Project 2

ID：0811562　　name：何祁恩

(a)code

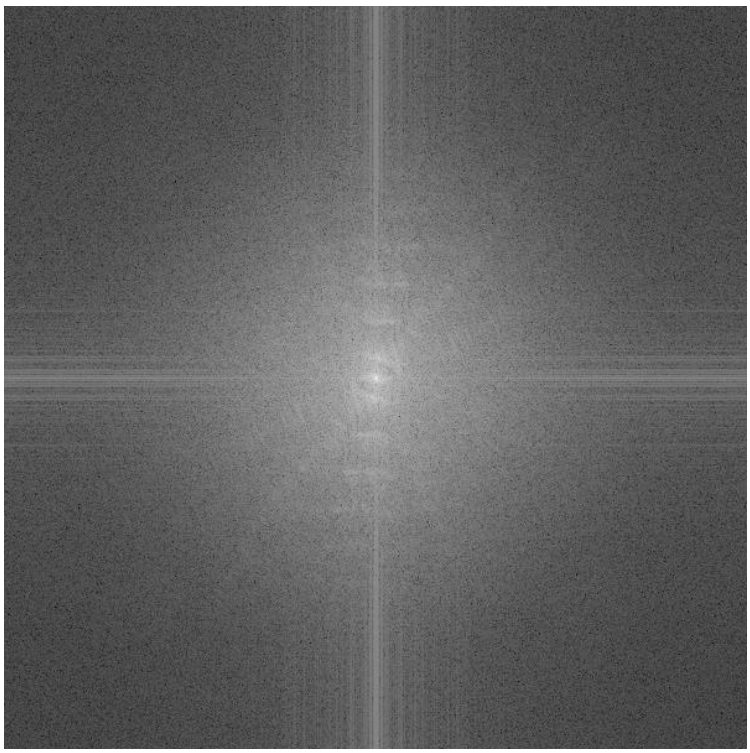
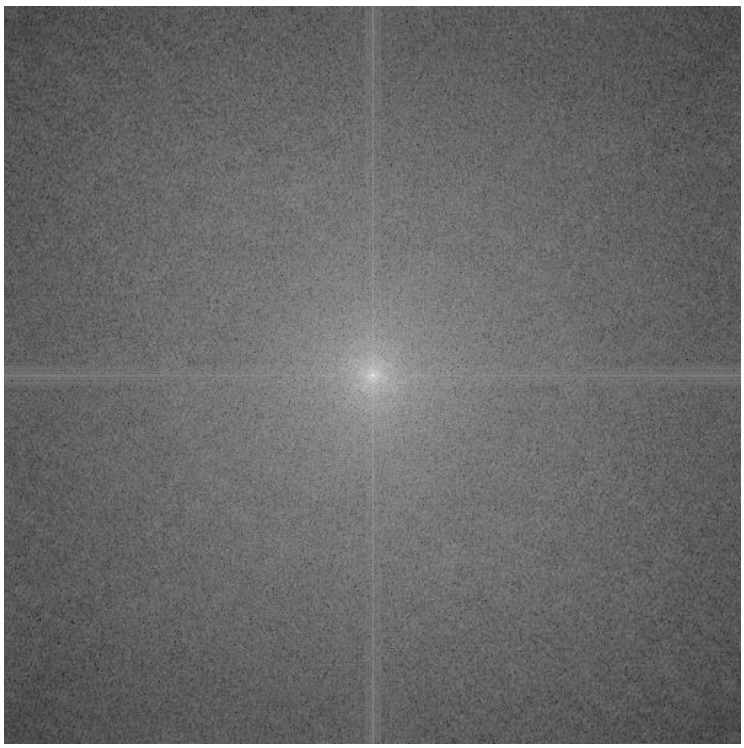
```
clc;
clear;
close all;

filename = 'kid';
f = imread([filename,'.tif']);
[M, N] = size(f);

%(b)
F = fft2(double(f));
F_shift = fftshift(F);

%pading to avoid wraparound error
f_pad = zeros(2*M, 2*N);
f_pad(1:M, 1:N) = f;
f_pad_shift = zeros(2*M, 2*N);

for i = 1 : 2*M
    for j = 1 : 2*N
        f_pad_shift(i, j) = f_pad(i, j) * ((-1)^((i-1)+(j-1)));
    end
end
F_pad_shift = fft2(double(f_pad_shift));

%create Gaussian LPF and HPF (1200 x 1200 FILTER)
LPF = zeros(2*M,2*N);
D0 = 200;                                                    %D0 = 200!!!!!!!!!!!!!!!!!!!! not 100

for u = 1:2*M
    for v = 1:2*N
        D2 = (u-M)^2 + (v-N)^2;                              %center change to (M, N)
        LPF(u,v) = exp(-1*D2/(2*D0*D0));
    end
end

HPF = 1 - LPF;
```

```

%Image pass LPF
G_LPF_shift = F_pad_shift .* LPF;
g_LPF_shift = ifft2(double(G_LPF_shift));
g_LPF = zeros(2*M, 2*N);
for i = 1 : 2*M
    for j = 1 : 2*N
        g_LPF(i, j) = g_LPF_shift(i, j) * ((-1)^((i-1)+(j-1)));
    end
end


Re_g_LPF = real(g_LPF(1:M,1:N));


%Image pass HPF
G_HPF_shift = F_pad_shift .* HPF;
g_HPF_shift = ifft2(double(G_HPF_shift));
g_HPF = zeros(2*M, 2*N);
for i = 1 : 2*M
    for j = 1 : 2*N
        g_HPF(i, j) = g_HPF_shift(i, j) * ((-1)^((i-1)+(j-1)));
    end
end

Re_g_HPF = real(g_HPF(1:M,1:N));
```

```
63
64  %get top25 abs(Fshift)
65  Abs_F_shift = abs(F_shift);
66  Abs_F_shift = Abs_F_shift(1:M,1:N/2);
67  max_abs = zeros(25,1);
68  u = zeros(25,1);
69  v = zeros(25,1);
70  idx = 1;
71
72  while idx <= 25
73      sort_abs = sort(Abs_F_shift(:));
74      max_value = max(sort_abs);
75      [row,col] = find(Abs_F_shift == max_value);
76      [length, a] = size(row);
77      for j = 0:length - 1
78          u(idx+j,1) = row(j+1,1) - 1;
79          v(idx+j,1) = col(j+1,1) - 1;
80          max_abs(idx+j,1) = max_value;
81          Abs_F_shift(row(j+1,1),col(j+1,1)) = -inf;
82      end
83      idx = idx + length;
84  end
```

```
85
86   figure(1)
87   imshow(mat2gray(log10(1+ abs(F_shift))));
88   img1 = getimage(gcf);
89   imwrite(img1,['result/',filename,'_(600×600_DFT).tiff'], 'tiff', 'Resolution', 150)
90
91   figure(2)
92   imshow(LPF);
93   img2 = getimage(gcf);
94   imwrite(img2,['result/',filename,'_(1200×1200_LPF).tiff'], 'tiff', 'Resolution', 150)
95
96   figure(3)
97   imshow(HPF);
98   img3 = getimage(gcf);
99   imwrite(img3,['result/',filename,'_(1200×1200_HPF).tiff'], 'tiff', 'Resolution', 150)
100
101  figure(4)
102  imshow(uint8(Re_g_LPF));
103  img4 = getimage(gcf);
104  imwrite(img4,['result/',filename,'_(600×600_LPF_output).tiff'], 'tiff', 'Resolution', 150)
105
106  figure(5)
107  imshow(uint8(Re_g_HPF));
108  img5 = getimage(gcf);
109  imwrite(img5,['result/',filename,'_(600×600_HPF_output).tiff'], 'tiff', 'Resolution', 150)
110
```

