dO: A differentiable engine for Deep Lens design of computational imaging systems –Supplemental Document–

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

Abstract

In this supplemental document we provide implementation details, discussions, and additional results in support of the main paper: Lens system details (Section I); Optimization details (Section II); Design prescription and optimization details (Section III); Additional details on end-to-end wavefront coding (Section IV); Additional details on end-to-end large field-of-view co-design (Section V); Misalignment estimation optimization details (Section VI).

I. Lens System Details

A. Optical surfaces

In this work, we consider three specific types of parameterized lens surfaces to represent lens and freeform surfaces, yet in theory alternative parameterization forms (see [1]) should also work. Surfaces are defined in a Cartesian coordinate system (x, y, z), with z-axis being chosen as the optical axis (if any). Recall that we have formulated optical surfaces as parameterized surfaces in implicit form $f(x, y, z; \boldsymbol{\theta}) = 0$ along with its spatial derivatives $\nabla f(x, y, z; \boldsymbol{\theta})$. Surface normals are normalized spatial derivatives, i.e., $\mathbf{n} = \nabla f / \|\nabla f\|$.

Aspheres: Let $\rho = x^2 + y^2$ since aspheric surfaces are axially symmetric. The sag distance function $s(\rho)$ of aspheric surfaces and its derivative with respect to ρ are:

$$s(\rho) = \frac{c\rho}{1 + \sqrt{1 - \alpha\rho}} + \sum_{i=2}^{n} a_{2i}\rho^i, \tag{S1}$$

$$s'(\rho) = c\frac{1 + \sqrt{1 - \alpha\rho} - \alpha\rho/2}{\sqrt{1 - \alpha\rho}\left(1 + \sqrt{1 - \alpha\rho}\right)^2} + \sum_{i=2}^{n} a_{2i} i \rho^{i-1}, \tag{S2}$$

where c is the curvature, $\alpha = (1 + \kappa)c^2$ with κ being the conic coefficient, and a_{2i}'s are higher-order coefficients. Spherical surfaces are special cases of aspheric surfaces, with $\kappa = 0$ and $a_{2i} = 0\,(i = 2, \cdots, n)$. In implicit form:

$$f(x, y, z; \boldsymbol{\theta}) = s(\rho) - z, \tag{S3}$$

$$\nabla f(x, y, z; \boldsymbol{\theta}) = \left(2s'(\rho)x, 2s'(\rho)y, -1\right), \tag{S4}$$

where differentiable parameters $\boldsymbol{\theta} = (c, \kappa, a_{2i})$.

XY polynomials: XY polynomial surfaces extend lens surface representation beyond axial symmetry. The implicit surface function $f(x, y, z; \boldsymbol{\theta})$ and its spatial derivatives are:

$$f(x, y, z; \boldsymbol{\theta}) = \sum_{j=0}^{J}\sum_{i=0}^{j} a_{i,j} x^i y^{j-i} + bz^2 - z, \tag{S5}$$

$$\nabla f(x, y, z; \boldsymbol{\theta}) = \left(\sum_{j=1}^{J}\sum_{i=0}^{j} a_{i,j} i x^{i-1} y^{j-i}, \sum_{j=1}^{J}\sum_{i=0}^{j} a_{i,j}(j-i) x^i y^{j-i-1}, 2bz - 1\right), \tag{S6}$$

where differentiable parameters $\boldsymbol{\theta} = (b, a_{i,j})$.

B-splines: We employ B-splines [2] to represent high degree-of-freedom freeform surfaces. In general, the sag distance function $g(x, y)$ is represented as a spline of degree (in our case, it is three, i.e. the cubic B-spline) on the rectangle area, with predefined number of knots and knot positions. With that, spline functions $S_{i,j}(x, y)$ are fixed, and $g(x, y)$ is determined by spline coefficients $c_{i,j}$:

$$f(x, y, z; \boldsymbol{\theta}) = \sum_{i}^{n}\sum_{j}^{m} c_{i,j} S_{i,j}(x, y) - z, \tag{S7}$$

$$\nabla f(x, y, z; \boldsymbol{\theta}) = \left(\sum_{i}^{n}\sum_{j}^{m} c_{i,j} \nabla_x S_{i,j}(x, y), \sum_{i}^{n}\sum_{j}^{m} c_{i,j} \nabla_y S_{i,j}(x, y), -1\right), \tag{S8}$$

where differentiable parameters $\boldsymbol{\theta} = (c_{i,j})$, and the spatial gradients of the spline functions $\nabla_x S_{i,j}$ and $\nabla_x S_{i,j}$ are efficiently evaluated via modified de-Boor's algorithm [2].

B. Optical elements

Optical elements are defined to contain at least two optical surfaces. To enable sensitivity analysis and further degree-of-freedom, each optical element is described by differentiable positional parameters $\boldsymbol{\theta} = (\theta_x, \theta_y, \mathbf{p}_{\text{shift}})$. Figure 1 demonstrates the physical meaning of these parameters, given a doubly convex lens (Thorlabs, LB1757).

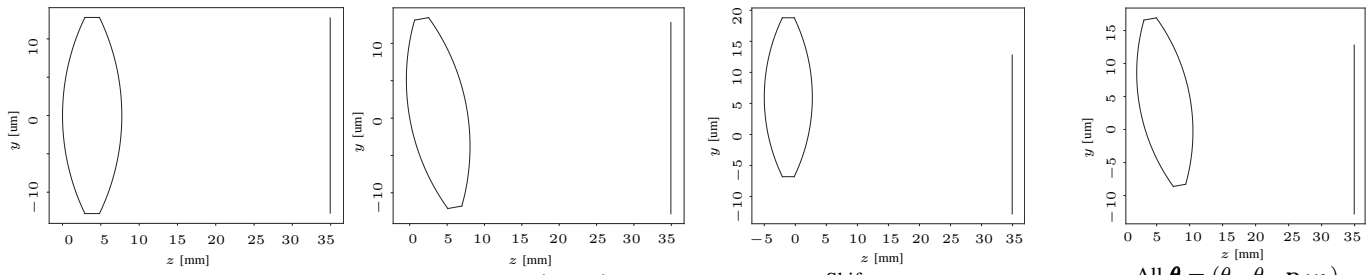


Fig. 1: Positional parameterization of an optical element.

