



Delft University of Technology

## Gradient descent-based freeform optics design for illumination using algorithmic differentiable non-sequential ray tracing

Koning, Bart de; Heemels, Alexander; Adam, Aurèle; Möller, Matthias

**DOI**

[10.1007/s11081-023-09841-9](https://doi.org/10.1007/s11081-023-09841-9)

**Publication date**

2023

**Document Version**

Final published version

**Published in**

Optimization and Engineering

**Citation (APA)**

Koning, B. D., Heemels, A., Adam, A., & Möller, M. (2023). Gradient descent-based freeform optics design for illumination using algorithmic differentiable non-sequential ray tracing. *Optimization and Engineering*, 25 (2024)(3), 1203-1235. <https://doi.org/10.1007/s11081-023-09841-9>

**Important note**

To cite this publication, please use the final published version (if applicable).

Please check the document version above.

**Copyright**

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

**Takedown policy**

Please contact us and provide details if you believe this document breaches copyrights.

We will remove access to the work immediately and investigate your claim.



# Gradient descent-based freeform optics design for illumination using algorithmic differentiable non-sequential ray tracing

Bart de Koning<sup>1,2</sup> · Alexander Heemels<sup>2</sup> · Aurèle Adam<sup>2</sup> · Matthias Möller<sup>1</sup>

Received: 1 March 2023 / Revised: 24 August 2023 / Accepted: 24 August 2023  
© The Author(s) 2023

## Abstract

*Algorithmic differentiable ray tracing* is a new paradigm that allows one to solve the forward problem of how light propagates through an optical system while obtaining gradients of the simulation results with respect to parameters specifying the optical system. Specifically, the use of algorithmically differentiable non-sequential ray tracing provides an opportunity in the field of illumination engineering to design complex optical system. We demonstrate its potential by designing freeform lenses that project a prescribed irradiance distribution onto a plane. The challenge consists in finding a suitable surface geometry of the lens so that the light emitted by a light source is redistributed into a desired irradiance distribution. We discuss the crucial steps allowing the non-sequential ray tracer to be differentiable. The obtained gradients are used to optimize the geometry of the freeform, and we investigate the effectiveness of adding a multi-layer perceptron neural network to the optimization that outputs parameters defining the freeform lens. Lenses are designed for various sources such as collimated beams or point sources, and finally, a grid of point sources approximating an extended source. The obtained lens designs are finally validated using the commercial non-sequential ray tracer LightTools.

---

✉ Alexander Heemels  
A.N.M.Heemels@tudelft.nl

Bart de Koning  
B.deKoning@tudelft.nl

Aurèle Adam  
A.J.L.Adam@tudelft.nl

Matthias Möller  
M.Moller@tudelft.nl

<sup>1</sup> Numerical Analysis, Delft University of Technology, Mekelweg 4, 2628 CD Delft, South Holland, Netherlands

<sup>2</sup> Imaging Physics, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, South Holland, Netherlands

**Keywords** Differentiable ray tracing · Freeform lens · B-spline surface · Optical design · Neural network · Illumination optics

## 1 Introduction

In the field of illumination optics, optical engineers design optical elements to transport the light from a source, which can be an LED, laser, or incandescent lamp, to obtain a desired irradiance (spatial density of the luminous flux) or intensity (angular density of the luminous flux) (Grant 2011). To transport the light from the source to the target, the optical engineer can construct a system consisting of various optical elements such as lenses, mirrors, diffusers, and light guides (John 2013). One particular type of optic used in automotive and road lighting applications is the freeform lens, a lens without any form of symmetry (Falaggis et al. 2022; Mohedano et al. 2016). Designing these lenses is a complex problem and is often solved using the assumption that the source has zero-étendue, thus using point sources and collimated beams. Under this assumption, the problem of finding a suitable geometry of a freeform lens or mirror which redistributes the light from the source to the target can be formulated as a Monge–Kantorovich mass transport problem. The geometry of the lens can then be obtained by solving the Monge–Ampere equation (Ries and Muschaweck 2002; Wu et al. 2013; Prins et al. 2015) or by solving the Monge–Kantorovich problem through optimization (Doskolovich et al. 2018) such as the supporting quadratic method (Fournier et al. 2010; Oliker et al. 2015). Alternatively, ray mapping methods (Feng et al. 2013; Bösel and Gross 2018) can be used, which first construct a mapping determining where the ray leaving the source ends up on the target and then uses this mapping to calculate the geometry of the lens. An in-depth discussion of the listed methods is given by Wu et al. (2018) and Romijn (2021, Chapter 5). Great effort is involved in extending these methods to account for varying amounts of optical surfaces (Anthonissen et al. 2021), Fresnel losses (Van Roosmalen et al. 2022), surface scattering (Kronberg et al. 2023), volume varying properties (Lippman and Schmidt 2020), or solving the problem for finite étendue source such as LED's (Tukker 2007; Fournier et al. 2009; Wester et al. 2014; Sorgato et al. 2019; Wei et al. 2021; Zhu et al. 2022; Muschaweck 2022). However, combining multiple of these effects still remains a major challenge.

The performance of an illumination system is evaluated using ray tracing, which is the process of calculating the path of a ray originating from a source through the optical system. Sequential ray tracers such as Zemax (Ansys 2023) and Code V (Synopsys 2023), primarily used in the design of imaging optics, trace a small number of rays to determine the quality of the image. Non-sequential ray tracers such as LightTools (Synopsis 2023) and Photopia (Itioptrics 2023) use many rays to simulate the optical flux through the system and share similarities with the rendering procedures in computer graphics (Pharr et al. 2016), with the main difference being that the rays are traced from source to detector.

Algorithmically differentiable ray tracing, a generalization of differential ray tracing (Feder 1968; Stone and Forbes 1997; Oertmann 1989; Chen and Lin 2012), is a tool that is being developed for both sequential (Volatier et al. 2017; Sun et al. 2021; Wang et al. 2022; Nie et al. 2023) and non-sequential (Nimier-David et al. 2019; Jakob

et al. 2022) ray tracing. *Differential ray tracing* obtains system parameter gradients using numerical or algebraic differentiation. The gradient can be calculated numerically using numerical differentiation or the adjoint method (Givoli 2021), requiring the system to be ray traced twice, once for its current state and once with perturbed system parameters. Analytic expressions for the gradient can be obtained by tracing the rays analytically through the system, calculating where the ray intersects the system's surfaces and how the ray's trajectory is altered. However, these expressions can become long and complicated depending on the system. In addition, the method is limited to optics described by conics as finding analytic ray surface intersection with surfaces of higher degrees becomes complicated or even impossible. Algorithmic differentiable ray tracing can handle these issues by obtaining the gradients with one single forward simulation without limitations on the number or type of surfaces. In addition, it can be seamlessly integrated into gradient-descent-based optimization pipelines. A modern framework for this is *Physics Informed Machine Learning* (Karniadakis et al. 2021), where a neural network is trained to approximate the solution to a physics problem formulated using data, a set of differential equations, or an implemented physics simulation (or a combination of these).

We investigate the reliability of designing freeform lenses with B-spline surfaces (Piegl and Tiller 1996) using algorithmically differentiable non-sequential ray tracing and gradient-based optimization to redirect the light of a light source into a prescribed irradiance distribution. The source models will be the collimated light source, point source, and finally, sources with a finite extent. The results are validated using the commercial ray trace program LightTools (Synopsis 2023). In addition, we investigate the effectiveness of optimizing a network to determine the optimal B-spline control points as proposed in Möller et al. (2021) and Gasick and Qian (2023), and compare it to optimizing the control points directly and seeing the possible speed-up.

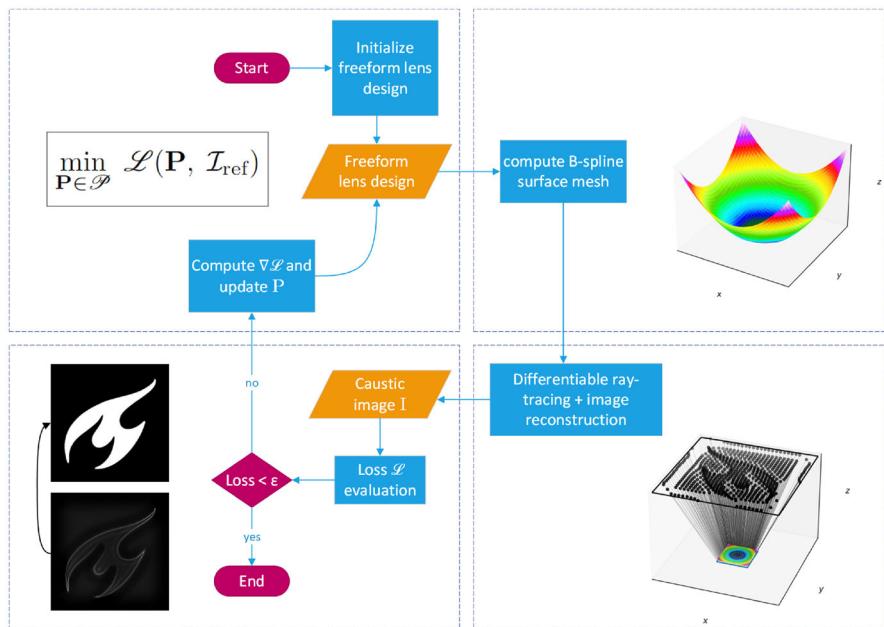
## 2 Gradient-based freeform design

The overall structure of our pipeline is depicted in Fig. 1. A freeform surface is defined by the parameters  $P \in \mathcal{P}$ , where  $\mathcal{P}$  is the set of permissible parameter values. This surface is combined with a flat surface to create a lens, and an irradiance distribution  $\mathcal{I}$  is produced by tracing rays through the lens onto a screen. The irradiance distribution is compared to a target  $\mathcal{I}_{\text{ref}}$  yielding a loss  $\mathcal{L}(\mathbf{P}, \mathcal{I}_{\text{ref}})$ . The optimization problem we are trying to solve can then be formulated as

$$\min_{\mathbf{P} \in \mathcal{P}} \mathcal{L}(\mathbf{P}, \mathcal{I}_{\text{ref}}), \quad (1)$$

which we solve by using gradient descent.

The freeform surface of the lens is defined in terms of a B-spline surface. From a manufacturing standpoint, this is convenient since B-spline surfaces can be chosen to be  $C^1$  smooth (in fact, B-spline surfaces can be  $C^n$  smooth for arbitrarily large  $n$ ). From an optimization perspective, B-spline surfaces have the property that the control points that govern the shape of the surface and which will be optimized have a local



**Fig. 1** Overview of our learning-based freeform design pipeline

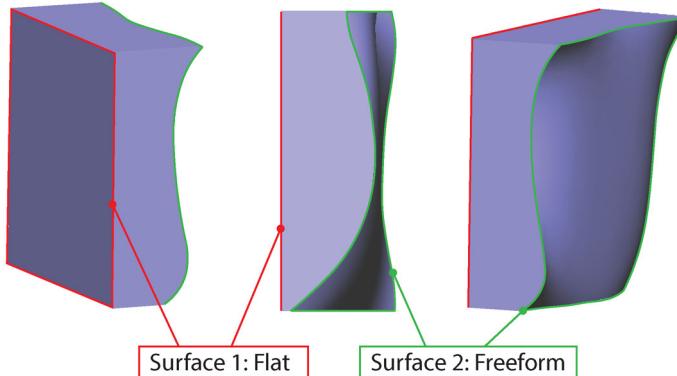
influence on the surface geometry, which in turn has a local influence on the resulting irradiance distribution.

