Image Enhancement in the Frequency Domain

UNIT 3 | Image Processing

Yuba Raj Devkota | NCCS | CSIT 5th Sem

Unit 3	**Image Enhancement in the Frequency Domain**	**Teaching Hours (8)**
Introduction	Introduction to Fourier Transform and the frequency Domain, 1-D and 2-D Continuous Fourier transform, 1-D and 2-D Discrete Fourier transform	1 hr
Properties of Fourier Transform	Logarthmic, Separability, Translation, Periodicity, Implications of Periodicity and symmetry	1 hr
Smoothing Frequency Domain Filters	Ideal Low Pass Filter, Butterworth Low Pass Filter, Gaussian Low Pass Filter	1 hr
Sharpening Frequency Domain Filters	Ideal High Pass Filter, Butterworth High Pass Filter, Gaussian High Pass Filter, Laplacian Filter	1 hr
Fast Fourier Transform	Computing and Visualizing the 2D DFT (Time Complexity of DFT), Derivation of 1-D Fast Fourier Transform, Time Complexity of FFT, Concept of Convolution, Correlation and Padding.	2 hrs.
Other Image Transforms	Hadamard transform, Haar transform and Discrete Cosine transform	2 hrs.

Image Enhancement in Frequency Domain

Image enhancement in the frequency domain works by transforming an image from the spatial (pixel) domain into the frequency domain, modifying its frequency components, and then transforming it back. This approach is especially useful for noise reduction, smoothing, sharpening, and periodic noise removal.

re 3 | Figure 4 | Figure 5 | Figure 6 | Figure 7 | Figure 8 | im | im_gray

408×414 uint8

	1	2	3	4	5	6
1	33	39	39	46	51	51
2	45	47	47	50	52	52
3	49	48	48	47	46	46
4	46	44	44	39	36	36
5	46	44	44	39	36	36
6	50	46	46	41	35	35
7	56	53	53	48	42	42
8	56	53	53	48	42	42
9	48	47	47	46	45	45
10	32	34	34	37	42	42
11	33	32	32	34	40	40
12	33	32	32	34	40	40
13	27	29	29	33	37	37
14	41	46	46	50	51	51
15	41	46	46	50	51	51

Frequency Domain Representation

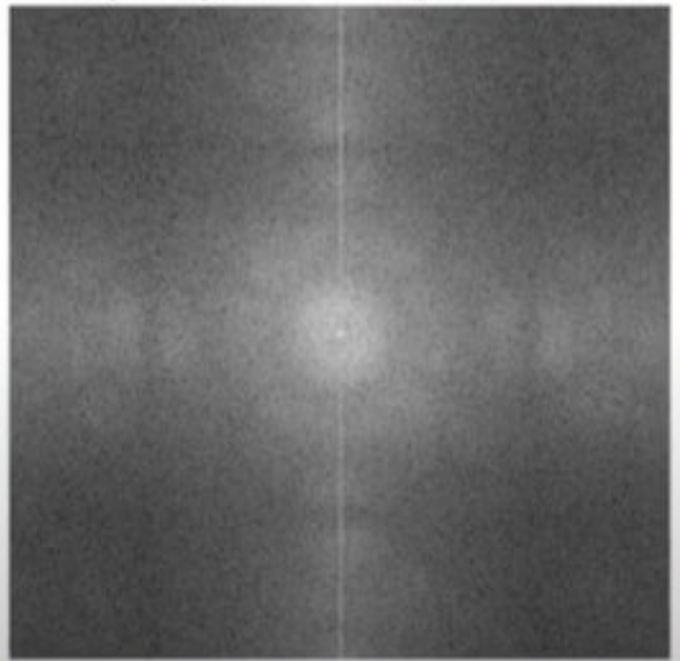
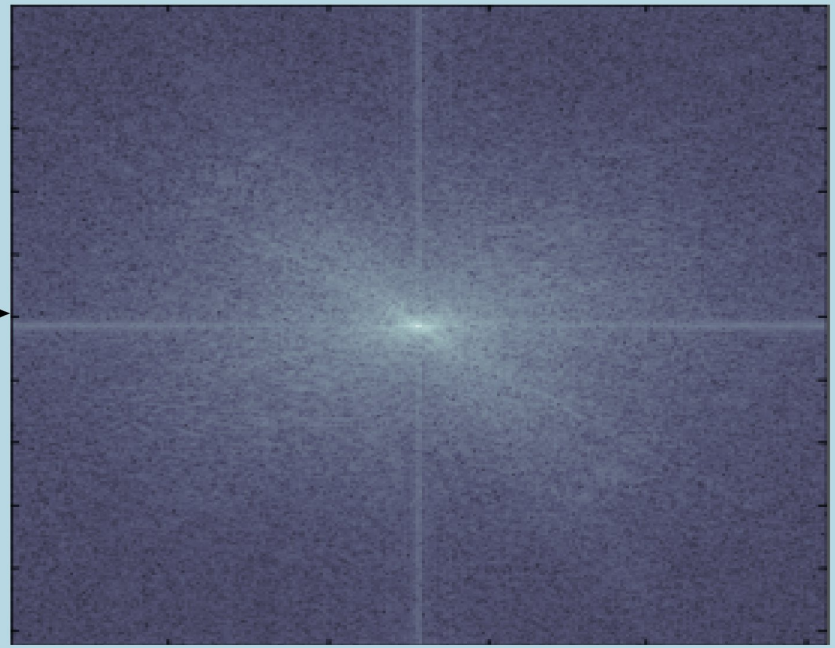


Figure 5 | Figure 6 | Figure 7 | Figure 8 | Figure 9 | Figure 10 | Figure 11 | im | im_gray | F_mag_log

408×414 complex double

	1	2	3	4
1	563	1.2540e+02 + 7.4604e+02i	-2.0653e+02 - 2.6965e+02i	5.7955e+01 - 1.3250e+02i
2	3.4572e+01 - 7.4093e+02i	6.2275e+02 + 3.0819e+02i	-1.1240e+02 - 4.4475e+01i	-1.7886e+02 - 2.7004e+02i
3	-1.9492e+02 - 3.5612e+02i	-8.1591e+01 + 1.9606e+02i	2.8925e+02 + 1.7267e+02i	-1.8710e+01 - 1.1230e+02i
4	9.9654e+01 + 5.5445e+02i	2.7094e+02 - 2.0555e+02i	-2.1178e+02 - 4.1219e+00i	-1.1322e+02 + 6.6989e+02i
5	-6.2924e+02 + 3.9648e+02i	5.8798e+02 - 5.4845e+02i	-3.7472e+02 + 5.4867e+01i	5.7135e+02 - 1.4306e+01i
6	-5.2654 - 1.1572i	-6.0181e+02 + 8.6877e+01i	5.5720e+02 + 5.2299e+02i	-9.1967e+01 + 2.7592e+02i
7	5.2332e+02 - 6.8658e+02i	-4.8072e+02 + 8.2406e+02i	2.6013e+01 - 2.9693e+02i	-1.6458e+02 - 1.0815e+02i
8	-5.8680e+02 + 2.3801e+02i	7.9953e+01 + 1.1179e+02i	5.6770e+02 + 1.5510e+02i	-2.1570e+02 - 8.1892e+01i
9	2.5913e+02 + 1.2170e+02i	2.8732e+02 - 2.5347e+02i	-5.7905e+02 - 3.8429e+02i	3.5600e+02 + 2.7501e+01i
10	-1.9304e+02 + 2.4179e+02i	3.3754e+02 - 3.2286e+02i	-2.1706e+02 + 1.8203e+02i	4.5312e+01 - 1.9841e+02i
11	5.0426e+02 - 2.0504e+01i	-5.8410e+02 - 3.0598e+01i	4.6916e+02 + 4.1675e+02i	-3.1892e+02 - 9.8888e+01i
12	-3.6034 -29.5433i	78.8237 + 0.6180i	-3.9006e+02 + 9.4479e+00i	8.3029e+02 - 3.7875e+01i
13	-2.0855e+02 - 2.8889e+02i	6.5255e+02 + 1.8463e+02i	-6.6152e+01 - 3.8918e+02i	3.2795e+02 + 4.8134e+02i
14	1.2150e+02 + 2.8288e+02i	2.1660e+02 + 4.0214e+02i	-4.2933e+02 - 6.3192e+02i	2.1316e+02 + 2.7192e+02i
15	-3.6419e+02 + 9.1827e+01i	5.0590e+01 - 4.1215e+02i	-9.3613e+01 + 1.1507e+02i	9.0698e+01 + 4.5351e+02i

Fourier Transform



Spatial Domain

Frequency Domain

Fourier Transform (FT)

The Fourier Transform decomposes an image into its sinusoidal components of different frequencies. In image processing, it tells us how fast pixel intensities change across the image.

- ***Low frequencies → smooth regions, background, gradual intensity changes***
- ***High frequencies → edges, fine details, noise***

The graphical representation of the sine function with its magnitude and phase spectra is shown in Figure-1.

