CO543 – Image Processing
Lab 05
Fourier Transformation

Objectives

- Computing the Fourier Transform for an image and displaying the spectral image.
- Designing of filters in the frequency domain (lowpass and highpass filters) and applying them to images.

Fourier Transform is used to analyze the frequency characteristics of various filters. For images, 2D Discrete Fourier Transform (DFT) is used to find the frequency domain. A fast algorithm called Fast Fourier Transform (*FFT*) is used for calculation of DFT.

Both openCv and Numpy have packages to do Fourier Transform. We use the numpy package in this lab. np.fft.fft2() provides us the frequency transform which will be a complex array.

fft.fft2(a, s=None, axes=(- 2, - 1), norm=None)

Its first argument is the input image, which is gray-scale. Second argument is optional which decides the size of the output array.

If it is greater than the size of the input image, the input image is padded with zeros before calculation of FFT. If it is less than input image, input image will be cropped. If no arguments passed, Output array size will be the same as input.

Numpy has three functions to compute the DFT:

1. fft - for one dimension (useful for audio)
2. fft2 - for two dimensions (useful for images)
3. fftn - for n dimensions

Numpy has three functions that compute the inverse DFT:

1. ifft
2. ifft2
3. ifftn

Let's try an example...

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

img = cv.imread('car.jpg',0) ## read image as grayscale
f = np.fft.fft2(img) ## pass grayscale image

fshift = np.fft.fftshift(f) ## Shift the zero-frequency component to the center of the spectrum.
magnitude_spectrum = 20*np.log(np.abs(fshift)) ## apply logarithm, otherwise the image can not identify easily, change and see the changes.

plt.subplot(121), plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```

Change the input image and observe the difference in the Spectrum. (Use pattern, rotate, add padding … etc)

Reconstruct image from inverse fourier transform (spectrum) In [2]:

```
I2 = np.fft.ifft2(f) # f is transformed image
I3 = np.real(I2) # get the real part

plt.subplot(121), plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Spectrum'), plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(I3, cmap = 'gray')
plt.title('Reconstructed'), plt.xticks([]), plt.yticks([]) plt.show()
```

In the 'Magnitude Spectrum' image, the whiter region at the center shows the low frequency content which is high. Following steps will show you how to apply a filter to an image in the frequency domain.

Basic Steps in DFT Filtering

The following summarize the basic steps in DFT Filtering

1. Obtain the Fourier transform:
    - F=fft2(f)
2. Generate a filter function, H
3. Multiply the transform by the filter:
    - G=H.*F
4. Compute the inverse DFT:
    - g=ifft2(G)
5. Obtain the real part of the inverse FFT of g:
    - g2=real(g);

Let's try a low-pass filter for an image..

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('car.jpg',0)

img_float32 = np.float32(img)

dft = cv2.dft(img_float32, flags = cv2.DFT_COMPLEX_OUTPUT) dft_shift =
np.fft.fftshift(dft)

rows, cols = img.shape
crow, ccol = rows//2 , cols//2 # center

# create a mask first, center square is 1, remaining all zeros mask = np.zeros((rows, cols,
2), np.uint8)
mask[crow-30:crow+30, ccol-30:ccol+30] = 1
```

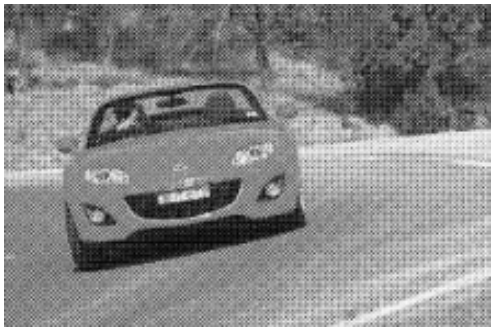
```
# apply mask and inverse DFT
fshift = dft_shift*mask
f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:,:,0],img_back[:,:,1])

plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
plt.title('Low-pass Image'), plt.xticks([]), plt.yticks([]) plt.show()
```

Observe the resulting image.

Lab Tasks

1. Apply high pass laplacian filter on Car.jpg image.
2. Apply ideal high-pass filter on Car.jpg image for D0=100
3. Apply ideal low-pass filter on Car.jpg image for D0=100
4. Apply FFT2, IFFT2, low-pass Gaussian filter, and high-pass laplacian filter on Car.jpg image.
5. Apply the necessary filter and correct the noise in the image. Image file is uploaded.
6. Apply the sobel operator (filter) on Car.jpg in the Fourier domain to detect edges.



7. Discuss applying Butterworth and Chebyshev filters and compare the output image with the Gaussian Filter image (You may use a preferred image to discuss the characteristics of the output images in Q7.)

Submission

You need to submit a jupyter notebook file containing the relevant programs and functions named according to the relevant question names as indicated in the lab sheet. Make sure to include the input images you used to run the codes as well.

You need to submit a PDF file displaying screenshots of the code, results from your code (your input and output images under each section after performing the required functions).

You can submit a single ZIP file as e17XXXlab05.zip including all:

- Final report in .pdf format
- Python source codes
- All Input and output images

Note: XXX indicates your registration number in all cases.