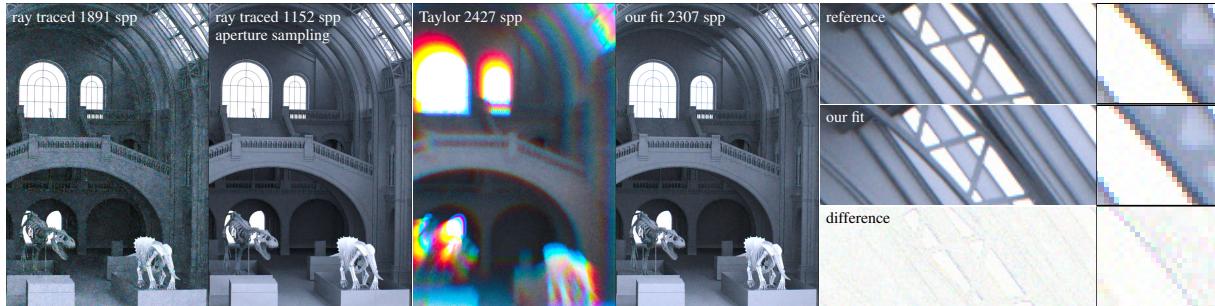


# Efficient Monte Carlo Rendering with Realistic Lenses

Johannes Hanika and Carsten Dachsbacher,

Karlsruhe Institute of Technology, Germany



**Figure 1:** Equal time comparison (40min,  $640 \times 960$  resolution): rendering with a virtual lens (Canon 70-200mm f/2.8L at f/2.8) using spectral path tracing with next event estimation and Metropolis light transport (Kelemen mutations [[KSKAC02](#)]). Our method enables efficient importance sampling and the degree 4 fit faithfully reproduces the subtle chromatic aberrations (only a slight overall shift is introduced) while being faster to evaluate than ray tracing, naively or using aperture sampling, through the lens system.

### **Abstract**

In this paper we present a novel approach to simulate image formation for a wide range of real world lenses in the Monte Carlo ray tracing framework. Our approach sidesteps the overhead of tracing rays through a system of lenses and requires no tabulation. To this end we first improve the precision of polynomial optics to closely match ground-truth ray tracing. Second, we show how the Jacobian of the optical system enables efficient importance sampling, which is crucial for difficult paths such as sampling the aperture which is hidden behind lenses on both sides. Our results show that this yields converged images significantly faster than previous methods and accurately renders complex lens systems with negligible overhead compared to simple models, e.g. the thin lens model. We demonstrate the practicality of our method by incorporating it into a bidirectional path tracing framework and show how it can provide information needed for sophisticated light transport algorithms.

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

## 1. Introduction

In the endeavor of ever increasing the realism of physically-based renderings, much progress has been made in the last years in improving light transport algorithms and material models. However, for mimicing real photography we also have to pay attention to accurately simulate camera lenses. Only then can we reproduce bokeh and depth of field effects accurately and match real footage for computer-generated imagery, e.g. in the movie industry.

In contrast to popular lens models in computer graphics, real-world lenses often consist of several to dozens of lens elements and an aperture. Although straightforward, path or light tracing through these lenses is impractical as it requires an exorbitant number of paths for converged images (see Fig. 1). The problem (as opposed to the pinhole or thin lens model) is that it is not possible to specifically create camera rays into the scene, or connect light paths to the sensor; and most of the paths will not even propagate through the lenses because they hit the diaphragm or the lens housing.

A camera model which is useful for *efficient* Monte Carlo rendering must support the following with little overhead and deterministic results:

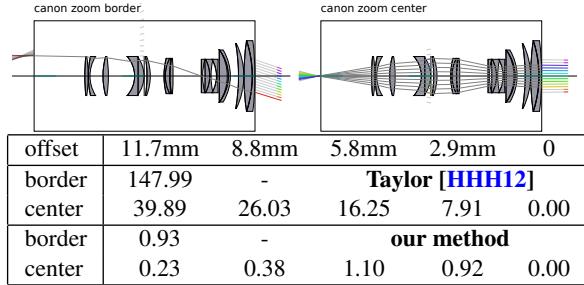
- sampling a new path from the sensor: sample sensor point, sample aperture, get direction,
- connecting to the lens: sample pupil, get sensor point,
- computing the probability densities for both aforementioned cases to support bidirectional (Markov chain) Monte Carlo methods.

In this paper, we present a camera model that can reproduce realistic lenses, such as a Canon 70–200mm f/2.8L or a Lensbaby, with high accuracy yet meeting the above criteria. As in polynomial optics [HHH12] we represent the lens with a polynomial approximation, however, we show how to reduce the approximation error by 1–2 orders of magnitude (measured in pixels, see Fig. 2) compared to previous work. Our further key contributions are direct sampling techniques and the computation of probability densities based on the Jacobians of the polynomial. We demonstrate our method in a bidirectional Monte Carlo framework where it is crucial that path and light tracing yield exactly equivalent results, and compare to direct ray tracing as well as previous work in terms of accuracy and computation time. Our method requires only negligibly more computation than a thin lens model and enables importance sampling without need for tabulation and handles small apertures efficiently. It supports all aberrations, vignetting, enables focusing via sensor shifts and faithfully reproduces interesting point spread functions.

## 2. Previous Work

**Lens Systems in Raytracing** Simulation of more sophisticated camera lens distortion models than the thin lens model was introduced to computer graphics by Kolb et al. [KMH95]. Other aberrations, wavelength dependent effects and some diffraction effects were added later by Steinert et al. [SDHL11]. Their methods reproduce a variety of realistic effects, but were not practical because of long computation times and the need for tabulation for every lens configuration (aperture, focus). Furthermore, the integration into existing Monte Carlo rendering frameworks was hard due to the specific set of requirements of such frameworks, e.g. the evaluation of probability densities.

Brute force ray tracing through lens systems is not only computationally expensive, but also complicates importance sampling: casting a ray from the sensor through the aperture is an “SDS” problem (the aperture with specular vertices before and after). Specialized sampling techniques, such as Metropolis light transport with manifold walks [Jak13] inside the lens, are possible but intricate and most importantly: complex camera paths complicate every transport path in the image. Our sampling scheme avoids the explicit representation of the specular chain while being similar to specular manifold walks in effect.