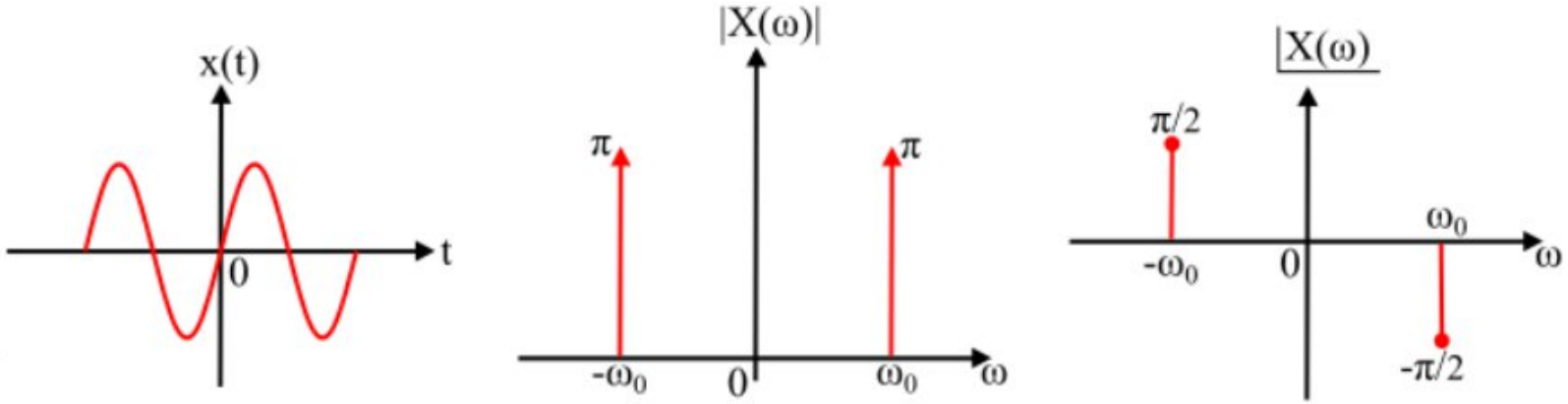
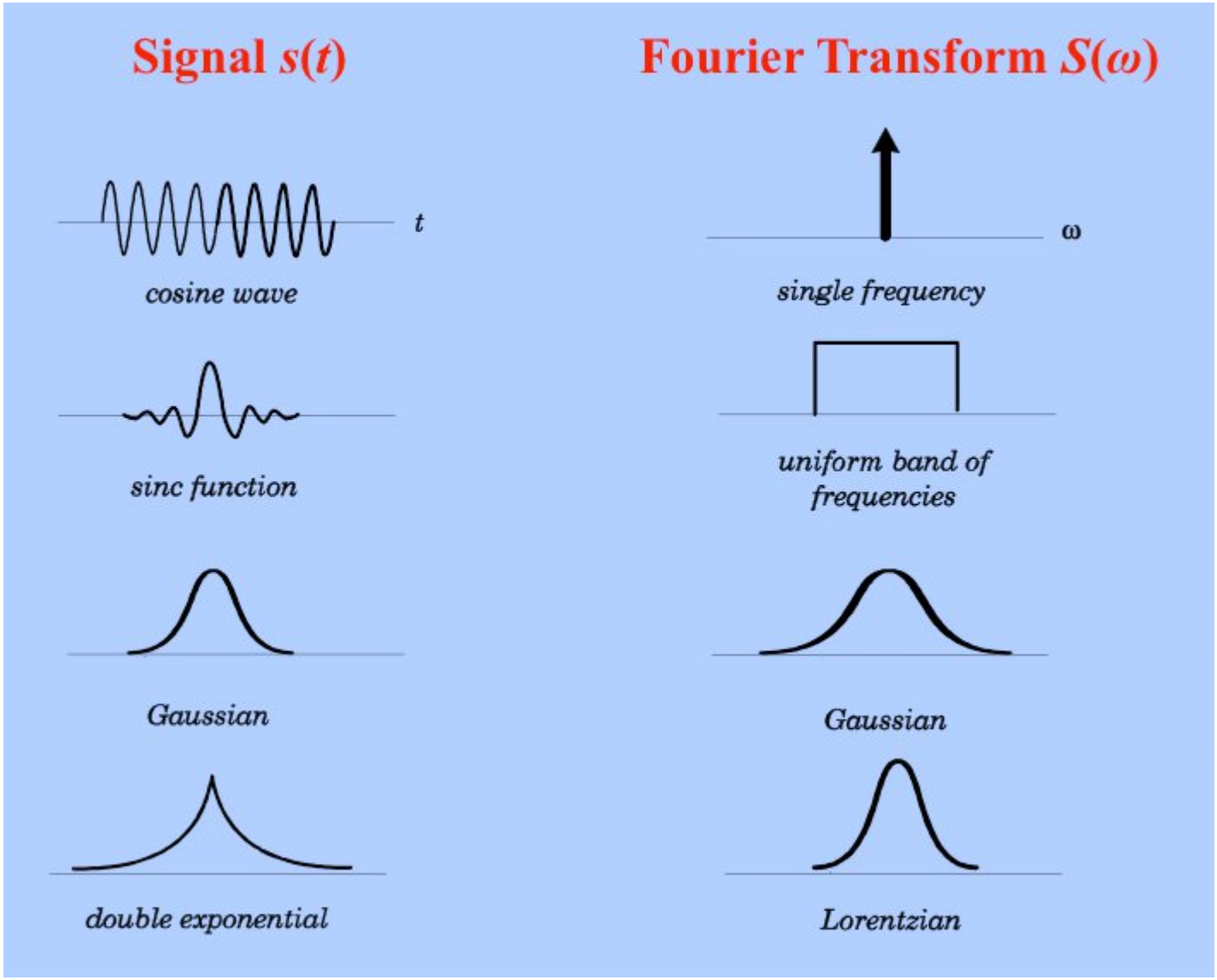


