

# Two-Dimensional Discrete Fourier Transform: Implementation and Analysis without FFT

Vaibhav Sharma (202351154), Devyash Saini (202351030)

*Indian Institute of Information Technology, Vadodara (IIITV)*

**Abstract**—This report presents a comprehensive implementation of the Two-Dimensional Discrete Fourier Transform (2-D DFT) from first principles without using ready-made Fast Fourier Transform (FFT) libraries. The work demonstrates four fundamental aspects: (1) generation and visualization of an  $8 \times 8$  2-D DFT basis as a  $64 \times 64$  image grid, (2) creation of binary test images containing rectangular regions with user-defined parameters, (3) computation and visualization of 2-D DFT magnitude and phase spectra, and (4) analysis of frequency domain characteristics through image centering using the  $(-1)^{x+y}$  transformation. The implementation validates theoretical DFT principles and illustrates the relationship between spatial and frequency domain representations. Results demonstrate the computational feasibility of direct DFT calculation and its effectiveness in analyzing image frequency content.

## I. INTRODUCTION

The Discrete Fourier Transform (DFT) is a fundamental tool in digital image processing, providing a mathematical framework for analyzing images in the frequency domain. While Fast Fourier Transform (FFT) algorithms are commonly used for computational efficiency, understanding the underlying DFT formulation is essential for grasping frequency analysis concepts.

This laboratory work implements the 2-D DFT using direct computation based on the mathematical definition, deliberately avoiding optimized FFT implementations to reinforce fundamental understanding. The system explores DFT basis functions, applies transformation to synthetic binary images, and demonstrates the effect of spatial domain operations on frequency spectra.

### A. Objectives

The primary objectives of this work are:

- Generate and visualize the complete basis set of an  $8 \times 8$  2-D DFT
- Implement 2-D DFT computation without using ready-made FFT functions
- Analyze frequency domain characteristics of rectangular binary images
- Demonstrate frequency spectrum shifting through image centering

## II. THEORETICAL BACKGROUND

### A. Two-Dimensional DFT

The 2-D DFT transforms a spatial domain image  $f(x, y)$  of size  $M \times N$  into its frequency domain representation  $F(u, v)$ :

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

where  $u = 0, 1, \dots, M-1$  and  $v = 0, 1, \dots, N-1$  represent frequency indices.

Using Euler's formula:

$$e^{-j\theta} = \cos(\theta) - j \sin(\theta) \quad (2)$$

The DFT can be expressed as:

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) [\cos(\theta_{ux, vy}) - j \sin(\theta_{ux, vy})] \quad (3)$$

where:

$$\theta_{ux, vy} = 2\pi \left( \frac{ux}{M} + \frac{vy}{N} \right) \quad (4)$$

### B. DFT Basis Functions

The 2-D DFT can be viewed as a decomposition of the input image into a weighted sum of basis functions. Each basis function  $\phi_{u,v}(x, y)$  corresponds to a specific frequency  $(u, v)$ :

$$\phi_{u,v}(x, y) = e^{-j2\pi(\frac{ux}{M} + \frac{vy}{N})} \quad (5)$$

For an  $N \times N$  image, there are  $N^2$  basis functions forming a complete orthogonal basis.

### C. Magnitude and Phase Spectra

The DFT result is complex-valued and can be represented in polar form:

$$\text{Magnitude: } |F(u, v)| = \sqrt{\text{Re}^2[F(u, v)] + \text{Im}^2[F(u, v)]} \quad (6)$$

$$\text{Phase: } \angle F(u, v) = \arctan \left( \frac{\text{Im}[F(u, v)]}{\text{Re}[F(u, v)]} \right) \quad (7)$$

### D. Image Centering

Multiplying an image by  $(-1)^{x+y}$  in the spatial domain shifts the origin of the DFT to the center of the frequency plane:

$$f_c(x, y) = f(x, y) \cdot (-1)^{x+y} \quad (8)$$

This transformation moves the DC component (zero frequency) from the corners to the center, facilitating frequency analysis.

