Eurographics Symposium on Rendering 2016
E. Eisemann and E. Fiume
(Guest Editors)

Volume 35 (2016), Number 4

Sparse high-degree polynomials for wide-angle lenses

Emanuel Schrade†, Johannes Hanika and Carsten Dachsbacher
Karlsruhe Institute of Technology, Germany.

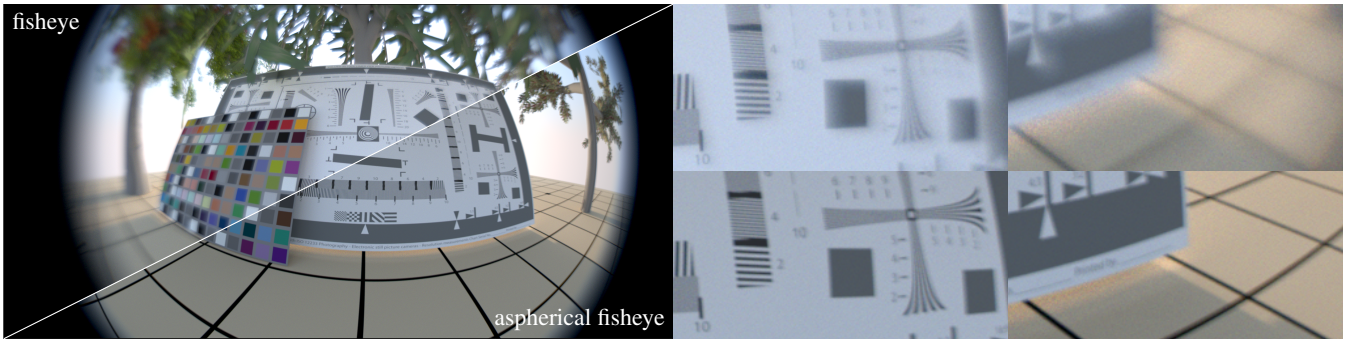

Figure 1: *We accurately model the optical properties of lens systems and show efficient rendering of lens effects using physically-based path tracing as well as interactive previews. The figure shows two different fisheye lens designs at wide open aperture f/2.8 (one is using aspherical elements). The grid on the floor has about 1/3 meter spacing. While they both deliver sharp images in the centre, they show very different, distinct aberrations in lateral and out of focus areas: the spherical fisheye is slightly brighter but has strong coma resulting in overall contrast loss and shallow depth of field. The aspherical lens is better corrected and thus sharper, but shows subtle chromatic aberrations.*

Abstract

Rendering with accurate camera models greatly increases realism and improves the match of synthetic imagery to real-life footage. Photographic lenses can be simulated by ray tracing, but the performance depends on the complexity of the lens system, and some operations required for modern algorithms, such as deterministic connections, can be difficult to achieve. We generalise the approach of polynomial optics*, i.e. expressing the light field transformation from the sensor to the outer pupil using a polynomial, to work with extreme wide angle (fisheye) lenses and aspherical elements. We also show how sparse polynomials can be constructed from the large space of high-degree terms (we tested up to degree 15). We achieve this using a variant of* orthogonal matching pursuit *instead of a Taylor series when computing the polynomials. We show two applications: photorealistic rendering using Monte Carlo methods, where we introduce a new aperture sampling technique that is suitable for light tracing, and an interactive preview method suitable for rendering with deep images.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing

1. Introduction

In photography the lens used to take a shot greatly influences the overall appearance of the result. For instance the bokeh of a lens, i.e. the quality of the out of focus blur, contributes much to how the subject can be separated from the background and the perceived impression of the blur. Similar holds for vignetting or blur caused by spherical aberration.

If such footage is to be combined with synthetic images, e.g. in visual effects work, it is important to maintain the overall look of the lens which has been used to take the real-world images. Recently, creating content for virtual reality headsets has spawned renewed interest in fisheye lenses with field of view around 180 degrees. One approach is to use a near-distortion free lens for the live shoot and simulate the lens effects on both rendering and live footage in post-production. This may require to shoot with more than one camera at once to cover the full 180 degree field of view. The other approach is to closely model the real lens and use this model in rendering.

† schrade@kit.edu

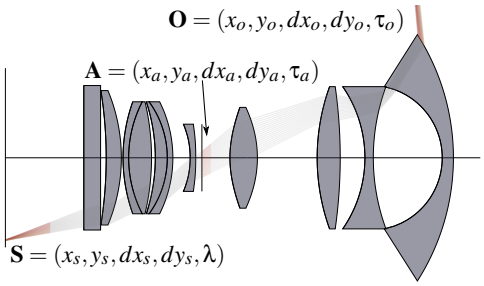

Figure 2: *Schematic of a fisheye lens with aspherical elements (JP,2014-052503,A). The light field in plane/plane parametrisation with wavelength at the sensor $(x_s, y_s, dx_s, dy_s, \lambda)$ is transformed to another plane/plane light field with Fresnel transmittance at the aperture $(x_a, y_a, dx_a, dy_a, \tau_a)$, and further to the outgoing light field on the outer pupil (the last lens element) $(x_o, y_o, dx_o, dy_o, \tau_o)$ which we parametrise as sphere/sphere to support outgoing angles deviating from the optical axis by up to 180 degrees.*

Our method is applicable in both scenarios through a simple and powerful model for the transformation of the 5D light field at the sensor (2D position, 2D direction, wavelength) to a 5D light field at the outer pupil of the lens system (2D position, 2D direction, transmittance). This transformation maps rays on the sensor to the outer pupil and includes aberrations that appear in the lens system. Since the light field transformation achieved by photographic lenses is smooth, polynomials have a long tradition of being used as the natural basis to describe them [Sei57].

In this paper, we extend polynomial optics for computer graphics [HHH12, HD14] to support more generic lenses, while at the same time improving precision, simplifying the procedure, and bounding computational complexity. In particular, we

- re-parametrise the light field to support extreme wide angles (fisheye lenses) (Sec. 3.1) and integrate attenuation due to internal reflection to our model.
- show how to find a high-degree, sparse polynomial model of the lens system (Sec. 3.2) to support for example aspherical elements without the need of an analytic series expansion.
- introduce an aperture sampling method suitable for light tracing (Sec. 4.1) that can be used to connect light path vertices in the scene to the sensor.
- show an algorithm to compute a preview of lens effects from deep images at interactive rates (Sec. 4.2).

We provide source code to produce the lens model including ray tracing, polynomial fitting, and the interactive visualisation.

2. Background and Previous Work

Photographic objectives are typically described as tables with curvature radii and distances to the next element along the optical axis, as well as material parameters including information about the spectral index of refraction [Sch16]. Such lens schematics can be obtained from books [Met48], databases [ODS10], or individual patents, and the full lens geometry can be reconstructed from this (see Fig. 2).

Optics There is a large body of literature on optics [Hec01, Kin92] and lens design [Smi00, Smi05]. The foundation of our work is geometrical optics, i.e. we ignore diffraction and interference. We also assume that the index of refraction is piecewise constant and only changes at the interface in between materials. Extending our ray tracing to support other cases is possible [BG12, ABW14] but we did not explore it.

