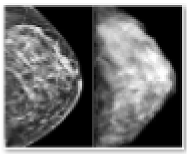
Final project report

1. Down Load 2 medical images such mammograms, ultrasound, xray, etc… and go downtown and take 2 pictures of downtown sky scraper neighborhoods. It is better to have a few buildings (1-3) so your edge detector does not get overwhelmed

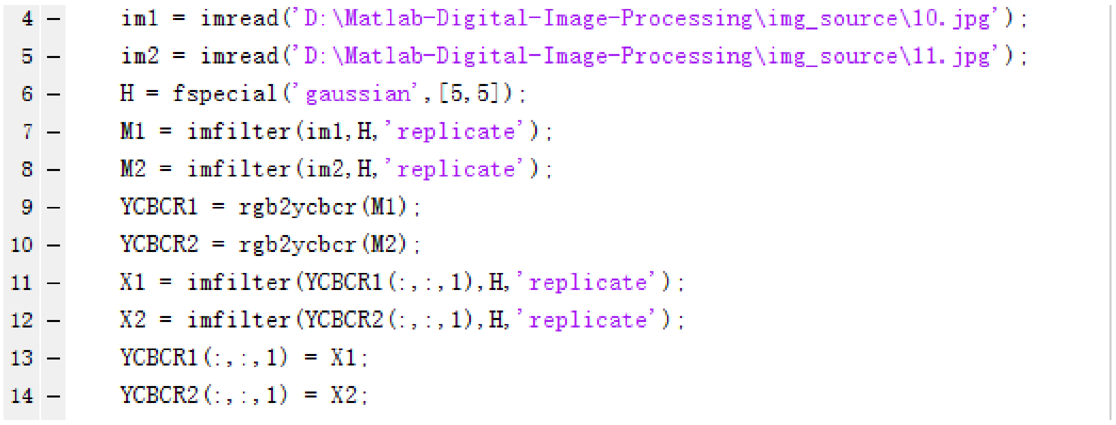

1.jpg

2.jpg

10.jpg

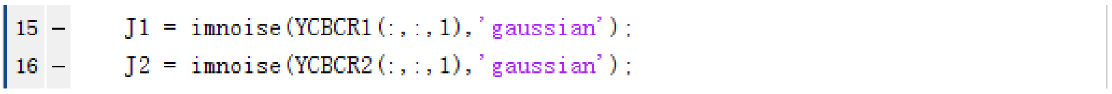
11.jpg

2. If it is a color image convert to YCbCr and only extract the Y channel



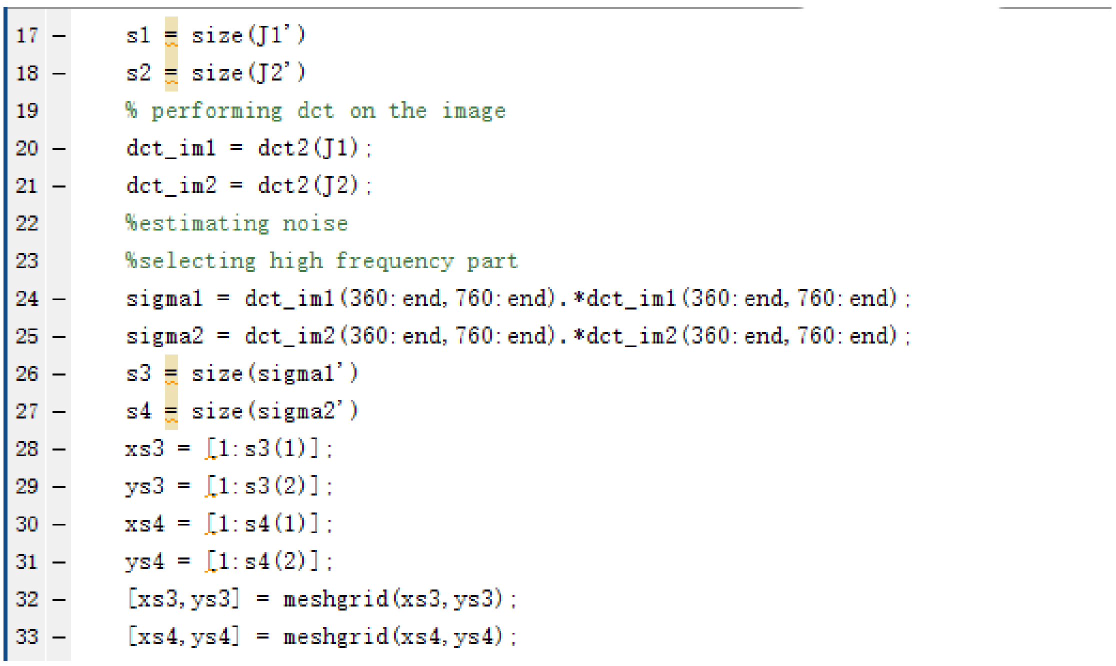
```
im1 = imread('D:\Matlab-Digital-Image-Processing\img_source\10.jpg');
im2 = imread('D:\Matlab-Digital-Image-Processing\img_source\11.jpg');
H = fspecial('gaussian',[5,5]);
M1 = imfilter(im1,H,'replicate');
M2 = imfilter(im2,H,'replicate');
YCBCR1 = rgb2ycbcr(M1);
YCBCR2 = rgb2ycbcr(M2);
X1 = imfilter(YCBCR1(:,:,1),H,'replicate');
X2 = imfilter(YCBCR2(:,:,1),H,'replicate');
YCBCR1(:,:,1) = X1;
YCBCR2(:,:,1) = X2;
```

3. Use 'imnoise' function in MATALB to add Gaussian noise to the images.

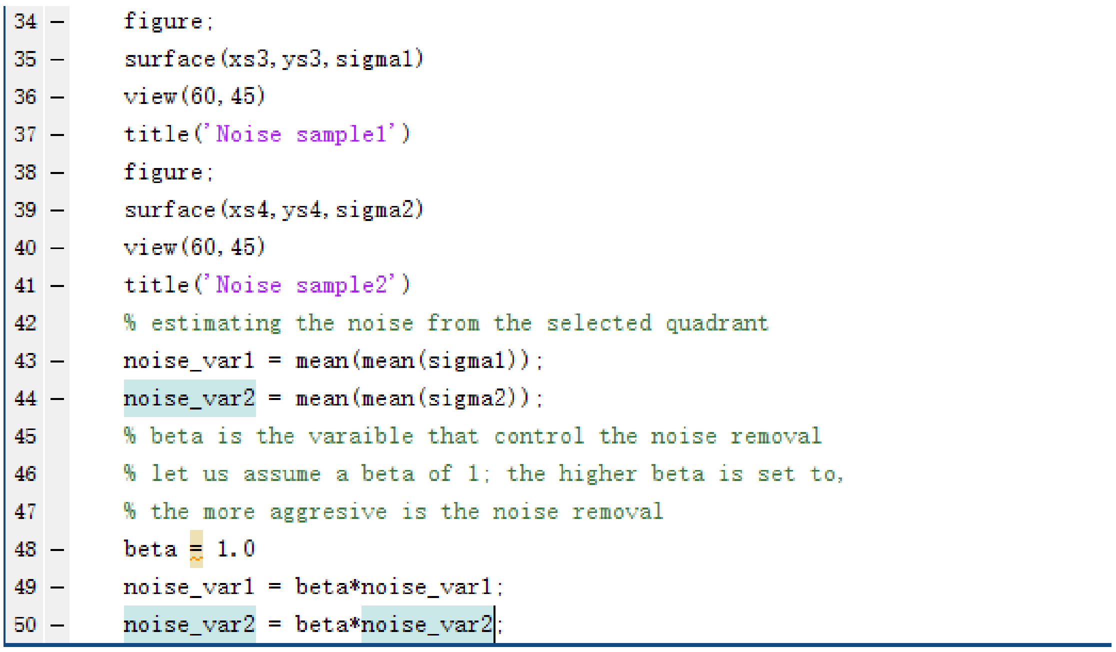


```
J1 = imnoise(YCBCR1(:,:,1),'gaussian');
J2 = imnoise(YCBCR2(:,:,1),'gaussian');
```

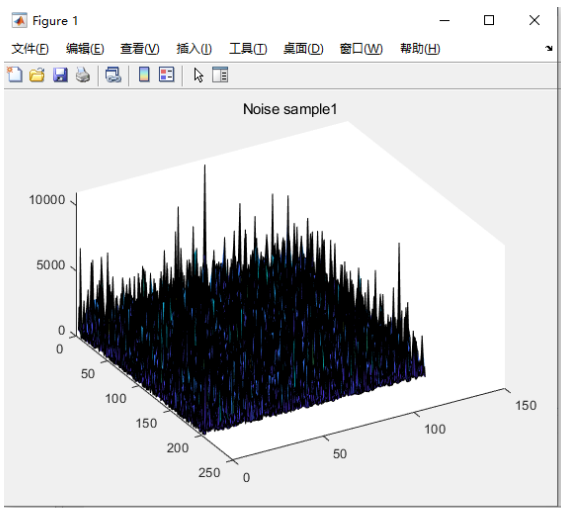
4. Perform 2D DCT and look at the high frequency areas the bottom right quadrant and estimate the noise constant

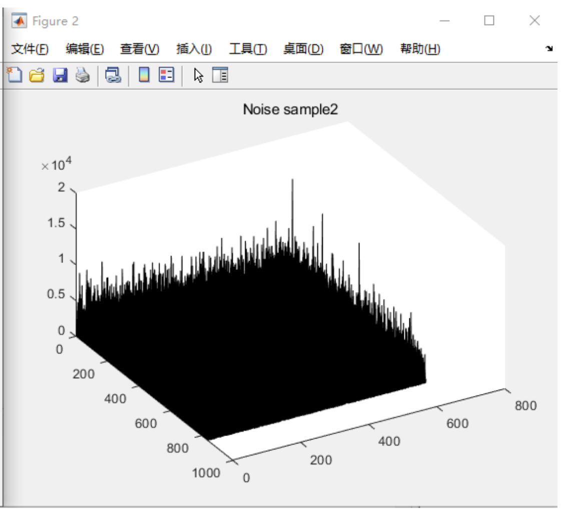


```matlab
s1 = size(J1')
s2 = size(J2')
% performing dct on the image
dct_im1 = dct2(J1);
dct_im2 = dct2(J2);
%estimating noise
%selecting high frequency part
sigma1 = dct_im1(360:end,760:end).*dct_im1(360:end,760:end);
sigma2 = dct_im2(360:end,760:end).*dct_im2(360:end,760:end);
s3 = size(sigma1')
s4 = size(sigma2')
xs3 = [1:s3(1)];
ys3 = [1:s3(2)];
xs4 = [1:s4(1)];
ys4 = [1:s4(2)];
[xs3,ys3] = meshgrid(xs3,ys3);
[xs4,ys4] = meshgrid(xs4,ys4);
```



```matlab
figure;
surface(xs3,ys3,sigma1)
view(60,45)
title('Noise sample1')
figure;
surface(xs4,ys4,sigma2)
view(60,45)
title('Noise sample2')
% estimating the noise from the selected quadrant
noise_var1 = mean(mean(sigma1));
noise_var2 = mean(mean(sigma2));
% beta is the varaible that control the noise removal
% let us assume a beta of 1; the higher beta is set to,
% the more aggresive is the noise removal
beta = 1.0
noise_var1 = beta*noise_var1;
noise_var2 = beta*noise_var2;
```



