Fourier Optics

Santiago R. [redacted], Birge Sükrü Tok [redacted] [1]
[1] *Physics Institute, Humboldt University to Berlin, Germany*
Instructor: Dr. Shuwei Jin
(Protocol Date: May 11, 2021; Experiment Date: 19.April 2021)

Fourier optics involve the study of optical systems by using Fourier transforms to filter and otherwise manipulate images in the spatial domain. Using a spatial light modulator (SLM) to modulate the phase ϕ of polarized light from a monochromatic $\lambda = 532nm$ laser, first the characteristic pixel pitch $p = (15.8 \pm 0.05)\mu m$ of the SLM was determined, as well as the applied gray values for characteristic phase shifts $A_{2\pi} = 176$, $A_{max} = 65$, $A_{min} = 21$. After characterising the SLM, a set of Fourier transformations of spatial images by a lens, together with software masking/filtering and holography were performed and compared to computational models. Finally, a series of orbital angular momentum beams (OAM) were generated and qualitatively described.

I Introduction

I.1 Theory

Spatial light modulators (SLMs) are optical devices capable of modulating the phase ϕ and/or intensity I of light beams, thus finding a large amount of applications in fields such as holography or microscopy, with the former being useful for the study of optical systems using methods such as mask filtering or Fourier transformations by a lens -this is known as Fourier optics.

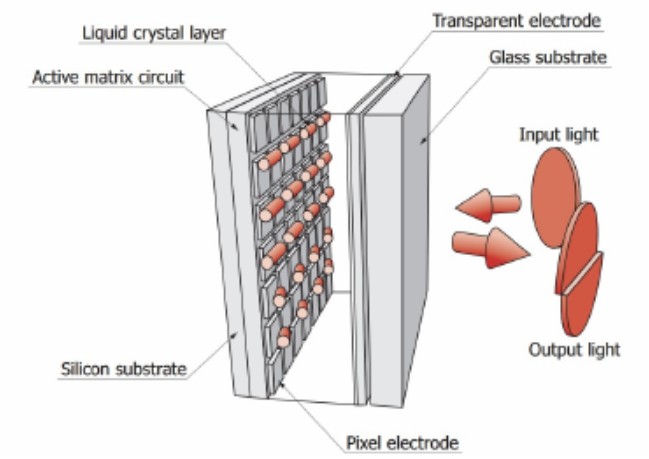


Figure 1: Design of the Hamamatsu X15213 SLM [1]

SLMs can all work very differently from each other and modulate multiple characteristics of light waves other than just their phase, but for these experiments using a liquid-crystal-on-silicon Hamamatsu X15213 SLM, only the phase ϕ of the beams was modulated. The Hamamatsu SLM works herein by having a liquid crystal layer above a pixelated, reflecting silicon substrate with an active matrix circuit and pixel electrodes built in and sitting below another glass substrate with a transparent electrode. The liquid crystals can then be reoriented on a per pixel basis by using the electric fields between the pixels in the silicon and the transparent electrode after applying a voltage between difference them, thus also changing the tilt-dependant extraordinary refractive index $n_e(\theta)$ of each point in the liquid crystal layer. This behavior is described [1] by the equation

$$\frac{1}{n_e(\theta)^2)} = \left(\frac{cos(\theta)}{n_e(0)}\right)^2 + \left(\frac{sin(\theta)}{n_o}\right)^2 \tag{1}$$

with n_o being the constant ordinary refractive index and $\theta \propto \frac{1}{V}$ for any given area in the crystal layer covered by a pixel. From this, a phase difference $\Delta\phi$ of all light incident on an area covered by a pixel of applied voltage V_2 is generated according to

$$\Delta\phi = \frac{2\pi}{\lambda}(n_e(V_1) - n_e(V_2)) \cdot 2d \tag{2}$$

with V_1 being the voltage applied on the upper transparent electrode and d the thickness of the crystal layer. Due to manufacturing constraints, the output wave also acquires a curve-like phase due to the SLM liquid crystal layer and silicon substrate also not being perfectly flat.

I.2 Setup

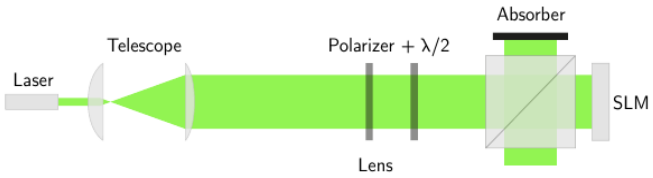


Figure 2: Initial Setup [1]

Although the setup changes multiple times for each part of the experiment, some elements in the optical path remain constant in order for the SLM to operate properly. First, a $\lambda = 532nm$ monochromatic laser which has its beam of light spread out over a larger area by a telescope beam expander is used as a light source for the SLM throughout the whole experiment. From there, the expanded laser beam goes through a polarizer as well as a $\lambda/2$ gate which respectively serve the purpose of polarizing and changing the polarization direction of the light waves arriving on the SLM. After that, a beam splitter redirects part of the light towards an absorber and another towards the SLM where it is modulated. The phase shifted light beams are then redirected once more by the beam splitter towards a target camera/mirror/screen below.

II Characterization of the SLM

For the first set of experiments, the Hamamatsu X15213 SLM itself is characterized by measuring the pixel size and determining the characteristics of its phase modulation in relation to applied grey values.

II.1 Pixel Pitch

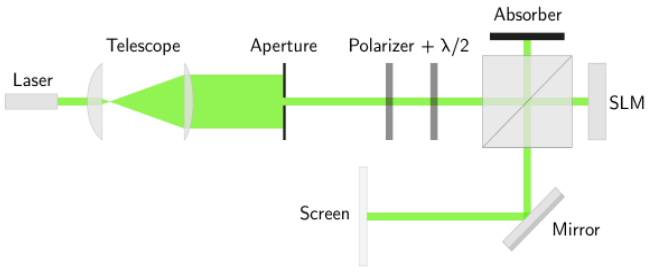


Figure 3: Setup with screen and aperture [1]

The pixel pitch determines the average distance p between pixels within the spatial light modulator, measured from the middle of a pixel to the middle of another. By having the pixelated structure of the SLM act as a 2D diffraction grating - and thus the pixel pitch serve as the spacing $g = p$ - the resulting diffraction pattern of maxima on a screen below the beam splitter satisfies the equation

$$psin(\theta_M) = m\lambda \tag{3}$$

During this experiment, an additional circular aperture is put in front of the expanded laser beam before the polarizer so as to produce a point-like beam that can be utilized for measuring the angular separation between diffraction orders. A screen with millimeter paper in front of it is also placed at a distance $D = (278 \pm 2)mm$ below the beam splitter which is $M = (31 \pm 2)mm$ away from the SLM, so that the angular distance can be inferred through the relation

$$\theta_m = arctan\left(\frac{d}{D+M}\right) \ , \ u_\alpha = \sqrt{\left(\frac{\partial\theta_m}{\partial d}u_d\right)^2 + \left(\frac{\partial\theta_m}{\partial D}u_D\right)^2} \tag{4}$$

Measuring the angular distance θ_m of the vertical and horizontal diffraction orders and performing two linear fits according to equation (3) and a least squares algorithm with $a = \frac{\lambda}{p}$ delivers

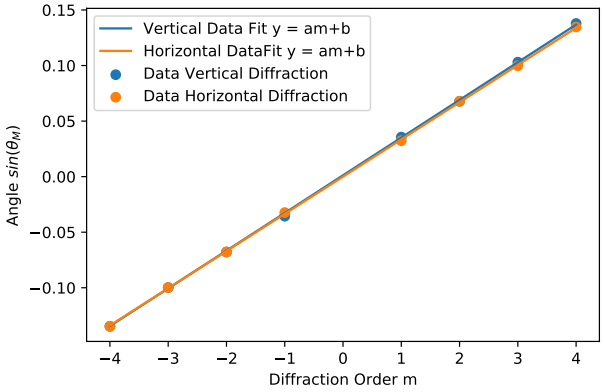


Figure 4: Linear Fit $sin(\theta_m) = am + b$

with the outputted fit parameters

Parameter	Value
a_H	(0.0339 ± 0.0001)
a_V	(0.0335 ± 0.0001)
b_H	(0.0011 ± 0.00035)
b_V	$(-4.024e-9 \pm 0.0003)$