Matrix Optics Linear (or matrix) optics has been introduced almost two centuries ago [Gau41]. In matrix optics a ray is defined by its distance y to the optical axis and by the angle α between the ray's direction and the optical axis. The calculations can be simplified using the paraxial approximation, i.e. it is assumed that y and α are close to zero. With these assumptions a first-order Taylor series expansion around $(y, \alpha) = (0, 0)$ can be used to construct matrices describing, e.g. the refraction at a spherical element or the propagation within a medium with constant refractive index. A matrix describing a whole lens system can be calculated by simply multiplying the appropriate matrices for each refraction and propagation. With this system matrix many rays can be transferred through the lens system by multiplying their vector representation to the matrix. In computer graphics, ray traced lens flares [HESL11] have been extended using matrix optics to obtain a fast, practical algorithm with lower precision [LE13].

The transformation of a lens system can also be quickly evaluated using polynomial optics which we discuss next.

Polynomial Optics Seidel [Sei57] showed that interesting aberrations only arise for non-linear approximations, i.e. higher order coefficients are required. Hullin et al. [HHH12] approximated lens systems through polynomials to calculate lens flares. They derive polynomials for refraction and reflection of rays with spherical and cylindrical interfaces through Taylor series expansion. As in matrix optics, the polynomial for the complete lens system can be built by inserting polynomials into each other. The resulting polynomials, however, are quite approximate when evaluated far from the point of expansion (the optical axis) and can contain many coefficients when constructed to high degree (degrees higher than 5 or 7 are hardly manageable in a Taylor expansion).

There are simple sparse polynomials used in practice to model lens distortions, e.g. the polynomials by Chen et al. [CJC*10] or the ones used in the open-source library *lensfun* [ZBK07]. They operate on 2D images, i.e. the position on the sensor, and thus do not transform the full light field.

Work in the area of lens design includes calculating the Taylor polynomial to obtain correction polynomials modelling the error of an aberrated ray compared to linear optics, for instance see Buchdahl [Buc68]. These polynomials can not only be used for approximating aberrations of a lens system [Hop76] but also for lens design [ZHB10]. For use in rendering these polynomials are impractical, as many terms have to be evaluated per ray.

Lenses in Physically-based Rendering In rendering, the simplest model is the pinhole camera; or the thin lens model (where the aperture and the outer pupil coincide) which simulates a single, infinitely thin lens without aberrations.

Kolb at al. [KMH95] introduced a more sophisticated model including distortions and identified the problem of guiding rays from the sensor through the aperture such that a large fraction of the samples actually reaches the outer pupil. Their approach has been improved by tabulated pupil sampling [SDHL11] which, however, requires significant precomputation and storage for every focus distance and aperture stop. Wu et al. [WZHX13] propose a technique to efficiently render spectral bokeh effects by explicit ray tracing through the lens system.

Hanika and Dachsbacher [HD14] refine the Taylor expansion by introducing a fitting step for higher precision when evaluating the polynomial far from the optical axis. The polynomial is then used in Monte Carlo path tracing, for which they introduced an aperture sampling technique. It ensures that most rays (well above 90% for real lenses) find the way through the aperture. This technique, however, cannot be used for the other direction (light tracing), i.e. when connecting path vertices in the scene to the outer pupil. In such cases, many connections get rejected due to absorption inside the lens system.

Hanika and Dachsbacher parametrise light fields in plane/plane space. That is, rays store their intersections with two planes separated by $dz = 1$ orthogonal to the optical axis. In this representation the free space propagation of rays can be described exactly through a linear polynomial, and enables precise refocusing of the lens by moving the sensor without the need to recompute the polynomial. We will make use of this parametrisation for the sensor light field.

Lenses in Interactive Rendering Effects such as depth of field or chromatic aberrations can be rendered by rasterising and accumulating the scene multiple times with different camera parameters [HA90] or projection matrices [HSS97]. A different approach is to apply lens effects as a post-processing step on a single image with depth information [MH14]. Occluded regions are not part of such an image, but a layered representation [LES10] of the scene can greatly enhance the resulting image. Lee et al. even model effects like spherical and chromatic aberration by tracing several rays per pixel through the scene layers.

As ray tracing an entire lens system is costly for real-time rendering, most previous work considers only single lenses. Especially when only first-order aberrations like depth of field are of interest, it is easy to calculate the circle of confusion, i.e. the size of a ray bundle originating in one point in the scene on the sensor from the depth of the scene point.

3. Sparse Polynomials for Lightfield Transformation

Polynomials have proven to be a good mathematical tool to accurately model lens aberrations in optics, and we also use this representation in our work. As previous work [Sei57, Buc68, HHH12, HD14] we treat a lens system as a black box with an incident light field at one side (the sensor), and an outgoing light field on the other side (outer pupil). For different focus distances, we move the sensor away from the lens system to focus distances closer to the outer pupil.

To construct the polynomials, we fit them to a set of ray traced ground truth samples which we obtain by brute force random walks through the lens system[†]. Our ray tracing supports both anamorphic (cylindrical) and aspherical elements.

In particular, we construct two 5×5 polynomial systems (see Fig. 2 for illustration and notation):

$$P_a(\mathbf{S}) : (x_s, y_s, dx_s, dy_s, \lambda) \mapsto (x_a, y_a, dx_a, dy_a, \tau_a) \quad (1)$$

$$P_o(\mathbf{S}) : (x_s, y_s, dx_s, dy_s, \lambda) \mapsto (x_o, y_o, dx_o, dy_o, \tau_o) \quad (2)$$

computing respectively an aperture light field $P_a(\mathbf{S})$ and the outgoing light field $P_o(\mathbf{S})$ from the sensor light field $\mathbf{S}$. A ray with wavelength λ goes through points $(x_{\{s,a,o\}}, y_{\{s,a,o\}})$ with directions $(dx_{\{s,a,o\}}, dy_{\{s,a,o\}})$ on the sensor, aperture, and outer pupil respectively. $\tau_{\{a,o\}}$ is the product of Fresnel transmittances along the path through the lens system. The polynomial $P_o(\mathbf{S})$ is sufficient to transform a ray through the lens system however, $P_a(\mathbf{S})$ is needed for aperture clipping / sampling. (see Sec. 4.1 / Fig. 5)

For high accuracy, we want to employ polynomials of high degree. Since this results in a combinatorial explosion of coefficients (requiring us to deal with tens of thousands of terms), we need to find a sparse solution (Sec. 3.2).

3.1. Coordinate Spaces

In the following we briefly discuss the different coordinate spaces which we need to be able to focus the image on the sensor, and to parametrize directions of up to $\pm180^\circ$.

Camera and World Space First, we transform from world space into camera space, where the z-axis is aligned with the optical axis and the origin is at the outer pupil, i.e. the sensor is set back by the lens length; we measure lengths in millimetres. The transform is essentially a rotation and, if world space lengths are measured in millimetres, too, incurs no Jacobian determinant.

