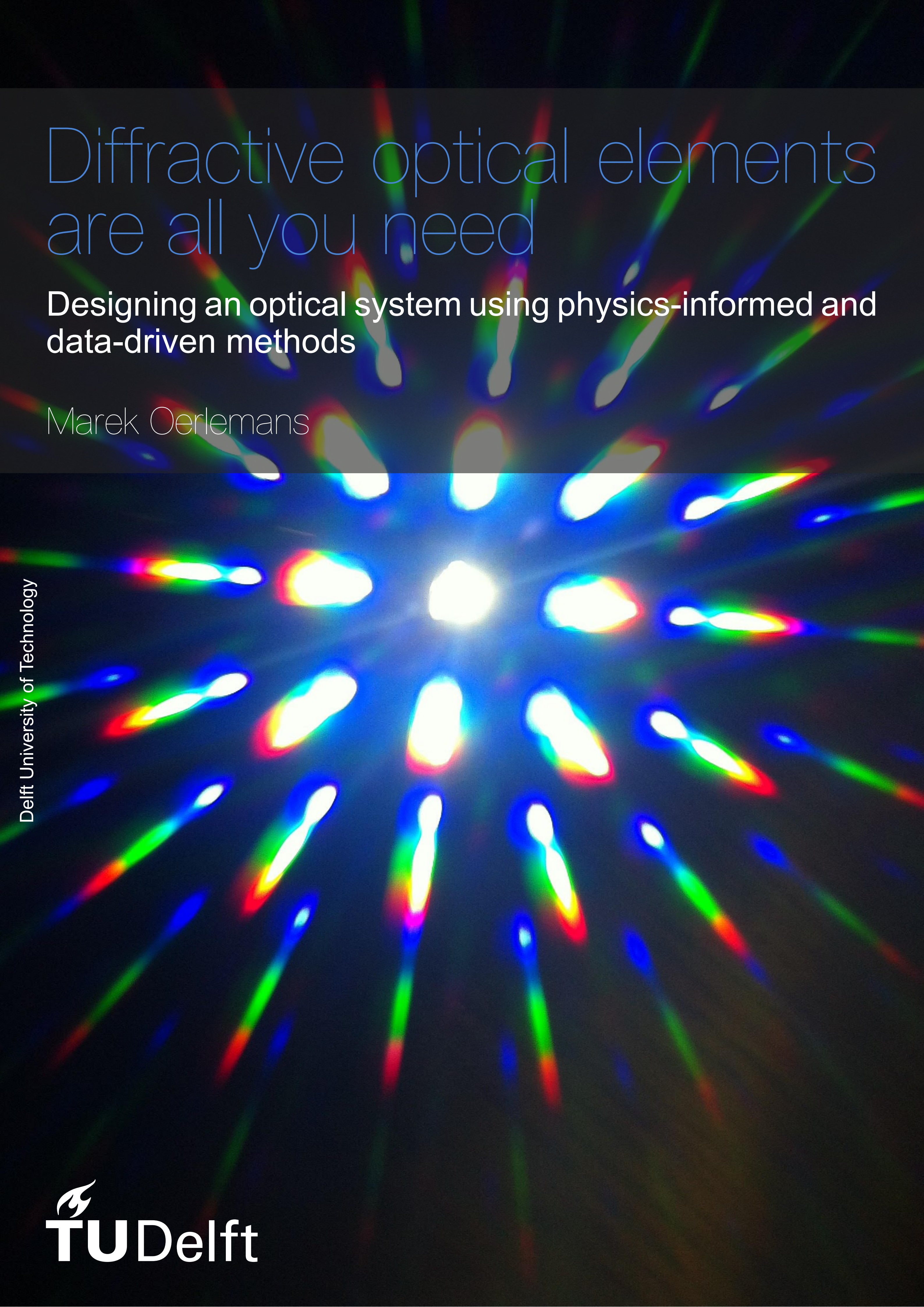
Diffractive optical elements are all you need

Designing an optical system using physics-informed and data-driven methods

Marek Oerlemans

Delft University of Technology



TU Delft

Diffractive optical elements are all you need

Designing an optical system using physics-informed and data-driven methods

by

Marek Oerlemans

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday July 7, 2022 at 14:00.

Student number:	4970071	
Project duration:	September 1, 2021 – July 7, 2022	
Thesis committee:	dr. M. Möller,	TU Delft, supervisor
	dr. A. Adam,	TU Delft, supervisor
	ir. A. M. N. Heemels,	TU Delft
	dr. H. N. Kekkonen,	TU Delft

Front cover image taken from WikiMedia Commons

Preface

The work you are about to read concludes my past six years of studying at university. This journey started for me at Erasmus University in Rotterdam, where I started a bachelor in Econometrics in 2016. As the econometrics program is practically the same as a mathematics program in the first year, this first year already build mathematical interest in me. My interest in mathematics only grew from then on. Where econometrics is mostly applied mathematics, this led me to a gap in my knowledge and an interest to discover more rigorous mathematics. I tried to fill this gap by taking the minor in applied mathematics in my third bachelor year. Here I was introduced to partial differential equations, numerical mathematics and real analysis. Even though I almost failed my first course ever, the course in partial differential equations (as I had none of the prerequisites and had not really seen differential equations at all), I had a very good time during my minor.

Filling some gaps in my mathematical needs I chose to study financial econometrics. But already in one of the first courses, I felt I missed the rigour and reason how one could solve a numerical problem (something about the Black-Scholes equation). The year after I returned to TU Delft to fill this gap in my knowledge by starting the master program in mathematics. The one thing that I feel is so special about this program is the absolute freedom I had to choose courses. I already pursued this study because of interest and now I could fill it with everything I liked. This is also one of the reasons I find myself now, a little shorter than two years after I started the master, with 139 study points obtained instead of the normal 120 study points.

Another special thing about this master is that it leads to such broad projects. Besides this work, I have done an internship in medical imaging and projects in fluid dynamics and optimization. Had you told me two years ago that I would be doing optics, I would have laughed at you. These broad applications of mathematics have also led me to my next step, which is a PhD position at the Netherlands Cancer Institute.

That this time in Delft was such a wonderful time for me was made possible by a certain number of people. I would like to say thank you to my supervisors Matthias, Aurèle and Alex for all the hours of supervision. I would like to say thanks to other students with whom I have worked during this master. And I would like my girlfriend, friends and family for supporting me.

The text you are about to read will lead you through the valleys of optics and the peaks of deep learning. What I think makes this work so special are the completely different fields coming together. I hope you become as curious about the possibilities of optics by reading as I became by writing this.

Marek Oerlemans
Rotterdam, July 2022

Summary

In this work, we consider how to optimize an optical system, specifically one with diffractive optical elements (DOE). We start by describing optical theory called Fourier optics also known as wave optics. This type of optics is found by making assumptions from the Maxwell equations for magnetic and electrical fields. This leads us to the Rayleigh-Sommerfeld diffraction integral, which we need to propagate light.

To optimize an optical system, we introduce the standard optimization methods used when gradients are available and also dive into data-driven methods. Two well-known algorithms in each category: the Adam optimization method, which is an extension of normal gradient descent methods, and the UNet convolutional neural network. To make the optimization methods work with our physics simulation, we use an automatically differentiable implementation which gives the gradients for the optimization.

Combining the two optimization methods with our optics engine, we optimize optical designs such that the resulting intensity on the sample plane resembles some target intensity. We are able to optimize systems with single and multiple DOE and for high and low resolution DOE designs. We find that more lenses makes the optimization better and increases the variability in the created projection. We also find that increasing the resolution severely slows down the optimization with the Adam method.

Although, the optimization method Adam is well suited for this optimization task. It becomes computationally very expensive on high resolution due to the physics simulation at every optimization step. Some physical simulations require high resolution to make sure the simulation does not contain to much artefacts. We show that the data-driven approach has potential to solve this issue. We train a network that takes as input a target intensity and outputs the lens that produces that intensity.

Combining these results, we conclude that modern optimization methods are well suited for optical system optimization and we find that there is a large untapped potential for data-driven methods in optics.

Contents

1 Introduction

The study of optics dates back to the ancient Egyptians and Mesopotamiens, who developed the first lenses using polished crystals. Optics has gained a lot of traction since then and applications such as glasses and optics applied to computer chips cannot be overlooked in modern society. The design of lenses is an active topic of research as it poses a difficult inverse problem. If one desires to know what light a lens projects, a light could shined through the lens and the intensity it produces can be observed on a sample plane. However, if one wants to know what type of lens creates a certain projection, this cannot be told directly from the projection. Here, the forward step would be shining a light through the lens and one would have to change the lens a little bit to improve the projection bit by bit. Similar to the process at an optometrist. You tell the optometrist if the current lens is more or less clear than the previous lens and step-by-step you find the correct strength for your eyes. This is a rather inefficient procedure of trial and error, which often is the result when solving an inverse problem.

Lenses in glasses are usually of simple shapes. These shapes are symmetric and their most important property is the focal point. However, in certain applications these simple symmetric lenses do not have enough variety. We use a freeform lens, which is a lens that has a non-symmetric, but still a continuous, surface. These lenses find application in many different use cases. An ingenious example of freeform lens capability is an application to street lights. LED-lights are energy efficient, but illuminate a large area with a low intensity. For street lightning, a uniform intensity in a focused area right in front of the light is desired, this can be achieved by freeform lenses ([30]). Freeform lenses are applied to improve camera lenses and applications of freeform lenses are found in caustic design. This last application is similar to our research, which will be discussed below.

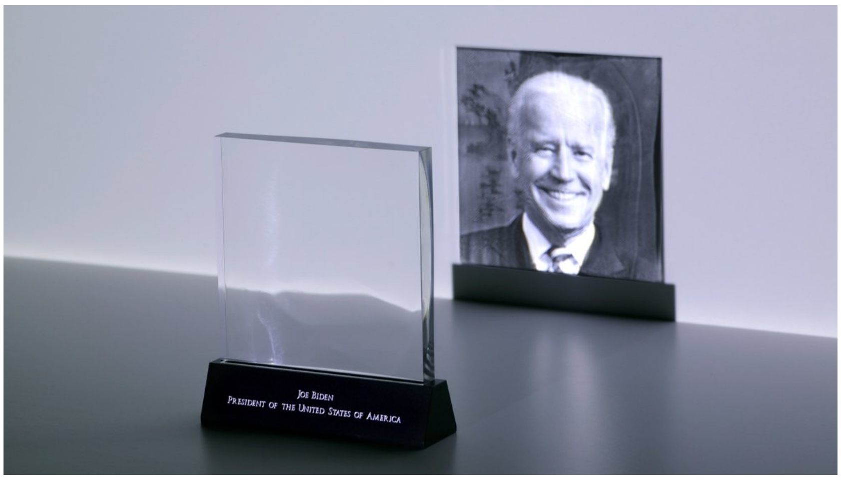


Figure 1.1: Example of an application with a projection through a freeform lens made by Rayform.

Unfortunately, with a highly variable freeform lens, the search space for optimal lens parameters increases in size. This is where recent advances in computational mathematics are helpful. Decades ago iterative algorithms like the Gerchberg-Saxton algorithm for phase retrieval were introduced. Moreover, over time multiple approaches to design freeform lenses have been made using elliptic differential equation or by turning the problem into a minimization problem. We also approach this problem as a minimization problem.

Alongside optics, the area of statistical learning and machine learning has seen broad improvements recently. Due to the increase in computing power, larger networks can be optimized which results in for example hand written digit recognition with over ninety percent accuracy (see for example [22]). These deep learning applications are also interesting for our problem. We apply these similar methodologies to optics. In more recent work, it has also been suggested to implement physical knowledge in neural networks, to combine the physical knowledge and the broad generalization of these deep networks.

We bridge the gap between applied optics theory and recent computational advances in deep learning. Several possibilities have already been shown in, for example, deep diffractive neural networks. We use these methods to our advantage and extend their results to freeform lenses as the current literature on freeform elements with our application is limited. We do this all with one central question in mind:

How can we optimize a system consisting of diffractive optical elements efficiently?

In this thesis we answer this question by the combination of state-of-the-art optimization, neural network models and with optical systems, which we specify more clearly later on. We start by applying certain optimization algorithms directly to the optical problems and see that these optimization methods are able to optimize the system, but this approach does not work in all circumstances. Therefore, a deep learning pipeline is proposed to make the setup work with higher resolutions. Lastly, it is shown how our methods compare and our optimization pipeline is applied to problems with multiple sources. This final experiment looks into optimizing an optical system with multiple sources in combination with multiple lenses. Fourier optics in combination with gradient descent type algorithms give a good foundation to optimize optical systems. In situations where this setup fails, a network can be trained and improves the system with a data-driven approach. This shows that new advances in data-driven modelling are able to improve optics research.

This thesis is divided in five chapters. After the introduction, the second chapter concerns all theoretical foundations to our methods. We subdivide this chapter into four subsections about theoretical optics, numerical optics, deep learning and optimization, and parametric surfaces. The third chapter discusses what has been done in literature and at the optics group of TU Delft as well. The fourth chapter discusses our experiments. Building on the theory, this chapter discusses our methods and shows what methods work and which did not. Chapter five concludes the thesis and gives our outlook on interesting research directions in the future.

2

Theoretical foundations

2.1. Fourier Optics

In this section the theory of optical physics is introduced. The framework of Fourier optics is considered, which is a part of scalar wave optics. Starting with the general Maxwell equations for electric and magnetic fields, scalar diffraction theories such as Rayleigh-Sommerfeld diffraction are derived. On top of this, simplifications in either the near-field or the far-field given by Fresnel or Frauenhofer diffraction, respectively, are considered. This section is mostly based on theory discussed in [20]

2.1.1. Wave Equation

The Maxwell equations are a set of partial differential equations that together describe how electromagnetic fields are related to themselves and their sources. In the absence of free charge, the equations are given by

$$\begin{aligned} \nabla \times \vec{\mathcal{E}} &= -\mu \frac{\partial \vec{\mathcal{H}}}{\partial t}, \\ \nabla \times \vec{\mathcal{H}} &= \epsilon \frac{\partial \vec{\mathcal{E}}}{\partial t}, \\ \nabla \cdot \epsilon \vec{\mathcal{E}} &= 0 \\ \nabla \cdot \mu \vec{\mathcal{H}} &= 0. \end{aligned} \tag{2.1.1}$$

Here, $\vec{\mathcal{E}}$ is a vector describing the real electric field in three dimensions. $\vec{\mathcal{H}}$ is the real magnetic field also in three dimensions. The parameters μ and ϵ are, respectively, the permeability and the permittivity in the medium. We assume that the medium has certain properties, such as constant permittivity in the whole medium. $\times$ and $\cdot$ are the vector cross product and vector dot product, respectively. The ∇ operator is the gradient to the specific dimension. Cross-multiplying the first equation on both sides by the gradient operator and substituting the second equation in the right-hand side. Rewriting the left-hand side with $\nabla \times \nabla \times \vec{\mathcal{E}} = \nabla(\nabla \cdot \vec{\mathcal{E}}) - \nabla^2 \vec{\mathcal{E}}$ and substituting the third equation, where ϵ is assumed to be constant. The following equations are found

$$\begin{aligned} \nabla^2 \vec{\mathcal{E}} - \frac{n^2}{c^2}\frac{\partial^2 \vec{\mathcal{E}}}{\partial t^2} &= 0, \\ \nabla^2 \vec{\mathcal{H}} - \frac{n^2}{c^2}\frac{\partial^2 \vec{\mathcal{H}}}{\partial t^2} &= 0, \end{aligned} \tag{2.1.2}$$

where

$$n^2 = \frac{\epsilon}{\epsilon_0} \quad \text{and} \quad c^2 = \frac{1}{\mu_0 \epsilon_0}. \tag{2.1.3}$$

Here, n is the refractive index of the medium, ϵ_0 is the vacuum permittivity and c is the light speed of propagation in the medium. Since all elements of $\vec{\mathcal{E}}$ and all elements of $\vec{\mathcal{H}}$ follow the above equations, they can be posed in a single equation

$$\nabla^2 u - \frac{n^2}{c^2}\frac{\partial^2 u}{\partial t^2} = 0, \tag{2.1.4}$$

where $u(\vec{x}, t)$ is a function of position $\vec{x} = (x, y, z)$ and time. This equation is known as the wave equation. Certain assumptions were made on the medium and thus the equation is only an approximation for media that closely follow these assumptions.

Considering monochromatic waves, the function $u(\vec{x}, t)$ can be separated as follows

$$u(\vec{x}, t) = \Re\left\{U(\vec{x}) \exp\left(-2\pi j v t\right)\right\} \tag{2.1.5}$$

$$U(\vec{x}) = A(\vec{x}) \exp\left(-j\phi(\vec{x})\right), \tag{2.1.6}$$

$$U(r) = \frac{A(r)}{r} \exp\left(jkr + \phi\right), \tag{2.1.7}$$

where j is the imaginary unit and $\mathcal{R}\{\cdot\}$ denotes the real part of a complex number. The last equation shows the change to polar coordinates, this transformation is appropriate if the waves are spherical, i.e. originating from the center of the coordinate system. In polar form we have $r = \sqrt{x^2 + y^2 + z^2}$. Substituting $u(\vec{x}, t)$ in the wave equation, the following is found

$$\left(\nabla^2 + k^2\right) U = 0. \tag{2.1.8}$$

Here, the wave number $k = 2\pi n v / c$ and the wavelength $\lambda = 2\pi / k$ are introduced. Eq. 2.1.8 is called the time-independent Helmholtz equation.

2.1.2. Huyghens principle

In optics, we assume the field is known at a plane at one location. This is used to describe the field at a different plane. That is, the complex field at $\vec{x}_0 = (x', y', 0)$ is given and the complex field at (x, y, z) is found using the methods of optics. Calculating this field given an earlier field is also known as diffraction. Using equation (2.1.6) would give us the influence to the complex field only from this point $(x', y', 0)$. However, to compute the full complex field we can use the Huygens-Fresnel principle. This principle states that every point on a wavefront acts as a point source and combining these sources together form the wavefront.

According to this principle, the complex field at (x, y, z) is given by summing the contributions from all point sources.

2.1.3. Rayleigh-Sommerfeld diffraction

A plate is placed on a plane $z = 0$. This plate has a rectangular aperture in the center. An aperture is an opening through which light can pass. The light not passing through the aperture is blocked.

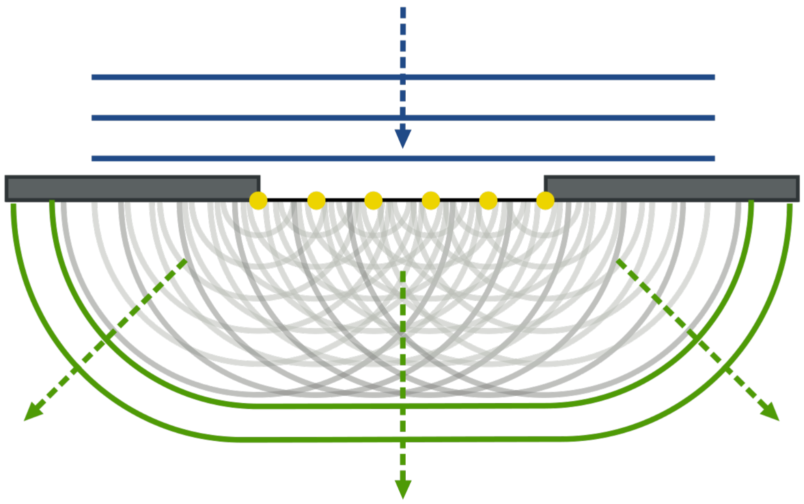


Figure 2.1: Refraction on an aperture according to the Huygens-Fresnel principle. The blue lines are the incoming light, the yellow dots represent the light in the aperture as a source according to the Huygens principle and the green lines represent the light behind the aperture. The aperture lies at $z = 0$. Reprinted from Wikimedia.

By the Huygens principle the whole of the aperture is considered a source. Taking infinitesimally small steps changes the summation of the sources into integration. Using equation (2.1.6) in polar form this results in the following

$$U(x, y, z) = \int_{\text{aperture}} U(x', y', 0)\frac{\exp(jkr)}{r}dx'dy', \tag{2.1.9}$$

where $r = \sqrt{(x-x')^2 + (y-y')^2 + z^2}$. As this is not rigorous, a better approach uses the method of Green's functions. Green's function is the impulse response solution to a differential equation. The convolution of Green's function and the source gives the solution to a differential equation. A short overview of the calculations is given here. Starting with Green's theorem,

$$\int_V U\nabla^2 G - G\nabla^2 U dV = \int_{\partial V} U\frac{\partial G}{\partial n} - G\frac{\partial U}{\partial n}ds, \tag{2.1.10}$$

which relates the integral of two complex valued functions $U(x)$ and $G(x)$ and their derivatives over the volume V with an integral over the closed boundary ∂V. It is assumed that the first and second partial derivatives exists and are continuous on ∂V. $\partial/\partial n$ denotes the partial derivative towards the outward normal direction. Let both U and G be equations that satisfy the Helmholtz equation (2.1.8), the following holds

$$\int_V U\nabla^2 G - G\nabla^2 U dV = -\int_V UGk^2 - GUk^2 dV = 0,$$

where the Helmholtz equation is substituted and the right-hand side is cancels out. Thus deriving

$$\int_{\partial V} U\frac{\partial G}{\partial n} - G\frac{\partial U}{\partial n} ds = 0, \tag{2.1.11}$$

choosing the area of integration in a specific manner this results in

$$U(x, y, z) = \frac{1}{4\pi}\int_{\text{aperture}} \frac{\partial U}{\partial n}G - U\frac{\partial G}{\partial n} ds, \tag{2.1.12}$$

For a more detailed derivation the reader is advised to look at [20] equation (3-24), where the derivation is done more formally. In deriving this equation three assumptions are made as [20] shows:

1. The scalar theory is valid
2. Both U and G satisfy the wave equation
3. The Sommerfeld radiation condition is satisfied.

The first assumptions implies that the theory to derive the wave equation holds. The third assumption implies that only outgoing waves are considered. In our optical setup, we make sure that this condition is met. Choosing Green's function G in a smart fashion is key. This is also the main difference between the Kirchhoff diffraction, another well known method which is not considered as it has certain disadvantages, and the Rayleigh-Sommerfeld diffraction. The choice for Green's function is

$$G(x, y, z) = \frac{\text{exp}(jkr)}{r} - \frac{\text{exp}(jk\tilde{r})}{\tilde{r}}, \tag{2.1.13}$$

where r and $\tilde{r}$ are the radii as depicted in Figure 2.2. If Green's function was generated only by a point source at P_0, only the first term would appear. However, this causes computational inaccuracies. Therefore, a second mirrored point source on the other side of the aperture is considered as seen in Figure 2.2.

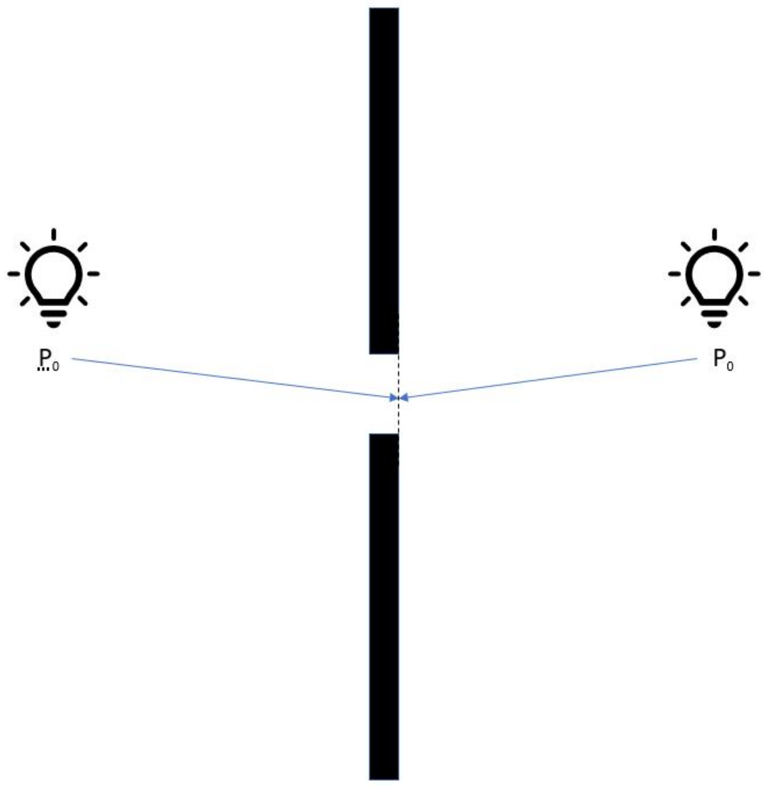


Figure 2.2: The setup of point sources for the derivation of Green's function. Recreated from [20], Figure 3.8.

The derivative of Green's function is given by

$$\frac{\partial G}{\partial n} = 2jk\cos(\theta)\frac{\exp(jkr)}{r} \tag{2.1.14}$$

$$= 2jk\frac{z}{r}\frac{\exp(jkr)}{r}, \tag{2.1.15}$$

where θ is the angle between the normal vector and the radius (r_{01} in Figure 2.2). Substituting this in the integral, the Rayleigh-Sommerfeld diffraction formula is found

$$U(x,y,z) = \frac{1}{j\lambda}\int_{\text{aperture}} U(x',y',0)\frac{z}{r}\frac{\exp(jkr)}{r}dx'dy'. \tag{2.1.16}$$

The above equation is similar to equation 2.1.7 apart from the factor z/r. Rayleigh-Sommerfeld diffraction is an attractive method to use because its validity is not influenced by the sampling distance, which will be seen in the next section is not always the case.

2.1.4. Fresnel and Fraunhofer approximation

The integral in equation 2.1.16 is difficult to calculate. Numerical integration would work. However, assumptions on the distance to the source (that is, assume z is of certain length), can simplify the integral. Taking z large and using Taylor approximations, we find the following

$$\begin{aligned} r &= \sqrt{(x-x')^2+(y-y')^2+z^2} \\ &= z\sqrt{1+\frac{(x-x')^2+(y-y')^2}{z}} \\ &\approx z+\frac{(x-x')^2+(y-y')^2}{2z} \quad \text{use } \sqrt{1+s}\approx 1+\frac{s}{2} \\ &= z+\frac{x^2+y^2}{2z}+\frac{x'^2+y'^2}{2z}-\frac{xx'+yy'}{z} \\ &\approx \frac{x^2+y^2}{2z}-\frac{xx'+yy'}{z} \end{aligned}$$

Using this approximation in the exponent and elsewhere approximating $r \approx z$, we find the following changes in the Rayleigh-Sommerfeld integral when excluding the last step, keeping the $\frac{x'^2+y'^2}{2z}$ terms. What remains is a Fourier transform as approximation:

$$\begin{aligned} U(x,y,z) \approx & \frac{\exp\left(jk\left(z+\frac{x^2+y^2}{2z}\right)\right)}{j\lambda z} \\ & \int_{\text{aperture}} \left\{U(x,y,0)\exp\left(jk\frac{x'^2+y'^2}{2z}\right)\right\}\exp\left(-jk\frac{xx'+yy'}{z}\right)dx'dy'. \end{aligned} \tag{2.1.17}$$

This approximation is called the Fresnel approximation or diffraction. The Fresnel diffraction is valid in the near field. For the Fresnel approximation to hold, the higher order terms of the phase must be negligible. This gives the condition $F\theta^2/4 \ll 1$, where F is the Fresnel number and θ is the maximum angle between the ray from the screen and the normal vector. Simplifying this is $z^3 \gg W^4/\lambda$, where W is the size of the aperture. Adding the last simplification in the approximation of the radius r, the calculating gives the following

$$U(x,y,z) \approx \frac{\exp\left(jk\left(z+\frac{x^2+y^2}{2z}\right)\right)}{j\lambda z}\int_{\text{aperture}} U(x,y,0)\exp\left(-jk\frac{xx'+yy'}{z}\right)dx'dy', \tag{2.1.18}$$

$$= \frac{\exp\left(jk\left(z+\frac{x^2+y^2}{2z}\right)\right)}{j\lambda z}\mathcal{F}\{U(z=0)\}\left(\frac{x}{\lambda z},\frac{y}{\lambda z}\right). \tag{2.1.19}$$

Thus assuming that z is large enough compared to the aperture, the integral changes into an even simpler Fourier transform (where $\mathcal{F}$ is the Fourier operator). This integral is valid in the far-field, which can be quantified as

$$z \gg \frac{W^2}{\lambda}, \tag{2.1.20}$$

where W is again the largest size of the diffracting aperture. This approximation is called the Fraunhofer approximation.

