CHINESE OPTICS LETTERS

Vol. 21, No. 5 | May 2023

Parallel ray tracing through freeform lenses with NURBS surfaces

Haisong Tang (汤海松)[1,2], Zexin Feng (冯泽心)[1,2*], Dewen Cheng (程德文)[1,2], and Yongtian Wang (王涌天)[1,2]

[1] Beijing Engineering Research Center of Mixed Reality and Advanced Display, School of Optics and Photonics, Beijing Institute of Technology, Beijing 100081, China

[2] MOE Key Laboratory of Optoelectronic Imaging Technology and Systems, Beijing Institute of Technology, Beijing 100081, China

*Corresponding author: fzx84@126.com

Received December 19, 2022 | Accepted March 7, 2023 | Posted Online May 4, 2023

We implement Monte Carlo-based parallel ray tracing to achieve quick irradiance evaluation for freeform lenses with non-uniform rational B-splines (NURBS) surfaces. We employ the inverse transform sampling method to sample rays uniformly from the Lambertian light source and adopt the analytical form of the B-spline basis function to achieve fast surface interpolation. When performing parallel calculations for the intersections between the rays and the NURBS surfaces, we propose a parameter transformation method to avoid the parameters escaping from the defined range in the iteration process. Simulation results of two complex picture-generating freeform lenses show that our method is fast and effective.

Keywords: propagation methods; computation methods; nonimaging optics; lenses.
DOI: 10.3788/COL202321.052201

1. Introduction

Freeform optics, which refers to refractive or reflective optical elements with freeform surfaces, is considered a revolution in modern optics[1]. Freeform surfaces are those surfaces that do not have rotational or translational symmetries[2]. Compared with traditional optical surfaces, including spherical, aspherical, and cylindrical surfaces, freeform surfaces offer much greater design freedom, allowing optical systems to achieve previously unimaginable imaging[3–5], illumination[6,7], and laser beam shaping performances[8]. The performance evaluation of these freeform optical systems is highly dependent on the light transport simulation from source to target. An efficient light transport computation is crucial in finding the optimal freeform optical systems through optimization techniques.

Ray tracing is currently the most popular and powerful technique for light transport simulation. The main operation of ray tracing is to acquire the intersection points between light rays and surfaces. The computation efficiency of this process is strongly influenced by the surface complexities. Freeform surfaces for imaging can be conveniently described by polynomials, e.g., XY polynomials[3]. However, plenty of illumination problems, e.g., picture-generating freeform lens design[6], are solved numerically. The resulting freeform surfaces, which contain many local curvature features, are hard to describe by using polynomial expansions. Time and memory consumption of ray tracing could increase rapidly as the surface representation becomes more and more complex. Generally, iterative methods have to be employed to obtain the intersection points, which heavily rely on the initial solutions. The meshing method is commonly used to find appropriate initial solutions[9–11]. In this method, the grids are checked one by one to judge if the intersection occurs on the current grid. However, this method becomes less efficient when the required number of mesh grids is large.

Optical surfaces that are highly freeform and contain both global and local structures can be represented by B-splines, especially by its subset, non-uniform rational B-splines (NURBS)[12,13]. However, it is not easy to determine the intersection points, even their initial approximations, between the rays and the NURBS surface with high efficiency. The clipping method can be employed to accelerate the ray-tracing process of NURBS surfaces[14,15]. This method adaptively divides the surface parameters into uniform parts on the 2D parameter space to find the region where the light ray intersects with the surface. Then, the region is gradually subdivided to determine an appropriate initial solution. However, this method suffers from reporting wrong intersections because the sampling points on the surface are not uniformly distributed when parameters are uniformly sampled[15,16].

Aside from the intersection problem, another important issue is the way of implementing ray tracing. Although some deterministic methods have been discussed, it is restricted in many applications[17]. In the evaluation and optimization process, the Monte Carlo (MC) ray-tracing method gains wider acceptance as the gold standard for calculating light transport through

optical systems[18–20]. MC ray tracing randomly samples light rays on a light source with a specific probability and traces the light path through the optical system by Snell's laws. Then, it accumulates the light energy transmitted to the receiver to calculate the irradiance distribution.

Some commercial softwares can perform MC ray tracing of freeform optical elements and systems, but the related algorithms are rarely discussed publicly. Many self-developed ray-tracing programs are successfully applied to artificial image simulation[21–27], aberration analysis[28–32], and optical inspection[33–36]. Some entire ray-tracing pipelines for nonimaging systems have been described in detail[11,20]. Most of the current MC ray-tracing algorithms trace rays one by one, based on the same law and formula.

The ray-tracing process could be accelerated if plenty of rays are traced simultaneously. However, it is a high-dimensional problem that the parallel intersection process requires simultaneous acquisition of the initial solutions of all current rays. The traditional way of obtaining an initial solution to the intersection problem of a single ray and the freeform surface requires simultaneous computation on mesh grids, which is not suitable for parallel ray tracing. In addition, when the initial solution is not accurate enough, the parameters of the intersection could easily escape from the defined range as the iteration progresses, resulting in many calculation errors.

We implement MC parallel ray tracing (PRT) for freeform illumination lenses with NURBS surfaces to realize a fast irradiance evaluation. We achieve uniform sampling by inverse transform sampling and fast surface interpolation by adopting the analytical form of the B-spline basis function. We propose a parameter transform method to reduce the accuracy requirement of the initial solution during the intersection process using Newton's method, which could improve the percentage of successfully traced light rays by parallel computation. This fast MC simulation program can facilitate the evaluation and optimization of freeform lenses. Section 2 describes in detail the parallel ray-tracing method for freeform NURBS surfaces. Section 3 gives two examples to demonstrate the effectiveness of the proposed method.

2. Methods

The purpose of the ray-tracing process is to trace the changes in the spatial parameters (x, y) and orientation parameters (r_x, r_y), which uniquely characterize a light ray in a three-dimensional space. (x, y, r_x, r_y) form a four-dimensional parameter space, which degenerates to a two-dimensional one for a special case when all the rays are emitted from a point light source. Figure 1 illustrates the procedures of our parallel ray-tracing algorithm. After defining the optical system parameters, we sample the light source randomly in both spatial locations and orientations, generating a random set of rays. We then implement MC ray propagation from the source, through the freeform lenses, to the receiver. The difficulty arises in simultaneously determining all the parameters (x, y, r_x, r_y) for the current set of rays transporting through a complex freeform surface. In this decisive and important step, we propose a parameter transformation method to reduce the ray-surface intersection errors in the iterative process of Newton's method. Finally, we count the number of rays that reach different positions of the receiver to determine the irradiance distribution. The three steps are described in detail in the following section.