**Figure 2:** A comparison of the pixel error for a 36mm/2k sensor back between previous work (Taylor expansion around the optical axis) and our method for  $\lambda = 500\text{nm}$ . The individual numbers in each row correspond to the differently colored rays.

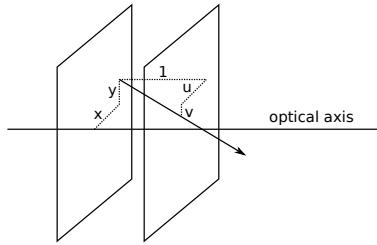
**Lens Approximations** The first-order paraxial approximation [Gau41] by ABCD matrices has been used for image formation in *pencil tracing* [STN87] and more recently for real-time lens flares [LE13]. Our approximation is third-order (fourth-order if wavelength is included) to support the aberrations described by Seidel optics [Sei57]. We perform first-order analysis to transform probability densities using the Jacobians of our functions.

Using polynomials to approximate lens distortions is common practice in photography applications [KZB<sup>\*</sup>07, CJC<sup>\*</sup>10]. However, these account for image distortions only and do not cover point spread functions (such as in Fig. 11).

**Polynomial Optics** Systems of polynomials from Taylor expansions have been used to approximate transformations through an optical system on the full four dimensional incident light field [HHH12]. These functions capture all interesting aberrations and enable fast rendering of lens flares. However, naively using this approach as a camera model in a Monte Carlo renderer reveals a number of weaknesses:

- the precision of the Taylor expansion is sufficient for lens flares, but its inaccuracy leads to unacceptable results with regard to image distortion, sharpness, and bokeh (e.g. more than 140 pixels at the image boundaries, see Fig. 2),
- practical rendering with lens systems requires the evaluation from both the sensor and the light sources,
- to perform multiple importance sampling probability density functions for both sampling techniques must be known,
- in the original work there is no easy way to focus the lens,
- the plane/sphere light-field parametrization in [HHH12] is suboptimal for image formation (see Sec. 3.1)

Some of these requirements involve finding roots of the polynomial system. For the general case, algorithms and software packages, such as [GKK<sup>\*</sup>04, LLT08], exist. However, our systems are comparatively small and we show how a simple Newton iteration converges well in our case.



**Figure 3:** We choose a paraxial plane/plane parametrization of the light field around the optical axis (see Sec. 3.1).

### 3. Background on Lens Systems and Light Fields

In this section we introduce the background for our work. We start by formalizing the light transport through a lens system and recapitulate the well-known thin lens model which we then generalize to complex lenses.

#### 3.1. Light Fields and Polynomial Optics

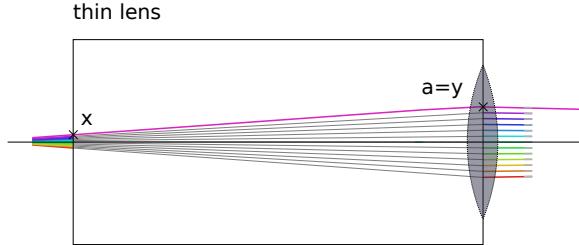
In order to describe the image formation – the gathering of light and focusing on the sensor plane – we use a plane/plane parameterized light field (see Fig. 3). The light field at a position  $\mathbf{x} = (x, y, u, v, \lambda)$  is parameterized based on the spatial position  $x, y$  in a plane perpendicular to the optical axis of the lens system,  $u, v$  define a direction vector  $\omega = (u, v, 1)$ , and  $\lambda$  denotes the wavelength of light.

As pointed out by Igehy [Ige99, Sec. 2], the linearized paraxial approximation used in pencil tracing [STN87] does not accurately represent the propagation of rays through empty space due to the plane/sphere parameterization. Using the plane/plane parameterization instead makes paraxial transport between parallel planes precise at first degree already. As a consequence, empty-space transport Jacobians are non-approximate, which enables focusing with our algorithm. Lastly, a plane/sphere parameterization would not allow rectilinear lenses with a minimal number of coefficients.

Lenses can be seen as a transformation of light fields as radiance propagates through the photographic objectives. Fig. 4 shows a simple lens system. The transport between two locations  $\mathbf{y}$  and  $\mathbf{x}$  can be approximated by a function  $P'(\mathbf{y}) = \mathbf{x}$  which transforms the lightfield  $(x, y, u, v, \lambda)$  at  $\mathbf{y}$  to  $(x', y', u', v', \lambda)$  at  $\mathbf{x}$  ( $\lambda$  is constant as we assume that lenses do not fluoresce). We use a  $5 \times 4$  polynomial system to represent this function [HHH12].

#### 3.2. The Thin Lens Model

In this section, we briefly recapitulate path and light tracing and importance sampling with the simple, well-known thin lens model. We use this model as an example to introduce the general notation required for more complex lens systems.



**Figure 4:** Schematic of the thin lens model. The aperture and the flat lens are imagined to fall into the same plane. The colors denote different rays (not wavelengths) with spatial offsets on the outer pupil. The point  $\mathbf{x}$  is on the sensor,  $\mathbf{a}$  is on the aperture and  $\mathbf{y}$  on the outer pupil.

To compute an image with Monte Carlo raytracing, we integrate the measurement contribution  $f$  over vertices  $x$  on the sensor and  $a$  on the aperture, where  $\omega_o$  is the direction of the ray at the outgoing pupil at  $y$  (Fig. 4). Note that  $a$  and  $y$  coincide in this case. The measurement contribution [Vea97] is defined as:

$$f = L(y, -\omega_o)G(y \leftrightarrow x)W(x), \quad (1)$$

where  $L$  denotes radiance,  $G$  is the geometric term, and  $W$  is the sensor responsivity.