Light Fields We deal with three light fields in a lens system: at the sensor, the aperture, and the outer pupil. As previous work [HD14] we use the plane/plane parametrisation for the ones at the sensor and the aperture. This enables us to quickly transform the sensor light field for refocusing, since a constant offset along the optical axis can be expressed accurately with a linear transform.

Fig. 3 illustrates the sphere/sphere parametrisation we use for the light field at the outer pupil. We chose this to support a field of view up to 360°. The position (x_o, y_o) is encoded by projecting the point on the outermost lens element onto the (x, y) plane perpendicular to the optical axis. We assume this last lens element (the outer pupil) is spherical in our implementation. The direction (dx_o, dy_o) is projected down from the normalised hemisphere defined by the intersection point of the ray with the sphere.

We choose a tangent frame such that there is no singularity in the hemisphere facing the scene (see Fig. 3, left). The poles of this parametrisation are at $\pm y$ instead.

[†] We restrict the ray tracing to transmission events; only for simulating lens flares (not of interest in this work) it would be necessary to calculate the reflection at each interface as well and to find suitable polynomials.

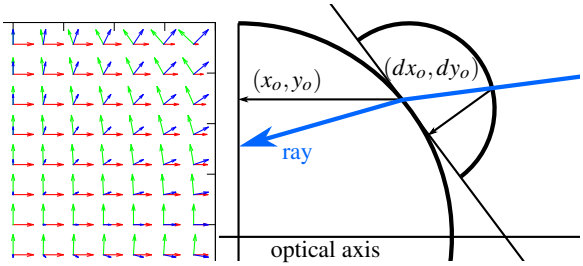

Figure 3: *Spherical coordinates on the outer pupil. A ray is parametrised as position projected perpendicular to the optical axis (x_o, y_o) and the normalised direction is similarly projected to a disk centred around the normal at the point of intersection, yielding (dx_o, dy_o). The tangent frames are oriented as shown on the left (blue: normal, red: tangent in x-z-plane, green: bitangent orthogonal to the other two vectors), to avoid a singularity on the optical axis. The three vectors are basis vectors in our local coordinate frame used for representing ray directions.*

More precisely, to transform a vector to this local coordinate system, we compute the matrix $\mathbf{T}(\mathbf{n})$ for a normal $\mathbf{n}$ on the sphere:

$$\mathbf{T}(\mathbf{n}) = \begin{bmatrix} n_z/l & -n_x n_y/l & n_x \\ 0 & l & n_y \\ -n_x/l & -n_y n_z/l & n_z \end{bmatrix}, \quad l := \sqrt{n_x^2 + n_z^2} \tag{3}$$

One tangent vector (left column) is orthogonal to the normal $\mathbf{n}$ (right column) in the x-z-plane and the other is calculated through a cross product (middle column).

Since the output of the lens polynomial is in this local coordinate system, we need to transform each ray to camera space and then world coordinates for path tracing for instance.

Vector Space of Polynomial Coefficients Our polynomials consist of terms of the form

$$c \cdot x_s^{d_0} y_s^{d_1} dx_s^{d_2} dy_s^{d_3} \lambda_s^{d_4} \text{ with degree } \sum_{i=0}^{4} d_i \leq d. \tag{4}$$

The coefficients $\mathbf{c}$ (together with the standard $+$ and $\cdot$ operations) form a vector space $\mathbf{c} \in \mathcal{P}$, which is isomorphic to $\mathbb{R}^{N(d)}$ for degree d. For instance degree $d = 11$ will result in $N(d) = \binom{n+d}{d} = 4368$ coefficients for $n = 5$ variables and we are interested in finding a sparse solution $\mathbf{c}' = (c'_0, c'_1, \ldots, c'_{s-1}) \in \mathbb{R}^s$ with s nonzero coefficients such that the sum of squared errors

$$\sum_{(\mathbf{S},\,\mathbf{O}) \in \text{Samples}} \left\| \mathbf{O} - P_o(\mathbf{S}) \right\|_2^2 \tag{5}$$

is minimal for the ray traced reference samples $(\mathbf{S}, \mathbf{O})$. A sparse solution makes storage and evaluation more efficient.

3.2. Normalised Orthogonal Matching Pursuit

Higher-degree polynomials Naturally, better polynomial fits can be achieved by including coefficients for terms with higher degree. As we have to find polynomial terms for five output variables in five input variables, going to high degree causes a combinatorial explosion and many more coefficients will have to be considered ($5 \cdot N(d)$ for odd degrees d from 5 to 15: 1260, 3960, 10010, 21840, 42840, 77520). This can quickly become a problem in the whole processing pipeline: when constructing the polynomial, during fitting, storage, or evaluation of the resulting model later on.

We observe that we can get more precise fits when including terms of higher degree, even when employing an overall lower number of coefficients. We use a maximum of 40 coefficients per output variable, i.e. a maximum of 200 total which gives a good trade-off between the number of terms to evaluate and small error (Sec. 6). To obtain such fits, we use a simple approach which avoids the Taylor series altogether.

Since finding the best coefficients for the vector space of all (mixed) powers of (x, y, dx, dy, λ) is a linear problem, we use linear least squares to directly compute the optimal fit. This enables fast fitting of high-degree polynomials (we tried up to degree 15) with tens of thousands of coefficients. Next we show how to pick the most important coefficients for an efficient evaluation of the resulting polynomials later on.

Finding a Sparse Coefficient Vector We start from the full configuration of a degree d polynomial in the variables of the light field at the sensor $(x_s, y_s, dx_s, dy_s, \lambda)$.

We construct one polynomial for every output variable. For this, we initialise a dense $M \times N(d)$ matrix, where M is the number of ray traced reference light field samples $(\mathbf{S}, \mathbf{O})$:

$$\hat{\Phi} = \begin{pmatrix} x_1 & y_1 & \cdots & \lambda_1^{d-1} dy_1 & \lambda_1^d \\ x_2 & y_2 & \cdots & \lambda_2^{d-1} dy_2 & \lambda_2^d \\ \cdots & & & & \\ x_M & y_M & \cdots & \lambda_M^{d-1} dy_M & \lambda_M^d \end{pmatrix}, \tag{6}$$

with the evaluation of each of the mixed terms (columns in $\hat{\Phi}$) for each of the samples (rows in $\hat{\Phi}$). We want to find a sparse coefficient vector $\mathbf{c}$ such that $\left\| \hat{\Phi} \cdot \mathbf{c} - \mathbf{b} \right\|_2$ is minimal for ray traced references on the outer pupil stored in $\mathbf{b}$. (i.e. $\mathbf{b}$ stores the outgoing light field). We use ten times more ray tracing samples than coefficients , to make the system overdetermined and to sufficiently sample the light fields (i.e. $M = 10 \cdot N(d)$). We will solve this system once for each of the ten equations: five for the light field at the aperture $\mathbf{A}$, and five at the outer pupil $\mathbf{O}$ (see Fig. 2).