## 2.1 The lens model using a B-spline surface

We define a lens as in Fig. 2 as the volume between a flat surface and a B-spline surface, with a uniform refractive index. A B-spline surface  $\mathbf{S}$  in  $\mathbb{R}^3$ , see Fig. 3, is given by Piegl and Tiller (1996, Eq. (3.11)):

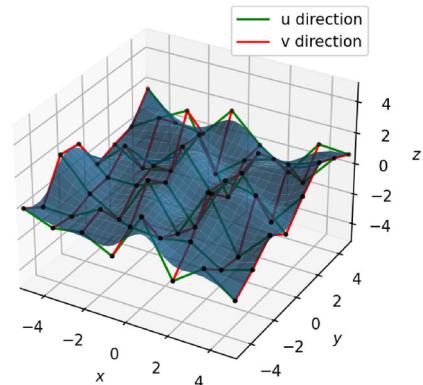
$$\mathbf{S}(u, v) = \sum_{i=0}^{n_1} \sum_{j=0}^{n_2} N_{i,p}(u) N_{j,q}(v) \mathbf{P}_{i,j}, \quad (u, v) \in [0, 1]^2. \quad (2)$$

The size of the control net grid of control points is defined using  $n_1$  and  $n_2$ , which are positive integers and  $\mathbf{P}_{i,j} \in \mathbb{R}^3$  and  $N_{i,p}$  and  $N_{j,q}$  are univariate B-spline basis functions of degree  $p$  and  $q$  and are recursively defined by the Cox–de Boor formula (Piegl and Tiller 1996, Eq. (2.5)). The basis functions are  $p$ -degree piece-wise polynomials defined using knots  $u_i$  which are non-decreasing real numbers in  $[a, b]$  collected in a knot vector (Piegl and Tiller 1996, Eq. (2.13)):



**Fig. 2** The used lens type: a volume enclosed between a flat surface and a freeform surface with a uniform refractive index

**Fig. 3** A B-spline surface of degrees  $(p, q) = (3, 3)$  with indicated directions of the  $u$  and  $v$  parameters. The control points are shown in black



$$\mathcal{V} = \left( \underbrace{0, \dots, 0}_{p+1}, u_{p+1}, \dots, u_{r-p-1}, \underbrace{1, \dots, 1}_{p+1} \right) \in \mathbb{R}^{r+1}. \quad (3)$$

The interior knots are chosen to be equispaced, i.e.

$$u_i = \frac{i - p}{n_1 - p + 1}, \quad i = p, \dots, n_1 + 1. \quad (4)$$

### 2.1.1 Linearizing the B-spline parametrizations

The volume  $V \subset \mathbb{R}^3$  of the modeled lens has a rectangular extent  $[-r_x, r_x] \times [-r_y, r_y]$  in the  $(x, y)$ -plane with  $2r_x$  and  $2r_y$  being the width and height of the lens, respectively.

The lens volume is enclosed on one side by a B-spline surface

$$\mathbf{S}(u, v) = (X(u, v), Y(u, v), Z(u, v)), \quad (5)$$

where  $X, Y, Z$  are the individual coordinate parameterizations, for instance

$$X(u, v) = \sum_{i=0}^{n_1} \sum_{j=0}^{n_2} N_{i,p}(u) N_{j,q}(v) P_{i,j}^x, \quad (6a)$$

$$Y(u, v) = \sum_{i=0}^{n_1} \sum_{j=0}^{n_2} N_{i,p}(u) N_{j,q}(v) P_{i,j}^y. \quad (6b)$$

For simplicity of calculations on ray-sampling and ray-intersection (Sect. 2.2), it is helpful to define the mapping  $(u, v) \mapsto (X(u, v), Y(u, v))$  in a way that it is analytically invertible. Therefore the coordinates of the control points are chosen such that the parametrizations  $X$  and  $Y$  are linear:

$$X : u \mapsto (2u - 1)r_x \in [-r_x, r_x], \quad (7a)$$

$$Y : v \mapsto (2v - 1)r_y \in [-r_y, r_y]. \quad (7b)$$

In general,  $X$  and  $Y$  are degree  $p$  and  $q$  piece-wise polynomials, respectively, and thus not linear. Linearity can be achieved by making use of the nodal representation of the B-spline basis functions (Cohen et al. 2010, Eq. (23)):

$$u = \sum_{i=0}^{n_1} u_{i,p}^* N_{i,p}(u), \quad u \in [0, 1], \quad u_{i,p}^* = \frac{u_{i+1} + \cdots + u_{i+p}}{p}, \quad (8)$$

which provides a specific knot vector-dependent linear combination of the basis functions that yields the identity function on the domain  $[0, 1]$ . The values  $u_{i,p}^*$  are called the Greville abscissae (Farin 2002, Sect. 8.6).

We assume that the  $P_{i,j}^x$  are independent of  $j$ , and choose  $j = 0$  as a representative. Then we obtain by the definition of  $X$ :

$$X(u, v) = \sum_{i=0}^{n_1} \sum_{j=0}^{n_2} P_{i,j}^x N_{i,p}(u) N_{j,q}(v) \quad (9a)$$

$$= \sum_{i=0}^{n_1} P_{i,0}^x N_{i,p}(u) \underbrace{\sum_{j=0}^{n_2} N_{j,q}(v)}_{=1} \quad (9b)$$

where  $\sum_{i=0}^{n_1} N_{i,p}(u) = 1$  by the partition of unity property of the basis functions (Piegl and Tiller 1996, P2.4). Now we see that if we define  $P_{i,j}^x := u_{i,p}^*$  then  $X(u) = u$ .

Thus if we apply the mapping  $u \mapsto (2u - 1)r_x$  to both sides of Eq. (8), we obtain

$$(2u - 1)r_x = \sum_{i=0}^{n_1} (2u_{i,p}^* - 1)r_x N_{i,p}(u). \quad (10)$$

This equality can be understood by expanding the 1 into the sum over all  $N_{i,p}(u)$  by again exploiting the partition of unity property. Thus if we define  $P_{i,j}^x := (2u_{i,p}^* - 1)r_x$  and equivalently  $P_{i,j}^y := (2v_{j,q}^* - 1)r_y$ , then Eqs. 7a and 7b are satisfied.

The lens is then defined as the volume in  $\mathbb{R}^3$  enclosed by the B-spline surface  $\mathbf{S}$  and the flat surface given by  $z = z_{\text{in}}$  on  $[-r_x, r_x] \times [-r_y, r_y]$ :

$$\begin{aligned} V = & \left\{ (x, y, z) \in \mathbb{R}^3 \mid z_{\text{in}} \leq z \right. \\ & \left. \leq Z\left(X^{-1}(x), Y^{-1}(y)\right), |x| \leq r_x, |y| \leq r_y \right\}. \end{aligned} \quad (11)$$

For the arguments of  $Z(u, v)$  the inverses of  $X$  and  $Y$  are used:

$$X^{-1}(x) = \frac{1}{2} \left( \frac{x}{r_x} + 1 \right), \quad Y^{-1}(y) = \frac{1}{2} \left( \frac{y}{r_y} + 1 \right). \quad (12)$$

### 2.1.2 Lens constraints

To let the lens be well-defined the surfaces of the lens should not intersect:

$$z_{\text{in}} < Z(u, v), \quad (u, v) \in [0, 1]^2. \quad (13)$$

By the convex hull property of B-spline surfaces (Piegl and Tiller 1996, P3.22) it suffices that

$$P_{i,j}^z > z_{\text{in}} \quad \forall (i, j). \quad (14)$$

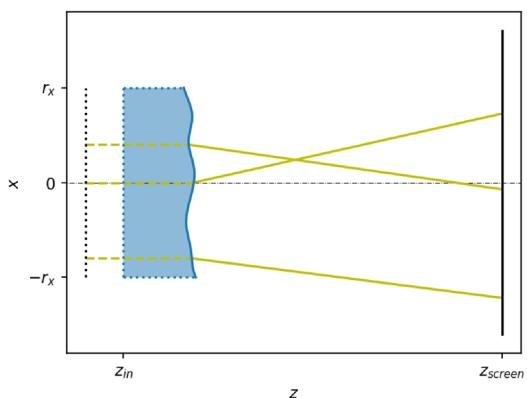
Manufacturing can require that the lens has some minimal thickness  $\delta$ , so that the constraint is stronger:

$$P_{i,j}^z \geq \delta + z_{\text{in}} \quad \forall (i, j). \quad (15)$$

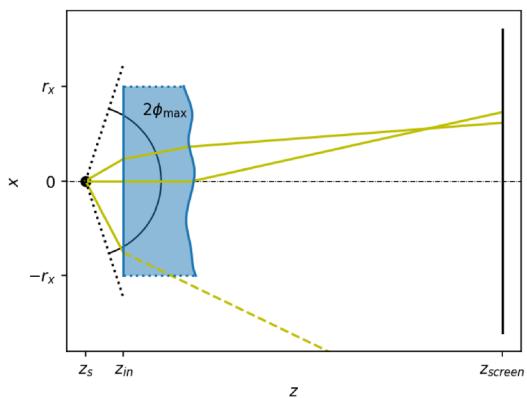
### 2.2 Differentiable ray tracer

Our implementation traces rays from a source through the flat lens surface and the freeform lens surface to the detector screen as depicted in Figs. 4 and 5. Other ray paths, e.g., total internal reflection at lens surfaces, are not considered and it is assumed that the contribution of these to the resulting irradiance distribution is negligible.

**Fig. 4** Schematic of the ray tracing with a collimated beam



**Fig. 5** Schematic of the ray tracing with a point source



### 2.2.1 Sources and ray-sampling

Non-sequential ray tracing is a Monte-Carlo approximation method of the solution to the continuous integration formulation of light transport through an optical system. For a detailed discussion of this topic, see Pharr et al. (2016, Ch. 14). Thus to perform ray tracing, the light emitted by a source must be discretized into a finite set of rays

$$l : t \rightarrow \mathbf{o} + \hat{\mathbf{d}}t, \quad (16)$$

where  $\mathbf{o}$  is the origin of the ray and  $\hat{\mathbf{d}}$  its normalized direction vector. Both collimated beam and point sources will be considered, see Figs. 4 and 5, respectively.

Tracing rays from a collimated beam can be understood from Fig. 4. The path of all rays from the source plane to the B-spline surface is a line segment parallel to the  $z$ -axis. Therefore, we can sample the incoming rays directly on the B-spline surface, with  $\hat{\mathbf{d}} = (0, 0, 1)^\top$ . By the linearity of  $X$  and  $Y$  sampling on the B-spline domain  $[0, 1]^2$  is analogous to sampling on the lens extent  $[-r_x, r_x] \times [-r_y, r_y]$  in terms of distribution. Rays are sampled in a (deterministic) square grid on  $[0, 1]^2$ .

For a point source, each ray starts at the location of the source, and the direction vector  $\hat{\mathbf{d}}$  is sampled over the unit sphere  $\mathbb{S}^2$ . More precisely,  $\hat{\mathbf{d}}$  is given by

$$\hat{\mathbf{d}} = (\cos \theta \sin \phi, \sin \theta \sin \phi, \cos \phi)^\top, \quad (17)$$

with  $\theta \in [0, 2\pi)$  and  $\phi \in [0, \phi_{\max}]$  for some  $0 \leq \phi_{\max} < \frac{\pi}{2}$ , see Fig. 5.  $\phi_{\max}$  is chosen as small enough to minimize the number of rays that miss the lens entrance surface but large enough such that the whole surface is illuminated. For instance, if the source is on the  $z$ -axis, then  $\phi_{\max} = \arctan\left(\frac{\sqrt{r_x^2 + r_y^2}}{z_{\text{in}} - z_s}\right)$  where  $z_{\text{in}}$  is the  $z$ -coordinate location of the entrance surface and  $z_s$  the  $z$ -coordinate of the source. To uniformly sample points on this sphere segment,  $\theta$  is sampled (non-deterministically) uniformly in  $[0, 2\pi)$  and  $\phi$  is given by

$$\phi = \arccos(1 - (1 - \cos \phi_{\max})a) \quad (18)$$

where  $a$  is sampled (non-deterministically) uniformly in  $[0, 1]$ . This sampling is used to produce the results in Sect. 3.

For the point source, the calculation of the intersection of a ray with the B-spline surface is non-trivial. This calculation comes down to finding the smallest positive root of the  $p + q$  degree piece-wise polynomial function

$$f(t) = Z\left(\binom{o_u}{o_v} + \binom{d_u}{d_v}t\right) - d_z t - o_z, \quad (19)$$

if such a root exists and yields a point in the domain of  $Z$ . Here the subscripts  $u$  and  $v$  denote that the ray is considered in  $(u, v, z)$  space instead of  $(x, y, z)$  space, so for instance

$$o_u = X^{-1}(o_x) = \frac{1}{2}\left(\frac{o_x}{r_y} + 1\right), \quad d_v = \frac{d_y}{2r_y}. \quad (20)$$

The roots of Eq. (19) cannot generally be found analytically for  $p + q > 4$ , and thus an intersection algorithm is implemented, which is explained in the next section.