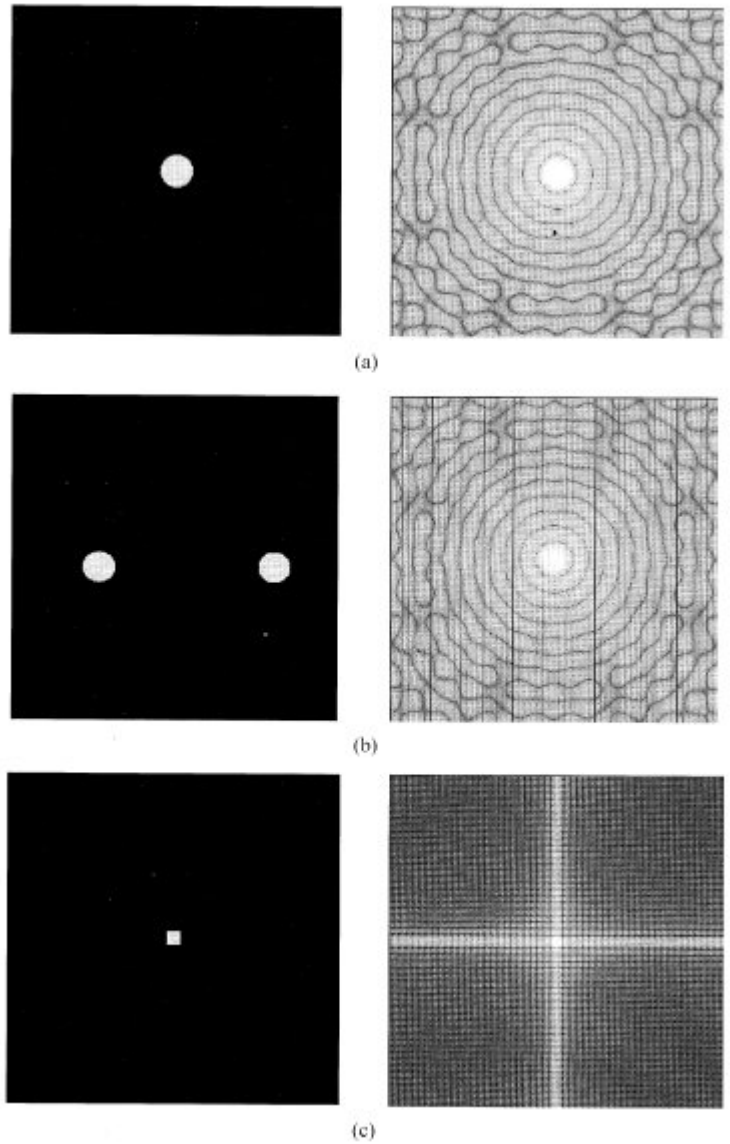
Figure-1

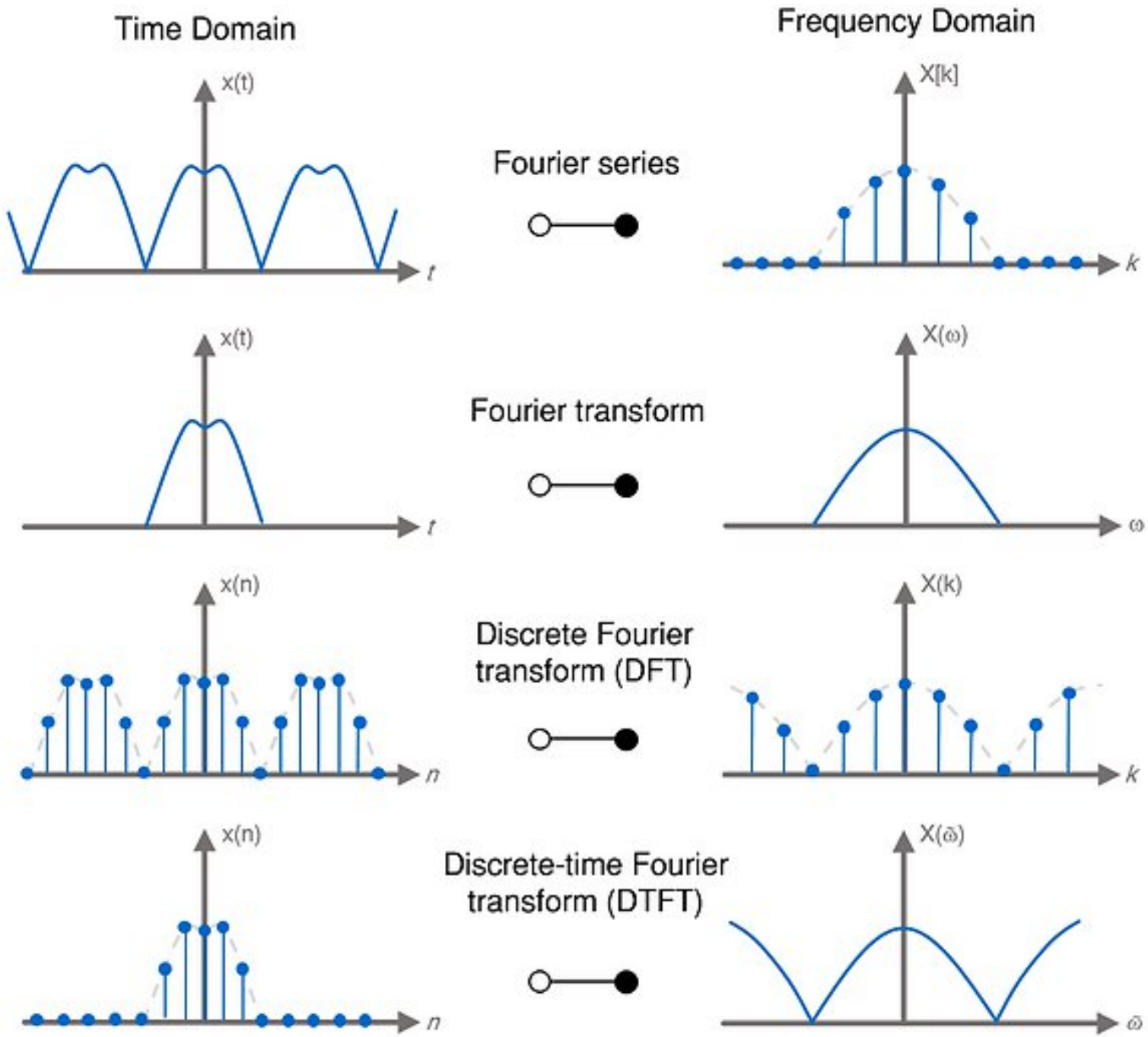
- **Magnitude Spectrum**: shows strength of frequencies
- **Phase Spectrum**: contains structural information of the image

☞ ***Important: Phase is often more important than magnitude for image reconstruction.***

Continuous and Discrete FT (1D / 2D)







Continuous Fourier Transform (1D / 2D)

Discrete Fourier Transform (1D / 2D)

1. 1D Continuous Fourier Transform (CTFT)

Used for **continuous-time signals** (e.g., analog signals).

Forward Transform

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

Inverse Transform

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

Where:

- $f(x)$ → continuous signal
- $F(u)$ → frequency representation
- u → frequency variable
- $j = \sqrt{-1}$

✔ Shows how a signal is composed of infinite sinusoidal waves.

2. 1D Discrete Fourier Transform (DFT)

Used for **finite-length digital signals**.

Forward DFT

$$F(u) = \sum_{x=0}^{N-1} f(x)\, e^{-j2\pi \frac{ux}{N}}$$

Inverse DFT

$$f(x) = \frac{1}{N} \sum_{}^{N-1} F(u)\, e^{j2\pi \frac{ux}{N}}$$

Where:

- N → number of samples
- $f(x)$ → discrete signal
- $F(u)$ → discrete frequency spectrum

✔ Used in **FFT algorithms**, audio processing, and image processing.

3. 2D Continuous Fourier Transform

Used for **continuous images** $f(x, y)$.

Forward Transform

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

Inverse Transform

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v)\, e^{j2\pi(ux+vy)}\, du\, dv$$

Where:

- (x, y) → spatial coordinates
- (u, v) → frequency coordinates

✔ Foundation for **optical systems** and theoretical image analysis.

4. 2D Discrete Fourier Transform (DFT) ⭐ (Most Important for DIP)

Used for **digital images** of size $M \times N$.

Forward 2D DFT

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

Inverse 2D DFT

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

Where:

- $f(x,y)$ → digital image
- $F(u,v)$ → frequency spectrum
- M, N → image dimensions

✔ Used for **image enhancement, filtering, compression, and restoration**.

Type	Dimension	Signal Type	Application
1D Continuous FT	1D	Analog	Communication systems
1D Discrete FT	1D	Digital	Audio, speech processing
2D Continuous FT	2D	Continuous image	Optical imaging
2D Discrete FT	2D	Digital image	Image enhancement (DIP)

- **1D → signals, 2D → images**
- **Continuous → integrals**
- **Discrete → summations**
- 2D DFT is the **most commonly used** in image processing

1️⃣ Simple Image (Spatial Domain)

Consider a **4×4 grayscale image** (very small so it's easy to understand):

$$f(x,y) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

👉 This is a **constant image** (all pixels have same intensity).

2️⃣ Fourier Transform of the Image

The **2-D Discrete Fourier Transform (DFT)** is:

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

What happens here?

- Since **all pixel values are constant**
- The image has **no variation**
- So it contains **only low frequency (DC component)**

Fourier Result:

$$F(u,v) = \begin{bmatrix} 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- ✔ The value **16** is at **(0,0)** → called the **DC component**
- ✔ All other frequencies are **zero**

3️⃣ Meaning of Fourier Components

- **DC component (0,0)** → Average brightness of image
- **High frequencies** → edges and details
- **Low frequencies** → smooth areas

Since image is flat, **no edges → no high frequencies**

1 Calculation of F(0,0)

Substitute $u = 0,\ v = 0$:

$$F(0,0) = \sum_{x=0}^{3}\sum_{y=0}^{3} 1 \cdot e^{-j2\pi(0+0)}$$

$$e^{0} = 1$$

So:

$$F(0,0) = \sum_{x=0}^{3}\sum_{y=0}^{3} 1$$

Total number of pixels = $4 \times 4 = 16$

$$\boxed{F(0,0) = 16}$$

✔ This is the **DC component (average intensity × total pixels)**

2 Calculation of F(0,1)

Substitute $u = 0,\ v = 1$:

$$F(0,1) = \sum_{x=0}^{3}\sum_{y=0}^{3} 1 \cdot e^{-j2\pi\left(0+\frac{y}{4}\right)}$$

$$= \sum_{x=0}^{3}\sum_{y=0}^{3} e^{-j2\pi y/4}$$

Now evaluate the **inner sum (over y)**:

$$y = 0 \Rightarrow e^{0} = 1$$

$$y = 1 \Rightarrow e^{-j\pi/2} = -j$$

$$y = 2 \Rightarrow e^{-j\pi} = -1$$

$$y = 3 \Rightarrow e^{-j3\pi/2} = +j$$

Add them:

$$1 + (-j) + (-1) + (j) = 0$$

So for **each x**, sum = 0

$$F(0,1) = \sum_{x=0}^{3} 0 = \boxed{0}$$

4️⃣ Inverse Fourier Transform (Back to Image)

The **Inverse DFT** is:

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

Substituting $F(0,0) = 16$:

$$f(x,y) = \frac{16}{16} = 1$$

Reconstructed Image:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

- ✔ Same image obtained back
- ✔ **No information loss**

Fourier Transform converts an image from spatial domain into frequency components, and Inverse Fourier Transform reconstructs the original image by combining those frequencies.

Properties of Fourier Transform

1. Linearity

If

$$f(x,y) = af_1(x,y) + bf_2(x,y)$$

then

$$F(u,v) = aF_1(u,v) + bF_2(u,v)$$

✔ Useful when combining images or signals.

2. Translation (Shift) Property

Shifting an image in spatial domain affects **only the phase**, not magnitude.

$$f(x - x_0, y - y_0) \Rightarrow F(u,v)e^{-j2\pi(\frac{ux_0}{M} + \frac{vy_0}{N})}$$

✔ Explains why magnitude spectrum remains centered.

3. Scaling Property

Scaling in spatial domain causes **inverse scaling** in frequency domain.

- Small objects → wide frequency spread
- Large objects → narrow frequency spread

4. Rotation Property

If an image is rotated in spatial domain, its Fourier transform rotates by the **same angle**.

✔ Helpful in pattern recognition.

5. Convolution Theorem (Very Important ⭐)

Convolution in spatial domain ↔ Multiplication in frequency domain

$$f(x,y) * h(x,y) \leftrightarrow F(u,v) \cdot H(u,v)$$

✔ This is the **basis of frequency-domain filtering**.