Table 1: Fit-Parameter for $sin(\theta)$

such that the resulting pixel pitch $p = \frac{\lambda}{a}$ after computing the arithmetic mean of both measurements becomes

$$p = (15.8 \pm 0.05)\mu m \tag{5}$$

II.2 Amplitude Modulation

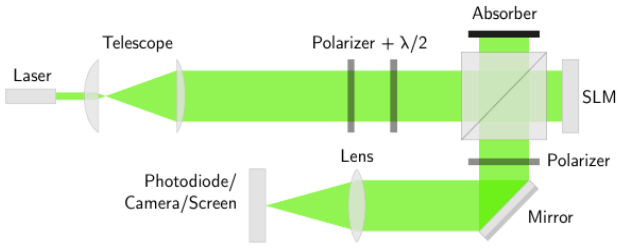


Figure 5: Setup with polarizer and photodiode [1]

For the second part of the SLMs characterization, the phase shift $\Delta\phi$ in relation to the applied gray scale value A is evaluated. For the setup in this occasion, an additional polarizer in crossed configuration towards the first one is placed below the beam splitter. In addition, the light is then redirected by a mirror and focused by a lens onto a photodiode connected to an oscilloscope for measuring the modulation of the incoming photon intensity. This intensity follows the equation [1]

$$I(A) = (I_{max} - I_{min})cos^2(sA + \delta) + I_{min} \tag{6}$$

which can be used to model the voltage-time measurement $U(t) \propto I(A)$ of the oscilloscope as the SLM progressively increases its applied gray values by reevaluating the expression as

$$U(t) = (U_{max} - U_{min})cos^2(wt + \delta_U) + U_{min} \tag{7}$$

Here, the expression holds true that

$$wt \propto sA \Leftrightarrow A \approx \frac{t}{4.14} \cdot 256 - 1 \tag{8}$$

where t is chosen such that an interesting phase shift $\Delta\phi_U = wt \Leftrightarrow \frac{\Delta\phi_U}{w} = t$ in the intensity modulation takes place- in this case for $\Delta\phi_U = [\frac{\pi}{4}, \frac{3\pi}{4}, 2\pi]$, which each correspond to the first minimum, maximum and 2π cycle. Performing a non-linear fit according to equation (7) using a least squares algorithm and assuming an uncertainty for the measured voltages of $u_V = 3\%$ results in

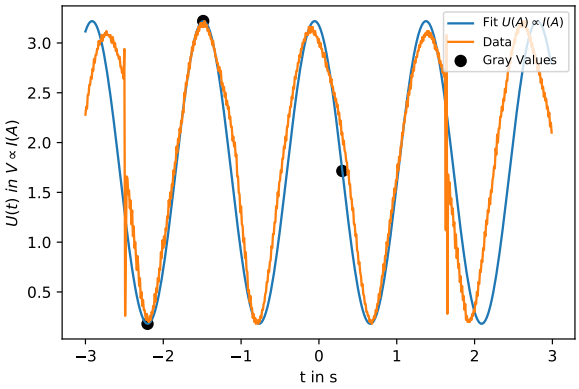


Figure 6: Non Linear Fit Amplitude Modulation

Due to convergence issues with the algorithm, only the w and δ_U parameters were fitted, while $(U_{max} - U_{min})$ and U_{min} were directly computed from the maximum/minimum voltage values within the experimental data. In addition, only the data points between the two cutoff spikes left and right of the data set were used for the fit, since the SLM resets at those points, causing the data to no longer converge with the model before and after them. The computed and fitted parameters are then

Parameter	Value
$(U_{max} - U_{min})$	$(3.04 \pm 0.09)V$
w	(2.1965 ± 0.0055)
δ_U	(0.12 ± 0.007)
U_{min}	$(0.18 \pm 0.005)V$

Table 2: Fit-Parameter for $U(t)$

Thus, the gray values for the characteristic phase shifts computed according to equation (8) with the fitted parameter w as according to $\frac{\Delta\phi_U}{w} = t$ become

$$A_{2\pi} = 176\ , \quad A_{max} = 65\ , \quad A_{min} = 21 \tag{9}$$

III Fourier Optics

III.1 Theory

As known from many fields and its applications, by applying a Fourier transformation, a function or measured signal can be broken down to a superposition of different periodic functions (or modes) and then described by the sum or integral of these individual functions. This is achieved in the case of a continuous 2D function $f(x, y)$ through evaluation of the integral

$$f(x,y) = \int \int_{-\infty}^{\infty} F(\nu_x, \nu_y) exp[i2\pi(\nu_x x + \nu_y y)] d\nu_x d\nu_y \tag{10}$$

where $F(\nu_x, \nu_y)$ is the Fourier transformation of the original function as computed through the integral

$$F(\nu_x, \nu_y) = \int \int_{-\infty}^{\infty} f(x,y) exp[i2\pi(\nu_x x + \nu_y y)] dx dy \tag{11}$$

or in the case of a set of data points through Discrete Fourier Transformation algorithms (DFT, with one of the most used examples being a Fast Fourier Transformation algorithm or FFT). In a 1D data set, this returns a set of sinusoidal waves with given amplitude, frequency and phase. In 2D data, such as images, these phases, which are complex number values, result in an angle with respect to the axes, whereas in 1D data, these phases, in this case real number valued, result in a shift of the waves.

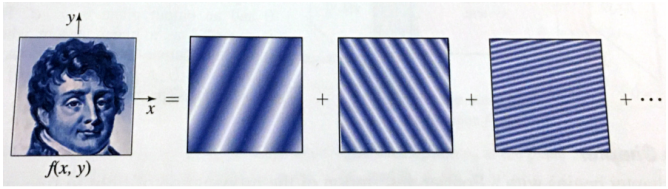


Figure 7: Depiction of a 2D Fourier Transformation.

When making use of the Fourier transformation, it is favorable to transform the data to the frequency domain (from here on, the Fourier transform of an image). This results in a representation of the frequencies and amplitudes of its modes. For a more graphical example, the Fourier transformation of a 2D image represents reoccurring shapes and patterns within the image as lines while sharply defined border and other characteristic appear as high frequency components closer to the center. This is specially noticeable when applying low or high pass filters by masking the high or low frequency components of the Fourier transform before reconverting it into a conventional image through an inverse Fourier transformation F^{-1}.

A common use case for such a transformation is in image (JPEG) and audio (mp3) file compression in computers. In the former, the image is broken down to 8X8 pixel blocks which are Fourier transformed. In the latter, a similar principle is applied, where samples, depending on the so called quality, normally at 44100Hz are taken of which frames consisting of 1152 samples are Fourier Transformed. The resulting Fourier Transform is then stored. Both methods reduce the amount of data from the amplitude data at any given time to a limited amount of modes. However, in both methods continuity is ignored. The blocks/frames are treated in isolation, leading to possible corruption of the data. Along with the fact that the smallest modes are not stored in JPEG, is the reason why high compression ratios cause corruption in image files and why a clicking may be heard in some audio files. [2]

III.2 Fourier Transformations by a Lens

Lenses are able to perform a Fourier transform of an image by focusing the incident plane waves coming in at the angles (θ_x, θ_y) onto a focal plane such that $(\theta_x, \theta_y) \to (x, y) = (\theta_x f, \theta_y f)$, where the complex amplitude of these points (x, y) is proportional to the Fourier transform $F(\nu_x, \nu_y) = F(\frac{x}{\lambda f}, \frac{y}{\lambda f})$ evaluated at the frequencies corresponding to those points.

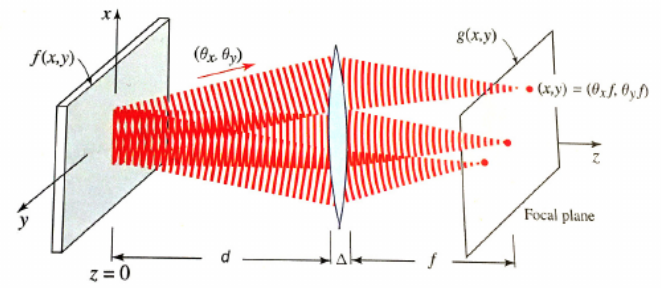


Figure 8: Plane waves being projected onto points (x, y)

For this part of the experiment, the Fourier transformations from various pictures were projected with such an optical setup onto a camera sensor for image capturing and then compared with the numerical computations recreating the same Fourier transforms using a FFT algorithm on the image data.