C. Lens system

We follow the standard lens design pipeline [3], [4] 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 (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. This is demonstrated in Figure 2 with a double-Gauss lens [5]. 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.

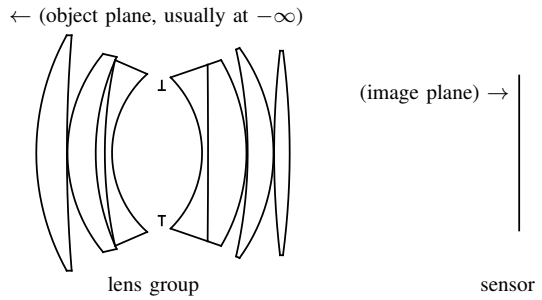
Depending on the needs, rays can be traced through the lens system in two different modes, *forward mode* or *backward mode*. In the forward mode, rays are traced starting from the object plane towards the image plane. This is the preferable way in lens design for aberration analysis, e.g., generation 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.

The above two ray tracing procedures are unitedly described as in Algorithm 1: The lens system can be formulated as a "black-box" operator $\mathcal{A}(\cdot)$ that is a function of all lens parameters $\boldsymbol{\theta}$. The lens system $\mathcal{A}$ transforms input ray $\{\mathbf{o}_{\text{in}}, \mathbf{d}_{\text{in}}\}$ into output ray $\{\mathbf{o}_{\text{out}}, \mathbf{d}_{\text{out}}\}$ at wavelength λ:

$$\mathcal{A}(\{\mathbf{o}_{\text{in}}, \mathbf{d}_{\text{in}}\}, \lambda; \boldsymbol{\theta}) = \{\mathbf{o}_{\text{out}}, \mathbf{d}_{\text{out}}\}. \tag{S9}$$

Ray propagation through a lens system involves two major steps, finding the ray-surface intersection point, 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.

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.



type	distance	1/curvature	diameter	material
O	20	0	100	AIR
S	5.0	78.360	76.0	1.79668/45.5
S	9.8837	469.477	76.0	AIR
S	0.1938	50.297	64.0	1.77279/49.4
S	9.1085	74.376	62.0	AIR
S	2.9457	138.143	60.0	1.6727/32.2
S	2.3256	34.326	51.0	AIR
A	16.0698	0	49.6	OCCULDER
S	13.0	-34.407	48.8	1.74/28.3
S	1.938	-2906.977	57.0	1.77279/49.4
S	12.403	-59.047	60.0	AIR
S	0.3876	-150.890	66.8	1.78797/47.5
S	8.333	-57.890	67.8	AIR
S	0.1938	284.630	66.0	1.78797/47.5
S	5.0388	-253.217	66.0	AIR
I	74.1	0	86.53	AIR

Fig. 2: Lens system schematic and its prescription file.

Algorithm 1 Ray tracing through a lens system $\mathcal{A}$.

procedure $\mathcal{A}(\{\mathbf{o}_{\text{in}}, \mathbf{d}_{\text{in}}\};\ \lambda)$
  Initialize $\{\mathbf{o}^{(0)}, \mathbf{d}^{(0)}\} \leftarrow \{\mathbf{o}_{\text{in}}, \mathbf{d}_{\text{in}}\}$
  for optical surface f_i $(\forall i = 1, \cdots, N)$ **do**
    Find intersection point $\mathbf{o}^{(i)}$
    Compute refraction direction $\mathbf{d}^{(i)}$
    if valid **then**
      update ray as $\{\mathbf{o}^{(i)}, \mathbf{d}^{(i)}\}$
    end if
  end for
  return $\{\mathbf{o}_{\text{out}}, \mathbf{d}_{\text{out}}\} \leftarrow \{\mathbf{o}^{(N)}, \mathbf{d}^{(N)}\}$
end procedure

II. Optimization

Recall to optimize a design, a merit function $\epsilon(\cdot)\colon \mathbb{R}^{2m} \mapsto \mathbb{R}$ 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:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}}\ \epsilon(\mathbf{p}(\boldsymbol{\theta})). \tag{S10}$$

Given the derivatives $\Delta\boldsymbol{\theta}$ from back-propagating Eq. (S10), $\mathbf{dO}$ 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, maximum system overall size, Eq. (S10) can be modified by adding a vector constraint function.

A. Optimization spirit

Specific optimization method depends on the number of variables. When the number of variables is small (for example $\boldsymbol{\theta} \in \mathbb{R}^n$, $n < 20$), damped least squares [6] are employed to efficiently optimize Eq. (S10), 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 $\mathbf{dO}$. When n is large, popular gradient descent methods such as Adam [7] is employed. If desirable, additional regularization terms are possible, for example when solving Eq. (S21). This optimization flexibility feature differentiates $\mathbf{dO}$ from existing software.

Recall our goal is to perform design optimization by minimizing the error metric function in Eq. (S10), for the engine to find a set of optimal parameters $\boldsymbol{\theta}^*$ that minimizes $\epsilon(\mathbf{p}(\boldsymbol{\theta}))$. The scalar-valued error function $\epsilon(\mathbf{p}(\cdot))$ is not necessarily linear and can only be evaluated numerically. Depending on the dimensionality of variables, gradient descent or damped least squares are employed to optimize Eq. (S10).

