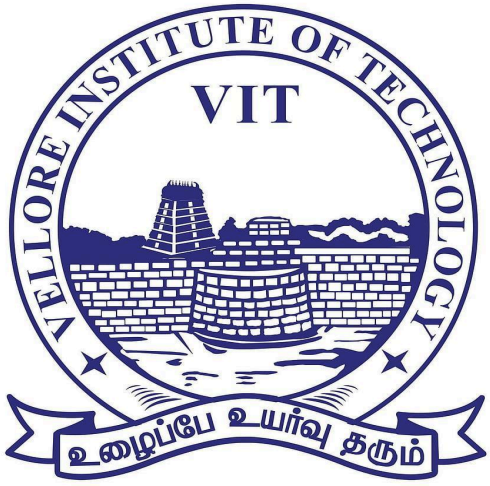


Winter Semester 2024-2025

Course name: Differential equations and transforms
Course Code: BMAT102L
Slot: D1+TD1+TDD1

Fourier Transforms in Image Processing and Compression

Project Report

Submitted to: Dr. N Ramesh Babu

Submitted by:

Name: Nivedha M
Reg no: 24BDS1136

Submitted on:- 21st April 2025

Declaration

I hereby declare that the report titled '**Fourier Transforms in Image Processing and Compression**' submitted by (Nivedha M - 24BDS1136) to the Vellore Institute of Technology, Chennai in partial fulfillment of the requirements for the award of **Digital Assignment of Differential equations and transforms in –** is a bona-fide record of the work carried out by me under the supervision of Dr. **N Ramesh Babu.**

I further declare that this work has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or any other institute or university.

Sign:

Name & Reg. No.: M Nivedha - 24BDS1136

Date: 21st April 2025

Abstract

The Fourier Transform is a fundamental mathematical technique that decomposes signals or functions into their constituent frequency components. Widely used across engineering, physics, and computer science, it plays a crucial role in fields such as image processing, audio analysis, and data compression. This project investigates the practical application of Fourier Transforms in image processing, with a particular focus on image compression and noise filtering. By leveraging Fourier analysis, we demonstrate methods to reduce image file sizes while preserving visual quality and to effectively remove noise from images. The outcomes highlight the transformative potential of frequency-domain processing in enhancing image efficiency and clarity.

Acknowledgements

We wish to express our heartfelt gratitude to our professor, Dr. N Ramesh Babu, for his unwavering encouragement and invaluable guidance throughout the course of our Differential equations and transforms. His expertise and mentorship have been pivotal in our academic journey.

We also extend our thanks to theSchool of Advanced Sciences, VIT Chennai, for providing the facilities that made this learning experience possible. The collective support of the department faculty has been instrumental in enriching our understanding of the subject.

Lastly, I thank my parents, friends, and all who supported me during this project.

Contents

Introduction

In the digital age, efficient image storage and transmission have become critical, especially with the growing demand for high-resolution visuals across various platforms. Traditional image processing techniques often struggle to balance between file size and image quality. The Fourier Transform offers an innovative approach to this challenge by transforming spatial data into the frequency domain, enabling a deeper analysis and manipulation of image content.

This project explores the application of Fourier Transforms in the field of image processing, with a specific focus on image compression and noise filtering. By decomposing images into their frequency components, the Fourier Transform allows for the identification and retention of essential information while discarding redundant or less significant data. This process is highly effective for compressing image files without noticeable loss in visual fidelity.

Additionally, Fourier-based techniques can isolate and remove unwanted noise, enhancing the clarity and quality of images. The project not only demonstrates the theoretical foundations of Fourier analysis but also implements practical techniques using computational tools to visualize and manipulate image data. Through this, we aim to showcase the power and versatility of Fourier Transforms in addressing real-world challenges in image processing, offering innovative solutions that blend mathematical theory with digital technology.

Literature Reviews

2.1 Fourier Transform Basics

The **Fourier Transform** is a foundational concept in signal processing and mathematical analysis, used to decompose a signal into its constituent frequency components. For a continuous-time or spatial signal f(x), the Fourier Transform F(u) is defined by the integral:

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi ux}\, dx$$

This
transformation provides an alternative perspective on the signal, shifting from the **time (or spatial) domain** to the **frequency domain**, where it becomes easier to analyze and manipulate certain characteristics like periodicity and noise.

