

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	Logarithmic, 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.

Spatial Domain vs Frequency Domain

- **Spatial Domain:**

- The spatial domain refers to the use of an image's space coordinates (x, y) directly for processing. In this domain, each point in an image retains its position and amplitude value (such as brightness or color). Most common image manipulations like brightness adjustment, contrast enhancement, and cropping are done in the spatial domain.
- In simpler terms, the spatial domain represents an image exactly as it appears visually. If you were to manipulate an image in the spatial domain, you might adjust the pixels directly based on their location.

- **Frequency Domain:**

- The frequency domain converts the spatial representation of the image into frequencies that make up the image. This conversion is typically done using mathematical transformations such as the Fourier Transform. The frequency domain representation of an image shows how often certain tones, textures, and patterns occur across the space of the image.
- Processing in the frequency domain is useful for operations that are complex or inefficient in the spatial domain, like filtering or compression. For example, removing noise or enhancing specific details of an image might be more effectively handled in the frequency domain.

Spatial Domain Representation



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

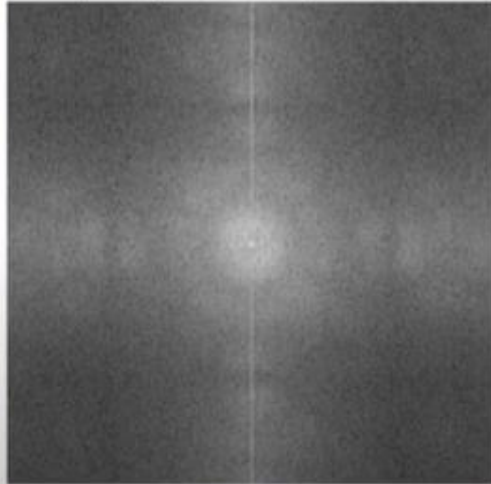


Figure 5 × Figure 6 × Figure 7 × Figure 8 × Figure 9 × Figure 10 × Figure 11 × im × im_gray × F_mag_log

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

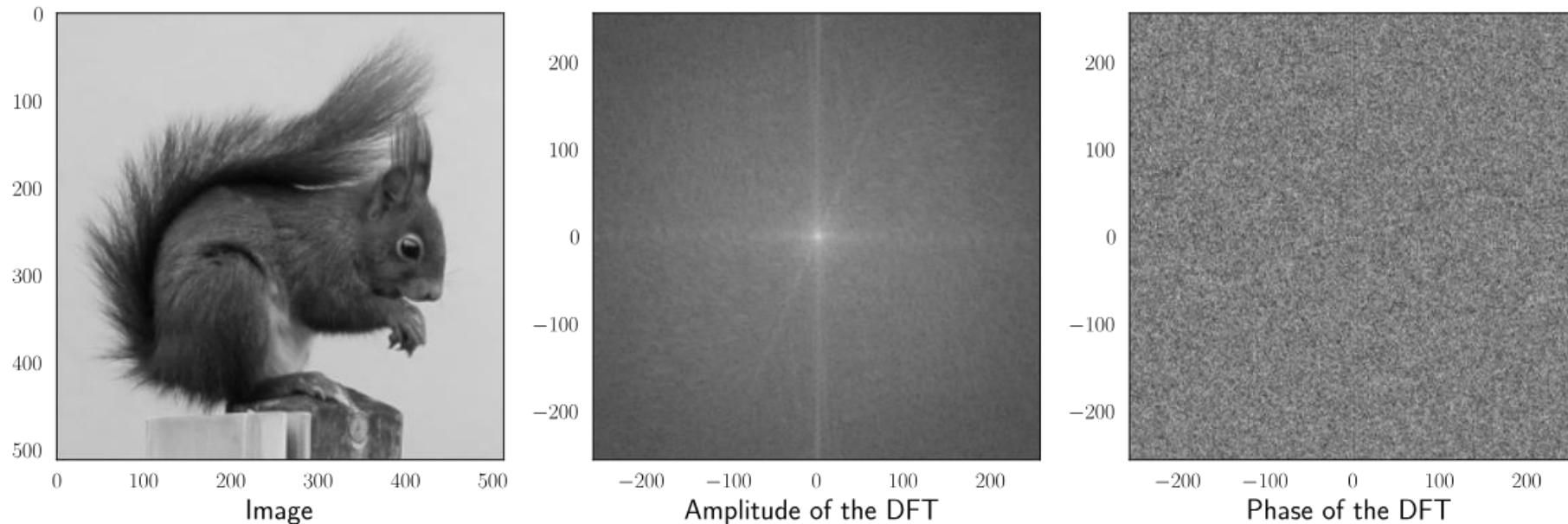
Image Enhancement in Frequency Domain

- For image enhancement, the image from spatial domain is taken to frequency domain and then processed. After that inverse transform is applied to bring back to spatial domain.
- In frequency domain, filters are applied to smoothening and Sharpening of image, by removing high or low frequencies.
- Once filter is applied, the changes will be done in whole image. In spatial domain, the change was one pixel by pixel.
- There are two types of filters i.e. low pass filter and high pass filter.
- Low pass filter smoothenes the images by removing high frequency components in image while high pass filter sharpens the image by removing low frequency components in images.
- Smoothening the images means blurring the image and removing the noise while sharpening the image means removing most of the backgrounds.
- Both Low pass and high pass filter can be of three types:
 1. Ideal Filter
 2. Butterworth Filter and
 3. Gaussian Filter

Fourier Transform in Image Processing

- The Fourier Transform is a powerful tool in image processing for analyzing, processing, and enhancing images based on their frequency content. Its applications range from basic image enhancements to more complex operations such as image compression, watermarking, and pattern recognition.
- The Fourier Transform is a mathematical technique that is widely used in image processing to analyze the frequency components of an image. Essentially, **it transforms a spatial domain image into the frequency domain**. This process involves decomposing an image into its sine and cosine components. The primary reason for applying the Fourier Transform in image processing is its ability to characterize image features by frequency, rather than by spatial location.
- Here are some key points about the Fourier Transform in image processing:
 - 1. Frequency Domain Representation:** The Fourier Transform converts an image from its spatial domain to the frequency domain, where each point represents a particular frequency contained in the spatial domain image. Low frequencies correspond to the slow-changing, smooth areas of the image, while high frequencies correspond to the fast-changing, detailed parts.
 - 2. Filtering:** One of the main applications of the Fourier Transform in image processing is in filtering. For example, a high-pass filter can be used to enhance the edges of an image by allowing high-frequency components to pass through while blocking low-frequency components. Conversely, a low-pass filter can help in smoothing the image by removing high-frequency noise.