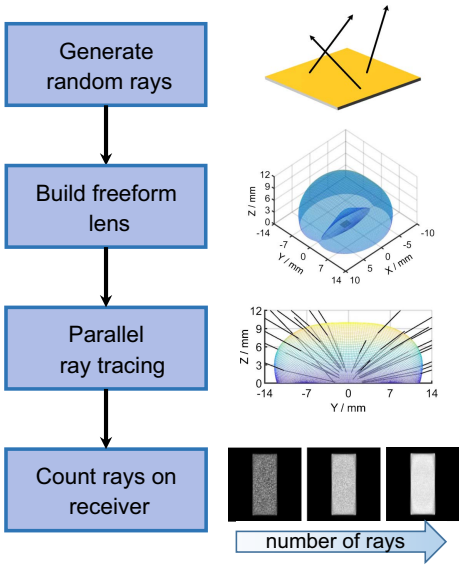

Fig. 1. Flow chart of our parallel ray-tracing algorithm.

2.1. Sampling rays from the light source

The light source is required to be sampled randomly in both spatial locations and orientations. Different sampling probability density functions $P(x, y, r_x, r_y)$ should ensure that $w(x, y, r_x, r_y) \cdot P(x, y, r_x, r_y)$ is proportional to $L(x, y, r_x, r_y) \cdot \cos\theta$, where L is the radiance distribution of the source, w denotes the weights of the sampled rays, and θ denotes the angles from the source surface normal to the sampled rays. The initial values of w can be set as 1, and then P is proportional to the intensity distribution of the light source. However, this can add computation complexity when L is not a constant.

Here, we first sample the position coordinates and solid angle of the light source uniformly and then assign the initial weights proportional to $L \cdot \cos\theta$. In this way, there is an equivalent relation between the radiance distribution L and the weights of the sampled light rays,

$$L(x, y, r_x, r_y) \cdot \cos\theta \delta S \delta\Omega \propto E_\mu\left(\sum_{i=0}^{k} w_i\right). \tag{1}$$

Herein, δS denotes a small area on the surface around (x, y), and $\delta\Omega$ denotes a small solid angle around (x, y, r_x, r_y). $E_\mu(\sum_{i=0}^{k} w_i)$ is the mathematical expectation of the summed weight of the light rays inside the small space $\delta S \delta\Omega$, and k denotes the number of rays inside the small space.

In a spherical coordinate system, (r_x, r_y) can be replaced with the zenith angle θ and the azimuth angle φ, as shown in Fig. 2(a). Note that the light rays sampled uniformly with θ and φ are not uniformly distributed in space[19]. Since the magnitude of the small solid angle $\delta\Omega$ around (θ, φ) is proportional to $\sin\theta\delta\theta\delta\varphi$, uniform sampling of the solid angle should guarantee that the probability is proportional to φ and $\sin\theta$.

We employ the inverse transform sampling method[37] to obtain random variables with arbitrary probability distribution $f(x)$, where $x \in [0, 1]$. This method first calculates the cumulative probability distribution function $F(x) = \int_{-\infty}^{\infty} f(x)\mathrm{d}x$ according to the required probability distribution function $f(x)$. A uniformly distributed random variable $Y = F(x)$ can be generated. Finally, the random variable X with probability distribution $f(X)$ can be obtained as $X = F^{-1}(x)$. For the sampling of the zenith angle here, we first obtain a random uniformly distributed variable Y between 0 and 1; then the probability distribution function $X = \arccos(1 - Y)$ is a sine function. Thus, uniform sampling of random rays in space can be generated based on setting the probability distribution functions as $\theta \sim \sin\theta$, $\varphi \sim U(0, 2\pi)$.

Figures 2(b) and 2(c) illustrate the ray distributions of a Lambertian light source sampled in two strategies: $\theta \sim \sin\theta$ and $\theta \sim U(0, 0.5\pi)$, where φ is uniformly sampled for each strategy. The pseudo-code of the random rays sampling process is shown in Algorithm 1.

Algorithm 1. Random rays sampling (RRS) process. The function RAND( ) denotes the generation of a uniformly distributed random number in [0,1].

Input: length and width of the rectangular light source (a, b)

Output: starting points $\hat{\mathbf{s}}$, ray directions $\hat{\mathbf{r}}$, weights w

```
function RRS(a, b)
    σ1 ← RAND(), σ2 ← RAND(), σ3 ← RAND(), σ4 ← RAND()
    x0 ← (σ1 − 0.5)a, y0 ← (σ2 − 0.5)b, z0 ← 0
    θ ← arccos(1 − σ3), φ ← 2πσ4
    rx ← sin θ cos φ, ry ← sin θ sin φ, rz ← cos θ
    ŝ ← (x0, y0, z0), r̂ ← (rx, ry, rz)
    w ← cos θ
    return ŝ, r̂, w
end function
```

2.2. Light propagation through freeform optical surfaces

After generating a sequence of random rays from the light source, we trace the rays propagating through the freeform optical surfaces represented with NURBS. The key operation is the determination of the intersection points between rays and surfaces. Thus, we accelerate the intersection process by modifying the surface interpolation strategy and introducing a parameter transformation method. After obtaining the intersection points, we use Snell's law in vector form to calculate the outgoing ray vectors.

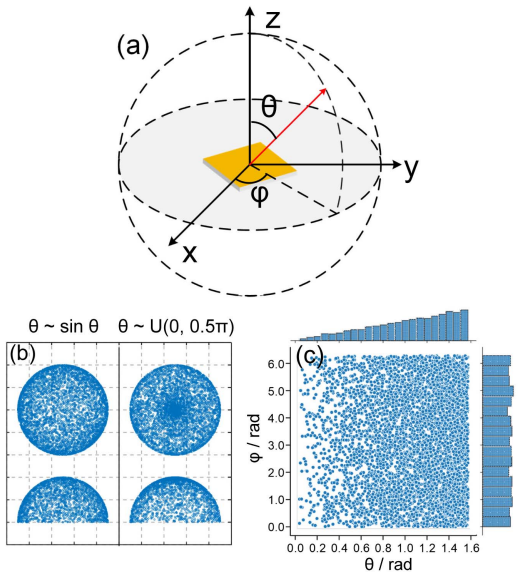

Fig. 2. (a) Zenith angle θ and azimuth angle φ of a randomly sampled ray. (b) The illustration of two different sampling strategies, $\theta \sim \sin\theta$ and $\theta \sim U(0, 0.5\pi)$, where φ is uniformly sampled. (c) The distribution of the sampling points with azimuth and zenith angles corresponding to the uniform spatial sampling.

2.2.1. Intersection acquisition on optical surface

An NURBS surface is obtained as the tensor product of two NURBS curves parameterized by two parameters u and v,

$$\mathbf{Q}(u, v) = \sum_{i=0}^{m_s-1} \sum_{j=0}^{n_s-1} N_{i,p}(u) N_{j,q}(v) R_{i,j} \mathbf{P}(i, j), \tag{2}$$

where u and v lie in $[0, 1]$, $\mathbf{P}(i, j)$ denotes the control point, $R_{i,j}$ is the weight of $\mathbf{P}(i, j)$, and $N_{i,p}(u)$ and $N_{j,q}(v)$ are the basis functions of the B-splines of degree p and q in the u and v directions, respectively. The basis function $N_{i,p}(u)$ is defined by the De Boor Cox recursion formula[12],

$$\begin{cases} N_{i,0} = \begin{cases} 1 & \text{if } u \in [u_i, u_{i+1}) \\ 0 & \text{if else} \end{cases}, \\ N_{i,p} = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u), \\ \text{define}: \frac{0}{0} = 0. \end{cases} \tag{3}$$

Herein, u_i is the selected node which divides $[0, 1]$ into $m_s + p$ segments. We employ the quasi-uniform nodes here, which can simplify the formula for the B-spline basis functions.

