Project 2

ID：311512064　　　name：鄧書桓

(a)code

```
import cv2
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from operator import itemgetter

if __name__ == "__main__":

    img = cv2.imread('kid.tif', 0)

    # generate 2d fourier type
    f_img = np.fft.fft2(img)
    # shift picture
    img_shift = np.fft.fftshift(f_img)

    # (b)
    # turn into magnitute specturm
    magnitude_spectrum = 20*np.log(np.abs(img_shift))
    PILimage = Image.fromarray(magnitude_spectrum)
    PILimage = PILimage.convert('RGB')
    PILimage.save("img/(b)kid_magnitude_spectrum.png", dpi=(150, 150))

    # padding to 1200*1200
    img_padding = cv2.copyMakeBorder(
        img, 0, 600, 0, 600, cv2.BORDER_CONSTANT)
    # generate 2d fourier type
    f_img_padding = np.fft.fft2(img_padding)
    # shift picture
    img_fshift_padding = np.fft.fftshift(f_img_padding)
    # turn into magnitute specturm
    magnitude_spectrum_padding = 20*np.log(np.abs(img_fshift_padding))

    # producing HPF, LPF
    M, N = img_padding.shape
    H_LP = np.zeros((M, N), dtype=np.float32)
    # ( (100 **2) * pi )/ 600**2 = ( (D0**2) * pi)/1200*2, D0 = 200
    D0 = 200
    for u in range(M):
        for v in range(N):
            D = np.sqrt((u - M/2)**2 + (v - N/2)**2)
            H_LP[u, v] = np.exp(-(D**2) / (2 * (D0**2)))
```