3. **Image Compression:** Fourier Transform is also useful in image compression techniques. By transforming an image into the frequency domain, it's possible to identify and eliminate frequencies that have minimal impact on the visual quality of the image, thus reducing the image size without significant loss of quality.
4. **Pattern Recognition and Analysis:** In the frequency domain, certain patterns or characteristics of the image might be more easily identified than in the spatial domain, making the Fourier Transform a valuable tool in pattern recognition and image analysis applications.
5. **Phase and Magnitude:** The output of the Fourier Transform consists of a complex number for each frequency, containing both a magnitude (the amount of the frequency present) and a phase (the orientation of the sine wave associated with the frequency). The magnitude spectrum often provides insight into the texture and composition of the image, while the phase spectrum carries information about the spatial arrangement of the image's components.



$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\omega) e^{j\omega t} d\omega$$

$$\cos(\omega t) + j \sin(\omega t)$$

$$X(j\omega_1) \times \left(\text{cosine wave} \rightarrow t + j \text{sine wave} \rightarrow t \right)$$

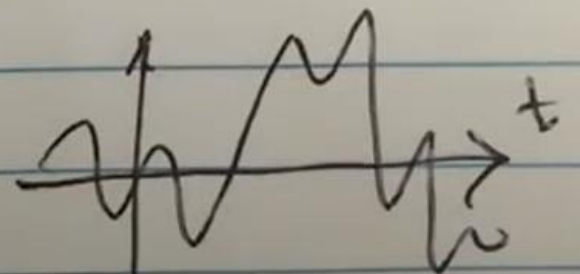
⋮

⋮

⋮

$$X(j\omega_2) \times \left(\text{cosine wave} \rightarrow t + j \text{sine wave} \rightarrow t \right)$$

$$+ = x(t)$$



1D/2D Continuous and Discrete Fourier Transform

- The Fourier Transform (FT) is a mathematical operation that transforms a signal from its original domain (often time or space) into a representation in the frequency domain. The FT comes in various forms depending on the nature of the signal (continuous or discrete) and its dimensionality (1D or 2D). Understanding these different forms is crucial in fields ranging from signal processing to image analysis.

1D Continuous Fourier Transform

The 1D Continuous Fourier Transform is used for continuous signals, which are functions of a continuous variable (like time). It's defined by the integral:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$$

- $f(t)$ is the original continuous signal in the time domain.
- $F(\omega)$ is the Fourier Transform of $f(t)$, representing the signal in the frequency domain.
- ω is the angular frequency.

1D Discrete Fourier Transform

The 1D Discrete Fourier Transform (DFT) is used for sequences of samples typically obtained by sampling a continuous signal. It's defined by the sum:

$$F(k) = \sum_{n=0}^{N-1} f(n) e^{-j \frac{2\pi}{N} kn}$$

- $f(n)$ is the discrete signal in the time domain.
- $F(k)$ is the DFT of $f(n)$, representing the signal in the frequency domain.
- N is the total number of samples.
- k is the index in the frequency domain corresponding to a specific frequency.

2D Continuous Fourier Transform

The 2D Continuous Fourier Transform is applied to continuous functions of two variables, typically spatial variables like x and y for images. It's defined as:

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

- $f(x, y)$ is the continuous two-dimensional signal.
- $F(u, v)$ is its Fourier Transform, with u and v being spatial frequencies in the x and y directions, respectively.

2D Discrete Fourier Transform

The 2D Discrete Fourier Transform is used for discrete two-dimensional signals, such as digital images. It's defined as:

$$F(u, v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) e^{-j2\pi(\frac{um}{M} + \frac{vn}{N})}$$

- $f(m, n)$ is the discrete two-dimensional signal (e.g., a digital image).
- $F(u, v)$ is the DFT of $f(m, n)$, representing the image in the frequency domain.
- M and N are the dimensions of the image (height and width, respectively).
- u and v are indices in the frequency domain.

Each version of the Fourier Transform provides a framework for analyzing signals or images in terms of their frequency content, which is invaluable for applications like signal filtering, image compression, and feature extraction. The choice between these forms depends on the nature of the data (continuous vs. discrete) and its dimensionality (1D vs. 2D).

Properties of Fourier Transform

Fourier transform is the input tool that is used to decompose an image into its sine and cosine components. Following are the properties of Fourier Transform.

- **Linearity:**

Addition of two functions corresponding to the addition of the two frequency spectrum is called the linearity. If we multiply a function by a constant, the Fourier transform of the resultant function is multiplied by the same constant. The Fourier transform of sum of two or more functions is the sum of the Fourier transforms of the functions.

Case I.

If $h(x) \rightarrow H(f)$ then $ah(x) \rightarrow aH(f)$

Case II.

If $h(x) \rightarrow H(f)$ and $g(x) \rightarrow G(f)$ then $h(x)+g(x) \rightarrow H(f)+G(f)$

1. Linearity:

$$x_1(t) \rightleftharpoons X_1(j\omega)$$

$$x_2(t) \rightleftharpoons X_2(j\omega)$$

$$\alpha x_1(t) \rightleftharpoons \alpha X_1(j\omega)$$

$$\beta x_2(t) \rightleftharpoons \beta X_2(j\omega)$$

$$\underbrace{\alpha x_1(t) + \beta x_2(t)}_{x(t)} \rightleftharpoons \alpha X_1(j\omega) + \beta X_2(j\omega)$$

proof:-

$$\begin{aligned} x(t) &\rightleftharpoons X(j\omega) \\ X(j\omega) &= \int_{-\infty}^{\infty} x(t) \cdot e^{-j\omega t} dt \\ &= \int_{-\infty}^{\infty} [\alpha x_1(t) + \beta x_2(t)] \cdot e^{-j\omega t} dt \\ &= \underbrace{\alpha \int_{-\infty}^{\infty} x_1(t) \cdot e^{-j\omega t} dt}_{X_1(j\omega)} + \underbrace{\beta \int_{-\infty}^{\infty} x_2(t) \cdot e^{-j\omega t} dt}_{X_2(j\omega)} \\ X(j\omega) &= \alpha X_1(j\omega) + \beta X_2(j\omega) \end{aligned}$$

2: Conjugation:

$$\begin{aligned} x(t) &\Longleftrightarrow X(j\omega) \\ \check{**} \quad x^*(t) &\Longleftrightarrow X^*(-j\omega) \end{aligned}$$

The term "conjugation" in image processing typically refers to the concept of the "complex conjugate" in the context of Fourier transforms, which are a crucial tool in many image processing techniques. When dealing with complex numbers, the complex conjugate of a number is formed by changing the sign of the imaginary part. This concept is widely used in signal and image processing, particularly in the manipulation and analysis of Fourier transforms.

Fourier transforms decompose an image into its frequency components, and the complex conjugate plays a role in operations such as filtering, compression, and reconstruction of images.

