Yukun Duan
Computer Vision for HCI AU'22
Homework Assignment #2

1)
I tried ten different sigma values for this problem, the outcome is below with sigma value changes from 40 to 0.5. Clearly, as the sigma value decrease, the clarity of the picture will be higher, and noise will decrease accordingly:

σ= 40:

σ= 20:

σ=15:

σ=10:

σ= 5:

σ= 4:

σ= 3:

σ= 2:

σ= 1:(That’s the original)

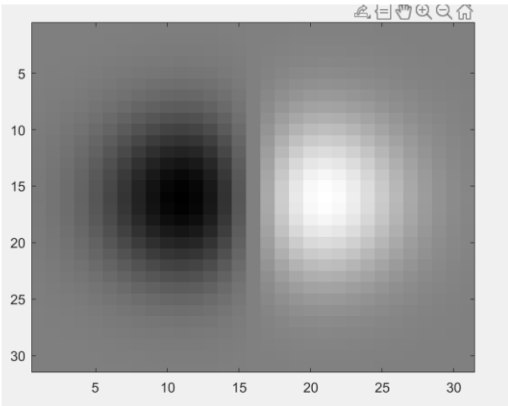
σ= 0.5:

Talking with my friend, he told me that when **σ=5** he can recognize human's face.

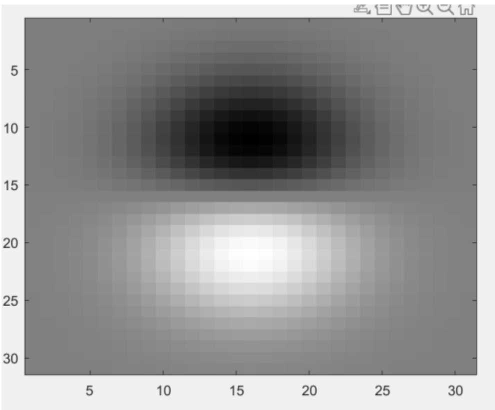
2)

I design the gaussDeriv2D(sigma) function which can generate 2D Gaussian derivative mask Gx and Gy when given a sigma value. I tested the function with sigma value equals to 5, and here is my Gx and Gy mask in 2D:

Gx:



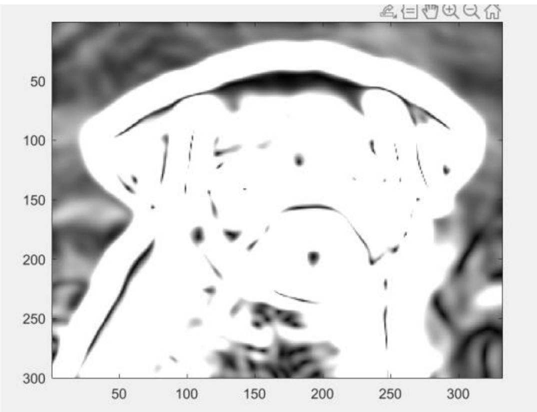
Gy:



We can find that the positive derivative lobe is on the side of the increasing direction of each axis for each mask.

3)

I directly use a grayscale puppy image. Just like the previous question, I take sigma as 5, and here is the image gradient magnitude:



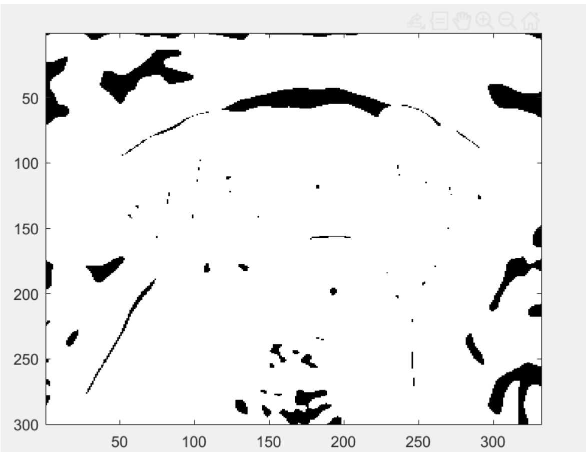
And compare with the original picture:

We can find that the gradient magnitude only keeps the basic outline of the puppy, other details are missing. Through other tests, we are still able to find that the smaller sigma tends to contain more details of edges, while the larger sigma will deprecate some edges and only remain the essential ones.

