dO: A Differentiable Engine for Deep Lens Design of Computational Imaging Systems

Congli Wang, Ni Chen, and Wolfgang Heidrich, *Fellow, IEEE*

***Abstract*—Computational imaging systems algorithmically post-process acquisition images either to reveal physical quantities of interest or to increase image quality, e.g., deblurring. Designing a computational imaging system requires co-design of optics and algorithms, and recently Deep Lens systems have been proposed in which both components are end-to-end designed using data-driven end-to-end training. However, progress on this exciting concept has so far been hampered by the lack of differentiable forward simulations for complex optical design spaces. Here, we introduce dO (DiffOptics) to provide derivative insights into the design pipeline to chain variable parameters and their gradients to an error metric through differential ray tracing. However, straightforward back-propagation of many millions of rays requires unaffordable device memory, and is not resolved by prior works. dO alleviates this issue using two customized memory-efficient techniques: differentiable ray-surface intersection and adjoint back-propagation. Broad application examples demonstrate the versatility and flexibility of dO, including classical lens designs in asphere, double-Gauss, and freeform, reverse engineering for metrology, and joint designs of optics-network in computational imaging applications. We believe dO enables a radically new approach to computational imaging system designs and relevant research domains.**

***Index Terms*—End-to-end lens design, image reconstruction, memory-efficient backpropagation, physics-based learning.**

I. Introduction

CAMERAS are designed with a non-trivial tradeoff between image quality (optical aberrations) and practical considerations (constraints, cost, form-factor, fabrication availability). In traditional optical design complex lens assemblies are simulated using ray-tracing and then semi-manually tuned to satisfy the performance requirements. Commercial software for automated lens design is nowadays the norm, such as ZEMAX [1] and Code V [2].

Computational imaging systems extend the capability of conventional imaging pipelines by having an additional degree of freedom in software. The spirit is to customize cameras or imaging modalities, and acquisition images are regarded as optically encoded information about the physical world. Computational methods, e.g., model-based numerical optimization or data-driven machine learning, are applied to raw images and decode the information. Examples are wavefront coding [3] and coded aperture variants [4], [5], [6] to extend depth-of-field or to perceive depth, cameras to capture individual rays (i.e., light fields) [7], [8], and cameras capable of high dynamic ranges [9].

The design of computational imaging systems is particularly challenging in that the design tradeoff happens in both hardware and software. As such the final design typically occupies a very high dimensional design space, that complicates easy and intuitive solutions. One way to explore the design space and evolve the co-design is to do stochastic optimization by gradient-descent, as shown in prior end-to-end trained Deep Lens works [10], [11], [12], [13], [14]. Such an approach requires the derivative of the final image concerning individual optical design parameters. This derivative-aware modeling is non-trivial for complex lens groups, and hence fully differentiable optical models have so far only been available for simplistic design spaces such as a diffractive optical element or a lensless mask, as demonstrated by various application-oriented deep optics designs, for image classification [15], depth estimation [16], [17], [18], lensless imaging [19], HDR acuisitions [20], [21], extended depth-of-field [22], computational micrscopy [23], [24], and for versatile purposes [25], [26]. See also [27], [28] for perspectives. These are a long way from the expressive power of ray-tracing based systems like ZEMAX [1] and Code V [2].

To partially tackle these challenges, derivative-aware lens design engines [29], [30], [31] were inspired by automatic differentiation (AD) [32], a fundamental technique in deep learning. The main distinction of a derivative-aware engine compared to conventional ones is *differentiability:* The availability of derivative information relating design parameters and the error metric. Via differential ray tracing [30], design parameters and their gradients are chained to the error metric through a so-called computational graph, on which back-propagation results in how each design parameter should quantitatively change to reduce the error metric. Together with gradient-based optimization, the obtained derivatives provide a searching direction in the hyper-parameter space to locally guide evolution of current design, improving performance in terms of the error metric. Combined with deep neural networks, such a differentiable engine could be employed for generating lens designs [33], [34] or image restoration [35], [36]. Recent works [13], [14] rely on differentiable ray tracing for end-to-end designs in

Manuscript received 29 December 2021; revised 24 July 2022; accepted 26 September 2022. Date of publication 14 October 2022; date of current version 20 October 2022. This work was supported by individual baseline funding of King Abdullah University of Science and Technology. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Henry Arguello. *(Corresponding author: Wolfgang Heidrich.)*

The authors are with the Visual Computing Center, King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia (e-mail: congli.wang@kaust.edu.sa; ni.chen@kaust.edu.sa; wolfgang.heidrich@kaust.edu.sa).

This article has supplementary downloadable material available at https://doi.org/10.1109/TCI.2022.3212837, provided by the authors.

Digital Object Identifier 10.1109/TCI.2022.3212837

computational imaging. Similar trends also appear in other research domains for solving inverse problems utilizing this additional amount of information on derivatives, in computer graphics [37], 3D reconstruction [38], ptychography [39], [40], head-mounted displays calibration [41], lens metrology [42], and phase microscopy [43].

However, one main challenge of direct applying back-propagation to gradient computations, is memory-hunger, in that the underlining computational graph could grow large when many millions of Monte Carlo rays are sampled, as required by rendering a megapixel image for a design algorithm to process with. As such, intermediate variables and computations could easily fill up device memory, limiting the scope of scaling-up, hence reducing the overall performance of a derivative-aware pipeline. Compared to model-based (or, physics-based) learning scenarios [44], our applications require the image formation to be computed in a Monte Carlo fashion and cannot be explicitly stated, and thus memory-efficient techniques rely on known image formation models like [44] are not directly applicable. Similar memory-hunger issue exists also in differentiable rendering for end-to-end learning [37], [45], and solutions have been proposed [46]. However, an optical design engine differs from a general-purpose graphics renderer, in that optical surface geometry representations are well-parameterized surfaces (e.g., aspheric or freeform splines) rather than discrete meshes. For non-spherical representations, the parameterization is so complicated that there are no analytical solutions for ray-surface intersection. As a result, ray-surface intersections are dominant computations, and become the memory-hunger bottleneck in rendering sensor image and its associated derivatives to design parameters. Specific problems like this in a differentiable optical design engine, are problems that this paper seeks crafted solutions for.

This memory issue has not been fully addressed in previous derivative-aware ray tracing works [13], [29], [30], [33] due to different application-oriented goals. In [13], the implementation of a differentiable ray tracing scheme enables image rendering, and thus rays are traced backwardly starting from the sensor plane, through the lenses, landing at the scene. Gradients are naively accumulated by unrolling iterations of a ray-surface intersection root finder, i.e. straightforward back-propagation. This most frequent yet time-critical operation consumes a large amount of memory and computing resources that limit the number of rays permissible for each render batch. Further, the gradient aggregation at the sensor plane suffers additionally to the large amount of rays sampled per pixel, and [13] sidesteps this issue by splitting the sensor image into blocks and rendering them independently. However this issue can be addressed in a more principled fashion where gradients are computed and back-propagated in a way that is efficient in terms of both compute time and memory. This is made possible by computing the gradients instead of directly back-propagating the computational graph, as explained below.

In this work we propose **dO** (DiffOptics), a derivative-aware lens design optimization pipeline using differentiable ray-tracing. To back-propagate from the error metric to design variable parameters in a memory-efficient way, we analyze the gradients in situations of ray-surface intersection and adjoint back-propagation, to enable validity and efficiency for gradient computation and inference. Based on the obtained derivatives, advanced algorithms could be developed for specific design applications.

Various applications are demonstrated using the proposed derivative-aware pipeline, ranging from classical usage such as spherical and aspherical lens group design optimization, local sensitivity analysis, freeform surface optimization, setup misalignment estimation, to advanced computational imaging applications. We believe that these examples highlight the potential and possible application domains shown by **dO**. This diverse application range differentiates **dO** from previous works as a general-purpose derivative-aware pipeline based on geometric ray tracing.

In contrast to earlier works that rely on external ray tracing engines such as [13], **dO** is fully implemented in PyTorch, which makes it easily portable and simplifies integration with existing machine learning image reconstruction methods, while at the same time utilizing GPU compute resources in an efficient fashion. Source code will be available at [47].

II. dO: Method and Approach

A. Overview

We use geometric ray tracing to model the light transport in a sequential-mode lens design. A lens system is parameterized by a vector variable $\boldsymbol{\theta} \in \mathbb{R}^n$, a collection of n-number design parameters. Starting from one end of the lens system, sampling rays are sequentially traced through a set of parameterized optical surfaces, intersecting only once for each surface, while traveling towards the other end of the lens system. Rays can be traced starting the object plane towards the image plane, as the preferable way in lens design for aberration analysis. Alternatively, rays are traced reversely starting from the image plane towards the object plane preferably in graphics for image rendering. For the i^{th} ray, intersection on the image (object) plane is determined by ray tracing, as a "black-box" function of $\boldsymbol{\theta}$, denoted as $\mathbf{p}_i(\boldsymbol{\theta}) \in \mathbb{R}^2$. Given m number of sampling rays, the collection of $\mathbf{p}_{1,\ldots,m}$ is known as the spot diagram in lens design, denoted as $\mathbf{p} \in \mathbb{R}^{2m}$, a concatenation of vectors.