**Path tracing** samples a point  $x$  proportional to a probability density function (PDF)  $p(x)$  on the sensor, and a point  $a \sim p(a)$  on the aperture. This defines the ray direction  $\omega_s = a - x$  off the sensor with density  $p(\omega_s) = p(a)/G(a \leftrightarrow x)$ . The probability density of the outgoing point and direction  $(y, \omega_o)$  is then the density of  $(x, \omega_s)$  transported through the optical system. In this simple case  $p(y, \omega_o) = p(x)p(\omega_s)G(a \leftrightarrow x)$ . This means that the geometric term cancels out of the combined PDF, but stays in the measurement contribution. The estimator then becomes

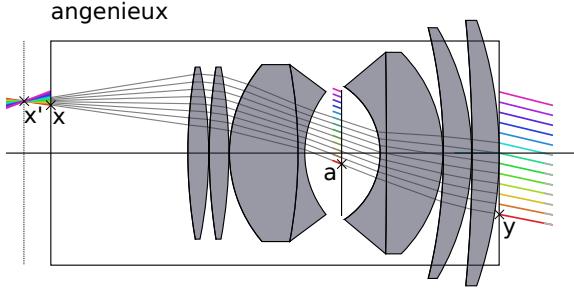
$$\frac{1}{N} \sum L(y, -\omega_o) \frac{G(a \leftrightarrow x)W(x)}{p(x)p(a)}. \quad (2)$$

**Light tracing** creates paths starting from light sources and connects path vertices to a point on the lens  $y \sim p(y)$  with incident direction  $\omega_o$ . Again, the inward facing direction  $\omega_s$  and  $x$  follow deterministically, resulting in an on-surface density  $p(x) = p(\omega_s)G(y \leftrightarrow x)$ . The estimator is then simply

$$\frac{1}{N} \sum L(y, -\omega_o)W(x)/p(y), \quad (3)$$

as the geometric term  $G(y \leftrightarrow x)$  appears in both the sampling density  $p(x)$  and the measurement contribution and thus cancels out, which makes the weights simple to compute.

Note that often the inverse of the geometric term  $G(y \leftrightarrow x)$  is implicitly baked into the sensor response  $W(x)$  to avoid the vignetting effect that comes with the cosine terms.



**Figure 5:** Schematic of an Angenieux lens. The transformation of the lightfield from outer pupil  $\mathbf{y}$  to  $\mathbf{x}$  is considered to be a black box function evaluation  $P'(\mathbf{y}) = \mathbf{x}$ . The light field at  $\mathbf{x}$  is then transported to  $\mathbf{x}'$  on the sensor in focus.

#### 4. Beyond the Thin Lens Model

To support arbitrary lens systems, we need to extend the probability densities in Sec. 3.2 to account for lens elements between the aperture and both the outer pupil and the sensor (compare Fig. 4 and 5). We also need to be able to focus the lens by other means than just implicitly altering the focal length of the imagined lens element in the thin lens model.

##### 4.1. Geometric Terms from Transport Jacobians

We start from the thin lens model (Fig. 4): to evaluate the geometric term  $G(a \leftrightarrow x)$  required in the path tracing case, we express the transport between  $a$  and  $x$  in paraxial plane/plane parametrization and assume that we can represent the mapping of the light field at  $\mathbf{x} = (x, y, u, v, \lambda)$  to  $\mathbf{a} = (x', y', u', v', \lambda)$  using a polynomial system  $P_a(\mathbf{x}) = \mathbf{a}$  (For an intuition what these polynomials look like we provide example equations in the additional material). Ignoring wavelength for now, the Jacobian of the transport operator  $P(\mathbf{x})$  in empty space is

$$J_{\mathbf{x} \rightarrow \mathbf{a}} = \begin{pmatrix} I & T \\ 0 & I \end{pmatrix} \in \mathbb{R}^{4 \times 4} \text{ with } T = \begin{pmatrix} d & 0 \\ 0 & d \end{pmatrix} \quad (4)$$

where  $I$  is the  $2 \times 2$  identity matrix and  $d$  is the distance along the optical axis between  $\mathbf{a}$  and  $\mathbf{x}$ . This means  $(u, v)$  stay constant and  $(x, y)$  are moved by distance times direction. Note that in this simple case the first-order version of our function is already precise.

This analysis is required to compute the first-order differentials as they appear when expressing the geometric term as ratio of differentials  $G = d\omega^\perp / dA$  (e.g. when performing a change of variables in integration). This is closely related to the formulation using  $ABCD$  matrices [STN87, Sec 4.2]. For path tracing, the geometric term in plane/plane parameterization  $G^{\parallel}(a \leftrightarrow x)$  can then be written as:

$$G^{\parallel}(a \leftrightarrow x) = 1/|T|, \quad (5)$$

because  $T$  is the block which maps directional components

to spatial ones. In the case of the thin lens model, this corresponds to the inverse square law  $1/d^2$ . Note that we need additional cosine terms to convert the plane/plane parametrization to solid angle and further to projected solid angle (the space of the geometric term, see Appendix A). Finally, this yields the well-known formula for the geometric term for projected solid angle to vertex area measure:

$$G(a \leftrightarrow x) = \cos^4 \theta / d^2. \quad (6)$$

In a more complex lens model, the probability densities  $p(x), p(a)$  and  $p(y)$  are derived analogously to the thin lens model from the transport polynomials, the only difference is that there can be additional lens elements between  $\mathbf{x}$  and  $\mathbf{a}$  and between  $\mathbf{a}$  and  $\mathbf{y}$  (see Fig. 5). In effect, the measurement contribution (1) remains the same, only  $G(y \leftrightarrow x)$  now becomes a *generalized geometric term* [Jak13]. As a consequence, the Jacobian  $J_{\mathbf{x} \rightarrow \mathbf{a}}$  will be different, but the relevant determinant will still be formed from the  $B$  component of the  $ABCD$ -matrix (mapping angular densities to spatial densities), analogous to Eq. (5). Note that light tracing does not need the evaluation of any Jacobians (Sec. 3.2).

##### 4.2. Accurate Polynomials for Image Formation

After introducing the basic concept of our lens model, the next step is to derive an accurate polynomial mapping representing complex lens systems with equally high precision over the entire image plane. Hullin et al. [HHH12] used a Taylor series to approximate the mapping of a lens system, analytically expanded around the center of the screen. As a consequence, the quality of the approximation severely degrades towards the image boundaries. While this is tolerable for lens flare rendering, it is unacceptable for image formation (see Fig. 1.2 and 13).

