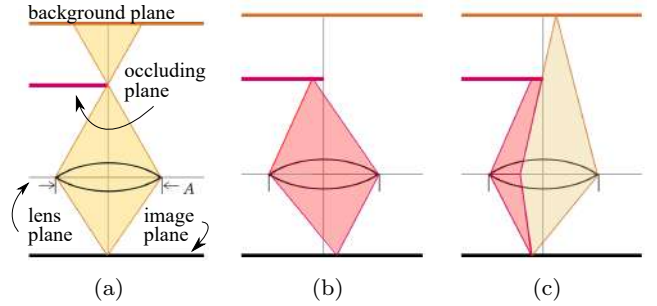


Fig. 4: Schematic representation of partial occlusion. (a) A finite area on the background contributes to a single point on the image plane. (b) A point on the foreground is imaged to a point on the image plane. (c) A point on the background and a finite region on the foreground both contribute to the same image point.

counterclockwise from the horizontal direction). φ defines the orientation of the so-called *cylindrical axis*, and C indicates the optical power perpendicular to the axis.

3.1 Partial-Occlusion Effects

A very challenging aspect of realistic vision simulation is dealing with partial occlusion effects. This is illustrated in Fig. 3, which shows a scene with a foreground plant (occluder) and some elements in the background (flower, plate, and wall). When the foreground is in focus (Fig. 3a), it appears opaque and occludes portions of the background, which in turn appears blurred. However, when the background is in focus (Fig. 3b), the blurred foreground exhibits a see-through effect.

Fig. 4 shows a schematic view of the partial-occlusion problem. Due to the finite lens and aperture (*i.e.*, pupil) sizes, when focusing on the foreground (Fig. 4a), a finite region on the background contributes to the formation of a single point on the image plane. The mixture of colors from those regions causes the background to appear blurry. In contrast, a single point on the occluder solely

contributes to a single point on the image plane (Fig. 4b). In this case, no color mixing between background and foreground occurs and the foreground appears sharp.

Fig. 4c illustrates the situation when the background is in focus. In this case, a single point on the background reflects several rays that contribute to a point on the image plane, whereas a finite area on the occluder also contributes to the formation of that very same point. This time, background and foreground colors mix together, resulting in an apparent translucency of the occluder (Fig. 3b). If information from the entire scene is available, all the effects in Fig. 4 can be simulated using ray tracing. Image-based solutions, however, have trouble simulating these effects. Our technique produces a plausible artifact-free solution for partial occlusions.

4 Simulating Low-Order Aberrations

Partial-occlusion effects emerge naturally from ray casting. One possible way of performing ray casting against an RGB-D image is to use a fragment-shader-based ray-height-field intersection solution performed in texture space [19, 18]. This would, however, require casting a cone of rays for each individual output pixel, which would hurt performance. We propose a different strategy that employs scene-depth discretization and a tree data structure to accelerate rendering.

We discretize the depth range spanned by the input RGB-D image using planes regularly spaced in diopters. Thus, the depth of each input pixel is approximated to the depth of its closest plane in the scene. While this introduces some discretization error, since diopters are expressed as the inverse of the distance in meters, such an error will be smaller closer to the viewer, where it would be more noticeable. Moreover, it can be bound by choosing the number of planes used for the discretization. In practice, we have not observed artifacts due to depth discretization. The concept of depth-range discretization is illustrated in Fig. 5a using only three planes for simplicity. For all scenes shown in this paper, with the exception of the eye chart ones, we used 14 planes to discretize their depth ranges.

We use a tree data structure to guide the traversal of rays into the resulting discrete scene representation. This data structure tracks the ray paths that contribute to the central output pixel (Fig. 5b). The same relative paths computed for the central one are used to evaluate the remaining output pixels. The rationale behind such a reuse is the fact that the *fovea* is a small region (on the retina) with approximately 1 degree in diameter that does not exhibit significant changes in aberrations [4]. The human visual system builds a mental picture of their surroundings by systematically scanning the scene

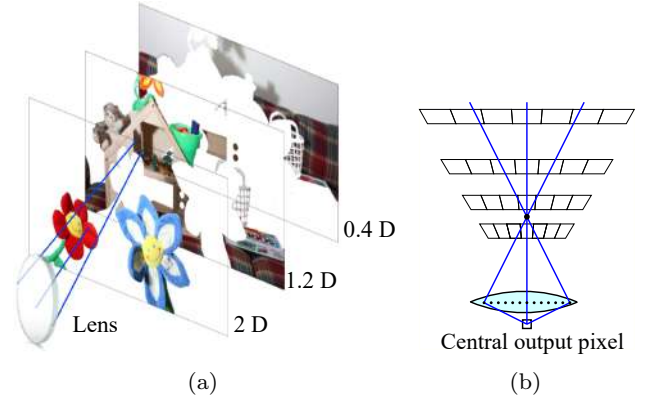


Fig. 5: Plane-discretized scene and the isoplanatic assumption. (a) Each pixel is assigned to its closest plane (in dioptic distance). (b) Rays are only cast for the formation of a single on-axis output pixel, and reused for all other points (isoplanatic assumption).

and projecting it on the fovea. Therefore, the most prominent perceived aberrations will be those registered by 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* [4] and works well in practice. Other researchers make the same assumption, albeit implicitly [21, 26, 12].