The computation volume required for interpolation increases exponentially with the number of control points and the interpolation degree. The major reason is that the recursive method repeatedly computes the formula with a large storage of accumulative data and no zero $N(u_i)$ only when $u_i \in [u_i, u_{i+p+1}]$. These unnecessary calculations slow down the intersection point-finding process. Abert *et al.* transferred the computationally expensive recursion into single instruction multiple-data (SIMD) suitable form to reduce the time cost of executed commands[13]. Jester *et al.* proposed the B-spline quasi-interpolation scheme, which improves the computational speed of B-spline surfaces within a certain error range[38]. To improve the speed of the interpolation process, we employ the analytical form of the B-spline basis function of degree 2[13]. In addition, only the control points whose corresponding basis functions are not 0 are considered in the interpolation process to avoid meaningless operations. Such a strategy can promote calculation efficiency greatly when the control point number is much larger than the interpolation order.

After specifying the interpolation strategy, let us now turn to the calculation of the intersection between the rays and the NURBS surface. Suppose we have a ray parameterized as $(x(t), y(t), z(t)) = \hat{\mathbf{s}} + \hat{\mathbf{r}} \cdot t$, where $\hat{\mathbf{s}} = (x_0, y_0, z_0)$ is the starting point, and $\hat{\mathbf{r}} = (r_x, r_y, r_z)$ denotes the ray direction. The intersection can be formulated by the following equation:

$$\mathbf{Q}(u, v) = \hat{\mathbf{s}} + \hat{\mathbf{r}} \cdot t, \tag{4}$$

where $\mathbf{Q} = (Q_1, Q_2, Q_3)$ denotes a point on the NURBS surface. After expressing the ray direction with θ and φ, Eq. (4) can be rewritten as

$$\begin{cases} Q_1(u, v) - x_0 = \sin\theta \cos\varphi \cdot t \\ Q_2(u, v) - y_0 = \sin\theta \sin\varphi \cdot t \\ Q_3(u, v) - z_0 = \cos\theta \cdot t \end{cases}. \tag{5}$$

We rotate φ and θ around the z-axis and y-axis, respectively, so that the new z-axis coincides with the direction of the light ray. This way can simplify the intersection-solving process[39]. The transformation matrices describing the rotations are

$$\mathbf{R}_z = \begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R}_y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{6}$$

where $\mathbf{R}_z$ describes the rotation around the z-axis, and $\mathbf{R}_y$ describes the rotation around the y-axis.

After applying the rotation matrices to Eq. (5), we obtain

$$\begin{aligned} &\mathbf{R}_y \cdot \mathbf{R}_z \cdot \begin{bmatrix} Q_x(u, v) - x_0 \\ Q_y(u, v) - y_0 \\ Q_z(u, v) - z_0 \end{bmatrix} \\ &= \mathbf{R}_y \cdot \mathbf{R}_z \cdot \begin{bmatrix} \sin\theta \cdot \cos\varphi \cdot t \\ \sin\theta \cdot \sin\varphi \cdot t \\ \cos\theta \cdot t \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ t \end{bmatrix}. \end{aligned} \tag{7}$$

We take the first two equations, which could be represented as $f_1(u, v) = 0$ and $f_2(u, v) = 0$. Then, the parameters (u, v) of the intersection point on the surface can be solved by using Newton's method. Newton's iteration step is given by

$$\begin{bmatrix} u_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} u_k \\ v_k \end{bmatrix} - \mathbf{J}^{-1} \begin{bmatrix} f_1(u_k, v_k) \\ f_2(u_k, v_k) \end{bmatrix}, \tag{8}$$

where (u_k, v_k) is the approximate solution of the kth iteration, and $\mathbf{J}$ is the Jacobian matrix,

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial u} & \frac{\partial f_1}{\partial v} \\ \frac{\partial f_2}{\partial u} & \frac{\partial f_2}{\partial v} \end{bmatrix}. \tag{9}$$

Newton's method needs a good initial estimate. Most of the methods divide the surface into many triangular meshes to find a suitable initial solution. If the meshing is sufficiently detailed, then the initial solution could lead to a good iteration result. Such a strategy is not suitable for parallel computation here.

A slight deviation from the exact initial solution could cause the value of u or v to exceed the range of [0,1], resulting in surface interpolation errors. We adopt a parameter transformation method to solve this problem. In this method, a new pair of parameters, (α, β), are employed to replace (u, v), and their relationships can be described as

$$u = \frac{1}{e^{-4\alpha} + 1}, \qquad v = \frac{1}{e^{-4\beta} + 1}. \tag{10}$$

The parameter transformation has the following characteristics:

$$\begin{cases} \alpha, \beta \in (-\infty, +\infty), u, v \in (0, 1) \\ \frac{\mathrm{d}u}{\mathrm{d}\alpha} > 0, \frac{\mathrm{d}v}{\mathrm{d}\beta} > 0, \frac{\mathrm{d}u}{\mathrm{d}\alpha}\big|_{\alpha=0} = 1, \frac{\mathrm{d}v}{\mathrm{d}\beta}\big|_{\beta=0} = 1 \\ u|_{\alpha=0} = 0.5, v|_{\beta=0} = 0.5 \end{cases}. \tag{11}$$

Such a transformation can avoid the values of (u, v) exceeding [0, 1]. The intersection equations now become $f_1(\alpha, \beta) = 0$ and $f_2(\alpha, \beta) = 0$. The new Jacobian matrix $\mathbf{J}$ in Newton's method can be acquired based on the chain rule

$$\begin{cases} \frac{\partial f}{\partial \alpha} & = & \frac{\partial f}{\partial u} \cdot \frac{\mathrm{d}u}{\mathrm{d}\alpha} \\ \frac{\partial f}{\partial \beta} & = & \frac{\partial f}{\partial v} \cdot \frac{\mathrm{d}v}{\mathrm{d}\beta} \end{cases}. \tag{12}$$

Herein, f can be f_1 or f_2. We obtain the initial solution using a coarse mesh on the surface. For the kth iteration, (α_k, β_k) can be calculated from (u_k, v_k) by the inverse operation of Eq. (10). The Newton iteration formula Eq. (8) is changed into

$$\begin{bmatrix} \alpha_{k+1} \\ \beta_{k+1} \end{bmatrix} = \begin{bmatrix} \alpha_k \\ \beta_k \end{bmatrix} - \mathbf{J}^{-1} \begin{bmatrix} f_1(\alpha_k, \beta_k) \\ f_2(\alpha_k, \beta_k) \end{bmatrix}. \tag{13}$$

Such a parameter transformation could reduce the accuracy requirement of the initial solution to the iterative computation. Thus, we can adopt surface points with a relatively coarse density as the initial solution, which can save a lot of time.