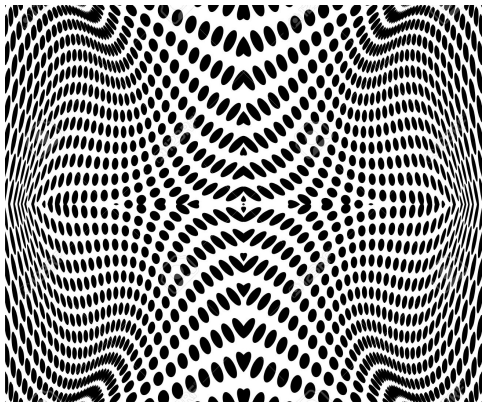


Figure 9: Abstract Black-White pattern

The first such image contains a repeating pattern in black and white with an overarching, curved shape that is captured in the Fourier transforms below due to their periodic nature.

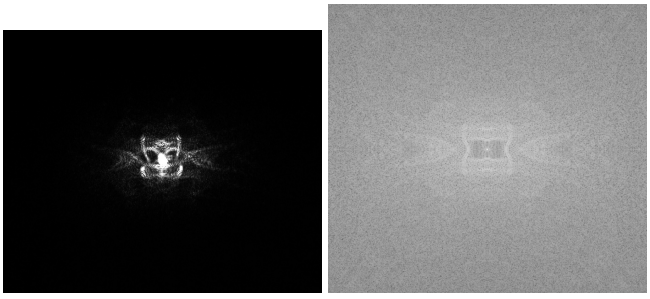


Figure 10: Spatial (left) and numerical fast (right) Fourier transform of Figure 8

Observing the spatial Fourier transform by the lens and comparing it to the numerical prediction by a FFT algorithm, a resemblance in the overall shape of the center from both images can be inferred. The spatial Fourier transform sharply declines in brightness away from the center however, making the shapes towards the edges of the image plane shown by the FFT prediction impossible to make out. Furthermore, the spatial Fourier transform has the intensity maximum in the center further spread out and less defined than the FFT, with details overall being resolved much less sharply due to imperfections in the optical setup projecting the Fourier transformation on the image sensor limiting this when compared to the numerical FFT.

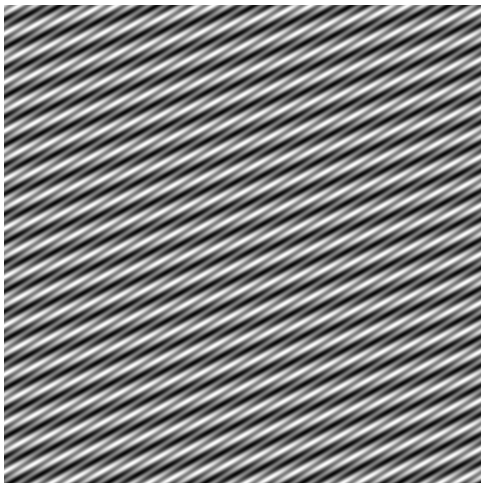


Figure 11: Two Harmonic lines pattern

For the second image, an image of repeating patterns was chosen, where two different sets of parallel lines continuously intersect. For this image, the spatial and numerical Fourier transforms become

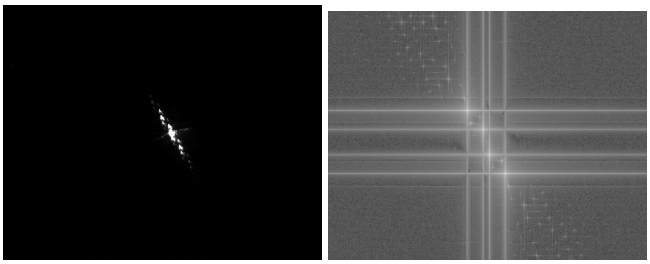


Figure 12: Spatial (left) and numerical fast (right) Fourier transform of Figure 10

where the diagonal outline from upper-left to lower-right is the most recognizable shape the two Fourier transforms have in common. Asides from that, the intensity of the projected pattern sharply declines away from the center again in the spatial Fourier transform like with the first picture, and the strong diffraction-cross like shapes of the computational FFT are completely absent/only the vertical lines in the center maxima can be made out in the left Fourier transform. In addition, a sharp intensity maximum in the center is present in both transformations, with the one in the spatial transform being once again more spread out than the pixel-like maximum predicted by the FFT.

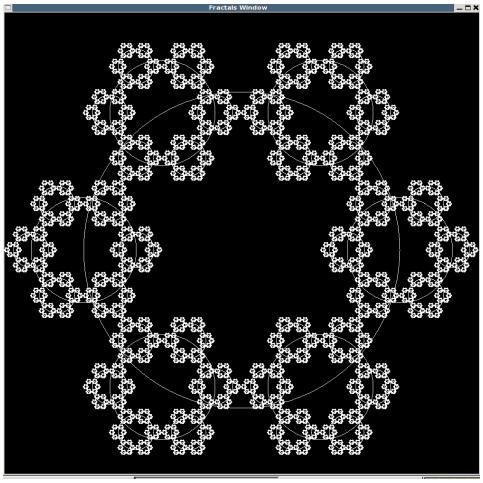


Figure 13: Fractal pattern

The last image involves a fractal repeating pattern which when projected onto their Fourier domain produces another repetitive pattern resembling the original fractal. From the Fourier transformations below

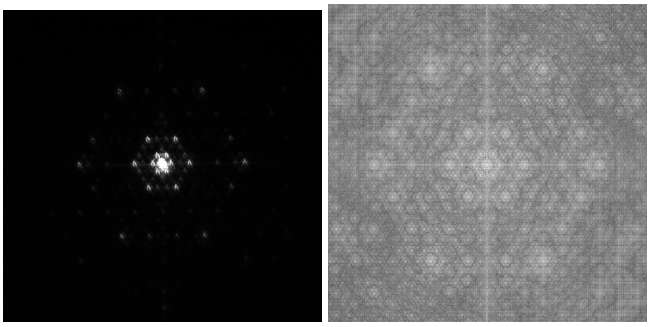


Figure 14: Spatial (left) and numerical fast (right) Fourier transform of Figure 12

it follows again that the spatial transformation resolves much less details of the resulting fractal than its computational peer. Nonetheless, the same resemblance from the upper examples is preserved, with the same rough, hexagonal layout of intensity maximums in the left spatial transform as the smaller fractals in the FFT to the right. The diffraction-like cross starting from the middle in the FFT is completely absent however, and beyond the layout of the maxima, not much of the shape of the smaller fractals is resolved in the spatial Fourier transform.

III.3 Mask filtering in the Fourier Plane

Data may be filtered using a Fourier transform, making use of the fact that the Fourier transform maps the modes of an image on a plane, depending on the frequency, angle (for 2D Data) and magnitude of the mode.

$$x = \lambda \nu_x f \implies \nu x = \frac{x}{\lambda f} y = \lambda \nu_y f \implies \nu y = \frac{y}{\lambda f} \quad (12)$$

This mapping can also be used to calculate the cutoff frequency for a circular filter. With this particular SLM, the cutoff frequency is calculated by transforming the mapping equations of a lens using the fact that all points on a circle have the same distance to the origin.

$$r = \lambda \nu_(c) f \implies \nu_c = \frac{r}{\lambda f} \quad (13)$$

or for a filter of diameter D pixels:

$$\nu_c = \frac{Dp}{2\lambda f} \quad (14)$$

Using this property, limitations may be applied to the data through educated guesses. For example; in a 1D data set consisting of a measurement of regular oscillations with random noise, a threshold may be applied to the magnitude data of the Fourier transform, thus taking only the ”strong” regular oscillations into account. Meanwhile, in astronomy and microscopy, regular imperfections in the image caused by natural phenomena or known regular imperfections in the data capture device (lenses, coatings, sensors) may be filtered out with a high precision, assuming the frequencies of these are known.

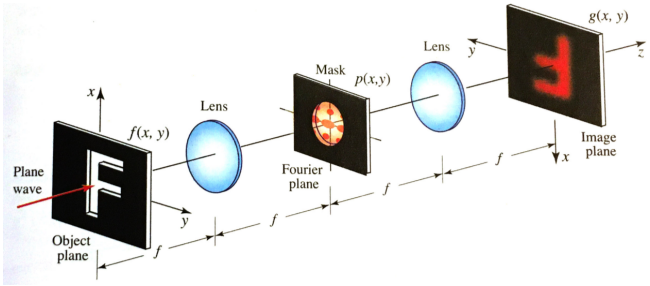


Figure 15: Depiction of a basic 4f system. Every element is 1 f apart from the next.