2.1.5. Adding a lens

In the above discussion, only apertures have been discussed and lenses were ignored. That is because in the limit of a thin lens, the lens only adds a phase shift to the field U. Let U_b and U_a denote the field before and after the lens, respectively. Then $U_a(x, y, z) = t(x, y, z)U_b(x, y, z)$, where $t(x, y, z) = \exp(jkn\Delta(x, y))$ is the phase shift induced by the lens. Here, n is the refractive index of the lens and $\Delta(x, y)$ is the thickness at the specific x, y-coordinates. In [20] an overview is given with the above in mind and considering multiple lenses. However, as we consider freeform lenses, $\Delta(x, y)$ denotes an arbitrary function for now. For an optical system it is also interesting what happens with multiple lenses behind each other. To simulate light through multiple lenses, the field needs to be calculated directly in front of the second (or next) lens to find the field behind the lens. Thus diffraction is done multiple times.

2.1.6. Coherency of light

Coherency is the manner in which light interferes with other light. Completely coherent light could interfere with other light, this happens when the temporal and spatial correlation between the light is high. Incoherent light does not interfere with other light. Coherent light from two sources is added together as if it were light from the same source as with the Huygens-Fresnel principle. With coherent light we add the fields of the light to each other during propagation. Incoherent light on the other hand does not interfere and we therefore cannot add the complex fields but the intensity. Simply said; with incoherent light the intensity profile after diffraction are added to each other, with coherent light the complex fields are added to each other. Within the scope of this thesis we will focus on incoherent light.

2.2. Numerical optics

In this chapter we look at the numerical methods used to implement a physically correct simulation of light. Firstly, the simulation of sources is discussed. Secondly, propagation with multiple methods is discussed. What methods are favourable and what restrictions should be taken into account. Lastly, aspects of lenses are discussed. Examples of lenses are mentioned, which are used to generate various intensity profiles.

2.2.1. Sources

In this section, simulating the complex field of a source is elaborated on. Three different sources are considered: the plane wave, the point source and the Gaussian beam. Lastly, this section describes how a source changes if it comes into our system at an angle.

Plane wave

A plane wave consists of a constant phase and a constant amplitude over the entire sampling domain. Thus the formula for its field is given by

$$u_{\text{plane wave}}(\vec{x}) = A \times e^{i \times \phi},$$

where A is the (spatially constant) amplitude of the field and ϕ is the (spatially constant) phase of the field.

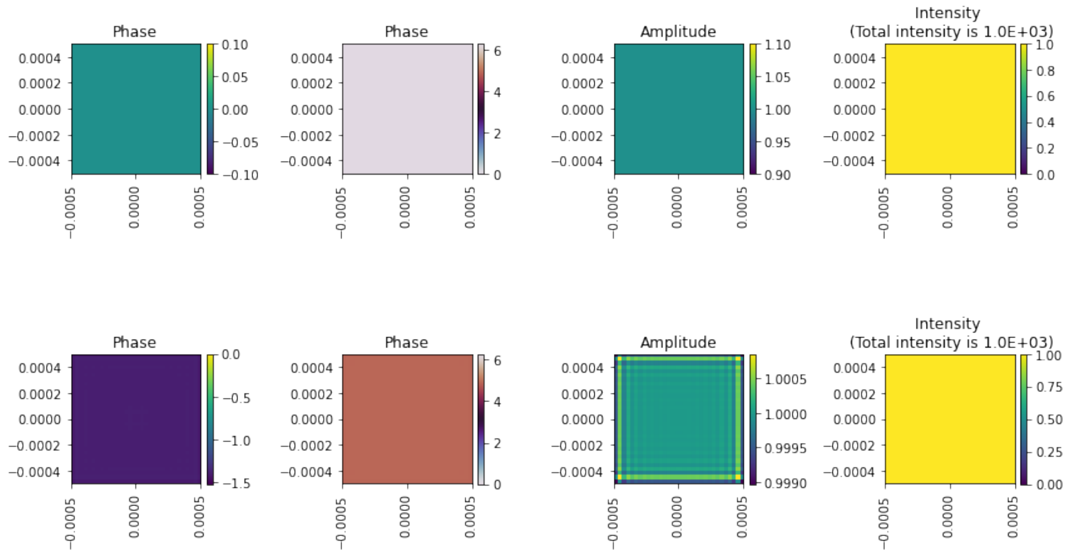


Figure 2.3: An example of a plane wave before and after propagation.

The plane wave seen in Figure 2.3 is as expected. On the border aliasing can be seen in the amplitude after propagation, this is caused by the low resolution.

Point source

Another source with properties similar to a plane wave at specific distances is the point source. Sources like a LED light could be though of like a point source. A point source is the light of a single pixel propagated in every direction. The point source is therefore a symmetric round shape. The field at position $\vec{x}$ due to a point source at position $\vec{x}_0$ is given by

$$u_{\text{point source}}(\vec{x}, \vec{x}_0) = \frac{\exp(ikR)}{4\pi R}, \tag{2.2.1}$$

where k is the wave number and $R = \sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2}$ is the distance between $\vec{x}$ and $\vec{x}_0$

Equation 2.2.1 is a fundamental solution to the Helmholtz equation

$$\nabla^2 u + k^2 u = -\delta(\vec{x} - \vec{x}_0).$$

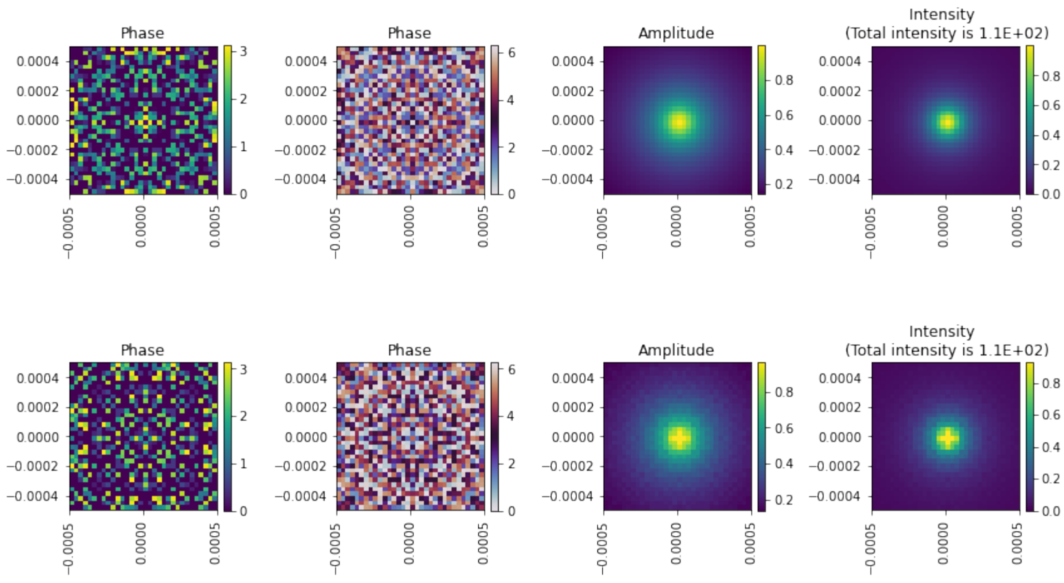


Figure 2.4: An example of a point source before and after propagation.

As can be seen in Figure 2.4, the point source has an amplitude as expected, around one point. The phase has multiple axis of symmetry through the centre as expected. The phase is however not completely symmetrical as we would expect as a unit impulse is also completely symmetrical.

Gaussian beam

Another common source is the Gaussian beam and has similar properties to a laser beam. It is named Gaussian as its intensity is in the form of a Gaussian function. The Gaussian beam is dependent on a parameter w_0 called the waist radius, which is related to the beam size at the point $z = 0$. The complex field of a Gaussian beam is given by

$$u_{\text{Gaussian beam}}(\vec{x}, \vec{x}_0) = \frac{w_0}{w(z)} \exp\left(\frac{-r^2}{w(z)^2}\right) \exp\left(ikz + ik\frac{r^2}{2R(z)} - i\psi(z)\right),$$

where the following holds:

- $r = \sqrt{(x-x_0)^2 + (y-y_0)^2}$
- k is the wavelength

- $w(z) = w_0\sqrt{1+\left(\frac{z}{z_R}\right)^2}$ is the radius where the total intensity has decreased by $1/e^2$
- $z_R = \frac{\pi w_0^2}{k}$
- $R(z) = z\left(1+\left(\frac{z_R}{z}\right)^2\right)$ is the radius of curvature
- $\psi(z) = \arctan\left(\frac{z}{z_R}\right)$ is also known as the Gouy phase. This is a correction for the phase in the near-field.

Note that w_0 should be large enough compared to the wavelength to be physically correct.

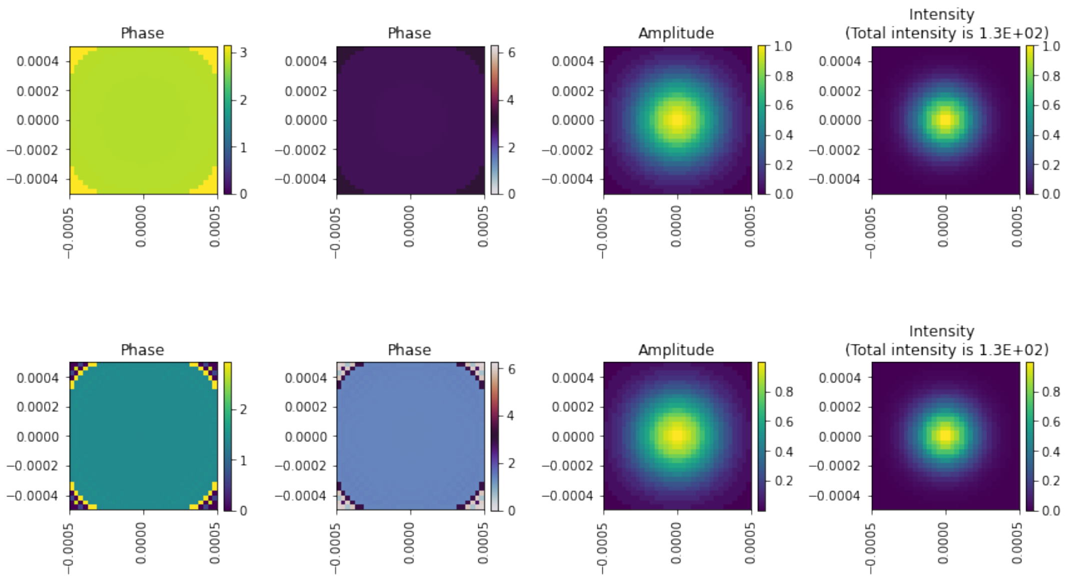


Figure 2.5: An example of a Gaussian beam before and after propagation.

From Figure 2.5, the intensity, similar to a point source, is around the center. The phase in this source is zero, therefore the intensity does not diverge of our sampling plane. This is similar behaviour as expected from a collimated laser.

Other numerical aspects of sources

To simulate a wave incident at an angle a linear phase term $u_{\text{change angle}}$ is added.

$$u_{\text{change angle}}(\vec{x},\theta,\psi) = e^{ik(x\sin\theta\sin\psi + y\cos\theta\sin\psi)},$$

such that the new field with an angle becomes

$$u_{\text{new}}(\vec{x},\theta,\psi) = u_{\text{change angle}}(\vec{x},\theta,\psi)u_{\text{source}}(\vec{x}).$$

Here, u_{source} denotes the old source and θ and ψ denote the change of angle.

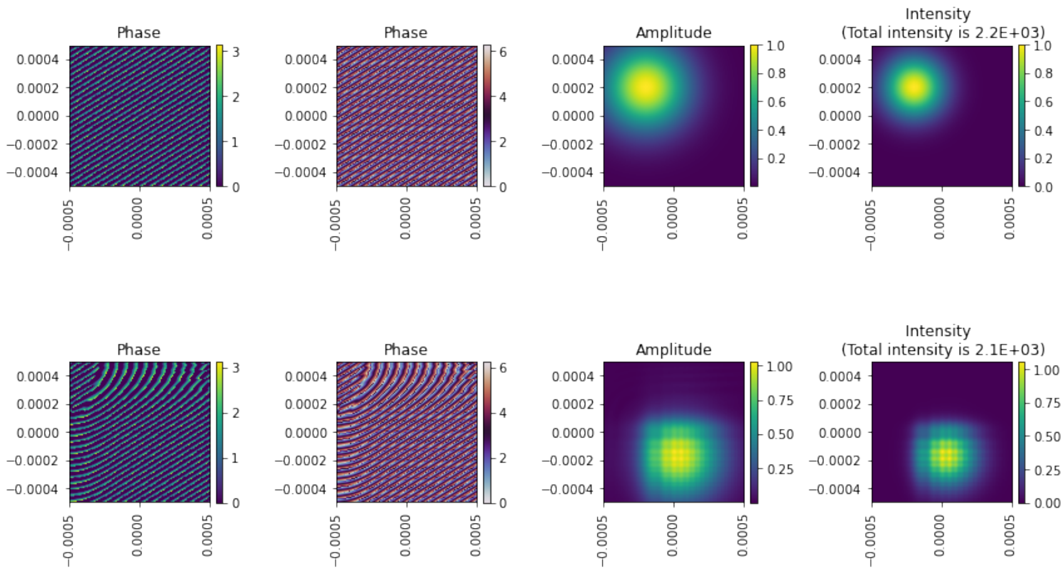


Figure 2.6: An example of a Gaussian beam at an angle before and after propagation. To make this feasible we had to increase the resolution and increase the propagation distance compared to previous images. Note also that due to the source field touching the sides the light has a square shape in the upper left corner.

In Figure 2.6, a clear shift from the top corner to the middle is seen. The intensity pattern itself does not change, only the position. The top and left sides of the Gaussian beam are straight due to effects from the numerical propagation. For point sources one would move the source in the xy-plane to create an angle.

2.2.2. Lenses

The methods to design a lens can be divided in two classes: lenses which are described per pixel and parametrizable lenses. In this section we show propagation results with both these types of lenses.

Diffractive Optical Elements

Diffractive optical elements (DOE) are the most straight-forward lenses. DOEs are not surfaces described by continuous functions, but the lens thickness or phase change is determined per pixel. This comes at the cost of having certain speckle effects that continuous lenses do not have. However, DOEs also have more freedom as they are not restricted to be continuous and are therefore able to generate more complex patterns.

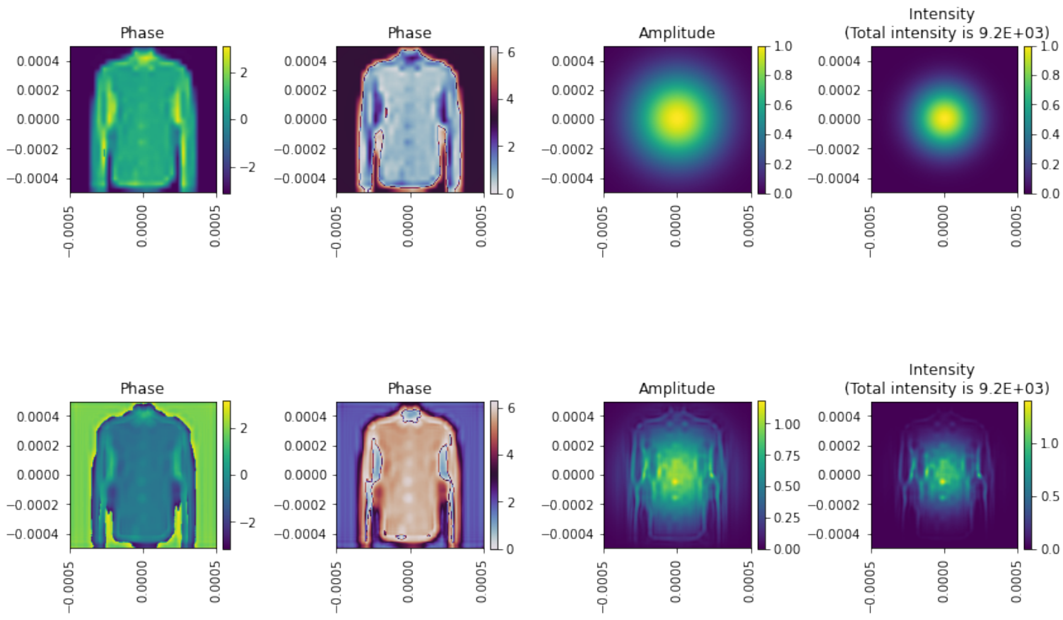


Figure 2.7: An image of a shirt from the FMNIST dataset used as a lens. The top part shows the field before propagation which is a combination of the Gaussian beam and a phase from the FMNIST dataset. The bottom images are of the propagated field.

In Figure 2.7, a system with a Gaussian beam is simulated, which is then given a phase change with the shirt from the FMNIST as DOE. The FMNIST image was first scaled such that pixel contrast is between a range of 0 and 2π. In the propagated field we clearly see that the intensity resembles the shirt. This shows that the shape of the lens and the intensity it generates are connected.

Parametric lenses

To make the optimization more feasible, a restriction could be made to lenses described by parametric surfaces. Such a surface can be a NURBS surface or a radial basis function. Lenses designed with continuous parametric surfaces have the advantage that they are less prone to speckle effects.

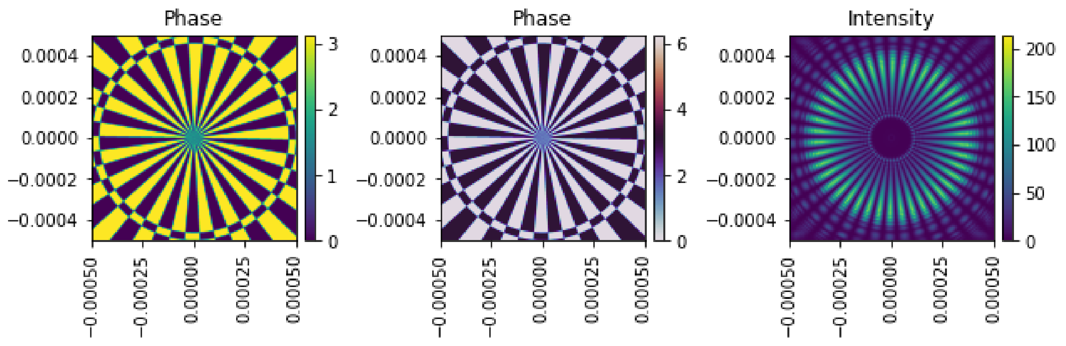


Figure 2.8: Radial basis polynomial before and after diffraction.

In Figure 2.8, a radial basis surface is used to describe the surface of a lens. This generates an intensity similar to the lens.

We also consider B-spline lenses (We write B-spline lenses instead of lenses modelled by B-splines). We discuss B-splines and other parametrics surfaces in section 2.4. Other B-spline lenses are also considered in the results in chapter 4.

2.2.3. Diffraction

Diffraction is the process of propagating the light from directly behind the lens to the sampling plane. In this section the fast Fourier transform (FFT), two methods of diffraction and improvements to make the system more well-behaved are described.

Fast Fourier Transform

The Fast Fourier Transform (FFT) lies at the heart of our diffraction methods and we improve our diffraction methods by improving on the weaknesses of the FFT. Therefore, we give a short introduction to the FFT.

To numerically calculate the Fourier transform of a function

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi\xi x} dx.$$

The integral can be replaced by a sum also known as the discrete Fourier transform (DFT)

$$\hat{f}(\xi) = \sum_{n=0}^{N-1} f(x_n) e^{-i2\pi\frac{kn}{N}}.$$

As a computer can only handle finite values this makes our interval of calculation limited, thus affecting analysis close to or at the border of our input. A naive implementation would have computational costs of $\mathcal{O}(N^2)$, a better implementation named the fast Fourier transform takes $\mathcal{O}(N \log N)$. In the FFT the above sum is split in the terms with even n and odd n and these terms can be treated as two separate DFTs, recursively applying the same procedure to these two DFTs gives us an efficient algorithm. Due to this recursive application and splitting the domain in two, the FFT is most efficient if N is a power of two as for $2^{i-1} < N < 2^i$ there will be i recursive steps.

Fraunhofer propagation

As previously seen in section 2.1.4 Fraunhofer propagation is a Fourier transform multiplied by a complex term

$$U(\vec{x}) = \frac{\exp(ik(z + r^2/2z)}{i\lambda z} \mathcal{F}\{U_0\}\left(\frac{x}{\lambda z}, \frac{y}{\lambda z}\right),$$

where $r^2 = x^2 + y^2$. This should hold if $z \gg \frac{W^2}{\lambda}$. In this formulation $\mathcal{F}$ can be replaced by the FFT. From the above equation it follows that the computational costs of the Fraunhofer diffraction is dominated by the computational costs of one FFT.

Rayleigh-Sommerfeld propagation

As mentioned, the Fraunhofer diffraction is not accurate enough when not in the far field. Especially when multiple lenses are considered there is need for a more accurate method, as the propagation distance between the lenses is too short for methods like Fraunhofer. Numerically approximating the Rayleigh-Sommerfeld integral is such a method. We can do this in two ways.

To solve the Rayleigh-Sommerfeld integral, the angular spectrum (AS) method is introduced as in [61]. Revisiting the Rayleigh-Sommerfeld (RS) integral in equation 2.1.16

$$U(x,y,z) = \frac{1}{j\lambda}\int_{\text{aperture}} U(x',y',0)\frac{z}{r}\frac{\text{exp}(jkr)}{r}dx'dy'.$$

Let $A(\alpha,\beta,z) = \mathcal{F}\{U(\cdot,\cdot,z)\}$ be the Fourier transformation of the light field at distance z. Propagation over a distance z then admits the following reformulation

$$A(\alpha,\beta,z) = A(\alpha,\beta,0)G(\alpha,\beta,z), \tag{2.2.2}$$

where $G(\alpha,\beta,z) = \text{exp}\left(jz\sqrt{k^2-\alpha^2-\beta^2}\right)$ is the optical transfer function. The function G is a solution to the Helmholtz equation. Let the inverse Fourier transform is denoted by $\mathcal{F}^{-1}$, respectively. We then find that

$$\begin{aligned} U(x,y,z) &= \mathcal{F}^{-1}\{A(\alpha,\beta,z)\}, && (2.2.3)\\ &= \mathcal{F}^{-1}\{A(\alpha,\beta,0)G(\alpha,\beta,z)\}, && (2.2.4)\\ &= \mathcal{F}^{-1}\{\mathcal{F}\{U(x,y,0)\}G(\alpha,\beta,z)\}. && (2.2.5)\end{aligned}$$

Numerically, a fast Fourier transform (FFT) is done instead of a real one and thus the two terms in the inverse Fourier transform are multiplied element-wise as describe in [61].

Another method to solve the RS integral is to numerically integrate the RS integral. Following the methods of [61] here. Applying the inverse Fourier transform to our optical transfer function $\mathcal{F}^{-1}\{G(\alpha,\beta,z)\} = g(x,y,z) = \frac{1}{2\pi}\frac{\text{exp}(jkr)}{r}\left(\frac{1}{r}-jk\right)$, there are similarities between the original RS integral and these terms. Continuing, this gives

$$U(x,y,z) = \int_{\text{aperture}} U(x',y',0)g(x-x',y-y',z)dx'dy'. \tag{2.2.6}$$

The above equation is a convolution integral, which was seen with Green's function as well. Following [61] and choosing the correct (zero-padded) matrices, this can again be solved with aid of fast (inverse) Fourier transforms as follows

$$U = \text{IFFT}\{\text{FFT}\{U(x',y',0)\}\text{FFT}\{g(x',y',z)\}\}\Delta x'\Delta y'. \tag{2.2.7}$$

Summarizing, two methods to solve the Rayleigh-Sommerfeld integral are described. One based on the angular spectrum method and another based on numerical integration. Both methods consist of sequentially applying the fast Fourier transform and have similar computational costs according to [61]. Furthermore, both methods have their benefits and drawbacks. As the angular spectrum method seems to be more popular and gives better results in our implementation, we choose this method. The angular spectrum method has computational costs dominated by the FFT and its inverse, this is therefore double the computational costs of the Fraunhofer diffraction.

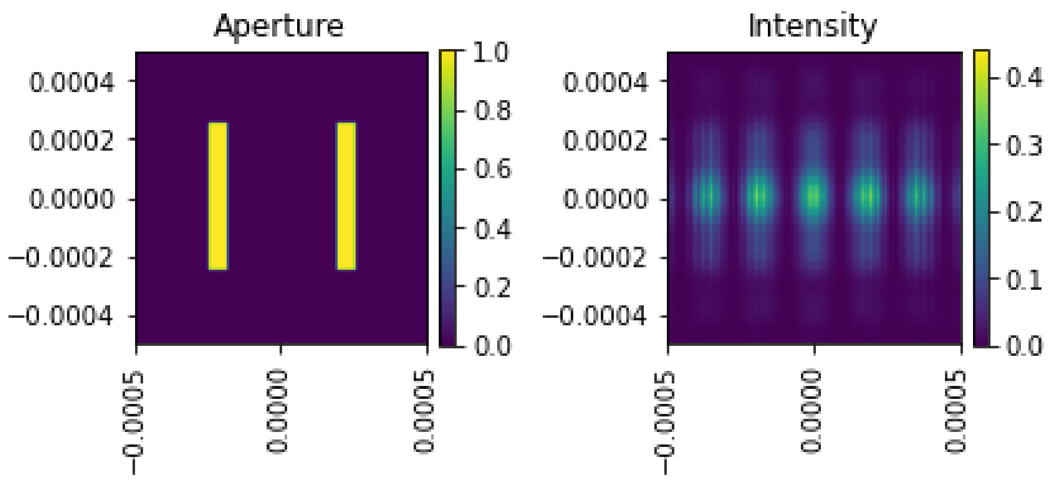


Figure 2.9: An example of a double slit before and after propagation with Rayleigh-Sommerfeld diffraction.

In Figure 2.9 the double slit aperture and light after propagation are shown. The double slit is a famous experiment were an uninformed person would think it would create two rectangles on the sample plane, however multiple rectangle like intensities should appear on the sample plane. The intensity is as expected, multiple intensity peaks appear on the sample plane.

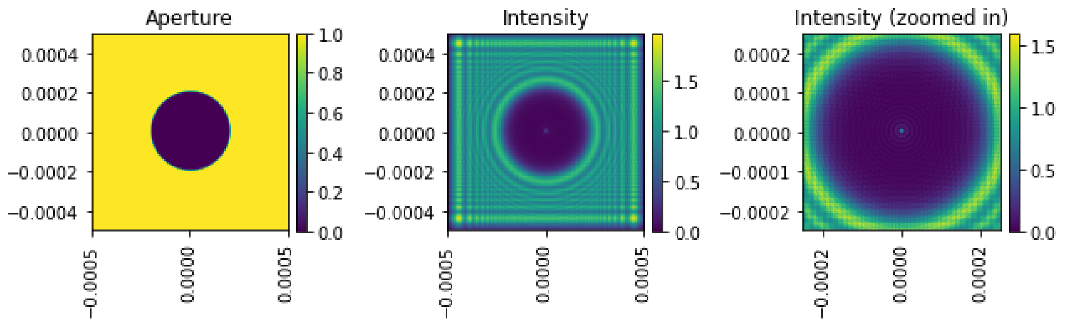


Figure 2.10: An example of a Poisson spot before and after propagation with Rayleigh-Sommerfeld diffraction. What is remarkable to the Poisson spot is that even though light in the middle is completely blocked it still forms a small spot in the middle. The Poisson spot is named after Simeon Denis Poisson who ridiculed the propagation theory of Fresnel, as he showed that with Fresnel's calculation such a spot should appear and this should be nonsense. As a response Fresnel showed that the spot indeed appeared in physical experiments validating his theory.

In Figure 2.10, the Poisson spot is generated. For the Poisson spot we expect the circular block to be seen on the sample plane as well as a small intensity peak in the middle of the block. We see both these things. We also see that the intensity shows a discontinuous pattern, this is caused by aliasing.

Padding and aliasing

As we use the fast Fourier transform in this work, we have to take into account numerical artefacts. To combat effects on the edge of our images all fields are padded with zeros before propagation in such a way that the total dimension is a power of two for efficiency.

Furthermore, we also experience aliasing when using certain sources, with a Gaussian beam for example. Especially, when there is a large difference in intensity between parts of the field. A way to solve this problem is to increase the resolution. In our settings a resolution of $N = 2^{10} = 1024$ generates correct intensities, however this brings a clear computational costs.

