Image Deblurring with Neural Networks Using Fourier Optics

Sihan Shao

1 Imaging Forward Model

Based on the paraxial and Fraunhofer approximations, the Point Spread Function (PSF) can be derived using a generalized pupil function $Q_{\lambda,z}(s,t)$. The blurred image $I_{s,z,\lambda}(x,y)$, which is captured by the sensor, can be mathematically described as the convolution between the ideal sharp image $I_{z,\lambda}(x,y)$ and the PSF:

$$I_{s,z,\lambda}(x,y) = I_{z,\lambda}(x,y) * h_{\lambda,z}(x,y), \tag{1}$$

where $*$ represents the convolution operation. To achieve this convolution-based imaging model, we define a generalized pupil function $Q_{\lambda,z}(s,t)$, which accounts for effects such as defocus and chromatic aberration. The PSF itself is obtained from the pupil function through the Fourier transform (Fraunhofer approximation):

$$h_{\lambda,z}(x,y) \propto \left| \mathcal{F}\{Q_{\lambda,z}(s,t)|_{\frac{x}{\lambda z_i}, \frac{y}{\lambda z_i}}\} \right|^2 \tag{2}$$

where $\mathcal{F}$ represents the Fourier transform.

1.1 Electric field and plane wave

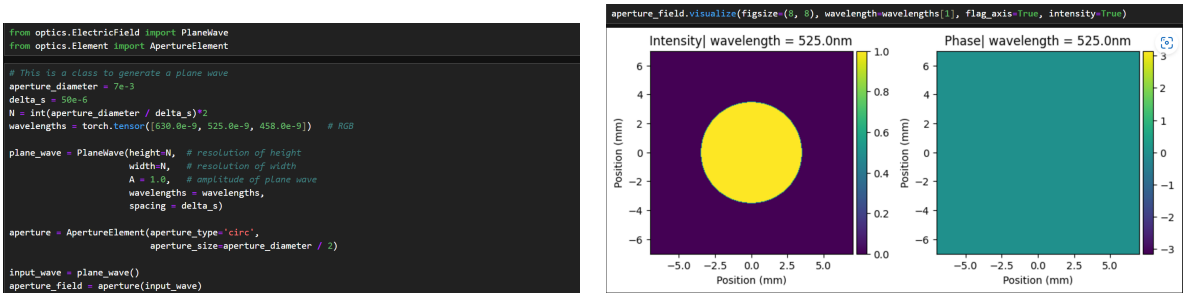
The code for this assignment is largely derived from the codebase I developed during my master's thesis. In this codebase, I first introduced an **Electricfield** data type to store spectral and spatial information, along with a simple visualization method. The **Electricfield** class contains a 4-dimensional tensor, where each dimension corresponds to the field components in the xy plane, the wavelength, and the spatial profile. The built-in methods allow both phase and amplitude distributions to be visualized for each predefined wavelength. Based on the **Electricfield** class, I also defined a **PlaneWave** class that can be modulated by the generalized pupil function. Fig. 1 shows the code to define a plane wave with a 7 mm diameter aperture.

1.2 Generalized Pupil function

To achieve this functionality for this imaging model, I implemented the class named **GeneralizedPupil**. This **GeneralizedPupil** class consists of several methods that together form the imaging forward model. These methods are designed to compute the defocus, apply aberrations, calculate the PSF, and visualize the results.

1.2.1 Constructor Method: __ init __

The constructor initializes the optical system parameters, including the reference focal length, refractive index, minimum and maximum object-to-lens distances, lens-to-sensor distance, and aperture diameter.



(a) (b)

Figure 1: Code example to generate a plane wave with 7 mm aperture and visualization method.

```

class GeneralizedPupil(nn.Module):
    "Generalized Pupil Function, considering the chromatic abberation"
    def __init__(self, ref_foclen, ref_n, zf, zs, d):
```

Listing 1: Constructor Method

1.2.2 calc_defocus Method

This method calculates the defocus parameter ΔD based on the refractive index values provided for different wavelengths and the object distance.

