Fourier Optics and Computational Imaging

Angled Illuminations in 4f System and Advantages in Resolution

---

Team Number:- ***10***

Team Members:- ***Aman Razdan***

Utkarsh Jain

Course:- ***B.Sc. (Hons.) Physics***

Year Of Study: ***3rd***

Hindu College, University of Delhi

Submitted on: March 8, 2025

Abstract

In this project, the effect of ***angled illumination*** in *4f systems* has been studied in detail, and an effort has been made to highlight the importance of ***Ptychography*** in overcoming the limitation of the Numerical Aperture. Most imaging systems and microscopes are based on the 4f system model. Like some other optical systems, the 4f system is limited by its numerical aperture, due to the low NA of the lens. However, by employing angled illumination, we can virtually overcome the aperture limitation by shifting the image in the *Fourier domain*, thus capturing higher frequency components as well. Several algorithms can achieve this, the entire class of which is referred to as the *Fourier Ptychography Algorithms*. In this paper, one such algorithm has been designed and implemented on a standard 512×512 grayscale image and the results obtained have been presented and analysed. Further, the algorithm used has also been discussed in detail.

1 Mathematical Background

This section will begin with the explanation of some basic mathematical concepts which underlay the core of all computational imaging and that of our algorithm. We will begin with the Fourier Transform, and follow it with the key properties, and its implications in convolutions.

1.1 Fourier Transform:

The foundation for this project lays entirely on the shoulder of one concept which is the **Fourier Transform**. The Fourier transform *(FT)* is an *integral transform* that takes a function as input and outputs another function that describes the extent to which various frequencies are present in the original function[4]. Simply, FT maps a signal/function in space or time to a complex valued function in frequency domain. Mathematically speaking, let a function $f(x)$ represent some signal in space, then the Fourier transform is given by

$$F(\nu) = \mathcal{F}\{f(x)\} = \int_{-\infty}^{\infty} f(x) e^{-i2\pi\nu x}\, dx, \tag{1}$$

where ν is the spacial frequency corresponding to the function $f(x)$. Similarly, for a 2-D function $f(x, y)$, the Fourier Transform is given by,

$$F(\nu_x, \nu_y) = \mathcal{F}\{f(x, y)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y) e^{-i2\pi(\nu_x x + \nu_y y)}\, dxdy, \tag{2}$$

It is important to mention that the transform is not applicable on every function, i.e., it applies on a class of functions which fulfil certain conditions known as the ***Dirichlet conditions*** as given:

1. The function $f(x)$ must be **absolutely integrable** over the entire interval, i.e.,
$$\int_{-\infty}^{\infty} |f(x)|\, dx < \infty$$
2. The function $f(x)$ has a finite number of maxima and minima in every finite interval of x.
3. The function $f(x)$ has a *finite* number of discontinuities in every finite interval of x. Also, each of these discontinuities must be *finite*.

We know from Euler's relation that $e^{i\theta} = \cos(\theta) + i\sin(\theta)$, so eq. 1 can be written as

$$F(\nu) = F_{\text{real}}(\nu) + F_{\text{img}}(\nu),$$

where $F_{\text{real}}(\nu)$ is the real part and $F_{\text{img}}(\nu)$ is the imaginary part of the function $F(\nu)$. Therefore, the magnitude of the function $F(\nu)$ is given by,

$$|F(\nu)| = \sqrt{F^2_{\text{real}}(\nu) + F^2_{\text{img}}(\nu)} \tag{3}$$

and the phase of the function $F(\nu)$ is given by,

$$\angle F(\nu) = \tan^{-1}\left(\frac{F_{\text{img}}(\nu)}{F_{\text{real}}(\nu)}\right) \tag{4}$$

Another point of information is that prior to this, no mention of the *periodicity* of the function was done. This is because the Fourier Transform works for both *periodic and aperiodic* functions. However,

in case of periodic functions, it is often convenient to represent the function as a trigonometric series, and this series is known as a **Fourier Series**. The Fourier series of a complex-valued P-periodic function $s(x)$, integrable over the interval $[0, P]$ is defined as:

$$s(x) = \sum_{-\infty}^{\infty} c_n e^{i2\pi \frac{n}{P} x},$$

such that the Fourier coefficients c_n are complex numbers defined by

$$c_n = \frac{1}{P} \int_0^P s(x) e^{i2\pi \frac{n}{P} x} \, dx$$

Finally, like the FT of a function in spatial domain is mapped to the frequency domain, we can perform the Inverse Fourier Transform (IFT) on a function in the frequency domain to find its analog in the spatial domain, as shown below:

$$f(x) = \mathcal{F}^{-1}\{F(\nu)\} = \int_{-\infty}^{\infty} F(\nu) e^{i2\pi\nu x} \, d\nu \tag{5}$$

The equations **??** and 5 together are referred to as the *Fourier Transform Pair* denoted by

$$f(x) \iff F(\nu)$$

1.2 Properties of the Fourier Transform:

Understanding the properties of the Fourier transform enable us to make the most out of this method. This properties are as follows:

→ **Linearity Property:** If $f(x)$ and $g(x)$ are two function with the transforms $F(\nu)$ and $G(\nu)$ respectively, then

$$\mathcal{F}\{af(x) + bg(x)\} = a\mathcal{F}\{f(x)\} + b\mathcal{F}\{g(x)\} = aF(\nu) + bG(\nu)$$

→ **Spatial Shifting Property:** If $f(x)$ is a function with transform $F(\nu)$, then we have

$$\begin{aligned}
\mathcal{F}\{f(x - x_0)\} &= \int_{-\infty}^{\infty} f(x - x_0) e^{-i2\pi\nu x} \, dx \\
&= \int_{-\infty}^{\infty} f(x') e^{-i2\pi\nu x'} e^{-i2\pi\nu x_0} \, dx \\
&= e^{-i2\pi\nu x_0} \mathcal{F}\{f(x)\} \\
&= e^{-i2\pi\nu x_0} F(\nu)
\end{aligned}$$

→ **Frequency Shifting Property:** The frequency shifting property states that

$$\begin{aligned}
\mathcal{F}\{e^{i2\pi\nu_0 x} f(x)\} &= \int_{-\infty}^{\infty} f(x) e^{-i2\pi(\nu - \nu_0)x} \, dx \\
&= F(\nu - \nu_0)
\end{aligned}$$

→ **Scaling Property:** If $f(x)$ is a function with the transform $F(\nu)$, and $a > 0$ is some constant, then

$$\begin{aligned}
\mathcal{F}\{f(ax)\} &= \int_{-\infty}^{\infty} f(ax) e^{-i2\pi\nu x} \, dx \\
&= \frac{1}{|a|} \int_{-\infty}^{\infty} f(x') e^{-i2\pi \frac{\nu}{a} x'} \, dx \\
&= \frac{1}{|a|} F\left(\frac{\nu}{a}\right)
\end{aligned}$$

→ **Differentiation Property:** If $f(x)$ is a function with the transform $F(\nu)$, then

$$\mathcal{F}\left\{\frac{df}{dx}\right\} = i2\pi\nu F(\nu)$$

More generally, we have

$$\mathcal{F}\left\{\frac{d^n f}{dx^n}\right\} = (i2\pi\nu)^n F(\nu)$$

→ **Integration Property:** If $f(x)$ is a function with the transform $F(\nu)$, then

$$\mathcal{F}\left\{\int f(x)\,dx\right\} = \frac{1}{i2\pi\nu}F(\nu) + \delta(\nu)$$

→ **Shear Theorem:** If $f(x, y)$ is a function with the transform $F(\nu_x, \nu_y)$ and a, b are some constants, then

$$\mathcal{F}\{f(x + ay, y)\} = F(\nu_x, \nu_y - a\nu_x)$$
$$\text{and, } \mathcal{F}\{f(x, y + bx)\} = F(\nu_x - b\nu_y, \nu_y)$$

And finally, a very important property, known as the Parseval's Theorem, is defined as follows,

→ **Parseval's Theorem:** For a function $f(x)$ and its FT $F(\nu)$, the theorem states that the total energy (or power) of the function in the space domain is equal to the total energy (or power) of its Fourier Transform in the frequency domain, i.e.,

$$\int_{-\infty}^{\infty} |f(x)|^2\,dx = \int_{-\infty}^{\infty} |F(\nu)|^2\,d\nu$$

1.3 Convolution and Fourier Transform:

Convolution is a mathematical operation on two functions (f and g) that produces a third function $(f * g)$. It is defined in terms of the following integral:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(\xi)g(x - \xi)\,d\xi \tag{6}$$

Graphically, it expresses how the 'shape' of one function is modified by the other. Now, there exists a wonderful theorem known as **Convolution Theorem** which states that under suitable conditions, the Fourier transform of a convolution of two functions (or signals) is the product of their Fourier transforms. More generally, convolution in one domain (e.g., spatial domain) equals point-wise multiplication in the other domain (e.g., frequency domain). A proof of this is as follows:

$$\begin{aligned}
\mathcal{F}\{f * g\} &= \int_{-\infty}^{\infty} (f * g)(x)e^{-i2\pi\nu x}\,dx \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\xi)g(x - \xi)e^{-i2\pi\nu x}\,d\xi dx \\
&= \int_{-\infty}^{\infty} f(\xi)e^{-i2\pi\nu\xi}\,d\xi \int_{-\infty}^{\infty} g(x - \xi)e^{-i2\pi\nu(x-\xi)}\,dx \\
&= F(\nu) \cdot G(\nu)
\end{aligned}$$

2 Introduction to the Physical Aspects

This section involves the understanding of the physical concepts like wave nature of light, behaviour of lens and working of 4f systems, with brief overview of Numerical Aperture. Finally, a theoretical explanation of the need for Ptychography has been provided. Following few topics are based on the exposition given in [1] and [2].

2.1 Wave Theory of Light:

The wave theory, initially developed by Christian Huygens in 1690, suggested that light travels in longitudinal waves similar to sound waves in air. However, it was Fresnel and Young who showed that light in fact propagates as a transverse wave and this successfully explained the reflection, refraction, interference, diffraction, and polarization of light waves. There are certain key terminologies related to this theory which would be used several times across this paper and these are:

***Wave Front*:** A wave front is defined as a surface over which the *phase* of the wave is *constant*. In a particular wave front, at a given moment of time, all particles of the medium are undergoing the same motion. Although there are several kinds of wave fronts depending upon the nature of the source, within the scope of this paper, we are interested in Plane Wave Front. The equation of a wave characterised by planar wave fronts travelling along the $z-$direction is given by

$$U(x, y; z) = Ae^{ikz}, \tag{7}$$

where A is the amplitude of the wave and $k = \dfrac{2\pi}{\lambda}$ is the propagation constant with λ as the wavelength. Further, if we have a *Tilted Plane Wave*, propagating along some vector $\vec{r}$ in space, then we have a more general equation of the wave is

$$U(\vec{r}) = Ae^{i\vec{k}\cdot\vec{r}} = Ae^{i(k_x x + k_y y + k_z z)}, \tag{8}$$

where $k = |\vec{k}| = \sqrt{k_x^2 + k_y^2 + k_z^2} = \dfrac{2\pi}{\lambda}$

***Huygens-Fresnel Principle*:** The Huygens–Fresnel principle states that every point on a wavefront is itself the source of *spherical wavelets*, and the secondary wavelets emanating from different points mutually interfere. The sum of these spherical wavelets thus form a new wavefront. This principle helps in understanding various phenomena like wave propagation in *near-field diffraction* and *far-field limit*, etc.