Convolution Combines a pixel with its neighbors while in multiplication No neighborhood involved

6. Correlation Property of Fourier Transform:

The correlation property of the Fourier Transform states that correlation in the spatial (or time) domain corresponds to multiplication with the complex conjugate in the frequency domain. This property is very important in image processing, pattern matching, and template matching.

$$\boxed{\mathcal{F}\{f(x) \star g(x)\} = F(u)\, G^*(u)}$$

Where:

- $F(u) = \mathcal{F}\{f(x)\}$
- $G(u) = \mathcal{F}\{g(x)\}$
- $G^*(u) \rightarrow$ complex conjugate of $G(u)$

Conjugation ensures:

- *Phase alignment, Real-valued energy for auto-correlation*
- *Without conjugate, it becomes convolution, not correlation*

The complex conjugate of a complex number is obtained by changing the sign of its imaginary part.

If a complex number is:

$$z = a + jb$$

then its **complex conjugate** is:

$$\boxed{z^* = a - jb}$$

Where:

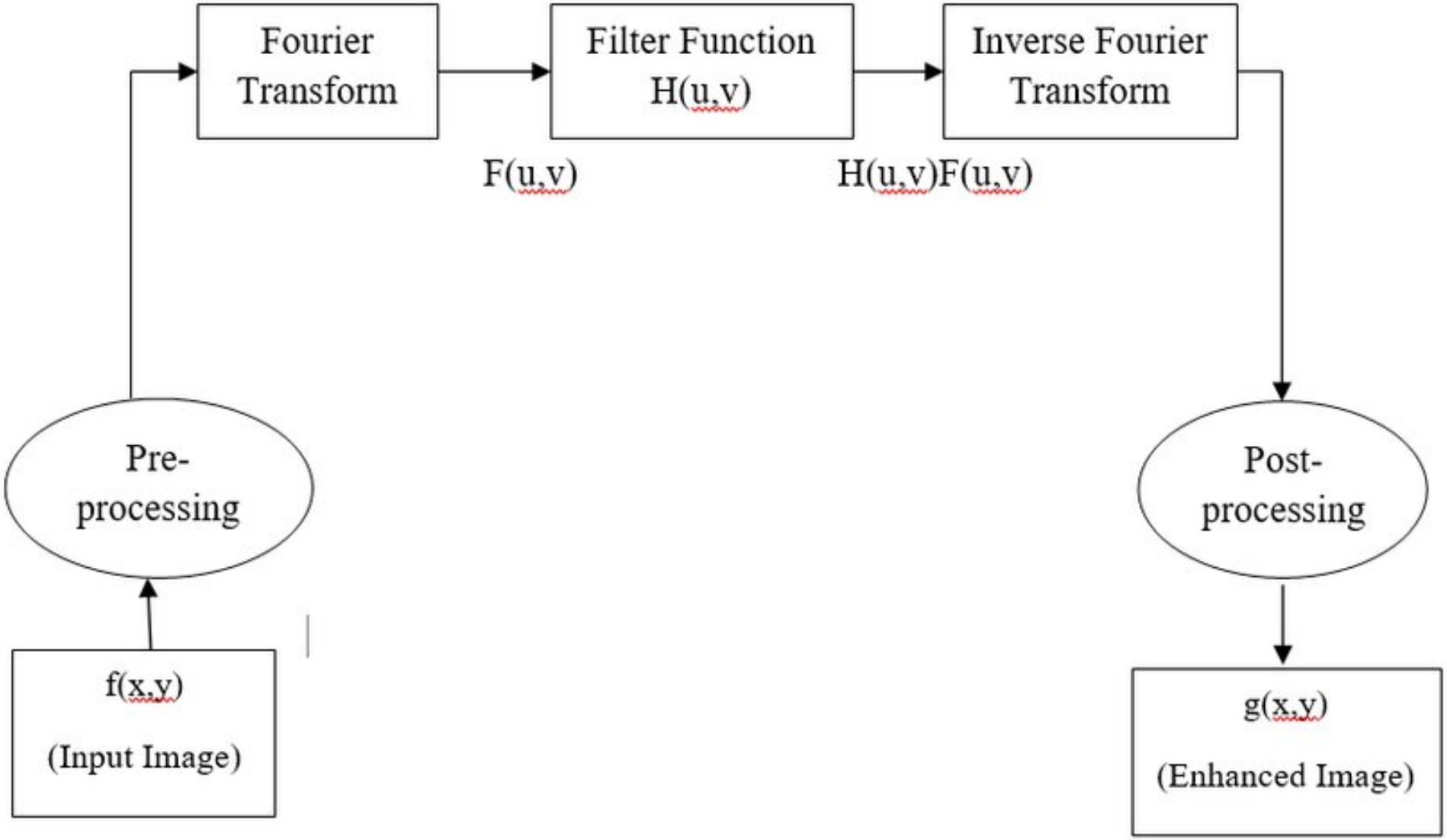
- $a \rightarrow$ real part
- $b \rightarrow$ imaginary part
- $j = \sqrt{-1}$

7. Energy Conservation (Parseval's Theorem)

Total energy in spatial domain equals total energy in frequency domain.

$$\sum |f(x,y)|^2 = \frac{1}{MN} \sum |F(u,v)|^2$$

Steps for filtering in frequency domain



Sharpening Frequency Filters (High Pass Filters)

In the frequency domain, High-Pass Filters (HPF) are mainly used for sharpening and edge enhancement, not smoothing. They work by attenuating low-frequency components (smooth background) and preserving or enhancing high-frequency components (edges, fine details).

- "Low-pass filters → smoothing"
- "High-pass filters → sharpening"

Concept of High-Pass Filtering

- Low frequencies: slow gray-level changes (background, smooth regions)
- High frequencies: rapid changes (edges, noise, fine details)

A high-pass filter:

$$H(u,v) = \begin{cases} 0, & \text{low frequencies} \\ 1, & \text{high frequencies} \end{cases}$$

The enhanced image is obtained as:

$$G(u,v) = H(u,v)\,F(u,v)$$

Then apply **Inverse FFT** to get the spatial image.

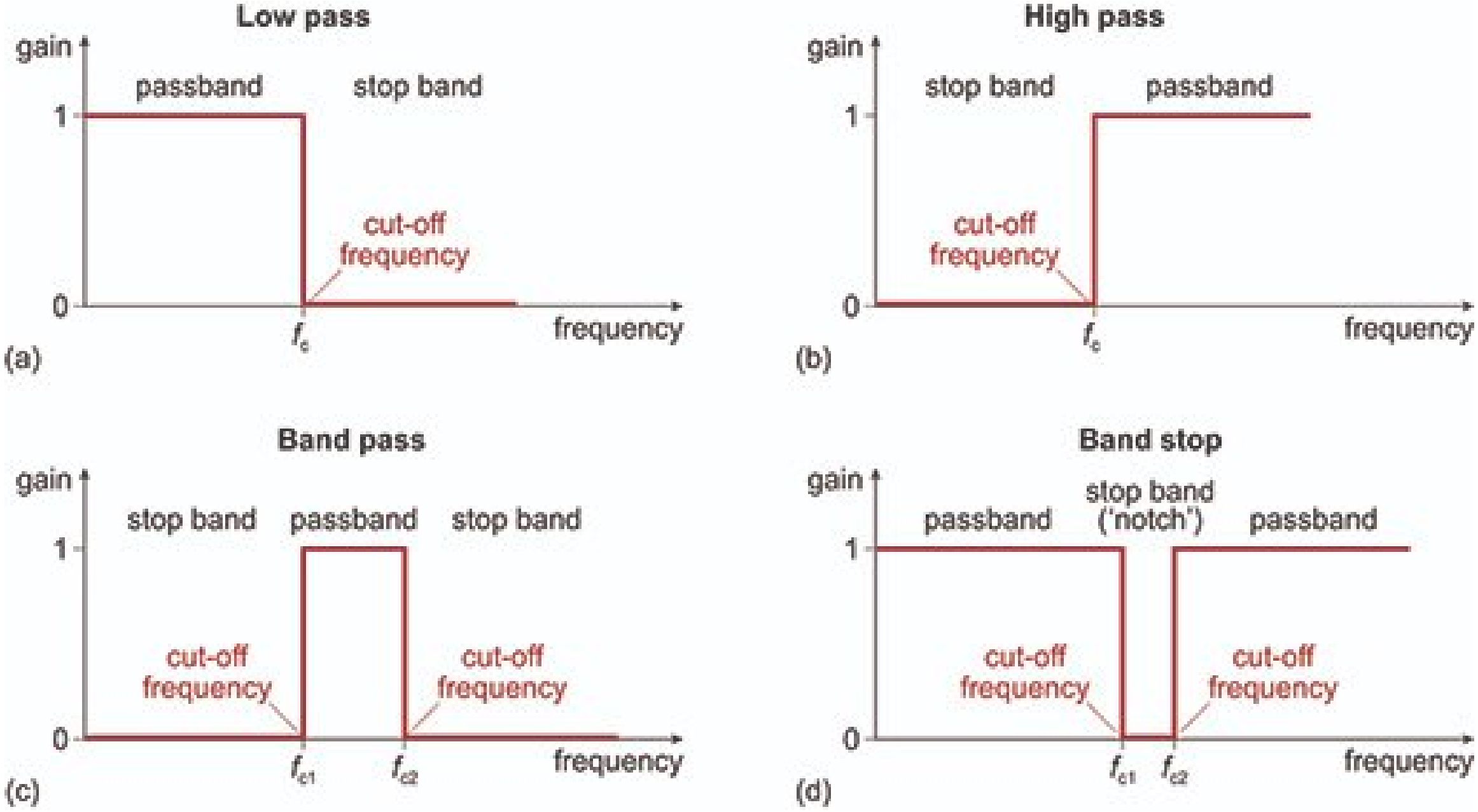

Low pass
gain
passband
stop band
1
0
cut-off
frequency
f_c
frequency
(a)
High pass
gain
stop band
passband
1
0
cut-off
frequency
f_c
frequency
(b)
Band pass
gain
stop band
passband
stop band
1
0
cut-off
frequency
cut-off
frequency
f_{c1}
f_{c2}
frequency
(c)
Band stop
gain
passband
stop band
('notch')
passband
1
0
cut-off
frequency
cut-off
frequency
f_{c1}
f_{c2}
frequency
(d)