2.2.2. Refraction on the optical surface

Once we obtain the intersection points between the rays and the surface, we determine the refraction directions of the outgoing rays based on Snell's law. As illustrated in Fig. 3, $\hat{\mathbf{i}}$ denotes the unit incident ray vectors, $\hat{\mathbf{t}}$ denotes the unit outgoing ray vectors, $\hat{\mathbf{n}}$ denotes the unit normal vectors to the intersection points on the surface, and n_1 and n_2 denote the refractive indices of the incident and exit media. Then, $\hat{\mathbf{t}}$ can be obtained based on Snell's law in vector form,

$$\hat{\mathbf{t}} = \frac{1}{n_2} \left\{ n_1 \hat{\mathbf{i}} + \left[\sqrt{n_2^2 - n_1^2 + n_1^2 (\hat{\mathbf{n}} \cdot \hat{\mathbf{i}})^2} - n_1 \hat{\mathbf{n}} \cdot \hat{\mathbf{i}} \right] \hat{\mathbf{n}} \right\}. \tag{14}$$

Fresnel loss occurs when a light ray passes through a refractive optical surface. The reflectivities r_s and r_p for s-polarized light and p-polarized light, respectively, are written as

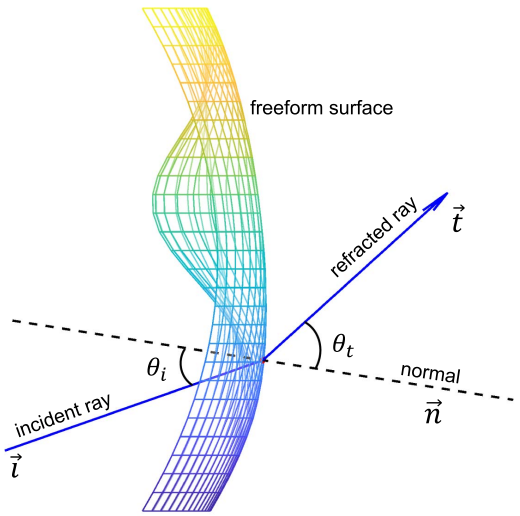

Fig. 3. Refraction on freeform surface.

$$r_s = \frac{n_1 \cos\theta_i - n_2 \cos\theta_t}{n_1 \cos\theta_i + n_2 \cos\theta_t}, \qquad r_p = \frac{n_1 \cos\theta_t - n_2 \cos\theta_i}{n_1 \cos\theta_t + n_2 \cos\theta_i}, \tag{15}$$

where θ_i and θ_t denote the incident and refracted angles, respectively. The reflectances R_s and R_p for s-polarized light and p-polarized light, respectively, can be obtained as $R_s = r_s^2$ and $R_p = r_p^2$. The transmittances T_s and T_p for s-polarized light and p-polarized light, respectively, can be simply obtained as $T_s = 1 - R_s$ and $T_p = 1 - R_p$. For natural light or typical LED light, the total transmittance T can be taken as the average of T_s and T_p. We adjust the weights $w(x, y, r_x, r_y)$ of the light rays according to the transmittance T during ray tracing.

The pseudo-code for simulating the light transport through a freeform surface is provided in Algorithm 2. Herein, we consider ray tracing as successful when the cosine distance $d = 1 - \mathrm{dot}(\mathbf{Q} - \hat{\mathbf{s}}, \hat{\mathbf{r}}) / \|\mathbf{Q} - \hat{\mathbf{s}}\|_2$ is less than the allowed error ε, where $\mathbf{Q}$ is the found intersection point.

Algorithm 2. Light transport simulation through a freeform NURBS surface. During the parallel ray-tracing process, we specify a fixed number of iterations in the INTER( ) function and record the number of rays that satisfy $d_k \le \varepsilon$.

Input: incident ray parameters $\hat{\mathbf{s}}, \hat{\mathbf{r}}, w$, surface control points $\mathbf{P}$, refractive indices n_1, n_2

Output: outgoing ray parameters $\hat{\mathbf{s}}', \hat{\mathbf{r}}', w'$

function TRACE($\hat{\mathbf{s}}, \hat{\mathbf{r}}, w, \mathbf{P}, n_1, n_2$)
  $u, v \leftarrow$ INTER($\hat{\mathbf{s}}, \hat{\mathbf{r}}, \mathbf{P}$), $\hat{\mathbf{i}} \leftarrow \hat{\mathbf{r}}$
  $\mathbf{Q}, \partial\mathbf{Q}/\partial u, \partial\mathbf{Q}/\partial v \leftarrow$ NURBS($u, v, \mathbf{P}$)
  get surface normal vectors $\hat{\mathbf{n}}$
  get refraction directions $\hat{\mathbf{t}}$ by Eq. (14)
  get reflectances R_s and R_p based on Eq. (15)
  $T \leftarrow 1 - (R_s + R_p)/2$
  $\hat{\mathbf{s}}' \leftarrow \mathbf{Q}, \hat{\mathbf{r}}' \leftarrow \hat{\mathbf{t}}, w' \leftarrow T \cdot w$
  return $\hat{\mathbf{s}}', \hat{\mathbf{r}}', w'$
end function
function INTER($\hat{\mathbf{s}}, \hat{\mathbf{r}}, \mathbf{P}$) ▷ intersection acquisition
  generate rough mesh grids $\mathbf{u}_0, \mathbf{v}_0 \in [0, 1]$
  $\mathbf{A}, \partial\mathbf{A}/\partial\mathbf{u}, \partial\mathbf{A}/\partial\mathbf{v} \leftarrow$ NURBS($\mathbf{u}_0, \mathbf{v}_0, \mathbf{P}$)
  $\mathbf{d} \leftarrow 1 - \mathrm{dot}(\mathbf{A} - \hat{\mathbf{s}}, \hat{\mathbf{r}}) / \|\mathbf{A} - \hat{\mathbf{s}}\|_2$
  $d_{\min} = \min(\mathbf{d})$
  $i \leftarrow \mathrm{index}(\mathbf{d}, d_{\min})$
  $u_k \leftarrow \mathbf{u}_0(i), v_k \leftarrow \mathbf{v}_0(i), d_k = d_{\min}$ ▷ initial solution

$\alpha_k \leftarrow -\ln(1/u_k - 1)/4, \beta_k \leftarrow -\ln(1/v_k - 1)/4$

while $d_k > \varepsilon$ **do** ▷ ε is the allowed error

$\mathbf{Q}, \partial\mathbf{Q}/\partial u, \partial\mathbf{Q}/\partial v \leftarrow \text{NURBS}(u_k, v_k, \mathbf{P})$

