Assignment 1: Image propagation using
- Angular Spectrum Method -

Indian Institute of Technology Delhi
Computational Optical Imaging
PYL 759

`Tamal Majumder`
Entry No: 2023PHS7226

Assignment done under the supervision of
Prof. Kedar Khare

10 th August, 2024

1 Problem Statement

The objective of this project is to analyze image propagation using the Angular Spectrum Method. The steps involved are as follows:

1. **Select a Grayscale Image:**
   - Choose a grayscale image with dimensions greater than 200 × 200 pixels.
2. **Scale Pixel Values:**
   - Scale the pixel values of the image by a factor β, where $\beta = \frac{\pi}{2\times 255}$. This adjusts the pixel values to the range $[0, \frac{\pi}{2}]$.
3. **Create the Complex Field:**
   - Define a complex field $u(x, y)$ using the formula:

     $$u(x, y) = \exp[i\beta g(x, y)]\ \text{circ}(x^2 + y^2 \leq 100^2)$$

     where $g(x, y)$ is the scaled grayscale image and 'circ' denotes a circular aperture with a radius of 100 pixels.
4. **Define Parameters:**
   - Set the pixel size to 5 µm and the wavelength to 0.5 µm.
   - Define the propagation distance as 100 µm.
5. **Propagate the Complex Field:**
   - Use the Angular Spectrum Method to propagate the complex field $u(x, y)$ over 100 µm:
     - Perform a Fourier transform on $u(x, y)$.
     - Apply a propagation transfer function.
     - Use a low pass filter to avoid aliasing.
     - Perform an inverse Fourier transform to get the propagated field.
6. **Compute and Display Intensity:**
   - Calculate the intensity $|u(x, y)|^2$ of the propagated field.
   - Display this intensity as an image.

2 Results

The following images illustrate Phase images at different distances:

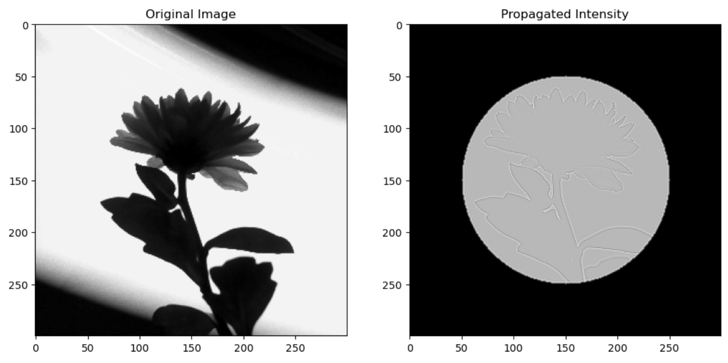


Figure 1: Propagated 1000 micrometer

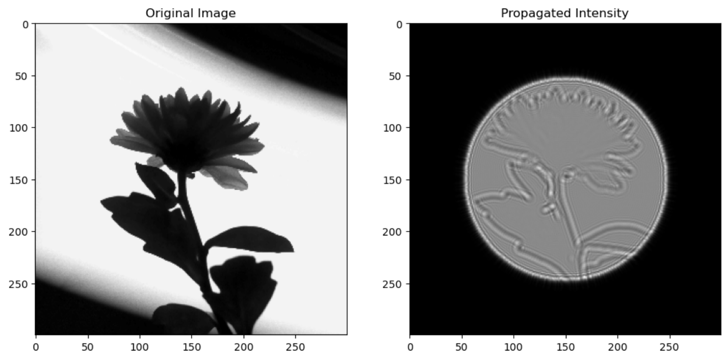


Figure 2: Propagated 1 mm (1000 micrometer)

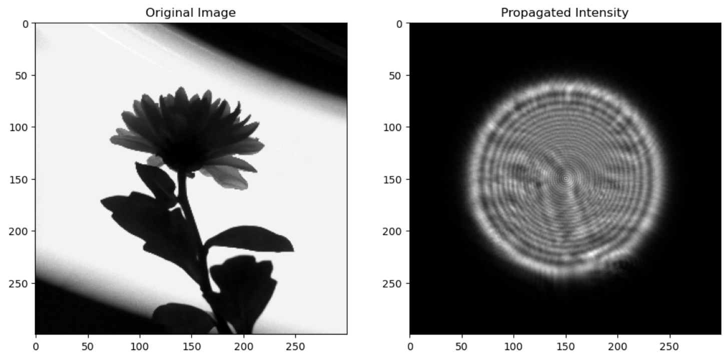


Figure 3: Propagated 10 mm (10000 micrometer)

3 A Brief Explanation about the Parameters

In this project, we investigated the propagation of a grayscale image using the Angular Spectrum Method, a technique for analyzing how phase objects influence light propagation.