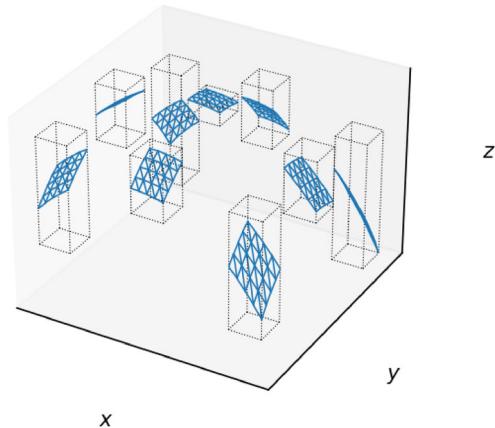
## 2.2.2 B-spline surface intersection algorithm

The intersection algorithm is based on constructing a triangle mesh approximation of the B-spline surface and computing intersections with that mesh.

### Triangle mesh intersection phase 1: bounding boxes

Checking every ray against every triangle for intersection is computationally expensive, so it is helpful to have bounding box tests that provide rough information about whether the ray is even near some section of the B-spline surface. B-spline theory

**Fig. 6** Triangles and corresponding bounding box for a few knot span products of a spherical surface



provides a tool for this: the strong convex hull property, which yields the bounding box

$$B_{i_0, j_0} = [u_{i_0}, u_{i_0+1}) \times [v_{j_0}, v_{j_0+1}) \times [z_{i_0, j_0}^{\min}, z_{i_0, j_0}^{\max}] \quad (21)$$

where  $z_{i, j}^{\min}$  and  $z_{i, j}^{\max}$  are the minimum and maximum  $z$ -values of the control points that affect the B-spline surface on the knot span product  $[u_{i_0}, u_{i_0+1}) \times [v_{j_0}, v_{j_0+1})$ , hence those with indices  $i_0 - p \leq i \leq i_0, j_0 - q \leq j \leq j_0$ . Formulated in terms of  $Z(u, v)$  this yields

$$z_{i_0, j_0}^{\min} \leq Z(u, v) \leq z_{i_0, j_0}^{\max}, \quad (u, v) \in [u_{i_0}, u_{i_0+1}) \times [v_{j_0}, v_{j_0+1}) . \quad (22)$$

Examples of such bounding boxes are shown in Fig. 6.

There are two steps in applying the bounding boxes in the intersection algorithm. First, a test for the entire surface (in  $(u, v, z)$ -space):

$$[0, 1]^2 \times \left[ \min_{i, j} P_{i, j}^z, \max_{i, j} P_{i, j}^z \right] . \quad (23)$$

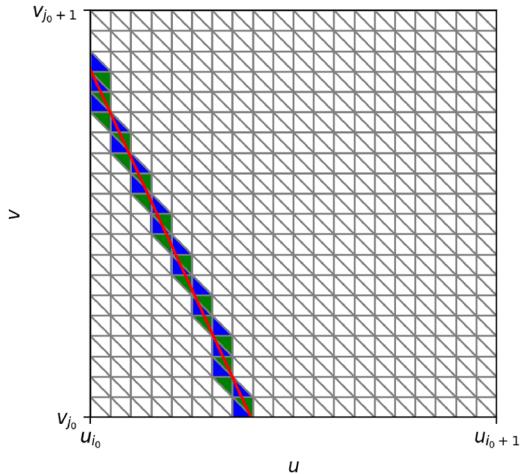
Second, a recursive method where, starting with all knot span products, each rectangle of knot span products is divided into at most 4 sub-rectangles for a new bounding box test until individual knot span products are reached.

*Triangle mesh intersection phase 2:  $(u, v)$ -space triangle intersection* Each non-trivial knot span product  $[u_{i_0}, u_{i_0+1}) \times [v_{j_0}, v_{j_0+1})$  is divided into a grid of  $n_u$  by  $n_v$  rectangles. Thus we can define the boundary points

$$u_{i_0, k} = u_{i_0} + k \Delta u_{i_0}, \quad \Delta u_{i_0} = \frac{u_{i_0+1} - u_{i_0}}{n_u}, \quad k = 0, \dots, n_u, \quad (24a)$$

$$v_{i_0, \ell} = v_{j_0} + \ell \Delta v_{j_0}, \quad \Delta v_{j_0} = \frac{v_{j_0+1} - v_{j_0}}{n_v}, \quad \ell = 0, \dots, n_v. \quad (24b)$$

**Fig. 7** Example of which triangles are candidates for a ray-surface intersection with the ray plotted in red, based on their  $u, v$ -domain



Each rectangle is divided into a lower left and an upper right triangle, as demonstrated in Fig. 7. In this figure it is shown for a ray projected onto the  $(u, v)$ -plane in some knot span which triangles are candidates for an intersection in  $(u, v, z)$ -space. This is determined by the following rules:

- A lower left triangle is intersected in the  $(u, v)$ -plane if either its left or lower boundary is intersected by the ray;
- An upper right triangle is intersected in the  $(u, v)$ -plane if either its right or upper boundary is intersected by the ray.

The intersection of these boundaries can be determined by finding the indices of the horizontal lines at which the vertical lines are intersected:

$$\ell_k = \left\lfloor \frac{o_v + (u_{i_0,k} - o_u) \frac{d_v}{d_u} - v_{j_0}}{\Delta v_{j_0}} \right\rfloor, \quad (25)$$

and analogously  $k_\ell$ .

*Triangle mesh intersection phase 3:  $u, v, z$ -space triangle intersection* A lower left triangle can be expressed by a plane

$$T(u, v) = Au + Bv + C \quad (26)$$

defined by the following linear system:

$$\begin{pmatrix} u_{i_0,k} & v_{j_0,\ell} & 1 \\ u_{i_0,k+1} & v_{j_0,\ell} & 1 \\ u_{i_0,k} & v_{j_0,\ell+1} & 1 \end{pmatrix} \begin{pmatrix} A \\ B \\ C \end{pmatrix} = \begin{pmatrix} [1.75]z_{i_0,k}^{j_0,\ell} \\ z_{i_0,k+1}^{j_0,\ell} \\ z_{i_0,k}^{j_0,\ell+1} \end{pmatrix}. \quad (27)$$

Here we use the following definition:

$$z_{i_0,k}^{j_0,\ell} = Z(u_{i_0,k}, v_{j_0,\ell}). \quad (28)$$

This yields the plane

$$T(u, v) = z_{i_0,k}^{j_0,\ell} + n_u \left( z_{i_0,k+1}^{j_0,\ell} - z_{i_0,k}^{j_0,\ell} \right) \frac{u - u_{i_0,k}}{u_{i_0+1} - u_{i_0}} \quad (29)$$

$$+ n_v \left( z_{i_0,k}^{j_0,\ell+1} - z_{i_0,k}^{j_0,\ell} \right) \frac{v - v_{j_0,\ell}}{v_{j_0+1} - v_{j_0}}. \quad (30)$$

Note that to define this triangle, the B-spline basis functions are evaluated at fixed points in  $[0, 1]^2$  independent of the rays or the  $P_{i,j}^z$ . This means that for a lens that will be optimized these basis function values can be evaluated and stored only once rather than in every iteration, for computational efficiency.

Computing the intersection with the ray  $\tilde{\mathbf{r}}(t) = \tilde{\mathbf{o}} + \tilde{\mathbf{d}}t$  is now straight-forward, and yields

$$t_{\text{int}} = -\frac{C + \langle \tilde{\mathbf{o}}, \mathbf{n} \rangle}{\langle \tilde{\mathbf{d}}, \mathbf{n} \rangle}, \quad \mathbf{n} = \begin{pmatrix} 0 \\ 1 \\ \partial_u T \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \\ \partial_v T \end{pmatrix} = \begin{pmatrix} A \\ B \\ -1 \end{pmatrix}, \quad (31)$$

where  $\mathbf{n}$  is a normal vector to the triangle, computed using the cross product. This also explains why  $\langle \tilde{\mathbf{d}}, \mathbf{n} \rangle = 0$  does not yield a well-defined result: in this situation the ray is parallel to the triangle.

The last thing to check is whether  $\tilde{l}(t_{\text{int}})$  lies in the  $(u, v)$ -domain of the triangle, which can be checked by three inequalities for the three boundaries of the triangle:

$$o_u + d_u t_{\text{int}} \geq u_{i_0,k} \quad (32a)$$

$$0 \leq o_v + d_v t_{\text{int}} - v_{j_0,\ell} < \frac{n_u}{n_v} \frac{v_{j_0+1} - v_{j_0}}{u_{i_0+1} - u_{i_0}} (u_{i_0,k+1} - (o_u + d_u t_{\text{int}})). \quad (32b)$$

The computation for an upper right triangle is completely analogous. The upper triangle has a closed boundary, whereas the lower triangle has an open one and vice versa, which means that the  $(u, v)$  domains of the triangles form an exact partition of  $[0, 1]^2$ . Thus the triangle mesh is ‘water-tight’, meaning that no ray intersection should be lost by rays passing in between triangles.

## 2.3 Image reconstruction

The ray tracing produces an irradiance distribution in the form of an image matrix  $\mathcal{I} \in \mathbb{R}_{\geq 0}^{n_x \times n_y}$ , where the elements correspond to a grid of rectangles called pixels that partition the detector screen positioned at  $z = z_{\text{screen}} > \max_{i,j} P_{i,j}^z$ . The screen

resolution  $(n_x, n_y)$  and the screen radii  $(R_x, R_y)$  together yield the pixel size

$$(w_x, w_y) = \left( \frac{2R_x}{n_x}, \frac{2R_y}{n_y} \right). \quad (33)$$

For reasons explained later in this section, sometimes a few ‘ghost pixels’ are added, so the effective screen radii are

$$R_x^* := R_x + \frac{v_x - 1}{2} w_x, \quad R_y^* := R_y + \frac{v_y - 1}{2} w_y, \quad (34)$$

and the effective screen resolution is  $(n_x + v_x - 1, n_y + v_y - 1)$  where  $v_x$  and  $v_y$  are odd positive integers whose meaning will become clear later in this section.

Producing the irradiance distribution from the rays that intersect the detector screen is called image reconstruction (Pharr et al. 2016, Sect. 7.8). The way that a ray contributes to a pixel with indices  $i, j$  is governed by a reconstruction filter

$$F_{i,j} : [-R_x, R_x] \times [-R_y, R_y] \rightarrow \mathbb{R}_{\geq 0}, \quad (35)$$

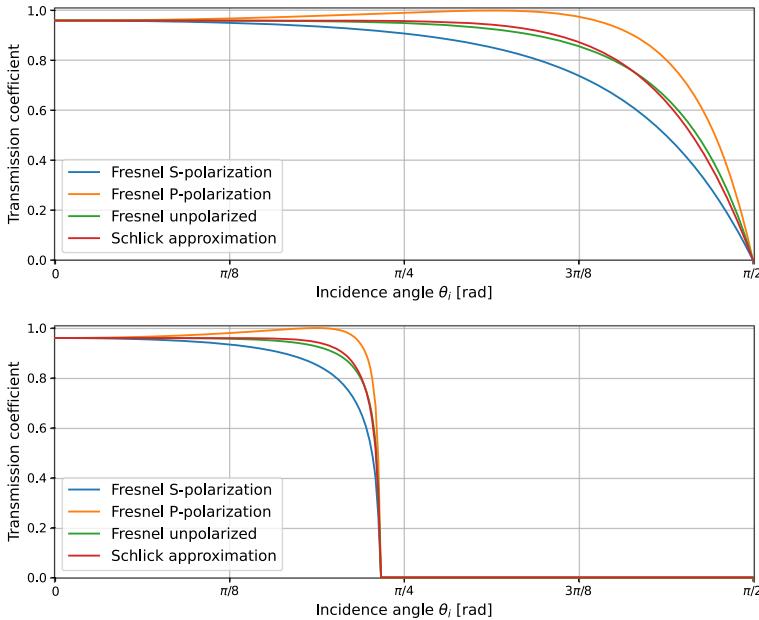
yielding for the irradiance distribution

$$\mathcal{I}_{i,j} = \sum_{k=1}^N \omega_k F_{i,j}(\mathbf{x}_k), \quad (36)$$

for a set of ray intersections  $\{\mathbf{x}_k\}_{k=1}^N$  with corresponding final ray weights  $\{\omega_k\}_{k=1}^N$ . The ray weights are initialized at the sampling of the ray at the source. They are slightly modified by the lens boundary interactions as a small portion of the light is reflected rather than refracted. The amount by which the ray weights are modified is governed by the Fresnel equations (Fowles 1989, Sect. 2.7.1). In our implementation, the Fresnel equations are approximated by Schlick’s approximation (Schlick 1994, Eq. (24)) which allows us to approximate the specular reflection coefficient of unpolarized light where  $\mathcal{R}_0 = ((\eta_1 - \eta_2)/(\eta_1 + \eta_2))^2$  is the reflection coefficient at normal incidence and  $\eta_1, \eta_2$  are the refractive indices of the material before and after the surface respectively. Depending on the ratio  $\mathcal{E} = \eta_1/\eta_2$ , the factor  $\cos \theta_x$  is either the incidence angle with respect to the surface normal  $\theta_x = \theta_i$  or the transmitted angle  $\theta_x = \theta_t$ :

$$\mathcal{R} = \begin{cases} \mathcal{R}_0 + (1 - \mathcal{R}_0)(1 - \cos \theta_i)^5 & \text{if } \eta_1/\eta_2 < 1, \\ \mathcal{R}_0 + (1 - \mathcal{R}_0)(1 - \cos \theta_t)^5 & \text{if } \eta_1/\eta_2 > 1. \end{cases} \quad (37)$$

The transmission coefficient is then given by  $\mathcal{T} = 1 - \mathcal{R}$ . The transmission coefficients are shown in Fig. 8 for both cases. In the current implementation, all ray weights are initialized equally. The precise value does not matter since the relationship between the initial and final weights is linear. The loss function (Sect. 2.5) compares scaled versions of the produced and target irradiance distribution.



