Justine John "JJ" A. Serdoncillo
October 16, 2023

Cool Images: HW 2

Problem #3: Programming Exercise

Part A: Morphing Lena + Man

For the first part of the programming exercise, we were tasked to create a new image using the phase and magnitude spectrum of the lena and man respectively. The original figures and fourier spectra are as shown below. The last row corresponds to the new image using the new phase and magnitude. We can see the importance of the phase because even if we have the magnitude of the lena fourier spectrum, the figure looks pretty messy and still somewhat cool.

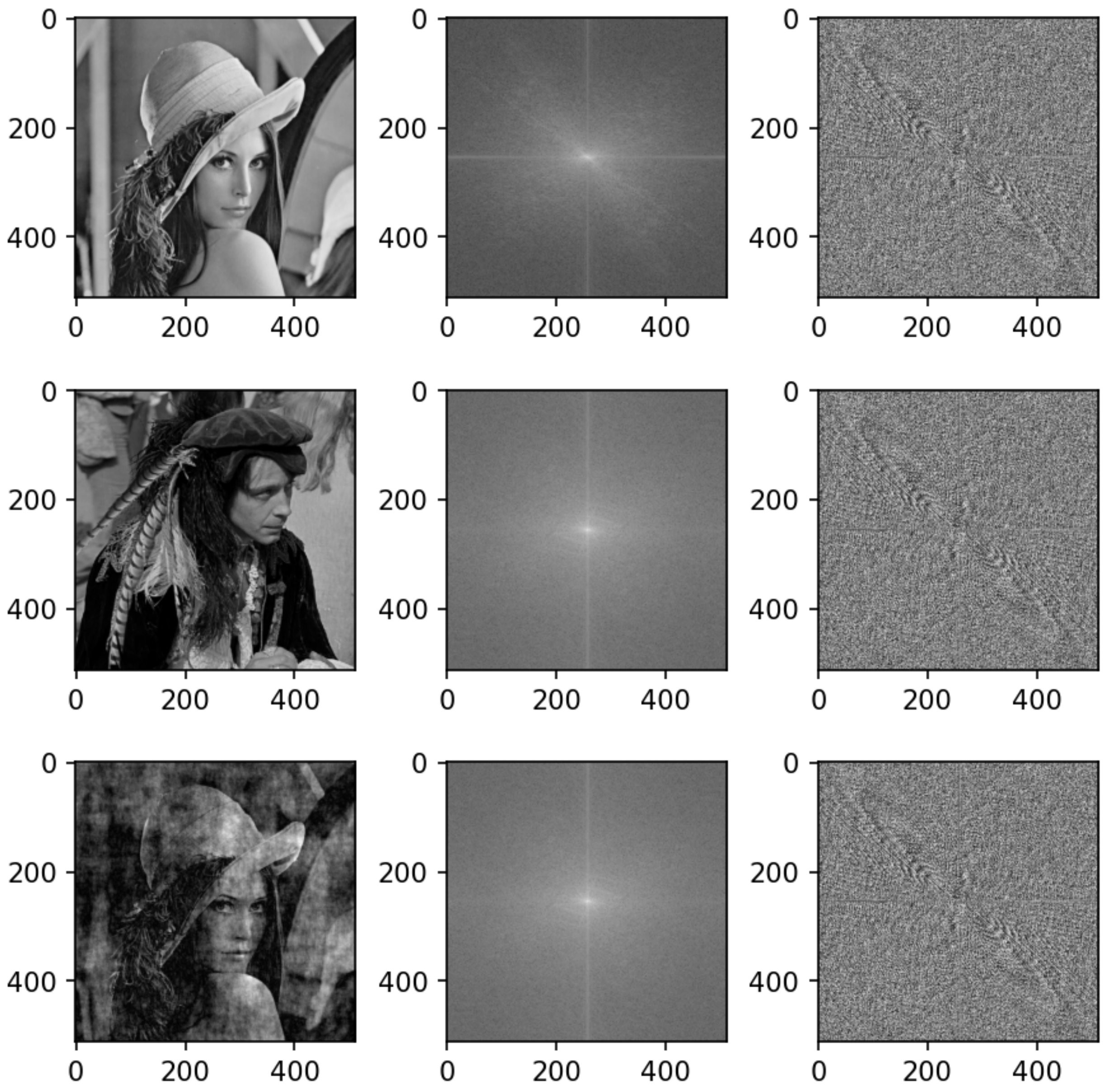

Figure 1: Adam and Eve merge together

Part B: DCT and DFT

For the second part of the programming exercise, we will try to compress the images by using DCT and DFT methods. The following figures show the compressed version of the lena image. It can be seen that the DCT method makes it look better while also using lower space for storage which is perfect.