Proof :-

$$x(t) \Longleftrightarrow X(j\omega)$$

$$X(j\omega) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j\omega t} dt$$

$$X^*(j\omega) = \int_{-\infty}^{\infty} x^*(t) \cdot e^{j\omega t} dt$$

$$\overset{\omega \rightarrow -\omega}{X^*(-j\omega)} = \int_{-\infty}^{\infty} x^*(t) \cdot e^{-j\omega t} dt$$

$$x^*(t) \Longleftrightarrow X^*(-j\omega)$$

3. Area Under Time Domain:

$$\text{Area under } \underline{x(t)} = \int_{-\infty}^{\infty} \underline{x(t)} dt$$

$$X(j\omega) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j\omega t} dt$$

$$\downarrow \omega=0$$

$$X(0) = \int_{-\infty}^{\infty} x(t) dt = \text{area under } x(t)$$

$$** \checkmark \text{ area under } x(t) = X(j\omega)|_{\omega=0}$$

4. Area Under Frequency Domain:

$$\text{Area under } \underline{X(j\omega)} = \int_{-\infty}^{\infty} \underline{X(j\omega)} d\omega$$

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\omega) \cdot e^{j\omega t} d\omega$$

$$\downarrow t=0$$

$$x(0) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\omega) d\omega$$

$$\underline{2\pi x(0)} = \int_{-\infty}^{\infty} X(j\omega) d\omega = \text{area under } X(j\omega)$$

$$* \checkmark \text{ area under } X(j\omega) = 2\pi x(t)|_{t=0}$$

Scaling:

Scaling is the method that is used to change the range of the independent variables or features of data. If we stretch a function by the factor in the time domain then squeeze the Fourier transform by the same factor in the frequency domain.

$$\text{If } f(t) \rightarrow F(w) \text{ then } f(at) \rightarrow (1/|a|)F(w/a)$$

Differentiation:

Differentiating function with respect to time yields to the constant multiple of the initial function.

$$\text{If } f(t) \rightarrow F(w) \text{ then } f'(t) \rightarrow jwF(w)$$

Convolution:

It includes the multiplication of two functions. The Fourier transform of a convolution of two functions is the point-wise product of their respective Fourier transforms.

$$\begin{aligned} \text{If } f(t) &\rightarrow F(w) \text{ and } g(t) \rightarrow G(w) \\ \text{then } f(t)*g(t) &\rightarrow F(w)*G(w) \end{aligned}$$

Frequency Shift:

Frequency is shifted according to the co-ordinates. There is a duality between the time and frequency domains and frequency shift affects the time shift.

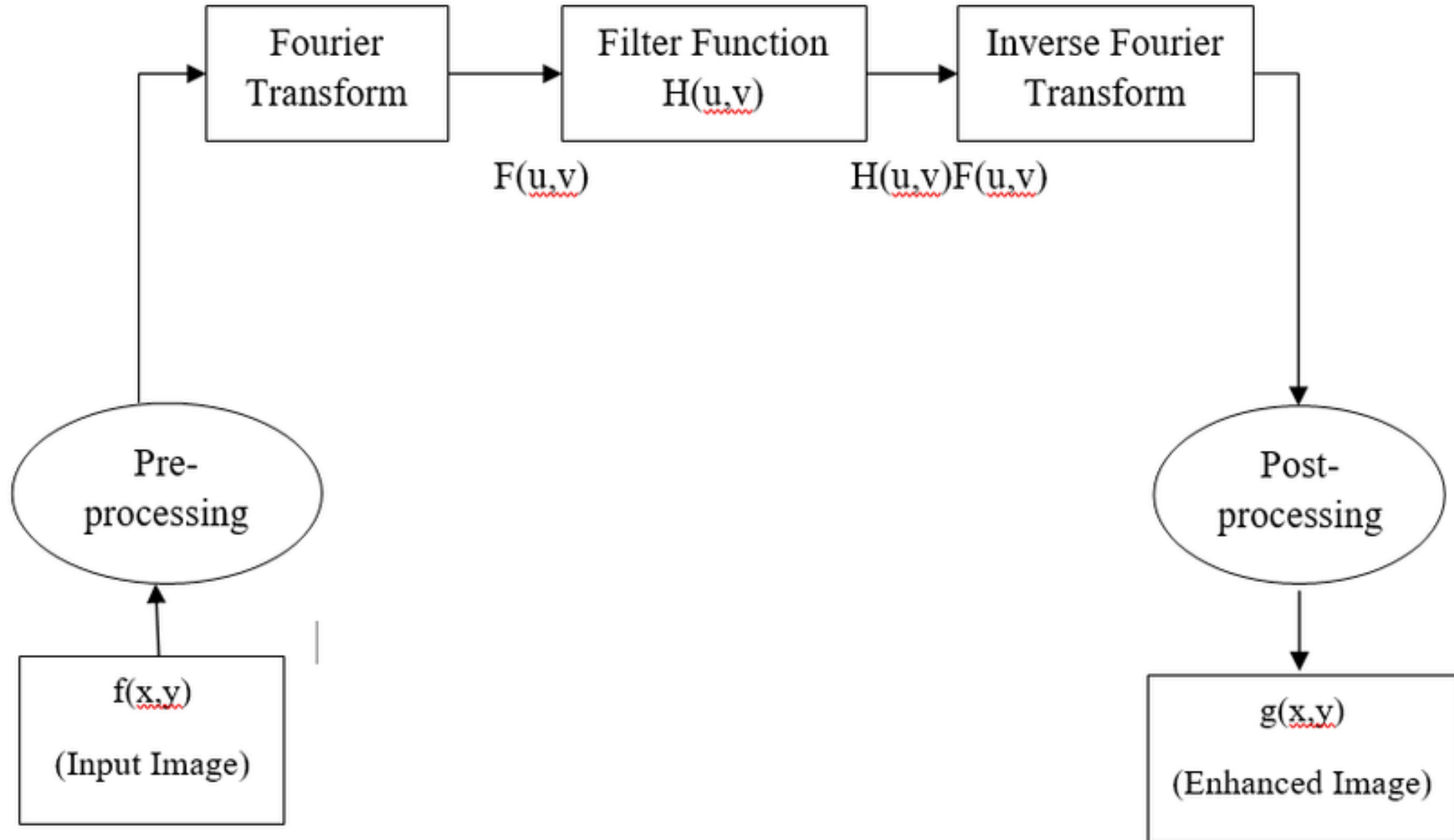
$$\text{If } f(t) \rightarrow F(\omega) \text{ then } f(t)\exp[j\omega't] \rightarrow F(\omega - \omega')$$

Time Shift:

The time variable shift also effects the frequency function. The time shifting property concludes that a linear displacement in time corresponds to a linear phase factor in the frequency domain.

$$\text{If } f(t) \rightarrow F(\omega) \text{ then } f(t - t') \rightarrow F(\omega)\exp[-j\omega t']$$

Steps for filtering in frequency domain



$f(x, y)$

1. Image $f(x, y)$ of size $M \times N$

2. $f(x, y) (-1)^{x+y}$

3. $F(u, v) \xrightarrow{\text{DFT}} \text{F.T. of } f(x, y)$

4. $H(u, v) \rightarrow \text{Filter in freq domain.}$

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

5. $\underline{g(x, y)} = \mathcal{F}^{-1}[G(u, v)] (-1)^{x+y}.$

Image Smoothing filters

We can achieve smoothing in frequency domain through high-frequency attenuation (lowpass filtering). Low-pass filters have a multitude of applications in various fields. Here are some common ones:

- 1.Audio Processing:** Low-pass filters are widely used in audio processing to remove high-frequency noise or to simulate the effect of distance (since high-frequency sounds diminish over distance).
- 2.Data Communication:** In data communication systems, these filters are used to limit the bandwidth of the transmitted signal, reducing interference.
- 3.Image Processing:** In digital image processing, low-pass filters are used to blur images or to reduce high-frequency noise.
- 4.Electronics:** They are used in electronic circuits to block high-frequency noise in power supplies.

