LAPORAN PRAKTIKUM PENGOLAHAN CITRA DIGITAL

12. 2D FOURIER TRANSFORM

Disusun oleh :

Nama :

NPM :

Kelas : IF4..

PROGRAM STUDI INFORMATIKA

FAKULTAS ILMU KOMPUTER DAN REKAYASA

UNIVERSITAS MULTI DATA PALEMBANG

2024

TUTORIAL : 2D FOURIER TRANSFORM

Goal

The goals of this tutorial are to learn how to compute and display the FT of an image and how to develop filters to be used in the frequency domain.

Objectives

- Learn how to use the fft2 function to compute the FT of a monochrome image.
- Learn how to visualize the FT results.
- Learn how to generate filters to be used in the frequency domain.

What You Will Need

- `distmatrix.m`

Procedure

To generate the FT of an image (a 2D function), we use the IPT function fft2, which implements the FFT algorithm.

1. Load the cameraman image, convert it to `double` (one of the data classes accepted as an input to `fft2`), and generate its FT.

```
I = imread('cameraman.tif');
Id = im2double(I);
ft = fft2(Id);
```

Question 1 What are the minimum and maximum values of the resulting discrete Fourier transform coefficients for the cameraman image?

To view the spectrum of the image, that is, the amplitude component of the FT results, we must shift the zero-frequency (DC) component to the center of the image using the `fftshift` function.

2. Shift the FT array of results.

```
ft_shift = fftshift(ft);
```

From your answer to Question 1, you should by now know that the range of values in the FT array of results (`ft`) extends well beyond the typical values of a grayscale image ([0, 255]). Consequently, we will try to display the resulting spectrum as an image using the "scaling for display purposes" option of function `imshow`.

3. Display the FT results, remapped to a grayscale range.

```
figure, subplot(1,2,1), imshow(abs(ft_shift),[]), ...
    title('Direct remap');
```

Question 2 Why are we required to use the `abs` function when displaying the `ft_shift` image?

Question 3 Howdid we remap the image to a different (narrower) grayscale range?

As you may have noticed, directly remapping the image to the grayscale range does not give us any useful information (only a white pixel at the center of the image the DC component of the FT results with all other pixels displayed as black). This suggests that there might be a significant amount of detail in other frequencies that we just cannot see.

We can perform the log transformation within the `imshow` function call and then remap the adjusted values to the grayscale range by specifying '[]' in the second parameter.

4. Display the log of the shifted FT image.

```
subplot(1,2,2), imshow(log(1 + abs(ft_shift)), []), ...
    title('Log remap');
```

Question 4 How does the log remap compare to the direct remap?

Distance Matrices

In the second part of this tutorial, we will look at distance matrices. To specify and implement frequency-domain filters, we must first generate a matrix that represents the distance of each pixel from the center of the image. We can create such matrix using the `distmatrix` function. The function takes two parameters, `M` and `N`, and will return the distance matrix of size `M × N`.

5. Close any open figures.
6. Generate a distance matrix that is the same size as the image I.

```
[M, N] = size(I);
D = distmatrix(M, N);
```

We can visualize the distance matrix in a 3D mesh plot, but we must first shift the image similar to the way we shifted the frequency spectrum earlier.

7. Create a 3D mesh plot of the distance matrix.

```
D_shift = fftshift(D);
figure, mesh(D_shift)
```

Question 5 Explain the shape of the 3D plot.

After obtaining the distance matrix, we can generate the filter of our choice. From that point, filtering in the frequency domain is as simple as multiplying

the filter by the FT image and converting back to the spatial domain. These processes will be examined in the remaining tutorials of this chapter.