3.1 Image Selection and Scaling

We began by choosing a grayscale image with dimensions greater than 1000×1000 pixels. Grayscale images represent different shades of gray, where pixel values correspond to intensity levels. Then, we scaled these pixel values by a scaling factor $\beta = \frac{\pi}{2\times 255}$. This scaling converts pixel values from the typical 0-255 range to a phase range $[0, \frac{\pi}{2}]$, which is crucial for accurately modeling phase modulation. The scaling ensures that the intensity values are appropriately transformed into phase values.

3.2 Creating the Complex Field

We defined a complex field $u(x, y)$ using the formula:

$$\exp[i\beta g(x, y)]\ \mathrm{circ}(x^2 + y^2 \leq 100^2)$$

- $\exp[i\beta g(x, y)]$: This term represents the phase modulation induced by the scaled grayscale image $g(x, y)$. The exponential function models the phase shift caused by the image's intensity.
- $\texttt{circ}(x^2 + y^2 \leq 100^2)$: This represents a circular aperture function with a radius of 100 pixels. It defines the region where the phase modulation occurs by restricting the complex field to a circular area of radius 100 pixels, centered at the origin. This function effectively simulates the optical effect of an aperture or lens by confining the phase modulation to this specified circular region.

3.3 Defining Experimental Parameters

We set the pixel size to 5 µm and the wavelength (λ) to 0.5 µm (Green Light). The pixel size determines the spatial resolution of the image, while the wavelength affects the spatial frequency components of the field. The propagation distance was set to 100 µm, specifying how far the phase-modulated field travels.

3.4 Propagating the Complex Field Using the Angular Spectrum Method

The Angular Spectrum Method involves several steps:

- **Fourier Transform:** We performed a 2D Fourier transform of the complex field to shift the problem into the frequency domain. This transformation simplifies the application of the propagation effects.
- **Propagation Transfer Function:** We calculated the transfer function H, which models how each frequency component propagates over the distance. The transfer function is crucial for accurately simulating the propagation effects.
- **Low Pass Filtering:** To prevent aliasing and ensure accurate representation, we applied a low pass filter. This filter removes high-frequency components beyond the Nyquist frequency, which could otherwise introduce artifacts in the final image.
- **Inverse Fourier Transform:** Finally, we performed an inverse Fourier transform to convert the field back to the spatial domain, providing the propagated field as an image.

3.5 Computing and Displaying Intensity

We computed the intensity $|u(x, y)|^2$ of the propagated field. Intensity, which is the square of the magnitude of the complex field, provides a visual representation of how the phase modulation affects the light distribution after propagation. This intensity image allows us to observe and analyze the impact of the phase object on the propagated light.

4 Conclusion

Overall, this approach demonstrates the application of Fourier optics in analyzing the influence of a phase object on light propagation. By scaling pixel values, creating a complex field with a circular aperture, and using the Angular Spectrum Method for propagation, we effectively simulated and interpreted the optical effects induced by the phase variations in the image.

::::: C O D E :::::

In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
import cv2
```

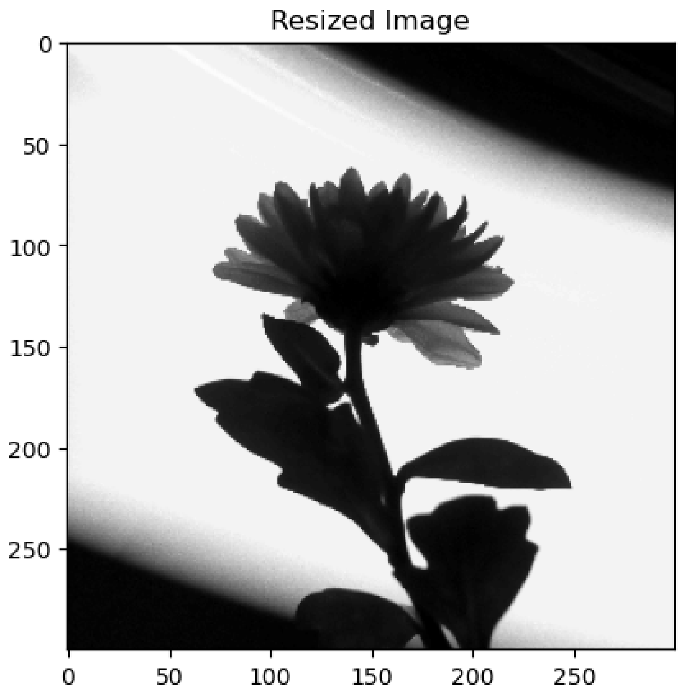
Parameters

In [2]:
```python
# Parameters
p = 5e-6  # pixel pitch (in meters)
lambda_= 500e-9  # wavelength (in meters)
k = 2 * np.pi / lambda_  # wavenumber
z = 100e-6  # propagation distance (in meters)
beta = np.pi / (2 * 255)  # scaling factor
```

Load the image and convert to grayscale

In [3]:
```python
# Load the image and convert to grayscale
image_path = r'C:\Users\987ta\Desktop\Test2.jpg'  # Replace with your image path
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