To optimize a design, a merit function $\epsilon(\cdot) : \mathbb{R}^{2m} \mapsto \mathbb{R}$ is applied to $\mathbf{p}$, producing a scalar error $\epsilon(\mathbf{p}(\boldsymbol{\theta})) \in \mathbb{R}$, as an indicator for design performance. Design optimization aims to solve for an optimal $\boldsymbol{\theta}^*$ by minimizing the error:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \ \epsilon(\mathbf{p}(\boldsymbol{\theta})). \tag{1}$$

For example, the usual merit function is the spot RMS error, by comparing the current spot diagram against a target one $\mathbf{p}_{\text{target}}$, in a least square sense:

$$\epsilon(\mathbf{p}) = \|\mathbf{p} - \mathbf{p}_{\text{target}}\|^2. \tag{2}$$

In **dO**, the merit function can also be at the irradiance level, when optimizing in the image space rather in the intersection space, e.g. in Refs. [13], [48]. This requires integrating a sensor image $I(\mathbf{p})$ from intersections $\mathbf{p}$, given a pre-defined pixel size. The error merit can be defined such that the produced image

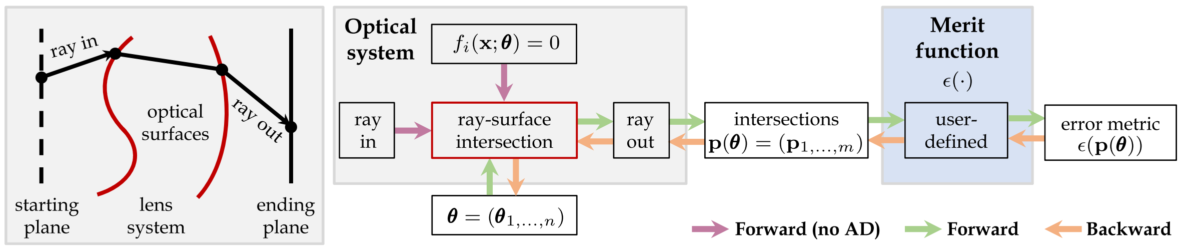

ray in
optical surfaces
ray out
starting plane
lens system
ending plane
Optical system
$f_i(\mathbf{x};\boldsymbol{\theta}) = 0$
ray in
ray-surface intersection
ray out
$\boldsymbol{\theta} = (\boldsymbol{\theta}_{1,\ldots,n})$
intersections $\mathbf{p}(\boldsymbol{\theta}) = (\mathbf{p}_{1,\ldots,m})$
Merit function $\epsilon(\cdot)$
user-defined
error metric $\epsilon(\mathbf{p}(\boldsymbol{\theta}))$
Forward (no AD)
Forward
Backward

Fig. 1. Ray tracing with back-propagation. To be derivative-aware, all modules must be differentiable so that gradients can be back-propagated from the error metric $\epsilon(\mathbf{p}(\boldsymbol{\theta}))$ to variable parameters $\boldsymbol{\theta}$. This is achieved by two stages of the reverse-mode AD: the forward and the backward passes. To ensure differentiability and efficiency, a custom ray-surface intersection solver is introduced in Section II-C. Instead of unrolling iterations for forward/backward, only the forward (no AD) is computed to obtain solutions at surfaces $f_i = 0$, and gradients are amended afterwards as in (7).

is close to a target image I_{target} in a least square sense using pixel-by-pixel comparison:

$$\epsilon(I(\mathbf{p})) = \|I(\mathbf{p}) - I_{\text{target}}\|^2. \tag{3}$$

Crucially, the advantage of such an irradiance-based merit function allows for the consideration of software reconstruction modules such as deep neural networks in the design process. This allows for end-to-end designs in which the optics and the image processing software are jointly optimized, and design solutions are found in which the optical design maximizes the ability of the software module to restore quality images.

Given a merit function $\epsilon(\cdot)$, our goal is to evolve variable parameters $\boldsymbol{\theta}$ iteratively towards an optimal $\boldsymbol{\theta}^*$, using gradient-based optimization. This requires computing $\partial\epsilon/\partial\boldsymbol{\theta} \in \mathbb{R}^n$, the partial derivatives that indicate how design parameters affect the error metric locally. Assuming $\epsilon(\mathbf{p}(\boldsymbol{\theta}))$ is a continuous function of $\boldsymbol{\theta}$, by the chain rule, the partial derivatives in (1) expand as, without or with I:

$$\frac{\partial\epsilon}{\partial\boldsymbol{\theta}} = \frac{\partial\mathbf{p}}{\partial\boldsymbol{\theta}}\frac{\partial\epsilon}{\partial\mathbf{p}}, \tag{4a}$$

$$\frac{\partial\epsilon}{\partial\boldsymbol{\theta}} = \frac{\partial\mathbf{p}}{\partial\boldsymbol{\theta}}\frac{\partial I}{\partial\mathbf{p}}\frac{\partial\epsilon}{\partial I}. \tag{4b}$$

In the following, we rely on using AD for computing (4). In AD, sole Jacobian matrices are rarely evaluated, instead the vector-Jacobian products are computed for a given perturbation, in our case denoted as $\Delta\epsilon$ and $\Delta\boldsymbol{\theta}$, with $\mathcal{J}_{(\cdot)}^T$ denoting the Jacobian transposes, (4) rewrites as:

$$\Delta\boldsymbol{\theta} = \mathcal{J}_{\mathbf{p}}^T\mathcal{J}_{\epsilon}^T\Delta\epsilon, \tag{5a}$$

$$\Delta\boldsymbol{\theta} = \mathcal{J}_{\mathbf{p}}^T\mathcal{J}_{I}^T\mathcal{J}_{\epsilon}^T\Delta\epsilon. \tag{5b}$$

Given a desired decrease $\Delta\epsilon$ in the error metric, our goal in this paper is to compute the corresponding variable parameter changes $\Delta\boldsymbol{\theta}$ (i.e., the derivatives), by evaluating (5), and preferably being memory-efficient for scaling up to large design problems.

B. Ray Tracing With Back-Propagation

A straightforward way to compute the derivatives in (5) is by using AD to perform all the computations in ray tracing. This requires using a derivative-aware numerical library, e.g., PyTorch [49], to track and compute both the primal value and its derivatives for every elementary arithmetic operation. For our

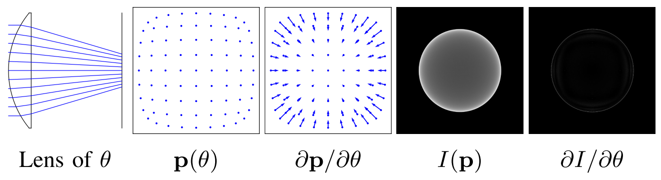


Fig. 2. Derivative-aware property of **dO**. In this example, the first surface curvature $\theta \in \mathbb{R}$ of a lens is under investigation. **dO** can obtain a spot diagram $\mathbf{p}(\theta)$ and derivatives $\partial\mathbf{p}/\partial\theta$, or a rendered image $I(\mathbf{p})$ and derivative $\partial I/\partial\theta$.

multiple-input and single-output case, i.e., many variable parameters and one scalar error metric, for numerical efficiency it is preferable to employ the reverse-mode AD, or back-propagation in the context of machine learning. Briefly speaking, to perform back-propagation on (1) to calculate the derivatives, two stages are required, the forward pass and the backward pass.

Fig. 1 illustrates this process by applying the reverse-mode AD to ray tracing. In the forward pass, values of differentiable parameters ($\boldsymbol{\theta}$ in our case) are inputs from the starting plane to the ending plane, by ray tracing, to obtain intersections $\mathbf{p}(\boldsymbol{\theta})$. With a user-defined merit function, an error metric $\epsilon(\mathbf{p}(\boldsymbol{\theta}))$ is calculated. The forward pass defines a computational graph in progressive that relates all differentiable variables, and non-differentiable variables are not needed for the forward pass with AD (i.e., forward with no AD in Fig. 1). In the backward pass, this computational graph is back-propagated to compute gradients following the chain rule, starting from the previously obtained error metric $\epsilon(\mathbf{p}(\boldsymbol{\theta}))$, to compute variable derivatives $\Delta\boldsymbol{\theta}$ as in (5). Fig. 2 illustrates a derivative-aware example where for illustrative purposes the partial derivatives are shown for interpretation. For example, the derivatives of the spot diagram are the "flow" diagram $\partial\mathbf{p}/\partial\theta$ revealing the motion of the spot diagram when changing θ, where the arrows indicate moving directions and lengths indicate magnitudes. In this case, increasing θ will focus the rays, affecting more in the peripheral spots than the central ones. Similarly, the derivative of the rendered image is a "derivative" image $\partial I/\partial\theta$ that tells how would the pixel value changes when changing θ. The "derivative image" indicates that the peripheral pixels are more sensitive than the central ones, as expected.

To ensure differentiability, all modules must be differentiable, i.e., modular computations are derivative-aware so that both

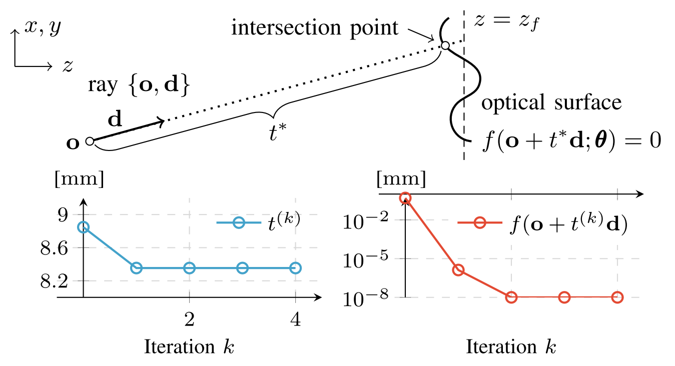

x, y
z
intersection point
$z = z_f$
ray $\{\mathbf{o}, \mathbf{d}\}$
$\mathbf{d}$
t^*
$\mathbf{o}$
optical surface
$f(\mathbf{o} + t^*\mathbf{d}; \boldsymbol{\theta}) = 0$
[mm]
$t^{(k)}$
Iteration k
[mm]
$f(\mathbf{o} + t^{(k)}\mathbf{d})$
Iteration k

Fig. 3. The ray-surface intersection problem (6) is solved by Newton's method. Once the optimal t^* is obtained, the AD-version t is computed as in (7).

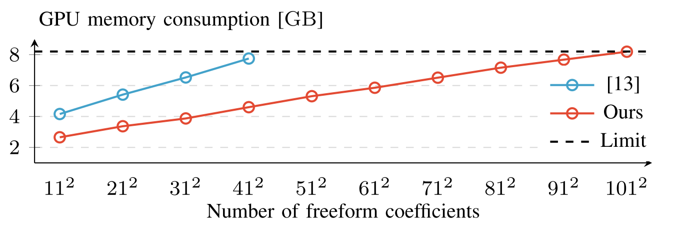


Fig. 4. Comparison between the two ray-surface intersection methods when optimizing Fig. 14. Our method reduces the required memory by ~ 6 times.

