Computational Lens Design

Arjun Teh

CMU-CS-25-144

November 2025

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Ioannis Gkioulekas, Co-Chair
Matthew O'Toole, Co-Chair
James McCann
Bernd Bickel, ETH Zürich

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science.

This research was supported in part by the National Science Foundation, Google, and the Air Force Office of Scientific Research.

Keywords: Differentiable Rendering, Computational Design, Lens Design, Ray Tracing, Gradient Index Materials, Markov Chain Monte Carlo

To Pirate, Bandit, Renegade and Snowy
forcing me to stop and smell the roses.

Abstract

Cameras are ubiquitous in modern life, from smartphones to autonomous vehicles. To produce high-quality, aberration-free images, manufacturers invest significant resources in precision lens design in order to produce cameras that meet the demands of all these downstream applications. However, traditional lens design tools are limited to a set of lens design tasks, and the burden of designing lenses is largely placed on an expert designer. The methods developed in this thesis seek to address fundamental limitations of existing lens design tools, which only focus on optimizing continuous parameters of a fixed lens design for sharpness. These contributions enable the optimization of complex lens designs that were previously computationally expensive or impossible to optimize, which allows for designers to more quickly and efficiently explore the vast design space of lenses.

Firstly, we develop a method for calculating unbiased gradients of light throughput, enabling the optimization of lens speed. We show previous methods ignore the effect of the lens pupil when calculating gradients and thus result in biased estimates. In order to address the pupil, we devise an object referred to as a warp field that accurately accounts for the effects of the pupil on the objective function. With pupil aware gradients, we enable the direct optimization of lens speed, or light throughput. We demonstrate the effectiveness of our method on a variety of lens design tasks, including enumerating a tradeoff space between throughput and sharpness.

Secondly, we develop a Markov chain Monte Carlo (MCMC) method that combines gradient-based optimization of continuous parameters with discrete mutations that change the number of elements in a lens design. A lens with a fixed number of elements is limited by a Pareto front the describes how much throughput and sharpness can be achieved. By allowing the number of elements to change during optimization, we can explore a larger design space and find designs that are better in terms of both throughput and sharpness. We show that our method constitutes a valid sampler and is effective at finding high quality lens designs.

Lastly, we address how to add more complicated types of lens elements to our framework. Gradient index lenses are hard to optimize with traditional techniques because of the large number of variables to optimize. We derive a method for calculating gradient through nonlinear ray tracing that has an order of magnitude lower memory requirements than existing methods. This enables our method to explore many different applications of gradient index optics, such as building mutliview displays and building more efficient fiber optics.

Acknowledgments

The work in this thesis would not have been possible without the support and mentorship from both my advisors, Ioannis Gkioulekas and Matthew O'Toole. I am indebted to their patience and guidance throughout this process through the highs and lows of my time at CMU. Both the personal and professional support from both of them has shaped my own identity and ability as a researcher. I would also like to thank my committee members for their valuable feedback as well, Bernd Bickel and Jim McCann, along with my collaborator Delio Vicini for his input for coming up with experiments to verify the work.

My time at CMU is full of cherished memories of lunch time at the graphics lab lounge, speaking with friends and colleagues about our research problems and discussing the various topics within computer graphics. Specifically, Mark, Jenny, Hossein, Bailey all contributed to creating a vibrant office space that made me eager to come to work every day. Along with my friends in the CSD program, Siva, Jalani, Kevin, Abhiram, David, and many others, thank you for the camaraderie and for sharing this journey through the PhD.

I wouldn't have even started on this journey without the encouragement and support from my family and friends. Chirag Sakhuja, thank you for encouraging me to apply to graduate school and for being a constant in my life during the whole of my PhD. To my parents, thank you for your unconditional love and support throughout my life and for giving me the opportunity to pursue higher education in the first place. Your enthusiasm for science is in large part why I chose this path. My siblings have also grounded me in life, helping me maintain perspective and balance throughout the process. And finally, to all my friends, (Cassidy, Prakash, David, Alan, Ashkon, Daniel, Hunju, to name a few) thank you for being my day ones.

Contents

1 Introduction **1**
- 1.1 Limitations of current lens design tools . . . 2
- 1.2 Research contributions . . . 3
- 1.3 Thesis organization . . . 4
- 1.4 Landscape of camera designs . . . 4

2 Modeling and Simulating Lenses **7**
- 2.1 Lens construction . . . 7
- 2.2 Design objectives . . . 8
- 2.3 Geometric optics . . . 9
- 2.4 Ray tracing . . . 11
- 2.5 Sequential Lens Design . . . 12

3 Aperture-Aware Lens Design **15**
- 3.1 Related work . . . 16
- 3.2 Adjoint ray tracing . . . 18
- 3.3 Aperture-Aware Gradients . . . 19
- 3.4 Warp-field reparameterization . . . 22
- 3.5 Experiments . . . 24
- 3.6 Discussion . . . 27

4 Mixed Discrete-Continuous Design of Compound Lenses **33**
- 4.1 Related work . . . 34
- 4.2 Lens design objectives . . . 35
- 4.3 Markov chain Monte Carlo for optimization . . . 36
- 4.4 The RESTORE algorithm . . . 38
  - 4.4.1 Algorithm overview . . . 39
- 4.5 Lens mutations . . . 41
- 4.6 Experiments . . . 44
- 4.7 Discussion . . . 48

5 Adjoint Nonlinear Ray Tracing **51**
5.1 Related Work . . . 52
5.2 Theoretical background . . . 53
5.2.1 Nonlinear ray tracing . . . 53
5.2.2 Adjoint state method . . . 55
5.3 Differentiating w.r.t. Refractive Index . . . 56
5.4 Discretization of the adjoint equations . . . 59
5.5 Experiments . . . 63
5.6 Discussion . . . 72

6 Conclusion **75**
6.1 Summary of Contributions . . . 75
6.2 Discussion . . . 76

Bibliography **79**

List of Figures

1.1 Lens examples . . . 4

2.1 The simplest lens model. . . . 7
2.2 Tracing is an alternating series of propagations and transmissions. . . . 11
2.3 Sequential vs non-sequential lens design. . . . 12

3.1 Aperture-aware lens design overview . . . 15
3.2 Ray diagram of valid rays . . . 16
3.3 Biased versus unbiased gradients . . . 22
3.4 Entrance pupil shape . . . 22
3.5 Gradient comparison of aperture-aware gradients . . . 25
3.6 Aperture-aware pareto front . . . 29
3.7 Low-light performance comparison . . . 30
3.8 Motion blur performance comparison . . . 30
3.9 Vignetting comparison . . . 30
3.10 Optimizing a zoom lens . . . 31

4.1 Validation with a simple scene . . . 41
4.2 Types of mutations . . . 42
4.3 Examplee of paraxial equivalence . . . 43
4.4 Comparison with Metropolis-Hastings . . . 45
4.5 Comparison with brute force search . . . 46
4.6 Paraxial projection ablation . . . 47
4.7 Pareto front expansion . . . 48

5.1 Examples of refractive index fields . . . 51
5.2 Diagram of adjoint tracing procedure . . . 57
5.3 Runtime and memory use comparison for reverse-mode AD and the adjoint method . . . 64
5.5 Effect of volume resolution on optimization . . . 66
5.4 Optimized GRIN lens displaying two different images . . . 66
5.6 A multifocal display . . . 67
5.7 Caustic display for near and far fields . . . 68
5.8 Luneburg and Maxwell lens reconstructions . . . 69

5.9 Optimizing a GRIN optical fiber 70
5.10 Reconstruction of gas flows at varying scales 73

List of Tables

4.1 Ablation study for paraxial projection 47

5.1 Definitions of main terms used in the adjoint state method. 54
5.2 Comparison of the adjoint method with prior work. 71

Chapter 1

Introduction

Cameras play an incredibly important role in the modern day world. They exist in applications ranging from consumer cameras and scientific instruments to theatrical lighting and projector systems, and with almost every practical camera system, there is a lens that focuses light from the target scene onto a sensor. The design of these lens systems requires careful construction, since these cameras are often used in applications where the quality of the image and the speed at which the camera can operate are critical to performance. For this reason, highly trained optical engineers are tasked with building lenses that are accurate at the micron level and below.

Contemporary lens design, therefore, presents a multifaceted optimization challenge. A typical compound lens system is characterized by both continuous parameters, such as surface shape and thicknesses, and discrete choices, such as the number of elements and types of material. This mixed discrete-continuous parameter space creates a complex design landscape where the performance of the lens is highly sensitive to all of the choices of parameters. This design space has been historically hard for designers to navigate, necessitating assistance from theoretical and computational tools that help guide the search for performant designs. Geometric optics theory provides a simplified model for how light travels through the refractive materials that make up lenses [10, 68], which remains the basis of how optical design software such as Zemax [109] and CODE V [91] evaluate lens performance by default. Yet, even with these tools, design remains time consuming and requires a great deal of designer input. It is thus useful, and the purpose of this thesis, to develop algorithms that alleviate the burden on designers.

In parallel, the graphics community has developed methods for differentiable rendering, which enable gradient-based optimization of image based tasks [44, 53, 99]. These methods have been successfully applied to a variety of problems and are a great candidate for application to lens design. However, there are key differences in lens design from general rendering that make directly applying these methods to lens design challenging. In particular, lenses are mostly comprised of refractive elements, which is expensive to evaluate in a general rendering context.

The purpose of this thesis is to develop methods that leverage ideas and techniques from differentiable rendering to address the challenges of lens design, which will require us to

develop a set of theoretical and computational tools tailored to the unique requirements of lens design. We describe three main contributions that address limitations in current lens design tools:

- A method for calculating the unbiased gradient of light throughput with respect to lens parameters, enabling the optimization of lens speed.
- A Markov chain Monte Carlo (MCMC) method that combines gradient-based optimization of continuous parameters with discrete mutations that change the number of elements in a lens system.
- A method for calculating gradients of ray paths through gradient-index (GRIN) materials using constant memory, allowing for optimization of GRIN lenses.

1.1 Limitations of current lens design tools

Lens design has a long history in optics research and industry, and thus there are several textbooks providing detailed theory and engineering background [68, 84]. Modern lens engineers use dedicated software tools for simulation and optimization, such as Zemax [109] and CODE V [91]. These tools enable engineers to iterate on virtual lens designs before building expensive physical prototypes.

Despite being the industry standard, these commercial lens design tools have several limitations that constrain the design process:

Limited Objective Function Scope Commercial software offers functionality for optimizing designs with metrics for sharpness. However, certain quantities critical to modern applications—particularly light throughput efficiency—cannot be directly optimized using current gradient-based methods as it is a quantity that is not readily differentiable with ray tracing. Designers must rely on heuristic methods to improve throughput, which generally involves manually adjusting the parameters of the lens to improve throughput.

Manual Design Existing software primarily handles continuous parameter optimization within a fixed lens design. Designers need to manually edit the design by adding or removing elements from the lens before then running another continuous parameter optimization. This process requires substantial expert knowledge and intuition about what discrete changes are reasonable and have potential to yield a better lens. Indeed, Chapter 2 of Smith [84] gives a long list of prescriptions for different problems that lens designers encounter during design. These instructions, however, are suggestions that are not guaranteed to yield a better design, and thus the designer must manually try different combinations of these mutatations to hopefully find a better design.

Gradient calculation In order to perform optimization on a design, commercial software employs gradient descent based optimization techniques, which rely on calculating a gradient of the objective function with respect to the design parameters. These software packages

typically either use finite-difference approximations to compute gradients or analytic gradients. Finite-difference approximations often produce biased and numerically unstable estimates of the gradient and the computational complexity of the approximation scales linearly with the complexity of the lens system. For example, gradient-index (GRIN) lenses, have many degrees of freedom which make them computationally intractable to optimize with finite-difference methods. Analytic gradients are perhaps the most computationally efficient way to calculate gradients, but analytic forms for ray tracing can generally only be found for simpler settings like a single spherical lens element. This severely limits the complexity of optical elements that can be optimized in the system.

1.2 Research contributions

This dissertation addresses these limitations through the development of novel theoretical frameworks and computational methods for lens design optimization. The primary contributions in order of presentation are:

Aperture-aware gradients A method for calculating the *unbiased* gradient of light throughput with respect to lens parameters, enabling the optimziation of lens speed. We show why prior work produces biased gradients when using traditional methods for calculating gradients, and we show how to overcome this bias with our technique. We also show that it is possible to find sharper lenses when using aperture-aware gradient as compared to using biased gradients. Being able to optimize both sharpness and throughput allows us to enumerate a Pareto front of speed-sharpness tradeoff.

Mixed continuous-discrete lens design A Markov chain Monte Carlo (MCMC) method that combines gradient-based optimization of continuous parameters with discrete mutations that change the number of elements. We also construct a method for adding and removing elements from a design that does not change the performance of the lens up to first order. This method gives us a way to incorporate mutations that lens designers typically use when exploring lens designs, which allows us to explore the design space more effectively. We demonstrate the efficacy of this system by producing lens designs that are comparable in performance to those produced by designers.

Adjoint nonlinear ray tracing A method for calculating gradients of ray paths through gradient-index (GRIN) materials using constant memory. In order to optimize for GRIN lenses, commercial products rely on expensive finite-difference methods for calculating gradients. Modern graphics methods require the use of automatic differentiation which is still an expensive operation. Our method presented in Chapter 5 allows for optimization of GRIN lenses which was previously intractable with finite-difference methods. With more computational efficiency, we can then optimize more complex designs that exhibit interesting effects that traditional refractive lenses cannot achieve.

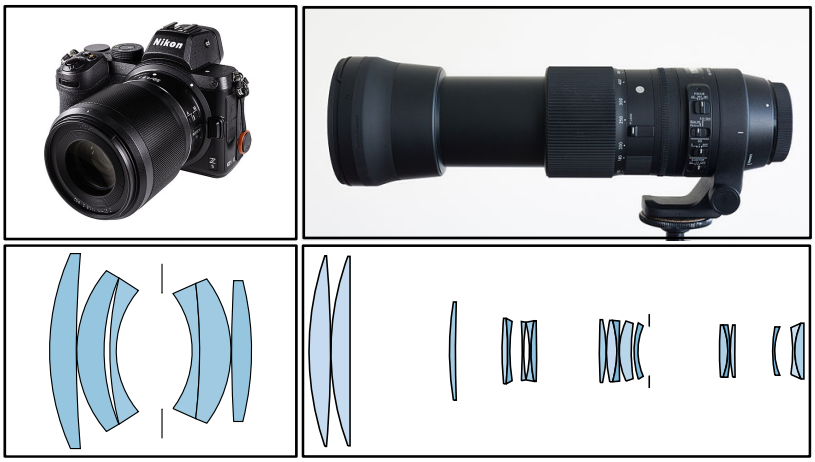


Figure 1.1: There are many different types of lens designs even when restricted to refractive sequential optics. Shown are two examples of lenses that we can address with the methods in this thesis. Images of the lenses are from commons.wikimedia.org.

1.3 Thesis organization

This thesis is organized first by describing in Chapter 2 how a lens is modeled and simulated with ray tracing. This will provide the theoretical background necessary to discuss the constributions of Chapter 3, which describes how address the optimization of light throughput. We will then discuss how to incorporate discrete variables into the optimization process in Chapter 4. After, Chapter 5 describes how to optimize gradient-index elements with gradient based optimization, which exemplifies ways in which the method from Chapter 4 can be extended to handle more complex types of optics. Finally, we conclude with a discussion of the limitations of our work and future research directions in Chapter 6.

1.4 Landscape of camera designs

Cameras come in many forms and are designed to image different phenomena of light. Most cameras today are constructed around a sensor that is usually built from a CMOS chip which is sensitive to light. The sensor is paired with an optical system that will focus light onto that sensor. There are many different types of optical systems that can be used to focus light onto a sensor. Optical elements are built from materials that either refract, reflect, and diffract light. Putting these elements together in a way that focuses light onto a sensor is what we refer to as an optical system.

When an optical element works by refracting or reflecting light, we categorize the elements as part of *geometric optics*. Geometric optics is a simplified model of light that represents light as rays that transport radiance as it travels through space and different materials. This model is useful for understanding how light behaves in optical systems, and it is the basis

for most lens design software, because of its relative computational simplicity. Geometric optics is not a complete model of light, but it is sufficient for many practical applications, such as for optical systems that are comprised of *geometric optical* elements. Examples of geometric optics are mirrors and dielectric materials, which are often used in telescopes and microscopes.

The most common optical system is a lens, which is a refractive optical element that bends light to focus it onto the sensor. These glass or plastic elements are usually stacked together to create a *compound lens* which often perform better than a single element because of the added degrees of freedom to remove aberrations and blur in the sensor image. Most cameras used in consumer and scientific applications are compound lenses. Compound lenses are the main focus of this thesis, and we will discuss them in more detail in Chapter 2.

If the optical element works by leveraging the wave nature of light, then we refer to them as wave optics. These are systems that manipulate light at the wavelength scale, such as diffraction gratings and holographic optical elements. These systems are often used in applications where the wavelength of light is important, such as in spectroscopy or interferometry. Wave optics is not the focus of this thesis, but it is an important area of optics that would also benefit from computational tools similar to those developed in this thesis.

Chapter 2

Modeling and Simulating Lenses

The main goal is of computational methods for designing lenses is to predict how a design will be before manufacturing a prototype. This is done by simulating how a lens will image an object given the design parameters. In order to simulate a lens design, we will need to have a geometric model for these lenses and a method for simulating how light propagates through the designed lens. This chapter provides an overview lenses are modeled and the theory for how to simulate light in the form of geometric optics.

2.1 Lens construction

We define a *lens* to be a device that will redirect light from a source point to a light sensor. It maps light emitting from points on a plane in space to points on a sensor, with the mapping being defined by the focal length of the lens. The plane that the object is on is referred to as the *object plane*, and the plane that the sensor is on is referred to as the *image plane*, or *sensor plane*. The focal length of the lens system dictates how much the object plane is

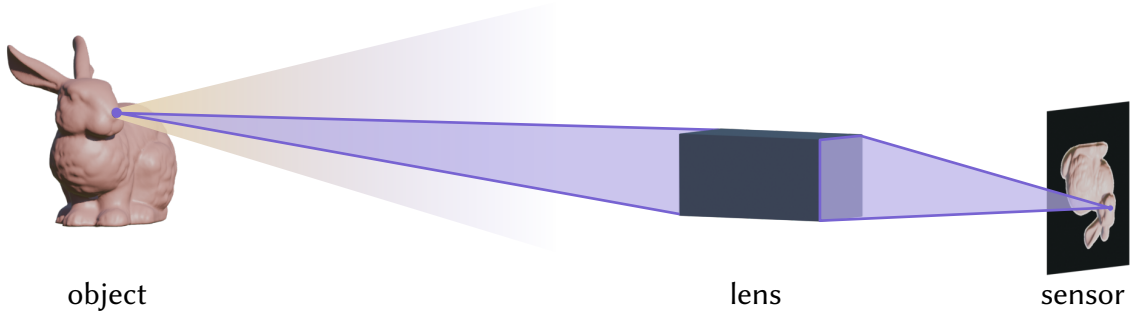


Figure 2.1: We consider a lens as a device that focuses light from a source point to a single point on a sensor. A lens is *sharp* if the light is directed to a very small area on the sensor. A lens is *fast* if the amount of light that is being focused on the sensor is large. The *focal length* of a lens dictates the size of the image on the sensor.

magnified on the sensor plane. This relationship is described by the *thin lens equation*,

$$\frac{1}{f} = \frac{1}{o} + \frac{1}{s}, \tag{2.1}$$

where f is the focal length of the lens, o is the distance from the lens to the object plane, and s is the distance from the lens to the sensor plane. After calculating the object and image distances, we can calculate the magnification, m, of the lens as,

$$m = \frac{s}{o}. \tag{2.2}$$

This tells us how far the point will be from the center of the sensor. The line that travels through the center of the sensor and is orthogonal to the object and sensor planes is referred to as the *optical axis* of the lens. The magnification of the lens is a scaling relative to this optical axis.

It is common practice when designing a lens to assume that the object is placed at infinity, since it possible to always move the image plane relative to the lens to refocus for objects at different distances from the lens. When the object plane is placed at infinity, then the image plane is placed exactly a focal length away from the lens, and each "point" on the the object plane is a bundle of parallel light rays, with the incident angle of the rays representing a different point on the object plane. This optical setup is referred to as *infinite conjugacy*, and is the default construction that most commerical lens design tools use, such as Zemax [109] and Code V [91]. In this thesis, we will also assume infinite conjugacy unless stated otherwise.

2.2 Design objectives

It is important to discuss how we can evaluate the quality of a particular lens. In this section, we describe the qualities of a "good" lens design, which we can formalize later in the thesis as optimization objectives.

Sharpness In practice, a lens is not capable of focusing light from a point source to an exact point on the sensor. Instead, the point source will map to an area on the sensor rather than a single point. The size of this area is referred to as the *spot error*, and the smaller the spot error, the sharper the image will be. No lens has perfect sharpness for all points on the object plane, and the sharpness often gets worse as the distance from the optical axis increases. Additionally, if the scale of the spot error is small enough relative to the wavelength of light, then the spot error will be dominated by effects not captured through ray tracing. At this point, the lens is said to be *diffraction limited* and is the best that can be achieved with ray tracing. Once a lens is diffraction limited, we consider the lens to be as sharp as possible.

Focal length A specification requirement for a lens design is the focal length, which describes how much magnification the lens will provide. A compound lens design will rarely have the *exact* focal length specified, but rather changes over the span of the sensor. This deviation from the ideal focal length is what gives rise to *distortion* in the image, which is a common lens aberration. We can measure the distortion by tracking how far the focus point is from the ideal focal length for different object heights. Depending on the application, distortion may not be a significant issue, but it is still important to consider when designing a lens.

Speed A lens design is fast if it allows a large amount of light to reach the sensor plane. This quality is often important for light sparse situations, such as night photography and sports photography. Essentially, speed is an area measure over how many rays reach the sensor plane from a point source in the scene. The unit of measurement for speed is the *f-number*, which is the ratio of the focal length of the lens to the diameter of the *entrance pupil*, which is the opening in the lens that light can travel through and successfully reach the sensor. A lower f-number indicates a faster lens design. The f-number of a lens also dictates the depth of field of the image which is the range of distances over which the image is in focus. Low f-numbers produce what is known as *bokeh* which is often a desirable quality in artistic photography. For this thesis, we work directly with throughput, rather than f-number.

Trade-offs If the lens has really large spot error, then the speed of the lens can be made arbitrarily large. Conversely, if the lens is very slow, then the spot error can very easily be made small, as the amount of light that needs to be focused is minimal. These edge cases exemplify the trade-off between the speed of the lens and the sharpness of the image. Indeed, the challenge of lens design is to find a lens that has both a good balance of speed and sharpness without making the lens too complicated to manufacture. Historically, photographic lenses have been designed to be sharp, and are improved by increasing the speed of the lens while minimally affecting the sharpness. This is often done by adding more elements to the lens design, which follows the trend of manufacturing generally becoming cheaper and more precise.

2.3 Geometric optics

There are many different ways to steer light. At human scale, light travels as a wave and can be manipulated by changing the refractive index of the material that the light is traveling through. There are optical devices that act on the wavefront of light, changing both the amplitude and phase of the wave. These devices fall under the category of *wave optics*, and are often used in applications like holography and interferometry. If the scale of the objects that we are studying is much larger than the wavelength of light, then we can more simply track the wavefront by tracking a *ray* that is orthogonal to the wavefront as it propagates through space. What results is a model in which light is treated as a collection of rays that travel through space and interact with objects by changing direction when encountering a

change in material. This model is known as *geometric optics*, which is the less accurate model of light propagation, but is computationally simpler.

Most commercial lenses are constructed from pieces of glass or plastic that are shaped to focus light. These optical elements are accurately modeled with geometric optics, in fact, the model error is usually negligible compared to the resolution of the sensors that are used within these imaging systems. For this reason, most commercial software for lens design, such as Zemax [109] and Synopsis [91], use the geometric model for designing lenses.

The methods of this thesis are built on geometric ray theory to both simulate the performance of a lens design but to also calculate the gradients of performance with respect to the design parameters. The following section provides a brief summary of the relevent theories used in this proposal. Further discussion of geometric ray theory and Hamiltonian optics can be found in Born and Wolf [10].

The primary object that we work with is the *light ray*, which is a description of how light will travel through objects in space. A light ray is defined by its position, x, and direction, v. The position of a light ray is the point in space where the light ray is currently located, and the direction of a light ray is the direction in which the light ray is traveling. In this thesis, we will use the notation $\omega = \{\mathrm{x}, \mathrm{v}\}$ to denote a light ray. In most cases, we use ω_0 to denote a ray that originates from a light source, and ω_{N+1} refers to a ray that has reached an end point, which is typically a sensor.

At a high level, we can think of a lens (or any geometric optical system) as a function, $f(\omega_0) = \omega_{N+1}$, that takes a light ray as input and traces the ray until it reaches a sensor.

In the context of design, we are interested in *optimizing* the parameters of a lens design, θ, to minimize a cost function, $\mathcal{C}$, that describes the performance of the lens design. This cost function is typically a function of how all of the light in a scene is focused on a sensor, and is of the form,

$$\mathcal{C}(\omega_0; \theta) = \mathcal{C}(f(\omega_0; \theta), \theta), \tag{2.3}$$

where ω_0 is a light ray that originates from some light source.

To optimize for the cost function, $\mathcal{C}$, we use gradient-based optimization, which requires computing the gradient of the cost function with respect to the lens parameters, θ,

$$\frac{\mathrm{d}\mathcal{C}}{\mathrm{d}\theta} = \frac{\partial \mathcal{C}}{\partial f}\frac{\partial f}{\partial \theta} + \frac{\partial \mathcal{C}}{\partial \theta}. \tag{2.4}$$

Calculating the partial derivatives of the cost function is often straightforward, as the cost function is typically simple, like the L2-norm. However, calculating the partial derivatives of the function f with respect to the lens parameters, θ, is more complex, since this function encapsulates the entire dynamics of light traveling through our lens. What follows is a deeper explanation of what this function, f, is and how it is implemented. Calculating this gradient is addressed in Chapter 3 of the thesis.

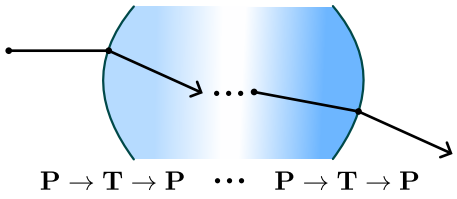


Figure 2.2: Tracing is an alternating series of propagations and transmissions.

2.4 Ray tracing

A light ray will travel in a straight line until it encounters a change in refractive index, at which point the ray will change direction according to Snell's law. Thus, f can be decomposed into repeated applications of two operations, *propagation* $\mathbf{P}$ and *transmission* $\mathbf{T}$, in alternating order at each surface. Given the current ray position and direction, $\mathbf{P}$ updates the ray position to the first intersection with the next lens surface; then $\mathbf{T}$ updates the ray direction at that intersection, using either Snell's law at refractive surfaces of singlets, or free-space transmission at open surfaces of aperture stops. Thus, starting at ω, ray tracing proceeds with the recurrence relation:

$$\mathrm{x}_{i+1} = \mathbf{P}\left(\mathrm{x}_i, \mathrm{v}_i, \theta\right), \tag{2.5}$$

$$\mathrm{v}_{i+1} = \mathbf{T}\left(\mathrm{x}_{i+1}, \mathrm{v}_i, \theta\right). \tag{2.6}$$

The subscript i indicates each step of the ray tracing process, as it iterates through the lens surface sequence. Ray tracing ends when the ray intersects the sensor at a position x_{N+1}, forming a ray $\mathrm{x}_0 \to \mathrm{x}_1 \to \ldots \mathrm{x}_N \to \mathrm{x}_{N+1}$, where N is the number of lens surfaces. Both $\mathbf{P}$ and $\mathbf{T}$ depend on the parameters θ that describe the lens geometry and material properties. Thus the sequence of positions x_i and directions v_i (including x_{N+1}) are functions of the lens parameters θ and the initial conditions ω—we make this dependence explicit or implied in equations depending on context. Importantly, for a fixed lens θ, the initial conditions ω completely determine, through Equations (2.5)–(2.6), a ray's position sequence x_i. Thus we can parameterize the space of rays through the lens using the initial conditions ω, and we will refer to each ω also as a ray.

Transmission Light rays passing through the interface of two different refractive materials will change direction according to Snell's Law,

$$\eta_1 \sin(\phi_1) = \eta_2 \sin(\phi_2),$$

where η_1 and η_2 are the refractive indices of the two materials and ϕ_1 and ϕ_2 are the incident and exiting angles, respectively. When rays travel at a glancing angle relative to the surface normal, then the ray of light will no longer transmit through the surface, but instead reflect. This is known as total internal reflection (TIR). For our purposes, when this happens, we simply terminate the tracing and ignore the ray. The angle at which TIR occurs is known as the *critical angle*, and is dependent on the refractive indices of the two materials that the refraction is ocurring.