1. **ns**: A list of refractive indices for different wavelengths.
2. **batch_size**: Specifies how many different object distances will be sampled.

```
    def calc_defocus(self, ns, batch_size=1):
        # calculate the focal lengths of corresponding R G B wavelength
        self.foclens = torch.from_numpy(
                self.ref_foclen * (self.ref_n - 1) / (np.array(ns) - 1)
                ).view(1, -1, 1, 1)

        # sample object distances
        self.zfs = torch.from_numpy(
                np.random.uniform(self.zf_min, self.zf_max, batch_size)
                ).view(-1, 1, 1, 1)

        # calculate the defocus parameter
        self.delta_Ds = (1 / self.zfs + 1 / self.zs - 1 / self.foclens).to(self.device
    )
```

Listing 2: A method for computing the defocusing parameters ΔD

In this method, focal lengths for different wavelengths are computed by using the reference focal length and refractive indices.

$$f_\lambda = f_G * \frac{(n_G - 1)}{(n_\lambda - 1)} \tag{3}$$

Then, random distances for the object z are sampled between zf_min and zf_max if a range. Finally, the defocus parameters ΔD are calculated, which measures the discrepancy between the ideal and actual focus due to lens aberrations and chromatic effects.

$$\Delta D = (\frac{1}{z} + \frac{1}{z_i} - \frac{1}{f_\lambda}) \tag{4}$$

1.2.3 to_aberration Method

This method applies phase aberrations due to defocus and chromatic aberration to an electric field representing light traveling through the optical system.

```
    def to_aberration(self, field):

        wavelengths = field.wavelengths[None, :, None, None]
        dx, dy = field.spacing[0], field.spacing[1]
        height, width = field.height, field.width

        X, Y = torch.meshgrid(torch.linspace(-dx * width / 2,
                                             dx * width / 2,
                                             width, dtype=dx.dtype),
                              torch.linspace(-dy * height / 2,
                                             dy * height / 2,
                                             height, dtype=dy.dtype),
                              indexing='xy')
        R = torch.sqrt(X**2 + Y**2).to(self.device)

        # apply phase aberration to the field
        phi_lens = 1j * torch.pi / wavelengths * self.delta_Ds * R[None, None, :,
    :]**2
        aberration = field.data * torch.exp(phi_lens)

        # apply aperture to the field
        aper_lens = R < self.r
        aberration *= aper_lens

        self.aberration = ElectricField(
        data=aberration,
        wavelengths=field.wavelengths,
        spacing = field.spacing
    )

        return self.aberration
```

Listing 3: A method for computing the aberration phase.

After calculating the defocus parameters, the phase aberration can be computed by:

$$Q_{\lambda,z}(s,t) = A(s,t) \exp j \frac{\pi}{\lambda} \Delta D[s^2 + t^2] \tag{5}$$

