Project #4

DIGITAL IMAGE PROCESSING

Student: 黎文雄 (Le Van Hung)

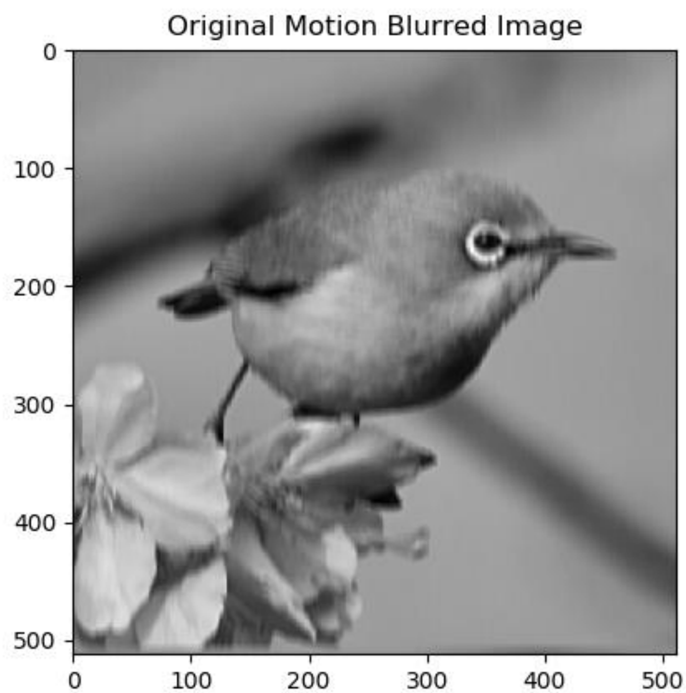
Student ID: 0860831

Requirement:

a. Estimate the direction of linear motion and the displacement

b. Construct and plot the restored image using H(u,v)

- *Your report should contain:*
- Source codes
- Figures of Fourier magnitude spectra of the degraded image
- Figure of the Fourier magnitude of degradation model H(u,v) for uniformly linear motion blurring
- Figure of the output image
- Model parameters: direction of linear motion, estimate of displacement in pixel