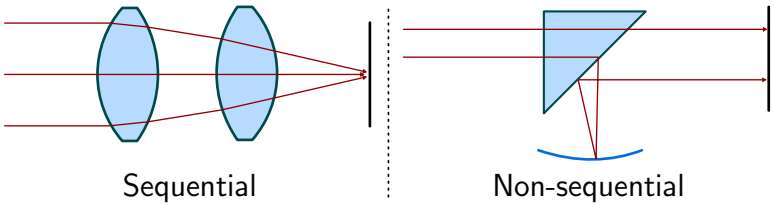


Figure 2.3: In a sequential lens design, all light rays pass through the lens surfaces in a fixed order. In a non-sequential lens design, different light rays may pass through a different ordering of surfaces, even interacting with the same surface multiple times.

Propagation When traveling in a single medium, a light ray will propagate in a straight line until it reaches *any* surface in the design. In the most general case, a spatial data structure is used to accelerate this search, such as a bounding volume hierarchy (BVH). This thesis focuses on sequential lens designs, where the order in which the ray intersects surfaces is known a priori. Thus, we can simply iterate through the surfaces in order. The theory presented here is readily applicable to the non-sequential case.

Supposing we know which surface the ray will intersect next, and that we can represent the surfaces implicitly (a function in which $g(\mathrm{x}) = 0$), we can compute i-th intersection point, x_i, by solving a constained optimization problem,

$$\mathrm{x}_i = \mathrm{x}_{i-1} + t^* \mathrm{v}_{i-1}, \tag{2.7}$$

$$t^* = \underset{t}{\operatorname{argmin}}\, t \quad \text{s.t.} \quad g(\mathrm{x}_{i-1} + t\mathrm{v}_{i-1}) = 0. \tag{2.8}$$

In many cases, this optimization problem can be solved analytically, such as when the surface is a plane or quadratic. When the surface is not formed analytically, we can use numerical optimization techniques, like Newton's method, to find the solution.

Whenever a light ray intersects a surface, most of the energy of the ray will be transmitted through teh surface, but some of the energy will be reflected back. The amount of energy that is reflected back is determined by the Fresnel equations,

2.5 Sequential Lens Design

The primary focus of this thesis is the design of *sequential* compound lenses, which are optical system where the order in which light passes through the components is fixed. This is in contrast to *non-sequential* optical systems, where the order of components can change based on the path of light through the system. Most lenses used in imaging systems (e.g. microscopes, telescopes, and cameras) are sequential lenses. In this chapter we will describe how to formulate the design of sequential lenses as an optimization problem and how the surfaces of these lenses are represented.

When a compound lens design is sequential, it will have the following properties and implications:

1. Rays entering the lens transmit through each refractive surface once and in order from lens entrance to sensor; we thus ignore geometric effects where rays violate this assumption (e.g., interreflections, glare).
2. Each compound lens is a collection of refractive singlets or cemented doublets (*lens elements*) that have spherical surfaces and are radially symmetric around the optical axis.
3. We assume geometric optics and thus ignore wave effects (e.g., diffraction).

Parameterization By virtue of the assumptions of sequential lens design, we can represent a compound lens as a sequence of refractive surfaces, each with a set of parameters that describe its geometry and optical properties. For each surface i in the lens, we need to following parameters to fully describe it:

- κ_i: the curvature of the surface
- d_i: the distance from this surface to the next surface in the lens.
- $\mathbf{R}_i$: the radius of the extent of the surface, orthogonal to the optical axis.
- η_i: the refractive index of the material after this surface.

This representation implicitly defines the sensor plane as the surface that the ray reaches after the *last* element defined in the representation. Similarly, we must define what the index of refraction is *before* the lens begins, which we will assume is air.

We can thus describe a compound lens as the ordered sequence of the surfaces of the lens elements—two for each singlet, one for each aperture stop. Additionally, we consider only geometric optics and focus on *sequential lens design*: that is, we consider only light rays that originate from an emitter surface before the lens, transmit once through all lens surfaces *in order*, and reach a sensor plane after the lens—we call such rays *valid rays* (Figure 3.2). Thus, we do not account for wave effects such as diffraction, or geometric effects involving invalid rays such as glare and interreflections.

With these parameters, we can then define an implicit function that describes the surface, i,

$$g(\mathrm{x}) = \kappa_i \|\mathrm{x}\| - 2\mathbf{P}_{\mathrm{oa}}\mathrm{x}, \tag{2.9}$$

where $\mathbf{P}_{\mathrm{oa}}$ is the projection onto the optical axis, and x is a point in 3D space. The surface is assumed to be centered at the origin, which we can always do by calculating the position of the lens based on the distances from the previous surfaces and offsetting the position of the ray when calculating an intersection.

Comparison to commerical products. Designing sequential lenses is perhaps the most common task in optical design. In commerical lens design software like Zemax [109], sequential lens tracing is the default mode of operation and is the main focus of the software. Similar to the work in this thesis, Zemax uses ray tracing to simulate the performance of a lens design and will also optimize the lens by calculating the gradient of lens performance metrics with respect to the lens parameters. Both Zemax and Synopsis [91] use the same parameterization as the one presented above. However, Zemax does not use automatic differentiation, and

instead uses a combination of analytic gradients and finite differences to approximate more complicated gradient terms. This is the common approach in most optical design softwares [91, 109], but it is not as efficient as using automatic differentiation, especially for complex lens designs. In this thesis, we will use automatic differentitation as well as derived analytic gradients that can be used in conjuction with automatic differentiation to calculate gradients in a much more memory and computationally efficient manner.

Chapter 3

Aperture-Aware Lens Design

A common design specification is requiring a lens that takes in as much light as possible, even at the cost of reduced sharpness. A light-efficient lens is especially important when imaging fast moving objects requiring a fast shutter (e.g., sports photography), or night scenes where available light is limited. However, existing design tools can only quantify the light efficiency of a lens (e.g., by computing its f-number), but cannot explicitly optimize it.

We augment the set of tools available for lens design by developing a technique to automatically optimize the light efficiency of a lens. Our key contribution is a new aperture-aware differentiable rendering technique for efficiently optimizing lens designs with respect to parameters controlling their entrance pupil. In particular, we express common lens design objectives as integrals over a domain that corresponds to the entrance pupil and depends on the lens parameters. We then derive estimators for the gradients of such objectives that correctly account for this dependence and are efficient to compute. We use our technique to explore the fundamental trade-off between light throughput and focus in lens design, and design lenses for applications where light throughput is critical, such as low-light and fast-motion settings.

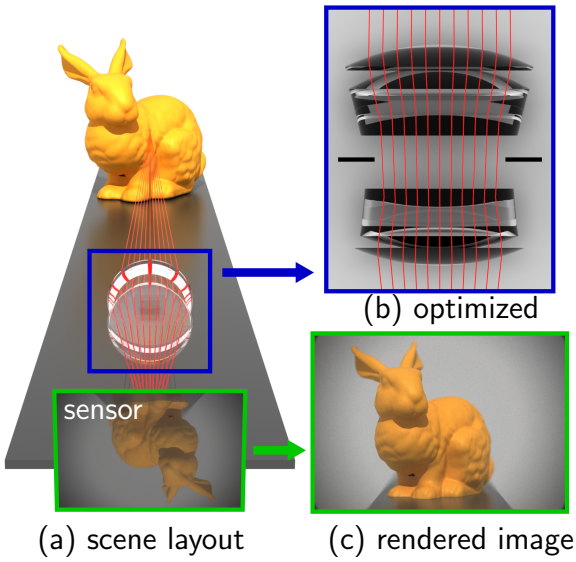


Figure 3.1: (a) The objective of this chapter is to design lenses capable of forming sharp images and having high light throughput. (b) We optimize the shape and position of individual lens elements and an aperture stop to gather the light from the scene. (c) Using off-the-shelf rendering software (e.g., Blender), we then model our lens and simulate its performance in forming images under various settings.

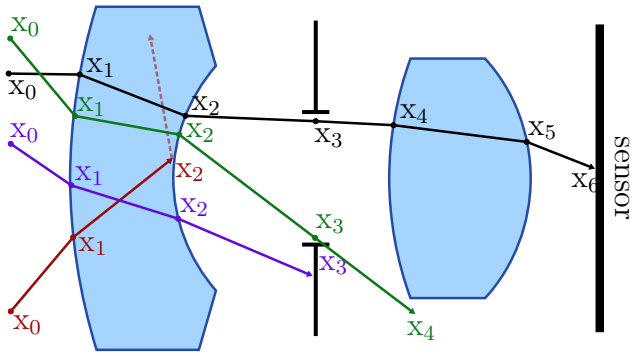


Figure 3.2: Rays travel through multiple refractive or aperture surfaces along the optical axis. A *valid* ray (black) travels through every surface once and in the correct order before reaching the sensor. An invalid ray either does not intersect a surface within its physical bounds (purple, green), or does so at a supercritical angle resulting in total internal reflection (red). We consider only valid rays in *sequential lens design*.

Our code for aperture-aware lens optimization is available at the project website.[1]

3.1 Related work

Existing software tools, both industrial and research [103], support searching for lens designs that match target specifications through gradient-based optimization. Our method expands these capabilities by enabling differentiation with respect to discontinuous parameters, such as the size of physical apertures and other pupils. This allows optimizing lens designs for important metrics, such as light throughput and vignetting.

Lens design Recent research in graphics and computational imaging has also looked at computational tools for lens design. Sun et al. [89] search through the lens design space using a stochastic mutation policy that adds and removes surfaces from the lens. By contrast, our method focuses on improving a fixed lens design without changing its number of surfaces. Damberg et al. [23] develop a method for generating caustic images using freeform lenses. This thesis is largely concerned with spherical sequential lens design, which are the most common type of lens used in consumer cameras.

Lens modeling Prior work in computer graphics has developed several approximate models for image formation through compound (multi-element) refractive lenses. Hullin et al. [38] derive a model for approximating ray tracing using polynomials. Tang and Kutulakos [92] provide a polynomial approximation of different optical aberrations in a lens. Both models trade off accuracy for efficiency, and thus are more suitable for inverse problems that require reasoning about the unknown scene that the lens is imaging. Tseng et al. [96] have used deep

[1] https://imaging.cs.cmu.edu/aperture_aware_lens_design/

learning to build differentiable approximations of the light transport inside a lens. These approximations, while efficient, do not provide sufficient accuracy for minimizing aberrations to the level required for photographic optics, and do not capture effects such as pupil size or light throughput. Our method uses geometric ray tracing, which is a more general and accurate model of light transport, and can thus optimize for such effects.

Automatic differentiation Most differentiable rendering architectures utilize, at different stages in their pipeline, automatic differentiation (AD) (*backpropagation*) [29]. AD analyzes the computational graph of a program, then iteratively applies the chain rule through the graph nodes to automatically calculate gradients of the program outputs with respect to its inputs. In computer graphics, Li et al. [54] build a domain-specific language for differentiable programming and use it to optimize a lens design in a setting similar to ours. We also use AD in both forward and reverse mode (backpropagation), to further enable the optimization of discontinuous lens parameters such as pupil sizes.

Refractive ray tracing Our method uses the classical geometric optics model of light transport through refractive materials, which has a long history as a modeling tool for refractive lenses [68]. In the context of rendering, this model translates to ray tracing with light paths that undergo a chain of specular (i.e., Dirac-delta) *refractive* events. Chen and Arvo [16] and Walter et al. [101] use the implicit function theorem to derive differentials for such reflective and refractive light paths with respect to their endpoints. Zeltner et al. [108] and Jakob and Marschner [43] use the same theory to efficiently sample and perturb specular chains for Monte Carlo rendering. Our method likewise computes differentials of specular paths, with respect to lens parameters.

Differentiable rendering In recent years, differentiable rendering has emerged as a core methodology for solving inverse problems within vision and graphics. A differentiable renderer calculates derivatives of images, or image-based objectives, with respect to scene parameters. Examples of differentiable rendering systems include Mitsuba 3 [45], Redner [53], and PSDR [110]. Recent works have enabled differentiable rendering with drastically lower memory requirements [99], efficient handling of parametric discontinuities [5], and support both explicit and implicit surface representations [6, 12, 100]. Our differentiable rendering method utilizes all three of these advances, and extends support to specular-manifold light paths that comprise solely interactions with smooth refractive interfaces and optical stops—such as the paths inside a lens.

Adjoint-state method The adjoint-state method is a classical methodology for efficiently differentiating optimization objectives constrained by partial differential equations (e.g., the rendering equation) [15, 34]. More recently, Chen et al. [18] used the adjoint-state method for differentiating ordinary differential equation systems represented as neural networks. In computer graphics, the adjoint-state method has found applications in problems including rigid-body dynamics and control [28], fluid control [59], and surface cutting [83]. In

differentiable rendering, Nimier-David et al. [63] and Stam [85] use the adjoint-state method to derive algorithms that decouple the computational complexity of ray tracing from the memory requirements of backpropagation. Vicini et al. [99] use a two-stage forward and backward tracing procedure to achieve constant memory complexity during derivative calculations. Similar to Wang et al. [103] and Teh et al. [93], we adopt an adjoint-state method for efficiently computing gradients of refractive ray tracing for lens optimization.

3.2 Adjoint ray tracing

We are interested in computing the gradient of the final position x_{N+1} with respect to the lens parameters θ. As the ray tracing process is a composition of alternating $\mathbf{P}$ and $\mathbf{T}$ operations, we can use the chain rule and compute this gradient by working backwards (i.e., *backpropagating*) through this composition. Differentiating Equations (2.5)–(2.6), we arrive at the backward recurrence:

$$\frac{\partial \mathrm{x}_{i+1}}{\partial \theta} = \frac{\partial \mathbf{P}}{\partial \mathrm{x}_i}\frac{\partial \mathrm{x}_i}{\partial \theta} + \frac{\partial \mathbf{P}}{\partial \mathrm{v}_i}\frac{\partial \mathrm{v}_i}{\partial \theta} + \frac{\partial \mathbf{P}}{\partial \theta}, \tag{3.1}$$

$$\frac{\partial \mathrm{v}_{i+1}}{\partial \theta} = \frac{\partial \mathbf{T}}{\partial \mathrm{x}_{i+1}}\frac{\partial \mathrm{x}_{i+1}}{\partial \theta} + \frac{\partial \mathbf{T}}{\partial \mathrm{v}_i}\frac{\partial \mathrm{v}_i}{\partial \theta} + \frac{\partial \mathbf{T}}{\partial \theta}. \tag{3.2}$$

At each iteration i, we can compute the gradients of $\mathbf{T}$ directly using automatic differentiation, and the gradients of $\mathbf{P}$ (an intersection operation) using implicit differentiation [61].

Computing the backward recurrence in Equations (3.1)–(3.2) requires knowing the ray path. As both $\mathbf{P}$ and $\mathbf{T}$ are bijective operations, we can recover this path by starting from the final position and direction and tracing backwards. Thus, we can calculate the gradients of the final position x_{N+1} with respect to θ without storing the ray path in memory using the following two-stage process.

Primal tracing: Starting from $\omega = \{\mathrm{x}_0, \mathrm{v}_0\}$, we use the recurrence of Equations (2.5)–(2.6) in the forward direction to compute the final ray position $\mathrm{x}_{N+1}(\omega, \theta)$ and direction $\mathrm{v}_{N+1}(\omega, \theta)$.

Adjoint tracing: Starting from $\{\mathrm{x}_{N+1}, \mathrm{v}_{N+1}\}$, we use jointly the recurrences of Equations (2.5)–(2.6) in the backward direction to retrace the ray, and Equations (3.1)–(3.2) to compute gradients.

Wang et al. [103] derive the same process using the adjoint state method. We then need to be able to compute the gradients of the $\mathbf{P}$ and $\mathbf{T}$ functions with respect to the lens parameters, θ.

Transmission Since refraction is a continuous process we can compute the gradient of the new direction with respect to the old direction through automatic differentiation.

Propagation To calculate the gradient terms of $\mathbf{P}$, we need to compute the derivatives with respect to the parameters θ, the position x, and the velocity v. Calculating the gradient

through automatic differentiation becomes expensive, as would require differentiating the entire iterative history of Newton's method. Instead, to calculate the gradient of interest,

$$\frac{\partial \mathbf{P}}{\partial \theta} = \frac{\partial \mathrm{x}_i}{\partial t^*}\frac{\partial t^*}{\partial \theta} \tag{3.3}$$

$$\frac{\partial \mathbf{P}}{\partial \mathrm{x}} = \frac{\partial \mathrm{x}_i}{\partial t^*}\frac{\partial t^*}{\partial \mathrm{x}}, \tag{3.4}$$

$$\frac{\partial \mathbf{P}}{\partial \mathrm{v}} = \frac{\partial \mathrm{x}_i}{\partial t^*}\frac{\partial t^*}{\partial \mathrm{v}}, \tag{3.5}$$

we need derive ${}^{\mathrm{d}t^*}/_{\mathrm{dx}}$, ${}^{\mathrm{d}t^*}/_{\mathrm{dv}}$ and ${}^{\mathrm{d}t^*}/_{\mathrm{d}\theta}$ directly.

Using the fact that the constraint needs to be satisfied at the solution, we can use implicit differentiation to compute the gradients, specifically, we differentiate the constraint with respect to x, v, and θ, noting that the solution t^* is implicitly a function of x, v, and θ, and solve for ${}^{\partial t^*}/_{\partial \mathrm{x}}$,

$$g(\mathrm{x}_{i-1} + t^*\mathrm{v}_{i-1}) = 0, \tag{3.6}$$

$$\frac{\partial g}{\partial \mathrm{x}}^\top \left(\mathbf{I} + \mathrm{v}_{i-1}\frac{\partial t^*}{\partial \mathrm{x}}^\top\right) = 0, \tag{3.7}$$

$$\frac{\partial t^*}{\partial \mathrm{x}} = \frac{-\frac{\partial g}{\partial \mathrm{x}}}{\left\langle \frac{\partial g}{\partial \mathrm{x}}, \mathrm{v}_{i-1}\right\rangle}. \tag{3.8}$$

We can follow a similar process to compute the gradients with respect to v and θ and obtain,

$$\frac{\partial t^*}{\partial \mathrm{v}} = \frac{-t^*\frac{\partial g}{\partial \mathrm{x}}}{\left\langle \frac{\partial g}{\partial \mathrm{x}}, \mathrm{v}_{i-1}\right\rangle}, \tag{3.9}$$

$$\frac{\partial t^*}{\partial \theta} = \frac{-1}{\left\langle \frac{\partial g}{\partial \mathrm{x}}, \mathrm{v}_{i-1}\right\rangle}\frac{\partial g}{\partial \theta}. \tag{3.10}$$

With these gradients, it is now possible to compute the gradients of the ray tracing process with respect to the initial ray position and direction as well as the scene parameters, θ. We note that we have still abstracted the calculation of ${}^{\partial g}/_{\partial \mathrm{x}}$, ${}^{\partial g}/_{\partial \mathrm{v}}$, and ${}^{\partial g}/_{\partial \theta}$. This is an implementation dependent quantity, and these terms can often be calculated with automatic differentiation when they cannot be calculated analytically.

3.3 Aperture-Aware Gradients

To design a compound lens, we minimize objectives that quantify its imaging performance, including objectives for blur and light efficiency. We express such objectives in the general form:

$$\mathcal{L}(\theta) \coloneqq \int_{\Omega(\theta)} \mathcal{C}(\mathrm{x}_{N+1}(\omega, \theta), \theta)\,\mathrm{d}\omega. \tag{3.11}$$

The scalar-valued function $\mathcal{C}$ describes a cost for each ray ω (e.g., squared distance of x_{N+1} from a target focus point). The integration is over the set of all valid rays, which we write as:[2]

$$\Omega(\theta) \coloneqq \left\{\omega \in \mathcal{E} \times \mathcal{S}^2 : g_i(\mathrm{x}_i(\omega,\theta), \mathrm{v}_i(\omega,\theta), \theta) < 0, i = 1, \ldots, N\right\}. \tag{3.12}$$

For each of the N lens surfaces, we use a scalar-valued implicit function g_i to describe the constraint that the ray must transmit through that surface—this *constraint function* is negative when that happens, and positive otherwise. As we explain next, in practice we need to specify more than one constraint function per lens surface, and Ω is the set of rays that satisfy all constraint functions for each surface. We do not explicitly indicate the multiple g_i for each i in Equation (3.12) to keep notation simple. We call $\Omega(\theta)$ the *entrance pupil* set, as it comprises all rays forming the entrance pupil image.

Constraint functions To understand how to define the constraint functions, we must consider the different ways a ray may fail to transmit through a lens surface, which we show in Figure 3.2. As we explain later in this section, we require that these constraint functions be differentiable with respect to all three of their arguments.

We consider first a refractive surface. A ray can fail to transmit through it due to: 1. straying too far from the optical axis; or 2. experiencing total internal reflection. In the former case, the ray position on the surface exceeds the surface's physical bounds. Given radially-symmetric surfaces, we can define a constraint function:

$$g_i^{\text{semi}}(\mathrm{x}, \mathrm{v}, \theta) \coloneqq \mathrm{dist}(\mathrm{x}) - \mathbf{R}_i(\theta), \tag{3.13}$$

where $\mathbf{R}_i$ is the lateral radius of the i-th surface (i.e., the size of the singlet it belongs to, determined by the lens parameters θ), and $\mathrm{dist}(\mathrm{x})$ is the lateral distance of a position x from the optical axis.

In the latter case, the ray intersects the surface at a supercritical angle. We can define a constraint function for this case as:

$$g_i^{\text{TIR}}(\mathrm{x}_i, \mathrm{v}_i, \theta) \coloneqq \left(\cos(\phi_i^{\text{crit}}(\theta))\right)^2 - \langle \mathrm{v}_i, \hat{\mathbf{n}}(\mathrm{x}_i, \theta)\rangle^2, \tag{3.14}$$

where ϕ_i^{crit} is the critical angle for the i-th surface (determined by the lens material parameters in θ). We use the differences of squared cosines as, empirically, doing so improves numerical stability.

Lastly, for an aperture stop surface, we can use the same constraint functions g_i^{semi}, g_i^{TIR}, in the former with $\mathbf{R}_i$ equal to the radius of the aperture stop opening, and in the the latter with $\phi_i^{\text{crit}} = \pi/2$ (i.e., the constraint is always negative and thus satisfied).

Differentiating lens design objectives Gradient-based optimization of a compound lens requires differentiating objectives as in Equation (3.11) with respect to the lens parameters θ.

[2]The measure $\mathrm{d}\omega$ is the product of the area $\mathrm{d}A(\mathrm{x})$ and solid angle $\mathrm{d}\sigma(\mathrm{v})$ measures.

If the domain of integration Ω were independent on θ, this gradient would equal:[3]

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta} \overset{\text{biased}}{=} \int_\Omega \frac{\mathrm{d}\mathcal{C}}{\mathrm{d}\theta}\,\mathrm{d}\omega = \int_\Omega \frac{\partial \mathcal{C}}{\partial \mathrm{x}_{N+1}} \frac{\partial \mathrm{x}_{N+1}}{\partial \theta} + \frac{\partial \mathcal{C}}{\partial \theta}\,\mathrm{d}\omega. \tag{3.15}$$

We could then use automatic differentiation to compute the gradients of the cost function $\mathcal{C}$, and differentiable ray tracing (Section 2.3) to compute the gradients of the final ray position x_{N+1}. Techniques such as DiffOptics [103] use this approach, which we refer to as the biased gradient approach.

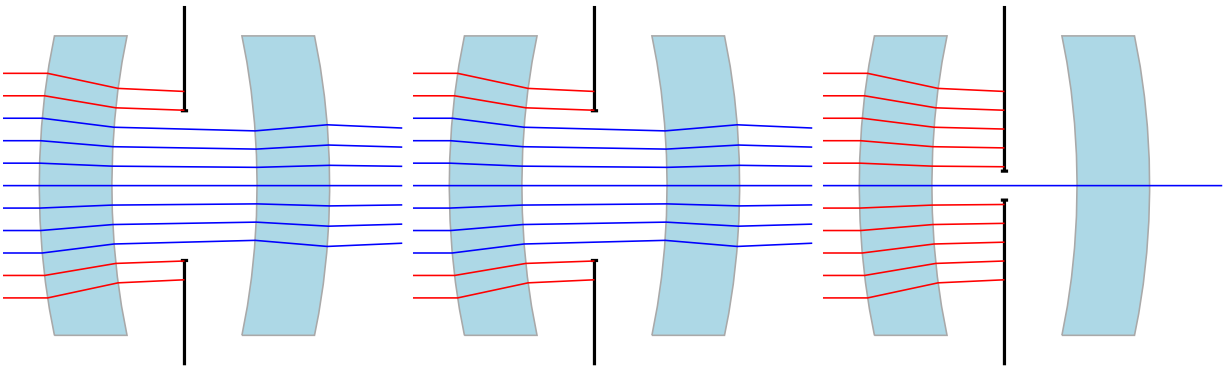
However, in general Ω *does* depend on the lens parameters θ. For example, the sizes of singlets and aperture stops in a compound lens determine its entrance pupil, and thus Ω. Then, we can use the Reynolds transport theorem [74] to write:

$$\begin{aligned}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta} &= \int_\Omega \frac{\mathrm{d}\mathcal{C}}{\mathrm{d}\theta}\,\mathrm{d}\omega - \int_{\partial\Omega(\theta)} \mathcal{C}\frac{\mathrm{d}g^*}{\mathrm{d}\theta}\,\mathrm{d}l(\omega) && (3.16)\\ &= \int_{\Omega(\theta)} \frac{\partial \mathcal{C}}{\partial \mathrm{x}_{N+1}} \frac{\partial \mathrm{x}_{N+1}}{\partial \theta} + \frac{\partial \mathcal{C}}{\partial \theta}\,\mathrm{d}\omega \\ &\quad - \int_{\partial\Omega(\theta)} \mathcal{C}\left(\frac{\partial g^*}{\partial \mathrm{x}^*}\frac{\partial \mathrm{x}^*}{\partial \theta} + \frac{\partial g^*}{\partial \mathrm{v}^*}\frac{\partial \mathrm{v}^*}{\partial \theta} + \frac{\partial g^*}{\partial \theta}\right)\mathrm{d}l(\omega). && (3.17)\end{aligned}$$

In Equation (3.16), the *entrance pupil boundary* $\partial\Omega(\theta)$ is the space of rays that satisfy all constraint functions g_i, *except* for one *active constraint* g^* that is equal to zero. The active constraint corresponds to a lens surface where the ray barely fails to transmit—i.e., the corresponding intersection point x^* is exactly on the surface boundary, or direction v^* is incident at exactly the critical angle.

Equation (3.16) shows that correctly differentiating objectives for compound lens optimization requires computing the *boundary integral* over $\partial\Omega(\theta)$. We show how to efficiently do so in Section 3.4. But first, we use an example to demonstrate the importance of using correct gradients for compound lens optimization. We consider a compound lens comprising an aperture stop between two singlets (Figure 3.3(a)). We optimize only the aperture radius to minimize imaging blur, using a cost function $\mathcal{C}(\mathrm{x}_{N+1}(\omega,\theta),\theta) \coloneqq \|\mathrm{x}_{N+1}(\omega,\theta) - \mathbf{o}\|^2$, where $\mathbf{o}$ is the center of the sensor plane. From geometric optics, the optimal solution involves reducing the aperture radius until we arrive at a pinhole camera, which produces a perfectly sharp image. As x_{N+1} does not depend on the aperture radius, the biased gradient of Equation (3.15) is always zero, and thus optimization using these gradients leaves the initial lens unaltered (Figure 3.3(b)). By contrast, optimization using the unbiased gradient of Equation (3.16) correctly results in a pinhole (Figure 3.3(c)).

[3]Throughout the thesis, we explicitly distinguish between *total derivatives* d/d and *partial derivatives* ∂/∂, e.g., in Equations (3.15)–(3.17) and Section 3.4.



(a) initial design (b) biased gradients (c) unbiased gradients

Figure 3.3: (a) We optimize the size of an aperture stop (in black) to minimize blur. (b) Doing so using biased gradients leaves the initial lens design unaltered. (c) Using unbiased gradients results in a pinhole, and thus perfectly sharp imaging, as expected from geometric optics.

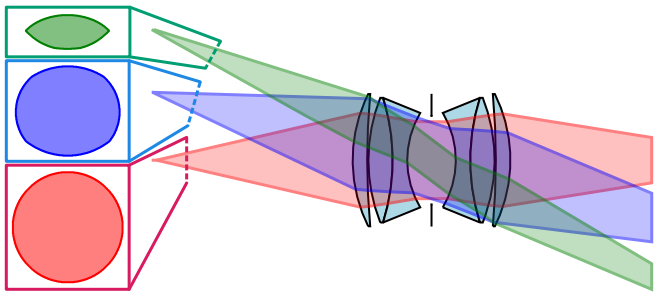


Figure 3.4: The shape of the entrance pupil $\Omega(\theta)$ is non-trivial even for radially symmetric lenses, becoming distorted for source points far from the optical axis. This behavior makes computing the entrance pupil boundary $\partial\Omega(\theta)$ challenging.