The phase aberrations are shown in Fig.2

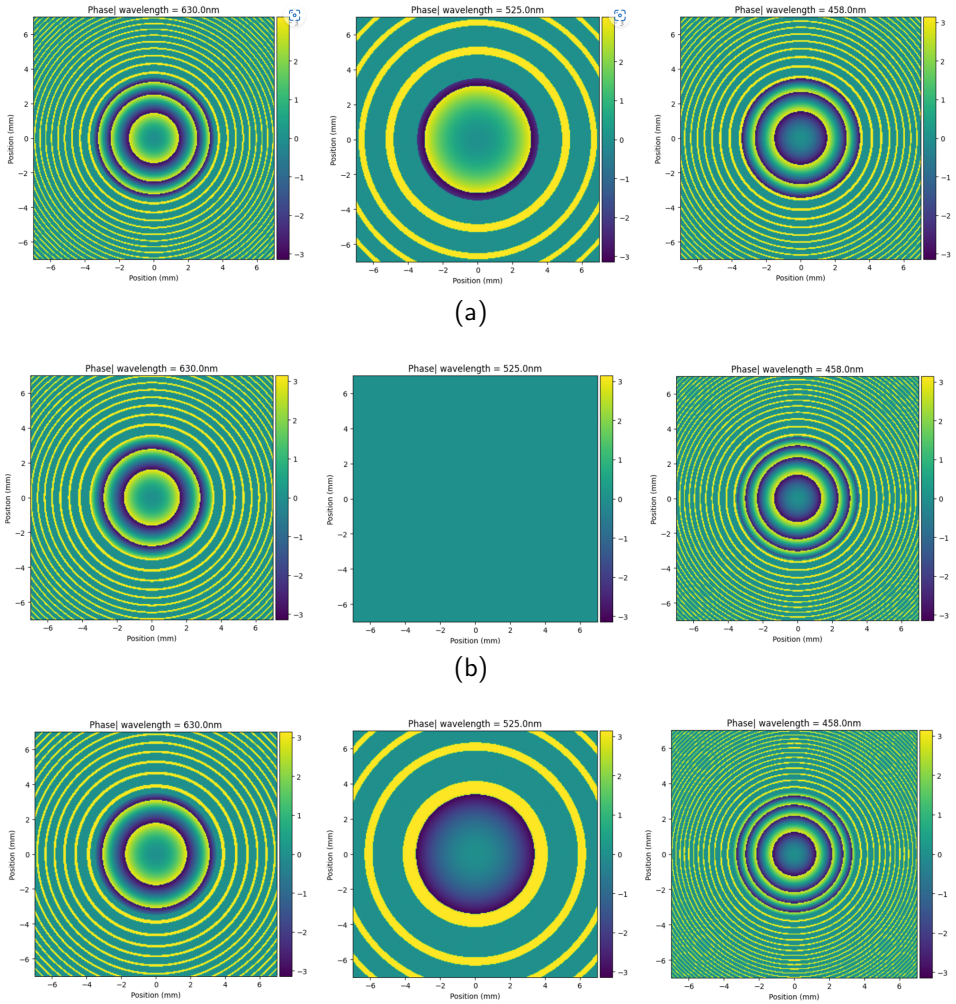


Figure 2: The phase aberrations when object distances are (a) z=1.8m, (b) z=2.0m, and (c) z=2.2m.

1.2.4 to_psf Method

This method computes the response function (PSF) of the aberration aperture based on the far-field approximation.

$$h_{\lambda,z}(x,y) \propto \left| \mathcal{F}\{Q_{\lambda,z}(s,t)|_{\frac{x}{\lambda z_i}, \frac{y}{\lambda z_i}}\} \right|^2 \tag{6}$$

```
    def to_psf(self, field, padding=True):

        if self.aberration is None:
            field = self.to_aberration(field)
        else:
            field = self.aberration
        # padding
        if padding:
```

```
            Horg, Worg = field.height, field.width
            Hpad, Wpad = Horg // 4, Worg // 4
            Himg, Wimg = Horg + 2 * Hpad, Worg + 2 * Wpad
            padded_field = pad(field.data, (Wpad, Wpad, Hpad, Hpad), mode='constant',
    value=0)

        else:
            Himg, Wimg = field.height, field.width

        # Computational fourier optics. Chapter 5, section 5.5.
        # obs sample interval
        self.dx_obs = field.wavelengths * self.zs / Himg / field.spacing[0]
        self.dy_obs = field.wavelengths * self.zs / Himg / field.spacing[1]

        field_data = ifftshift(fft2(fftshift(padded_field)))


        if padding:
            center_crop = torchvision.transforms.CenterCrop([Horg, Worg])
            field_data = center_crop(field_data)

        self.psfs = field_data.abs()**2 / torch.sum(field_data.abs()**2, dim=[2, 3],
    keepdim=True)

        return self.psfs
```

Listing 4: A method to compute the PSF.

Here, we normalize the PSF before the convolution, independently for each color channel, such that,

$$\sum_{x,y} h_{\lambda,z}(x,y) = 1 \tag{7}$$

When using the FFT to compute the Fraunhofer field, the source and observation plane side lengths are not generally the same. The observation plane side length and sample interval in terms of the source plane parameters can be computed by:

$$L_{obs} = \frac{\lambda z}{\Delta x_1}, \quad and \quad \Delta x_2 = \frac{\lambda z}{L_1} \tag{8}$$

we calculated the sampling interval at the observation plane to better visualize the PSF with realistic physical size.

1.2.5 show_psf Method

This method Visualizes the point spread function (PSF) for a specified wavelength. The point spread functions for different wavelengths at various object distances are shown in Fig.3.

1.3 Generating the blurred image

To generate a blurred image dataset, we first define a **MiniImagenetDataset** class to load images from the specific folder. Some examples of the sharp and blurred images are shown in Fig.4.