4.1 Light-Gathering Trees

The paths of the light rays starting at the central pixel in Fig. 5b define a tree-like structure. Its nodes store the relative pixel positions (computed with respect to the central one) of the potential scene elements that contribute to the central output pixel. The intuition behind using a tree to emulate ray casting is shown in Fig. 6. When a large number of rays are cast into the scene (Fig. 6a), a significant number of them end up traversing the same cells (pixels in a given plane of the scene discretization). We replace such groups of rays by arrows, as shown in Fig. 6b, where their weights indicate the number of rays represented by each arrow. This hierarchical sequence of arrows define a tree data structure: a *light-gathering tree* (LGT). Ray casting, in this case, is equivalent to traversing this tree from the root to its leaves. Furthermore, an LGT is *computed once* and reused for all pixels of the target image based on the isoplanatic assumption.

If a scene element is present on a certain plane, rays may end up reaching a cell behind it on the next plane, where no information is available (Fig. 3a, Fig. 6c). Our technique handles this situation by accessing color

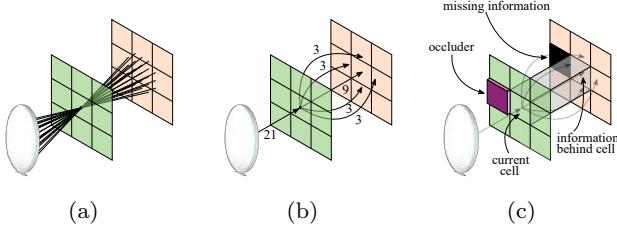


Fig. 6: Using a tree to gather scene information. (a) Rays cast into the scene. (b) Tree structure with weights on arrows indicating number of rays in bundles. (c) Occluder causes missing information in the second plane.

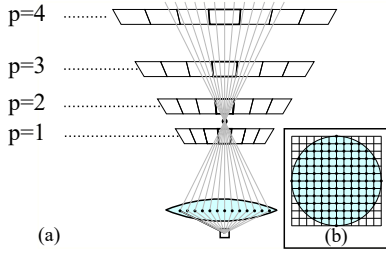


Fig. 7: Scene sampling rays emerging from the sensor central pixel are bent by a lens and converge to a focal point along the optical axis of the lens, and diverge after passing through it. (b) A regular grid defines the sampling positions on the lens.

information behind the cell visited just before hitting the occluded one (Fig. 6c) and using that as the missing color. Although the missing color information is unknown, our *disocclusion* technique provides a plausible solution (due to spatial coherence) and avoids the occurrence of color artifacts at depth discontinuities.

4.2 Light-Gathering Tree Construction and Usage

Given an input RGB-D image representing a scene, the nodes of an LGT are arranged according to a few parameters: *number of planes*, *nearest and farthest plane distances*, *focal distance*, *pupil size*, *number of traversal rays*, and *simulated low-order aberration parameters* (S , C , φ). Rather than casting rays into the scene in the traditional way during runtime, we traverse the LGT and sample the input image to perform the simulation. Thus, for a given subject, the nodes of an LGT need to be updated when the focal distance changes, as this leads to a change in accommodation. Updating the nodes of an LGT is a quick operation. For a typical LGT with 14 planes and 314 rays, it takes approximately 2.4 ms on a Core i5 3.2 GHz CPU using unoptimized C# code.

The process of building an LGT is illustrated in Figs. 6a and 6b. Nodes are all linked from parent to

child cells. Each node stores the following information: parent node, list of children nodes, cell position, number of intercepted rays, and weight (see Data Structure 1). To exploit the isoplanatic assumption, cell positions are stored as 2D coordinates relative to the center cell. Fig. 7 illustrates the process of sampling a scene to build an LGT. Each plane corresponds to a level of the LGT. Fig. 7b shows a regularly spaced grid over a disk representing the subject's pupil. The grid crossings are the starting positions of rays cast into the scene towards a focal point along the optical axis. Algorithm 1 summarizes the procedure for constructing LGTs.