3.4 Warp-field reparameterization

Computing the boundary integral in Equation (3.16) is challenging because, for arbitrary compound lenses, the entrance pupil boundary $\partial\Omega(\theta)$ is difficult to characterize analytically or sample rays from: $\partial\Omega(\theta)$ch:tracing replace the boundary integral with an integral over the entrance pupil $\Omega(\theta)$.

To this end, we first rewrite the boundary integral in Equation (3.16) into a form amenable to the divergence theorem as:

$$\int_{\partial\Omega(\theta)} \mathcal{C}\frac{\mathrm{d}g^*}{\mathrm{d}\theta}\,\mathrm{d}l(\omega) = \int_{\partial\Omega(\theta)} \left\langle \mathcal{C}\frac{\mathrm{d}g^*}{\mathrm{d}\omega}, \mathcal{V} \right\rangle \mathrm{d}l(\omega), \tag{3.18}$$

where the *warp field* $\mathcal{V}(\omega, \theta)$ is a vector field on Ω that must be differentiable and satisfy, as $\omega \to \partial\Omega(\theta)$,

$$\left\langle \frac{\mathrm{d}g^*}{\mathrm{d}\omega}, \mathcal{V} \right\rangle \to \frac{\mathrm{d}g^*}{\mathrm{d}\theta}. \tag{3.19}$$

Then, using the divergence theorem, Equation (3.18) becomes:

$$\int_{\partial\Omega(\theta)} \left\langle \mathcal{C}\frac{\mathrm{d}g^*}{\mathrm{d}\omega}, \mathcal{V} \right\rangle \mathrm{d}l(\omega) = \int_{\Omega(\theta)} \operatorname{div}\left(\mathcal{C}\mathcal{V}\right)\, \mathrm{d}\omega, \tag{3.20}$$

where $\operatorname{div}(\cdot)$ is the divergence operator. By using Equation (3.20) to replace the boundary integral in Equation (3.16) and combining the integrands, we can write the objective gradient as:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta} = \int_{\Omega(\theta)} \frac{\mathrm{d}\mathcal{C}}{\mathrm{d}\theta} - \operatorname{div}\left(\mathcal{C}\mathcal{V}\right)\, \mathrm{d}\omega. \tag{3.21}$$

We have thus eliminated the boundary integral, making it possible to compute unbiased Monte Carlo estimates of the objective gradient by sampling rays in the entrance pupil $\Omega(\theta)$. For each ray sample, we can evaluate the derivative terms in the integrand using differentiable ray tracing (for $\mathrm{d}\mathcal{C}/\mathrm{d}\theta$, as in Section 3.2) and forward-mode automatic differentiation (for the warp field).

We still need to find a warp field that satisfies the stated requirements. Bangaru et al. [6] and Vicini et al. [100] suggest:

$$\mathcal{V}(\omega,\theta) \overset{?}{:=} \frac{\frac{\partial g^*}{\partial \omega}}{\left\| \frac{\partial g^*}{\partial \omega} \right\|^2} \frac{\partial g^*}{\partial \theta}. \tag{3.22}$$

However, g^* is undefined inside $\Omega(\theta)$, where no constraint is active. Instead, we leverage the fact that $\mathcal{V}(\omega,\theta)$ only needs to have the correct direction as it approaches $\partial\Omega(\theta)$, and define $\mathcal{V}(\omega,\theta)$ using a mixture of such warp fields for all constraint functions:

$$\mathcal{V}(\omega,\theta) := \frac{1}{\sum_i w_i} \sum_i w_i \frac{\frac{\partial g_i}{\partial \omega}}{\left\| \frac{\partial g_i}{\partial \omega} \right\|^2} \frac{\partial g_i}{\partial \theta},\ w_i(\omega,\theta) := \frac{\tanh(-\alpha g_i)}{{g_i}^p}, \tag{3.23}$$

with hyperparameters $p > 2$ and $\alpha > 0$. Abusing notation, the summation is over both g^{TIR} and g^{semi} constraint functions for each lens surface i. Each weight w_i approaches infinity as the corresponding constraint function g_i gets closer to activation; conversely, it decreases as g_i decreases. Intuitively, when a ray is close to failing to transmit through a lens surface—through either exceeding its bounds or total internal reflection—the corresponding constraint function should contribute more to the warp field.

Comparison to prior work Vicini et al. [100] and Bangaru et al. [6] have used warp-field reparameterization for differentiable rendering of implicit surfaces, to deal with visibility-driven changes in the integration domain of the rendering equation. Their techniques require using a separate warp field of the form of Equation (3.22)—replacing our active constraint function with the implicit function defining the scene geometry—at each *stochastic* reflection or refraction event along a multi-bounce light path. By contrast, we define a single warp field for the entire multi-bounce light path. We can do so because we deal with only *specular* (i.e., Dirac-delta and therefore *deterministic*) refractions, and can thus parameterize the resulting

space of specular multi-bounce paths using the initial conditions ω (Section 3.2). In turn, we use the same parameterization to define a multi-bounce warp field as in Equation (3.23). Therefore, our technique enables differentiable rendering in lower-dimensional specular path manifolds [43, 108] in the presence of parameter-dependent integration domains, which is not possible in existing differentiable rendering implementations [45]. Even though we present our theory and technique in the context of optimizing compound refractive lenses, they should apply more broadly for other inverse problems involving specular—reflective or refractive—path manifolds, e.g., inverse rendering of specular geometry [55] or caustics [65, 79].

3.5 Experiments

Implementation details We implemented our warp-field technique (including the differentiable ray tracer) in JAX [11], and provide our implementation on the project website. We used this implementation for all our optimization experiments, for which we set $\alpha = 10$ and $p = 6$ in Equation (3.23), and ran Adam [50] for 10,000 gradient iterations. We ran all experiments on a desktop with an Intel i7-6700k CPU and NVIDIA Titan X GPU, where each experiment took approximately 10 min.

To render images with various lenses, we used either our ray tracer, or Blender [20] with the built-in Cycles renderer. In the latter case, we modelled lens geometry as a triangle mesh.

Lens modeling and initialization We parameterized all singlet surfaces as spherical surfaces centered on a fixed optical axis, each with four parameters: curvature, position on the optical axis, refractive index, and semidiameter. We parameterized aperture stops as planar surfaces with two parameters: position and semidiameter.

Except where we state otherwise, we initialized optimization from a double-Gauss design by Reiley [73], which is a reconstruction of a design by Rudolph [76]. This design has 5 singlets (10 refractive surfaces) and one aperture stop, resulting in $3 \times 10 + 2 = 32$ free parameters—we do not optimize refractive index.

Finite differencing To verify the correctness of the gradients we compute with our warp-field method, in Figure 3.5 we show the gradient images (also known as *forward gradients* Zhang et al. [111]) it produces when differentiating with respect to the curvature of the first refractive surface and the semidiameter of the aperture stop of the lens. We compare these images against those from finite differencing and from the method of Wang et al. [103], which computes biased gradients as in Equation (3.15). Gradients from our method match finite differencing and accurately account for changes in both curvature and aperture size; by contrast, the method of Wang et al. [103] cannot account for aperture size changes, as those impact the integration domain $\Omega(\theta)$ rather than per-ray quantities (final ray position or throughput). Of course, even though finite differencing is useful for validating correctness, it is impractical for gradient-based optimization due to its linear complexity with parameter dimensionality, and need for step-size fine-tuning.

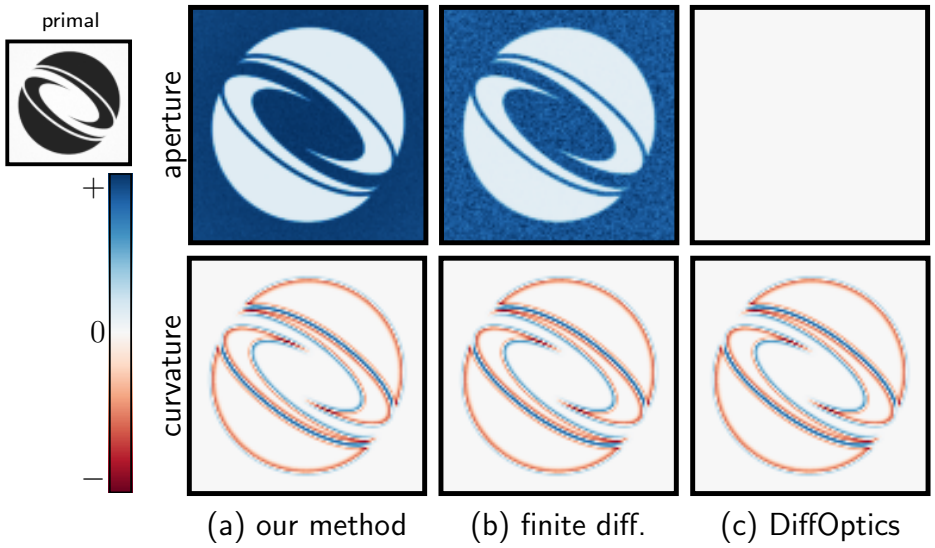


Figure 3.5: We consider an image of the SIGGRAPH logo through a lens, and differentiate it with respect to two lens parameters: (top row) aperture size, and (bottom row) curvature of the first refractive element. We compare the gradient images from: (a) our method, (b) finite differencing, and (c) DiffOptics [103]. Our method produces accurate gradients with respect to both lens parameters, and its results match those from finite differencing. By contrast, the biased gradients from DiffOptics cannot account for aperture size changes, producing a gradient image that is identically zero.

Light efficiency-sharpness trade-off The example of Figure 3.3 highlights a classical trade-off in lens design: In the absence of diffraction effects, we can increase image sharpness by shrinking the entrance pupil of the lens (e.g., making the aperture stop smaller, or refractive surfaces flatter); however, doing so comes at the cost of decreased light efficiency, as most light rays become invalid. Conversely, increasing the entrance pupil size improves light efficiency; but it also increases geometric aberrations, resulting in decreased sharpness.

We can use our technique navigate the lens design space and achieve different trade-offs between light efficiency and sharpness. To this end, we first define two objectives of the form of Equation (3.11):

$$\mathcal{L}_{\text{spot}}(\theta) \coloneqq \int_{\Omega(\theta)} \left\| \mathrm{x}_{N+1}(\omega, \theta) - \hat{\mathrm{x}}(\omega, \theta) \right\|^2 \mathrm{d}\omega, \tag{3.24}$$

$$\mathcal{L}_{\text{throughput}}(\theta) \coloneqq \int_{\Omega(\theta)} -1 \, \mathrm{d}\omega, \tag{3.25}$$

where $\hat{\mathrm{x}}(\omega, \theta)$ is the centroid of the sensor locations of all rays starting from the same source point $\mathrm{x}_0 \in \omega$. The *spot-size objective* $\mathcal{L}_{\text{spot}}$ measures the spot size produced on the sensor by rays starting from the same source point. Thus, this objective, common in lens design, encourages improved sharpness. The *throughput objective* $\mathcal{L}_{\text{throughput}}$ measures the size of the entrance pupil, with a negative sign to penalize rays becoming invalid. Thus, this objective encourages improved light efficiency.

We then optimize a compound lens using the *composite objective*:

$$\mathcal{L}(\theta) \coloneqq \mathcal{L}_{\text{spot}}(\theta) + \beta_{\text{NA}}\mathcal{L}_{\text{throughput}}(\theta). \tag{3.26}$$

The scalar weight β_{NA} allows us to control the relative importance of the spot-size and throughput objectives, and thus control how much we emphasize sharpness versus light efficiency. Gradient-based optimization of this composite objective critically depends on the ability to compute the unbiased gradient (3.16): the biased gradient of $\mathcal{L}_{\text{spot-size}}$ is inaccurate and, worse yet, the biased gradient (3.15) of $\mathcal{L}_{\text{throughput}}$ is *always zero*! This exploration of the light efficiency-sharpness tradeoff is only possible with our warp-field technique, and not those that use biased gradients [103].

Figure 3.6 shows an example where we optimize a 50 mm double-Gauss lens [22] using different weights β_{NA}. Our technique produces designs that represent different trade-offs between sharpness and light efficiency. We compare our designs against those from DiffOptics [103] and Zemax [109]: By using unbiased gradients to optimize both $\mathcal{L}_{\text{spot}}(\theta)$ and $\mathcal{L}_{\text{throughput}}(\theta)$, our warp-field technique produces lens designs that have higher throughput at comparable spot sizes, as well as designs with smaller spot sizes by giving up some throughput.

Figure 3.6 also shows rendered images and spot diagrams for some of the resulting lens designs.[4] We simulate the spot diagrams by focusing the lens at infinity and tracing rays parallel to the optical axis. We color the final ray points on the sensor based on how far the corresponding rays are from the optical axis. The spot diagram for the lowest β_{NA} value is much smaller than that for the highest value, indicating sharper focus; however, the resulting image is also a lot darker, indicating lower light efficiency. The figure also shows, for comparison, rendered images and spot diagrams for the designs from DiffOptics [103] and Zemax [109].

Low-light conditions Using a lens with high light efficiency is particularly important in low-light settings. In Figure 3.7, we simulate such a setting by constructing a candlelit scene. We render images of this scene using two lens designs from among those in Figure 3.6, one with low and another with high throughput, and incorporate Poisson and additive Gaussian noise [33] in the rendered images, assuming the same exposure time. The high-throughput lens results in a visibly brighter image—a difference also visible in the image intensity histograms—with higher signal-to-noise (SNR) ratio, at the expense of some sharpness.

Motion blur A situation where lens light efficiency is critical is in scenes with fast motion, where a high-throughput lens helps reduce exposure time, and thus motion blur. In Figure 3.8, we render images of a scene with a USAF resolution target moving from left to right, using the same two lenses as in Figure 3.7. In this experiment, we adjust exposure times so that

[4]The rendered images are slightly mismatched, because the optimization does not explicitly enforce a focal length constraint, and thus the resulting lenses produce images with different fields of view. In our experiments, the focal length of the optimized lenses deviated by around 1 mm from that of the initial design (50 mm).

both images reach the same brightness level. The low-throughput lens requires about four times longer exposure time, resulting in significant motion blur.

Vignetting The composite objective of Equation (3.26) penalizes low throughput, and thus encourages rays starting from source points away from the optical axis to reach the sensor. As a result, using this objective, and correctly minimizing it with the unbiased gradients from our warp-field technique, helps reduce vignetting artifacts in lens designs. Figure 3.9 shows images of a white balance target rendered using two lenses, optimized with our warp-field technique and Zemax. As Zemax uses biased gradients, it cannot optimize for the throughput of off-axis source points, and thus its designed lens results in significant vignetting. By contrast, the lens from our technique maintains high throughput for off-axis source points, drastically reducing vignetting.

Zoom lens In Figure 3.10, we use our method to improve a compound zoom lens for better sharpness and light efficiency. We start with a zoom lens design by Reiley [73], and optimize the surface curvatures and distances, by summing the composite objectives of Equation (3.26) for three different focal lengths—28 mm, 45 mm, and 75 mm. Our technique produces a zoom lens design that is generally better in terms of both overall sharpness and light efficiency. At focal lengths where both designs have similar sharpness, their performance can differ significantly across the field of view; for example, at 77 mm, the optimized lens has better sharpness in the center and worse towards the edges of the image.

3.6 Discussion

We discuss the limitations and applications of this technique outside the scope of this chapter, and how one may address them.

Interreflections An important effect that we did not explore is interreflections between and inside optical elements in a compound lens. In very bright scenes, such interreflections can become visible in the form of glare and lens flare. Our theory can, in principle, model this effect; however optimization of glare characteristics would require expensive Monte Carlo rendering, to account for all possible interreflections inside a lens in an unbiased manner. Simulating and optimizing such effects in a computationally efficient manner remains an open challenge.

Other geometric optical elements We have not considered many geometric optical elements that are commonplace in consumer photography and scientific imaging. Aspherical, Fresnel, and freeform lenses have many more degrees of freedom than spherical ones, and are common in projector and illumination systems. Beyond dioptric (i.e., refractive) systems, catoptric (i.e., reflective) systems are common in telescopes [27, 105]; and catadioptric (i.e., both reflective and refractive) systems [3] facilitate wide-angle imaging. As such systems can

be modeled using geometric optics and ray tracing, our theory directly supports or can be readily extended to support their design.

Beyond cameras We presented our technique in the context of photographic lenses, but it can apply to other domains. An example is augmented reality (AR) and virtual reality (VR), where design of eyepieces is important. VR displays use intricate configurations of optical elements [71] and benefit from high throughput designs [23]; our technique can help optimize existing such configurations or discover new ones. Another example is non-imaging optics, i.e., optical systems designed specifically to maximize light throughput [88]. Our technique can be suitable for optimizing designs in this domain.

Nonconvexity Our technique facilitates gradient-based optimization of complex objectives for compound lenses. However, these objectives are highly non-convex. Thus gradient-based optimization can get stuck in local minima. We can mitigate this issue by combining our technique with simulated annealing, or reasonable initializations from data-driven techniques [21].

Other design constraints and trade-offs Lens designs often need to accommodate application-specific constraints that go beyond sharpness and light efficiency. For example, the manufacturing process may constrain the realizable surface shapes. Alternatively, the intended use may constrain the lens form factor or cost. Such constraints are typically differentiable, and thus amenable to optimization using our technique. Even if they do not take the form of hard constraints, such application-specific considerations can introduce additional trade-offs that the lens design process must balance. In such cases, we can combine our technique with gradient-based multi-objective optimization techniques [78] to discover Pareto-optimal lens designs.

Wave optical elements Since our method relies directly on geometric optics, we cannot readily use these methods on optical elements that fundamentally rely on wave optics, such as diffractive optical elements and "metalenses" [13]. Such elements are becoming increasingly popular in both scientific and consumer applications, thanks to their miniaturization potential. Recent progress in wave optics rendering [87] can facilitate extensions in this direction.

Variable element number To produce performant compound lens designs, designers typically add and remove lens elements to examine whether different configurations can improve performance. From an optimization perspective, doing so corresponds to a discrete, non-differentiable operation that changes the dimensionality of the design space. Gradient-based optimization, and thus the technique presented in the chapter, does not accommodate such changes. We address this limitation in the next chapter, where we present a technique that introduces discrete operations into the optimization process.

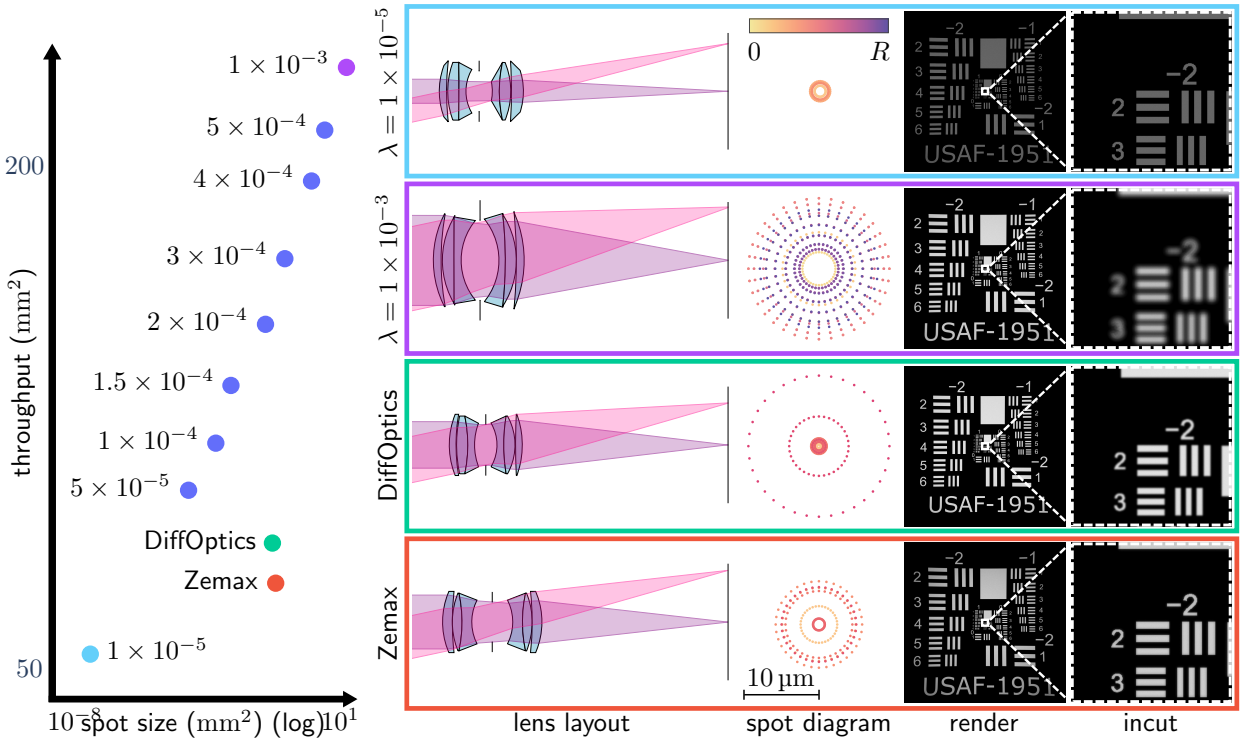


Figure 3.6: We use our warp-field technique to optimize the same initial design using the composite objective of Equation (3.26) with varying weights β_{NA}. The left plot quantifies the throughput and spot-size performance of the resulting set of lens designs, as well as designs optimized with DiffOptics [103] and Zemax [109]. By using unbiased gradients, our technique produces lens designs with better throughput at comparable spot sizes as DiffOptics and Zemax, which both use biased gradients. Our technique additionally produces multiple other designs that achieve different trade-offs between throughput and spot size. The right figures show the lens configurations, spot diagrams, and simulated images of USAF resolution targets for the designs corresponding to the lowest (first row) and highest (second row) β_{NA} values, DiffOptics (third row) and Zemax (fourth row). These figures help qualitatively assess the differences in spot size and throughput performance. For example, the spot-size-focused design in the first row has a smaller spot, but also transmits fewer rays than the throughput-focused design in the second row.

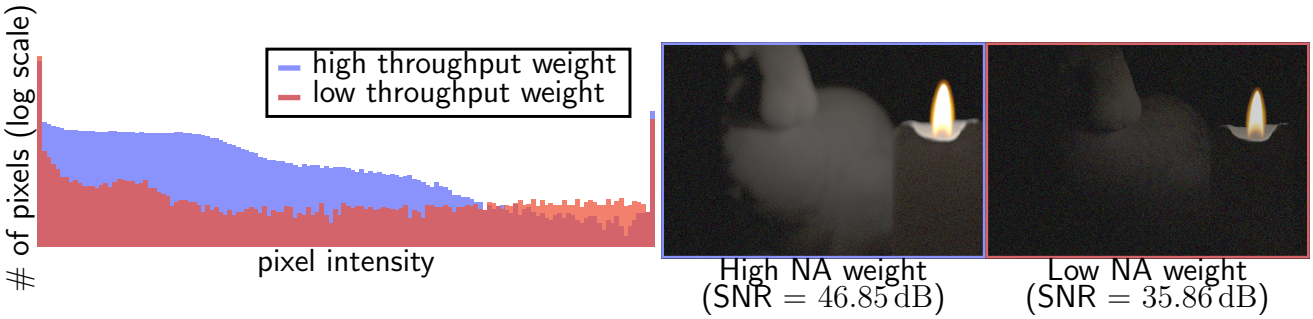


Figure 3.7: We simulate images of a candlelit scene using lenses optimized for high throughput ($\beta_{\mathrm{NA}} = 5 \times 10^{-4}$, bottom left) versus small spot size ($\beta_{\mathrm{NA}} = 1 \times 10^{-5}$, bottom right). We incorporate shot and read noise in rendered images—simulated for a Canon 7D sensor [19] at unity ISO (984)—then gamma-correct ($\gamma = 0.2$) the noisy images. We also plot the histogram of pixel intensities in the two images (top). The image from the high-throughput lens has less pronounced noise and an intensity distribution shifted towards larger values; but also slightly worse blur (e.g., at the candle flame).

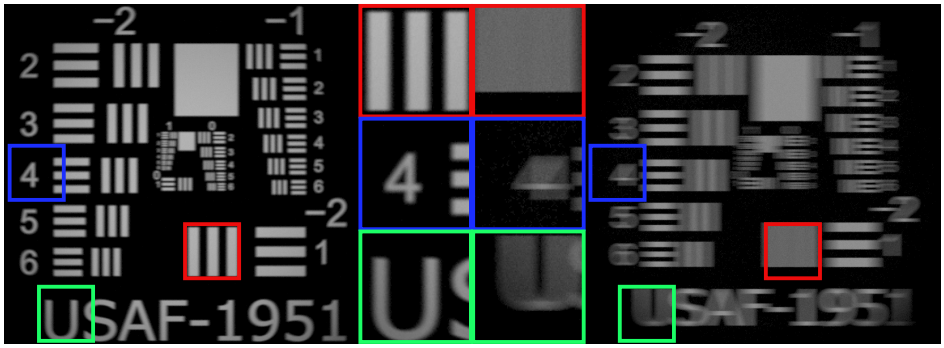


Figure 3.8: We simulate images of a moving USAF resolution target using lenses optimized for high throughput ($\beta_{\mathrm{NA}} = 5 \times 10^{-4}$, left) versus small spot size ($\beta_{\mathrm{NA}} = 1 \times 10^{-5}$, right). The high-throughput lens is well suited for this dynamic scene: it achieves the same overall brightness at one-fourth the exposure time, resulting in much reduced motion blur.

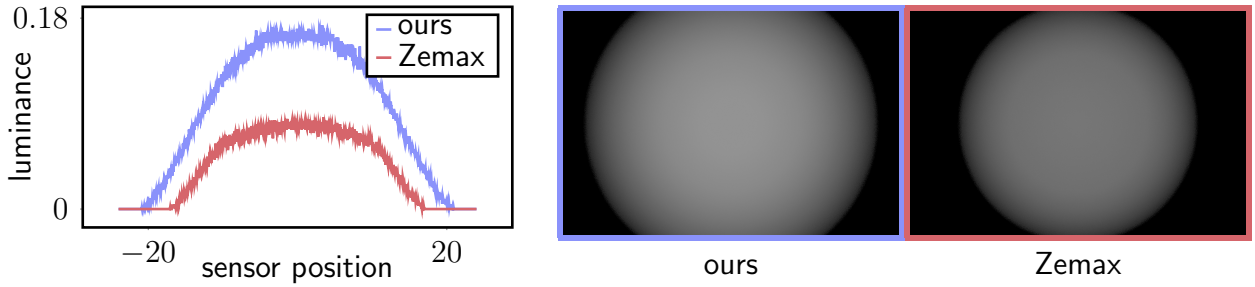


Figure 3.9: We simulate images of a white balance target using lenses optimized with our technique (bottom left) versus Zemax [109] (bottom right). We also plot the horizontal cross section of the two images (top). By emphasizing throughput, including from off-axis sources, the lens from our technique strongly mitigates vignetting away from the sensor center.

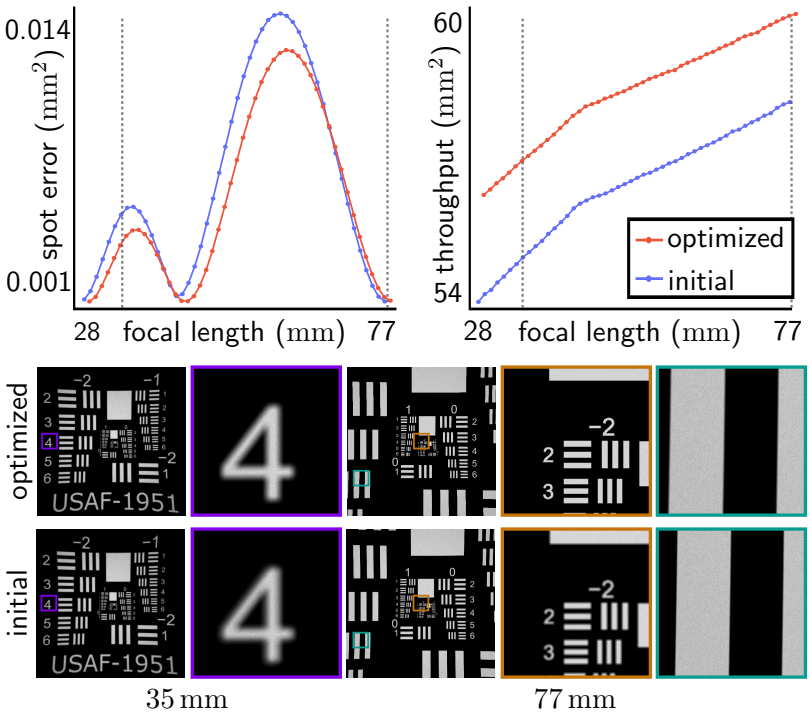


Figure 3.10: We use our technique to optimize a zoom lens design in the focal length range $28\,\mathrm{mm}$ to $77\,\mathrm{mm}$. We sum the composite objective of Equation (3.26) at three target focal lengths, $28\,\mathrm{mm}$, $45\,\mathrm{mm}$, and $77\,\mathrm{mm}$. We plot the spot-size error (top left) and throughput (top right) of the initial and optimized designs across the focal length range, and simulate images of a USAF resolution target at focal lengths $35\,\mathrm{mm}$ (bottom left) and $77\,\mathrm{mm}$ (bottom right). The optimized design has overall better spot-size error and throughput across the focal length range. At the target focal length $77\,\mathrm{mm}$, both lenses achieve the same overall spot-size error, but distribute the error to different parts of the sensor (insets).