Then the additive Gaussian noise is added to the blurred images to model the sensor noise:

$$I_{s,z,\lambda}(x,y) = I_{z,\lambda}(x,y) * h_{\lambda,z}(x,y) + N(x,y) \tag{9}$$

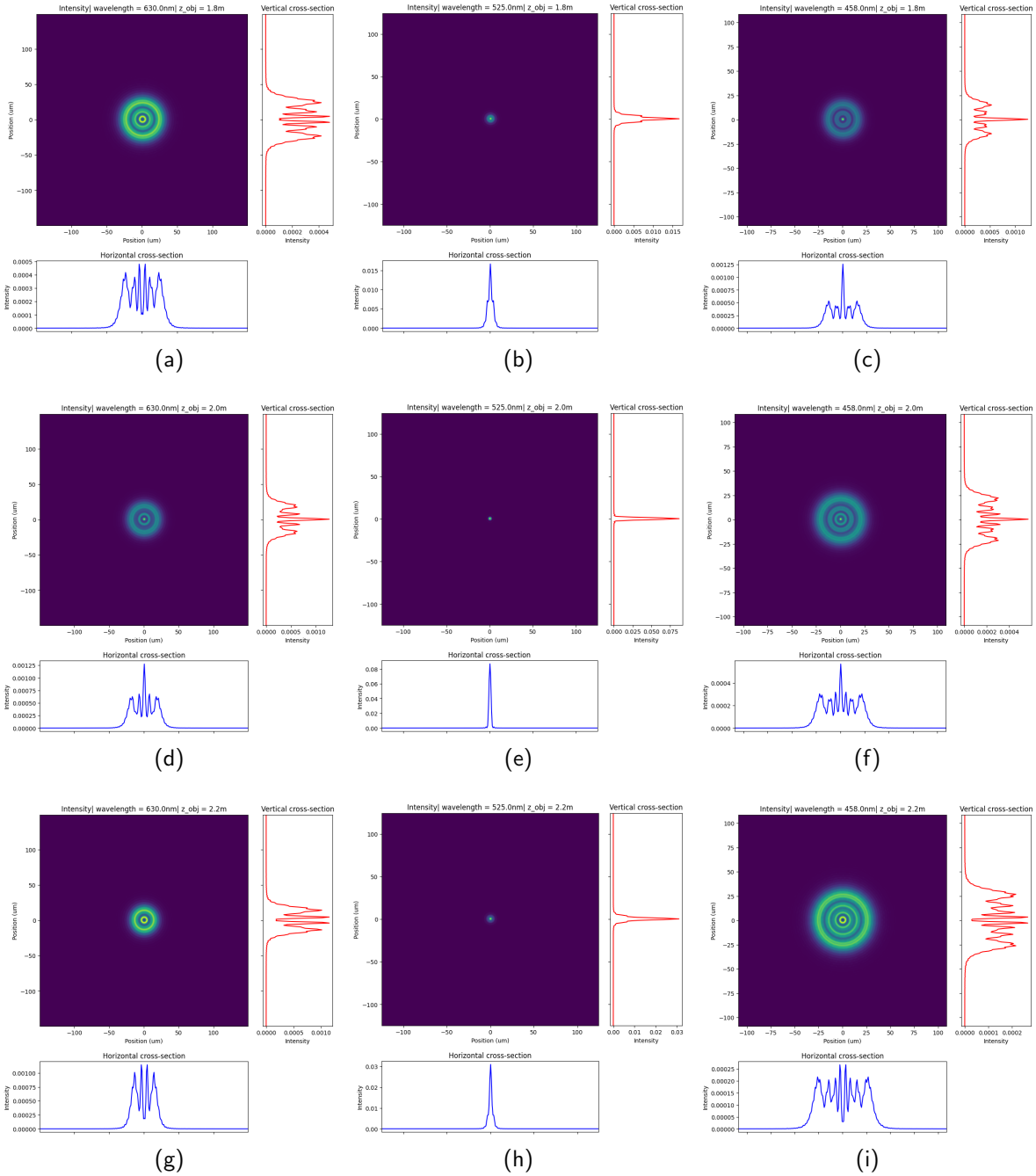


Figure 3: The PSFs when object distances are (a-c) z=1.8m, (d-f) z=2.0m, and (g-f) z=2.2m.