***Fresnel-Kirchhoff Diffraction Formula*:** This formula approximates light intensity and phase in optical diffraction,i.e., light fields in the boundary regions of shadows. The previously discussed Huygens-Fresnel Principle is also derived by this formula.

2.1.1 Fresnel Diffraction:

The Fresnel diffraction is an approximation of the Fresnel-Kirchhoff diffraction that can be applied to the propagation of waves in the near-field, i.e., when the distances between the source, obstacle (or, aperture) and the screen are finite and comparable. To understand the Fresnel diffraction integral formulation, we start with the expression of the electric-field diffraction pattern at a point (x, y, z) as obtained from the solution of the Helmholtz equation, given as follows (refer to Fig. 1):

$$E(x, y, z) = \frac{1}{i\lambda} \iint_{-\infty}^{\infty} E(x', y', 0) \frac{e^{ikr}}{r} \frac{z}{r} \left(1 + \frac{i}{kr}\right) dx' dy',$$

where $E(x', y', 0)$ is the electric field at the aperture and $r = \sqrt{(x - x')^2 + (y - y')^2 + z^2}$

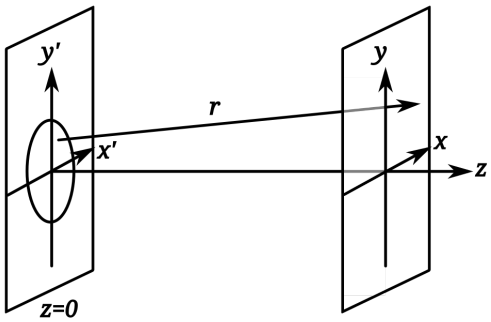


Figure 1: Diffraction Geometry, showing Aperture and Image plane

In order to simplify the above integral, we apply the *Fresnel approximation*, i.e.,

$$r \approx z + \frac{(x - x')^2 + (y - y')^2}{2z}$$

which then gives us the simpler formulation as

$$E(x, y, z) = \frac{e^{ikz}}{i\lambda z} \iint_{-\infty}^{\infty} E(x', y', 0) \exp\left(\frac{ik}{2z}\left[(x - x')^2 + (y - y')^2\right]\right) dx' dy' \tag{9}$$

This is the **Fresnel diffraction integral** and it means that, if the Fresnel approximation is valid, the propagating field is a *spherical wave*, originating at the aperture and moving along z. A closer inspection of the form of the integral highlights a resemblance to the 2-D Fourier transform, along with an additional phase term.

2.1.2 Fraunhofer Diffraction:

the Fraunhofer diffraction models the diffraction of waves when plane waves are incident on a diffracting object, and the diffraction pattern is viewed at a sufficiently long distance (a distance satisfying

Fraunhofer condition) from the object, and also when it is viewed at the focal plane of an imaging lens. If d is the largest size of the diffracting aperture, D is the distance between the aperture and the screen and λ is the wavelength of light, then the *Fraunhofer condition* is given by

$$D \gg \frac{d^2}{\lambda} \qquad \text{or,} \quad \frac{d^2}{D\lambda} \ll 1$$

The Fraunhofer diffraction equation (refer to Fig. 1) is thus given by,

$$E(x,y,z) \approx \frac{e^{ikz} e^{\frac{ik(x^2+y^2)}{2z}}}{i\lambda z} \iint_{\text{Aperture}} E(x',y',0) e^{-i\frac{k}{z}(xx'+yy')}\, dx'dy' \tag{10}$$

It can be seen that the integral in the above equations is the Fourier transform of the aperture function evaluated at frequencies,

$$\nu_x = \frac{x}{\lambda z}$$
$$\nu_y = \frac{y}{\lambda z}$$

2.2 Thin Lens Approximation:

When understanding the action of lens, it is important to realise that since the wavelength of light is less than a micron, it acts like a ray in the many common situations in which it encounters objects larger than a micron, such as lenses. Now, a thin lens is defined to be one whose thickness allows rays to refract, but does not allow properties such as dispersion and aberrations. An ideal thin lens has two refracting surfaces but the lens is thin enough to assume that light rays bend only once. Another important characteristic of a thin lens is that light rays through its centre are deflected by a negligible amount (refer to Fig. 2). The treatment of a lens as a thin lens is known as the *thin lens approximation.* Within the context of this paper, the lens we would consider will follow this approximation.

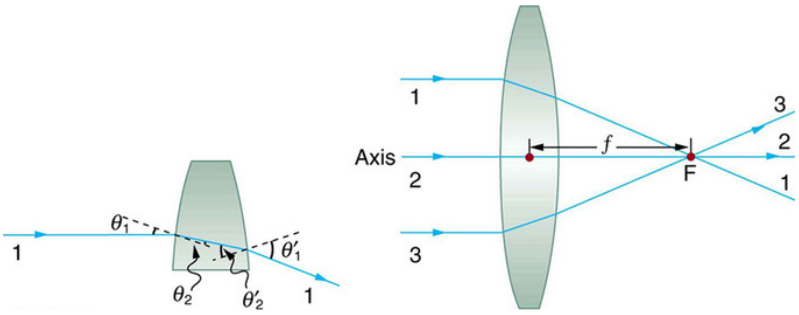


Figure 2: Ray tracing in Lens

2.3 Imaging System and the Numerical Aperture:

Imaging Systems are used for observation or image capture in a variety of applications including inspection or machine vision. Imaging Systems typically consist of a camera, imaging lens, along with an illumination source (see Fig. 3). Depending on the system setup, these can allow observed objects to be magnified or enhanced to ease the viewing or inspection of small or unclear objects. Furthermore, imaging systems can be self-contained units or modular component assemblies to meet the needs of a wide range of applications.

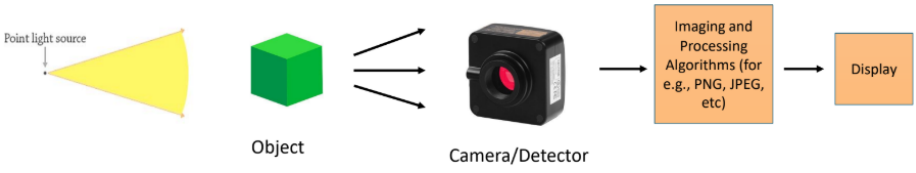


Figure 3: Components of an Imaging System

Now, every imaging system presents a limited capability in resolution that can be expressed as a function of the minimal distance that two infinitely small spatial features can be positioned near each

other while remaining two separable items in the image provided by the system. This is called the Rayleigh Resolution Limit, and is usually given as

$$\textbf{Rayleigh Resolution Limit}, \text{RL} = 1.22\frac{\lambda}{\text{NA}},$$

where NA is called the ***Numerical Aperture*** of the imaging system, and is defined as the light gathering power of the system. If θ_a is the maximum entrance angle of light from the source, entering the system, also called the *acceptance angle*, D is the size of the aperture of the imaging system and z is the distance between source and imaging system, then

$$\text{NA} = \sin(\theta_a) \approx \tan(\theta_a) = \frac{D}{z} \tag{11}$$

It is evident from the above equations that the ultimate goal of any imaging system is to maximize the numerical aperture, in order to improve the resolution limit and hence obtain precise and clear images.

2.4 Fourier Ptychography:

Before moving to the definition of ptychography and its algorithm, it is important that we understand the need to introduce such an algorithm. Many imaging systems, like the *microscopes* are limited by their small numerical aperture. This is because of the following reasons:

1. ***Long Working Distance*:** Many low-magnification objectives are designed to have a long working distance (the distance between the lens and the specimen). A longer working distance necessitates a smaller acceptance angle (θ_a), reducing the numerical aperture.
2. ***Depth of Field*:** A smaller NA increases the depth of field, which means more of the specimen remains in focus along the optical axis. This is advantageous for observing thicker samples.
3. ***Cost and Complexity*:** High-NA lenses require complex optical designs and high-quality glass, which increases manufacturing difficulty and cost. Simpler, lower-NA lenses can be sufficient for applications where high resolution is not needed.

However, there are issues associated with a smaller NA as well, like the *decreased resolution* and *reduced brightness*. So, the question arises: "***How can we virtually increase the Numerical Aperture (and hence improve the resolution) without physically altering the optical system?***" One of the popular solutions to this problem is the class of various algorithms known as Fourier ptychography. That is, we define ptychography as a computational imaging technique that involves the synthesis of a *wider numerical aperture* from a set of full-field images acquired at various *coherent illumination angles*, resulting in increased resolution. The reason this algorithm works is due to the shifting property of the Fourier Transform, and looking at the form of a tilted plane wave (see eq. 8), if were to take its FT, it would simply appear as a shifted version of the FT of a simple plane wave. This way, multiple angles of illumination result in the shifting of the image in the Fourier domain (refer to Fig. 4), thus helping in capturing higher frequency content while keeping the actual optical system intact.

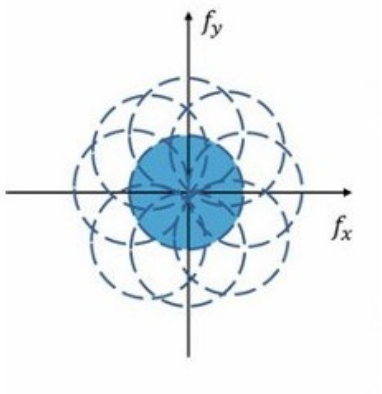


Figure 4: The spatial frequency coverage is illustrated by circles in the spatial frequency plane (Image Credit: Khare, Butola, Rajora)

We would now move on to the Methodology section to describe several aspects of the computation involved within this project.

3 Methodology

In this section, we would start with a detailed discussion on the Discrete and Fast-Fourier Transform and corresponding algorithms. This will be followed by a look into the world of Fourier Optics, starting with transmittance functions, and further developed by the understanding of the Angular Spectrum Method (ASM), Lens action and its association with the Fourier Transform, and finally the 4−f optical imaging system.