B. Unconstrained optimization methods

Gradient descent and variants: Since gradient information is available from back-propagation, Eq. (S10) can be easily optimized using gradient descent methods such as Adam [7], with the learning rate α_k being strategically tuned for each iteration k:

$$\boldsymbol{\theta}^{k+1} \leftarrow \boldsymbol{\theta}^k + \Delta\boldsymbol{\theta}, \qquad \Delta\boldsymbol{\theta} = -\alpha_k \left.\frac{\partial \epsilon}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^k}. \tag{S11}$$

Damped least squares: When the error function follows the form of $\epsilon(\boldsymbol{\theta}) = \sum_i \|\mathcal{A}_i(\boldsymbol{\theta})\|^2$ where $\mathcal{A}_i(\cdot)$ is a function that can be numerically evaluated, and the total number of variables is small, we employ the well-known damped least squares [6], [8] for optimizing Eq. (S10), which can be re-written as follows to simply notation:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}}\ \epsilon(\boldsymbol{\theta}), \quad \epsilon(\boldsymbol{\theta}) = \sum_i \|\mathcal{A}_i(\boldsymbol{\theta})\|^2. \tag{S12}$$

This problem is solved in an iterative fashion. At each iteration k, the damped least squares method solves for a least squares sub-problem with respect to a small variable change $\Delta\boldsymbol{\theta}$, which is Tikhonov regularized to enhance solution stability, with an iterative changing damping factor ρ_k:

$$\boldsymbol{\theta}^{k+1} \leftarrow \boldsymbol{\theta}^k + \arg\min_{\Delta\boldsymbol{\theta}}\ \sum_i \|\mathcal{A}_i(\boldsymbol{\theta}^k + \Delta\boldsymbol{\theta})\|^2 + \rho_k \|\Delta\boldsymbol{\theta}\|^2. \tag{S13}$$

By approximating $\mathcal{A}_i(\boldsymbol{\theta}^k + \Delta\boldsymbol{\theta})$ using first-order Taylor expansion:

$$\mathcal{A}_i(\boldsymbol{\theta}^k + \Delta\boldsymbol{\theta}) \approx \mathcal{A}_i(\boldsymbol{\theta}^k) + \mathcal{J}_i \Delta\boldsymbol{\theta}, \tag{S14}$$

where $\mathcal{J}_i$ is the Jacobian matrix at $\mathcal{A}_i(\boldsymbol{\theta}^k)$, Eq. (S13) is solved by the normal equation, with $\mathbf{I}$ denoting the identity matrix:

$$\left(\sum_i \mathcal{J}_i^T \mathcal{J}_i + \rho_k \mathbf{I}\right) \Delta\boldsymbol{\theta}^k = -\sum_i \mathcal{J}_i^T \mathcal{A}_i(\boldsymbol{\theta}^k). \tag{S15}$$

One nice feature of using automatic differentiation is that the right hand side can be efficiently evaluated using back-propagation (the backward mode), with the left hand side being obtained using the forward mode.

C. Constraint handling

When there are constraints in the design, e.g., positive air-spacing, minimum glass thickness or back focal length, maximum air-spacing overall size, Eq. (S10) needs to be constrained, which can be re-phrased as two vectors $\mathbf{b}_l$ and $\mathbf{b}_h$, i.e., a bounding box constraint:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \ \epsilon(\boldsymbol{\theta}), \qquad \text{s.t.} \quad \mathbf{b}_l \leq \boldsymbol{\theta} \leq \mathbf{b}_h. \tag{S16}$$

This linear constraint turns the original unconstrained problem into a constrained one, preventing $\partial\epsilon/\partial\boldsymbol{\theta}$ to be evaluated at boundaries. Consequently, the unconstrained optimization methods in the previous subsection are revised by simply projecting the variable to a feasible solution space after obtaining $\Delta\boldsymbol{\theta}$ from Eq. (S11) or Eq. (S15), at each iteration k applying an element-wise maximum-minimum operation to $\boldsymbol{\theta}^k + \Delta\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^{k+1} \leftarrow \max(\mathbf{b}_l, \min(\boldsymbol{\theta}^k + \Delta\boldsymbol{\theta}, \mathbf{b}_h)). \tag{S17}$$

III. Design Prescriptions

A. Spherical aberration optimization

We minimize the spherical aberration of an asphere lens as described in Table I. We optimized these parameters using damped least squares.

TABLE I: Asphere lens prescription. Changed variables are marked in **bold**.

(a) Initial design

#	type	distance	1/curvature	diameter	material	conic	higher aspheric coefficients
0	O	0	∞	0	AIR		
1	AS	0	20.923	50.0	1.5229/58.50	-0.6405	2.0e-06
2	S	21.0	∞	50.0	AIR		
3	I	26.0	∞	50.0	AIR		

(b) Optimized design

#	type	distance	1/curvature	diameter	material	conic	higher aspheric coefficients
0	O	0	∞	0	AIR		
1	AS	0	**20.9403**	50.0	1.5229/58.50	**-0.6262**	**1.6405e-06**
2	S	21.0	∞	50.0	AIR		
3	I	26.0	∞	50.0	AIR		

B. Photographic camera design optimization

The Nikon lens prescription is shown in Table II. Our target is to reset the aspheric surfaces to spherical surfaces, and the goal is to use our ray tracing engine to minimize RMS spot size by optimizing all lens surface curvatures, the conic and aspherical coefficients of Surface 17 & 18. We optimized these parameters using damped least squares.

C. Sensitivity analysis

In sensitivity analysis we analyzed a nominal Cooke triplet design, shown in Table I. The goal was to analyze the potential effect of how each lens element misalignment would affect the optical performance of the design. These potential effects are illustrated by the Jacobian matrices in the main document.

TABLE I: Cooke triplet prescription.