2 Deblurring Neural Network

U-Net [1] is employed for deblurring in this report. The implementation can be found in *./network/unet.py* in the source code. Alternative networks which are implemented in source code, such as NAFNet [2] and Restormer [3], could also be used for this task and as comparisons. However, U-Net

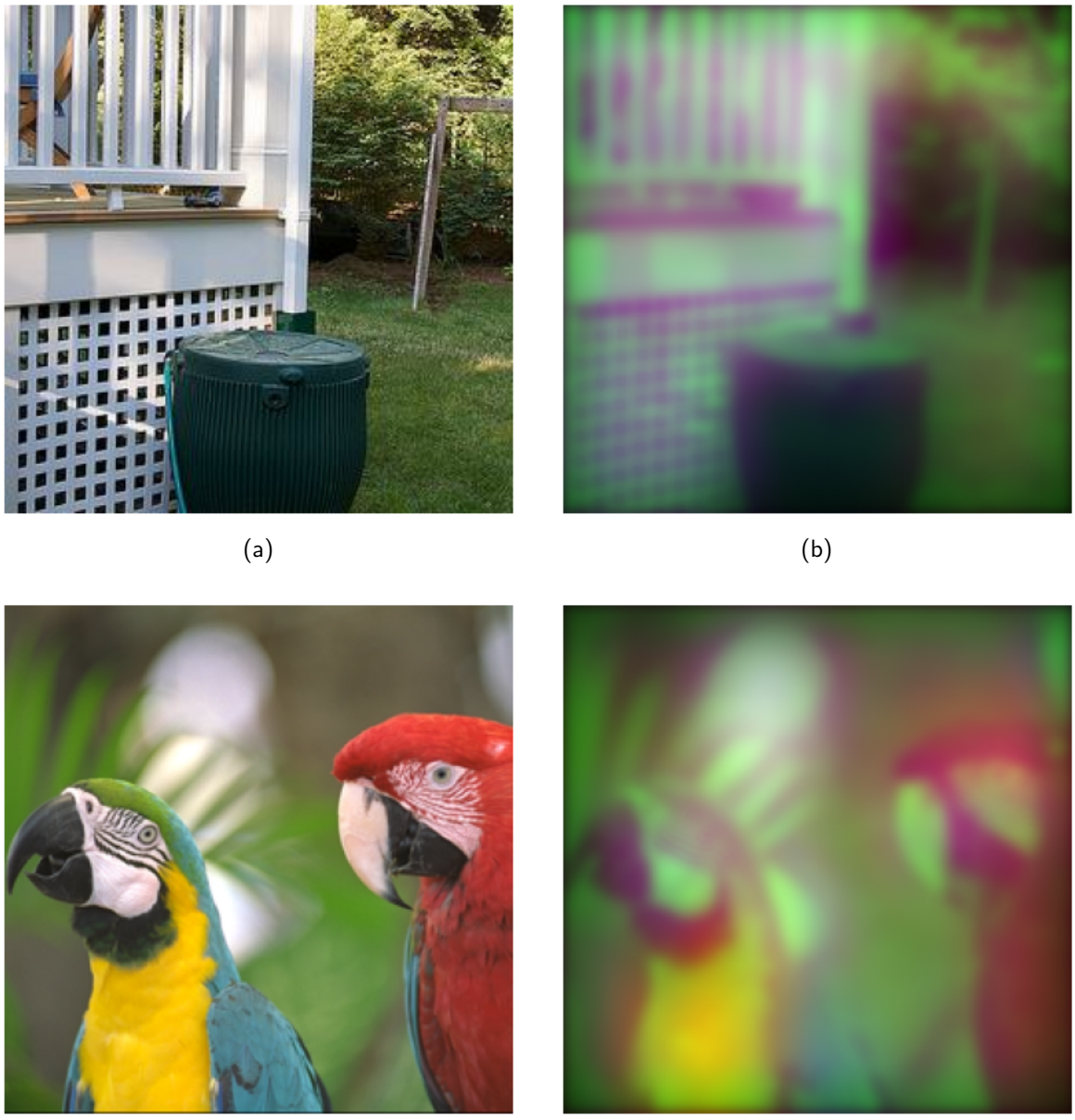
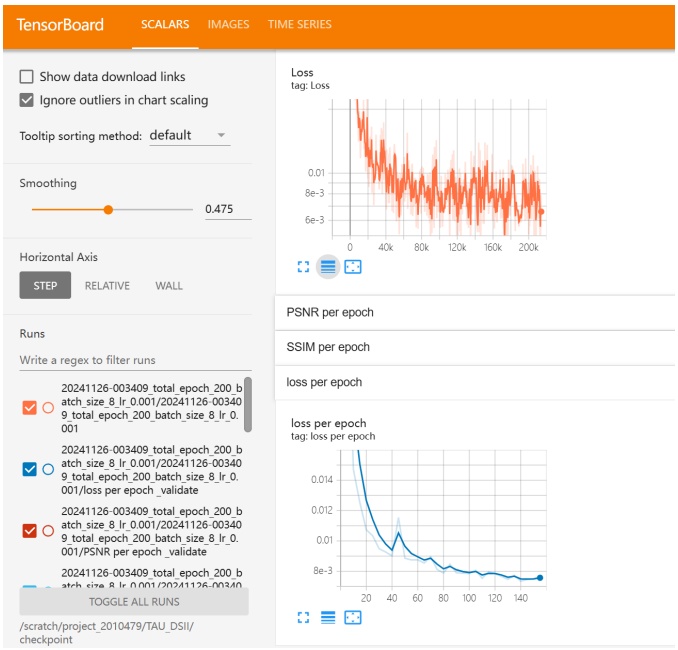


