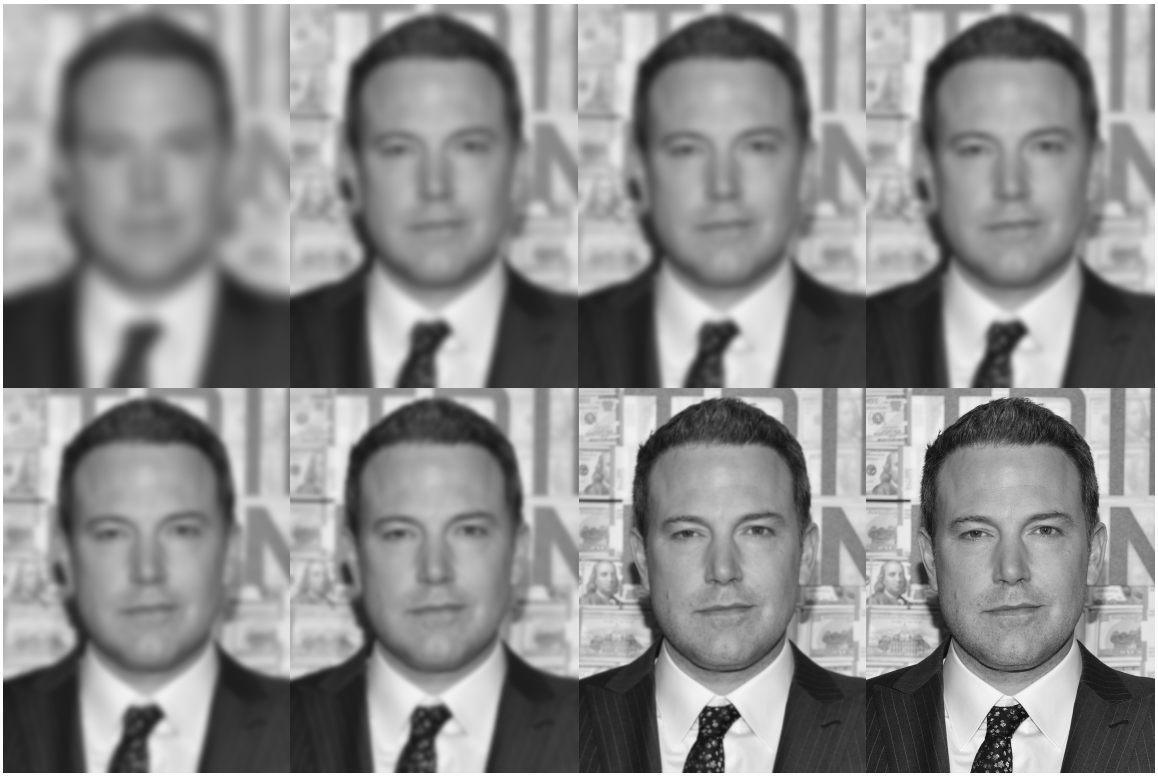
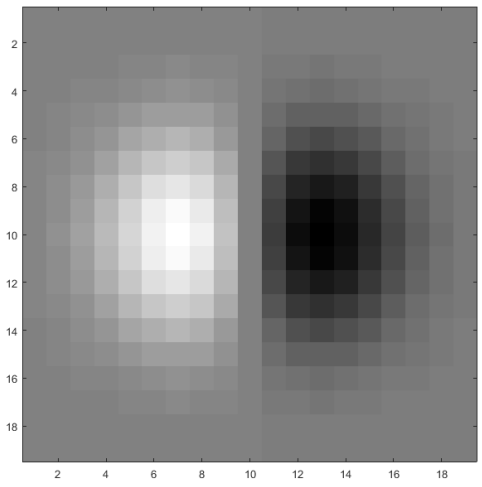
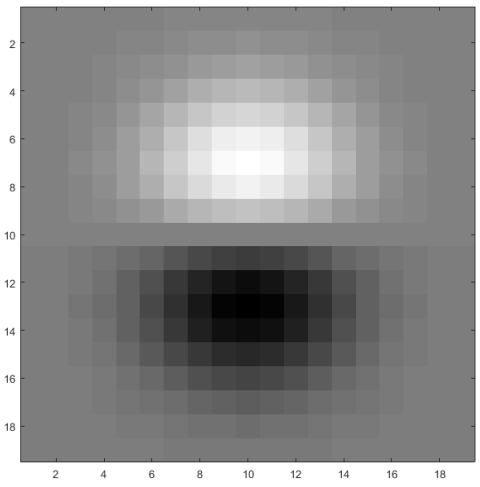
1) Perform Gaussian smoothing on the grayscale image affleck_gray.png (or affleck_gray_flip.png). Try with multiple sigma values, starting with larger values (e.g., from 20 to .5). When does the face become recognizable to your friends? [2 pts]