# Resize the image to 250x250 pixels
image_resized = cv2.resize(image, (300,300))
plt.imshow(image_resized, cmap='gray')
plt.title('Resized Image')
plt.show()
print('Resized pixel value: ', image_resized[100, 100])
```

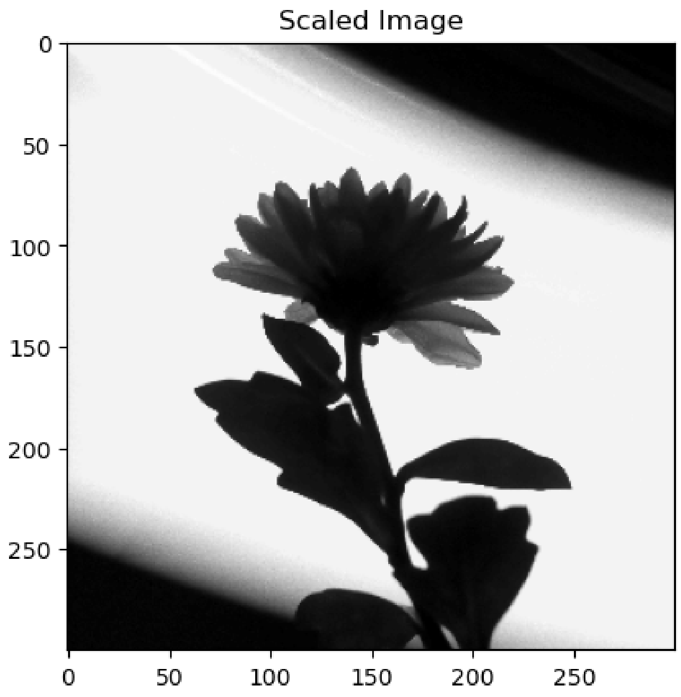