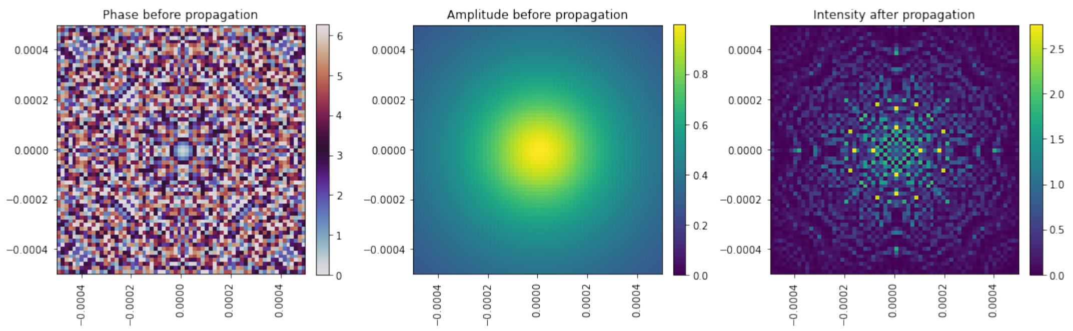


Figure 2.11: A propagated point source with a resolution of 64×64

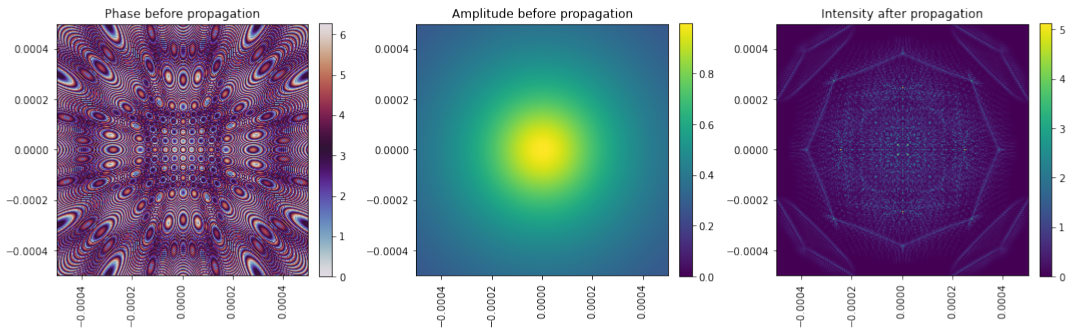


Figure 2.12: A propagated point source with a resolution of 256×256

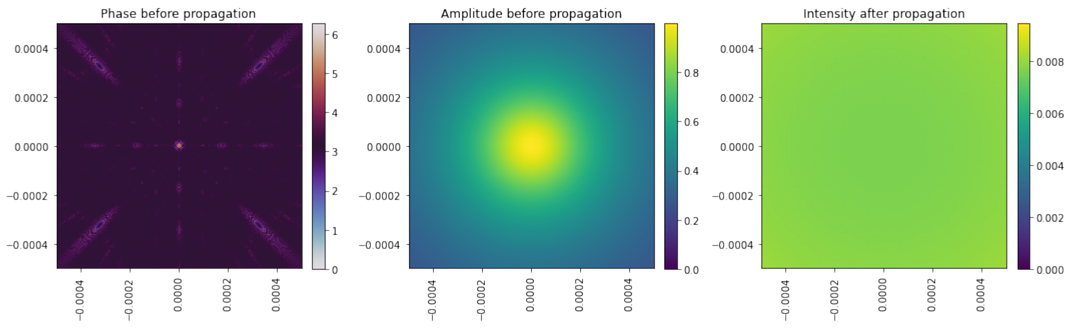


Figure 2.13: A propagated point source with a resolution of 2048×2048

The above pictures show the effect of aliasing. Propagating over a distance of 1cm should result in an intensity similar to a plane wave, which is only seen in the last figure. In the other two intensities we can see two different patterns in the intensity.

2.3. Deep Learning

In this section deep learning is discussed. The section starts with the foundation of neural networks and extends this with different types of layers, such as convolutional, pooling and residual layers. Afterwards, different network architecture combining these layers are discussed. The section is concluded by elaborating on the optimization methods used to optimize optical systems and neural networks. The theory in this section is based on [29].

2.3.1. Neural networks

Neural networks are universal function approximators, allowing a neural network to approximate any function, as long as the network is large enough ([24]). The input of a neural network can be a scalar, a vector or another type of tensor, denoted by X and the output vector is denoted as y, this can also be a scalar, a vector or a tensor.

The input is multiplied by a weight W_1 and a bias b_1 is added, which should have the correct dimension. This results in $W_1X + b_1$, which is a linear function. To let the network express complex (non-linear) relations between variables a non-linear transformation is done. Thus a one-layer network is

$$y = \sigma(W_1X + b_1), \tag{2.3.1}$$

where σ is a non-linear function, also called an activation function. In literature, the Sigmoid function $\sigma(x) = 1/(1+e^{-x})$ or the rectified linear unit (ReLU) $\sigma(x) = \max\{0, x\}$ are commonly used. The activation function is applied element-wise.The Rectified Linear Unit (ReLU) is chosen as it mimics the way neurons in a brain activate on signals.

In a network with multiple layers, the output of the previous layer becomes the input of the next layer. Thus a two-layer network becomes

$$y = \sigma(W_2\sigma(W_1X + b_1) + b_2) \tag{2.3.2}$$

and a general neural network becomes

$$y = \sigma(W_n\sigma(W_{n-1}\sigma(...) + b_{n-1}) + b_n). \tag{2.3.3}$$

Given data pairs for input and output this setup allows us to find weights and biases to approximate any functional relationship between input and output variables. A loss function, such as the mean squared error (MSE), is minimized to find the weights and biases that best fit the data. Let (X_i, y_i) for $i = 1, ..., N$ be the data, then the MSE is given by $MSE(X_{1:N}, y_{1:N}) = \frac{1}{N}\sum_{i=1}^{N} ||y_i - \hat{y}(X_i)||_2^2$. Here, $\hat{y}(\cdot)$ is the output of the network given the parameters as in Equation (2.3.3).

Even though neural networks are universal function approximators, finding the correct weights and biases is not trivial. Optimizing a single layer network to have a 128×128 image as input and a full weight matrix already has over 16000 optimizable parameters. Therefore, we introduce several application specific layers that have inductive biases. Inductive biases are properties some layers should have based on the type of input, such as translational invariance for images.

2.3.2. Convolutional layers

If all the parameters in the weight matrix W are optimized for, that layer is called fully connected (FC) as all input values are connected to the output values. Thic could cause many inefficient connections due to an underlying structure in the data. An example is image analysis, in imaging the information in pixels mainly relates to the pixels around themselves. Thus the network should be restricted to have this inductive bias. Therefore, convolutional layers were introduced.

Convolutional layers consist of multiple discrete convolutions. A convolution of a function f with a function g is denoted by

$$f * g[n] = \sum_m f[m]g[n-m], \tag{2.3.4}$$

where the sum is taken over the entire discrete domain, the rectangular brackets denote that we are working on a discrete domain.

An example of a convolution would be to take $g[-1] = -1$, $g[0] = 2$, $g[1] = -1$ and zero elsewhere. We then have $\frac{1}{n^2} f * g[i] = \frac{-f[i-1]+2f[i]-f[i+1]}{n^2}$, which is the finite difference approximation of the second derivative. This is one example of a convolution. The output consist of a weighted summation of neighboring pixels. If $g[i]$ is non-zero over the entire domain, we are essentially back to the fully connected layer.

The kernel size of a convolution denotes the non-zero amount of values in the kernel g. A kernel size of three means that we take into account the left and right neighbour. In deep learning convolutional layers are used in image recognition as they are able to use the structure of an image, where pixels lying next to each other should be connected. On top of this, convolutional layers are invariant to translations in images, which is exploited in object recognition. To work on images we could use a two dimensional convolution and to work on three dimensional volume images, we can use three dimensional convolutions.

The input of a convolution is an image or tensor and the output is a tensor. The input image or tensor usually consists of three dimension, height width and amount of channels. In the input the channels often relate to color, RGB images are three channels. Later in the network each channel describes a different feature. The input in a two dimensional convolution on an image would be *batch size* $\times$ *height* $\times$ *width* $\times$ *# of channels in input* and the output would be *batch size* $\times$ *height* $\times$ *width* $\times$ *# of channels for output*. Let us denote the number of input channels by c_{in} and the number of output channels by c_{out}. Then a single convolutional layer has $c_{\text{out}} \times (c_{\text{in}} \times$ kernel width $\times$ kernel height $+ 1)$ number of parameters, as each output needs a bias and the convolution parameters. The 1D and 3D convolution parameters work similarly.

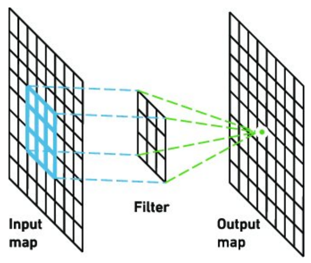


Figure 2.14: A visual representation of a convolutional layer taken from [70].

2.3.3. Upsampling layers

The inverse of a convolutional layer or a transposed convolutional layer is known as an upsampling layer. These layers act like a convolutional layer but increase the dimension of our image instead of decrease.

Upsampling multiplies every pixel of the input image with a kernel and then places the outcome in an output image. Then moving to the next pixel and depending the stride the output of this next pixel will be added on the overlapping pixels in the output image. In Figure 2.15 a visual example of an upsampling layer is given.

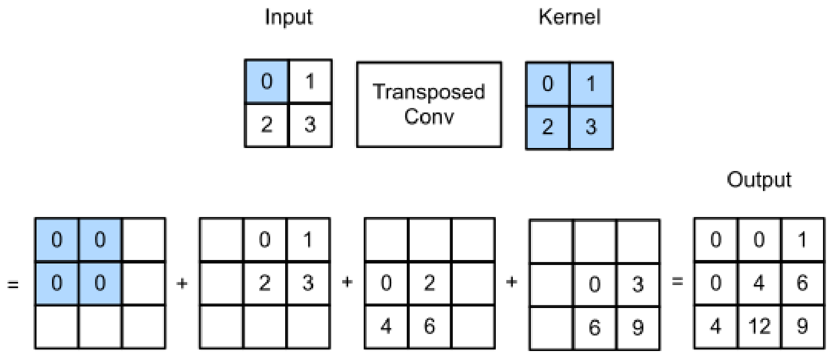


Figure 2.15: An example of a transposed convolution with stride 1. The effects of every pixel in the output image are added if they overlap due to the chosen stride. Image taken from d2l.ai.

2.3.4. Max-pooling layers

In neural networks for images, the input is often larger than the output. In a classification case such as handwritten image set MNIST, we have an input size of $28 \times 28 = 784$ and 10 possible output numbers (the probability the input resembles a particular number). In convolutional layers the dimension can be reduced by changing the number of output channels or by moving the convolutional kernel multiple pixels at a time. To capture better effect, average-pooling and max-pooling layers are introduced. Similar to a convolution layer, these layers look at the input tensor and consider a certain kernel, however the value this operation outputs is either the maximum value in the kernel or the average value of all kernel items. Furthermore, were a convolutional layer moves per pixel such that when moving the kernel one pixel, the previous pixel still has an effect on the output, the pooling layers consider each pixel only one time. A 2×2 kernel thus reduces the amount of values after a layer by a factor of four. An average pooling layer can also be mimicked by a convolutional layer with fixed weights and a fixed stride (the step-size of the kernel).

2.3.5. Back propagation

As we will see in Section 2.3.10, our optimization methods require gradients. These gradient are calculated through back propagation. This is best understood by a clear example. Let us define our network to be a one layer network as in equation (2.3.1), for simplicity the data, weights and biases are taken to be one-dimensional. Let us take the sigmoid function as the activation function $\sigma(\cdot)$ and the loss is the MSE. As for multiple data points we would sum the losses, we consider $(\hat{X}, \hat{y})$ to be a single data point. Then to derive the derivative of the loss with respect to the weight the weight

$W_1 \in \mathbb{R}$ is

$$\begin{aligned}\frac{\partial}{\partial W_1}\frac{1}{2}||\sigma\left(W_1\hat{X}+b_1\right)-\hat{y}||^2 &= \left(\sigma\left(W_1\hat{X}+b_1\right)-\hat{y}\right)\frac{\partial\sigma\left(W_1\hat{X}+b_1\right)}{\partial W_1}\\ &= \left(\sigma\left(\cdot\right)-\hat{y}\right)\left(\sigma(\cdot)(1-\sigma(\cdot))\right)\frac{\partial W_1\hat{X}+b_1}{\partial W_1}\\ &= \left(\sigma\left(\cdot\right)-\hat{y}\right)\left(\sigma(\cdot)(1-\sigma(\cdot))\right)\hat{X}.\end{aligned}$$

This shows that the chain rule is important in back propagation. Moreover, from the above work we could quickly also find the derivative with respect to b_1 as only the last step changes. This shows the power of back propagation and by saving the gradient at each step, we can quickly recover the gradient for multiple parameters which need to be optimized.

Calculating the gradient is done automatically in packages such as Facebook's PyTorch, in such packages the computer keeps track of the computational graph, as each chain rule application can be seen as a node in a graph. Note that this procedure also has drawbacks. For example, using the sigmoid function as the activation function and W_1X+b_1 going to infinity, the sigmoid function approaches one. Therefore, the gradient is $\sigma(\cdot)(1-\sigma(\cdot))\approx 0$ and thus the whole derivative becomes approximately zero. As we will see later this in turn means that our optimization step vanishes. This is called the vanishing gradient problem and can be solved by multiple techniques, which we discuss in Section 2.3.7.

2.3.6. Batch normalization

A technique that makes back propagation more stable is called batch normalization. When training a network with multiple layers there is a technical artifact called internal co-variate shift. As weights in the first few layers of the network get updated, the distribution of their output values changes such that the input of the deeper layer also has a different distribution. The deeper layers chase a moving target when optimizing. To standardize the distribution of inputs in each layer we apply the so-called batch normalization. This concept was introduced to the artificial intelligence world in [28]. The application of batch normalization layers is quite simple. We take the mean and the variance of the output of the last layer and subtract the mean and divide by the variance to get the standardized input in the next layer. The distribution of our input values has mean zero and variance one. Therefore, the optimization no longer chases a moving target.

2.3.7. Residual connections

A method to combat vanishing gradients mainly used in convolutional networks is called a residual connection. Residual connections were first introduced in [22]. In a residual layer, the input of the layer would be added to the output of the layer. Thus if the input of a layer is the output of a convolutional layer, the input would be a tensor of dimension $50\times100\times100$ in channels, width, height format. Assume a convolutional layer with output $25\times100\times100$ inside our residual block, then the output after adding the residual is $75\times100\times100$ as all the channels of the input are added to the channels of the output.

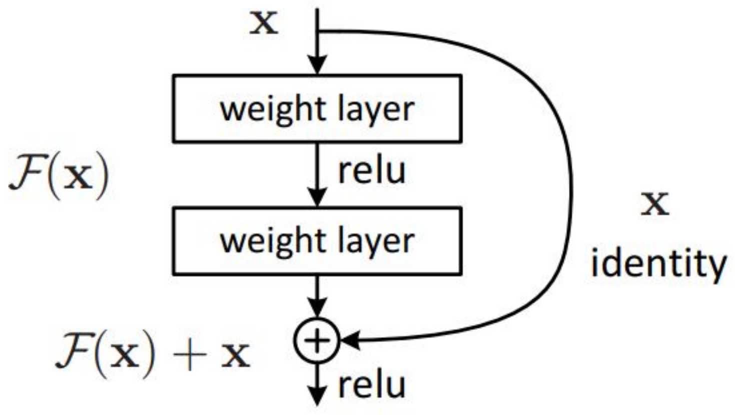

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.16: A residual layer.

The main benefit of this approach is that even though an activation function is applied on the output of our layer, the gradient information does not vanish as all information also skips this activation function and thus the gradient flows through to the next layer. The idea behind this method is also that the network can learn the change between input and output instead of the whole transformation.

2.3.8. Physics informed deep learning

A recent application in the field of both deep learning and physics is known as physics informed deep learning. The idea was first introduced in [50]. In general physics are described by partial differential equations, these equations are often modeled by solving them on a grid, so derivatives can be approximated on the grid. However, as illustrated above, the design of neural networks makes it possible to calculate their derivatives with respect to the weights of the network. In the same manner, derivatives of a neural network can also be calculated with respect to the input. In the simplest setting a physics informed neural network tries to solve a boundary value problem $\mathcal{L}p(t) = f(t)$ for $p(t)$, where $\mathcal{L}$ is a differential operator and $f(t)$ is source. The loss of the network is $\sum_i ||\mathcal{L}p(t_i) - f(t_i)||^2$ and minimize this. The function should be close to the solution to our boundary value problem. Note that we still have to evaluate the loss on a finite amount of grid points, but the main feature is that the derivatives are no longer approximated. Moreover, it is also possible to combine this approach with data and find unknown parameters in the equation simultaneously with solving the equation.

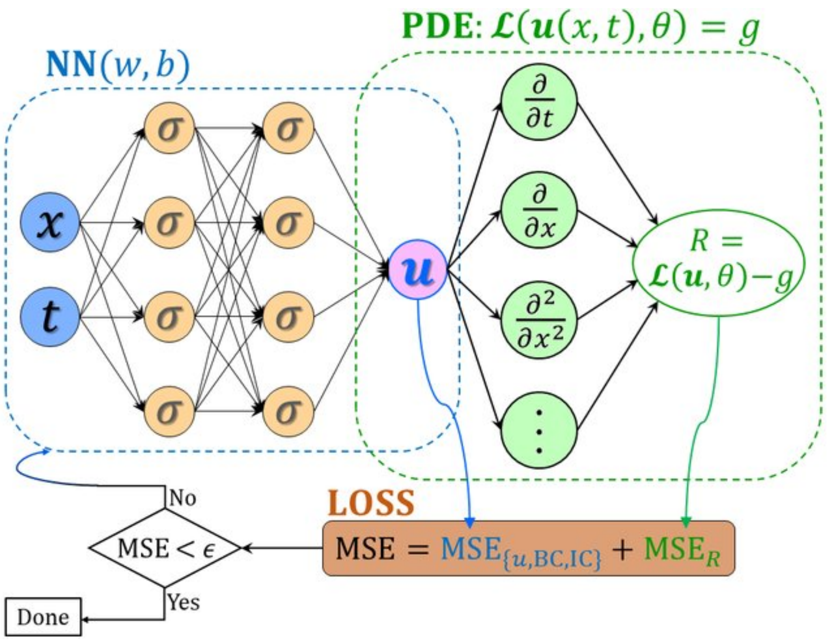

PDE: $\mathcal{L}(u(x,t),\theta) = g$
NN(w,b)
x
t
σ
u
$\frac{\partial}{\partial t}$
$\frac{\partial}{\partial x}$
$\frac{\partial^2}{\partial x^2}$
$\vdots$
$R = \mathcal{L}(u,\theta) - g$
LOSS
MSE $= \text{MSE}_{\{u,\text{BC},\text{IC}\}} + \text{MSE}_R$
MSE $< \epsilon$
No
Yes
Done

Figure 2.17: A Physics informed neural network architecture as found in [37]. Note that the given MSE consists of error terms on the PDE, the initial conditions (IC) and the boundary conditions (BC).

Recent advancements in physics informed deep learning have shown that it is also possible to learn operator between function spaces and even Banach spaces. In [34], it is shown that neural networks are able to learn entire families of PDEs and outperform other methods. In [65] DeepONets are introduced which learn operators based on data and underlying physics.

2.3.9. Network Architectures

In the previous chapter, we have introduced multiple different kinds of layers that are able to give information to the next hidden or output layer in a different way. These layers by themselves do not mean much, however when combined in a specific manner they prove to be versatile. We discuss tricks that work in practice. We discuss three types of convolutional neural networks, that is, network structures using convolutions. We start with a standard convolutional neural network (CNN). After this we discuss residual convolutional neural networks and lastly we discuss the state-of-the-art network we use in combination with our optical system: the UNet.

Convolutional Neural Networks

The name convolutional neural network can be given to any neural network that uses convolutions anywhere in its architecture. Convolutional neural networks have become popular since they won the ImageNet competition (https://www.image-net.org/). A typical example of a convolutional neural network is given in Figure 2.18

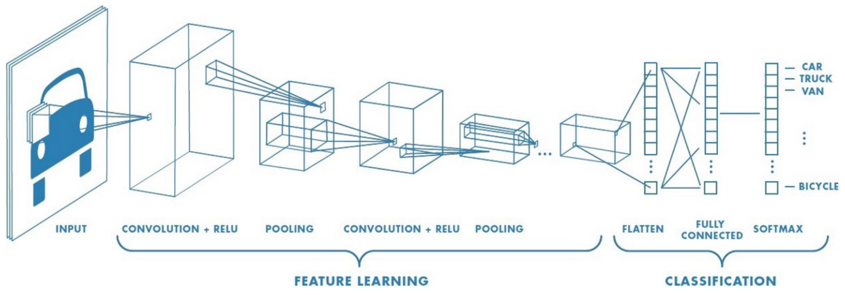


Figure 2.18: The architecture of a typical convolutional neural network taken from towardsdatascience.com.

In neural networks the structure as seen in Figure 2.18 is very common, where the first part of the network decreases the dimension of the input in a limited amount of features and the second part uses these features for classifications or other output. This structure is called an encoder-decoder structure, where the first part is called the encoder and the second part the decoder. For classification a fully connected output is used, whereas the encoder is mainly dependent on the input data.

In Figure 2.18, the typical layers used in a convolutional neural network are also seen. As discussed convolutional layers are used to find features in the image. Pooling layers are then used to aggregrate these features so that only important ones remain and the dimension of the feature space is reduced. Finally, a non-linearity like a ReLU function is used to add non-linearity to the network. This allows to estimate various complex non-linear relations. A batch-normalization layer is added before or after a convolution to help with normalizing the data at every step of the network.

ResNet

To get the current state-of-the-art convolutional neural networks we have to add residual connections to our list of techniques we use so far. Residual connections append the input of the layer to the output.

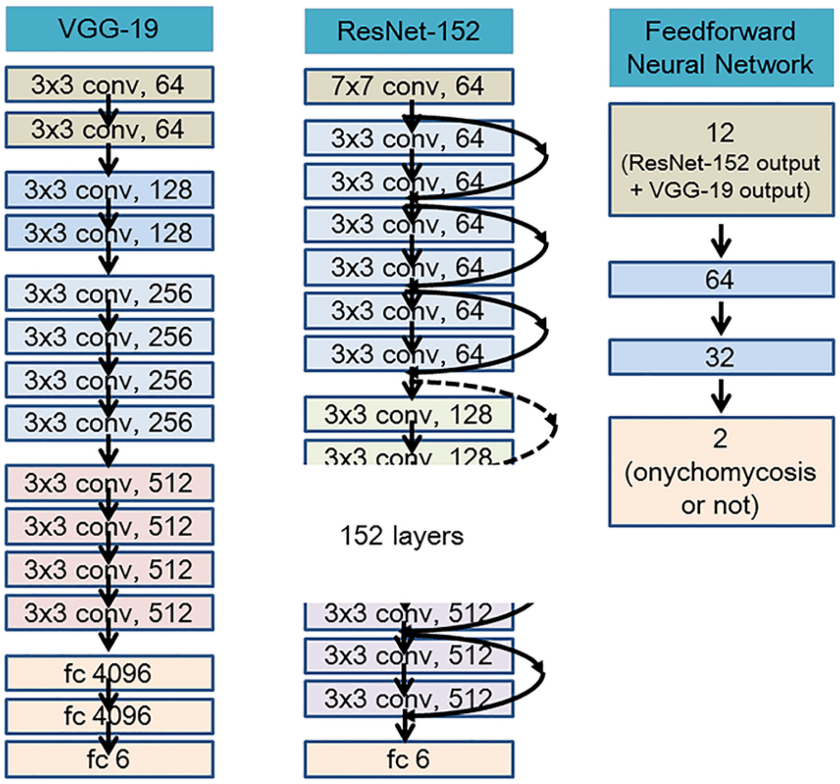


Figure 2.19: A comparison between a residual network and two convolutional neural networks. The VGG-19 model has around 143 million parameters and the ResNet-152 has approximately 160 million parameters ([22]).

The number of parameters in convolutional neural network can be very large and this is where the residual connections work best. Due to them linking the input and output of every layer they help with the gradient flow allowing for very deep networks. Currently ResNets are the state-of-the-art in image classification.

UNet

UNet is a structure for networks that take in a picture and output a same-sized picture. The UNet architecture was introduced in [56] for biomedical segmentation, but UNet has also proven to be useful as a generative model. Generative models generate an output image or tensor given some input data, in contrast to classification models that assign classes to data.

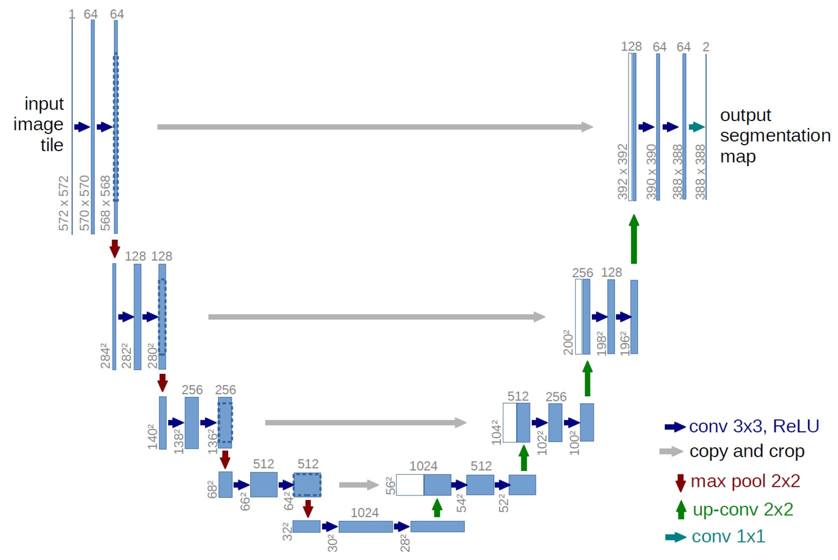


Figure 2.20: The UNet architecture clearly shows where it got its name. Taken from [56].

As shown in Figure 2.20 we recognize that UNet has an encoder-decoder structure. The encoder consists of a number of combinations of convolutions, pooling layers and non-linearities reducing the dimension of the latent space to only the important features. Then the features are upsampled with upsampling convolutions. The main innovation of the UNet was that during the upsampling a residual connection is concatenated from the encoder layer. This allows again these networks to capture many different features.

2.3.10. Optimization

In this section we introduce the optimization method we use for both the lens design parameters and the neural network weights and biases. These methods work only when gradients are available.

Gradient Descent

Optimizing a function when gradients are available, one can use a linear approximation of the function and minimize that. When minimizing the linear function we have to take into account that we are making a linear approximation and thus should not take a step to large. Let the current point be x_0. This gives us the following equation

$$\min_x f(x) \approx \min_x f(x_0) + (x - x_0)\nabla f(x_0) + \frac{1}{2\eta}||x - x_0||_2^2.$$

Here, the first and second term are given by linearizing and the last term is a penalty to stay close to the current point of evaluation. We use the η here for weighting the penalty. We set the derivative to zero on the right-hand side to find an extreme point and see the following

$$
\begin{aligned}
0 &= \nabla_x \left(f(x_0) + (x - x_0)\nabla f(x_0) + \frac{1}{2\eta}||x - x_0||_2^2 \right) \\
0 &= \nabla f(x_0) + \frac{1}{\eta}(x - x_0) \\
x &= x_0 - \eta \nabla f(x_0).
\end{aligned}
$$

Replacing x by x_t and x_0 by x_{t-1} we get the famous gradient descent algorithm in step t. This allows us to optimize any function with a gradient by walking down along the path of steepest descent also known as the negative gradient. The above approach is a simplified derivation of an algorithm called mirror descent. More on mirror descent can be found in this lecture at http://www.cs.cmu.edu/ 15850/notes/lec19.pdf.

The parameter η is known as the stepsize as it determines what size of a step we take. This is very important as one could imagine applying gradient descent to the function $f(x) = x^2$, it is not hard to see that choosing $\eta = 1$ would jump back and forth and not converge and that $\eta > 1$ would actually diverge away from the optimum when to far from the optimum, except if you start at the optimum.