The 3 types of lowpass filters

1. Ideal Low Pass Filter
2. Butterworth Lowpass Filters
3. Gaussian Lowpass Filters

1. Ideal Low Pass Filter

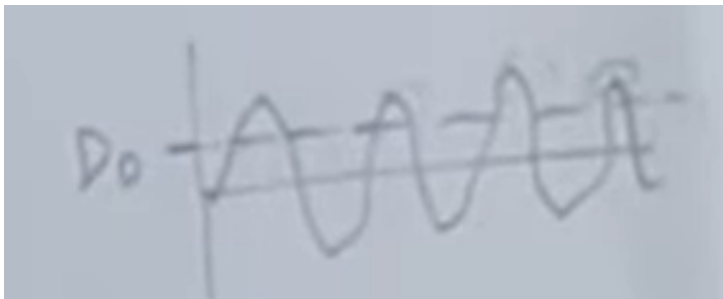
The ideal lowpass filter is used to cut off all the high-frequency components of Fourier transformation. Below is the transfer function of an ideal lowpass filter.

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

$$D(u, v) = \left[\left(u - \frac{M}{2} \right)^2 + \left(v - \frac{N}{2} \right)^2 \right]^{\frac{1}{2}}$$



ideal lowpass



The problem with an ideal high pass filter or low pass filter is that the edges are blurred. It means that we cannot differentiate between the background and the image end.

2. Butterworth Low Pass Filter

Butterworth Lowpass Filter is used to remove high-frequency noise with very minimal loss of signal components.

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



Butterworth Lowpass

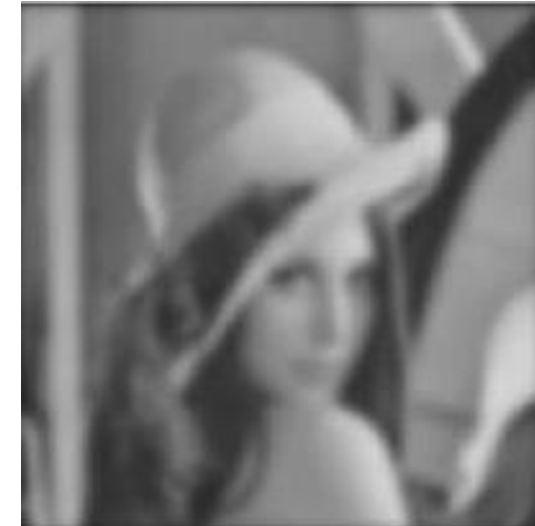
The value of n should be kept less value for clear edges. As the value of n increases, it becomes more to ideal low pass filter.

3. Gaussian Low Pass Filter

The transfer function of Gaussian Lowpass filters is shown below:

$$H(u, v) = e^{-D^2(u, v) / 2D_0^2} \quad (8)$$

The advantage is that it reduces low frequency noise



Gaussian Lowpass

Image Sharpening filters

- A highpass filter is used for passing high frequencies but the strength of the frequency is lower as compared to cut off frequency.
- Sharpening is a highpass operation in the frequency domain. As lowpass filter, it also has standard forms such as Ideal highpass filter, Butterworth highpass filter, Gaussian highpass filter.



Original image



Gaussian highpass



Ideal highpass



Butterworth highpass

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

Implementing a Gaussian High Pass Filter for image smoothing in the frequency domain involves several steps. The Gaussian High Pass Filter is actually used for image sharpening by attenuating the low-frequency components more than the high-frequency components of the image. Here's a brief outline of how to implement it:

1. Convert the Image to Frequency Domain:

- Start by loading the image and convert it to grayscale if it is not already.
- Apply a Fourier Transform (using functions like FFT) to the image to convert it from the spatial domain to the frequency domain.

2. Create a Gaussian High Pass Filter:

- Determine the size of the image and create a two-dimensional array for the filter with the same dimensions.
- For each element in the filter array, calculate the distance from the center of the image. The Gaussian function for the High Pass Filter can be defined as $1 - e^{-\frac{d^2}{2\sigma^2}}$, where d is the distance from the center and σ is the standard deviation that controls the spread of the filter.
- Normalize the filter so that the filter values are spread between 0 and 1.

3. Apply the Filter in Frequency Domain:

- Multiply the Fourier Transformed image by the Gaussian High Pass filter. This step attenuates the low frequencies more and keeps the high frequencies, effectively sharpening the image.

4. Inverse Fourier Transform:

- Apply the Inverse Fourier Transform to the filtered image to convert it back to the spatial domain.

5. Post-processing:

- The resulting image might require normalization or scaling to display it properly as the final output can have values outside the usual display range.

This process enhances the edges and fine details in an image, which is the opposite of smoothing. If you were aiming for image smoothing, you might consider using a Gaussian Low Pass Filter instead, which softens the image by reducing high-frequency components.

How will you implement Butterworth High pass frequency domain filter for image sharpening in the frequency domain? Describe in brief.

Implementing a Butterworth High Pass Filter for image sharpening in the frequency domain follows a structured series of steps, similar to implementing Gaussian filters but with a different mathematical model for the filter itself. Here's how you can go about it:

1. Convert the Image to Frequency Domain:

- Load the image and convert it to grayscale if necessary.
- Apply the Fast Fourier Transform (FFT) to convert the spatial domain image into the frequency domain.

2. Create a Butterworth High Pass Filter:

- Determine the dimensions of the image and create a two-dimensional filter matrix of the same size.
- Calculate the distance $d(u, v)$ from the center of the frequency matrix to each point (u, v) in the matrix.
- The Butterworth High Pass Filter (BHPF) formula is given by $H(u, v) = 1 - \frac{1}{1 + \left(\frac{D_0}{d(u, v)}\right)^{2n}}$,

where:

- D_0 is the cutoff frequency, determining how much of the frequency content is cut off.
- n is the order of the filter, controlling the sharpness of the transition between passed and attenuated frequencies.
- $d(u, v)$ is the distance from the center of the frequency plane to each point (u, v) .
- Elements closer to the center (low frequencies) are more attenuated, and those further away are less attenuated (high frequencies).