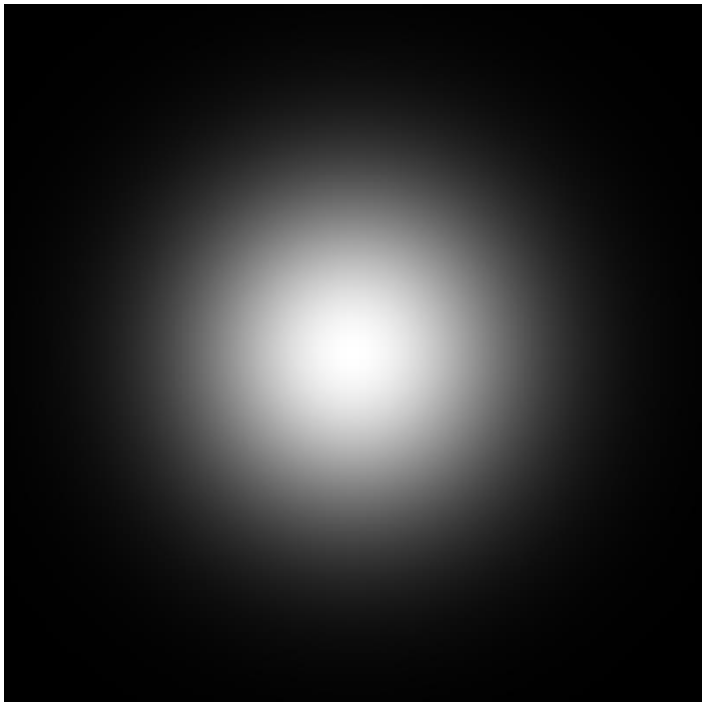
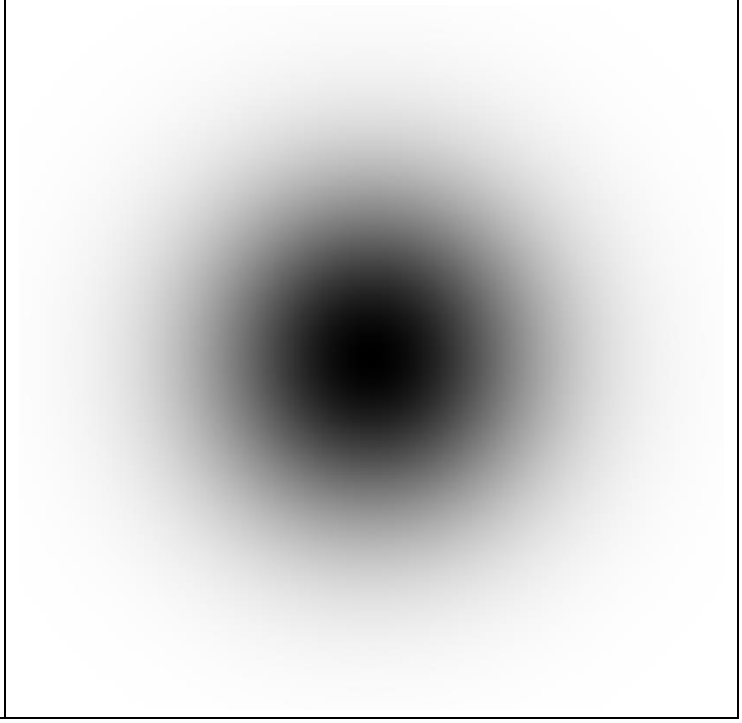
(b) Fourier magnitude spectra (in Log scale) of kid and fruit (600x600 DFT)