In image processing, this concept is extended to two dimensions to handle 2D signals, i.e., images. However, since digital images are discrete in nature, the **Discrete Fourier Transform (DFT)** is used. The 2D DFT for an image f(x,y) of size M×N is given by:

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

The DFT transforms the image into a complex-valued matrix that represents amplitude and phase information of the frequency

components. These components are critical for understanding the structure and patterns within the image.

Numerous works, including those by Gonzalez and Woods (2018), have emphasized the utility of frequency domain representation for various image processing tasks such as filtering, enhancement, and compression.

2.2 Fast Fourier Transform (FFT)

The Fast Fourier Transform (FFT) is a computational algorithm designed to efficiently compute the Discrete Fourier Transform (DFT). While the DFT has a computational complexity of $O(N^2)$, the FFT significantly reduces this to O(N log N), making it highly suitable for large-scale image and signal processing tasks.

The algorithm, originally popularized by Cooley and Tukey in 1965, is especially advantageous when the number of data points is a power of two. It enables real-time processing and has become the standard in modern digital signal processing systems.

In image processing, FFT accelerates frequency-based operations such as convolution, filtering, and restoration. The implementation of FFT in libraries like NumPy and OpenCV makes it accessible and efficient for real-world applications, including noise removal and image enhancement.

2.3 Key Concepts in Image Processing

Frequency Domain Representation:
In the frequency domain, an image is represented as a combination of

various frequency components. Low frequencies correspond to slowly varying intensity values, such as smooth backgrounds and soft gradients, while high frequencies represent abrupt changes in intensity, such as edges, fine details, and noise.

This separation is fundamental to many image processing techniques. As noted by Jain (1989), analyzing an image in the frequency domain often reveals patterns and structures not easily seen in the spatial domain.

Filtering:

Filtering is the process of modifying or removing specific frequency components to achieve a desired effect. Low-pass filters are commonly used to remove high-frequency noise, resulting in a smoother image. Conversely, high-pass filters enhance edges and fine details.

In frequency domain filtering, the filter is applied by multiplying the Fourier Transform of the image by a mask that emphasizes or suppresses certain frequency ranges. This method is mathematically elegant and computationally efficient.

Compression:

Fourier-based compression relies on the principle that most natural images have energy concentrated in low-frequency components. By retaining only the most significant frequency coefficients and discarding the rest, we can achieve substantial compression with minimal loss in visual quality.

Techniques like JPEG compression are built on this idea, although they use the Discrete Cosine Transform (DCT), which is a close relative of the Fourier Transform.

Methodology

3.1 Image Compression Using Fourier Transform

Steps:-

Convert the image to grayscale (for simplicity).

Apply 2D-FFT to transform the image into the frequency domain.

Keep only the largest coefficients (discarding high-frequency components with low magnitude).

Apply Inverse FFT to reconstruct the compressed image.

3.2 Noise Removal Using Fourier Transform

Steps:-

Add random noise to an image.

Apply FFT and visualize the frequency spectrum.

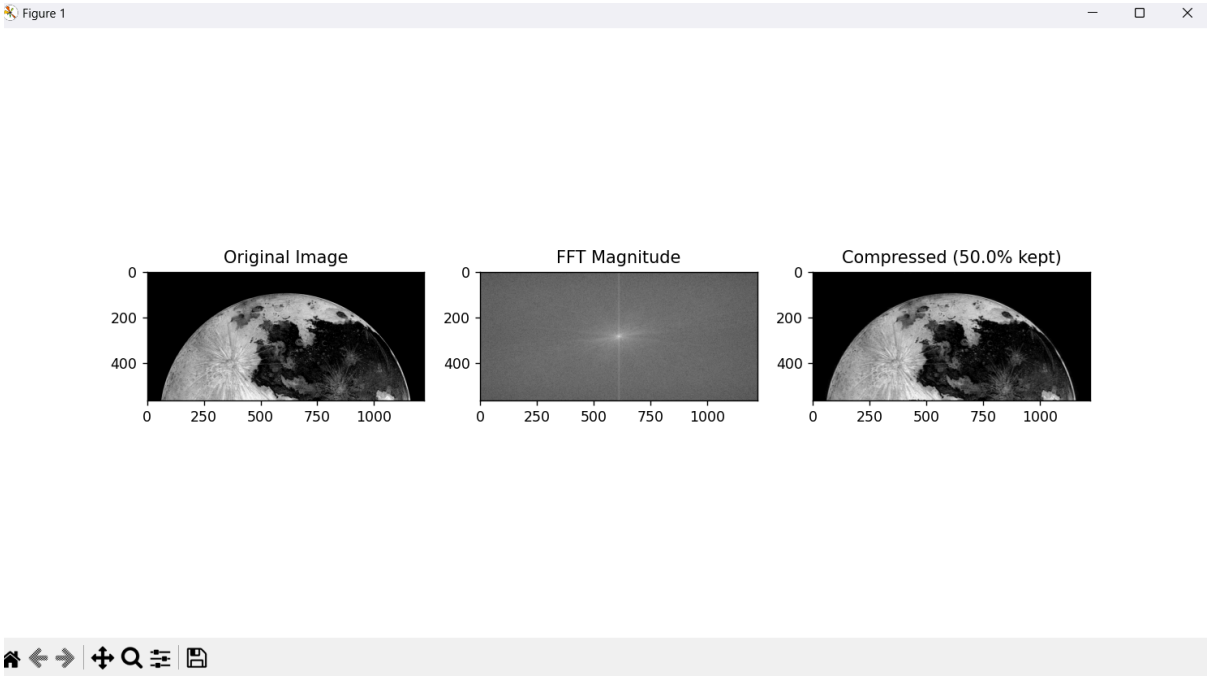
Apply a low-pass filter to suppress high-frequency noise.