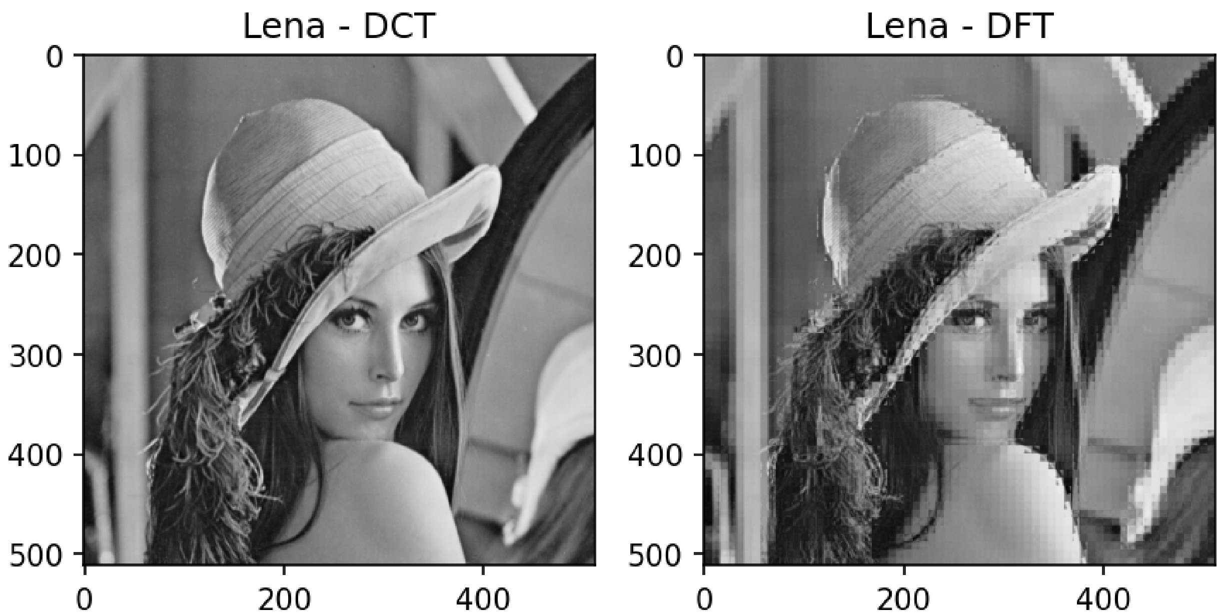

Figure 2: Lena cosine and lena fourier

Part C: Sharpening

For the last part of the programming exercise, the images are sharpened by using a high pass filter. This is pretty cool because it is actually pretty simple but gives out a decent sharpened image. Coins are usually dull but I guess this is a way to make it sharper.