In this part, this method is made use of to show that lenses can not only Fourier transform images, but also invert this transformation along with filters applied. To achieve this, a so called ”4f system” is constructed, where the SLM is used to apply a circular high- or low pass filter (HPF or LPF), or a linear slit- or bar filter.

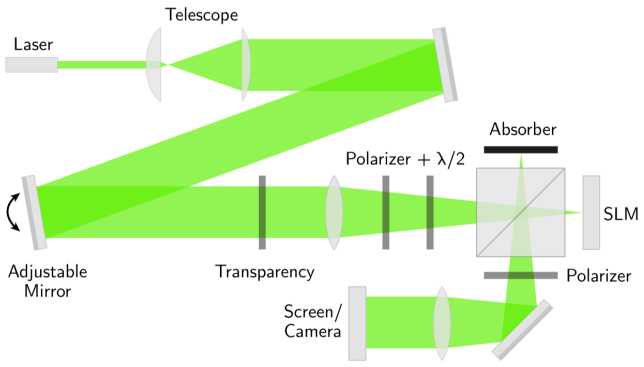


Figure 16: Depiction of the 4f system utilized. The SLM is used to apply filters onto the image from the transparency. The thus produced image is then captured using a camera.

The resultant images are then compared to computer generated filtered images. The generation of

such is achieved by Fourier transforming the original image and multiplying the intensity values of the resultant image by a matrix, where the indices corresponding to the values to be filtered out are equal to zero. Since only the frequency and angle filtering are being observed, this multiplication suffices to apply the filter to the Fourier transformed image. Once the inverse Fourier transform is applied to this filtered Fourier transform, the filtered Image is yielded.

For this experiment, 3 Images were utilized

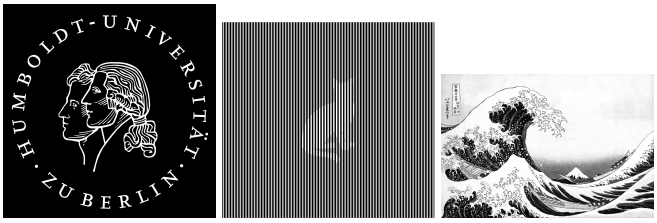


Figure 17: Images Used

The logo of the Humboldt university, as this has sharp edges, and may be beneficial in visualizing edge detection; a silhouette of a cat, obscured by a grating, which shows the linear filtering of regular imperfections; and finally a picture of The Great Wave of Kanagawa, a prime example of the Ukio-e movement in Japan from the 1830s depicting a large wave -likely a rouge wave, mistakenly assumed to be a tsunami- with Mount Fuji in the background located in the trough of the wave. The latter was chosen as an example of a more complex image. For each 3 measurements were taken using visually interesting filter sizes.

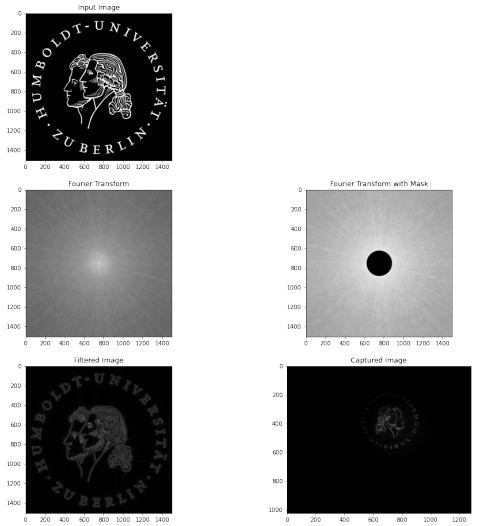


Figure 18: Depiction of the Humboldt University Logo filtering process with a HPF of width 130px.

Looking at the Logo of the university only from a single filter type, the effect may not be discerned easily from the captured image as it is rather dim. However, putting images of a HPF next to that of a LPF along with the numerically generated filtered images, the difference may be observed more easily.

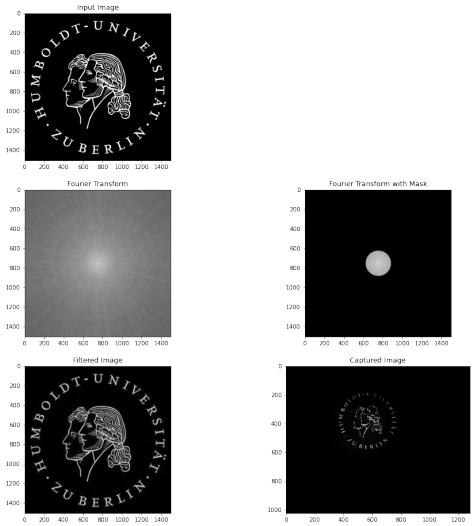


Figure 19: Depiction of the Humboldt University Logo filtering process with a LPF of width 130px.

Looking at the numerically generated filtered images, the HPF removes the infill, whereas the LPF is slightly blurring the edges. This implies that the infill was generated by the lower frequency components, whereas the details or edges -the steep gradient parts- were generated by higher frequency components. From this we can gather that just as in a Taylor series, the higher order -in this case frequency- the better the image detail. Judging by the brightness and thickness of the lettering, one can assume that indeed only the borders are left in the HPF. Whereas the LPF obscured the edges, which is especially apparent looking at the faces depicted. The higher diameters led to a moderation of the effect on a LPF and to a larger effect on the HPF; meaning thinner edges on the HPF displayed by a dimmer image with thinner looking letters, and more detail in the faces of the LPF captured image.

In the case of the cat silhouette, the difference seems nonexistent if the view is limited to the silhouette and comparing vertical LPF and HPF at the same width. However, with a narrow vertical LPF, the grating almost disappears.

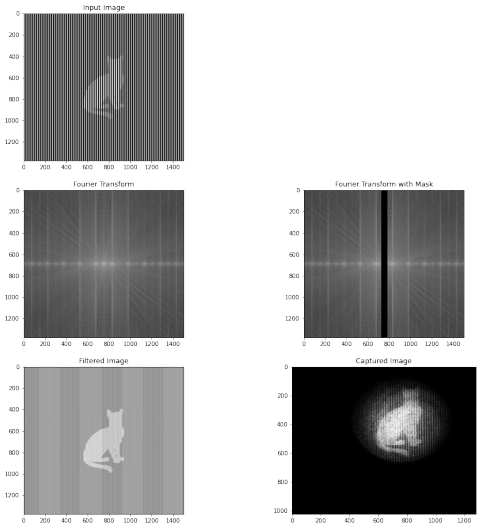


Figure 20: Depiction of the cat silhouette image filtering process with a barfilter of width 56px.

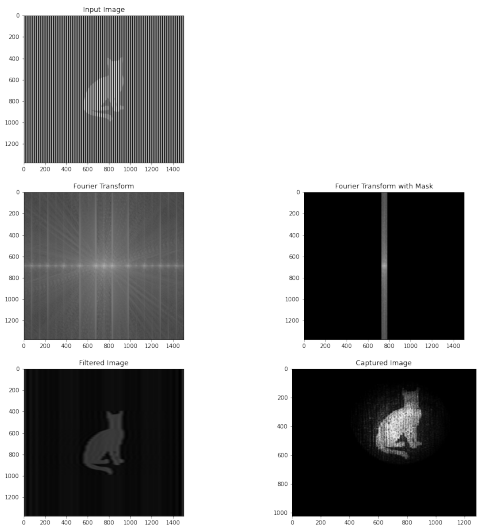


Figure 21: Depiction of the Humboldt University Logo filtering process with a slit filter of width 56px.

This implies that the detailed grating is generated by lower frequencies. With a narrow bar filter, the silhouette which was broken up by the grating looks like a complete silhouette. The slit filtered image contrasts this with the clearly visible black bands, breaking up the silhouette. A possible explanation for this would be, that a bar filter causes a 1D blurring of an image, similar to the blurring and detail loss we observed in the LPF above. With both filters however, the rough grain texture of the grating could no longer be observed-the wider bands of dark and bright bars of the grating about half the size of the cat, looking like a freshly mown football field. Looking at the computer generated Fourier transform of this image, it would have been interesting to view the image with horizontal slit- and bar filters.

Finally, when observing the same process on a more complex image -The Great Wave of Kanagawa-, similar effects as described above can be observed. The edge filtering aspect of the HPF however, is inferred just as with the Humboldt University logo, whereas the blur effect of the LPF was clearly observable.