**Fig. 8** (Top) Transmission coefficient for  $\mathcal{E} < 1$  with  $\eta_1 = 1$  and  $\eta_2 = 1.5$ ; (Bottom) Transmission coefficient for  $\mathcal{E} > 1$  with  $\eta_1 = 1.5$  and  $\eta_2 = 1$

In the simplest reconstruction case, the value of a pixel is given by the sum of the weights of the rays that intersect the detector screen at that pixel (called box reconstruction in Pharr et al. 2016, Sect. 7.8.1). In this case the reconstruction filter of pixel  $i, j$  is simply the indicator function of the pixel  $[(i-1)w_x, iw_x) \times [(j-1)w_y, jw_y)$ .

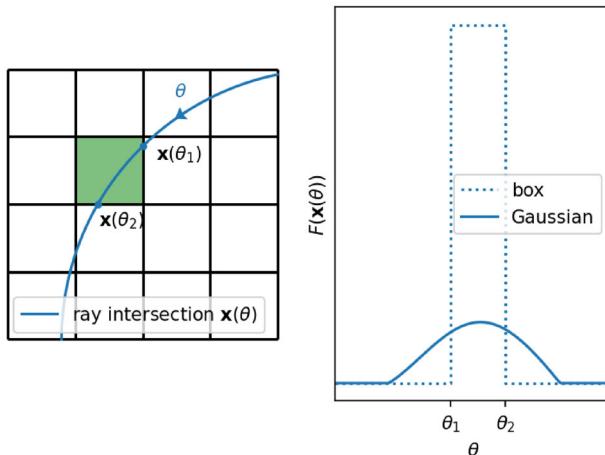
To obtain a ray tracing implementation where the irradiance  $\mathcal{I}$  is differentiable with respect to geometry parameters of the lens, say, the parameter  $\theta$ , the irradiance distribution must vary smoothly with this parameter. The dependency on this parameter is carried from the lens to the screen by the rays through the screen intersections  $\mathbf{x}_k = \mathbf{x}_k(\theta)$ . Thus to obtain a useful gradient  $\frac{\partial \mathcal{I}}{\partial \theta}$  the filter function  $F_{i,j}$  should be at least  $C^1$ , see Fig. 9 which is achieved by introducing a filter function that spreads out the contribution of a ray over a kernel of pixels of size  $(v_x, v_y)$  centered at the intersection location. For the conservation of light, we require that  $\sum_{i,j} F_{i,j}(\mathbf{x}) \equiv 1$ .

Therefore, the Gaussian reconstruction function is introduced, based on the identically named one described in Pharr et al. (2016, Sect. 7.8.1). This filter function is based on the product

$$\tilde{F}_{i,j}(x, y; \alpha, v_x, v_y) := f_i^x(x; \alpha, v_x) f_j^y(y; \alpha, v_y), \quad (38)$$

where

$$f_{i_0}^x(x; \alpha, v_x) = \begin{cases} e^{-\alpha(x - c_{i_0}^x)^2} - e^{-\alpha(\frac{v_x w_x}{2})^2} & \text{if } |x - c_i^x| < \frac{v_x w_x}{2}, \\ 0 & \text{otherwise.} \end{cases} \quad (39)$$



**Fig. 9**  $\mathbf{x}(\theta)$  in the left plot shows the intersection location of a ray with the screen, dependent on a lens geometry parameter  $\theta$ . The right plot then shows the reconstruction filter value for the green pixel in the left plot dependent on  $\theta$ . In order to obtain a useful gradient of the pixel value with respect to  $\theta$ , a smooth reconstruction filter is needed

The centers of the pixels are given by

$$(c_i^x, c_j^y) := \left( \left( i + \frac{1}{2} \right) w_x - R_x, \left( j + \frac{1}{2} \right) w_y - R_y \right). \quad (40)$$

Note that the support of  $\tilde{F}_{i,j}$  is of size  $v_x w_x$  by  $v_y w_y$ , the size of the kernel on the detector screen. The normalized reconstruction filter is then given by

$$F_{i,j}(x, y; \alpha, v_x, v_y) = \frac{\tilde{F}_{i,j}(x, y; \alpha, v_x, v_y)}{\sum_{i',j'} \tilde{F}_{i',j'}(x, y; \alpha, v_x, v_y)}. \quad (41)$$

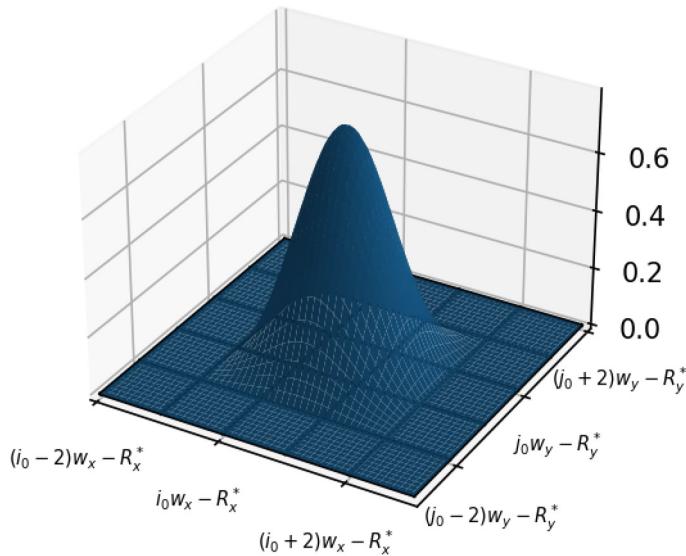
The function  $F_{i,j}$  is plotted in Fig. 10. Note that the function is not differentiable at the boundary of its support. However, in our numerical experiments we have not observed any problems in the optimization loop.

Gaussian image reconstruction is shown in Fig. 11 for various values of  $v_x = v_y$ . There is a trade-off here since the larger  $v_x$ , and  $v_y$  are the blurrier the resulting image is, and the larger the computational graph becomes, but also the larger the section of the image is that is aware of a particular ray which yields more informative gradients.

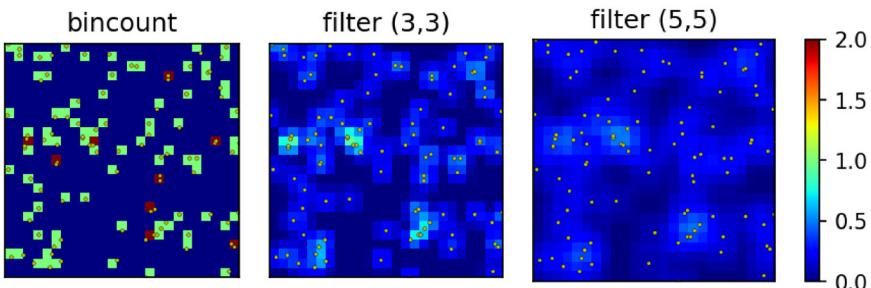
Up to this point, this section has discussed the ray tracing part of the pipeline, the next subsections will discuss the role of the neural network and the optimization.

## 2.4 Multi-layer perceptron as optimization accelerator

Several neural network architectures are considered, all with a trivial input of 1, meaning that the neural networks will not, strictly speaking, be used to approximate a



**Fig. 10** Gaussian reconstruction filter  $F_{i_0, j_0}$  for  $\alpha = 1$  and  $(v_x, v_y) = (3, 3)$



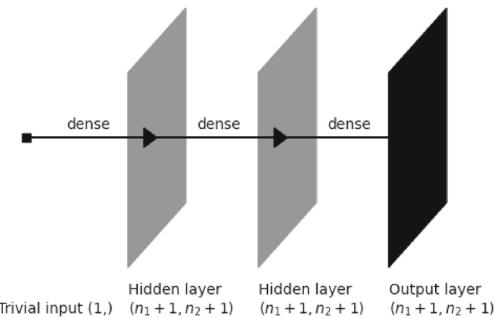
**Fig. 11** Image reconstruction based on a small set of ray-screen intersections, for bincount and various reconstruction filter sizes and  $\alpha = 1$

function since the considered domain is trivial. Non-trivial network inputs of system parameters like the source location will probably be part of follow-up research.

In this configuration, the neural network can be considered a transformation of the space over which is optimized: from the space of trainable neural network parameters to the space of control point  $z$ -coordinate values. The goal of choosing the network architecture is that optimizing the trainable neural network parameters of this architecture yields better training behavior than optimizing the control point  $z$ -coordinate values directly. The used networks are multi-layer perceptrons (MLPs), feed-forward networks consisting of several layers of neurons, as depicted in Fig. 12. The considered architectures are:

1. No network at all.
2. A sparse MLP where the sparsity structure is informed by the overlap of the B-spline basis function supports on the knot spans. In other words: this architecture aims

**Fig. 12** The dense multi-layer perceptron architecture based on the size of the control net  $(n_1 + 1) \times (n_2 + 1)$



to precisely let those control points ‘communicate’ within the network that share influence on some knot span product on the B-spline surface, yielding a layer with the same connectivity as a convolutional layer with kernel size  $(2p + 1, 2q + 1)$ . However, each connection has its own weight and each kernel its own bias, instead of only having a weight per element of the convolution kernel and one single bias for all kernels.

3. Larger fully connected architectures are also considered, with 3 layers of control net size. Note that two consecutive such layers yield many weight parameters:  $n^4$  for a square control net with ‘side length’  $n$ .

The hyperbolic tangent was chosen as the activation function for all neurons because it flattens out when the input values are high. This enables it to restrict the control points’ movement to a specific domain. This is further discussed in Sect. 2.4.1.

#### 2.4.1 Control point freedom

Control over the range of values that can be assumed by the control point  $z$ -coordinates is essential to make sure that the system stays physical (as mentioned in Sect. 2.1.2), but also to be able to take into account restrictions imposed on the lens as part of mechanical construction in a real-world application. Note that the restriction  $P_{i,j}^z > z_{\text{in}}$  for the control points being above the lens entrance surface is not critical for a collimated beam simulation since, the entrance surface can be moved arbitrarily to the  $-z$  direction without affecting the ray tracing.

