Objectives:

(i) To learn about 2D discrete fourier transform.

(ii) To learn how to represent image in frequency domain.

(iii) To learn about noise removal using DFT.

Introduction: The discrete fourier transform is a fundamental tool in image processing that can represent an image in frequeny domain from spatial domain. In spatial domain, an image is represented by pixels with their corresponding intensity values. On the other hand, in frequency domain, an image is represented by its frequency components. Here high frequency components correspond to sharp edges and low frequency components correspond to smooth variations.

A 2D DFT converts an into a complex array that consist of

(i) Power spectrum

(ii) Phase angle

We can apply various filtering on an image after converting it into frequency domain.

To apply filtering, an image is preprocessed, then converted into fourier domain, then multiply it with filter function, finally apply inverse DFT to get the filtered image in spatial domain.

Notch reject filter is a specific type of band-reject filter that targets and eliminates a narrow band of frequency around a particular center frequency. This filter is useful for removing ~~noise~~ periodic noise that manifest as stripes or grid-like patterns in an image.

It rejects frequencies in a predefined neighbourhood about the center of the frequency rectangle. H is usually symmetric about the origin, So if there is a ~~noth~~ notch at (u, v) then there must have a notch at (-u, -v).

The ideal notch reject filter is mathematically represented as

$$H(u,v) = \prod_{k=1}^{Q} H_k(u,v) \cdot H_{-k}(u,v)$$

High pass-filter transfer function

Here, the center of the filter are (u_k, v_k) and $(-u_k, -v_k)$, the center of the frequency rectangle is $\left(\frac{M}{2}, \frac{N}{2}\right)$ where M, N are the no. of rows and columns of the input image.

The a

The previous form can be generalized as

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

The distance ~~d~~ computation for each filter transfer function are given by,

$$D_k(u,v) = \left[\left(u - \frac{M}{2} - u_k\right)^2 + \left(v - \frac{N}{2} - v_k\right)^2 \right]^{\frac{1}{2}} \text{ and}$$

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

Classwork pseudo-code:

1. import necessary librarie.
2. Read image in grayscale mode.
3. Convert into frequency domain and shifter origin to actual image center.
4. Find the magnitude spectrum.
5. Take log and scale it as needed
6. Calculate the phase angle.
7. Create the notch filter.
   Iterate all ~~row~~ pixel of the image and place either 0 or 1 based on the distance from the center point.

8. Multiply the notch with magnitude spectrum to get the final image.

9. Apply inverse transform and show the final image.

Discussion: DFT in image processing can break down an image into various frequency component. Theses frequencies can reveal details like edges and smooth area. It can be used for noise reduction, image sharpening, compression etc.

Conclusion: Discrete fourier transform performs very well in Image ~~processing~~ with repetitive pattern or where specific frequencies hold significant info. It is widely used in noise reduction, sharpening, watermarking for copyright protection etc. Though it offers valuable tools in image processing, but it may not suitable approach for all image types.

References:

① Lab documents
② https://vincmazet.github.io/bip/filtering/fourier.html

Drawing portrait using DFT and epicycles.

methodology:

(i) Read the input image in grayscale mode.

(ii) Apply Canny edge detection to detect edges.

(iii) Generate the edge points in correct order.

a) start at point (0,0).

b) Apply BFS to find the nearest white pixel.

c) Apply DFS twice to detect edge points in correct order.

d) merge all the points into a single list.

e) Continue doing (a) to (e) until all pixels are processed.

(iv) Generate the fourier co-efficients.

(v) Animate using library (matplotlib).