```
Resized pixel value:  11
```

Scale the image pixel values

In [4]:
```
# Scale the image pixel values
image_scaled = beta * image_resized
plt.imshow(image_scaled, cmap='gray')
plt.title('Scaled Image')
print('Scaled pixel value: ', image_scaled[100, 100])
```

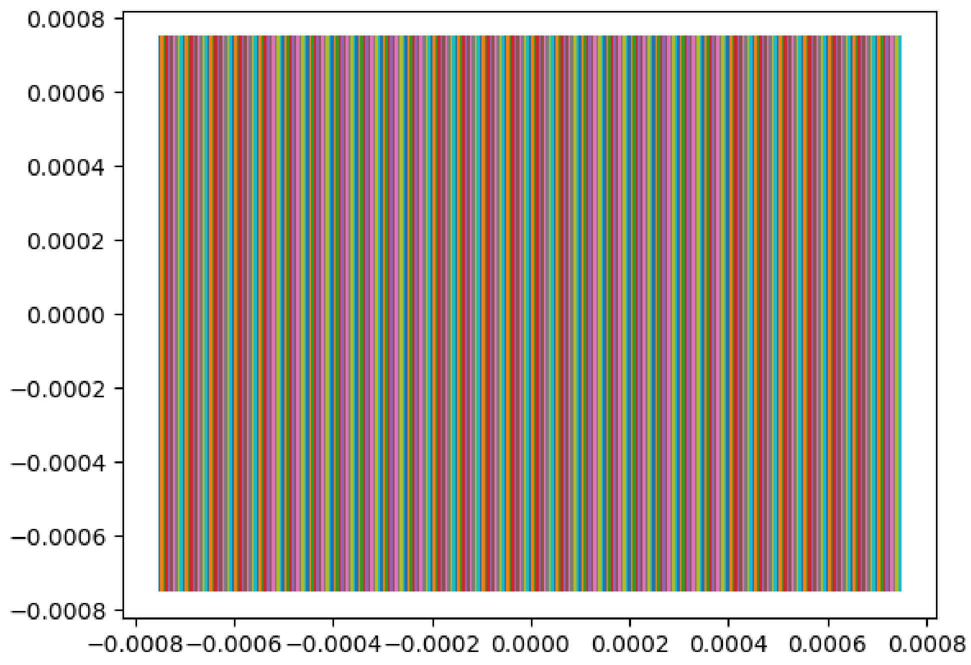
```
Scaled pixel value:  0.06775984154801515
```



In [5]:
```
# Get the dimensions of the image
rows, cols = image_resized.shape
```

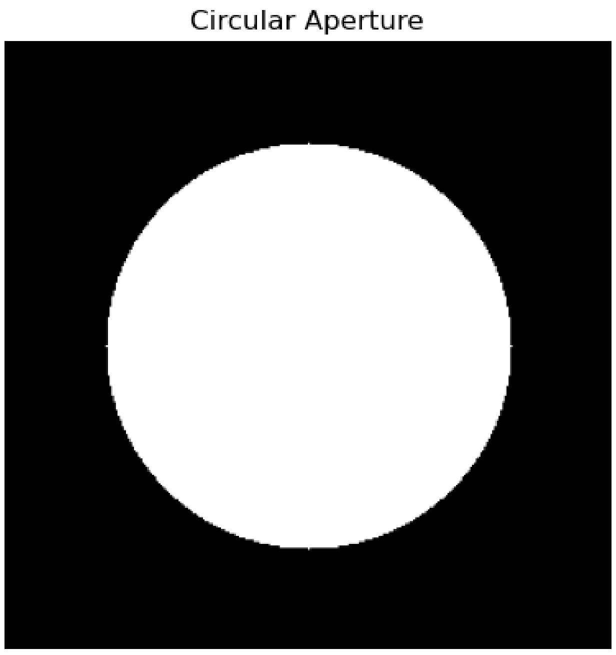
Create the coordinate grids

In [6]:
```
# Create the coordinate grids
x = np.linspace(-cols//2, cols//2 - 1, cols) * p
y = np.linspace(-rows//2, rows//2 - 1, rows) * p
x, y = np.meshgrid(x, y)
plt.plot(x,y)
plt.show()
```



Create the circular aperture function (circ function)

In [7]:
```
aperture_radius = 100  # radius in pixels
circ = (x**2 + y**2) <= (aperture_radius * p)**2
plt.imshow(circ, cmap='gray')
plt.title('Circular Aperture')
plt.axis('off')
plt.show()
```

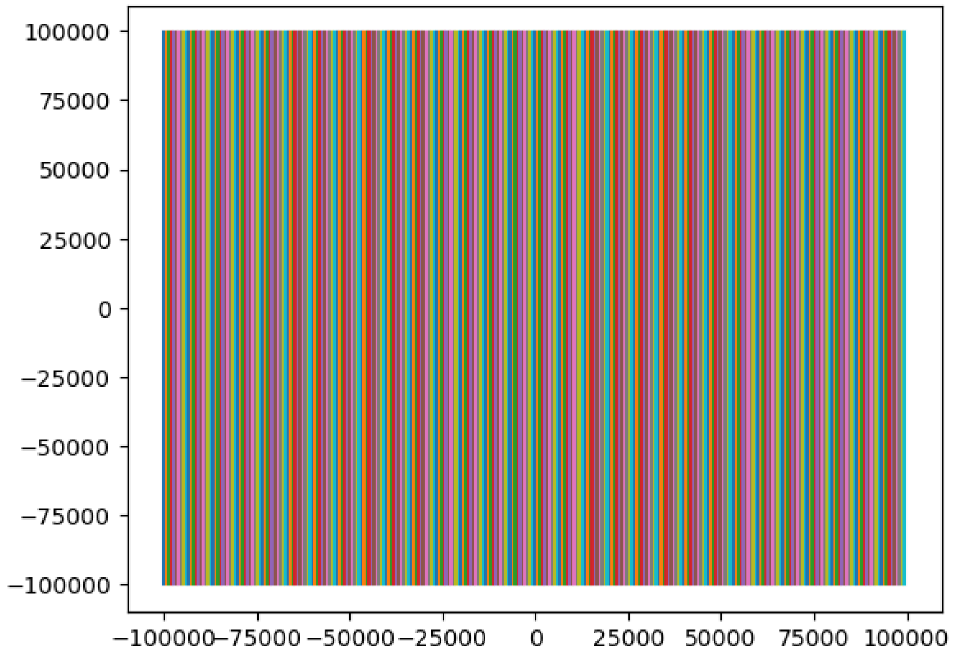


In [8]:
```
# Create the complex field u(x, y) = exp[iθg(x, y)] circ(x² + y²)
u = np.exp(1j * image_scaled) * circ
```

In [9]:
```
# Fourier transform of the complex field
U = np.fft.fftshift(np.fft.fft2(np.fft.ifftshift(u)))
```

Create the frequency grids

In [10]:
```
fx = np.fft.fftshift(np.fft.fftfreq(cols, d=p))
fy = np.fft.fftshift(np.fft.fftfreq(rows, d=p))
fx, fy = np.meshgrid(fx, fy)
plt.plot(fx,fy)
plt.show()
```



Compute the propagation transfer function