Fig. 3: original image

Fig. 4: Result of Gaussian low pass filter

Fig. 5: Gaussian high pass filter

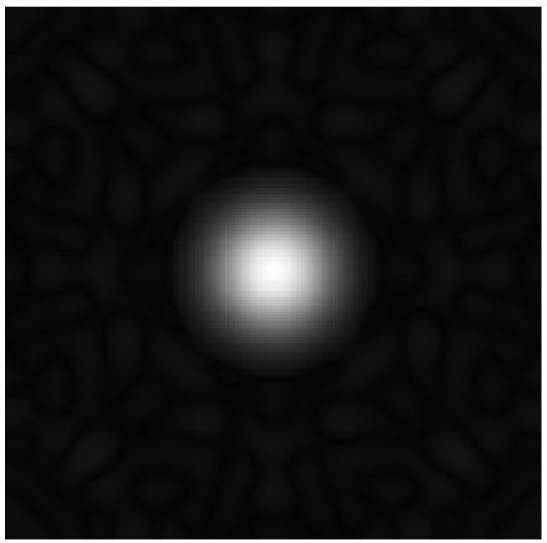


Fig. 6: Low pass filter in the frequency domain

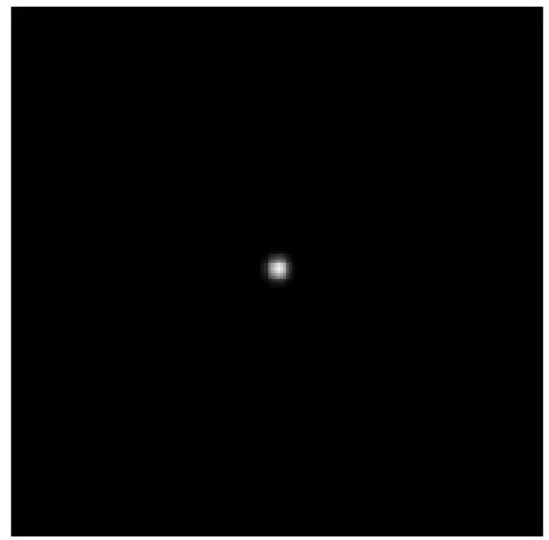


Fig. 7: Low pass filter in the spatial domain

Fig. 8: High pass filter in the frequency domain

Types of High Pass Filters in F.D.

1. Ideal High-Pass Filter (IHPF)

The simplest high-pass filter.

$$H(u,v) = \begin{cases} 0, & D(u,v) \leq D_0 \\ 1, & D(u,v) > D_0 \end{cases}$$

Where:

- $D(u,v)$ = distance from the frequency origin
- D_0 = cutoff frequency

✔ Removes all low frequencies sharply

2. Butterworth High-Pass Filter

Provides a **smooth transition** between low and high frequencies.

$$H(u,v) = \frac{1}{1 + \left(\frac{D_0}{D(u,v)}\right)^{2n}}$$

Where:

- n = order of the filter

✔ Less ringing than Ideal HPF

✔ Widely used in practice

*In image processing, "ringing" refers to an unwanted visual artifact that appears as **ripples, halos, or oscillations** near sharp edges or high-contrast boundaries in an image. It's also sometimes called the **"Gibbs phenomenon"** in signal processing.*

3. Gaussian High-Pass Filter (GHPF)

Most smooth and natural sharpening filter.

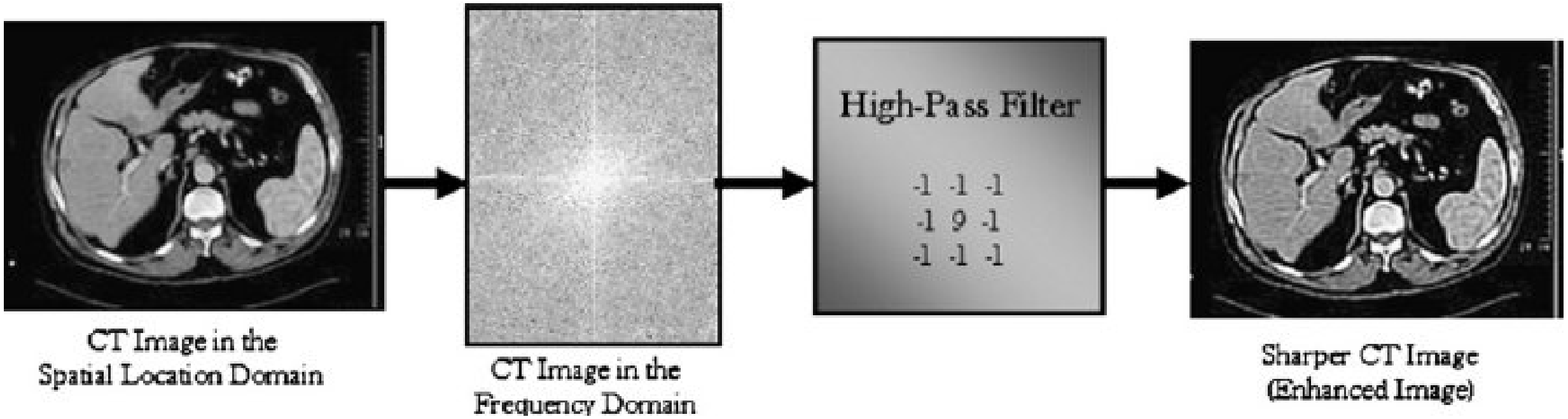
$$H(u,v) = 1 - e^{-\frac{D^2(u,v)}{2D_0^2}}$$

- ✔ No ringing
- ✔ Best visual quality
- ✔ Most preferred in image enhancement

Filter	Transition	Ringing	Quality
Ideal HPF	Abrupt	High	Poor
Butterworth HPF	Moderate	Medium	Good
Gaussian HPF	Smooth	None	Excellent

Effect of High-Pass Filtering on Image

- Background becomes less dominant
- Edges become clearer
- Fine details are enhanced



Describe in brief that how do you implement Gaussian High Pass Frequency Filter for image sharpening in the frequency domain?

Steps to Implement Gaussian High-Pass Frequency Filter

1. Read and preprocess the image

- Convert the image to **grayscale**
- Convert pixel values to floating point

---

2. Compute the Fourier Transform

- Apply **2D FFT** to the image:

$$F(u,v) = \text{FFT}\{f(x,y)\}$$

3. Shift the zero frequency to the center

- Use frequency shift so that low frequencies are at the center:

$$F_c(u,v) = \text{fftshift}(F(u,v))$$

4. Construct Gaussian High-Pass Filter

First compute distance from the center:

$$D(u,v) = \sqrt{(u-u_0)^2 + (v-v_0)^2}$$

Gaussian High-Pass Filter:

$$\boxed{H(u,v) = 1 - e^{-\frac{D^2(u,v)}{2D_0^2}}}$$

Where:

- D_0 = cutoff frequency (controls sharpening strength)

5. Apply the filter in frequency domain

$$G(u,v) = H(u,v)\,F_c(u,v)$$

6. Inverse shift and inverse FFT

- Apply inverse shift:

$$G_s(u,v) = \text{ifftshift}(G(u,v))$$

- Apply inverse FFT:

$$g(x,y) = \text{IFFT}\{G_s(u,v)\}$$

7. Take magnitude and display result

- Take absolute value
- Normalize for display

Key Result

- Low frequencies (smooth background) are suppressed
- High frequencies (edges, fine details) are enhanced
- Image appears **sharper**

Smoothing Frequency Filters (Low Pass Filters)

In frequency-domain image enhancement, Low-Pass Filters (LPF) are used for smoothing an image. They work by preserving low-frequency components (slow intensity variations) and suppressing high-frequency components (edges and noise).

- "Low-Pass Filters → Smoothing / Noise reduction"
- "High-Pass Filters → Sharpening"

Basic Idea of Low-Pass Filtering

- **Low frequencies** → smooth regions, background
- **High frequencies** → edges, noise

The filtering operation is:

$$G(u,v) = H(u,v)\,F(u,v)$$

where

- $F(u,v)$ = Fourier transform of image
- $H(u,v)$ = low-pass filter
- $G(u,v)$ = filtered spectrum

After applying **Inverse FFT**, we get the smoothed image.

Types of Smoothing (Low-Pass) Frequency Filters

1. Ideal Low-Pass Filter (ILPF)

The simplest low-pass filter with a **sharp cutoff**.

Transfer Function

$$H(u,v) = \begin{cases} 1, & D(u,v) \leq D_0 \\ 0, & D(u,v) > D_0 \end{cases}$$

Where:

- $D(u,v)$ = distance from frequency origin
- D_0 = cutoff frequency

Characteristics

✔ Preserves all frequencies below D_0

2. Butterworth Low-Pass Filter (BLPF) ⭐

Provides a **smooth transition** between passband and stopband.