Since the final activation function  $\tanh$  has a finite range  $(-1, 1)$ , this can easily be mapped to a desired interval  $(z_{\text{min}}, z_{\text{max}})$ :

$$y_{i,j} \mapsto z_{\text{min}} + \frac{1}{2}(y_{i,j} + 1)(z_{\text{max}} - z_{\text{min}}), \quad (42)$$

which can even vary per control point if desired. Here  $y_{i,j}$  denotes an element of the total output  $Y$  of the network. The above can also be used as an offset from certain fixed values:

$$y_{i,j} \mapsto f(P_{i,j}^x, P_{i,j}^y) + z_{\text{min}} + \frac{1}{2}(y_{i,j} + 1)(z_{\text{max}} - z_{\text{min}}). \quad (43)$$

The resulting B-spline surface approximates the surface given by  $f(x, y) + \frac{1}{2}(z_{\text{max}} + z_{\text{min}})$  if  $Y \approx 0$  can be used to optimize a lens that is globally at least approximately

convex/concave. The choice of the hyperbolic tangent activation function accommodates this: since this activation function is smooth around its fixed point 0 when initializing the weights and biases of the network close to 0, there is no cumulative value-increasing effect in a forward pass through the network so that indeed  $Y \approx 0$  in this case.

For comparability, in the case without a network, the optimization is not performed directly on the control point  $z$ -coordinates. Instead, for each control point, a new variable for optimization is created, which is passed through the activation function and the correction as in Eqs. (42) or (43) before being assigned to the control point.

## 2.5 The optimization

The lens is optimized such that the irradiance distribution  $\mathcal{I}$  projected by the lens approximates a reference image  $\mathcal{I}_{\text{ref}}$ , where  $\mathcal{I}, \mathcal{I}_{\text{ref}} \in \mathbb{R}_{\geq 0}^{n_x \times n_y}$ . The loss function used to calculate the difference between the two uses the normalized matrices:

$$\widehat{\mathcal{I}} = \frac{\mathcal{I}}{\sum_{i,j}^{n_x, n_y} \mathcal{I}_{i,j}} \quad \text{and} \quad \widehat{\mathcal{I}}_{\text{ref}} = \frac{\mathcal{I}_{\text{ref}}}{\sum_{i,j}^{n_x, n_y} \mathcal{I}_{\text{ref},i,j}}. \quad (44)$$

The loss function is given by

$$\mathcal{L}(\mathcal{I}; \mathcal{I}_{\text{ref}}) = \frac{1}{\sqrt{n_x n_y}} \|\widehat{\mathcal{I}} - \widehat{\mathcal{I}}_{\text{ref}}\|_F, \quad (45)$$

where  $\|\cdot\|_F$  is the Frobenius matrix norm, which is calculated as follows:

$$\|\mathcal{A}\|_F = \sqrt{\sum_i^{n_x} \sum_j^{n_y} |a_{i,j}|^2}. \quad (46)$$

Figure 1 shows the conventional stopping criterion of the loss value being smaller than some  $\varepsilon > 0$ , but in our experiments, we use a fixed number of iterations.

The neural network parameters (weights and biases) are updated using the Adam optimizer (Kingma and Ba 2014) by back-propagation of the loss to these parameters.

## 3 Results

Several results produced with the optimization pipeline discussed in the previous sections are displayed and discussed in this section. The implementation mainly uses PyTorch, a Python wrapper of Torch (Collobert et al. 2002) and run on a HP ZBook Power G7 Mobile Workstation with a NVIDIA Quadro T1000 with Max-Q Design GPU.

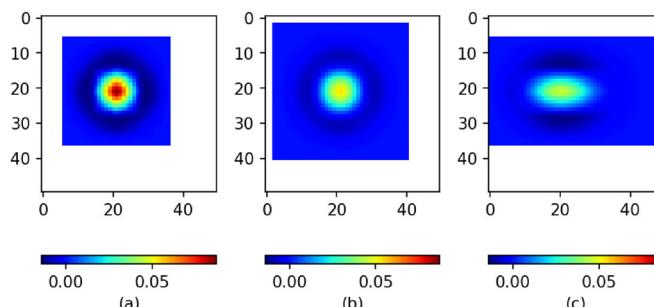
Most of the results have been validated with *LightTools* (Synopsis). Lens designs were imported as a point cloud, then interpolated to obtain a continuous surface, and all simulations were conducted using  $10^6$  rays.

Units of length are mostly unspecified since the obtained irradiance distributions are invariant under uniform scaling of the optical system. This invariance to scaling is reasonable as long as the lens details are orders of magnitude larger than the wavelength of the incident light such that diffraction effects do not play a role. Furthermore, the irradiance distributions are directly proportional to the scaling of all ray weights and thus the source power, so the source and screen power also need no unit specification. Note that relative changes have a non-trivial effect, like changes to the power proportion between sources or the distance proportions of the optical system.

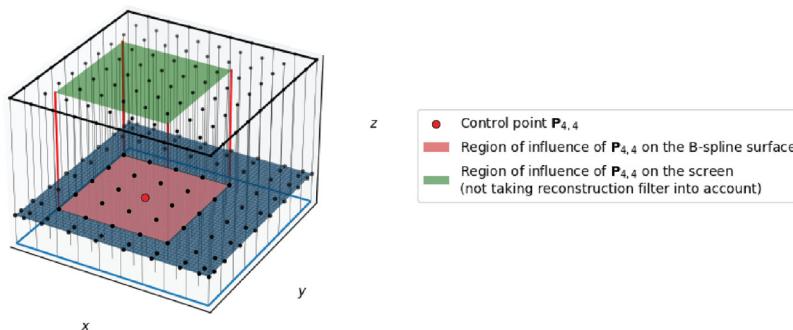
### 3.1 Irradiance derivatives with respect to a control point

This section gives a simple first look at the capabilities of the implemented differentiable ray tracer: computing the derivative of an irradiance distribution with respect to a single control point. Figure 13 shows the derivative of an irradiance distribution produced by a collimated ray bundle through a flat lens for various B-spline degrees and reconstruction filter sizes, and Fig. 14 shows what one of these systems looks like. The overall ‘mountain with a surrounding valley’ structure can be understood as follows: as one of the control points rises, it creates a local convexity in the otherwise flat surface. This convexity has a focusing effect, redirecting light from the negative valley region toward the positive mountain region.

Noteworthy of these irradiance derivatives is also their total sum: (a)  $-1.8161 \times 10^{-8}$ , (b)  $3.4459 \times 10^{-8}$ , (c)  $9.7095 \times 10^{-5}$ . These are small numbers with respect to the total irradiance of about 93 and therefore indicate conservation of light; as the control point moves out of the flat configuration, at first, the total amount of power received by the screen will not change much. This is expected from cases (a) and (b), where the control point does not affect rays that reach the screen on the boundary pixels. However, in all cases, all rays intersect the lens at right angles. Around  $\theta = 0$ , the slope of Schlick’s approximation is very shallow, indicating a small decrease in refraction in favor of reflection.



**Fig. 13** Gradients of an irradiance distribution of a collimated beam through a flat lens (parallel sides), with respect to the  $z$ -coordinate of one control point. The zeros are masked with white to show the extend of the influence of the control point. These irradiance distributions differ by: **a** degrees (3, 3), reconstruction filter size (3, 3), **b** degrees (3, 3), reconstruction filter size (11, 11), **c** degrees (5, 3), reconstruction filter size (3, 3)



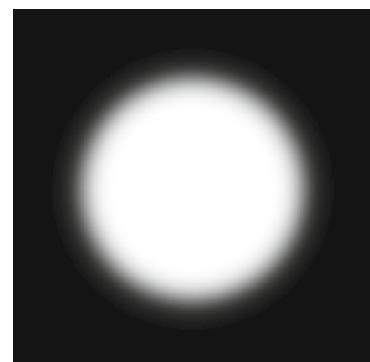
**Fig. 14** Demonstration of how one control point influences the irradiance distribution in the case of a flat lens with B-spline degrees (3, 3) and a collimated beam

### 3.2 Sensitivity of the optimization to initial state and neural network architecture

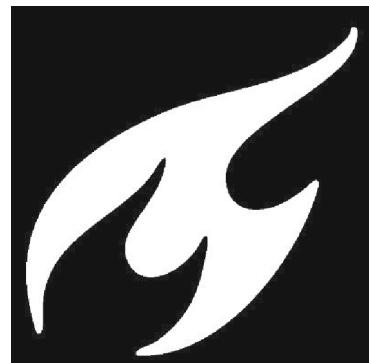
As with almost any iterative optimization procedure, choosing a reasonable initial guess of the solution is crucial for reaching a good local/global minimum. For training neural networks, this comes down to how the network weights and biases are initiated. In this section, we look at three target illuminations: the circular top hat distribution (Fig. 15), the TU Delft logo (Fig. 16), and an image of a faceted ball (Fig. 17). We design lenses to produce these distributions from a collimated ray beam, given various neural network architectures (introduced in Sect. 2.4) and parameter initializations.

*Circular top hat distribution from collimated ray beam* Figure 18 shows the progress of the loss over 1000 iterations, with each iteration taking 2.5 s. Learning rates of  $10^{-3}$ ,  $10^{-4}$  and  $10^{-5}$  were used for different network architectures: random sparse (RS), no network (NN), uniform sparse (UN), and uniform dense (UD). In this context, uniform implies that the initial trainable parameter values are sampled from a small interval:  $U([-10^{-4}, 10^{-4}])$ , except for the no-network case; this is initialized with all zeros. Figures 19, 20, 21, 22 and 23 depict the freeform surfaces and irradiance distributions obtained throughout optimization using a learning rate of  $10^{-4}$ . More information about the initial parameters used in these simulations can be found in Table S1 in the supplementary materials. The figures depicting the irradiance distributions and

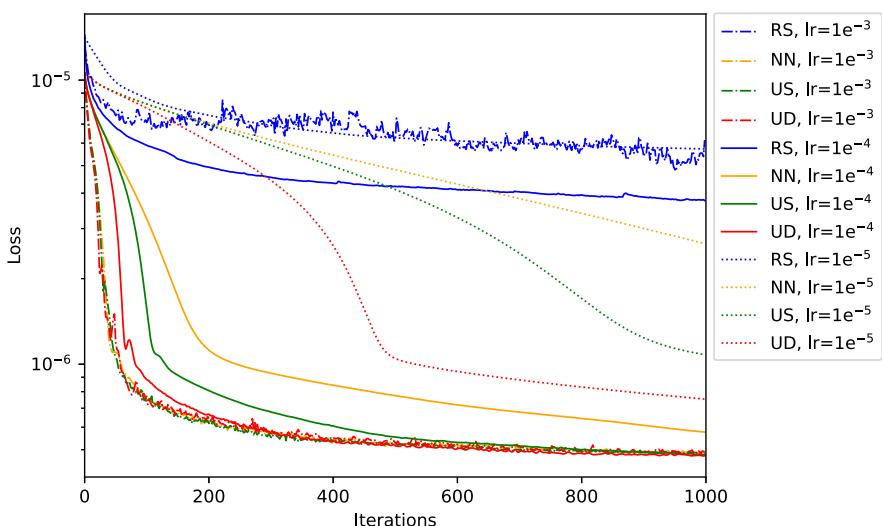
**Fig. 15** The circular top hat target illumination



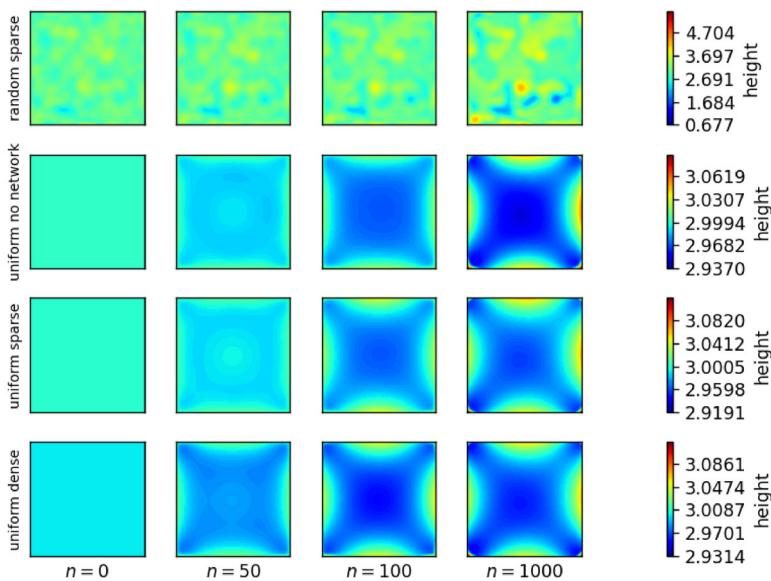
**Fig. 16** The TU Delft flame target illumination