Chapter 4

Mixed Discrete-Continuous Design of Compound Lenses

Gradient-based optimization has become an essential part of the design process for making compound lenses as performant as possible. Recent advances in differentiable rendering facilitate faster and more efficient gradient-based optimization of continuous lens parameters (e.g., shapes of individual elements or distances between them) [90, 94, 96, 103]. Though such methods are invaluable for improving lens designs with a fixed *topology* (number and type of lens elements), they cannot optimize the lens topology itself. It is up to the designer to perform discrete changes manually—strategically adding, changing, or removing elements from a base design—before handing the design back to the optimizer for further improvement. And critically, changing the lens topology is often the *only* option available for meeting the stringent requirements of challenging design tasks, e.g.: drastically improving sharpness when transitioning a lens from a half-frame to a full-frame sensor; increasing speed when designing a lens for extreme low-light conditions; or maintaining sharpness and speed when adapting a lens to smaller form factors.

Another way to assist designers is by generating good starting points and performing global search through the use of deep learning or genetic algorithms [21, 39, 113]. These methods can help seed the design process, but the seed points they output are often suboptimal: though they can make discrete changes to lens topology, they cannot effectively optimize continuous lens parameters, and thus cannot accurately assess the potential of different lens topologies. This inability limits the scope of lens designs that can be explored automatically. As a result, pushing the envelope in lens performance still requires expert designers to manually explore edits and iterate between discrete and continuous optimization.

These considerations highlight a critical gap in automated design of compound lenses: existing methods optimize only continuous or only discrete parameters of a system whose performance critically depends on *joint* optimization of both types of parameters. In light of this gap, we seek a method that can perform mixed discrete-and-continuous lens optimization automatically, to better assist expert designers. To this end, we introduce a method that uses *Markov chain Monte Carlo* (MCMC) sampling for compound lens design. Our method allows combining gradient-based optimization of continuous lens parameters with *transdimensional*

mutations that alter the number of lens elements. We make such a combination practical through two core contributions:

- A sampling algorithm that facilitates mixing gradient updates and mutations without sacrificing optimization performance.
- A set of mutations that use a projection operation to propose lenses with varied topologies that nevertheless remain paraxially equivalent to an original design.

Altogether, these contributions allow our method to explore a much larger design space of compound lenses than previously possible in an automated manner. We show experimentally that our method finds lenses with improved sharpness and speed compared to prior work that focuses on only gradient-based optimization. In the supplement, we provide interactive visualizations and code for our method, which we will make open-source upon acceptance.

4.1 Related work

Lens design exploration The design space of lenses is highly nonconvex and contains local minima. Prior work uses simulated annealing [112], genetic algorithms [39], or large language models [113] to mitigate these issues. These methods can help find performant designs, but use a predetermined number of elements.

To expand the search space, Betensky [8] proposed exchanging parts of a design with paraxially equivalent designs. Similarly, we use paraxial optics to choose design mutations. Alternatively, Sun et al. [89] search over a library of off-the-shelf elements to directly generate a design without continuous optimization. Deep learning has also been used to generate initial designs for further continuous optimization [21]. Our method combines continuous optimization with discrete mutations, without requiring learning.

MCMC for computational design Practitioners in other areas often use MCMC methods for discrete optimization problems. Closely related to our work are applications of MCMC to computational design problems. For example, Yeh et al. [107] use reversible-jump MCMC to decide the placement of furniture in virtual rooms. Desai et al. [24] use MCMC to optimize the placement of components of a mechanical assembly. More recently, Barda et al. [7] use MCMC to optimize for the design of sheet metal parts.

MCMC for rendering In computer graphics, MCMC finds application in rendering to estimate light transport integrals [98]. Related to our work are methods that mix sampling distributions of varying dimensionality [9, 64]. Other approaches use local gradients or Hessians to guide path sampling [52, 56].

Quasi-stationary Monte Carlo Metropolis-Hastings adjusted processes reject samples, which can be inefficient, especially when each sample is costly to generate (e.g., in optimization). Quasi-stationary Monte Carlo (QSMC) methods sample from a target distribution without rejections. Unlike Metropolis-Hastings, QSMC methods do not require

reversible transitions or detailed balance for convergence. Pollock et al. [70] and Wang et al. [102] proposed QSMC algorithms that sample from a target distribution using Brownian motion. In rendering, Holl et al. [37] use the jump restore algorithm [102] to mix global and local dynamics in Metropolis light transport. We use the jump restore framework to develop a sampler that navigates the design space of lenses.

4.2 Lens design objectives

We optimize a compound lens θ with respect to a combination of lens design objectives that emphasize sharpness, speed, and conformity to focal length specifications. We define each loss for rays starting from the same location on the target plane, then aggregate losses for many such locations.

1. *Sharpness:* We use a variance-based *spot size loss*:

$$\mathcal{L}_{\text{spot}}(\mathrm{x}_0;\theta) \coloneqq \int_{(\mathrm{x}_0,\mathrm{v}_0)\in\Omega(\theta)} \left\|\mathrm{x}_{\mathrm{s}}(\mathrm{x}_0,\mathrm{v}_0;\theta) - \tilde{\mathrm{x}}(\mathrm{x}_0)\right\|^2 \mathrm{dv}_0, \tag{4.1}$$

$$\tilde{\mathrm{x}}(\mathrm{x}_0) \coloneqq \frac{\int_{(\mathrm{x}_0,\mathrm{v}_0)\in\Omega(\theta)} \mathrm{x}_{\mathrm{s}}(\mathrm{x}_0,\mathrm{v}_0;\theta)\,\mathrm{dv}_0}{\int_{(\mathrm{x}_0,\mathrm{v}_0)\in\Omega(\theta)} \mathrm{dv}_0}. \tag{4.2}$$

2. *Speed:* We use a *throughput loss* inspired by Teh et al. [94]:

$$\mathcal{L}_{\text{throughput}}(\mathrm{x}_0;\theta) \coloneqq 1 - \frac{1}{T_0}\int_{(\mathrm{x}_0,\mathrm{v}_0)\in\Omega(\theta)} \mathrm{dv}_0, \tag{4.3}$$

   where T_0 is the maximum throughput determined by the maximum size of the design, which we specify as a hyperparameter.

3. *Focal length:* We use a loss based on the target focal length f,

$$\mathcal{L}_{\text{focal}}(\mathrm{x}_0;\theta) \coloneqq \left\|\tilde{\mathrm{x}}(\mathrm{x}_0) - \mathrm{x}_{\text{thin}}(\mathrm{x}_0;f)\right\|^2, \tag{4.4}$$

   where $\mathrm{x}_{\text{thin}}(\mathrm{x}_0;f)$ is the sensor point predicted for x_0 from the Gaussian lens formula for a thin lens of focal length f.

We include an additional regularization term that prevents lens elements from becoming thinner than a hyperparameter d_{min}:

$$\mathcal{L}_{\text{thickness}}(\theta) \coloneqq \sum\nolimits_{k=1}^{K} \max\left(d_{\text{min}} - t_k, 0\right)^2, \tag{4.5}$$

where t_k is the thickness of the k-th lens element.

Our total loss is a weighted combination of these losses, aggregated over multiple target locations where appropriate:

$$\begin{aligned}\mathcal{L}(\theta) \coloneqq \sum\nolimits_{m=1}^{M} \big(&w_{\text{spot}}\mathcal{L}_{\text{spot}}(\mathrm{x}_0^m;\theta) + w_{\text{throughput}}\mathcal{L}_{\text{throughput}}(\mathrm{x}_0^m;\theta) \\ &+ w_{\text{focal}}\mathcal{L}_{\text{focal}}(\mathrm{x}_0^m;\theta)\big) + w_{\text{thickness}}\mathcal{L}_{\text{thickness}}(\theta).\end{aligned} \tag{4.6}$$

When the number K of lens elements (and thus length N of θ) is fixed a priori, the loss of Equation (4.6) is differentiable using adjoint sequential ray tracing [94]. Therefore, for fixed N, the design of a compound lens can be done using gradient-based optimization, an approach that has been popular in recent work [90, 94, 96, 103].

Unlike this prior work, we are interested in designing *all aspects* of a compound lens, including both the discrete-valued number K of lens elements and their continuous-valued parameters θ. Though the discrete parameter is not amenable to gradient-based optimization, we would like any such method to continue to use gradient-based optimization for the continuous parameters, given its prior success. We enable *mixed discrete-continuous design* by treating it as a sampling problem that we attack with *Markov chain Monte Carlo* (MCMC) algorithms. To this end, we first convert the loss in Equation (4.6) into a positive Boltzmann distribution to be *maximized*:

$$\pi(\theta) \coloneqq e^{-\frac{\mathcal{L}(\theta)}{T}}, \tag{4.7}$$

where T is a *temperature* constant that controls the "sharpness" of the distribution. Intuitively, higher T allows sampling to explore more regions of the design space, whereas lower T forces sampling to explore only high-value regions. We treat T as a fixed hyperparameter, though future work could explore *simulated annealing* methods that progressively lower temperature over time.

4.3 Markov chain Monte Carlo for optimization

One way to approach solving an optimization problem is to build a sampler for the Boltzmann distribution defined in Equation (4.7). Then, with high probability, the sampler will produce samples that are close to locally optimal solutions. This approach is flexible because of that fact that we have not imposed any restrictions on the properties of the variables that we are optimizing over. The state space could be comprised of both continuous and discrete variables. For this reason, using sampling in the lens design case is a natural choice, as we want to be able to add the number of elements in the lens as part of the optimization process.

In order to build this sampler, we turn to *Markov chain Monte Carlo* (MCMC) methods, which are a class of algorithms that sample from a target distribution by constructing a Markov chain whose stationary distribution is the target distribution. Markov chains are a more convenient object to work with since we can reason about local mutations between states, rather than directly working with a global distribution that might be computationally intractable.

Metropolis-Hastings and its pitfalls. Probably the most popular MCMC method is the *Metropolis-Hastings* (MH) algorithm. Given a current compound lens θ, it uses a *proposal distribution* p to sample a lens proposal, $\theta' \sim p(\theta \to \theta')$, and compute an acceptance probability:

$$\alpha \coloneqq \min\left\{1, \exp\left(\frac{\mathcal{L}(\theta) - \mathcal{L}(\theta')}{T}\right) \cdot \frac{p(\theta' \to \theta)}{p(\theta \to \theta')}\right\}. \tag{4.8}$$

The proposal is randomly either accepted with probability α and used to update the lens ($\theta \leftarrow \theta'$), or rejected leaving the lens unchanged. The exponential term in Equation (4.8) favors accepting proposals θ' that improve (reduce) $\mathcal{L}$. Under certain conditions on p, this algorithm will sample lenses proportional to $\mathcal{L}$.

The appeal of MH for mixed discrete-continuous design lies in the great flexibility they afford the user in selecting the proposal distribution p. We can use a *mixture* proposal distribution:

$$p(\theta \to \theta') = \beta p_{\text{continuous}}(\theta \to \theta') + (1-\beta) p_{\text{discrete}}(\theta \to \theta'), \tag{4.9}$$

which at random updates lens parameters θ either continuously by sampling $p_{\text{continuous}}$—a *perturbation* that leaves the dimension N of θ the same—or discretely by sampling p_{discrete}—a *mutation* that changes N. Moreover, for $p_{\text{continuous}}$, we can use *Langevin perturbations* $\theta' \sim \mathcal{N}(\theta - \eta \, {}^{\mathrm{d}\mathcal{L}}/_{\mathrm{d}\theta}, \eta)$, or equivalently:

$$\theta' \leftarrow \theta - \eta \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta} + \sqrt{\eta} \cdot \varepsilon, \tag{4.10}$$

where ε is a vector of normal random variates, and the gradient term can be computed using adjoint sequential ray tracing. The Langevin perturbations in Equation (4.10) mimic gradient descent, except for the addition of noise, allowing MH to incorporate gradient-based optimization of continuous lens parameters.

MH with mixed Langevin perturbations and discrete mutations has been successful elsewhere in computer graphics, notably in work on Monte Carlo rendering [52, 56] that also shows how to use Langevin perturbations with adaptive stepsizes (e.g., as in Adam [50]). Unfortunately, we have empirically found it difficult to use this approach for mixed discrete-continuous design of compound lenses, for two reasons:

1. The sharpness and speed of a compound lens θ with many elements (e.g., $K \geq 4$) is *very* sensitive to random perturbations. Langevin perturbations include noise ε (Equation (4.10)), and thus almost always produce badly performing lenses that are rejected. Preventing this behavior requires keeping the noise variance—and thus stepsize η—very small. The result in either case is that optimization requires prohibitively many sampling iterations.
2. Lens mutations that have high acceptance probability generally require refining a lens after increasing or decreasing its elements. Such post-mutation refinement requires continuous optimization, which makes it challenging—or impossible—to compute the reverse proposal distribution $p(\theta' \to \theta)$ in Equation (4.8).

We provide experimental evidence for both issues in Section 5.5. Fundamentally, these issues arise from the requirement for *reversible proposals* in MH, which necessitates the addition of noise in Equation (4.10) and the computation of $p(\theta' \to \theta)$ in Equation (4.8). In Section 4.4, we address both issues by using a different MCMC sampling algorithm that lifts the reversibility requirement, alongside providing other advantages for design. Then, in Section 4.5, we design effective discrete lens mutations, leveraging the flexibility from no longer being constrained by reversibility.

Algorithm 1 SingleRestoreStep$(\theta, R, \gamma\pi, C)$

Input: *A compound lens θ, a reservoir R, a weight parameter γ, a target distribution π, a constant C.*

Output: *A new lens $\tilde{\theta}$, an updated reservoir R.*

▷ *Perform a gradient step*
$\tilde{\theta} \leftarrow$ GradientStep(π, θ)
▷ *Compute the termination probability*
$\kappa \leftarrow \frac{\pi(\theta)-\pi(\tilde{\theta})+C}{\pi(\theta)+C}$
▷ *Check whether to terminate perturbation sequence*
if SampleUniform$[0,1] < \kappa$ **then**
    ▷ *Update the reservoir with the current lens*
    $R \leftarrow$ UpdateReservoir$(R, \tilde{\theta})$
    ▷ *Sample and mutate new lens*
    if SampleUniform$[0,1] < \gamma$ **then**
        $\tilde{\theta} \leftarrow$ SampleGlobal()
    else
        $\tilde{\theta} \leftarrow$ SampleReservoir(R)
    $\tilde{\theta} \leftarrow$ MutateLens$(\tilde{\theta})$
▷ *Return new lens and updated reservoir*
return $\tilde{\theta}, R$

4.4 The RESTORE algorithm

We propose to use an alternative MCMC sampling algorithm based on so-called *randomly exploring and stochastically regenerating* (RESTORE) processes [102]—we use the name RESTORE also for the sampling algorithm associated with these processes. RESTORE uses two proposal distributions, one for small perturbations, and another for large changes potentially including independent sampling. The RESTORE literature refers to these distributions as *local dynamics* and *regeneration probability* (resp.); for clarity, we will continue to use the terms *perturbation* and *mutation* (resp.) we introduced in the previous section.

Starting from an initial lens θ, RESTORE performs a sequence of perturbations updating the lens ("local dynamics simulation"). At each perturbation, it uses the current lens θ to compute a *termination probability* to randomly decide whether to continue perturbations or terminate the sequence. Upon termination, it updates θ using the mutation distribution ("regeneration"), then restarts a perturbation sequence. Compared to MH, RESTORE has no rejections, and does not require the perturbation and mutation distributions to be reversible—both properties that benefit optimization, as prior work in other graphics areas has demonstrated [37]. Instead of reversibility, RESTORE relies on careful selection of the mutation distribution and termination probability to ensure it samples the target distribution π [102]. First we will describe the original RESTORE algorithm, then we will detail our

choices for the perturbation and mutation distributions. These mutations will then determine the termination probability we should use in the algorithm. Along the way, we elaborate on advantages over MH. We summarize the version of RESTORE we use in Algorithm 1.

4.4.1 Algorithm overview

The RESTORE algorithm runs by mixing two types of operations: perturbations and global regenerations through an exponential clock with a rate κ. The algorithm at each step samples two exponential random variables T_{local} and T_{global} with rate 1 and κ, respectively. Whichever of the two is smaller determines the next step of the algorithm. If $T_{\text{local}} < T_{\text{global}}$, then the algorithm performs a perturbation step, otherwise it performs a regeneration step. Each step yields a weighted sample with a weight of $\min(T_{\text{local}}, T_{\text{global}})$. This procedure converges to the correct target distribution if

$$\tilde{\kappa} = \frac{Q^*\pi(\theta)}{\pi(\theta)} + C\frac{\mu(\theta)}{\pi(\theta)}, \tag{4.11}$$

where Q^* is the adjoint infinitesimal generator of the perturbation distribution, μ is the regeneration distribution, and C is a constant that ensures that the right-hand side is positive. In order to use RESTORE, we need to specify the perturbation process as well as its infinitesimal generator along with a global regeneration distribution.

Perturbation distribution. As RESTORE does not require reversible proposals, we use gradient perturbations *without noise*:

$$\theta^* \leftarrow \theta - \eta\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta}. \tag{4.12}$$

Thus a perturbation sequence becomes equivalent to gradient descent. The lack of noise is crucial for performance: Whereas in Langevin perturbations (Equation (4.10)) the noise term required keeping step size η small, in Equation (4.12) we can use large step sizes to accelerate optimization. We use Adam [50] to determine adaptive step sizes from the history of lenses θ in the current perturbation sequence. As we explain below, the sequence terminates randomly with a probability that adapts to how much progress gradient steps make towards improving the lens θ.

The gradient perturbation process is a jump process, which means that it *jumps* from on state to another in a discrete manner. The adjoint infinitesimal generator of the gradient perturbation is

$$Q^*_{\text{gradient}}\pi(\theta^*) = \pi(\theta) - \pi(\theta^*). \tag{4.13}$$

This form comes from plugging in Equation Equation (4.12) into the definition of the adjoint infinitesimal generator of a jump process [26].

Mutation distribution. RESTORE requires that the mutation distribution can sample any lens θ with non-zero probability. In practice, the choice of mutation distribution is further constrained by its role in determining the termination probability—through an integral operator relationship that is difficult to compute analytically except for trivial distributions (e.g., uniform sampling). We follow Pollock et al. [70], who show that using a mutation distribution that selects from a reservoir of previous samples allows a tractable approximation of the termination probability (described below).

In particular, we first sample a lens θ from a mixture distribution

$$\mu(\theta) = \gamma p_{\text{global}}(\theta) + (1-\gamma) p_{\text{reservoir}}(\theta, R), \tag{4.14}$$

where: 1. p_{global} samples a lens θ by sampling each of its parameters independently and uniformly within some upper and lower bounds; 2. $p_{\text{reservoir}}$ samples a lens from a reservoir R of previously sampled lenses, formed as we explain below; 3. $\gamma \in [0, 1]$ is a hyperparameter. We set γ to a small value, to ensure that the mutation distribution satisfies the RESTORE requirement while mostly sampling from the reservoir. After sampling a lens θ, we mutate it as we describe in Section 4.5. As the overall mutation distribution does not need to be reversible, our mutations allow altering the number of lens elements, then refining the lens before resuming a perturbation chain.

We form the reservoir R by keeping track of the top N lenses (in terms of π) sampled upon perturbation chain terminations. The reservoir effectively allows RESTORE to "backtrack" to a previous high-performing lens, when a perturbation chain gets trapped exploring a bad region of the design space. This backtracking is important as without it the mutation distribution would be extremely unlikely to find a reasonable lens by randomly sampling the design space. Instead, using the reservoir allows "warm starting" a new perturbation chain from a previous reasonable lens. As the chain continues, the reservoir R will contain samples from the target distrbution, so the regeneration distribution μ that we are effectively sampling from is an approximation to our target distribution,

$$\mu(\theta) \approx \pi(\theta). \tag{4.15}$$

Termination probability Given our specifications of perturbations and global regenerations, we can write, $\tilde{\kappa}$ as

$$\tilde{\kappa} \leftarrow \frac{\pi(\theta) - \pi(\theta^*)}{\pi(\theta)} + C. \tag{4.16}$$

For optimization, the associated weights with each sample are unnecessary, since we are searching for the *maximum* of the target distribution. We can therefore simply calculate the probability of terminating the current gradient descent sequence, κ, as

$$\kappa \leftarrow \frac{\pi(\theta) - \pi(\tilde{\theta}) + C}{\pi(\theta) + C}. \tag{4.17}$$

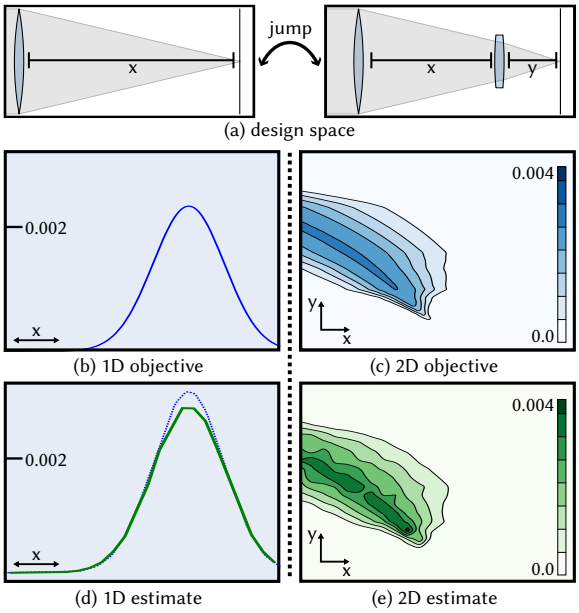


Figure 4.1: We construct a toy lens design problem *(a)* to validate the sampling accuracy of our method. The sampler can choose between optimizing the distance of a singlet from the sensor (1D target distribution, *left*), or the distances between two singlets and the sensor (2D target distribution, *right*). Even with an approximate termination probability, our method *(d-e)* correctly samples from the target distributions *(b-c)* in both 1D and 2D.

Intuitively, κ increases, and thus termination becomes more likely, when the relative improvement of π after a gradient step becomes small. Thus, RESTORE *adaptively* determines to stop perturbations and use a mutation when gradient optimization gets stuck at a local maximum of π. Though Equation (4.16) is an approximation, we show in Figure Figure 4.1 that using it in RESTORE still allows sampling from the target distribution π if we account for the weights calculated by the timers.

4.5 Lens mutations

We implement four mutations in our method (Figure 4.2): singlet addition, singlet removal, singlet gluing, and doublet splitting. These mutations are inspired by strategies suggested in lens design reference textbooks [84, Chapter 2]. During sampling, we uniformly sample an element of the current compound lens, then uniformly sample a mutation strategy for the selected element, and lastly apply a *refinement* process. We use singlet addition as an example to explain this refinement process, but the same process applies to the other mutations. Additionally, our refinement process is extensible to other types of mutations, e.g., changing the material type or flipping the orientation of lens elements.

If we simply add a singlet to the current compound lens, the resulting lens will almost certainly have drastically worse focusing and throughput performance than the original lens. Though RESTORE will still attempt to optimize the mutation proposal, optimization

will likely get stuck fast requiring a new mutation—and thus wasting the previous ones. We therefore need a way to quickly fine-tune the lens from a mutation proposal so that it performs closer to the original lens. To do so, we use *ray transfer matrix analysis*—a paraxial approximation to ray tracing—as a proxy for Equation (4.6) that allows quickly improving lens focusing performance.

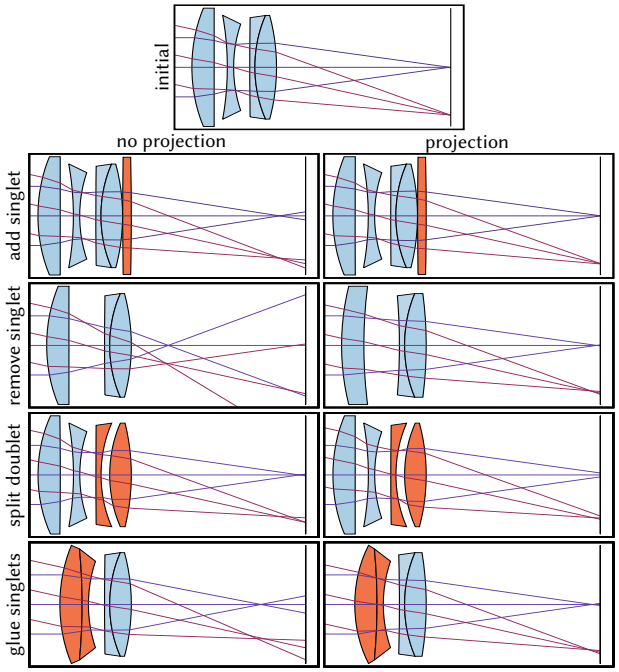


Figure 4.2: We consider four types of mutations to a compound lens. Applying such a mutation on its own often leads to a lens with significantly worse performance than the original. We alleviate this issue by using a *paraxial projection* refinement process that updates the mutated lens to have the same focusing behavior (up to first order) as the original lens.

Paraxial lens optics. With the paraxial approximation of radially symmetric optics, we parameterize a light ray using the 2D-vector of its distance y and angular deviation ϕ from the optical axis. We also model a sequential compound lens using a 2×2 *ray transfer matrix* M. Then, we can model propagation of the ray through the lens using matrix-vector multiplication:

$$\begin{bmatrix} \phi_{\mathrm{f}} \\ y_{\mathrm{f}} \end{bmatrix} = M \begin{bmatrix} \phi \\ y \end{bmatrix}, \tag{4.18}$$

We overview this matrix for compound lenses comprising spherical elements, and refer to Pedrotti et al. [68, Chapter 18] for details.

As a ray traveling through a lens undergoes a sequence of propagation and refraction operations, it suffices to explain how to model each such operation paraxially as a ray transfer matrix. Propagation by a distance d corresponds to a matrix:

$$M_p(d) \coloneqq \begin{bmatrix} 1 & 0 \\ d & 1 \end{bmatrix}, \tag{4.19}$$

Refraction at a spherical surface with curvature κ corresponds to:

$$M_r(\kappa, \eta_{\mathrm{o}}, \eta_{\mathrm{i}}) \coloneqq \begin{bmatrix} \frac{\eta_{\mathrm{i}}}{\eta_{\mathrm{o}}} & \frac{\kappa(\eta_{\mathrm{i}} - \eta_{\mathrm{o}})}{\eta_{\mathrm{o}}} \\ 0 & 1 \end{bmatrix}, \tag{4.20}$$

where η_{i} and η_{o} are the refractive indices before and after refraction.

We can then model a compound lens as the product of the refraction and propagation matrices corresponding to its elements. For example, a singlet is represented paraxially as $M_r M_p M_r M_p$, where the last matrix considers the distance from the final lens surface to the sensor plane. Given a compound lens θ, we denote its ray transfer matrix $M(\theta)$, which can always be written in the form:

$$M(\theta) = \begin{bmatrix} a & -\frac{1}{f} \\ b & c \end{bmatrix}, \tag{4.21}$$

where f is the focal length of the lens and a, b, and c are coefficients that encode other first-order properties of the lens. Importantly, $M(\theta)$ provides a fixed-sized abstraction of the compound lens, no matter the complexity of its design (e.g., number of elements). Moreover, two compound lenses will have similar (up to first order) focusing behavior if their ray transfer matrices are equal, no matter how different their internal designs. Therefore, we can define the following notion of approximate equivalence between two compound lenses—emphasizing focusing of rays parallel to the optical axis and at infinite conjugacy ($y = 1, \phi = 0$).

Definition 1: Paraxial equivalence

Two compound lenses θ_a and θ_b are *paraxially equivalent* if

$$M(\theta_a)\begin{bmatrix}0\\1\end{bmatrix} = M(\theta_b)\begin{bmatrix}0\\1\end{bmatrix}.$$

Mutation procedure. After adding a singlet to a lens θ, we refine the augmented lens θ^* so that it is paraxially equivalent to θ. We formulate this refinement as a constrained optimization problem;

$$\min_{\theta^*} \|\theta - \theta^*\|^2 \quad \text{s.t.} \quad (M(\theta) - M(\theta^*))\begin{bmatrix}0\\1\end{bmatrix} = 0. \tag{4.22}$$

Optimization excludes entries of θ^* corresponding to the added singlet. This refinement process, which we term *paraxial projection*, ensures that θ^* has the same first-order behavior as θ, and applies to our other mutation operations (Figure 4.2) exactly analogously.

