



Real-time ray transfer for lens flare rendering using sparse polynomials

Andrea Bodonyi^{1,2} · István Csoba^{1,2} · Roland Kunkli¹

Accepted: 29 August 2024
© The Author(s) 2024

Abstract

While lens flares are often undesired artifacts of the imaging process, they are also essential for increasing the level of realism in video games and serve as a powerful artistic tool for photography and filmography. For these reasons, computationally reproducing lens flares has always received special attention. Due to the cost of analytical ray tracing, existing solutions are unable to simultaneously achieve the performance needed for real-time environments and retain the ability to simulate arbitrarily complex ghost shapes. Although polynomial optics has been successfully used to increase the efficiency of ray tracing in multiple rendering areas, no complete and validated solution exists that correctly models all aspects of lens flares. This paper presents our polynomial optics-based method for efficiently and accurately ray tracing lens flare ghosts. Our approach successfully models the shape, energy absorption, chromatic effects, and blocking of lens flare rays by partitioning the input domain into local fitting zones. We demonstrate that our model provides a considerable speedup and high accuracy compared to the analytical approach and achieves better fitting speed, output quality, and rendering performance than the naïve application of polynomial optics. The source code for our implementation is available on [GitHub](#).

Keywords Lens flare rendering · Polynomial optics · Ray tracing · Camera lens effects · Sparse polynomial regression

1 Introduction

The lens flare phenomenon of an optical system is caused by the unwanted diffraction and internal reflection of light rays as they enter and travel through the optical system. Conceptually, lens flare can be divided into two main components: starburst (caused by diffraction) and ghosts (the result of internal reflection). Although often viewed as an undesired artifact of the imaging process, they are essential artistic elements in the field of photography and filmography [1, 2] and contribute significantly to increasing the level of realism in video games. Furthermore, they also have several critical scientific applications, such as the ability to study the impact of

glare on a variety of everyday tasks and provide visual cues of high-intensity light sources in synthetic imagery [3].

Because of their high relevance, numerous solutions exist in the scientific literature for computationally simulating lens flares. However, current methods all make compromises to achieve their desired goals, be it plausibility (by omitting physical correctness) or simulation performance (by relying on ray tracing). Consequently, no existing solution can accurately and efficiently simulate arbitrarily shaped, physically accurate lens flares while satisfying the performance requirements of real-time applications.

Despite the importance of high-performance, physically plausible lens flare simulation, recent works generally focus on removing lens flares [4–7] and not their computational synthesis. Most of these methods rely heavily on the use of training data comprising images with lens flare [5, 8]. These datasets are typically constructed using digital cameras, making the process difficult, highly error-prone, and excessively time-consuming [5]. Because of these reasons, computational synthesis of lens flare training images can be another potential area for high-performance lens flare simulation techniques.

✉ Andrea Bodonyi
bodonyi.andrea@inf.unideb.hu

István Csoba
csoba.istvan@inf.unideb.hu

Roland Kunkli
kunkli.roland@inf.unideb.hu

¹ Faculty of Informatics, University of Debrecen, Debrecen, Hungary

² Doctoral School of Informatics, University of Debrecen, Debrecen, Hungary

Although polynomial optics [9] has been outlined as a promising tool for solving these problems, previous polynomial optics-based lens flare simulation approaches have all ignored key aspects of lens flares and omitted the validation of the resulting simulations, severely limiting the practicality of these methods.

In this paper, we present our polynomial optics-based ray transmission model for simulating the lens flares of optical systems. Our proposed method facilitates the high-quality, real-time simulation of lens flare ghosts while properly treating all important characteristics of the phenomenon. Compared to analytical ray tracing, our solution achieved a two-to-five-times speedup in all tested scenarios and provided a considerable increase in performance and accuracy over the naïve application of polynomial optics. Our main contributions are:

- A polynomial optics-based ray-transfer model tailored to the simulation of lens flare ghosts.
- A custom scheme built on partial fitting regions and shared polynomial bases, which greatly improves the fitting speed, reduces the number of polynomial terms, and increases the polynomial evaluation performance.
- A data filtering and model fitting procedure for our proposed model, which efficiently constructs small polynomial systems for the input optical system, thereby facilitating the highly accurate, real-time simulation of the optical system's lens flares.
- An evaluation of rendering quality and performance of polynomial optics-based lens flare rendering with respect to analytically ray-traced reference simulations. Example output simulations generated using our proposed method are shown in Fig. 1 for two different optical systems.

2 Previous work and motivation

2.1 Rendering *lens* flares

The earliest approaches for rendering lens flares build on the use of texture sprites. The physical properties of the optical system are ignored, and the textures are placed empirically on the image. Kilgard [10] suggested an algorithm that positions the flare textures on a line that goes through the center of the image. King [11] then extended this approach by altering the flare intensities based on their distances from the image center. Later, Maughan [12] proposed a method for determining the ghost intensities accounting for the partial occlusion of the light source in the scene, in which approach was further improved by Sekulic [13] with a GPU-based solution for the calculation. Oat [14] replaced the use of textures with a postprocessing filter applied to the high-intensity points

of the image. Finally, Alspach [15] proposed a vector-based representation for modeling the lens flare ghosts.

The first physically based algorithm for simulating lens flare ghosts was proposed by Chaumond [16], who used path tracing with the refracting interfaces of a real optical system. Keshmirian [17] then applied photon mapping as an alternative for path tracing, and Steinert et al. [18] proposed an algorithm considering the material coatings for producing spectral lens flares. Lendermann et al. [19] then considered the precise diffraction simulation and utilized optical systems with arbitrary focal lengths and aperture sizes. Hanika and Dachsbacher [20] used a Monte Carlo renderer to further improve precision. As an alternative to these optics-based approaches, Walch et al. [21] processed digital images, whereby the lens flares in the input images were located, and a set of Bézier curves were fit to their shapes, facilitating the synthesis of lens flares with arbitrary light source positions.

Although the algorithms described above can be used to generate plausible results, they are computationally demanding. To overcome this issue, Hullin et al. [22] utilized sparse ray grids and interpolation to significantly reduce the number of ray-tracing operations. Their algorithm traces a coarse ray grid through the optical system for all ghosts, the results of which are interpolated and rasterized on the GPU to produce the output. While practically usable (as demonstrated by Pekkarinen and Balzer [2]), a huge disadvantage of this method is that the ghosts are rendered individually, severely limiting the rendering throughput for large numbers of lens flares. Bodonyi and Kunkli [23] solved this problem by using screen-space tiles and software-based rasterization to efficiently accumulate the contributing quads for each tile and produce the final result in a single step.

To further improve rendering performance, Lee and Eisemann [24] used paraxial ray tracing and transfer matrices to obtain the lens flare ghost positions, sizes, and intensities. Hennessy [25] traced a single ray through the optical system to obtain the color and utilized the effective aperture of the optical system to calculate the ghost intensity. A significant drawback of these approaches lies in the inability to handle complex ghost shapes because each ghost is rendered as a textured quad.

The creation of the starburst component of the lens flare phenomenon is similar to the glare effect in the human eye because they both result from the diffraction and light scattering of the obstacles in the path of light. Kakimoto et al. [26] applied Fraunhofer diffraction to simulate the glare in the human eye. Van den Berg et al. [27] then utilized the Rayleigh–Debye–Gans approximation for modeling the scattering by the molecules inside the human eye. Ritschel et al. [3] modeled the glare produced by the human eye using Fresnel's diffraction integral and the fast Fourier transform. Similar to these methods, the starburst component of the lens



Fig. 1 Examples of lens flare renderings using our proposed polynomial optics-based ray-transfer approach for a Heliar Tronnier (left) and Angenieux (right) camera system and two light sources. Ray transfer took, respectively, 0.43 ms (left) and 0.83 ms (right) on an NVIDIA

TITAN Xp GPU. For reference, the total lens flare simulation took 1.84 ms (left) and 2.01 ms (right) at a display resolution of 1920×1080

flare phenomenon was first modeled by Hullin et al. [22] using the Fourier transform of the aperture image. Joo et al. [28] improved this method by simulating the lens imperfections coming from the manufacturing process and extended the algorithm to handle aspheric lenses. Finally, Scandolo et al. [29] proposed a hierarchical algorithm for the fast computation of the Fraunhofer diffraction.

2.2 Polynomial optics

Analytical ray tracing produces highly realistic images but requires high computational expenses. Therefore, researchers have proposed using polynomial systems to approximate the result of ray tracing in a much faster manner.

From such methods, Hullin et al. [9] created a construction kit for obtaining the ray-transfer functions using their truncated Taylor expansions. They utilized a system of polynomials for all lens surfaces in the optical system, from which the final transfer function was created by concatenating the polynomials. Schrade et al. [30] then extended the polynomial method for wide-angle lenses and developed an improved polynomial-fitting method by selecting the most relevant terms from all the candidates. Later, Zheng and Zheng [31] utilized adaptive polynomial regression to construct sparse polynomial lens models, which further improved the accuracy and speed of the polynomial-fitting procedure. Goossens et al. [32] suggested a method for fitting polynomials to existing ray-traced data.

Regarding lens flare rendering with polynomial optics, Hullin et al. [9] demonstrated that their construction kit approach can simulate lens flares in offline environments. However, their approach ignored surface reflectance and ray blocking by the lens housing, which significantly altered the resulting flare renderings. Pekkarinen [2] also utilized Taylor polynomials to render lens flares but extended the method of Hullin et al. by testing the image sensor for lens house blocking and using Shlick's formula to coarsely approximate the

transmitted energy at the front lens surface. Later, Dilorio [33] proposed further optimizations to reduce the number of polynomial terms, but they only evaluated their results using a very small, synthetic lens setup. Recently, Sabatschus [34] used sparse polynomial systems for modeling lens flares. However, the presented results did not indicate real-world usability because the author could not utilize the resulting polynomials to produce plausible lens flare renderings.