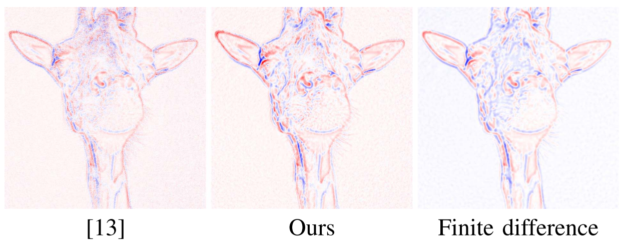


Fig. 5. Gradient images computed using different ray-surface intersection methods. Compared to gradient image computed by [13], our approach produces a cleaner one, due to the memory allowance for sampling more rays, as the consequence of low memory consumption of computing the intersections. The finite difference result suffers from slight blurriness due to numerical rounding errors, because the finite difference values of $\theta_{+/-} = \{10^{-6}, -10^{-6}\}$ were approaching the single floating point precision.

the primals and their derivatives are computed and tracked in a computational graph. This requires both the optical system and the merit function to be fully differentiable, so that $\boldsymbol{\theta}$, $\mathbf{p}$, $\epsilon(\mathbf{p}(\boldsymbol{\theta}))$, and the desired derivatives $\Delta\boldsymbol{\theta}$ given $\Delta\epsilon$, can all be related through ray tracing.

C. Differentiable Ray-Surface Intersection

To perform ray tracing from surface to surface, the crucial part in a differentiable optical system is to calculate ray-surface intersection, where an intersection point $\mathbf{x} \in \mathbb{R}^3$ is computed. The intersection lies on an optical surface described by an implicit function $f(\mathbf{x}; \boldsymbol{\theta}) = 0$, and lies along the ray direction. Given a ray $\{\mathbf{o}, \mathbf{d}\}$ of origin $\mathbf{o} \in \mathbb{R}^3$ and direction $\mathbf{d} \in \mathbb{R}^3$ of unit length, $\mathbf{x} = \mathbf{o} + t\mathbf{d}$ for a positive marching distance $t \in \mathbb{R}_+$. Fig. 3 illustrates this problem, which finds $t > 0$ such that:

$$f(\mathbf{x}; \boldsymbol{\theta}) = f(\mathbf{o} + t\mathbf{d}; \boldsymbol{\theta}) = 0. \tag{6}$$

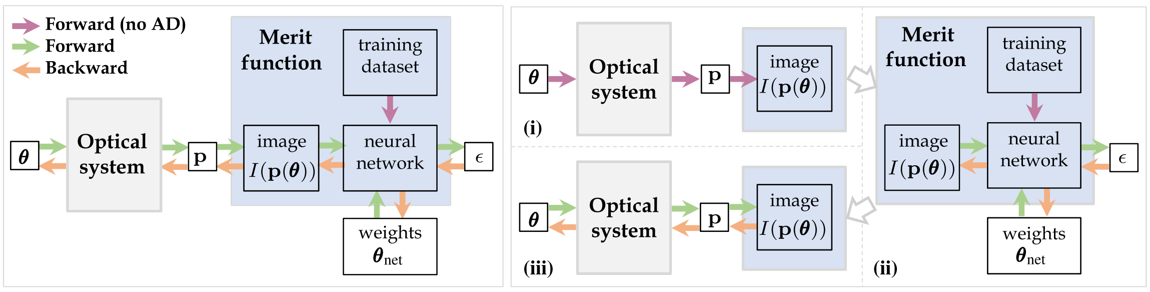
It can be solved using iterative root finders (e.g., Newton's method) implemented in AD, unrolling the iterations for gradient evaluations, so that the solution t and its gradients can be related to the lens parameters $\boldsymbol{\theta}$, as in Ref. [13]. However, this straightforward approach is memory consuming because of storing the intermediate iteration variables. This issue can be avoided by taking advantage of the fact that solution to (6) is independent on initialization of t, and hence the optimal solution t^* can be first solved with no AD, no need for storing intermediate states, and calculate the AD-version t by one Newton iteration:

$$t \leftarrow t^* - \frac{f(\mathbf{x}; \boldsymbol{\theta})}{\mathbf{d} \cdot \nabla f(\mathbf{x}; \boldsymbol{\theta})}, \quad \text{with} \quad \mathbf{x} = \mathbf{o} + t^*\mathbf{d}, \tag{7}$$

where $\cdot$ denotes inner product, and $\nabla f(\mathbf{x}; \boldsymbol{\theta})$ denotes the spatial derivatives of the implicit function with respect to $\mathbf{x}$, parameterized by $\boldsymbol{\theta}$. Refer to Supplemental Document for specific forms of $f(\cdot)$ and $\nabla f(\cdot)$. Example surfaces are aspheres, XY polynomials, and B-splines. Notice our approach does not depend on specific iteration solvers chosen for solving (6), yet keeping the solution differentiable. Our approach is simple yet effective, and to our best knowledge is the first to address this memory issue in differentiable optics research.

We employ Newton's method for obtaining t^*, initialized by a non-singular estimate $t^{(0)} = (z_f - \mathbf{o}_z)/\mathbf{d}_z$ as in Fig. 3, and the iteration stops when residual is smaller than the tolerance. Convergence of $t^{(k)}$ and $f(\mathbf{o} + t^{(k)}\mathbf{d}; \boldsymbol{\theta})$ is within a few iterations of k. For spherical lenses, Newton's method converges in theory within exactly one iteration. Fig. 4 shows the superiority of the proposed approach, in comparison against the unrolled approach [13]. The unrolled approach needs to store intermediate states for AD, and consumes a large memory that cannot be affordable on a single GPU, and hence limiting the total number of freeform coefficients in the optimization. In contrast, our proposed differentiable solver drastically alleviates this issue, and imagine how often ray-surface intersection happen in ray tracing the design. This improvement allows **dO** to perform memory-efficient back-propagation when evaluating $\mathcal{J}_{\mathbf{p}}^T$ in (5), no matter what merit function is chosen.

Our proposed method improves gradient estimates for Deep Lens designs, by allowing more rays to be sampled per pixel. In Fig. 5 a doublet aspheric lens (see Table III in Supplementary Document for prescription) is under investigation, with its first surface's 4th aspheric coefficient $\theta = 0$ being differentiated. Given a planar texture image object, **dO** renders the corresponding sensor image I and its gradient $\partial I/\partial\theta$. Fig. 5 shows the same gradient image computed using method in [13] and ours, as well as a finite difference version for reference, which is inaccurate due to limited numerical precision. Method in [13] is limited to fewer rays per pixel because of high memory consumption in computing ray-surface intersection, producing a noisy gradient image. In contrast, our method allows for more rays to be sampled per pixel, yielding a cleaner gradient image. This example demonstrates the superiority of our approach over [13], and the benefit of applying our approach for robust gradient estimate, and hence to Deep Lens training.



(a) Back-propagation learning **(b)** Adjoint back-propagation learning

Fig. 6. Adjoint back-propagation. In end-to-end learning for computational imaging applications, the merit function contains a rendering image, a neural network, and its weights as variables. Rendering the image could use many millions of rays, and the unaffordable memory limits total ray samples in back-propagation (a), and hence compromising the performance. Adjoint back-propagation (b) alleviates this issue by splitting computations into multiple stages (i), (ii), and (iii).

D. Adjoint Back-Propagation

Evaluating (5) depends also on the specific merit function chosen. For simple metrics such as the spot RMS in conventional lens designs, evaluating (5a) does not involve many rays and the memory issue is not urgent. However, when the merit function is in the image space, e.g. (3), and involves rendering images, for example a megapixel resolution image $I(\mathbf{p})$ may take an intensive sampling of many millions of rays. In this scenario, direct back-propagating on (5b) could be problematic and may require impractical amounts of memory in the backward pass, making derivative evaluations prohibitive in practice.

This issue can be partially alleviated by exploiting a key insight from (5b) that, the target $\Delta\boldsymbol{\theta}$ contains two parts: (i) ray-tracing relevant derivatives $\mathcal{J}_{\mathbf{p}}^T$ determined by the current design, and (ii) error-metric relevant derivatives $\mathcal{J}_I^T \mathcal{J}_\epsilon^T$ determined by the chosen merit function $\epsilon(\cdot)$. This separability property enables us to split computations into multiple passes, eventually alleviating the back-propagation memory issue. A similar memory-efficient approach was proposed in Ref. [46].

The separability property reflects by introducing an intermediate derivative image ΔI to (5b):

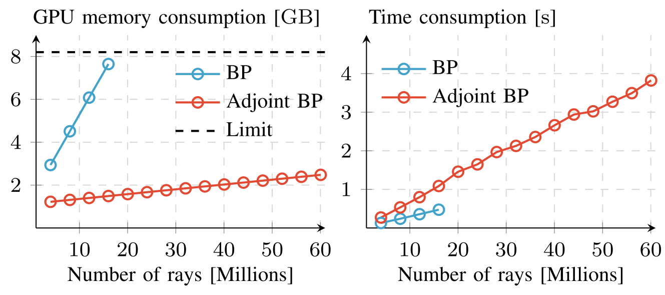
$$\Delta I = \mathcal{J}_I^T \mathcal{J}_\epsilon^T \Delta\epsilon, \tag{8a}$$

$$\Delta\boldsymbol{\theta} = \mathcal{J}_{\mathbf{p}}^T \Delta I. \tag{8b}$$

By first evaluating (8a) and then (8b), the original (5b) can be eventually evaluated. The computations are hence split into three sequential stages:

i) Forward computation (no AD). This obtains a rendered image $I(\boldsymbol{\theta})$ solely, with no derivative information on $\boldsymbol{\theta}$. Since no computational graph is created or stored, this stage has low memory requirements.
ii) Forward and backward on the metric function $\epsilon(I)$. This treats the rendered image I as a differentiable variable, and obtains ΔI by back-propagating (8a).
iii) Forward and backward on the optical system. This gives the final desired variable derivatives $\Delta\boldsymbol{\theta}$ by back-propagating (8b).

The above procedure, termed the adjoint back-propagation, allows the backward pass to be performed at stages (ii,iii), and hence alleviating the memory issue as a whole. The adjoint terminology is relevant to the adjoint methods in other research fields, in that the computations of back-propagation are separated with the adjoint part being computed without storing intermediate variables from the forward part, as in (8).