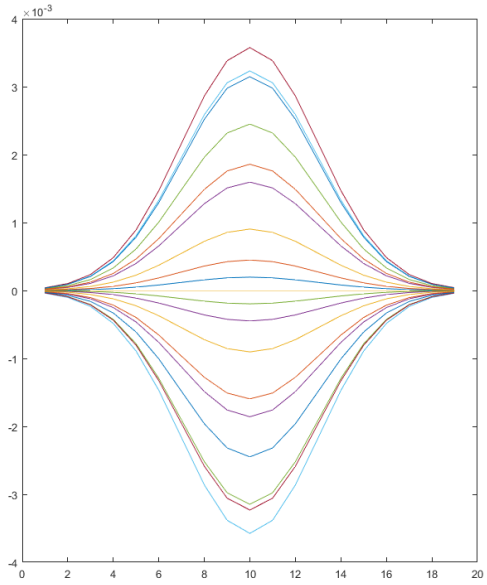


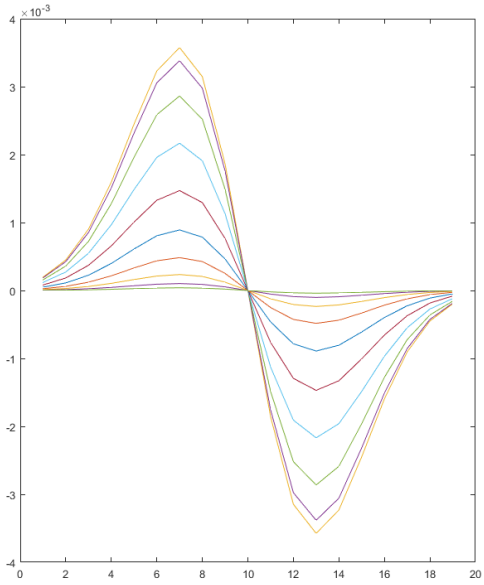
I tried sigma from 20 to 1. The image from the first to the last with sigma = 20, 10, 9, 8, 7, 6, 2, 1. When the sigma value becomes smaller, the image becomes more recognizable. When sigma = 8, the face becomes recognizable.

2) Write a function to compute and display the 2D Gaussian derivative masks Gx and Gy for a given sigma. Sample the Gaussian derivative/ gradient (2D) equation directly (see class notes) at the appropriate x,y (row, col) locations. Note: each mask is a square 2D matrix. Please ensure that the positive derivative lobe is on the side of the increasing direction of each axis for each mask (use a negative sign in your equations if needed). Plot each mask (either in 2D or 3D). [3 pts]









The first image is Gx from imagesc, the second image is Gy from imagesc, the third one is the plot of Gx, and the fourth one is the plot of Gy. I used equation for derivative of Gaussian to make the masks, and the sigma value I set is 3.0. Also, I used both plot and imagesc to display Gx and Gy.