#	type	distance	1/curvature	diameter	material
0	O	0.0	∞	0.0	AIR
1	S	0.0	22.014	19.0	1.6204/60.31
2	S	3.259	-435.760	19.0	AIR
3	S	6.008	-22.213	8.0	1.6200/36.37
4	S	1.000	20.292	7.72	AIR
5	S	4.750	79.684	15.0	1.6204/60.31
6	S	2.952	-18.395	15.0	AIR
7	I	42.208	∞	36.346	AIR

TABLE II: Nikon lens prescription. Surface 12 is the pupil plane, Surface 17 and 18 are aspheric surfaces, and Surface 23 is the sensor plane. Changed variables are marked in **bold**.

(a) Original design

#	type	distance	1/curvature	diameter	material	conic	higher aspheric coefficients
0	O	0	∞	0	AIR		
1	S	0	5.267	1.694	1.5168/64.12		
2	S	0.102	0.961	1.392	AIR		
3	S	0.309	1.442	1.322	1.9027/35.72		
4	S	0.246	10.280	1.250	1.5955/39.21		
5	S	0.083	1.215	1.092	AIR		
6	S	0.411	-1.099	1.048	1.6990/30.05		
7	S	0.088	2.918	1.172	1.9108/35.25		
8	S	0.258	-1.669	1.202	AIR		
9	S	0.009	1.643	1.248	1.5928/68.62		
10	S	0.379	-1.412	1.226	1.7205/34.70		
11	S	0.069	-2.572	1.214	AIR		
12	A	0.118	∞	1.110	OCCLUDER		
13	S	0.604	-0.973	0.952	1.5927/35.31		
14	S	0.051	-24.080	0.980	AIR		
15	S	0.009	2.376	1.086	1.5928/68.62		
16	S	0.282	-1.306	1.138	AIR		
17	AS	0.239	-7.317	1.208	1.6935/53.20	0	(-0.240, -0.4268)
18	AS	0.122	-2.200	1.254	AIR	0	(-0.05053, -0.3491, 0.1459, 0.07718)
19	S	0.154	-1.545	1.324	1.4875/70.44		
20	S	0.083	-7.257	1.424	AIR		
21	S	0.750	∞	2.400	1.5168/64.12		
22	S	0.074	∞	2.400	AIR		
23	I	0.043	∞	1.912	AIR		

(b) Optimized design

#	type	distance	1/curvature	diameter	material	conic	higher aspheric coefficients
0	O	0	∞	0	AIR		
1	S	0	**6.843**	1.694	1.5168/64.12		
2	S	0.102	**0.972**	1.392	AIR		
3	S	0.309	**1.439**	1.322	1.9027/35.72		
4	S	0.246	**11.065**	1.250	1.5955/39.21		
5	S	0.083	**1.232**	1.092	AIR		
6	S	0.411	**-1.095**	1.048	1.6990/30.05		
7	S	0.088	**2.825**	1.172	1.9108/35.25		
8	S	0.258	**-1.662**	1.202	AIR		
9	S	0.009	**1.644**	1.248	1.5928/68.62		
10	S	0.379	**-1.314**	1.226	1.7205/34.70		
11	S	0.069	**-2.537**	1.214	AIR		
12	A	0.118	∞	1.110	OCCLUDER		
13	S	0.604	**-0.980**	0.952	1.5927/35.31		
14	S	0.051	**-15.827**	0.980	AIR		
15	S	0.009	**2.502**	1.086	1.5928/68.62		
16	S	0.282	**-1.379**	1.138	AIR		
17	AS	0.239	**-8.237**	1.208	1.6935/53.20	**132.663**	**(-0.1154)**
18	AS	0.122	**-2.053**	1.254	AIR	**-0.883**	**(0.0349)**
19	S	0.154	**-1.581**	1.324	1.4875/70.44		
20	S	0.083	**-7.257**	1.424	AIR		
21	S	0.750	∞	2.400	1.5168/64.12		
22	S	0.074	∞	2.400	AIR		
23	I	0.043	∞	1.912	AIR		

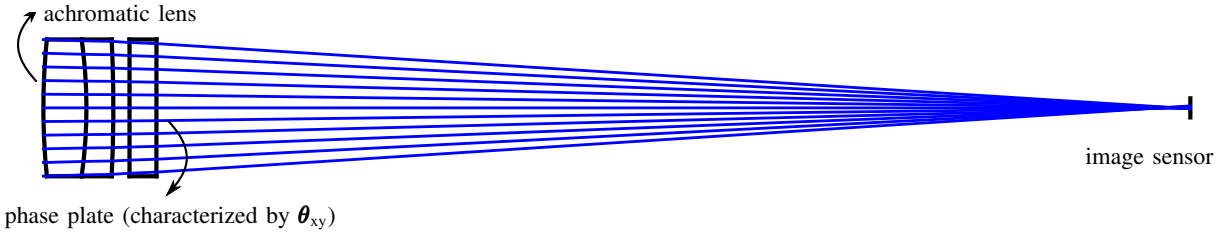


Fig. 3: Layout diagram of the end-to-end wavefront coding imaging system. The field of view of this F/8 imaging system is around 1°. Here, the central view is depicted.

IV. End-to-end wavefront coding implementation details

A. Imaging system and phase plate parameterization

The imaging system consists of an achromatic doublet and a phase plate, as shown in the ray tracing layout diagram Figure 3. Prescription of the imaging system is in Table II. The wavefront coding phase plate is parameterized by $\boldsymbol{\theta}_{\text{xy}} \in \mathbb{R}^4$, the cubic polynomial coefficients. Thus, the phase plate height map is described as:

$$h(x, y; \boldsymbol{\theta}_{\text{xy}}) = \boldsymbol{\theta}_{\text{xy},1} x^3 + \boldsymbol{\theta}_{\text{xy},2} x^2 y + \boldsymbol{\theta}_{\text{xy},3} x y^2 + \boldsymbol{\theta}_{\text{xy},4} y^3. \tag{S18}$$

TABLE II: Image system prescription.