$d_k \leftarrow 1 - \text{dot}(\mathbf{Q} - \hat{\mathbf{s}}, \hat{\mathbf{r}})/\|\mathbf{Q} - \hat{\mathbf{s}}\|_2$

get Jacobian matrix $\mathbf{J}$ by Eqs. (9) and (12)

get $\alpha_{k+1}, \beta_{k+1}$ by Eq. (13)

$\alpha_k \leftarrow \alpha_{k+1}, \beta_k \leftarrow \beta_{k+1}$

$u_k \leftarrow 1/(e^{-4\alpha_k} + 1), v_k \leftarrow 1/(e^{-4\beta_k} + 1)$

end while

return u_k, v_k

end function

function NURBS($\mathbf{u}, \mathbf{v}, \mathbf{P}$) ▷ surface interpolation

generate quasi-uniform node vectors $(u_1, u_2, \ldots, u_i, \ldots)$ and $(v_1, v_2, \ldots, v_j, \ldots)$ using shape ($\mathbf{P}$)

get $N_{i,p}(u), N_{j,q}(v)$ based on Eq. (3)

get interpolation points $\mathbf{Q}$ and the partial derivatives $\partial\mathbf{Q}/\partial u, \partial\mathbf{Q}/\partial v$, while $R_{ij} = 1$ based on Eq. (2)

return $\mathbf{Q}, \partial\mathbf{Q}/\partial u, \partial\mathbf{Q}/\partial v$

end function

After finishing the ray tracing through the entrance surface, the ray tracing through the second surface can be implemented based on the same procedure, where the intersection points on the entrance surface are set as the starting points of the next ray tracing. Such a process is repeated until the light rays hit the receiver.

2.3. Energy statistics on the receiver

After finishing the ray tracing through all the freeform surfaces and determining the intersection points between the rays and the target plane, we count the number of rays for each small grid of the discretized receiver.

Suppose that the total radiant power emitted from the light source is W_0. The weight of the ith ray is denoted as $w_{0,i}$, and the number of rays is n. The radiant power of a ray with the unit weight can be acquired by

$$W_{\text{unit}} = \frac{W_0}{\sum_{i=1}^{n} w_{0,i}}. \tag{16}$$

Suppose that the range of the receiver is $\{(x, y)|x_{\min} < x < x_{\max}, y_{\min} < y < y_{\max}\}$, which is discretized into $m_c \times n_c$ cells. The area of each cell can be obtained as

$$S_{\text{cell}} = \frac{(x_{\max} - x_{\min}) \cdot (y_{\max} - y_{\min})}{m_c \cdot n_c}. \tag{17}$$

The energy weight of the ith light ray that hits the receiver is changed into $w_{i'}$. The total number of rays of the jth cell is n_j. The irradiance value of the jth cell, $j = 1, 2, \ldots, m_c \times n_c$, can be acquired by

$$E_j = \frac{W_{\text{unit}} \times \sum_{i=1}^{n_j} w_{i'}}{S_{\text{cell}}}. \tag{18}$$

The pseudo-code of the energy statistic process and the arithmetic flow of our PRT are shown in Algorithm 3. Note that the more the total traced rays, the higher the signal-to-noise ratio (SNR) of the simulated irradiance distribution. A discussion of the appropriate number of rays to be traced for a certain setting can be found in Ref. [40].

Algorithm 3. Pseudo-code for PRT. We use the GPU to trace batches of light rays simultaneously. The function $\text{ZEROS}(m_c, n_c)$ means generating an $m_c \times n_c$ zeros matrix.

Input: light source size (a, b), total energy W_0, number of rays n, control points $(\mathbf{P}_1, \mathbf{P}_2)$ of the two surfaces, refractive indices (n_1, n_2) of air and lens, z-coordinate z_r of receiver, range $\{[x_{\min}, x_{\max}], [y_{\min}, y_{\max}]\}$ of the receiver, number of cells (m_c, n_c)

Output: the irradiance distribution on the receiver E

$\mathbf{x}_r \leftarrow \mathbf{y}_r \leftarrow \mathbf{w}_r \leftarrow \text{ZEROS}(n), w_s \leftarrow 0$

for $i = 1$ to n **do**

$\hat{\mathbf{s}}_0, \hat{\mathbf{r}}_0, w_0 \leftarrow \text{RRS}(a, b)$

$w_s \leftarrow w_s + w_0$

$\hat{\mathbf{s}}_1, \hat{\mathbf{r}}_1, w_1 \leftarrow \text{TRACE}(\hat{\mathbf{s}}_0, \hat{\mathbf{r}}_0, w_0, \mathbf{P}_1, n_1, n_2)$

$\hat{\mathbf{s}}_2, \hat{\mathbf{r}}_2, w_2 \leftarrow \text{TRACE}(\hat{\mathbf{s}}_1, \hat{\mathbf{r}}_1, w_1, \mathbf{P}_2, n_2, n_1)$

$(x, y, z) \leftarrow \hat{\mathbf{s}}_2, (r_x, r_y, r_z) \leftarrow \hat{\mathbf{r}}_2$ ▷ decomposition

$t \leftarrow (z_r - z)/r_z$

$\mathbf{x}_r(i) \leftarrow x + t \cdot r_x, \mathbf{y}_r(i) \leftarrow y + t \cdot r_y, \mathbf{w}_r(i) \leftarrow w_2$

end for

$\mathbf{w}_r \leftarrow (W_0/w_s) \cdot \mathbf{w}_r$

$E \leftarrow \text{STATIC}(\mathbf{x}_r, \mathbf{y}_r, \mathbf{w}_r, m_c, n_c, x_{\min}, x_{\max}, y_{\min}, y_{\max})$

function STATIC($\mathbf{x}, \mathbf{y}, \mathbf{w}, m_c, n_c, x_{\min}, x_{\max}, y_{\min}, y_{\max}$)

$E \leftarrow \text{ZEROS}(m_c, n_c)$

for $i = 1$ to m_c **do**

for $j = 1$ to n_c **do**

$S_{\text{cell}} = (x_{\max} - x_{\min}) \cdot (y_{\max} - y_{\min})/(m_c \cdot n_c)$

```
        w_ij ← sum w of the rays on the ijth cell
        E_ij ← w_ij / S_cell
      end for
    end for
    return E
  end function
```

3. Results

To demonstrate the effectiveness of our method, we implement irradiance evaluation of the picture-generating freeform lenses designed with the iterative wavefront tailoring (IWT) method[41]. As shown in Fig. 4, the light rays emitted from a light source propagate through the entrance and exit optical surfaces and then generate a complex irradiance distribution of the picture type at the receiver. The entrance surface can be a spherical, aspherical, or freeform surface, and the exit surface is a freeform one. Note that the freeform surface profiles of the picture-generating freeform lenses can be very complex, which is difficult to be described by conventional polynomials, e.g., XY or Zernike polynomials.