**Initial Polynomial** We recall that all interesting aberrations are phenomenologically present when using polynomials of degree 3 or higher [Sei57]. This is why in our approach the minimal Taylor expansion of degree 3 (4 with wavelength dependency, see below) serves as a starting point for a subsequent optimization of the polynomial. We create this expansion using the same approach as previous work, by symbolically inserting analytically created Taylor expansions of individual lens elements into each other. This process is iterated through the lens description table, starting at the outer pupil towards the sensor. The polynomials for each individual element (such as transmission through a spherical element or propagation through empty space) are created using a computer algebra program, the coefficients of the combined polynomials are computed in C++ code [HHH12].

**Optimization** We then use non-linear least squares (Levenberg-Marquardt) optimization over the coefficients to match a set of randomly ray traced ground truth samples. We create a set of several thousands of valid paths through

the lens system by first sampling the sensor uniformly and shooting towards the inner pupil of the lens system. We transform the light field at every intersection using Snell's law. This results in many absorbed rays (e.g. hitting the aperture or lens body, see Table 1), and we only keep surviving paths for fitting. The paths are samples of the transformation of the light field at the sensor to the light field at the outer pupil. The error measure when fitting the polynomial is the L2 norm of the 4D difference vector of a path's computed transformation and the polynomial evaluated for the same path through the lens system. We use Levmar [Lou04] to optimize the coefficients of the polynomials; derivatives are computed via finite differences. This proved to work well for all lenses we tested (except the fisheye lens, see Fig. 6), as the initial Taylor series, although unacceptable for image formation, is not completely off.

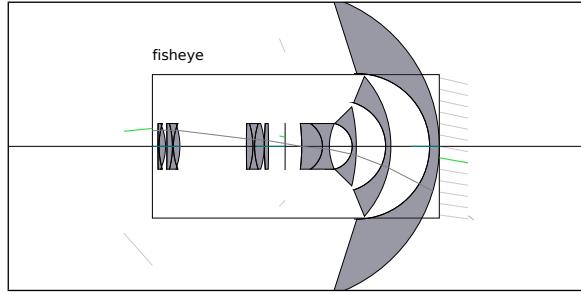
**Spectral Dimension** The wavelength  $\lambda$  is a dimension in the polynomial (not discretized before fitting) so we can change it continuously in a spectral rendering context. If grey transport is sufficient, this dimension can be left constant and the symbol  $\lambda$  can be removed from the formulas.

**Discussion** As an alternative to polynomials, we also implemented 5D piecewise cubic tensor product splines which provide high precision interpolation at low degree. Unfortunately the 5D domain of definition of the lens system is not aligned with the tensor product cube: when choosing locations on the sensor and the inner pupil as parameters only a small percentage of that domain actually corresponds to paths passing through the lens. Even unphysical, mathematically smooth extensions of the domain as proposed by Hullin et al. [HESL11] reaches its limits when a ray completely misses the spherical element, or when Fresnel's laws dictate total internal reflection. This leads to many unknown control points with huge impact on the behaviour of the interpolation, which then need to be optimized, too. Lastly, evaluating the splines will not be data-free and slower to compute.

**Example** To summarize the process, we give an example how to compute the Jacobian  $J_{a \rightarrow x}$  for the lens shown in Fig. 5. It has eight combined propagations and interface interactions on the way from the sensor to the aperture. We first go through the lens description table and extract polynomials  $P_i$  for each interface  $i$  and the following propagation up to the next interface. The combined polynomial is then  $P_{a \rightarrow x} = P_7 \circ P_6 \circ P_5 \cdots P_0$ . The fitting process then optimizes the coefficients of the polynomial to better match ray traced ground truth. The Jacobian is constructed by symbolically taking the 16 partial derivatives  $dP_{\{x,y,u,v\}} / d\{x,y,u,v\}$ .

### 4.3. Focusing

For practical reasons, focusing in real lenses is usually adjusted by moving only a few lenses inside the housing. In our virtual lenses, we can achieve the same effect by moving



**Figure 6:** Fail case of our plane/plane parametrization choice, a fisheye lens with 220 degrees field of view. The densities are extremely compressed through the optical system and the Taylor expansion (green line segment at pupil, aperture, and sensor) does not line up with the ray traced ground truth (gray path) at all. Even if fitting was done for this system, plane/plane coefficients could never represent a field of view larger than 180 degrees. See the additional material for a working wide angle example.

the sensor plane instead (which allows us to keep the polynomial fixed).

In terms of our representation this means that we express the transport through lens elements (between  $x$  and  $y$ ) as one polynomial as described above, and add another short transport step between  $x$  and the point on the sensor  $x'$  (Fig. 5). In path tracing we sample  $x'$  and determine the outgoing direction (see Sec. 5.2), and then intersect this ray with the lens black box yielding  $x$ . In this case the Jacobian undergoes another transform,  $J_{x' \rightarrow x}$ , multiplied from the right to the one described in Sec. 3.2. The subblock determinant has to be taken of the product of these two matrices. We can do that without approximation because first-order transport through empty space is lossless in plane/plane parametrization. In the case of light tracing we intersect the ray exiting the lens black box at  $x$  with the sensor to determine  $x'$ ; as before, no additional Jacobian is required in this case.

## 5. Monte Carlo Rendering with Polynomial Lenses

In this section we describe how to use our lens model, i.e. the improved polynomial representation of a camera lens (Sec. 4.2), in Monte Carlo rendering algorithms.