3.1 Discrete Fourier Transform (DFT)

DFT is a special case of the Fourier transform that is amenable to machine computation. If $f(x)$ and $F(\nu)$ form a FT pair, then the computation of the corresponding DFT approximation requires to sample the given $f(x)$ and $F(\nu)$ at a finite number of discrete points in the spacial and frequency domains respectively. Let the function $f(x)$ be sampled at steps $x_j = j\Delta x,\ j = 0, 1, 2, \ldots, (N-1)$, such that the total spatial length used to sample $f(x)$ is given as $L = N\Delta x$. So, the integral transform now becomes a summation with $f(x_j) = f_j$, i.e.

$$F(\nu) = \sum_{j=0}^{N-1} f_j e^{-i2\pi\nu x_j} \Delta x$$

Now, to incorporate all frequency components in the given space domain signal, we generate a frequency grid as $\nu_k = k\Delta\nu,\ k = 0, 1, 2, \ldots, (N-1)$ and $\Delta\nu = \dfrac{1}{L} = \dfrac{1}{N\Delta x}$. So, the above equation becomes

$$F_k = \sum_{j=0}^{N-1} f_j e^{-i2\pi\nu_k x_j} = \sum_{j=0}^{N-1} f_j e^{-i2\pi(k\Delta\nu)(j\Delta x)},$$

where $F_k = F(\nu_k)$. Now, using $\Delta\nu = \dfrac{1}{N\Delta x}$, we have

$$F_k = \sum_{j=0}^{N-1} f_j e^{-i2\pi\frac{kj}{N}} \tag{12}$$

Thus, eq. 12 gives the expression for Discrete Fourier Transform. Further, the factor $\dfrac{1}{\Delta x}$ is called the *Sampling Rate* and according to the ***Nyquist Sampling Theorem***, this sampling rate must be at least *twice* the largest frequency component of $f(x)$, for the DFT and FT to be equivalent. In matrix form, the above equation is given as

$$\begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ \vdots \\ F_{N-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \ldots & \omega^{N-1} \\ 1 & \left(\omega^2\right)^1 & \left(\omega^2\right)^2 & \left(\omega^2\right)^3 & \ldots & \left(\omega^2\right)^{N-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \left(\omega^{N-1}\right)^1 & \left(\omega^{N-1}\right)^2 & \left(\omega^{N-1}\right)^3 & \ldots & \left(\omega^{N-1}\right)^{N-1} \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{N-1} \end{bmatrix},$$

where $\omega = e^{i\frac{2\pi}{N}}$.So, we can write above matrix equation as

$$\hat{F}_k = \Omega_{kj}\hat{f}_j$$

This matrix equation will be useful for explaining the efficacy of the Fast Fourier Transform (FFT) algorithm.

3.2 Fast Fourier Transform (FFT)

As defined by the eq. 12, the computation for DFT requires N^2 complex operations, and consequently for data samples of even moderate size, DFT calculations can be extremely time-consuming. In order to improve the computationally efficiency, the Fast Fourier Transform was introduced was introduced to calculate the DFT of a sequence. It is a *divide-and-conquer* algorithm that recursively breaks the DFT into smaller DFTs to reduce the number of computations. In fact, FFT successfully reduces the complexity from $O(N^2)$ to $O(N \log N)$, where N is the size of the data. The reason FFT is possible

is due to the *symmetries* in the DFT. If we again take a look at eq. 12, we can see

$$\begin{aligned} F_{k+N} &= \sum_{j=0}^{N-1} f_j e^{-i2\pi \frac{(k+N)j}{N}} \\ &= \sum_{j=0}^{N-1} f_j e^{-i2\pi \frac{kj}{N}} e^{-i2\pi j} \\ &= \sum_{j=0}^{N-1} f_j e^{-i2\pi \frac{kj}{N}} = F_k \end{aligned}$$

We can thus generalize $F_{k+jN} = F_k$, for any integer j. In 1965, ***Cooley and Tukey*** made use of this symmetry property and showed that we can calculate DFT more efficiently if we continue to divide the problem into smaller ones. That is, the whole series is divided into two parts, the even and odd number parts such that each part is a smaller DFT of half the length as shown below:

$$\begin{aligned} F_k &= \sum_{j=0}^{N-1} f_j e^{-i2\pi \frac{kj}{N}} \\ &= \sum_{m=0}^{\frac{N}{2}-1} f_{2m} e^{-i2\pi \frac{k(2m)}{N}} + \sum_{m=0}^{\frac{N}{2}-1} f_{2m+1} e^{-i2\pi \frac{k(2m+1)}{N}} \\ &= \sum_{m=0}^{\frac{N}{2}-1} f_{2m} e^{-i2\pi \frac{km}{\frac{N}{2}}} + e^{-i2\pi \frac{k}{N}} \sum_{m=0}^{\frac{N}{2}-1} f_{2m+1} e^{-i2\pi \frac{km}{\frac{N}{2}}} \end{aligned}$$

For each term, we have $0 \leq m \leq \dfrac{N}{2}$ but $0 \leq k \leq N$. Therefore, half of the values will be the same due to the symmetry property as described above. Thus, we only need to calculate half of the fields in each term. Again, we can continue to divide each term into half with the even and odd values until it reaches the last two numbers, at which point the calculation becomes really simple. This describes the recursive nature of the FFT algorithm. We can similarly show in the matrix equation above that,

$$\hat{F}_n = \begin{bmatrix} I_{n/2} & -D_{n/2} \\ I_{n/2} & -D_{n/2} \end{bmatrix} \begin{bmatrix} \Omega_{n/2} & 0 \\ 0 & \Omega_{n/2} \end{bmatrix} \begin{bmatrix} f_{n/2}\big|_{even} \\ f_{n/2}\big|_{odd} \end{bmatrix}$$

$$\text{where, } D_{n/2} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & \omega & 0 & \dots & 0 \\ 0 & 0 & \omega^2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \omega^{n-1} \end{bmatrix}$$

And here as well, $\Omega_{n/2}$ can be subdivided further into smaller and simpler components. And due to the diagonal nature of the matrices $I_{n/2}$ and $D_{n/2}$, and the smaller size of $\Omega_{n/2}$, the FFT computation much more faster and efficient.

3.3 Transmittance, Point-Spread and Transfer Functions

The term transmittance is defined as the *ratio* of transmitted optical power to the incident optical power for some object, for example an optical system. And so, the ***transmittance function (TF)*** of an optical system is a *waveform* with a shape that dictates the *fraction* of input light that is transmitted at various wavelengths. Mathematically, if $g(x_1, y_1)$ is an input field which passes through a surface having transmittance function $t(x_1, y_1)$, then the output field is given by

$$u(x_1, y_1) = t(x_1, y_1) \cdot g(x_1, y_1)$$

Now, transmittance function $t(x, y)$ is of three types:

→ **Complex TF:** $t(x, y) = A(x, y)e^{i\theta(x,y)}$

→ **Amplitude TF:** $t(x, y) = A(x, y)$, i.e. $\theta(x, y) = 0$

→ **Phase TF:** $t(x, y) = e^{i\theta(x,y)}$, i.e. $\forall x, y,\ A(x, y) = 1$

Now, the Point-Spread function (PSF) of an optical system describes the system's response to a unit impulse, or a point, and hence the name. The Fourier Transform of the PSF gives the transfer

function, and it is defined as a measure of the ability of the system to transmit spatial information in the spatial frequency domain. In order to mathematically derive the proper expressions for them, we must know what we mean by an impulse. A unit impulse in the spatial domain is described by a *dirac-delta* function defined at some point in space. An interesting property of the delta function is that when it is *convolved* with a function, the result is just the copy of the function at the position of the delta function. Another important thing we need to know is that the optical systems, for most standard purposes, are *Linear Time-Invariant Systems*, i.e. for some input $x(t)$ to the system and some response $h(t)$ of the system, the output is simply

$$y(t) = h(t) * x(t)$$

Now, let $\mathcal{S}$ be a linear transformation describing an optical system's response to an input field, say $g(x_1, y_1)$, then we know

$$\begin{aligned} g_1(x_1, y_1) &= (g_1 * \delta)(x_1, y_1) \\ &= \iint_{-\infty}^{\infty} g_1(\xi, \eta)\delta(x_1 - \xi, y_1 - \eta)\, d\xi d\eta \end{aligned}$$

So, passing this input through the system, we get

$$\begin{aligned} g_2(x_2, y_2) &= \mathcal{S}\{g_1(x_1, y_1)\} \\ &= \mathcal{S}\{(g_1 * \delta)(x_1, y_1)\} \\ &= \mathcal{S}\left\{\iint_{-\infty}^{\infty} g_1(\xi, \eta)\delta(x_1 - \xi, y_1 - \eta)\, d\xi d\eta\right\} \\ &= \iint_{-\infty}^{\infty} g_1(\xi, \eta)\mathcal{S}\{\delta(x_1 - \xi, y_1 - \eta)\}\, d\xi d\eta \end{aligned}$$ [i]

Here, $\textbf{PSF} = \mathcal{S}\{\delta(x_1 - \xi, y_1 - \eta)\}$. Hence, this gives

$$g_2(x_2, y_2) = g_1(x_1, y_1) * \text{PSF} \tag{13}$$

Applying FT on both sides gives

$$G_2(\nu_{x_2}, \nu_{y_2}) = G_1(\nu_{x_1}, \nu_{y_1}) \cdot \mathcal{F}\{\text{PSF}\} = G_1(\nu_{x_1}, \nu_{y_1}) \cdot H, \tag{14}$$

where H is the transfer function.

3.4 Angular Spectrum Method (ASM)

The angular spectrum method is a technique for modelling the propagation of a wave field, i.e. for a given field $E(x, y, 0)$, we wish to find the field after it has propagated by a distance z given by $E(x, y, z)$. This technique involves expanding a complex wave field into a summation of large (ideally infinite) number of plane waves of the same frequency and propagating in different directions. ASM generally consists of the following steps:

1. ***Sample*** the complex (real and imaginary) components of a field over a grid of points lying in a cross-sectional plane within the field, i.e. we begin by discretizing $E(x, y, 0)$ along a $x - y$ grid.
2. Compute the ***2D-FFT*** of the field - this will decompose the field into a 2D "angular spectrum" of component plane waves each travelling in a unique direction.
3. Multiply each point in the 2D-FFT by a ***propagation term*** which accounts for the phase change that each plane wave will undergo on its journey to the prediction plane. This propagation term is calculated to be $e^{i\alpha z}$, where $\alpha = \sqrt{k^2 - 4\pi^2(f_x^2 + f_y^2)}$.
4. Take the ***2D-IFFT*** of the resulting data set to yield the field over the prediction plane.

3.5 Lens Action

We know that in wave optics, the change in the speed of propagation through glass introduces a phase-shift on the wave field. This phase-shift will be proportional to the length of the path traversed by light. In addition, if we have a thin lens the effects of propagation within the lens can be neglected. Therefore, the wave field is only multiplied by a phase factor. As discussed before, within this paper, we assume that our lens is thin, i.e., the incoming field at the lens is only multiplied by a phase factor. If we take a look at Fig. 5,

[i] $\mathcal{S}$ operates only on δ because it acts on x_1 and y_1, and the system is linear

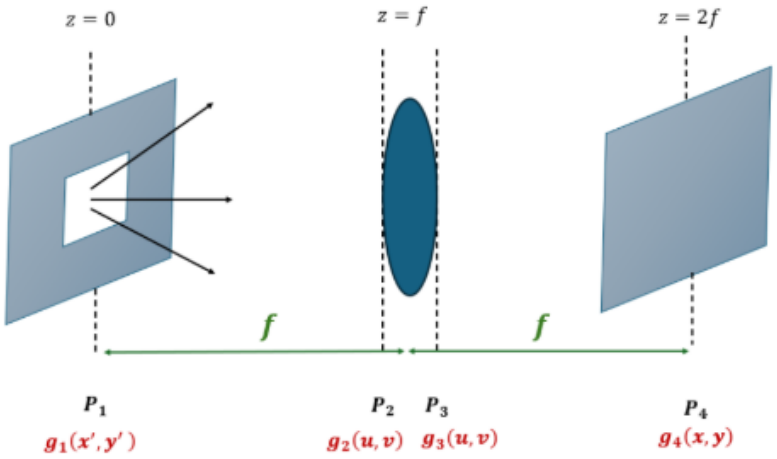


Figure 5: Action of Lens

The incoming field at plane P_2 is obtained by the ASM of the source field at P_1 through a distance equal to the focal length, f. This field is then multiplied by the phase factor introduced by the lens. This factor is called the *transmittance function of the lens* and is given by

$$t_l(u,v) = e^{-i\frac{\pi}{\lambda f}(u^2+v^2)}$$

Finally, the output field at P_4 is obtained by computing the ASM of the field at P_3 through a distance f. However, we saw from Section. 2.1.2, $\frac{u}{\lambda f} = \nu_u$ and $\frac{v}{\lambda f} = \nu_v$, so we find

$$t_l(u,v) = e^{-i\pi(\nu_u u + \nu_v v)},$$

which is the Fourier Transform of the input field at P_2. Hence, the output field at P_4 is actually the FT of the source field, magnified by a factor of

$$M \propto \lambda f$$

.