(a) Memory comparison **(b)** Timing comparison

Fig. 7. Memory and time comparison between back-propagation (BP) and adjoint back-propagation (adjoint BP) with respect to the number of sampling rays when the merit function is in the image space, i.e., evaluating (5b). Due to excessive memory requirements, the baseline method cannot proceed beyond 16 millions of rays, whereas the adjoint method is capable of that meanwhile maintaining a low memory request.

This computational spirit is depicted in Fig. 6, where the merit function is chosen deliberately to be the one used in end-to-end learning [10], [11], [12], where the rendered image $I(\mathbf{p}(\boldsymbol{\theta}))$ is fed to a neural network whose own weights $\boldsymbol{\theta}_{\text{net}}$ are adjustable during training, as a set of additional differentiable parameters in the design optimization process. In end-to-end learning, the goal is to optimize both the optical system parameters $\boldsymbol{\theta}$ and network weights $\boldsymbol{\theta}_{\text{net}}$ such that the hardware-software combination produces the optimal image quality for the neural network to sharpen raw sensor images. See Figs. 15 and 16 for examples.

Fig. 7 shows a memory consumption comparison between the two back-propagation methods, applying to Fig. 2, assuming a dummy merit function of (3). The adjoint back-propagation takes more time, but the memory consumption maintains at a low-level, allowing for aggregations and sampling scale-ups. This memory-efficiency enables a large number of sampling rays for faithful image rendering for practical usage in end-to-end computational imaging designs.

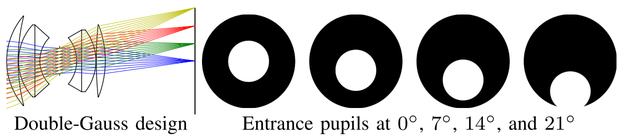


Fig. 8. Entrance pupil calculation. By tracing a dense grid at the very first optical surface, whose aperture is the dark circle, overlaid on which we can obtain the entrance pupil area (bright region) for subsequent ray spatial sampling, at different viewing angles.

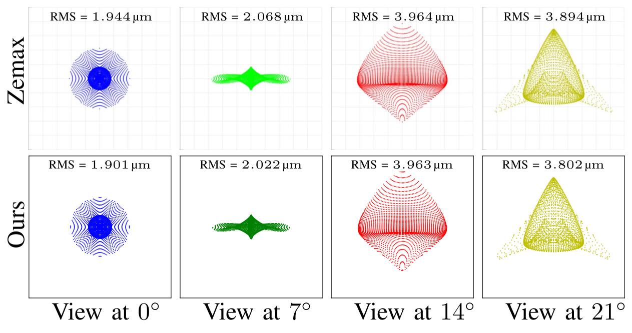


Fig. 9. Spot diagrams and RMS spot errors that produced by **dO** highly resemble those by Zemax.

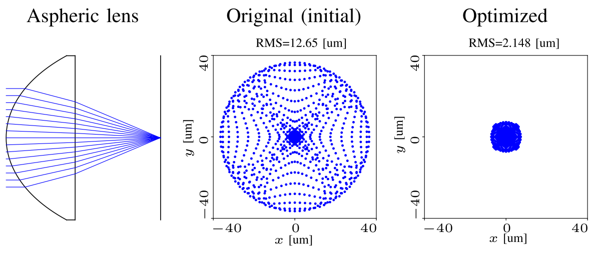


Fig. 10. Spherical aberration minimization. Differentiable parameters $\boldsymbol{\theta}$ are surface curvature, conic and 4th asphere coefficient. Our solver converges in 1 s.

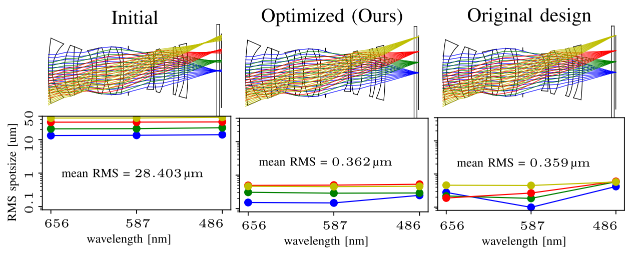


Fig. 11. Nikon lens group total aberration minimization. Starting from an all-spherical design, with the differentiable parameters $\boldsymbol{\theta}$ being all surface curvatures and aspheric coefficients of two surfaces, our optimized version achieves similar performance as the original design.

III. Implementation

A. Optimization

Given the derivatives $\Delta\boldsymbol{\theta}$ from back-propagating (5), **dO** performs optimization and iteratively changes the variable parameters $\boldsymbol{\theta}$. When there are constraints in the design, e.g., positive air-spacing, minimum glass thickness or back focal length,

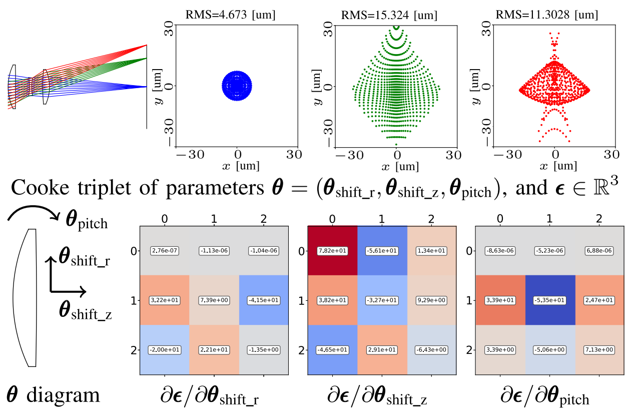


Fig. 12. Tolerancing a nominal lens system. The sensitivity matrices are readily obtained as the Jacobians in **dO**. Here, $\boldsymbol{\epsilon}$ and $\partial\boldsymbol{\epsilon}/\partial\boldsymbol{\theta}$ represent the sensitivity matrices in three different field of views.

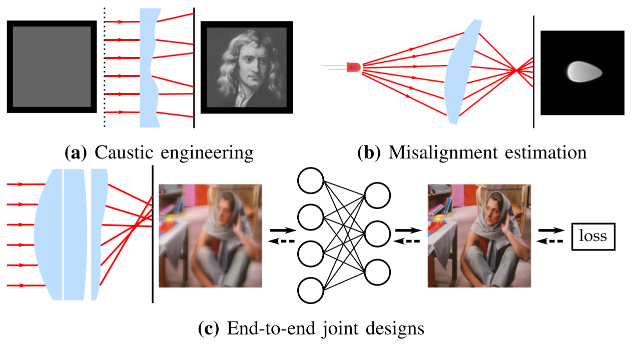


Fig. 13. Overview of advanced applications. (a) Caustic engineering aims to design a freeform surface to produce a target irradiance pattern at certain distance (Fig. 14). (b) Real setup misalignment can be estimated by using **dO** in reverse as a black-box solver (Fig. 17). (c) End-to-end designs that jointly optimize lenses and image post-processing algorithms (here, neural networks) for computational imaging applications (Fig. 15, 16).

maximum system overall size, (1) can be modified by adding a vector constraint function.

The specific optimization method depends on the number of variables. When the number of variables is small (for example $\boldsymbol{\theta} \in \mathbb{R}^n$, $n < 20$), Classical damped least squares [50] are employed to efficiently optimize (1), that the required Jacobians can be constructed from the derivative vectors. In this case it does not take full advantage of the derivative-aware property of **dO**. This is in contrast to the case when n is large (e.g. for free-form surface optimization) and the Jacobians are too large to be explicitly constructed, and popular gradient descent methods such as Adam [51] can be employed. If desirable, additional regularization terms are possible, for example when solving (15). This optimization flexibility feature differentiates **dO** from existing optical design software.

B. Lens System

We follow the standard lens design pipeline [52], [53] to model a lens systems. We focus on the sequential mode, where starting from one end of the lens system, rays are sequentially traced through a sequence of parameterized optical surfaces

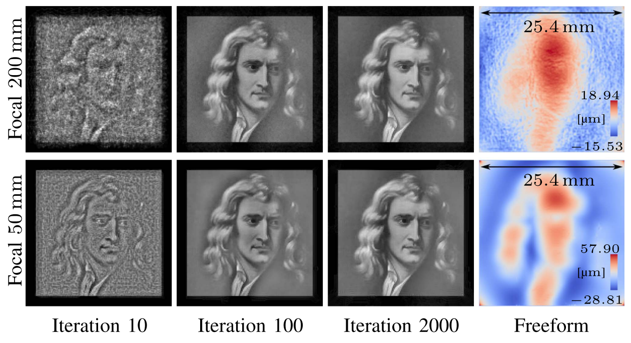


Fig. 14. Caustic engineering. We optimize a freeform surface to refractive collimated light into a target irradiance distribution at two focal lengths, at wavelength $532.8\,\mathrm{nm}$ with a refractive index of 1.5. **dO** produces a smooth freeform surface, and optimizes the caustic structures efficiently as shown in the intermediate states. See Visualization 1 for the optimization process.

(including the image plane, i.e., the sensor plane), intersecting only once for each surface, while traveling towards the other end of the lens system. In the sequential mode, the exact visibility ordering of the surfaces is known a priori, and thus no need for finding the closest surface intersection when performing ray tracing.

Ray propagation through a lens system involves two major steps, finding the ray-surface intersection point (finding intersection points by solving (6)), and refraction of the ray at material interfaces with chromatic effects. Only valid rays are traced in continuity, whereas invalid rays happen when the intersections are outside of the lens geometry or when total internal reflection takes place.

At material interfaces, transmitted direction $\mathbf{d}_t$ is determined from surface normal direction $\mathbf{n} = \nabla f / \|\nabla f\|$ and incident direction $\mathbf{d}_i$, by Snell's law [54]:

$$\mathbf{d}_t = \mathbf{n}\sqrt{1 - (1 - \cos^2 \psi_i)\eta^2} + \eta(\mathbf{d}_i - \mathbf{n}\cos\psi_i), \quad (9)$$

where $\cos\psi_i = \mathbf{d}_i \cdot \mathbf{n}$ and $\eta = n_i/n_t$ is the ratio of refraction indices of the two materials. Refractive index follows Cauchy's equation $n(\lambda) = A + B/\lambda^2$, with A and B determined from central refractive index and Abbe number.