3. Apply the Filter in Frequency Domain:

- Multiply the FFT of the image by the Butterworth High Pass filter matrix. This operation enhances the high frequencies while attenuating the low frequencies, thereby sharpening the image.

4. Inverse Fourier Transform:

- After applying the filter, convert the image back from the frequency domain to the spatial domain using the Inverse FFT (IFFT).

5. Post-processing:

- The output image might need adjustments such as normalization or scaling to bring the pixel values back into an appropriate range for display.

Fast Fourier Transform

- The Fourier Transform is a mathematical operation that decomposes a time-domain signal into its constituent frequencies. In essence, it converts a waveform into a representation in the frequency domain, highlighting the amplitude and phase of different frequency components. This technique is invaluable across various scientific disciplines, providing insights into the underlying characteristics of complex signals.
- However, the conventional Fourier Transform can be computationally intensive, especially for signals with numerous data points. This is where the FFT steps in as a game-changer. The FFT algorithm streamlines the computation of the Fourier Transform by exploiting symmetries in the mathematical calculations. This results in a significant reduction in computational complexity, making it an order of magnitude faster than the standard Fourier Transform for large datasets
- **FFT's Significance:** In signal processing, FFT swiftly computes the Fourier Transform, which is vital for understanding frequency components. It enhances performance in RF circuits and electronics.
- **Efficient Computation:** FFT exploits symmetries to simplify calculations, significantly reducing complexity compared to standard Fourier Transform.

FFT Advantages Over Standard Fourier Transform

The FFT algorithm streamlines the computation of the Fourier Transform by exploiting symmetries in the mathematical calculations. This significantly reduces computational complexity, making it an order of magnitude faster than the standard Fourier Transform for large datasets.

- **Computational Complexity:** The computational complexity of the traditional Fourier Transform is $O(N^2)$, where N is the number of samples in the input signal. As the number of samples increases, the computation time grows quadratically. The FFT algorithm dramatically reduces the computational complexity to $O(N \log N)$, a significant improvement.
- **Divide and Conquer Approach:** The traditional approach calculates each frequency component of the Fourier Transform independently, summing up the contributions of each sample for every frequency. The FFT algorithm employs a divide-and-conquer strategy to compute the Fourier Transform efficiently. It decomposes the input sequence into smaller subproblems, recursively computes their Fourier Transforms, and then combines these results to obtain the final transform.
- **Complexity of Trigonometric Calculations:** The traditional approach involves a lot of trigonometric calculations for computing sine and cosine functions, which can be computationally expensive. The FFT algorithm optimizes these trigonometric calculations by using properties of complex numbers and symmetry in the data. This further contributes to the speedup in computation.
- **Memory Access Patterns:** The traditional approach involves accessing the input data scattered, which might not be cache-friendly and can slow down the computation. The FFT algorithm is designed to use memory more efficiently and take advantage of cache-friendly access patterns, further enhancing its speed.

Feature	Fourier Transform (FT)	Fast Fourier Transform (FFT)
Definition	A mathematical transform used to convert a signal from its original domain (often time or space) into the frequency domain.	An algorithm designed to efficiently compute the Discrete Fourier Transform (DFT) of a sequence, or its inverse.
Computational Complexity	Depends on the method of calculation, but the direct calculation from the definition has a computational complexity of $O(N^2)$ for N data points.	Significantly reduces computational complexity to $O(N\log N)$, making it much faster for computing the DFT of large sequences.
Type of Signals	Applicable to both continuous and discrete signals, but in practice, numerical methods are used to compute FT for discrete signals.	Specifically designed for discrete signals and sequences, making it ideal for digital signal processing.
Application Domain	Theoretical and practical applications across various fields including signal processing, physics, and electrical engineering.	Primarily used in digital signal processing, data compression, and in fields that require fast and efficient computation of Fourier transforms.
Purpose	Provides a theoretical foundation for understanding and analyzing signals in the frequency domain.	Provides a practical and efficient means for computing the transformation of signals into the frequency domain, especially for digital applications.
Efficiency	Less efficient computationally when directly implemented for large data sets.	Highly efficient, especially for large datasets or where computational resources are limited.
Implementation	Theoretical analysis and numerical methods are often required for its computation.	Directly implemented using algorithms in software and hardware for fast computation.