#	type	distance	1/curvature	diameter	material	note
0	O	0.0	∞	0.0	AIR	
1	S	0.0	62.8	12.7	1.51680/64.17	
2	S	4.0	-45.7	12.7	1.67270/32.21	
3	S	6.5	-128.2	12.7	AIR	
4	S	8.0	∞	12.7	1.51680/64.17	
5	C	10.5	N.A.	12.7	AIR	phase plate; characterized by Eq. (S18)
6	I	96.0	∞	1.7664	AIR	

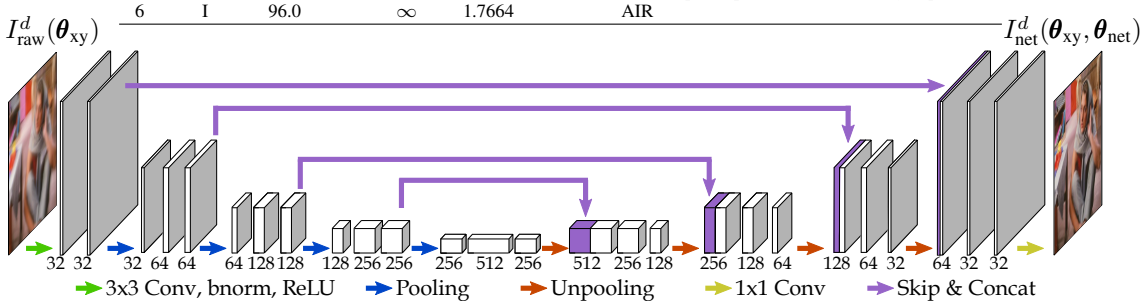


Fig. 4: U-Net architecture used in end-to-end training. All sub-stage blocks were connected by 3x3 Conv, batch-norm followed by ReLU (green arrows).

B. Image simulation

We assume the object is a planar texture image plane, located at distance d away from the imaging system. When using the backward mode for simulation image generation, the acquired raw image is a function of d and $\boldsymbol{\theta}_{\text{xy}}$, denoted as $I^d_{\text{raw}}(\boldsymbol{\theta}_{\text{xy}})$. Due to the huge memory consumption in the back-propagation pass, total number of ray samples is limited. The image resolution was set to 256 by 256, with a pixel pitch of $6.9\,\mu\text{m}$, with an exit pupil aperture sampling of 7 by 7. The exact pupil sampling location was randomly perturbed each time to avoid strike-like artifacts. Though the ray sampling rate is low, we have empirically verified that the rendered images were of high enough quality for post-processing by a neural network. We rendered channels of an RGB image at three different wavelengths, $656.2725\,\text{nm}$, $587.5618\,\text{nm}$, and $486.1327\,\text{nm}$.

C. Optimization and loss functions

In extended depth of field applications, the goal is to deconvolve $I^d_{\text{raw}}(\boldsymbol{\theta}_{\text{xy}})$ and recover the ground truth image I_{gt}. This task could be accomplished using a post-processing algorithm (here, a neural network) parameterized by $\boldsymbol{\theta}_{\text{net}}$ as a black-box solver, such that the post-processed output image is close to the ground truth image, for each d:

$$\text{Net}(I^d_{\text{raw}}(\boldsymbol{\theta}_{\text{xy}}); \boldsymbol{\theta}_{\text{net}}) = I^d_{\text{net}}(\boldsymbol{\theta}_{\text{xy}}, \boldsymbol{\theta}_{\text{net}}) \approx I_{\text{gt}}. \tag{S19}$$

Optimizing Eq. (S19) is a joint processing of optimizing optics parameterization $\boldsymbol{\theta}_{\text{xy}}$ and algorithm parameterization $\boldsymbol{\theta}_{\text{net}}$. We employed the standard U-Net [9] architecture as the post-processing algorithm, aiming to deconvolve the blurry raw images at different defocus distances. More advanced neural network architectures are possible, our choice here only offers a demonstration for the end-to-end capability of our differentiable engine. To optimize $\boldsymbol{\theta}_{\text{xy}}$ and $\boldsymbol{\theta}_{\text{net}}$, we generated a synthetic dataset for training, using samples from ImageNet [10]. To ensure a robust reconstruction, we optimize the following loss function (shown in the main paper):

$$\min_{\boldsymbol{\theta}_{\text{xy}}, \boldsymbol{\theta}_{\text{net}}} \mathcal{L}(\boldsymbol{\theta}_{\text{xy}}, \boldsymbol{\theta}_{\text{net}}) = \mathcal{L}_{\text{mse}} + \mathcal{L}_{\text{tv}} + \mathcal{L}_{\text{vgg16}}, \tag{S20}$$

by employing the following loss functions to regularize Eq. (S19):

$$\mathcal{L}_{\text{mse}} = \alpha_1 \| I^d_{\text{net}}(\boldsymbol{\theta}_{\text{xy}}, \boldsymbol{\theta}_{\text{net}}) - I_{\text{gt}} \|^2, \qquad \text{(MSE error)}$$

$$\mathcal{L}_{\text{tv}} = \alpha_2 \| \nabla I^d_{\text{net}}(\boldsymbol{\theta}_{\text{xy}}, \boldsymbol{\theta}_{\text{net}}) \|_1, \qquad \text{(total variation)}$$

$$\mathcal{L}_{\text{vgg16}} = \alpha_3 \| \mathcal{V}(I^d_{\text{net}}(\boldsymbol{\theta}_{\text{xy}}, \boldsymbol{\theta}_{\text{net}})) - \mathcal{V}(I_{\text{gt}}) \|^2, \qquad \text{(pretrained VGG-16 feature error)}$$

where $\alpha_i (i = 1, 2, 3)$ are trade-off parameters, $\mathcal{V}(\cdot)$ denotes extracted features from 15 layers from pretrained VGG-16. We implemented the network in PyTorch, and used Adam [7] for training, with a number of 800 epochs and 10 iterations for each image per d. The training took around three days to finish on a GPU (Nvidia, GeForce RTX 2080 Ti). Figure 4 shows the U-Net architecture. Additional results are shown in Figure 5.