Data Structure 1: Node structure

```

1 struct Node:
2   Node* parent           // parent node
3   List<Node*> children
4   int x, y               // relative pos
5   int rayCount           // # of traversed rays
6   float wgt              // rayCount / total # rays

```

Algorithm 1: LGT Construction

```

input  : RGB-D image (Img), closest (CP) and
         farthest (FP) plane distances, pupil size (A),
         number of planes (NP), focal distance (f),
         number of rays (NR), aberration
         parameters (S, C,  $\varphi$ ), field of view (FOV)
output : LGT represented by s as root node
1 s ← CreateOrReuseNode(0, 0, null)
2 for ray ← 1 to NR:
   // depends on f, S, C,  $\varphi$ , A, NR
3   Compute  $x_s, y_s, x_\Delta, y_\Delta$  for this ray
4   p ← s // first parent is root
5   for plane ← 1 to NP:
6     z ← PlaneDist(plane) // Use CP, FP, NP
       // discret. x, y using FOV, Img size
7     (x, y) ← mkCellCoord( $x_s + x_\Delta \cdot z, y_s + y_\Delta \cdot z$ )
8     nd ← CreateOrReuseNode(x, y, p)
9     nd.rayCount ← nd.rayCount + 1
10    nd.wgt ← nd.rayCount / NR
11    p ← nd // next parent is curr node

```

Vision-realistic simulation is performed by gathering the contributions of each portion of the scene hit by the LGT rays, which are then integrated into the final colors of the corresponding output pixels. Algorithm 2 summarizes this process.

4.3 Determining Ray Directions for Astigmatism

LGTs support one or two focal points in order to simulate defocus (myopia, hyperopia, and accommodation)

Algorithm 2: LGT Usage

```

input : RGB-D image (Img), LGT's root node (s)
output : RGB image (OutImg) vision rendering
1 OutImg  $\leftarrow$  mkBlankImg(Img.Size)
  // Iterate over all image pixels
2 for  $c_x, c_y \leftarrow$  PixelCoordinates(Img):
3   OutImg[ $c_x, c_y$ ].RGB  $\leftarrow$  GatherLight(s, 0)
4 return OutImg
  // npi:node plane idx, ipi:image plane idx
5 function GatherLight (nd, npi):
6    $z \leftarrow$  Img[ $c_x + nd.x, c_y + nd.y$ ].Depth
7   ipi  $\leftarrow$  mkPlaneIndex(z)
8   if ipi < npi:
9     p  $\leftarrow$  nd.parent
10    return nd.wgt  $\cdot$  Img[ $c_x + p.x, c_y + p.y$ ].RGB
11  elseif ipi = npi:
12    return nd.wgt  $\cdot$  Img[ $c_x + nd.x, c_y + nd.y$ ].RGB
13  wgt  $\leftarrow$  0
14  for each child in nd.children:
15    wgt  $\leftarrow$  wgt + GatherLight(child, npi + 1)
16  return wgt

```

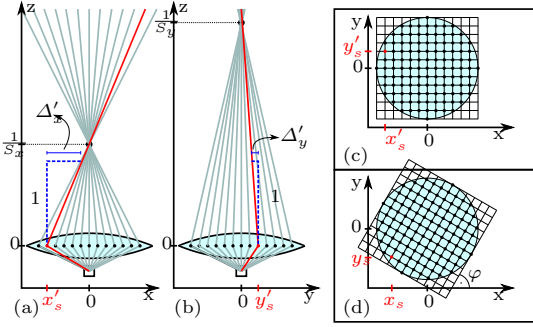


Fig. 8: Ray casting for an astigmatic optical system using a left-handed coordinate system. Each ray is defined by a starting position $(x_s, y_s, 0)$ and direction $(x_\Delta, y_\Delta, 1)$, given by Eqs. (1) and (2), respectively. (a) Top view (xz plane) of the scene. (b) Side view (yz plane) of the scene. (c) Back view (xy plane) of the scene when $\varphi = 0$. (d) Back view (xy plane) of the scene when $\varphi \neq 0$.

and astigmatism, respectively. We adopt a left-handed coordinate system, as shown in Fig. 8. For the case of astigmatism, the cylinder axis lies on the xy plane making an angle φ with the horizontal axis (Fig. 8d). For each ray, its starting position $(x_s, y_s, 0)$ and direction $(x_\Delta, y_\Delta, 1)$ are obtained using Eqs. (1) and (2):

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{bmatrix} \begin{bmatrix} x'_s \\ y'_s \end{bmatrix}, \quad (1)$$

$$\begin{bmatrix} x_\Delta \\ y_\Delta \end{bmatrix} = \begin{bmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{bmatrix} \begin{bmatrix} x'_\Delta \\ y'_\Delta \end{bmatrix}, \quad (2)$$

where $(x'_s, y'_s, 0)$ and $(x'_\Delta, y'_\Delta, 1)$ correspond to the ray starting position and direction for the case of an axis-aligned grid (*i.e.*, $\varphi = 0^\circ$) (Fig. 8c). For defocus, φ is

always 0, $x'_\Delta = -x'_s S$, and $y'_\Delta = -y'_s S$. For astigmatism, the values of x'_Δ and y'_Δ are computed as

$$x'_\Delta = -x'_s S \quad \text{and} \quad y'_\Delta = -y'_s S_{pC}, \quad (3)$$

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

5 Results

We implemented the described technique as a Unity compute shader as well as a MATLAB script. We evaluated our method both qualitatively and quantitatively. The results of these experiments show that it produces realistic simulations of accommodation and low-order aberrations. For a typical scene, such as the one shown in Fig. 1 our simulation takes about 6 ms per frame on an RTX 2080 Ti GPU.

