Assignment 3

COMP 478 Image Processing

Etienne Pham Do

40130483

Etienne Pham Do 40130483

COMP 478 Assignment 3

1.

$$H(u,v) = \sum_{x=-1}^{1}\sum_{y=-1}^{1} F(u,v)e^{-j2\pi(\frac{ux_0}{M}+\frac{vy_0}{N})}$$

$$= \frac{1}{4}\sum_{x=-1}^{1} e^{\frac{-j2\pi ux}{M}} \sum_{y=-1}^{1} e^{\frac{-j2\pi vy}{N}}$$

$$= \frac{1}{4}(e^{\frac{j2\pi u}{M}} + 0 + e^{\frac{-j2\pi u}{M}})(e^{\frac{j2\pi v}{N}} + 0 + e^{\frac{-j2\pi v}{N}})$$

$$= \frac{1}{4}(2\cos\left(\frac{2\pi u}{M}\right))(2\cos\left(\frac{2\pi v}{N}\right))$$

2.

a)

Let Radon(f(x,y)) = g(ρ,θ), and f and g be f(x,y) and g(x,y)

$$Radon(af+bg) = \iint_{-\infty}^{\infty}\left(af(x,y)+bg(x,y)\right)\delta(xcos\theta + ysin\theta - \rho)dxdy$$

$$= aRadon\left(f(x,y)\right) + bRadon\left(g(x,y)\right)$$

$$Radon(af+bg) = aRadon(f) + bRadon(g)$$

b)

$$Radon\left(f(x-x_0, y-y_0)\right) = \iint_{-\infty}^{\infty} f(x-x_0,y-y_0)\delta((x-x_0)cos\theta + (y-y_0)sin\theta - \rho)dxdy$$

$$= \iint_{-\infty}^{\infty} f(x-x_0,y-y_0)\delta(xcos\theta - x_0cos\theta + ysin\theta - y_0sin\theta - \rho)dxdy$$

$$= g(\rho - x_0cos\theta - y_0sin\theta, \theta)$$

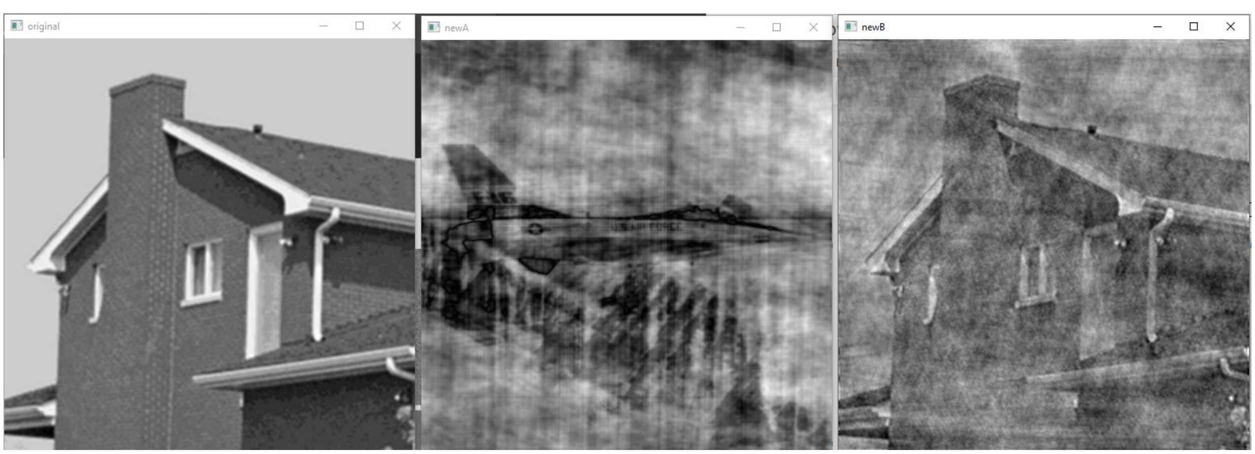
Programming

1.

```
import numpy as np

import cv2


#loading the images

Ia = cv2.imread('house.tif', 0)

Ib = cv2.imread('jet.tiff', 0)


#applying the fourier transforms and center the transforms

Fa = cv2.dft(np.float32(Ia), flags = cv2.DFT_COMPLEX_OUTPUT)

dft_shift_a = np.fft.fftshift(Fa)

Fb = cv2.dft(np.float32(Ib), flags = cv2.DFT_COMPLEX_OUTPUT)

dft_shift_b = np.fft.fftshift(Fb)


#getting the magnitudes and phases

magA, phaseA = cv2.cartToPolar(dft_shift_a[:,:,0], dft_shift_a[:,:,1])

magB, phaseB = cv2.cartToPolar(dft_shift_b[:,:,0], dft_shift_b[:,:,1])


#applying the switch in phases and merge the values in new frequency arrays

realA, imagA = cv2.polarToCart(magA, phaseB)

realB, imagB = cv2.polarToCart(magB, phaseA)

mergedCartA = cv2.merge([realA, imagA])

mergedCartB = cv2.merge([realB, imagB])


#undoing the shift

mergedCartA_ishift = np.fft.ifftshift(mergedCartA)

mergedCartB_ishift = np.fft.ifftshift(mergedCartB)
```

```
#inverse fourier transform
newA = cv2.idft(mergedCartA_ishift)
newB = cv2.idft(mergedCartB_ishift)
newA = cv2.magnitude(newA[:,:,0], newA[:,:,1])
newB = cv2.magnitude(newB[:,:,0], newB[:,:,1])

#converting back to 8 bit images from 32 bit
newA = cv2.normalize(newA, None, alpha=0, beta=255, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_8U)
newB = cv2.normalize(newB, None, alpha=0, beta=255, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_8U)

#showing the results
cv2.imshow('original', Ia)
cv2.imshow('newA', newA)
cv2.imshow('newB', newB)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

I_2 should have a reconstruction closer to the original image I_A since it has the phase of F_A, which has information about the shape features of I_A.



2.

```
import numpy as np

import cv2

import time


houseImg = cv2.imread('house.tif', 0)


#Laplacian of Gaussian

start = time.time()

gaussianBlur = cv2.GaussianBlur(houseImg, (7,7), 0)

laplacianOfGuassian = cv2.Laplacian(gaussianBlur, cv2.CV_64F)

end = time.time()

print(str(end - start))


#Canny edge detection

start = time.time()

cannyEdge = cv2.Canny(houseImg, 100, 200)

end = time.time()

print(str(end - start))


cv2.imshow('LoG', laplacianOfGuassian)

cv2.imshow('Canny', cannyEdge)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

a)

Laplacian of Gaussian:

- apply Gaussian blur on image

- apply Laplacian on Gaussian

Canny edge detection:

- Noise reduction
- Gradient calculation
- Non-maximum suppression
- Double threshold
- Edge linking

b)

Edge linking consists of transforming low intensity pixels into high intensity pixels if there are neighboring high intensity pixels. The first method needs this step as it would be useful to reduce noise.

c)

Laplacian of Gaussian: kernel size affects performance, used size 7x7 since it makes the performance faster and has less noise.

Canny edge: lower and upper thresholds affect performance, used 100 and 200 since it makes the performance faster and has less noise.

d)

Laplacian of Gaussian shows the edges of the house, but has more noise than the Canny edge output, which shows the general outline of the house.