Figure 4: Example of the sharp (left) and blurred images (right)

was chosen as the baseline for the following reasons:

1. U-Net is a lightweight architecture with fewer parameters compared to other networks.
2. For resource-constrained mobile devices, lightweight networks are more suitable for achieving faster inference.

The loss function used for deblurring is the L1 loss, defined as:



(a)

Figure 5: TensorBoard was used to visualize the training process, including the loss values for the training and test datasets, as well as SSIM and PSNR evaluations on the test dataset.

$$\text{loss} = \sum |I_{\text{sharp}} - I_{\text{blurred}}| \tag{10}$$

Other loss functions, such as PSNR, SSIM, and VGG loss, were also implemented in *./utils/losses.py* and can be combined to achieve better results [4]. However, due to time limits, only the L1 loss was used in this work.

We train the whole model for 100 epochs with a batch size of 8 using Adam optimizer. The initial learning rate is 0.001 with decay rates of 0.995 for each epoch. All expierments are built on the CSC Puhti high-performance computing cluster at Finland with 4 NVIDIA Tesla V100 GPUs and implemented by Pytorch. The total training time is 2 days. More training details, such as tensorboard for visualization shown in Fig.5, can be found in *./train_unet.py*.

2.1 Comparison of loss function

In this section, I examine the impact of the different loss functions on the performance of networks. The employed loss functions as L1 loss and L1 loss combined with SSIM loss [4] as:

$$\begin{aligned} \mathcal{L}_1 &= \mathcal{L}^{l_1} \\ \mathcal{L}_2 &= \alpha \mathcal{L}^{l_1} + (1-\alpha)\mathcal{L}^{SSIM} \end{aligned} \tag{11}$$

where $\alpha = 0.3$ in my experiments.

I applied two loss functions to different neural networks, such as U-Net and Restormer, and compared the Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) values on the test dataset. Table 1 presents the impact of different loss functions on the reconstruction results for