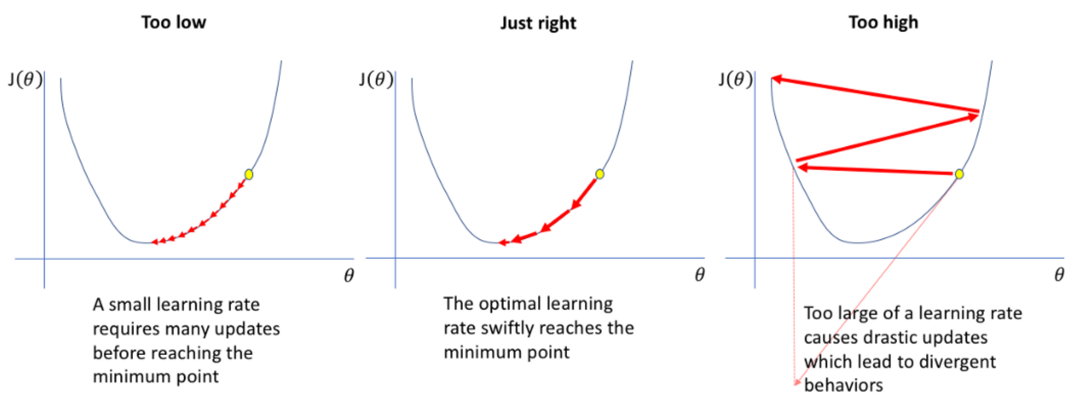

Too low
Just right
Too high
J(θ)
θ
A small learning rate requires many updates before reaching the minimum point
The optimal learning rate swiftly reaches the minimum point
Too large of a learning rate causes drastic updates which lead to divergent behaviors

Figure 2.21: Choosing the step size is important in gradient descent. Taken from medium.com

Adam

To optimize the network we use the *Adam* optimizer [33]. This optimizer is an extension of the normal gradient descent and works as follows

$$
\begin{aligned}
& t \leftarrow t + 1 \\
& g_t \leftarrow \nabla_\theta f_t(\theta_{t-1}) && \text{(get gradient)} \\
& m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) \cdot g_t && \text{(update biased first moment estimate)} \\
& v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) \cdot g_t^2 && \text{(update biased second moment estimate)} \\
& \hat{m}_t \leftarrow m_t / (1 - \beta_1^t) && \text{(bias correction)} \\
& \hat{v}_t \leftarrow v_t / (1 - \beta_2^t) && \text{(bias correction)} \\
& \theta_t \leftarrow \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon) && \text{(update parameters)}
\end{aligned}
$$

The strength of this method is that it works with exponential averages of gradient and squared gradient to make to make the right step and not behave very erratically. This also allows the method to step over local minima, as the step does not vanish directly from a low gradient value. As we use regularization, which we discuss further later on, we could define that as a squared weight term in the loss function or we simple change the second operation in the *Adam* algorithm to $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1}) + \lambda\theta_{t-1}$. Here, λ is the amount of influence the regularization to has. This step can be derived from the loss function approach as well. The standard values, the authors suggest, for the parameters are $\lambda = 0$, $\alpha = 1 \times 10^{-6}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Increasing λ, adds more regularization. The β's influence how long previous values are taken into account in the current optimization step. When dealing with noisy data β_1 and β_2 should be lowered so the noise averages out over more iterations. The learning rate α is the most important. The learning rate should be low enough to not step over minima, but at the same time the learning rate should also be high enough so the optimization moves away from non optimal zones faster. Higher learning rate could also help with stepping over local minima. In normal stochastic gradient descent we would decrease the learning rate every iterate, however as *Adam* is already an adaptive gradient method this is not necessary for convergence. Adaptive in this sense means that the overall learning rate changes based on the steps. In practice both a constant and a decreasing learning rate seem to work.

Regularization

Another concept in machine learning we use is regularization. When training on a lot of data the network is prone to overfitting. Overfitting is the concept of having the network optimize very well on the training set, but not learning the right features and thus performing bad on out-of-sample data.

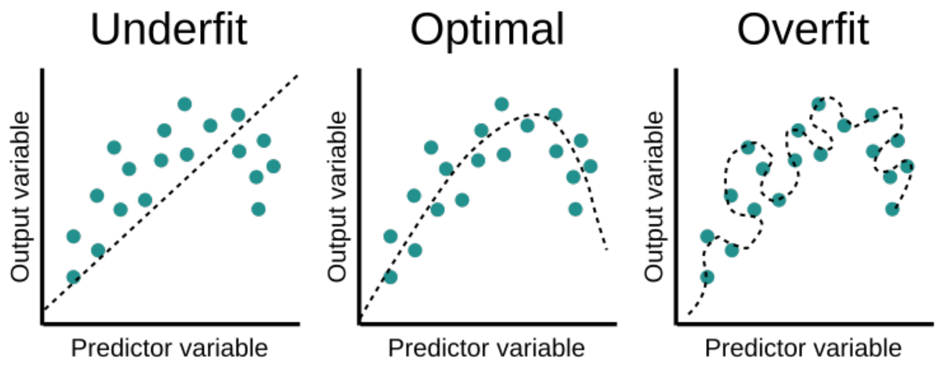


Figure 2.22: The data can be fitted by a simple line. By optimizing a high order polynomial we are also able to get minimal loss, but this does not capture the essence and will give a high error when evaluated on points not in the dataset. Figure taken from educative.io.

To make sure overfitting does not happen an extra parameter is added to the loss function. This parameter is the square of the weights, therefore the weights are forced to stay small. Regularization, the name for this procedure, actually is a bias-variance trade-off. We trade the unbiasedness of the network to reduce the variance, as we can see from Figure 2.22. In deep neural networks there is also a phenomenon called double descent, where adding more network parameters actually causes another descent in the loss after the bias-variance trade-off has already been taken into account. This phenomenon is described in [42]. In our research we also need to take this into account.

2.4. Parametric surfaces

In this section we describe different surface parametrization techniques that we have used to represent lenses. A parametric surface is a surface that is defined by certain parameters and allows to be calculated at every point in the domain, in contrast to grid-wise defined surfaces. We discuss B-splines and NURBS first. After this we also discuss radial basis functions.

2.4.1. B-splines

In this section we explain what B-spline curves and surfaces are. We start with the former and name its properties, after that we extend this to the B-spline surfaces that we use for our lens design.

B-splines (which is an abbreviation for basis-splines) are a specific set of polynomial basis functions that are used to fit a curve or surface along a certain set of control points. Let us denote the number of control points as n and denote the coordinates of the points as P_i, where i ranges from 0 to n. We would like to find basis functions with the property that our curve C satisfies the expression $C(u) = \sum_{i=0}^{n} f_i(u)P_i$. From this summation one can see that possible continuity is independent of the control points and only dependent on the basis functions themselves. To form the basis functions we first define the knot-vector $U = (u_0, ..., u_m)$, for which we have that the knots are non-decreasing $a \leq u_{i-1} \leq u_i \leq u_{i+1} \leq b$ for $i = 1, ..., m-1$, a and b denote our interval here. Let us define $N_{i,p}$ to be the ith basis function of degree p (note that here $i = 0, ..., m$, where m is the number of knots). Then the B-splines basis functions are defined recursively as

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{else} \end{cases} \tag{2.4.1}$$

$$N_{i,p}(u) = \omega_{i,p-1}(u)N_{i,p-1}(u) + (1 - \omega_{i+1,p-1}(u))N_{i+1,p-1}(u), \tag{2.4.2}$$

here we define

$$\omega_{i,p}(x) = \begin{cases} \frac{u-u_i}{u_{i+p}-u_i}, & \text{if } t_{i+p} \neq t_i \\ 0, & \text{else} \end{cases} \tag{2.4.3}$$

This recursive formula is know as the Cox de Boor recursion formula ([47]). Due to this definition, in the interval $[u_i, u_{i+1})$ only $N_{i-p,p}, ..., N_{i,p}$ are non-zero and the basis function $N_{i,p}$ is only nonzero in the interval $[u_i, u_{i+p+1})$. For $p = 0$ the step function is found.

An example of these basis functions with $p = 2$ can be found in Figure 2.23.

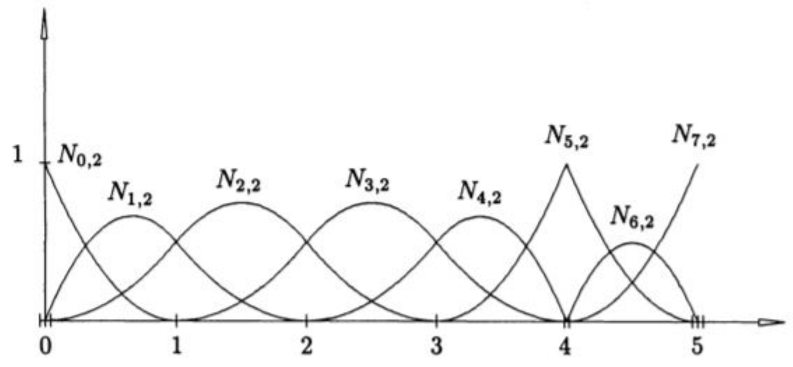


Figure 2.23: The quadratic B-spline basis functions with knot vector $U = (0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5)$. This figure is extracted from [48].

Figure 2.23 also clearly shows the impact the knotvector has on the B-splines. When the multiplicity of a knot equals k, then the continuity at that point is **C**$^{p-k}$. We see that for $u = 0$ and $u = 5$, we have multiplicity three such that the curve C is discontinuous at these locations. For $u = 4$ we have multiplicity two, thus the curve itself is continuous here, but not its derivatives. All other interior knots have multiplicity one, thus these basis functions and their derivatives are continuous at the knot locations. Another observation from this figure is that when the first and last knot have multiplicity $p + 1$, then there is exactly one basis function active at that point. We call such a knot vector open. Furthermore, as the B-splines are a weighted sum with their weights summing up to one, we must also have for $u \in [u_i, u_{i+1})$ that $\sum_{i=1}^{n} N_{j,p}(u) = \sum_{j=i-p}^{i} N_{j,p}(u) = 1$.

For knot vector U of length $n + p + 1$ we can define a p-times continuously differentiable B-spline curve in the following way

$$C(u) = \sum_{i=1}^{n} N_{i,p}(u) P_i, \qquad (2.4.4)$$

where $P_i \in \mathbb{R}^d$ are the control points. This is illustrated in Figure 2.24.

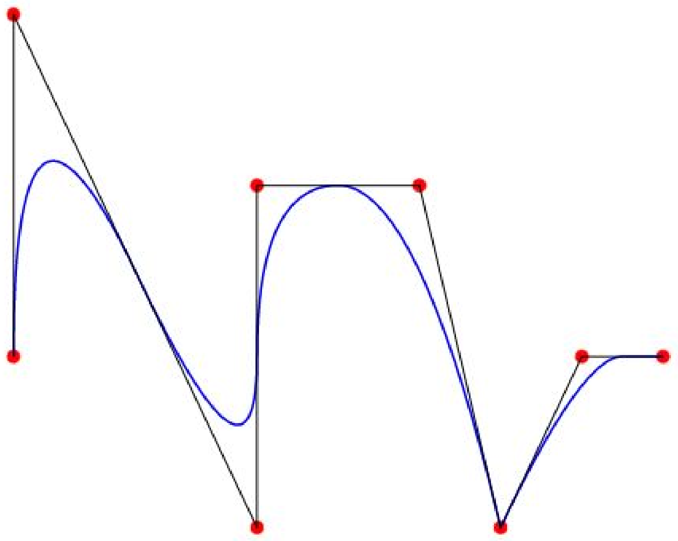
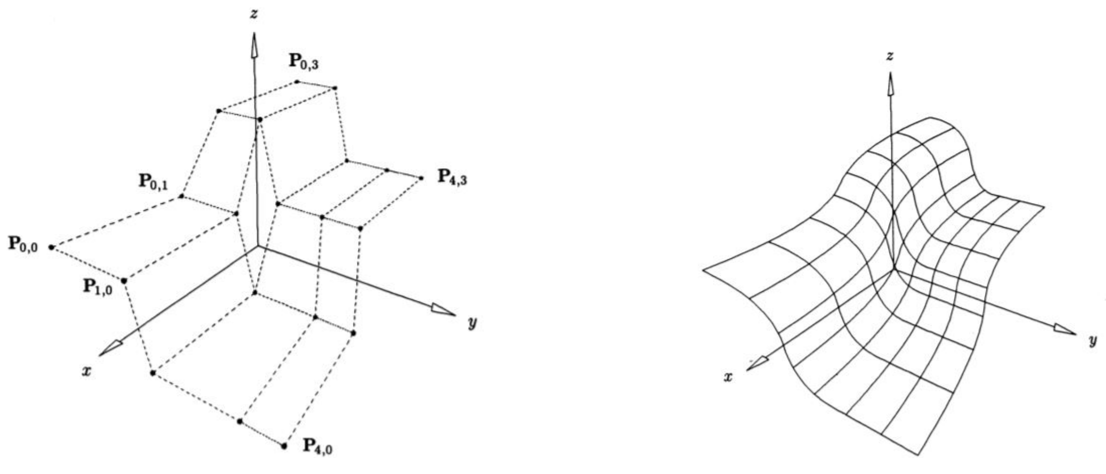


Figure 2.24: A B-spline curve in $\mathbb{R}^2$. The red dots denote the control points. Taken from [26]

To make a B-spline surface in $\mathbb{R}^3$, we choose the following. A surface consist of a tensor product of B-spline, for simplicity we assume that our p holds for both basis functions in each term. And furthermore let the knot vectors U and V be open and of length $n+p+1$ and $m+p+1$ and denote B-splines from knot vector U as $N_{i,p}(u)$ and B-splines from knot vector V as $M_{j,p}(v)$. Then we describe the B-spline surface which is p-times continuously differentiable as

$$S(u,v) = \sum_{j=1}^{m} \sum_{i=1}^{n} N_{i,p}(u) M_{j,p}(v) P_{i,j}. \tag{2.4.5}$$

Here, $P_{i,j} \in \mathbb{R}^3$ are the control points. An example of a B-spline surface and its control points can be found in Figure 2.25.



(a) The control points.

(b) The B-spline surface.

Figure 2.25: On the left-hand side we see the control points and on the right-hand side we see the B-spline surface. Images taken from [48].

B-spline curves and surface are themselves already able to fit a variety of curves, however a shape as a circle can only be approximated and never fitted exactly. It would require many degrees to get a good approximation. A more flexible parametrization technique that extends B-splines are non uniform rational B-splines (NURBS). NURBS use rational B-splines in the following way. Let $\{P_i \in \mathbb{R}^d\}_{i=0:n}$ be the control points and let $\{w_i \in \mathbb{R}\}_{i=0:n}$ be the weights. NURBS curves are defined as follows

$$\begin{aligned} C(u) &= \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} && (2.4.6)\\ &= \sum_{i=0}^{n} R_{i,p}(u) P_i, && (2.4.7) \end{aligned}$$

with

$$R_{i,p} = \frac{N_{i,p}(u) w_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i}. \tag{2.4.8}$$

A NURBS curve is more general than a simple B-spline as it can approximate with rational functions as well. It does however cost n new weight parameters. We can recover basic B-splines by setting all weight equal to a constant. When applying an affine transformation to the curve it is enough to apply it to the control points. The same properties that hold for B-splines with regard to continuity and differentiability also hold for NURBS. Extending this definition of NURBS curves to NURBS surfaces similar steps are taken as with B-spline surfaces. A NURBS surface $S(u, v)$ is defined as follows

$$\begin{aligned} S(u,v) = &= \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u) M_{j,p}(v) w_{i,j} P_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p} M_{j,p}(v) w_{i,j}} && (2.4.9)\\ &= \sum_{i=0}^{n}\sum_{j=0}^{m} R_{i,j,p}(u,v) P_{i,j}, && (2.4.10) \end{aligned}$$

with

$$R_{i,j,p}(u,v) = \frac{N_{i,p}(u) M_{j,p}(v) w_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p} M_{j,p}(v) w_{i,j}}. \tag{2.4.11}$$

We saw already that we could calculate the basis functions using the Cox de Boor recursion formula (2.4.2), however this procedure is not very efficient as on most intervals a part of the B-splines are zero. Thus to make the algorithm more efficient we need to make sure these calculations do not happen. In [9] a solution for this problem is proposed which changes the recursion into multiple efficient matrix multiplications. To show how this works, we first have to revisit the weight function (2.4.3) to build the B-splines. Using this function $\omega_{i,p}(x)$ we define the B-spline factor matrix as follows

$$T_p(x) = \begin{pmatrix} 1-\omega_{i-p+1,p}(x) & \omega_{i-p+1,p}(x) & 0 & \dots & 0 \\ 0 & 1-\omega_{i-p+2,p}(x) & \omega_{i-p+2,p}(x) & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1-\omega_{i,p}(x) & \omega_{i,p}(x) \end{pmatrix}. \tag{2.4.12}$$

$T_p(x)$ is a $p \times (p+1)$ band-limited matrix and this matrix can thus be used in sparse matrix computations. Let us denote $T^p(x) = T_1(x)T_2(x)...T_p(x)$, then we can rewrite a B-spline with control point vector **P** as $C(u) = T^p(u)\mathbf{P}$. To extend this to a B-spline surface, we first have to introduce the Kronecker product $\otimes$, which is defined as

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{pmatrix}$$

and let us also define the vectorization of a matrix as

$$\text{Vec}(A) = \begin{bmatrix} a_{11}, & \dots, & a_{m1}, & a_{12}, & \dots, & a_{m2}, & \dots, & a_{1n}, & \dots, & a_{mn} \end{bmatrix}^\top$$

. We denote $N(u) = T^p(u)$ and $M(v) = T^q(v)$, then these are exactly the B-spline matrices in the x and y direction. We rewrite the B-spline surface as

$$S(u,v) = \sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p}(u)M_{j,p}(v)P_{i,j} \tag{2.4.13}$$

$$= (N(u) \otimes M(v))\text{Vec}(P^\top_{i:i+p+1,j:j+q+1}). \tag{2.4.14}$$

This allows us to efficiently calculate the values on the surface, without multiplying a lot of zeros due to the limited support of B-splines. We can easily extend these results for B-splines to the NURBS setting.

2.4.2. Radial basis functions

Next to B-splines in optics three other parametrization are also used as seen in [57]. They are the Zernike polynomials, the XY polynomials and radial basis functions.

The Zernike polynomials are a set of polynomials that are orthogonal on the unit disk. The definition of Zernike polynomials is split up in even and odd polynomials. The even polynomials are defined as

$$Z_n^m(\rho,\psi) = R_n^m(\rho)\cos(m\psi) \tag{2.4.15}$$

and the odd polynomials are defined as

$$Z_n^{-m}(\rho\psi) = R_n^m(\rho)\sin(m\psi), \tag{2.4.16}$$

. Here, $R_n^m(\rho)$ are radial polynomials and by odd and even we mean that m is odd or even. The Zernike polynomials have applications in modelling aberrations in lenses.

Another method to model a surface are the XY-polynomials. These polynomials are different compared to Zernike polynomials as they are not orthogonal and similar to B-splines in that they use basis functions. The XY polynomials use basis functions of the form $x^n y^m$. The XY polynomials have more degrees of freedom when compared to a Zernike polynomial.

A method not necessarily based on polynomials are the radial basis functions (RBFs). We can describe a surface in terms of RBFs as

$$S(u,v) = \sum_{i=1}^{n} w_n\phi_n(x,y), \tag{2.4.17}$$

where we define $\phi_n(x, y) = \psi(||(x, y) - (x_n, y_n)||_2)$. These surfaces are thus build of weighted sums of radially symmetric functions. Typical choices for ψ in optics are for example Gaussian functions.

Other polynomials used in freeform optics are Chebyshev polynomials, Legendre polynomials and Jacobi polynomials. An overview of the different methods is given in [73].

We found that generating Zernike polynomials is very computationally expensive, due to the summation in the radial polynomial part. Therefore, we choose to use the radial basis idea of radial basis functions and Zernike polynomials to generate random radial basis functions (that is radial basis functions with random coefficients) as example lenses which we use later. We use B-spline surfaces to represent lenses and radial basis functions to generate a variety of diffraction patterns, as Zernike polynomials are a subset of these basis functions.

3 Previous Research

3.1. Related Literature

In this section, we discuss the literature in the fields necessary for our research. We divide the literature interesting for us in three categories: phase retrieval, freeform and diffractive lens design and lastly deep learning and optics.

3.1.1. Phase retrieval and deep holography

The inverse problem of finding a freeform lens that produces a certain distribution, is strongly connected to the problem of phase retrieval. This problem is given by finding the appropriate phase of a given input source amplitude and a target intensity. To solve this problem a popular algorithm was proposed in 1972 in [18]. In [19] an iterative algorithm is proposed which has as input two different intensities and the task of the algorithm is to find the phase change to propagate from one to the other. This algorithm, called the Gerchberg-Saxton (GS) algorithms, works by iterating between forward and backward Fourier transforms on the source and target intensity, iteratively until a stopping criterion is met. Figure 3.1 illustrates the general principle of the algorithm. The GS algorithm has applications in multiple fields, in optics it is mainly used in holography and coherent imaging, but it also has applications in other imaging sciences.

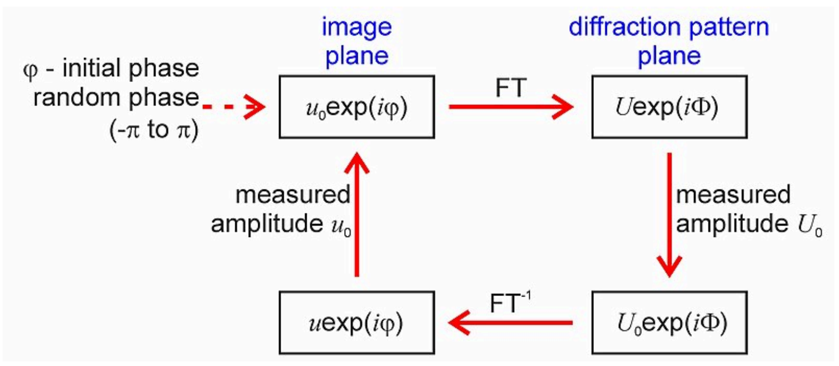


Figure 3.1: The Gerchberg-Saxton algorithm.

It can be proven that the GS algorithm is monotonically decreasing its objective function ([18]), which is remarkable given that the algorithm is mostly applying Fourier transforms. In [16] it was shown that the GS algorithm is equivalent to a gradient descent algorithm. Where it is also shown that the convergence of the GS algorithm is actually not as good as other gradient descent and Fourier based methods. However, the GS algorithm remains the standard in the literature. Recently, a deep learning approach to phase retrieval was taken in [59], where a convolutional neural network inspired by the GS algorithm is proposed. In other literature we also see the use of convolutional neural networks which work with the methods we discussed in previous sections. An example is [55], where a network is made to extract the phase from a single intensity image. The structure of this network can be seen in Figure 3.2.

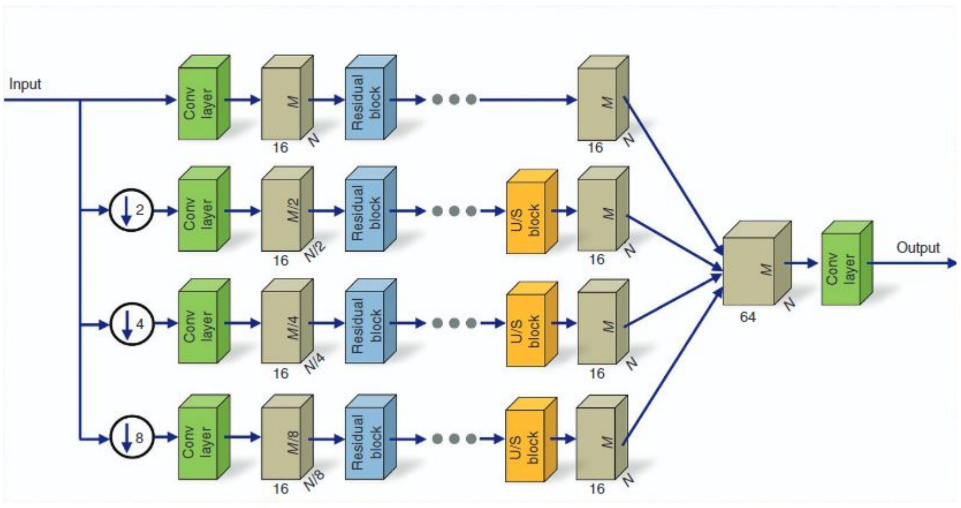


Figure 3.2: The network structure used for phase and amplitude recovery in [55].

Figure 3.2 also gives an idea of the networks used in these types of problems. In [41] a convolutional neural network is used for phase recovery and denoising. Metzler et al. also compare their method to multiple other established methods, such as gradient descent based methods and show that their method has better convergence properties. [21] gives a more theoretical overview of the subject and based on the theory they propose a method with a generative model.

Phase recovery is connected to the field of holography as it solves a similar problem. In the field of holography, the challenge is to reconstruct the whole wave field from a reference beam. Thus considering phase-only holography, this would essentially be equivalent to phase recovery. Deep learning has also made an impact on the field of holography. An example of this is [23], where the inverse process of finding the hologram is solved by giving the network a target pattern, letting the network suggest hologram and applying an optical propagation, the forward process, to the suggested hologram. This gives an error between the target pattern and the reproduced pattern, which should be minimized. For the network Horisaki, Takagi, and Tanida use a UNet like convolutional network, which also shares properties with the above shown network of [55]. In [54] a review is also given with applications in, e.g., microscopy and material science. Where the conventional optics leave the 3D nature of light and only consider the phase and amplitude, Ren et al. considers 3D vectorial holography and shows that they are able to train a deep network to solve the inverse problem in this setting as well. Cheremkhin et al. show that they are able to find a diffractive optical elements as output of the network, where the input is the intensity image. The network is trained on the MNIST dataset of written digits. In [25] again an unsupervised UNet approach is discussed. Another interesting development is [62], where a similar network is proposed as in [23], but optimized to be memory efficient and being able to operate on less powerful devices than standard computers. This enables the techniques to be applied in almost real-time on embedded systems.

3.1.2. Freeform and diffractive optical elements

Two fields in optics that are also relevant to our research are the field of freeform optics and the field of diffractive optics. Freeform optics studies the properties of freeform lenses, whereas diffractive optics studies diffractive optical elements. Freeform lenses are lenses which do not have any rotational symmetry. Where normal lenses in, for example, glasses focus the light on a specific point, freeform lenses are able to shape the light in any way. Diffractive optical elements (DOE) consist of phase plates, which change the phase of the light, this is similar to freeform lenses. However, DOEs can consists of non-continuous surfaces, where a freeform lens has a continuous surface.

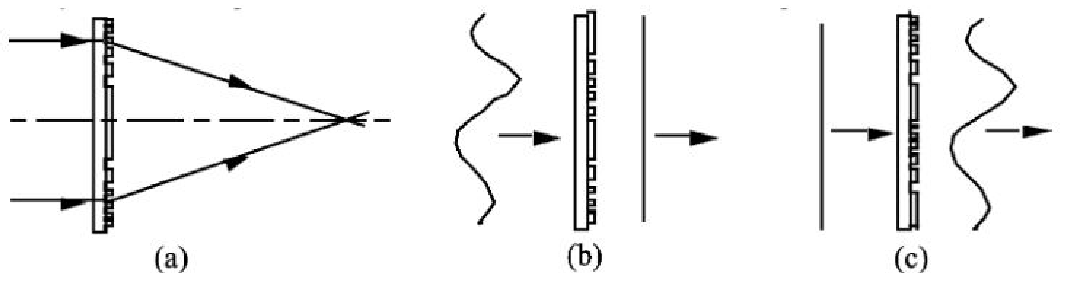


Figure 3.3: Different diffractive optics as shown in [43]: (a) Shows a DOE used as a diffractive lens. (b) shows a DOE as a wavefront corrector. (c) shows a null corrector.

The theory of designing diffractive optical elements is very strongly connected to the contents of the previous section on phase retrieval as with phase retrieval one could make a phase plate or a diffractive optical element to get the correct intensity.

Freeform lenses have a wide range of applications, but one application is well researched in literature. Freeform lenses are used extensively in the control of illumination. As the lenses can shape the illumination pattern of a lens in any way this has applications in normal lighting especially for light emitting diodes (LEDs), which have a very broad intensity pattern. Most of the light is too weak to be useful. However, LEDs are very energy efficient and thus we would like to use their capabilities. To fix this problem in, for example, street lighting, we could use a freeform lens to make a uniform intensity pattern such that a part of the road is well lit. Such optical systems for street lights are described in [63], in which an actual working optical system is shown. Similar design for street lighting is also considered in [74] and [64]. [31] and [14] show theoretical results, which confirm the power of freeform lenses for street illumination. The aforementioned papers all consider the challenge of illumination control. Other applications which include the use of freeform optics to create see-through lenses with a large field-of-view as described in [10]. Also Peng et al. shows that it is possible to create a freeform camera lens that has better field-of-view properties than normal lenses. An overview of applications in freeform optics and other freeform industrial parts is given in[58].