3) Compute and display the gradient magnitude of an image (search the web for an interesting image that has strong vertical and horizontal boundaries; convert to grayscale if necessary; make sure to upload the image with your code in the Carmen submission). [2 pts]

The first image is the original image, the second image is the one with Gx filter, the third one is the one with Gy filter, and the fourth one is the gradient magnitude of an image. I used the spiderman image ,which has strong vertical and horizontal boundaries and convert it into grayscale before I apply filters, and the sigma value is 3.0. From the gradient magnitude of an image, we already can see the edges.

4) Threshold and display the magnitude image with different threshold T levels. [2 pts]

The magnitude images have threshold T > 0.1, 2, 5 and 10. The larger the threshold, the clear the edge. However, when the threshold level is too large, some edges will disappear. At T > 2 levels, the edges are most clear.

5) Compare the above results with the Sobel masks. [2 pts]

The images with the Sobel masks and threshold T > 0.1, 2, 5, 10, 50 and 90. The larger the threshold, the clear the edge. And the images will be noisy if the threshold is too small. When threshold > 50, the edge can become very clear.

6) Run the MATLAB canny edge detector, edge(Im,'canny'), on your image and display the default results. How does it compare? (Python: you can use skimage or opencv packages) [2 pts]

Canny detector generate clear edges!