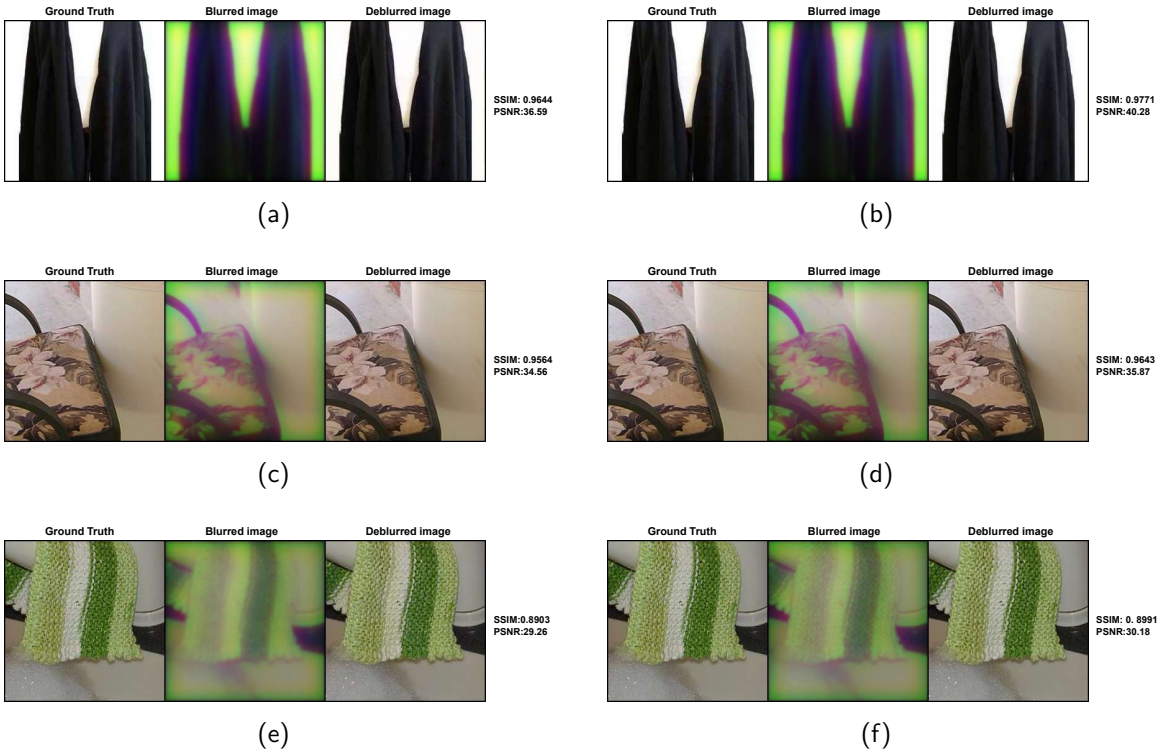


Figure 6: The deblurred results for different networks are presented. The left side displays the results reconstructed by U-Net, while the right side shows those generated by Restormer.

the Mini-Imagenet test dataset. The results indicate that incorporating SSIM into the loss function, represented as $\mathcal{L}_2$, leads to an improvement in the SSIM metric. However, U-Net with $\mathcal{L}_1$ achieves a higher PSNR value compared to U-Net with $\mathcal{L}_2$. In the case of Restormer, incorporating SSIM into the loss function results in improvements in both the PSNR and SSIM metrics.

Table 1: Quantitative comparison of the deblurring performance with different loss functions and architectures.

Architecture	**Params**	**Loss function**	**PSNR**(dB) ↑	**SSIM** ↑
U-net	10.79 M	$\mathcal{L}_1$	29.96	0.8931
U-net	10.79 M	$\mathcal{L}_2$	29.59	0.8948
Restormer	15.20 M	$\mathcal{L}_1$	30.83	0.9076
Restormer	15.20 M	$\mathcal{L}_2$	31.02	0.9112

2.2 Comparison of architecture

Additionally, Table 1 also demonstrates the reconstruction performance for different neural network. Comparing to the U-net, the Restormer achieves around 1dB and 0.01 improvement for PSNR and SSIM, respectively. The visual comparison is presented in Fig.6. Those images for randomly sampled from the test dataset and images from left and right side are results from the U-net and Restormer, respectively.

3 Feature Work

In this report, I first design a single lens imaging system considering the chromatic aberration and object distance, then construct a neural network to deblurre captured images. As all optical components in this codebase are implemented by Pytorch, which support auto-differentiation. In the future work, we can build a end-to-end design pipeline which jointly optimizes the a single DOE and corresponding neural network like some existing work [5, 6, 7].

The basic idea of the future work is listed here:

1. **Point source to sphere wave at the aperture plane**: Based on the paraxial approximation, the object can be modeled as a point source with different distances and PSF of imaging system is shift-invariant.
   (a) The sampling interval Δx_0 and Δy_0 should be carefully considered and keep the propagation simulation accurate. The physics lengths of the field should be twice as the aperture size to ensure correct FFT.
   (b) The sphere wave at the aperture plane is defined as:
   $$u_{in} = \frac{r_{\max}}{r} e^{i \cdot k \cdot r} \quad \text{where} \quad r = \sqrt{(x - x_0) + (y - y_0) + (z - z_0)^2} \tag{12}$$
   where k is the wavenumber for RGB and z_o is the object distance that can be varied during the optimization.
2. **The modulation of DOE at aperture plane.**
   (a) The wave field u_{in} then pass through the camera aperture and the DOE resulting in the changes of the amplitude and phase as:
   $$u_{out} = u_{in} \cdot A(x', y') e^{i \cdot k \cdot (n_\lambda - 1) h(x', y')} \tag{13}$$
   where n_λ is the refractive indexes for RGB wavelengths and $h(x', y')$ is the height profile of DOE needs to be optimized along with the neural network.
   (b) The sampling interval of $h(x', y')$ should be larger than that in wave field due to the fabrication limits. Also, the height map of DOE will be continuous if without any constraint which would result in non-ideal fabrication error.
3. **Propagation to the sensor plane.**
   (a) The modulated field then propagates in free space by a distance z to the sensor plane as:
   $$U_s(x, y, z_1) = \frac{e^{ikz}}{i\lambda z} \iint U_{out} e^{\frac{ik}{2z}((x-x')^2 + (y-y')^2)} dx' dy' \tag{14}$$
   This formulation uses the Fresnel propagation operator, which is an accurate model for near and far distances when $\lambda \ll z$.
   (b) The point spread function of the imaging system can be derived as:
   $$p_\lambda(x, y) \propto |U_s(x, y, z_1)|^2 \tag{15}$$
4. **From PSF to sensor image.**
   (a) Under the assumption of paraxial approximation, the image formation is a shift-invariant convolution of the image and the PSF. Consequently, the off-axis aberrations like coma or chromatic off-axis aberration are neglected.