Though in this work we focus on the sequential mode where optical surfaces are fixed in a known order, non-sequential mode should also be possible with proper extensions and modifications on the current ray tracing engine.

C. Two Tracing Modes: Forward and Backward

Depending on the needs, rays can be traced through a lens system in two different modes, *forward mode* or *backward mode*. In the forward mode, rays are traced starting from the object plane towards the image plane. This is the preferable way in lens design for aberration analysis, e.g., generation for spot diagrams. In the backward mode, rays are traced reversely, starting from the image plane towards the object plane. This is a sampling efficient way for sensor image rendering, and thus is the preferable way in computer graphics to render realistic images. We will be using these two modes interchangeably depending on specific needs.

D. Differentiable Image Rendering From Intersections

Once tracing is finished, a synthetic image $I(\mathbf{p})$ can be generated given the intersection points $\mathbf{p}$, provided with a proper image integrator. This process (termed rendering) of pixelization from a continuous light signals to discretized pixel values, is handled differently in the two tracing modes.

Reconstruction filter in the forward mode: In the forward tracing mode, performance analysis is conduct to understand the optical property of the current design, where rays are purposely generated according to analysis-specific criteria. This process may involve gradient computations, and differentiability is desired in that the generated image pixel values are differentiable to intersection point movements.

To ensure differentiability, the derivatives of this mapping from intersection to pixel values need to be continuous. **dO** employs a linear interpolation scheme to round the fractional intersection point $\mathbf{p}$ into the nearest four neighboring pixels, with the corresponding four pixel values linearly weighted.

Monte Carlo integrator in the backward mode: In the backward tracing mode, rays are initialized from the pixelated sensor plane, and are traced outwards, and the major goal is to render a physically correct image for the current design given a specific scene. This is achieved by integration of the rendering equation [55], and is evaluated by renderers using a Monte Carlo integrator for discrete sampling the integral. This backward tracing mode is utilized in end-to-end system designs, see Section V-B.

E. Entrance/Exit Pupil Computation for all Fields

To perform ray tracing for a lens system, the entrance pupil has to be determined first, which is the area over the very front lens element where rays from a given viewing angle will finally reach the sensor plane. This is easily determined for paraxial angles, but not for larger angles. Fig. 8 demonstrates this vignette effect of a double-Gauss design [56], with the calculated entrance pupils shown at different views. Our engine determines the entrance pupil by ray tracing a dense grid (1025×1025) at specific viewing angles. Entrance pupils are determined if the sampled rays propagate through all the optical elements successfully. Exit pupils can be determined in a similar manner as described in [57].

F. Derivative-Aware Implementation

dO is implemented from scratch in PyTorch [49]. In comparison to [13], which is based on Mitsuba2 [37], **dO** is therefore more portable and very easy to combine with existing deep neural network code for image reconstruction. This makes it easier to integrate **dO** in various end-to-end design problems while at the same time offering more degrees of freedom in terms of flexibility for additional applications. Due to the PyTorch backend, **dO** effectively uses GPU compute resources and due to the algorithmic improvements outlined in the previous section, it is significantly more memory efficient than previous end-to-end differentiable ray-tracing systems. **dO** produces highly identical results to modern lens design software. To verify this, a double-Gauss design [56] is under sanity check with Zemax [1],

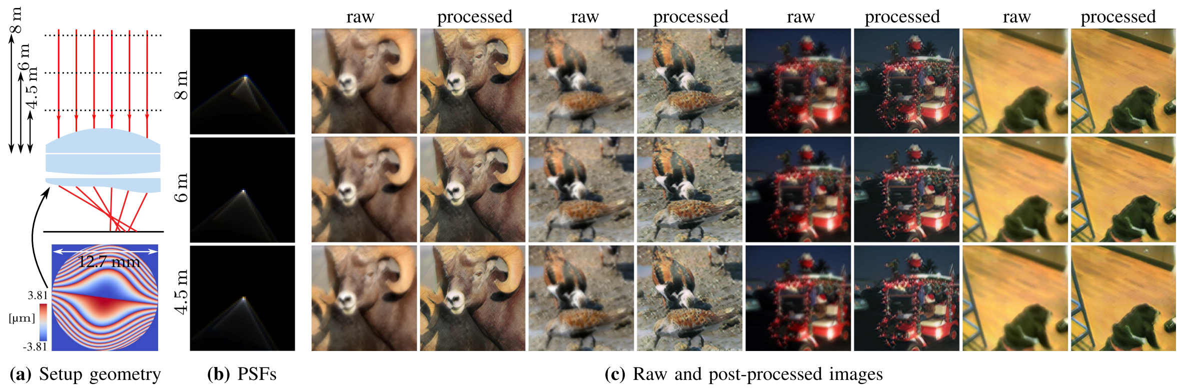


Fig. 15. **dO** can perform end-to-end wavefront coding, i.e., jointly optimize phase optics profile and deconvolution algorithm for extended depth of field applications, here by simulation. (a) Setup geometry and optimized phase optics. (b) Central PSFs at different distances. (c) Raw and post-processed images by the neural network.

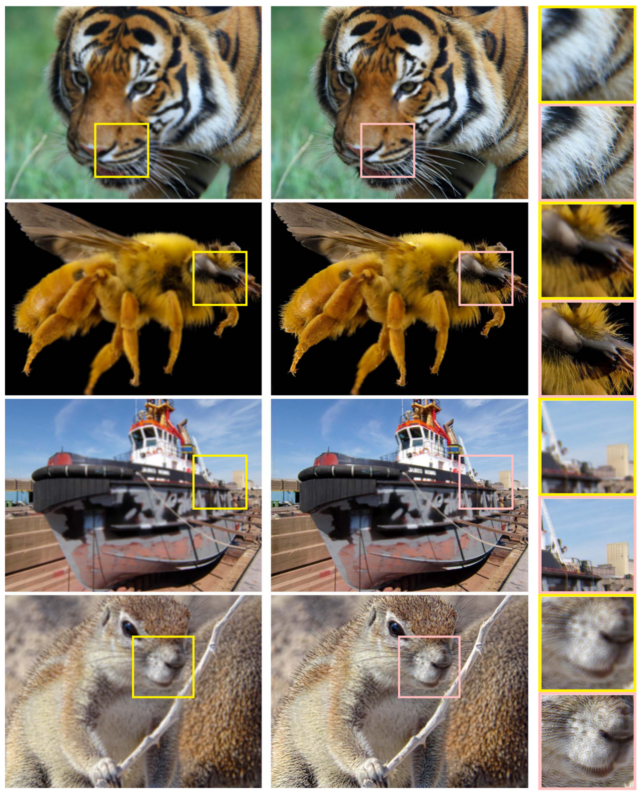


Fig. 16. **dO** can perform end-to-end large FOV imaging by jointly optimizing aspheric coefficients and a neural network for large FOV applications, here by simulation. Raw and post-processed images by the neural network.

using single wavelength $\lambda = 587.56\,\mathrm{nm}$ at four field of views, as in Fig. 9. The spot diagrams and the RMS errors are almost identical to the Zemax results despite a slight variation due to different aperture sampling strategies.

IV. Classical Applications

dO can manage classical design problems, as will be demonstrated in this section. Spot RMS error at different viewing angles is chosen as the merit function $\epsilon(\cdot)$ for design optimization. See Supplemental Document for lens prescriptions and full details.

A. Design Optimization

Spherical aberration minimization: The first example is to optimize the aspherical coefficients of an asphere lens to minimize the axial spherical aberration. In Fig. 10, parameters of a revised asphere lens design (Thorlabs, ACL5040 U) are optimized in the hope of reducing the axial RMS spot. Compared to the initial design, our differentiable engine ends up with a nearly six times smaller RMS spot.

Photographic camera design optimization: The engine can also optimize complicated lens group for design optimization. Fig. 11 shows the second example to re-parameterize curvatures and aspheric coefficients of a Nikon patent design to minimize the total RMS spot error at different field of views ($0°$, $10°$, $20°$, $32.45°$), at three wavelengths ($656.27\,\mathrm{nm}$, $587.56\,\mathrm{nm}$, $486.13\,\mathrm{nm}$). The design is initialized by removing all aspheric coefficients, showing large aberrations. After optimization, our optimized design shows a comparable mean RMS error of the original design. This example demonstrates the capability of **dO** to perform multi-lens and aspheres optimization.

B. Tolerance Analysis

In tolerancing a lens system, or known as sensitivity analysis, a presuming small, linear parameter perturbation $\widehat{\Delta\boldsymbol{\theta}}$ is enforced, and the total effect of the perturbation, $\widehat{\Delta\epsilon}$, is calculated through the nominal system to determine the potential effects. To avoid confusion between perturbation and the gradients considered before, we use the $\widehat{(\cdot)}$ notation. This is mathematically paraphrased by relating $\widehat{\Delta\epsilon}$ to $\widehat{\Delta\boldsymbol{\theta}}$ through the derivatives, known as the Jacobian-vector product in AD terminology:

$$\widehat{\Delta\epsilon} = \mathcal{J}_\epsilon \mathcal{J}_\mathbf{p} \widehat{\Delta\boldsymbol{\theta}}. \tag{10}$$

Notice there is a reciprocal relationship between (10) and (5). In the forward analysis, $\widehat{\Delta\boldsymbol{\theta}}$ is given to compute $\widehat{\Delta\epsilon}$, whereas in the inverse analysis, $\widehat{\Delta\epsilon}$ is given to compute $\widehat{\Delta\boldsymbol{\theta}}$, assuming proper prior probabilities or regularization on $\widehat{\Delta\boldsymbol{\theta}}$. Despite alternative methods such as finite difference, analytical gradients [58], or the wavefront differential method [59], **dO** can evaluate (10) intrinsically, without additional implementation efforts.

In Fig. 12, a Cooke triplet is under sensitivity analysis, with all the optical element positional misalignment parameters $\boldsymbol{\theta}$ being toleranced. **dO** can obtain the Jacobian matrix, known as the sensitivity matrix for further analysis. See Supplemental Document for misalignment parameter definitions.