7) Submit your report containing all code, test images, printouts of images, and discussion of results. Make a HW2.m (or python equivalent) script to do the above tasks and call needed functions. Upload your report, code, and selected images to Carmen. [2 pts]

```
% Bo Yang
% CSE5524– HW2
% 01/23/2022

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Problem 1

%sigma 20
sigma=20; % use different values
G = fspecial('gaussian', 2*ceil(3*sigma)+1, sigma);
faceIm = double(imread('affleck_gray.png'));
```

```
gIm = imfilter(faceIm, G, 'replicate');
imshow(gIm/255); % double images need range of 0-1
imwrite(uint8(gIm), 'gIm_20.bmp');

%sigma 10
sigma=10; % use different values
G = fspecial('gaussian', 2*ceil(3*sigma)+1, sigma);
faceIm = double(imread('affleck_gray.png'));
gIm = imfilter(faceIm, G, 'replicate');
imshow(gIm/255); % double images need range of 0-1
imwrite(uint8(gIm), 'gIm_10.bmp');

%sigma 9
sigma=9; % use different values
G = fspecial('gaussian', 2*ceil(3*sigma)+1, sigma);
faceIm = double(imread('affleck_gray.png'));
gIm = imfilter(faceIm, G, 'replicate');
imshow(gIm/255); % double images need range of 0-1
imwrite(uint8(gIm), 'gIm_9.bmp');

%sigma 8
sigma=8; % use different values
G = fspecial('gaussian', 2*ceil(3*sigma)+1, sigma);
faceIm = double(imread('affleck_gray.png'));
gIm = imfilter(faceIm, G, 'replicate');
imshow(gIm/255); % double images need range of 0-1
imwrite(uint8(gIm), 'gIm_8.bmp');

%sigma 7
sigma=7; % use different values
G = fspecial('gaussian', 2*ceil(3*sigma)+1, sigma);
faceIm = double(imread('affleck_gray.png'));
gIm = imfilter(faceIm, G, 'replicate');
imshow(gIm/255); % double images need range of 0-1
imwrite(uint8(gIm), 'gIm_7.bmp');

%sigma 6
sigma=6; % use different values
G = fspecial('gaussian', 2*ceil(3*sigma)+1, sigma);
faceIm = double(imread('affleck_gray.png'));
gIm = imfilter(faceIm, G, 'replicate');
imshow(gIm/255); % double images need range of 0-1
imwrite(uint8(gIm), 'gIm_6.bmp');

%sigma 2
sigma=2; % use different values
G = fspecial('gaussian', 2*ceil(3*sigma)+1, sigma);
faceIm = double(imread('affleck_gray.png'));
gIm = imfilter(faceIm, G, 'replicate');
imshow(gIm/255); % double images need range of 0-1
imwrite(uint8(gIm), 'gIm_2.bmp');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Problem 2 at end of file
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Problem 3

[Gx, Gy] = gaussDeriv2D(3.0);
myIm = imread('hw2.jpg');
myIm = double(rgb2gray(myIm));
gxIm = imfilter(myIm, Gx, 'replicate');
gyIm = imfilter(myIm, Gy, 'replicate');
magIm = sqrt(gxIm.^2 + gyIm.^2);
imagesc(gxIm);
%pause;
imwrite(gxIm, 'gxIm.bmp');
imagesc(gyIm);
%pause;
imwrite(gyIm, 'gyIm.bmp');
imagesc(magIm);
%pause;
imwrite(magIm, 'magIm.bmp');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Problem 4
tIm = magIm > 0.1;
imagesc(tIm);
%pause;
imwrite(tIm, 'tIm_0.1.bmp');

tIm = magIm > 2;
imagesc(tIm);
%pause;
imwrite(tIm, 'tIm_2.bmp');

tIm = magIm > 5;
imagesc(tIm);
%pause;
imwrite(tIm, 'tIm_5.bmp');

tIm = magIm > 10;
imagesc(tIm);
%pause;
imwrite(tIm, 'tIm_10.bmp');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Problem 5
Im = imread('hw2.jpg');
Im = double(rgb2gray(Im));
Fx = -fspecial('sobel')';
fxIm = imfilter(Im,Fx);
Fy = -fspecial('sobel');
fyIm = imfilter(Im,Fy);
magIm = sqrt(fxIm.^2 + fyIm.^2);

tIm = magIm > 0.1;
imagesc(tIm);
%pause;
imwrite(tIm, 'tIm_sobel_0.1.bmp');
```

```
tIm = magIm > 2;
imagesc(tIm);
%pause;
imwrite(tIm, 'tIm_sobel_2.bmp');

tIm = magIm > 5;
imagesc(tIm);
%pause;
imwrite(tIm, 'tIm_sobel_5.bmp');

tIm = magIm > 10;
imagesc(tIm);
%pause;
imwrite(tIm, 'tIm_sobel_10.bmp');

tIm = magIm > 50;
imagesc(tIm);
%pause;
imwrite(tIm, 'tIm_sobel_50.bmp');

tIm = magIm > 90;
imagesc(tIm);
%pause;
imwrite(tIm, 'tIm_sobel_90.bmp');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Problem 6

Im = imread('hw2.jpg');
grayIm = rgb2gray(Im);
edges = edge(grayIm,'canny');
imagesc(edges);
%pause;
imwrite(edges, 'edges.bmp');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Problem 2
function [Gx, Gy] = gaussDeriv2D(sigma)


 mask_size = 2*ceil(3*sigma)+1;
 Gx = zeros(mask_size, mask_size);
 Gy = zeros(mask_size, mask_size);
 y = -ceil(3*sigma);
for i = 1: mask_size
     x = -ceil(3*sigma);
    for j = 1:mask_size
        Gx(i,j) = (exp((-1)*(x^2 + y^2)/ (2*sigma^2))) * (((-1) *x)/(2*
pi*sigma^4));
       Gy(i,j) = (exp((-1)*(x^2 + y^2)/ (2*sigma^2))) * (((-1) *y)/(2*
pi*sigma^4));
       x = x + 1;
    end
    y = y + 1;
```

```
 end
imagesc(Gx);
axis('image');
colormap('gray');
imwrite(uint8(Gx), 'Gx.bmp');
%pause;
plot(Gx)
%pause;
plot(Gy)
%pause;
imagesc(Gy);
axis('image');
colormap('gray');
imwrite(uint8(Gy), 'Gy.bmp');
%pause;


end
```