(b) Further account for the wavelength sensitivity κ_λ of the sensor for each RGB channel:

$$I_c(x,y) = \int (I_\lambda * p_\lambda)(x,y) \cdot \kappa_\lambda d\lambda \tag{16}$$

(c) The image I_c is captured by the sensor with specific sensor pixel size and corrupted by noise. Often, the sensor pixel size is larger than sampling interval of wave field so we need to downsample I_c as:

$$y_c = S(I_c) + \eta \tag{17}$$

where S is the pixel intergration and sampling operator and $\eta \quad \mathcal{N}(0, \sigma^2)$ is Gaussian read noise.

5. **Reconstruction neural network and end-to-end optimization**

(a) We define a neural network G with its parameters Θ, the deblurred image can be processed by NN as:

$$\hat{I}_c = G_\Theta(y_c) \tag{18}$$

(b) During the end-to-end optimization, our final goal is to minimize the loss function to find a optimal DOE height profile $h^\star(x', y')$ and NN parameters $\Theta^\star$ as:

$$h^\star(x', y'), \Theta^\star = \arg\min_{h(x,y),\Theta} \mathcal{L}\big(G_\Theta(y_c), I_c\big) \tag{19}$$

4 Conclusion

In this assignment, I implemented a single-lens imaging system and a corresponding deblurring neural network. First, I developed code to simulate chromatic aberrations and depth-dependent point spread functions (PSFs) derived from Fourier optics. Blurred images were then calculated by convolving the PSFs with sharp images. Corresponding visualization methods were also implemented to better understand the aberration phase and the distribution of the PSFs.

Next, I trained two neural networks, U-Net and Restormer, to perform deblurring. Experiments showed that both networks successfully deblurred images for this imaging system, achieving approximately 29.5 dB PSNR and 0.895 SSIM on the test dataset. Notably, Restormer outperformed U-Net, highlighting the advantages of transformer-based architectures.

Finally, I proposed an approach to further improve deblurring performance by jointly optimizing the imaging system and the corresponding neural network. I believe this end-to-end optimization method can be applied to my future doctoral research projects.

References

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer, 2015, pp. 234–241.

[2] L. Chen, X. Chu, X. Zhang, and J. Sun, "Simple baselines for image restoration," in *European conference on computer vision*. Springer, 2022, pp. 17–33.

[3] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, and M.-H. Yang, "Restormer: Efficient transformer for high-resolution image restoration," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 5728–5739.

[4] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Transactions on computational imaging*, vol. 3, no. 1, pp. 47–57, 2016.

[5] V. Sitzmann, S. Diamond, Y. Peng, X. Dun, S. Boyd, W. Heidrich, F. Heide, and G. Wetzstein, "End-to-end optimization of optics and image processing for achromatic extended depth of field and super-resolution imaging," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–13, 2018.

[6] S.-H. Baek, H. Ikoma, D. S. Jeon, Y. Li, W. Heidrich, G. Wetzstein, and M. H. Kim, "Single-shot hyperspectral-depth imaging with learned diffractive optics," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 2651–2660.

[7] L. Li, L. Wang, W. Song, L. Zhang, Z. Xiong, and H. Huang, "Quantization-aware deep optics for diffractive snapshot hyperspectral imaging," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 19 780–19 789.