kid	fruit
	

(c) Magnitude responses of Gaussian LPF and HPF D0 = 200! Not 100!!!!!!!!!!!

D0 = 100 pixels based on the original image size (600 * 600), but we perform a gaussian filter on (1200 * 1200) to avoid wraparound error, so we should choose D0' by calculating :

(100^2 * pi) / 600^2 = ((D0')^2 * pi) / 1200^2 -> D0' = 200 pixels for expanding the image to double size

LPF (1200x1200)	HPF (1200 x 1200)
	

(d) 4 output images

Kid LPF, HPF

Fruit LPF, HPF

The output image from HPF is so black, but we still can see some edges detected.

(e) start from left top

Kid in descending order:

u	v
301	299
300	299
299	299
298	299
297	299
299	297
302	298
298	298
298	294
302	299
302	296
299	298
304	298
316	298
299	294
301	296
317	298
296	296
296	298
316	297
300	294
298	292
297	296
298	297
301	297

Fruit in descending order

u	v
300	299
301	297
300	298
296	299
303	297
300	297
299	299
295	299
302	297
297	298
301	294
298	299
300	295
302	299
304	299
303	299
296	294
299	298
303	298
299	296
296	296
306	299
297	296
299	297
302	295

Feedback:

I think I didn't do anything wrong in this project, I follow the PPT process given in class. However, I think the given D0 is not so good to observe the output image after Gaussian LPF and HPF. I do check the definition of 2D DFT taught in class has the same formula as Matlab implementation:

2D DFT pair f: non periodic function (M * N)

$$F(u,v)=\begin{cases}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)W_M^{ux}W_N^{vy}, & \begin{matrix}0\le u\le M-1\\ 0\le v\le N-1\end{matrix}\\ 0,\ \text{otherwise}\end{cases}$$

$$W_M = e^{-j\frac{2\pi}{M}}$$ analysis equation

$$f(x,y)=\begin{cases}\frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)W_M^{-ux}W_N^{-vy}, & \begin{matrix}0\le x\le M-1\\ 0\le y\le N-1\end{matrix}\\ 0,\ \text{otherwise}\end{cases}$$

synthesis equation

Matlab implementation of 2D DFT:

2-D Fourier Transform

This formula defines the discrete Fourier transform Y of an m-by-n matrix X:

$$Y_{p+1,q+1}=\sum_{j=0}^{m-1}\sum_{k=0}^{n-1}\omega_m^{jp}\omega_n^{kq}X_{j+1,k+1}$$

ω_m and ω_n are complex roots of unity:

$$\omega_m = e^{-2\pi i/m}$$
$$\omega_n = e^{-2\pi i/n}$$

i is the imaginary unit. p and j are indices that run from 0 to $m-1$, and q and k are indices that run from 0 to $n-1$. This formula shifts the indices for X and Y by 1 to reflect matrix indices in MATLAB®.