### 5.1. Bidirectional Transport through Lenses

In Sec. 4 we introduced our model and improved the accuracy of the polynomial representation. This polynomial  $P'(\mathbf{y}) = \mathbf{x}$  transports light from the outer pupil to the sensor. For the opposite direction (path tracing), we need the adjoint polynomial  $P(\mathbf{x}) = \mathbf{y}$ . In principle, we can construct it in a straightforward way by processing the lens description table in the reverse order.

```

evaluate_reverse:
input : y[x,y,u,v] point on the outer pupil
output: x[x,y,u,v] point on the sensor

// evaluate (approximate) adjoint polynomial
x = P'(y)
// Newton iterations
for(int it=0;it<8;it++) {
    // forward evaluate to get error vector
    y* = P(x[y,-u,-v])
    delta = y - y*[x,y,-u,-v]

    // adjoint Jacobian
    Ji = jacobian' (y*[x,y,-u,-v])
    x += Ji * delta
}

```

**Figure 7:** Pseudo code of the reverse evaluation procedure. Usually 4-8 steps are enough to get a converged solution.

However, every evaluation  $P(\mathbf{x})$  or  $P'(\mathbf{y})$  has an  $O(d+1)$  approximation error for polynomials of degree  $d$  and this error causes different results for path and light tracing, i.e. evaluating  $P(\mathbf{x})$ , flipping the sign of the direction and then evaluating  $P'(\cdot)$  on that will not end up on the same point. Our improvement of the precision of  $P(\mathbf{x})$  (Sec. 4.2) to match the reciprocal ray traced solution more closely reduces the error significantly compared to a pure Taylor expansion. Nevertheless, there still remain errors for large apertures and wide angles.

In order to remove the discrepancy between the two directions we declare one as the “reference” and adjust the other direction to match it. We decided for the path tracing direction  $P(\mathbf{x})$  as reference, as this case is more common in rendering than connecting a light path to the camera. To accomplish the matching of light tracing, we use Newton’s iteration with  $\mathbf{x} = P'(\mathbf{y})$  as the initial guess and the adjoint Jacobian  $J'(\mathbf{y})$  in subsequent iteration steps. Note that we use  $J'(\mathbf{y})$  rather than  $J^{-1}(\mathbf{y})$ , as the adjoint Jacobian is both faster to compute and numerically more stable than the matrix inversion. Fig. 7 details this procedure as pseudo code.

## 5.2. Importance Sampling the Aperture

When sampling locations on and directions off the sensor to generate rays (path tracing case) many of these rays will hit the housing or the aperture (and be absorbed), which can be avoided by importance sampling the aperture. To this end, we proceed analogously to path tracing with the thin lens model and sample points  $x'$  on the sensor and  $a$  on the aperture. The sampling densities are those described in Sec. 4.

In contrast to the thin lens model, there are lens elements between these two points and we first need to determine the directional component  $(u, v)$  at  $\mathbf{x}$ . The direction at  $a$  will then follow from  $\mathbf{a} = P_a(\mathbf{x})$ .

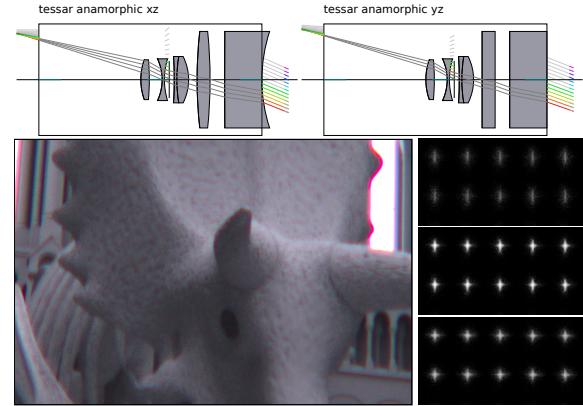
```

sample_aperture:
input: x'[x,y,u,v]
      a [x,y,u,v]
      [x,y] fixed and [u,v] initial guess
      dist, distance x' to x
output: updated [u,v] components

for(int k=0;k<4;k++) {
    // move to start of polynomial blackbox
    x = [x'[x,y] + dist * x'[u,v], x'[u,v]]
    // predict point a* on aperture
    a* = Pa(x)
    Ja(x) = [ dx/du  dx/dv ]
             [ dy/du  dy/dv ]
    Ji = invert(Ja)
    delta = (a - a*)[x,y]
    x'[u,v] += Ji * delta
}
a[u,v] = a*[u,v]

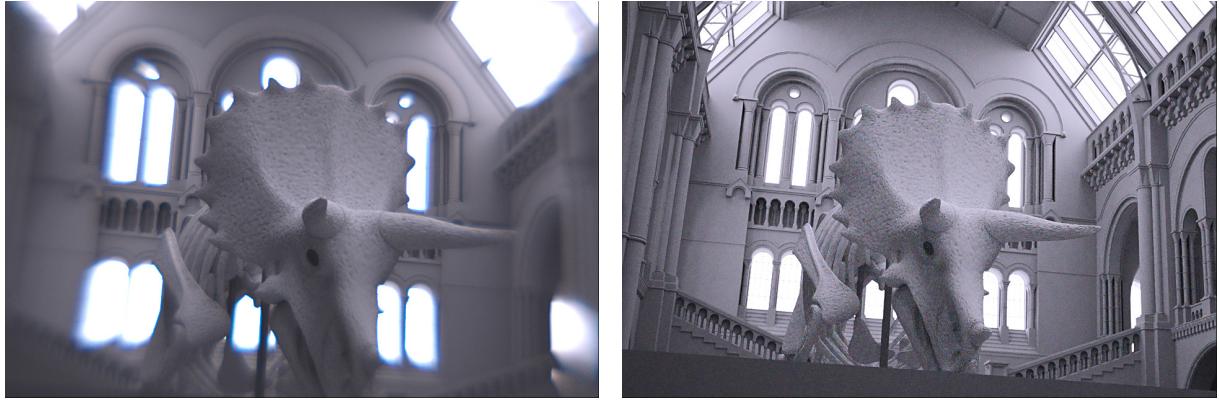
```