III.4 Holography

Holography is the method of recording a wave front such that it may be reproduced later. It is well know for the application of recording and reproducing 3D images. Here the SLM as is used as a hologram source, which then in turn projects the image from which it was generated at the focal point of a lens. The hologram is generated by means of an error reduction algorithm, specifically the Gerchberg-Saxton algorithm, which for repeated iterations delivers higher accuracy holograms via phase masks numerically generated from a source image.

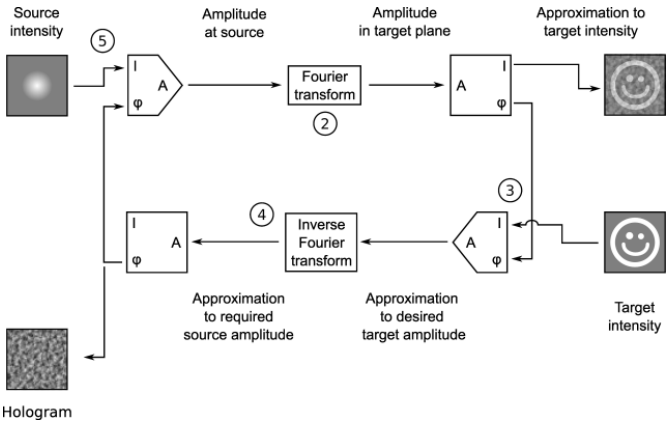


Figure 22: Schematic of an error reduction algorithm like the Gerchberg-Saxton.

Here a random phase matrix is used as the initial hologram data alongside the Fourier transform of the source image. The transferring function (A) reads

$$A = I * e^{i\phi}$$

and can be implemented through Python code as a function with inbuilt for-loop as follows

```
def GXalg(image, iterations):
    phase_mask_source =
    ↪  np.random.rand(*img.shape)
    phase_mask = np.ones(img.shape)
    amplitude_img = img
    amplitude_func = np.ones(img.shape)

    input_signal =
    ↪  amplitude_img*np.exp(phase_mask_
    ↪  source * 1j)

```

```
    for i in range(iterations):
            print("iteration ", i+1,
            ↪  "of", iterations)
            signal, amplitude_signal =
            ↪  dft(input_signal)

            phase_mask =
            ↪  get_phase(signal)
            signal_init = amplitude_func
            ↪  * np.exp(phase_mask * 1j)
            signal[:,:,0] =
            ↪  np.real(signal_init)
            signal[:,:,1] =
            ↪  np.imag(signal_init)
            input_signal, amplitude =
            ↪  idft(signal)
            phase_mask_source =
            ↪  get_phase(input_signal)
            input_signal = amplitude_img
            ↪  *
            ↪  np.exp(phase_mask_source
            ↪  * 1j)

    clear_output()
    return phase_mask
```

The full dependencies alongside the output from the code can be found attached below as a Jupyter Notebook. For one iteration, the algorithm returns the following phase mask and reconstructed image

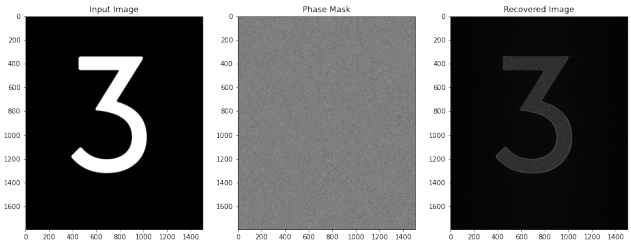


Figure 23: Generated Hologram for 1 Iteration

where even for a single iteration, one already gets an image similar to the original when reconstructing the outputted phase mask. There is a lack of contrast in the mapped intensity which can be seen in the darker color of what would have been white in the original image, as well as a grain-like gray pattern in its stead.

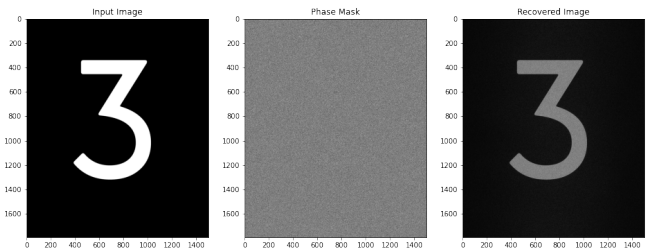


Figure 24: Generated Hologram for 2 Iterations

Further applying a second iteration, this lack of contrast is noticeably improved, with the white of the original image now being displayed as a much clearer gray color in the reconstruction from the phase mask. The grain-like pattern is still present nonetheless.

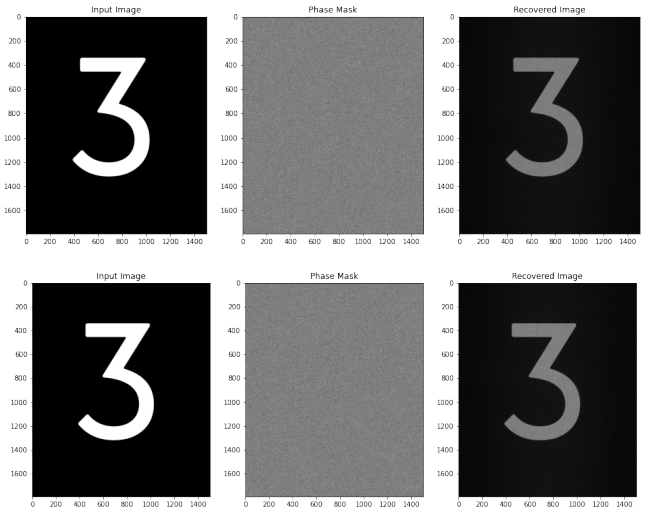


Figure 25: Generated Hologram for 5 (top) and 20 (below) Iterations

Iterating even further between five and twenty times, the contrast does not noticeably improve any more however. This is due to the convergent nature of the algorithm, which noticeably decreases the error in the first two or three iterations and then doesn't decrease it much further even after dozens more. In order to compare and make an estimate of the error σ and its reduction as a function of the amount of iterations N, the difference between the matrix norms of the original image M_S and the reconstructed image $M_T(N)$ out of the phase mask may be compared such that

$$\sigma(N) = abs(||M_S|| - ||M_T(N)||) \tag{15}$$

Plotting these differences then delivers

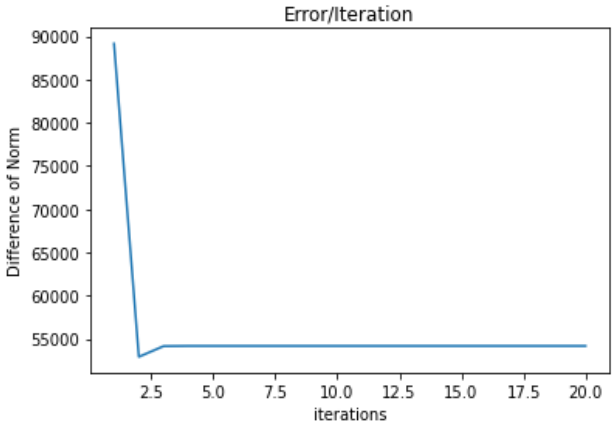


Figure 26: Estimated error $\sigma(N)$

which is consistent with the observed effects on the images above, where the computation of a second iteration strongly reduces the error already, and where extra iterations seem to stagnate in reducing it further.