### III. METHODOLOGY

#### A. Task 1: DFT Basis Generation

---

**Algorithm 1** Generate 2-D DFT Basis Functions

---

**Require:** DFT size  $N$  (default: 8)  
**Ensure:** Basis function arrays (real and imaginary parts)

- 1: Initialize arrays:  $B_{\text{real}}[N^2][N][N]$ ,  $B_{\text{imag}}[N^2][N][N]$
- 2:  $\text{idx} \leftarrow 0$
- 3: **for**  $u = 0$  to  $N - 1$  **do**
- 4:   **for**  $v = 0$  to  $N - 1$  **do**
- 5:     **for**  $x = 0$  to  $N - 1$  **do**
- 6:       **for**  $y = 0$  to  $N - 1$  **do**
- 7:          $\theta \leftarrow -2\pi \left( \frac{ux}{N} + \frac{vy}{N} \right)$
- 8:          $B_{\text{real}}[\text{idx}][x][y] \leftarrow \cos(\theta)$
- 9:          $B_{\text{imag}}[\text{idx}][x][y] \leftarrow \sin(\theta)$
- 10:       **end for**
- 11:     **end for**
- 12:      $\text{idx} \leftarrow \text{idx} + 1$
- 13:   **end for**
- 14: **end for**
- 15: **return**  $B_{\text{real}}, B_{\text{imag}}$

---

1) *Algorithm:* The magnitude of each basis function is computed and arranged in an  $8 \times 8$  grid, producing a  $64 \times 64$  visualization image.

```
def generate_dft_basis_2d(N=8):
    basis_real = np.zeros((N * N, N, N))
    basis_imag = np.zeros((N * N, N, N))

    idx = 0
    for u in range(N):
        for v in range(N):
            for x in range(N):
                for y in range(N):
                    angle = -2 * np.pi * ((u * x / N) + (v * y / N))
                    basis_real[idx, x, y] = np.cos(angle)
                    basis_imag[idx, x, y] = np.sin(angle)
            idx += 1

    return basis_real, basis_imag
```

#### B. Task 2: Binary Rectangle Image Creation

A  $64 \times 64$  binary image is generated with a rectangular region set to 1 (white) and the background set to 0 (black). User inputs specify:

- Top-left corner position  $(r_{\text{top}}, c_{\text{left}})$
- Rectangle dimensions: width  $w$  and height  $h$

Input validation ensures:

$$0 \leq r_{\text{top}}, c_{\text{left}} < 64 \quad (9)$$

$$w, h > 0 \quad (10)$$

$$r_{\text{top}} + h \leq 64 \quad (11)$$

$$c_{\text{left}} + w \leq 64 \quad (12)$$

#### C. Task 3: 2-D DFT Computation

---

**Algorithm 2** Compute 2-D DFT

---

**Require:** Input image  $f[M][N]$   
**Ensure:** DFT coefficients  $F[M][N]$  (complex)

- 1: Initialize  $F[M][N] \leftarrow 0 + 0j$
- 2: **for**  $u = 0$  to  $M - 1$  **do**
- 3:   **for**  $v = 0$  to  $N - 1$  **do**
- 4:      $\text{sum} \leftarrow 0 + 0j$
- 5:     **for**  $x = 0$  to  $M - 1$  **do**
- 6:       **for**  $y = 0$  to  $N - 1$  **do**
- 7:          $\theta \leftarrow -2\pi \left( \frac{ux}{M} + \frac{vy}{N} \right)$
- 8:          $\text{sum} \leftarrow \text{sum} + f[x][y] \cdot (\cos \theta + j \sin \theta)$
- 9:       **end for**
- 10:     **end for**
- 11:      $F[u][v] \leftarrow \text{sum}$
- 12:   **end for**
- 13: **end for**
- 14: **return**  $F$

---

1) *Algorithm:*

```
def dft_2d(image):
    M, N = image.shape
    dft_result = np.zeros((M, N), dtype=complex)

    for u in range(M):
        for v in range(N):
            sum_val = 0.0 + 0.0j
            for x in range(M):
                for y in range(N):
                    angle = -2 * np.pi * ((u * x / M) + (v * y / N))
                    sum_val += image[x, y] * (
                        np.cos(angle) + 1j *
                        np.sin(angle))
            dft_result[u, v] = sum_val

    return dft_result
```

#### D. Task 4: Centered Image DFT

The image centering transformation is applied:

```
def center_image(image):
    M, N = image.shape
    centered_image = np.zeros((M, N))

    for x in range(M):
        for y in range(N):
            centered_image[x, y] = image[x, y] *
                ((-1) ** (x + y))

    return centered_image
```

The centered image undergoes 2-D DFT computation, and results are compared with the non-centered version.

#### IV. COMPUTATIONAL COMPLEXITY

The direct 2-D DFT implementation has computational complexity:

$$\mathcal{O}(M^2 N^2) \quad (13)$$

For a  $64 \times 64$  image:

- Number of DFT coefficients:  $64 \times 64 = 4,096$
- Operations per coefficient:  $64 \times 64 = 4,096$
- Total operations:  $\approx 16.8$  million

This demonstrates why FFT algorithms ( $\mathcal{O}(N^2 \log N)$ ) are preferred for practical applications, though direct computation remains valuable for educational purposes.