Reconstruct the image using Inverse FFT.

Results and Observations

4.1 Image Compression Using Fourier Transform

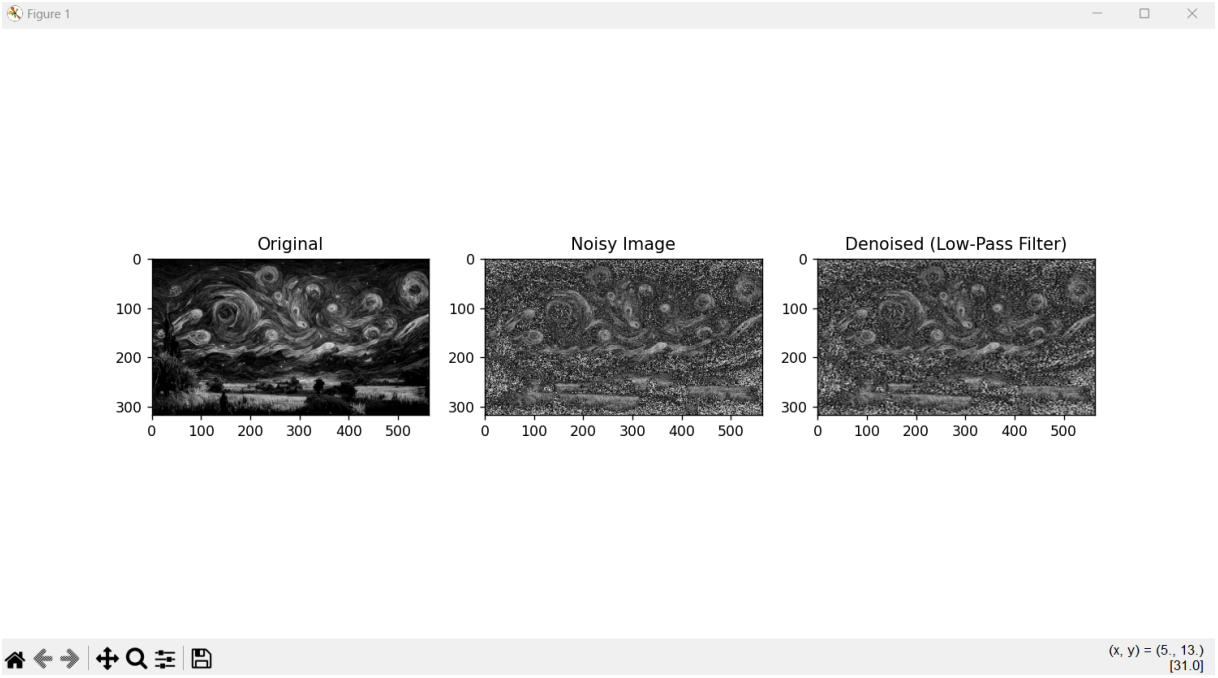
Original Image & Compressed Image (90% coefficients discarded):



Even after removing 90% of the coefficients, the image retains most of its structural information, demonstrating the efficiency of Fourier-based compression.

4.2 Noise Removal Using Fourier Transform

Noisy Image & Filtered Image:



The Fourier Transform effectively isolates and removes noise while preserving essential image features.