V. End-to-end large field-of-view implementation details

A. Imaging system and training

The imaging system consists of a cemented doublet of an F/5, 40° large field-of-view end-to-end imaging system, as shown in the ray tracing layout diagram Figure 6. Prescription of this imaging system is in Table III. Bold fonts are differentiable

parameters that are changable during the end-to-end training. With only a doublet and three surfaces, compared to a standard Cooke triplet design, the total degree of freedom is insufficient to fully compensate most lower-order aberrations, and hence the blurriness is unavoidable. Here, we anticipate the trained neural network to act as an image reconstruction layer appended to remove the unwanted blurriness. Training (loss function and strategy) is similarly to Section IV, but the neural reconstruction algorithm is based on a modified version of Ref. [11]. See Figure 7 for additional results.

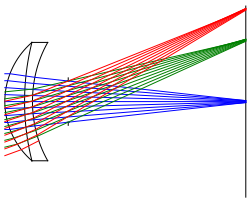


Fig. 6: Layout diagram.

#	type	distance	1/curvature	diameter	material	4^{th} aspheric coefficient
0	O	0.0	∞	0.0	AIR	
1	AS	0.0	**19.163**	30.0	1.62040/60.31	**0**
2	S	5.0	51.503	30.0	1.620/36.37	
3	AS	2.006	**30.347**	30.0	AIR	**0**
4	A	9.063	∞	10.466	OCCLUDER	
5	I	45.0	∞	48.437	AIR	

TABLE III: Image system initial prescription.

VI. Misalignment estimation

In the main paper, we demonstrate misalignment estimation with a revised cost function on MSE error metric. This is important for a successful estimation, because the overlapping area is too small for the current estimate I and the target real image I_{real}, as such the gradient of the MSE error $\|I - I_{\text{real}}\|^2$ is close to zero regardless of local spot movements, resulting in the so-called stagnation problem for optimization. To overcome this issue, we regularize the MSE error with a centroid alignment loss, with $\mathcal{C}(\cdot)$ being the centroid function for a given image. That is:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \ \epsilon(\boldsymbol{\theta}), \qquad \epsilon(\boldsymbol{\theta}) = \|I(\boldsymbol{\theta}) - I_{\text{real}}\|^2 + \mu\|\mathcal{C}(I(\boldsymbol{\theta})) - \mathcal{C}(I_{\text{real}})\|^2, \tag{S21}$$

where μ is a tradeoff parameter to balance between MSE error and spot position. We minimize Eq. (S21) by alternating between Adam and damped least squares. Specifically, each optimization contains 50 Adam + 15 damped least squares + 50 Adam + 15 damped least squares + 50 Adam iterations. We first optimized sensor distance and light source origin For $\theta_x = 0$ misalignment, then fixed these values for subsequent estimations, and optimized for lens origins instead. Optimization details are shown in Figure 8, Figure 9, Figure 10, Figure 11, and Figure 12.

References

[1] J. Ye, L. Chen, X. Li, Q. Yuan, and Z. Gao, "Review of optical freeform surface representation technique and its application," *Optical Engineering*, vol. 56, no. 11, p. 110901, 2017.

[2] C. de Boor, *A practical guide to splines*. springer-verlag New York, 1978, vol. 27.

[3] J. M. Geary, *Introduction to lens design: with practical ZEMAX examples*. Willmann-Bell Richmond, VA, USA:, 2002.

[4] C. Kolb, D. Mitchell, and P. Hanrahan, "A realistic camera model for computer graphics," in *Proceedings of the 22nd annual conference on computer graphics and interactive techniques*, 1995, pp. 317–324.

[5] K. Wakamiya, "Great aperture ratio lens," 1984, u.S. Patent 4448497.

[6] J. Meiron, "Damped least-squares method for automatic lens design," *Journal of the Optical Society of America*, vol. 55, no. 9, pp. 1105–1109, 1965.

[7] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *International Conference for Learning Representations*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[8] M. J. Kidger, "Use of the levenberg-marquardt (damped least-squares) optimization method in lens design," *Optical Engineering*, vol. 32, no. 8, pp. 1731–1739, 1993.

[9] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2009, pp. 248–255.

[11] O. Kupyn, T. Martyniuk, J. Wu, and Z. Wang, "DeblurGAN-v2: Deblurring (orders-of-magnitude) faster and better," in *IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8878–8887.