3.6 4f Imaging System

If we closely examine the system given in Fig. 6, it consists of 2 lenses placed at a distance $z = 2f$. From the physical understanding of lens action, we know that the field at the output plane P_3 will be the inverted source field of the same size. But the idea of key interest here, is the plane P_2 which is called as the ***Fourier Plane*** or the ***Filter Plane***. This plane is called Fourier Plane because it is in this plane that we have the Fourier Transform of the source field. However, it is also called the Filter plane as it is in this plane, that we place a *low-pass*, *high-pass* or *band-pass* filters depending upon the requirement of the experiment. For the purposes of our project, we have applied a low-pass filter, to highlight the efficacy of our Ptychography Algorithm. The steps of simulating the 4f system is same applying two lens actions back to back.

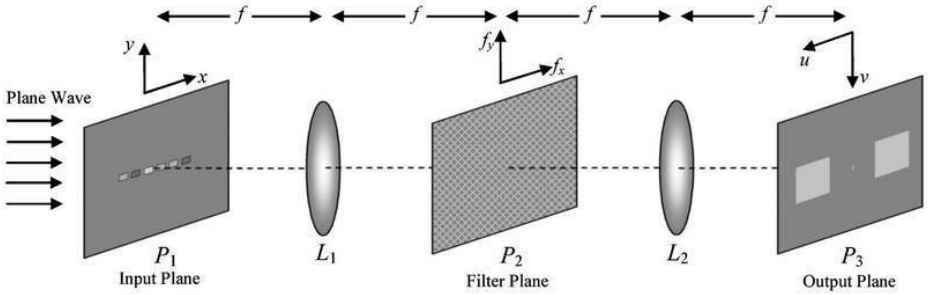


Figure 6: A 4f Imaging System

4 Computational Aspects

In this section, we describe the ***Python*** implementation of the problem described in Section. 2.4. The general procedure is as follows:

→ **Step-1:** ***Importing the required libraries:***

```
from numpy.fft import fft2,ifft2,fftshift,ifftshift ## fourier transform operations
import matplotlib.pyplot as plt                     ## for plotting
import numpy as np                                    ## for numerical calculation
import scipy                                        ## only used for fftconvolve
import cv2                                           ## for convenient grayscaling
plt.style.use('seaborn-v0_8-poster')

## Please note: This notebook has been programmed to handle square images only.
```

→ **Step-2:** ***Defining the Class for simulating Imaging Systems:***

1. The class supports 2 input formats- either *string* (filename) or *numpy* array.

```
class fourier_optics:

    def __init__(self,file,mode,ld=500,p=3):
        '''
        reads the image file and initializes the object
        arguments:-
        file: either ndarray or str (the name of the image file)
        mode: str, the mode of the image, 'grayscale' or 'rgb'
        ld: float, wavelength of the light in nm. Default is 500 nm
        p: float, pixel size in micrometer. Default is 3 micrometer


        returns: ndarray, the image data in the desired mode
        '''

        ## Our class supports 2 input formats- either string(filename) or numpy array:
        if type(file)==np.ndarray:
            self.data=np.array(file)
        else:
            ## if file is being and it is rgb, we have to convert it into grayscale:
            if mode not in ['grayscale','rgb']:
                raise ValueError('mode must be "grayscale" or "rgb"')

            self.data=cv2.imread(file)
```

2. Many properties of the system are used multiple times. To avoid calculating them multiple times, we define these as object attributes.

```
## We create several attributes that are required later.
        self.mode=mode
        self.ld=ld*(1e-9)
        self.p=p*(1e-6)
        self.k=2*np.pi/self.ld
        self.dx=self.dy=self.p
        N=self.data.shape[1]
        x=np.arange(-N/2,N/2,1)*self.dx
        y=np.arange(-N/2,N/2,1)*self.dy
        x,y=np.meshgrid(x,y)
        self.x=x
        self.y=y
        fX=np.arange(-1/2,1/2,1/N)/self.dx
        fY=np.arange(-1/2,1/2,1/N)/self.dy
        fX,fY=np.meshgrid(fX,fY)
        self.fX=fX
        self.fY=fY
```

→ **Step-3:** ***Creating the key functions:***

1. When we simply copy a list, we get a *shallow copy*. It means that when we make changes in new array using subscript, old array is also changed. To avoid this we make deep copy.

```
def deep_copy(self):
    '''
    returns a deep copy of the object
    arguments: None
    returns: fourier_optics object
    '''
    return fourier_optics(self.data,'grayscale',self.ld/(1e-9),self.p/(1e-6))
```

2. Function for the Fourier Transform;

```
def fourier_transform(self):
        '''
        returns the fourier transform of the image
        arguments: None
        returns fourier transform of the image
        '''
        return fftshift(fft2(ifftshift(self.data)))
```

3. Function for the Fraunhofer and Fresnel Diffraction;

```
def fraunhoffer_diffraction(self):
    '''
    returns the fraunhoffer diffraction pattern of the image
    arguments: None
    returns: fraunhoffer diffraction pattern of the image
    '''
    return self.fourier_transform()

def fresnel_diff(self,z):
    '''
    returns fresnel diffraction pattern of the image at a distance z
    arguments:
    z: float, distance of propagation in meters
    returns: fresnel diffraction pattern of the image at a distance z
    '''
    self.fourier_transform()
    alpha=np.sqrt(self.k**2-(4*(np.pi**2)*(self.fX**2+self.fY**2)))
    H=np.exp(1j*alpha*z)
    self.data*=H
    output=fftshift(ifft2(ifftshift(self.data)))
    return output
```

4. Function for adding and removing *padding*;

```
def add_padding(self,pad_th):
    '''
    adds padding to the image
    arguments:
    pad_th: int, the padding thickness
    returns: None
    '''
    self.data=np.pad(self.data,pad_th)
    N=np.shape(self.data)[0]
    x=np.arange(-N/2,N/2,1)*self.dx
    y=np.arange(-N/2,N/2,1)*self.dy
    x,y=np.meshgrid(x,y)
    self.x=x
    self.y=y
    fX=np.arange(-1/2,1/2,1/N)/self.dx
    fY=np.arange(-1/2,1/2,1/N)/self.dy
    fX,fY=np.meshgrid(fX,fY)
    self.fX=fX
    self.fY=fY
```

```
    def remove_padding(self,pad_th):
        '''
        removes padding from the image
        arguments:
        pad_th: int, the padding thickness
        returns: None
        '''
        self.data=self.data[pad_th:-pad_th,pad_th:-pad_th]
        N=np.shape(self.data)[0]
        x=np.arange(-N/2,N/2,1)*self.dx
        y=np.arange(-N/2,N/2,1)*self.dy
        x,y=np.meshgrid(x,y)
        self.x=x
        self.y=y
        fX=np.arange(-1/2,1/2,1/N)/self.dx
        fY=np.arange(-1/2,1/2,1/N)/self.dy
        fX,fY=np.meshgrid(fX,fY)
        self.fX=fX
        self.fY=fY
```

5. Function to simulate the effect of *angled illumination*;

```
    def angled_illumination_phi(self,theta:float,phi:float):
        '''
        simulates the effect of angled illumination
        arguments:
        theta: float, Polar angle- the angle of illumination w.r.t. z-axis (in degrees)
        phi: float, Azimuthal angle- the direction of angled illumination in xy plane
        wrt +ve x-axis (in degrees)
        '''
        I0=self.data
        theta=np.pi*theta/180
        phi=np.pi*phi/180
        I=I0*np.exp(1j*self.k*(np.sin(theta)*(self.x*np.cos(phi)+self.y*np.sin(phi))))
        self.data=I
```

6. Function for extracting the data from the object;

```
    def image_data(self):
        '''
        returns the image data
        arguments: None
        returns: ndarray, the image data
        '''
        return self.data
```

→ **Step-4:** ***Defining the elementary functions of Fourier Optics:***

```
    def ASM(self,z):
        ''' This function computes the output image based on the
            angular spreading of the light waves
            corresponding to the transmittance function; A.K.A Fresnel Diffraction
        '''
        inp=self.data
        F_inp=fftshift(fft2(ifftshift(inp)))
        alpha=np.sqrt(self.k**2-4*np.pi**2*(self.fX**2+self.fY**2))
        H=np.exp(1j*alpha*z)
        F_out=F_inp*H
        out=fftshift(ifft2(ifftshift(F_out)))
        self.data=out

    def Lens(self,f):
        '''
        This function computes the image, at f, of an object place at f.
        Arguments:
        f: focal length of the lens
        '''
```

```
        self.ASM(f)
        lens_trans=np.exp(-1j*np.pi*(self.x**2+self.y**2)/(self.ld*f))
        self.data*=lens_trans
        self.ASM(f)


    def low_pass(self,f_max):
        '''
        simulates the effect of a low pass filter.
        arguments:
        f_max: maximum allowed frequency
        '''
        lp=np.where((self.fX**2+self.fY**2)<=f_max**2,1,0)
        self.data*=lp

    def four_f_system(self,f,f_max):
        '''
        simulates the effect of a 4f system
        arguments:
        f: float, focal length of the lens
        f_max: float, cut-off frequency of the low pass filter
        returns: None
        '''
        self.Lens(f)
        self.low_pass(f_max)
        self.Lens(f)
```

→ **Step-5:** ***Normalizing the data:***

```
    def normalize(self):
        '''
        normalizes the image data. Maps the data range to 0-1.
        arguments: None
        returns: None
        '''
        self.data=(self.data-np.min(self.data))/(np.max(self.data)-np.min(self.data))
```