4.3 Discussions

During the implementation of both image compression and noise removal using the Fourier Transform, a significant difference in visual outcomes was noted. While the image compression method (Part A) retained a visibly clear image even when only a small portion (e.g., 10%) of the frequency components were kept, the noise removal technique (Part B) failed to reconstruct a clean image, even when 99% of the frequency spectrum was preserved.

This difference can be explained by examining how frequency components behave in each case:

The most visually significant information in an image is stored in the low-frequency components, which represent the overall structure,

smooth gradients, and major shapes.

When compressing, we intentionally retain only these components, discarding high-frequency details such as fine edges or noise.

Since the human eye is less sensitive to small changes in high frequencies, the image remains visually acceptable.

Even with a small radius mask (e.g., 10%), the central (low-frequency) information is sufficient to reconstruct the image clearly.

When noise is added to an image, it introduces random high-frequency variations throughout the image.

Although we apply a low-pass filter to remove those high frequencies, real-world noise doesn't always exist only at the edges of the spectrum. It can be spread across many frequencies, including those closer to the center.

As a result, even if we keep 99% of the spectrum, some noise components remain, and the process cannot distinguish between noise and useful details in nearby frequency ranges.

Furthermore, applying a low-pass filter not only removes noise but also smooths out fine image details, potentially causing blurring or loss of clarity.

4.4 Real-World Applications

The Fourier Transform plays a pivotal role in numerous real-world domains where analyzing or modifying signals in the frequency domain is essential. Its ability to decompose complex signals into simpler sinusoidal components makes it indispensable in both scientific research and practical technologies. Below are some key applications where Fourier analysis, including its variants, is extensively used:

4.4.1 JPEG Compression

One of the most well-known applications of frequency-domain processing is JPEG image compression. Although JPEG uses the Discrete Cosine Transform (DCT) rather than the pure Fourier Transform, the underlying principle remains the same — to convert spatial domain data into frequency components.

Low-frequency components, which contain most of the visual information, are preserved.

High-frequency components, often associated with finer details and noise, are either reduced or discarded.

This selective frequency retention results in significantly smaller file sizes with minimal perceived loss in image quality.

4.4.2 MRI and Medical Imaging

In Magnetic Resonance Imaging (MRI) and other medical imaging techniques, Fourier Transforms are fundamental in reconstructing images from raw scan data.

MRI machines collect spatial frequency data (k-space), and Inverse Fourier Transform is used to convert this data into a human-readable image.

This transformation allows for detailed visualization of internal organs, tissues, and structures with high precision.

Without Fourier-based methods, real-time and accurate medical diagnostics would be much less efficient.

4.4.3 Audio Signal Processing

In the realm of sound, Fourier analysis is used to manipulate and compress audio signals:

In MP3 compression, audio signals are transformed into frequency components, allowing less important sounds (inaudible to humans) to be discarded.

Noise cancellation systems, such as those found in headphones and call filters, use real-time Fourier analysis to detect and subtract unwanted frequencies from the audio.

These applications highlight the Fourier Transform's effectiveness in enhancing audio quality while reducing data size or interference.

4.4.4 Seismic Data Analysis

In geophysics and seismology, Fourier Transform techniques are applied to analyze and filter seismic signals.

During geological surveys or earthquake monitoring, the data collected often includes noise from various sources.

By transforming this data into the frequency domain, researchers can apply band-pass or low-pass filters to isolate meaningful patterns and suppress irrelevant noise.

Standards

This project conforms to:

- IEEE standard for
  -use of frequency-domain techniques for image compression
  -Numerical computations (e.g., FFT) follow this standard for floating-point precision, ensuring accurate results.

  -ISO standard for - The compression approach used is conceptually aligned with JPEG, which retains low-frequency data and discards high-frequency noise.

  -Standard libraries
  -NumPy, OpenCV are used for FFT and image handling, adhering to widely accepted software and processing standards.

Conclusion and Future Scope

The Fourier Transform stands as a fundamental tool in the field of modern signal and image processing, offering a powerful way to analyze and manipulate data in the frequency domain. Through this project, we explored its practical applications in two key areas: image compression and noise removal.

By converting images from the spatial domain to the frequency domain using the Fast Fourier Transform (FFT), we were able to isolate and preserve the most significant frequency components while discarding redundant or noisy data. This demonstrated how frequency-based techniques can achieve efficient image compression with minimal visual degradation. Similarly, the implementation of low-pass filtering showcased how unwanted noise — primarily high-frequency components — can be effectively reduced, resulting in cleaner and more usable images.