#### V. RESULTS AND DISCUSSION

##### A. DFT Basis Visualization

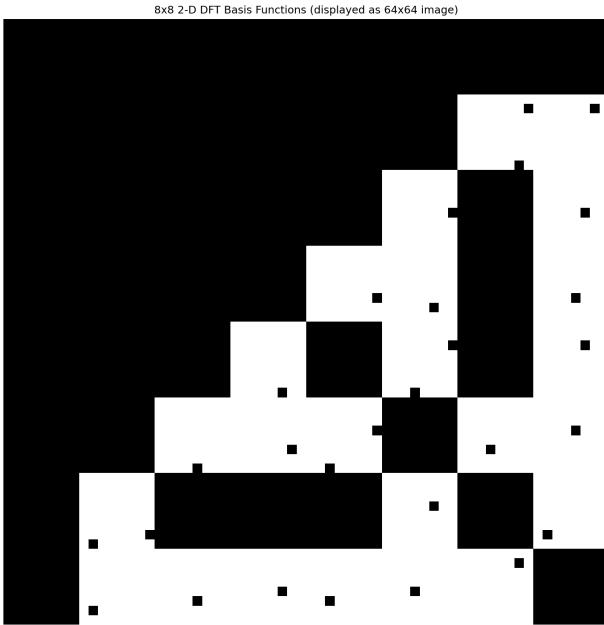
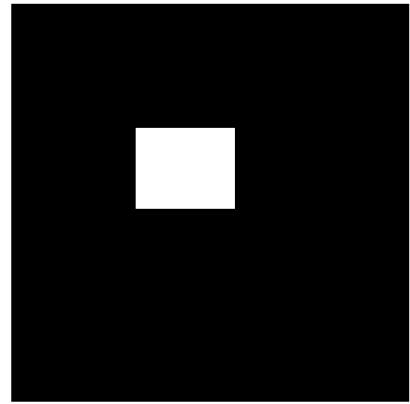


Fig. 1:  $8 \times 8$  2-D DFT basis functions displayed as a  $64 \times 64$  image. Each  $8 \times 8$  block represents one basis function with increasing spatial frequencies from top-left to bottom-right.

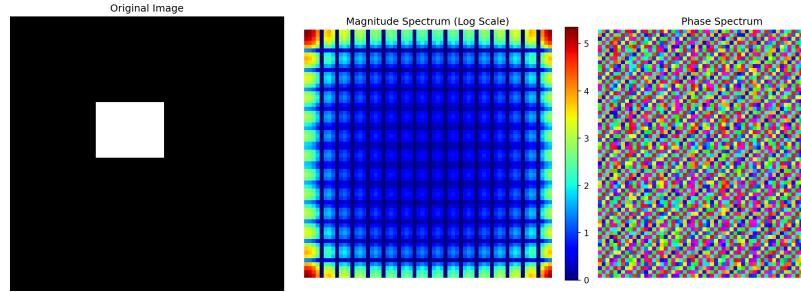
Figure 1 displays all 64 basis functions of an  $8 \times 8$  2-D DFT. Key observations:

- The top-left block  $(0,0)$  represents the DC component (uniform intensity)
- Horizontal frequency increases from left to right
- Vertical frequency increases from top to bottom
- Higher frequency basis functions exhibit more rapid oscillations

Binary Image with Rectangle



(a) Binary rectangle



(b) DFT analysis

Fig. 2: Binary rectangle image and its 2-D DFT magnitude/phase spectra.

##### B. Rectangle Image and DFT

The rectangle image (Figure 2) demonstrates:

- Strong DC component at corners (low frequency energy concentration)
- Magnitude spectrum shows symmetric patterns along horizontal and vertical axes
- Phase spectrum reveals directional information
- Sharp edges in spatial domain create high-frequency components

##### C. Centered Image Analysis

The centering transformation (Figure 3) produces:

- Checkerboard pattern in spatial domain due to  $(-1)^{x+y}$  multiplication
- DC component shifted to spectrum center in frequency domain
- Symmetric distribution of frequency energy around center
- Easier interpretation of low and high-frequency content

##### D. Comparative Analysis

Table I summarizes key differences between standard and centered DFT representations.