The existing polynomial optics-based lens flare simulation approaches outlined above have several significant limitations. Firstly, the correct per-element evaluation of blocking by the lens housing and the evaluation of the Fresnel equations for calculating the transmitted and reflected energy were ignored in all these works. Secondly, results verification with ground-truth renderings was not performed in any of the successful approaches. Thirdly, previous works did not focus on facilitating high-performance, GPU-based evaluation, which usually leads to large polynomial systems, substantially increasing the cost of evaluating the polynomial system. Finally, relating to the previous problem, the rendering performance was not studied in detail in any related works.

2.3 Summary and motivation

With recent works mostly focusing on removing lens flare ghosts rather than their digital synthesis, to our knowledge, no solution is available in the literature capable of the physically correct, real-time rendering of lens flare ghosts for arbitrarily complex optical systems. Although recently Bodonyi and Kunkli [23] provided a solution for the efficient rasterization of ghosts, the computational cost of ray tracing remains a severe bottleneck for the physically accurate, real-time simulation of ghosts. Direct analytical approaches are only suitable for real-time applications if the size of the optical system (and thus the number of simulated flares) is small and if only simple (planar and spherical) surface elements

are utilized. Although polynomial regression can be used to overcome these limitations for many components of the image-synthesis pipeline, the complexity of lens flare light paths and resulting flare shapes make it highly nontrivial to apply such methods to real-time, high-fidelity lens flare rendering, as demonstrated by all relevant prior work in this area [2, 9, 33, 34]. Additionally, the validity of these models is also questionable, as no proper validation has been performed on any of these models. Finally, while transfer matrices [24] can also be utilized for efficiently performing ray tracing for lens flare rendering, this approach cannot handle complex ghost shapes, severely degrading the plausibility of the resulting simulations. The main motivating factor for this work was to overcome these limitations; namely, to facilitate the real-time ray tracing of arbitrarily shaped lens flare ghosts using polynomial optics, for arbitrary optical system complexity and with the proper treatment of all aspects of lens flares.

3 Mathematical background

3.1 Structure and properties of an optical system

To perform analytical ray tracing with a real optical system, knowing its physical properties is necessary. For our goals, we confine our camera structure to a single aperture and multiple lens elements, with each lens element comprising two or more refractive surfaces.

The optical system is represented as a series of planar and spherical surface elements, which increases the simplicity of the model and the efficiency of the ray-tracing process. Each element contains several physical surface properties, including the refractive index, material, radius of curvature, thickness, and height. Finally, we utilize a texture mask to model the shape of the aperture. Table 1 and Fig. 2 demonstrate an example of such a system description.

Note that real cameras may contain elements that cannot be described by planar and spherical surfaces. However, our proposed method can be trivially extended to support such surfaces if the necessary analytical intersection formulas are available. Their representations and the corresponding intersection tests are harder to obtain and properly implement, which is the main reason why we ignored these surfaces in our work. However, the corresponding computations are often substantially slower, and thus, optical systems with such elements would benefit even more from the polynomial approximation strategy outlined in this paper.

3.2 Rendering ghosts with sparse ray tracing

Lens flare ghosts are generated by the light rays that undergo an even number of reflections, as only such rays can reach the optical system's sensor. The energy of the light ray decreases

with every reflection, and therefore, the visibility of the resulting ghost becomes significantly lower as the number of reflections gets higher. Because of this reason, Hullin et al. [22] considered light paths with only two reflections because ghosts with more reflections are often nearly invisible. Furthermore, light passing through the aperture multiple times is likely to be mostly blocked by the aperture. It thus can be ignored with a negligible impact on the plausibility of the simulation.

To render lens flare ghosts using coarse ray tracing, Hullin et al. [22] traced a sparse grid of parallel rays through the optical system for all simulated ghosts, which yields a projected ray grid on the sensor. The resulting sparse grid is then filled using hardware-accelerated interpolation. During light traversal, the amount of reflected energy and the reflected or refracted ray directions are calculated using analytical formulas as the rays interact with each lens surface of the optical system. Besides these parameters, their approach keeps track of several other ray properties necessary for evaluation. These parameters include the aperture-plane coordinates for sampling the aperture mask texture and the highest distance from the optical axis (normalized using the height of the optical system and referred to as *relative radius*) encountered during ray tracing, which is used to detect if the aperture or the lens housing blocks a ray.

3.3 Polynomial optics

Polynomials are a set of functions obtained by summarizing a set of *monomials*. Therefore, a polynomial $p(x)$ can be formulated in the following form:

$$p(x) = \sum_{k=0}^t m_k, \quad (1)$$

where $x = (x_1, x_2, \dots, x_d)$ is the d dimensional input vector of the polynomial, t is the number of monomials, and m_k is the k th monomial. The m_i monomial is the product of a coefficient and each input variable raised to a specific power, formally written in the following way:

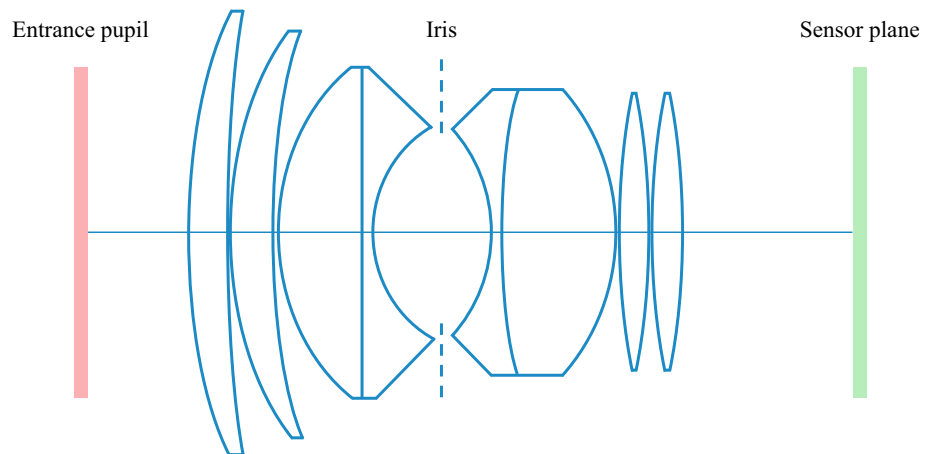
$$m_i = c_i \cdot \prod_{j=1}^d x_j^{p_j^i}. \quad (2)$$

Based on the presence of these monomials, a polynomial is called *dense* if all monomials for a given degree have a nonzero coefficient ($c_i \neq 0$ for all i) and *sparse* otherwise.

Polynomials come with a large set of advantages, including their smoothness, the ability to easily fit to arbitrary input data, their comparatively low evaluation costs, and the simple evaluation of their derivatives. In practice, ray transmission with polynomials is typically performed by constructing a distinct polynomial for each output ray parameter, which

Table 1 Physical structure of an Angenieux double-Gauss lens (USP 2701982A, $f/11$, 100 mm focal length, 14 refracting surfaces)

Radius	Thickness	Material	Refractive index	Height
164.13	10.99	SF5	1.6751	27.0
559.20	0.23	air		27.0
100.12	11.45	BAF10	1.6689	25.5
213.54	0.23	air		25.5
58.04	22.95	LAK9	1.6913	20.5
2551.10	2.58	SF5	1.6751	20.5
32.39	15.66	air		13.5
10,000	15.00	air (iris)		12.75
− 40.42	2.74	SF15	1.6992	12.5
192.98	27.92	SK16	1.6204	18.0
− 55.53	0.23	air		18.0
192.98	7.98	LAK9	1.6913	17.5
− 225.30	0.23	air		17.5
175.08	8.48	LAK9	1.6913	17.5
− 203.55	55.742	air		17.5

Fig. 2 Schematic of the Angenieux lens described in Table 1

directly transforms the input ray from a selected plane related to the optical system (such as the entrance pupil) to another plane of interest (such as the aperture plane or the optical system's sensor). This approach facilitates the omission of the underlying optical system structure and treats the system as a black box once the corresponding polynomial system is constructed. A schematic of this approach is shown in Fig. 3.

4 Our method

We constructed a polynomial optics-based algorithm for the real-time, physically accurate simulation of lens flare ghosts for highly complex optical systems with a large number of reflective surfaces. Conceptually, our approach consists of two main phases: precomputation and rendering. During precomputation, we perform polynomial fitting for the ghosts

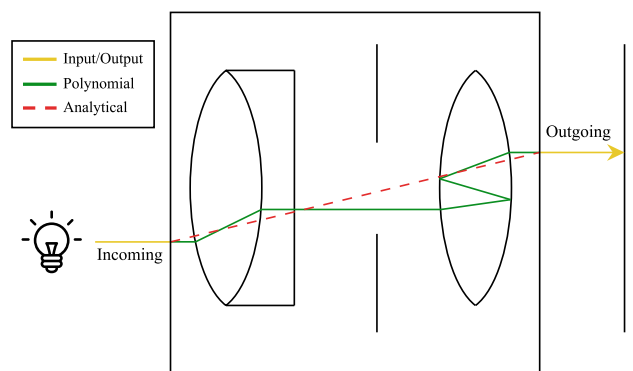
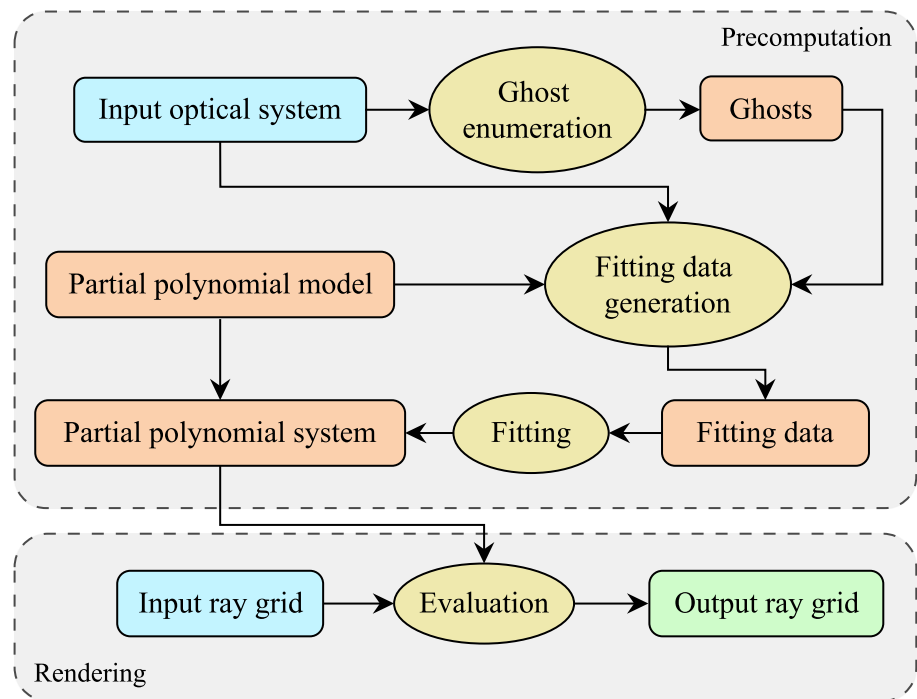
**Fig. 3** Visualization of the difference between traditional ray tracing and polynomial optics. Given an incoming ray, analytical ray tracing evaluates each surface of the optical system sequentially to produce the output (green). With polynomial optics, the incoming ray is directly transformed into the output ray, disregarding the physical structure of the optical system (red)