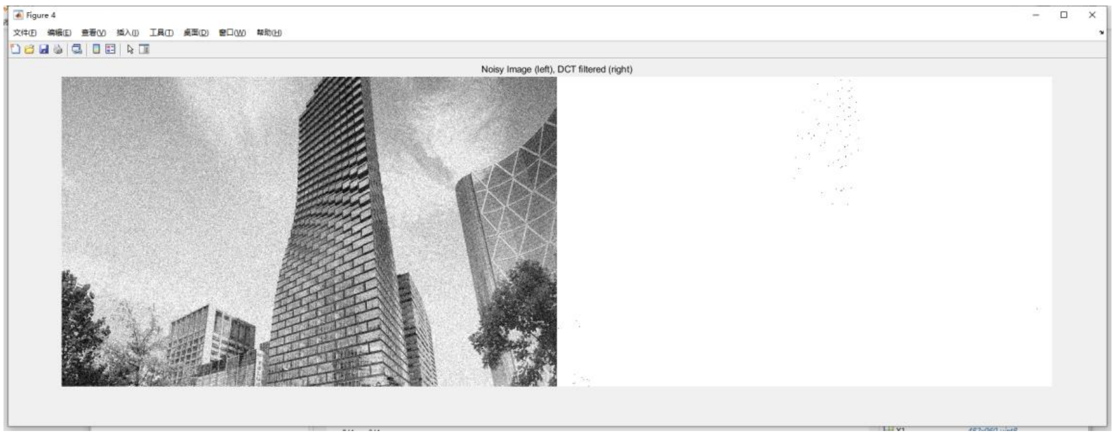


noise_var1	636.9226
noise_var2	615.6696

5. Use this estimated noise constant to perform a DCT Wiener Filter on each image

```
signal_var1 = dct_im1.*dct_im1 + 0.001;
signal_var2 = dct_im2.*dct_im2 + 0.001;
wiener_filter1 = 1 + (noise_var1./signal_var1);
wiener_filter1 = 1./wiener_filter1;
wiener_filter2 = 1 + (noise_var2./signal_var2);
wiener_filter2 = 1./wiener_filter2;
% Apply the Wiener Filter DCT coefficients
filtered_dct1 = dct_im1.*wiener_filter1;
filtered_dct2 = dct_im2.*wiener_filter2;
% Inverse DCT transform to reconstruct image
filtered_dct_image1 = idct2(filtered_dct1);
filtered_dct_image1 = 255.*imadjust(filtered_dct_image1, [], [0,1]);
figure;
imshowpair(J1, uint8(filtered_dct_image1), 'montage');
title('Noisy Image (left), DCT filtered (right)');
filtered_dct_image2 = idct2(filtered_dct2);
filtered_dct_image2 = 255.*imadjust(filtered_dct_image2, [], [0,1]);
```

```
figure;
imshowpair(J1, uint8(filtered_dct_image1), 'montage');
title('Noisy Image (left), DCT filtered (right)');
filtered_dct_image2 = idct2(filtered_dct2);
filtered_dct_image2 = 255.*imadjust(filtered_dct_image2, [], [0,1]);
figure;
imshowpair(J2, uint8(filtered_dct_image2), 'montage');
title('Noisy Image (left), DCT filtered (right)');
```



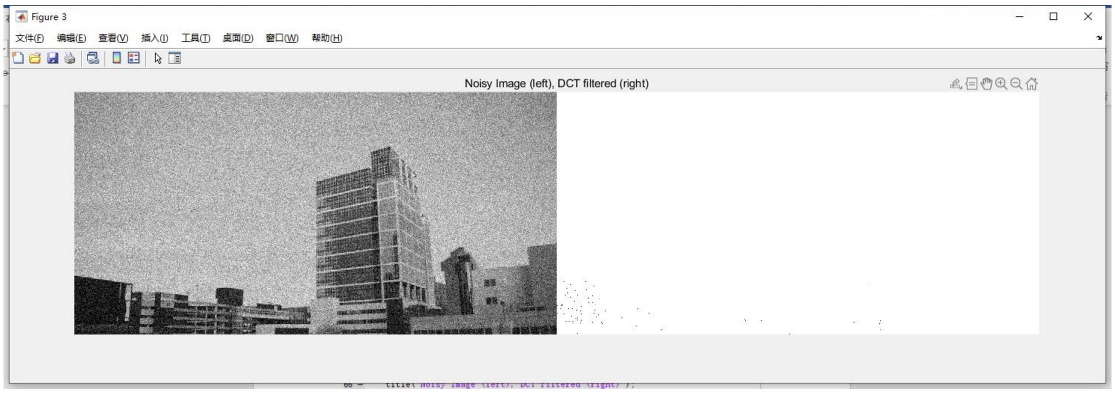
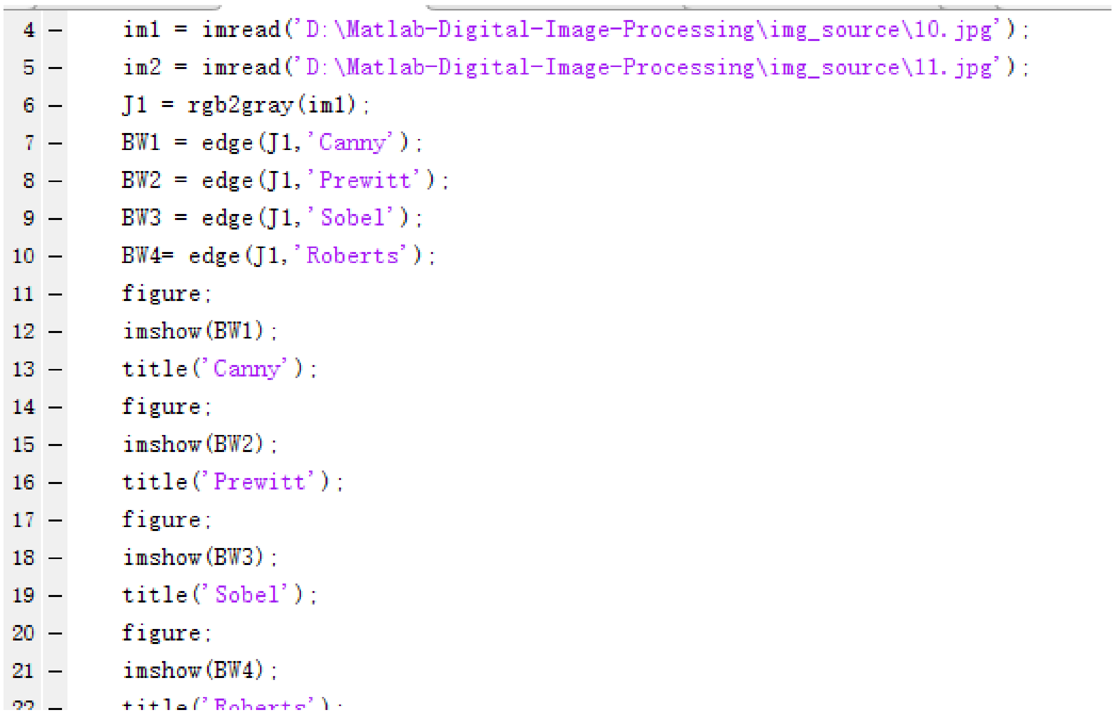


Figure 3
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Noisy Image (left), DCT filtered (right)

6. Use MATALB 'edge' function to detect edges using 'Sobel', 'Prewitt','Roberts',and 'Canny'. For each image you will have 4 segmented images.



```
im1 = imread('D:\Matlab-Digital-Image-Processing\img_source\10.jpg');
im2 = imread('D:\Matlab-Digital-Image-Processing\img_source\11.jpg');
J1 = rgb2gray(im1);
BW1 = edge(J1,'Canny');
BW2 = edge(J1,'Prewitt');
BW3 = edge(J1,'Sobel');
BW4= edge(J1,'Roberts');
figure;
imshow(BW1);
title('Canny');
figure;
imshow(BW2);
title('Prewitt');
figure;
imshow(BW3);
title('Sobel');
figure;
imshow(BW4);
title('Roberts');
```