**Figure 8:** Pseudo code of the optimization procedure. Usually 2-4 steps are enough to get a converged solution.

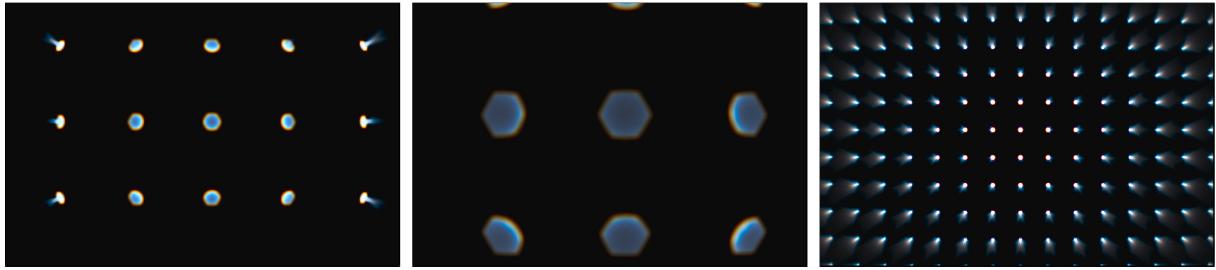


**Figure 9:** Tessar 100mm design with a custom anamorphic converter using cylindrical elements at f/8.0. The illustrations in the top row show xz and yz projections. The image shows clearly visible aberrations caused by the poor design of the converter. Right: point spread functions for the channels red, green, blue (from top to bottom).

In principle we have to connect two points ( $x$  and  $a$ ) through a specular chain of interactions (lens elements), which is related to manifold exploration [Jak13], however, we do not have a valid seed path to start from. But again, we can make use of our polynomial lens description: it proved to be sufficient to use the direction pointing from  $x$  to  $a$  ignoring the lens system at first as an initial guess, and refine using Newton’s method. This iteration (Fig. 8) uses the  $2 \times 2$  block  $(dx, dy) \times (1/du, 1/dv)$  out of the Jacobian and converges quickly. The parameter  $dist$  in the pseudo code handles sensor offsets for focusing (see Sec. 4.3).



**Figure 10:** Images taken with a single bi-convex, uncorrected 35mm lens equipped with an aperture at f/2.8. Left: rendering with a fitted polynomial of degree 4. Right: using a polynomial of degree 1 using plain Taylor coefficients: this model has too few degrees of freedom and a fit to the ray traced solution would result in an overall blurry image.



**Figure 11:** Point spread functions evaluated with our fit for a simple lens (single bi-convex lens, same as Fig. 10). The shape of the dots changes towards the borders of the image due to clipping at the hexagonal aperture. The right image shows how the dots which are too close to be in focus turn red whereas the points too far away turn blue due to chromatic aberrations.

## 6. Implementation Details

We compute the initial polynomial from the Taylor expansion using the source code provided by Hullin et al. [HHH12], and use the `levmar` package [Lou04] to optimize its coefficients. In order to obtain high performance, we compile the polynomials  $P(\mathbf{x})$ ,  $P'(\mathbf{y})$  and  $P_a(\mathbf{x})$  as `c99` code to be included in our rendering framework. The resulting code is 20x (forward evaluation)–60x (aperture sampling) faster than interpreting the polynomial from lists of coefficients. Note that it is necessary to write the code out in full floating point precision (i.e. using `%g` not `%f` in `fprintf`) as the coefficients can be very small. We found, however, that single precision floating point arithmetic is sufficient for our needs.

As we simulate transport through dispersive camera lenses, we use spectral rendering: we use bidirectional path tracing with multiple importance sampling weighting [Vea97] where every path transports spectral radiance for one differential wavelength. To assure that the bokeh matches exactly between path tracing and light tracing, we clip the rays at the inner pupil, the outer pupil, and the aper-

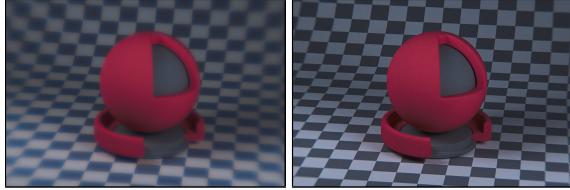
ture. We only need to do that explicitly for the openings which were not sampled directly.

## 7. Results

All our images were created on an Intel Core i7 hexacore with 3.20GHz. Fig. 9 shows results for an anamorphic lens design. These are well represented in our framework since we work on the full 4D light field.

Our formulas degenerate to a rectilinear model if only first-order terms are used (see Fig. 10), resulting in a transform similar to the thin lens model. Fig. 11, using the same simple bi-convex lens as in the previous figure, shows point spread functions to illustrate that our method produces the five Seidel aberrations (coma, spherical aberration, curvature of field, astigmatism, distortion) and chromatic aberration. Fig. 12 illustrates a combined effect of aberrations, where a small f-stop value introduces overall contrast loss and a slight color shift.

Table 1 shows the survival rates of samples through the lens system, i.e. the ratio of primary rays created off the sensor actually exiting the lens into the scene. The maximum



**Figure 12:** A simple lensbaby-like lens (achromatic doublet) at f/2.8 (left) and f/8.0 (right). Stopping down improves sharpness and contrast.

number reported for pre-tabulated pupil sampling [SDHL11, Table 1] is 87.6% for the Brendel Tessar design at f/2.8. Our data-free aperture sampling yields 99.1% for this setting, and 100% at f/4.0. As can be seen, our technique yields close to optimal survival rate, even though we only explicitly sample the aperture and not the combination of outer pupil and aperture. Apparently, good lens design takes care that rays passing the aperture also penetrate through the outer pupil. This is also supported by the scatter plots in [SDHL11, Fig. 9] and Fig. 11 (middle), which suggests that the elliptical shape of the point spread function is due to distortion of the aperture shape, not due to clipping at the outer pupil. The reduced rate of survivors in pupil sampling [SDHL11] is probably due to the conservative nature of the fitted circle bounding the *pixel pupil*. Also note that we can sample the shape of the aperture directly (e.g. six rounded blades vs. full circle).