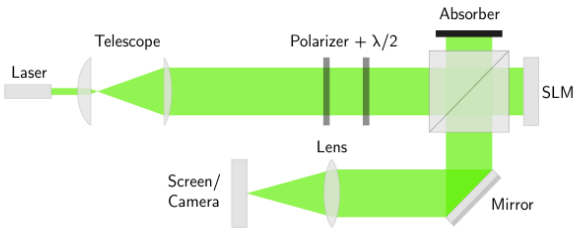


Figure 27: Setup for holography with the SLM [**?**]

In order to compare the above prediction from the Python algorithm with actual holograms produced using the SLM, the setup was modified as in Figure 27 and the SLM used as a pure phase modulator projecting the light with the help of a lens onto a camera sensor. Using the SLM ToolBox program with a Correction Bitmap for compensating the uneven shape of the SLMs surface and an inbuilt phase mask generator based on the same Gerchberg-Saxton algorithm, the following hologram was then displayed and captured

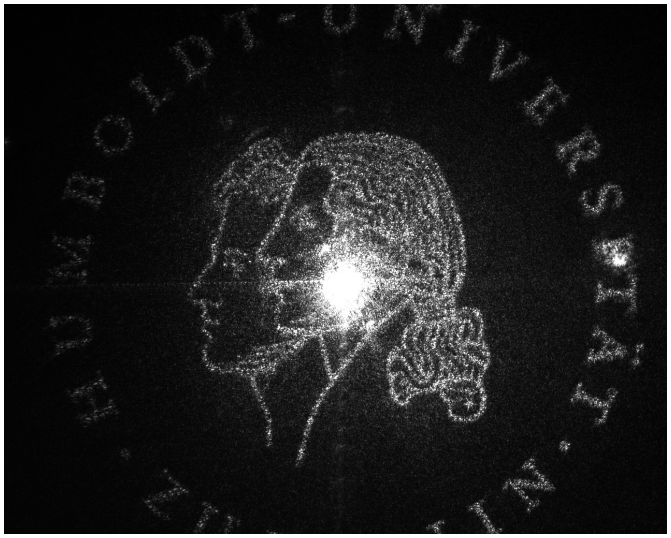


Figure 28: Hologram of the HU logo, 1 iteration

The image generated by the SLM from a phase mask computed with only one iteration is observed to have much the same graininess as the ones computed above, both due to the computational model, but also now due to the limits imposed on the overall resolution by the pixelated structure of the SLM and other optical imperfections within the setup. Likewise due to the later reason, an intensity maximum absent in the computational predictions is observed in the middle of the image due to diffraction effects in the optical setup.

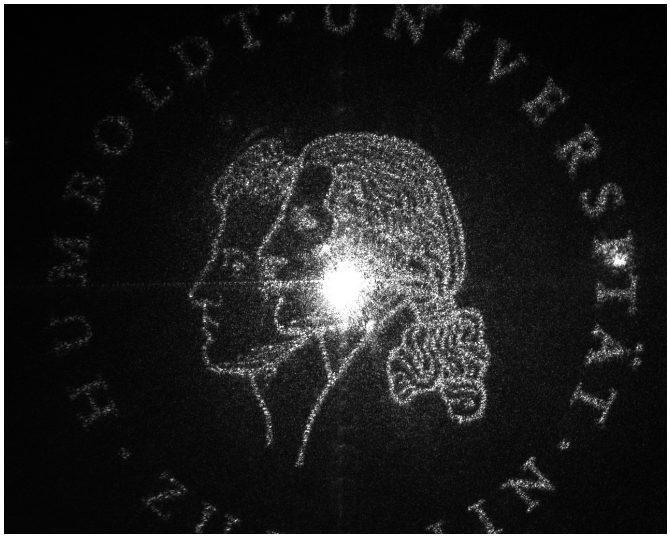


Figure 29: Hologram of the HU logo, 10 iterations

Iterating the phase masks further for a total of ten times, the projected hologram improves slightly in quality. Specially for the resolved details, finer features which were completely absent in Figure 28 -like eyebrows, face- or hairlines- can now be somewhat made out. The intensity maximum from diffraction at the center of the image is not affected by this, however.

III.5 Orbital Angular Momentum Beams

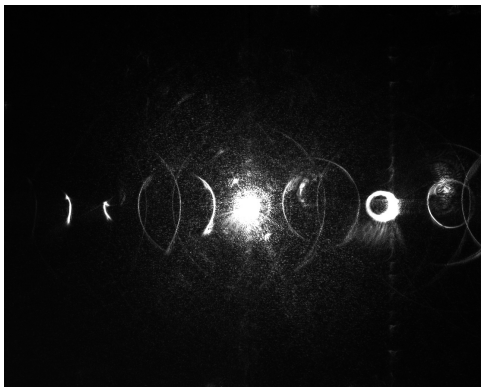


Figure 30: Axicon pattern with doghnut shaped intensity profile beam to the right

Orbital angular momentum beams (OAM) are a form of higher order spatial light modes known as Laguerre Gaussian beam, where only the azimuthal component has been modified by the SLM. Such beams have a vortex-shaped phase structure and their projected intensity profile appears to have a doughnut-like shape, as seen in one of the profiles to the right of Figure 14. These OAM beams can be generated with the SLM-Toolbox program using either a vortex-shaped profile or as in this case an axicon profile.

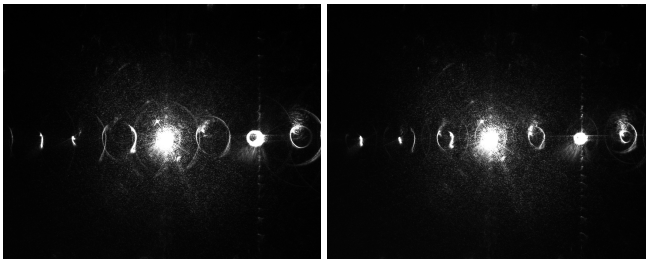


Figure 31: Intensity profile of an OAM beam with different azimuthal orders

Changing the azimuthal order AZ by modifying the topological charge of beam -which signifies the amount of azimuthal transitions in steps of 2π- changes the form of the intensity profile by either enlarging the circumference of the circular doughnut with increasing AZ or making it smaller until the hole in the middle of the "doughnut" is closed completely.
OAM modes are of interest for applications like microscopy below the diffraction limit of light or high bandwidths communications, and their feasibility in fiber-based links is currently under research for high speed communication purposes.

IV Discussion

The experimental findings support most of the theory involving the SLM characterization and the theory on spatial Fourier transformations, masking and holography by behaving as predicted by the mathematical models and computational predictions.

Starting with the characterization of the SLM, the computed value for the Pixel Pitch of $p = (15.8 \pm 0.05)\mu m$ is in line with the reference value $p_{ref} = 12.5\mu m$ [3] from the literature, with the deviation outside the uncertainty boundary likely corresponding to a measurement error for the distance between the beam splitter and the SLM, which was difficult to measure during the experiment due to obstructions. Furthermore, the computed gray scale values for the characteristic phase shifts $A_{2\pi} = 176$, $A_{max} = 65$, $A_{min} = 21$ also agree with the predicted values during the experiment, where a first computation of these from time readouts in the oscilloscope delivered initial estimates of $A_{2\pi.est} = 174$, $A_{max.est} = 61$, $A_{min.est} = 18$.

Proceeding towards the imaging of the spatial Fourier transformation, the numerical FFTs showed a good degree of agreement with the overall outline of the Fourier transformations by the lens optical system, with an overall loss in detail and intensity maxima further spread out and compromising surrounding features in the spatial transformation. These can be attributed to imperfections in the optical setup however and were to be expected, owing also to the limited pixel resolution of the SLM.

As for the mask filtering in the Fourier plane, it was observed that the lenses do indeed behave like a Fourier and inverse Fourier transformer if set up appropriately. Some of the effects were inferred however. In order to observe the full effects, a higher resolution camera, or a larger image may be used -that is assuming the lenses can provide such resolutions.
What is interesting however, is to observe the 1D blurring effect hypothesized above. With a more complex image this effect becomes more apparent.

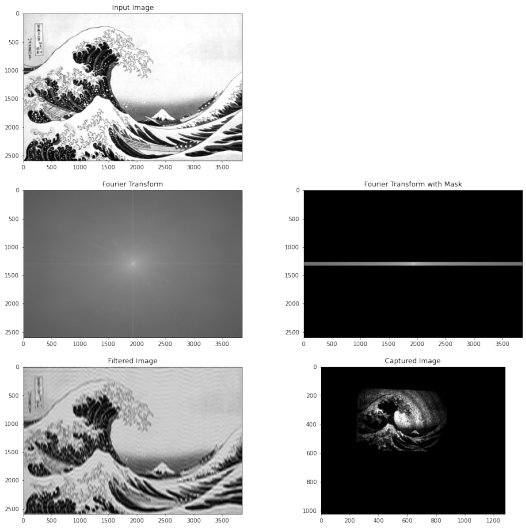


Figure 32: Depiction of the The Great Wave of Kanagawa filtering process with a horizontal slit filter of width 65px.

When looking at the computer generated image, the vertical blur effect is visible. The same is not entirely valid for the captured image. A blur effect is visible, but the detail captured is so little, that a discrimination between horizontal or vertical cannot be made. This does hint to the hypothesis being correct. However, to say for certain a better quality capture image is needed.

