Project 2

王國倫 0860077 電控所碩一

- Source codes

```
# library
import numpy as np
import cv2
from matplotlib import pyplot as plt
%matplotlib inline
```

```
# load picture from google drive
from google.colab import drive
drive.mount('/content/drive')
```

```
# loaded the image in grayscale
image = cv2.imread('/content/drive/My Drive/Bird 2.tif',0)

# convert from uint8 into float32
image_float32 = np.float32(image)

# Computed the 2-d discrete Fourier Transform
dft = cv2.dft(image_float32, flags = cv2.DFT_COMPLEX_OUTPUT)

# Shift the zero-frequency component to the center of the spectrum.
dft_shift = np.fft.fftshift(dft)

# compute magnitude spectrum in log scale
one = np.ones(dft_shift.shape[:2])
magnitude_spectrum =np.log(one+cv2.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))
```

```
# split whole frequency region to half left
x = magnitude_spectrum[0:512,0:256]

# transfrom to 1 dimension
t = x.ravel()
```

```
# get top 25 DFT frequency
y = t.argsort()[-25:]
```

```
# get position (u,v) of top 25 DFT frequency
pos = []
for i in y:
    pos.append([i//256,i%256])
pos.reverse()
```

```
# list top 25 DFT frequency
pos
```

```
# plot magnitude spectrum
plt.imshow(magnitude_spectrum, cmap = "gray")
plt.colorbar()
plt.title("Magnitude Spectrum")
```

```
# discrete fourier transform
dft = cv2.dft(image_float32, flags = cv2.DFT_COMPLEX_OUTPUT)

# shift to center
dft_shift = np.fft.fftshift(dft)
```

```
# create a ideal low pass filter and ideal high pass filter
def ILPF(diameter,shape):

    result = np.zeros(shape,dtype=np.uint8)
    center = [(shape[0]-1)/2,(shape[1]-1)/2]

    for i in range(shape[0]):
        for j in range(shape[1]):
            if(((i-center[0])**2+(j-center[1])**2)**0.5<diameter):
                result[i,j]=1
    return result

def IHPF(diameter,shape):

    result = np.zeros(shape,dtype=np.uint8)
    center = [(shape[0]-1)/2,(shape[1]-1)/2]

    for i in range(shape[0]):
        for j in range(shape[1]):
            if(((i-center[0])**2+(j-center[1])**2)**0.5>=diameter):
                result[i,j]=1
    return result
```

```
# implement ideal low pass filter and high pss filter
low = ILPF(30,image.shape[:2])
high = IHPF(30,image.shape[:2])
```

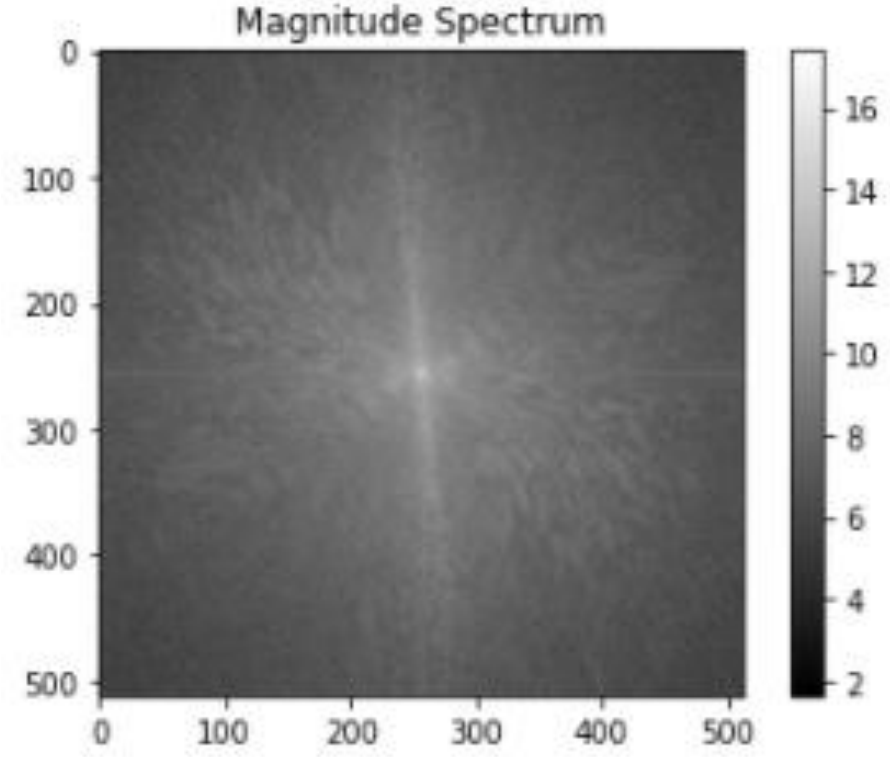
```
# apply mask and inverse DFT
fshift = np.zeros(dft_shift.shape)
for i in range(dft_shift.shape[2]):
    fshift[:,:,i] = dft_shift[:,:,i]*low
f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:,:,0],img_back[:,:,1])
```

```
# plot image by using ideal low pass filter
plt.figure(figsize=(15,9))
plt.imshow(img_back, cmap = "gray")
plt.title("output image for ideal low pass filter")
```