The ray tracing performances are evaluated by the time consumption, the intersection precision, and the success rate. The intersection precision is measured by the maximum or average value of the Euclidean distances from the calculated intersection points on the exit surface to the light rays. The success rate τ is defined as the ratio of the number of successfully traced rays to the total number of traced rays.

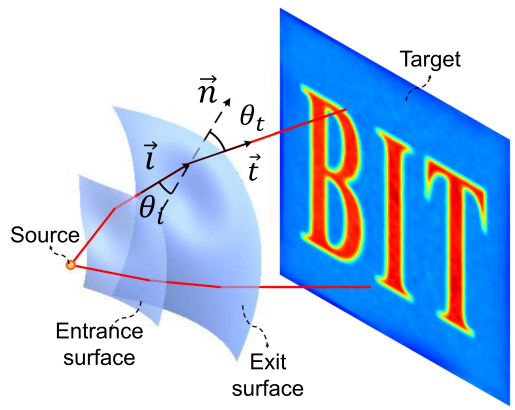

Fig. 4. Schematic of the ray tracing from the source, through the entrance and exit surfaces of the freeform lens, to the target, generating a complex irradiance pattern.

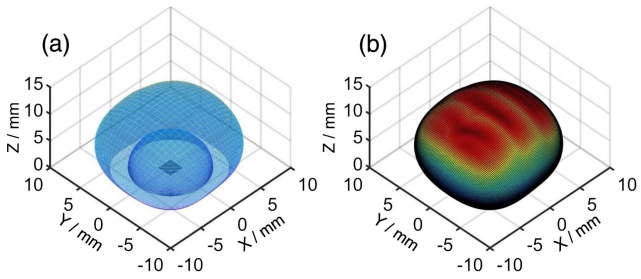

Fig. 5. Illustration of (a) the spherical-freeform lens and an extended light source and (b) the continuous change of the freeform exit surface.

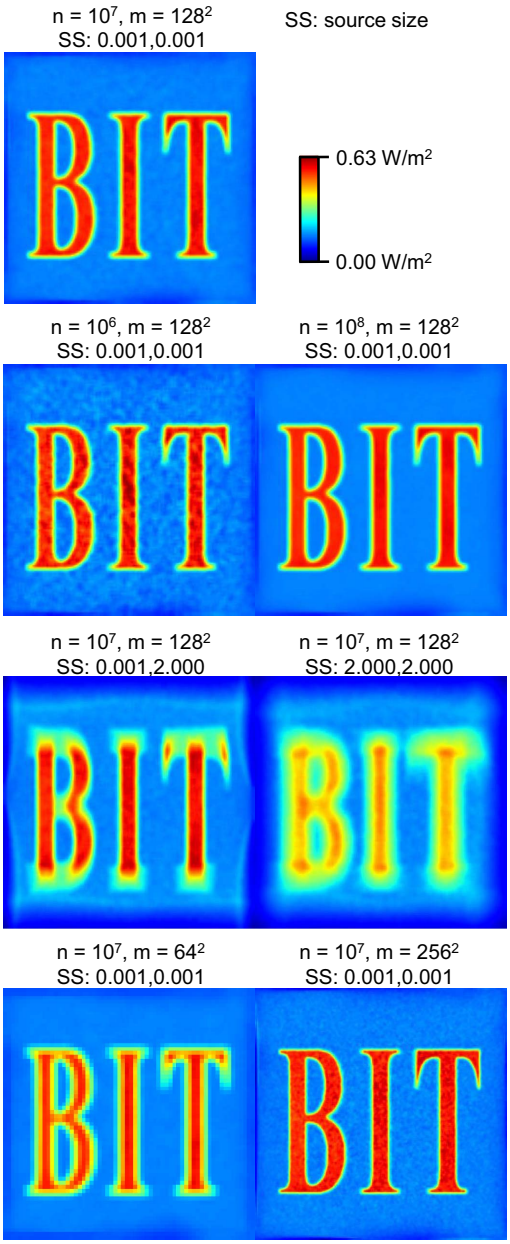

Fig. 6. Simulated irradiance distributions of the first example under different settings of the number of rays, the number of cells, and the source size. Each simulated irradiance distribution is in the range of $\{(x,y) \mid -1000\ \text{mm} < x < 1000\ \text{mm}, -1000\ \text{mm} < y < 1000\ \text{mm}\}$ and filtered by a 3×3 uniform mask.

Table 1. Performances of the Proposed PRT Algorithm Evaluated by the Time Consumption, the Intersection Precision, and the Success Rate τ.

Number of rays		10^7	10^6	10^8	10^7	10^7	10^7	10^7
Source size		(0.001,0.001)	(0.001,0.001)	(0.001,0.001)	(0.001,0.001)	(0.001,0.001)	(0.001,2.0)	(2.0,2.0)
Number of bins		(128,128)	(128,128)	(128,128)	(64,64)	(256,256)	(128,128)	(128,128)
Time cost (s)		11.16	1.30	112.86	11.13	11.12	11.11	11.17
Intersection precision (mm)	max	2.80×10^{-4}	2.40×10^{-4}	2.83×10^{-4}	2.75×10^{-4}	2.70×10^{-4}	2.72×10^{-4}	2.64×10^{-4}
	mean	5.33×10^{-8}	5.31×10^{-8}	5.35×10^{-8}	5.33×10^{-8}	5.32×10^{-8}	5.34×10^{-8}	5.34×10^{-8}
Success rate τ		0.99978	0.99978	0.99973	0.99978	0.99978	0.99965	0.99945

Our computer is equipped with an Intel i9-12900 k CPU with x86-64 architecture and an NVIDIA RTX3090 GPU with 24 GB video memory and NVIDIA Ampere architecture. The programming language is Python 3.11. We use the GPU to trace batches of light rays simultaneously and the CPU to control the calculation flow. We use the CuPy package of Python for surface modeling, intersection solving, and other parallel scientific calculations. The computing time is mainly affected by the GPU performance.

The first example is the ray tracing of a spherical-freeform lens as shown in Fig. 5(a). The refractive index of the lens is 1.4932. This lens is aimed at converting the light distribution of an LED source into a 2000 mm × 2000 mm irradiance distribution at $z = 1000$ mm, forming the letters "BIT". The maximum dimensions of the lens along the x, y, and z directions are 16.97 mm, 16.28 mm, and 12.01 mm, respectively. It can be seen from Fig. 5(b) that the freeform exit surface has complex local features which are directly related to the target irradiance pattern. The entrance surface is 50 × 50 3D grids sampled from a semi-sphere of 5 mm radius, which is also expressed with NURBS. The complex exit freeform surface is described by 256 × 256 control points. We employ a 3 × 3 uniform filter for the simulated irradiance distribution to decrease the noise of the MC method. Figure 6 shows the simulated irradiance distributions of our algorithm under different settings of the number of light rays n, the number of receiver cells $m = m_c \times n_c$, and the source sizes. Table 1 gives the corresponding time consumption, precision of the solved intersections, and the success rates.