```
In [11]: alpha = np.sqrt(np.maximum(k**2 - (2 * np.pi * fx)**2 - (2 * np.pi * fy)**2, 0))  # Use maximum to avoid com
         H = np.exp(1j * alpha * z)
```

Low pass filter (optional but recommended to avoid aliasing artifacts)

```
In [12]: cutoff_freq = 1 / (2 * p)  # Nyquist frequency
         low_pass_filter = np.sqrt(fx**2 + fy**2) < cutoff_freq
         U_filtered = U * low_pass_filter
```

```
In [13]: U1 = U_filtered * H
```

Inverse Fourier transform to get the propagated field

```
In [14]: u1 = np.fft.ifftshift(np.fft.ifft2(np.fft.fftshift(U1)))
```

Compute the intensity of the propagated field

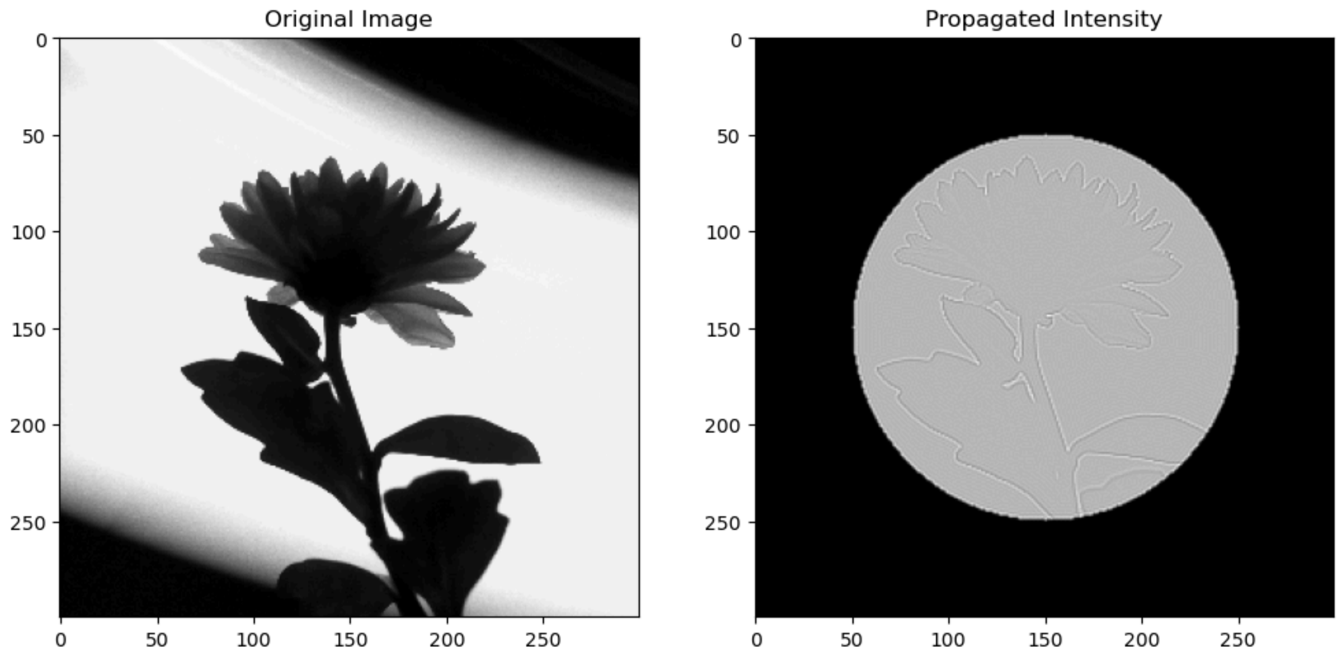
```
In [15]: intensity = np.abs(u1)**2
```

Plot the results

```
In [16]: plt.figure(figsize=(12, 6))

         plt.subplot(1, 2, 1)
         plt.title("Original Image")
         plt.imshow(image_resized, cmap='gray')

         plt.subplot(1, 2, 2)
         plt.title("Propagated Intensity")
         plt.imshow(intensity, cmap='gray')
         plt.show()
```


Original Image
Propagated Intensity