Obtaining a sparse solution $\mathbf{c}'$ for the problem $\hat{\Phi} \cdot \mathbf{c}' = \mathbf{b}$ is not trivial. For instance just dropping the smallest coefficients does not work well. Thus, we employ a variant of orthogonal matching pursuit (OMP), which has been found to yield good results and to be simple to implement [TG07]. Although OMP is tailored towards input corrupted by noise – which is not the case for our ray traced input as transmission through the lens system is deterministic – the algorithm is still a good match because the polynomial approximation may result in some smoothing of details. The procedure (outlined in Alg. 1) takes the full matrix $\hat{\Phi}$ and the ray traced target vector $\mathbf{b}$, and returns a sparse matrix Φ and a corresponding coefficient vector $\mathbf{c}'$. The number of nonzero coefficients s can be set as a user input, thus giving us control over precision and the ability to bound the required computation when evaluating the polynomial.

The particular changes we made to the original algorithm to yield

better results is to calculate the exact sum of squared errors with fitted coefficients (Alg. 1, line 3) instead of approximating the error through a dot product. After adding s vectors the error is still not minimal so that we allow for replacing vectors afterwards (similar to [ARN06, JTD11]) if this reduces the error (Alg. 1, line 7).

Algorithm 1 Orthogonal Matching Pursuit with replacement

Require: $\hat{\Phi}, \mathbf{b}, s$ ▷ full matrix, target, number of nonzero coefficients

1: $\mathbf{c}' \leftarrow 0, \quad \Phi \leftarrow 0, \quad \mathbf{r} \leftarrow \mathbf{b}$
2: **for** $i = 1 \dots N(d)$ **do**
3: $\quad \mathbf{a}_m = \text{argmin}_{\mathbf{a}_m} \{\|(\Phi \cup \mathbf{a}_m) \cdot \mathbf{c}' - \mathbf{b}\|_2\}, \mathbf{a}_m = \hat{\Phi}_{\cdot,m}$
4: $\quad$ **if** $i < s$ **then**
5: $\quad\quad \Phi \leftarrow \Phi \cup \mathbf{a}_m$ // add column
6: $\quad$ **else** // look for best replacement
7: $\quad\quad \Phi_{\cdot,k} \leftarrow \mathbf{a}_m, \text{argmin}_k \{\|(\Phi \backslash \mathbf{a}_k \cup \mathbf{a}_m) \cdot \mathbf{c}' - \mathbf{b}\|_2\}$
8: $\quad$ **end if**
9: $\quad$ // linear least squares
10: $\quad \mathbf{c}' \leftarrow \text{argmin}_{\mathbf{c}'} \{\|\Phi \cdot \mathbf{c}' - \mathbf{b}\|_2\}$
11: $\quad \mathbf{r} \leftarrow \mathbf{b} - \Phi \cdot \mathbf{c}'$
12: $\quad$ **if** $\|\mathbf{r}\|_2 < \varepsilon$ **break**
13: **end for**
14: **return** $\Phi, \mathbf{c}'$

Algorithm 2 Aperture Sampling from the Light

Require: $\hat{\mathbf{o}}, \hat{\mathbf{A}}_{xy}$ ▷ 3D scene point, 2D on aperture

1: $\mathbf{S} = (x_s, y_s, dx_s, dy_s, \lambda) \leftarrow (0,0,0,0,\lambda)$
2: **for** $i = 1 \dots 100$ **do**
3: $\quad$ // aperture point from current sensor estimate
4: $\quad \mathbf{A}_{xy} \leftarrow P_a(\mathbf{S})$
5: $\quad$ // error vector on aperture
6: $\quad \Delta\mathbf{A}_{xy} \leftarrow \mathbf{A}_{xy} - \hat{\mathbf{A}}_{xy}$
7: $\quad$ // Jacobian aperture point $\mapsto$ sensor direction
8: $\quad J_a \leftarrow \mathrm{d}\mathbf{A}_{xy} / \mathrm{d}\mathbf{S}_\omega$
9: $\quad$ // 1) update direction based on aperture position
10: $\quad \mathbf{S}_\omega \leftarrow \mathbf{S}_\omega + J_a^{-1} \cdot \Delta\mathbf{A}_{xy}$
11: $\quad \mathbf{O} \leftarrow P_o(\mathbf{S})$ // full outer pupil ray
12: $\quad \mathbf{o} \leftarrow$ project $\mathbf{O}$ to plane at target point $\hat{\mathbf{o}}$
13: $\quad \Delta\mathbf{O}_\omega \leftarrow$ direction offset to point to target
14: $\quad$ // Jacobian sensor position $\mapsto$ outer pupil direction
15: $\quad J_o \leftarrow \mathrm{d}\mathbf{O}_\omega / \mathrm{d}\mathbf{S}_{xy}$
16: $\quad$ // 2) update position based on outer pupil direction
17: $\quad \mathbf{S}_{xy} \leftarrow \mathbf{S}_{xy} + J_o^{-1} \cdot \Delta\mathbf{O}_\omega$
18: $\quad$ **if** $\|\Delta\mathbf{O}_\omega\|_2 < \varepsilon$ and $\|\Delta\mathbf{A}_{xy}\|_2 < \varepsilon$ **break**
19: **end for**
20: **return** $\mathbf{S}$

4. Applications

4.1. Physically-based Rendering

In Monte Carlo light transport we handle paths starting at the sensor in the same way as Hanika and Dachsbacher [HD14]: by sampling a point on the aperture, solving for position and direction on the sensor $\mathbf{S}$, and then simply evaluating the polynomial. For the reverse direction, i.e. when connecting a transport path started at a light source to the camera, they compute an approximation to the inverse polynomial $\mathbf{S} = P_o^{-1}(.)$ to transport a ray from the outer pupil to the sensor. Next, they iteratively evaluate the forward polynomial, compute the error on the outer pupil, and propagate it back until they find the point on the outer pupil that was given as an input. This procedure ensures that path tracing and light tracing yield the same result, but it is not an efficient sampling technique because many of the constructed rays will be obstructed by the aperture. This is because the procedure requires sampling the connecting point on the outer pupil instead of the point on the aperture.

We deviate from this in two main aspects. Firstly, we sample the aperture instead of the outer pupil. This is a big improvement as we will see later, especially for fisheye lenses. Secondly, we only use the forward polynomials and their derivatives. While simplifying the implementation, this also ensures identical results for path and light tracing. Given a point on the aperture and in the scene, we solve for a position and direction $\mathbf{S}$ on the sensor and again evaluate the polynomial. This solve operation, however, is completely different to the path tracing case, since we have a 2D constraint far away from the outer pupil (the first path vertex in the scene) and on the other side of the polynomial's evaluation direction.