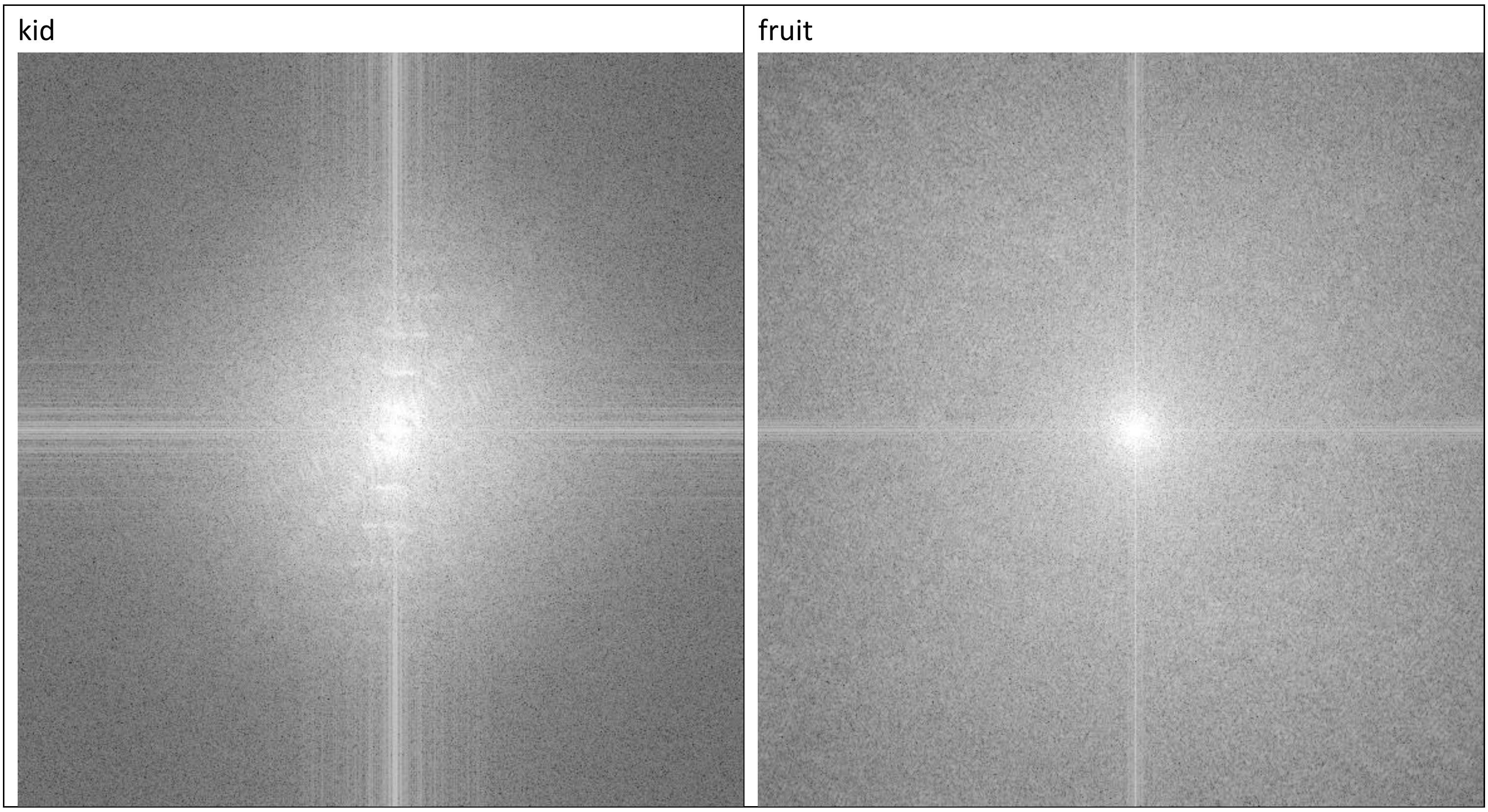
```python
Gshift_LP = img_fshift_padding * H_LP
G_LP = np.fft.ifftshift(Gshift_LP)
g_LP = np.abs(np.fft.ifft2(G_LP))
# PILimage = Image.fromarray(np.abs(Gshift_LP).astype(np.uint8))
# PILimage.save("img/kid_Gshift_LP.png", dpi=(150, 150))

# produce HPF
H_HP = 1 - H_LP
Gshift_HP = img_fshift_padding * H_HP
G_HP = np.fft.ifftshift(Gshift_HP)
g_HP = np.abs(np.fft.ifft2(G_HP))
# PILimage = Image.fromarray(np.abs(Gshift_HP).astype(np.uint8))
# PILimage.save("img/kid_Gshift_HP.png", dpi=(150, 150))

PILimage = Image.fromarray((H_LP*255).astype(np.uint8))
PILimage.save("img/kid_LPF.png", dpi=(150, 150))
PILimage = Image.fromarray((H_HP*255).astype(np.uint8))
PILimage.save("img/kid_HPF.png", dpi=(150, 150))
PILimage = Image.fromarray(g_LP[0:600, 0:600].astype(np.uint8))
PILimage.save("img/kid_output_LPF.png", dpi=(150, 150))
PILimage = Image.fromarray(g_HP[0:600, 0:600].astype(np.uint8))
PILimage.save("img/kid_output_HPF.png", dpi=(150, 150))

# (e)
r = 600
c = 300
e_list = np.zeros((r*c, 3))
k = 0
for i in range(r):
    for j in range(c):
        e_list[k] = [j, i, magnitude_spectrum[j, i]]
        k = k+1
sorted_list = sorted(e_list, key=itemgetter(2), reverse=True)
print("(e) tables of top 25 DFT")
for i in range(25):
    print(sorted_list[i][0:2])

plt.subplot(211)
plt.imshow(H_LP, cmap='gray')
plt.title('LPF'), plt.xticks([]), plt.yticks([])
plt.subplot(212)
plt.imshow(H_HP, cmap='gray')
plt.title('HPF'), plt.xticks([]), plt.yticks([])
plt.show()
```

```