Fig. 4 Overview of the main steps of our proposed polynomial optics-based lens flare ghost rendering approach



of the input optical system to obtain the corresponding polynomial systems, using a custom polynomial model that is subdivided into individual regions and a corresponding fitting process. During the rendering phase, we then utilize the polynomial systems obtained during precomputation to perform ray transmission, using GPU-based vectorization to efficiently evaluate the polynomial systems and coarse ray tracing with interpolation to achieve real-time performance. An overview of the main steps of our method is shown in Fig. 4.

During the description of our algorithm, we only consider directional light sources (with parallel incoming light rays), which is commonly done in other relevant works as well. The size and intensity of the starburst are typically overbearing and cause the ghost patches to be highly unnoticeable when the light source is close to the observer. The amount of light emanated by such sources is also typically small enough that the corresponding ghosts are mostly invisible. Most practically relevant sources of lens flare ghosts are distant light sources, where incoming rays are nearly parallel. Additionally, as outlined in Sect. 3.1, we only considered planar and spherical lens surface elements, in which case the optical system exhibits rotational symmetry. Consequently, only one incidence angle needs to be considered as an input parameter; therefore, the length of the polynomial-fitting procedure can be significantly shortened. We note that these simplifications do not hinder the usability of our method and are only meant to reduce the length and complexity of the model description;

our approach can be trivially extended to handle not rotationally symmetrical optical systems with point lights and spot lights via a small change of the polynomial parametrization.

4.1 Our polynomial model of lens flares

Previous uses of polynomial optics typically focused on offline ray-tracing applications. Consequently, these models did not focus on minimizing the number of input variables and included several outputs (most notably, the outgoing ray direction) that are unnecessary for simulating lens flares. Additionally, previous works typically relied on the straightforward approach of using the 2D ray coordinate and direction vectors as inputs, which can make the number of required sparse polynomial terms excessively large. Correctly modeling per-lens transmittance, reflectance, and lens house blocking is also difficult with such input variables.

To construct a polynomial model that is better suited for real-time lens flare ghost rendering, we used the following input and output variables:

$$(x_p, y_p, r_p, \bar{r}_p, \theta, \lambda) \rightarrow (x_a, y_a, x_s, y_s, \tau_s, \rho_s), \quad (3)$$

where (x_p, y_p) is the ray position on the entrance pupil, r_p is the radius component of the polar coordinates corresponding to (x_p, y_p) , $\bar{r}_p = 1 - r_p$, θ is the light incidence angle, and λ is the light wavelength. Regarding outputs, (x_a, y_a) and (x_s, y_s) are the ray positions on the aperture and the sensor, and τ_s and ρ_s are the accumulated ray intensity and aggregated

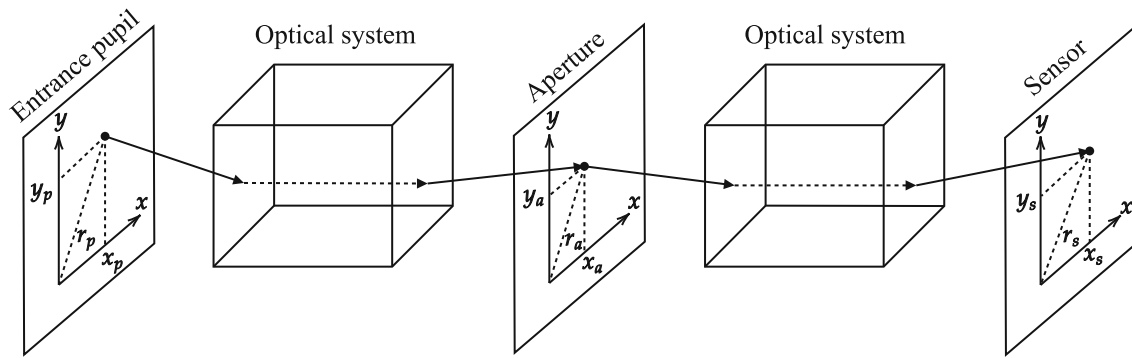


Fig. 5 Visualization of the main planes and coordinate systems used by our polynomial lens flare transmission model defined in (4)

relative radius on the sensor. Figure 5 visualizes the relevant planes and variables.

As opposed to previous models, since our algorithm assumes rotational symmetry, we can simply utilize the incidence angle θ directly instead of the full 2D input direction vectors. This approach reduces the dimensionality of the input vectors and thus increases the performance of polynomial evaluation. Another notable difference between our model and previous approaches is the use of the radius and inverted radius components (r_p and $\bar{r}_p$) as inputs. By including these terms in the outputs of our polynomials, they can describe round shapes more precisely with fewer terms, directly impacting rendering performance. This property is critical for correctly modeling the ρ_s relative radius component, which will be discussed below.

Similar to previous works [2, 9], our polynomial model also includes the aperture- and image-space coordinates as outputs, which is necessary to determine the final ghost coordinates and to handle blocking by the optical system's aperture and sensor. However, previous approaches allowed ghost paths to cross the aperture multiple times [2, 9], which requires multiple separate aperture coordinates to be modeled with polynomials. As noted by Hullin et al. [22] and outlined in Sect. 3.2, such ghosts tend to be mostly blocked by the aperture, making their contributions largely negligible. By constraining our model to only include ghost paths with a single aperture crossing, we can substantially reduce the number of output variables and simplify and speed up the ray evaluation process with minimal impact on the resulting renderings.

As outlined in Sect. 2.2, previous applications of polynomial optics for simulating lens flares ignored the per-element Fresnel transmittance and reflectance altogether and, at best, modeled the intensity using a single approximation taken at the front lens [2]. Lens manufacturers spend considerable effort on minimizing ghosts by using special, per-element anti-reflection coatings, which dramatically affect the final appearance of ghosts. To facilitate the correct simulation of arbitrary lens and anti-reflection coating materials, we

included the accumulated intensity term (τ_s) in our model, which we obtained as a product of the corresponding Fresnel transmittance and reflectance terms evaluated at every interface of the optical system. This term is not only critical for ensuring that the resulting ghosts look plausible but also plays a critical role in ensuring that the resulting ghost intensities are valid and realistic.

For large, high-complexity optical systems, lens house blocking contributes significantly to the final shape of the ghost. The *relative radius* term outlined in Sect. 3.2 is a crucial element of handling per-element lens house blocking. The relative radius encodes the maximum distance between the light ray and the optical axis that was encountered on the ray's path as it hit each optical element, normalized by the heights of the corresponding lenses. Unlike previous approaches where the relative radius and per-element lens house blocking were fully ignored, the ρ_s output variable in our model allows our approach to handle lens house blocking correctly. Note that because we use the aperture-plane coordinates to perform aperture clipping, we ignore the aperture during the calculation ρ_s , reducing the complexity of the fitting data.

Previous approaches used polygon tests to implement aperture blocking. To make the process simpler and faster, we utilized an aperture texture and hardware-accelerated texture sampling to determine if a ray can pass through the aperture. Our aperture mask stores the per-pixel signed distance from the closest aperture edge, which we sample using the aperture-plane coordinates of the ray. A consequence of our proposed approach is that it is possible to use the aperture distance directly in place of the aperture coordinates, which would reduce the number of output variables. Furthermore, the clip factor and the relative radius parameters could be aggregated in a single scalar clip factor term, reducing the number of output parameters by two and removing the need for the texture sampling process during ray tracing. In practice, however, real-world aperture shapes tend to be much more mathematically complex, and therefore, both approaches significantly increase the number of polynomial

terms needed to fit to the input data accurately, leading to worse performance despite the reduced number of input values. Finally, the dynamic runtime change of aperture shapes is also impossible with such approaches. Despite these problems, depending on the characteristics of the application, using a single clip factor term can be a viable solution for a considerable increase in rendering performance.

Because of the reasons outlined above, we used the six output variables listed in (4) during our tests. However, it is important to highlight that directly utilizing the aperture distance or a combined clip factor is a potential alternative approach that can be evaluated for any given target optical system used in practice.