The aperture sampling routine for light tracing is summarised in Alg. 2. We start with a given wavelength λ, a point $\hat{\mathbf{o}}$ in the scene and a sampled point $\hat{\mathbf{A}}_{xy}$ on the aperture. We search for a position and direction $\mathbf{S}$ on the sensor such that the ray goes through the specified aperture point and such that, after leaving the outer pupil, the ray intersects the point $\hat{\mathbf{o}}$ in the scene. To achieve this, we employ a two-stage Newton method: the first stage (Alg. 2, line 10) updates the sensor direction based on the difference between the sampled aperture position $\hat{\mathbf{A}}_{xy}$ and the current estimate $\mathbf{A}_{xy}$. The second stage (Alg. 2, line 17) updates the sensor position based on the difference in outgoing direction. It is important to note that the difference in direction $\Delta\mathbf{O}_\omega$ is computed in camera space first and then transformed to the sphere/sphere parametrisation of the outer pupil light field. The algorithm terminates when both errors, in aperture and outgoing direction, are smaller than a defined threshold ε (we use 1e-4 in our implementation).

Transforming Sampling Densities To transform sampling densities between measure spaces, we need to compute Jacobian determinants [Kol33], [HD14, Sec. 4.1]. These will also need to respect the Jacobian of the transformation to sphere/sphere space. Note that for path tracing, most implementations are interested in an outgoing direction density in projected solid angle which is the same as the direction in sphere/sphere parametrisation; in this case no additional density transformation has to take place.

For light tracing, we need to adjust the Monte Carlo estimator as we now sample a point on the aperture. Starting from the sample contribution for path tracing:

$$c_{pt} = \frac{f}{p} = W(\mathbf{S}) / \left(p(\mathbf{A}_{xy}) \left\| \frac{\mathrm{d}\mathbf{S}_\omega}{\mathrm{d}\mathbf{A}_{xy}} \right\| \right) \tag{7}$$

where f is the measurement contribution, p the probability density function (PDF), $W(\mathbf{S})$ the sensor responsivity, and the Jacobian $\left\|\mathrm{d}\mathbf{S}_\omega / \mathrm{d}\mathbf{A}_{xy}\right\|$ signifies the change in direction at the sensor $\mathbf{S}_\omega$ with

respect to change of position on the aperture $\mathbf{A}_{xy}$. For light tracing this becomes:

$$c_{lt} = \frac{f}{p} = W(\mathbf{S}) / \left(p(\mathbf{A}_{xy}) \cdot \|J_v\| \cdot \left\| \frac{\mathrm{d}\mathbf{O}_{xy}}{\mathrm{d}\mathbf{A}_{xy}} \right\| \right) \tag{8}$$

where the Jacobians here transform vertex area measure at the aperture to the spatial part on the outer pupil $\mathbf{O}_{xy}$. Note that an additional Jacobian $\|J_v\| = \cos\theta = \sqrt{R - \mathbf{O}_x^2 - \mathbf{O}_y^2}/R$, where R is the curvature radius of the last lens element, is needed to transform the projected hemisphere position measure of $\mathbf{O}_{xy}$ to vertex area measure on the actual spherical surface of the last lens element. We compute this via the shared measurement space $\mathrm{d}\mathbf{S}_\omega$, as

$$\left\| \mathrm{d}\mathbf{O}_{xy} / \mathrm{d}\mathbf{A}_{xy} \right\| = \left\| \mathrm{d}\mathbf{O}_{xy} / \mathrm{d}\mathbf{S}_\omega \right\| \cdot \left\| \mathrm{d}\mathbf{S}_\omega / \mathrm{d}\mathbf{A}_{xy} \right\| \tag{9}$$

from 2×2 sub-matrices in the analytic Jacobians of the polynomials $P_o(\mathbf{S})$ and $P_a(\mathbf{S})$, respectively. Next event estimation for light tracing can then proceed, in the same way as with a thin lens model, by evaluating the geometry term from the point on the outer pupil to the scene point. The normal needs to be the geometric normal on the outer pupil lens element, and both cosines have to be included in the geometry term since the outgoing direction is given in projected solid angle.

4.2. Interactive Rendering

The evaluation of the final polynomial is fast enough for use in interactive applications. Compared to previous work (e.g. [MH14] or [LES10]) we do not have to intersect rays with lenses, neither are we limited to lower order aberrations as long as the necessary terms are included in the polynomial. Using the aperture polynomial and its derivatives, we can generate rays that go through sampled points on the aperture. We can trace these rays through the scene for example by ray marching an RGB-D texture. We calculate the min max mipmap of the depth values stored in the texture, allowing us to propagate the ray in larger steps for areas farther away from possible intersection points. Larger step sizes accelerate the algorithm through a decreased number of texture accesses and a reduced number of iterations for finding an intersection. The complete algorithm of our GLSL shader is given in Alg. 3.

Algorithm 3 Interactive Preview

Require: tex, $(s_x, s_y), n$ ▷ RGB-D texture, sensor position, number of samples

for $i = 1 \ldots n$ **do**
  $(x_a, y_a) \leftarrow$ sample_aperture(i/n)
  // find (dx_s, dy_s) s.t. ray goes through (x_a, y_a)
  $\mathbf{S} \leftarrow$ solve_aperture$((x_s, y_s), (x_a, y_a))$
  $\mathbf{O}' \leftarrow$ toWorldspace$(P_o(\mathbf{S}))$
  // $(r_x, r_y) \in [0,1]$, $r_z \in [z_{min}, z_{max}]$ from mipmap
  $\mathbf{r} \leftarrow$ toTextureSpace$(\mathbf{O}')$
  $l \leftarrow \mathrm{level}_{max}$ // start with lowest resolution
  while no intersection found **do**
    $(d_{min}, d_{max}) \leftarrow$ tex$((r_x, r_y), l)$
    $d \leftarrow 0$ // distance for ray propagation
    if $r_z < d_{min}$ **then**
      $d \leftarrow (d_{min} - r_z)/r_{dz}$, $l \leftarrow l - 1$
    else if $r_z < d_{max}$ **then**
      if $d_{max} - d_{min} < \varepsilon$ **return** intersection
      $l \leftarrow l + 1$
    end if
    $d \leftarrow \min(d,$ distance to next texel at level $l)$
    Propagate r by a distance of d
  end while
end for

5. Implementation Details

We implemented the orthogonal matching pursuit and least squares fitting using the library Eigen. Due to the high degree of our polynomials the matrices and vectors we use for calculating the sparse polynomial (Sec. 3.2) contain so many terms that single-precision floating point variables are not precise enough. We observed this through an increasing error of the fitted polynomial when increasing the degree and hence, the number of coefficients. We ran most experiments with degree 11. Note that going much higher than that will result in ill-conditioned polynomials and radial basis function interpolation may be better suited for even higher precision. This will result in much longer run times during evaluation, however.

6. Results

Aperture sampling Since aperture sampling from the light starts with the initial guess on the optical axis (i.e. $\mathbf{S} = (0,0,0,0,\lambda)$), we need slightly more iterations to reach convergence than in the path tracing case which has a more reasonable initial guess. Actual numbers depend a lot on the error threshold ε, viewing direction and lens, but in most cases Alg. 2 converges in less than 20 iterations. The path tracing case needs only about 4 iterations.

It is essential to fit the polynomials P_o and P_a to the same set of light field sample rays, to ensure consistency for the two-stage Newton method. Also, as the allowed fitting error is increased or the number of coefficients pushed too low (below $\approx$10 per output variable), the method will take more iterations and start to converge to a slightly biased solution.

