

Department of Electronics and Electrical Communication  
Engineering

Indian Institute of Technology Kharagpur

---

Digital Image and Video Processing Lab  
(EC69211)



Experiment No: 3

**Title:** Frequency Domain Transformation

**Date of submission:** 28.08.2024

**Name:** Unnati Singh

**Roll No.:** 21EC39027

**Instructors:**

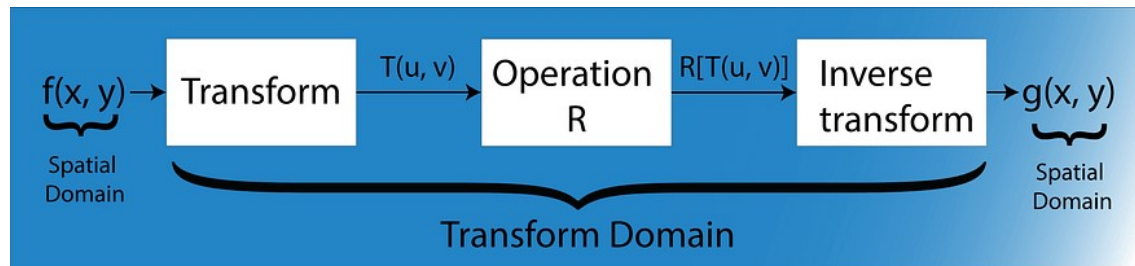
Prof. Saumik Bhattacharya

Prof. Prabir Kumar Biswas

## Introduction:

In mathematics, physics, electronics, control systems engineering, and statistics, the frequency domain refers to the analysis of mathematical functions or signals with respect to frequency (and possibly phase), rather than time, as in time series or spatial domain for images.

Image Transformation mainly follows three steps:



An image can be represented in the form of a 2D matrix where each element of the matrix represents pixel intensity. This state of 2D matrices that depict the intensity distribution of an image is called Spatial Domain. For the RGB image, the spatial domain is represented as a 3D vector of 2D matrices.

In frequency-domain methods are based on Fourier Transform of an image. Roughly, the term frequency in an image talks about the rate of change of pixel values.

Many times, image processing tasks are best performed in a domain other than the spatial domain. Moreover, it is easy to detect some features in a particular domain, i.e., a new information can be obtained in other domains.

## Key Functions in code:

### 1. `FFT2D(image)`:

- **Purpose:** The function computes the 2D Fast Fourier Transform (FFT) of an input image and visualizes its magnitude and phase spectra, which are crucial for understanding the image's frequency components.
- **How It Works:**
  - **Input Handling:** The function handles inputs as a list of pixel values, a file path to an image, or a 2D NumPy array, converting them into a suitable 2D NumPy array format.
  - **1D FFT Computation:** A recursive 1D FFT function is applied to each row of the image array.
  - **2D FFT Computation:** The 1D FFT results from the rows are then used to apply the FFT to each column, yielding the 2D FFT of the image.

- **Magnitude and Phase Calculation:** The magnitude and phase spectra of the 2D FFT are computed using the absolute value and angle of the complex numbers.
- **FFT Shift:** The spectra are shifted so that the zero-frequency component is centred, improving visualization.
- **Visualization:** The magnitude and phase spectra are plotted, providing insight into the image's frequency content.


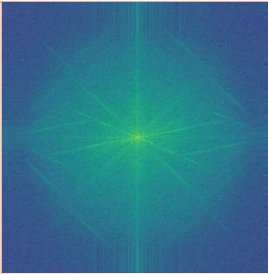
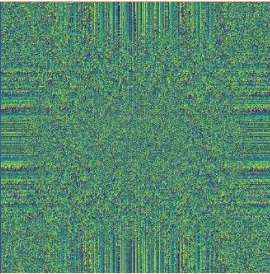


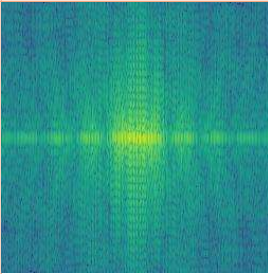
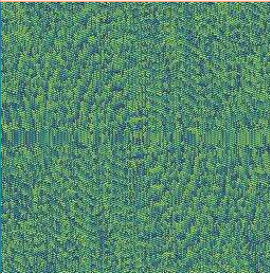


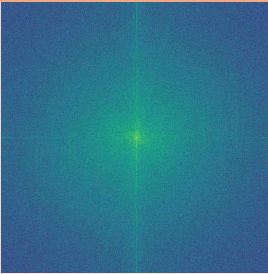
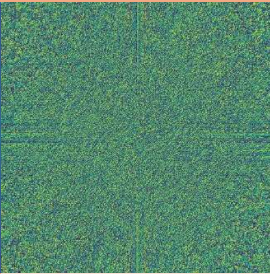