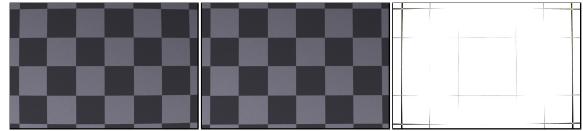
A performance evaluation of our algorithm is difficult: as it does not access any data, but is compute-only, some of its overhead is hidden by memory fetches during ray tracing in a full rendering system. Tracing a ray through the Canon zoom lens, for example, takes 26 times longer than evaluating the corresponding polynomial of degree 4 in a standalone unit test. We further implemented ray traced aperture sampling based on Jacobians from ray differentials (also used in Figure 1). We compute the offset at the inner pupil using transport matrices obtained from a manifold walk aiming to end at the aperture point [Jak13]. Ray tracing through the system does not always succeed (Sec. 4.2). Taking the optical axis as starting point for the iteration is possible, but leads to increased iteration counts. Our experiments use a constant number of iterations for better comparison and yield the following timings relative to the evaluation of the thin lens model:

thin lens model	1.00x
lens (no glass between aperture and sensor), degree 1	0.94x
lens (no glass between aperture and sensor), degree 4	1.18x
four lens elements, polynomial degree 4	2.98x
Canon zoom lens, polynomial degree 4	2.98x
Canon ray traced/ray differentials	85.00x

If no lens elements are placed between the aperture and the sensor, aperture sampling converges in the first iteration and is thus faster compared to the general case.

lens f/#	Canon zoom		Simple		Tessar		
	IS	RT	IS	RT	IS	RT	PS
f/16	100	0.2	100	0.7	100	2.4	-
f/11	100	0.3	100	1.5	100	5.0	-
f/8	100	0.6	100	2.9	100	9.4	85.7
f/5.6	100	1.2	100	5.9	100	19.3	-
f/4.0	100	2.4	100	11.6	100	37.5	-
f/2.8	100	4.9	100	23.7	99.1	64.2	87.6

**Table 1:** Effectiveness of aperture importance sampling. The numbers show the percentage of started rays that make it through the lens system without being absorbed by the housing or the aperture blades. IS is our importance sampling, RT straight forward ray tracing, PS pupil sampling (taken from [SDHL11] where available). Since we only sample the aperture, not the combination of aperture and outgoing pupil, the effectiveness of our method degrades slightly at smaller f-stop values.



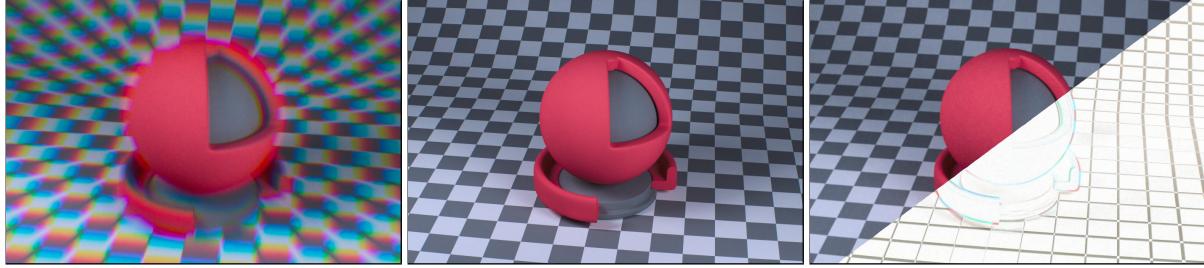
**Figure 14:** Rendering with the Canon zoom lens (left) and after lens corrections applied in external photography software (center). The software uses correction data of the real lens. Our method is so accurate that this correction can be directly applied and also straightens the barrel distortion of our renders. Right: difference image (inverted for print) before and after correction.

Fig. 13 shows how badly the plain Taylor expansion matches ground truth, even if wavelength-dependency is ignored. In contrast, we applied third-party lens correction software [KZB\*07] to our render to demonstrate that real world calibration data for this lens design matches the distortions we synthesize (see Fig. 14).

As additional material, we provide Maxima [Max13] sheets for the plane/plane parametrized Taylor expansion, as well as source code to compute an image using a simple lens system. The latter uses the *smallpt* source code [Bea10] and adds minimalistic lens code for path tracing (without aperture sampling).

## 8. Limitations and Future Work

A failure case for the plane/plane parametrization is a fisheye lens (Fig. 6). This is a widely acknowledged special case and handled separately in many models [CJC\*10]. We handle wide angle lenses reasonably well (see additional material). Extreme cases, however, require fall back to the slower ray traced aperture sampling.



**Figure 13:** A Canon zoom lens ( $f/2.8L 70-200mm$ ) at  $f/8.0$  rendered using Taylor expansion [HHH12] (left) and our fitted polynomial (center). The wavelength dimension is problematic in the Taylor series. The fit shows that this design is in fact an excellent lens. Right: using a gray Taylor series, i.e. ignoring the wavelength domain and fixing  $\lambda = 500nm$ , avoids color fringing. Note, however, how the difference image reveals the wrong barrel distortion.

While our importance sampling of the aperture works very well for the path tracing direction, we can only connect to the outer pupil in the light tracing case. Recall that we declared the path tracing polynomial  $P(\mathbf{x}) = \mathbf{y}$  as the reference which maps points on the sensor to the outer pupil. In aperture sampling in light tracing we would have a fixed point on the scene geometry and sample one on the aperture. Determining the inbetween point on the pupil using  $P(\mathbf{x}) = \mathbf{y}$  is difficult and subject to constraints regarding  $\mathbf{x}$  and  $\mathbf{a}$ . Another issue is that sampling the aperture is not the best strategy in this case, as certain object points will be imaged onto the sensor only through a small fraction of the aperture (Fig. 2, border case).

As Hullin et al. [HHH12] we do not consider Fresnel’s laws when evaluating ray transmission. Our results will thus be consistently too bright. As lens manufacturers try to optimize for high transmission this should be reasonably close to reality. Nevertheless, our primary goal is an efficient, practical rendering method and not lens design, but precision can always be increased by using higher order polynomials.

We implemented spherical and cylindrical elements. Prisms and aspherical elements could be handled in the same framework. Provided that there is a ray traced ground truth, aspherical elements could probably be added via fitting of an initial polynomial from a spherical element.

It would be interesting to apply the fitting procedure also to lens flare rendering and evaluate the visual difference of the improved accuracy.