Original image

a. Estimate the direction of linear motion and the displacement

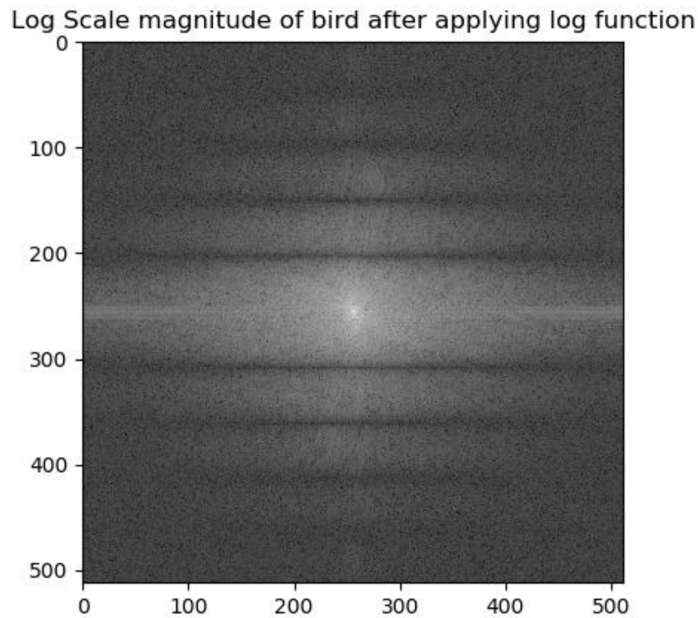


Figure1: The Fourier magnitude spectra of the degraded image

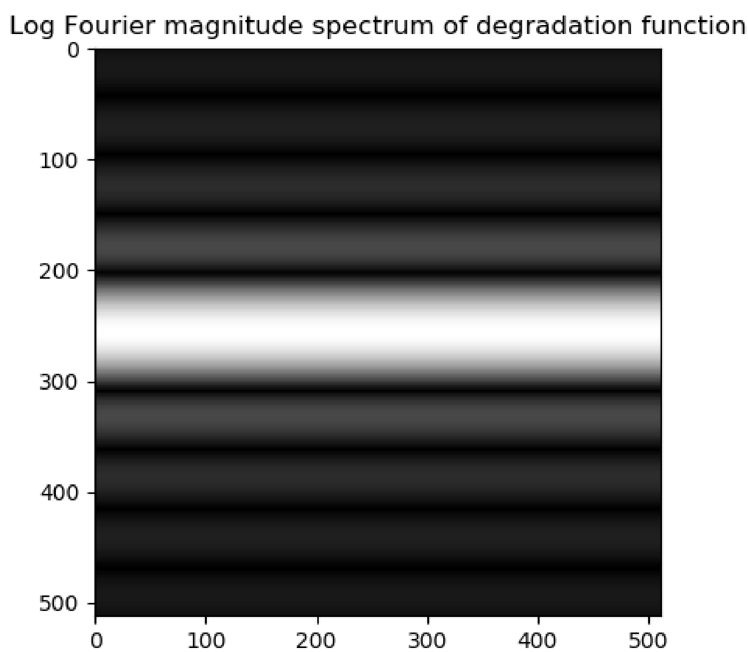
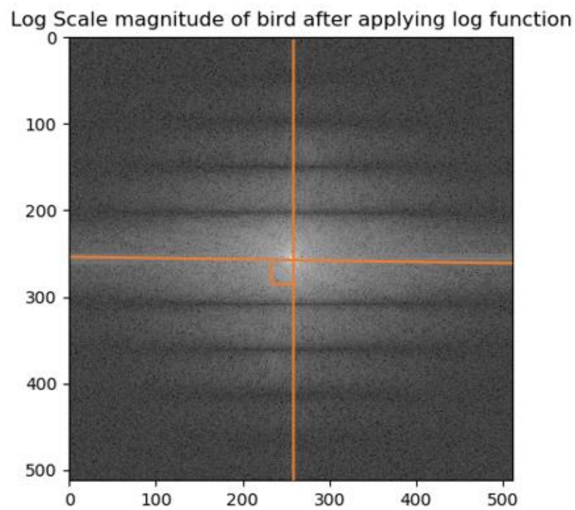
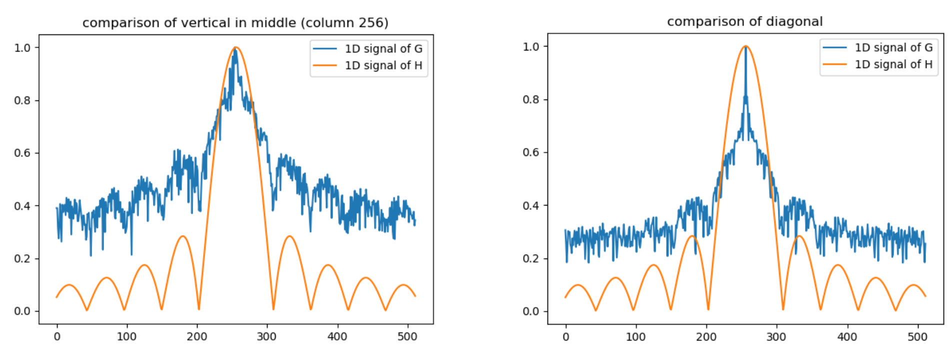


Figure2: The Fourier magnitude of the degradation model H(u,v) for uniformly linear motion blurring