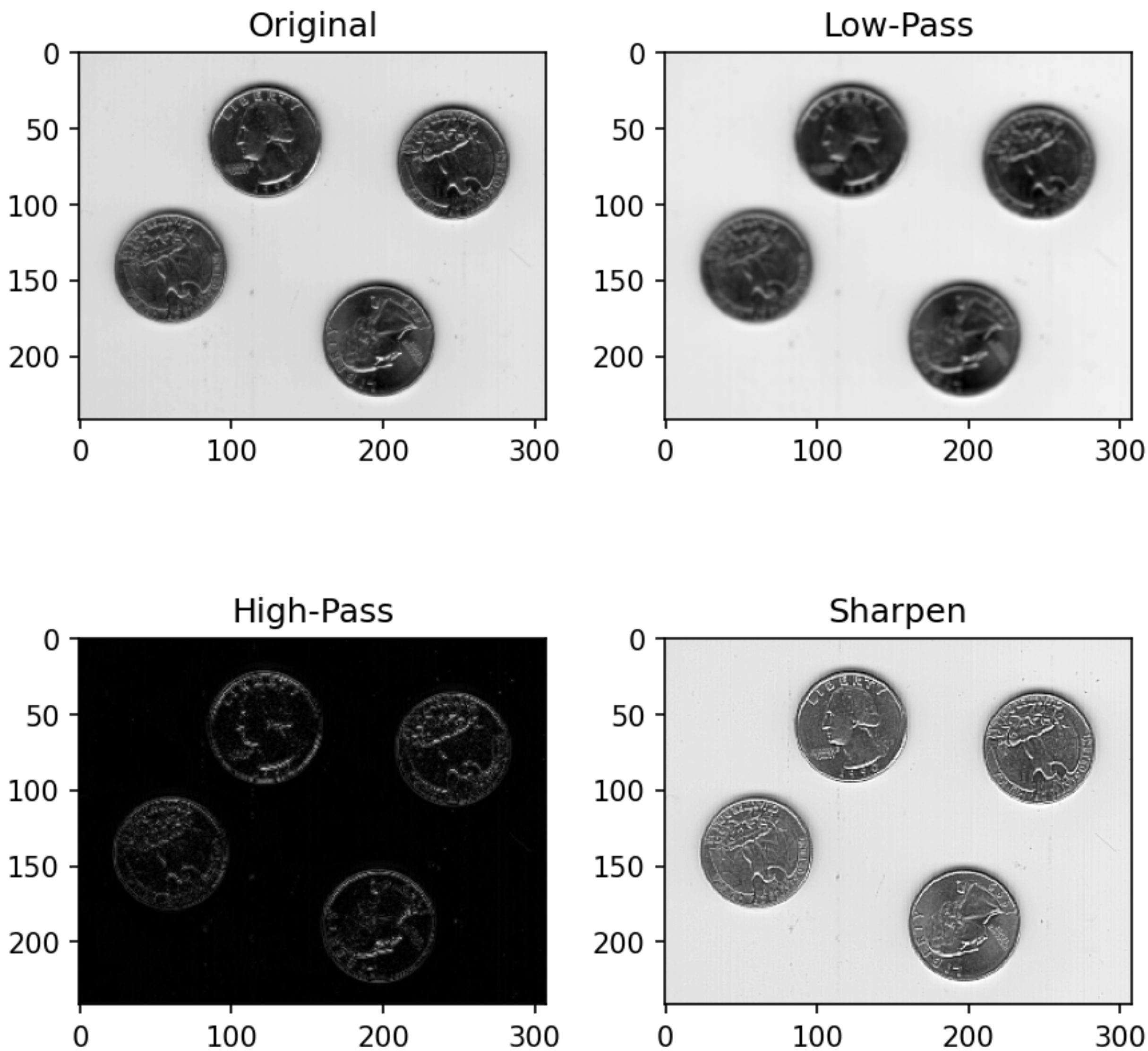

Figure 3: Sharp Coins ???