The shape design of a freeform lens is not a trivial problem. The design of freeform lenses is often performed in a setting of geometrical optics (also known as ray optics). In this setting [53] and [52] describe how the problem can be turned into a set of non-linear partial differential equations of the Monge-Ampere type. The equations are manipulated to be solvable by standard numerical methods, but this restricts the boundary conditions for which these equations can be solved. As the Monge-Ampere equations are of elliptic type, it can be shown that the finding of solutions to the equations is equivalent to solving a minimization problem. In [67] it is shown how to find a minimization problem for lens design. Wolansky and Rubinstein shows how the functional that is minimized is connected to other well-known equations, such as transport equations. The minimization of a functional is used more in literature, see for example [2] where a functional is found by reformulating the problem with an eikonal distribution. This functional is then minimized by gradient descent methods. We also see an optimal transport formulation again in [60]. [3] gives a set of generalized Monge-Ampere equations. One paper that differs from the rest is [35], where other papers mostly consider only one point source, here multiple sources are considered. This also has applications for our research, as we also consider multiple sources. With multiple sources Lin note that the equations become harder, but they are able to solve the problem by a weighted least squares approach. These conventional methods of freeform design are still being investigated as can be seen from [4], where the elliptical PDE problem is revisited again in a formulation which allows for even less restrictions on the lenses and the output and input irradiance required.

Where the previous paragraph mainly describes algorithms involving the finding of a solution to partial differential equations or by minimizing a functional, the output is a function which would describe the freeform lens. To restrict the problem it is also common to look at the design of freeform surfaces described by basis functions. Common surface representations are Bézier, B-spline or Non Uniform Rational B-Spline (NURBS) surfaces as [58] lists. These parametrized surfaces have advantages over non-parametrized surfaces as they have an mathematical expression which describes them by a low amount of parameters and they are often already available in computer aided design (CAD) and computer graphics applications. Applications in CAD is especially true for NURBS surfaces as can be found in [5]. Other surface representations are also found, such as in [7] where the basis functions are chosen as Gaussian functions. These Gaussian functions allow a variety of lenses and this is shown in practical applications. [17] discusses a basis of orthogonal polynomials and how to efficiently compute their coefficients. A specific algorithm to design freeform optics with NURBS is introduced in [68], they show how to change this problem into an overdetermined non-linear system and how to solve this system using the Gauss-Newton method. [15], [73] and [69] give an overview of surface representation. Wu et al. mentions that alongside the already mentioned NURBS and Gaussian functions, XY-polynomials and Zernike-polynomials are also used. This last class of Zernike-polynomials is actually used in optics as the polynomials have special properties, which help with lens abberation correction. A recent book on the same topic which also mentions multiple algorithms to find the correct parametrization is [32].

Another related topic in optics are the properties of freeform lenses themselves and why we would use them compared to diffractive optical elements. In certain application diffractive optical elements create speckles, a problem that freeform lenses do not have. The problem of creating a uniform square intensity for street lighting is approached in [45], where the problem is solved with diffractive optical elements. However, due to discontinuities in the phase of their optical elements it is not possible to create uniform intensity and the correct amplitude. [1] also considers this and gives a mathematical description of the speckles and give methods how to remove these speckles. A clear take away from [1] and [45] is that requiring continuity in the lenses would make sure that there are no speckles. This gives an advantage of freeform over diffractive optical elements due to their continuity. However, the continuity of a freeform lens also has consequences as [6] mentions. As freeform lenses only have continuous phase changes we cannot have multiple non-connected intensities in our output intensity. This is due to the reformulation of the previously mentioned eikonal function in terms of a linear assignment problem.

3.1.3. Deep learning in optics

In the previous paragraphs we saw that deep learning techniques are applied to the field of holography and phase recovery. Deep learning has also found its way to other fields within optics. An overview of deep learning in optics and photonics can be found in [66].

A specific application of deep learning in optics is the invention of deep diffractive neural networks. These networks are physical neural networks, where each layer consists of a diffractive optical element. When multiple of these DOEs are placed consecutively, they are able to approach non-linear relations similar to normal neural networks.

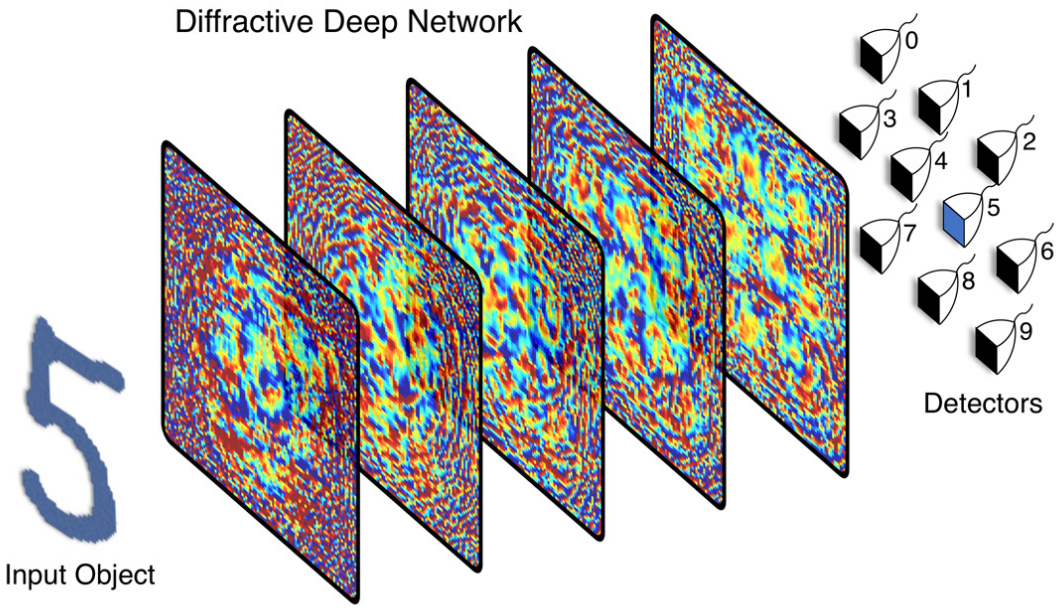


Figure 3.4: An example of a deep diffractive neural network as shown in [36]

These deep diffractive neural networks (D^2NN) are implemented digitally to optimize the phase plates as described in [36]. They test their D^2NN in a classification setting, the setup for this can be seen in Figure [36]. In [39] a further analysis shows that their D^2NN achieves a 97.18 percent accuracy when their network consists of five phase-only plates. In [44] and [38] it is shown that these networks are able to adapt to data that is shifted, rotated, scaled or misaligned in another way. This shows similarities between convolutional neural networks and these networks, as one of the advantages of convolutional layers is their ability to be invariant under shifting and rotation. Interestingly there have already been other applications of these networks than to classification. In [49] it is shown that D^2NN are able to solve the inverse problem of holography, which we discussed earlier in this section. In [40] it is furthermore shown that these networks are also able to solve the ill-posed inverse problem of reconstructing overlapping phase images.

More in line with our research there are also applications of deep learning to the design of lenses and other optical surfaces. We mostly see deep learning being applied to the generation of starting points for the process of lens optimization. The optimization of lens design is a difficult problem and the base of our research. In [71] a neural network is used to find the position of optical elements and the parameters for these elements. The elements are made by parametrized surfaces of polynomials such as XY- and Zernike polynomials. The input of the network are the optical system properties such as field of view and F number. This approach is also found in [12] and [8], where even more complex optical systems can be found. The latter also adds new optimization techniques, but the base remains similar.

3.2. Previous work at TU Delft

The results from this thesis builds upon the results from projects by other students. As the goals in this thesis and the other work done at TU Delft differ from the literature we discuss these results separately. This section summarizes both the results from [27] and [13]. The former considers the design of a freeform lens based on B-splines in one dimension and the latter extends these results to two dimension.

3.2.1. Imhof (2020)

In this thesis a one dimensional field is assumed. Imhof uses a physics informed self-supervised approach and shows that this produces desirable results. Self-supervised means that the network does not necessarily have data in input/output pairs for the network to train, but rather it has input and this input is also a term in the loss function. The setup of this thesis was to have the input of the network be the desired intensity pattern and the output of the network be the B-spline parameters. The loss function that makes this self-supervised first turns the B-spline values into a lens design and then uses the Fraunhofer diffraction to simulate propagation of light. The propagated intensity pattern is then compared to the desired intensity pattern, thus creating a trainable loss. In this thesis it was shown that for the one dimensional case this procedure generates desirable lenses. The network is trained with a number of intensity patterns that are feasible. The approach is shown to work when the B-splines have three degrees of freedom, which is quite limited. It is extensively discussed how the network is trained. This shows us that this approach should be feasible, at least with limited degrees of freedom.

3.2.2. Crijns (2021)

In this thesis the approach of Imhof is extended to the two dimensional case. This work still considers a far-field Fraunhofer diffraction for the optical propagation and considers the intensity pattern as the input for the network, where the output of the network are the B-spline parameters. This work differs from the one dimensional work as only one specific image is considered and the network is trained for this image. This turns the procedure into a fancy optimizer. To help the optimization procedure it is chosen to use multiple resolutions. This should make the optimization easier, as the optimization can first be done on a coarse grid with less degrees of freedom. The research concludes to show that it is quite hard to make the intensity pattern of the TU Delft logo.

4 Experiments

The goals of this thesis is to combine modern computer power, modern optimization methods and wave optics to optimize a system of diffractive optical elements. The optical system consists of the following three general parts. First, the source of light, this could be a number of distinct sources or one source. Second are diffractive optical elements. Here, we can choose to have multiple optical elements behind each other and possibly also have restrictions on the smoothness of the lens. As discussed before in Section 3.1, having smooth lenses is favorable as they suffer less from speckle effects. The last part of the optical system is the sampling plane. The intensity we optimize for on the sampling plane is given. In our optimization procedure the goal is to find the design parameters of diffractive optical elements such that the output intensity a system with these elements produces is close to the target intensity. Another element of our optimization procedure is the parameters we are optimizing over. This, in essence, determines what phase change the lens induces. In Chapter 2 of this thesis, we showed that we can simulate propagation of light in multiple different ways. The three main method were Fresnel, Fraunhofer and Rayleigh-Sommerfeld propagation. The main differences of these methods are when they are applicable, Fraunhofer is the most restrictive and requires the sampling plane to be far away from the lens. Rayleigh-Sommerfeld has the least restrictions. Although, we used Fraunhofer in earlier experiments and showed that we can optimize a simple optical system. We cannot use it in a system with multiple lenses as the propagation distance between the lenses is too small for Fraunhofer to be used. Therefore, we use Rayleigh-Sommerfeld diffraction. In the rest of this chapter we discuss a simple optical optimization using gradient descent, we discuss a data-driven neural network approach and finally we discuss an experiment where we optimize a difficult optical system using the two approaches together.

In most of the experiments shown below, we use the values given in Table 4.1 if not specified otherwise.

Table 4.1: Constants used in optical simulation

Name	Symbol	Value
Sample plane width		10^{-3}
Propagation distance	z	10^{-3}
Wavelength	λ	633×10^{-9}
Wavenumber	k	≈ 0.0036
Beam width	w_0	300×10^{-6}
NURBS control points grid size		$(10, 10)$

Our basic optical system setup is described in Figure 4.1. This basic optical setup is extended in the following results with multiple diffractive optical elements and multiple sources. Also the type of sources is varied in different experiments.

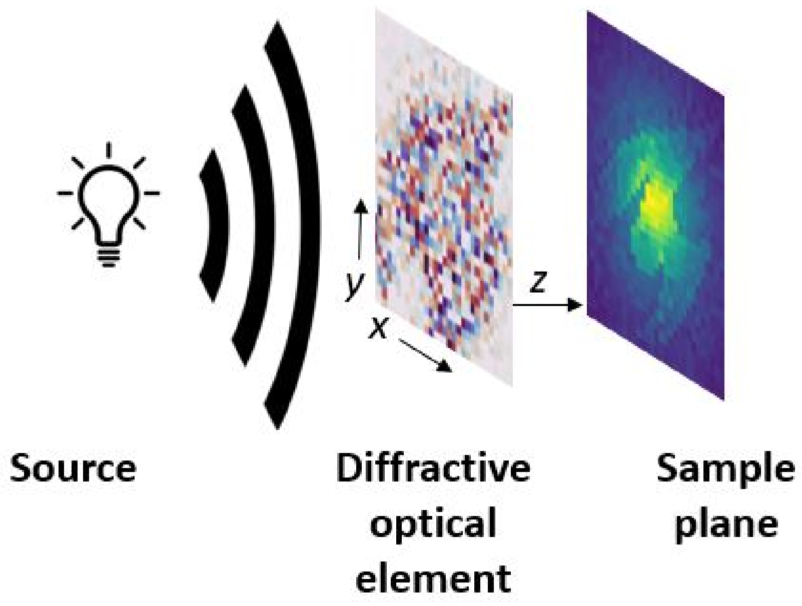


Figure 4.1: The setup of our optical system. We can have one or multiple sources next to each other (in the x and y direction) and one or multiple diffractive optical elements (in series, in the z-direction). We propagate light in the z-direction.

Table 4.2: In this table for each figure in the results the optical settings are reported. The resolution described holds for all elements in the system. When a DOE is not based on a NURBS surface this is noted in the sixth column.

Figure	Resolution	Source type	Number of sources	Number of DOEs	NURBS	Target
4.2	32×32	Plane wave	1	1	✗	TU Delft logo
4.3	32×32	Plane wave	1	1	✓	TU Delft flame
4.4	32×32	Point source	1	1	✗	TU Delft flame
4.5	32×32	Gaussian beam	1	1	✗	TU Delft flame
4.6	32×32	Plane wave	1	2	✗	TU Delft flame
4.7	32×32	Plane wave	1	2	✓	TU Delft flame
4.10	512×512	Point source	1	1	✗	FMNIST
4.11	512×512	Point source	1	1	✗	TU Delft flame
4.12	256×256	Point source	1	1	✗	FMNIST
4.13	1024×1024	Point source	1	1	✗	FMNIST
4.14	128×128	Point source	1	1	✗	Square
4.17	128×128	Point source	2	1	✗	TU Delft flame
4.18	128×128	Point source	3	1	✗	Multiple
4.20	128×128	Point source	3	2	✗	Multiple

Throughout this chapter the optical system changes for different experiments. In Table 4.2. This table is given to give a better overview of the different experiments and to compare directly what changes between experiments.

4.1. Optimizing a system of diffractive optical elements

We are tasked with optimizing the optical system. We propose multiple configurations of diffractive optical elements which we optimize such that the output intensity of the system resembles the target intensity profile. The different optical systems, vary in the amount of lenses, the sources, the diffraction method (Fraunhofer or Rayleigh-Sommerfeld), the parametric formulation of the surface and the different target intensities. We optimize the lens parameters by using the Adam gradient descent optimizing algorithm, we program our own optical system in the Pytorch package with back-propagation which allows for easily calculated gradients.

An overview of the figures in this section is found in Table 4.3

Table 4.3: In this table for each figure in the results the optical settings are reported. The resolution described holds for all elements in the system. When a DOE is not based on a NURBS surface this is noted in the sixth column.

Figure	Resolution	Source type	Number of sources	Number of DOEs	NURBS	Target
4.2	32×32	Plane wave	1	1	✗	TU Delft logo
4.3	32×32	Plane wave	1	1	✓	TU Delft flame
4.4	32×32	Point source	1	1	✗	TU Delft flame
4.5	32×32	Gaussian beam	1	1	✗	TU Delft flame
4.6	32×32	Plane wave	1	2	✗	TU Delft flame
4.7	32×32	Plane wave	1	2	✓	TU Delft flame

4.1.1. System of a single optical element

We first consider the simplest version, a single source and one phase plate, as we saw in Figure 1.1. The source is given by a plane wave which is normally incident and has unit amplitude. The lens only modulates the phase. The diffractive optical element can have phase changes at every pixel of the lens. Depending on the target intensity we also let the amplitude of the source field be variable, to allow the system to have enough energy to recreate the target intensity, meaning that the source term is multiplied by a term which we also optimize.

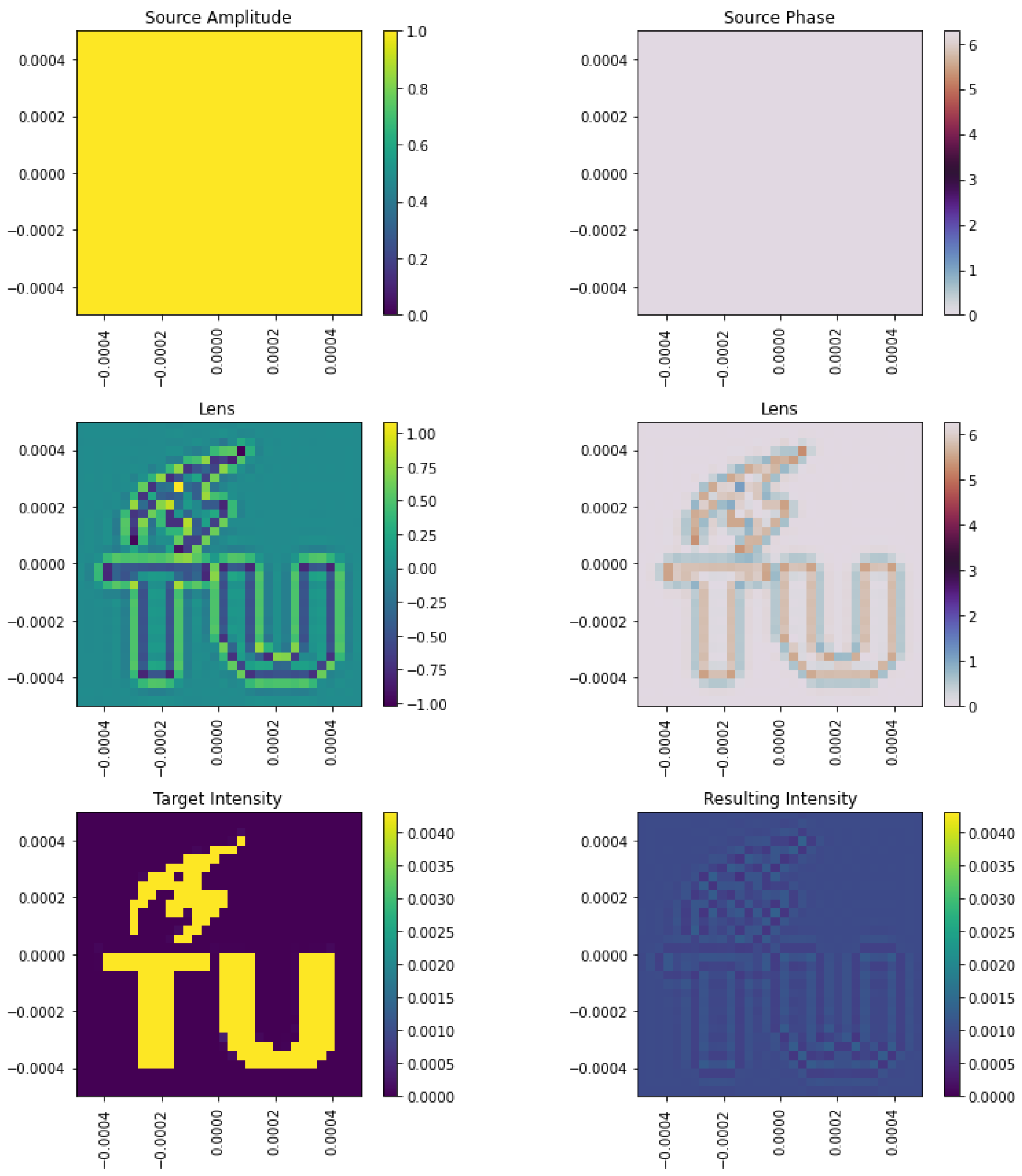


Figure 4.2: Optimized single DOE system with Rayleigh-Sommerfeld propagation. The first row shows a plane wave source. The second row shows the single optimized phase plate twice, the right one is corrected for the range of zero to 2π. The bottom row shows the target and resulting intensity.

The first results are shown in Figure 4.2. The top row of this figure shows the complex field of light created by the source of light. The middle row shows the phase change we have optimized for and the last row shows the target and resulting intensity. It can be seen that the TU Delft logo nicely comes back in the resulting intensity on the bottom right of the figure, however not with a filled logo and only the edges have contrast. The target intensity is recognizable in the lens. This is expected as we saw similar effects when propagating FMNIST items.

We could also consider the same single lens system but with a lens described by a NURBS surface. We also change the target intensity as a continuous lens cannot describe multiple non-connected elements. We then find the following.

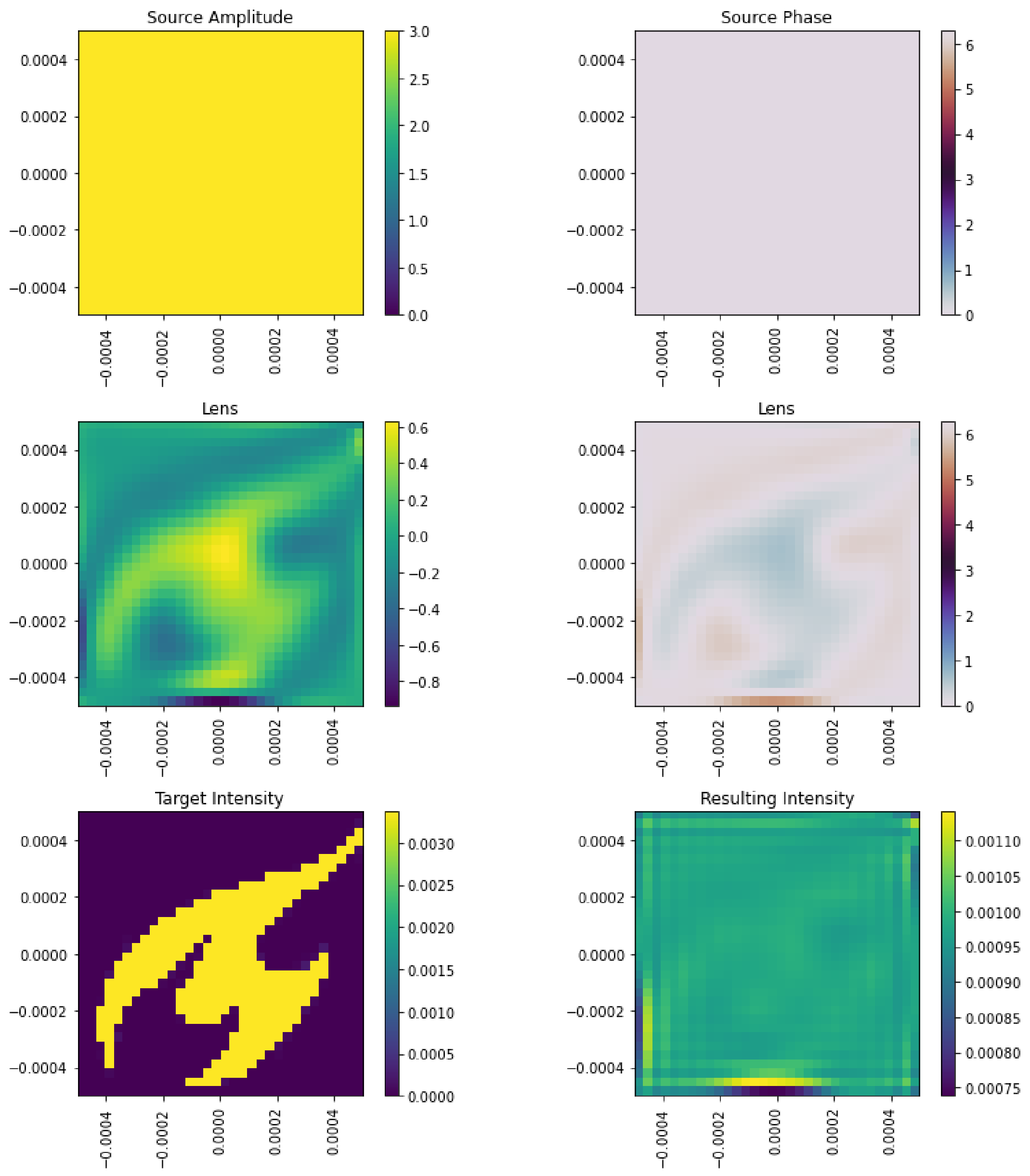


Figure 4.3: Optimized single NURBS lens system with Rayleigh-Sommerfeld propagation. The first row shows the source field, the second row shows the phase plate and the last row shows the target and generated intensity. Note that the lower right intensity is not on the same scale and that with the correct scale the small effect seen here is not visible.

Comparing the results in Figure 4.3 with the results from the previous optimization with the DOE in Figure 4.2, we directly see a clear difference. The resulting intensity is weak compared to the background as can be seen in the bottom row of Figure 4.3. Similarities between the target and the lens are visible in the middle row of the same figure, showing that the optimization changes the lens correctly. However, the low contrast in the intensity shows that this parametric surface is not suitable for this problem. Note that we changed the target intensity compared to the previous optimization, as continuous lenses are not able to generate non-continuous intensities. Thus the discontinuity between the letters and the flame of the TU Delft logo would not be feasible to generate. As this parametric surface is not feasible the following results are with diffractive optical elements.

Instead of the simple plane wave system, we also take a point source and a Gaussian beam as a source, as this is closer to real world application.

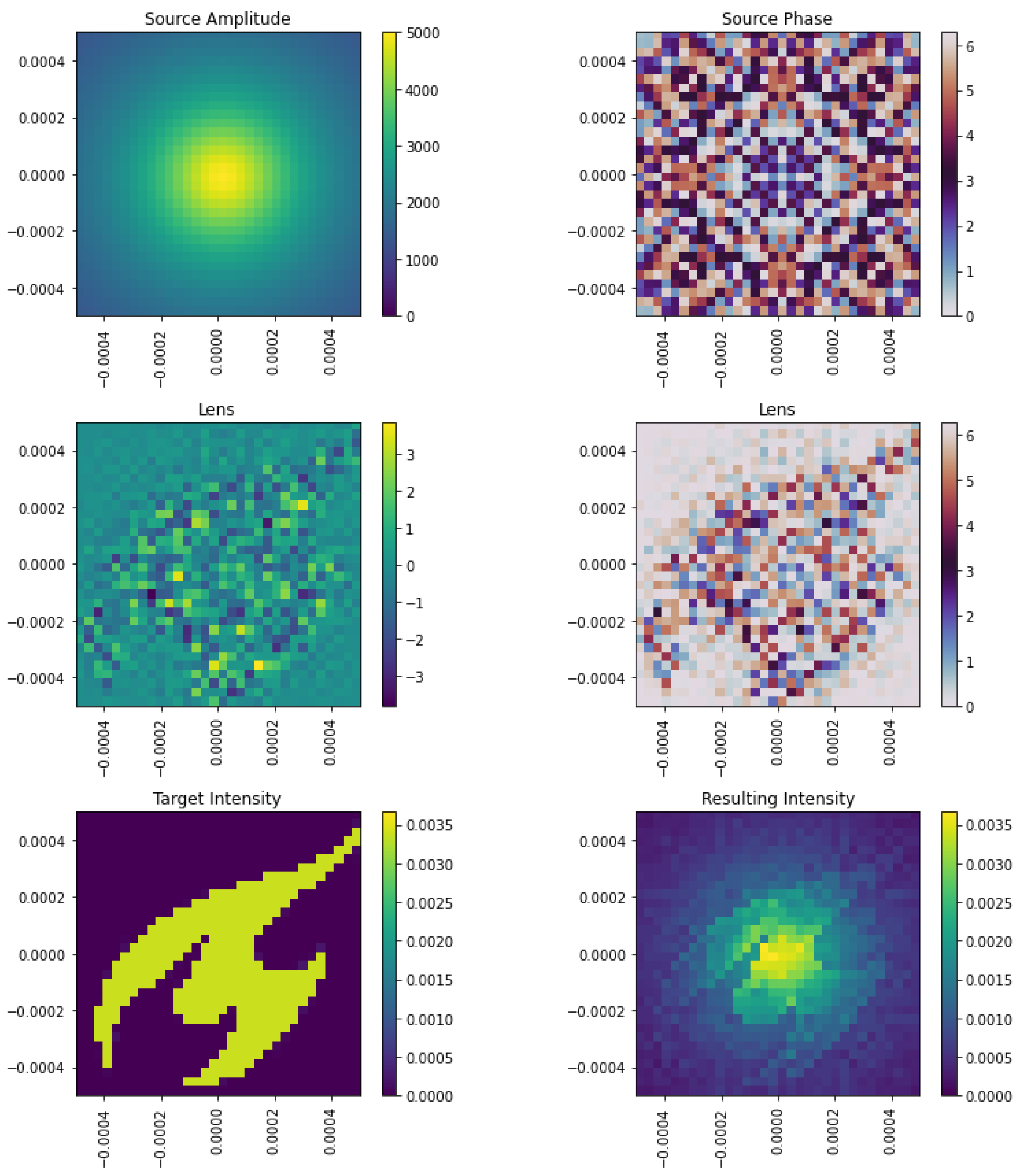


Figure 4.4: Using a point source with optimization of a DOE with a resolution of 32×32. The first row shows the source field, the second row shows the DOE phase and the last row shows the target and generated intensity.