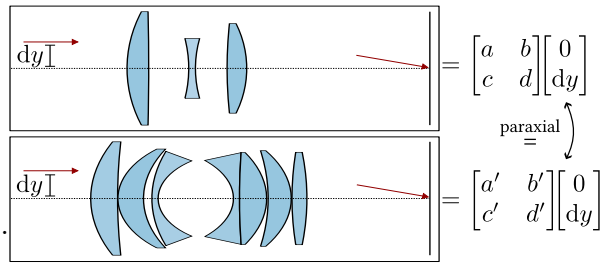


Figure 4.3: Example of paraxial equivalence. The 2×2 ray transfer matrix describes how a light ray parallel to the optical axis propagates through a compound lens (up to first order). If two lenses have the same effect on such a ray, they are *paraxially equivalent*.

We solve Equation (4.22) using Newton's method and Lagrange multipliers. Empirically, we found that the solver runtime is equivalent to about 30 gradient-based iterations using full (adjoint) ray tracing. We also found that using paraxial projection after a mutation results in much larger improvements in the lens objective than using gradient descent for the same amount of time. We note that paraxial projection is locally unique and thus has a full-rank Jacobian of θ^* with respect to θ—the supplement has a derivation. Though the Jacobian is not needed for our method, it can be used for MH-based sampling requiring reversibility.

Jacobian of manifold projection If we call the solution to Equation 4.22, θ^*, and the initial parameters, θ_0, then we are interested in the Jacobian of θ^* w.r.t. θ_0. This optimization problem can be differentiated using the implicit function theorem. We first

write the Lagrangian of the optimization problem.

$$\mathcal{L}(\theta^*, \theta_0, \lambda) = \|\theta^* - \theta_0\|^2 \tag{4.23}$$

$$+ \lambda^\top \left(M\left(\begin{bmatrix} \theta^* & \theta_{\text{add}} \end{bmatrix}^\top \right) - M(\theta_0) \right) \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{4.24}$$

Following the method of Lagrange multipliers, we know that the solution to the optimization problem is a stationary point of the Lagrangian,

$$\nabla_1 \mathcal{L} = 0 \tag{4.25}$$

$$\nabla_3 \mathcal{L} = 0, \tag{4.26}$$

where ∇_i refers to differentiation with respect to the i-th argument. We now assume that both θ and λ are functions of θ_0 and differentiate the Lagrangian w.r.t. θ_0,

$$\nabla_1 \nabla_1 \mathcal{L} \frac{\mathrm{d}\theta^*}{\mathrm{d}\theta_0} + \nabla_2 \nabla_1 \mathcal{L} + \nabla_3 \nabla_1 \mathcal{L} \frac{\mathrm{d}\lambda}{\mathrm{d}\theta_0} = 0, \tag{4.27}$$

$$\nabla_1 \nabla_3 \mathcal{L} \frac{\mathrm{d}\theta^*}{\mathrm{d}\theta_0} + \nabla_2 \nabla_3 \mathcal{L} + \nabla_3 \nabla_3 \mathcal{L} \frac{\mathrm{d}\lambda}{\mathrm{d}\theta_0} = 0. \tag{4.28}$$

Notice that both $\nabla_1 \mathcal{L}$ and $\nabla_2 \mathcal{L}$ have the same dimension, while $\nabla_3 \mathcal{L}$ has a dimension of the number of constraints (two in our case). Writing this out in matrix form,

$$\begin{bmatrix} \nabla_1 \nabla_1 \mathcal{L} & \nabla_3 \nabla_1 \mathcal{L} \\ \nabla_1 \nabla_3 \mathcal{L} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}\theta^*}{\mathrm{d}\theta_0} \\ \frac{\mathrm{d}\lambda}{\mathrm{d}\theta_0} \end{bmatrix} = \begin{bmatrix} -\nabla_2 \nabla_1 \mathcal{L} \\ -\nabla_2 \nabla_3 \mathcal{L} \end{bmatrix}, \tag{4.29}$$

and plugging in Equation 4.24 gives,

$$\begin{bmatrix} \mathbf{I} + \lambda^\top \nabla_1 \nabla_1 g & \nabla_1 g \\ \nabla_1 g^\top & \mathbf{0} \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}\theta^*}{\mathrm{d}\theta_0} \\ \frac{\mathrm{d}\lambda}{\mathrm{d}\theta_0} \end{bmatrix} = \begin{bmatrix} -\left(\mathbf{I} + \nabla_2 \nabla_1 g\right) \\ -\nabla_2 g^\top \end{bmatrix}, \tag{4.30}$$

where g is the constraint function in Equation 4.24. We can now perform a linear solve to find the Jacobian of the projection.

4.6 Experiments

We show results from experiments on several lens design tasks.

Implementation details. We implemented our method in Python, using JAX [11] for GPU acceleration, and Optimistix [72] for ray transfer matrix projection. We assume a standard 35 mm full-frame sensor, and optimize for four target plane positions in Equation (4.6), sampled so that their thin-lens sensor-plane projections are uniformly distributed from the center to the edge of the sensor. We set reservoir size to 5, and temperature so that the initial lens θ_0 has $\pi(\theta_0) \approx 0.5$. All our initial lenses are from Reiley [73]. On a workstation with an NVIDIA RTX 3090 GPU, our implementation performs 12000 iterations in about 40 min.

Comparison with Metropolis-Hastings In Figure 4.4, we compare our method with a reversible-jump MCMC sampler that uses the Metropolis-adjusted Langevin algorithm (i.e., MH with Langevin perturbations) [75]. As we explain in Section 4.3, the need to include noise in Langevin perturbations forces the MH-based sampler to use very small step sizes to ensure perturbations are accepted—thus slowing down optimization process. Additionally, the MH-based sampler rejects most mutation proposals as, even with projection, they are significantly worse than the current lens. By contrast, our method uses large step sizes for faster gradient optimization and, as it has no rejections, attempts to optimize mutation proposals before performing another mutation. As a result, our method consistently finds better lenses.

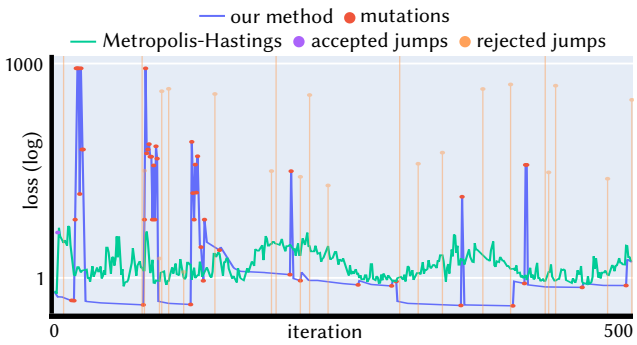


Figure 4.4: We compare our method with an MH-based sampler. The MH-based sampler (green) rejects most mutations proposals and can only perform small gradient steps. By contrast, our method (blue) efficiently optimizes all mutation proposals before trying another mutation.

Comparison with brute-force search In Figure 4.5, we compare our method against a baseline using brute-force search. Brute-force search works by adding or removing a singlet at every possible location in the compound lens, then optimizing the resulting lens using gradient descent. This baseline is simple to implement, but is prone to getting stuck in local minima and can become intractable as we increase the number of elements in the original lens or the number of elements we want to add. By contrast, our method can better avoid local minima and explore more lens designs by: adaptively adding and removing elements, using paraxial projection, and dynamically terminating stalled optimization.

We use four initial lenses for comparison: a 28 mm wide-angle lens, a 50 mm normal lens, a 105 mm macro lens, and a 135 mm telephoto lens. We run our method and the brute-force baseline for the same time (40 min), optimizing all lenses generated from additions and removals by the baseline for an equal number of gradient iterations. In all cases, our method finds lenses with better objective values, sharpness, and speed than the baseline.

Paraxial projection ablation Figure 4.6 and Table 4.1 show results from an ablation study evaluating the utility of paraxial projection. We run our method with and without paraxial projection for 5000 iterations. The table shows that our method samples lenses that improve on the initialization much more often with paraxial projection than without. Figure 4.6 shows how mutations are distributed during sampling for the 135 mm lens, with and without paraxial projection. In both cases, there is a "warm-up" period where our method must first populate the reservoir with good lenses, before it starts consistently finding better lenses. Using paraxial projection greatly shortens this period, resulting in overall

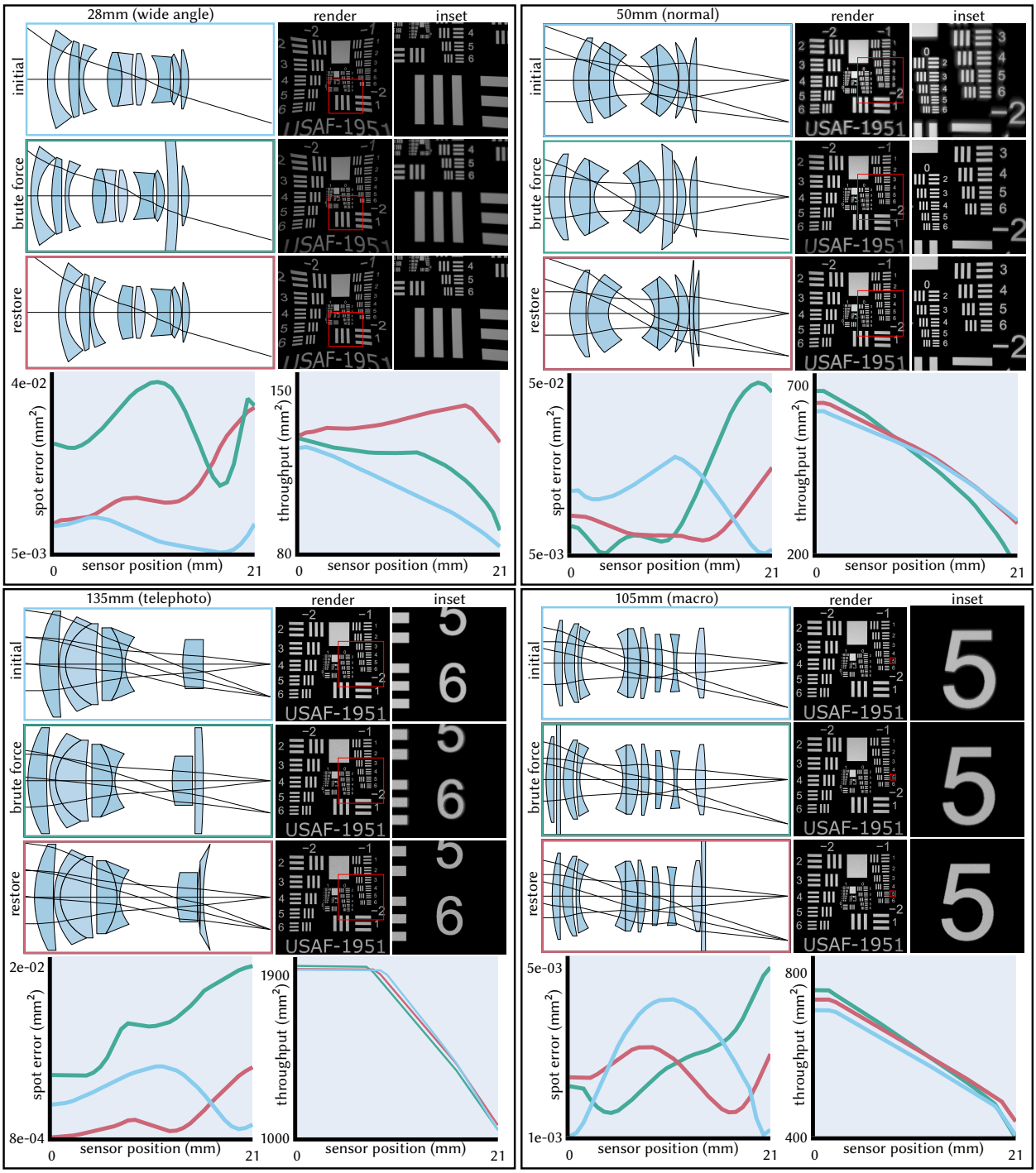


Figure 4.5: A comparison of our method with a baseline using brute force search. The baseline adds or removes an element at every possible location in the original compound lens, then uses gradient-based optimization for the resulting lens. Both methods run for $40\,\mathrm{min}$. We experiment on four different types of lenses: a $28\,\mathrm{mm}$ wide-angle lens, a $50\,\mathrm{mm}$ normal lens, $105\,\mathrm{mm}$ macro lens, and a $135\,\mathrm{mm}$ telephoto lens. In all cases, our method finds better lenses than baseline for the given objective function.

Table 4.1: Ablation study for paraxial projection. We run our method with and without paraxial projection, and report the fraction of iterations in which the sampled lenses have better objective value than the initial lens. We report results after 1000 and 5000 iterations.

	projection		no projection	
lens	1k	5k	1k	5k
wide-angle (28 mm)	0.356	0.358	0.225	0.241
normal (50 mm)	0.964	0.972	0.554	0.828
macro (105 mm)	0.118	0.424	0.0	0.015
telephoto (135 mm)	0.087	0.281	0.009	0.122

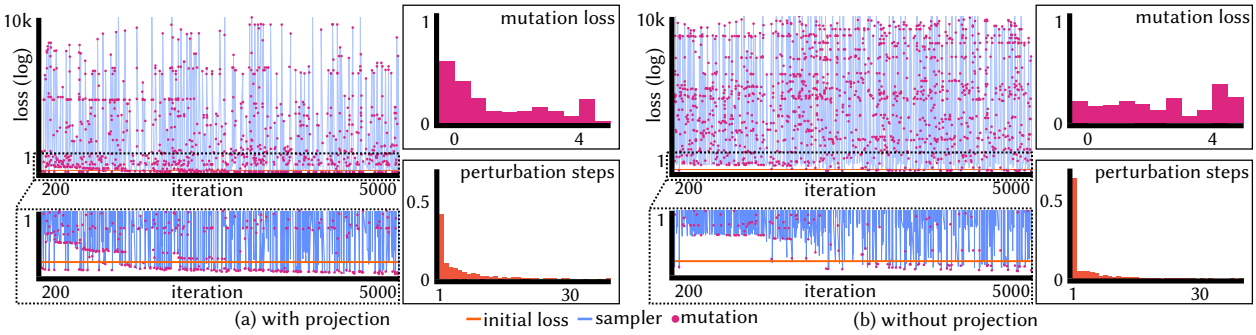


Figure 4.6: Two runs of our method with *(a)* and without *(b)* paraxial projection. We initialize both optimizations with the 135 mm lens and run for 5000 iterations. We also plot the loss of the initial lens *(orange)* for comparison. Using paraxial projection results in better mutations with lower losses, and perturbation sequences with longer durations. As the reservoir saturates with good lenses, both runs more consistently find better lenses. However, paraxial projection reduces this "warm-up" period.

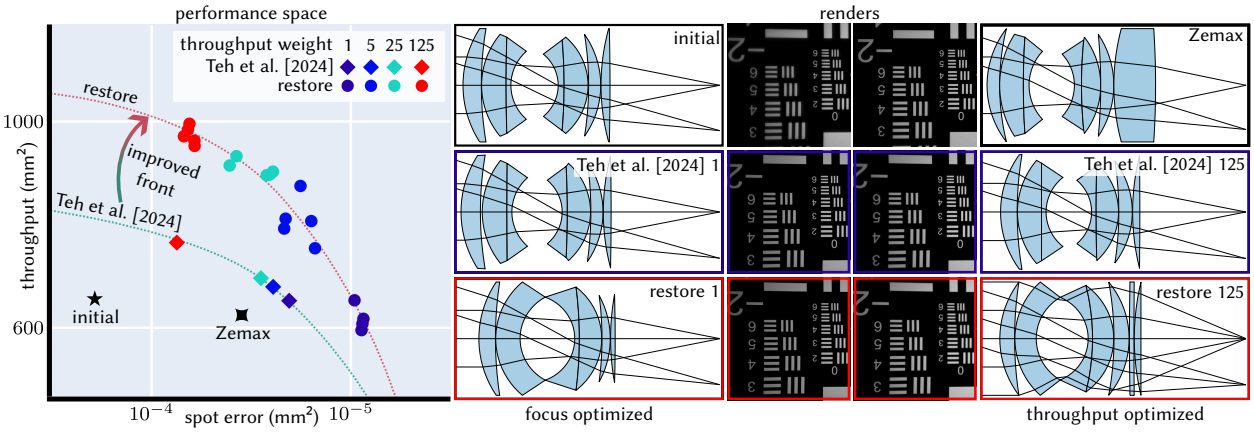


Figure 4.7: By varying the throughput and spot error weights in the optimization loss, we can explore designs that achieve different tradeoffs between lens speed and sharpness, tracing a Pareto front. Without mutations, lens designs are limited to the Pareto front determined by the initial design's topology. As our method can add and remove elements from the design, it is able to explore a larger space of designs and expand the Pareto front to achieve better tradeoffs.

better lenses.

Expanding the lens sharpness-speed Pareto front Figure 4.7 shows that our method allows exploring a richer tradeoff space of lens designs than using just gradient-based optimization of continuous lens parameters. Exploring the design space involves changing the weights of different losses in Equation (4.6), to place different emphasis on lens sharpness versus speed, creating a Pareto front of lens designs. Initialized with lenses from the Pareto front produced using only gradient-based optimization, similar to Teh et al. [94], our method with the same total loss finds designs that move past this front to more favorable parts of the design space. By varying both continuous and discrete lens parameters, our method produces lenses with better overall performance than previous methods.

4.7 Discussion

We presented a method for automated design of compound lenses that, uniquely among related work, can optimize both continuous and discrete parameters of a lens design. It uses Markov chain Monte Carlo sampling to combine gradient-based optimization of continuous parameters, and tailored mutations altering the number of lens elements. Our method uses the RESTORE algorithm for effective sampling, and paraxial projection to improve lens mutations. Our method finds better lenses than when using only gradient-based optimization, and expands the Pareto front possible when considering the lens speed-sharpness tradeoff. We discuss limitations of our method that point towards future research directions.

Other geometric optical elements Our method is currently limited to only spherical lens elements, due to our reliance on the ray transfer matrix for the paraxial projection step. First-order approximations to other types of lens elements (e.g., aspherics, cylindrical lenses, or even reflective elements) do exist, though as their accuracy is lower than for spherical lenses, it remains unclear how effective they would be when designing lens mutations.

Wave optics As we rely on geometric optics for ray tracing and paraxial projection, our method is limited to elements well approximated by geometric optics. Several recent works extend continuous lens design methods using gradient-based optimization to account for wave effects (e.g., diffraction) [17, 36]. Incorporating these works into our method would allow likewise extending mixed continuous-discrete lens design.

Initialization dependence Though our method is able to explore a larger design space than using only gradient-based optimization, it is still sensitive to initialization, requiring high-quality initial lens designs. Sampling such design at random is extremely unlikely, but data-driven methods [21] provide a potential solution to this problem. Such methods can work in conjunction with our method to, e.g., prepopulate the reservoir.

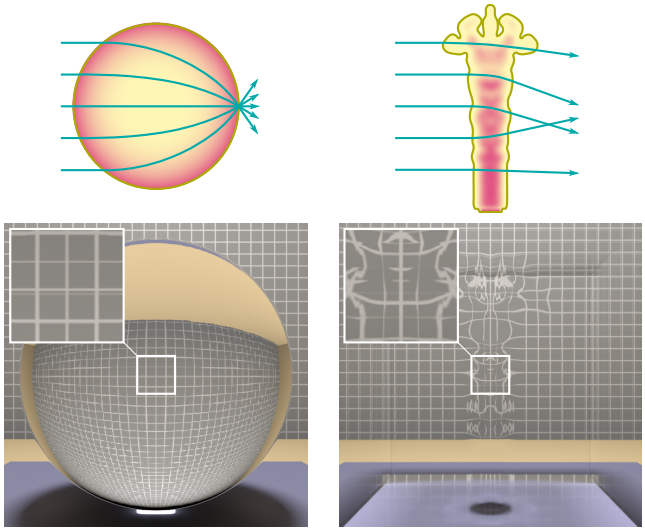
Paraxial projection Our paraxial projection procedure solves a nonconvex polynomial optimization problem with Newton's method, thus is sensitive to initialization and can get stuck in saddle points, though rarely an issue in practice. It is possible to rewrite the optimization problem as a sum-of-squares problem whose solution comes with a certificate of optimality. Directly using sum-of-squares, however, is slow and not practical for our method. Further investigation into optimal solutions to the paraxial projection problem is an interesting direction.

Manufacturing constraints and tolerances Though our method outputs good lenses in simulation, ensuring that these lenses are manufacturable and robust to manufacturing errors is essential for practicality. Our designs satisfy certain manufacturability constraints (e.g., element thickness), but there are other constraints not captured in our method (e.g., element distances). Additionally, our method does not account for manufacturing tolerances to improve robustness. Investigating how to incorporate manufacturing constraints and tolerances is an important future research direction.

Chapter 5

Adjoint Nonlinear Ray Tracing

Up until this point, the thesis has been concerned with spherical lenses, mainly for their simplicity. There are many other types of optical elements one can build that exhibit interesting optical properties. One such type of optical element is a *gradient-index* (GRIN) lens, which is an optical element that has a spatially-varying refractive index field. In this setting, rays of light will *bend continuously* through the medium instead of refracting at discrete interfaces. This chapter presents a method for differentiating GRIN lenses and other continuously-refractive media, which we call *adjoint nonlinear ray tracing* (ANRT). With ANRT, we can calculate the gradient of an objective function with respect to the refractive index field, which readily allows us to integrate these materials into the methods presented in the previous chapters.



(a) Luneburg lens (b) Gas flow

Figure 5.1: Examples of refractive index fields, including a Luneburg lens and a plume of gas. **(a)** The Luneburg lens focuses any incoming light direction to the antipodal point of the incoming direction. **(b)** The gas plume has a spatially-varying lower refractive index than the surrounding air, causing light to bend through the medium. To exaggerate distortion (see inset), we multiply the refractive index of the medium by $100\times$.

GRIN lenses, such as the Luneburg lens shown in Figure 5.1(a) or optical fibers used for telecommunication, use spatially-varying refractive index fields to focus or steer light. The phenomenon of continuously varying index of refraction also occurs in naturally, in the form of mirages and mixing fluids like mixed drinks. Mixing two gases with different refractive index also causes light to distort when passing through, as shown in Figure 5.1(b).

The theory for the geometric and radiometric properties of light paths undergoing continuous refraction is well-established in computer graphics. Given a volume with a known

refractive index field as input, there are many ray tracing procedures to efficiently determine the nonlinear path of light through a medium [1, 41, 86]. This enables the rendering of photorealistic images of these gradient-index fields, which we refer to as nonlinear ray tracing.

We are interested in building a differentiable version of nonlinear ray tracing. One approach is to use reverse-mode automatic differentiation (AD) to calculate gradients with respect to the refractive field. Reverse-mode AD records the computation that occurs during the simulation, then calculates the derivative of the output rays with respect to the refractive index profile. This requires a large amount of memory to store the computation graph, especially when working with 3D volumes. This means that the more accurate the simulation is, the more memory automatic differentiation will require.

We instead present a method for differentiating the dynamics of continuous refraction *without* constructing the computation graph. We employ the adjoint state method to derive a set of ordinary differential equations for calculating the gradient. We then show how to discretize and efficiently simulate these equations. Our technique can optimize for refractive index fields based on two types of objectives: image objectives and geometric objectives. We use our technique to explore multiple types of optimization objectives for a number of applications, including: (i) optimizing the focusing properties of GRIN optical fibers and lenses, (ii) designing novel displays based on refractive index fields, and (iii) reconstructing unknown refractive index fields from a set of images. To ensure reproducibility and facilitate follow-up work, we provide our code on the project website: `http://imaging.cs.cmu.edu/adjoint_nonlinear_tracing/`.

5.1 Related Work

Nonlinear ray tracing Our focus is on *continuously-refractive media*, where the refractive index changes continuously from one location to another. As a consequence of the eikonal equation of geometric optics, light propagating inside such media travels along curved rays, rather than (piecewise-)linear rays [51]. Stam and Languénou [86] use Lagrangian optics to convert the eikonal equation into a second-order differential equation, which they then use to describe and numerically trace these curved rays. We follow Gröller [30] and term this process *nonlinear ray tracing*. Sharma et al. [82] reformulate these equations to make nonlinear ray tracing more numerically stable. Ihrke et al. [41] use nonlinear ray tracing to render images due to light wavefronts propagating through continuously-refractive media. In practice, many such media additionally exhibit volumetric scattering. The combined effects of light curving and scattering can be described using the refractive radiative transfer equation [1], which can be simulated using variants of volumetric path tracing [67] and photon mapping [31] techniques.

Gradient-index optics Unlike conventional refractive optics, GRIN optics work by using a continuously-varying refractive index field to guide light along curved light paths. These optics can produce optical effects and aberration characteristics that are not possible with

conventional refractive optics. Teichman et al. [95] survey GRIN optic designs, their advantages and disadvantages compared to conventional refractive optics, and ways to fabricate different GRIN profiles. As an alternative to fabrication, it is possible to sculpt *virtual* GRIN optics in various media (e.g., water, tissue) using acoustic waves [14, 47, 67, 80, 81].

Two common examples of GRIN optics, for which the refractive index fields are known analytically, are the Luneburg lens [57] and Maxwell fisheye lens [58]. They focus collimated beams or point sources (respectively), regardless of the orientation of the lens, the angle of incidence of the beam, or the location of the point source [51]. Another common example is GRIN waveguides, which can be used in place of conventional optical fibers [25]. Similar to our work, Balasubramanian et al. [4] use differentiable ray tracing to design refractive index fields for new types of GRIN optics. Their technique is based on automatic differentiation, and thus suffers from slow computation and large memory consumption. Our adjoint derivation addresses both shortcomings.

Reconstruction of transparent materials Some transparent materials such as gas clouds are continuously-refractive media. Atcheson et al. [2] develop an algorithm that, given a set of image measurements of a gas cloud, solves for the refractive index everywhere inside the cloud. Their approach assumes that light rays through the cloud are approximately linear, which makes it possible to recover the refractive index field through Poisson integration. Ihrke [40] and Ji et al. [46] both relax the path linearity assumption by iterating between nonlinear path tracing and refractive index estimation. However, these iterative schemes ignore either the exit direction or the exit position when path tracing. In contrast to these prior works, our method can recover the refractive index field even in cases of large deflection. Schröder and Schuster [77] introduce theoretical analysis and an algorithm for reconstructing continuously-refractive media using time-of-flight measurements. Our method can also take advantage of time-of-flight information.

5.2 Theoretical background

We use this section to present the theory of nonlinear ray tracing within media with continuously-varying refractive index, and the adjoint state method for differentiating objectives subject to partial differential equation (PDE) constraints. We refer to Kravtsov and Orlov [51] for a more detailed discussion of nonlinear ray tracing, and to Plessix [69] for the adjoint state method.

5.2.1 Nonlinear ray tracing

Nonlinear ray tracing refers to the geometric optics description of how light propagates in *continuously-refractive media*. These are media where the refractive index η varies continuously from point to point. When traveling from point x_1 to point x_2 inside such a medium $\mathcal{M} \subset \mathbb{R}^3$,

Table 5.1: Definitions of main terms used in the adjoint state method.

symbol	type	description
σ	$[0, \infty)$	Parameterization of time
η	$\mathbb{R}^3 \to [1, \infty)$	Refractive index field
$\nabla\eta$	$\mathbb{R}^3 \to \mathbb{R}^3$	Spatial gradient of refractive index
$\text{Hess}(\eta)$	$\mathbb{R}^3 \to \mathbb{R}^{3\times3}$	Hessian of the refractive index
x	$[0, \infty) \to \mathbb{R}^3$	Position of the light ray
v	$[0, \infty) \to \mathbb{R}^3$	Velocity of the light ray
$\boldsymbol{\lambda}$	$[0, \infty) \to \mathbb{R}^3$	Adjoint state variable of position
$\boldsymbol{\mu}$	$[0, \infty) \to \mathbb{R}^3$	Adjoint state variable of velocity
$\mathcal{C}_i$	$\mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$	Inner cost function
$\mathcal{F}_i$	$\mathbb{R} \to \mathbb{R}$	Outer cost function

light will trace a curved ray that is a stationary point of the *optical Lagrangian*

$$L(R) \equiv \int_R \eta(\mathrm{x}(s))\,\mathrm{d}s, \tag{5.1}$$

where R is any curve contained in $\mathcal{M}$ that starts at x_1 and ends at x_2; and s is the geometric-length (arc-length) parameterization of this ray. Equation (5.1) is the *optical length* of ray R, that is, geometric length weighted by the local refractive index. Therefore, light travels along curved rays that correspond to extrema (local maxima or minima) and saddle points of optical length.[1]

Using the Euler-Lagrange equation for stationarity of the optical Lagrangian of Equation (5.1), we can derive the *ray equation of geometric optics* [10] for light rays inside a continuously-refractive medium:

$$\frac{\mathrm{d}}{\mathrm{d}s}\left(\eta\frac{\mathrm{dx}}{\mathrm{d}s}\right) = \nabla\eta, \tag{5.2}$$

where, to simplify notation, we made the dependence of the refractive index η and its gradient $\nabla\eta$ on the location $\mathrm{x}(s) \in \mathcal{M}$ implicit. Throughout the chapter, we adopt a change of variable proposed by Sharma et al. [82] and defined through the differential relationship

$$\mathrm{d}\sigma \equiv \frac{\mathrm{d}s}{\eta}. \tag{5.3}$$

We term σ the *canonical parameter*, a name we will justify shortly. By reparameterizing light rays in terms of σ, Equation (5.2) becomes

$$\frac{\mathrm{d}}{\mathrm{d}\sigma}\left(\frac{\mathrm{dx}}{\mathrm{d}\sigma}\right) = \eta\nabla\eta. \tag{5.4}$$

[1]The optical length is directly proportional to time, given that $\mathrm{d}t = \eta/c_o\,\mathrm{d}s$, where c_o is the speed of light in vacuum. Therefore, stationary points of the optical Lagrangian of Equation (5.1) also correspond to stationary points of time.