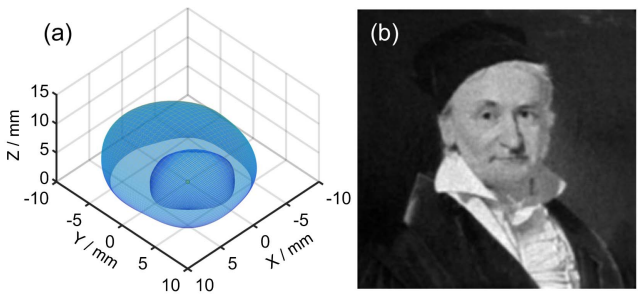

Fig. 7. The second simulation example. (a) The double-freeform lens with a point-like source and (b) its simulation result.

As shown in Table 1, for each case of the first example, the success rate τ is higher than 99.9%. τ could be reduced to below about 96% without using the parameter transformation method. In addition, the parameter transformation method improves the convergence of Newton's method for solving the intersection points. The iterative solutions would not jump out of the allowed range of the parameters in the early stage of the iterative process. However, the intersection calculation errors can rise rapidly toward the edge of the surface ($u, v \to 0$ or 1) since the derivatives ($\mathrm{d}\alpha/\mathrm{d}u, \mathrm{d}\beta/\mathrm{d}v$) of the transformation functions tend to 0. This problem could be avoided by extending the surface so that the active area is bounded away from the edge of the surface.

The second example is about the ray tracing of a more complex double-freeform lens with the maximum dimensions of 15.33 mm × 16.47 mm × 10.04 mm [see Fig. 7(a)], whose refractive index is also 1.4932. Each of the freeform surfaces contains 1024 × 1024 control points. This freeform lens aims at generating a portrait of Johann Carl Friedrich Gauss[42] with the size of 240 mm × 240 mm at $z = 100$ mm from a point-like source whose size is 0.001 mm × 0.001 mm. We perform the PRT using 10^7 rays. The simulated irradiance distribution is shown in Fig. 7(b). The maximum and average values of the distances from the calculated intersection points on the exit surface to the light rays are 2.33×10^{-7} mm and 5.16×10^{-8} mm, respectively. Our program only takes ~11.52 s to perform the simulation, and τ is about 99.951%.

We analyze the time complexity of our PRT program for the first example with respect to the number of rays, the number of receiver cells, the size of the light source, and the number of control points. We use the control variable method to practically test the time consumption of the program. The results are presented in Fig. 8. The time complexity for the number of rays is $O(n)$. Because n is much larger than the number of GPU cores, parallel computing is still done in batches on the hardware. As shown in Fig. 8(a), the simulation time increases linearly as n increases. For the number of receiver cells, the total time complexity is approximately $O(1)$. As shown in Fig. 8(b), as m increases, the simulation time remains almost unchanged. The reason is

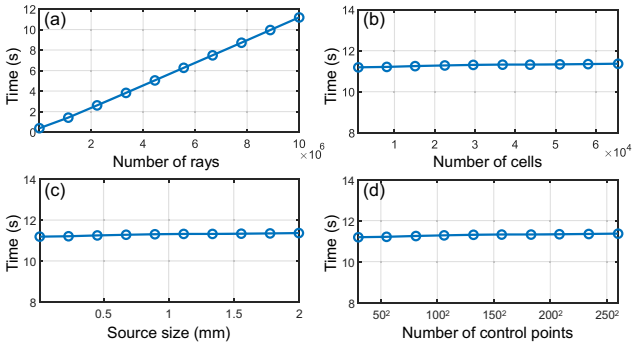

Fig. 8. Variations of the runtime with (a) the number of rays, (b) the number of receiver cells, (c) the source size, and (d) the number of control points (of each surface) for the first example.

that most of the time is spent on the intersection-seeking process. Theoretically, the time complexity of the ray energy statistical process is $O(m)$. However, since n is typically much larger than m to ensure low-level Monte Carlo noise, the impact of m on the overall simulation time is negligible. The total time complexity for the source size is about $O(1)$, as illustrated in Fig. 8(c). The source size only affects the range of the starting points of the randomly sampled rays. We adopt the 2-order analytic form of the NURBS basis functions, which can avoid many recursive operations in base function modeling. Thus, the total time complexity for the control points of the lens surfaces is $O(1)$, as demonstrated in Fig. 8(d).

4. Conclusion and Discussion

We have proposed a PRT method for freeform lenses with complex NURBS surfaces. In this method, we employ the inverse transform sampling strategy to uniformly sample rays in space. We also adopt the analytical form of the B-spline basis function of 2 degrees to achieve a fast surface interpolation. More importantly, we proposed a parameter transform method to reduce the initial solution requirement of Newton's method for solving intersections, which thus increases the success rate of parallel ray tracing. Two examples verify that the proposed PRT method is efficient and effective for quick irradiance evaluations of complex freeform irradiance tailoring lenses.

The proposed PRT method is also applicable to freeform surfaces with other expressions, including XY polynomials, Zernike polynomials, and radial base functions. Our method can help designers quickly evaluate their design results, especially for freeform surfaces with high degrees of freedom. As a relatively fast irradiance evaluation technique, our method can also be applied to the optimal design of freeform lenses. The proposed method is currently restricted to those cases in which a single light ray does not have multiple intersections with a surface. However, such a restriction is fulfilled by most of the freeform lens designs for illumination and beam-shaping applications. Future work may include the development of non-sequential PRT algorithms and the application of freeform illumination lens design using the automatic differentiation technique in the framework of MindSpore.

Acknowledgement

This work was supported by the National Key Research and Development Program of China (No. 2017YFA0701200), the National Natural Science Foundation of China (No. 11704030), and the CAAI-Huawei MindSpore Open Fund (No. CAAIXSJLJJ-2021-014A).