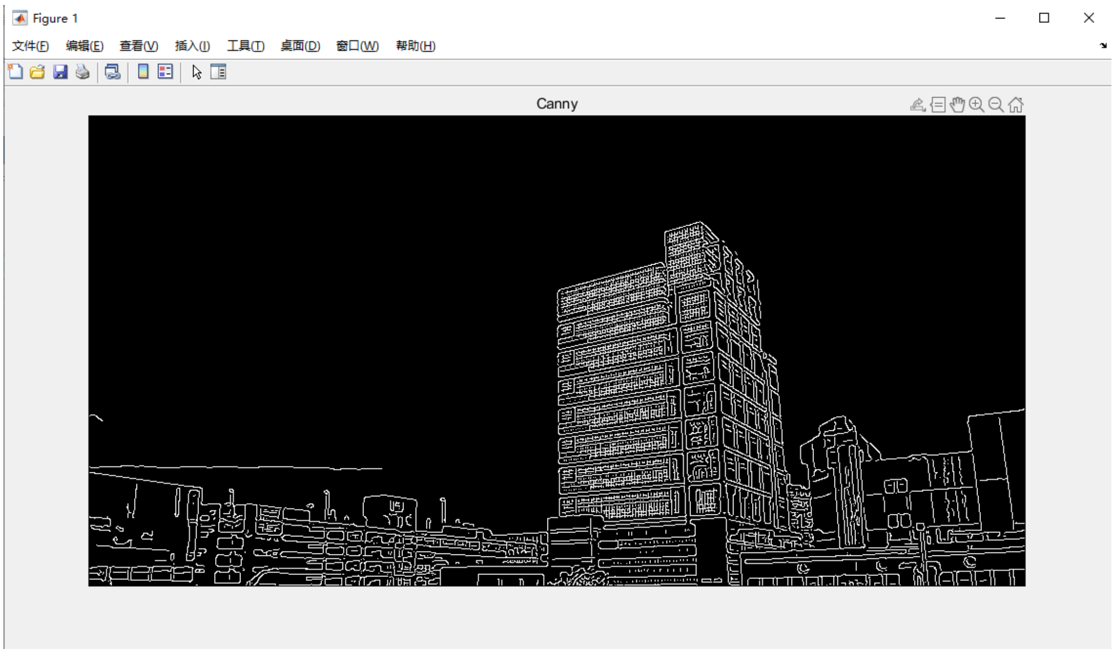


Figure 1
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Canny

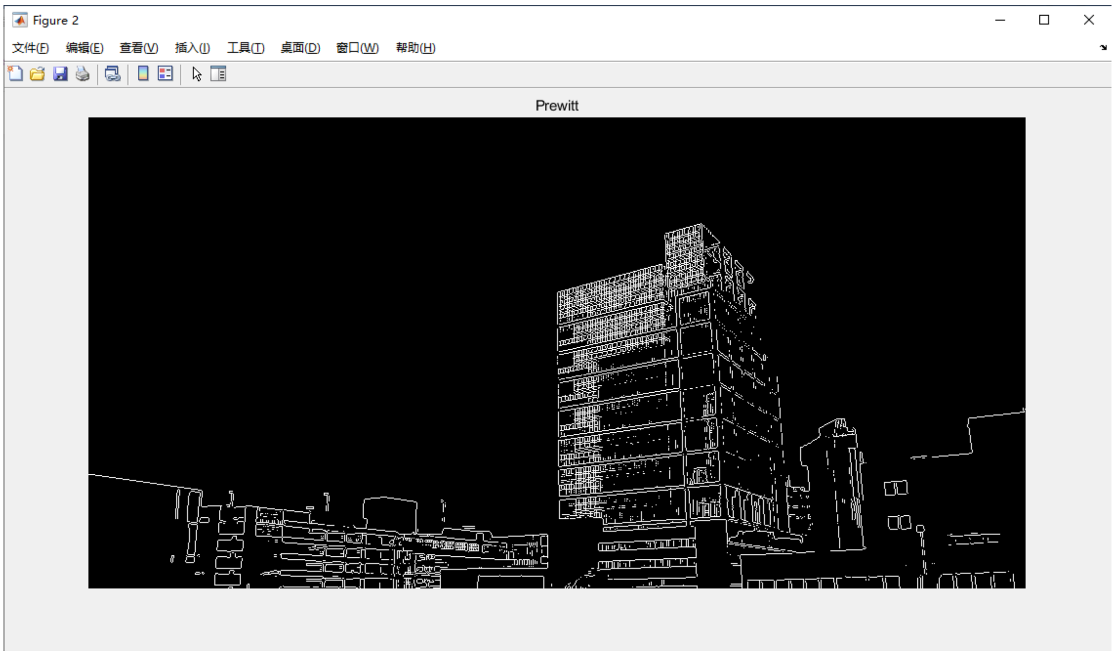


Figure 2
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Prewitt

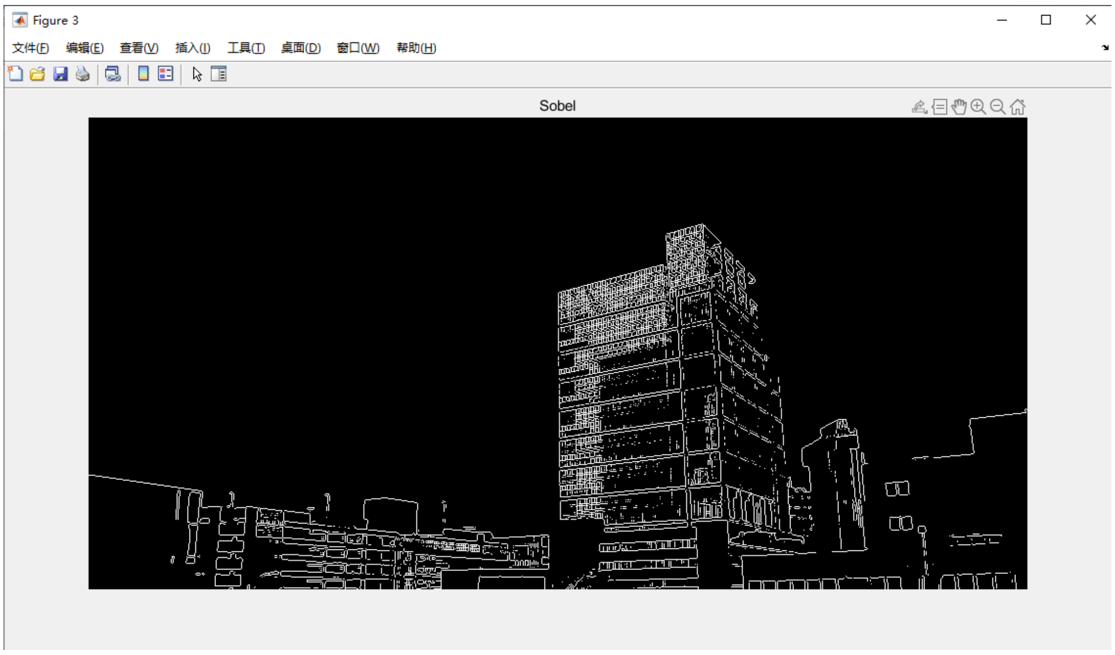


Figure 3
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Sobel

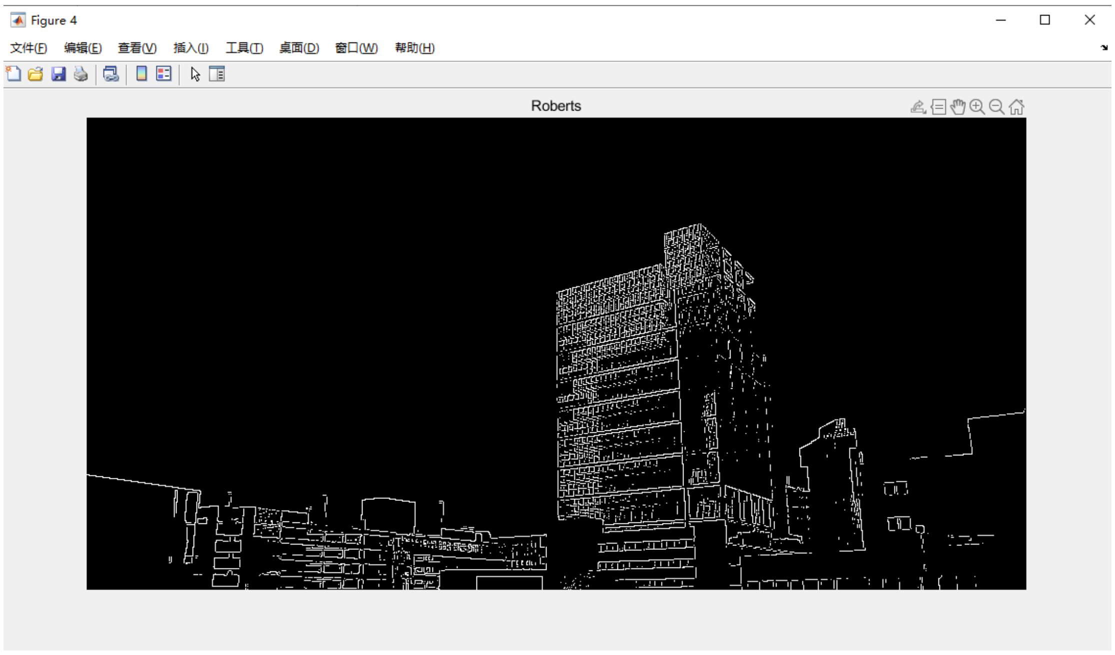
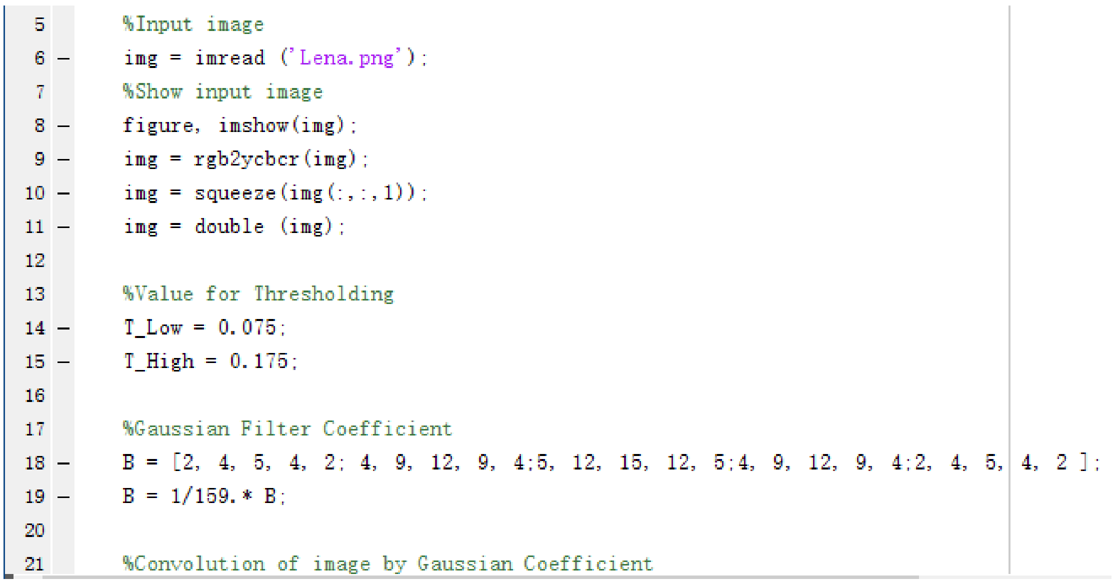


Figure 4
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Roberts

7. Use the MATLAB code provided on Ed 'Canny_v0' to perform a canny edge detection on the 'Lena.png' image also provided. Make sure you optimize the two thresholds in the script.

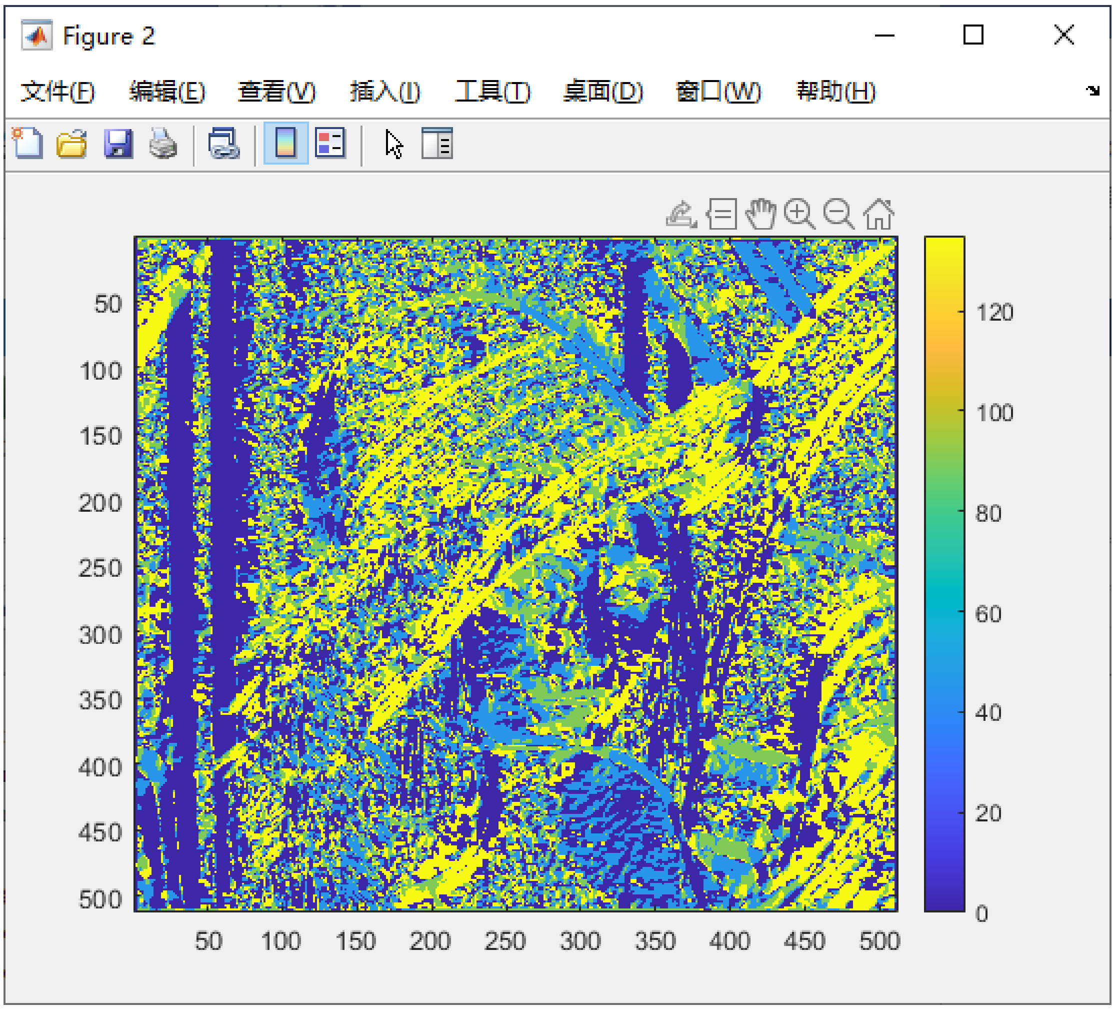


```
%Input image
img = imread ('Lena.png');
%Show input image
figure, imshow(img);
img = rgb2ycbcr(img);
img = squeeze(img(:,:,1));
img = double (img);

%Value for Thresholding
T_Low = 0.075;
T_High = 0.175;

%Gaussian Filter Coefficient
B = [2, 4, 5, 4, 2; 4, 9, 12, 9, 4;5, 12, 15, 12, 5;4, 9, 12, 9, 4;2, 4, 5, 4, 2 ];
B = 1/159.* B;

%Convolution of image by Gaussian Coefficient
```



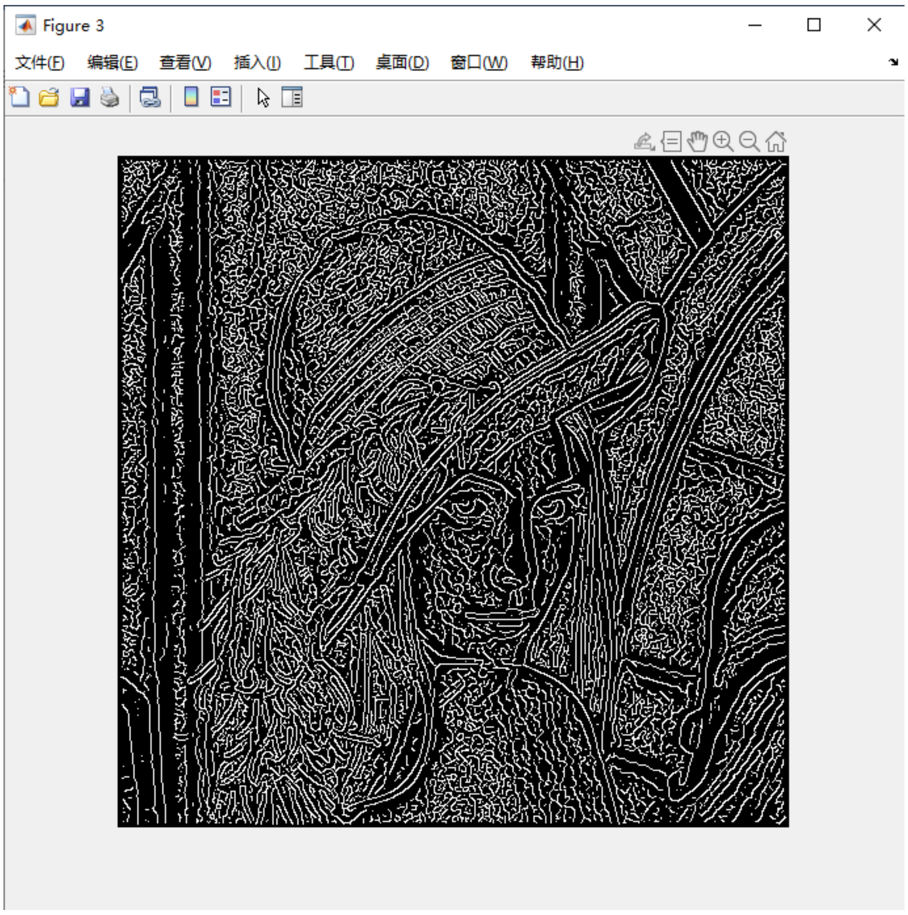


Figure 3
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)