Performing objective comparisons requires ground truth data. While low-order aberrations can be induced by placing extra lenses in front of a camera, getting reliable depth values for these same scenes is a non-trivial task. According to our experience, RGB-D sensors such as the Microsoft Kinect, StereoLabs ZED camera, and Intel RealSense tend to produce invalid depth values at depth discontinuities, and the severity of this issue increases with the proximity of the objects to the sensor (larger disparity). This is a highly undesirable condition for our application, as depth discontinuities are critical for simulating partial occlusions, and the results of the simulations can be better appreciated at close range where objects present more details.

An alternative to RGB-D sensors is the use of CNN-based single-image depth-estimation techniques [22, 14], which can produce impressive results given the difficulty of the problem they try to solve. While their results can be used for applications such as One Shot 3D Photography [9] that only requires relative and approximate depth values, they are not appropriate for our context. Fig. 9 compares depth maps obtained using a multiview stereo [23] and the CNN-based technique in [22] and shows that CNN-based techniques suffer from various limitations. For instance, they can change the depth order of scene objects (see the left flower's stem compared to its petals in Fig. 9 (top)), incorrectly assign depth variations based on color differences (see the blue and white petal in Fig. 9 (top)), and miss important details (*e.g.*, the net in Fig. 9 (bottom)).

Due to the difficulty of acquiring ground-truth depth for real scenes, in this paper we perform qualitative evaluations on several photographs for which depth has been estimated using computer vision techniques (*e.g.*, Fig. 1). We limit quantitative evaluations to real scenes

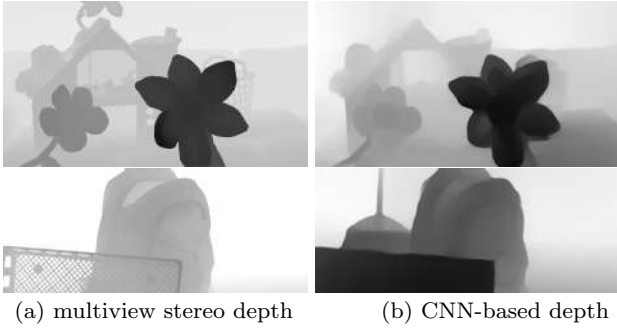


Fig. 9: Comparisons of depth maps generated by multi-view stereo (a) and by a CNN-based single-image depth-estimation technique [22] (b).

containing a single plane, for which depth can be reliably estimated.

5.1 Qualitative Evaluation

For the qualitative experiments, we use a set of RGB-D images whose depth ranges cover several diopters. The images were obtained from a stereo online dataset that offers 23 color images with corresponding disparity maps, which were then converted to depth information [23]. The per-pixel depth values expressed in meters were computed as:

$$Z = \frac{b \times f / (d + dpp)}{1000}, \quad (4)$$

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 [23]. 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. Early tests have shown that even slightly misplaced depth values can result in noticeable artifacts in the simulations, such as the introduction of light or dark auras around objects. To avoid such artifacts, we manually adjusted the depth map using the distance from objects that were correctly registered and roughly correspond to the same depth.

Figure 1 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 Fig. 1a the subject is focusing on the game box, causing the blue and white flower to appear blurry. In Fig. 1b the focus has moved to the blue and white flower, making the game box to look defocused. Fig. 10 illustrates the view of hyperopic subject (-0.3 D) observing the same scene

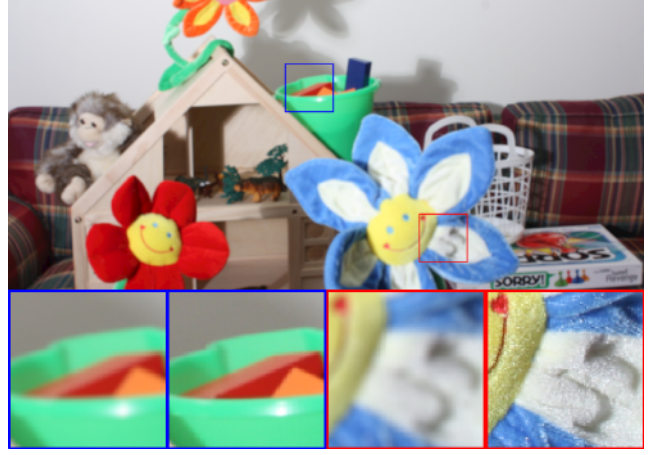


Fig. 10: Simulated view of a hyperopic subject (-0.3 D). Note how closer objects appear blurrier than far away ones. The pairs of sub-images compare simulated (left) and original (right) patches.

showing in Figure 1. Note how closer objects appear blurrier than far away ones.