Lastly, by introducing the velocity $\mathrm{v} \in \mathbb{R}^3$, we can separate this second-order ordinary differential equation (ODE) into a system of first-order ODEs known as *Hamilton's equations*,

$$\frac{\mathrm{dx}}{\mathrm{d}\sigma} = \mathrm{v}, \tag{5.5}$$

$$\frac{\mathrm{dv}}{\mathrm{d}\sigma} = \eta \nabla \eta. \tag{5.6}$$

Equations (5.5)-(5.6) are known as the *Newton's law form* or *canonical form* of Hamilton's equations, because of their similarity with Newton's equations of motion in mechanics [51]. We can interpret these equations as a moving particle that is subject to forces equal to $\eta \nabla \eta$. This exact analogy with Newton's equations of motion is a consequence of the use of σ from Equation 5.3 to parameterize light rays, justifying the name canonical parameter. As we discuss in Section 5.4, the use of canonical parameter σ also allows us to discretize and numerically simulate these equations using *reversible* symplectic integrators [32]. The reversibility property will be critical when discretizing our adjoint nonlinear ray-tracing formulation.

5.2.2 Adjoint state method

The adjoint state method allows computing derivatives of optimization objectives subject to constraints in the form of (ordinary or partial) differential equations. Such optimization problems often arise in physics-based inverse problems. The general form of the optimization problem that we will consider is:

$$\begin{aligned} \min_{\theta} \quad & \mathcal{G}(\mathbf{p}) \\ \text{s.t.} \quad & \mathcal{S}(\mathbf{p}; \theta) = 0, \end{aligned} \tag{5.7}$$

where $\mathcal{S}$ is the set of differential equations describing the underlying dynamics, and $\mathcal{G}$ is the cost function of the inverse problem. $\mathbf{p}$ is the configuration, or *state variables*, describing the physics, and θ is the *control variables* of the dynamics. The adjoint state method differentiates the optimization objective, making it possible to use gradient-based optimization techniques to solve optimization problems of the form shown in Equation (5.7).

The adjoint state method proceeds by first converting the constrained optimization problem of Equation (5.7) into an unconstrained optimization problem, through the use of Lagrange multipliers,

$$\min_{\theta, \mathbf{p}, \boldsymbol{\lambda}} \quad \mathcal{L}(\mathbf{p}, \theta, \boldsymbol{\lambda}), \tag{5.8}$$

$$\mathcal{L}(\mathbf{p}, \theta, \boldsymbol{\lambda}) \equiv \mathcal{G}(\mathbf{p}) - \langle \boldsymbol{\lambda}, \mathcal{S}(\mathbf{p}; \theta) \rangle. \tag{5.9}$$

The *Lagrangian* $\mathcal{L}$ augments the original cost function $\mathcal{G}$ with the original constraints, scaled by slack variables $\boldsymbol{\lambda}$ known as the *adjoint state*. The adjoint state has the same dimensionality as the configuration of the dynamics.

The Lagrangian $\mathcal{L}$ will have the same minimum as the original cost function $\mathcal{G}$ when all the variables except for θ are critical points. This is equivalent to finding values $\mathbf{p}^*$ for the configuration and $\boldsymbol{\lambda}^*$ for the adjoint state such that

$$\mathrm{d}_{\mathbf{p}}\mathcal{L}(\mathbf{p}^*, \theta, \boldsymbol{\lambda}^*) = 0, \tag{5.10}$$
$$\mathrm{d}_{\boldsymbol{\lambda}}\mathcal{L}(\mathbf{p}^*, \theta, \boldsymbol{\lambda}^*) = 0. \tag{5.11}$$

Combining these equations with the definition of the Lagrangian in Equation (5.9), we have

$$(\,\mathrm{d}_{\mathbf{p}}\mathcal{S})(\mathbf{p}^*; \theta)\boldsymbol{\lambda}^* - \,\mathrm{d}_{\mathbf{p}}\mathcal{G}(\mathbf{p}^*) = 0, \tag{5.12}$$
$$\mathcal{S}(\mathbf{p}^*; \theta) = 0. \tag{5.13}$$

Equation (5.13) simply requires that we satisfy the constraints of the original optimization problem of Equation (5.7). Equation (5.12) allows us to solve for $\boldsymbol{\lambda}^*$. Lastly, taking the derivative of the Lagrangian with respect to θ, we have

$$\mathrm{d}_{\theta}\mathcal{L}(\mathbf{p}^*, \theta, \boldsymbol{\lambda}^*) = -\boldsymbol{\lambda}^* \,\mathrm{d}_{\theta}\mathcal{S}(\mathbf{p}^*; \theta). \tag{5.14}$$

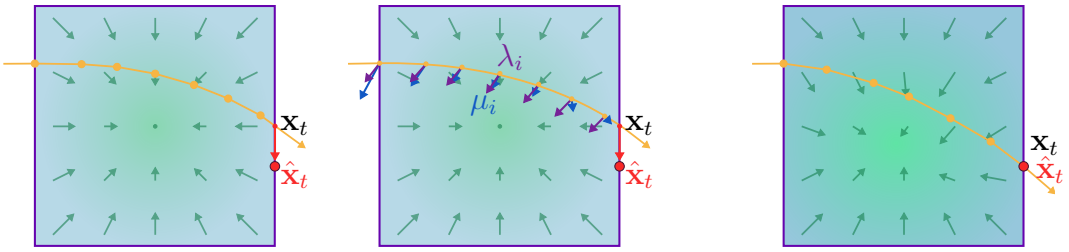
Equation (5.14) equals the derivative of the constrained objective of the optimization problem of Equation (5.7). With this gradient in hand, we can solve this problem using any gradient-based optimization algorithm. Importantly, Equation (5.12) will be a set of differential equations defined by the derivatives of the original differential equations $\mathcal{S}$ with respect to the configuration $\mathbf{p}$. We can use these equations to compute $\boldsymbol{\lambda}^*$ first, and use the result to compute the product $\boldsymbol{\lambda}^* \,\mathrm{d}_{\theta}\mathcal{S}(\mathbf{p}^*; \theta)$ directly, without having to explicitly construct the, typically very high-dimensional, *Jacobian* $\mathrm{d}_{\theta}S(\mathbf{p}^*; \theta)$.

5.3 Differentiating w.r.t. Refractive Index

In this section, we use the adjoint state method to derive an expression for differentiating optimization objectives constrained by Hamilton's equations (5.5)-(5.6). Concretely, we are concerned with optimization problems of the form:

$$\begin{aligned}
\min_{\eta} \sum_{i=1}^{N} \mathcal{F}_i\Bigg[\iint_{(\mathrm{x}_0, \mathrm{v}_0)\in\Omega} \mathcal{C}_i(\mathrm{x}\,(\sigma_f; \eta, \mathrm{x}_0, \mathrm{v}_0)\,, \mathrm{v}\,(\sigma_f; \eta, \mathrm{x}_0, \mathrm{v}_0))\mathrm{d}\mathrm{x}_0\,\mathrm{d}\mathrm{v}_0 \Bigg] \\
\text{s.t.}\quad \dot{\mathrm{x}}(\sigma; \eta, \mathrm{x}_0, \mathrm{v}_0) = \mathrm{v}, \quad \forall\sigma \in [0, \sigma_f], \\
\dot{\mathrm{v}}(\sigma; \eta, \mathrm{x}_0, \mathrm{v}_0) = \eta\nabla\eta, \quad \forall\sigma \in [0, \sigma_f], \\
\mathrm{x}(0; \eta, \mathrm{x}_0, \mathrm{v}_0) = \mathrm{x}_0, \\
\mathrm{v}(0; \eta, \mathrm{x}_0, \mathrm{v}_0) = \mathrm{v}_0,
\end{aligned} \tag{5.15}$$

where the dot refers to differentiation with respect to the canonical parameter σ; Ω is the set of possible initial conditions for light rays (e.g., their location and velocity on a light source); σ_f parameterizes the end of the ray (e.g., when it exits the medium or reaches a sensor); $\mathcal{C}_i$



(a) Forward tracing (b) Backward tracing (c) Updated refractive index field

Figure 5.2: The adjoint tracing procedure involves three key steps. **(a)** First, the ray is traced forward, where each vertex is an integration step in the simulation. An error value is computed, e.g., the distance between the terminal position x_t and a target position $\hat{\mathrm{x}}_t$. **(b)** This error is then backward traced through the volume to compute the optimization gradient. Because of the reversibility property of the forward tracing procedure, we can retrace the same set of vertices *without* actually storing them—which would otherwise be extremely memory demanding. **(c)** Finally, the refractive index field η is updated using the gradient. This cycle is repeated until convergence. Each point on the trajectory of the ray has an associated x, v, $\boldsymbol{\lambda}$, and $\boldsymbol{\mu}$.

is any function of the location and velocity at the end of the ray; and $\mathcal{F}_i$ is any function of integrals over multiple rays. For simplicity, we will generally use x and v in place of the full $\mathrm{x}(\sigma;\eta,\mathrm{x}_0,\mathrm{v}_0)$ and $\mathrm{v}(\sigma;\eta,\mathrm{x}_0,\mathrm{v}_0)$. Moreover, we will use $\mathrm{x}(\sigma_f)$ and $\mathrm{v}(\sigma_f)$ when referring to the ray's end position and velocity, respectively.

In the simplest case of the optimization problem of Equation (5.15), rays travel for some fixed amount of time σ_f, specified as part of the problem description. In practice, this time may be defined implicitly, e.g., as the time when the ray exits the medium or crosses through some specific surface inside the medium. In this case, σ_f is itself a function of the unknown refractive index field η. In both cases, differentiation reduces to solving the same pair of ordinary differential equations (Equations (5.19)-(5.20)), but with different boundary conditions. Thus, for simplicity, our derivation here assumes that σ_f is fixed and independent of η, and we show a derivation for the more general case in the supplement.

Our formulation will allow us to compute the derivative of optimization objectives such as Equation (5.15) with respect to refractive index, assuming derivatives of functions $\mathcal{F}_i$ and $\mathcal{C}_i$ are available (e.g., through analytic or automatic differentiation). We explore two main types of losses as specializations of the optimization problem of Equation (5.15): image losses and geometric losses.

For image losses, the optimization problem of Equation (5.15) searches for a refractive index field η such that rendered radiometric measurements through that field match target measurements by a sensor. In this case, $\mathcal{C}_i$ becomes the path contribution function; the integral corresponds to a radiometric path integral expression [97]; and $\mathcal{F}_i$ is a loss function that compares the rendered and target radiometric measurements. An example image loss is:

$$\mathcal{F}_i = \left\| \hat{I}_i - \iint_{(\mathrm{x}_0,\mathrm{v}_0)\in\Omega} W_{e,i}(\mathrm{x}(\sigma_f),\mathrm{v}(\sigma_f)) L_e(\mathrm{x}_0,\mathrm{v}_0) \mathrm{d}\mathrm{x}_0 \mathrm{d}\mathrm{v}_0 \right\|^2. \tag{5.16}$$

In this example, $\mathcal{C}_i$ equals the product of the sensitivity function $W_{e,i}$ of the i-th sensor and

the source emission function L_e, and $\mathcal{F}_i$ equals the $\mathcal{L}_2$ loss between the rendered measurement and the actual measurement $\hat{I}_i$ at that sensor. Summing over all N sensors (e.g., all pixels of an image) completes the loss.

For geometric losses, the optimization problem of Equation (5.15) searches for a refractive index field η such that rays traced through the field have end conditions satisfying specified properties. In this case, $\mathcal{C}_i$ becomes a loss function that compares the end conditions of the ray against the desired properties; the integral accumulates this loss for all rays; $N = 1$; and $\mathcal{F}$ is the identity function. An example of a geometric loss is:

$$\mathcal{C}_i = \|\mathrm{x}(\sigma_f) - \hat{\mathrm{x}}\|^2, \tag{5.17}$$

where $\mathcal{C}$ is the $\mathcal{L}_2$ loss between the ray's end position $\mathrm{x}(\sigma_f)$ and a target position $\hat{\mathrm{x}}$ where we want all rays to arrive.

Adjoint nonlinear ray tracing We now apply the adjoint state method to compute the derivative of the optimization objective of Equation (5.15). For simplicity, we differentiate the function inside the integral of Equation (5.15), which we denote as $\mathcal{C}$. The total derivative additionally requires the term $\frac{\mathrm{d}\mathcal{F}}{\mathrm{d}\mathcal{C}}$, which is easy to compute.

Using the terminology of Section 5.2.2, the configuration variables are $\mathbf{p} \equiv (\mathrm{x}, \mathrm{v})$, and the control variable is $\theta \equiv \eta$. Therefore, we introduce a pair of adjoint state variables $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ that have the same dimensionality as the corresponding configuration variables. We can then form the Lagrangian as:

$$\begin{aligned}\mathcal{L} = \mathcal{C}(\mathrm{x}(\sigma_f), \mathrm{v}(\sigma_f)) &- \int_0^{\sigma_f} \boldsymbol{\lambda}^\top (\dot{\mathrm{x}} - \mathrm{v})\, \mathrm{d}\sigma \\ &- \int_0^{\sigma_f} \boldsymbol{\mu}^\top (\dot{\mathrm{v}} - \eta \nabla \eta)\, \mathrm{d}\sigma.\end{aligned} \tag{5.18}$$

Solving for the critical points of the Lagrangian with respect to $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, we obtain Hamilton's Equations (5.5)-(5.6) with initial conditions $\mathrm{x}_0, \mathrm{v}_0$—that is, the constraints of the optimization problem of Equation (5.15), as expected. Solving for the critical points with respect to x and v, we have

$$\dot{\boldsymbol{\lambda}} = -\Big(\nabla\eta(\nabla\eta)^\top + \eta\,\mathrm{Hess}(\eta)\Big)\boldsymbol{\mu}, \qquad \forall\sigma \in [0, \sigma_f] \tag{5.19}$$

$$\dot{\boldsymbol{\mu}} = -\boldsymbol{\lambda}, \qquad \forall\sigma \in [0, \sigma_f] \tag{5.20}$$

$$\boldsymbol{\lambda}(\sigma_f) = \frac{\partial \mathcal{C}}{\partial \mathrm{x}}, \tag{5.21}$$

$$\boldsymbol{\mu}(\sigma_f) = \frac{\partial \mathcal{C}}{\partial \mathrm{v}}. \tag{5.22}$$

We provide the details of this derivation in the supplement. We make the following observations. First, Equations (5.19)-(5.20) are a system of first-order ODEs on the adjoint state variables, with boundary conditions specified by Equations (5.21)-(5.22) at the

propagation *end*, σ_f. We term Equations (5.19)-(5.20) the *adjoint equations*. Second, computing the boundary conditions in Equations (5.21)-(5.22) requires knowing the ray's end position $\mathrm{x}(\sigma_f)$ and velocity $\mathrm{v}(\sigma_f)$. Third, even though the ODEs in Equations (5.19)-(5.20) evolve only the adjoint state variables $\boldsymbol{\lambda}, \boldsymbol{\mu}$, they require evaluating the refractive index η and its derivatives at all intermediate ray locations $\mathrm{x}(\sigma)$, $\forall \sigma \in [0, \sigma_f]$.

These observations suggest the following two-stage procedure for computing the adjoint state variables. At the first stage, we evolve Hamilton's Equations (5.5)-(5.6) with initial conditions $\mathrm{x}_0, \mathrm{v}_0$ *forward* in σ, until propagation ends at σ_f. At the second stage, we first use the ray's end position and velocity to compute the initial conditions $\boldsymbol{\mu}(\sigma_f), \boldsymbol{\lambda}(\sigma_f)$ of Equations (5.21)-(5.22). We then evolve the adjoint Equations (5.19)-(5.20) with initial conditions $\boldsymbol{\mu}(\sigma_f), \boldsymbol{\lambda}(\sigma_f)$ *backward* in σ, for which we travel in reverse along the ray we traced during the first stage. We refer to the second stage procedure as *backward tracing*. In Section 5.4, we leverage the special structure of Hamilton's equations and the adjoint equations, and devise discrete numerical procedures for efficiently implementing this two-stage procedure, without the need to store the trajectory of the ray.

Once we have computed the adjoint state variables, we can compute the gradient of the objective of Equation (5.15) as

$$\mathrm{d}_\eta \mathcal{L} = \int_0^{\sigma_f} (\eta \nabla(\,\mathrm{d}\eta) + \,\mathrm{d}\eta \nabla \eta)^\top \boldsymbol{\mu}\,\mathrm{d}\sigma. \tag{5.23}$$

Figure 5.2 visualizes the steps of the overall procedure. The differential term $\mathrm{d}\eta$ in Equation (5.23) will depend on the spatial representation of the refractive index field. We assume that we compute the refractive index using a function $\mathcal{N}$ with parameters θ, $\eta(\mathrm{x}) = \mathcal{N}(\mathrm{x}; \theta)$. Then, we can replace $\mathrm{d}\eta(\mathrm{x}) = \frac{\mathrm{d}\mathcal{N}(\mathrm{x};\theta)}{\mathrm{d}\theta}\,\mathrm{d}\theta$. We note that Equation (5.23) requires also computing spatial gradients of $\mathrm{d}\eta$, i.e., the derivatives of the underlying refractive index field representation. Thus, we can use Equation (5.23) to compute derivatives of the objective of Equation (5.15) with respect to the parameters of any representation of the refractive index field that: (i) supports point, gradient (for Equations (5.6) and (5.20)), and Hessian (for Equation (5.20)) queries; and (ii) has point and gradient queries that are differentiable with respect to the representation parameters θ. We refer to the supplement for the derivation of these equations for the case where $\mathcal{N}$ corresponds to trilinear interpolation. Other representations that satisfy these requirements include smooth interpolation schemes (linear, spline, and so on), different grid types, and neural fields [106].

5.4 Discretization of the adjoint equations

We now discuss how to numerically implement the two-stage procedure we derived in Section 5.3 for differentiating the optimization objective of Equation (5.15) with respect to the refractive index field.

Algorithm 2 ForwardTrace$(\eta, \mathrm{x}_0, \mathrm{v}_0, \Delta\sigma)$

Input: *A refractive field* η*, initial position* x_0*, initial direction* v_0*, step size* $\Delta\sigma$*.*
Output: *A new position* x_f*, a new direction* v*.*

▷ *Initialize parameters*
$\mathrm{x} \leftarrow \mathrm{x}_0$
$\mathrm{v} \leftarrow \mathrm{v}_0$
while insideVolume(x) **do**
    $\eta, \nabla\eta \leftarrow \text{interpolate}(\eta, \mathrm{x})$
    $\mathrm{v} \leftarrow \mathrm{v} + \eta \cdot \nabla\eta \cdot \Delta\sigma$
    $\mathrm{x} \leftarrow \mathrm{x} + \mathrm{v} \cdot \Delta\sigma$
$\mathrm{x}_f \leftarrow \mathrm{x}$
$\mathrm{v}_f \leftarrow \mathrm{v}$
▷ *Return new position and direction*
return $\mathrm{x}_f, \mathrm{v}_f$

Forward tracing During the first stage of our procedure, we need to evolve Hamilton's Equations (5.5)-(5.6) forward in σ. We choose to use a *symplectic and reversible integrator* to perform this numerical integration. When applied to a Hamiltonian system of ODEs, symplectic integrators have well-documented stability properties that help keep discretization error bounded even along very long integration trajectories. Additionally, reversible integrators will be important during the second stage of our procedure when we perform backward tracing, as we discuss later in this section. Many popular symplectic integrators are also reversible. For more information on symplectic and reversible integrators, we refer to Hairer et al. [32] and Kharevych et al. [48].

For our experiments, we use the symplectic Euler integrator, though we can easily change to other symplectic integration schemes (e.g., the leapfrog integrator [62, 67]). Applying this integrator to Hamilton's Equations (5.5)-(5.6) results in the following discretized evolution equations:

$$\mathrm{x}_i = \mathrm{x}_{i-1} + \mathrm{v}_i \Delta\sigma, \tag{5.24}$$
$$\mathrm{v}_i = \mathrm{v}_{i-1} + \eta(\mathrm{x}_{i-1}) \nabla\eta(\mathrm{x}_{i-1}) \Delta\sigma. \tag{5.25}$$

This integration scheme is *explicit*: the evolution equations compute the values of x and v at the i-th step using only their values and the values of the refractive index field η at the previous $(i-1)$-th step, and thus without the need for a nonlinear solver. Algorithm 2 summarizes our procedure for the forward tracing stage.

Backward tracing During the second stage of our procedure, we need to evolve the adjoint Equations (5.19)-(5.20) backward in σ, along with retracing in reverse direction the ray we traced during the first stage. For this, we need to: first discretize the adjoint equations, as we did for Hamilton's equations in Equations (5.24)-(5.25); and second, evaluate both sets of discrete equations in the backward σ direction.

Algorithm 3 BACKWARDTRACE$(\eta, \mathrm{x}_0, \mathrm{v}_0, \Delta\sigma)$

Input: *A refractive field* η*, initial position* x_0*, initial velocity* v_0*, position gradient* dx*, direction gradient* dv*, step size* $\Delta\sigma$*.*

Output: *A gradient w.r.t parameters,* $\mathrm{d}\theta$*.*

▷ *Initialize parameters*
$\mathrm{x} \leftarrow \mathrm{x}_f$
$\mathrm{v} \leftarrow \mathrm{v}_f$
$\boldsymbol{\mu} \leftarrow \mathrm{dv}$
$\boldsymbol{\lambda} \leftarrow \mathrm{dx} + \mathrm{dv} \cdot \Delta\sigma$
$\mathrm{d}\theta \leftarrow 0$
while insideVolume(x) **do**
    ▷ *Backward tracing*
    $\mathrm{x} \leftarrow \mathrm{x} - \mathrm{v} \cdot \Delta\sigma$
    $\eta, \nabla\eta \leftarrow \mathrm{interpolate}(\eta, x)$
    $\mathrm{v} \leftarrow \mathrm{v} - \eta \cdot \nabla\eta \cdot \Delta\sigma$
    ▷ *Accumulate the gradients in the adjoint variables*
    $nxx \leftarrow \mathrm{Hess}(\eta, x)$
    $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \left(\nabla\eta \cdot \nabla\eta^\top + \eta \cdot nxx\right) \boldsymbol{\mu}\Delta\sigma$
    $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \boldsymbol{\lambda}\Delta\sigma$
    $\delta\theta \leftarrow \delta\theta + (\boldsymbol{\mu} \cdot \nabla\eta)\frac{\mathrm{d}\mathcal{N}}{\mathrm{d}\theta} + \eta\left(\boldsymbol{\mu} \cdot \nabla\frac{\mathrm{d}\mathcal{N}}{\mathrm{d}\theta}\right)$
▷ *Return the accumulated gradient*
return $\mathrm{d}\theta$

The adjoint equations share the same symplectic structure as Hamilton's equations. Therefore, we can discretize them using the same symplectic Euler integrator we used for Hamilton's equations:

$$\begin{aligned}\boldsymbol{\lambda}_i = \boldsymbol{\lambda}_{i-1} - \big(&\nabla\eta(\mathrm{x}_{i-1})(\nabla\eta(\mathrm{x}_{i-1}))^\top \\ &+ \eta(\mathrm{x}_{i-1})\,\mathrm{Hess}(\eta(\mathrm{x}_{i-1}))\big)\boldsymbol{\mu}_i\Delta\sigma. \qquad (5.26)\end{aligned}$$

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i-1} - \boldsymbol{\lambda}_{i-1}\Delta\sigma, \qquad (5.27)$$

Then, we can invert these discrete relationships to evolve both Hamilton's equations and the adjoint equations in reverse, as required during the backward tracing stage:

$$\mathrm{x}_{i-1} = \mathrm{x}_i - \mathrm{v}_i\Delta\sigma, \qquad (5.28)$$

$$\mathrm{v}_{i-1} = \mathrm{v}_i - \eta(\mathrm{x}_{i-1})\nabla\eta(\mathrm{x}_{i-1})\Delta\sigma, \qquad (5.29)$$

$$\begin{aligned}\boldsymbol{\lambda}_{i-1} = \boldsymbol{\lambda}_i &\\ + \Big(&\nabla\eta(\mathrm{x}_{i-1})(\nabla\eta(\mathrm{x}_{i-1}))^\top + \eta(\mathrm{x}_{i-1})\,\mathrm{Hess}(\eta(\mathrm{x}_{i-1}))\Big)\boldsymbol{\mu}_i\Delta\sigma, \qquad (5.30)\end{aligned}$$

$$\boldsymbol{\mu}_{i-1} = \boldsymbol{\mu}_i + \boldsymbol{\lambda}_{i-1}\Delta\sigma. \qquad (5.31)$$

Equations (5.28)-(5.31) are simply rearranged versions of the forward discretized evolution Equations (5.24)-(5.27), respectively. The resulting numerical scheme is also *explicit*: the backward evolution equations update all quantities at the $(i-1)$-th step using only their values and the values of the refractive index field η at the (i)-th step. Algorithm 3 summarizes our backward tracing procedure.

We make two observations. First, we note that Equations (5.28)-(5.29) will query the exact same locations x and refractive index values η as during forward tracing (up to numerical precision error). Therefore, we do *not* need to store the ray locations traced during forward tracing. We only need to use the final position and velocity $(\mathrm{x}(\sigma_f), \mathrm{v}(\sigma_f))$ at the end of the forward tracing stage. Experimentally, we found that the relative difference between points on the (discretized) paths produced by forward and backward tracing is on the order of 10^{-6}. As a result, the memory use of our combined forward tracing and backward tracing algorithms is constant, instead of scaling linearly with step-size and number of steps, as in conventional reverse-mode AD.

Second, we note that the explicit and exact reversibility properties of our two-stage procedure are due to two important choices we made in our formulation. The first choice is our use of a reversible symplectic integrator. Using a non-reversible integrator (e.g., Euler scheme) would result in the locations x (and thus queried refractive index values η) during backward tracing being different from those during forward tracing. In turn, this difference would result in biased gradient estimates.[2] The second choice is our use of the canonical parameterization σ and the ensuing Newton's law form of Hamilton's equations (5.5)-(5.6). Using a different parameterization of Hamilton's equations (e.g., arc-length s, as in Ihrke et al. [41]) would result—even with a reversible integrator—in *implicit* backward equations, requiring expensive root finding procedures (e.g., Newton's method) to evolve. This is even though the corresponding continuous Hamilton's and adjoint equations are reversible. We demonstrate this in the supplement.

Step size During both the forward and backward tracing stages, it is important to select an appropriate step size $\Delta\sigma$ for integration. Decreasing the step size increases the accuracy of the integration, at the cost of increased computation. In particular, in the case of our two-stage procedure, halving the step size would increase tracing computation by roughly $2\times$, given the need to trace the ray twice, forward and backward. A procedure based on reverse-mode AD would have a similar increase in computation, given the need to forward-trace a longer path and then parse a larger computational graph. However, such a procedure based on reverse-mode AD would have *additionally* increased memory requirements, given the need to store this larger computational graph. By contrast, our two-stage procedure has constant memory requirements. This highlights an important advantage of our two-stage procedure, especially when optimizing refractive index fields at higher resolutions (which requires smaller step sizes for accurate ray tracing).

[2]By "biased", we mean that the gradients computed by our procedure would not match those computed using automatic differentiation on the loss evaluation routine, which involves forward tracing.

5.5 Experiments

We compare the performance of our technique with other differentiable rendering alternatives, in terms of both memory use and computational efficiency. Additionally, we show experiments using different cost functions and refractive index field representations, to show the diversity of applications of our framework. We show results for designing displays, optimizing GRIN optics, and reconstructing different types of objects. In the supplement, we show additional results that validate the accuracy of our computed gradients, and demonstrate the importance of the reversibility properties we discuss in Section 5.4.