Transfer Function

$$H(u,v) = \frac{1}{1 + \left(\frac{D(u,v)}{D_0}\right)^{2n}}$$

Where:

- n = order of filter (controls sharpness)

Characteristics

- ✔ Less ringing than Ideal LPF
- ✔ Adjustable smoothness using order n
- ✔ Widely used in practice

3. Gaussian Low-Pass Filter (GLPF) ⭐ ⭐

Most commonly used smoothing filter.

Transfer Function

$$H(u,v) = e^{-\frac{D^2(u,v)}{2D_0^2}}$$

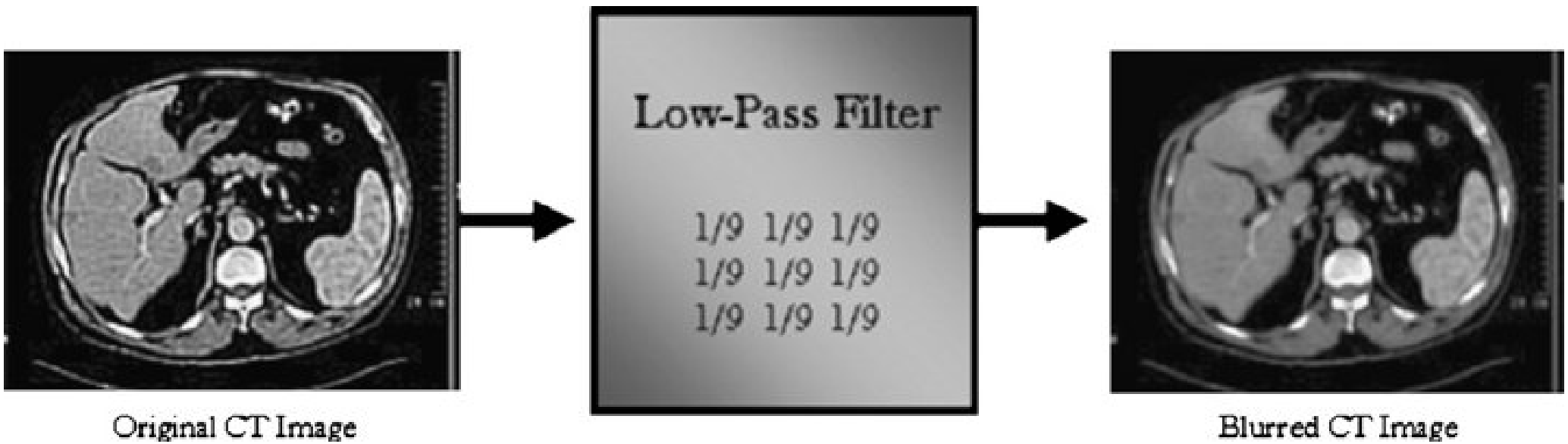
Characteristics

- ✔ **No ringing effect**
- ✔ Very smooth and natural results
- ✔ Best visual quality for smoothing

Ideal LPF	Abrupt	High	Poor
Butterworth LPF	Moderate	Medium	Good
Gaussian LPF	Smooth	None	Excellent

Effect of Low-Pass Filtering on Image

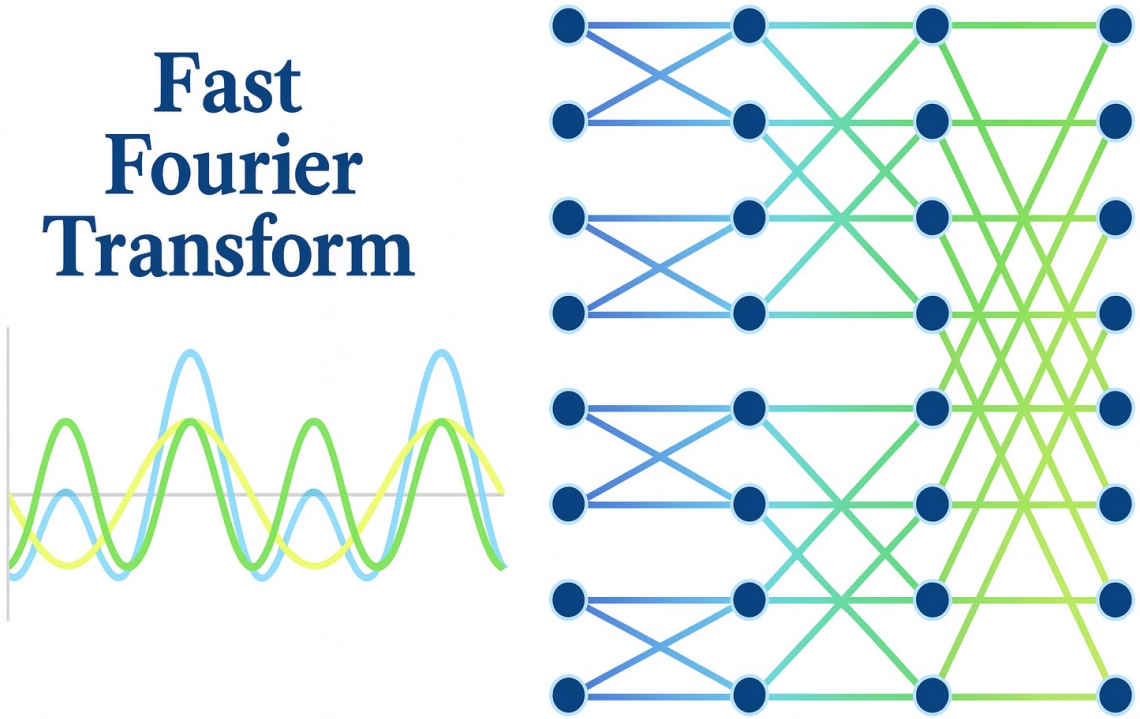
- Image becomes **blurred**
- Noise is reduced
- Fine details and edges are softened



Fast Fourier Transform (FFT)

- ***Fast Fourier Transform (FFT) is an efficient algorithm used to compute the Discrete Fourier Transform (DFT) and its inverse. The main purpose of FFT is to convert a signal or sequence from the spatial/time domain into the frequency domain with much less computation compared to direct methods.***

- ***In image processing and signal processing, FFT helps us analyze frequency components, such as low-frequency (smooth areas) and high-frequency (edges, noise) information.***



Advantage of FFT Over FT (FFT vs FT)

- ***Fourier Transform (FT) is a mathematical concept that represents a signal as a sum of sinusoids. For digital signals, we practically use Discrete Fourier Transform (DFT).***

- ***FFT is not a new transform, but a fast computational technique to calculate DFT efficiently.***

Key differences are:

- FT / DFT focuses on *what* transformation is done
- FFT focuses on *how fast* the transformation is computed
- Direct DFT computation requires $\mathbf{O(N^2)}$ operations
- FFT reduces complexity to **O(N log N)**
- FFT is suitable for real-time and large-size data (images, audio, video)

FFT (Fast Fourier Transform) – Formula

FFT is an **efficient algorithm** to compute the **DFT (Discrete Fourier Transform)**.

DFT Formula

For an N-point signal $x(n)$:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn}$$

where,

$$W_N = e^{-j\frac{2\pi}{N}}$$

Key FFT Idea (Divide & Conquer)

FFT splits the DFT into:

- **Even indexed samples**
- **Odd indexed samples**

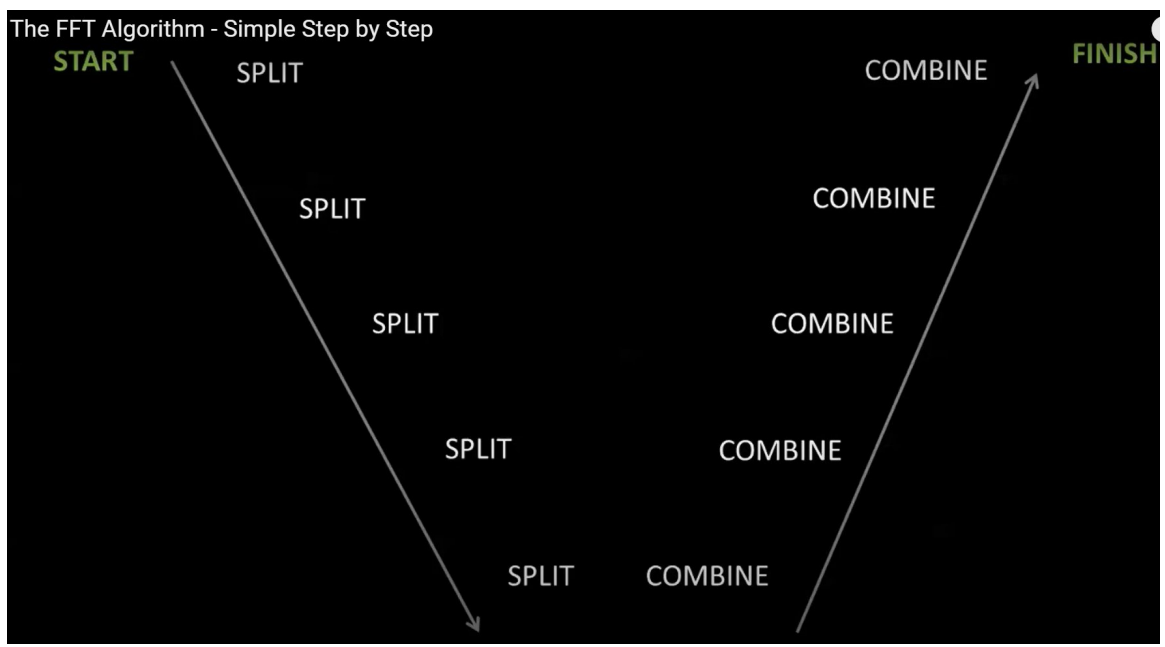
$$X(k) = E(k) + W_N^k O(k)$$

$$X(k + N/2) = E(k) - W_N^k O(k)$$

where

$E(k)$ = DFT of even samples

$O(k)$ = DFT of odd samples



Explain FFT Algorithm for one dimensional Case.