    plt.subplot(321)
    plt.imshow(img, cmap='gray')
    plt.title('Originnal img')
    plt.subplot(322)
    plt.imshow(magnitude_spectrum, cmap='gray')
    plt.title('(b)magnitude_spectrum')
    plt.subplot(323)
    plt.imshow(np.abs(Gshift_LP), cmap='gray')
    plt.title('output specturm of Gaussian LPF')
    plt.subplot(324)
    plt.imshow(np.abs(Gshift_LP), cmap='gray')
    plt.title('output specturm of Gaussian HPF')
    plt.subplot(325)
    plt.imshow(g_LP[0:600, 0:600], cmap='gray')
    plt.title('output of Gaussian LPF')
    plt.subplot(326)
    plt.imshow(g_HP[0:600, 0:600], cmap='gray')
    plt.title('output of Gaussian HPF')
    plt.axis('off')
    plt.show()
######################################################

    img = cv2.imread('fruit.tif', 0)

    # generate 2d fourier type
    f_img = np.fft.fft2(img)
    # shift picture
    img_shift = np.fft.fftshift(f_img)

    # (b)
    # turn into magnitute specturm
    magnitude_spectrum = 20*np.log(np.abs(img_shift))
    PILimage = Image.fromarray(magnitude_spectrum)
    PILimage = PILimage.convert('RGB')
    PILimage.save("img/(b)fruit_magnitude_spectrum.png", dpi=(150, 150))

    # padding to 1200*1200
    img_padding = cv2.copyMakeBorder(
        img, 0, 600, 0, 600, cv2.BORDER_CONSTANT)
    # generate 2d fourier type
    f_img_padding = np.fft.fft2(img_padding)
    # shift picture
    img_fshift_padding = np.fft.fftshift(f_img_padding)
    # turn into magnitute specturm
    magnitude_spectrum_padding = 20*np.log(np.abs(img_fshift_padding))
```