→ **Step-6:** ***Intended Project work:***

1. Simulation of angled illumination and plotting the result (Along x-axis).

```
## initializing values
f=10e-3;f_max=2e4  # f is focal length and f_max is maximum frequency
                   # allowed in fourier aperture.

## creating the object
img=fourier_optics('graytestimg.png','grayscale',550,1.6) # initializing fourier_optics
                                                          # object


## preprocessing data
img.normalize()
img.add_padding(128)

sin_theta_range=np.linspace(0,np.sin(7.5*np.pi/180),5) # we are selecting sin(theta) range
                                                       # instead of theta range so that
ang_range=np.arcsin(sin_theta_range)*180/np.pi         # different angled illumination are
                                                       # equidistant in fourier domain.
for ang in ang_range:
    plt.figure(figsize=(15,5),layout='constrained')

    img1=img.deep_copy() # we are creating a deep copy of data for this particular
                         # illumination so that original data is not modified.
    img1.angled_illumination_phi(ang,0)
    img1.Lens(f)
    img1.remove_padding(128)
    plt.subplot(1,3,1)
```

```
plt.imshow(abs(img1.image_data()/np.max(img1.image_data()))**0.4,cmap='gray')
plt.axis(False),plt.title('Fourier Plane Input'),plt.colorbar()
img1.add_padding(128)

img1.low_pass(f_max)
img1.remove_padding(128)
plt.subplot(1,3,2)
plt.imshow(abs(img1.image_data()/np.max(img1.image_data()))**0.4,cmap='gray')
plt.axis(False),plt.title('Fourier Plane Output'),plt.colorbar()
img1.add_padding(128)

img1.Lens(f)
img1.remove_padding(128)
img1.data=img1.data/np.max(img1.data)
img1=abs(img1.image_data())
plt.subplot(1,3,3),plt.imshow(img1,cmap='gray'),plt.axis(False)
plt.title('Output through 4-F system'),plt.colorbar()

plt.suptitle(f'For $\\theta$={np.round(ang,2)}° , $\phi$=0°',
             fontsize=25,fontweight=20)
plt.show()
```

2. Repeating the algorithm for different orientations of angled illumination:
   (a) Along y-axis;

```
## initializing values
f=10e-3;f_max=2e4

## creating the object
img=fourier_optics('graytestimg.png','grayscale',550,1.6)
## preprocessing data
img.normalize()
img.add_padding(128)

sin_theta_range=np.linspace(0,np.sin(7.5*np.pi/180),5)

ang_range=np.arcsin(sin_theta_range)*180/np.pi

for ang in ang_range:
    plt.figure(figsize=(15,5),layout='constrained')

    img1=img.deep_copy()
    img1.angled_illumination_phi(ang,90)
    img1.Lens(f)
    img1.remove_padding(128)
    plt.subplot(1,3,1)
    plt.imshow(abs(img1.image_data()/np.max(img1.image_data()))**0.4,cmap='gray')
    plt.axis(False),plt.title('Fourier Plane Input'),plt.colorbar()
    img1.add_padding(128)

    img1.low_pass(f_max)
    img1.remove_padding(128)
    plt.subplot(1,3,2)
    plt.imshow(abs(img1.image_data()/np.max(img1.image_data()))**0.4,cmap='gray')
    plt.axis(False),plt.title('Fourier Plane Output'),plt.colorbar()
    img1.add_padding(128)

    img1.Lens(f)
    img1.remove_padding(128)
    img1.data=img1.data/np.max(img1.data)
    img1=abs(img1.image_data())
    plt.subplot(1,3,3),plt.imshow(img1,cmap='gray'),plt.axis(False)
    plt.title('Output through 4-F system'),plt.colorbar()
```

```
        plt.suptitle(f'For $\\theta$={np.round(ang,2)}° , $\phi$=90° ',
                     fontsize=25,fontweight=20)
        plt.show()
```

(b) Along 45° wrt x-axis;

```
## initializing values
f=10e-3;f_max=2e4

## creating the object
img=fourier_optics('graytestimg.png','grayscale',550,1.6)

## preprocessing data
img.normalize()
img.add_padding(128)

sin_theta_range=np.linspace(0,np.sin(7.5*np.pi/180),5)

ang_range=np.arcsin(sin_theta_range)*180/np.pi

for ang in ang_range:
    plt.figure(figsize=(15,5),layout='constrained')

    img1=img.deep_copy()
    img1.angled_illumination_phi(ang,45)
    img1.Lens(f)
    img1.remove_padding(128)
    plt.subplot(1,3,1)
    plt.imshow(abs(img1.image_data()/np.max(img1.image_data()))**0.4,cmap='gray')
    plt.axis(False),plt.title('Fourier Plane Input'),plt.colorbar()
    img1.add_padding(128)

    img1.low_pass(f_max)
    img1.remove_padding(128)
    plt.subplot(1,3,2)
    plt.imshow(abs(img1.image_data()/np.max(img1.image_data()))**0.4,cmap='gray')
    plt.axis(False),plt.title('Fourier Plane Output'),plt.colorbar()
    img1.add_padding(128)

    img1.Lens(f)
    img1.remove_padding(128)
    img1.data=img1.data/np.max(img1.data)
    img1=abs(img1.image_data())
    plt.subplot(1,3,3),plt.imshow(img1,cmap='gray'),plt.axis(False)
    plt.title('Output through 4-F system'),plt.colorbar()

    plt.suptitle(f'For $\\theta$={np.round(ang,2)}° , $\phi$=45°',
                 fontsize=25, fontweight=20)
    plt.show()
```

→ **Step-7:** ***Extended Project work- Fourier Ptychography:***

1. Defining the features of the Ptychography algorithm.

```
def ptychography(file_name:str,ld: float,p: float,focal_l:float,max_frequency:float,
                 mode:str,multiple_of_max_freq:int =3,plot_graph:bool=True):
    '''
    This function performs the ptychography algorithm in order to increase the resolution
    of the image.

    Arguments:-

    file_name:              name of the image file on which ptychography has to be
                            demostrated
    ld:                     wavelength of illuminating light(in nm)
```

```
p:                      pixel size (in micrometer)
focal_l:                focal length (in m)
max_frequency:          maximum frequency allowed in fourier plane. It simulates the
                        effect of low numerical aperture.
multiple_of_max_freq:   Maximum frequency you wish to capture devided by max_frequency
plot_graph:             Whether you want to plot the results or not.
'''

def fft(data):
    return fftshift(fft2(ifftshift(data)))

def ifft(data):
    return fftshift(ifft2(ifftshift(data)))

# prepocess data
data_padded=fourier_optics(file_name,mode=mode,ld=ld,p=p)
data_padded.add_padding(128)
data_padded.normalize()
new_data=data_padded.data

# initializing data required for processing later.
delta=np.zeros(np.shape(data_padded.fX))
N=np.shape(delta)[0]
f_data_final=np.zeros(np.shape(delta),dtype=complex)
d_f=data_padded.fX[0,1]-data_padded.fX[0,0]
loc_of_1_f=(0.5*max_frequency/d_f)

# simulating angled illumination and capturing data using an imaging system-4f system
for i in np.arange(-multiple_of_max_freq,multiple_of_max_freq+1):
    for j in np.arange(-multiple_of_max_freq,multiple_of_max_freq+1):

        '''*************************************************************************'''
        '''calculating specifications of angled illumination'''
        ## calculating theta
        r=np.sqrt(i**2+j**2)
        theta=np.arcsin(0.5*r*max_frequency*ld*1e-9)*180/np.pi

        ## calculating phi
        if i==0:
            if j==0:
                phi=0
            elif j>0:
                phi=90
            else:
                phi=-90

        else:
            phi=np.arctan(j/i)*180/np.pi
            if i<0 and j<0:
                phi+=180
            if i<0 and j>0:
                phi+=180
            if j==0 and i<0:
                phi=180
        '''*************************************************************************'''

        ## simulating capturing the data
        angled_illumination=fourier_optics(new_data,'grayscale',ld,p)
        angled_illumination.angled_illumination_phi(theta,phi)
        angled_illumination.four_f_system(focal_l,max_frequency)

        ## post-processing the data obtained from the imaging system
```

```
        f_data=angled_illumination.data
        f_data=fft(f_data)
        lp=np.where((data_padded.fX**2+data_padded.fY**2)<=(max_frequency**2),1,0)
        f_data*=lp

        ''' superposing /stiching data together corresponding to different
            illumination in fourier domain.
        We do so by by moving the data centered at origin to (i,j)(if origin is
        at (0,0) ). This done by convolving the data with a delta function at (i,j).
        '''
        i_loc=int(np.round(i*loc_of_1_f,1))
        j_loc=int(np.round(j*loc_of_1_f,1))
        delta[N//2-1+j_loc,N//2-1+i_loc]=1 #creating a delta function from zero matrix
        f_data=scipy.signal.fftconvolve(f_data,delta,mode='same')
        delta[N//2-1+j_loc,N//2-1+i_loc]=0 #reseting the delta function to zero matrix
        ## To avoid adding overlapping area twice.
        multiply_array=np.where(np.abs(f_data_final)<=np.abs(f_data),1,0)

        f_data_final+=(f_data*multiply_array) #Finally superposing
```

2. Plotting the results:

```
if plot_graph:
    plt.figure(figsize=(20,20))

    ## Ptychography
    plt.subplot(222)
    plt.title("Fourier domain with ptychography")
    plot_data=f_data_final
    plot_data=np.abs(plot_data)
    plot_data=((plot_data-np.min(plot_data))/(np.max(plot_data)-np.min(plot_data)))
    plt.imshow(plot_data**0.4,cmap='gray')
    plt.colorbar()

    plt.subplot(224)
    plt.title("Image with ptychography")
    plot_data=ifft(f_data_final)
    plot_data=np.abs(plot_data)
    plot_data=((plot_data-np.min(plot_data))/(np.max(plot_data)-np.min(plot_data)))
    plt.imshow(plot_data[-129:127:-1,-129:127:-1],cmap='gray')
    plt.colorbar()

    ## for demostrating improvement
    plt.subplot(223)
    plt.title("Image without angled \n illumination & without ptychography")
    compare_data=fourier_optics(new_data,mode=mode,ld=ld,p=p)
    compare_data.four_f_system(focal_l,max_frequency)
    plot_data=compare_data.data
    plot_data=np.abs(plot_data)
    plot_data=((plot_data-np.min(plot_data))/(np.max(plot_data)-np.min(plot_data)))
    plt.imshow(plot_data[-129:127:-1,-129:127:-1],cmap='gray')
    plt.colorbar()

    plt.subplot(221)
    plot_data=fft(compare_data.data)
    plt.title("Fourier domain without angled \n illumination & without ptychography")
    plot_data=np.abs(plot_data)
    plot_data=((plot_data-np.min(plot_data))/(np.max(plot_data)-np.min(plot_data)))
    plt.imshow(plot_data**0.4,cmap='gray')
    plt.colorbar()

    plt.suptitle("Fourier Ptychography",fontsize=25,fontweight=20)
    plt.show()
```

```
        return f_data_final
```

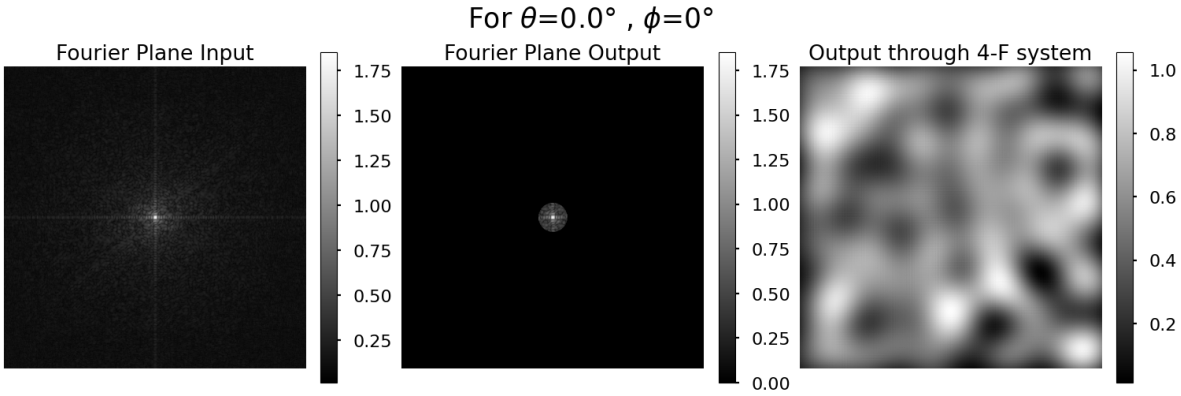
→ **Step-8:** ***Testing the algorithm on our test image:***