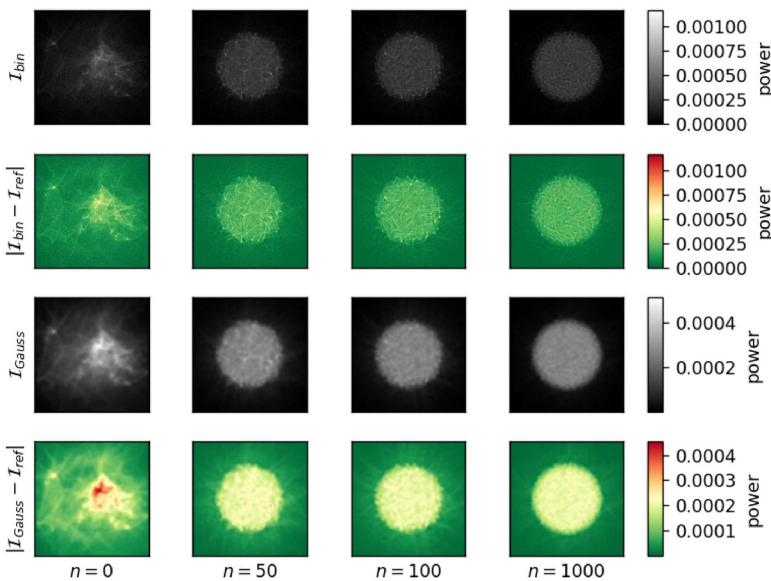
**Fig. 17** The faceted ball target illumination



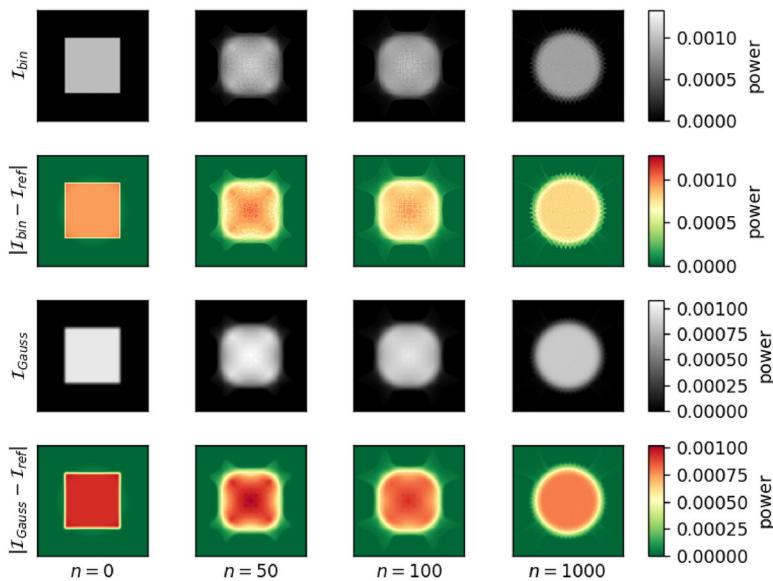
**Fig. 18** Loss progress over the iterations for the pipeline-setups: random sparsely connected (RS), no network (NN), uniform sparsely connected (US), and uniform densely connected (UD) using learning rates  $10^{-3}$ ,  $10^{-4}$ , and  $10^{-5}$  for forming a top hat distribution from a collimated beam



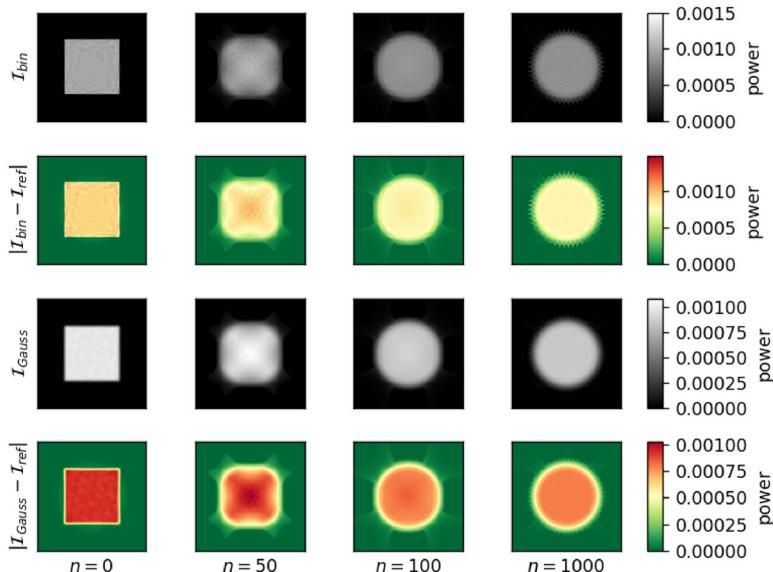
**Fig. 19** The lens height field after initialization ( $n = 0$ ), and  $n = 50, 100$  and  $1000$  iterations respectively, using a learning rate of  $10^{-4}$  and different network architectures (Sect. 2.4) and network parameter initializations (Sect. 3.2)



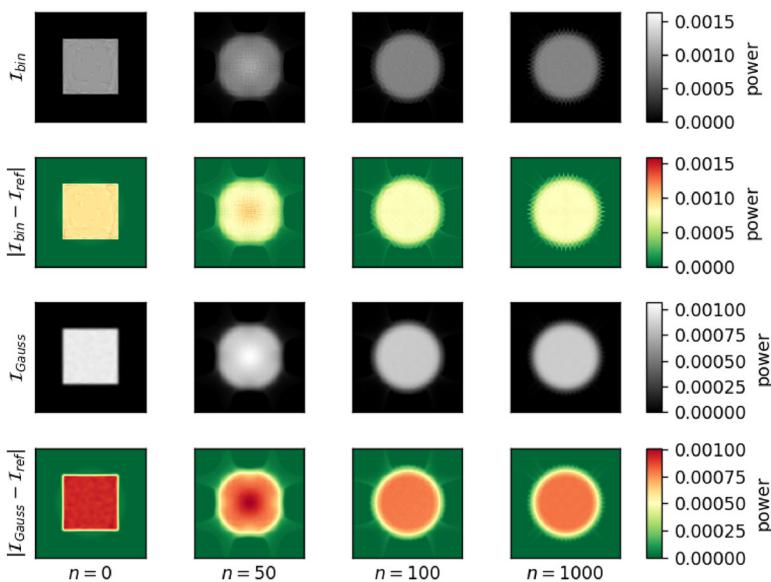
**Fig. 20** Irradiance distributions and pixel-wise errors in the optimization progress of a lens with a random sparse network and a learning rate of  $10^{-4}$  towards a circular top hat illumination



**Fig. 21** Irradiance distributions and pixel-wise errors in the optimization progress of a flat lens without a network and a learning rate of  $10^{-4}$  towards a circular top hat illumination



**Fig. 22** Irradiance distributions and pixel-wise errors in the optimization progress of a flat lens with a sparse network and a learning rate of  $10^{-4}$  towards a circular top hat illumination



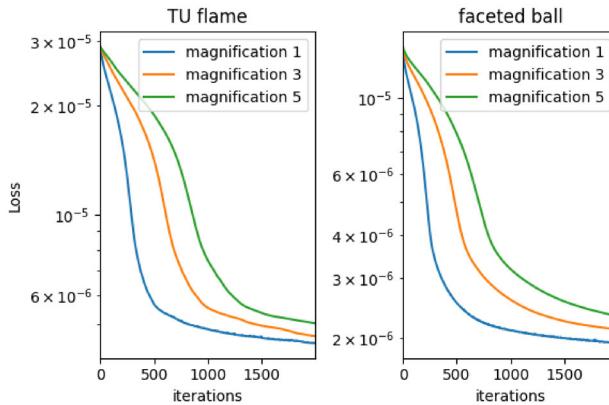
**Fig. 23** Irradiance distributions and pixel-wise errors in the optimization progress of a flat lens with a dense network and a learning rate of  $10^{-4}$  towards a circular top hat illumination

surfaces for the other learning rates can be found in Figs. 1–12 of the supplementary materials.

The first notable difference between randomly and uniformly initialized sparse neural networks is that the uniformly initialized converge loss values are much lower than the randomly initialized, regardless of the chosen learning rate. There are two possible explanations for why this is the case. First, due to the random initialization of the network, the lens is also randomly initialized, leading the optimizer to converge to a different local minimum than the uniform initialized networks, as shown in Fig. 19. Second, as the parameter values are randomly initiated, the influence of certain nodes on certain control points might be unbalanced, resulting in some control points having a much higher contribution throughout the optimization than other control points.

When comparing the different learning rates of the optimizations done using no network, US, and UD, we see that when using a learning rate of  $10^{-3}$ , all three cases converge to a loss of  $5 \times 10^{-7}$  in roughly 600 iterations as shown in Fig. 18. At a learning rate of  $10^{-4}$ , the uniform dense network initially decreases slower but catches up to the other networks using a learning rate of  $10^{-3}$  at around 400 iterations, eventually converging to a similar loss value of  $5 \times 10^{-7}$ . While the uniform sparse network takes longer to reach the same loss value, it still manages to do so after 1000 iterations. When decreasing the learning rate to  $10^{-5}$ , none of the cases fully converge. However, we observe the same behavior of the uniform dense network, converging faster than the uniform sparse network, which again converges faster than the no networks case.

Another property of the uniformly initialized cases is their preservation of symmetry in these setups. As Fig. 19 shows, this leads to much simpler lenses, which are likely less sensitive to manufacturing errors due to their relative lack of small detail. It is

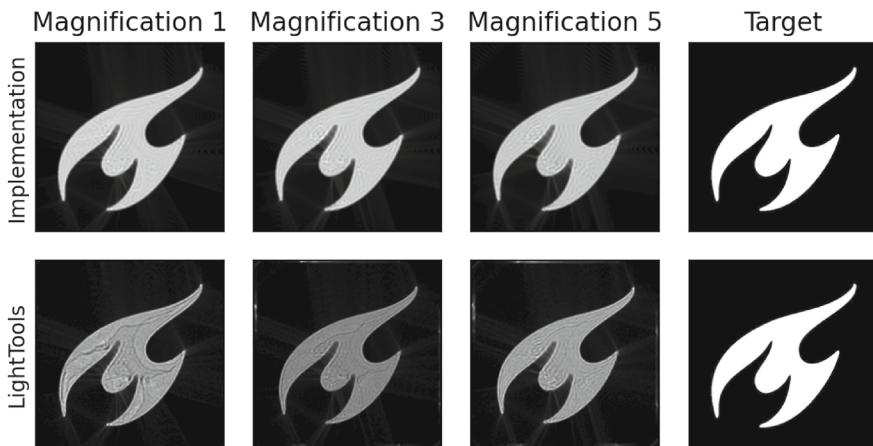


**Fig. 24** Loss progress for the various magnifications and target distributions

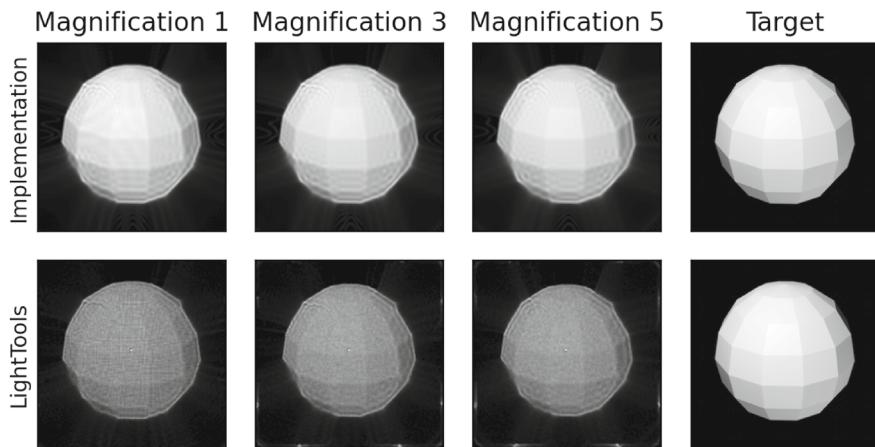
interesting to note that if the sparse network is initialized with all parameters set to 0, the result is identical to the no-network case, as only the biases in the last layer achieve non-zero gradients.