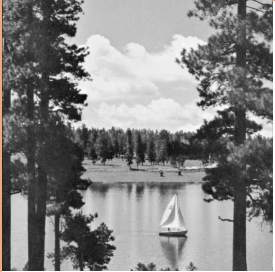
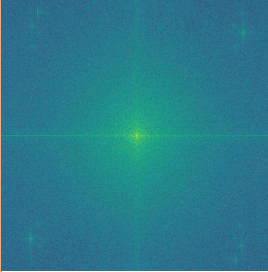
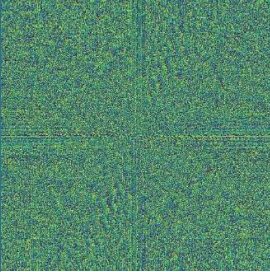
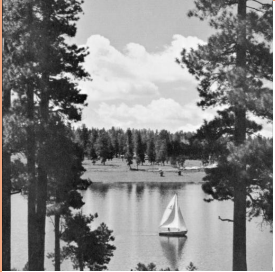
## 2. `inv_FFT2D(fft_image):`

- **Purpose:** The function computes the inverse 2D Fast Fourier Transform (iFFT) to reconstruct the original image from its frequency domain representation.
- **How It Works:**
  - **Inverse 1D FFT Computation:** A recursive inverse FFT function (`inverse_FFT`) is applied to each row of the 2D FFT image.
  - **Inverse 2D FFT Computation:** The results of the inverse 1D FFTs from the rows are used to apply the inverse FFT to each column, reconstructing the original 2D image.
  - **Normalization:** The final result is normalized by dividing by the length of the input, ensuring accurate reconstruction of the image.

## 3. `process_image(image_path):`

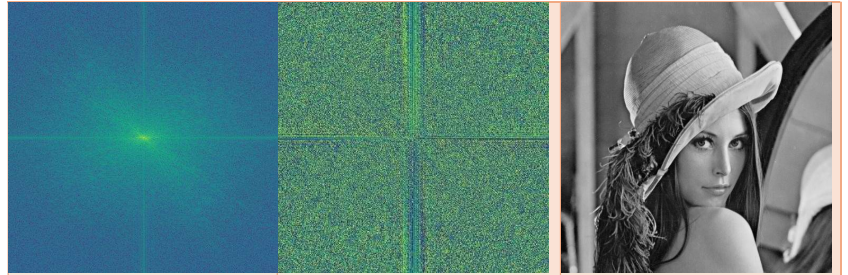
- **Purpose:** The function processes an image by applying Fourier-based transformations, modifying its frequency domain, and then reconstructing and displaying the resulting image as asked in the question.
- **How It Works:**
  - **Image Loading and Grayscale Conversion:** The image is loaded from the specified path and converted to a grayscale NumPy array.
  - **Centering the Image:** The pixel values are multiplied by  $(-1)^{x+y}$  to shift the image's frequency components, a common step to center the zero-frequency component before applying the Fourier Transform.
  - **2D FFT Application:** The centered image undergoes a 2D Fast Fourier Transform (using the `FFT2D` function), converting it to the frequency domain.
  - **Complex Conjugate:** The complex conjugate of the frequency domain representation is computed, which is used for further processing.
  - **Inverse 2D FFT:** The inverse FFT (`inv_FFT2D`) is applied to the conjugated frequency data to bring it back to the spatial domain.
  - **Decentering the Image:** The resultant image is multiplied by  $(-1)^{x+y}$  again to reverse the initial centering step.
  - **Displaying the Image:** The final processed image is displayed using `matplotlib`, showing the effect of the transformations applied.

Output Images:

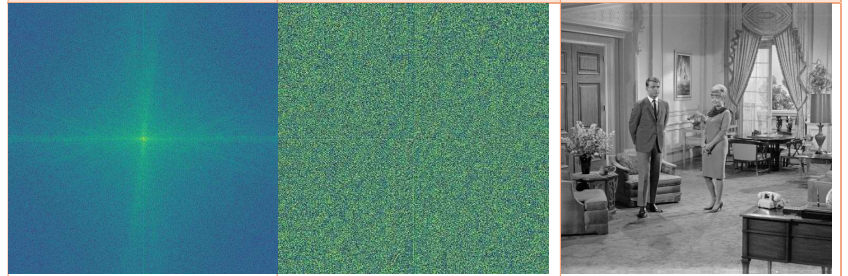
	<i>Image</i>	<i>Magnitude Spectrum</i>	<i>Phase Spectrum</i>	<i>Reconstructed image from IFFT</i>
1. cameraman.jpg				
2. dip.tif				
3. jetplane.jpg				
4. lake.jpg				