For the point source we see in Figure 4.4, that the phase plate corrects for the phase of the source. Even-though, the pattern that is visible in the source is not clearly seen in the phase plate. The resulting intensity looks very much like the target, but we also see effects from the amplitude of the source.

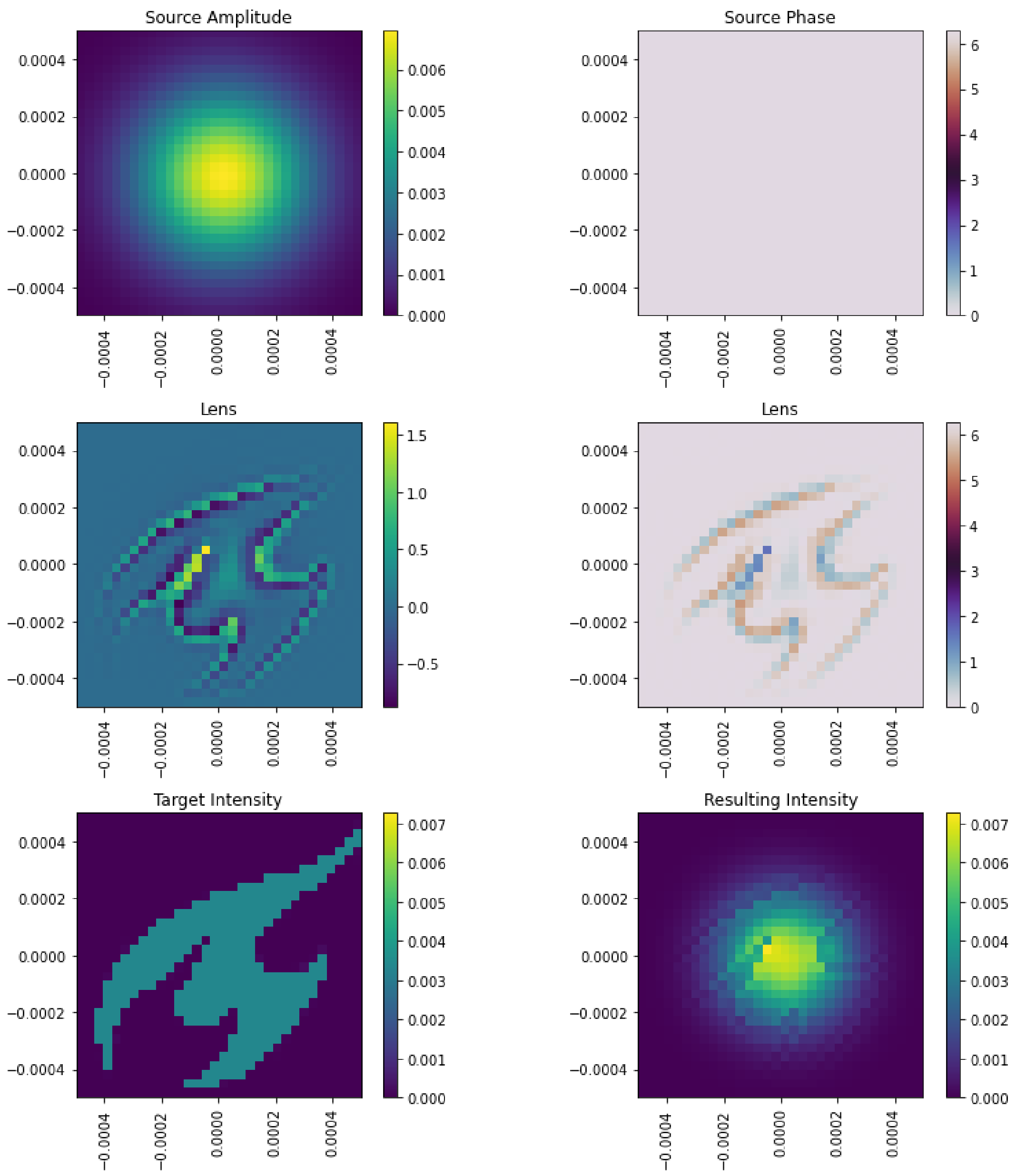


Figure 4.5: Using a Gaussian beam with optimization of a DOE with a resolution of 32×32. The first row shows the source field, the second row show the DOE phase and the last row shows the target and resulting intensity.

The Gaussian beam mainly illuminates the centre of the DOE, therefore, causing the optimization to also only focus on the centre as can be seen in Figure 4.5. We thus see this effect in the phase plate and the resulting intensity. Apart from that the result is similar the the previous result, however the phase plate does not need to correct for the phase of the source.

4.1.2. Multiple lenses

To increase the abilities of the optical system to generate more detailed intensities on the sample plane, we introduce a system with multiple lenses. We show the result again for DOEs and NURBS lenses.

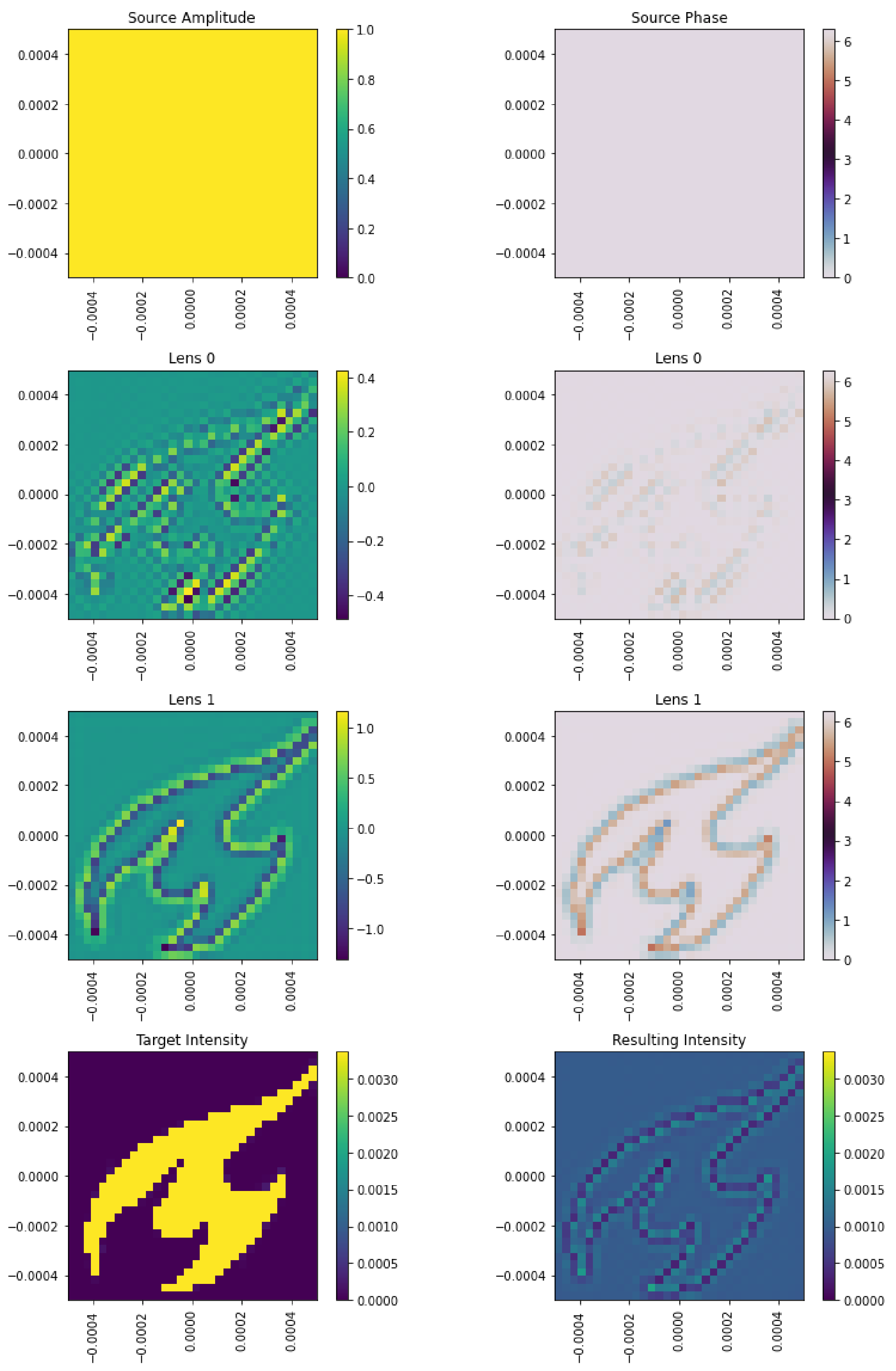


Figure 4.6: Optimized double DOE system. The first row shows the source fields, the middle two rows show the first and second DOE phases and the last row shows the target and resulting intensity.

The parts of a double DOE system which is optimized for a target intensity are seen in Figure 4.6. Although we did not show a single DOE system with this target we can see very well that the resulting intensity has more definition along the edges in the bottom right in Figure 4.6. The middle of the generated intensity is still not filled, but the shape is more recognizable. What is also interesting is that both DOEs resemble the target pattern, but that the first DOE seems to focus on more coarse edge patterns, whereas the second DOE focusses along the whole edge.

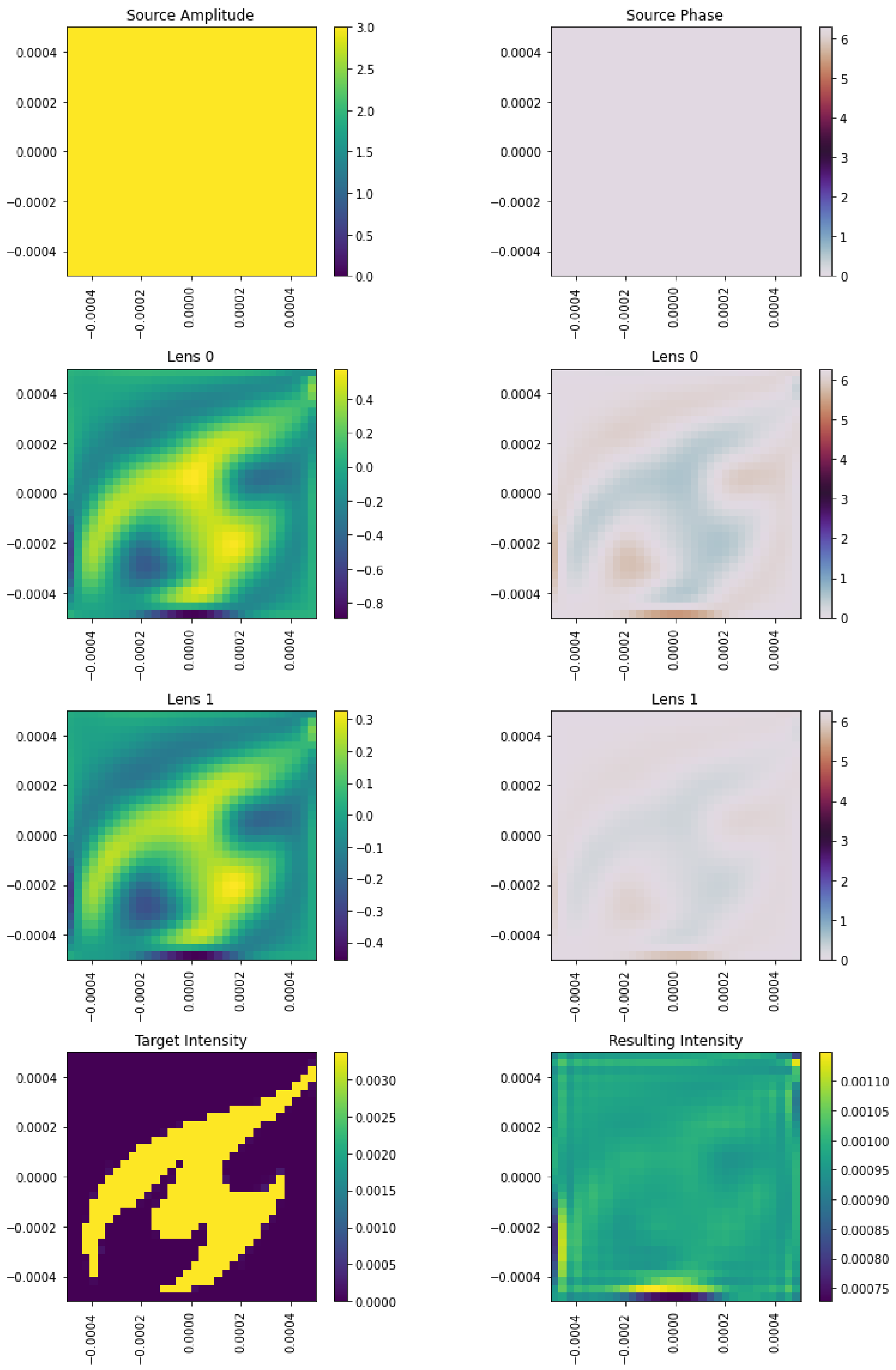


Figure 4.7: Optimized double NURBS lens system. The first row shows the source field, the middle two rows show the two NURBS generated phases and the last row shows the target and generated intensity. Note that the lower right intensity is not on the same scale and that with the correct scale the small effect seen here is not visible.

For the optimized NURBS system in Figure 4.7 the same happens as before as seen. The output intensity is so low that we have to change the range to make it visible. We see again the target intensity in the two lenses, this time there is not a particular focus on certain sections in one of the lenses. This could be caused by the restrictiveness of the NURBS.

We found that for a logo like the TUD logo, no more than two lenses are required as the third lens did not improve the output visually. We also found that the best approach was to optimize both lenses at the same time, instead of optimizing the first and adding the second one later. The idea of this approach was that the first lens would capture the main image and the second lens would improve details, however this proved futile and the second lens would not change the output intensity further. Although these results are visually confirmed, an additional lens does decrease the loss after optimization as seen in Figure 4.8.

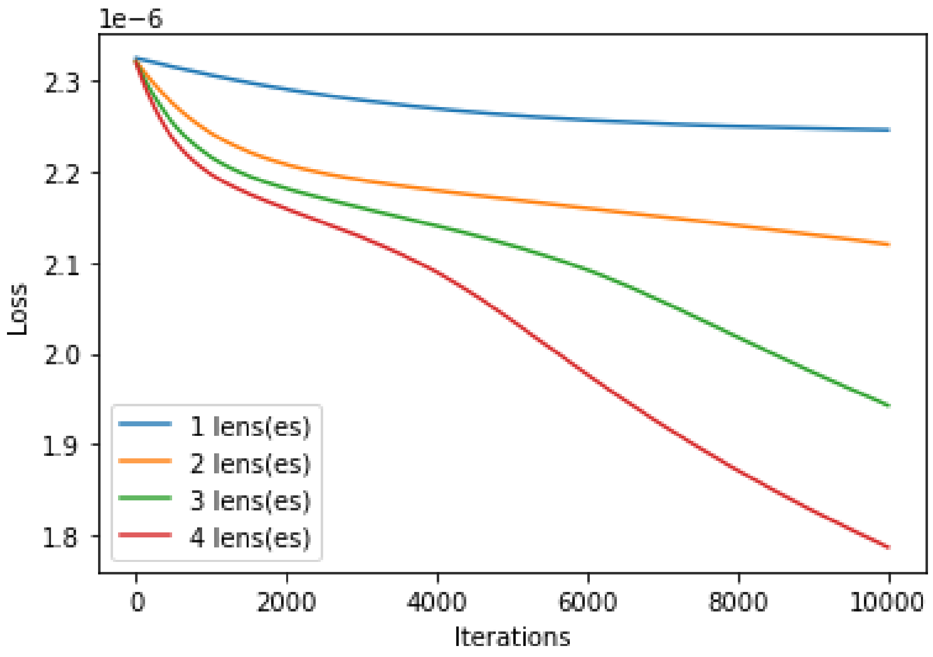


Figure 4.8: The loss function over the iterations for a different number of lenses in the system. Note that every iteration becomes more expensive with more lenses as more propagations must be done.

As the main computational costs of the optimization is the simulation of the diffraction and adding a lens adds a diffraction, the computation time increases when adding lenses. Thus adding an extra lens in our computation allows generating intensities more similar to the target, but it comes at an optimization cost.

4.1.3. High resolution

Some sources, for example a Gaussian beam, require a high resolution to have a realistic propagation. It is therefore interesting to see how well our previous methods work for higher resolution lenses. Higher resolution has two main consequences, the first is that optimization steps are longer as per step we need to simulate the light propagation to calculate the loss and the gradients, which becomes more expensive for larger matrices. A second consequence is that for the DOEs, we optimize for every lens pixel and thus with a higher resolution we have to optimize more phase changing pixel and thus more parameters. In this section we return to a setting with one source, one lens and one target intensity, but with a high resolution.

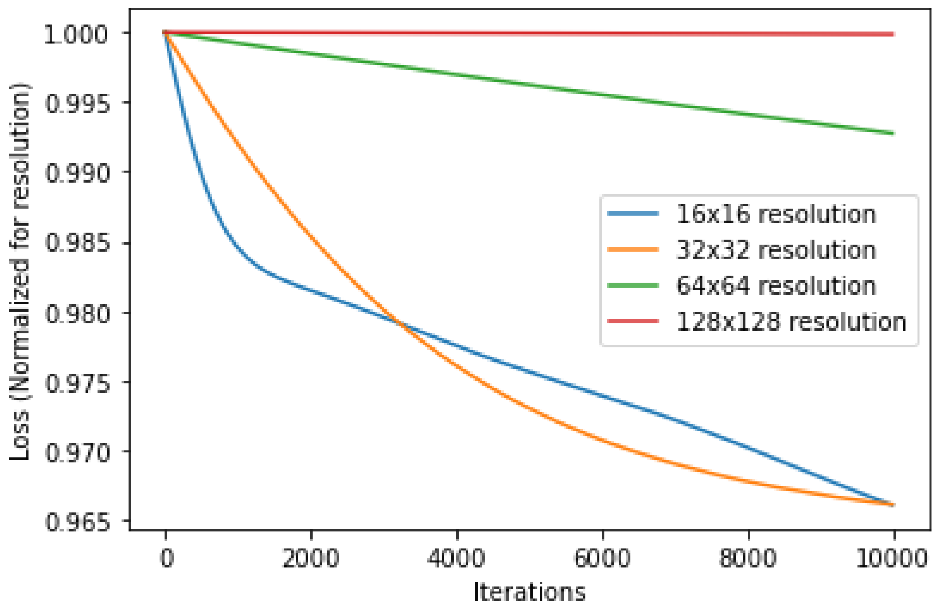


Figure 4.9: The iterations versus the loss for different resolutions. As the loss is resolution depended we scale the loss at the first iteration to be 1.0.

Looking at Figure 4.9, compared to a low resolution such as 32×32 it almost seems as if a resolution four times higher of 128×128 does not decrease in loss. This scale difference clearly shows the difficulty in optimizing for higher resolution. For certain settings, we are able to optimize optical system with DOEs with a resolution of 128×128, but even this resolution is a challenge. Figures made at that resolution are those in section 4.3

4.2. Data-driven methods for optics

As we saw in the previous section and in Chapter 2, for some sources we need a better optimization procedure for high resolution systems. Therefore, we use a data-driven approach to solve this problem. We use the UNet architecture discussed in Section 2.3.9. The input of the network is the target intensity and we train the output to be the DOE.

The UNet architecture originally stems from application in medical imaging such as segmentation, where a probability is given to each pixel if it belongs to a certain class. We use the UNet in a different capacity as a generative network to generate our DOE surfaces. As convolutional neural networks with upsampling have been successful in generative tasks we feel that the UNet should be up for the task as the decoder has exactly such a structure. We use the standard UNet structure as described in [56], we change the number of down- and up-sampling layers to be three.

To train the network we need to specify the loss function. A simple loss function would be a supervised loss: we would start with a certain phase plate, calculate the intensity distribution of this phase plate (call this the target distribution). This target distribution is the input of the network. The loss will be the difference between starting phase plate and the phase plate that the network outputs. This method is supervised as we need to know the phase plate to begin with, which is usually not known in the system we optimize. Another method is the unsupervised physics informed method. We start with a target intensity and the network guesses a phase plate with this intensity. We then use this phase plate to calculate the resulting intensity. We compare the two intensities to each other. This method is unsupervised as it does not need a beginning phase plate. In experiments we actually see that both methods perform reasonably well and we see that combining them might be beneficial. Thus we combine the two losses and scale them such that they are both of equal magnitude. Adding both methods does make the whole system supervised, but in the training phase that is not a problem as we generate training data ourselves.

$$\begin{aligned} \textit{Loss} &= \omega\mathcal{L}_{\text{sup}} + \mathcal{L}_{\text{phy}} \\ \mathcal{L}_{\text{sup}}(\textit{target phase}, \textit{target intensity}) &= ||\textit{target phase} - f_\theta(\textit{target intensity})||_2^2 \\ \mathcal{L}_{\text{phy}}(\textit{target intensity}) &= ||\textit{target intensity} - \textit{propagate}(f_\theta(\textit{target intensity}))||_2^2, \end{aligned}$$

here, $\mathcal{L}_{\text{phy}}$ denotes the physics informed loss, which clearly depends on the propagation of the physics and $\mathcal{L}_{\text{sup}}$ denotes the supervised loss, which does not depend on the physics directly through the loss.

An overview of the results shown here is found in Table 4.4.

Table 4.4: In this table for each figure in the results the optical settings are reported. The resolution described holds for all elements in the system. When a DOE is not based on a NURBS surface this is noted in the sixth column.

Figure	Resolution	Source type	Number of sources	Number of DOEs	NURBS	Target
4.10	512×512	Point source	1	1	✗	FMNIST
4.11	512×512	Point source	1	1	✗	TU Delft flame
4.12	256×256	Point source	1	1	✗	FMNIST
4.13	1024×1024	Point source	1	1	✗	FMNIST

We are able to train a network in a relatively simple setting. We use the FMNIST dataset to be our phases which we train with. We also experimented with training with other types of input phases, such as totally random, but the model does not train well on data that is unstructured such as random data. We use that standard parameters for the Adam optimizer. We find the result of this in Figure 4.10.

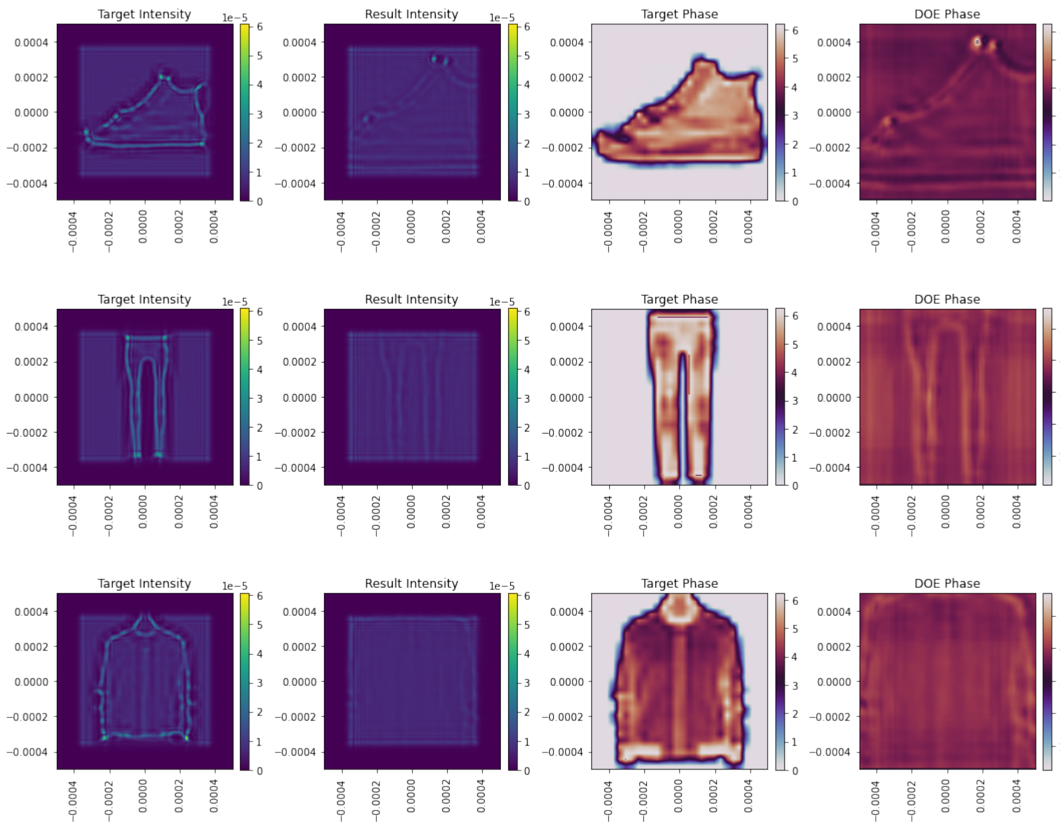


Figure 4.10: Results from the UNet architecture given the target intensity as input this gives us the result phase which we propagate to find the result intensity. The first column shows the different target intensities generated with the target phases in the third column, the second column shows the resulting intensities generated with the phases in the fourth column. We see a weird effect where the network output looks enlarged compared to the ground truth.

Figure 4.10 shows that the UNet is able to capture at least the essence of every picture. We see that both the DOE phase and the resulting intensity look like the targets. We also see a weird enlarging effect, this effect remains with more training or other sources which is remarkable.

We also find that the UNet performs very well on data not similar to the training or validation data. Although above pictures are already generated from the validation set and thus unseen data, they are the same structure. Applying this approach to the TU Delft flame intensity, we find that it generates light similar to this target in 4.11.

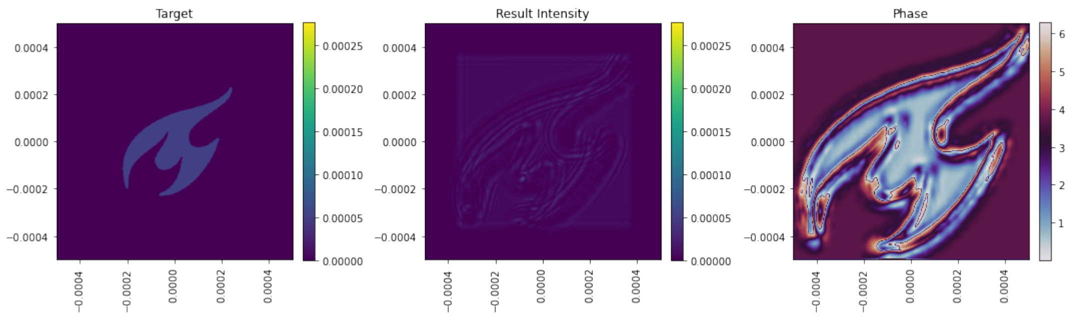


Figure 4.11: Applying the UNet to the TU Delft flame logo, which has a completly different structure than the trainings images. We again see an enlargement. The left subfigure shows the target intensity, the middle shows the intensity generated by the output phase of the network which is seen in the right image.

For the TU Delft flame we see that the model is making a DOE and intensity as expected. The enlargement seen on the FMNIST set is seen here as well.

We also tried to make the system able to generalize to different sources, but were not successful yet. We propose two ideas for this system which handle the source in a different way. When representing the source we could see it as a two channel image, where one channel is the phase of the source and the other channel is the amplitude of the source. Our first proposal to put this in a UNet would be to concatenate these two extra channels to the input channel that we already have from the target intensity. However, as the source and target intensity have different information not connected to each other in the encoder part, we also propose to add a second encoder arm to the UNet for the source.

An interesting property that the UNet approach has, is that although we trained the system for a resolution of 512×512 pixels, the system works fairly well on other resolutions such as 256×256 or 128×128 as long as the target intensity in this system does not contain to many artifacts from the optical propagation at low resolution. This can be seen in Figure 4.12.