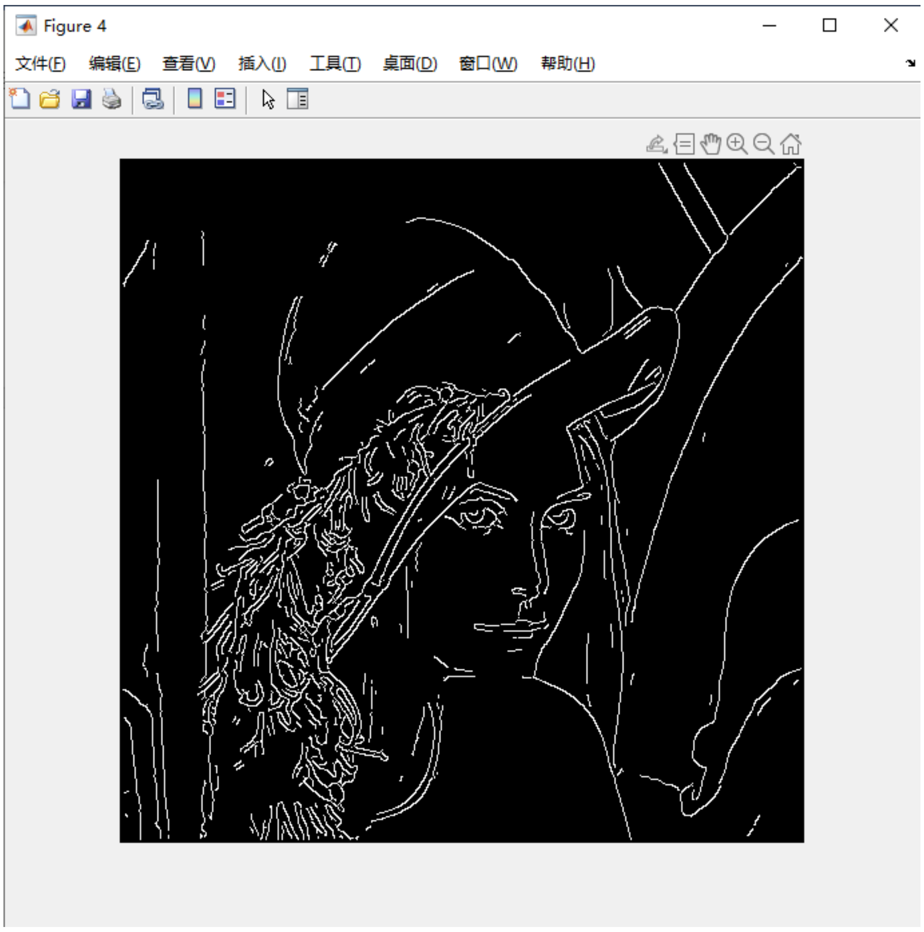
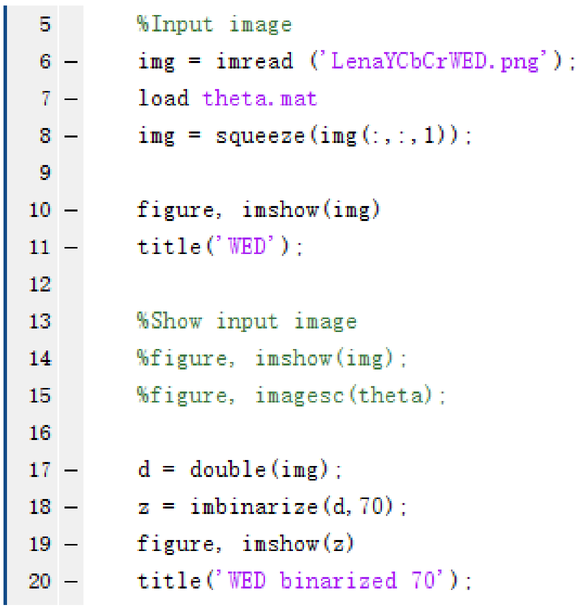


Figure 4
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)

8. Read the provided image 'lenaYCbCrWED' into MATLAB. This image was obtained by using a wide support kernel in the DCT domain to smooth and perform the gradient operations. Run the MATLAB code 'WED_postproc' on this image and optimize the thresholds again. This code only performs the non-maximum suppression and thresholding.



```
%Input image
img = imread ('LenaYCbCrWED.png');
load theta.mat
img = squeeze(img(:,:,1));

figure, imshow(img)
title('WED');

%Show input image
%figure, imshow(img);
%figure, imagesc(theta);

d = double(img);
z = imbinarize(d,70);
figure, imshow(z)
title('WED binarized 70');
```