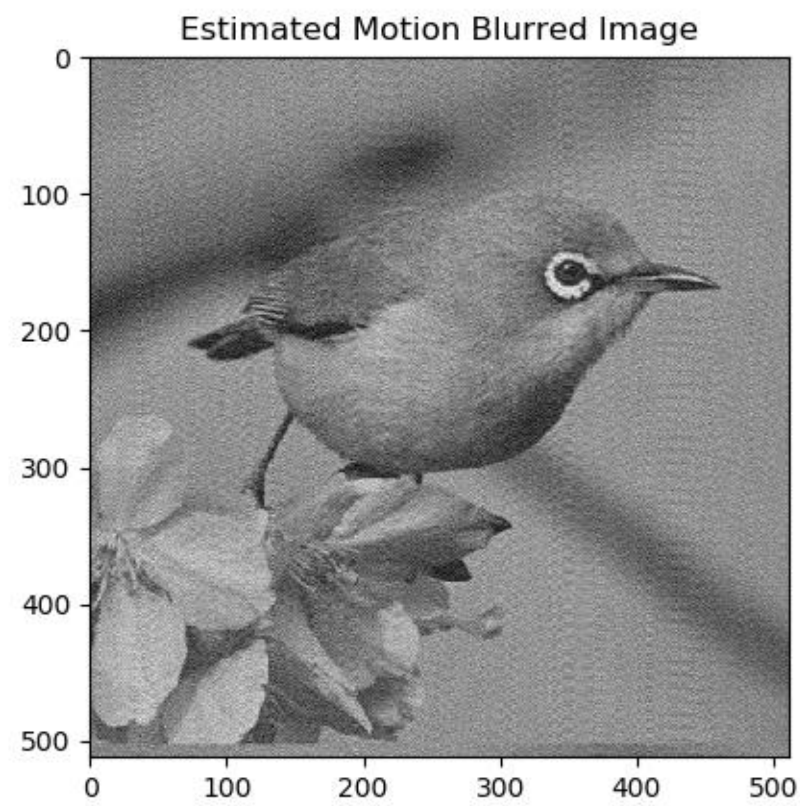
- **Model Parameter:**



- ✓ Direction of linear motion: 0 degree with respect to x axis
- ✓ Estimate of displacement in pixel: a = 0.0188, b = 0 and T = 1
- ✓ To find the best selection of parameter a. Align and compare the 1D scanned Fourier spectra of blurred image and linear-motion mathematic model. In this case, b=0 because of the direction of linear motion.



b. **Construct and plot the restored image using the H(u,v) obtained.**



Sources code: (using Python)

```
# import library
from PIL import Image
import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing

# import image
image = cv2.imread("image-pj4 (motion blurring).tif",
cv2.IMREAD_GRAYSCALE)
g = np.array(image, dtype='float')
shape_img = np.shape(g)
N = shape_img[0]

# multi fp with (-1)^(x+y) for centerring
```

```
gc = np.zeros_like(g)
for x in range(N):
    for y in range(N):
        gc[x, y] = g[x, y] * pow(-1, (x + y))

# DFT
Gc = np.fft.fft2(gc)
#get abs of F
Gc_abs = np.abs(Gc)

# apply log function
Gc_abs_log= np.log(1 + Gc_abs)

#############################################################
# define pare
a = 0.0188
b = 0
T = 1
#because theta = 0 with respect to horizontal axis
#Establishing degradation function
H = np.zeros_like(Gc)
for u in range(N):
    for v in range(N):
        k = np.pi * (a * (u - 256) + b * (v - 256))
        if (u ==256):
            H[u,v] = T*np.exp(-1j*k)
        else:
            H[u, v] = (T*np.sin(k) * np.exp(-1j * k)) / k

H_abs = np.abs(H)

# apply log function
H_abs_log= np.log(1 + H_abs)

plt.imshow(H_abs_log,cmap='gray')
plt.title("Log Fourier magnitude spectrum of degradation function")
plt.show()

#Find F(u,v)
F = np.zeros_like(Gc)
for u in range(N):
    for v in range(N):
        F[u,v] = Gc[u,v]/H[u,v]

#inverse DFT
fc = np.fft.ifft2(F)
```

```
# get real of fc
fc_real = fc.real
fc_real = np.array(fc_real,dtype='float')

f_e = np.zeros_like(fc_real)
for x in range(N):
    for y in range(N):
        f_e[x,y] = fc_real[x,y]*pow(-1,(x+y))


# Fourier magnitude of ouput include centering
plt.title("Estimated Motion Blurred Image")
plt.imshow(f_e, cmap ='gray')
plt.show()

plt.title("Original Motion Blurred Image")
plt.imshow(image, cmap ='gray')
plt.show()

plt.imshow(Gc_abs, cmap='gray')
plt.title("Fourier magnitude of bird before applying log function")
plt.show()

plt.imshow(Gc_abs_log, cmap='gray')
plt.title("Log Scale magnitude of bird after applying log function")
plt.show()


#comparison of vertical in middle
Align_G = Gc_abs_log[:,255:256]/max(Gc_abs_log[:,255:256])
Align_H = H_abs_log[:,255:256]/max(H_abs_log[:,255:256])
#compare plot
plt.plot(Align_G,label = '1D signal of G')
plt.plot(Align_H,label = '1D signal of H')
plt.title("comparison of vertical in middle (column 256)")
plt.legend()
plt.show()

#comparison of diagonal
Align_G = []
Align_H = []
for u in range(N):
    for v in range(N):
        if (u==v):
            Align_G.append(Gc_abs_log[u,v])
```

```
            Align_H.append(H_abs_log[u,v])
Align_G = np.array(Align_G)
Align_H = np.array(Align_H)

Align_G = Align_G/max(Align_G)
Align_H = Align_H/max(Align_H)
#compare plot
plt.plot(Align_G,label = '1D signal of G')
plt.plot(Align_H,label = '1D signal of H')
plt.title("comparison of diagonal")
plt.legend()
plt.show()
```