Fig. 4 shows an equal sample comparison of path tracing and the two light tracing methods: via pupil sampling [HD14] and via aperture sampling (Alg. 2). The second method is far superior, even for wide open apertures. Fig. 5 illustrates why: it visualises the set of visible samples as well as the set we obtain via aperture sampling on the full pupil. Especially for fisheye lenses, the image of the aperture is very small on the outer pupil and will result in a lot of absorbed samples if not sampled carefully.

Accuracy of sparse polynomials We compare the precision and number of coefficients of our sparse polynomial to the Taylor polynomial and to the complete polynomial containing all terms up to a defined degree (Tbl. 1). All results were calculated with the same 15,000 reference rays. We used the same rays for error evaluation as for fitting. For the Taylor polynomial the average sum of squared errors per ray decreases with increasing degree, as higher-order terms help to model the lenses aberrations more precisely. The error of the complete polynomial is in most cases slightly smaller but the polynomial contains many more coefficients. Fig. 6 shows the error on

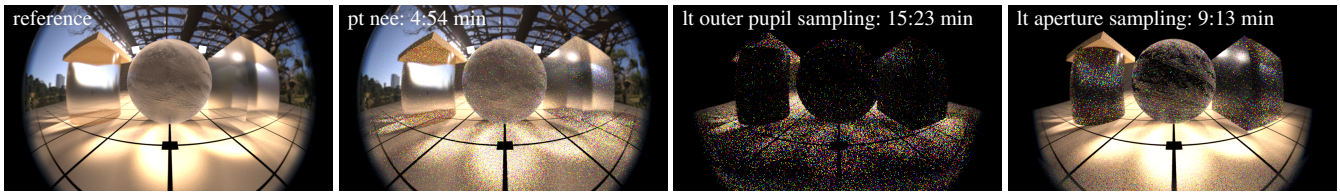

Figure 4: *Equal sample comparison (512 spp) between rendering algorithms for the aspherical fisheye lens (pt nee: path tracing with next event estimation, lt: light tracing). The light tracer with outer pupil sampling [HD14] performs very badly, as expected when looking at the fraction of visible samples on the outer pupil (see Fig. 5). Background image from hdrlabs.com.*

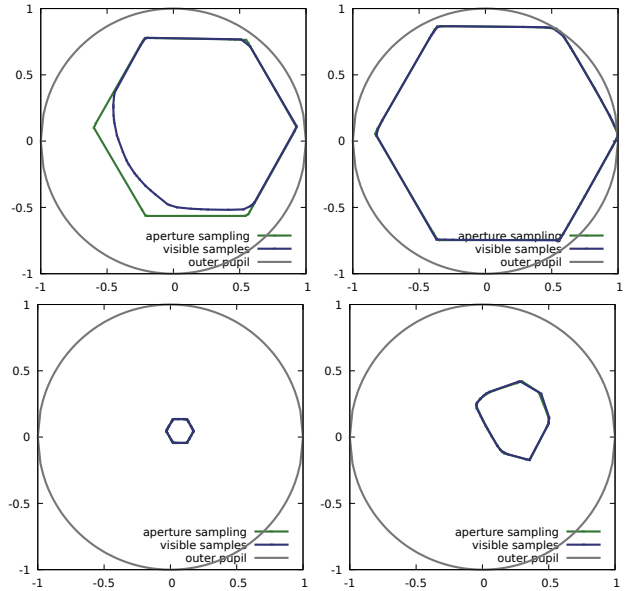

Figure 5: *Sampling efficiency illustrated as convex hulls of samples on the outer pupil (normalised to outer pupil housing radius). The lenses are (left to right, top to bottom): very fast double Gauss lens (large aperture), classic double Gauss, the aspherical fisheye lens, an anamorphic lens. The aperture has six-blades and was as wide open as possible. The benefits of aperture sampling increase with higher f-stop. Previous work [HD14] samples the outer pupil (grey circle) for light tracing.*

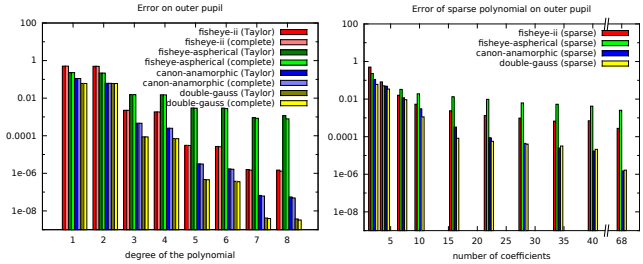

Figure 6: *Errors of the Taylor polynomial and the complete polynomial for four different lenses for degrees 1–8. For sparse polynomials with small numbers of terms the error first gets reduced greatly but changes only slowly afterwards when adding more terms.*

	fisheye-ii	fisheye-aspherical	canon-anamorphic	double-gauss
Taylor 1 (2)	$5.07 \cdot 10^{-1}$	$2.29 \cdot 10^{-1}$	$1.11 \cdot 10^{-1}$	$6.06 \cdot 10^{-2}$
Complete 1 (6)	$5.07 \cdot 10^{-1}$	$2.28 \cdot 10^{-1}$	$1.11 \cdot 10^{-1}$	$6.06 \cdot 10^{-2}$
Sparse (2)	$5.07 \cdot 10^{-1}$	$2.28 \cdot 10^{-1}$	$1.11 \cdot 10^{-1}$	$6.06 \cdot 10^{-2}$
Taylor 2 (4)	$4.97 \cdot 10^{-1}$	$2.17 \cdot 10^{-1}$	$6.18 \cdot 10^{-2}$	$6.03 \cdot 10^{-2}$
Complete 2 (21)	$4.97 \cdot 10^{-1}$	$2.17 \cdot 10^{-1}$	$6.17 \cdot 10^{-2}$	$6.02 \cdot 10^{-2}$
Sparse (4)	$\mathbf{8.21 \cdot 10^{-2}}$	$\mathbf{5.07 \cdot 10^{-2}}$	$\mathbf{4.85 \cdot 10^{-2}}$	$\mathbf{3.43 \cdot 10^{-2}}$
Taylor 3 (16)	$2.26 \cdot 10^{-3}$	$1.54 \cdot 10^{-2}$	$4.63 \cdot 10^{-4}$	$8.76 \cdot 10^{-5}$
Complete 3 (56)	$\mathbf{2.25 \cdot 10^{-3}}$	$1.54 \cdot 10^{-2}$	$4.62 \cdot 10^{-4}$	$8.69 \cdot 10^{-5}$
Sparse (16)	$2.40 \cdot 10^{-3}$	$\mathbf{1.34 \cdot 10^{-2}}$	$\mathbf{3.25 \cdot 10^{-4}}$	$\mathbf{8.22 \cdot 10^{-5}}$
Taylor 4 (28)	$1.86 \cdot 10^{-3}$	$1.52 \cdot 10^{-2}$	$2.52 \cdot 10^{-4}$	$7.07 \cdot 10^{-5}$
Complete 4 (126)	$1.85 \cdot 10^{-3}$	$1.49 \cdot 10^{-2}$	$2.50 \cdot 10^{-4}$	$6.98 \cdot 10^{-5}$
Sparse (28)	$\mathbf{9.72 \cdot 10^{-4}}$	$\mathbf{6.26 \cdot 10^{-3}}$	$\mathbf{4.40 \cdot 10^{-5}}$	$\mathbf{4.02 \cdot 10^{-5}}$

Table 1: *The average of the sum of squared errors per ray of our sparse polynomial is in almost all cases smaller than the error of both the complete polynomial and the Taylor polynomial with comparable numbers of coefficients per equation.*

the outer pupil for different lenses for both the Taylor polynomial and the complete polynomial.