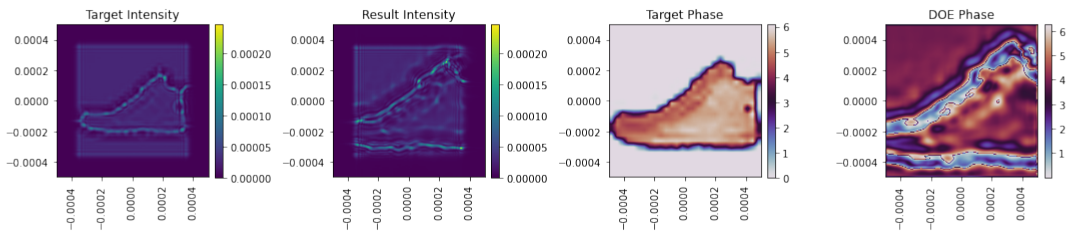


Figure 4.12: Using a network trained for 512×512 images applied to 256×256 resolution images. The first image shows the target intensity, which is the network input, generated by the phase in the third image. The second image is generated by phase seen in the last image, which is the network output.

More interesting than going from high to low resolution would be the other way around. Optics simulation at a high resolution are very expensive. Our UNet solves this problem as it does not iterate for optimization after it has been trained. On top of this, as the UNet is able to handle higher resolutions that really shows potential in this method.

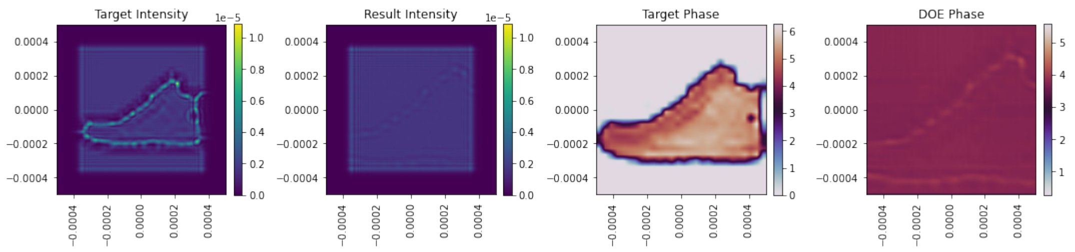


Figure 4.13: Using a network trained for 512×512 images applied to 1024×1024 resolution images. The first image shows the target intensity, which is the network input, generated by the phase in the third image. The second image is generated by phase seen in the last image, which is the network output.

In Figure 4.13, we see that the UNet is able to generalize to higher resolution as well as the two plots on the left show a similar intensity. Although the image is not as clear, there is potential. Comparing to the result we have of the UNet evaluated on the same resolution as training, the result intensity and phase show less contrast than the intensity and phase from the lower resolution.

We actually saw that in other literature that neural networks are used to generate starting points for the optimization ([72]). As can be seen from our results as well, the neural networks do produce output that is for a system that does not require iterations, however applying an iterative optimizer might be beneficial so the lens is able to capture more small details. We found that gradient descent does not handle high resolution optimization, even given a starting point. That gradient descent works worse on high dimensions was expected from Figure 4.9, but we would have hoped that gradient descent would be possible when closer to the optimum.

Another network we tried was a neural network with NURBS parameters describing the surface of the lens as output. However, during research we found that it was not possible to train a network that outputted the correct parameters. We first tried without optical simulation, thus the input of the network was a surface (or matrix of z-coordinates) and the output of the network should be the z-coordinates of the B-spline surface (we fixed the other coordinates). As for B-splines we know that their neighboring points affect each other more than far away points we tried a convolutional encoder network. For the decoder we tried two thing: a fully connected decoder and a UNet decoder. As the output points also have a spatial structure we hoped this would work as well. Unfortunately, this approach did not work. We do not have a clear explanation, but clearly the B-spline structure was too hard for the network.

4.3. Incoherent caustic design for multiple sources

We saw that extending the system with multiple lenses, allows the system to optimize for more difficult target intensities. In this section we consider a system with multiple sources. A system with multiple sources, like a system with multiple lenses, has different properties. Optimizing with a second DOE the generated intensities had more details, does the same thing also happen with a second source? We are also interested to see if one system can be optimized to generate different target intensities when a different source is turned on, this would resemble the inverse of the described diffractive neural network classifiers from [36].

An overview of the results in this section is given in Table 4.5

Table 4.5: In this table for each figure in the results the optical settings are reported. The resolution described holds for all elements in the system. When a DOE is not based on a NURBS surface this is noted in the sixth column.

Figure	Resolution	Source type	Number of sources	Number of DOEs	NURBS	Target
4.14	128×128	Point source	1	1	✗	Square
4.17	128×128	Point source	2	1	✗	TU Delft flame
4.18	128×128	Point source	3	1	✗	Multiple
4.20	128×128	Point source	3	2	✗	Multiple

To get an intuition what happens when a source is added we simulate a system with a square in Figure 4.14.

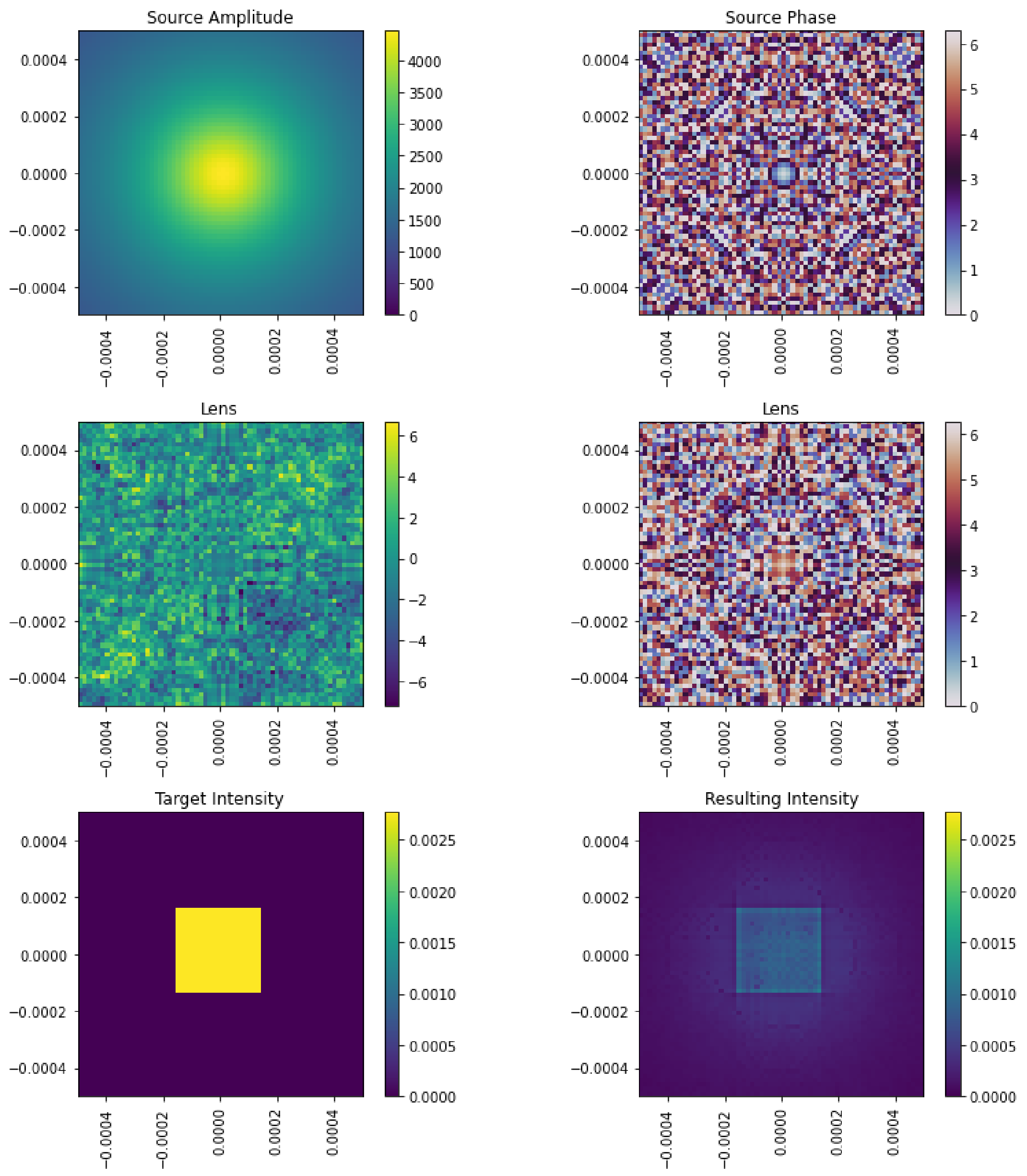


Figure 4.14: Single point source single DOE system optimized to form a square intensity. The first row shows the source fields, the second row shows the phase of the DOE and the last row shows the target and resulting intensity after optimization.

As we work with incoherent light a second source is not correlated with the first, thus moving the source around and comparing resulting intensities shows what happens if a second source is added. We expect the intensity to also translate with the source.

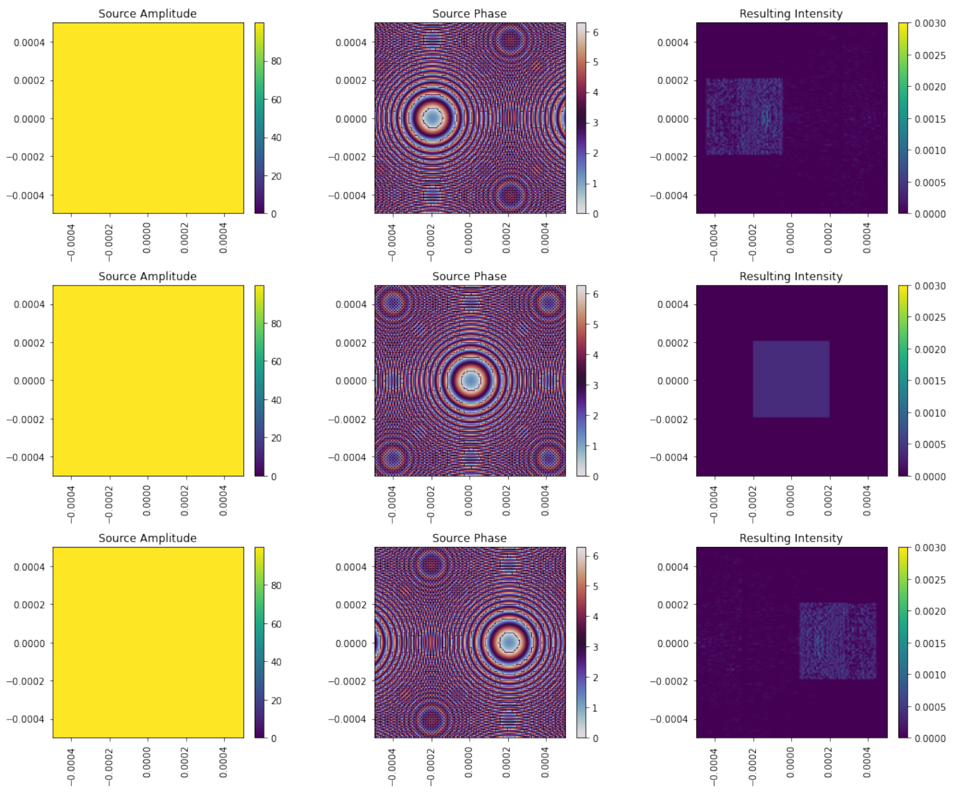


Figure 4.15: Using the previous DOE and moving the point source horizontally. The first column shows the amplitude of the three different sources, the second shows the phase of the sources and the last column shows the intensity generated by the combination of source and the phase from Figure 4.14.

In Figure 4.15, we see that as the source translates the intensity also translates as expected. Therefore, we can make a system with a single lens and multiple sources that repeats its intensity in multiple places as Figure 4.16 shows.

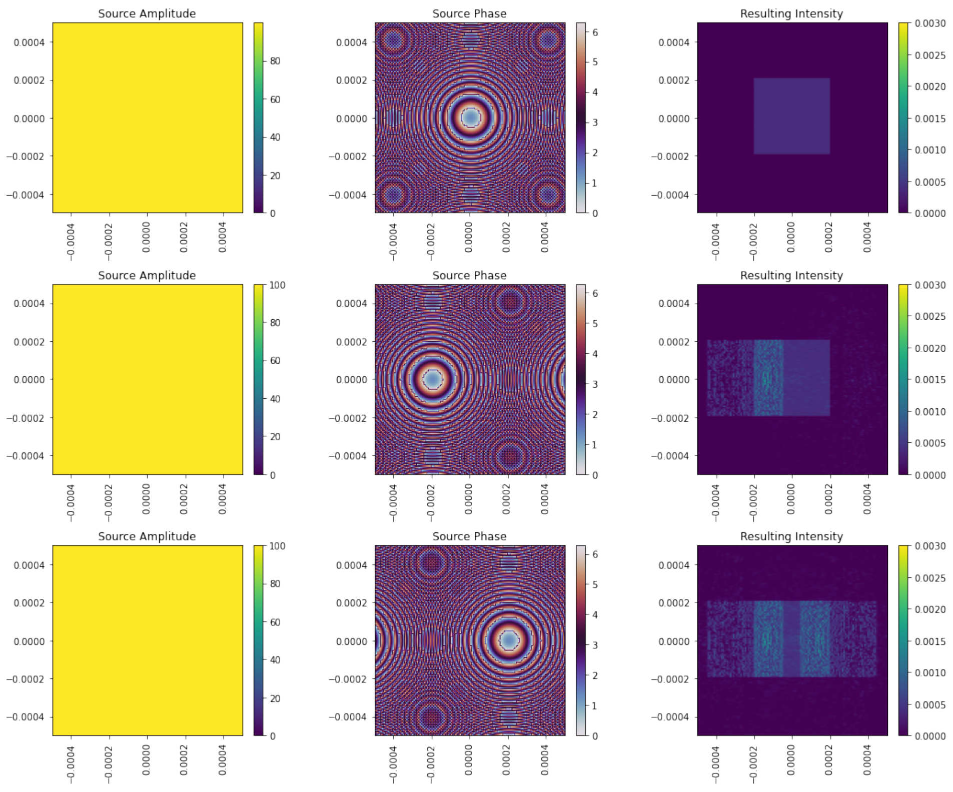


Figure 4.16: Using a DOE that generates a square intensity and adding a source every row. The first column shows the amplitude of the sources, the second column shows the phase of the sources and the third column shows the cumulative intensity from adding the source every row to the system.

We thus see that with multiple sources we could create a whole connected intensity to be used in, for example, street lighting. A single lens system could be used in combination with multiple sources to create a long uniform intensity along the road.

A system with multiple sources can also be optimized. We revisit the TU Delft flame logo and optimize it with two sources.

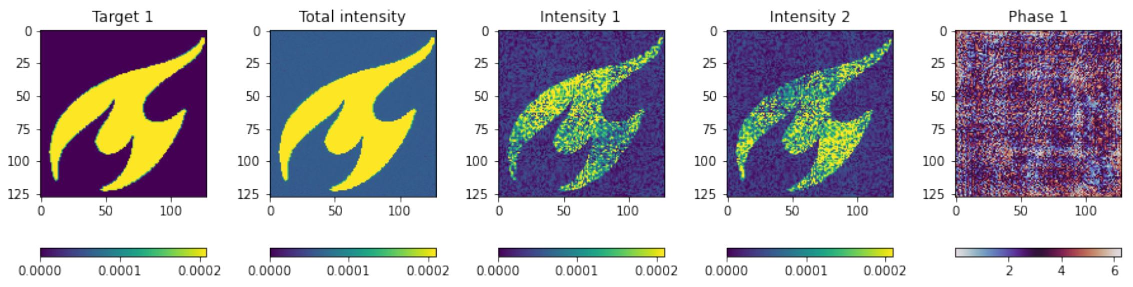


Figure 4.17: Optimizing a system with one lens and two sources for a single target intensity. The first two images show the target and the resulting intensity generated by adding the two intensities in the third and fourth image. These intensities are generated by propagation two sources trough the phase in the last image.

In the optimization in Figure 4.17 it seems that the lens divides the light of the two sources as if every other pixel belongs to one source to generate the intensity. This gives the two partial intensity a granular look. This result was not expected due to the limit to the system as it has only one lens. We also see that the intensity shown in the left two figures of Figure 4.17 are very similar and that also the middle is filled of the resulting intensity in the second subfigure.

Although in the street light example this shifting of intensity distribution with moving of the source and optimizing a system for one target seem nice, we could also take it one step further. We optimize a system that generates two different target intensity distributions depending on which source is turned on. We first consider only one lens and later extend this to multiple lenses.

Optimizing for one lens is difficult as we saw already that changing the source position only shifts the intensity distribution. The outcome of optimizing for three different sources and intensities is shown in Figure 4.18.

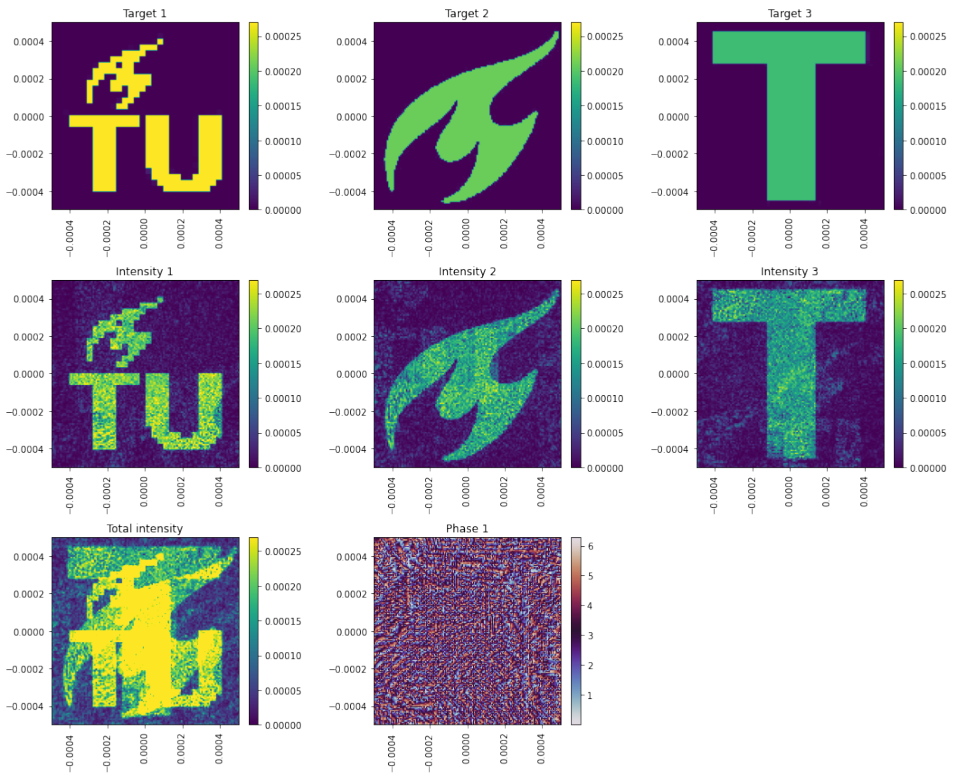


Figure 4.18: Optimizing one lens to have a different intensity pattern based on where the source is. The first row shows the three different targets, the second row shows the resulting intensities from each source and the last row shows the cumulative intensity of all sources and the phase to generate all these intensities.

The result shown in Figure 4.18 was not expected, as it should not be possible to generate two different patterns when the source is only changing its position.

What is then interesting to see is that when moving the source from one location to the other what would happen.

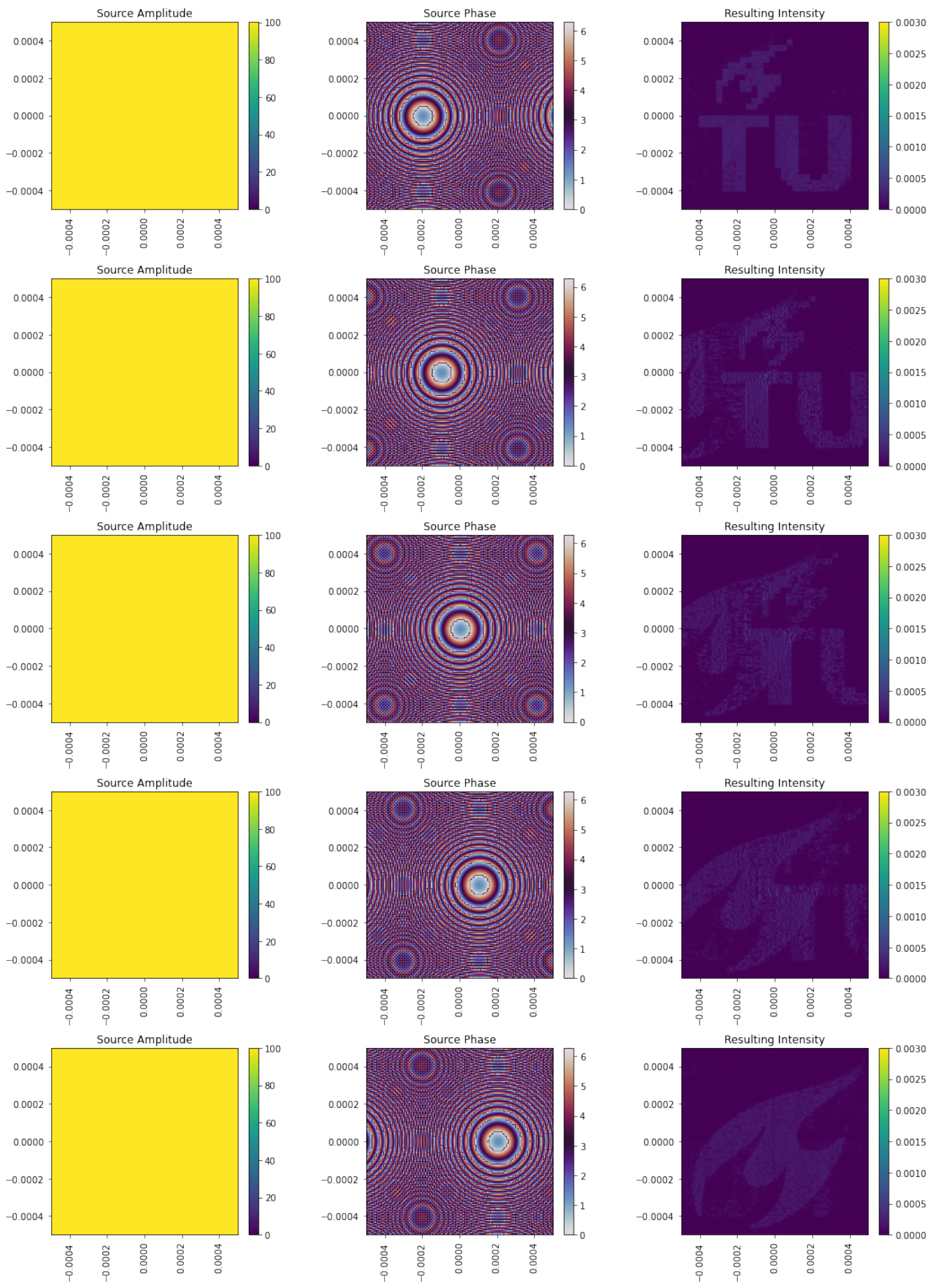


Figure 4.19: Moving the source between the two optimized points give an interpolation of the two target distribution. The first column shows the amplitude of the sources, the second column shows the phase of the sources and the last column shows the resulting intensity generated by the phase seen in Figure 4.18

We see in Figure 4.19, that the intensity shifts over the sample plane and our explanation for this result is that the two intensities created by the different sources shift outside the sampling plane when shifting from one source to the next. Thus there would not be a conflict in generating two intensities.

Adding a lens should give the system a lot more options and the light should not shift around. Adding a second lens does increase the number of parameters to be optimized, but it should have a positive impact on our results.

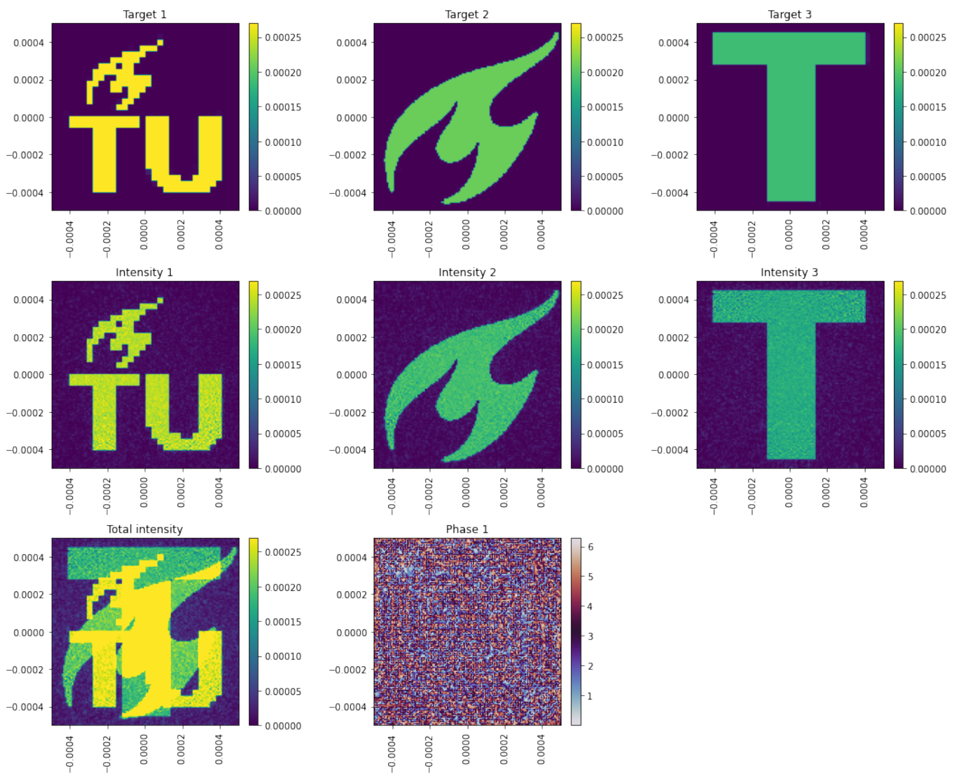


Figure 4.20: Optimizing a system with three sources, two lenses and three different target illumination patterns. The first row shows the three different targets, the second row shows the resulting intensities from each source and the last row shows the cumulative intensity of all sources and the phases to generate all these intensities.

In Figure 4.20 the result from simulation with two DOEs is seen. When comparing the one DOE system (Figure 4.18) and two DOEs system, there is less noise in the two DOEs system.

5 Conclusion

In this thesis we investigated the optimization of optical systems with diffractive optical elements. We started by introducing optical theory as created in the study of wave optics. This theory is derived from the Maxwell equations and uses that light at a certain point is the sum of all light at an earlier point in the system. We use the Rayleigh-Sommerfeld diffraction in combination with the angular spectrum method to simulate our optics.

To optimize the system we pursue two directions. We discuss modern methods in gradient descent algorithms, specifically the Adam optimizer. Neural networks are discussed as a type of data driven approach to solve our optimization problem. We have particular focus on convolutional neural networks such as UNets. Another field we visit is the field of parametrizable surfaces, this gives us possible lens describing surfaces.

We combine this all to answer the question: How can we optimize a system consisting of diffractive optical elements efficiently?

We show that it is possible to optimize either diffractive optical elements or the lens parameters for a NURBS generated lens in our optical setup. We show that for different sources of light this is possible. Multiple lenses behind each other allow to generate more detailed intensities. In other experiments We were able to use data-driven methods to generate lenses, which circumvented the issues created by high resolution and have a potential to speed up the process. We also saw the possibilities of using optical optimization with multiple sources.

The UNet not only showed that it is capable of generating the correct DOE for the target intensity, but also showed that it can be used on even higher resolutions than what it was trained on. The UNet has potential and can be extended, but as we tried to do this with a variable source term as input this proved to be hard.

Our other experiments concerned multiple sources. In this setting we were able to optimize with one lens, but the result got better with two lenses. However, when we shift the source from one side to the other, the intensity slowly fades in and out of the sample plane. We were not able to correct for this and in this current setting if the sample plane would be made larger the other intensity (belonging to the other source) should appear, as it is shifted of the small sample plane in the current setting. This is the desired effect as it, generates a truly different intensity given a different source. More research in this area should be done, as it also has potential.

Concluding, the Rayleigh-Sommerfeld diffraction provides us with an optics propagation method which is able to handle a wide variety of circumstances. Furthermore, in combination with gradient based solvers optical systems based on this diffraction are able to optimize, as long as the physical settings of the system are set correctly. These methods do not perform well in high resolution settings. We propose a data-driven method and show that this achieves desired results even for resolutions higher than the data it is trained with. We furthermore show how our optimization works in a system with multiple sources.