The FFT Algorithm - Simple Step by Step

START

SPLIT

SPLIT

SPLIT

SPLIT

SPLIT

COMBINE

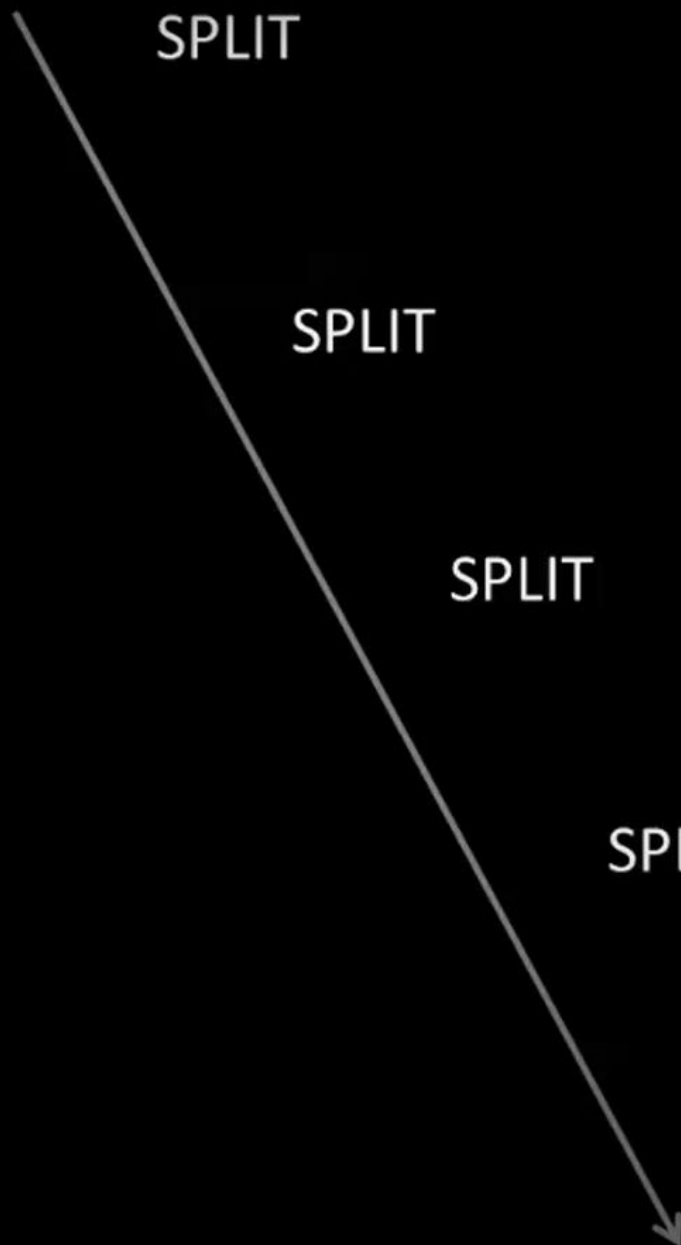
COMBINE

COMBINE

COMBINE

COMBINE

FINISH



```
vector<complex<double>> FFT(vector<complex<double>> &samples) {  
    // find the number of samples we have  
    int N = samples.size();  
  
    // Execute the end of the recursive even/odd splits once we only  
    one sample  
    if(N==1) {return samples;}  
  
    /* Split the samples into even and odd subsums */  
    // Find half the total number of samples  
    int M = N/2;  
  
    // Declare an even and an odd complex vector  
    vector<complex<double>> Xeven(M, 0);  
    vector<complex<double>> Xodd(M, 0);
```

```
// Input the even and odd samples into respective vectors
for(int i=0;i!=M;i++)
{
    Xeven[i]=samples[2*i];
    Xodd[ i]=samples[2*i+1];
}

// Perform the recursive FFT operation on the odd and even sides
vector<complex<double>> Feven(M,0) ;
Feven = FFT(Xeven);
vector<complex<double>> Fodd(M,0) ;
Fodd = FFT(Xodd);
```

```

/*----- END RECURSION -----*/
// Declare vector of frequency bins
vector<complex<double>> freqbins(N,0);
// Combine the values found
for(int k=0;k!=N/2;k++)
{
    // For each split set, we need to multiply a k-dependent complex
    // number by the odd subsum
    complex<double> cmplxexponential = polar(1.0,-2*pi*k/N)*Fodd[k];
    freqbins[k] = Feven[k]+cmplxexponential;

    // Everytime you add pi, exponential changes sign
    freqbins[k+N/2] = Feven[k]-cmplxexponential;
}

return freqbins;
}

```


FAST FOURIER TRANSFORM

$$\text{frequency bin } F_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{j2\pi kn}{N}}$$

k: 0, ..., # of samples *n: 0, ..., # of samples*

(1 operation) 1 sample: $F_0 = x_0 * \text{exponential}$

(4 operations) 2 samples: $F_0 = x_0 * \text{exponential} + x_1 * \text{exponential}$
 $F_1 = x_0 * \text{exponential} + x_1 * \text{exponential}$

(9 operations) 3 samples: $F_0 = x_0 * \text{exponential} + x_1 * \text{exponential} + x_2 * \text{exponential}$
 $F_1 = x_0 * \text{exponential} + x_1 * \text{exponential} + x_2 * \text{exponential}$
 $F_2 = x_0 * \text{exponential} + x_1 * \text{exponential} + x_2 * \text{exponential}$

Complexity: N^2

0-20000 Hz, 1 Hz spacing

$20000^2 = 400$ million operations


≈ 760 years

Complexity: $N \log_2 N$

0-20000 Hz, 1 Hz spacing

$20000 * \log_2 20000 = 28600$ operations

≈ 198 days

$$F_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{j2\pi kn}{N}}$$

$$F_k = \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-\frac{j2\pi k(2m)}{N}} + \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-\frac{j2\pi k(2m+1)}{N}}$$

x_0, x_2, x_4, \dots x_1, x_3, x_5, \dots

EVEN INDEX **ODD INDEX**

$$F_k = \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-\frac{j2\pi km}{N/2}} + C_k \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-\frac{j2\pi km}{N/2}}$$

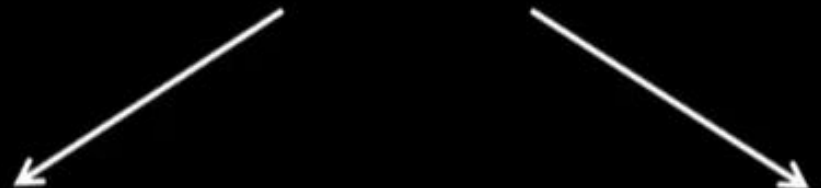
SYMMETRY IDENTITY

$$\cos\left(-\frac{2\pi km}{N/2}\right) = \cos\left(-\frac{2\pi(N/2 + k)m}{N/2}\right)$$

$$\sin\left(-\frac{2\pi km}{N/2}\right) = \sin\left(-\frac{2\pi(N/2 + k)m}{N/2}\right)$$

$$e^{-\frac{j2\pi km}{N/2}} = e^{-\frac{j2\pi(N/2 + k)m}{N/2}}$$

$$F_k = \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-\frac{j2\pi km}{N/2}} + C_k \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-\frac{j2\pi km}{N/2}}$$



Two white arrows point from the first and second terms of the equation above to the two terms of the equation below, illustrating the decomposition of the DFT into two parallel summations.

$$\sum_{m=0}^{N/4-1} x_{4m} \cdot e^{-\frac{j2\pi km}{N/4}} + e^{\frac{-j4\pi k}{N}} \sum_{m=0}^{N/4-1} x_{4m+2} \cdot e^{-\frac{j2\pi km}{N/4}}$$

Every “split level” halves the number of operations to calculate the DFT

$$\sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-\frac{j2\pi km}{N/2}}$$

$$e^{-\frac{j2\pi k}{N}} \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-\frac{j2\pi km}{N/2}}$$

$$\sum_{m=0}^{N/4-1} x_{4m} \cdot e^{-\frac{j2\pi km}{N/4}}$$

$$e^{-\frac{4\pi k}{N}} \sum_{m=0}^{N/4-1} x_{4m+2} \cdot e^{-\frac{j2\pi km}{N/4}}$$

$$\sum_{m=0}^{N/4-1} x_{4m+1} \cdot e^{-\frac{j2\pi km}{N/4}}$$

$$e^{-\frac{j4\pi k}{N}} \sum_{m=0}^{N/4-1} x_{4m+3} \cdot e^{-\frac{j2\pi km}{N/4}}$$

$$F_0 = x_0 + x_2 + x_1 + x_3 = 0 + 1 + 0 + (-1) = 0$$

$$F_0^e = 1$$

$$F_0^o = -1$$

Fast Fourier Transform Applications in RF and Electronics

The Fast Fourier Transform (FFT) algorithm finds extensive applications in the field of radio frequency (RF) and electronics due to its ability to efficiently analyze and process signals.