With our sparse polynomials we can define the number of coefficients per equation exactly for fast evaluation later on. In most cases our fitting algorithm finds the best terms from the degree 11 polynomial so that the average error is smaller than the one of the Taylor polynomial with the same number of coefficients. (Tbl. 1) Adding even more terms to the polynomial has only a minor influence on the error. (see Fig. 6)

Fig. 7 shows the average transmittance of the outgoing light field **O** plotted vs. the distance to the optical axis on the outer pupil. For most long or moderately wide angle lenses this plot is almost constant, as for the 100mm lens on the right. The left plot shows a fisheye with slightly above 180 degree field of view, with clearly visible vignetting due to Fresnel transmittance. This is frequently reduced by applying lens coatings and is thus important to model.

Timings Since it is impossible to compare a fisheye render to a thin lens base line, we used a double Gauss lens instead. While a single evaluation of a camera sample or connection is significantly more expensive than the thin lens counter part, we found that in a setting with moderate complexity for visual effects work the overhead becomes negligible (see Fig. 8).

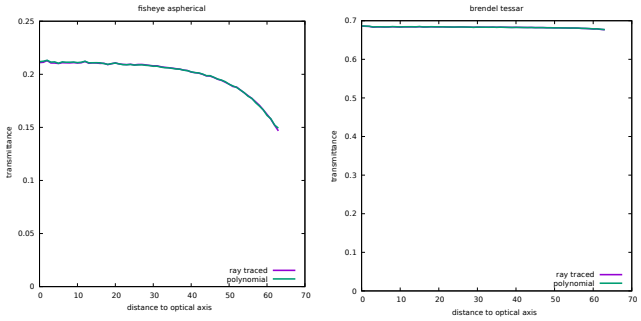

Figure 7: *Fresnel transmittances for two lenses: a fisheye and a 100mm prime, plotted versus position on the outer pupil and averaged over all other dimensions. While this can be reasonably assumed to be constant for long lenses, there is significant vignetting for fisheye lenses.*

Figure 8: *The test scene [Nic15] we used to evaluate the speed impact of our camera model over the thin lens model in a production complexity setting (33M primitives). Bidirectional path tracing took 185s for one sample/pixel in a 2048×1152 render with the shown double Gauss lens (top), and 184s using the thin lens with a closely matching field of view (bottom). The speed difference is mainly due to the slightly different set of paths to be traced.*

	fisheye-ii	fisheye-aspherical	canon-anamorphic	double-gauss
original OMP	$1.69 \cdot 10^{-3}$ (18 s)	$1.15 \cdot 10^{-2}$ (24 s)	$1.59 \cdot 10^{-3}$ (21 s)	$6.51 \cdot 10^{-5}$ (22 s)
with fit	$1.36 \cdot 10^{-3}$ (1:02 min)	$9.25 \cdot 10^{-3}$ (1:02 min)	$6.75 \cdot 10^{-5}$ (0:58 min)	$5.85 \cdot 10^{-5}$ (1:03 min)
with replacement	$9.58 \cdot 10^{-4}$ (42:11 min)	$5.40 \cdot 10^{-3}$ (44:18 min)	$3.85 \cdot 10^{-5}$ (37:10 min)	$3.61 \cdot 10^{-5}$ (40:10 min)
replacement and fit	$8.55 \cdot 10^{-4}$ (6:58 min)	$6.27 \cdot 10^{-3}$ (6:01 min)	$4.04 \cdot 10^{-5}$ (3:37 min)	$3.86 \cdot 10^{-5}$ (5:56 min)

Table 2: *Comparison of the errors and execution times of the different variants of the orthogonal matching pursuit algorithm. (28 terms per equation from a degree 8 polynomial fitted single threaded to 2000 samples on an Intel Core i7-4790k processor)*

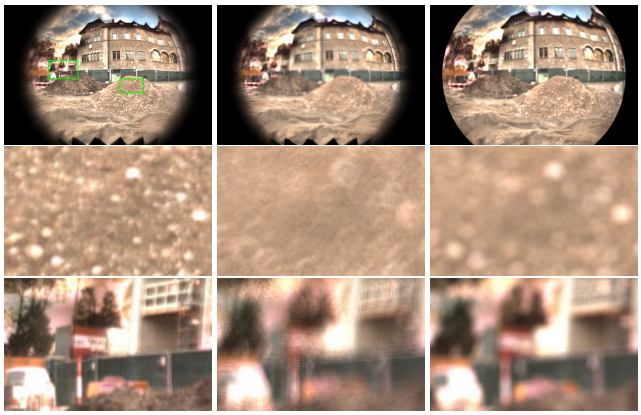

Figure 9: *Our preview shows lens effects applied on RGB-D cubemaps (source: [ZKP13]) at interactive rates (137 ms per 1080x720 image with 144 samples per pixel on a AMD Radeon R9 390 graphics card) when the scene is in focus (left), out of focus (middle) and out of focus with a smaller aperture (right).*

Orthogonal Matching Pursuit While the original orthogonal matching pursuit algorithm is fast, the error can be reduced further by allowing the replacement of terms. In our implementation this includes an exhaustive search for the least important terms in the current sparse solution. By spending more time on choosing important terms (e.g. through fitting the coefficients and calculating the squared error we want to minimize) we can speed-up the algorithm (see Tbl. 2). Our goal is to calculate a polynomial with few coefficients and small error and it is hence acceptable to spend more time once for finding a solution, as it saves time when evaluating the polynomial. The errors for sparse polynomials in Tbl. 1 were obtained allowing replacements and calculating the least squares fit for every term to be added (fourth row in Tbl. 2).

Interactive preview Evaluating our sparse polynomials can be used as a post-process working on images with depth information. While there exist more elaborate algorithms on interactive as well as real-time rendering lens effects, our implementation is usable as an interactive preview. (Fig. 9)

7. Conclusion and Future Work

We presented a polynomial model to describe the light field transformation happening in photographic lens systems, which is suitable to describe extreme wide angle systems such as fisheye lenses. We showed how to create a sparse model for fast evaluation and how to use the derivatives of the polynomials to perform aperture sampling for deterministic connections to the sensor. This is useful for bidirectional path tracing and variants of Metropolis light transport which move the first path vertex after the camera. We also demonstrated how the sparse polynomial model is useful in interactive applications as it can be evaluated very quickly on the GPU.