4.2 Partial fitting zones

In the polynomial lens flare model described by (4), each polynomial system describes the properties of a single ghost over the entire domain of incidence angles and light wavelengths. Such an approach is a viable solution for several applications of polynomial optics (as demonstrated by the recent works outlined in Sect. 2.2) when real-time performance is not important or if the ray paths are coherent with a much more predictable behavior. However, due to the complex behavior of the accumulated relative radius and intensity terms, this representation can often lead to heavily increased numbers of polynomial terms and, in turn, drastically increases the evaluation time of the resulting polynomial systems, as indicated by recent works [2, 9, 33, 34] and our practical experiments. Furthermore, the relative radius can exhibit drastic changes near the end of its visible region (before the ghost gets fully blocked by the lens housing), which is not only difficult to properly model using a single polynomial system but also has an adverse effect on the overall accuracy of the resulting polynomial systems.

To overcome these problems, we handled the distinct light wavelengths individually and divided the input domain of incidence angles into partial local zones. A distinct polynomial system was then constructed and fit for all discrete combinations of incidence angle and wavelength. A visualization of our proposed, region-based polynomial representation is shown in Fig. 6. The number of partial regions is a user-configurable parameter that we use to sample the entire range uniformly. During polynomial evaluation, the in-between ranges are filled by the linear interpolation of the neighboring polynomial systems. Such an approach provides much better fitting characteristics in the local regions and thus reduces the number of polynomial terms needed to represent the same ghost shape. Furthermore, the incidence angle and light wavelength parameters are also constant for each unique combination, eliminating them from the inputs of the polynomial systems, leading to the following reduced

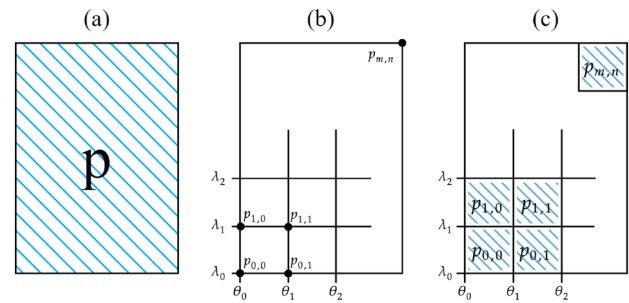


Fig. 6 Visual comparison of **a** the full polynomial model, **b** our proposed partial model, and **c** the hybrid variation of our partial

formulation:

$$(x_p, y_p, r_p, \bar{r}_p) \rightarrow (x_a, y_a, x_s, y_s, \tau_s, \rho_s), \quad (4)$$

which is significantly faster to evaluate than (4).

To reduce the amount of data needed by the individual polynomial systems and facilitate the efficient interpolation of neighboring polynomials during evaluation, we utilized a uniform system of polynomial bases for each polynomial belonging to the same lens flare ghost. Therefore, only the monomial coefficients vary with the different incidence angles and light wavelengths. Formally, each polynomial corresponding to the same ghost can be defined as:

$$p_{mn}(x) = \sum_{k=0}^t c_k^{mn} \cdot v_k, \quad v_k = \prod_{j=1}^d x_j^{l_j^k}, \quad (5)$$

where m and n are the indices of the incidence angle and light wavelength samples, v_k is the shared polynomial basis, and c_k^{mn} is the corresponding expansion coefficient. Consequently, the amount of data is significantly lowered, which greatly impacts the memory bandwidth needed for polynomial evaluation. Furthermore, this approach substantially increases the interpolation speed because the same shared basis can be evaluated only once per ray and only the monomial coefficients must be interpolated.

As a hybrid alternative solution, the local polynomial system could be constructed by considering both the current and the subsequent incidence angles, which would remove the need for interpolation and reduce the number of operations required during the evaluation. A comparison between the different representations is visualized in Fig. 6. The main drawback of this approach is that it leads to more complex polynomial systems, increasing the number of terms necessary to obtain an accurate polynomial representation. Additionally, this setup requires the use of the full polynomial system outlined in (4), which would have an adverse effect on the cost of evaluation. Our practical experiments showed that the combined effect of these factors leads to much higher

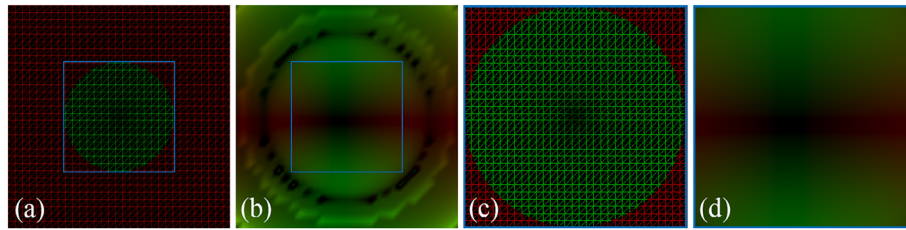


Fig. 7 **a, b** False-color visualization of rays traced through the entire front lens and an example of corresponding output variables, with the bounding region highlighted in blue. **c, d** Visualization of rays traced through only the bounding region and the corresponding target values

computational costs than our other approach outlined above. Thus, our interpolation-based approach is ideal for maximizing the performance of polynomial evaluation.

4.3 Fitting data generation

A significant drawback of the construction kit approach proposed by Hullin et al. [9] is that the per-element approximation errors are propagated through the entire system, substantially reducing the accuracy of the resulting polynomials. Additionally, the per-element polynomials can only be obtained by finding the roots of complex analytical formulas using expensive software. More recent approaches [20, 30, 31] overcome these limitations by using training datasets and devising a suitable fitting procedure to obtain the polynomials corresponding to the entire optical system. As this approach has been shown to yield better results, we also used fitting to obtain the desired polynomial systems.

To perform fitting with our proposed partial polynomial model (outlined in Sect. 4.2), we first need to construct the fitting data to obtain the exact polynomials and coefficients that approximate the lens flare ghosts of the optical system. Creating a proper data generation approach was crucial because fitting to inappropriate data can drastically reduce fitting precision. The quality of the fitting data is critical for the accuracy of the resulting polynomial system and for the number of polynomials required to reach the desired precision. Therefore, special attention is needed to construct the fitting dataset in a way that satisfies these goals. The main steps of our proposed fitting data generation approach are summarized in Algorithm 1. Conceptually, the fitting dataset generation comprises a series of steps that are performed for each unique combination of ghost, wavelength, and incidence angle; throughout the description of our method, we outline the details of generating the dataset for a single given parameter triplet.

For data generation purposes, we perform analytical ray tracing with a coarse ray grid against the physical structure of the optical system. To this end, we first define an axis-aligned bounding rectangle on the entrance pupil of the optical system, which tightly bounds the region that is used as the starting point of the rays. The purpose of this region is

to focus the dataset on rays that contribute to the visible part of the ghost; that is, a bounding rectangle of the rays that can reach the sensor without being blocked by any obstacle (lens housing, aperture), not suffering total internal reflection, and carrying a sufficient amount of energy.

To identify the bounding region, we trace a dense ray grid through the optical system in a preliminary step and determine the axis-aligned bounding rectangle of the region where such rays are present. The impact of this region on the resulting ray-traced fitting data is visualized in Fig. 7. If no valid rays can be found in the test grid, we check the neighboring incidence angles; if a valid neighbor is present, we use their bounding region. Finally, if no incidence angle in the proximity yields valid rays, we mark the incidence angle as invalid and output no bounding region. This step is essential to avoid visible artifacts at incidence angles where the ghost changes its visibility.

Given the bounding region, we proceed with the ray tracing of the final ray grid (with a user-configurable grid size) to obtain the initial fitting dataset, which provides the initial dataset. We then perform a series of postprocessing steps to obtain the final output fitting dataset. First, we determine for all rays whether it is an internal, external, or boundary element of the ray grid, which we define as all neighboring rays being valid (internal), invalid (external), or a mixture of both types (boundary). An example of ray categorization is visualized in Fig. 8. The generated data is then split into a valid and an invalid subset, with boundary rays assigned to the corresponding set based on the validity of the ray itself, disregarding its neighbors. Finally, both valid and invalid sets are subsampled using a uniform random sampling, from which the final output fitting data is created by combining the two subsampled datasets. Due to the nature of uniform sampling, the subsampled fitting dataset retains the original characteristics of the original data, ensuring that the sampling process only causes a fitting performance increase without any significant loss of information. Furthermore, constructing the fitting dataset from a distinct valid and invalid subset guarantees that both regions are accurately represented in the subsampled dataset, which is critical when the two regions have different distributions (like in the peripheral areas). Properly representing invalid regions is essential for

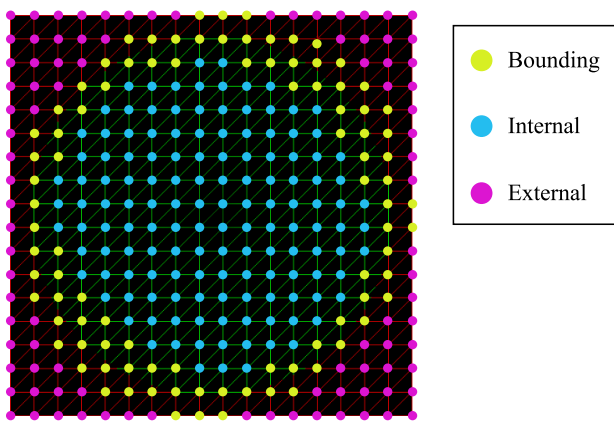


Fig. 8 Visualization of internal, external, and bounding rays

achieving adequate fitting quality and avoiding interpolation artifacts during rendering.