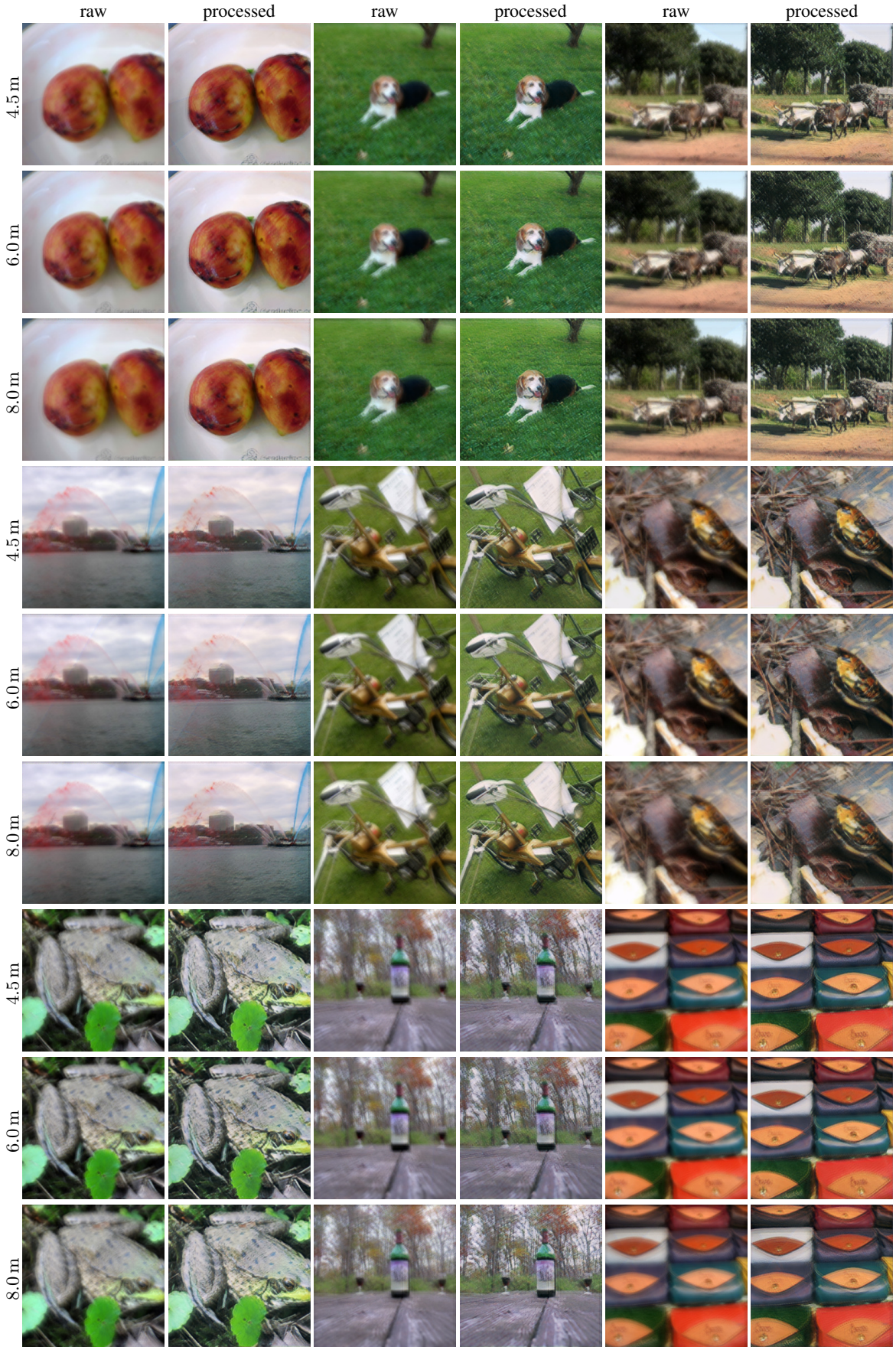
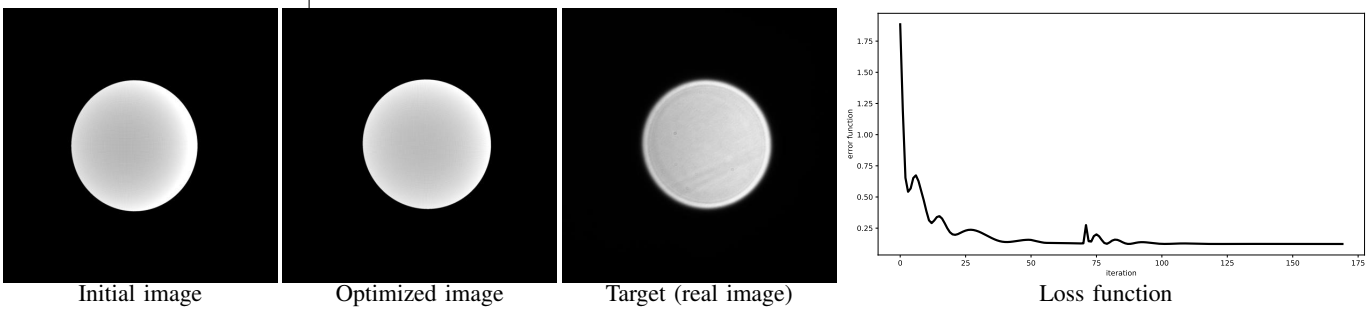


Fig. 5: Additional results on end-to-end wavefront coding. Here raw and post-processed images by the neural network are shown at three different defocus distances.

Fig. 7: Additonal results of large field-of-view end-to-end training.

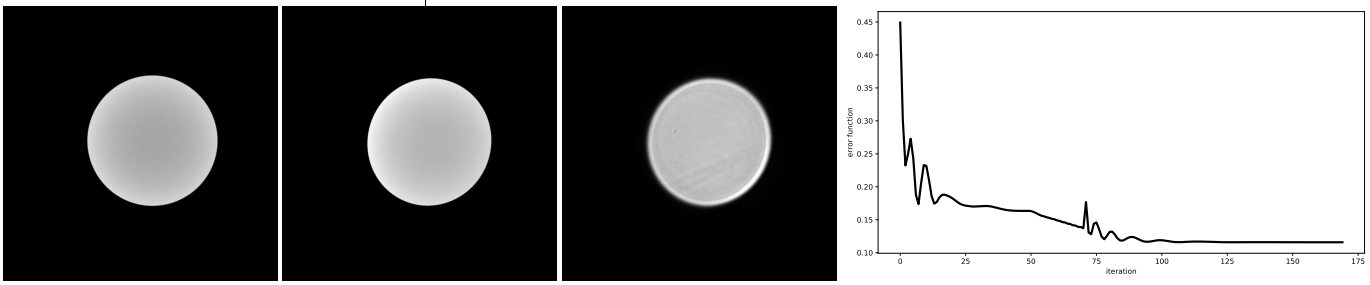
	sensor distance [mm]	light origin [mm,mm,mm]	lens yaw θ_x [°]	lens pitch θ_y [°]
initial	130.0	(0.0, 0.0, -660.0)	0	0
optimized	123.05	(0.65, -2.00, -864.01)	2.45	1.34



Initial image Optimized image Target (real image) Loss function

Fig. 8: Positional estimation for small angle (around $\theta_x = 0°$) misalignment.