V. Advanced Applications

Thanks to differentiability and the versatile optimizer, **dO** can be combined with advanced post-processing computational algorithms for complex designs or setup reverse-engineering, and beyond the usual application range of lens design software. Such domain-specific applications are deemed not easily configurable with existing design software, as three examples shown in Fig. 13.

A. Caustic Engineering

Caustic engineering aims to produce a target image by pixel-wisely changing the directions of a directional light source, by optimizing a freeform optical surface [60], [61]. Here, we demonstrate caustic engineering as one of the applications of **dO**. We parameterize the desired surface in spline freeform, and thus this representation ensures smoothness and higher-order continuities of the surface, in contrast to the mesh-based representations in alternative differentiable solvers [37]. The freeform knot positions are fixed, spline coefficients are optimized and consequently changing the surface geometry. This flexibility of geometry representation allows **dO** to perform freeform surface optimization. We employ the forward mode to render caustic images. To enable a satisfactory reconstruction, the desired freeform surface is assumed to be smooth and hence is represented by B-splines with a large degree of freedom approximately equal to the pixel number of the target image. The error metric was the standard mean-square-error (MSE), and we optimize the B-spline coefficients $\boldsymbol{\theta}$ that characterize the freeform surface:

$$\min_{\boldsymbol{\theta}} \ \|I(\boldsymbol{\theta}) - I_{\text{target}}\|^2. \tag{11}$$

(11) is optimized using Adam [51] because construction of Jacobian is computationally prohibitive due to the large number of variables in (11). Fig. 14 shows the results along with intermediate optimization states, where the optimized freeform surface is shown in height maps. The reconstruction image is contrast-preserving, and the optimized freeform surface is smooth. Although in this particular example, the contrast in our result is not high compared to alternative specific solvers, e.g., using optimal transportation [60] or iterative warping [61], our result shows freeform designs would be more approachable when a plug-and-play differentiable engine such as **dO** is available.

B. End-to-end Computational Imaging Designs

Computational imaging designs [10], [62] require jointly optimization of hardware and software as a whole part. For example, a blurry image I can be de-blurred by a designed algorithm or a neural network $\mathcal{N}(\cdot)$, and becomes a potentially sharper image $\mathcal{N}(I)$. In end-to-end designs, this design process involves an evolution of both the lens design (parameterized by $\boldsymbol{\theta}$) and the post-processing algorithm $\mathcal{N}(\cdot)$ (parameterized by $\boldsymbol{\theta}_{\text{net}}$, e.g., an untrained neural network of weights $\boldsymbol{\theta}_{\text{net}}$). End-to-end design optimizations are usually accomplished in a supervised setting, by using a training set as target images I_{target} to educate and evolve the hardware/software variable parameters. This design pipeline and the back-propagation procedure has been depicted in Fig. 6. An end-to-end design error metric involves both $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_{\text{net}}$, and is re-phrased as:

$$\epsilon(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{net}}) = \mathcal{L}(\mathcal{N}(I(\boldsymbol{\theta}); \boldsymbol{\theta}_{\text{net}}), I_{\text{target}}), \tag{12}$$

where $\mathcal{L}(\cdot, \cdot)$ is a general loss metric function to quantify a total error difference between the post-processed image $\mathcal{N}(I(\boldsymbol{\theta}); \boldsymbol{\theta}_{\text{net}})$ and the ground truth target image I_{target}. The goal in the following is to minimize (12) for two end-to-end computational imaging designs, using **dO**.

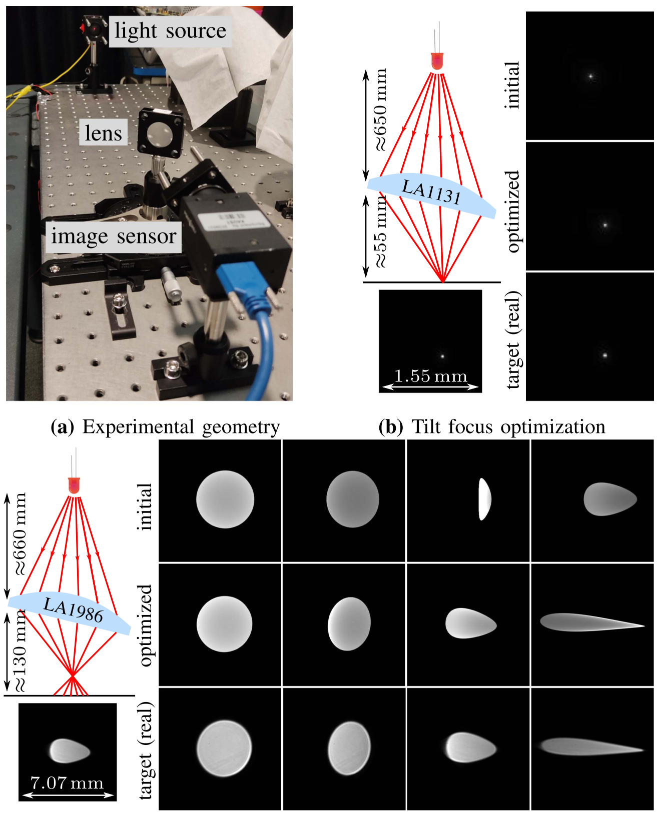
End-to-end design for extended-depth-of-field imaging: In wavefront coding, a pupil plane phase modulator is introduced to deliberately distort input lights for point-spread-function (PSF) engineering, e.g., cubic phase plate [3] that produces depth-invariant PSFs for extended-depth-of-field, and coded aperture [4] and lattice-focal [63] that produce depth-sensitive PSFs for depth retrieval. This process is a joint optical-algorithmic problem in that the final image is the output from a sequential appliance of the encoding optics and the decoding algorithm. Prior end-to-end approach relies on paraxial approximation [10], ignoring the spatially variant nature of PSFs. Here, **dO** provides an initial solution to break this limitation, in that the optical system is faithfully reproduced by ray tracing. We parameterize the phase plate using only third-order XY polynomials, and end-to-end jointly optimize the polynomial coefficients $\boldsymbol{\theta}_{\text{XY}}$ and U-Net [64] parameters $\boldsymbol{\theta}_{\text{net}}$ for extended depth of field applications, as in Fig. 15. Specifically, the total error metric in (12) consists an MSE loss $\mathcal{L}_{\text{mse}}$ and a channel feature loss $\mathcal{L}_{\text{vgg16}}$ on pre-trained VGG16 [65]:

$$\min_{\boldsymbol{\theta}_{\text{XY}}, \boldsymbol{\theta}_{\text{net}}} \ \epsilon(\boldsymbol{\theta}_{\text{XY}}, \boldsymbol{\theta}_{\text{net}}) = \mathcal{L}_{\text{mse}} + \mathcal{L}_{\text{vgg16}}. \tag{13}$$

See Supplemental Document for full details on network architecture, definitions of loss terms $\mathcal{L}$, and training set generation. Initialized from a null zero phase, the optimized phase in Fig. 15(a) exhibits a similar structure as in [3], verified by the central PSFs in Fig. 15(b). Given the blurry raw input images, the post-processed images in Fig. 15(c) reveal sharp features.

End-to-end design for large field-of-view (FOV) imaging: Large field-of-view (FOV) imaging is challenging for cemented doublets due to the severe aberrations that cannot be fully compensated because of the lack of design degree of freedoms. We argue this large aberrations can be corrected not through additional optical components, but through a computational post-processing algorithm that applies on the perceived aberration image. Using an end-to-end learning, this optical aberration limitations can be addressed by using a post-processing neural network, here we use [66]. The training follows (12), optimizing:

$$\min_{\boldsymbol{\theta}_{\text{asphere}}, \boldsymbol{\theta}_{\text{net}}} \ \epsilon(\boldsymbol{\theta}_{\text{asphere}}, \boldsymbol{\theta}_{\text{net}}) = \mathcal{L}_{\text{mse}} + \mathcal{L}_{\text{vgg16}}. \tag{14}$$



(c) Misalignment optimization for different angular misalignment

Fig. 17. **dO** can be employed as a general-purpose solver to back-engineer real setup misalignment parameters so that simulation and reality match, demonstrated by the high similarity between optimized images and the target (real) ones. (a) Experimental setup includes an LED light source, a lens, and an image sensor. (b) LA1131 was in focus but tilted. (c) LA1986 was out of focus and tilted. Optimized images share high visual similarity to the real target ones, revealing the success of our approach. See Visualization 2 for the optimization process.

A cemented doublet parameterized by its aspheric coefficients $\boldsymbol{\theta}_{\text{asphere}}$ and network parameters $\boldsymbol{\theta}_{\text{net}}$ are jointly optimized following a similar metric as in (13). Fig. 16 shows how the original blurry images can be post-processed and be deblurred, after the end-to-end learning, justifying the applicability of **dO**.

C. Misalignment Estimation

Finally, we show real experimental results to leverage potentials of the proposed differentiable pipeline, **dO**, by using it in reverse to estimate misalignment of an experimental setup. This task of misalignment estimation is not possible with previous differentiable renderers, e.g., Mitsuba2 [37] or Ref. [13], because of geometric representation incompatibility and missing implementations for optics element perturbations, and most importantly, the memory efficiency of **dO** allows for large number of rays to be sampled per pixel for gradient estimates. In Fig. 17(a), a pinpoint light source, consists of an LED (central wavelength $622\,\mathrm{nm}$) and an iris of radius $0.4\,\mathrm{mm}$, is placed far in front of a misaligned plano-convex lens. The lens was able to rotate freely, but the exact angular values were unknown, and the imperfect mounting leads to slight yet noticeable tilting. The setup is imaged by a monochromatic CMOS sensor (FLIR, GS3-U3-51S5M-C, pixel size of $3.45\,\mu\mathrm{m}$). Without knowing the exact position and angle parameters $\boldsymbol{\theta}$, e.g., light source position, sensor to lens distancing, lens yaw/pitch angles, it is very challenging to reproduce experimental measurements by manual parameter tuning a simulation setup. In Fig. 17, we show the success of **dO** to estimate such misalignment parameters, by minimizing the MSE error between the simulation image $I(\boldsymbol{\theta})$ and the target real captured image I_{real}. We employ the forward mode to render $I(\boldsymbol{\theta})$. To escape local stationary points and to regularize gradient computation, we enforce centroid alignment between the two images, by denoting $\mathcal{C}(\cdot)$ as an operator for calculating image centroid:

$$\min_{\boldsymbol{\theta}} \; \|I(\boldsymbol{\theta}) - I_{\text{real}}\|^2 + \mu \|\mathcal{C}(I(\boldsymbol{\theta})) - \mathcal{C}(I_{\text{real}})\|^2, \tag{15}$$

where μ is a tradeoff parameter to balance between MSE and alignment errors. Empirically, (15) is optimized by Adam and damped least squares in alternation. The optimization usually takes $< 0.5\,\mathrm{min}$ to finish for a megapixel image resolution on a GPU (Nvidia, GeForce RTX 2080 Ti).