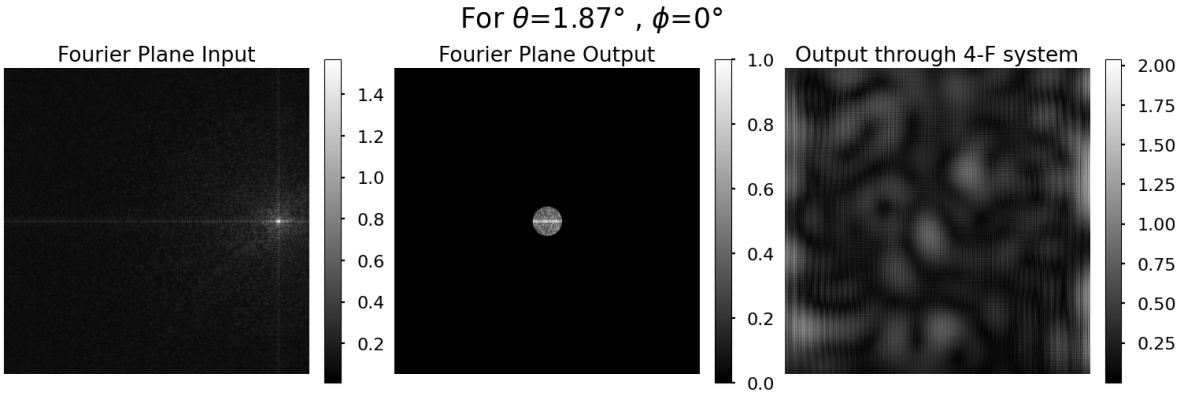
```
## initializing specifications
focal_l=10e-3;max_frequency=2e4;ld=550;p=1.6   #ld is in nm and p is in micron

## ptychography
high_res_micro=ptychography('microsample.jpg',ld,p,focal_l,max_frequency,'grayscale',5)
```

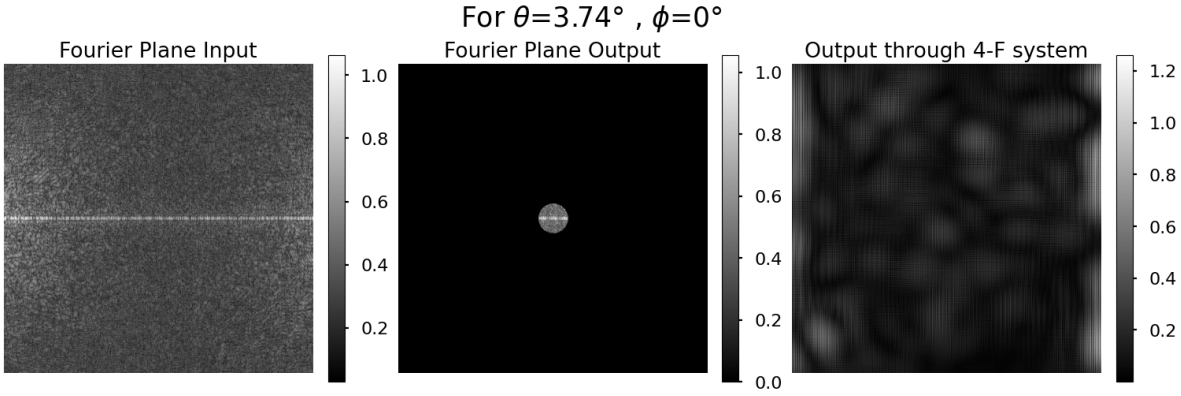
5 Results and Discussion

In this section, we present the plots of our findings and try to verify all that we have stated before.

5.1 Multiple Angled-Illumination:







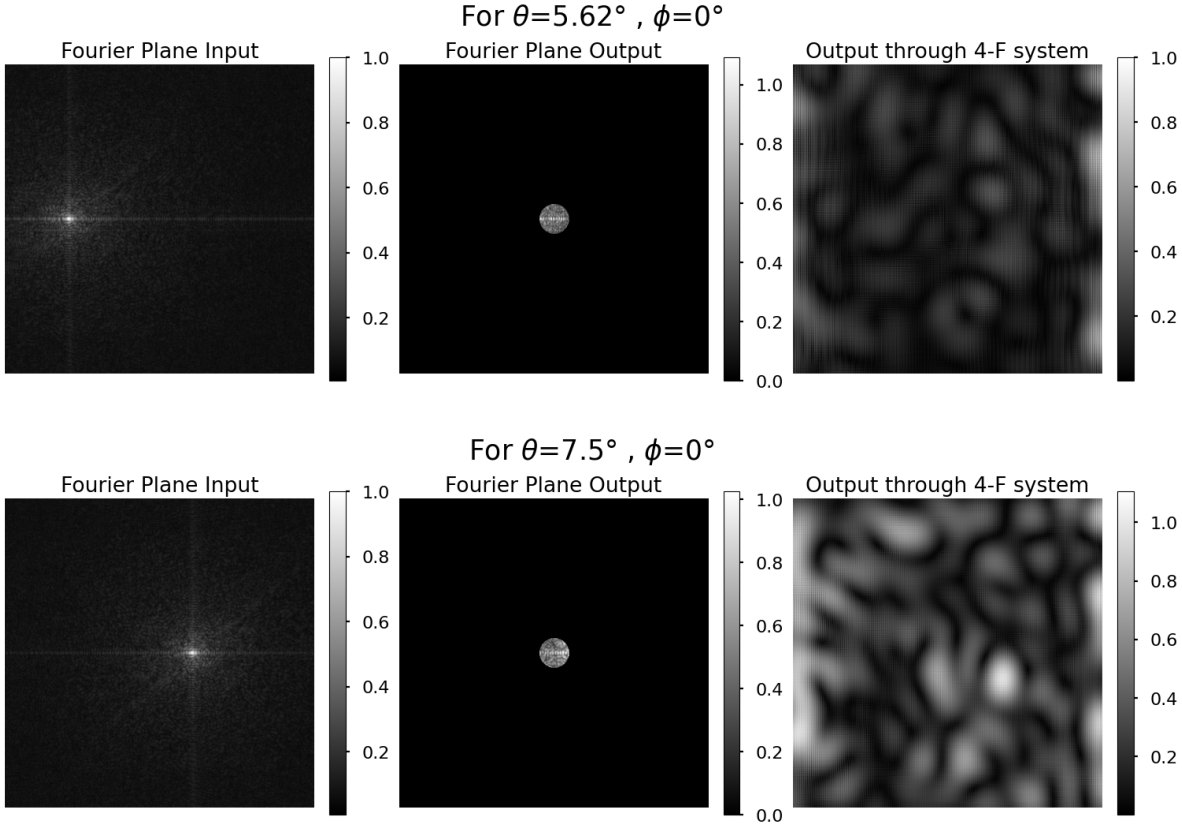
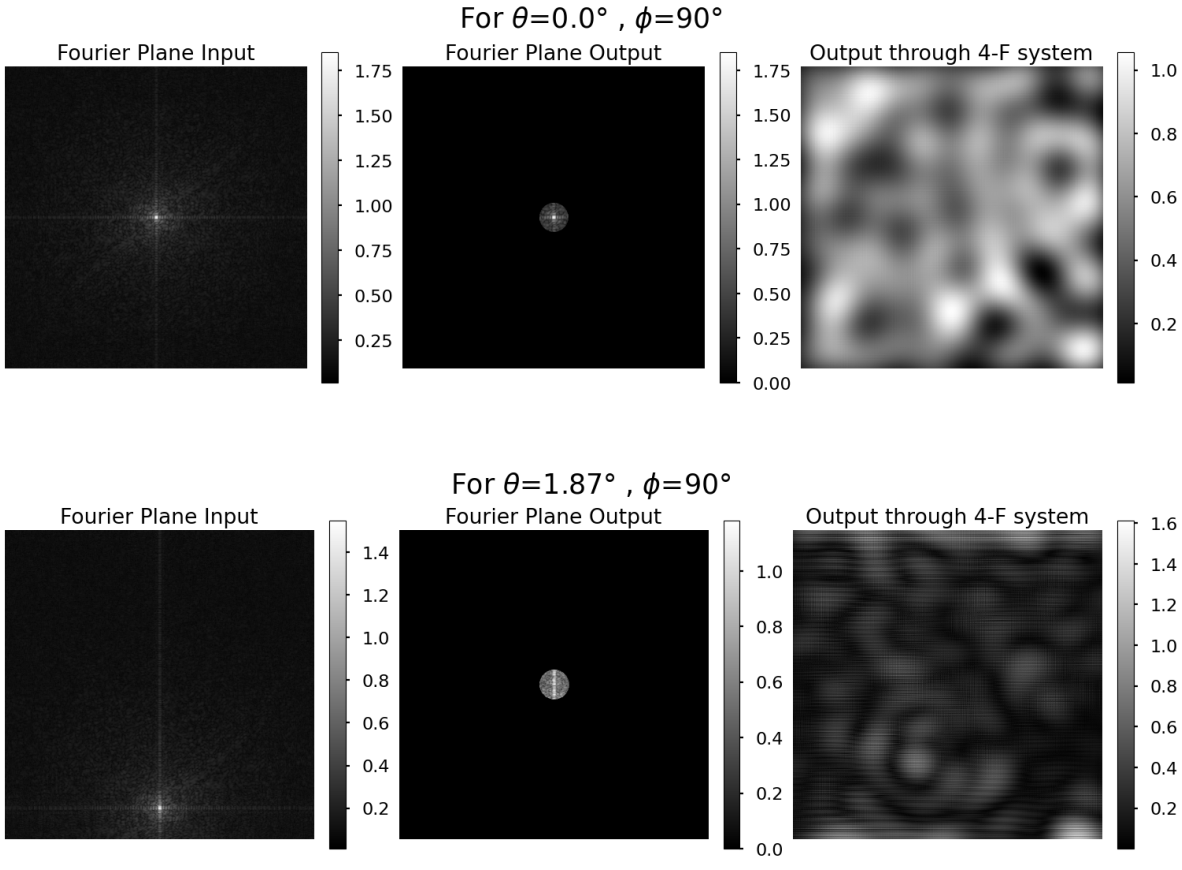