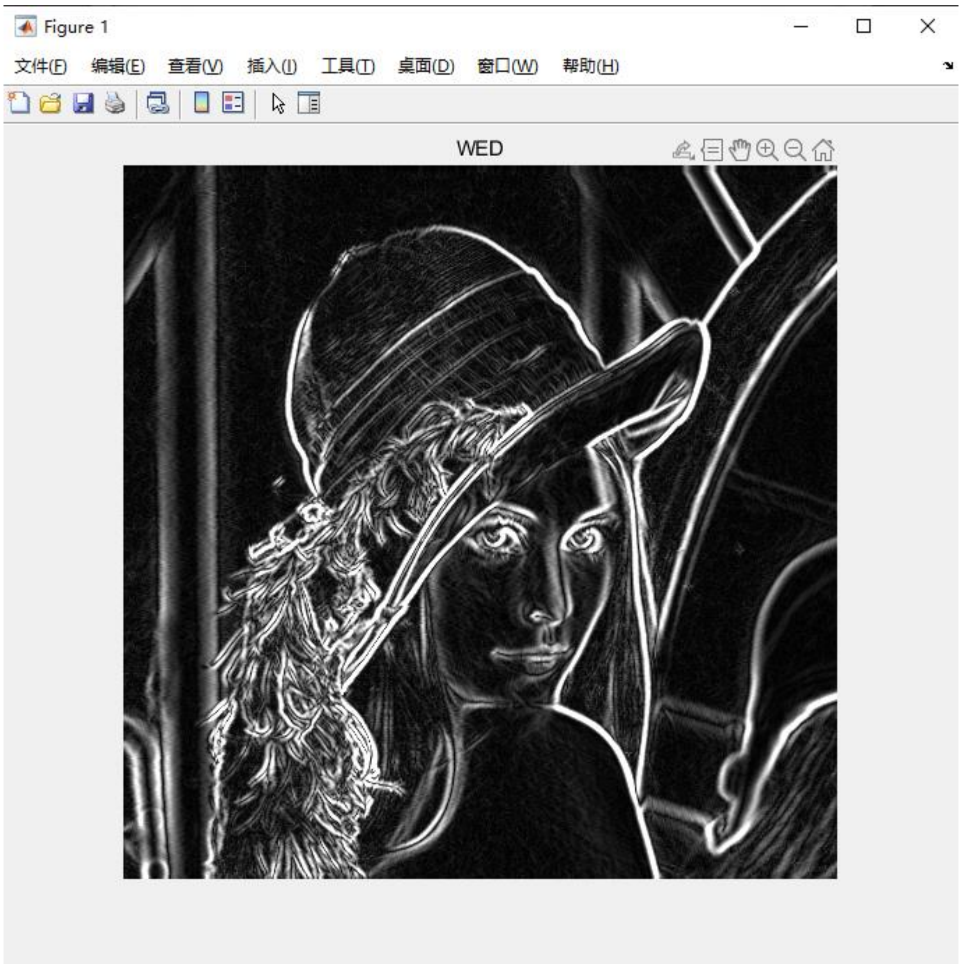


Figure 1
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
WED

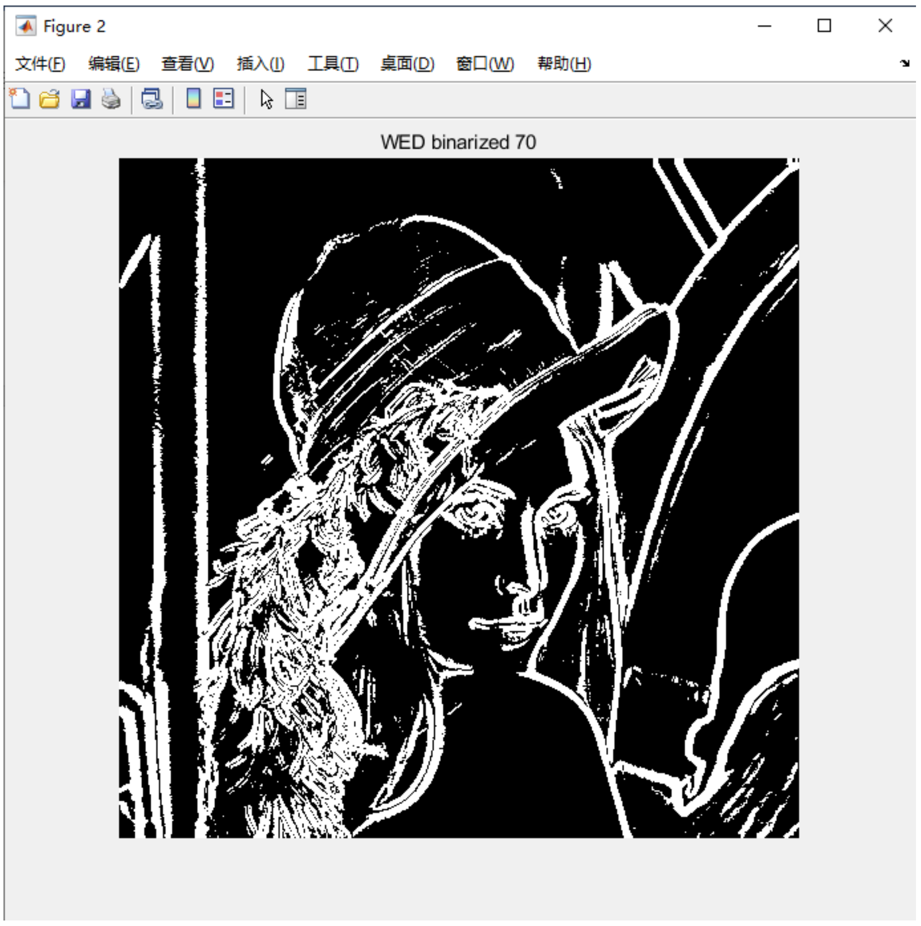


Figure 2
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
WED binarized 70

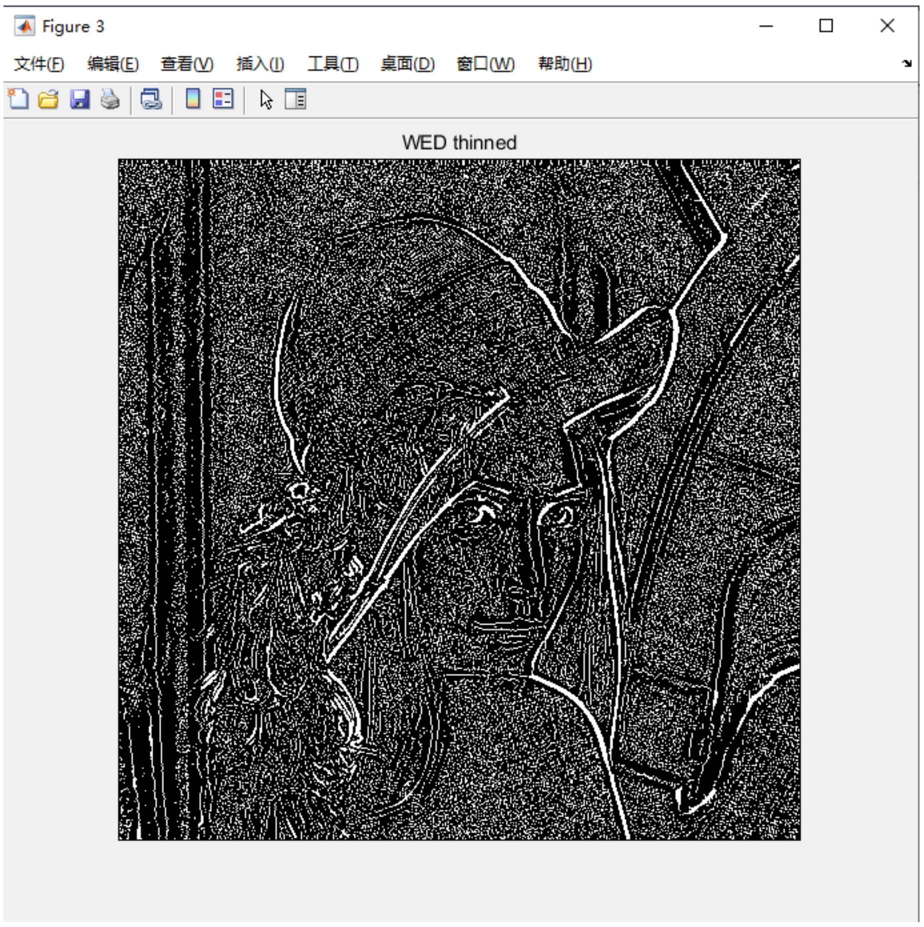


Figure 3
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
WED thinned

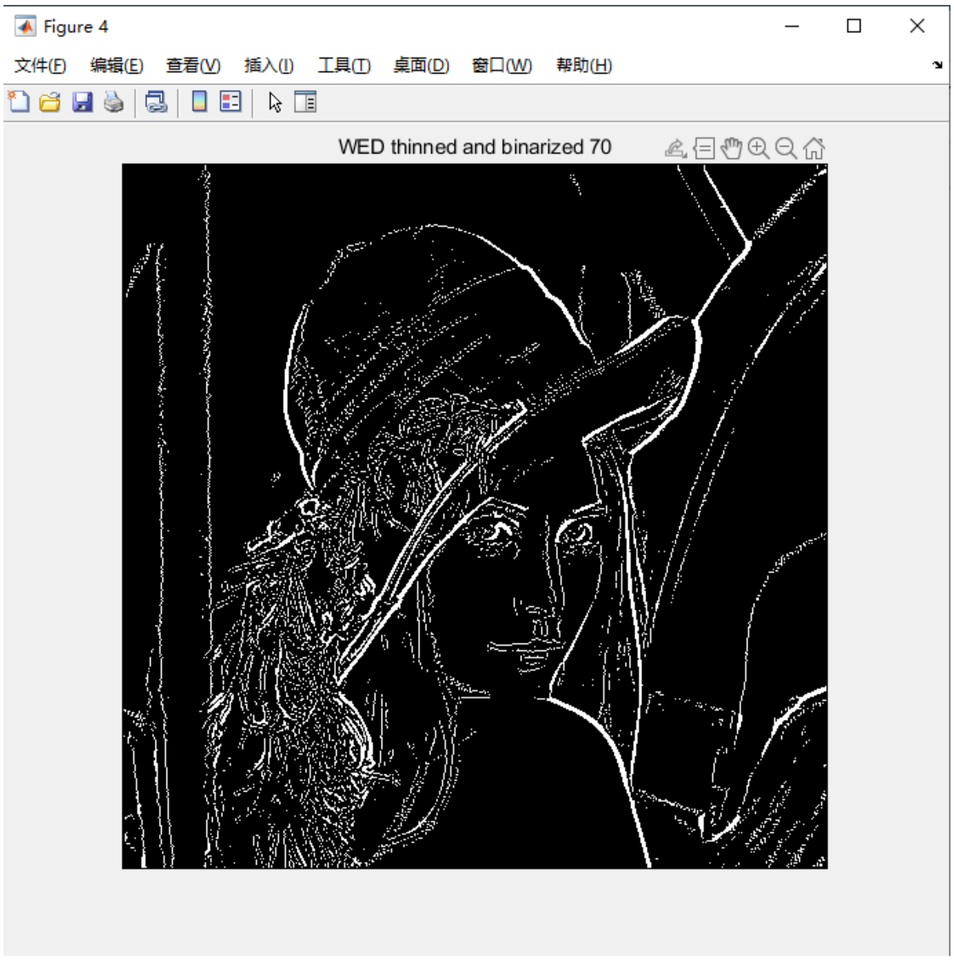


Figure 4
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
WED thinned and binarized 70

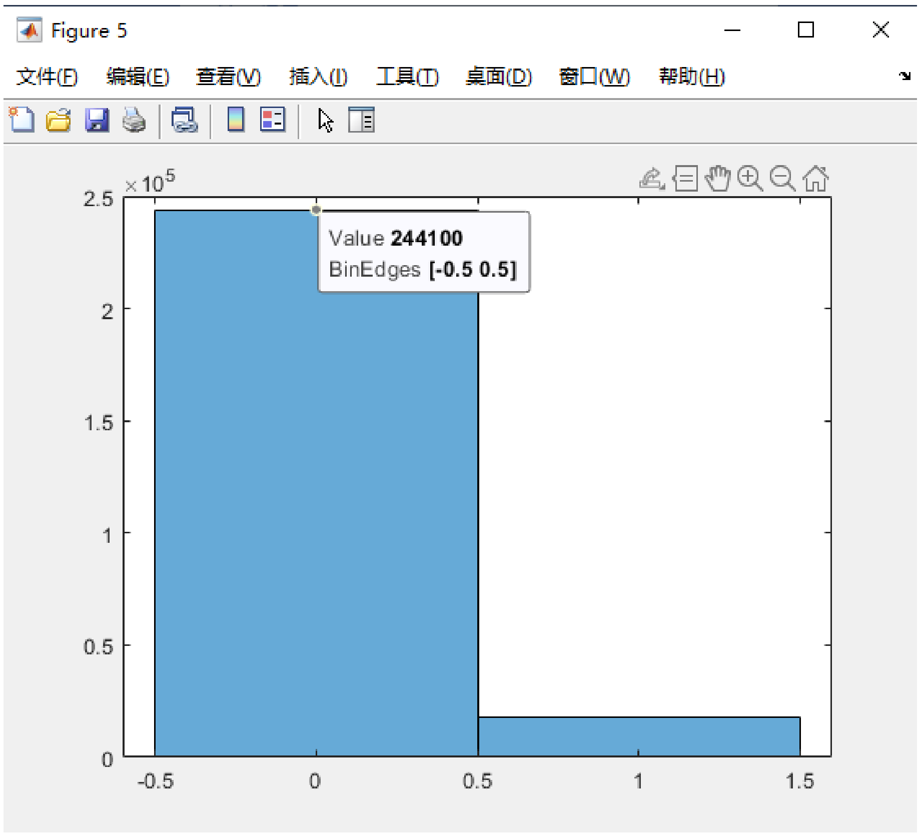


Figure 5
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
×10^5
2.5
2
1.5
1
0.5
0
-0.5
0
0.5
1
1.5
Value 244100
BinEdges [-0.5 0.5]

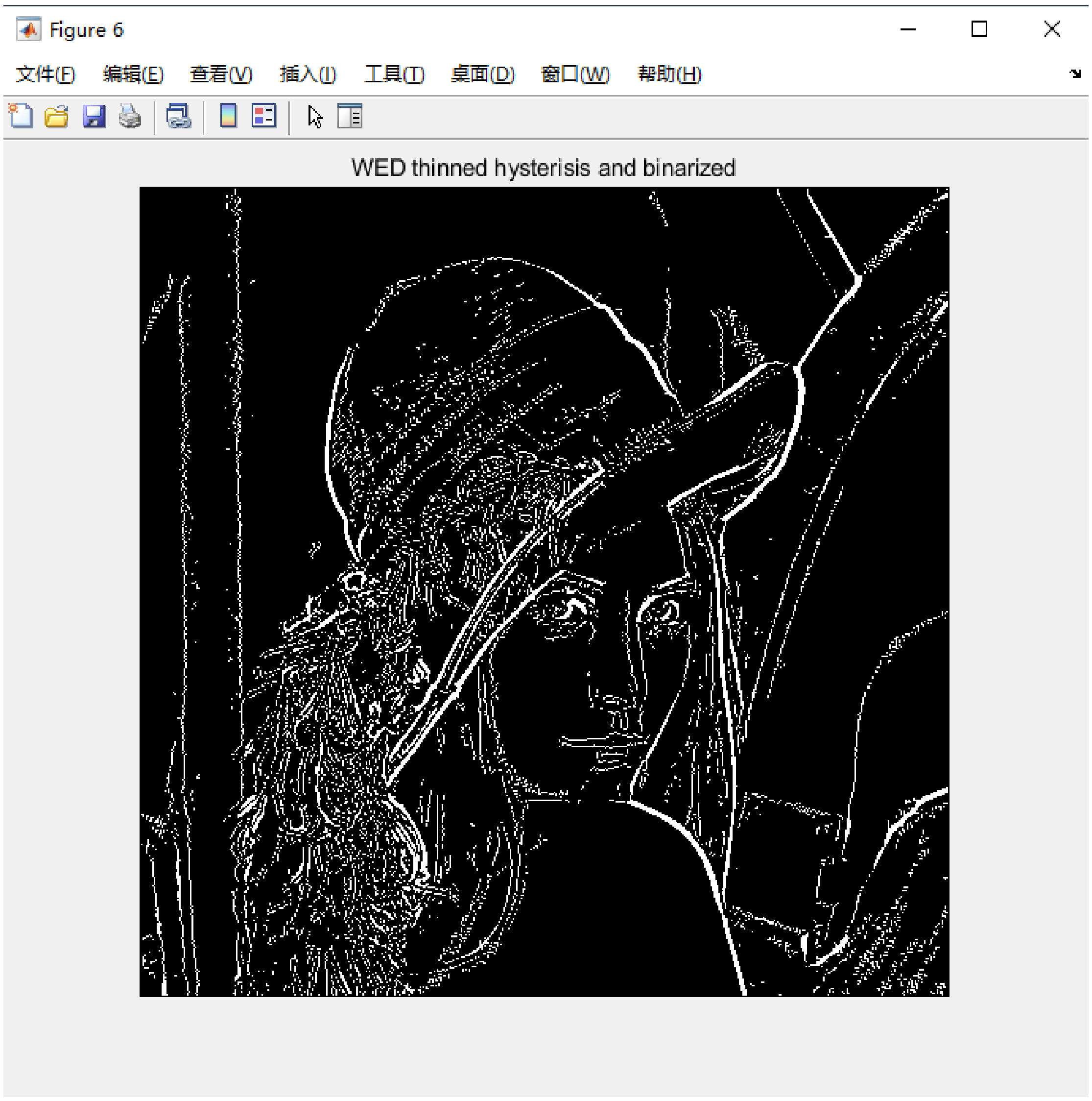
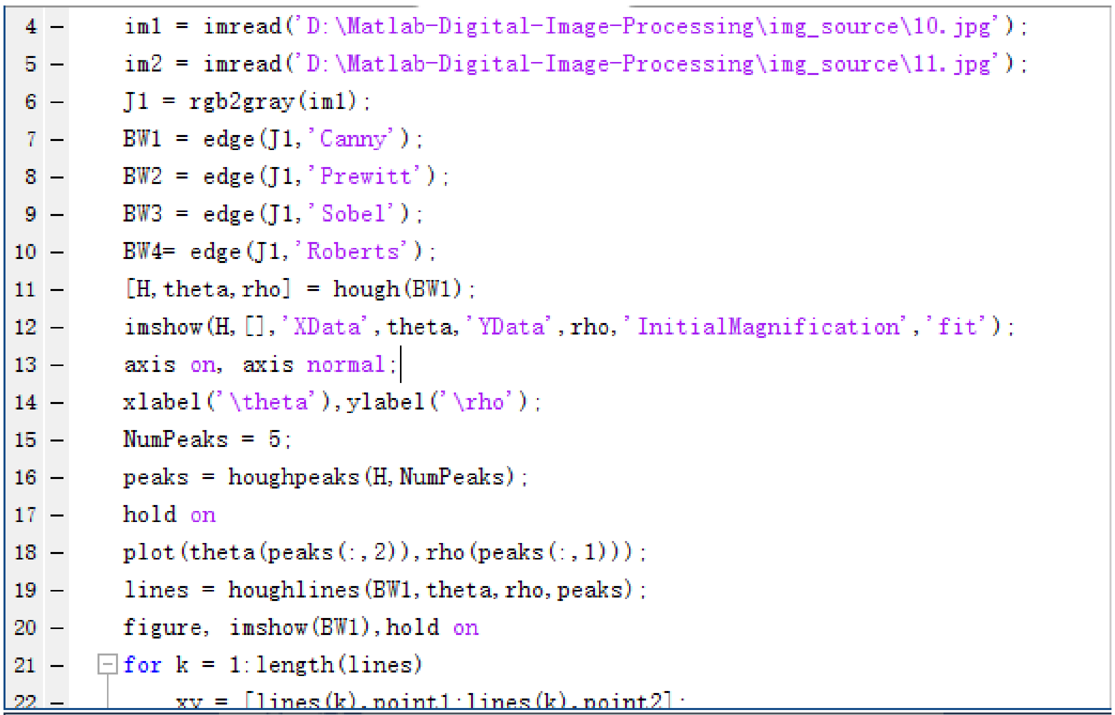


Figure 6
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
WED thinned hysterisis and binarized

