Digital Image Processing Project 2

Name: 彭晨益

Student ID: 310512054

(a)Source code

```
import argparse
import os
import csv
import cv2

import numpy as np

from matplotlib import pyplot as plt


def check_folder(path):
    if not os.path.exists(path):
        os.makedirs(path)
    return path


def fft_2d(img, n_x, n_y):
    """

    Parameters
    ------------
    img: input image
    n_x: DFT dimension of x
    n_y: DFT dimension of y

    Returns
    -----------
    return 2d fft result shape: (n_x, n_y)
    """
    f = np.fft.fft2(img,(n_x, n_y))
    fshift = np.fft.fftshift(f)

    return fshift

def ifft_2d(img):
    """
    Parameters
```

```
    ------------
    img: input image

    Returns
    ------------
    return 2d ifft result shape: (n_x, n_y)
    """
    fshift = np.fft.ifftshift(img)
    result = np.fft.ifft(fshift)
    return result



def save_img(src, title, save_path):
    """
    save the image

    Parameters
    ---------
    src: image
    title: string
    save_path: where to save the image

    """
    global SAVE
    plt.figure()
    plt.imshow(src, cmap="gray")
    plt.title(title)
    plt.axis("off")


    if SAVE and not os.path.exists(save_path + ".png"):
        plt.savefig(save_path + ".png", dpi=200, bbox_inches='tight')

def generate_GLPF(shape=(1200,1200), d_0=100):
    """
    generate 2-D Gaussian lowpass filter


    Parameters
    ---------------
    shape: dimension of the gaussian lowpass filter (M, N)
    d_0: cutoff frequency
```

```
    Returns
    ---------------
    Gaussian lowpass filter shape: (M, N) with cutoff frequency d_0

    """
    m, n = shape
    x = np.linspace(0, m, num=m)
    y = np.linspace(0, n, num=n)
    xv, yv = np.meshgrid(x, y, sparse=False)
    h = np.exp(-((xv - m/2)*(xv - m/2) + (yv - n/2)*(yv - n/2)) / (2. * d_0 *
d_0) )

    return h



def filtering(img, filter):
    """
    Filter the gray-scale image with the given filter

    Parameters
    -------
    img: gray-scale image shape: (m x n)
    filter: filter shape: (2*m, 2*n)


    Returns
    -------
    result_list: list
    which stores images during the process

    """
    result_list = []
    m, n = img.shape
    x = np.arange(0, 2*n)
    xv, yv = np.meshgrid(x,x)

    pad_signal = np.zeros((2 * m, 2 * n))
    pad_signal[0:m, 0:n] = img

    pad_signal_shift = pad_signal * ((-1)**(xv + yv))
```