We evaluate the performance of our LGT-based approach to simulate accommodation and low-order aberrations by comparing it against an optimized ray tracer written using the NVIDIA Optix ray tracing engine [17]. The measurements were performed on an i5 3.2 GHz CPU with 16 GB DDRAM and an NVIDIA RTX 2080 Ti GPU with 11 GB of memory and 68 ray-tracing acceleration cores. The Optix engine takes advantage of these acceleration cores. The LGTs were created on-the-fly on the CPU by tracing 314 rays (Fig. 7).

Fig. 11 compares a simulation produced by our technique against the Optix ray-traced result for a myopic individual with 3 diopters (*i.e.*, $S = 3$ D). Both approaches used the same plane-discretized scene representation: $1,440 \times 990$ pixels and 14 scene planes. The ray tracer was adapted to apply the same disocclusion method as ours (described in the last paragraph of Section 4.1), as without it the ray-traced images would exhibit distracting artifacts. For this example, our simulation takes approximately 6 ms per frame. Under the same time budget, the Optix ray tracer can trace 2 samples per pixel, producing a noisy image (Fig. 11 bottom right). By exploiting the isoplanatic assumption, our technique harnesses the LGT structure to approximate the contributions of 314 samples per pixel, thus producing smooth anti-aliased results (Fig. 11 bottom left).

Figure 12 shows a scene containing an Adirondack chair. The input color and depth images are shown in Figs. 12a and 12b. The scene’s plane index map is illustrated in Fig. 12c, where each color indicates a different plane numbered from 0 (closest) to 13 (farthest) accord-



Fig. 11: Simulated view of myopic subject ($S = 3$ D) using our technique (left) and ray tracing (right), both running in roughly 6 ms. The ray-traced image shows artifacts due to the usage of only two samples per pixel

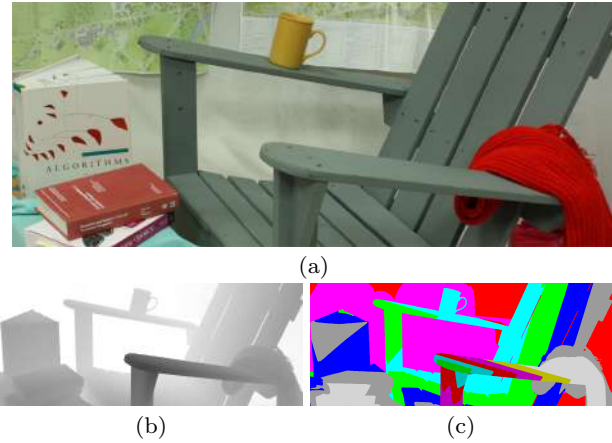


Fig. 12: Adirondack chair. (a) Reference image. (b) Depth image. (c) Field discretization plane set.

Table 1: Plane indices and their encoded colors.

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

ing to Table 1. Figs. 13a and 13b 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 dipters and, as such, its image appears sharper in Fig. 13b compared to its appearance in Fig. 13a.

Fig. 14 illustrates the combined simulation of myopia and astigmatism ($S = 1$ D, $C = 3$ D, $\varphi = 20^\circ$). Note