We also repeated the optimization for a different target irradiance distribution: the flame of the TU Delft logo (Fig. 16), the results of which can be seen in Figs. 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 and 29 of the supplementary materials. Despite the more complex target, we observed the same convergence behavior for all the network types.

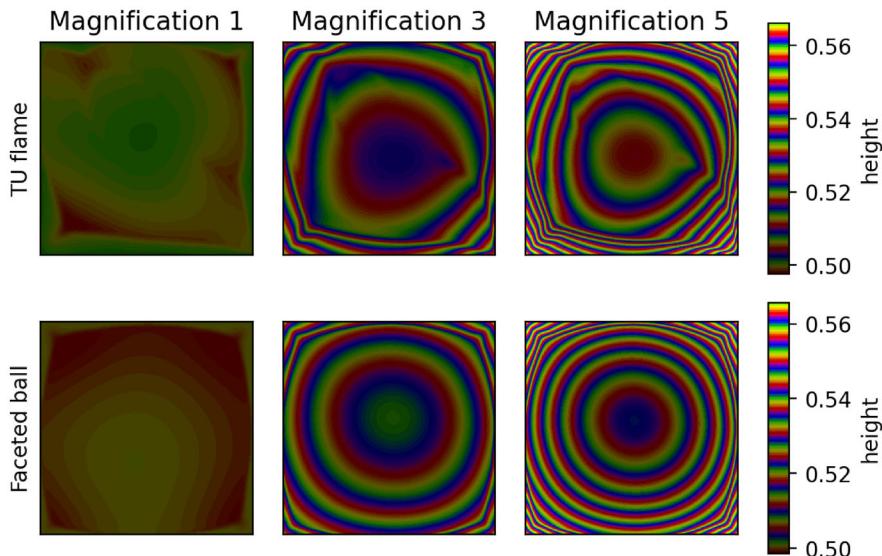
No rigorous investigation has been conducted to the extent that this behavior of increased convergence speed carries over to other target distributions and system configurations and what the optimal hyper-parameters are. A thorough investigation of the hyper-parameter space that defines a family of network architectures could reveal where in the increase of the architecture complexity, diminishing returns for optimiz-



**Fig. 25** Implementation and LightTools irradiance distributions of the TU flame target from the final lens design of the optimization



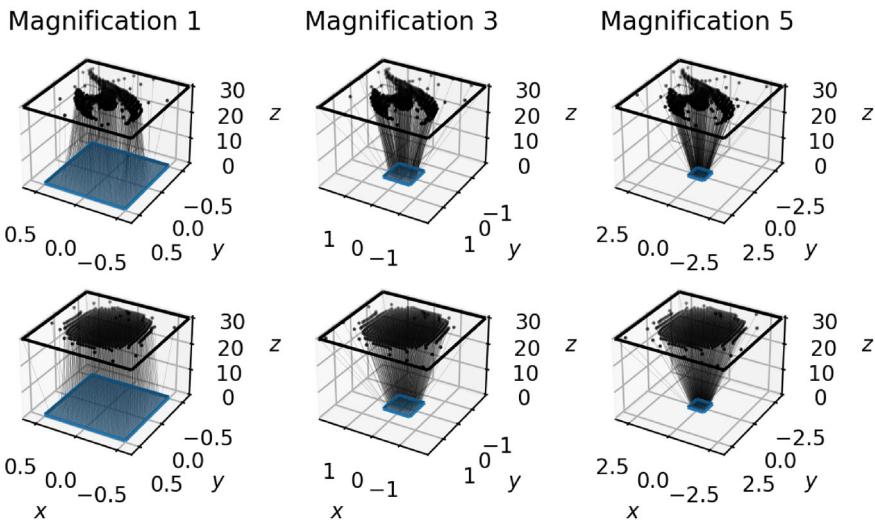
**Fig. 26** Implementation and LightTools irradiance distributions of the faceted ball target from the final lens design of the optimization



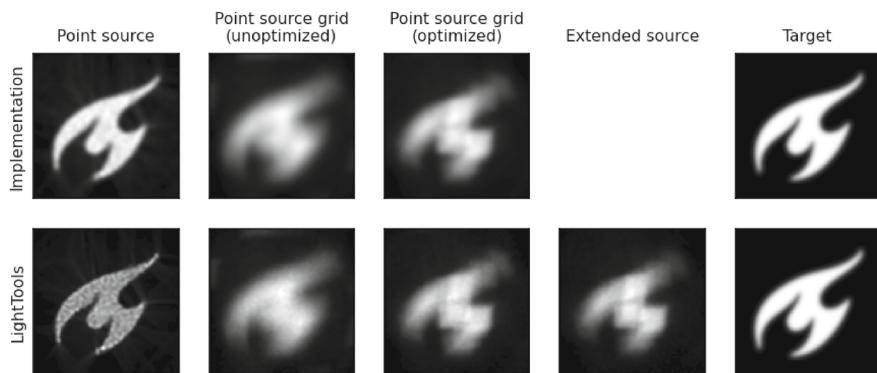
**Fig. 27** The lens designs for the different magnifications and two target distributions

ing these lenses arises. However, based on these initial findings the uniform dense network is used for all the following optimizations in the results.

*TU flame and faceted ball from collimated beam* In what follows, we consider complex target distributions: the TU Delft flame (for a complex shape) and a faceted ball (for a target with various brightness levels). Here we still use the collimated beam, but lenses are now optimized for various magnifications; see Table 1. These magnifications are defined as the scaling of the screen size with respect to the smallest screen size (0.64, 0.64). Both optimizations ran for 2000 iterations with a constant learning rate of



**Fig. 28**  $25 \times 25$  traced rays through the final lens designs for the different magnifications and two target distributions



**Fig. 29** The final irradiation distribution of the lens optimizations with point sources and the corresponding LightTools validations. The extended source is not implemented in our ray tracer, but is approximated by the point source grid, the difference in irradiance is checked in LightTools Fig. 30 in the supplementary material

**Table 1** The screen size and control point offset function  $f$  used per magnification in the TU flame and faceted ball optimizations (distances in centimeters)

Magnification	Screen size	$f(x, y)$	Starting shape type
1	(0.64, 0.64)	$\frac{1}{2}$	Flat
3	(1.92, 1.92)	$\frac{1}{2} + 8 - \sqrt{8^2 - x^2 - y^2}$	Concave
5	(3.20, 3.20)	$\frac{1}{2} + 4 - \sqrt{4^2 - x^2 - y^2}$	Concave

$10^{-5}$ , each iteration took about 4 s each. The other parameters of these optimizations are shown in Table S2 of the supplementary materials.

The final irradiance distributions and corresponding LightTools results are shown in Figs. 25 and 26, respectively. These figures show that the optimization pipeline can handle these more complex target illuminations well. The validation in LightTools shows some artifacts in the obtained irradiance. Which are most notable in the TU flame magnification 1 case (Fig. 25), where lines can be seen flowing through the distribution. However, these artifacts are absent in the distribution obtained using the differentiable ray tracer. The cause of this difference lies in the Gaussian ray blurring, which filters out these detailed artifacts, resulting in a much more uniform result. By visual inspection, based on the LightTools results, one would probably rate these results in the exact opposite order than as indicated by the losses shown in Fig. 24.

A potential explanation of the increase in loss with the magnification factor in Fig. 24 is that the bigger the screen is: the rays require higher angles to reach the edges of the screen, which is apparent in the cases of magnification 3 and 5 Fig. 28. This results in a larger sensitivity of the irradiance to the angle with which a ray leaves the screen. This in turn gives larger gradients of the irradiance with respect to the control points. Therefore the optimization takes larger steps in the neural network parameter space, possibly overshooting points that result in a lower loss.

For the magnification, 3 and 5, the irradiance distributions from LightTools show artifacts at the screen boundaries. A possible explanation for this is that the way the B-spline surfaces are transferred to LightTools is inaccurate at the surface boundaries.<sup>1</sup> This is because surface normals are inferred from fewer points on the B-spline surface at the boundary than in the middle of the surface by LightTools.

Furthermore, a significant amount of rays are lost during optimization because the target illuminations are black at the borders, so rays near the screen boundary will be forced off the screen by the optimization. Once rays are off the screen, they no longer contribute to the loss function. Once a ray misses the screen, the patch on the B-spline surface these rays originate from does not influence the irradiance and, thus, the loss function. However, this does not mean that this patch is idle for the rest of the optimization, as this patch can be in support of a basis function that corresponds to a control point that still affects rays that hit the screen. Therefore, the probability of getting idle lens patches with this setup decreases with the B-spline degrees since these determine the size of the support of the B-spline basis functions but might, in some cases, lead to oscillatory behavior, with rays alternating between hitting and missing the screen.

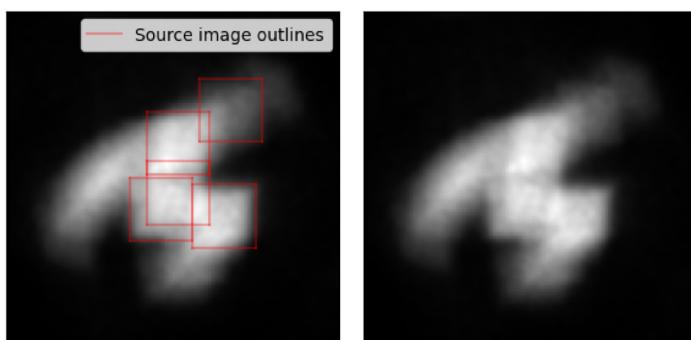
Figure 27 shows the optimized B-spline lens surface height field. A densely varying color map is chosen since the deviations from a flat or smooth concave shape are quite subtle, which is due to the large lens exit angle sensitivity of the ray-screen intersections since the ratio lens size to screen size is large with respect to the ratio lens size to screen distance.

<sup>1</sup> Assuming only rays from the B-spline surface boundaries reach the screen boundary area.

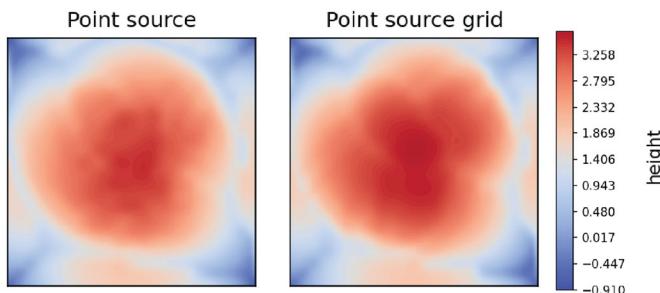
### 3.3 Optimization with a point source and a grid of point sources

We now consider an optimization that uses the B-spline intersection algorithm. First, we design a lens with one point source at  $(0, 0, -5)$  with  $5 \times 10^5$  rays to form the TU flame. After 200 iterations, we change the source to an equispaced grid of  $25 \times 25$  point sources with  $10^3$  rays each on  $[-1, 1] \times [-1, 1] \times \{-5\}$ , approximating a source of non-negligible size. The learning rate was kept constant at  $10^{-3}$  throughout the whole optimization procedure. The target irradiance is blurred using a Gaussian kernel, as creating sharp irradiance distributions is very challenging for finite etendue sources. The other (hyper-) parameters of this optimization can be found in Table S3 of the supplementary materials. Due to the additional B-spline intersection procedures, each iteration takes approximately 50 s. The resulting final irradiance distribution and LightTools validation can be seen in Fig. 29. The final irradiance distribution is similar to that obtained by LightTools, indicating that ray tracing with the implemented B-spline intersection algorithm works correctly. The irradiance is blurred due to the reconstruction filter. The single-source point optimization performs well, although the illumination is less uniform than in the collimated beam case (Figs. 25 and 26). The non-uniformity can be attributed to the Gaussian reconstruction filter used during optimization, as it smoothes out the small uniformities.