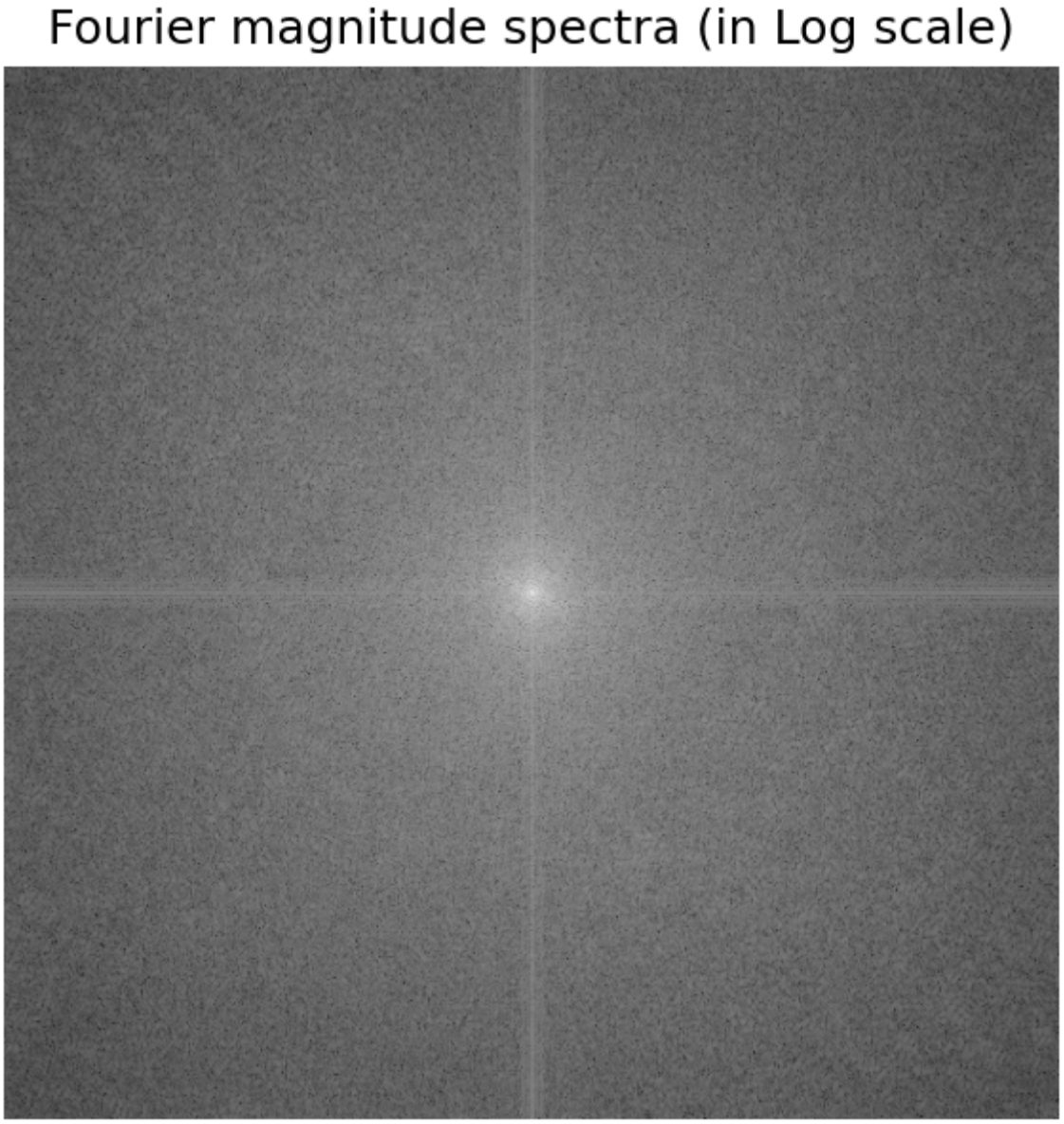
```
    fft_result = np.fft.fft2(pad_signal_shift, (m*2, n*2))

    filtered_result = fft_result * filter

    reconstruct_result = np.fft.ifftshift(np.fft.ifft2(filtered_result*
((-1)**(xv + yv))))

    final_result = np.abs(reconstruct_result[0:m, 0:n])

    result_list.append(pad_signal)
    result_list.append(pad_signal_shift)
    result_list.append(np.abs(filtered_result))
    result_list.append(np.abs(reconstruct_result))
    result_list.append(final_result)

    return result_list

def plot_all_result(img_list, img_title, col=4, row=2):
    """
    plot all figures based on given img_list and img_title

    Params
    ----------
    img_list: list of the images, and shape of each object in list is (h, w, 3).
    img_title: list of the image titles



    """
    fig, ax = plt.subplots(row, col, figsize=(16, 16))
    if row == 1:
        for j in range(col):
            ax[j].imshow(img_list[j ], cmap="gray")
            ax[j].set_title(img_title[j])
            ax[j].axis("off")
    else:
        temp = 0
        for i in range(row):
            for j in range(col):
                if len(img_list)> temp:

                    ax[i, j].imshow(img_list[temp], cmap="gray")
                    ax[i, j].set_title(img_title[temp])
```

```
                    ax[i, j].axis("off")
                    temp = temp+1
    plt.show()

def find_highest_k_freq(img, k):
    """
    find the (u, v) of highest k frequency in the half-left region of the image

    Params
    -------
    img: DFT result shape:(M, N)
    k: number of how many highest (u, v) pairs would return

    """
    m, n = img.shape

    left_half = np.abs(img[:m, :n//2])
    print(left_half.shape)
    indices =  np.argpartition(left_half.flatten(), -2)[-k:]
    indices = np.vstack(np.unravel_index(indices, left_half.shape)).T
    return indices

def save_csv(data, filename):
    with open((filename+ 'most_freq_(u,v).csv'), 'w', newline='') as f:
        writer = csv.writer(f)
        writer.writerow(['u', 'v'])
        writer.writerows(data)


if __name__ == "__main__":


    parser = argparse.ArgumentParser()
    parser.add_argument("--save", help="Whether to save the image",
action="store_true")
    args = parser.parse_args()

    SAVE = args.save

    img_save_path = os.getcwd()
    print(img_save_path)
```