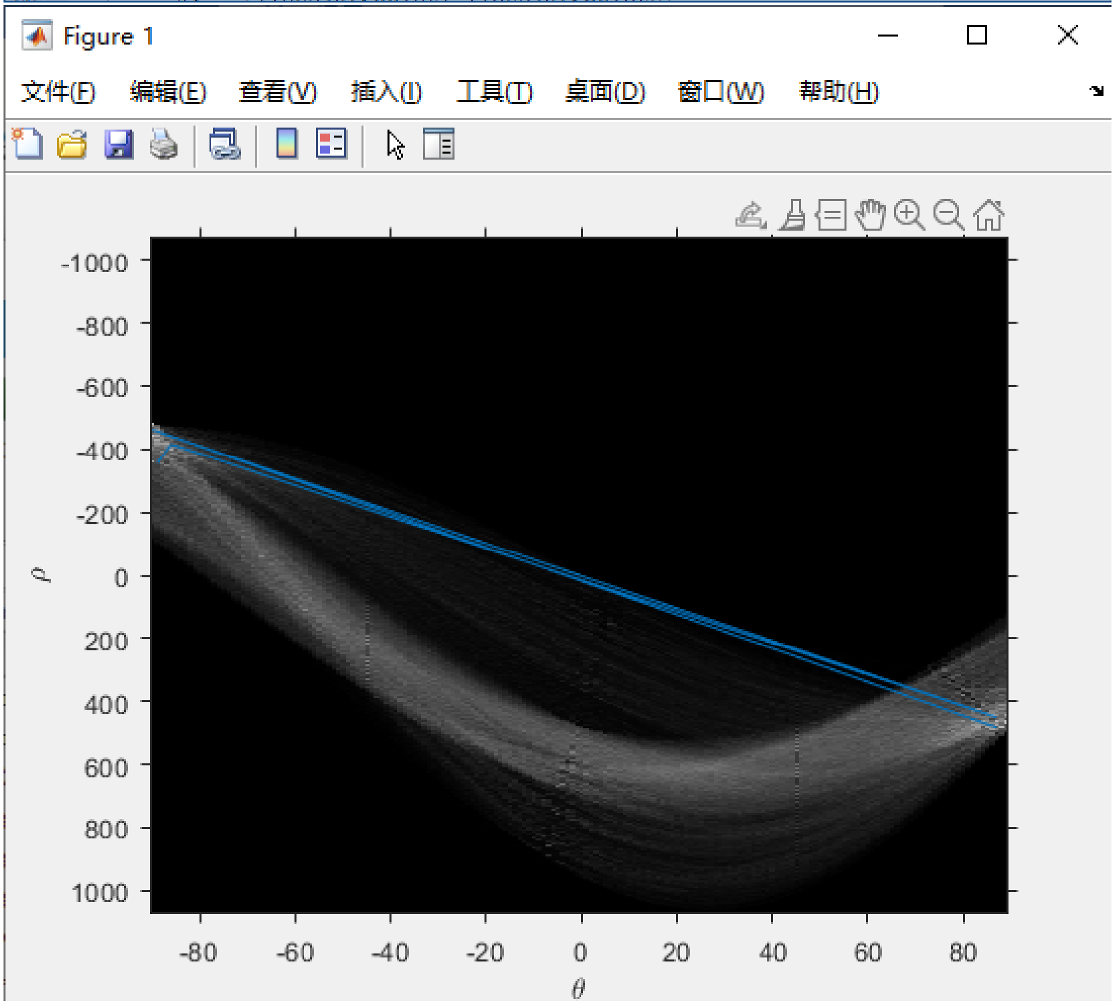
9. Compare the the results of steps 7 and 8 and find out which works better and which one has the more detail and connectivity.

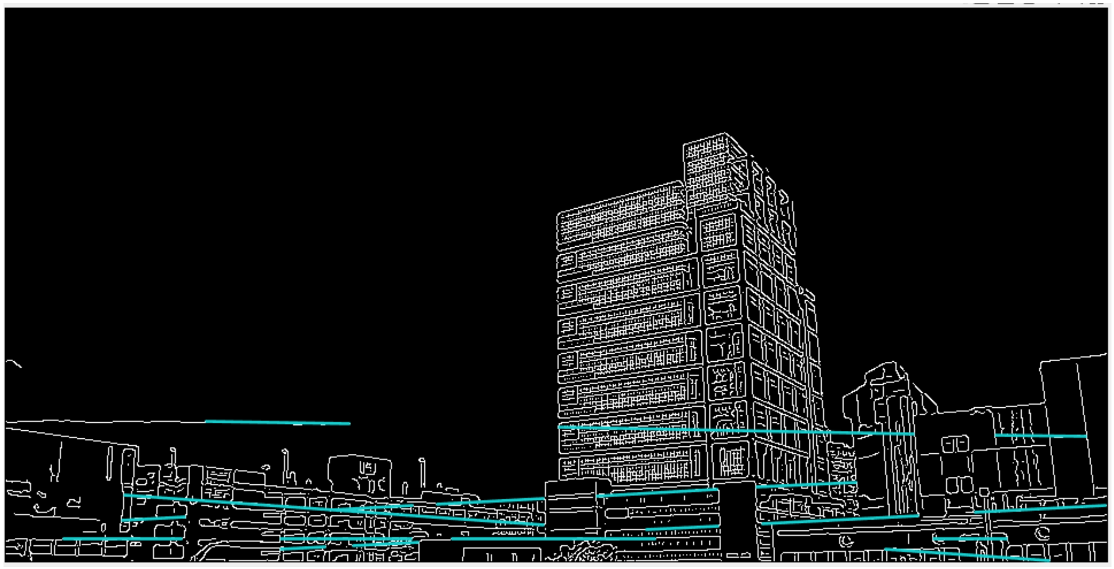
I think the result of step8 is better, because the effect of step7 is very good for the outer frame of the character, but the processing of details is a little bit worse. The result of step8, in my opinion, is better for the recognition of portraits.

10. Use the city scape images segmented in the previous step to perform the Hough Transform using the MATLAB function 'Hough' and see if you can find the lines in the city scape pictures

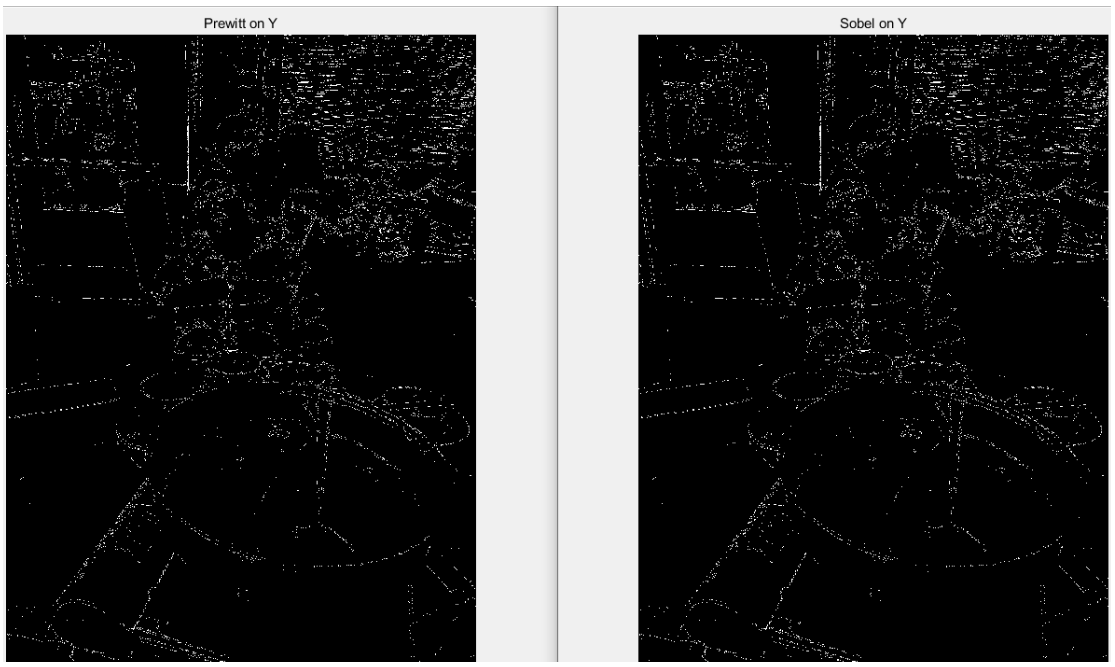


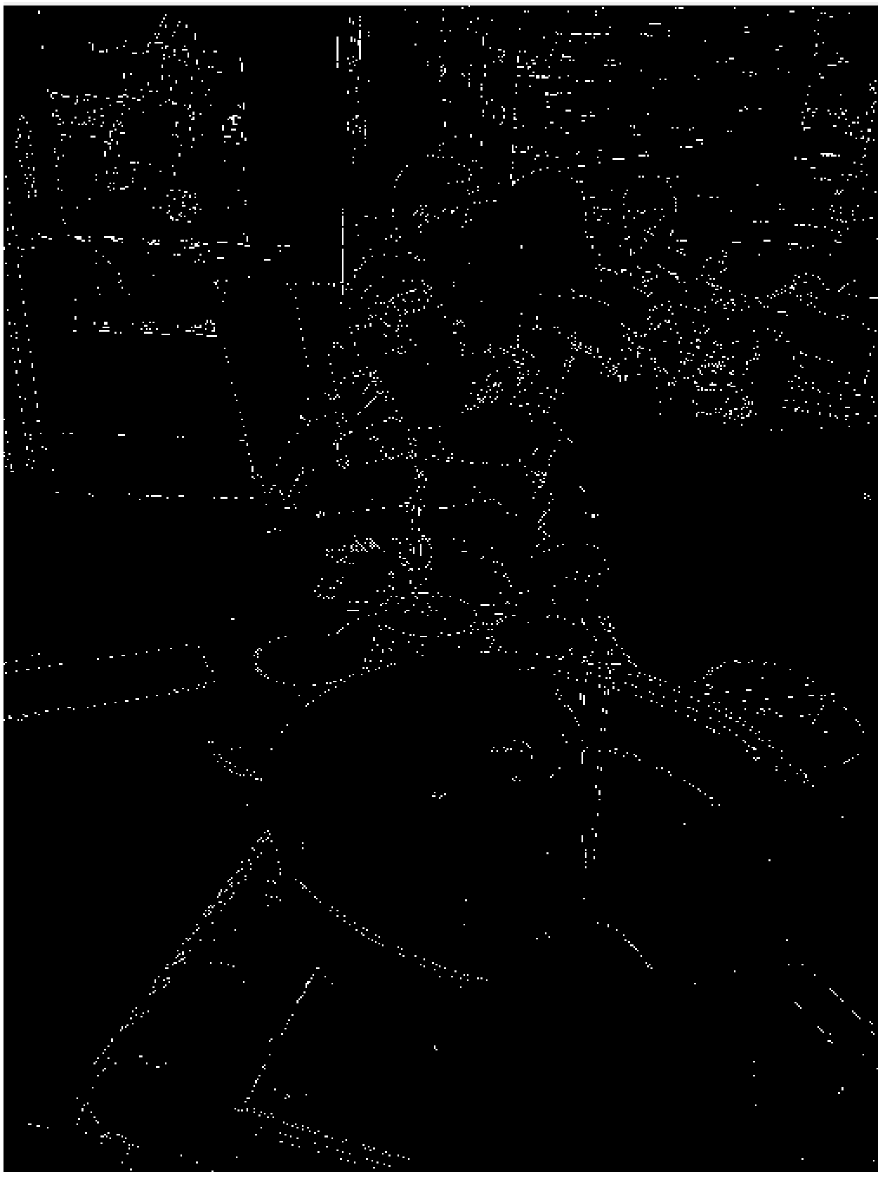
```
    im1 = imread('D:\Matlab-Digital-Image-Processing\img_source\10.jpg');
    im2 = imread('D:\Matlab-Digital-Image-Processing\img_source\11.jpg');
    J1 = rgb2gray(im1);
    BW1 = edge(J1,'Canny');
    BW2 = edge(J1,'Prewitt');
    BW3 = edge(J1,'Sobel');
    BW4= edge(J1,'Roberts');
    [H,theta,rho] = hough(BW1);
    imshow(H,[],'XData',theta,'YData',rho,'InitialMagnification','fit');
    axis on, axis normal;
    xlabel('\theta'),ylabel('\rho');
    NumPeaks = 5;
    peaks = houghpeaks(H,NumPeaks);
    hold on
    plot(theta(peaks(:,2)),rho(peaks(:,1)));
    lines = houghlines(BW1,theta,rho,peaks);
    figure, imshow(BW1),hold on
  for k = 1:length(lines)
        xy = [lines(k).point1;lines(k).point2];
```