Appendix

Python Code

```python
'''
Justine Serdoncillo
EE 5561 - Image Processing
Problem Set 2
October 16, 2023
'''

# %% Problem Statement
"""
    a) [3 pts] Use the phase of the Fourier spectrum of Lena image, and the magnitude of the
    Fourier spectrum of the kneeling man image (provided to you), to generate a combined image
    with the corresponding phase and magnitude Fourier spectra. Display the input images and
    their relevant Fourier specta, and the final output image.

    b) [6 pts] Take the DCT of each distinct (i.e. not sliding) 8    8. Keep the largest (in
    magnitude) 10 DCT coefficients, and set the rest to zero. Take the inverse DCT of each
    block to generate a new image. Do the same with DFT and inverse DFT (i.e. FFT). Display
    the images.
    (Hint: In Python, it will be helpful to define a function to extract/process non-overlapping
    sliding blocks and store the results in a new array. For MATLAB, the built-in function
    blkproc may be helpful. Also defining a function that performs the given transform, then
    keeps the largest 10 coefficients and then does the inverse transform will also be useful.)

    c) [9 pts] Perform the sharpening exercise with the coins image ( eight .tif ).
    Read-in the image, then perform lowpass filtering with an averaging filter (use imfilter/signal.convolve2d).
    Generate the high-pass image by subtracting this low-pass image from the original.
    Generate the sharpened image as original image plus 2 times the high-pass image. Display the
    original, low-pass, high-pass and sharpened images.
"""
# %%

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as img
import scipy
import imageio
from scipy.fftpack import dct, idct
from scipy.signal import convolve2d

def fft2c(img):
    return np.fft.fftshift(np.fft.fft2(np.fft.ifftshift(img), norm = 'ortho'))

def ifft2c(freq):
    return np.fft.ifftshift(np.fft.ifft2(np.fft.fftshift(freq), norm = 'ortho'))

def dct2(img):
    return dct(dct(img.T, norm='ortho').T, norm='ortho')

def idct2(coeff):
    return idct(idct(coeff.T, norm='ortho').T, norm='ortho')

def moving_average_filter(img, kernel_size):
    kernel = np.ones((kernel_size[0], kernel_size[1])) / (np.product(kernel_size))
    return convolve2d(img, kernel, mode='same', boundary='wrap')

# %% Problem 3 Part A
def prob3a():
    # Load different images as an ndarray
    lena_img = np.asarray(img.imread('lena512.bmp'))
    man_img = imageio.v2.imread("man.png")

    # Take the fourier transform of the lena and man image
```

```
    lena_freq = fft2c(lena_img)
    lena_phase = np.angle(lena_freq)
    man_freq = fft2c(man_img)
    man_phase = np.angle(lena_freq)

    # Morph using magnitude of the man and the phase of lena
    man_mag = np.abs(man_freq)
    morph_img = ifft2c(man_mag * np.exp(1j * lena_phase))

    # Plot the figures
    fig, ax = plt.subplots(3,3, figsize=(6,6), dpi=150)
    ax[0,0].imshow(lena_img, cmap="gray")
    ax[0,1].imshow(np.log(np.abs(lena_freq)), cmap="gray")
    ax[0,2].imshow(lena_phase, cmap="gray")

    ax[1,0].imshow(man_img, cmap="gray")
    ax[1,1].imshow(np.log(np.abs(man_freq)), cmap="gray")
    ax[1,2].imshow(man_phase, cmap="gray")

    ax[2,0].imshow(np.abs(morph_img), cmap="gray")
    ax[2,1].imshow(np.log(np.abs(man_freq)), cmap="gray")
    ax[2,2].imshow(lena_phase, cmap="gray")
    fig.tight_layout()


 # %% Problem 3 Part B
def prob3b():
    lena_img = np.asarray(img.imread('lena512.bmp'))
    lena_DCT = np.zeros(lena_img.shape)
    lena_DFT = np.zeros(lena_img.shape)
    #lena_DCT = dct2(lena_img)
    #fig, ax = plt.subplots()
    #ax.imshow(lena_DCT)

    size = 8
    for row in range(int(round(lena_img.shape[0]/size))):
        for col in range(int(round(lena_img.shape[1]/size))):
            temp_img = dct2(lena_img[row*size:(row+1)*size,col*size:(col+1)*size])
            g = np.unravel_index(np.argsort(np.abs(temp_img.ravel()))[::-1][:10], [size,size
    ])
            TEMP_IMG = np.zeros((size,size))
            for index in g:
                TEMP_IMG[index] = temp_img[index]
            lena_DCT[row*size:(row+1)*size,col*size:(col+1)*size] = idct2(TEMP_IMG)

            temp_img = fft2c(lena_img[row*size:(row+1)*size,col*size:(col+1)*size])
            g = np.unravel_index(np.argsort(np.abs(temp_img.ravel()))[::-1][:10], [size,size
    ])
            TEMP_IMG = np.zeros((size,size))
            for index in g:
                TEMP_IMG[index] = temp_img[index]
            lena_DFT[row*size:(row+1)*size,col*size:(col+1)*size] = ifft2c(TEMP_IMG)

            #if (row+1) % 16 == 0 and (col+1) % 16 == 0:
            #fig, ax = plt.subplots()
            #ax.imshow(lena_DCT)

    fig, ax = plt.subplots(1,2, figsize=(6,3), dpi=150)
    ax[0].imshow(lena_DCT, cmap="gray")
    ax[0].set_title("Lena - DCT")

    ax[1].imshow(lena_DFT, cmap="gray")
    ax[1].set_title("Lena - DFT")
    fig.tight_layout()

    pass


# %% Problem 3 Part C
def prob3c():
    # Apple low_pass filter using convolution
    coins_img = np.complex64(plt.imread('eight.tif'))
```

```
    coins_img = np.asarray(img.imread('eight.tif'))
    #print(coins_img[200,100])
    coins_low_pass = moving_average_filter(coins_img, [3,3])
    #print(coins_low_pass[200,100])
    coins_high_pass = coins_img - coins_low_pass # max is 104 which is unusually high
    #print(coins_high_pass[200,100])
    #coins_sharp = np.abs(coins_img + 2 * coins_high_pass)
    coins_sharp = np.clip(np.abs(coins_img + 2 * coins_high_pass), 0, 255)
    #print(coins_sharp[200,100])

    # Plot the figures
    fig, ax = plt.subplots(2,2, figsize=(6,6), dpi=150)
    ax[0,0].imshow(np.abs(coins_img), cmap="gray")
    ax[0,0].set_title("Original")

    ax[0,1].imshow(np.abs(coins_low_pass), cmap="gray")
    ax[0,1].set_title("Low-Pass")

    ax[1,0].imshow(np.abs(coins_high_pass), cmap="gray")
    ax[1,0].set_title("High-Pass")
    print(np.max(coins_high_pass))

    ax[1,1].imshow(np.abs(coins_sharp), cmap="gray")
    ax[1,1].set_title("Sharpen")
    fig.tight_layout()
    print(np.max(coins_sharp))

# %% Main Function
if __name__ == "__main__":
    prob3a()
    prob3b()
    prob3c()
```