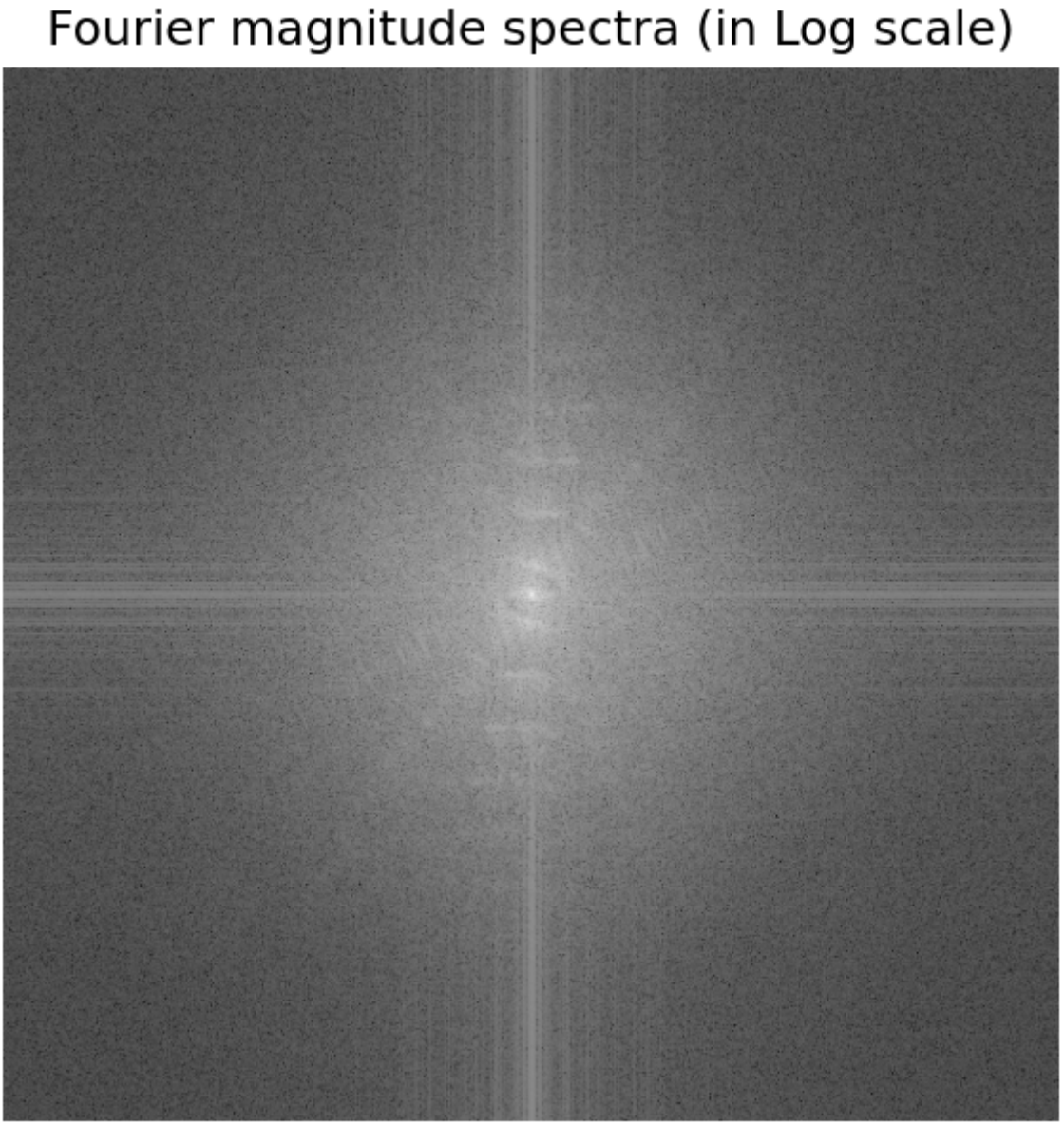
```
    # Load images
    img_fruit = cv2.imread('fruit.tif', 0)
    img_kid = cv2.imread('kid.tif', 0)

    # Do the 2D-DFT of the images with fft
    fruit_fft = fft_2d(img_fruit, 600, 600)
    kid_fft = fft_2d(img_kid, 600, 600)

    fruit_magnitude_spectrum = 20 * np.log(np.abs(fruit_fft))
    kid_magnitude_spectrum = 20 * np.log(np.abs(kid_fft))

    # Generate the Gaussian lowpass and highpass filter
    glpf = generate_GLPF(shape=(1200,1200), d_0=200)
    ghpf = 1 - glpf

    # filter the image with the given filter
    kid_lpf = filtering(img_kid, glpf)
    kid_hpf = filtering(img_kid, ghpf)
    fruit_lpf = filtering(img_fruit, glpf)
    fruit_hpf = filtering(img_fruit, ghpf)

    kid_lpf.insert(0,img_kid)
    title_list = ["1", "2", "3", "4", "5","6"]

    #plot all result
    plot_all_result(kid_lpf, title_list, col=4, row=2)

    # save the (u, v) coordinates of the highest 25 frequencies in the half left
region of (b)
    fruit_csv = find_highest_k_freq(fruit_fft, 25)
    kid_csv = find_highest_k_freq(kid_magnitude_spectrum, 25)
    save_csv(fruit_csv, "fruit_")
    save_csv(kid_csv, "kid_")

    # save the result images
    save_img(fruit_magnitude_spectrum, "Fourier magnitude spectra (in Log
scale)", "img_fruit/fruit_fourier_magnitude_spectrum")
    save_img(kid_magnitude_spectrum, "Fourier magnitude spectra (in Log scale)",
"img_kid/kid_fourier_magnitude_spectrum")
    save_img(fruit_lpf[-1], "Gaussian lowpass filter result",
"img_fruit/fruit_glpf")
    save_img(kid_lpf[-1], "Gaussian lowpass filter result", "img_kid/kid_glpf")
```

```
    save_img(fruit_hpf[-1], "Gaussian highpass filter result",
"img_fruit/fruit_ghpf")
    save_img(kid_hpf[-1], "Gaussian highpass filter result", "img_kid/kid_ghpf")

    # whether to show the saved images
    # plt.show()
```