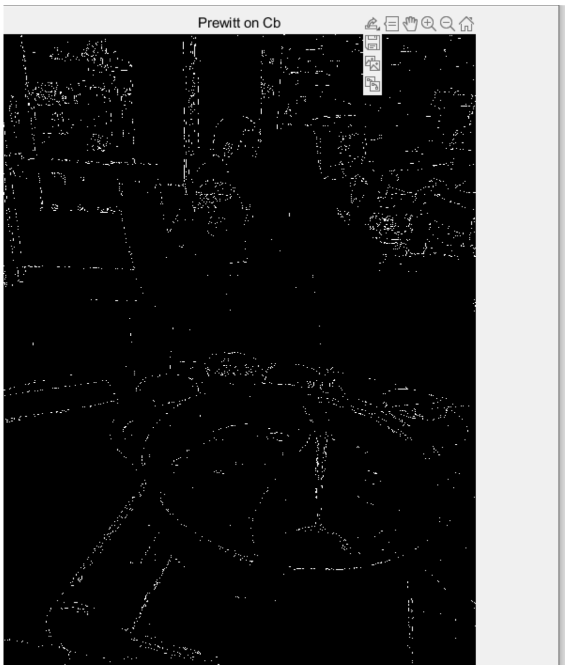


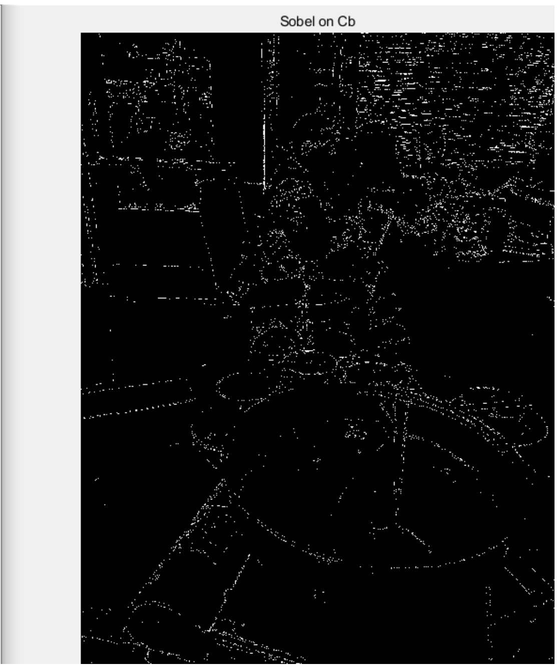


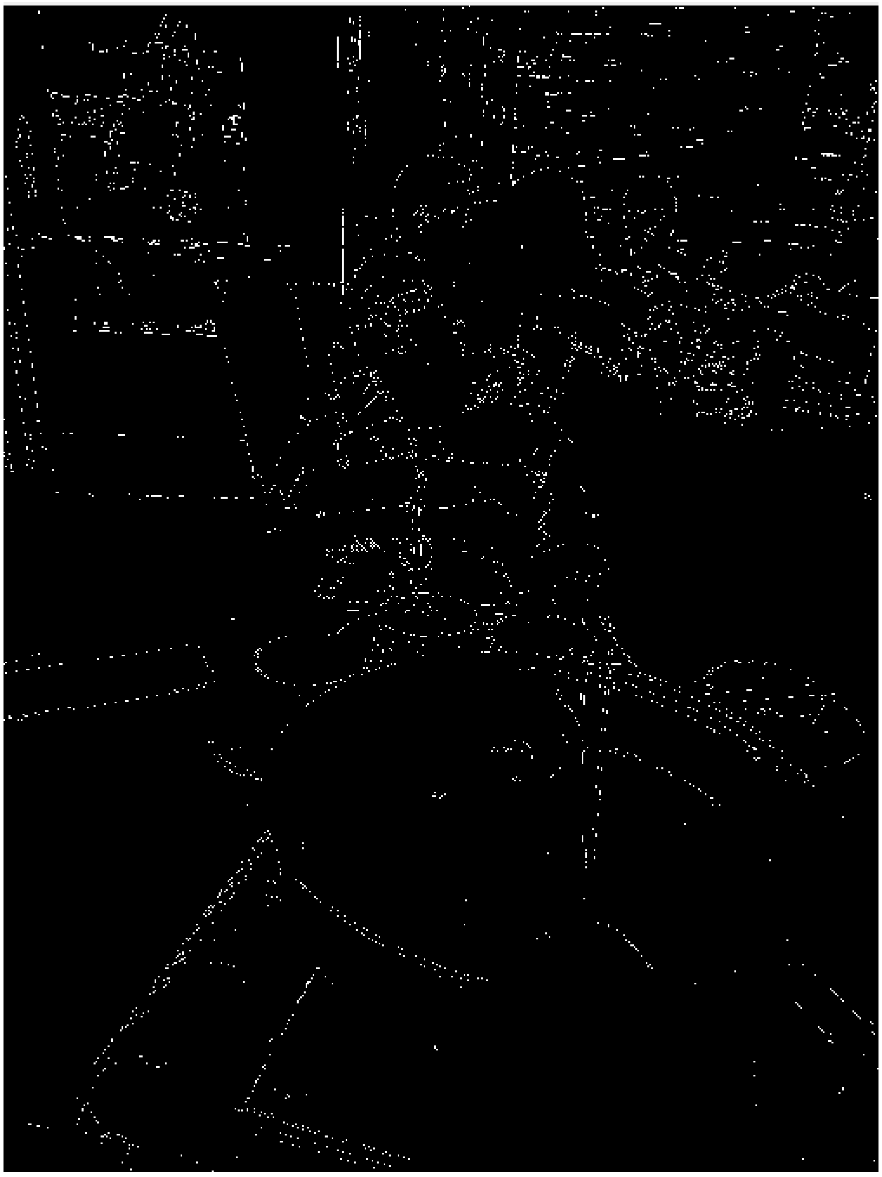
11. Use your smartphone to take a picture of your friends, may be a group picture
    a. Download to our computer and read image into MATLAB
    b. Convert the image into YCbCr
    c. Run Prewitt, Sobel and the provided 'Canny_v0' edge detector on the y, Cr and Cb channels separately. Which one gives you the most definition of the faces in the image?


Prewitt on Y
Sobel on Y




Prewitt on Cb


Sobel on Cb



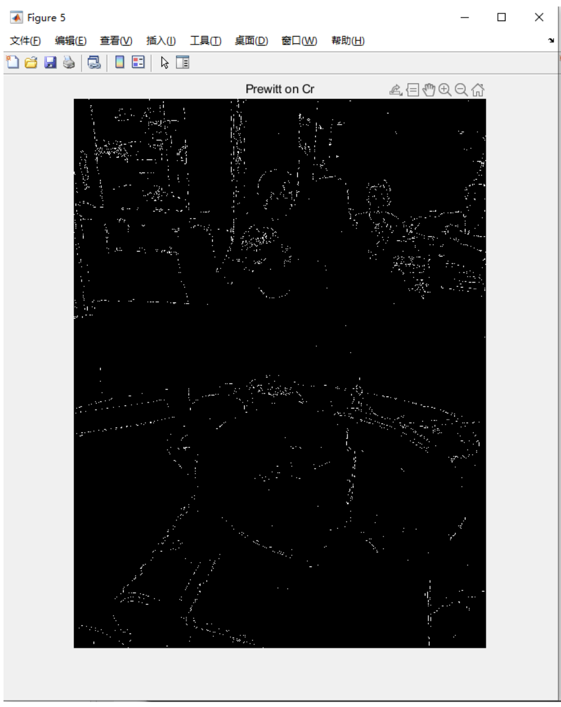


Figure 5
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Prewitt on Cr

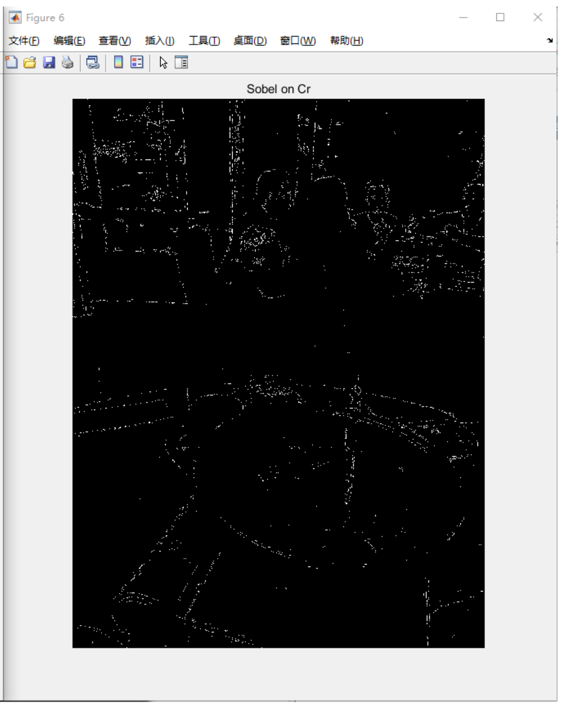
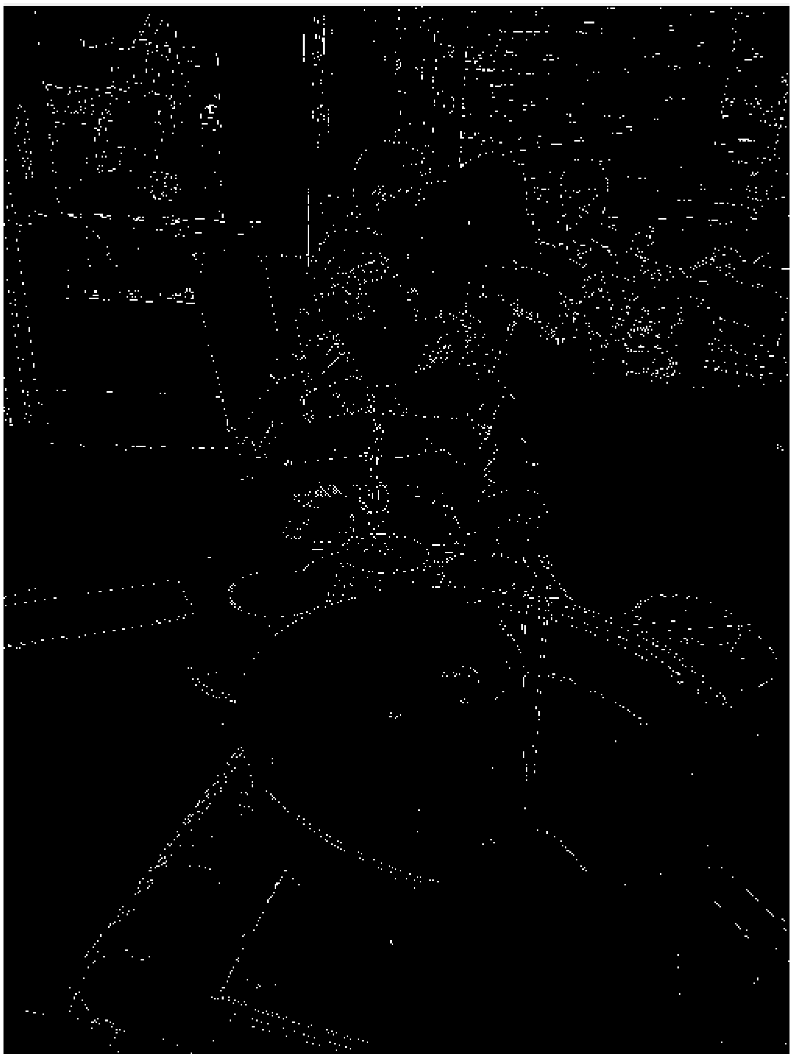



Figure 6
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Sobel on Cr

The 'Canny_v0' edge detector on the y depict the face great

12. Perform a 2D Haar Wavelet transform using the Y channel of 'Lena.png' image using WaveletMain.m
    a. Extend the file so you can see level 3 and level 4 images.
    b. At each wavelet level calculate the absolute magnitude image by squaring the horizontal edge image and adding to the square of the vertical edge image and then taking the square root.
    c. compare the result from part 'b' above to the results in problems 7 and 8

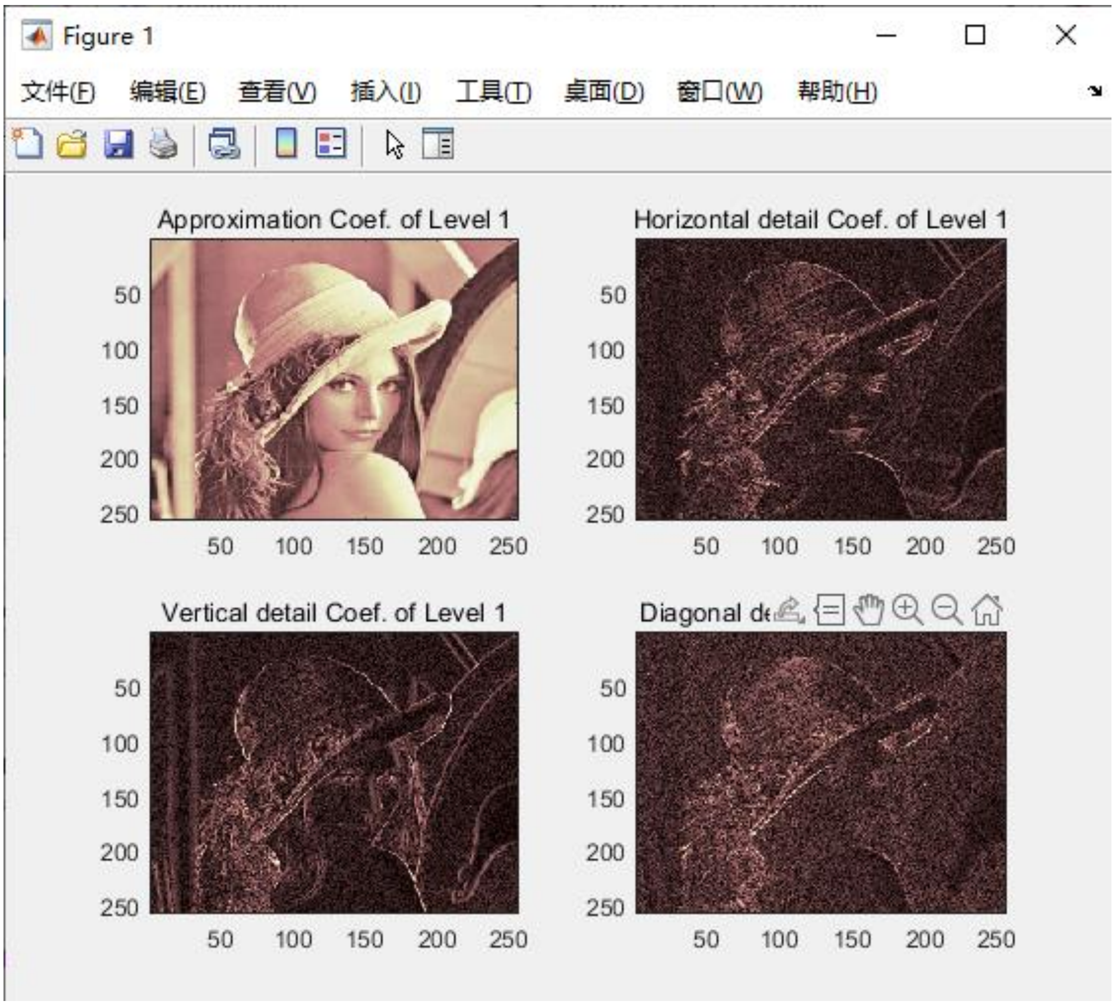


Figure 1
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Approximation Coef. of Level 1
Horizontal detail Coef. of Level 1
Vertical detail Coef. of Level 1
Diagonal de
50
100
150
200
250