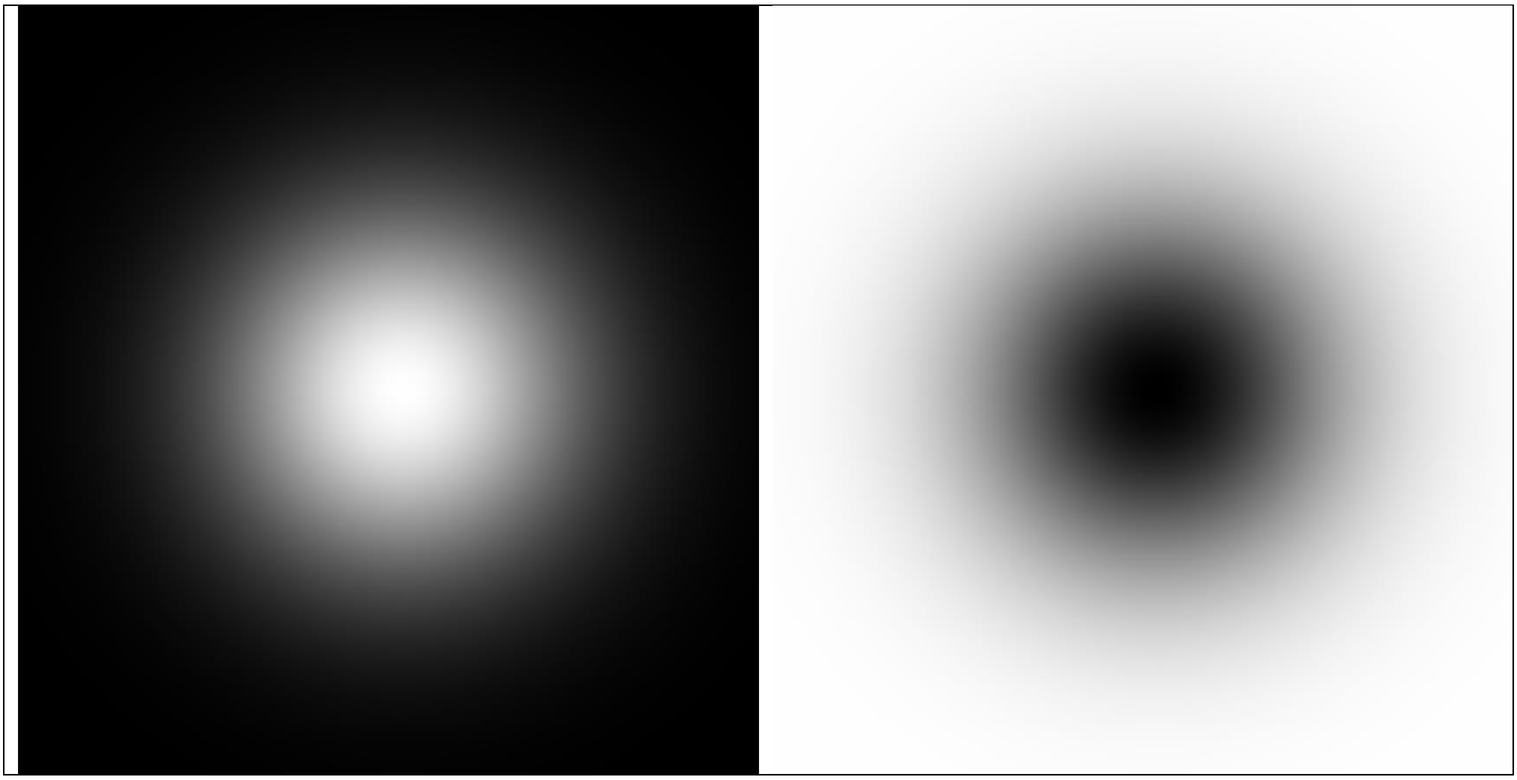
```python
# producing HPF, LPF
M, N = img_padding.shape
H_LP = np.zeros((M, N), dtype=np.float32)
# ( (100 **2) * pi )/ 600**2 = ( (D0**2) * pi)/1200*2, D0 = 200
D0 = 200
for u in range(M):
    for v in range(N):
        D = np.sqrt((u - M/2)**2 + (v - N/2)**2)
        H_LP[u, v] = np.exp(-(D**2) / (2 * (D0**2)))

Gshift_LP = img_fshift_padding * H_LP
G_LP = np.fft.ifftshift(Gshift_LP)
g_LP = np.abs(np.fft.ifft2(G_LP))
# PILimage = Image.fromarray(np.abs(Gshift_LP).astype(np.uint8))
# PILimage.save("img/kid_Gshift_LP.png", dpi=(150, 150))

# produce HPF
H_HP = 1 - H_LP
Gshift_HP = img_fshift_padding * H_HP
G_HP = np.fft.ifftshift(Gshift_HP)
g_HP = np.abs(np.fft.ifft2(G_HP))
# PILimage = Image.fromarray(np.abs(Gshift_HP).astype(np.uint8))
# PILimage.save("img/kid_Gshift_HP.png", dpi=(150, 150))

PILimage = Image.fromarray((H_LP*255).astype(np.uint8))
PILimage.save("img/fruit_LPF.png", dpi=(150, 150))
PILimage = Image.fromarray((H_HP*255).astype(np.uint8))
PILimage.save("img/fruit_HPF.png", dpi=(150, 150))
PILimage = Image.fromarray(g_LP[0:600, 0:600].astype(np.uint8))
PILimage.save("img/fruit_output_LPF.png", dpi=(150, 150))
PILimage = Image.fromarray(g_HP[0:600, 0:600].astype(np.uint8))
PILimage.save("img/fruit_output_HPF.png", dpi=(150, 150))

# (e)
r = 600
c = 300
e_list = np.zeros((r*c, 3))
k = 0
for i in range(r):
    for j in range(c):
        e_list[k] = [j, i, magnitude_spectrum[j, i]]
        k = k+1
sorted_list = sorted(e_list, key=itemgetter(2), reverse=True)
print("(e) tables of top 25 DFT")
```

```
for i in range(25):
    print(sorted_list[i][0:2])

plt.subplot(211)
plt.imshow(H_LP, cmap='gray')
plt.title('LPF'), plt.xticks([]), plt.yticks([])
plt.subplot(212)
plt.imshow(H_HP, cmap='gray')
plt.title('HPF'), plt.xticks([]), plt.yticks([])
plt.show()

plt.subplot(321)
plt.imshow(img, cmap='gray')
plt.title('Originnal img')
plt.subplot(322)
plt.imshow(magnitude_spectrum, cmap='gray')
plt.title('(b)magnitude_spectrum')
plt.subplot(323)
plt.imshow(np.abs(Gshift_LP), cmap='gray')
plt.title('output specturm of Gaussian LPF')
plt.subplot(324)
plt.imshow(np.abs(Gshift_LP), cmap='gray')
plt.title('output specturm of Gaussian HPF')
plt.subplot(325)
plt.imshow(g_LP[0:600, 0:600], cmap='gray')
plt.title('output of Gaussian LPF')
plt.subplot(326)
plt.imshow(g_HP[0:600, 0:600], cmap='gray')
plt.title('output of Gaussian HPF')
plt.axis('off')
plt.show()
```