FFT (Fast Fourier Transform) – Formula

FFT is an **efficient algorithm** to compute the **DFT (Discrete Fourier Transform)**.

DFT Formula

For an N-point signal $x(n)$:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn}$$

where,

$$W_N = e^{-j\frac{2\pi}{N}}$$

Step-by-Step FFT Algorithm (1-D, Radix-2 DIT)

Step 1: Input Sequence

Given a 1-D sequence:

$$x(0), x(1), x(2), \ldots, x(N-1)$$

Step 2: Divide the Sequence

Split the signal into:

- **Even indexed samples**
- **Odd indexed samples**

$$x_e(n) = x(2n)$$

$$x_o(n) = x(2n+1)$$

Step 3: Compute Smaller DFTs

Recursively compute:

- DFT of even part → $E(k)$
- DFT of odd part → $O(k)$

Each sub-problem is of size $N/2$.

Step 4: Twiddle Factor

Define the twiddle factor:

$$W_N^k = e^{-j\frac{2\pi}{N}k}$$

Step 5: Combine (Butterfly Operation)

For $k = 0$ to $\frac{N}{2} - 1$:

$$X(k) = E(k) + W_N^k \, O(k)$$

$$X\left(k + \frac{N}{2}\right) = E(k) - W_N^k \, O(k)$$

This operation is called the **butterfly computation**.

Step 6: Repeat Recursively

Repeat steps 2–5 until 2-point DFTs are reached:

$$X(0) = x(0) + x(1)$$

$$X(1) = x(0) - x(1)$$

FFT Example (4-Point FFT)

Given signal

$$x(n) = [1,\ 2,\ 3,\ 4]$$

Step 1: Separate Even & Odd Samples

Even indices:

$$x_e = [x(0), x(2)] = [1,\ 3]$$

Odd indices:

$$x_o = [x(1), x(3)] = [2,\ 4]$$

Step 2: Compute 2-Point DFTs

DFT of even part

$$E(0) = 1 + 3 = 4$$

$$E(1) = 1 - 3 = -2$$

DFT of odd part

$$O(0) = 2 + 4 = 6$$

$$O(1) = 2 - 4 = -2$$

Step 3: Twiddle Factor

For $N = 4$:

$$W_4 = e^{-j\pi/2}$$

$$W_4^0 = 1, \quad W_4^1 = -j$$

Step 4: Combine Results

$$X(0) = E(0) + W_4^0 O(0) = 4 + 6 = 10$$

$$X(1) = E(1) + W_4^1 O(1) = -2 + (-j)(-2) = -2 + 2j$$

$$X(2) = E(0) - W_4^0 O(0) = 4 - 6 = -2$$

$$X(3) = E(1) - W_4^1 O(1) = -2 - 2j$$

Final FFT Output

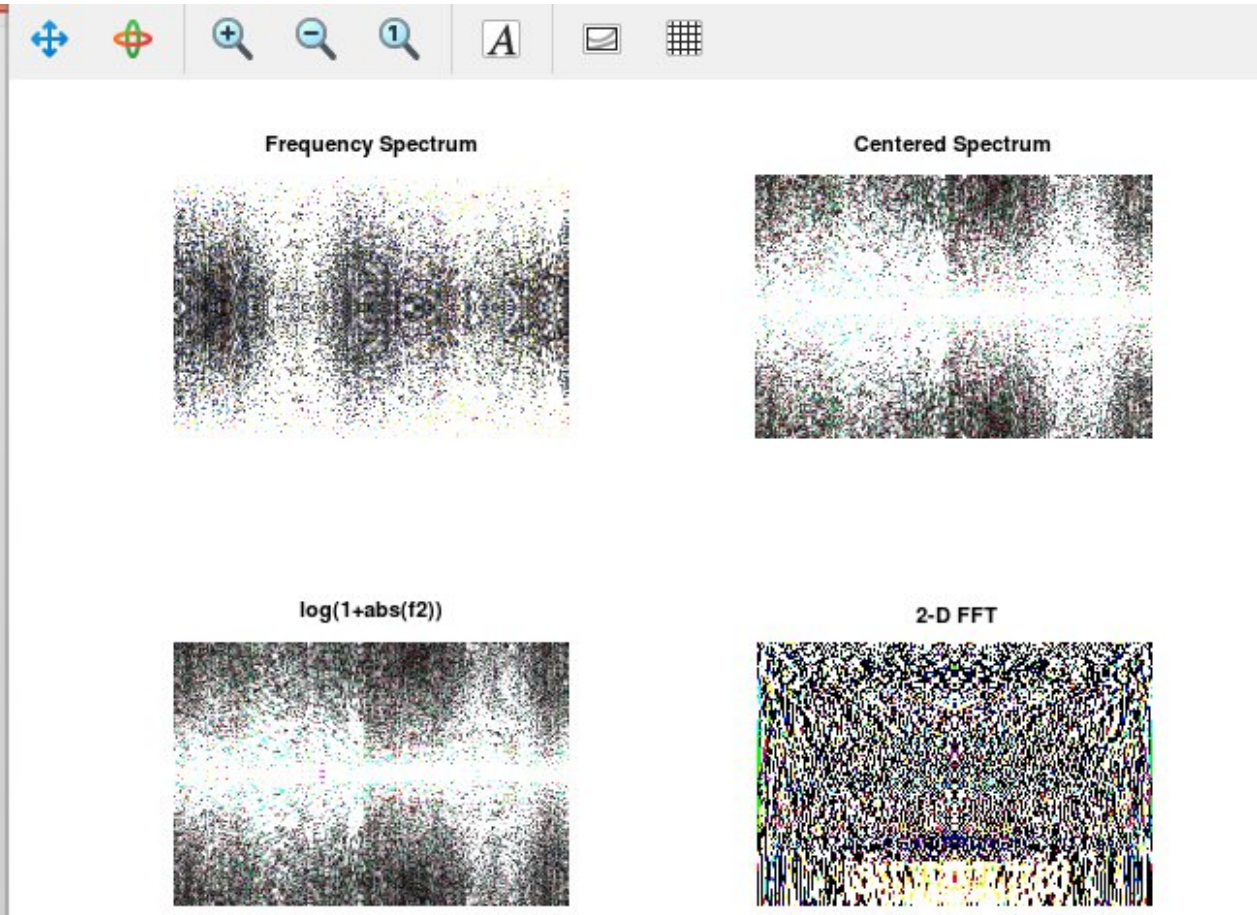
$$X(k) = [\ 10,\ -2 + 2j,\ -2,\ -2 - 2j\ ]$$

Why FFT is Important

FFT reduces DFT computation from $O(N^2)$ to $O(N \log N)$, making frequency-domain processing of images and signals fast.

Lab 09: FFT

```
clear all; % clear all variables
close all; % close all figures
clc;       % clear command window
% import image package
pkg load image;
% read image
l=im2double(imread('tiger.jpg'));
f1=fft(l);
f2=fftshift(f1);
subplot(2,2,1); imshow(abs(f1)); title('Frequency Spectrum');
subplot(2,2,2); imshow(abs(f2)); title('Centered Spectrum');
f3=log(1+abs(f2));
subplot(2,2,3); imshow(f3); title('log(1+abs(f2))');
l=fft2(f1);
l1=real(l);
subplot(2,2,4); imshow(l1);title(' 2-D FFT');
```



Other Image Transformations

Apart from Fourier Transform and Fast Fourier Transforms, some other Image Transformations in Frequency Domain includes:

- Hadamard Transform (HT)
- Haar Transform (Haar)
- Discrete Cosine Transform (DCT)

Hadamard Transform (HT)

- ***The Hadamard Transform is a linear, orthogonal transform that converts a signal or image from the spatial (or time) domain into a set of coefficients using only addition and subtraction.***
- ***It uses a matrix whose elements are +1 and −1, called the Hadamard matrix.***

Unlike Fourier Transform, Hadamard Transform does not use sine or cosine functions.

Hadamard Transform Formula

For an N-point signal x (where $N = 2^m$):

$$Y = H_N x$$

where

- x = input signal vector
- Y = transformed output
- H_N = Hadamard matrix of order N

The inverse transform is:

$$x = \frac{1}{N} H_N Y$$

Hadamard Matrix

2×2 Hadamard Matrix

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

4×4 Hadamard Matrix

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Example of Hadamard Transform (Simple 1-D Example)

Given input signal

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Step 1: Use Hadamard Matrix H_2

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Step 2: Compute Transform

$$Y = H_2\, x = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$Y = \begin{bmatrix} 1(1) + 1(2) \\ 1(1) - 1(2) \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$$

Final Output

$$Y = [\,3,\ -1\,]$$

Inverse Hadamard Transform (Check)

$$x = \frac{1}{2} H_2 Y = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 3 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

✔ Original signal recovered.