Two lenses of different focal lengths (Thorlabs, LA1131, focal length of $50\,\mathrm{mm}$; LA1986, focal length of $125\,\mathrm{mm}$) were under investigation, as shown in Fig. 17(b) and Fig. 17(c). In Fig. 17(b), the lens was focused but tilted, and the goal is to re-parameterize the simulation to fit real measurements. In Fig. 17(c), the lens was slightly de-focused, introducing a blurry bright disk on the sensor plane. With an increase of angular misalignment, the image smears and elongates in the horizontal direction. Initialized from a coarse setup estimation, the final optimized images match real images visually well. Refer to Supplemental Document for full results. This example proves the validity of the differentiability of **dO**, and demonstrates the possibility of using **dO** for setup calibration. The simplicity of using the proposed engine for misalignment estimation allows integrating self-calibration into existing numerical modeling pipelines. From a broader perspective, we demonstrate the possibility of using a differentiable ray tracer such as **dO**, as a general inverse solver for metrology problems.

VI. Discussion and Conclusion

In principle, ray optics limits the application range of **dO** in that the wave nature of light is ignored. Resolution demanding imaging applications towards diffraction-limited performance such as telescope or microscope designs are hence not possible at this point. In methodology, we rely on gradient descent and damped least squares as the optimization techniques, and thus a number of iteration steps are required for convergence. Due to its local optimization nature, the solver suffers from the local minima problem and the initialization sensitivity issue. In software, **dO** relies mostly on reverse-mode AD, which is known to be memory-consuming especially for large amount of Monte Carlo samples, though partially alleviated, can still limit the number of rays used when optimizing the design. More memory-efficient approaches could be explored. We also expect faster speed performance after careful code improvements.

The current ray tracing engine could be further enhanced. From design perspective, more surface representations could be implemented, e.g., Laguerre, Hermite, and Zernike polynomials. New tracing methods are possible, for example paraxial tracing [67] and Gaussian beam tracing [68]. Also, new features could be implemented, e.g., diffraction and gratings, Fresnel equation and polarization ray tracing, coatings, stray light and ghost analysis, and non-sequential tracing. From application perspective, further applications could be explored, e.g., lens metrology [69], realistic lens rendering [70], and wavefront sensor designs [71], [72]. Hardware-software end-to-end optimization for domain-specific applications [10], [11], [12], [62], or semi-supervised training for automatic lens design [33], [34] are also target topics. We believe **dO** serves as an initial starting point towards these applications.

To conclude, we have proposed **dO**, a differentiable ray tracing engine for lens design. Board applications are demonstrated, ranging from classical design optimization and sensitivity analysis, to computationally intensive freeform design, end-to-end learning for computational imaging applications, and to challenging experimental misalignment estimation. We envision the potential of **dO**, as it opens up an exciting aspect to bring first-order gradient insights into lens design and relevant problems.

References

[1] Zemax, “Zemax opticstudio,” 2013. [Online]. Available: https://www.zemax.com/

[2] Synopsys, Inc., “Code V,” Accessed: Oct. 2022. [Online]. Available: https://www.synopsys.com/optical-solutions/codev.html

[3] E. R. Dowski and W. T. Cathey, “Extended depth of field through wavefront coding,” *Appl. Opt.*, vol. 34, no. 11, pp. 1859–1866, 1995.

[4] A. Levin, R. Fergus, F. Durand, and W. T. Freeman, “Image and depth from a conventional camera with a coded aperture,” *ACM Trans. Graph.*, vol. 26, no. 3, 2007, Art. no. 70.

[5] C. Zhou, S. Lin, and S. Nayar, “Coded aperture pairs for depth from defocus,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 325–332.

[6] O. Cossairt and S. Nayar, “Spectral focal sweep: Extended depth of field from chromatic aberrations,” in *Proc. IEEE Int. Conf. Comput. Photogr.*, 2010, pp. 1–8.

[7] R. Ng and P. Hanrahan, “Light field photography with a hand-held plenoptic camera,” *Comput. Sci. Tech. Rep.*, vol. 2, no. 11, pp. 1–11, 2005.

[8] A. Veeraraghavan, R. Raskar, A. Agrawal, A. Mohan, and J. Tumblin, “Dappled photography,” *ACM Trans. Graph.*, vol. 26, no. 3, 2007, Art. no. 69.

[9] P. E. Debevec and J. Malik, “Recovering high dynamic range radiance maps from photographs,” in *Proc. 24th Annu. Conf. Comput. Graph. Interactive Techn.*, 1997, pp. 369–378. [Online]. Available: https://doi.org/10.1145/258734.258884

[10] V. Sitzmann et al., “End-to-end optimization of optics and image processing for achromatic extended depth of field and super-resolution imaging,” *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–13, 2018.

[11] Y. Peng, Q. Sun, X. Dun, G. Wetzstein, W. Heidrich, and F. Heide, “Learned large field-of-view imaging with thin-plate optics,” *ACM Trans. Graph.*, vol. 38, no. 6, pp. 1–14, 2019.

[12] X. Dun, H. Ikoma, G. Wetzstein, Z. Wang, X. Cheng, and Y. Peng, “Learned rotationally symmetric diffractive achromat for full-spectrum computational imaging,” *Optica*, vol. 7, no. 8, pp. 913–922, 2020.

[13] Q. Sun, C. Wang, Q. Fu, X. Dun, and W. Heidrich, “End-to-end complex lens design with differentiate ray tracing,” *ACM Trans. Graph.*, vol. 40, no. 4, pp. 1–13, 2021.

[14] E. Tseng et al., “Differentiable compound optics and processing pipeline optimization for end-to-end camera design,” *ACM Trans. Graph.*, vol. 40, no. 4, pp. 1–19, 2021.

[15] J. Chang, V. Sitzmann, X. Dun, W. Heidrich, and G. Wetzstein, “Hybrid optical-electronic convolutional neural networks with optimized diffractive optics for image classification,” *Sci. Rep.*, vol. 8, no. 1, pp. 1–10, 2018.

[16] J. Chang and G. Wetzstein, “Deep optics for monocular depth estimation and 3D object detection,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 10193–10202.

[17] Y. Wu, V. Boominathan, H. Chen, A. Sankaranarayanan, and A. Veeraraghavan, “Phasecam3-D – Learning phase masks for passive single view depth estimation,” in *Proc. IEEE Int. Conf. Comput. Photogr.*, 2019, pp. 1–12.

[18] H. Ikoma, C. M. Nguyen, C. A. Metzler, Y. Peng, and G. Wetzstein, “Depth from defocus with learned optics for imaging and occlusion-aware depth estimation,” in *Proc. IEEE Int. Conf. Comput. Photogr.*, 2021, pp. 1–12.

[19] K. Monakhova, J. Yurtsever, G. Kuo, N. Antipa, K. Yanny, and L. Waller, “Learned reconstructions for practical mask-based lensless imaging,” *Opt. Exp.*, vol. 27, no. 20, pp. 28075–28090, 2019.

[20] Q. Sun, E. Tseng, Q. Fu, W. Heidrich, and F. Heide, “Learning rank-1 diffractive optics for single-shot high dynamic range imaging,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1386–1396.

[21] C. A. Metzler, H. Ikoma, Y. Peng, and G. Wetzstein, “Deep optics for single-shot high-dynamic-range imaging,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1375–1385.

[22] S. Tan, Y. Wu, S.-I. Yu, and A. Veeraraghavan, “Codedstereo: Learned phase masks for large depth-of-field stereo,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 7170–7179.

[23] K. Yanny et al., “Miniscope3d: Optimized single-shot miniature 3D fluorescence microscopy,” *Light: Sci. Appl.*, vol. 9, no. 1, 2020, Art. no. 171.

[24] H. Pinkard et al., “Learned adaptive multiphoton illumination microscopy for large-scale immune response imaging,” *Nature Commun.*, vol. 12, no. 1, 2021, Art. no. 1916.

[25] V. Boominathan, J. K. Adams, J. T. Robinson, and A. Veeraraghavan, “Phlatcam: Designed phase-mask based thin lensless camera,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 7, pp. 1618–1629, Jul. 2020.

[26] J. N. Martel, L. K. Mueller, S. J. Carey, P. Dudek, and G. Wetzstein, “Neural sensors: Learning pixel exposures for HDR imaging and video compressive sensing with programmable sensors,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 7, pp. 1642–1653, Jul. 2020.

[27] G. Wetzstein et al., “Inference in artificial intelligence with deep optics and photonics,” *Nature*, vol. 588, no. 7836, pp. 39–47, 2020.

[28] Z. Ballard, C. Brown, A. M. Madni, and A. Ozcan, “Machine learning and computation-enabled intelligent sensor design,” *Nature Mach. Intell.*, vol. 3, no. 7, pp. 556–565, 2021.

[29] M. Hillenbrand, A. Hoffmann, D. P. Kelly, and S. Sinzinger, “Fast nonparaxial scalar focal field calculations,” *Schedae Inform.*, vol. 31, pp. 169–175, 2014.

[30] J.-B. Volatier, Álvaro Menduiña-Fernández, and M. Erhard, “Generalization of differential ray tracing by automatic differentiation of computational graphs,” *J. Opt. Soc. Amer. A*, vol. 34, no. 7, pp. 1146–1151, 2017.