(b) Fruit ‘s Fourier magnitude spectra

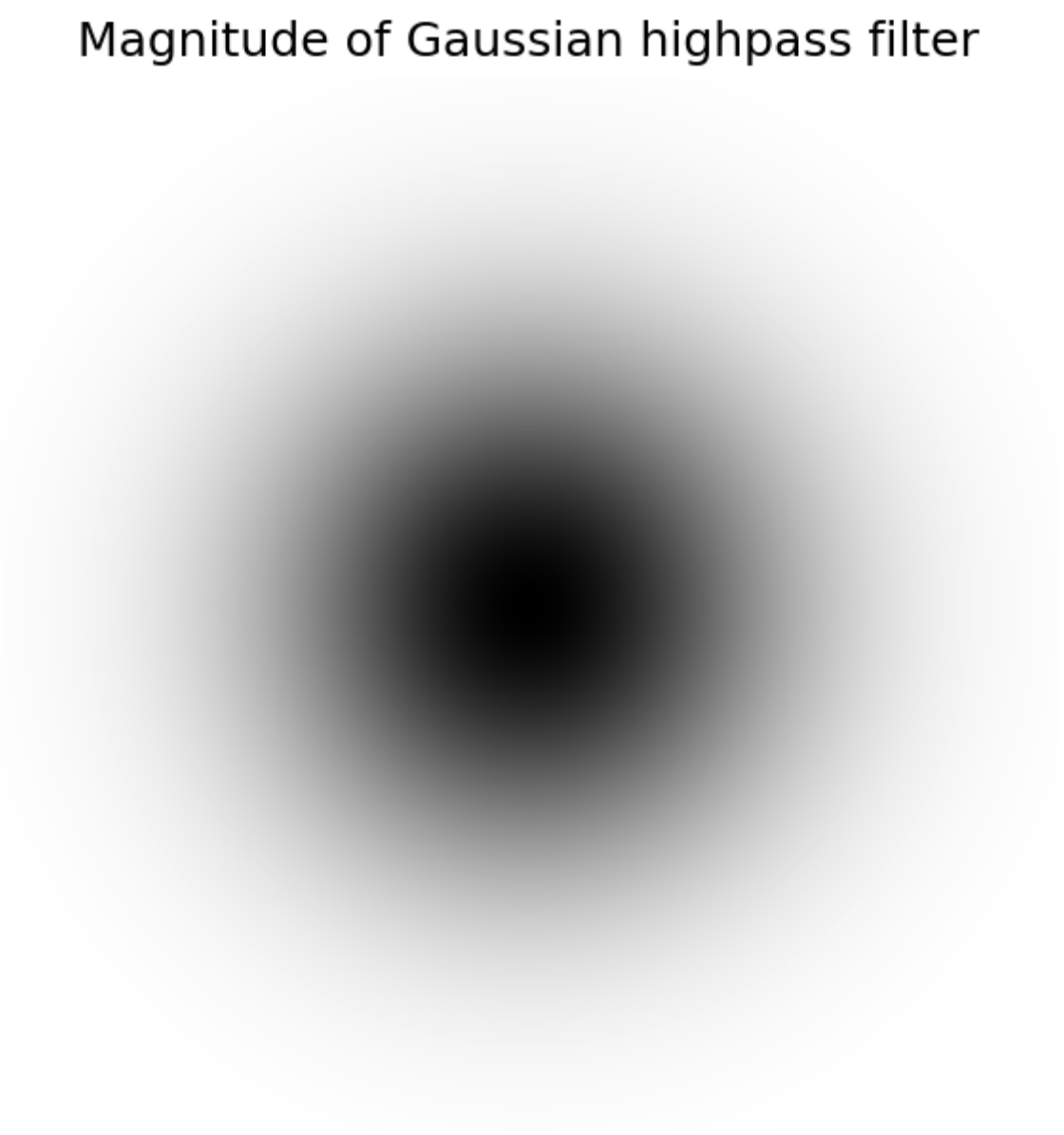


Kid ‘s Fourier magnitude spectra



(c) Magnitude response of Gaussian lowpass filter

Magnitude response of Gaussian highpass filter



(d) Filtered results

Gaussian highpass filter result

Gaussian lowpass filter result

Gaussian highpass filter result

(e)Tables of top 25 DFT frequencies (u,v) of b) in the left half frequency region

kid: (ascending order)

u,v

301,296

316,298

300,298

300,295

300,294
299,298
299,297
299,294
302,299
298,298
298,297
298,294
298,292
297,299
297,296
296,298
296,296
288,299
315,297
315,298
316,296
298,299
299,299
300,299
301,299

fruit: (ascending order)

u,v
295,291
296,298
296,297
296,296
295,296

296,293
296,292
299,298
302,297
295,299
304,299
297,298
298,299
300,295
303,299
302,299
296,294
301,294
299,299
296,299
303,297
300,298
300,297
301,297
300,299