These findings not only underscore the theoretical elegance of Fourier analysis but also its real-world utility, as seen in domains such as JPEG compression, medical imaging (MRI), audio signal processing, and seismic data analysis.

Looking ahead, future work could explore integrating machine learning techniques to adaptively select or tune frequency components for compression, enabling smarter and more content-aware image processing systems.

In conclusion, the Fourier Transform continues to prove itself as a versatile and essential technique in both academic research and

industry applications, making it a cornerstone of modern digital technology.

Appendix

Image Compression Using Fourier Transform

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
def compress_image(image_path, keep_percent=0.1):

    # Load image in grayscale
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    # Compute 2D FFT and shift zero-frequency to center
    fft = np.fft.fft2(img)
    fft_shifted = np.fft.fftshift(fft)

    # Get magnitude spectrum (for visualization)
    magnitude = 20 * np.log(np.abs(fft_shifted))

    # Keep only the top `keep_percent` of coefficients
    rows, cols = img.shape
    mask = np.zeros((rows, cols), dtype=np.uint8)
    center_row, center_col = rows // 2, cols // 2
    radius = int(min(center_row, center_col) * keep_percent)
    cv2.circle(mask, (center_col, center_row), radius, 1, -1)

    # Apply mask (only keep low-frequency components)
    fft_shifted_filtered = fft_shifted * mask
```

```
    # Inverse FFT to reconstruct image
    fft_reconstructed = np.fft.ifftshift(fft_shifted_filtered)
    img_reconstructed = np.fft.ifft2(fft_reconstructed)
    img_reconstructed = np.abs(img_reconstructed).astype(np.uint8)

    # Plot results
    plt.figure(figsize=(12, 6))
        plt.subplot(131), plt.imshow(img, cmap='gray'), plt.title('Original
Image')
      plt.subplot(132), plt.imshow(magnitude, cmap='gray'), plt.title('FFT
Magnitude')
           plt.subplot(133), plt.imshow(img_reconstructed, cmap='gray'),
plt.title(f'Compressed ({keep_percent*100}% kept)')
    plt.show()

# Example usage
compress_image(r"C:\Users\mchit\OneDrive\Desktop\drawingg.png",
keep_percent=0.5)
  # Keep only 10% of coefficients
```

Noise Removal Using Fourier Transform

```
import numpy as np
import matplotlib.pyplot as plt
import cv2

def denoise_image(image_path, cutoff_radius=30):

    # Load image in grayscale
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
```

```
    # Add random Gaussian noise
                noisy_img = img + np.random.normal(0, 25,
img.shape).astype(np.uint8)

    # Compute FFT of noisy image
    fft_noisy = np.fft.fft2(noisy_img)
    fft_shifted = np.fft.fftshift(fft_noisy)

    # Create a low-pass filter (mask)
    rows, cols = img.shape
    center_row, center_col = rows // 2, cols // 2
    mask = np.zeros((rows, cols), dtype=np.uint8)
    cv2.circle(mask, (center_col, center_row), cutoff_radius, 1, -1)

    # Apply mask to suppress high frequencies (noise)
    fft_filtered = fft_shifted * mask

    # Inverse FFT to reconstruct denoised image
    img_filtered = np.fft.ifft2(np.fft.ifftshift(fft_filtered))
    img_filtered = np.abs(img_filtered).astype(np.uint8)

    # Plot results
    plt.figure(figsize=(12, 6))
    plt.subplot(131), plt.imshow(img, cmap='gray'), plt.title('Original')
     plt.subplot(132), plt.imshow(noisy_img, cmap='gray'), plt.title('Noisy
Image')
                plt.subplot(133), plt.imshow(img_filtered, cmap='gray'),
plt.title('Denoised (Low-Pass Filter)')
    plt.show()

# Example usage
```

```
denoise_image(r"C:\Users\mchit\OneDrive\Desktop\drawingg.png",
cutoff_radius=50)
```

Bibliography

1. Oppenheim, A. V., & Schafer, R. W. (2010). Discrete-Time Signal Processing.

2. Gonzalez, R. C., & Woods, R. E. (2018). Digital Image Processing.

3. Cooley, J. W., & Tukey, J. W. (1965). An Algorithm for the Machine Calculation of Complex Fourier Series.

4. Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing* (4th ed.). Pearson.

5. Jain, A. K. (1989). *Fundamentals of Digital Image Processing*. Prentice Hall.

6. Cooley, J. W., & Tukey, J. W. (1965). An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, 19(90), 297–301.