Algorithm 1: Our proposed method for fitting data

Input: optical system $\mathcal{O}$, light wavelengths Λ , maximum incidence angles $\tilde{\theta}$, number of incidence angles n_θ , number of bounding and sampling rays n_b, n_s , number of valid and invalid output samples n_v, n_i

Output: fitting dataset $\mathcal{D}$

$\mathcal{D} \leftarrow \{\}$
 $\mathcal{G} \leftarrow \text{EnumerateGhosts}(\mathcal{O})$
 $\Theta \leftarrow \text{SampleAngles}(n_\theta, [0, \tilde{\theta}])$
foreach $(g, \theta, \lambda) \in \mathcal{G} \times \Theta \times \Lambda$ **do**
 $R_p \leftarrow \text{GenerateRaysOnPupil}(\mathcal{O}, n_b)$
 $R_s \leftarrow \text{TraceRays}(\mathcal{O}, R_p, g, \theta, \lambda)$
 $b \leftarrow \text{CalculateValidRayBounds}(R_s)$
 $R_p^b \leftarrow \text{GenerateRaysBounded}(b, n_s)$
 $R_s^b \leftarrow \text{TraceRays}(\mathcal{O}, R_p^b, g, \theta, \lambda)$
 $\hat{\mathcal{D}} \leftarrow \text{PostProcessRays}(R_s^b)$
 $\mathcal{D}_v, \mathcal{D}_i \leftarrow \text{SplitByValidity}(\hat{\mathcal{D}})$
 $\mathcal{D}_v^s \leftarrow \text{UniformRandomSamples}(\mathcal{D}_v, n_v)$
 $\mathcal{D}_i^s \leftarrow \text{UniformRandomSamples}(\mathcal{D}_i, n_i)$
 $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_v^s \cup \mathcal{D}_i^s$
end

4.4 Polynomial fitting

In possession of the partial polynomial model and the fitting data, polynomial fitting can be performed, which produces the exact polynomials describing the ray transmission for the ghosts of the optical system. Several different methods are available in the scientific literature for this goal, with significantly different computation times and fitting precisions. From these methods, we used the approach described in [31] as a starting point due to its ability to produce accurate results in a low number of steps. We then extended this method to simultaneously determine the shared polynomial bases of the partial model and the coefficients belonging to the local zones and modified it to minimize the number of polynomial terms after reaching the desired precision.

We perform fitting for each unique combination of ghost, wavelength, and output attribute. We start the process with a polynomial comprising a single term and iteratively apply the expansion and reduction operators proposed in [31] to refine the polynomial. An essential aspect of the process is only to insert a new term into the current polynomial if the resulting reduction of error is meaningful (above a certain threshold) and the number of polynomial terms is small enough, which we both control using user-configurable parameters. The process is performed until an absolute error value is reached. Our experiments show that the impact of the error on the rendering quality is negligible under a certain threshold, whereas the additional polynomial terms increase the cost of evaluating the polynomial system. Therefore, keeping the number of polynomial terms minimal is essential for achieving adequate rendering performance.

To evaluate a possible new state generated by the operators, we apply a custom error function and an aggregation operator to determine the overall precision of the polynomial from the individual local fitting zones. The main purpose of the aggregation operator is to produce an error value that properly describes the overall behavior of the polynomials of the local zones. To this end, we calculate the error of a given state by using the mean-squared error (MSE) aggregation operator with the error function of the sub-regions:

$$\Omega(\Theta) = \frac{\sum_{\theta \in \Theta} (\omega(\theta))^2}{|\Theta|}, \quad (6)$$

where Θ is the full set of fitting datasets for each incidence angle, θ is the fitting dataset corresponding to a given incidence angle, and ω is the error function for a single local fitting zone, which we define as follows:

$$\omega(\theta) = \frac{\sum_{r \in \theta} w(r) \cdot (\hat{y}(r) - y(r))^2}{|\theta|}, \quad (7)$$

where r is a single fitting data entry, $\hat{y}(r)$ and $y(r)$ are the true (ray traced) and generated (using the polynomial system) values of the output variable for the entry r , and $w(r)$ is the weighting term for r , which we defined as follows:

$$w(r) = \begin{cases} w_r, & \text{if } y_{\min} \leq y(r) \leq y_{\max} \\ 1, & \text{otherwise} \end{cases}, \quad (8)$$

where w_r is a user-configurable parameter that depends on the validity of r , and $y_{\min}$ and $y_{\max}$ are the lowest and highest valid values for the target variable y . During our experiments, we used the following w_r weights: $w_i = 1$ for interior, $w_b = 1$ for boundary, and $w_e = 0$ for exterior rays.

The main focus of our proposed error function is to accurately fit the polynomials in the internal and bounding regions, where the results contribute to the visible parts

of the lens flare ghosts. Furthermore, our weighting scheme defined in (9) guarantees that invisible entries in the external regions only contribute to the fitting error if they would generate incorrect values during evaluation, increasing the overall robustness of the fitting process. Since the behavior of the mathematically continuous values stemming from analytical ray tracing is unpredictable and highly volatile in these regions, seeking a fully accurate fit would make the process much more complex while providing no tangible benefits. Because the ghost is invisible in these regions, the exact values of its properties are irrelevant; the only important factor is to be on the right side of the clipping threshold. Our proposed error function and weighting scheme satisfies this condition since invalid entries only contribute to the error if the corresponding generated values are inappropriate for evaluation.

4.5 Polynomial evaluation

After successfully fitting the polynomials to the optical system, we can proceed with the rendering phase, executed on the GPU. Previous polynomial optics-based lens flare rendering methods employed dense ray tracing, which is unsuitable for real-time applications. To overcome this limitation, we utilized the algorithm of Hullin et al. [22]; we only trace a coarse ray grid through the optical system and then fill in the gaps using hardware-accelerated interpolation. In our case, the key difference lies in the ray-tracing process, as rather than evaluating the lens surface elements iteratively using analytical intersection formulas, we transform the rays in one step from the entrance pupil onto the sensor using the polynomials obtained by fitting.

As outlined in Sect. 4.2, our model uses discrete horizontal incidence angles. Therefore, to handle intermediate angles, we interpolate the rotation-dependent polynomial coefficients (c_k^{mn} in (6)) using the angle of incidence of the light source generating the simulated lens flare and evaluate the polynomial model using the interpolated coefficients.

To handle axial rotation (rotation around the optical axis), the ray grid resulting from polynomial evaluation is rotated in image space with the appropriate axial rotation value. However, this process inappropriately transforms the texture coordinate space, leading to incorrectly rotated aperture shapes. To solve this problem, we apply the inverse of the axial rotation to the texture coordinates generated by the polynomial system, which cancels the rotation introduced by transforming the ray grid. The problem and our proposed solution are visualized in Fig. 9.

Lastly, efficiently performing the calculations and keeping the required operations minimal are critical for achieving optimal rendering performance on the GPU. Therefore, we took several steps to ensure that the cost of polynomial evaluation is appropriate for real-time applications. First,

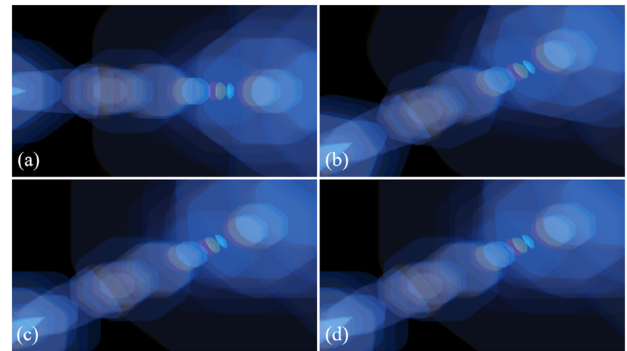


Fig. 9 Visualization of the main steps for obtaining the final ray-traced simulations. **a** Output of polynomial evaluation. **b** Results rotated around the optical axis. **c** Texture coordinates counter-rotated, providing the final output. **d** Ray-traced reference

we determine the relevant polynomial coefficients that contribute to the current frame and store them in *group shared* GPU memory. As a result, the access time of the cached polynomials is significantly reduced. Furthermore, we also applied native GPU vectorization for the parallel evaluation of the polynomial systems belonging to different output variables, leading to better utilization of the GPU resources and making the evaluation process much more efficient.

5 Results

5.1 Test setup

To evaluate our proposed method, we created a reference implementation using the C++ programming language and the OpenGL graphics API. Fitting data was generated using GPU-based analytical ray tracing, with polynomial fitting implemented on the CPU, using multithreading. Polynomial evaluation and rendering were then performed on the GPU using the tiled ghost rasterization approach of Bodonyi and Kunkli [18] to generate the final outputs from the results of the polynomial ray-tracing approach.

We also implemented direct analytical ray tracing as proposed by Hullin et al. in [22], which we used to generate the reference outputs and evaluate the performance improvements of our method. Finally, we also implemented naïve polynomial fitting [9, 34] using the polynomial system described by (4), whereby each ghost is represented via a single polynomial system, and no local fitting zones are utilized. A visual comparison of the two polynomial approaches is shown previously in Fig. 6. We note, however, that the original polynomial optics-based methods omitted the relative radius and intensity terms. However, we included these two variables when implementing the naïve approach to focus on

evaluating the improvements stemming from our partial fitting method. In reality, the difference between the previous and our proposed approaches would be even more substantial if we also ignored these two attributes in our study. However, it would make it difficult to evaluate the impact of our partial polynomial-fitting solution.

To evaluate the performance of our proposed method, we used five different optical systems with varying complexities, which we obtained from the examples of the OpenLensFlare framework [35]:

- A Heliar Tronnier lens (USP 2645156, $f/16$, 100 mm focal length, 8 refracting surfaces, 13 ghosts).
- An Angenieux double-Gauss lens (USP 2701982A, $f/11$, 100 mm focal length, 14 refracting surfaces, 42 ghosts).
- An Itoh zoom lens (USP 4196968, $f/32$, 146 mm focal length, refracting 18 surfaces, 73 ghosts).
- A Nikon tele zoom lens (S.53–JPS53131852A/22, 200 mm focal length, 21 refracting surfaces, 142 ghosts).
- A Canon tele zoom lens (USP 5537259, $f/22$, 135 mm focal length, 33 refracting surfaces, 312 ghosts).

The simulated outputs are shown in Fig. 10 for all optical systems and with each rendering algorithm.

Regarding test hardware, we used an AMD Ryzen 7 1700 3.00 GHz CPU and an NVIDIA TITAN Xp graphics card for all the measurements presented in this paper.