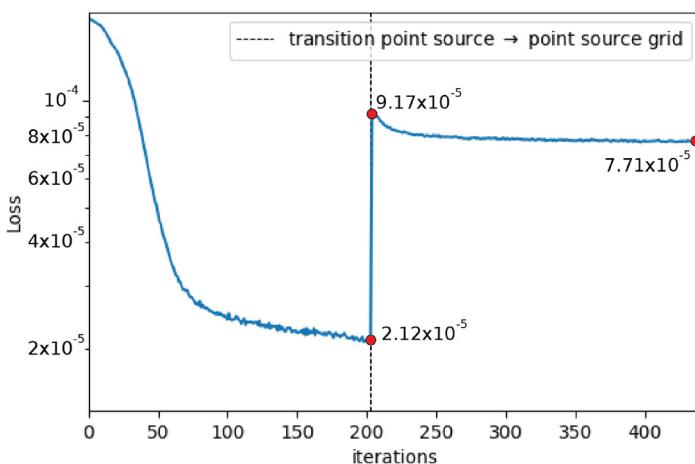
Finding a lens design that redirects light from a source of non-negligible size into a desired irradiance distribution is a complex problem for which it is hard to indicate how good the optimal irradiance distribution can become. The progress of the loss, as seen in Fig. 32, shows that the optimization can still improve the loss, even after the transition to the grid of point sources. Interestingly, looking at Fig. 29 again, the optimization seems to adopt the coarse strategy of filling up the target distribution with images of the source square, as shown in Fig. 30. This strategy does hinder the possible quality of the final irradiance distribution as the image of the source on the target is larger than the fine details in the desired irradiance. This issue can potentially be resolved by optimizing both the front and back surfaces of the freeform, as this will cause the image of the source to change shape depending on where it ends up on the target screen (Figs. 31 and 32).



**Fig. 30** Indication of images of the source square in the irradiance distribution obtained by LightTools using the point source grid



**Fig. 31** Height fields of the lenses optimized for the TU flame with point sources



**Fig. 32** Loss over the iterations optimizing for the TU flame. The system is initiated with a point source, and after  $\sim 200$  iterations the point source is replaced by an equispaced grid of  $25 \times 25$  point sources

## 4 Conclusion

We demonstrated that non-sequential differentiable ray tracing is a viable tool for designing freeform lenses for collimated beams, points, and extended sources. Using a B-spline allows for the design of a continuous surface, which is desirable for manufacturing, and its control point allows for locally altering the irradiance distribution. For both cases, collimated and point source lens designs were found that could accurately project the desired irradiance distribution in both the differentiable ray tracer and in commercial software LightTools. To address the discrepancies between the differentiable ray tracer results and the LightTools validations, which were most apparent when optimizing for a collimated beam, the ray blur can gradually be decreased during optimization. This approach can maintain stable gradient calculation and also allow the discrepancies to be detected during optimization.

For the source with a finite extent, the optimizer improved upon the design obtained for a point source. However, the final irradiance distribution was made up of images of the source, which hinders the minimum that can be obtained as the source image is

larger than the details in the desired irradiance distribution. This issue can be resolved by optimizing multiple surfaces simultaneously, as the image of the source on the target plane can then be optimized to vary with location.

Using a neural network to remap the optimization space provides an interesting way to increase the convergence speed of the optimization. However, further investigation is required to see whether this generally holds and what the effect is on other network architectures. Furthermore, the implementation transfer learning can transfer knowledge from past optimizations to new optimization scenarios, potentially reducing the time required for optimization.

The developed ray tracing implementation is currently a proof of concept and needs to be optimized for speed. The B-spline intersection algorithm, in particular, adds roughly a factor of 10 to the computation time. A significant speedup can be achieved here by leveraging efficient lower-level GPU programming languages, such as CUDA.

**Supplementary Information** The online version contains supplementary material available at <https://doi.org/10.1007/s11081-023-09841-9>.

**Acknowledgements** We acknowledge support by NWO-TTW Perspectief program (P15-36) “Free-form scattering optics”.

**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

Ansys (2023) Zemax. <https://www.zemax.com/>

Anthonissen MJH, Romijn LB, ten Thije Boonkamp JHM, IJzerman WL (2021) Unified mathematical framework for a class of fundamental freeform optical systems. *Opt Express* 29:31650. <https://doi.org/10.1364/oe.438920>

Bösel C, Gross H (2018) Double freeform illumination design for prescribed wavefronts and irradiances. *JOSA A* 35(2):236–243

Chen YB, Lin PD (2012) Second-order derivatives of optical path length of ray with respect to variable vector of source ray. *Appl Opt* 51(22):5552. <https://doi.org/10.1364/AO.51.005552>

Cohen E, Martin T, Kirby RM, Lyche T, Riesenfeld RF (2010) Analysis-aware modeling: understanding quality considerations in modeling for isogeometric analysis. *Comput Methods Appl Mech Eng* 199:334–356. <https://doi.org/10.1016/j.cma.2009.09.010>

Collobert R, Bengio S, Mariéthoz J (2002) Torch: a modular machine learning software library. Technical report, Idiap

Doskolovich LL, Bykov DA, Andreev ES, Bezus EA, Oliker V (2018) Designing double freeform surfaces for collimated beam shaping with optimal mass transportation and linear assignment problems. *Opt Express* 26(19):24602–24613

Falaggis K, Rolland J, Duerr F, Sohn A (2022) Freeform optics: introduction. *Opt Express* 30:6450. <https://doi.org/10.1364/oe.454788>

Farin G (2002) Curves and surfaces for CAGD, 5th edn. Morgan Kaufmann Publishers, Burlington

Feder DP (1968) Differentiation of ray-tracing equations with respect to construction parameters of rotationally symmetric optics. *JOSA* 58(11):1494–1505

Feng Z, Huang L, Jin G, Gong M (2013) Designing double freeform optical surfaces for controlling both irradiance and wavefront. *Opt Express* 21(23):28693–28701

Fournier FR, Cassarly WJ, Rolland JP (2009) Designing freeform reflectors for extended sources. In: Nonimaging optics: efficient design for illumination and solar concentration VI, Volume 7423. SPIE, pp 11–22

Fournier FR, Cassarly WJ, Rolland JP (2010) Fast freeform reflector generation using source-target maps. *Opt Express* 18(5):5295–5304

Fowles GR (1989) Introduction to modern optics. Courier Corporation

Gasick J, Qian X (2023) Isogeometric neural networks: a new deep learning approach for solving parameterized partial differential equations. *Comput Methods Appl Mech Eng* 405:115839. <https://doi.org/10.1016/j.cma.2022.115839>

Givoli D (2021) A tutorial on the adjoint method for inverse problems. *Comput Methods Appl Mech Eng* 380:113810. <https://doi.org/10.1016/j.cma.2021.113810>

Grant BG (2011) Field guide to radiometry. SPIE, Bellingham

Jakob, W, Speirer S, Roussel N, Vicini D (2022). DR. JIT: a just-in-time compiler for differentiable rendering. *ACM Trans Graph (TOG)* 41(4):1–19

John RK (2013) Illumination engineering: design with nonimaging optics. Wiley, Hoboken

Karniadakis GE, Kevrekidis IG, Lu L, Perdikaris P, Wang S, Yang L (2021) Physics-informed machine learning. *Nat Rev Phys* 3:422–440. <https://doi.org/10.1038/s42254-021-00314-5>

Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. [arXiv:1412.6980](https://arxiv.org/abs/1412.6980)

Kronberg VC, Anthonissen MJ, ten Thije Boonkamp JH, IJzerman WL (2023) Two-dimensional freeform reflector design with a scattering surface. *JOSA A* 40(4):661–675

Lippman DH, Schmidt GR (2020) Prescribed irradiance distributions with freeform gradient-index optics. *Opt Express* 28:29132–29147. <https://doi.org/10.1364/OE.404456>

ltoptics (2023) Photopia. <https://www.ltoptics.com/en/optical-design-software-photopia.html>

Mohedano R, Chaves J, Hernández M (2016) Free-form illumination optics. *Adv Opt Technol* 5:177–186. <https://doi.org/10.1515/aot-2016-0006>

Möller M, Toshniwal D, van Ruiten F (2021) Physics-informed machine learning embedded into isogeometric analysis. In: Mathematics: key enabling technology for scientific machine learning. Platform Wiskunde, Amsterdam, pp 57–59

Muschawec JA (2022) Tailored freeform surfaces for illumination with extended sources. In: SPIE-International Society Optical Engineering. Presented at SPIE Optical Engineering + Applications, San Diego, California, 3 October 2022, p 9

Nie Y, Zhang J, Su R, Ottevaere H (2023) Freeform optical system design with differentiable three-dimensional ray tracing and unsupervised learning. *Opt Express* 31(5):7450–7465

Nimier-David M, Vicini D, Zeltner T, Jakob W (2019) Mitsuba 2: a retargetable forward and inverse renderer. *ACM Trans Graph* 38:1–17. <https://doi.org/10.1145/3355089.3356498>

Oertmann FW (1989) Differential ray tracing formulae; applications especially to aspheric optical systems. In: Optical design methods, applications and large optics, volume 1013, Hamburg, Germany, pp 20–26. SPIE. Presented at 1988 international congress on optical science and engineering, Hamburg, Germany, 13 April 1989

Oliker V, Rubinstein J, Wolansky G (2015) Supporting quadric method in optical design of freeform lenses for illumination control of a collimated light. *Adv Appl Math* 62:160–183

Pharr M, Jakob W, Humphreys G (2016) Physically based rendering: from theory to implementation. Morgan Kaufmann Publishers, Burlington

Piegl L, Tiller W (1996) The NURBS book. Springer, Berlin

Prins C, Beltman R, ten Thije Boonkamp J, IJzerman WL, Tukker TW (2015) A least-squares method for optimal transport using the Monge–Ampere equation. *SIAM J Sci Comput* 37(6):B937–B961

Ries H, Muschawec J (2002) Tailored freeform optical surfaces. *JOSA A* 19(3):590–595

Romijn L (2021) Generated Jacobian equations in freeform optical design: mathematical theory and numerics. Ph.D Thesis, Mathematics and Computer Science. Proefschrift

Schlick C (1994) An inexpensive BRDF model for physically-based rendering. *Comput Graph Forum* 13:233–246. <https://doi.org/10.1111/1467-8659.1330233>

Sorgato S, Chaves J, Thienpont H, Duerr F (2019) Design of illumination optics with extended sources based on wavefront tailoring. *Optica* 6:966–971. <https://doi.org/10.1364/OPTICA.6.000966>

Stone BD, Forbes GW (1997) Differential ray tracing in inhomogeneous media. *J Opt Soc Am A* 14(10):2824. <https://doi.org/10.1364/JOSAA.14.002824>

Sun Q, Wang C, Fu Q, Dun X, Heidrich W (2021) End-to-end complex lens design with differentiate ray tracing. *ACM Trans Graph* 40:1–13. <https://doi.org/10.1145/3450626.3459674>

Synopsis (2023) Lighttools. <https://www.synopsys.com/optical-solutions/lighttools.html>

Synopsis (2023) Code v. <https://www.synopsys.com/optical-solutions/codev.html>

Tukker TW (2007) Efficient collimator design for extended light sources with the flux tube method. In: Presented at SPIE optical engineering + applications, San Diego, California, 18 September 2007

Van Roosmalen A, Anthonissen M, IJzerman W, ten Thije Boonkamp J (2022) Fresnel reflections in inverse freeform lens design. *JOSA A* 39(6):1045–1052

Volatier JB, Álvaro M-F, Erhard M (2017) Generalization of differential ray tracing by automatic differentiation of computational graphs. *J Opt Soc Am A* 34:1146. <https://doi.org/10.1364/josaa.34.001146>

Wang C, Chen N, Heidrich W (2022) dO: A differentiable engine for deep lens design of computational imaging systems. *IEEE Trans Comput Imaging* 8:905–916

Wei S, Zhu Z, Li W, Ma D (2021) Compact freeform illumination optics design by deblurring the response of extended sources. *Opt Lett* 46(11):2770–2773

Wester R, Müller G, Völl A, Berens M, Stollenwerk J, Loosen P (2014) Designing optical free-form surfaces for extended sources. *Opt Express* 22(102):A552–A560

Wu R, Feng Z, Zheng Z, Liang R, Benítez P, Miñano JC, Duerr F (2018) Design of freeform illumination optics. *Laser Photon Rev* 12:1700310. <https://doi.org/10.1002/lpor.201700310>

Wu R, Liu P, Zhang Y, Zheng Z, Li H, Liu X (2013) A mathematical model of the single freeform surface design for collimated beam shaping. *Opt Express* 21(18):20974–20989

Zhu Z, Wei S, Fan Z, Ma D (2022) Freeform illumination optics design for extended led sources through a localized surface control method. *Opt Express* 30(7):11524–11535

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