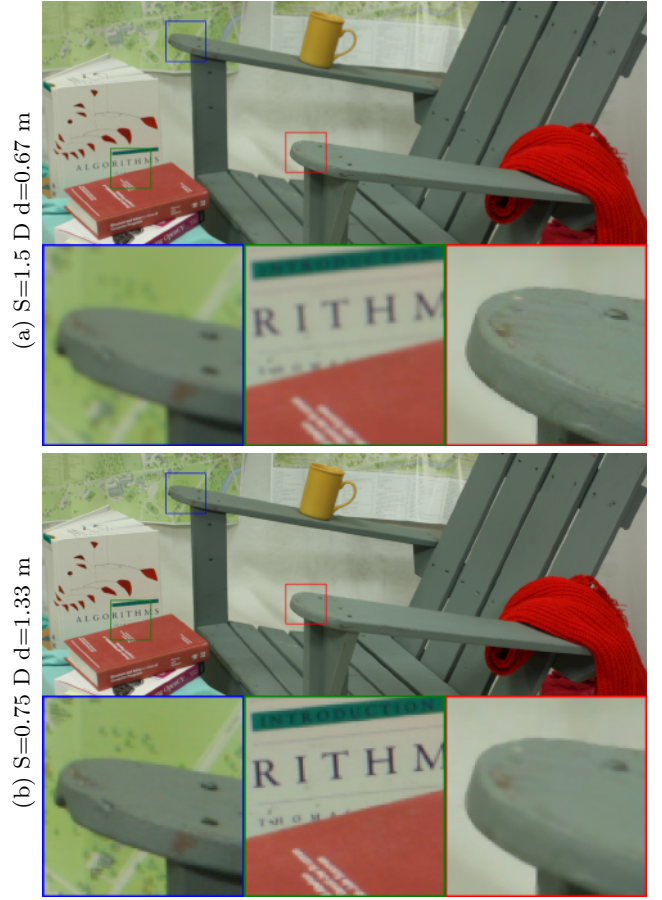


Fig. 13: Myopic simulations for 1.5 D (a) and 0.75 D (b), which correspond to focusing at the two armrests (red and blue insets) located approximately at 0.67 m and 1.33 m from the observer. The white book (green inset) is closer to the farthest armrest and, therefore, appears sharper in Fig. 13b.

that the anisotropic blurriness on the book cover and on the mug handle (red inset) is more pronounced at 110° , the direction perpendicular to φ . Along such direction, the dioptric power is given by $S+C = 4$ D.

Fig. 15 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. Fig. 16a 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). Fig. 16b shows a simulation 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).

The supplemental materials include a video, captured in real time, illustrating the use of our technique.



Fig. 14: Simulating myopia and astigmatism: $S = 1$ D, $C = 3$ D, $\varphi = 20^\circ$.

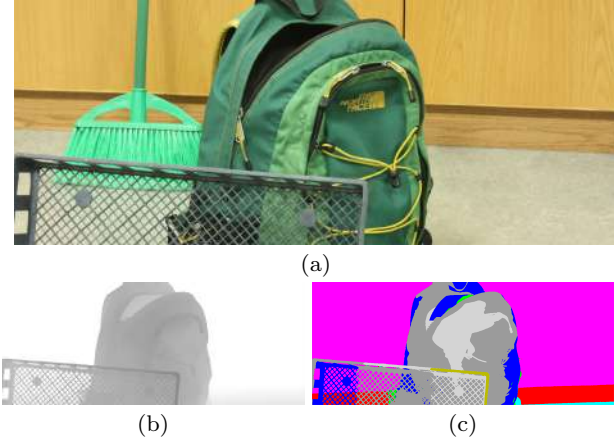


Fig. 15: Backpack and broom on the background. (a) Reference image. (b) Depth image. (c) Field discretization plane set.

5.2 Quantitative Evaluation

For the quantitative evaluation, we took a set of pictures of eye charts with a Canon Rebel T6 DSLR camera putting extra lenses 54 mm in front of the camera's original lens to induce low-order aberrations. The chart was 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 (Table 2). 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. For these eye chart scenes, the resulting LGTs have a single layer.

The use of an external lens introduces changes to the camera's optical system, resulting in changes in



Fig. 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.

magnification, brightness, and chromatic aberrations of the captured ground-truth images. Such changes need to be compensated for in our synthesized results for proper comparisons. Appendix A describes the procedure we used to compensate for these factors.

5.2.1 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. Fig. 17 (a) shows five eye-chart pictures captured with the following camera settings: ISO 100, exposure $1/40$ s, $f=20$ mm, and $f/5$. Fig. 17 (b) shows the pictures in Fig. 17 (a) after minification and brightness correction. Such minification was performed using MATLAB's interpolation function `interp2`. Fig. 17 (c) shows the ground-truth pictures captured by placing an external lens (whose parameters



Fig. 17: Inducing aberrations by placing extra lenses in front of a camera ($v=54$ 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 obtained with an extra lens in front of the camera. (d) Simulated results produced by our technique.

are described in the corresponding rows of Table 2) in front of the camera lens. Fig. 17 (d) shows the simulated results produced by our technique using Fig. 17 (b) as input and not taking chromatic aberration into account. Note their similarity to the corresponding ground-truth images. Table 2 shows SSIM and PSNR values for the results shown in Fig. 17 (d). SSIM values ranged from 0.94 up to 0.99 and PSNR values ranged from 32.4 up to 43.0, both with and 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 improve the results to justify its additional computation. Thus, in the paper, we only show simulated results without considering chromatic aberration.

Table 2: Extra lens parameters, SSIM and PSNR values for the simulated results in Fig. 17, with (CA) and without (NCA) considering chromatic aberration. "Row" is the row number in Fig. 17.

Row	Lens			CA		NCA	
	S (D)	C (D)	φ	SSIM	PSNR	SSIM	PSNR
1	0	-1	86°	0.959	34.426	0.959	34.401
2	0	-0.5	82°	0.942	32.434	0.942	32.411
3	-0.25	-2.25	69°	0.952	36.382	0.950	36.312
4	+1	0	0°	0.986	42.220	0.987	42.370
5	+2.25	0	0°	0.991	43.204	0.991	43.078

5.3 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. It also does not take into account high-order aberrations, which can be represented with Zernike polynomials. High-order aberrations, however, tend to cause only subtle, hard to notice artifacts, which are not corrected for by spectacles or by contact lenses.