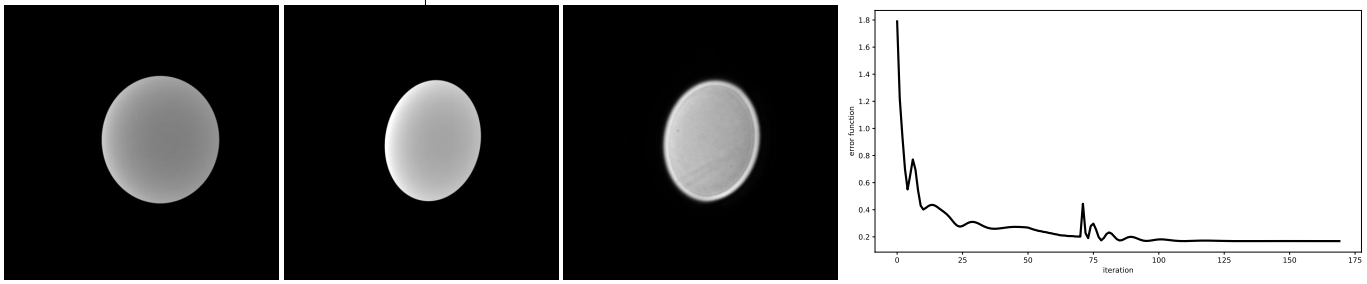
	lens origin [mm,mm,mm]	lens yaw θ_x [°]	lens pitch θ_y [°]
initial	(0, 0, 0)	-1	0
optimized	(-0.0065, -0.1122, -0.5569)	-2.72	1.20



Initial image Optimized image Target (real image) Loss function

Fig. 9: Positional estimation for small angle (around $\theta_x = 5°$) misalignment.

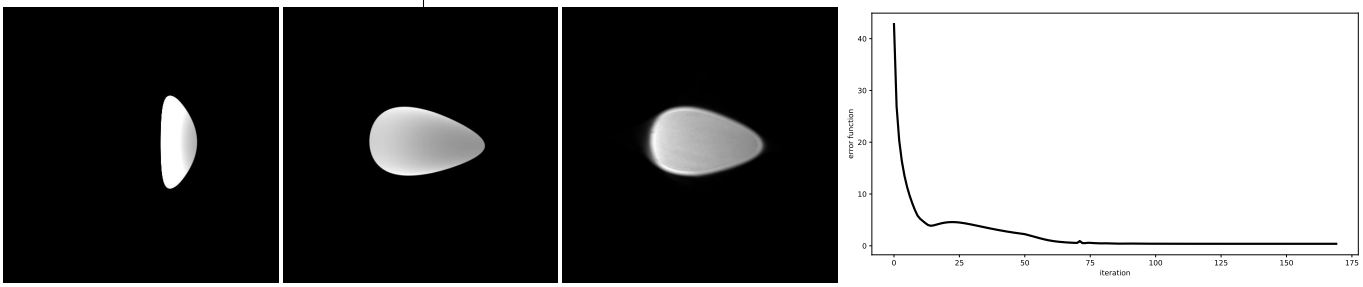
	lens origin [mm,mm,mm]	lens yaw θ_x [°]	lens pitch θ_y [°]
initial	(0, 0, 0)	-6	0
optimized	(-0.0239, -0.2938, -0.2993)	-9.6423	1.6439



Initial image Optimized image Target (real image) Loss function

Fig. 10: Positional estimation for small angle (around $\theta_x = 10°$) misalignment.

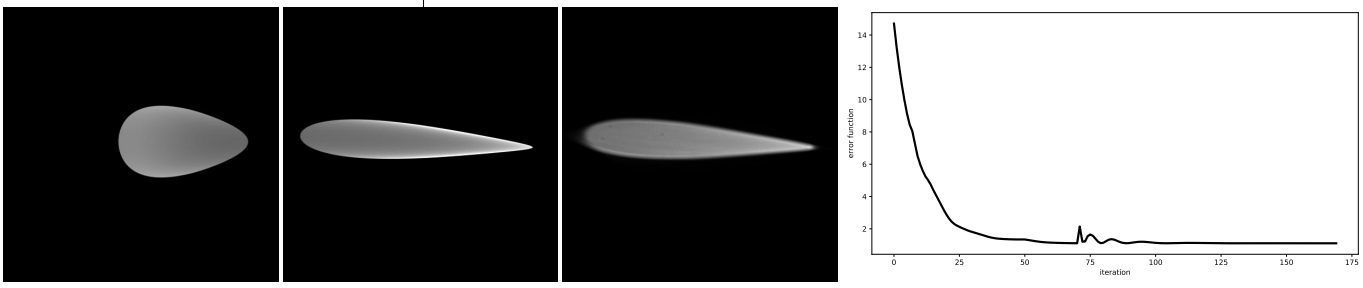
	lens origin [mm,mm,mm]	lens yaw θ_x [°]	lens pitch θ_y [°]
initial	(0, 0, 0)	-21	0
optimized	(-0.0229, -0.7511, -1.6763)	-25.0738	1.9460



Initial image Optimized image Target (real image) Loss function

Fig. 11: Positional estimation for small angle (around $\theta_x = 25°$) misalignment.

	lens origin [mm,mm,mm]	lens yaw θ_x [°]	lens pitch θ_y [°]
initial	(0, 0, 0)	-26	0
optimized	(-0.0251, -0.8970, -2.6649)	-30.0409	1.4138



Initial image Optimized image Target (real image) Loss function

Fig. 12: Positional estimation for small angle (around $\theta_x = 30°$) misalignment.