5.2 Model fitting performance

We first evaluate the fitting characteristics for the naïve and our proposed polynomial models. To this end, we measured the total fitting time, average number of resulting polynomial terms, and average fitting error for all output variables and optical systems. For both polynomial setups, we used three separate wavelengths and 241 incidence angle samples with a 0.5° step size ($\theta \in [0^\circ, 60^\circ]$), 80×80 sized ray grids for generating the fitting data, and selected 3,200 samples using uniform random sampling from the resulting datasets, with a 1:1 split of valid and invalid entries. The maximum number

of polynomial terms was five for the small (Heliar Tronnier, Angenieux, Itoh) and 10 for the large (Nikon, Canon) optical systems. The results are summarized in Table 2.

As can be seen from the measurements, our proposed method achieves very high fitting accuracy for all output variables and optical systems. Compared to the naïve approach, the error generated by our proposed solution is at least an order of magnitude lower for all optical systems, with substantially higher observable gains in the individual output variables. Furthermore, this improved accuracy is achieved with significantly lower numbers of polynomial terms, which is critical for achieving optimal, real-time rendering performance.

Regarding fitting performance, the total computation time scales with the complexity of the optical system, which is expected as the corresponding number of ghosts (and thus the amount of fitting to perform) grows with the optical system complexity. For our algorithm, fitting can take several minutes (Heliar Tronnier) to a few hours (Canon zoom). We believe such performance is perfectly suitable for real applications, as fitting must be performed only once, with the results being reusable in the application.

Concerning the naïve approach, the overall length of the fitting is higher with all test systems and can last almost four times longer in the most complex scenario (Canon zoom). The main reason for this behavior lies in the resulting number of terms; the polynomial regression algorithm [31] that forms the basis of our fitting procedure works by generating new candidate states via the forward and backward operators applied to the current terms present in the polynomial, which causes an exponential increase of computation as the size of the polynomial grows. Therefore, the reduction of polynomial size achieved with our proposed method is critical for rendering performance and plays an essential role in keeping the corresponding fitting times low.

5.3 Rendering accuracy

Next, we evaluated the rendering quality of the naïve and our proposed polynomial models. To this end, we compared the

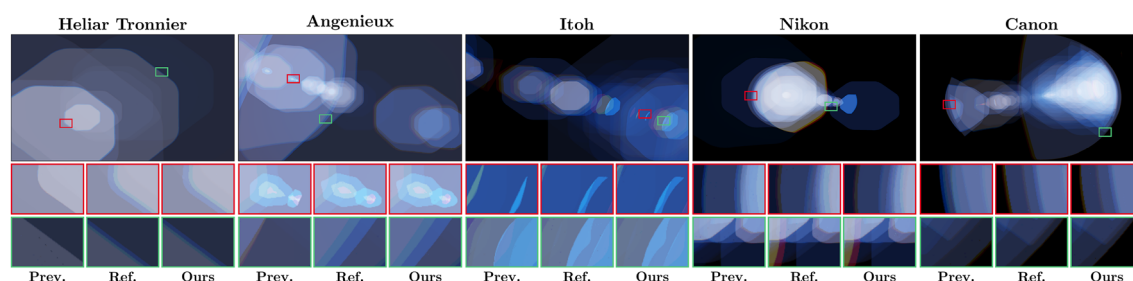


Fig. 10 Lens flare simulations for the five different optical systems tested in this paper using the full polynomial model (*prev.*), analytical ray tracing (*ref.*), and our proposed partial polynomial model (*ours*)

Table 2 Computation times, average number of polynomial terms, and average errors for all output variables and the entire optical system, using all five optical systems tested in this paper with the naïve (*prev.*) and our proposed partial (*ours*) polynomial models

Variable	Method	Heliar Tronnier			Angenieux			Itoh			Nikon			Canon		
		Comp.time	Avg. no.terms	Avg. error	Comp.time	Avg. no.terms	Avg. error	Comp.time	Avg. no.terms	Avg. error	Comp.time	Avg. no.terms	Avg. error	Comp.time	Avg. no.terms	Avg. error
x_a	Prev	0.16 h	5.00	$1.30 \cdot 10^{-1}$	0.40 h	5.00	$2.91 \cdot 10^{-2}$	0.12 h	5.00	$1.18 \cdot 10^{-2}$	3.06 h	10.00	$1.05 \cdot 10^{-2}$	4.61 h	9.95	$8.13 \cdot 10^{-3}$
	Ours	0.15 h	4.69	$7.22 \cdot 10^{-4}$	0.20 h	4.95	$3.03 \cdot 10^{-4}$	0.18 h	3.82	$4.13 \cdot 10^{-5}$	0.30 h	5.77	$1.83 \cdot 10^{-4}$	0.44 h	5.92	$7.88 \cdot 10^{-4}$
y_a	Prev	0.18 h	5.00	$2.80 \cdot 10^{-3}$	0.39 h	5.00	$1.96 \cdot 10^{-3}$	0.10 h	4.96	$1.20 \cdot 10^{-3}$	1.85 h	8.32	$1.89 \cdot 10^{-3}$	2.61 h	8.96	$1.17 \cdot 10^{-3}$
	Ours	0.07 h	4.54	$1.14 \cdot 10^{-4}$	0.09 h	4.79	$7.41 \cdot 10^{-5}$	0.01 h	3.66	$2.67 \cdot 10^{-5}$	0.14 h	4.97	$4.00 \cdot 10^{-5}$	0.21 h	4.77	$1.40 \cdot 10^{-4}$
x_s	Prev	0.14 h	5.00	$5.40 \cdot 10^{-1}$	0.38 h	5.00	$1.42 \cdot 10^{-1}$	0.13 h	5.00	$1.31 \cdot 10^{-1}$	3.21 h	10.00	$6.82 \cdot 10^{-2}$	4.26 h	10.00	$3.62 \cdot 10^{-2}$
	Ours	0.15 h	5.00	$1.50 \cdot 10^{-2}$	0.29 h	5.00	$3.95 \cdot 10^{-3}$	0.07 h	4.97	$1.83 \cdot 10^{-4}$	0.99 h	7.30	$9.64 \cdot 10^{-3}$	1.09 h	7.59	$3.73 \cdot 10^{-3}$
y_s	Prev	0.19 h	5.00	$4.80 \cdot 10^{-2}$	0.41 h	5.00	$2.11 \cdot 10^{-2}$	0.15 h	5.00	$1.35 \cdot 10^{-2}$	3.28 h	9.78	$8.76 \cdot 10^{-3}$	4.33 h	9.92	$7.65 \cdot 10^{-3}$
	Ours	0.15 h	5.00	$2.90 \cdot 10^{-3}$	0.18 h	5.00	$1.22 \cdot 10^{-3}$	0.03 h	4.93	$2.23 \cdot 10^{-5}$	0.53 h	6.36	$1.30 \cdot 10^{-3}$	0.56 h	6.48	$7.01 \cdot 10^{-4}$
τ_s	Prev	0.10 h	4.69	$6.90 \cdot 10^{-6}$	0.21 h	4.76	$6.54 \cdot 10^{-6}$	0.08 h	4.33	$1.69 \cdot 10^{-6}$	0.70 h	5.58	$1.68 \cdot 10^{-6}$	0.89 h	5.08	$1.23 \cdot 10^{-6}$
	Ours	0.11 h	3.54	$1.24 \cdot 10^{-7}$	0.16 h	2.86	$8.70 \cdot 10^{-8}$	0.02 h	1.44	$2.29 \cdot 10^{-8}$	0.12 h	1.80	$5.43 \cdot 10^{-8}$	0.21 h	1.73	$4.13 \cdot 10^{-8}$
ρ_s	Prev	0.16 h	5.00	$3.04 \cdot 10^{-2}$	0.38 h	5.00	$3.73 \cdot 10^{-2}$	0.14 h	5.00	$2.39 \cdot 10^{-2}$	3.20 h	10.00	$5.42 \cdot 10^{-2}$	4.68 h	10.00	$2.55 \cdot 10^{-2}$
	Ours	0.18 h	5.00	$4.92 \cdot 10^{-3}$	0.33 h	5.00	$4.07 \cdot 10^{-3}$	0.11 h	5.00	$5.43 \cdot 10^{-4}$	1.46 h	8.13	$9.76 \cdot 10^{-3}$	2.92 h	9.32	$2.36 \cdot 10^{-3}$
Entire sys-tem	Prev	0.97 h	4.95	$1.30 \cdot 10^{-1}$	2.29 h	4.96	$3.86 \cdot 10^{-2}$	0.77 h	4.88	$3.01 \cdot 10^{-2}$	15.54 h	8.95	$2.40 \cdot 10^{-2}$	21.73 h	8.99	$1.31 \cdot 10^{-2}$
	Ours	0.85 h	4.63	$3.95 \cdot 10^{-3}$	1.36 h	4.60	$1.60 \cdot 10^{-3}$	0.30 h	3.97	$1.36 \cdot 10^{-4}$	3.76 h	5.72	$3.49 \cdot 10^{-3}$	5.78 h	5.97	$1.29 \cdot 10^{-3}$

simulated outputs to the analytical references using the peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) metrics. The per-pixel SSIM maps and the PSNR and mean SSIM values are shown in Fig. 11.

As the results demonstrate, our proposed approach approximates the ground-truth analytical references with high accuracy. Most of the inaccuracies can be found near the ghost edges; the root cause for this behavior stems from the per-pixel clipping with the ρ_s relative radius parameter, which exaggerates any inaccuracies resulting from the polynomial approximation. We also highlighted a few areas with the biggest observable differences in Fig. 10, which clearly demonstrate that the errors are marginal, and that the simulation is fully plausible despite these errors.