When trying to match footage of a certain lens, it can be hard to obtain the exact lens description table. If it is available, it can be even harder to obtain data for potentially applied coatings. Our current implementation describes the outgoing light field in sphere/sphere coordinates, regardless of the last lens element. For cylindrical or aspherical outer pupils, this could be changed to more closely represent the actual lens geometry, to simplify the geometry term handling for path tracing. Furthermore, we do not treat zoom lenses so far. If required, this can be handled as an additional input variable of the polynomials.

The performance of the OMP fitter we employ can be time consuming (see Tbl. 2). On the other hand, standard OMP does not yield the best results out of the box. We see room for future improvement here.

As our implementation of the interactive preview is straightforward it would be interesting to see how large the overhead for evaluating the lens polynomial is in an existing framework for example the one by Lee et al. [LES10].

References

[ABW14] AMENT M., BERGMANN C., WEISKOPF D.: Refractive radiative transfer equation. *ACM Trans. on Graphics 33*, 2 (Apr. 2014), 17:1–17:22. 2

[ARN06] ANDRLE M., REBOLLO-NEIRA L.: A swapping-based refinement of orthogonal matching pursuit strategies. *Sparse Approximations in Signal and Image Processing 86*, 3 (2006), 480 – 495. 5

[BG12] BAHRAMI M., GONCHAROV A. V.: Geometry-invariant gradient refractive index lens: analytical ray tracing. *Journal of Biomedical Optics 17*, 5 (2012), 055001–1–055001–9. 2

[Buc68] BUCHDAHL H. A.: *Optical aberration coefficients*. Dover Publications, 1968. 2, 3

[CJC*10] CHEN S., JIN H., CHIEN J., CHAN E., GOLDMAN D.: *Adobe Camera Model*. Tech. rep., Adobe Systems Inc, January 2010. 2

[Gau41] GAUSS C.: *Dioptrische Untersuchungen*. Dieterich, 1841. 2

[HA90] HAEBERLI P., AKELEY K.: The accumulation buffer: hardware support for high-quality rendering. *ACM SIGGRAPH Computer Graphics 24*, 4 (1990), 309–318. 3

[HD14] HANIKA J., DACHSBACHER C.: Efficient Monte Carlo rendering with realistic lenses. *Computer Graphics Forum (Proceedings of Eurographics) 33*, 2 (April 2014), 323–332. 2, 3, 5, 6, 7

[Hec01] HECHT E.: *Optics*, 4 ed. Addison Wesley, August 2001. 2

[HESL11] HULLIN M. B., EISEMANN E., SEIDEL H.-P., LEE S.: Physically-based real-time lens flare rendering. *ACM Trans. Graph. (Proc. SIGGRAPH 2011) 30*, 4 (2011), 108:1–108:9. 2

[HHH12] HULLIN M., HANIKA J., HEIDRICH W.: Polynomial Optics: A construction kit for efficient ray-tracing of lens systems. *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering) 31*, 4 (July 2012). 2, 3

[Hop76] HOPKINS G. W.: Proximate ray tracing and optical aberration coefficients. *JOSA 66*, 5 (May 1976), 405–410. 2

[HSS97] HEIDRICH W., SLUSALLEK P., SEIDEL H.-P.: An image-based model for realistic lens systems in interactive computer graphics. In *Graphics Interface* (May 1997), Davis W., Mantei M., Klassen V., (Eds.), pp. 68–75. 3

[JTD11] JAIN P., TEWARI A., DHILLON I. S.: Orthogonal matching pursuit with replacement. In *Advances in Neural Information Processing Systems 24*, Shawe-Taylor J., Zemel R. S., Bartlett P. L., Pereira F., Weinberger K. Q., (Eds.). Curran Associates, Inc., 2011, pp. 1215–1223. 5

[Kin92] KINGSLAKE R.: *Optics in Photography*. SPIE Publications, 1992. 2

[KMH95] KOLB C., MITCHELL D., HANRAHAN P.: A realistic camera model for computer graphics. In *ACM Trans. on Graphics (Proc. SIGGRAPH)* (1995), pp. 317–324. 3

[Kol33] KOLMOGOROV A.: *Grundbegriffe der Wahrscheinlichkeitsrechnung*. Julius Springer, Berlin, 1933. 5

[LE13] LEE S., EISEMANN E.: Practical real-time lens-flare rendering. *Computer Graphics Forum 32*, 4 (2013), 1–6. 2

[LES10] LEE S., EISEMANN E., SEIDEL H.-P.: Real-Time Lens Blur Effects and Focus Control. *ACM Trans. on Graphics (Proc. SIGGRAPH) 29*, 4 (2010), 65:1–7. 3, 6, 9

[Met48] METTE W. W.: *The Zeiss Index of Photographic Lenses*. Tech. rep., ZEISS (CARL) JENA (GERMANY), Nov. 1948. 2

[MH14] MOERSCH J., HAMILTON H. J.: Variable-sized, circular bokeh depth of field effects. In *Proceedings of Graphics Interface 2014* (2014), GI '14, pp. 103–107. 3, 6

[Nic15] NICHOLS C.: Wikihuman : The open source digital human project. FMX talk, 2015. 8

[ODS10] OPTICAL DATA SOLUTIONS I.: The LensVIEW database. http://ods-inc.com/, 2010. 2

[Sch16] SCHOTT AG: Optical glass catalogue, February 2016. 2

[SDHL11] STEINERT B., DAMMERTZ H., HANIKA J., LENSCH H. P. A.: General spectral camera lens simulation. In *Computer Graphics Forum* (2011), vol. 30, pp. 1643–1654. 3

[Sei57] SEIDEL L.: *Über die Theorie der Fehler, mit welchen die durch optische Instrumente gesehenen Bilder behaftet sind, und über die mathematischen Bedingungen ihrer Aufhebung*. Abhandlungen der Naturwissenschaftlich-Technischen Commission bei der Königl. Bayerischen Akademie der Wissenschaften in München. Cotta, 1857. 2, 3

[Smi00] SMITH W. J.: *Modern Optical Engineering*. McGraw-Hill, 2000. 2

[Smi05] SMITH W. J.: *Modern Lens Design*. McGraw-Hill, 2005. 2

[TG07] TROPP J. A., GILBERT A. C.: Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory 53*, 12 (Dec. 2007), 4655–4666. 4

[WZHX13] WU J., ZHENG C., HU X., XU F.: Rendering realistic spectral bokeh due to lens stops and aberrations. *The Visual Computer 29*, 1 (2013), 41–52. 3

[ZBK07] ZABOLOTNY A., BRONGER T., KRAFT S.: The Lensfun Project, 2007. URL: http://lensfun.sourceforge.net/. 2

[ZHB10] ZHENG N., HAGEN N., BRADY D. J.: Analytic-domain lens design with proximate ray tracing. *JOSA 27*, 8 (Aug 2010), 1791–1802. 2

[ZKP13] ZEISL B., KOSER K., POLLEFEYS M.: Automatic registration of rgb-d scans via salient directions. In *Computer Vision (ICCV), 2013 IEEE International Conference on* (2013), IEEE, pp. 2808–2815. 8