Implementation details We have created two implementations of our two-stage forward and backward tracing procedure: one in Pytorch [66], and another in C++ using the Enoki library [42]. We use the two implementations to compare against reverse-mode AD, as implemented both by Pytorch's autograd and in Enoki. For our design and reconstruction experiments, we use the C++ implementation, combined with Pytorch for its gradient-based optimizers and visualization tools.

For all our results, we use the Adam optimizer [49], and initialize the refractive index field to be $\eta(\mathrm{x}) = 1$ everywhere. To ensure that the recovered reconstruction is physically plausible, after every gradient descent iteration, we project η to be greater than or equal to 1 (projected gradient descent). We also use a multiresolution approach to accelerate optimization convergence: During optimization, we periodically double the resolution of the volume we use to represent the refractive index field. We select the step size used for tracing to always be smaller than the width of a voxel in the volume, meaning that we periodically decrease the step size during optimization. We also impose a constraint that the volume boundary has a refractive index of 1, by clamping the values at the boundary at each iteration. We run all of our experiments on an NVIDIA RTX 3090 GPU, with runtimes ranging between 10-40 minutes. We use the Mitsuba renderer with support for continuously-refractive media [67] for rendering visualizations of the results.

We note that, because of the finite step size used for tracing, a traced ray will end at some point past the volume boundary. We deal with this by tracing the ray back to the boundary to compute the boundary conditions for the adjoint equations, then begin the backward tracing stage from the actual post-boundary end location.

Performance We compare the computational efficiency and performance requirements of our technique, against the reverse-mode AD implementations of Pytorch and Enoki. In terms of memory, our method requires the initial and final positions and velocities of the ray. It also needs to keep track of a second volume that maintains the refractive index gradients $\delta\eta$. By contrast, reverse-mode AD additionally needs to store the computation graph, which grows with the number of integration steps taken during the simulation. This means that reverse-mode AD has a linear memory complexity with respect to step size, whereas the memory complexity of our method is constant.

To demonstrate these advantages, we perform quantitative comparisons of our C++

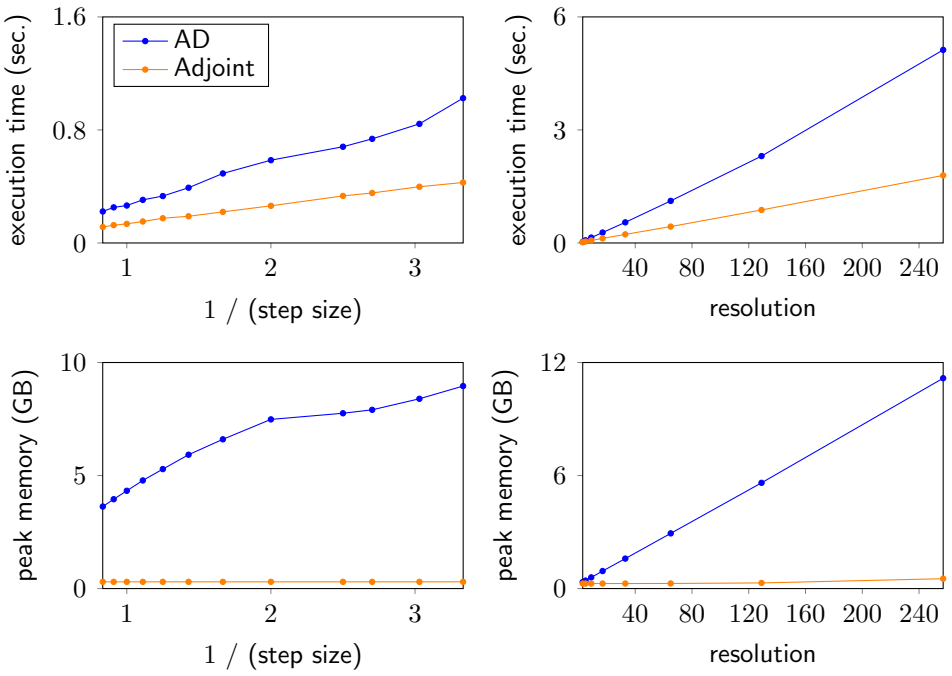


Figure 5.3: Runtime and memory use comparison for reverse-mode AD and the adjoint method. The number of steps along a ray is directly proportional to resolution and inversely proportional to step size. For the adjoint method, memory use is constant and runtime increases linearly as a function of the number of steps along a ray. By contrast, memory usage and runtime for reverse-mode AD are both significantly higher. This is because reverse-mode AD requires keeping track of the entire light path to compute gradients.

implementation against one that uses Enoki's reverse-mode AD. In the supplement, we show additional comparisons of our Pytorch implementation against Pytorch's reverse-mode AD. For our comparisons, we look at runtime and peak memory usage of performing forward and backward tracing operations, as a function of increasing volume resolution and decreasing step size.

Figure 5.3 shows the comparison results. As expected, our method has constant memory usage with respect to step size, whereas reverse-mode AD has a linear dependence on step size. Increasing the volume resolution also affects reverse-mode AD significantly more than our method. As resolution increases, the memory footprint of the volume increases by N^3. However, the step size needs to decrease along with the increase in resolution so that traced rays sample the volume voxels properly. The decrease in step size dominates the memory resources more so than the cubic increase in the volume size, which is why we see a linear dependence on resolution for reverse-mode AD as well.

Our method also performs better in terms of runtime compared to reverse-mode AD. Theoretically, our method has the same asymptotic complexity as reverse-mode AD, as traversal of the computation graph (for reverse-mode AD) and the backward tracing procedure (for our method) will both take as long as the forward tracing procedure. We attribute the better runtime observed for our method in practice to the much larger number of memory accesses that reverse-mode AD needs to perform.

We note that Enoki can perform graph simplification during automatic differentiation, which gives better memory performance than standard reverse-mode AD (so called, hybrid-mode automatic differentiation [29]). In our comparisons with Enoki, we keep graph simplification turned on. However, our workload requires use of the gather operation to query the spatial volume, which prevents Enoki from using certain aggressive graph simplification techniques.

Novel view displays We present three experiments for designing refractive index fields that produce different displays. The first experiments is generating a multiview display that shows two images in two different directions. The second is building a multifocal display that produces images at different focusing distances with accurate defocus blur. The third is a caustic design problem where our method produces a refractive index field that generates a caustic pattern in both the near and far fields.

For the multiview display, we replicate the experimental setting presented in Nimier-David et al. [62], where the task is to generate a refractive index volume that produces two different images simultaneously using two perpendicular light sources. The sources are collimated beams. We use an image loss in this case where, using the notation of Equation (5.15), the outer and inner cost functions are

$$\mathcal{F}_i^{\text{multiview}}(a) \equiv \left\| a - \hat{I}_i \right\|^2, \tag{5.32}$$

$$\mathcal{C}_i^{\text{multiview}}(\mathrm{x}, \mathrm{v}) \equiv I_i(\mathrm{x}, \mathrm{v}), \tag{5.33}$$

and summation is over the two views.

Figure 5.5: Effect of volume resolution. We optimize volumes of different resolution to reproduce a picture of Albert Einstein, under the same setting as in Figure 5.4. As the volume resolution increases, so does reproduction accuracy. The image is courtesy of Yousuf Karsh. ©Yousuf Karsh.

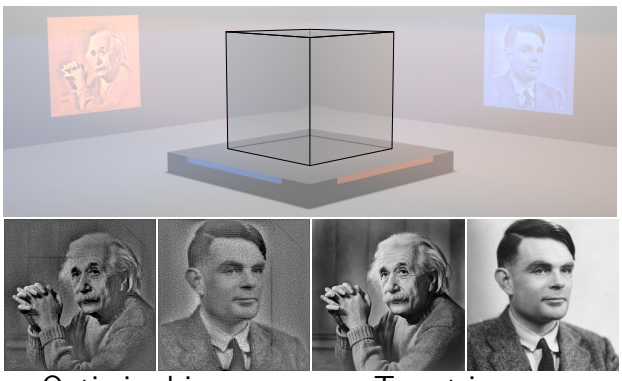


Figure 5.4: The optimized GRIN lens displaying two images. **(Top)** Two collimated beams of light (red and blue) simultaneously illuminate two faces of a cubic GRIN lens, which steers the light to form two distinct images on a wall. **(Bottom)** The optimized images of Albert Einstein and Alan Turing, and the corresponding target images. The image of Albert Einstein is a portrait by Yousuf Karsh. ©Yousuf Karsh. The portrait of Alan Turing is by Elliot & Fry Studio. ©National Portrait Gallery.

Figure 5.4 shows the results. Our method allows us to optimize a higher-resolution volume than that used by Nimier-David et al. [62] (256^3 versus 150^3 voxels; trying to use 256^3 with the reverse-mode AD implementation in a single pass resulted in an "out-of-memory" error). We note that our experiments produce different results than those in Nimier-David et al. [62], due to differences in the experimental setup.

Thanks to its constant memory complexity, our method enables optimizing refractive index fields of higher resolution than what is possible using reverse-mode AD. To demonstrate the importance of this capability, in Figure 5.5, we repeat the experiment of Figure 5.4 using refractive index volumes of different resolutions. Increasing the volume resolution helps ameliorate discretization artifacts due to trilinear interpolation, which in turn results in more accurate reproduction of the target image.

For the multifocal display, we optimize a refractive index field that can generate a focal stack of some input scene. We can simulate the focal stack by placing image planes at different distances from the refractive index volume, as shown in Figure 5.6. The distance of each plane then corresponds to projecting an image at a different focusing distance, with appropriate defocus blur. We select focusing distances equally spaced in diopter space. We then run optimization with an image loss where, using the notation of Equation (5.15), the

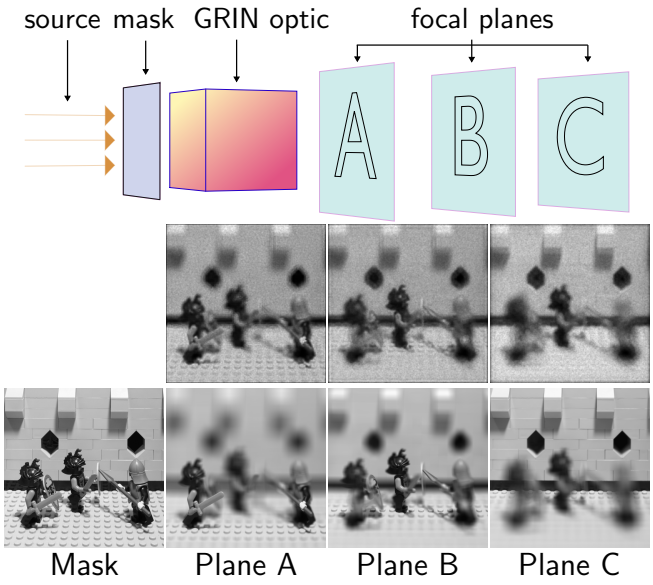


Figure 5.6: A multifocal display. **(Row 1)** Collimated light passes through a mask to form an all-in-focus projected image of a lego scene. Placing a GRIN lens in front of the mask produces a 2D intensity distribution that can change as a function of distance. We optimize this GRIN lens to create a focal stack of this lego scene. **(Row 2)** The optimized intensity distribution at different plane positions, where plane A is the closest to the GRIN lens and plane C is the furthest. **(Row 3)** The target (ground truth) focal stack.

outer and inner cost functions are

$$\mathcal{F}_i^{\text{multifocal}}(a) \equiv \left\| a - \hat{I}_i \right\|^2, \tag{5.34}$$

$$\mathcal{C}_i^{\text{multifocal}}(\mathrm{x}, \mathrm{v}) \equiv I_i(\mathrm{x}, \mathrm{v}). \tag{5.35}$$

and summation is over the different focusing distances.

Figure 5.6 shows the results of the optimization. We form the target images by focusing the Lego knights lightfield from the Stanford light field dataset at different focusing distances. Our optimized refractive index fields can produce images that replicate the defocus blur effects in the input images.

Lastly, we design a refractive index field that generates circular caustics in both the near and far fields. [3] Instead of an image loss, for this caustic design, we use a geometric loss that encourages rays to land at a particular locus of points and have a particular direction. The use of such a loss function showcases the ability of our method to optimize both image and geometric objectives. To define the loss, we use the signed distance function of the target caustic patterns at the near and far field planes. In the notation of Equation (5.15), our loss

[3] By "far field" we mean the image we would obtain if we placed a sensor at a plane placed at the infinity focus of a lens. The location of rays on this plane is determined by their velocity v rather than their location x.

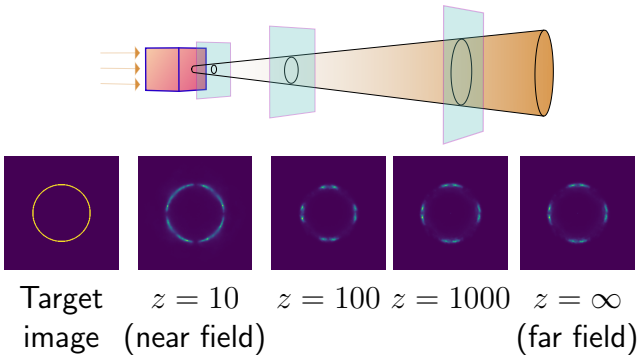


Figure 5.7: Caustic design. The caustic pattern remains in shape as the sensor moves away from the volume. Since the cost function does not promote uniform energy distribution, the caustic contains bright spots.

corresponds to using the outer and inner cost functions

$$\mathcal{F}^{\text{caustic}}(a) \equiv a, \tag{5.36}$$

$$\mathcal{C}^{\text{caustic}}(\mathrm{x}, \mathrm{v}) \equiv (\text{SDF}_{\text{near}}(\mathrm{x}))^2 + (\text{SDF}_{\text{far}}(\mathrm{v}))^2. \tag{5.37}$$

To compute the loss, rays are traced until they reach the end of the volume. There, the near-field part of the loss penalizes large distances (as measured by the SDF) of the ray end-location x from the target caustic. Likewise, the far-field part of the loss penalizes large distances of the ray end-velocity v from the target caustic.

Figure 5.7 shows the results. We note that the optimized refractive index volume successfully reproduces the target caustic in both near-field and far-field. However, some parts of the caustic are a lot brighter than other parts. This is because the loss function we use encourages rays to move toward the caustic, but *not* to spread uniformly along it. Our focus in this experiment is to show that our technique can optimize geometric losses, rather than to find the best loss for producing uniformly-illuminated caustics. In fact, the general problem of defining a transport map between the source and the target images is an active area of research [60, 79, 104]; our optimization procedure could be used with a geometric loss, to design refractive index field realizing such a transport map.

Optimizing GRIN optics We show experiments where we recover the refractive index field of a known GRIN lens, using only a description of the operation of the lens (i.e., the geometric mapping between incident and outgoing rays that the lens implements). We do this for the Luneburg and Maxwell lenses. The Luneburg lens [57] is a GRIN lens that focuses a point source at infinity to the antipodal point on the lens; whereas the Maxwell lens [58] focuses a point source placed at the surface of the lens to the antipodal point on the lens. These perfect focusing properties are realized using the radial refractive index profiles

$$\eta_{\text{Luneburg}}(r) \equiv \sqrt{2 - \left(\frac{r}{R}\right)^2}, \tag{5.38}$$

$$\eta_{\text{Maxwell}}(r) \equiv \frac{2}{1 + \left(\frac{r}{R}\right)^2}, \tag{5.39}$$

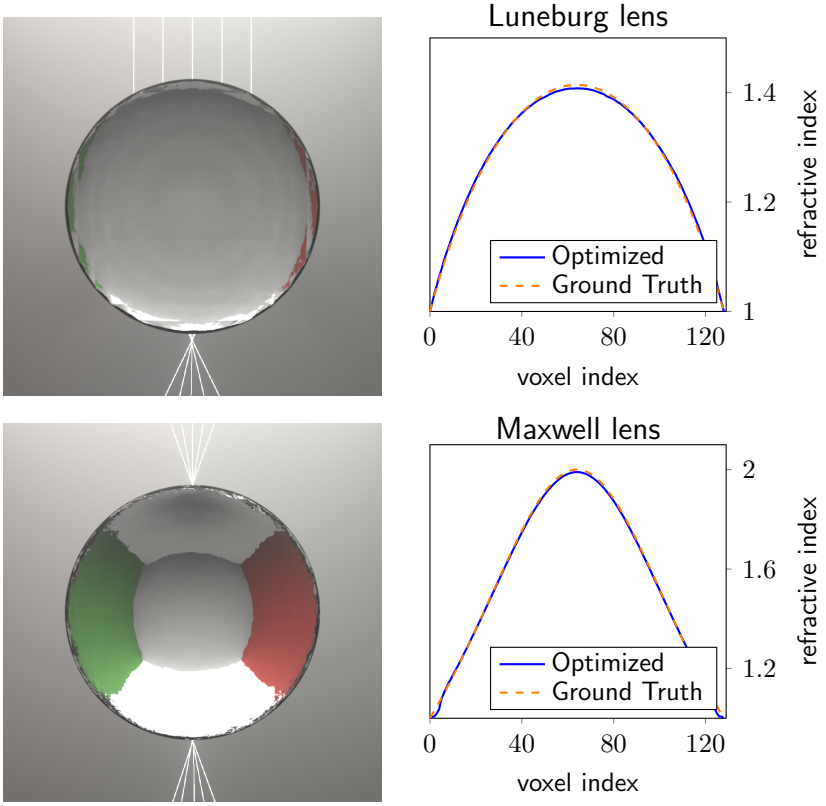


Figure 5.8: Renderings of the reconstructed Luneburg and Maxwell lenses. **(Col 1)** Renderings of both lenses in the Cornell box with lasers shining through the lens. **(Col 2)** A comparison of the center axes of the optimized volume and the ground truth lens. Qualitatively, the paths have little divergence between the two. The greatest disparity occurs at the boundary of the volume. This is because of the constraint we add during optimization which projects the boundary of the solution to a value of one.

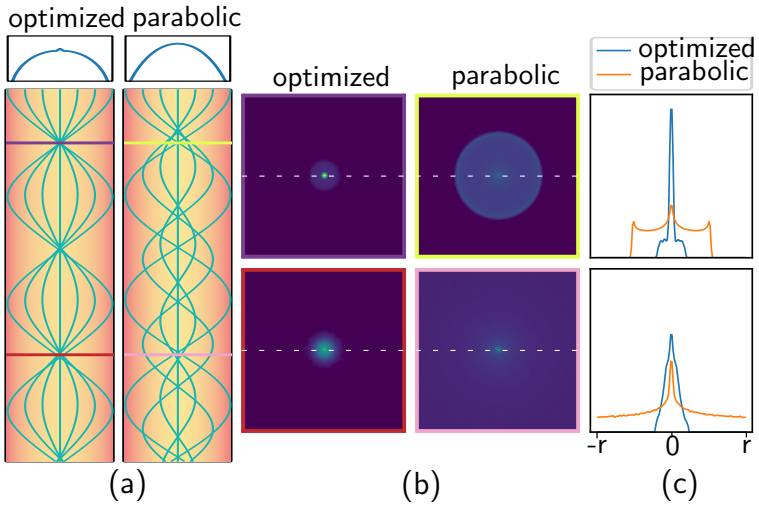


Figure 5.9: A comparison between the optimized fiber design and the parabolic profile. **(a)** A cross-section of each fiber with the ray trajectories. Light disperses the farther it travels in both fibers, but much less so in the optimized fiber. **(b)** Images of the focused source at each of the focus points in the fibers. The images from the optimized fiber are better focused. **(c)** A cross-section of the images, showing the PSF of the fiber at each of the focus points. The optimized fiber retains better focus at the farther hop.

where R is the radius of the lens, and r is the distance from the center of the lens. Even though the Luneburg and Maxwell lenses are primarily of theoretical interest, they provide useful groundtruth profiles for evaluating our method.

To recover these profiles, we optimize refractive index fields for the same geometric loss that, in the notation of Equation (5.15), uses the outer and inner cost functions [4]

$$\mathcal{F}^{\text{GRIN}}(a) \equiv a, \tag{5.40}$$

$$\mathcal{C}^{\text{GRIN}}_{\mathrm{x}_0,\mathrm{v}_0}(\mathrm{x},\mathrm{v}) \equiv \|\mathrm{x} - \hat{\mathrm{x}}(\mathrm{x}_0,\mathrm{v}_0)\|^2. \tag{5.41}$$

where $\hat{\mathrm{x}}(\mathrm{x}_0,\mathrm{v}_0)$ is the target antipodal point determined by the initial ray direction and position. In the case of the Luneburg lens, rays entering the lens with the same *direction* should reach the same end point. For the Maxwell lens, rays entering the lens from the same *point* should reach the same end point. At each gradient descent iteration, we pick at random six directions and a corresponding six target points to optimize over. Figure 5.8 shows the results. The optimized refractive index volumes closely match the analytic refractive fields for both the Luneburg and Maxwell lenses.

Next, we turn our attention to using our method to optimize a GRIN fiber. This is a long waveguide cable that has a rotationally-symmetric refractive index profile. As a result, light traveling through the fiber is curved toward the center of the fiber. As light travels through the medium and does not bounce off of the sides of the fiber, it can travel long distances with minimal loss of energy.

[4]We note that, for this experiment, we use a slightly more general inner cost function than in Equation (5.15), as we allow the inner cost function to change depending on the initial conditions of the ray. This does not affect our optimization formulation.

Table 5.2: Error values for the experiments in Figure 5.10, measured as $\mathcal{L}_2$ relative error from the ground truth.

Method	Orig. Data	10x	100x	1000x
Adjoint [ours]	0.0014	0.014	0.29	3.8
Atcheson et al. [2]	0.0110	0.113	1.28	19.3

The drawback to this design, however, is that light traveling through the fiber experiences dispersion. Rays starting at the center of the fiber have a smaller optical distance to travel compared to rays starting farther away from the center. This results in the waveform eventually being deformed as propagation distance increases. Figure 5.9 visualizes this dispersion. Hisatomi et al. [35] survey different radial profiles and their dispersion characteristics. The simplest such GRIN fiber profile is the parabolic one,

$$\eta_{\text{fiber}}(r) = \sqrt{2 - \left(\frac{r}{R}\right)^2}, \tag{5.42}$$

where r is the distance from the medial axis of the fiber, and R is the radius of the fiber.

We seek to design a profile that exhibits less dispersion compared to the parabolic profile. We start with a collimated source and have it focus to a point. Our cost function is

$$\mathcal{F}^{\text{fiber}}(a) \equiv a, \tag{5.43}$$

$$\mathcal{C}^{\text{fiber}}(\mathrm{x}, \mathrm{v}) \equiv \|\mathrm{x} - \hat{\mathbf{p}}\|, \tag{5.44}$$

where $\hat{\mathbf{p}}$ is the target focal point of the fiber. To enforce rotational symmetry, we use a refractive index field representation that specifies refractive index only as a function of radius.

Figure 5.9 shows the results. Using our optimized refractive index profile results in rays that focus better than using the parabolic profile, and that maintain better focus at multiple points throughout the fiber. At all focus points in the fiber, the optimized profile produces a more focused image than the parabolic profile.

Fuel injection reconstruction An application for our refractive index field optimization procedure is the reconstruction of transparent gas flows, similar to Atcheson et al. [2] and Ji et al. [46]. Both works use active sensing to obtain measurements of the light field entering and exiting a gas volume. They generate correspondences between incident and outgoing rays; then use this information to reconstruct the refractive index field of the volume by assuming that light rays are approximately linear. This is an accurate assumption given that, as the gas flow volume has refractive index changes on the order of 10^{-4}, most rays undergo very little deflection.

We reconstruct the fuel injection dataset from SFB 382 of the German Research Council (DFG) using the measurements as in these prior works. We optimize a geometric loss that, in

the notation of Equation (5.15), uses the outer and inner cost functions

$$\mathcal{F}^{\text{fuel}}(a) \equiv a, \tag{5.45}$$

$$\mathcal{C}^{\text{fuel}}_{\mathrm{x}_0,\mathrm{v}_0}(\mathrm{x},\mathrm{v}) \equiv \|\mathrm{x}-\hat{\mathrm{x}}(\mathrm{x}_0,\mathrm{v}_0)\|^2 + \|\mathrm{v}-\hat{\mathrm{v}}(\mathrm{x}_0,\mathrm{v}_0)\|^2, \tag{5.46}$$

where $\hat{\mathrm{x}}$ and $\hat{\mathrm{v}}$ are the measurements from the ground truth simulation. Importantly, our technique does not assume that the paths of the rays are known prior to the optimization (as Ji et al. [46] do), or linear (as Atcheson et al. [2] do).

Figure 5.10 compares our results with those from the technique of Atcheson et al. [2]. Along with experiments using the original dataset, we show experiments where we scale the refractive index values of the volume, to artificially generate scenes that produce much larger ray deflections. These artificial settings are not representative of the original application on gas flow reconstruction, but help highlight the ability of our technique to handle large deflections. As ray deflection increases, the technique of Atcheson et al. [2] produces reconstructions with more artifacts than in those produced by our technique. The remaining artifacts are because of the nonconvex nature of this inverse problem, and can potentially be reduced with more measurements. Table 5.2 compares the performance of our method and the method of Atcheson et al. [2] in terms of relative error from the ground truth, showing that our method improves performance using the same measurements.

5.6 Discussion

We presented a theory for differentiating optimization objectives constrained by nonlinear ray tracing equations with respect to the underlying refractive index field. We showed both a continuous formulation, and a discretization that lends itself to numerical evaluation. The resulting algorithm has constant memory complexity, and overall requires significantly less memory than previous differentiable rendering methods based on reverse-mode automatic differentiation. Our method supports different types of optimization objectives, involving image and geometric losses. Lastly, we demonstrated the utility of our method through simulated experiments, where we use it for a variety of design and reconstruction problems involving continuously-varying refractive index fields. We now discuss the limitations of our method and outline future research directions.

Initialization As with any gradient-based optimization procedure, our method requires a good initialization. Given that the optimization landscape is highly non-convex, it is easy for gradient descent to get trapped into local minima. In our experiments, initializing to a uniform refractive index field gave satisfactory results. Exploring better initialization schemes (e.g., reconstructions from techniques assuming a single refraction event [2]) is an important future research direction.

Sufficient measurements When using our technique for reconstruction tasks, it is still unclear how many and what measurements are required to correctly recover the underlying

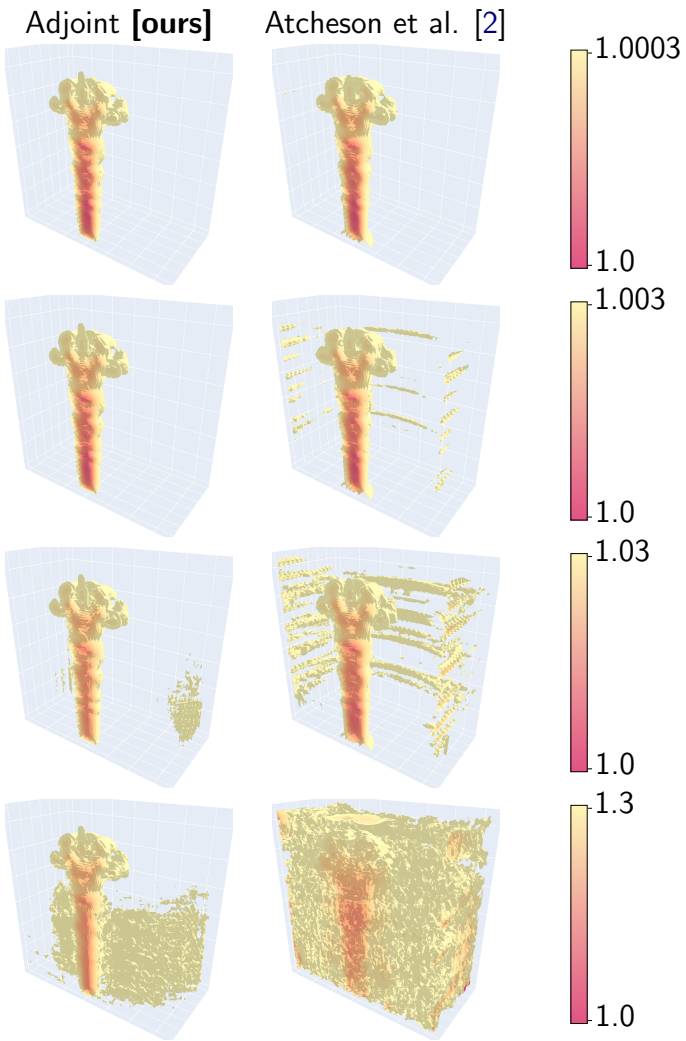


Figure 5.10: Isocontour visualizations of the reconstructions of an unknown refractive index field from a set of images with our method and Atcheson et al. [2]. Every row increases the magnitude of the refractive index field by a factor of 10. With increased refractive index gradients, the light rays deflect by larger amounts. The method proposed by Atcheson et al. [2] struggles to recover the field in such scenarios, because of a linear path assumption imposed on the propagation of light. In contrast, our method recovers the fuel injection scene even in the case of extreme ray deflections.

refractive index field. In the presence of ambiguities, it is possible to have zero measurement loss, while still recovering a different refractive index field. The analysis of what are sufficient measurements for unique refractive index field recovery is an important open problem.