Compared to the naïve approach, our proposed partial model greatly improves the simulation accuracy. As shown by the SSIM maps in Fig. 11, the errors of the full model are still mostly focused on the ghost edges. However, the inaccurately approximated clipping parameters manifest in displaced spectral channels and sometimes highly morphed ghost shapes, further exacerbated by the increased errors of the screen-space ray locations. Consequently, the simulations differ much more greatly from the analytical ground-truth references and achieve much lower levels of physical correctness. Our proposed polynomial model greatly reduces the levels of such inaccuracies.

One particularly challenging scenario is when the shape of the relative radius map is highly irregular. The root cause for this issue is that the shared polynomial basis cannot represent the rapidly changing shape of the clipping information exactly. An example of such a scenario is shown in Fig. 12 for the Nikon zoom lens, where a single erroneous ghost is highlighted with increased intensity. As demonstrated by this example, the difference is hard to notice despite the greatly increased intensity. The main reason for this is that such scenarios typically occur when a large portion of the incoming ray grid is blocked by the lens housing, rendering the corresponding ghost mostly invisible. Therefore, ghosts exhibiting such inaccuracies can be discarded without meaningfully reducing the plausibility of the simulation. Falling back to analytical ray tracing in such cases is another viable alternative to overcome this issue. Finally, using more refined polynomial input variables (such as the projection of the input ray grid to the entrance pupil) or tuning the error function to the individual optical system could also help.

Finally, rendered videos are also provided as supplementary material for a low (Heliar Tronnier), medium (Itoh), and high (Canon) complexity camera system and both polynomial solutions. These demonstrate the stability of our proposed solution in motion and compare the outputs of both models to the analytically ray-traced reference simulations.

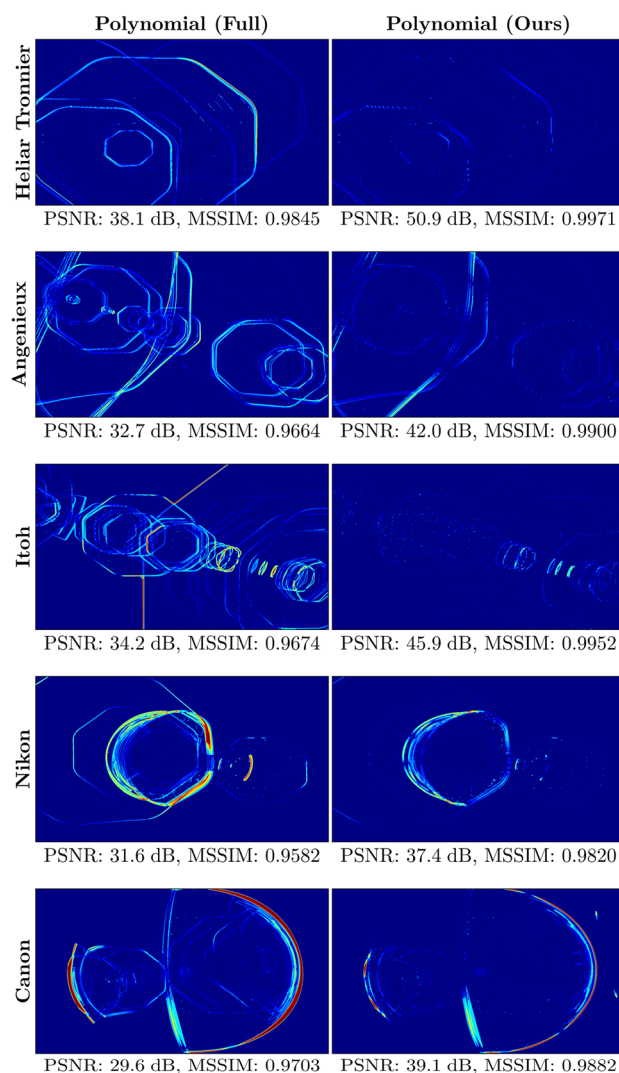


Fig. 11 Structural similarity index measure (SSIM) maps, peak signal-to-noise ratios (PSNR), and mean SSIM values (MSSIM) for the five different optical systems evaluated in this paper, using the naïve (full) and our proposed (partial) polynomial-fitting methods

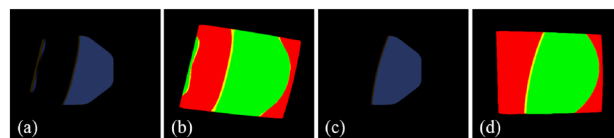


Fig. 12 Close-up of ghost #100 of the Nikon zoom lens, with a $10 \times$ intensity multiplier for better visibility. **a–b** Output simulation and false-color visualization of the corresponding relative radius generated with our partial polynomial approach. **c–d** Ground-truth reference simulations produced using analytical ray tracing

Table 3 Computation times of the ray transmission step and the total lens flare simulation for all the outputs shown in Fig. 10

	Analytical		Polynomial (prev.)		Polynomial (ours)	
	Ray trans	Total	Ray trans	Total	Ray trans	Total
Heliar Tronnier	0.75 ms	1.95 ms	0.52 ms	2.13 ms	0.32 ms	1.82 ms
Angenieux	3.32 ms	4.54 ms	1.37 ms	2.66 ms	0.89 ms	2.23 ms
Itoh	3.72 ms	4.48 ms	1.27 ms	2.21 ms	0.80 ms	1.76 ms
Nikon	10.54 ms	11.79 ms	4.35 ms	5.63 ms	2.27 ms	3.52 ms
Canon	33.65 ms	35.53 ms	8.54 ms	10.65 ms	4.51 ms	6.59 ms

Table 4 Memory requirements of the full (*prev.*) and our proposed partial (*ours*) polynomial models

	Heliar Tronnier	Angenieux	Itoh	Nikon	Canon
Prev	10.55 KB	34.18 KB	58.46 KB	208.41 KB	459.92 KB
Ours	1.00 MB	3.21 MB	4.82 MB	13.52 MB	30.99 MB

5.4 Rendering performance

With the fitting and rendering quality of our model established and validated, we proceed with the evaluation of the rendering performance of the analytical and polynomial approaches. To this end, we measured the length of the ray tracing and the full lens flare simulation processes for the outputs shown in Fig. 10. The results are summarized in Table 3.

As the measurements clearly show, our proposed method performs significantly better than all other tested solutions. Even in the simplest case (Heliar Tronnier), our partial polynomial approach is nearly twice as fast as analytical ray tracing and performs four to five times better than the brute-force solution in the most complex cases (Nikon, Canon). Compared to the naïve polynomial model, our method also shows increased performance, with considerable performance gains measured for the more complex optical systems. Critically, our proposed model also facilitates the real-time simulation of lens flares for arbitrary optical systems, which was previously impossible with analytical ray tracing or only by significantly sacrificing from the output quality with previous polynomial models, as demonstrated in Sect. 5.3.

5.5 Memory

To finish evaluating our polynomial ray transmission approach, we also measured the memory requirements of the full and our partial models for all optical systems. The results are summarized in Table 4.

As expected, compared to the naïve polynomial model, our proposed partial solution increases the memory footprint of the polynomial systems. However, the shared polynomial basis ensures that the amount of resulting data is still

well within reasonable limits. Both the offline (on disk) and online (in CPU and GPU memory) storage of our proposed model are feasible on commodity hardware. Consequently, we believe that our proposed method is perfectly suitable for real-world applications.

6 Conclusion

In this paper, we presented an efficient ray-transfer algorithm for the real-time simulation of camera lens flares. Our proposed approach works by subdividing the input space into local fitting zones and constructing a set of sparse polynomials with a common basis for each ghost generated by the optical system. Proper fitting quality and performance are achieved by carefully constructing the fitting datasets with prefiltering and subsampling, as well as by utilizing a custom error function that focuses the fitting procedure on the visible regions of the flare and ensures that incorrect external values are properly penalized. We evaluated the fitting characteristics, runtime performance, and rendering accuracy of our proposed approach using optical systems of varying complexity. Our results demonstrate that our proposed method can achieve fully real-time rendering performances that clearly outperform previous solutions while retaining the physical correctness of the simulation and faithfully replicating the outputs and physical correctness of analytical ray tracing.

Regarding future research areas, we would be mainly interested in testing the model with aspherical lens elements, as we expect our approach to provide substantially higher performance benefits compared to more complex analytical intersection formulas. We would also like to extend our model

to support dynamically changing the focal length of the optical system. While extending our proposed polynomial model and fitting procedure to include the focal length as an input variable is straightforward, altering the focal length on real-world objectives typically results in a drastic change in the internal structure of the optical system. Therefore, more work needs to be done to evaluate the performance of the straightforward solution and to optionally extend the model to be able to handle a dynamically changing optical system efficiently. Experimenting with different, more specialized polynomial input variables could also be beneficial to improve the fitting accuracy further and potentially reduce the number of terms needed to achieve the same rendering quality. We would also like to research potential optimization opportunities to speed up the performance of polynomial evaluation; we believe that polynomial systems are ideal for GPU evaluation, but the currently available tools could be more efficient and have proved to be the most limiting factor of performance throughput in our experiments. Finally, we believe it would be interesting to apply our model to other areas where the cost of analytical ray tracing is a key limitation, such as the simulation of aberrated human vision.

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1007/s00371-024-03625-7>.

Acknowledgements We gratefully acknowledge the support of NVIDIA Corporation with the donation of the TITAN Xp GPU used for this research.

Author Contributions A.B. conceived the idea. A.B. and I.Cs. designed the partial polynomial model and the corresponding fitting procedure. All the authors conducted the theoretical analysis. A.B. performed the experiments and analyzed the results. A.B. wrote the initial manuscript and prepared the figures with inputs from all authors. All authors reviewed the manuscript. R.K. supervised the project.

Funding Open access funding provided by University of Debrecen. The authors have no relevant financial or nonfinancial interests to disclose.