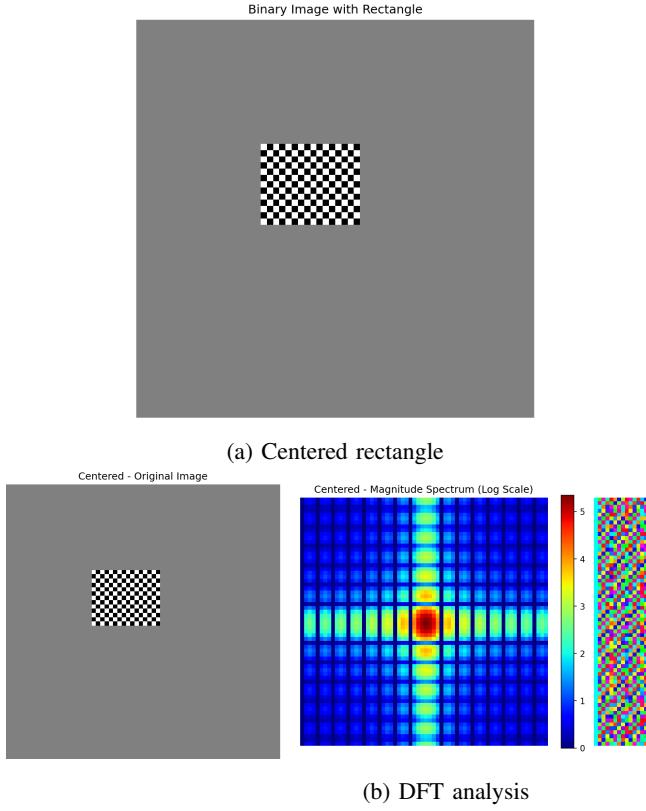


Fig. 3: Centered image (multiplied by  $(-1)^{x+y}$ ) and its 2-D DFT spectra.

TABLE I: Comparison of Standard vs. Centered DFT

Property	Standard DFT	Centered DFT
DC location	Corners	Center
Symmetry	4-fold	Radial
Low-freq region	Scattered	Concentrated
Visualization	Less intuitive	More intuitive
Computation	Same	Same

#### E. Spectrum Characteristics

For the rectangular binary image:

- **Magnitude spectrum:** Exhibits sinc-like patterns characteristic of rectangular functions in Fourier domain
- **Phase spectrum:** Contains discontinuities corresponding to edge locations
- **Symmetry:** Magnitude spectrum shows Hermitian symmetry due to real-valued input
- **Energy distribution:** Majority of energy concentrated in low frequencies

#### VI. VALIDATION

The implementation correctness is validated through:

- 1) **Basis orthogonality:** Verification that basis functions form an orthogonal set

- 2) **Parseval's theorem:** Energy conservation between spatial and frequency domains:

$$\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} |f(x, y)|^2 = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} |F(u, v)|^2 \quad (14)$$

- 3) **Symmetry properties:** Real-valued inputs produce conjugate-symmetric spectra
- 4) **DC component:**  $F(0, 0) = \sum_{x,y} f(x, y)$  equals sum of pixel values

## VII. CHALLENGES AND SOLUTIONS

### A. Computational Time

**Challenge:** Direct DFT computation for  $64 \times 64$  images requires significant time (several seconds to minutes).

#### Solution:

- Pre-compute trigonometric values
- Use optimized NumPy array operations where possible
- Provide progress feedback during computation

### B. Numerical Precision

**Challenge:** Floating-point arithmetic may introduce small errors.

**Solution:** Use 64-bit floating-point (double precision) for intermediate calculations.

### C. Visualization

**Challenge:** Large dynamic range in magnitude spectrum makes visualization difficult.

**Solution:** Apply logarithmic scaling:

$$M_{\log}(u, v) = \log(1 + |F(u, v)|) \quad (15)$$

## VIII. PRACTICAL APPLICATIONS

Understanding 2-D DFT fundamentals enables:

- **Image filtering:** Design of frequency-selective filters
- **Compression:** JPEG utilizes DCT (related to DFT)
- **Pattern recognition:** Frequency domain features for classification
- **Image enhancement:** Sharpening and noise reduction
- **Motion analysis:** Velocity estimation through phase correlation

## IX. CONCLUSION

This work successfully implemented a complete 2-D DFT system from first principles without using optimized FFT libraries. The implementation demonstrates:

- 1) Generation and visualization of DFT basis functions providing insight into frequency decomposition
- 2) Direct computation of 2-D DFT for arbitrary images validating theoretical formulations
- 3) Analysis of frequency domain characteristics for binary rectangular images
- 4) Effect of spatial domain centering on frequency spectrum localization

The results confirm that:

- DFT effectively transforms spatial information into frequency components
- Image edges and discontinuities create high-frequency content
- Centering operation facilitates intuitive frequency analysis
- Direct DFT computation, while slow, reinforces fundamental understanding

Future extensions could include:

- Implementing separable 2-D DFT for improved efficiency
- Applying frequency domain filtering operations
- Comparing with FFT implementations for performance analysis
- Extending to color images using multi-channel DFT

## X. ACKNOWLEDGMENT

The authors thank to Prof. Jignesh Patel for guidance during this laboratory work. Complete source code, documentation, and results are publicly available at:

<https://github.com/vaibhav-123-4/ImageProcessing>