Below are some of the

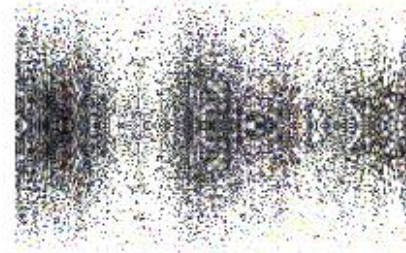
- **Spectrum Analysis:** FFT is used for identifying frequency components in RF signals, aiding in spectrum analysis, channel allocation, and interference detection.
- **Modulation and Demodulation:** FFT assists in extracting information from modulated RF signals by analyzing their frequency components.
- **Radar Systems:** In radar, FFT plays a pivotal role in pulse compression, enhancing range resolution and allowing for accurate target detection and tracking.
- **Wireless Communication:** FFT is essential in OFDM (Orthogonal Frequency Division Multiplexing) systems, enabling high-data-rate transmission by converting data into subcarriers that are orthogonal in the [frequency domain](#).
- **Signal Filtering:** FFT aids in designing and analyzing analog and digital filters for RF and electronic systems, facilitating precise frequency response shaping.

Lab 09: FFT

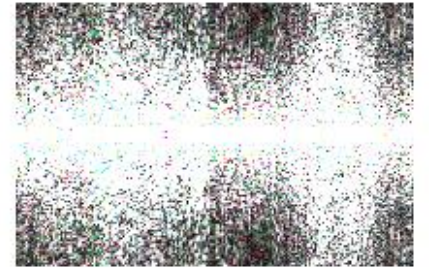
```
clear all; % clear all variables
close all; % close all figures
clc; % clear command window
% import image package
pkg load image;
% read image
l=im2double(imread('tiger.jpg'));
f1=fft(l);
f2=fftshift(f1);
subplot(2,2,1); imshow(abs(f1)); title('Frequency Spectrum');
subplot(2,2,2); imshow(abs(f2));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



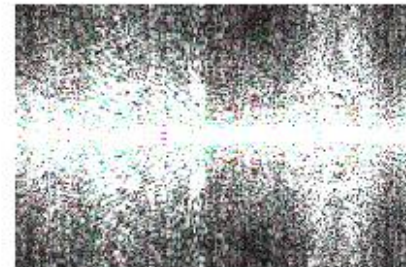
Frequency Spectrum



Centered Spectrum



log(1+abs(f2))



2-D FFT



Other Image Transforms

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

1. Hadamard Transform

- The Hadamard Transform is a mathematical transformation used in various fields, including signal processing, image processing, data compression, and quantum computing. It belongs to the group of linear orthogonal transformations, similar to the Fourier Transform, but it uses only real numbers and involves simpler arithmetic operations. The Hadamard Transform is particularly noted for its efficiency in computational algorithms and its ability to process data in binary form.
- In the context of image processing, the Hadamard Transform is used for tasks such as image compression, feature extraction, and noise reduction. It works by transforming the spatial domain of an image into a domain that represents the image in terms of its spatial frequencies. The key advantage of using the Hadamard Transform for image processing is its computational simplicity and speed, especially for images whose dimensions are powers of two. This makes it suitable for real-time image processing applications.
- The Hadamard Transform operates on data by applying a series of Hadamard matrices, which consist of 1s and -1s arranged in a specific pattern. The size of the Hadamard matrix used depends on the size of the image or the data block being processed. A common form of the Hadamard Transform is the Fast Hadamard Transform (FHT), which efficiently computes the transform using a divide-and-conquer strategy, significantly reducing the number of required computations.

- Hadamard is Symmetric and Non-Sine function.
- For 1D, $H.F = H * F$, while for 2D, $H.F = H * F * H^T$ where Transpose of H is equal to H in symmetric functions.

$$Q:] \quad \underline{1D}: f(x) = \{1, 2, 0, 3\}_{4 \times 1} \dots H^T$$

$$H = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}_{2 \times 2}$$

↖ ↘ → ①
↖ ↘ → ②

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}_{4 \times 4}$$

solⁿ:

$$F = H \cdot f$$

$$= \begin{bmatrix} \textcircled{1} & \textcircled{1} & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}_{4 \times 4} \begin{bmatrix} \textcircled{1} \\ \textcircled{2} \\ 0 \\ 3 \end{bmatrix}$$

$N=4$

$$f(x) \rightarrow F$$

$$1 + 2 + 0 + 3 = \begin{bmatrix} 6 \\ -4 \\ 0 \\ 2 \end{bmatrix}$$

2. Haar Transform

- The Haar Transform is another mathematical transformation used in signal and image processing, akin to the Hadamard and Fourier Transforms. It is particularly noted for its simplicity and efficiency in representing an image in terms of its local spatial frequencies, with applications in image compression, feature detection, and image analysis. The Haar Transform operates using Haar wavelets, which are square-shaped functions that represent different levels of detail in the image. Unlike the sinusoidal functions used in the Fourier Transform, Haar wavelets have a piecewise constant shape, making them particularly suited for detecting abrupt changes in signal or image intensity, such as edges.
- In image processing, the Haar Transform is applied to decompose an image into a set of basis functions (Haar wavelets) that vary in scale and position, allowing for the analysis or compression of the image at different resolutions. This transform is capable of quickly identifying areas of high contrast in an image while being computationally efficient, especially for binary and grayscale images.

Example:

Consider a simple grayscale image represented by a 1D array of 8 pixel values:

$$P = [10, 20, 30, 40, 50, 60, 70, 80]$$