Regarding the results from the holography segment, the Gerchberg-Saxon algorithm behaves as expected in finding a suitable phase mask that when inverse Fourier transformed delivers a reconstruction of the original image such that the phase mask can be used for holography in combination with the SLM. It converges quickly too and doesn't deliver many improvements to the error reduction for iterations beyond two or three. Finally, although the algorithm didn't work at first, further improvements to the original code via comparisons with [4] brought it to a usable state where the returned phase masks could be reconstructed again into a similar image to the source. The introduced graininess also matches the one found in the captured holography images using the SLM, with only characteristic optical aberrations from the SLM and laser setup deviating significantly from the predicted computational reconstructions of the phase mask.

Lastly, the generated OAM beams also responded as expected to increases and decreases in their azimuthal order, with the only difficulty during the experiment arising with acquiring a proper doughnut-shaped intensity profile using the vortex-shaped profile within the SLM-Toolbox software, thus leading to the need of generating this through the alternative axicon profile. Nonetheless and despite the difficulties and uncertainties described for the experiment above, the results shown further support the predicted behavior for the SLMs modulation of polarized light beams and the resulting applications in the field.

V References

[1] Daniel Lechner, Lucas Pache, Shuwei Jin : *Fourier Optics Spatial Light Modulator Lab Course*, 2007

[2] John Peacock, University of Edinburgh: *Compression of data using Fourier methods*, https://www.roe.ac.uk/japwww/teaching/fourier/compression.html, last accessed: May 11, 2021

[3] Hamamatsu: *LCOS-SLM (Liquid Crystal on Silicon - Spatial Light Modulator*, https://www.hamamatsu.com/resources/pdf/lsr/x15213_E.pdf, last accessed: May 11, 2021

[4] wxwang0104: *Phase retrieval single constraint*, https://github.com/wxwang0104/Phase_retrieval_single_constraint, last accessed: May 11, 2021

Gerchberg-Saxton Algorithm

May 11, 2021

[1]:
```
import numpy as np
%matplotlib inline
from matplotlib import pyplot as plt
from IPython.display import clear_output
import cv2
```

[2]:
```
def dft(data,shift=True,mag=True,cmplx=True):
    if cmplx==True:
        dft = cv2.dft(np.float32(data), flags = cv2.DFT_COMPLEX_OUTPUT)
    else:
        dft = cv2.dft(np.float32(data))

    if shift == True:
        dft_shift = np.fft.fftshift(dft)
    else:
        dft_shift=dft
    if mag == True:
        magnitude_spectrum = 20*np.log(cv2.magnitude(dft_shift[:,:
 ↪,0],dft_shift[:,:,1]))
        return dft_shift, magnitude_spectrum
    else:
        return dft_shift

def idft(data,shift=True,mag=True):
    if shift == True:
        idft_shift = np.fft.ifftshift(np.float32(data))
    else:
        idft_shift=np.float32(data)

    idft = cv2.idft(idft_shift)

    if mag == True:
        magnitude_spectrum = cv2.magnitude(idft[:,:,0],idft[:,:,1])
        return idft, magnitude_spectrum
    else:
        return idft
```

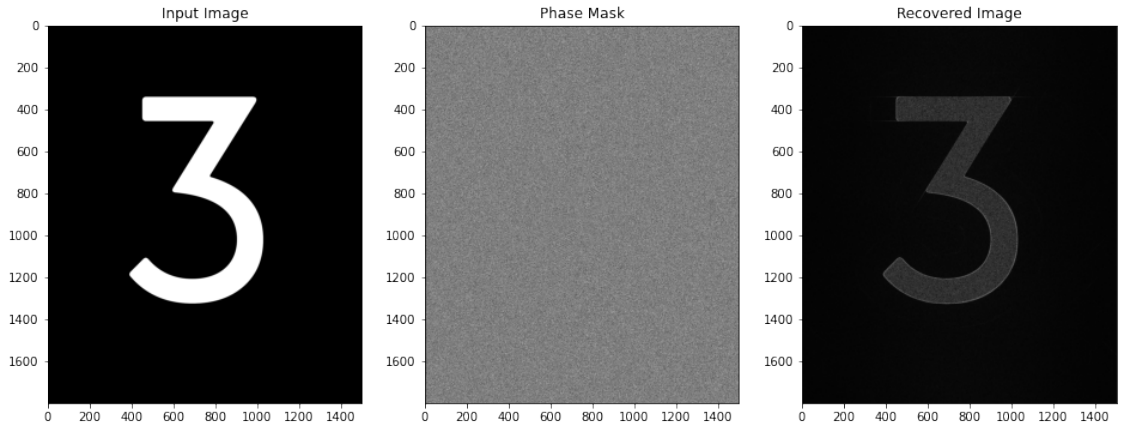
[1]:
```
def get_amplitude(data):
    return cv2.magnitude(data[:,:,0],data[:,:,1])

def get_phase(data):
    mag,ang = cv2.cartToPolar(data[:,:,0],data[:,:,1])
    return ang

def norm_compare(img,phase_mask):
    signal_plt = np.ones(phase_mask.shape+(2,))
    signal_plt[:,:,0] = np.real(np.exp(phase_mask*1j))
    signal_plt[:,:,1] = np.imag(np.exp(phase_mask*1j))
    recovered_img,recovered_img_mag = idft(signal_plt)
    factor = np.max(img)/np.max(recovered_img_mag)
    return (np.absolute(np.linalg.norm(img) - np.linalg.
 ↪norm(factor*recovered_img_mag)))

def GXalg(image, iterations, err_reduction_plot=False):
    phase_mask_source = np.random.rand(*img.shape)
    phase_mask = np.ones(img.shape)
    amplitude_img = img
    amplitude_func = np.ones(img.shape)

    input_signal = amplitude_img*np.exp(phase_mask_source * 1j)

    if err_reduction_plot==True:
        variance = np.ones(iterations)
        for i in range(iterations):
            print("iteration ", i+1, "of", iterations)
            signal, amplitude_signal = dft(input_signal)

            phase_mask = get_phase(signal)
            signal_init = amplitude_func * np.exp(phase_mask * 1j)
            signal[:,:,0] = np.real(signal_init)
            signal[:,:,1] = np.imag(signal_init)
            input_signal, amplitude = idft(signal)
            phase_mask_source = get_phase(input_signal)
            input_signal = amplitude_img * np.exp(phase_mask_source * 1j)

            variance[i] = norm_compare(image,phase_mask)

        x = np.arange(1,iterations+1,1)
        plt.plot(x,variance)
        plt.title("Error/Iteration")
        plt.xlabel("iterations")
        plt.ylabel("Difference of Norm")

    else:
```

```
        for i in range(iterations):
            print("iteration ", i+1, "of", iterations)
            signal, amplitude_signal = dft(input_signal)

            phase_mask = get_phase(signal)
            signal_init = amplitude_func * np.exp(phase_mask * 1j)
            signal[:,:,0] = np.real(signal_init)
            signal[:,:,1] = np.imag(signal_init)
            input_signal, amplitude = idft(signal)
            phase_mask_source = get_phase(input_signal)
            input_signal = amplitude_img * np.exp(phase_mask_source * 1j)

    clear_output()
    return phase_mask
```

0.0.1 1 Iteration

[4]:
```
img = cv2.imread(r"C:\Users\birge\Documents\Fourrier Optics\IMG\3.bmp", 0)
phase_mask= GXalg(img,1)
signal = np.ones(phase_mask.shape+(2,))
signal[:,:,0] = np.real(np.exp(phase_mask*1j))
signal[:,:,1] = np.imag(np.exp(phase_mask*1j))
recovered_img,recovered_img_mag = idft(signal)
```