## 9. Conclusion

We showed how a precise polynomial approximation to optical lens systems can be used for image formation in a bidirectional Monte Carlo path tracing context. The result closely matches the ray traced ground truth. The computational overhead over the well-known thin lens model is only small, and since we do not access any data it can, to some extent, be hidden between memory accesses taking place in ray tracing. Sampling the aperture results in close to optimal ray survival rates and all features of a modern Monte

Carlo renderer are supported. Bidirectional path tracing, as well as lens perturbations in Metropolis light transport are supported, as the generalized geometric term cancels out of the ratio measurement contribution/transition probability. Lastly, ray differentials can be implemented using the Jacobian of the polynomial system.

## References

- [Bea10] BEASON K.: smallpt: Global Illumination in 99 lines of C++. <http://www.kevinbeason.com/smallpt/>, 2010. [8](#)
- [CJC\*10] CHEN S., JIN H., CHIEN J., CHAN E., GOLDMAN D.: *Adobe Camera Model*. Tech. rep., Adobe, 2010. [2](#), [8](#)
- [Gau41] GAUSS C.: *Dioptrische Untersuchungen*. Dieterich, 1841. [2](#)
- [GKK\*04] GUNJI T., KIM S., KOJIMA M., TAKEDA A., FUJISAWA K., MIZUTANI T.: PHoM – a polyhedral homotopy continuation method for polynomial systems. *Computing* 73, 1 (July 2004), 57–77. [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. [5](#)
- [HHH12] HULLIN M. B., HANIKI J., HEIDRICH W.: Polynomial Optics: A construction kit for efficient ray-tracing of lens systems. *Computer Graphics Forum (Proc. of EGSR)* 31, 4 (July 2012). [2](#), [3](#), [4](#), [7](#), [9](#)
- [Ige99] IGEHY H.: Tracing ray differentials. *Proc. of SIGGRAPH* (1999), 179–186. [3](#)
- [Jak13] JAKOB W.: *Light transport on path-space manifolds*. PhD thesis, Cornell University, 2013. [2](#), [4](#), [6](#), [8](#)
- [KMH95] KOLB C., MITCHELL D., HANRAHAN P.: A realistic camera model for computer graphics. In *Proc. of SIGGRAPH* (1995), pp. 317–324. [2](#)
- [KSKAC02] KELEMEN C., SZIRMAY-KALOS L., ANTAL G., CSONKA F.: A simple and robust mutation strategy for the Metropolis light transport algorithm. In *Computer Graphics Forum (Proc. Eurographics)* (2002), pp. 531–540. [1](#)
- [KZB\*07] KRAFT S., ZABOLOTNY A., BRONGER T., ET AL.: lensfun: Open source lens correction library. [web page] <http://lensfun.berlios.de>, 2007. [2](#), [8](#)
- [LE13] LEE S., EISEMANN E.: Practical real-time lens-flare rendering. *Computer Graphics Forum* 32, 4 (2013), 1–6. [2](#)

- [LLT08] LEE T. L., LI T. Y., TSAI C. H.: HOM4PS-2.0: a software package for solving polynomial systems by the polyhedral homotopy continuation method. *Computing* 83, 2-3 (Nov. 2008), 109–133. [2](#)
- [Lou04] LOURAKIS M.: levmar: Levenberg-Marquardt nonlinear least squares algorithms in C/C++. [web page] <http://www.ics.forth.gr/~lourakis/levmar/>, Jul. 2004. [5](#), [7](#)
- [Max13] MAXIMA: Maxima, a computer algebra system, 2013. [8](#)
- [SDHL11] STEINERT B., DAMMERTZ H., HANICA J., LENSC H. P. A.: General spectral camera lens simulation. In *Computer Graphics Forum* (2011), vol. 30, pp. 1643–1654. [2](#), [8](#)
- [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](#), [4](#)
- [STN87] SHINYA M., TAKAHASHI T., NAITO S.: Principles and applications of pencil tracing. *Proc. of SIGGRAPH '87* (1987), 45–54. [2](#), [3](#), [4](#), [10](#)
- [Vea97] VEAUCH E.: *Robust Monte Carlo Methods for Light Transport Simulation.* PhD thesis, Stanford University, 1997. [3](#), [7](#)

## Appendix A: Converting Plane/Plane Jacobians to Projected Solid Angle

Consider the Jacobian of an optical system in plane/plane space:

$$J_{\mathbf{x}' \rightarrow \mathbf{a}} = J_{\mathbf{x} \rightarrow \mathbf{a}} \cdot F = \begin{pmatrix} A & B \\ C & D \end{pmatrix},$$

where  $F$  is the matrix that traces the ray from the focused sensor plane  $\mathbf{x}'$  to the start of the black box  $\mathbf{x}$ .

To apply this transformation to a ray  $\hat{\mathbf{x}'}$  in solid angle space (plane/sphere parametrization), we need to renormalize the directional part from  $L_2$  norm to  $L_\infty$  norm by multiplying the Jacobian from the right with

$$N = \begin{pmatrix} 1 \\ l \end{pmatrix}$$

$$J_{\mathbf{x}' \rightarrow \mathbf{a}} \cdot N = \begin{pmatrix} A & l \cdot B \\ C & l \cdot D \end{pmatrix}$$

where  $l$  is the length of the directional component  $(u, v)$  in plane/plane parametrization  $l = \sqrt{1 + u^2 + v^2}$  or  $1/l = \cos \theta$  which is used to convert  $(u, v)$  to solid angle space  $(\xi, \zeta)$ .

Along the lines of [STN87], the combined Jacobian determinant of the  $B$  block in solid angle space is then  $\frac{|dx dy|}{|d\xi d\zeta|} = |l \cdot B| = l^2 |B| = |B|/\cos^2 \theta$ . We then compute the generalized geometric term as the ratio between projected solid angle measure  $d\omega^\perp$  and vertex area measure  $dA$ :

$$\begin{aligned} G(x' \leftrightarrow a) &= d\omega^\perp / dA \\ &= \frac{d\xi d\zeta \cdot \cos \theta}{dx dy / \cos \theta} \\ &= \cos^4 \theta / |B|. \end{aligned}$$