Looking forward, the field of optimization in optics with diffractive optical elements is not very large. There is a limited amount of literature. For applications for our research, one could think of virtual reality but also about optical computers. The optical field is also largely unfamiliar with data-driven methods and current methods in optics to solve inverse problems can be solved more efficient with data-driven methods as we showed. Moreover, multiple sources also prove to be interesting in to generate different target intensities. Our analysis of multiple sources was limited to a small number of experiments and suffered from optical effects outside the sampling plane and thus is also an interesting topic to improve on this in the future. The combination of the field of (computational) optics and modern optimization techniques have a bright future ahead of them.

References

[1] Harald Aagedal et al. “Theory of speckles in diffractive optics and its application to beam shaping”. In: *http://dx.doi.org/10.1080/09500349608232814* 43.7 (1996), pp. 1409–1421. ISSN: 13623044. DOI: `10.1080/09500349608232814`.

[2] A A Belousov, L L Doskolovich, and S I Kharitonov. “A gradient method of designing optical elements for forming a specified irradiance on a curved surface”. In: (2008).

[3] J. H. M. ten Thije Boonkkamp, L. B. Romijn, and W. L. IJzerman. “Freeform lens design for a point source and far-field target”. In: *JOSA A, Vol. 36, Issue 11, pp. 1926-1939* 36.11 (Nov. 2019), pp. 1926–1939. ISSN: 1520-8532. DOI: `10.1364/JOSAA.36.001926`.

[4] Christoph Bösel and Herbert Gross. “Double freeform illumination design for prescribed wavefronts and irradiances”. In: *Journal of the Optical Society of America A* 35.2 (Feb. 2018), p. 236. ISSN: 1084-7529. DOI: `10.1364/JOSAA.35.000236`.

[5] C Brecher et al. *NURBS Based Ultra-Precision Free-Form Machining*. Tech. rep. 2006.

[6] Dmitry A Bykov et al. “Linear assignment problem in the design of freeform refractive optical elements generating prescribed irradiance distributions”. In: (2018). DOI: `10.1364/OE.26.027812`.

[7] Ozan Cakmakci et al. “4830) Optical systems design”. In: 220 (2008).

[8] Hang Chen et al. “Diffractive Deep Neural Networks at Visible Wavelengths”. In: *Engineering* 7.10 (2021), pp. 1483–1491. ISSN: 20958099. DOI: `10.1016/j.eng.2020.07.032`.

[9] Xianming Chen, Richard F. Riesenfeld, and Elaine Cohen. “An algorithm for direct multiplication of B-splines”. In: *IEEE Transactions on Automation Science and Engineering*. Vol. 6. 3. July 2009, pp. 433–442. DOI: `10.1109/TASE.2009.2021327`.

[10] Dewen Cheng et al. “Design of an optical see-through head-mounted display with a low f-number and large field of view using a freeform prism”. In: (2009).

[11] Pavel Cheremkhin et al. “Machine learning methods for digital holography and diffractive optics”. In: *Procedia Computer Science* 169 (2020), pp. 440–444. ISSN: 18770509. DOI: `10.1016/j.procs.2020.02.243`.

[12] Geoffroi Côté, Jean-François Lalonde, and Simon Thibault. “Deep learning-enabled framework for automatic lens design starting point generation”. In: *Optics Express* 29 (3 Feb. 2021), p. 3841. ISSN: 1094-4087. DOI: `10.1364/oe.401590`.

[13] L H Crijns. *PINN inspired Freeform Design Using Fraunhofer Diffrac-tion to ond Freeforms de-scribed by B-spline Sur-faces*. Tech. rep. 2021.

[14] Yi Ding et al. “Freeform LED lens for uniform illumination References and links”. In: (2008).

[15] F. Z. Fang et al. “Manufacturing and measurement of freeform optics”. In: *CIRP Annals - Manufacturing Technology* 62.2 (2013), pp. 823–846. ISSN: 00078506. DOI: `10.1016/j.cirp.2013.05.003`.

[16] J. R. Fienup. “Phase retrieval algorithms: a comparison”. In: *Applied Optics, Vol. 21, Issue 15, pp. 2758-2769* 21.15 (Aug. 1982), pp. 2758–2769. ISSN: 2155-3165. DOI: `10.1364/AO.21.002758`.

[17] G W Forbes. “Characterizing the shape of freeform optics References and links”. In: (2012).

[18] R W Gerchberg and W O Saxton. *A Practical Algorithm for the Determination of Phase from Image and Diffraction Plane Pictures*. Tech. rep. 2. 1969, pp. 237–246.

[19] R W Gerchberg and W O Saxton. *A Practical Algorithm for the Determination of Phase from Image and Diffraction Plane Pictures*. 1969, pp. 237–246.

[20] J.W. Goodman. *Introduction to Fourier Optics*. Tech. rep. 2005.

[21] Paul Hand, Oscar Leong, and Vladislav Voroninski. “Phase Retrieval Under a Generative Prior”. In: (2018).

[22] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[23] Ryoichi Horisaki, Ryosuke Takagi, and Jun Tanida. “Deep-learning-generated holography”. In: *Applied Optics* 57.14 (May 2018), p. 3859. ISSN: 1559-128X. DOI: `10.1364/ao.57.003859`.

[24] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5 (1989), pp. 359–366.

[25] M. Hossein Eybposh et al. “DeepCGH: 3D computer-generated holography using deep learning”. In: *Optics Express* 28.18 (Aug. 2020), p. 26636. ISSN: 1094-4087. DOI: `10.1364/oe.399624`.

[26] T. J.R. Hughes, J. A. Cottrell, and Y. Bazilevs. “Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement”. In: *Computer Methods in Applied Mechanics and Engineering* 194 (39-41 Oct. 2005), pp. 4135–4195. ISSN: 00457825. DOI: `10.1016/j.cma.2004.10.008`.

[27] Joost Imhof. *Freeform lens predictions by a Neural Network and B-splines*. Tech. rep. 2020.

[28] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[29] Gareth James et al. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[30] Jinbo Jiang et al. “Optical design of a freeform TIR lens for LED streetlight”. In: *Optik* 121 (19 Oct. 2010), pp. 1761–1765. ISSN: 0030-4026. DOI: 10.1016/J.IJLEO.2009.04.009.

[31] Jinbo Jiang et al. “Optical design of a freeform TIR lens for LED streetlight”. In: *Optik* 121.19 (Oct. 2010), pp. 1761–1765. ISSN: 0030-4026. DOI: 10.1016/J.IJLEO.2009.04.009.

[32] X. Jane Jiang and Paul J. Scott. “Free-form surface reconstruction”. In: *Advanced Metrology*. Elsevier, 2020, pp. 93–127. DOI: 10.1016/b978-0-12-821815-0.00005-8.

[33] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[34] Zongyi Li et al. “Fourier Neural Operator for Parametric Partial Differential Equations”. In: (Oct. 2020).

[35] Ku Chin Lin. “Designation of lenses with a single freeform surface for multiple point sources”. In: *Journal of the Optical Society of America. A, Optics, image science, and vision* 29.3 (Mar. 2012), p. 200. ISSN: 1520-8532. DOI: 10.1364/JOSAA.29.000200.

[36] Xing Lin et al. “All-optical machine learning using diffractive deep neural networks”. In: (2018).

[37] Xuhui Meng et al. “PPINN: Parareal Physics-Informed Neural Network for time-dependent PDEs”. In: (Sept. 2019). DOI: 10.1016/j.cma.2020.113250.

[38] Deniz Mengu, Yair Rivenson, and Aydogan Ozcan. “Scale-, Shift-, and Rotation-Invariant Diffractive Optical Networks”. In: *ACS Photonics* 8.1 (Jan. 2021), pp. 324–334. ISSN: 23304022. DOI: 10.1021/acsphotonics.0c01583.

[39] Deniz Mengu et al. “Analysis of Diffractive Optical Neural Networks and Their Integration with Electronic Neural Networks”. In: *IEEE Journal of Selected Topics in Quantum Electronics* 26.1 (Oct. 2018). DOI: 10.1109/JSTQE.2019.2921376.

[40] Deniz Mengu et al. “Classification and reconstruction of spatially overlapping phase images using diffractive optical networks”. In: (2021).

[41] Christopher A Metzler et al. “prDeep: Robust Phase Retrieval with a Flexible Deep Network”. In: (2018).

[42] Preetum Nakkiran et al. “Deep double descent: where bigger models and more data hurt”. In: *J. Stat. Mech* (2021), pp. 1742–5468. DOI: 10.1088/1742-5468/ac3a74.

[43] Donald C O et al. *Diffractive Optics: Design, Fabrication, and Test*. 2004.

[44] Aydogan Ozcan et al. “Misalignment resilient diffractive optical networks”. In: *Nanophotonics* 9.13 (Oct. 2020), pp. 4207–4219. ISSN: 21928614. DOI: 10.1515/nanoph-2020-0291.

[45] C. Paterson. “Diffractive Optical Elements with Spiral Phase Dislocations”. In: *http://dx.doi.org/10.1080/09500349414550771* 41.4 (Apr. 1994), pp. 757–765. ISSN: 13623044. DOI: 10.1080/09500349414550771.

[46] Yifan Peng et al. “Learned large field-of-view imaging with thin-plate optics”. In: *ACM Transactions on Graphics* 38.6 (Nov. 2019). ISSN: 15577368. DOI: 10.1145/3355089.3356526.

[47] Les Piegl and Wayne Tiller. “The NURBS Book”. In: (1995). DOI: 10.1007/978-3-642-97385-7.

[48] Les Piegl and Wayne Tiller. “The NURBS Book”. In: Monographs in Visual Communications (1995). DOI: 10.1007/978-3-642-97385-7.

[49] Md Sadman Sakib Rahman et al. “Ensemble learning of diffractive optical networks”. In: *Light: Science and Applications* 10.1 (Dec. 2021). ISSN: 20477538. DOI: 10.1038/s41377-020-00446-w.

[50] M. Raissi, P. Perdikaris, and G. E. Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational Physics* 378 (Feb. 2019), pp. 686–707. ISSN: 0021-9991. DOI: 10.1016/J.JCP.2018.10.045.

[51] Haoran Ren et al. “Three-dimensional vectorial holography based on machine learning inverse design”. In: (2020).

[52] Harald Ries and Julius Muschaweck. “Tailored freeform optical surfaces”. In: (2002).

[53] Harald Ries and Julius A. Muschaweck. “Tailoring freeform lenses for illumination”. In: *Novel Optical Systems Design and Optimization IV*. Vol. 4442. SPIE, Dec. 2001, pp. 43–50. DOI: 10.1117/12.449957.

[54] Yair Rivenson, Yichen Wu, and Aydogan Ozcan. “Deep learning in holography and coherent imaging”. In: *Light: Science and Applications* 8.1 (Dec. 2019). ISSN: 20477538. DOI: 10.1038/s41377-019-0196-0.

[55] Yair Rivenson et al. “Phase recovery and holographic image reconstruction using deep learning in neural networks”. In: *Light: Science and Applications* 7.2 (Feb. 2018), undefined–undefined. ISSN: 20477538. DOI: 10.1038/LSA.2017.141.

[56] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. DOI: 10.48550/ARXIV.1505.04597.

[57] E. Savio, L. De Chiffre, and R. Schmitt. “Metrology of freeform shaped parts”. In: *CIRP Annals - Manufacturing Technology* 56 (2 2007), pp. 810–835. ISSN: 00078506. DOI: 10.1016/j.cirp.2007.10.008.

[58] E. Savio, L. De Chiffre, and R. Schmitt. “Metrology of freeform shaped parts”. In: *CIRP Annals - Manufacturing Technology* 56.2 (2007), pp. 810–835. ISSN: 00078506. DOI: 10.1016/j.cirp.2007.10.008.

[59] Schlieder. *Learned Residual Gerchberg-Saxton Network for Computer Generated Holography*. Tech. rep. Sept. 2020, pp. 1–9.

[60] Yuliy Schwartzburg et al. “High-contrast computational caustic design”. In: *ACM Transactions on Graphics* 33.4 (2014). ISSN: 15577333. DOI: 10.1145/2601097.2601200.

[61] Fabin Shen and Anbo Wang. “Fast-Fourier-transform based numerical integration method for the Rayleigh-Sommerfeld diffraction formula”. In: (2006).

[62] Liang Shi et al. “Towards real-time photorealistic 3D holography with deep neural networks”. In: *Nature* 591.7849 (Mar. 2021), pp. 234–239. ISSN: 14764687. DOI: 10.1038/s41586-020-03152-0.

[63] Kai Wang et al. “Freeform LED lens for rectangularly prescribed illumination”. In: *Journal of Optics A: Pure and Applied Optics* 11.10 (Aug. 2009), p. 105501. ISSN: 1464-4258. DOI: 10.1088/1464-4258/11/10/105501.

[64] Kai Wang et al. “Freeform LED lens for uniform illumination”. In: *Opt. Express* 46.18 (2010), pp. 12958–12966.

[65] Sifan Wang, Hanwen Wang, and Paris Perdikaris. “Learning the solution operator of parametric partial differential equations with physics-informed DeepOnets”. In: *Science Advances* 7.40 (Mar. 2021). ISSN: 23752548. DOI: 10.1126/sciadv.abi8605.

[66] Gordon Wetzstein et al. “Inference in artificial intelligence with deep optics and photonics”. In: *Nature* 588.7836 (Dec. 2020), pp. 39–47. ISSN: 14764687. DOI: 10.1038/s41586-020-2973-6.

[67] Gershon Wolansky and Jacob Rubinstein. “Intensity control with a free-form lens”. In: *JOSA A, Vol. 24, Issue 2, pp. 463-469* 24.2 (Feb. 2007), pp. 463–469. ISSN: 1520-8532. DOI: 10.1364/JOSAA.24.000463.

[68] Rengmao Wu, José Sasián, and Rongguang Liang. “Algorithm for designing free-form imaging optics with nonrational B-spline surfaces”. In: *Applied Optics* 56.9 (Mar. 2017), p. 2517. ISSN: 0003-6935. DOI: 10.1364/ao.56.002517.

[69] Rengmao Wu et al. “Design of Freeform Illumination Optics”. In: *Laser & Photonics Reviews* 12.7 (July 2018), p. 1700310. ISSN: 1863-8899. DOI: 10.1002/LPOR.201700310.

[70] Hiromu Yakura et al. “Malware Analysis of Imaged Binary Samples by Convolutional Neural Network with Attention Mechanism”. In: *Conference: The 8th ACM Conference on Data and Application Security and Privacy*. Mar. 2018, pp. 127–134. DOI: 10.1145/3176258.3176335.

[71] Tong Yang, Dewen Cheng, and Yongtian Wang. “Direct generation of starting points for freeform off-axis three-mirror imaging system design using neural network based deep-learning”. In: *Optics Express* 27.12 (June 2019), p. 17228. ISSN: 1094-4087. DOI: 10.1364/oe.27.017228.

[72] Tong Yang, Guo Fan Jin, and Jun Zhu. “Automated design of freeform imaging systems”. In: *Light: Science and Applications* 6 (10 Oct. 2017). ISSN: 20477538. DOI: 10.1038/lsa.2017.81.

[73] Jingfei Ye et al. “Review of optical freeform surface representation technique and its application”. In: *Optical Engineering* 56.11 (Nov. 2017), p. 1. ISSN: 1560-2303. DOI: 10.1117/1.oe.56.11.110901.

[74] Zheng Zhenrong, Hao Xiang, and Liu Xu. “Freeform surface lens for LED uniform illumination”. In: (2009).

A
Comments on implementation and code

For this thesis we wrote a bunch of code which has been uploaded to the github repository of Alex Heemels at `https://gitlab.tudelft.nl/anmheemels/pinn-based-freeform-design/-/tree/Marek_branch/experiments`. What is mainly interesting in this code is the file under scripts and differentiable propagation called 'optical_field.py'. This file allows to build a differentiable optical field with as many sources/ lenses as one wants. Examples how to use this code can be found in the notebooks in the folder '\Execution_files'. Furthermore, in the execution files folder, the code for training the network can be found, as well as code for generating all pictures in this thesis.

An implementation for the neural network can be found in the folder with the scripts, where one can also find a NURBS implementation made by Bart de Koning, another master student in the group.

B
Extra figures

In the text we made some claims on optimizing multiple lenses at the same time and in what order one should optimize these lenses, here we show some extra images to verify our claims. The two claims were the following:

- Adding a third lens does not add a visual improvement over a two lens system
- First optimizing one lens and later adding the other to the optimization, does not help the optimization and with a similar amount of optimization steps the the resulting intensity of this method is visually worse than when optimizing both lenses at the same time.

We show the two lens system as a reference again

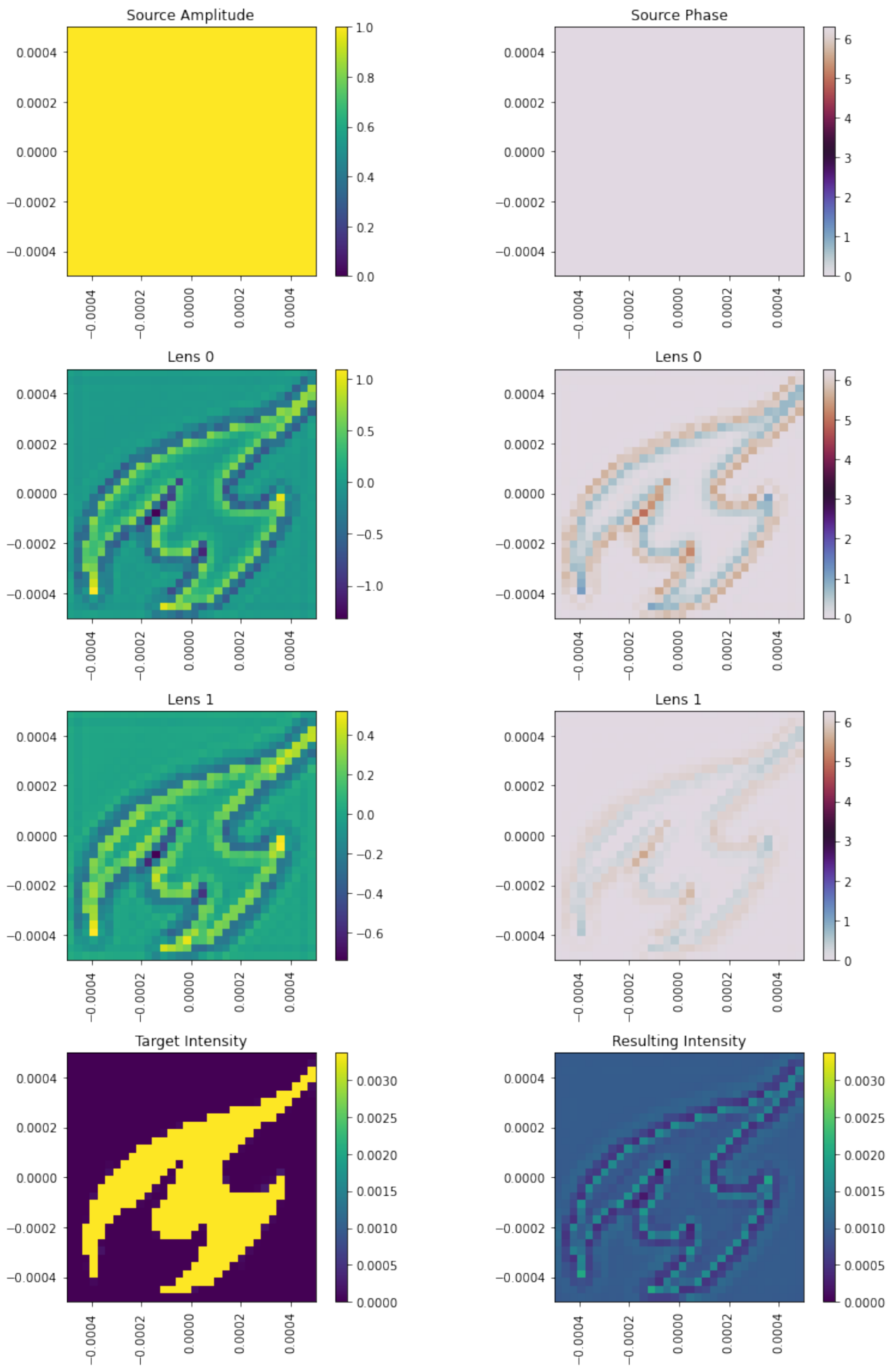


Figure B.1: A two DOE system given 10000 iterations.

We see the pattern that we saw already again. Two DOE's which both look similar to the target intensity.

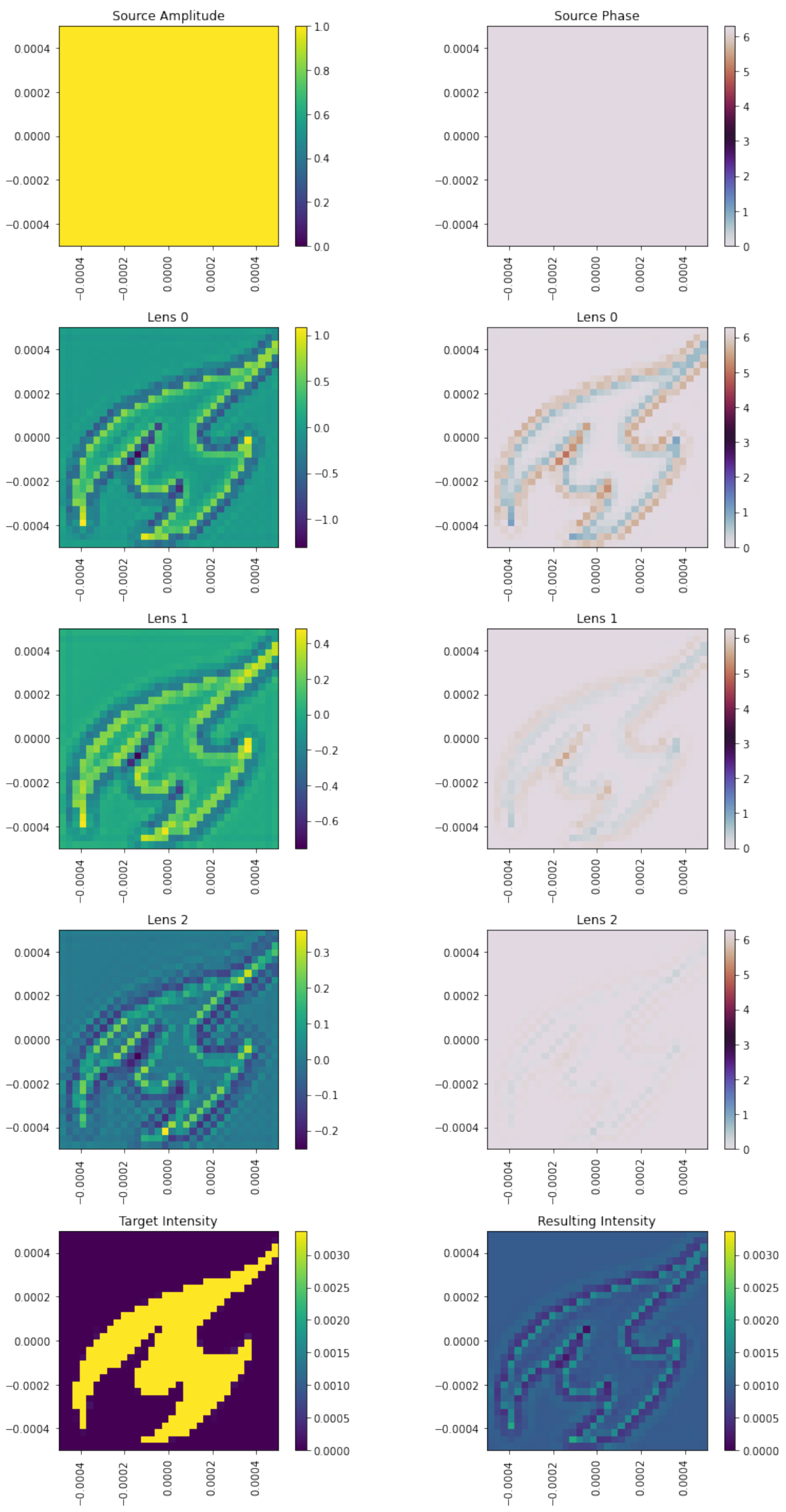


Figure B.2: A three DOE system given 10000 iterations.

We see that the third lens also follows the target intensity pattern. It does focus on other parts, but we also see that the phase is not very high. Visually the resulting intensity does not get a lot better. We did see however that the loss clearly decreases when adding multiple lenses. However, as the effect is not visually clear and it does have computational costs, we show mostly double DOE system results.

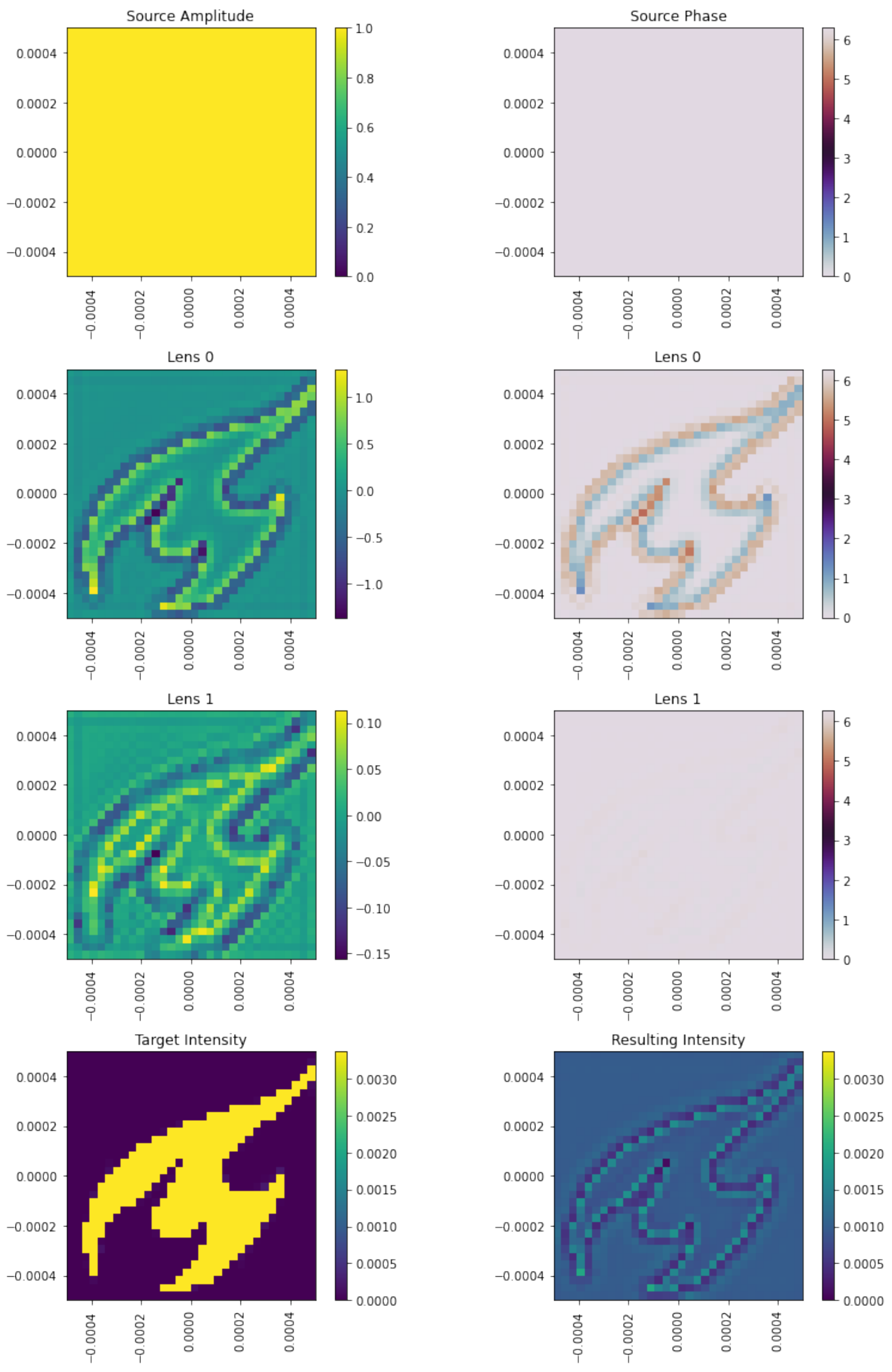


Figure B.3: Optimizing a two DOE system for 10000 iterations, where we add the second DOE to the optimization after 5000 iterations.

We see that the second DOE does start the target intensity as we would expect, but as we can see on the right hand side the effect is so small it becomes invisible there. We do see that if we give the complete system more iterations (including the second lens) that the lens does adapt. We conclude that it does not make sense to separate the lenses during optimization and that they should all be optimized at the same time.