2-D Inverse Fourier Transform

This formula defines the discrete inverse Fourier transform X of an m-by-n matrix Y:

$$X_{p,q}=\frac{1}{m}\sum_{j=1}^{m}\frac{1}{n}\sum_{k=1}^{n}\omega_m^{(j-1)(p-1)}\omega_n^{(k-1)(q-1)}Y_{j,k}$$

ω_m and ω_n are complex roots of unity:

$$\omega_m = e^{2\pi i/m}$$
$$\omega_n = e^{2\pi i/n}$$

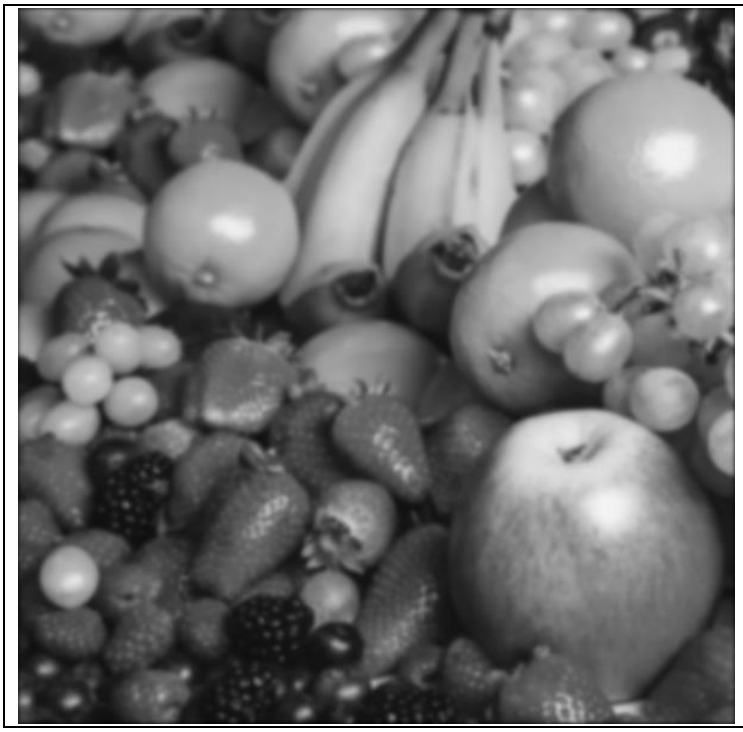
i is the imaginary unit. p runs from 1 to m and q runs from 1 to n.

In python, fft provided by NumPy seems to have a normalized coefficient different from the formula given above. Therefore, I choose Matlab to do this project.

Some other results (without any post-processing):

For Do = 100

Fruit LPF, HPF:

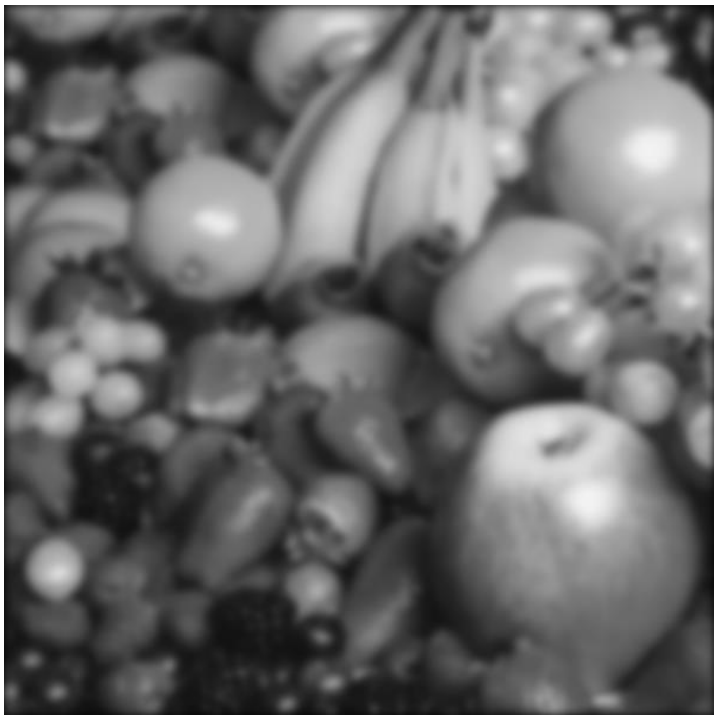



KID LPF, HPF:


The output result of LPF becomes a little blurry, and the output result of HPF becomes clear.

For D0 = 50

Fruit LPF, HPF:




KID LPF, HPF:


When D0 = 50, the output image from LPF becomes very blurry, and the output image from HPF detects more edges clearly. I think the difference between us might be the implementation detail under those package functions.

It is quite fun on this project, and I learn a lot!