(b) Fourier magnitude spectra (in Log scale) of kid and fruit



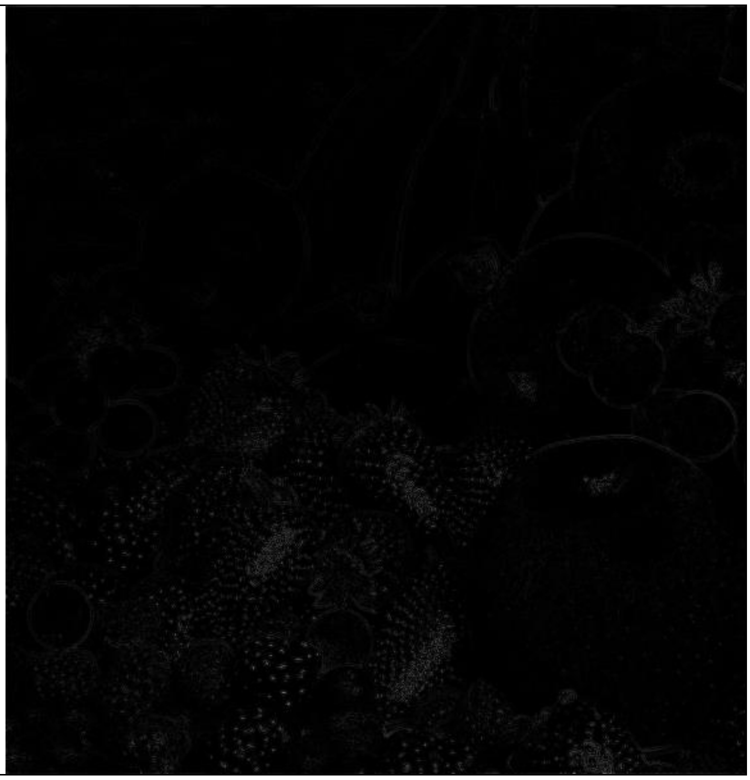
c) Magnitude responses of Gaussian LPF and HPF



(d) 4 output images

Kid LPF, HPF

Furit LPF, HPF



(e) start from left top

Kid

```
[299. 301.]
[299. 300.]
[299. 299.]
[298. 299.]
[298. 300.]
[297. 299.]
[299. 297.]
[298. 302.]
[298. 298.]
[298. 294.]
[298. 301.]
[298. 304.]
[299. 298.]
[296. 302.]
[284. 302.]
[297. 300.]
[299. 294.]
[299. 304.]
[283. 302.]
[296. 296.]
[296. 298.]
[284. 303.]
[283. 300.]
[298. 292.]
[297. 296.]
```

Fruit

```
[299. 300.]
[298. 300.]
[299. 303.]
[296. 299.]
[297. 303.]
[299. 299.]
[295. 299.]
[298. 303.]
[297. 298.]
[299. 306.]
[296. 300.]
[298. 299.]
[298. 301.]
[296. 301.]
[297. 301.]
[296. 294.]
[299. 298.]
[297. 302.]
[299. 296.]
[296. 296.]
[294. 301.]
[297. 296.]
[299. 297.]
[298. 305.]
[296. 298.]
```