Scattering Our theory and algorithms apply to media where there is only continuous refraction, and no volumetric scattering. However, most real-world materials have both continuous refraction and volumetric scattering. Currently, there exist forward rendering formulations based on the refractive radiative transfer equation for the simulation of such materials [1, 67]. We believe our techniques can be combined with the refractive radiative transfer equation, to enable differentiable rendering in materials that both scatter and continuously refract light.

Discretization bias Our procedure uses discrete numerical integration to simulate continuous ODEs. Inevitably, this introduces bias, which is larger as the simulation step size increases. However, we note that our derivation in Section 5.3 is continuous and thus unbiased. Investigating ways to simulate these continuous equations without discretization bias (e.g., Monte Carlo and randomization techniques) is an interesting future research direction.

Fabrication constraints In most of our simulated experiments, we did not take into account possible fabrication constraints in real-world design tasks, such as those discussed by Teichman et al. [95]. However, as we show in the GRIN fiber optimization example, it is possible to incorporate design constraints into our framework (e.g., rotational symmetry). Future work should investigate incorporating other types of design and fabrication constraints into our optimization framework.

Chapter 6

Conclusion

We summarize the contributions of this thesis and discuss the limitations and open research directions that could be explored.

6.1 Summary of Contributions

Aperture-aware gradients We developed a method for calculating *unbiased* gradients of lens throughput, which enables optimization of lens speed. Using automatic differentiation, or differentiating the ray tracing procedure directly will lead to biased results. We showed that this bias comes from ignoring an integral term that appears when differentiating an integral that has a domain that is dependent on the variable of differentiation. We derived a method for calculating the unbiased gradient, which requires constructing a ray tracing procedure that tracks the aperture of the lens.

Mixed continuous-discrete lens design In order to optimize over the number of elements in a design, we developed a Markov chain Monte Carlo (MCMC) method that combines gradient-based optimization of continuous parameters with discrete mutations that change the number of lens elements. We showed that this method can find better lenses than when using only gradient-based optimization and is able to expand the Pareto front of the speed-sharpness tradeoff.

Adjoint nonlinear ray tracing By looking directly at the nonlinear ray tracing equations, we derived a method for calculating gradients through nonlinear ray tracing using constant memory. Our key insight is that the trajectory of a ray in gradient index field can be completely described by the initial position and direction of the ray, or the final position and direction. This allows us to accumulate the gradient of the refractive index field in the same way as automatic differentitation, but without the need to store the entire ray trajectory.

6.2 Discussion

In this thesis, we made several simplifying assumptions and focused on specific types of optical elements. We discuss the limitations of our work, and how they can be addressed in future research.

Nonconvexity Our techniques facilitate gradient-based optimization of complex objectives for compound lenses. However, these objectives are highly non-convex. Thus gradient-based optimization will often get stuck in local minima. All of the methods developed in this thesis are sensitive to initialization. We can mitigate this issue by combining our technique with simulated annealing, or reasonable initializations from data-driven techniques [21].

Other geometric optical elements We have not considered many geometric optical elements that are commonplace in consumer photography and scientific imaging. Aspherical, Fresnel, and freeform lenses have many more degrees of freedom than spherical ones, and are common in projector and illumination systems. Beyond dioptric (i.e., refractive) systems, catoptric (i.e., reflective) systems are common in telescopes [27, 105]; and catadioptric (i.e., both reflective and refractive) systems [3] facilitate wide-angle imaging. As such systems can be modeled using geometric optics and ray tracing, our theory directly supports or can be readily extended to support their design.

Wave optical elements Other types of optical elements fundamentally rely on wave optics, such as diffractive optical elements and "metalenses" [13]. Such elements are becoming increasingly popular in both scientific and consumer applications, thanks to their miniaturization potential. As our theory relies on geometric optics, it does not apply to such optical elements. Recent progress in wave optics rendering [87] can facilitate extensions in this direction.

Beyond cameras We presented our technique in the context of photographic lenses, but it can apply to other domains. An example is augmented reality (AR) and virtual reality (VR), where design of eyepieces is important. VR displays use intricate configurations of optical elements [71] and benefit from high throughput designs [23]; our technique can help optimize existing such configurations or discover new ones. Another example is non-imaging optics, i.e., optical systems designed specifically to maximize light throughput [88]. Our techniques can be suitable for optimizing designs in this domain.

Other design constraints and trade-offs Lens designs often need to accommodate application-specific constraints that go beyond sharpness and light efficiency. For example, the manufacturing process may constrain the realizable surface shapes. Alternatively, the intended use may constrain the lens form factor or cost. Such constraints are typically differentiable, and thus amenable to optimization using our technique. Even if they do not take the form of hard constraints, such application-specific considerations can introduce

additional trade-offs that the lens design process must balance. In such cases, we can combine our technique with gradient-based multi-objective optimization techniques [78] to discover Pareto-optimal lens designs.

Manufacturing constraints and tolerances Though our method outputs good lenses in simulation, ensuring that these lenses are manufacturable and robust to manufacturing errors is essential for practicality. Our designs satisfy certain manufacturability constraints (e.g., element thickness), but there are other constraints not captured in our method (e.g., element distances). Additionally, our method does not account for manufacturing tolerances to improve robustness. Investigating how to incorporate manufacturing constraints and tolerances is an important future research direction.

Bibliography

[1] Marco Ament, Christoph Bergmann, and Daniel Weiskopf. Refractive radiative transfer equation. 5, 5.1, 5.6

[2] Bradley Atcheson, Ivo Ihrke, Wolfgang Heidrich, Art Tevs, Derek Bradley, Marcus Magnor, and Hans-Peter Seidel. Time-resolved 3d capture of non-stationary gas flows. 5.1, 5.2, 5.5, 5.5, 5.5, 5.6, **??**, 5.10

[3] Simon Baker and Shree K Nayar. A theory of single-viewpoint catadioptric image formation. *International journal of computer vision*, 35:175–196, 1999. 3.6, 6.2

[4] Manushanker Balasubramanian, Sawyer D. Campbell, and Douglas H. Werner. Highly-efficient grin lens optimization through differential ray tracing. 5.1

[5] Sai Bangaru, Tzu-Mao Li, and Frédo Durand. Unbiased warped-area sampling for differentiable rendering. *ACM Trans. Graph.*, 39(6):245:1–245:18, 2020. 3.1

[6] Sai Bangaru, Michael Gharbi, Tzu-Mao Li, Fujun Luan, Kalyan Sunkavalli, Milos Hasan, Sai Bi, Zexiang Xu, Gilbert Bernstein, and Fredo Durand. Differentiable rendering of neural sdfs through reparameterization. In *ACM SIGGRAPH Asia 2022 Conference Proceedings*, SIGGRAPH Asia '22, New York, NY, USA, 2022. Association for Computing Machinery. doi: 10.1145/3550469.3555397. URL https://doi.org/10.1145/3550469.3555397. 3.1, 3.4, 3.4

[7] Amir Barda, Guy Tevet, Adriana Schulz, and Amit Haim Bermano. Generative design of sheet metal structures. *ACM Trans. Graph.*, 42(4), July 2023. ISSN 0730-0301. doi: 10.1145/3592444. URL https://doi.org/10.1145/3592444. 4.1

[8] Ellis I. Betensky. Postmodern lens design. *Optical Engineering*, 32(8):1750 – 1756, 1993. doi: 10.1117/12.145079. URL https://doi.org/10.1117/12.145079. 4.1

[9] Benedikt Bitterli, Wenzel Jakob, Jan Novák, and Wojciech Jarosz. Reversible jump metropolis light transport using inverse mappings. *ACM Trans. Graph.*, 37(1), October 2017. ISSN 0730-0301. doi: 10.1145/3132704. URL https://doi.org/10.1145/3132704. 4.1

[10] Max Born and Emil Wolf. *Principles of optics: electromagnetic theory of propagation, interference and diffraction of light*. Elsevier, 2013. 1, 2.3, 5.2.1

[11] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-

Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax. 3.5, 4.6

[12] G. Cai, K. Yan, Z. Dong, I. Gkioulekas, and S. Zhao. Physics-based inverse rendering using combined implicit and explicit geometries. *Computer Graphics Forum*, 41(4), 2022. 3.1

[13] Praneeth Chakravarthula, Jipeng Sun, Xiao Li, Chenyang Lei, Gene Chou, Mario Bijelic, Johannes Froesch, Arka Majumdar, and Felix Heide. Thin on-sensor nanophotonic array cameras. *ACM Transactions on Graphics (TOG)*, 42(6):1–18, 2023. 3.6, 6.2

[14] Maysamreza Chamanzar, Matteo Giuseppe Scopelliti, Julien Bloch, Ninh Do, Minyoung Huh, Dongjin Seo, Jillian Iafrati, Vikaas S Sohal, Mohammad-Reza Alam, and Michel M Maharbiz. Ultrasonic sculpting of virtual optical waveguides in tissue. 5.1

[15] Guy Chavent. Identification of functional parameters in partial differential equations. 06 1974. 3.1

[16] Min Chen and James Arvo. Theory and application of specular path perturbation. *ACM Trans. Graph.*, 19(4):246–278, oct 2000. doi: 10.1145/380666.380670. URL https://doi.org/10.1145/380666.380670. 3.1

[17] Ni Chen, Congli Wang, and Wolfgang Heidrich. ∂h: Differentiable holography. *Laser & Photonics Reviews*, 17(9):2200828, 2023. 4.7

[18] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *NeurIPS*, 2018. 3.1

[19] Roger N. Clark. Clarkvision.com, 2023. URL https://clarkvision.com/articles/digital.sensor.performance.summary/index.html. Accessed: 2024-13-1. 3.7

[20] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2023. URL http://www.blender.org. 3.5

[21] Geoffroi Côté, Jean-François Lalonde, and Simon Thibault. Deep learning-enabled framework for automatic lens design starting point generation. *Opt. Express*, 29(3): 3841–3854, Feb 2021. doi: 10.1364/OE.401590. URL https://opg.optica.org/oe/abstract.cfm?URI=oe-29-3-3841. 3.6, 4, 4.1, 4.7, 6.2

[22] Arthur Cox. Photographic optics: a modern approach to the technique of definition. *(No Title)*, 1974. 3.5

[23] Gerwin Damberg, James Gregson, and Wolfgang Heidrich. High brightness hdr projection using dynamic freeform lensing. *ACM Trans. Graph.*, 35(3), may 2016. ISSN 0730-0301. doi: 10.1145/2857051. URL https://doi.org/10.1145/2857051. 3.1, 3.6, 6.2

[24] Ruta Desai, James McCann, and Stelian Coros. Assembly-aware design of printable electromechanical devices. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, page 457–472, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359481. doi: 10.1145/3242587.

3242655. URL https://doi.org/10.1145/3242587.3242655. 4.1

[25] Fouad El-Diasty. Evaluation of some grin fiber parameters and the associated fraction mode loss due to mechanically induced optical anisotropy. 5.1

[26] Stewart N Ethier and Thomas G Kurtz. *Markov processes: characterization and convergence*. John Wiley & Sons, 2009. 4.4.1

[27] Jonathan P Gardner, John C Mather, Mark Clampin, Rene Doyon, Matthew A Greenhouse, Heidi B Hammel, John B Hutchings, Peter Jakobsen, Simon J Lilly, Knox S Long, et al. The james webb space telescope. *Space Science Reviews*, 123: 485–606, 2006. 3.6, 6.2

[28] Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. Add: analytically differentiable dynamics for multi-body systems with frictional contact. *ACM TOG*, 2020. 3.1

[29] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM review, 2008. 3.1, 5.5

[30] Eduard Gröller. Nonlinear ray tracing: Visualizing strange worlds. 5.1

[31] Diego Gutierrez, A Muñoz, F Seron, E Jimenez, María de Luna, and Edificio Ada Byron. Global illumination in inhomogeneous media based on curved photon mapping. 5.1

[32] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric numerical integration*. Springer-Verlag, Berlin,. 5.2.1, 5.4

[33] Samuel W Hasinoff, Frédo Durand, and William T Freeman. Noise-optimal capture for high dynamic range photography. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 553–560. IEEE, 2010. 3.5

[34] Michael Hinze, René Pinnau, Michael Ulbrich, and Stefan Ulbrich. *Optimization with PDE constraints*. Springer, 2008. 3.1

[35] Makiko Hisatomi, Michael C. Parker, and Stuart D. Walker. Comparison of zoned microstructure fiber geometries for low-dispersion waveguiding. 5.5

[36] Chi-Jui Ho, Yash Belhe, Steve Rotenberg, Ravi Ramamoorthi, Tzu-Mao Li, and Nicholas Antipa. A differentiable wave optics model for end-to-end computational imaging system optimization, 2024. URL https://arxiv.org/abs/2412.09774. 4.7

[37] Sascha Holl, Hans-Peter Seidel, and Gurprit Singh. Jump restore light transport. *arXiv preprint arXiv:2409.07148*, 2024. 4.1, 4.4

[38] Matthias B. Hullin, Johannes Hanika, and Wolfgang Heidrich. Polynomial Optics: A construction kit for efficient ray-tracing of lens systems. *Computer Graphics Forum (Proceedings of EGSR 2012)*, 31(4), July 2012. 3.1

[39] Kaspar Höschel and Vasudevan Lakshminarayanan. Genetic algorithms for lens design: a review. *Journal of Optics*, 48(1):134–144, December 2018. ISSN 0974-6900. doi: 10.1007/s12596-018-0497-3. URL http://dx.doi.org/10.1007/s12596-018-0497-3.

4, 4.1

[40] Ivo Ihrke. Reconstruction and rendering of time-varying natural phenomena. 5.1

[41] Ivo Ihrke, Gernot Ziegler, Art Tevs, Christian Theobalt, Marcus Magnor, and Hans-Peter Seidel. Eikonal rendering: Efficient light transport in refractive objects. 5, 5.1, 5.4

[42] Wenzel Jakob. Enoki: structured vectorization and differentiation on modern processor architectures. https://github.com/mitsuba-renderer/enoki. 5.5

[43] Wenzel Jakob and Steve Marschner. Manifold exploration: A markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 31(4):58:1–58:13, July 2012. doi: 10.1145/2185520.2185554. 3.1, 3.4

[44] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. Mitsuba 3 renderer, 2022. https://mitsuba-renderer.org. 1

[45] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. Dr.jit: A just-in-time compiler for differentiable rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 41(4), July 2022. doi: 10.1145/3528223.3530099. 3.1, 3.4

[46] Yu Ji, Jinwei Ye, and Jingyi Yu. Reconstructing gas flows using light-path approximation. 5.1, 5.5, 5.5

[47] SeungYeon Kang, Elena Dotsenko, David Amrhein, Christian Theriault, and Craig B Arnold. Ultra-high-speed variable focus optics for novel applications in advanced imaging. In *Photonic Instrumentation Engineering V*. 5.1

[48] Liliya Kharevych, W Wei, Yiying Tong, Eva Kanso, Jerrold E Marsden, Peter Schröder, and Matthieu Desbrun. *Geometric, variational integrators for computer animation*. Eurographics Association. 5.4

[49] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 5.5

[50] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 3.5, 4.3, 4.4.1

[51] Yu A Kravtsov and Yu I Orlov. *Geometrical optics of inhomogeneous media*. Springer. 5.1, 5.1, 5.2, 5.2.1

[52] Tzu-Mao Li, Jaakko Lehtinen, Ravi Ramamoorthi, Wenzel Jakob, and Frédo Durand. Anisotropic gaussian mutations for metropolis light transport through hessian-hamiltonian dynamics. *ACM Trans. Graph.*, 34(6), November 2015. ISSN 0730-0301. doi: 10.1145/2816795.2818084. URL https://doi.org/10.1145/2816795.2818084. 4.1, 4.3

[53] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018. 1, 3.1

[54] Tzu-Mao Li, Michaël Gharbi, Andrew Adams, Frédo Durand, and Jonathan Ragan-Kelley. Differentiable programming for image processing and deep learning in halide. *ACM Transactions on Graphics (ToG)*, 37(4):1–13, 2018. 3.1

[55] Zhengqin Li, Yu-Ying Yeh, and Manmohan Chandraker. Through the looking glass: Neural 3d reconstruction of transparent shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1262–1271, 2020. 3.4

[56] Fujun Luan, Shuang Zhao, Kavita Bala, and Ioannis Gkioulekas. Langevin monte carlo rendering with gradient-based adaptation. *ACM Trans. Graph.*, 39(4), August 2020. ISSN 0730-0301. doi: 10.1145/3386569.3392382. URL https://doi.org/10.1145/3386569.3392382. 4.1, 4.3

[57] Rudolf K Luneberg. *Mathematical Theory of Optics. Providence*. Brown Univ. Press. 5.1, 5.5

[58] James Clerk Maxwell. Solutions of problems. 5.1, 5.5

[59] Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. Fluid control using the adjoint method. *ACM TOG*, 2004. 3.1

[60] Jocelyn Meyron, Quentin Mérigot, and Boris Thibert. Light in power: A general and parameter-free algorithm for caustic design. 5.5

[61] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3504–3515, 2020. 3.2

[62] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. 5.4, 5.5, 5.5

[63] Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. Radiative backpropagation: An adjoint method for lightning-fast differentiable rendering. *ACM TOG*, 2020. 3.1

[64] Hisanari Otsu, Anton S. Kaplanyan, Johannes Hanika, Carsten Dachsbacher, and Toshiya Hachisuka. Fusing state spaces for markov chain monte carlo rendering. *ACM Trans. Graph.*, 36(4), July 2017. ISSN 0730-0301. doi: 10.1145/3072959.3073691. URL https://doi.org/10.1145/3072959.3073691. 4.1

[65] Marios Papas, Wojciech Jarosz, Wenzel Jakob, Szymon Rusinkiewicz, Wojciech Matusik, and Tim Weyrich. Goal-based caustics. In *Computer Graphics Forum*, volume 30, pages 503–511. Wiley Online Library. 3.4

[66] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. 5.5

[67] Adithya Pediredla, Yasin Karimi Chalmiani, Matteo Giuseppe Scopelliti, Maysamreza

Chamanzar, Srinivasa Narasimhan, and Ioannis Gkioulekas. Path tracing estimators for refractive radiative transfer. 5.1, 5.1, 5.4, 5.5, 5.6

[68] Frank L Pedrotti, Leno M Pedrotti, and Leno S Pedrotti. *Introduction to optics*. Cambridge University Press, 2017. 1, 1.1, 3.1, 4.5

[69] Rene-Edouard Plessix. A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. 5.2

[70] Murray Pollock, Paul Fearnhead, Adam M Johansen, and Gareth O Roberts. Quasi-stationary monte carlo and the scale algorithm. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 82(5):1167–1221, 2020. 4.1, 4.4.1

[71] Yingsi Qin, Wei-Yu Chen, Matthew O'Toole, and Aswin C. Sankaranarayananan. Split-lohmann multifocal displays. *ACM Transactions on Graphics / SIGGRAPH*, 31, Aug 2023. doi: 10.1145/3592110. URL https://doi.org/10.1145/3592110. 3.6, 6.2

[72] Jason Rader, Terry Lyons, and Patrick Kidger. Optimistix: modular optimisation in jax and equinox. *arXiv:2402.09983*, 2024. 4.6

[73] Daniel Reiley. Lens designs, 2014. URL https://www.lens-designs.com/. Accessed: 2025-05-18. 3.5, 3.5, 4.6

[74] Osborne Reynolds. *Papers on Mechanical and Physical Subjects: The sub-mechanics of the universe*, volume 3. The University Press, 1903. 3.3

[75] Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341 – 363, 1996. 4.6

[76] Paul Rudolph. Object glass, U.S. Patent 583 336, May. 1897. 3.5

[77] Udo Schröder and Thomas Schuster. An iterative method to reconstruct the refractive index of a medium from time-of-flight measurements. 32(8):085009. doi: 10.1088/0266-5611/32/8/085009. URL https://doi.org/10.1088/0266-5611/32/8/085009. 5.1

[78] Adriana Schulz, Harrison Wang, Eitan Grinspun, Justin Solomon, and Wojciech Matusik. Interactive exploration of design trade-offs. *ACM Trans. Graph.*, 37(4), jul 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201385. URL https://doi.org/10.1145/3197517.3201385. 3.6, 6.2

[79] Yuliy Schwartzburg, Romain Testuz, Andrea Tagliasacchi, and Mark Pauly. High-contrast computational caustic design. *ACM Transactions on Graphics (TOG)*, 33(4): 1–11, 2014. 3.4, 5.5

[80] Matteo Giuseppe Scopelliti and Maysamreza Chamanzar. Ultrasonically sculpted virtual relay lens for in situ microimaging. 5.1

[81] Matteo Giuseppe Scopelliti, Hengji Huang, Adithya Pediredla, Srinivasa G Narasimhan, Ioannis Gkioulekas, and Maysamreza Chamanzar. Overcoming the tradeoff between confinement and focal distance using virtual ultrasonic optical waveguides. 5.1

[82] Anurag Sharma, Dhanwada Vizia Kumar, and Ajoy K. Ghatak. Tracing rays through

graded-index media: a new method. 5.1, 5.2.1

[83] Nicholas Sharp and Keenan Crane. Variational surface cutting. *ACM TOG*, 2018. 3.1

[84] Warren J Smith. *Modern optical engineering: the design of optical systems*. McGraw-Hill Education, 2008. 1.1, 1.1, 4.5

[85] Jos Stam. Computing light transport gradients using the adjoint method. *arXiv:2006.15059*, 2020. 3.1

[86] Jos Stam and Eric Languénou. Ray tracing in non-constant media. In *EGSR*. 5, 5.1

[87] Shlomi Steinberg and Ling-Qi Yan. A generic framework for physical light transport. *ACM Trans. Graph.*, 40(4), jul 2021. ISSN 0730-0301. doi: 10.1145/3450626.3459791. URL https://doi.org/10.1145/3450626.3459791. 3.6, 6.2

[88] Shlomi Steinberg, Nandor Bokor, and Nir Davidson. Two-mirror compact system for ideal concentration of diffuse light. *Journal of the Optical Society of America A*, 39(4):628, Mar 2022. ISSN 1520-8532. doi: 10.1364/josaa.447493. URL http://dx.doi.org/10.1364/JOSAA.447493. 3.6, 6.2

[89] Libin Sun, Brian Guenter, Neel Joshi, Patrick Therien, and James Hays. Lens factory: Automatic lens generation using off-the-shelf components, 2015. 3.1, 4.1

[90] Qilin Sun, Congli Wang, Qiang Fu, Xiong Dun, and Wolfgang Heidrich. End-to-end complex lens design with differentiate ray tracing. *ACM Trans. Graph.*, 40(4), jul 2021. ISSN 0730-0301. doi: 10.1145/3450626.3459674. URL https://doi.org/10.1145/3450626.3459674. 4, 4.2

[91] Synopsis. Code v optical design software, 2023. URL https://www.synopsys.com/optical-solutions/codev.html. 1, 1.1, 2.1, 2.3, 2.5

[92] Huixuan Tang and Kiriakos N. Kutulakos. What does an aberrated photo tell us about the lens and the scene? In *IEEE International Conference on Computational Photography (ICCP)*, pages 1–10, 2013. doi: 10.1109/ICCPhot.2013.6528316. 3.1

[93] Arjun Teh, Matthew O'Toole, and Ioannis Gkioulekas. Adjoint nonlinear ray tracing. *ACM Transactions on Graphics (TOG)*, 41(4):1–13, 2022. 3.1

[94] Arjun Teh, Ioannis Gkioulekas, and Matthew O'Toole. Aperture-aware lens design. In *ACM SIGGRAPH 2024 Conference Papers*, SIGGRAPH '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400705250. doi: 10.1145/3641519.3657398. URL https://doi.org/10.1145/3641519.3657398. 4, 2., 4.2, 4.6

[95] Jeremy Teichman, Jenny Holzer, Bohdan Balko, Brent Fisher, and Leonard Buckley. Gradient index optics at darpa. Technical report, Institute for Defense Analyses. 5.1, 5.6

[96] Ethan Tseng, Ali Mosleh, Fahim Mannan, Karl St-Arnaud, Avinash Sharma, Yifan Peng, Alexander Braun, Derek Nowrouzezahrai, Jean-Francois Lalonde, and Felix Heide. Differentiable compound optics and processing pipeline optimization for end-to-end camera design. *ACM Transactions on Graphics (TOG)*, 40(2), 2021. 3.1, 4, 4.2

[97] Eric Veach. *Robust Monte Carlo methods for light transport simulation*. Stanford University. 5.3

[98] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, page 65–76, USA, 1997. ACM Press/Addison-Wesley Publishing Co. ISBN 0897918967. doi: 10.1145/258734.258775. URL https://doi.org/10.1145/258734.258775. 4.1

[99] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. Path replay backpropagation: differentiating light paths using constant memory and linear time. *ACM Transactions on Graphics (TOG)*, 40(4):1–14, 2021. 1, 3.1, 3.1

[100] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. Differentiable signed distance function rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 41(4): 125:1–125:18, July 2022. doi: 10.1145/3528223.3530139. 3.1, 3.4, 3.4

[101] Bruce Walter, Shuang Zhao, Nicolas Holzschuch, and Kavita Bala. Single scattering in refractive media with triangle mesh boundaries. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605587264. doi: 10.1145/1576246.1531398. URL https://doi.org/10.1145/1576246.1531398. 3.1

[102] Andi Q Wang, Murray Pollock, Gareth O Roberts, and David Steinsaltz. Regeneration-enriched markov processes with application to monte carlo. 2021. 4.1, 4.4

[103] Congli Wang, Ni Chen, and Wolfgang Heidrich. do: A differentiable engine for deep lens design of computational imaging systems. *IEEE Transactions on Computational Imaging*, 8:905–916, 2022. 3.1, 3.1, 3.2, 3.3, 3.5, 3.5, 3.5, 3.6, 4, 4.2

[104] ShiLi Wei, ZhengBo Zhu, ZiChao Fan, and DingLin Ma. Least-squares ray mapping method for freeform illumination optics design. 5.5

[105] Raymond N Wilson. *Reflecting telescope optics II: manufacture, testing, alignment, Modern Techniques*. Springer Science & Business Media, 2013. 3.6, 6.2

[106] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. 5.3

[107] Yi-Ting Yeh, Lingfeng Yang, Matthew Watson, Noah D. Goodman, and Pat Hanrahan. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM Trans. Graph.*, 31(4), July 2012. ISSN 0730-0301. doi: 10.1145/2185520.2185552. URL https://doi.org/10.1145/2185520.2185552. 4.1

[108] Tizian Zeltner, Iliyan Georgiev, and Wenzel Jakob. Specular manifold sampling for rendering high-frequency caustics and glints. *ACM Trans. Graph.*, 39(4), August 2020. ISSN 0730-0301. doi: 10.1145/3386569.3392408. URL https://doi.org/10.1145/3386569.3392408. 3.1, 3.4

[109] Zemax. Opticstudio, 2023. URL https://www.ansys.com/products/optics/ansys-zemax-opticstudio. 1, 1.1, 2.1, 2.3, 2.5, 3.5, 3.6, 3.9

[110] Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. Path-space differentiable rendering. *ACM Trans. Graph.*, 39(4), August 2020. ISSN 0730-0301. doi: 10.1145/3386569.3392383. URL https://doi.org/10.1145/3386569.3392383. 3.1

[111] Ziyi Zhang, Nicolas Roussel, and Wenzel Jakob. Projective sampling for differentiable rendering of geometry. *ACM Transactions on Graphics (TOG)*, 42(6):1–14, 2023. 3.5

[112] Nenad Zoric, Yunfeng Nie, Simon Thibault, Radomir Prodanovic, and Lijo Thomas. Global search algorithms in an automated design of starting points for a deep-uv lithography objective. *Appl. Opt.*, 63(26):6960–6968, Sep 2024. doi: 10.1364/AO.532057. URL https://opg.optica.org/ao/abstract.cfm?URI=ao-63-26-6960. 4.1

[113] Nenad Zoric, Simon Thibault, Marie-Anne Burcklen, Lijo Thomas, Momcilo Krunic, and Yunfeng Nie. Combined Rule-Based and Generative Artificial Intelligence in the Design of Smartphone Optics. 4 2025. doi: 10.1364/opticaopen.28759226.v1. URL https://preprints.opticaopen.org/articles/preprint/Combined_Rule-Based_and_Generative_Artificial_Intelligence_in_the_Design_of_Smartphone_Optics/28759226. 4, 4.1