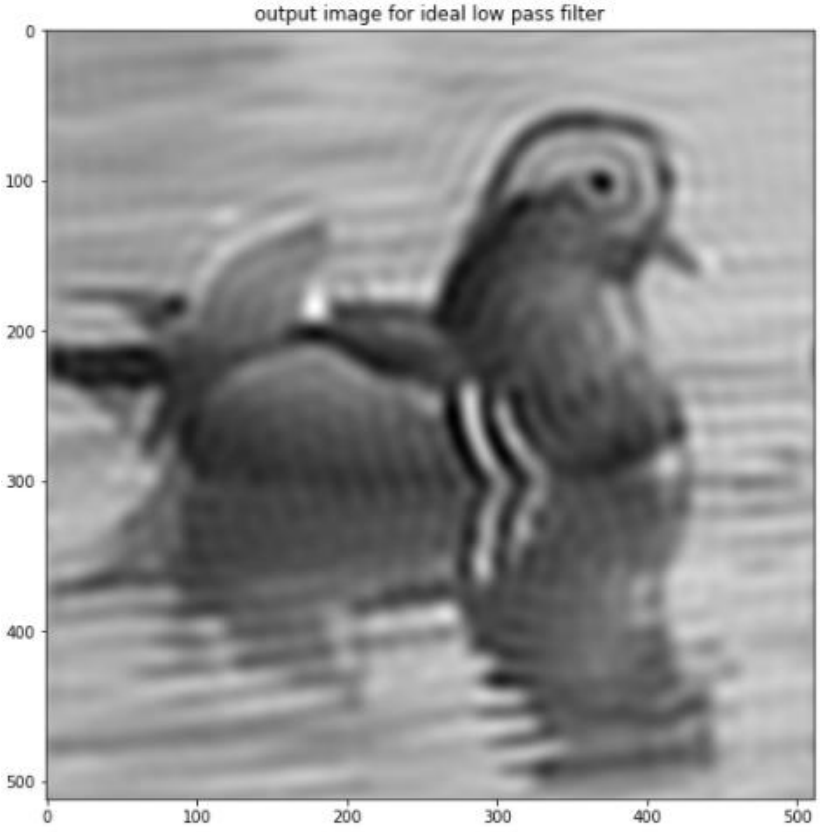
```
# apply mask and inverse DFT
fshift = np.zeros(dft_shift.shape)
for i in range(dft_shift.shape[2]):
    fshift[:,:,i] = dft_shift[:,:,i]*high
f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:,:,0],img_back[:,:,1])
```

```
# plot image by using ideal high pass filter
plt.figure(figsize=(15,9))
plt.imshow(img_back, cmap = "gray")
plt.title("output image for ideal high pass filter")
```

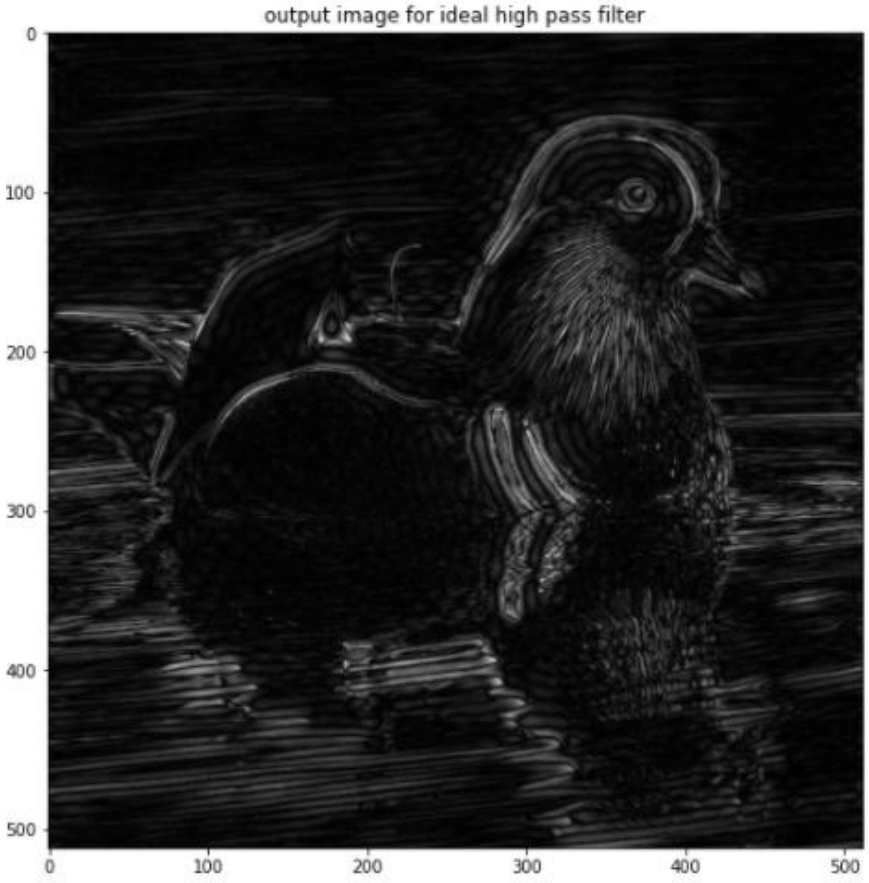
- Plot of DFT magnitude in Log scale



- Image constructed by DFT coefficients inside the circular region with radius = 30 pixels



- Image constructed by DFT coefficients outside the circular region with radius = 30 pixels



- Table of top 25 DFT frequencies (u,v) in the left half frequency region ($0 \leq u \leq M-1$, $0 \leq v \leq N/2-1$)

top	(u,v)
1	256, 254
2	256, 255
3	255, 255
4	257, 255
5	257, 254
6	253, 255
7	259, 254
8	258, 255
9	259, 255
10	253, 254
11	256, 253
12	258, 252
13	254, 254
14	258, 253
15	252, 253
16	248, 255
17	254, 255
18	254, 252
19	260, 254
20	262, 255
21	254, 253
22	255, 252
23	255, 254
24	252, 255
25	261, 254