The Haar Transform would proceed by averaging pairs of pixel values to represent the general trend (approximation) and subtracting them to capture the detail (differences). This process reduces the dimensionality of the data at each step, which is a desirable property for compression and feature extraction.

1.

Averaging and Differencing: Starting with the original pixel values, we calculate the averages and differences:

- Averages: $[15, 35, 55, 75]$ from pairs $(10, 20)$, $(30, 40)$, $(50, 60)$, $(70, 80)$
- Differences: $[-5, -5, -5, -5]$ calculated as the difference of each pair.

2.

Repeating the Process: The process is then repeated for the averaged values, leading to a further reduced representation:

- New averages: $[25, 65]$ from pairs $(15, 35)$, $(55, 75)$
- New differences: $[-10, -10]$

3.

Final Step: Apply the process one last time to get the final averaged value and its difference:

- Final average: $[45]$
- Final difference: $[-20]$

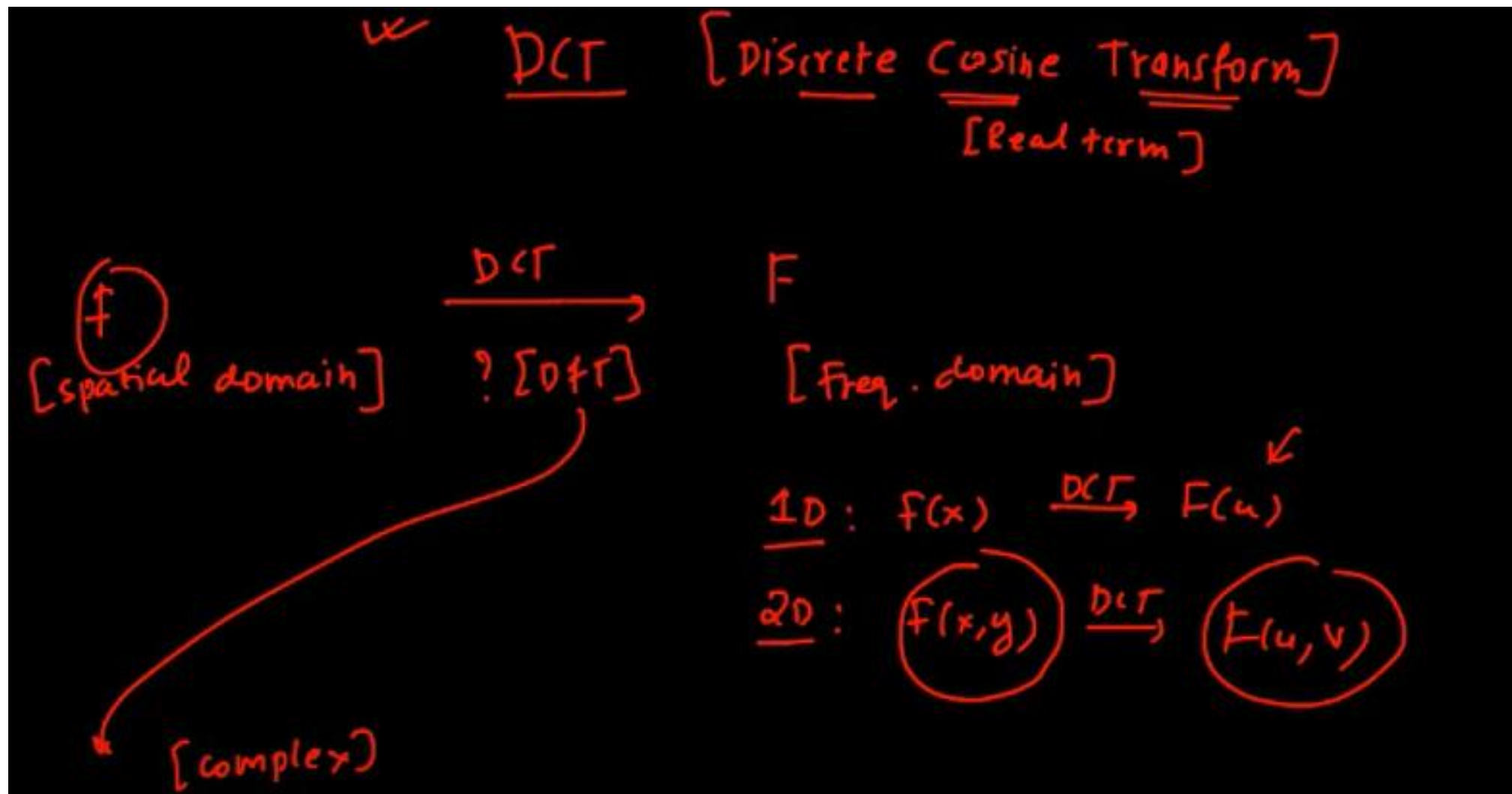
The Haar Transform of the original pixel array P results in a new array of coefficients that compactly represent the original image:

$$H(P) = [45, -20, -10, -10, -5, -5, -5, -5]$$

This new representation includes one coefficient for the overall average intensity of the image (45) and coefficients representing various levels of detail (-20, -10, -10, etc.). For image compression, one could choose to keep only the most significant coefficients (those representing the largest differences), which would allow the original image to be reconstructed with some loss of detail. This is the basis of many wavelet-based compression techniques, including those used in the JPEG2000 standard.

3. Discrete Cosine Transform

- The Discrete Cosine Transform (DCT) is a widely used technique in signal and image processing, especially prominent in the field of image and video compression. It belongs to the family of Fourier-related transforms, with the unique characteristic of using only cosine functions. The DCT transforms a sequence of spatial domain (time domain for signals) samples into a sum of cosine functions oscillating at different frequencies. Its popularity in compression, particularly for JPEG image compression and MPEG video compression, stems from its excellent energy-compaction properties: most of the signal information tends to be concentrated in a few low-frequency components of the DCT.
- There are several types of DCT, but the most commonly used in image processing is the DCT-II. It transforms a block of image pixels (e.g., 8x8, 16x16) from the spatial domain into the frequency domain, where each coefficient represents the amplitude of a certain frequency of cosine wave.
- In image compression like JPEG, the DCT coefficients are quantized (reducing the precision of the higher frequency coefficients more aggressively), which leads to compression. The process exploits the fact that high-frequency components are generally less perceptible to the human eye, allowing them to be compressed more without significantly affecting the perceived image quality.
- This ability of the DCT to compact the signal's energy into fewer coefficients makes it highly efficient for compression, as many of the higher frequency coefficients can be made very small (or even zero) without greatly impacting the visual fidelity of the reconstructed image.



We can transform spatial domain image to frequency domain using either DFT or DCT. While DFT contains complex terms in equation, DCT contains only the real terms. So the main aim is to convert $F(x,y)$ into $F(u,v)$.

$$F(u) \quad \left\{ \begin{array}{l} \text{v} \\ F(u,v) \end{array} \right.$$

$$F = C \cdot f$$

$$F = C \cdot f \cdot C^T$$

$$C =$$

Cosine T.M.

$$\sqrt{\frac{1}{4}} = 0.5$$



$$C(u,v) = \left(\sqrt{\frac{1}{N}} \right)$$

$$u=0, \quad 0 \leq v \leq N-1$$

$$C(u,v) = \sqrt{\frac{2}{N}} \cos \left[\frac{(2v+1)\pi u}{2N} \right]$$

$$\begin{array}{l} 1 \leq u \leq N-1 \\ 0 \leq v \leq N-1 \end{array}$$

u

$$4 \text{ } \phi \text{ } v \rightarrow 0 \text{ to } N+1$$

N=4

u=

v=

$$C(u,v) = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0 & 0.5 & -0.5 & 0.5 \\ 0 & 0.5 & -0.5 & 0.5 \\ 0 & 0.5 & -0.5 & 0.5 \end{bmatrix}$$

- For 1D, $F = C * F$ while for 2D, $F = C * F * C$ Transpose
- For the first row, value is given by $\text{root}(1/N)$ while for the rest of the rows, the value is given by the second formula.

$C(u, N) =$

	$v=0$	1	2	3
0	$\sqrt{0.5}$	0.5	0.5	0.5
1	0.653	0.2705	0.2705	-0.653
2	0.5	-0.5	-0.5	0.5
3	0.2705	-0.653	0.653	-0.2705

$\sqrt{\frac{1}{4}} = 0.5$

The matrix is symmetric, and the value 0.2705 is circled in the original image.

- Discrete Cosine Function (DCF) is not Symmetric function, so C^T is not equal to C .
- It is used in JPEG Compression
- It is a sinusoidal function, which contains only real parts unlike Discrete Fourier Transform (DFT)

Attempt any two questions. (2 × 10 = 20)

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