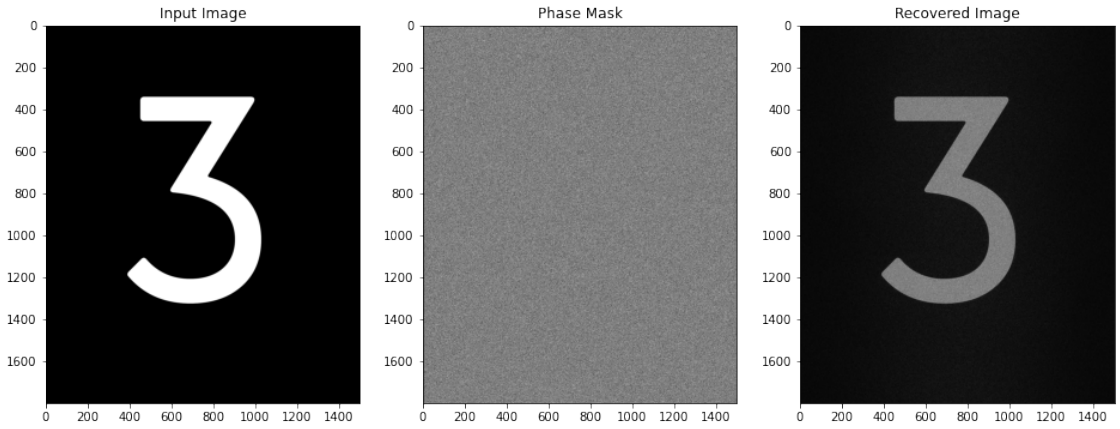
[5]:
```
fig = plt.figure(figsize=(16,16))
ax1 = fig.add_subplot(2,3,1)
ax1.imshow(img,cmap="gray")
ax1.title.set_text("Input Image")
ax2 = fig.add_subplot(2,3,2)
ax2.imshow(phase_mask,cmap="gray")
ax2.title.set_text("Phase Mask")
ax3 = fig.add_subplot(2,3,3)
ax3.imshow(recovered_img_mag,cmap="gray")
ax3.title.set_text("Recovered Image")
fig.savefig("1Hologram.png")
plt.show()
```



0.0.2 2 Iterations

[6]:
```
img = cv2.imread(r"C:\Users\birge\Documents\Fourrier Optics\IMG\3.bmp", 0)
phase_mask= GXalg(img,2)
signal = np.ones(phase_mask.shape+(2,))
signal[:,:,0] = np.real(np.exp(phase_mask*1j))
signal[:,:,1] = np.imag(np.exp(phase_mask*1j))
recovered_img,recovered_img_mag = idft(signal)
```

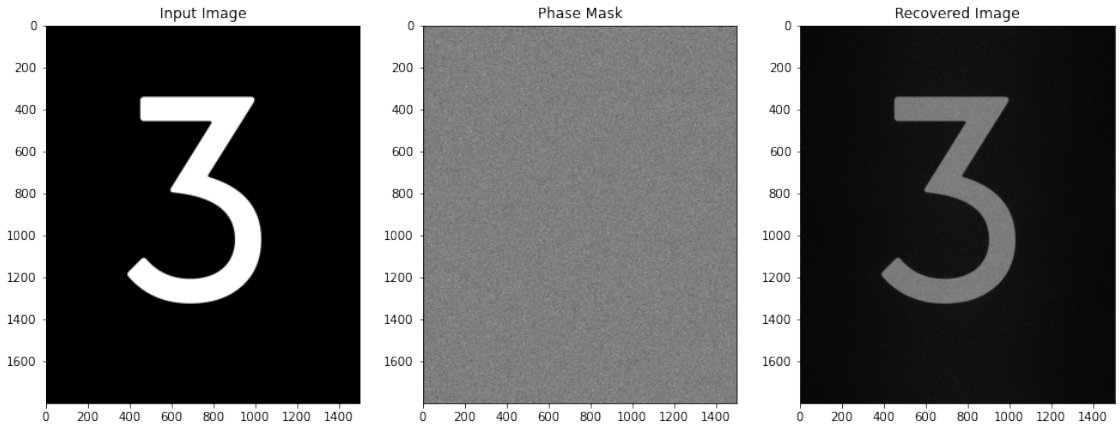
[7]:
```
fig = plt.figure(figsize=(16,16))
ax1 = fig.add_subplot(2,3,1)
ax1.imshow(img,cmap="gray")
ax1.title.set_text("Input Image")
ax2 = fig.add_subplot(2,3,2)
ax2.imshow(phase_mask,cmap="gray")
ax2.title.set_text("Phase Mask")
ax3 = fig.add_subplot(2,3,3)
ax3.imshow(recovered_img_mag,cmap="gray")
ax3.title.set_text("Recovered Image")
fig.savefig("2Hologram.png")
plt.show()
```



0.0.3 5 Iterations

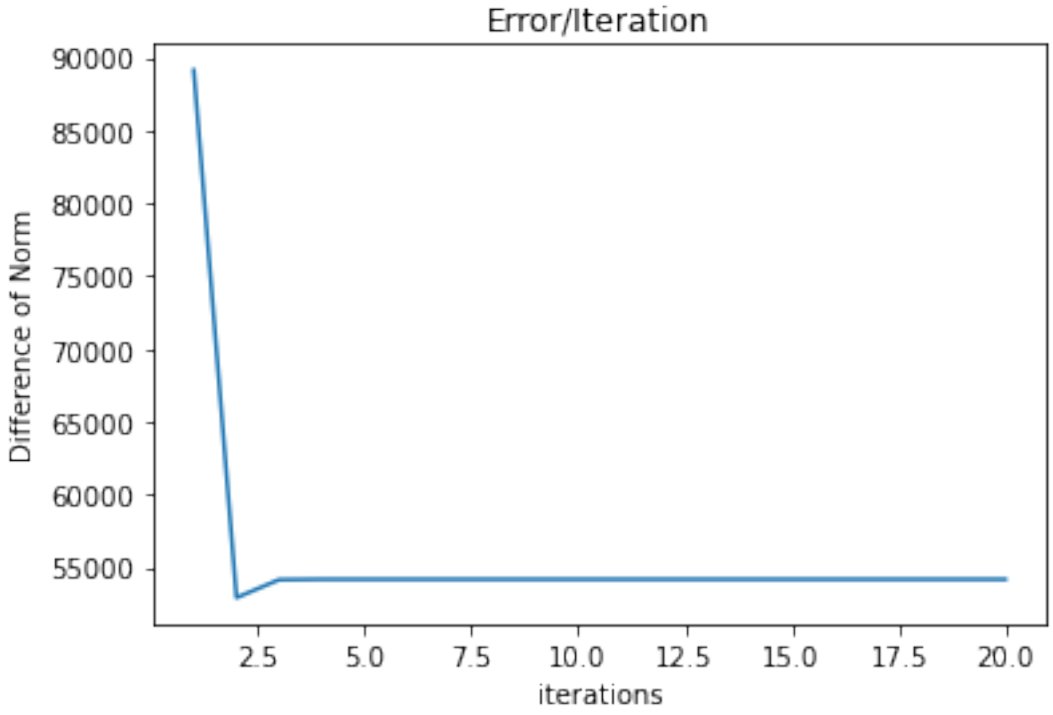
[8]:
```
img = cv2.imread(r"C:\Users\birge\Documents\Fourrier Optics\IMG\3.bmp", 0)
phase_mask= GXalg(img,5)
signal = np.ones(phase_mask.shape+(2,))
signal[:,:,0] = np.real(np.exp(phase_mask*1j))
signal[:,:,1] = np.imag(np.exp(phase_mask*1j))
recovered_img,recovered_img_mag = idft(signal)
```

[9]:
```
fig = plt.figure(figsize=(16,16))
ax1 = fig.add_subplot(2,3,1)
ax1.imshow(img,cmap="gray")
ax1.title.set_text("Input Image")
ax2 = fig.add_subplot(2,3,2)
ax2.imshow(phase_mask,cmap="gray")
ax2.title.set_text("Phase Mask")
ax3 = fig.add_subplot(2,3,3)
ax3.imshow(recovered_img_mag,cmap="gray")
ax3.title.set_text("Recovered Image")
fig.savefig("5Hologram.png")
plt.show()
```



0.0.4 20 Iterations + Error plot

[10]:
```
img = cv2.imread(r"C:\Users\birge\Documents\Fourrier Optics\IMG\3.bmp", 0)
phase_mask= GXalg(img,20, err_reduction_plot=True)
signal = np.ones(phase_mask.shape+(2,))
signal[:,:,0] = np.real(np.exp(phase_mask*1j))
signal[:,:,1] = np.imag(np.exp(phase_mask*1j))
recovered_img,recovered_img_mag = idft(signal)
```



[11]:
```
fig = plt.figure(figsize=(16,16))
ax1 = fig.add_subplot(2,3,1)
ax1.imshow(img,cmap="gray")
ax1.title.set_text("Input Image")
ax2 = fig.add_subplot(2,3,2)
ax2.imshow(phase_mask,cmap="gray")
ax2.title.set_text("Phase Mask")
ax3 = fig.add_subplot(2,3,3)
ax3.imshow(recovered_img_mag,cmap="gray")
ax3.title.set_text("Recovered Image")
fig.savefig("20Hologram.png")
plt.show()
```