Applications

- Image compression
- Pattern recognition
- Digital image processing
- Noise reduction

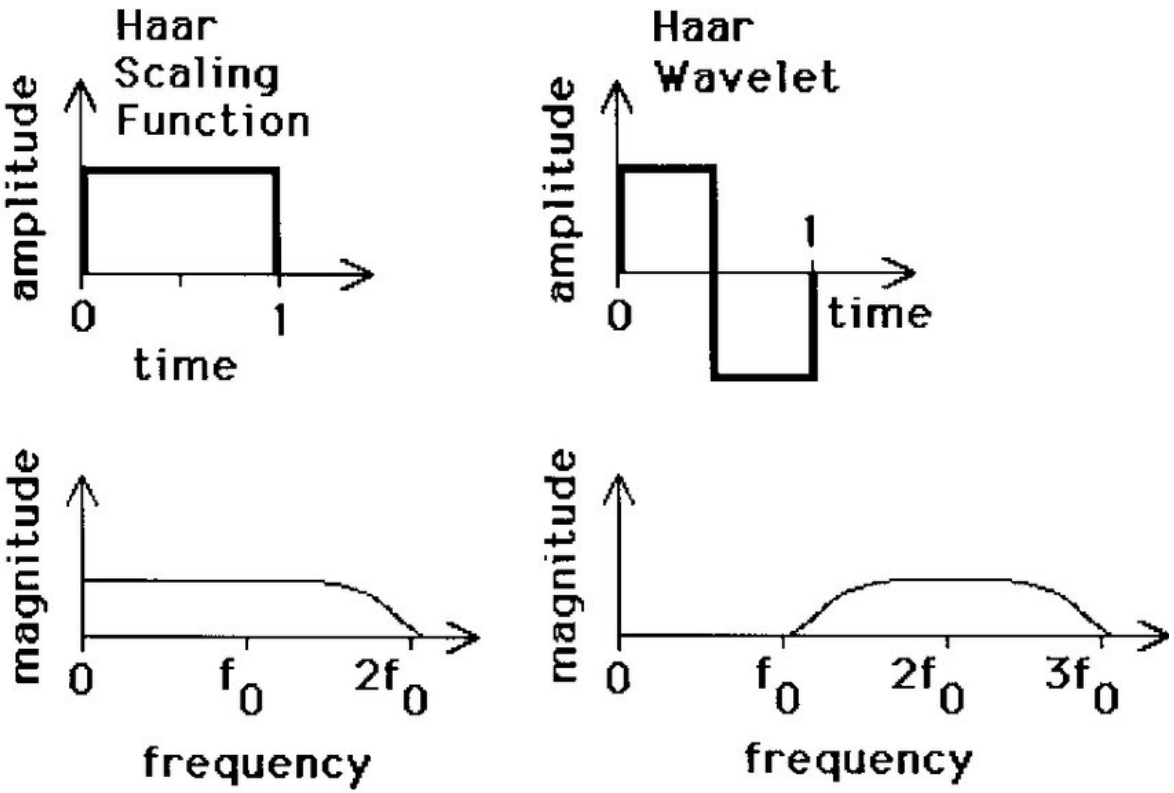
The Hadamard Transform is an orthogonal transform that represents a signal using +1 and −1 basis functions and requires only addition and subtraction operations.

Haar Transform

- ***The Haar Transform is the simplest wavelet transform used in signal and image processing. It represents a signal using average (approximation) and difference (detail) components at different scales.***
  It is mainly used for data compression, noise reduction, and feature extraction, especially in images.

☞ ***Haar Transform splits data into low-frequency (smooth part) and high-frequency (detail part) information.***

The Haar transform uses step (square) functions as basis functions, unlike Fourier transform which uses sine and cosine waves.



Example of 1D Haar Transform (8 Pixel Values)

Consider the following **1D array of 8 pixel values**:

$$X = [8,\ 6,\ 4,\ 2,\ 10,\ 12,\ 14,\ 16]$$

Step-1: Pairwise Average and Difference

Haar Transform uses:

$$\text{Average} = \frac{a+b}{2}, \quad \text{Difference} = \frac{a-b}{2}$$

Pair	Values	Average	Difference
1	(8, 6)	7	1
2	(4, 2)	3	1
3	(10, 12)	11	−1
4	(14, 16)	15	−1

Step-2: Haar Transform Output (Level-1)

$$H_1 = [7,\ 3,\ 11,\ 15,\ 1,\ 1,\ -1,\ -1]$$

- **First half** → Approximation coefficients
- **Second half** → Detail coefficients

Step-3: Further Decomposition (Optional – Full Haar)

Apply Haar again on approximation part $[7, 3, 11, 15]$:

Second Level

Pair	Values	Avg	Diff
1	(7, 3)	5	2
2	(11, 15)	13	−2

$$H_2 = [5,\ 13,\ 2,\ -2]$$

Third Level (Final)

Apply Haar on $[5, 13]$:

- Avg = 9
- Diff = -4

Final Haar Transform Coefficients

$$\boxed{[9,\ -4,\ 2,\ -2,\ 1,\ 1,\ -1,\ -1]}$$

Applications of Haar Transform

- Image compression (JPEG2000 – wavelet based)
- Edge detection
- Noise removal
- Feature extraction
- Fast computation (very simple and efficient)

Haar Transform is a wavelet transform that decomposes a signal into averages and differences, representing both spatial and frequency information efficiently.

Discrete Cosine Transform (DCT)

- ***The Discrete Cosine Transform (DCT) is a mathematical transform that converts a discrete signal or image from the spatial domain into the frequency domain using only cosine functions.***
- ***It concentrates most of the signal energy into few low-frequency coefficients, which makes it very useful for image and video compression.***
- ***👉 In image processing, DCT separates an image into low-frequency (smooth areas) and high-frequency (edges and fine details) components.***

Why Only Cosine?

Cosine functions are even and real, which makes DCT:

- More energy-efficient
- Better for compression
- Simpler than Fourier Transform (no sine terms)

Mathematical Definition (1D DCT)

For a signal $f(x)$ of length N:

$$F(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) \cos\left[\frac{\pi(2x+1)u}{2N}\right]$$

where

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}}, & u = 0 \\ \sqrt{\frac{2}{N}}, & u > 0 \end{cases}$$

Applications of DCT

- JPEG image compression
- MPEG video compression
- Image enhancement
- Pattern recognition
- Signal processing

u is the frequency index of spatial index x.

Consider a **4-point signal**:

$$f = [10,\ 12,\ 14,\ 16]$$

DCT Result (approximate):

$$F = [26,\ -4.46,\ 0,\ -0.32]$$

Discrete Cosine Transform converts spatial data into frequency components using cosine functions, concentrating most energy in low-frequency coefficients for efficient compression.

- **F(0) = 26** → DC coefficient (average / brightness)
- Remaining values → AC coefficients (details)

👉 Most energy is concentrated in the **first coefficient**, which proves why DCT is good for compression.

1 4-Point 1D DCT Matrix

For **N = 4**, the DCT can be written as:

$$\mathbf{C} = \mathbf{D}\,\mathbf{x}$$

where

- **x** = input vector
- **C** = DCT coefficients
- **D** = DCT matrix

$$\mathbf{D} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ 0.653 & 0.271 & -0.271 & -0.653 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.271 & -0.653 & 0.653 & -0.271 \end{bmatrix}$$

2 Input 1D Signal (4 pixel values)

$$\mathbf{x} = \begin{bmatrix} 8 \\ 6 \\ 4 \\ 2 \end{bmatrix}$$

3 Matrix Multiplication

$$\mathbf{C} = \mathbf{D}\mathbf{x}$$

First coefficient (DC term)

$$C(0) = \frac{1}{2}(8 + 6 + 4 + 2) = \frac{1}{2}(20) = 10$$

Second coefficient

$$C(1) = (0.653)(8) + (0.271)(6) - (0.271)(4) - (0.653)(2)$$

$$= 5.224 + 1.626 - 1.084 - 1.306$$

$$= 4.46$$

Third coefficient

$$C(2) = 0.5(8) - 0.5(6) - 0.5(4) + 0.5(2)$$

$$= 4 - 3 - 2 + 1 = 0$$

Fourth coefficient

$$C(3) = 0.271(8) - 0.653(6) + 0.653(4) - 0.271(2)$$

$$= 2.168 - 3.918 + 2.612 - 0.542$$

$$= 0.32$$

4 Final DCT Output

$$\boxed{\mathbf{C} = \begin{bmatrix} 10 \\ 4.46 \\ 0 \\ 0.32 \end{bmatrix}}$$

1. Differentiate between Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT). Explain the FFT algorithm for one-dimensional case. **[3+7=10]**

6. Describe in brief that how do you implement Gaussian High Pass Frequency domain filter for image smoothing in the frequency domain?

2. What is Fourier Transform and how can you apply it in the digital image processing? Explain the different properties of the Fourier Transform.(4+6)

2. Define Discrete Cosine Transform(DCT). Differentiate between Discrete Fourier Transform (DFT) and Fast Fourier Transform(FFR).

7. How will you implement Butterworth high Pass Frequency domain filter for image sharpening in the frequency domain? Describe in brief.