Data availability The datasets generated during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the

permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Pixar. The imperfect lens: Creating the look of Wall-E (2008). Wall-E Three-DVD Box
2. Pekkarinen, E., Balzer, M.: Physically based lens flare rendering in “The Lego Movie 2”. In: Proceedings of the 2019 Digital Production Symposium, pp. 1:1–1:3. Association for Computing Machinery, New York, NY, USA (2019). <https://doi.org/10.1145/3329715.3338881>
3. Ritschel, T., Ihrke, M., Frisvad, J.R., Coppers, J., Myszkowski, K., Seidel, H.-P.: Temporal glare: real-time dynamic simulation of the scattering in the human eye. *Comput. Graph. Forum* **28**(8), 183–192 (2009). <https://doi.org/10.1111/j.1467-8659.2009.01357.x>
4. Zou, G., Bai, H., Yuan, Y., Deng, T., Yin, Z., Wei, J.: Research on flare removal network based on channel attention mechanism and depthwise over-parameterized convolution. In: Proceedings of the 4th International Conference on Artificial Intelligence and Computer Engineering, pp. 919–926. Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3652628.3652781>
5. Wu, Y., He, Q., Xue, T., Garg, R., Chen, J., Veeraraghavan, A., Barron, J.T.: How to train neural networks for flare removal. In: O’Conner, L. (ed) Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 2219–2227. IEEE Computer Society, Los Alamitos, California, CA, USA (2021). <https://doi.org/10.1109/ICCV48922.2021.00224>
6. Jin, Z., Feng, H., Xu, Z., Chen, Y.: A Data Generation Method for Image Flare Removal Based on Similarity and Centrosymmetric Effect. *Photonics* **10**, 1072:1–1072:19 (2023). <https://doi.org/10.3390/photonics10101072>
7. Dai, Y., Lou, Y., Zhou, S., Li, C., Loy, C. C.: Nighttime Smartphone Reflective Flare Removal Using Optical Center Symmetry Prior. In: O’Conner, L. (ed) Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 20783–20791. IEEE Computer Society, Los Alamitos, California, CA, USA (2023). <https://doi.org/10.1109/CVPR52729.2023.01991>
8. Dai, Y., Li, C., Zhou, S., Feng, R., Loy, C. C.: Flare7K: A Phenomenological Nighttime Flare Removal Dataset. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds) Advances in Neural Information Processing Systems 35 (NeurIPS 2022), pp. 3926–3937. (2022). https://proceedings.neurips.cc/paper_files/paper/2022/hash/1909ac72220bf5016b6c93f08b66cf36-Abstract-Datasets_and_Benchmarks.html Accessed on 30 July 2024
9. Hullin, M.B., Hanika, J., Heidrich, W.: Polynomial optics: a construction kit for efficient ray-tracing of lens systems. *Comput. Graph. Forum* **31**(4), 1375–1383 (2012). <https://doi.org/10.1111/j.1467-8659.2012.03132.x>
10. Kilgard, M.J.: Fast OpenGL-rendering of Lens Flares. URL <https://www.opengl.org/archives/resources/features/KilgardTechniques/LensFlare/> (2000). Accessed 17 January 2024
11. King, Y.: 2D Lens Flare. In: DeLoura, MA. (ed) Game Programming Gems, pp. 515–518. Charles River Media, (2000)
12. Maughan, C.: Texture Masking for Faster Lens Flare. In: DeLoura, MA. (ed) Game Programming Gems 2, pp. 474–480. Charles River Media, (2001)
13. Sekulic, D.: Efficient Occlusion Culling. In: Fernando, R. (ed) GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics, pp. 487–503. Addison-Wesley, (2004)

14. Oat, C.: A Steerable Streak Filter. In: Engel, W.F. (ed) *ShaderX3*, vol. 2, pp. 341–348. Charles River Media, (2004)
15. Alspach, T.: Vector-based representation of a lens flare. US Patent 7-526-417 (2009)
16. Chaumond, J.: Realistic Camera - Lens Flares. URL <https://graphics.stanford.edu/wikis/cs348b-07/JulienChaumond/FinalProject> (2007). Accessed 17 January 2024
17. Keshmirian, A.: A physically-based approach for lens flare simulation. Master's thesis, University of California, San Diego. URL <https://escholarship.org/uc/item/5n07m4p6> (2008). Accessed 17 January 2024
18. Steinert, B., Dammertz, H., Hanika, J., Lensch, H.P.A.: General Spectral Camera Lens Simulation. *Comput. Graph. Forum* **30**(6), 1643–1654 (2011). <https://doi.org/10.1111/j.1467-8659.2011.01851.x>
19. Lendermann, M., Tan, J.S.Q., Koh, J.M., Cheong, K.H.: Computational imaging prediction of starburst-effect diffraction spikes. *Sci. Rep.* **8**, 16919 (2018). <https://doi.org/10.1038/s41598-018-34400-z>
20. Hanika, J., Dachsbacher, C.: Efficient Monte Carlo rendering with realistic lenses. *Comput. Graph. Forum* **33**(2), 323–332 (2014). <https://doi.org/10.1111/cgf.12301>
21. Walch, A., Luksch, C., Szabo, A., Steinlechner, H., Haaser, G., Schwärzler, M., et al.: Lens flare prediction based on measurements with real-time visualization. *Vis. Comput.* **34**(9), 1155–1164 (2018). <https://doi.org/10.1007/s00371-018-1552-4>
22. Hullin, M., Eisemann, E., Seidel, H.-P., Lee, S.: Physically-based real-time lens flare rendering. *ACM Trans. Graph.* **30**(4), 1081–1089 (2011). <https://doi.org/10.1145/2010324.1965003>
23. Bodonyi, A., Kunkli, R.: Efficient tile-based rendering of lens flare ghosts. *Comput. Graph.* **115**, 472–483 (2023). <https://doi.org/10.1016/j.cag.2023.07.019>
24. Lee, S., Eisemann, E.: Practical real-time lens-flare rendering. *Comput. Graph. Forum* **32**(4), 1–6 (2013). <https://doi.org/10.1111/cgf.12145>
25. Hennessy, P.: Implementation Notes: Physically Based Lens Flares. URL <https://placeholderart.wordpress.com/2015/01/19/implementation-notes-physically-based-lens-flares/> (2015). Accessed 17 January 2024
26. Kakimoto, M., Matsuoka, K., Nishita, T., Naemura, T., Harashima, H.: Glare generation based on wave optics. *Comput. Graph. Forum* **24**(2), 185–193 (2005). <https://doi.org/10.1111/j.1467-8659.2005.00842.x>
27. van den Berg, T.J.T.P., Hagenouw, M.P.J., Coppens, J.E.: The ciliary corona: physical model and simulation of the fine needles radiating from point light sources. *Invest. Ophth. Vis. Sci.* **46**(7), 2627–2632 (2005). <https://doi.org/10.1167/iovs.04-0935>
28. Joo, H., Kwon, S., Lee, S., Eisemann, E., Lee, S.: Efficient ray tracing through aspheric lenses and imperfect bokeh synthesis. *Comput. Graph. Forum* **35**(4), 99–105 (2016). <https://doi.org/10.1111/cgf.12953>
29. candolo, L., Lee, S., Eisemann, E.: Quad-based Fourier transform for efficient diffraction synthesis. *Comput. Graph. Forum* **37**(4), 167–176 (2018). <https://doi.org/10.1111/cgf.13484>
30. Schrade, E., Hanika, J., Dachsbacher, C.: Sparse high-degree polynomials for wide-angle lenses. *Comput. Graph. Forum* **35**(4), 89–97 (2016). <https://doi.org/10.1111/cgf.12952>
31. Zheng, Q., Zheng, C.: Adaptive sparse polynomial regression for camera lens simulation. *Vis. Comput.* **33**, 715–724 (2017). <https://doi.org/10.1007/s00371-017-1402-9>
32. Goossens, T., Lyu, Z., Ko, J., Wan, G.C., Farrell, J., Wandell, B.: Ray-transfer functions for camera simulation of 3D scenes with hidden lens design. *Opt. Express* **30**(13), 24031–24047 (2022). <https://doi.org/10.1364/OE.457496>
33. Dilorio, S.: Optimizations for Rendering Realistic Lens Flares in Polynomial Optics. Honors Theses, Union College, Schenectady, USA. URL <https://digitalworks.union.edu/cgi/viewcontent.cgi?article=1291&context=theses> (2015). Accessed: 29 February 2024
34. Sabatschus L.: Polyflare: Sparse Polynomial Modeling for Efficient Approximate Lens Flare Rendering. Bachelor's thesis, University of Bonn, Bonn, Germany (2022). Accessed 17 January 2024
35. Csoba, I.: OpenLensFlare: an Open-Source, Lens Flare Designing and Rendering Framework. In: Skala, V. (ed) *WSCG 2017: Short papers proceedings*, pp. 195–203. Vaclav Skala–UNION Agency (2017)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



data visualization.

Andrea Bodonyi received her M.Sc. degree in Computer Science from the University of Debrecen, Hungary in 2019. She is currently a Ph.D. student at the Doctoral School of Informatics, University of Debrecen, Hungary. She also works as an assistant lecturer at the Department of Data Science and Visualization, Faculty of Informatics, University of Debrecen. Her research interests are in computer graphics, real-time rendering based on camera-specific properties and



István Csoba received his M.Sc. degree in Computer Science from the University of Debrecen, Hungary in 2017. He is currently a Ph.D. student at the Doctoral School of Informatics, University of Debrecen, Hungary. His research interests mainly include computer graphics and real-time rendering, with a particular focus on massively parallel algorithms and the simulation of the characteristics of optical systems.



Roland Kunkli received his M.S. and Ph.D. degrees from the University of Debrecen, Hungary; he graduated as a teacher of mathematics and descriptive geometry (2008), and as a teacher of informatics (2012), then he received his Ph.D. degree in computer science in 2015. He currently works as an assistant professor at the Department of Data Science and Visualization, Faculty of Informatics, University of Debrecen, Hungary. His research interests are in computer

aided geometric design, computer graphics, and data visualization.