Wavefront errors are a function of accommodation, meaning that when a subject changes focal distance, aberrations might change as well [8]. 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 $0.5 \mu\text{m}$ for each of the astigmatism coefficient in the Zernike polynomials [8]. Our technique disregard such minor effects.

One should note that our simulation consists of re-sampling RGB-D images representing scenes. Therefore, it is not prone to chromatic aberration nor requires brightness adjustment. These issues were discussed in Section 5.2.1 and in Appendix A, respectively, in the context of the objective validation using additional lenses to induce low-order aberrations. The use of these lenses introduced chromatic aberration and brightness reduction, which needed compensation for proper quantitative evaluation using SSIM and PSNR metrics.

6 Conclusion

We presented a real-time technique for simulating accommodation and low-order aberrations of the human eye, considering real scenes. Real-time performance is achieved with the use of a novel tree data structure (LGT) in combination with a discretization of the scene depth using a uniform spacing in diopters. The use of the isoplanatic assumption leads to a light data structure that only needs to store the paths of rays cast through

few tree nodes. The LGT is built from an RGB-D image representing the scene, and low-order aberration parameters (S , C , φ), focal distance, and pupil size.

We demonstrated the effectiveness of our technique through a series of qualitative and quantitative experiments. For a typical scene represented by a $1,440 \times 990$ -pixel RGB-D image using 14 planes, our simulation takes about 6 ms per frame to approximate the contributions of 314 samples per pixel. Under the same time budget, an optimized ray tracer exploring hardware acceleration on a modern GPU only traces two samples per pixel.

Besides graphics applications, our technique 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 a training tool for medical students.

6.1 Future work

Combining our simulation with tracking technologies [10] in virtual-reality environments could be an effective way to allow one to experience varying levels of low-order aberrations. This could also be used to obtain realistic simulation of depth-of-field blur in VR applications.

An interesting direction of widening the scope of our simulations to consider microscopic environments is design LGTs that can handle diffraction effects using Wigner functions [15] to represent rays using the Huygens-Fresnel principle.

The use of a separable bokeh technique [16,7] could further improve the algorithm's performance. However, this would probably be limited to myopia and hyperopia, since the astigmatic bokeh is not circularly symmetric. Finally, one could implement higher-order aberrations replacing ray direction determination by sampling along the wavefront normals.

Compliance with Ethical Standards

Conflict of Interest: The authors declare they have no conflict of interest.

Funding: This work was funded by CNPq-Brazil (fellowships and grants 312975/2018-0, 423673/2016-5 and 131288/2016-4), and CAPES Finance Code 001.