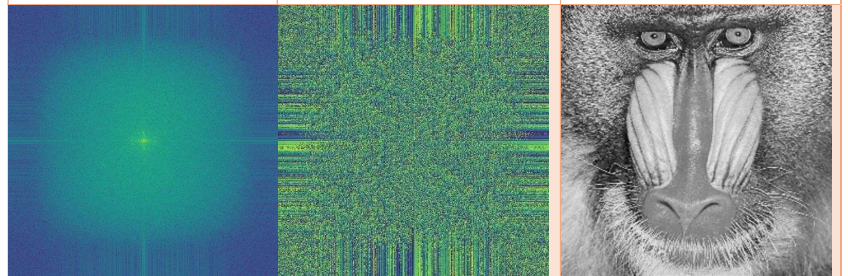
5. lena\_gray\_512.jpg



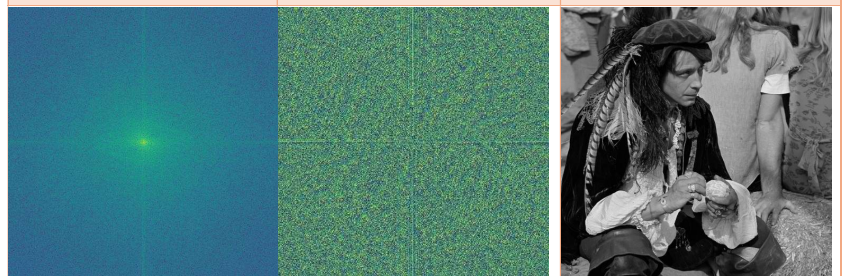
6. livingroom.jpg



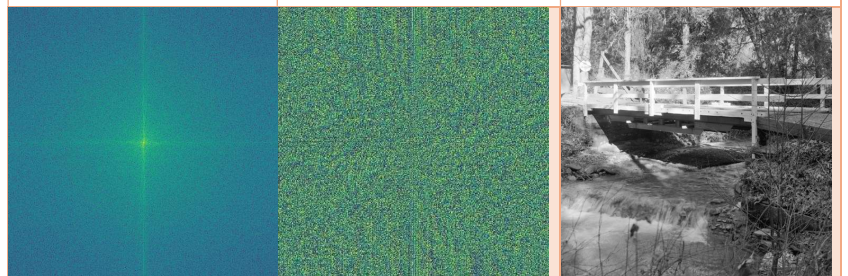
7. mandril\_gray.jpg



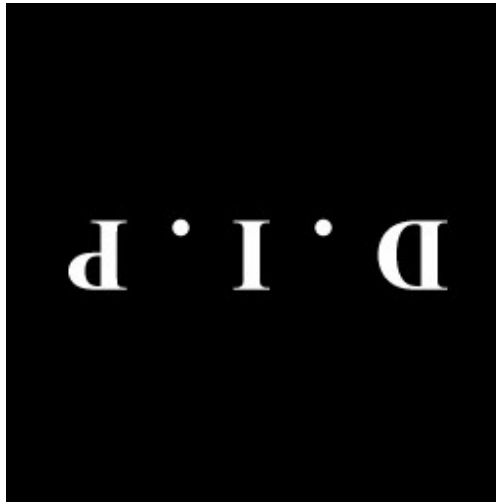
8. pirate.jpg



9. walkbridge.jpg



Additionally, we also had to perform a few specified transformations on the image ``dip.tif`` and the resulting image is as follows:



The complex conjugate of the transform helps calculate the symmetrical values of the DFT values thus the flipped version of the image in frequency domain. Hence, the inverse Fourier transform gives us a flipped image.

### **Discussion:**

The primary goal of this experiment was to understand and apply frequency domain transformations to image data, specifically through the use of the Fast Fourier Transform (FFT) and its inverse. The key steps involved transforming an image from the spatial domain to the frequency domain, modifying it, and then reconstructing the image to analyse the effects of these transformations.

The FFT transforms an image into its frequency components, with the magnitude spectrum showing the strength of various frequencies and the phase spectrum indicating spatial arrangements. High-frequency components, linked to edges, are located at the periphery, while low-frequency components, related to smooth areas, are centered. Centering the image in the frequency domain by applying  $\begin{bmatrix} (-1)^{x+y} \end{bmatrix}$  simplifies analysis by placing low frequencies in the center. Decentering by reapplying  $(-1)^{x+y}$  after processing ensures accurate spatial reconstruction during inverse FFT, preserving the image's original spatial coherence.

### **Q1. Fast Fourier Transform (FFT):**

The `FFT2D` function computes the 2D FFT of an image using a recursive implementation. It handles input as either a file path, numpy array, or list of pixel values. It visualizes and saves the magnitude and phase spectra of the image.

## Q2. Image Processing Steps:

1. **Multiplication by  $(-1)^{x+y}$** : This step is used to shift the zero-frequency component to the center of the image, which helps in better visualization of the frequency components.
2. **FFT Calculation**: Computes the frequency domain representation of the image.
3. **Complex Conjugate**: The complex conjugate of the transform helps calculate the symmetrical values of the DFT values thus the flipped version of the image in frequency domain. Hence, the inverse Fourier transform gives us a flipped image.
4. **Inverse FFT**: Reconstructs the spatial domain image from the modified frequency domain representation.
5. **Multiplication by  $(-1)^{x+y}$** : Re-applies the earlier shift to restore the original spatial domain characteristics.