References

1. S. Wills, "Freeform optics: notes from the revolution," Opt. Photon. News **28**, 34 (2017).
2. K. Garrard, T. Bruegge, J. Hoffman, T. Dow, and A. Sohn, "Design tools for freeform optics," Proc. SPIE **5874**, 58740A (2005).
3. J. P. Rolland, M. A. Davies, T. J. Suleski, C. Evans, A. Bauer, J. C. Lambropoulos, and K. Falaggis, "Freeform optics for imaging," Optica **8**, 161 (2021).
4. S. Mao, Y. Li, J. Jiang, S. Shen, K. Liu, and M. Zheng, "Design of a hyper-numerical-aperture deep ultraviolet lithography objective with freeform surfaces," Chin. Opt. Lett. **16**, 030801 (2018).
5. D. Cheng, H. Chen, T. Yang, J. Ke, Y. Li, and Y. Wang, "Optical design of a compact and high-transmittance compressive sensing imaging system enabled by freeform optics," Chin. Opt. Lett. **19**, 112202 (2021).
6. R. Wu, Z. Feng, Z. Zheng, R. Liang, P. Benítez, J. C. Miñano, and F. Duerr, "Design of freeform illumination optics," Laser Photonics Rev. **12**, 1700310 (2018).
7. Z. Feng, D. Cheng, and Y. Wang, "Iterative freeform lens design for prescribed irradiance on curved target," Opto-Electron. Adv. **3**, 200010 (2020).
8. Z. Feng, D. Cheng, and Y. Wang, "Iterative freeform lens design for optical field control," Photon. Res. **9**, 1775 (2021).
9. S. Ortiz, D. Siedlecki, L. Remón, and S. Marcos, "Three-dimensional ray tracing on Delaunay-based reconstructed surfaces," Appl. Opt. **48**, 3886 (2009).
10. S. Schedin, P. Hallberg, and A. Behndig, "Three-dimensional ray-tracing model for the study of advanced refractive errors in keratoconus," Appl. Opt. **55**, 507 (2016).
11. R. Kimura, "Accelerated ray tracing for headlamp simulation," M.A. thesis (University of Arizona, 2017).
12. J. Ye, L. Chen, X. Li, Q. Yuan, and Z. Gao, "Review of optical freeform surface representation technique and its application," Opt. Eng. **56**, 110901 (2017).
13. O. Abert, M. Geimer, and S. Muller, "Direct and fast ray tracing of NURBS surfaces," in *IEEE Symposium on Interactive Ray Tracing* (2006), p. 161.
14. S.-W. Wang, Z.-C. Shih, and R.-C. Chang, "An improved rendering technique for ray tracing Bézier and B-spline surfaces," J. Visual. Comp. Animat. **11**, 209 (2000).
15. A. Efremov, V. Havran, and H.-P. Seidel, "Robust and numerically stable Bézier clipping method for ray tracing NURBS surfaces," in *Proceedings of the 21st Spring Conference on Computer Graphics* (2005), p. 127.
16. S. Campagna, P. Slusallek, and H.-P. Seidel, "Ray tracing of spline surfaces: Bezier clipping, Chebyshev boxing, and bounding volume hierarchy a critical comparison with new results," Vis. Comput. **13**, 265 (1997).
17. A. Hirst, J. Muschaweck, and P. Benítez, "Fast, deterministic computation of irradiance values using a single extended source in 3D," Opt. Express **26**, A651 (2018).
18. H. W. Jensen and N. J. Christensen, "Photon maps in bidirectional Monte Carlo ray tracing of complex objects," Comput. Graph. **19**, 215 (1995).
19. S. N. Pattanaik and S. P. Mudur, "Computation of global illumination in a participating medium by Monte Carlo simulation," J. Visual. Comp. Animat. **4**, 133 (1993).
20. L. Szirmay-Kalos, "Monte-Carlo methods in global illumination," Ph.D. dissertation (Vienna University of Technology, 2000).
21. M. Nimier-David, D. Vicini, T. Zeltner, and W. Jakob, "Mitsuba 2: a retargetable forward and inverse renderer," ACM Trans. Graph. **38**, 203 (2019).

22. R. Olsson and Y. Xu, "An interactive ray-tracing based simulation environment for generating integral imaging video sequences," Proc. SPIE **6016**, 60160F (2005).
23. S. Lee, E. Eisemann, and H.-P. Seidel, "Real-time lens blur effects and focus control," ACM Trans. Graph. **29**, 65 (2010).
24. J. Hanika, "Spectral light transport simulation using a precision-based ray tracing architecture," Ph.D. dissertation (Fakultät für Ingenieurwissenschaften und Informatik, 2011).
25. D. S.-M. Liu and C.-W. Hsu, "Ray-tracing based interactive camera simulation," in *MVA2013 IAPR International Conference on Machine Vision Applications* (2013), p. 383.
26. E. Schrade, J. Hanika, and C. Dachsbacher, "Sparse high-degree polynomials for wide-angle lenses," Comput. Graph. Forum **35**, 89 (2016).
27. D. D. Zhdanov, V. A. Galaktionov, A. G. Voloboy, A. D. Zhdanov, A. A. Garbul, I. S. Potemin, and V. G. Sokolov, "Photorealistic rendering of images formed by augmented reality optical systems," Program Comput. Soft. **44**, 213 (2018).
28. J. Wu, C. Zheng, X. Hu, and F. Xu, "Rendering realistic spectral bokeh due to lens stops and aberrations," Vis. Comput. **29**, 41 (2013).
29. S. Lee and E. Eisemann, "Practical real-time lens-flare rendering," Comput. Graph. Forum **32**, 1 (2013).
30. B. Steinert, H. Dammertz, J. Hanika, and H. P. A. Lensch, "General spectral camera lens simulation," Comput. Graph. Forum **30**, 1643 (2011).
31. Y. Jeong, S. Lee, S. Kwon, and S. Lee, "Expressive chromatic accumulation buffering for defocus blur," Vis. Comput. **32**, 1025 (2016).
32. H. Joo, S. Kwon, S. Lee, E. Eisemann, and S. Lee, "Efficient ray tracing through aspheric lenses and imperfect bokeh synthesis," Comput. Graph. Forum **35**, 99 (2016).
33. T.-W. Schmidt, J. Novák, J. Meng, A. S. Kaplanyan, T. Reiner, D. Nowrouzezahrai, and C. Dachsbacher, "Path-space manipulation of physically-based light transport," ACM Trans. Graph. **32**, 129 (2013).
34. P. Rojo, S. Royo, J. Caum, J. Ramírez, and I. Madariaga, "Generalized ray tracing method for the calculation of the peripheral refraction induced by an ophthalmic lens," Opt. Eng. **54**, 025106 (2015).
35. R. Yao, X. Intes, and Q. Fang, "Generalized mesh-based Monte Carlo for wide-field illumination and detection via mesh retessellation," Biomed. Opt. Express **7**, 171 (2016).
36. M. Mohammadikaji, S. Bergmann, J. Beyerer, J. Burke, and C. Dachsbacher, "Sensor-realistic simulations for evaluation and planning of optical measurement systems with an application to laser triangulation," IEEE Sens. J. **20**, 5336 (2020).
37. S. Olver and A. Townsend, "Fast inverse transform sampling in one and two dimensions," arXiv:1307.1223 (2013).
38. P. Jester, C. Menke, and K. Urban, "B-spline representation of optical surfaces and its accuracy in a ray trace algorithm," Appl. Opt. **50**, 822 (2011).
39. L. Chen, Y. Gong, B. Li, and X. Ren, "Algorithm for intersecting line of free-form surfaces," J. Xi'An Jiaotong Univ. **34**, 70 (2000).
40. A. E. Burgess, "The Rose model, revisited," J. Opt. Soc. Am. A **16**, 633 (1999).
41. Z. Feng, D. Cheng, and Y. Wang, "Iterative wavefront tailoring to simplify freeform optical design for prescribed irradiance," Opt. Lett. **44**, 2274 (2019).
42. C. A. Jensen, "Johann Carl Friedrich Gauss," Wikimedia Commons, 2021, https://commons.wikimedia.org/w/index.php?title=File:Carl_Friedrich_Gauss_1840_by_Jensen.jpg&oldid=611380576.