References

1. Barsky, B., W. Bargteil, A., D. Garcia, D., Klein, S.: Introducing vision-realistic rendering. In: Proc. 13th Eurographics Workshop on Rendering, pp. 1–7 (2002)
2. Barsky, B.A.: Vision-realistic rendering: Simulation of the scanned foveal image from wavefront data of human subjects. In: Proc. of APGV '04, pp. 73–81. ACM (2004)
3. Barsky, B.A., Tobias, M.J., Horn, D.R., Chu, D.P.: Investigating occlusion and discretization problems in image-based blurring techniques. In: Proc. Vision, Video, and Graphics, VVG 2003, pp. 97–102 (2003)
4. Bedggood, P., Daaboul, M., Ashman, R.A., Smith, G.G., Metha, A.: Characteristics of the human isoplanatic patch and implications for adaptive optics retinal imaging. *Journal of Biomedical Optics* **13**(2), 1 – 7 – 7 (2008)
5. Cholewiak, S.A., Love, G.D., Srinivasan, P.P., Ng, R., Banks, M.S.: Chromablur: Rendering chromatic eye aberration improves accommodation and realism. *ACM Trans. Graph.* **36**(6), 210:1–210:12 (2017)
6. Cook, R.L., Porter, T., Carpenter, L.: Distributed ray tracing. In: Proc. of SIGGRAPH '84, pp. 137–145 (1984)
7. Garcia, K.: Circular separable convolution depth of field. In: ACM SIGGRAPH 2017 Talks, pp. 16:1–16:2 (2017)
8. He, J., Burns, S., Marcos, S.: Monochromatic aberrations in the accommodated human eye. *Vision Research* **40**(1), 41–48 (2000)
9. Kopf, J., Matzen, K., Alsisan, S., Quigley, O., Ge, F., Chong, Y., Patterson, J., Frahm, J.M., Wu, S., Yu, M., Zhang, P., He, Z., Vajda, P., Saraf, A., Cohen, M.: One shot 3d photography. *ACM Trans. Graphics* **39**(4) (2020)
10. Koulieris, G., Akrj, K., Stengel, M., Mantiuk, R., Mania, K., Richardt, C.: Near-eye display and tracking technologies for virtual and augmented reality. *Computer Graphics Forum* **38**(2), 493–519 (2019). DOI 10.1111/cgf.13654
11. Kraus, M., Strengert, M.: Depth-of-field rendering by pyramidal image processing. *Comput. Graph. Forum* **26**, 645–654 (2007)
12. Krueger, M.L., Oliveira, M.M., Kronbauer, A.L.: Personalized visual simulation and objective validation of low-order aberrations of the human eye. In: Proc. SIBGRAP'16, pp. 64–71 (2016)
13. Lee, S., Eisemann, E., Seidel, H.P.: Real-time lens blur effects and focus control. *ACM Trans. Graph.* **29**, 65:1–65:7 (2010)
14. Li, Z., Snavely, N.: Megadepth: Learning single-view depth prediction from internet photos. In: Computer Vision and Pattern Recognition (CVPR) (2018)
15. Luis, A.: Complementary Huygens principle for geometrical and nongeometrical optics. *European Journal of Physics* **28**, 231–240 (2007)
16. Niemitalo, O.: Circularly symmetric convolution and lens blur. <http://yehar.com/blog/?p=1495> (2011)
17. Parker, S.G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., McAllister, D., McGuire, M., Morley, K., Robison, A., Stich, M.: Optix: A general purpose ray tracing engine. *ACM Trans. Graph.* **29**(3), 66:1–66:13 (2010)
18. Policarpo, F., Oliveira, M.M.: Relief mapping of non-height-field surface details. In: Proc. 2006 Symposium on Interactive 3D Graphics and Games, I3D '06, pp. 55–62 (2006)
19. Policarpo, F., Oliveira, M.M., Comba, J.a.L.D.: Real-time relief mapping on arbitrary polygonal surfaces. In: Proc. 2006 Symposium on Interactive 3D Graphics and Games, pp. 155–162 (2005)
20. Polyanskiy, M.N.: Refractive index database. <https://refractiveindex.info>. Accessed 2019-01-23
21. Potmesil, M., Chakravarty, I.: Synthetic image generation with a lens and aperture camera model. *ACM Trans. Graph.* **1**(2), 85–108 (1982)

22. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V.: Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. arXiv:1907.01341 (2019)
23. Scharstein, D., Hirschmüller, H., Kitajima, Y., Krathwohl, G., Nesić, N., Wang, X., Westling, P.: High-resolution stereo datasets with subpixel-accurate ground truth. In: GCPR, *LNCIS*, vol. 8753, pp. 31–42 (2014)
24. Schedl, D., Wimmer, M.: A layered depth-of-field method for solving partial occlusion. *Journal of WSCG* **20**(3), 239–246 (2012)
25. Schwartz, S.H.: Visual perception: A clinical orientation, 4 edn. McGraw-Hill Medical Pub. Division, New York (2010)
26. Scofield, C.: 2 1/2-d depth-of-field simulation for computer animation. In: Graphics Gems III, pp. 36–38. M. Kaufmann (1992)
27. Shinya, M.: Post-filtering for depth of field simulation with ray distribution buffer. In: Proc. Graphics Interface '94, pp. 59–66 (1994)
28. Thibos, L., Applegate, R.A., Schwiegerling, J.T., Webb, R.: Standards for reporting the optical aberrations of eyes. *Journal of Refractive Surgery* pp. S652–S660 (2002)
29. Xiao, L., Kaplanyan, A., Fix, A., Chapman, M., Lanman, D.: Deepfocus: Learned image synthesis for computational display. In: ACM SIGGRAPH 2018 Talks, pp. 4:1–4:2 (2018)

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 [20], and correspond to the wavelengths $\lambda_r=700$ nm, $\lambda_g=510$ nm, and $\lambda_b=440$ nm [12].

A 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

$$M_{\varphi} = \frac{1}{1 - vS} \quad \text{and} \quad M_{\varphi^{\perp}} = \frac{1}{1 - vS_{pC}}, \quad (5)$$

where M_{φ} and $M_{\varphi^{\perp}}$ are, respectively, the magnification factors along the directions that make angles φ and $\varphi + 90^\circ$ with the horizontal axis. In the absence of astigmatism, the magnification is isotropic, with $M_{\varphi} = M_{\varphi^{\perp}}$. The *effective optical power* is then obtained as $S'_{\varphi} = SM_{\varphi}$, which is the same as $S'_{\varphi^{\perp}} = SM_{\varphi^{\perp}}$.

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 one, 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 has only been used for the sake of the validation experiment that uses an external lens. This is not required by the LGT technique itself.

Brightness adjustment is required to compensate for some amount of light that is reflected/absorbed by the extra lens, effectively not reaching the 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 brightness of 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 (Table 2).