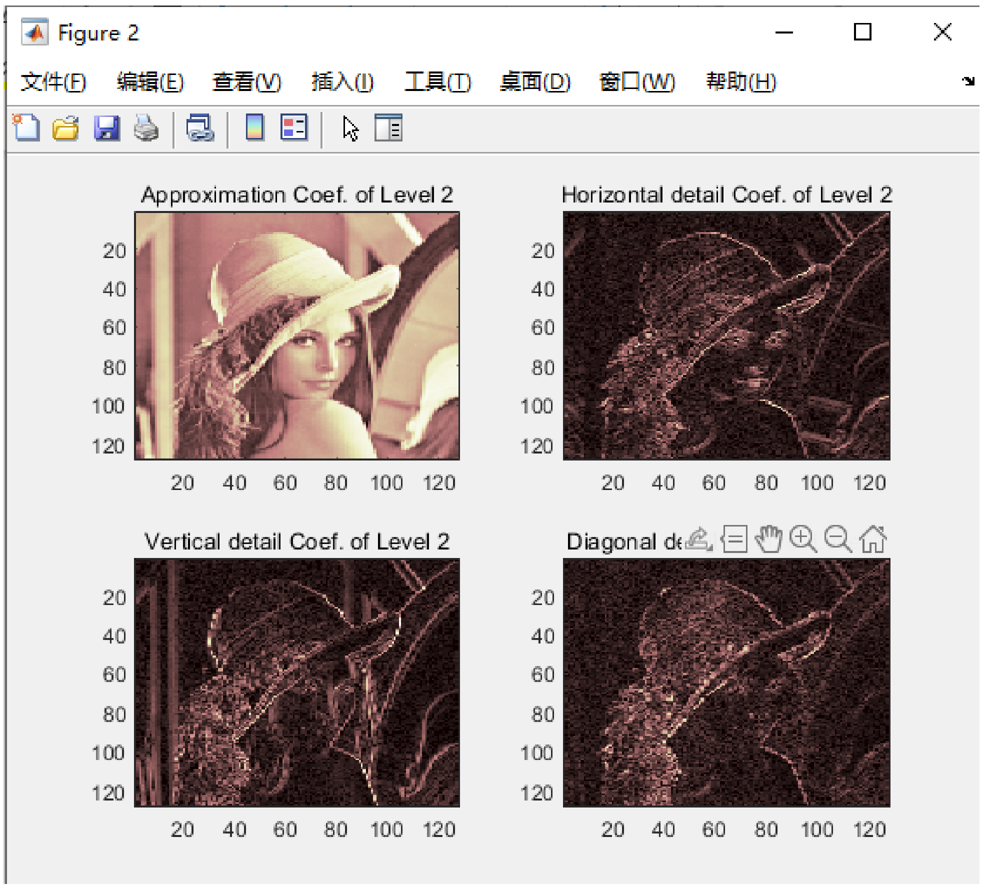


Figure 2
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Approximation Coef. of Level 2
Horizontal detail Coef. of Level 2
Vertical detail Coef. of Level 2
Diagonal de
20 40 60 80 100 120

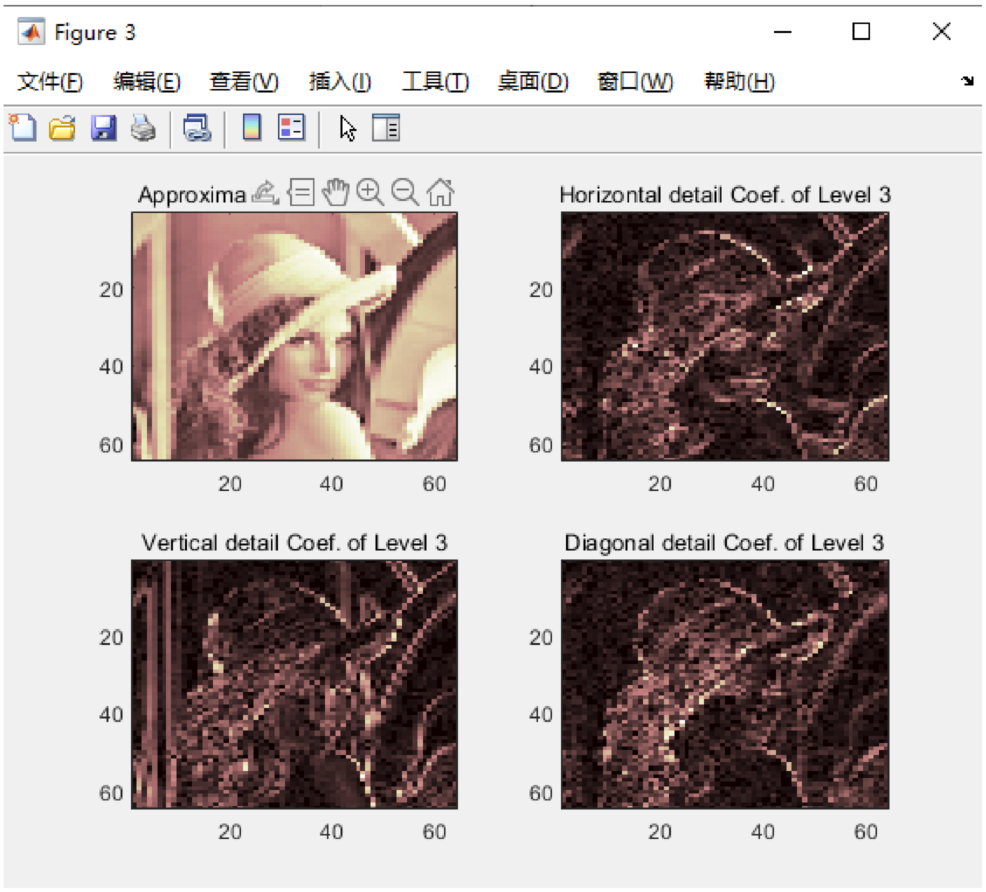


Figure 3
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Approxima
Horizontal detail Coef. of Level 3
Vertical detail Coef. of Level 3
Diagonal detail Coef. of Level 3
20 40 60

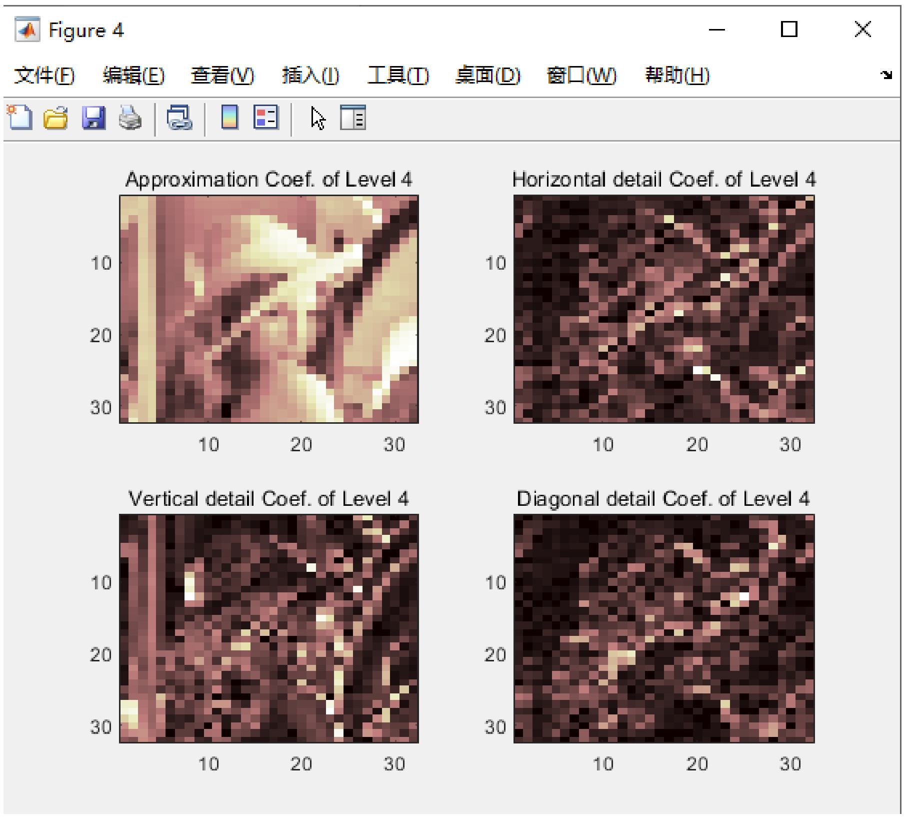


Figure 4
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Approximation Coef. of Level 4
Horizontal detail Coef. of Level 4
Vertical detail Coef. of Level 4
Diagonal detail Coef. of Level 4
10
20
30

```
ami = sqrt(H1.^2 + V1.^2);
```

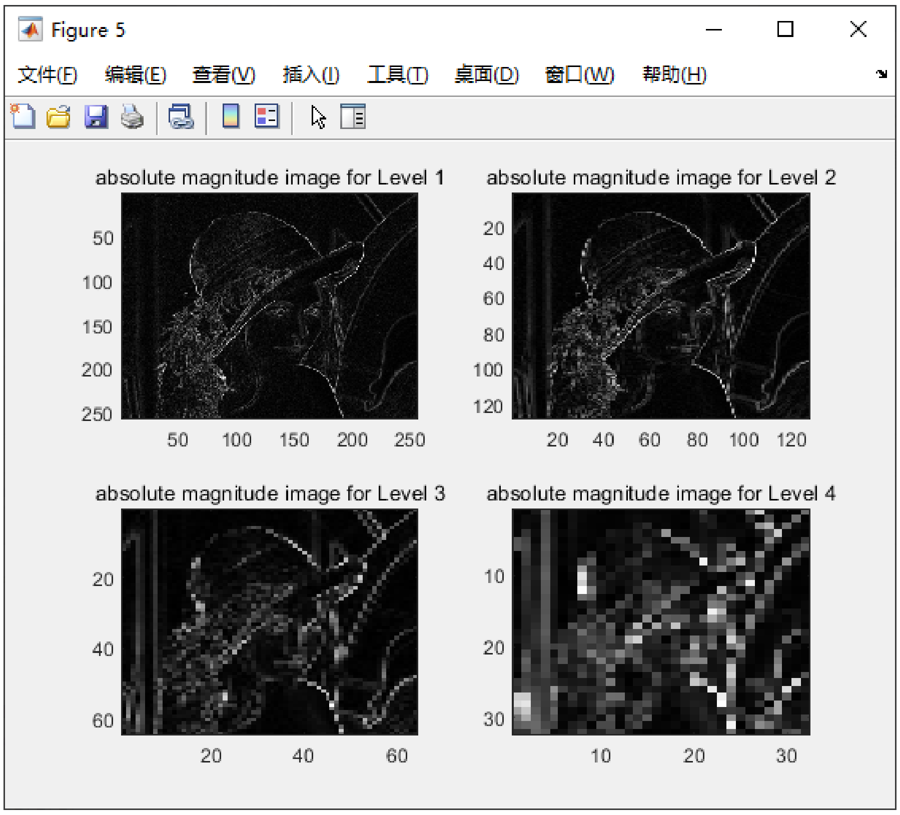


Figure 5
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
absolute magnitude image for Level 1
absolute magnitude image for Level 2
absolute magnitude image for Level 3
absolute magnitude image for Level 4

I think become more clear than in problem 7,8