[31] C. Wang, N. Chen, and W. Heidrich, “Lens design optimization by back-propagation,” in *Proc. Int. Opt. Des. Conf.*, 2021, Art. no. 120781O.

[32] O. Manzyuk, B. A. Pearlmutter, A. Radul, D. Rush, and J. M. Siskind, “Perturbation confusion in forward automatic differentiation of higher-order functions,” *J. Funct. Program.*, vol. 29, 2019, Art. no. e12.

[33] G. Côté, J.-F. Lalonde, and S. Thibault, “Extrapolating from lens design databases using deep learning,” *Opt. Exp.*, vol. 27, no. 20, pp. 28279–28292, 2019.

[34] G. Côté, J.-F. Lalonde, and S. Thibault, “Deep learning-enabled framework for automatic lens design starting point generation,” *Opt. Exp.*, vol. 29, no. 3, pp. 3841–3854, 2021.

[35] A. Halé, P. Trouvé-Peloux, and J.-B. Volatier, “End-to-end sensor and neural network design using differential ray tracing,” *Opt. Exp.*, vol. 29, no. 21, pp. 34748–34761, 2021.

[36] Z. Li, Q. Hou, Z. Wang, F. Tan, J. Liu, and W. Zhang, “End-to-end learned single lens design using fast differentiable ray tracing,” *Opt. Lett.*, vol. 46, no. 21, pp. 5453–5456, 2021.

[37] M. Nimier-David, D. Vicini, T. Zeltner, and W. Jakob, “Mitsuba 2,” *ACM Trans. Graph.*, vol. 38, no. 6, pp. 1–17, 2019.

[38] J. Krishna Murthy, G. Iyer, and L. Paull, “gradslam: Dense slam meets automatic differentiation,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 2130–2137.

[39] S. Ghosh, Y. S. G. Nashed, O. Cossairt, and A. Katsaggelos, “ADP: Automatic differentiation ptychography,” in *Proc. IEEE Int. Conf. Comput. Photogr.*, 2018, pp. 1–10.

[40] S. Kandel, S. Maddali, M. Allain, S. O. Hruszkewycz, C. Jacobsen, and Y. S. G. Nashed, "Using automatic differentiation as a general framework for ptychographic reconstruction," *Opt. Exp.*, vol. 27, no. 13, pp. 18653–18672, 2019.
[41] Q. Guo et al., "Raycast calibration for augmented reality hmds with off-axis reflective combiners," in *Proc. IEEE Int. Conf. Comput. Photogr.*, 2020, pp. 1–12.
[42] C. Wang, N. Chen, and W. Heidrich, "Towards self-calibrated lens metrology by differentiable refractive deflectometry," *Opt. Exp.*, vol. 29, no. 19, pp. 30284–30295, Aug. 2021.
[43] E. Bostan, R. Heckel, M. Chen, M. Kellman, and L. Waller, "Deep phase decoder: Self-calibrating phase microscopy with an untrained deep neural network," *Optica*, vol. 7, no. 6, pp. 559–562, 2020.
[44] M. Kellman et al., "Memory-efficient learning for large-scale computational imaging," *IEEE Trans. Comput. Imag.*, vol. 6, pp. 1403–1414, 2020.
[45] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen, "Differentiable monte carlo ray tracing through edge sampling," *ACM Trans. Graph.*, vol. 37, no. 6, 2018, Art. no. 222.
[46] M. Nimier-David, S. Speierer, B. Ruiz, and W. Jakob, "Radiative backpropagation," *ACM Trans. Graph.*, vol. 39, no. 4, 2020, Art. no. 146.
[47] C. Wang, N. Chen, and W. Heidrich, "dO Source Code," 2022, [Online]. Available: https://github.com/vccimaging/diffOptics
[48] H. Ries and J. Muschaweck, "Tailored freeform optical surfaces," *J. Opt. Soc. Amer. A*, vol. 19, no. 3, pp. 590–595, 2002.
[49] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Assoc., Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
[50] J. Meiron, "Damped least-squares method for automatic lens design," *J. Opt. Soc. Amer.*, vol. 55, no. 9, pp. 1105–1109, 1965.
[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980
[52] J. M. Geary, *Introduction to Lens Design: With Practical ZEMAX Examples*. Richmond, VA, USA: Willmann-Bell, 2002.
[53] C. Kolb, D. Mitchell, and P. Hanrahan, "A realistic camera model for computer graphics," in *Proc. 22nd Annu. Conf. Comput. Graph. Interactive Techn.*, 1995, pp. 317–324.
[54] M. Born and E. Wolf, *Principles of Optics*, 7th ed. Cambridge, U.K.: Cambridge Univ. Press, 2019.
[55] J. T. Kajiya, "The rendering equation," in *Proc. 13th Annu. Conf. Comput. Graph. Interactive Techn.*, 1998, pp. 157–164.
[56] J. G. Baker, "Highly corrected objective having two inner divergent meniscus components between collective components," U.S. Patent 2532751, 1949.
[57] M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*. San Mateo, CA, USA: Morgan Kaufmann, 2016.
[58] M. Rimmer, "Analysis of perturbed lens systems," *Appl. Opt.*, vol. 9, no. 3, pp. 533–537, 1970.
[59] D. P. Feder, "Differentiation of ray-tracing equations with respect to construction parameters of rotationally symmetric optics," *J. Opt. Soc. Amer.*, vol. 58, no. 11, pp. 1494–1505, 1968.
[60] Y. Schwartzburg, R. Testuz, A. Tagliasacchi, and M. Pauly, "High-contrast computational caustic design," *ACM Trans. Graph.*, vol. 33, no. 4, 2014, Art. no. 74. [Online]. Available: https://doi.org/10.1145/2601097.2601200
[61] G. Damberg and W. Heidrich, "Efficient freeform lens optimization for computational caustic displays," *Opt. Exp.*, vol. 23, no. 8, pp. 10224–10232, 2015. [Online]. Available: https://doi.org/10.1364/OE.23.010224
[62] M. D. Robinson and D. G. Stork, "Joint design of lens systems and digital image processing," in *Proc. Int. Opt. Des. Conf.*, 2006, Paper no. WB4.
[63] A. Levin, S. W. Hasinoff, P. Green, F. Durand, and W. T. Freeman, "4-D frequency analysis of computational cameras for depth of field extension," *ACM Trans. Graph.*, vol. 28, no. 3, 2009, Art. no. 97.
[64] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.- Assist. Interv.*, 2015, pp. 234–241.
[65] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015. [Online]. Available: http://arxiv.org/abs/1409.1556
[66] O. Kupyn, T. Martyniuk, J. Wu, and Z. Wang, "DeblurGAN-v2: Deblurring (orders-of-magnitude) faster and better," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 8878–8887.
[67] R. R. Shannon, *The Art and Science of Optical Design*. Cambridge, U.K.: Cambridge Univ. Press, 1997.
[68] G. A. Deschamps, "Gaussian beam as a bundle of complex rays," *Electron. Lett.*, vol. 7, no. 23, pp. 684–685, 1971.
[69] D. Wang et al., "Simultaneous multisurface measurement of freeform refractive optics based on computer-aided deflectometry," *Optica*, vol. 7, no. 9, pp. 1056–1064, 2020.
[70] M. Hullin, E. Eisemann, H.-P. Seidel, and S. Lee, "Physically-based real-time lens flare rendering," in *ACM Trans. Graph.*. New York, NY, USA: ACM2011, pp. 1–10.
[71] C. Wang, X. Dun, Q. Fu, and W. Heidrich, "Ultra-high resolution coded wavefront sensor," *Opt. Exp.*, vol. 25, no. 12, pp. 13736–13746, 2017. [Online]. Available: https://doi.org/10.1364/OE.25.013736
[72] C. Wang, Q. Fu, X. Dun, and W. Heidrich, "Modeling classical wavefront sensors," *Opt. Exp.*, vol. 28, no. 4, pp. 5273–5287, 2020.

Congli Wang received the B.Eng. degree in electrical engineering from Tianjin University, Tianjin, China, in 2015, and the M.Sc. and Ph.D. degrees in electrical engineering from the King Abdullah University of Science and Technology, Thuwal, Saudi Arabia, in 2016 and 2021. He is currently a Postdoctoral Researcher with the University of California, Berkeley, Berkeley, CA, USA. His research interests include computational imaging, wavefront sensing, and adaptive optics.

Ni Chen received the B.S. degree in software engineering from the Harbin Institute of Technology, Harbin, China, in 2008, the M.S. degree in electrical engineering from Chungbuk National University, Cheongju, South Korea, and the Ph.D. degree in electrical engineering from Seoul National University, Seoul, South Korea, in 2010 and 2014, respectively. She is currently a Researcher with the King Abdullah University of Science and Technology, Thuwal, Saudi Arabia. From 2014 to 2016, she was a Research Scientist with the University of Hong Kong, Hong Kong, and from 2016 to 2017, she was an Associate Professor with the Shanghai Institute of Optics and Fine Mechanics, Chinese Academy of Sciences, Beijing, China. From 2018 to 2019, she was a Research Assistant Professor with the Department of Electrical and Computer Engineering, Seoul National University. Her research interests include three-dimensional optical imaging and display.

Wolfgang Heidrich (Fellow, IEEE) received the Ph.D. degree from the University of Erlangen, Erlangen, Germany, in 1999. He was a Research Associate with the Computer Graphics Group of the Max-Planck-Institute for Computer Science in Saarbrucken, Germany, before joining UBC in 2000. He is currently a Professor of computer science with the King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia. He joined KAUST in 2014, after 13 years as a Faculty Member with the University of British Columbia, Vancouver, BC, Canada. His research interests include intersection of imaging, optics, computer vision, computer graphics, and inverse problems. His current research interests include computational imaging, focusing on hardware-software co-design of the next generation of imaging systems, with applications such as High-Dynamic Range imaging, compact computational cameras, hyperspectral cameras, to name just a few. He work on High-Dynamic Range Displays served as the basis for the technology behind Brightside Technologies, which was acquired by Dolby in 2007. He has chaired the papers program for both Siggraph Asia and the International Conference of Computational Photography among others. He was the recipient of a 2014 Humboldt Research Award.