Figure 7: Using angled-illumination along x-axis



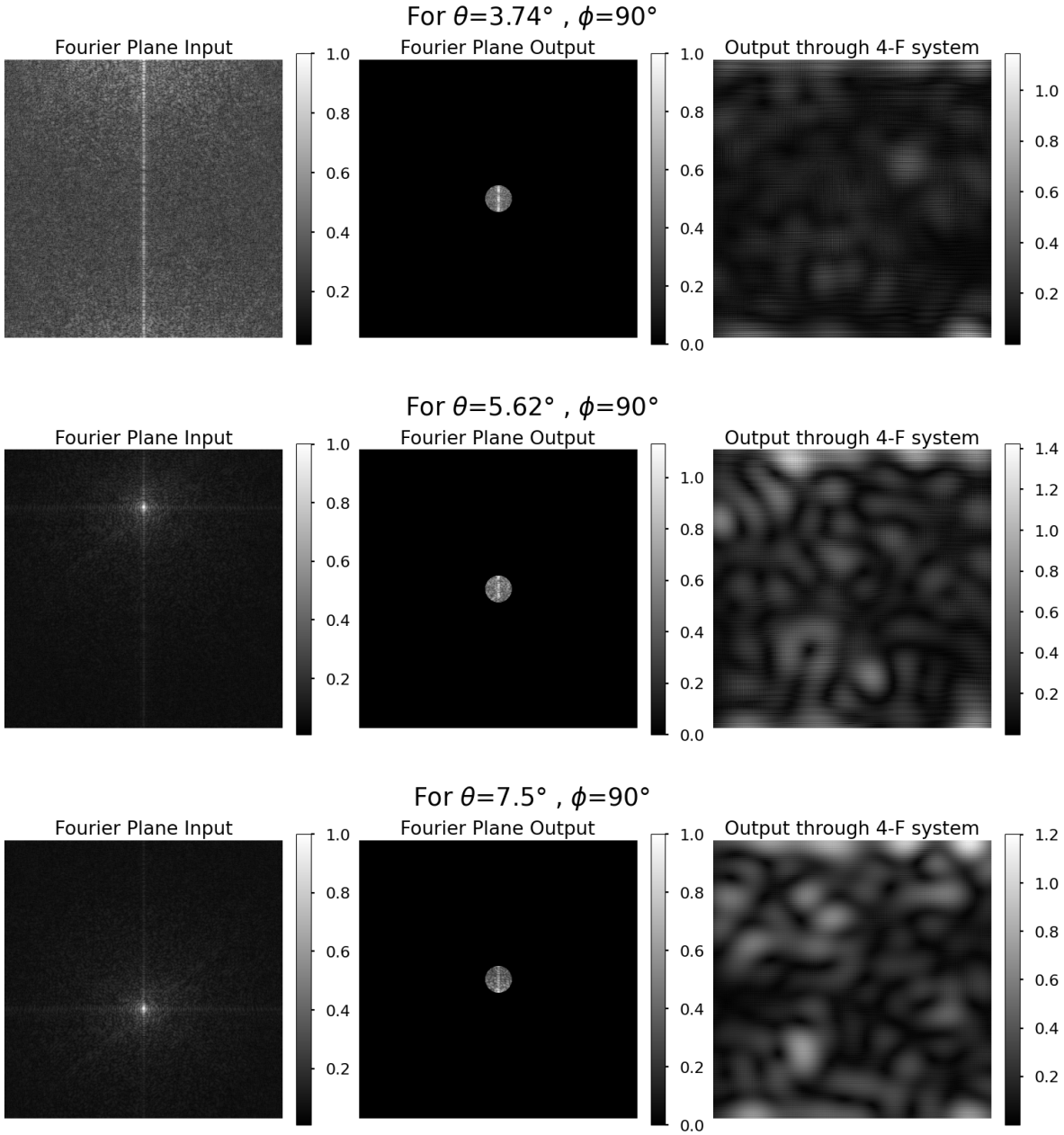
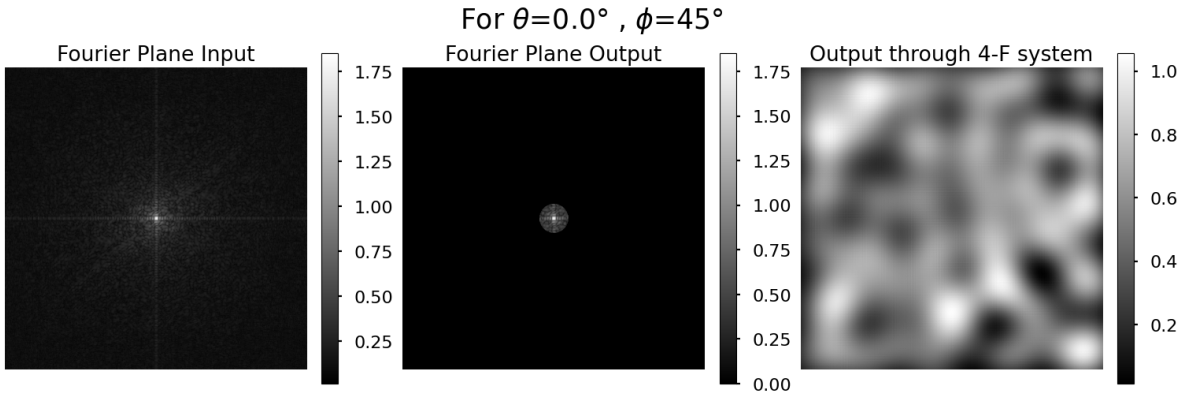


Figure 8: Using angled-illumination along y-axis



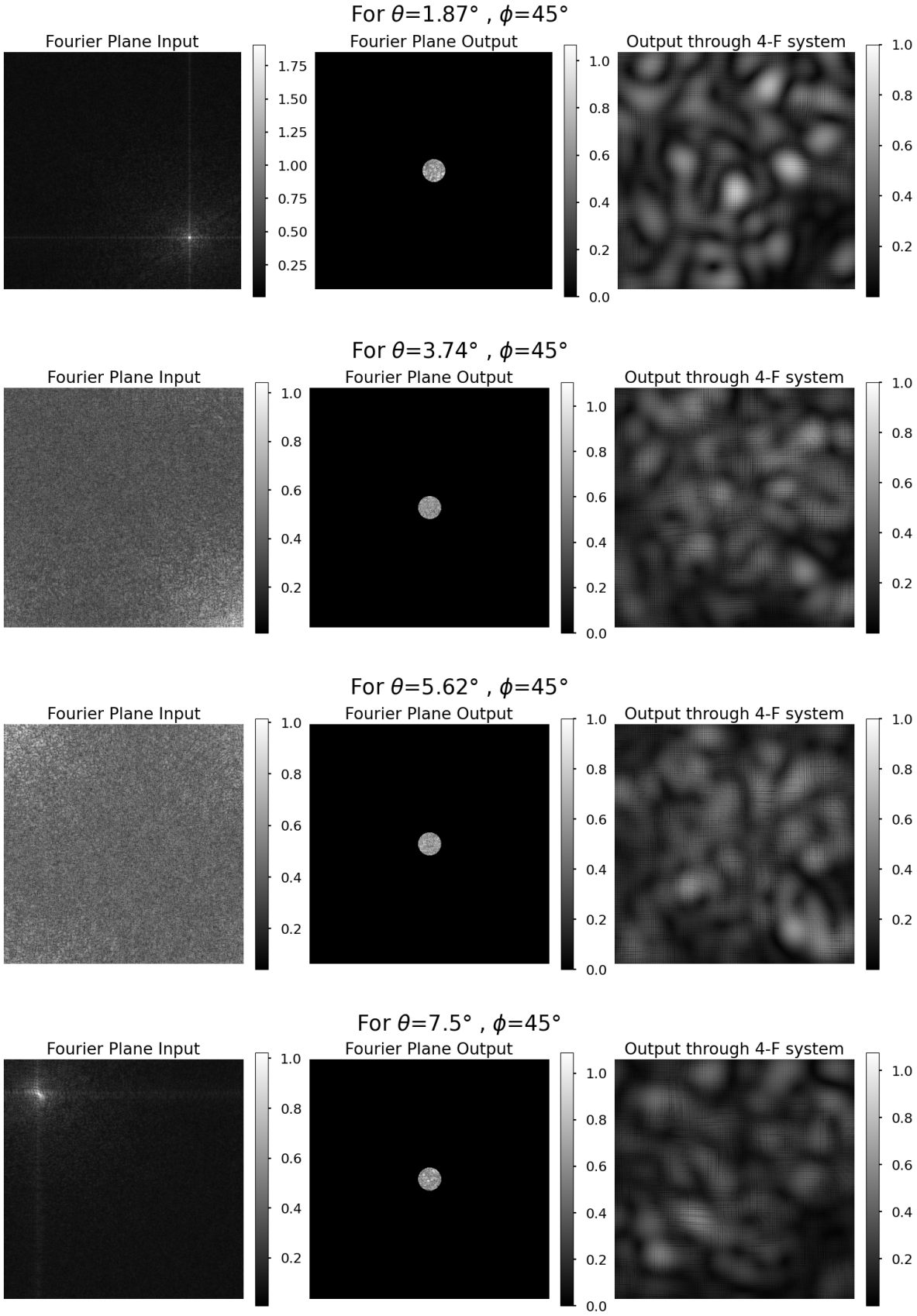


Figure 9: Using angled-illumination along xy-axis

As we discussed before, the frequency shifting property of the FT means that on illuminating the image transmittance function with different angles of illumination, i.e., with a tilted plane wave of different orientations, the FT of the image shifts in the Fourier domain, depending upon the angle and orientation. In the plots above, we can see that by taking 3 different orientations ($\phi = 0°, 90°$ and $45°$) and 5 different angles (θ), we see how the FT simply shifts along the respective axis/plane, by an

amount calculated as per the angle. This, in turn, effects the output images, as can be seen from the plots. Simply, different angles of illumination captures, to put it crudely, 'different perspectives' of the same image. And the idea of ptychography lies in the observation that these shifted Fourier Transforms can be stitched together into one field, to virtually yeild a bigger numerical aperture.

5.2 Fourier Ptychography Implementation:

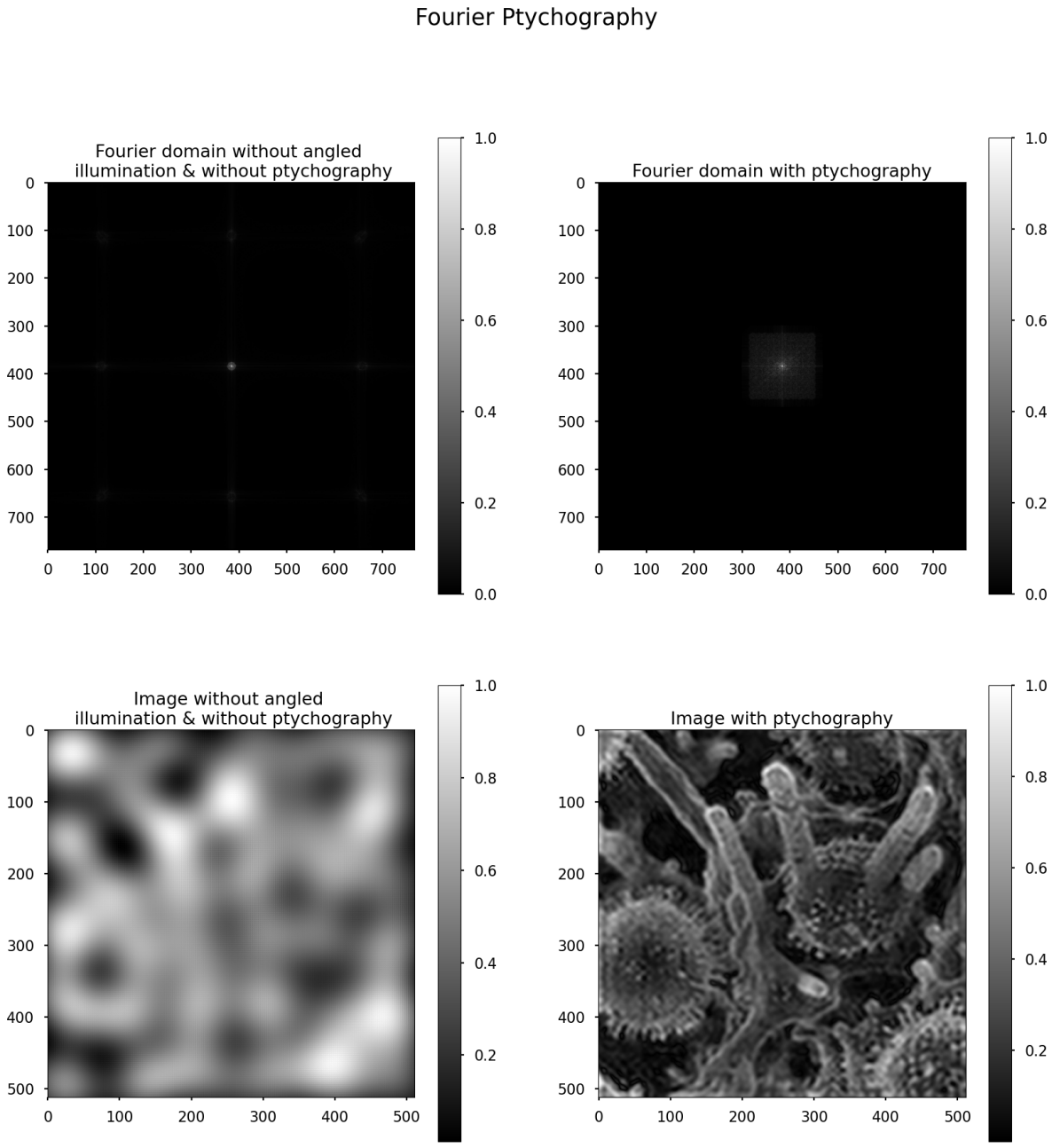


Figure 10: Our Ptychography Algorithm on a Bacterial Specimen

Following the general idea as given in Section. 2.4, we see that several shifted FTs when stitched together, as described in our algorithm, result in a virtually bigger numerical aperture and consequently a higher resolution image. While some high frequency noise is captured in the process, the difference in clarity of the 2 images, without and with Ptychography, is of a large significance. By working on effective overlapping and with further processing techniques, these noise components can be modulated further in order to enhance the quality of the image. And finally, the reason for choosing the bacteria image is because it highlights one of the areas of application of such an algorithm.

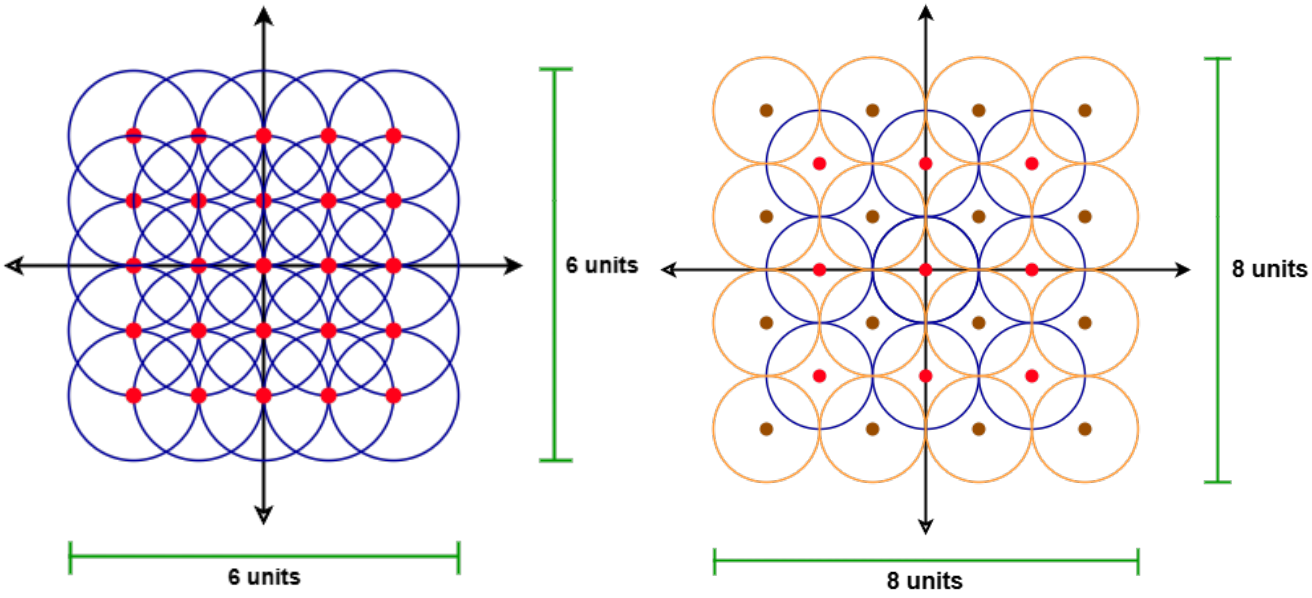
6 Novelty and Possible Improvements

As discussed before, Ptychography is not just one algorithm but rather a class of algorithms designed to achieve a similar purpose. Many popular authors and computational researchers have proposed several different methods of improving the resolution of an image by multiple illumination angles. However, the algorithm we have worked upon in this paper has 2 advantages over popular methods which gives it a touch of novelty. These are:

1. This is not a *convergence* algorithm. It will work for all types of images each and every time.
2. Since it is not a convergence algorithm, it does not require as much *computational power* as many other available alternatives.

However, every algorithm has rooms for improvement, and so does our. We noticed a few technicalities, which when improved upon, can make our algorithm even more efficient and effective. These are:

1. Selecting points in a way that overlap is less but coverage is complete (in order to make the algorithm even more computationally efficient). If we take a look at Fig. 11, both algorithms, with a similar *cost* of 25 circles, cover different areas. While our implementation covers only 36 sq. units, a better overlapping method covers 64 sq. units, which is a huge computational improvement.
2. Making the code compatible with RGB images further improves the range of applicability of our algorithm.



(a) Our Implementation (b) Another Improved Implementation

Figure 11: Possible improvement in implementation of Ptychography

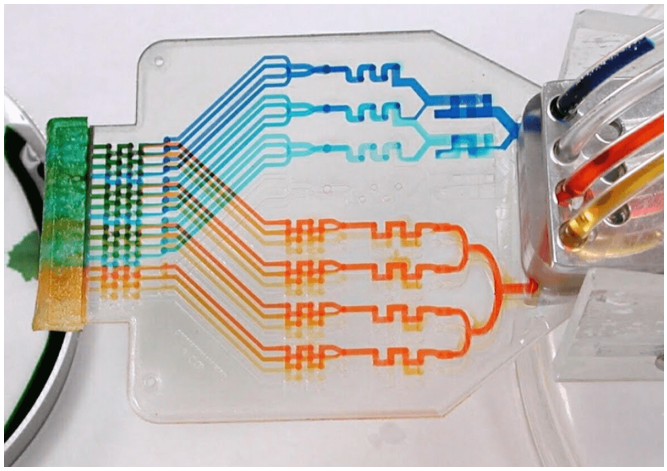
7 Applications

Fourier Ptychography (FP) and the concept of multiple angled-illuminiation is an idea of immense importance in various contexts of Computational Imaging. Its applications are ever-growing and wide-ranging. Some examples of which have been discussed as below[3]:

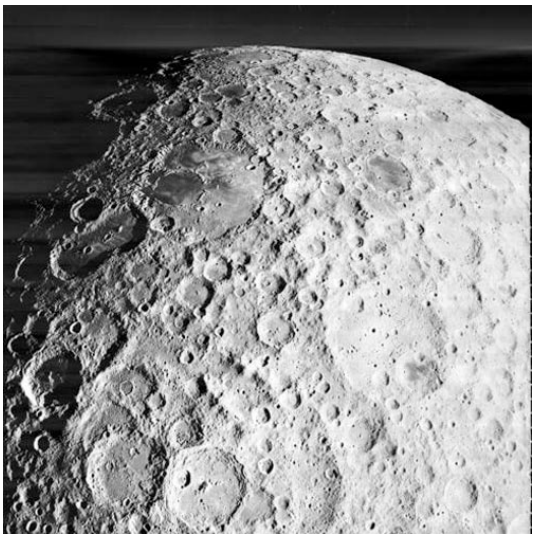
1. ***Semiconductor wafer inspection:*** FP's quantitative phase recovery of FP points to its potential to determine the surface topography of silicon wafers or other highly reflective surfaces that do not exhibit much contrast otherwise, for example to assist in defect detection.
2. ***Microfluidics:*** FP can replace holographic imaging methods used to monitor microfluidics, potentially offering the ability to image macroscopic areas at sufficient spatial and temporal resolutions in the future.
3. ***Phase Reconstruction Application:*** The ability of FP to quantitatively measure the phase with a non-interferometric setup makes it highly advantageous in phase imaging. This makes FP

important in fields like biomedical imaging, wherein many specimens are primarily transparent, as well as in metrology, to measure the topography of 3D surfaces.

4. ***Astrophysical Imaging:*** Imaging of celestial objects, such as that of the Moon, can be greatly benefitted by Fourier ptychography. We know that the Moon is lit only by the Sun, and while for a given elemental section of the moon the angle of illumination is constant, as the Moon changes its phase, we get different angles of illumination. This helps to build an enhanced image for the particular section of the moon.



(a) Microfluidics



(b) Lunar Imaging (Image Credit: Lunar and Planetary Institute)

Figure 12: Some Applications of Ptychography

8 Conclusion

Understanding optical phenomena and explanation of wave effects in terms of the Fourier transform and its properties marked the beginning of a revolutionary change in the field of Imaging. While Fourier optics became the new trend in optics, the growth of computational algorithms and advancement in technology meant that conducting experiments was not only limited to laboratories and high-end equipment. The simple observation relating diffraction, for example, to a slightly modified version of the Fourier transform allowed scientists to use computer systems to model the diffraction patterns for various optically interesting functions. Many years into the future, we have algorithms like Fourier ptychography, with immense applicability and importance in diverse fields, like metrology, biomedical imaging, semiconductor engineering, etc. Within the scope of this paper, an honest attempt has been made to explore and further build upon the idea of ptychography and the use of multiple angled-illumination to improve upon the resolution. The results obtained along with the algorithm used to compute them have been discussed in detail in the paper. Finally, some important applications have also been provided to develop a more holistic understanding of the topic.

References

[1] Joseph W Goodman. *Introduction to Fourier optics*. Roberts and Company publishers, 2005.

[2] Kedar Khare, Mansi Butola, and Sunaina Rajora. *Fourier optics and computational imaging*. Springer, 2015.

[3] Pavan Chandra Konda et al. "Fourier ptychography: current applications and future promises". In: *Optics express* 28.7 (2020), pp. 9603–9630.

[4] Atul Kumar Razdan and V Ravichandran. *Fundamentals of partial differential equations*. Springer, 2022.