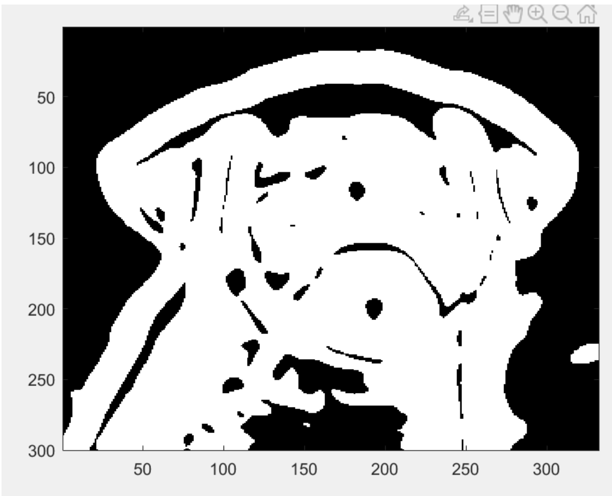
4)

I set T value at 0.3, 1, 3, 10 and 20, here are gradient magnitude images I get:

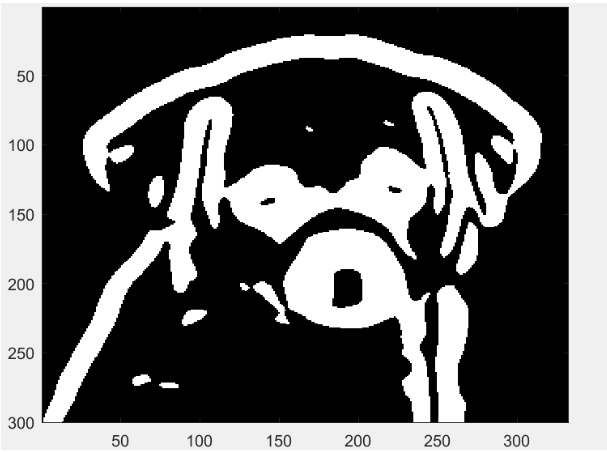
T=0.3:



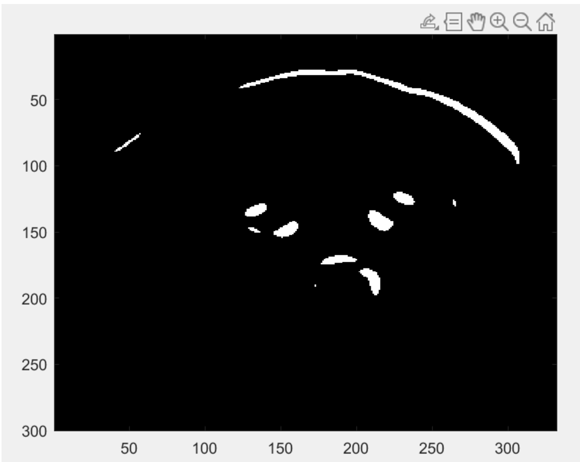
T=1:


50
100
150
200
250
300
50
100
150
200
250
300

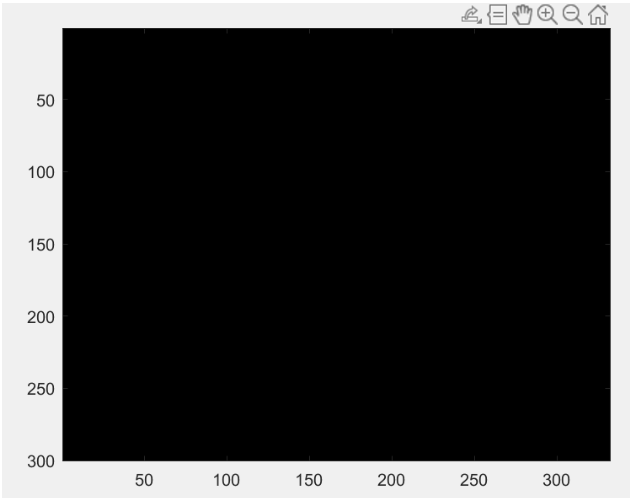
T=3:


50
100
150
200
250
300
50
100
150
200
250
300

T=10:

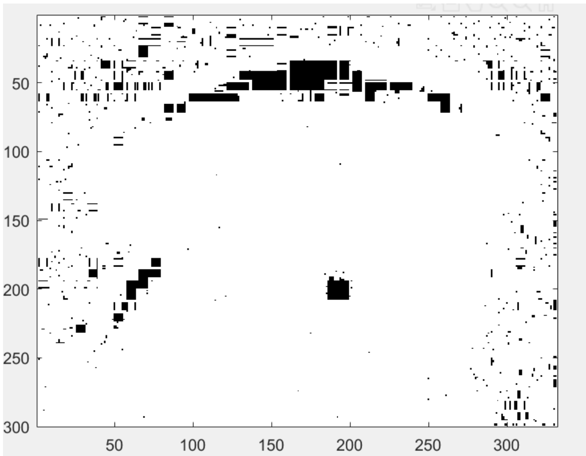

50
100
150
200
250
300
50
100
150
200
250
300

T=20:

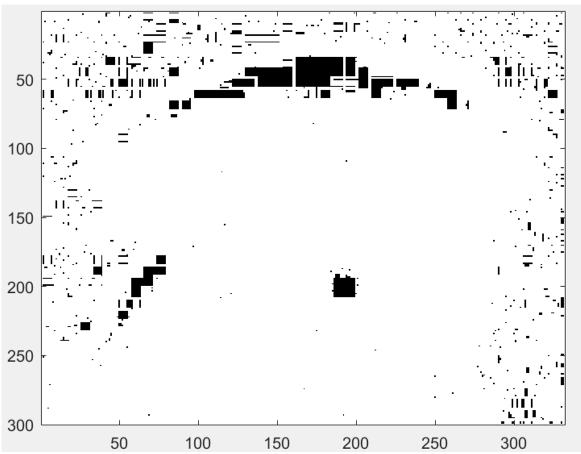


As T becomes larger, more details (edges) are removed from picture and picture will also become more abstract and unrecognizable. If T goes very large, like 20 used in my example, all details of the picture will disappear.
While when T goes very small, then too many edges show up. I personally think that when T=3, the edges are captured as expected and the contours are representative enough.

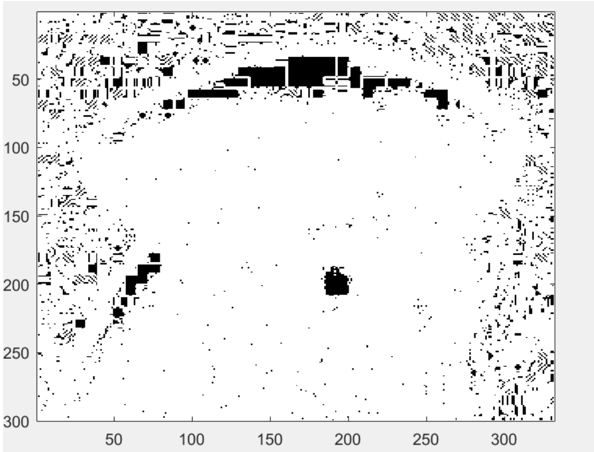
5)
T=0.3:



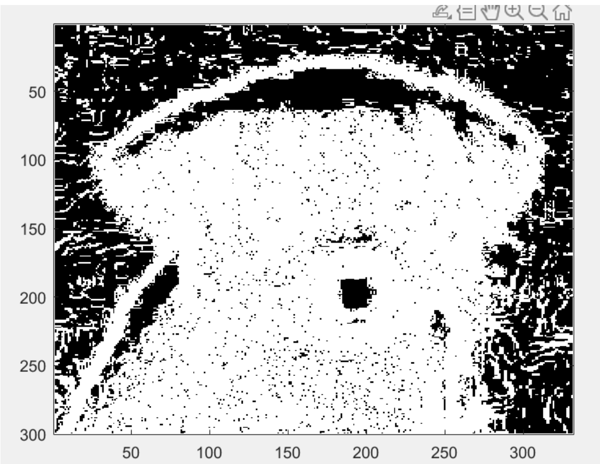
T=1:



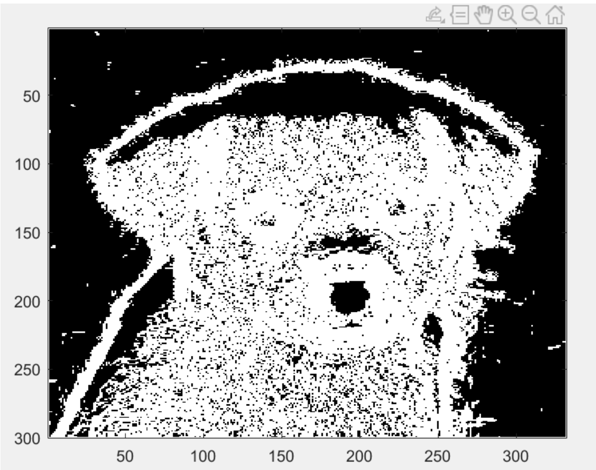
T=3:



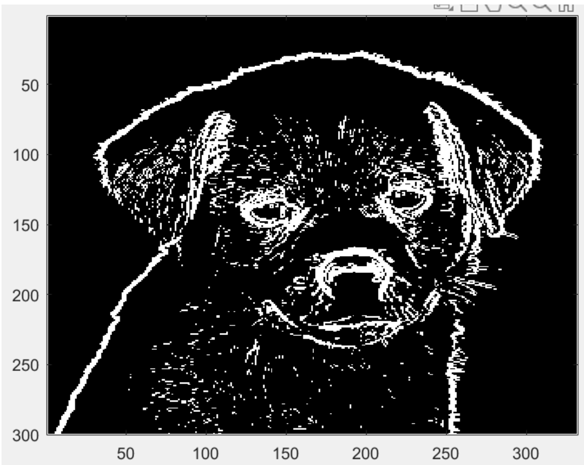
T=10:



T=20:



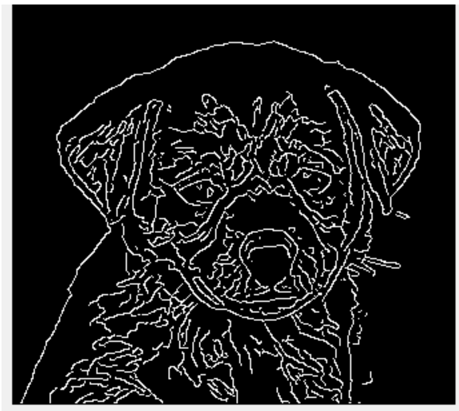
T=100:



The Sobel masks contain too many edges compared to the previous situation and make the picture blurry. I increased the T value to 100 and then the edges become clear, and small unnecessary edges are eliminated.

6)

We used canny to find edges in this question, and we can notice that the canny method did pretty well on edge detection, this image keeps most of the puppy's facial contours. And also, the canny technique can produce smoother edges compared to the Sobel filter.



Attached Code:

```
% Yukun Duan
% CSE5524 - HW2
% 8/31/2022

%% Problem 1
sigma=40.0;  % use largest value
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.bmp');
axis("image");
pause;

sigma=20.0;  % second try
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), 'gIm1.bmp');
axis("image");
pause;

sigma=15.0; % third try
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), 'gIm2.bmp');
axis("image");
pause;

sigma=10.0;   %fourth try
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), 'gIm3.bmp');
axis("image");
pause;

sigma=5.0;  % fifth try
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), 'gIm4.bmp');
axis("image");
pause;

sigma=4.0;  % sixth try
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), 'gIm5.bmp');
axis("image");
pause;

sigma=3.0;  % seventh try
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), 'gIm6.bmp');
axis("image");
pause;

sigma=2.0;   % eighth try
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), 'gIm7.bmp');
axis("image");
pause;

sigma=1.0;   % ninth try
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), 'gIm8.bmp');
axis("image");
pause;

sigma=0.5; % tenth try
```

```
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), 'gIm9.bmp');
axis("image");
pause;

%% Problem 2
[a,b] = gaussDeriv2D(5);  % let sigma value be 5
img_gx = imagesc(a);  % Display image with scaled colors
colormap('gray');
pause;

img_gy = imagesc(b);
colormap('gray');
pause;

%% Problem 3
[Gx,Gy] = gaussDeriv2D(5);  %let sigma value be 5
myIm = double(imread('pic.jpg'));
myim=imread('pic.jpg');
gxIm = imfilter(myIm, Gx, 'replicate');
gyIm = imfilter(myIm, Gy, 'replicate');
magIm = sqrt(gxIm.^2 + gyIm.^2);
imagesc(gxIm);
imagesc(gyIm);
imagesc(magIm);
pause;


%% Problem 4
T = 0.3;  % try T for three different value
tIm = magIm > T;
imagesc(tIm);
pause;

T = 1;
tIm = magIm > T;
imagesc(tIm);
pause;

T = 3;
tIm = magIm > T;
```

```
imagesc(tIm);
pause;

T = 10;
tIm = magIm > T;
imagesc(tIm);
pause;

T = 20;
tIm = magIm > T;
imagesc(tIm);
pause;

%% Problem 5
T = 0.3;
Fx = -fspecial('sobel')';
fxIm = imfilter(myIm,Fx);
Fy = -fspecial('sobel');
fyIm = imfilter(myIm,Fy);
magIm = sqrt(fxIm.^2 + fyIm.^2);
tIm = magIm > T;
imagesc(tIm);
pause;

T = 1;
Fx = -fspecial('sobel')';
fxIm = imfilter(myIm,Fx);
Fy = -fspecial('sobel');
fyIm = imfilter(myIm,Fy);
magIm = sqrt(fxIm.^2 + fyIm.^2);
tIm = magIm > T;
imagesc(tIm);
pause;

T = 3;
Fx = -fspecial('sobel')';
fxIm = imfilter(myIm,Fx);
Fy = -fspecial('sobel');
fyIm = imfilter(myIm,Fy);
magIm = sqrt(fxIm.^2 + fyIm.^2);
tIm = magIm > T;
imagesc(tIm);
pause;
```

```
T = 10;
Fx = -fspecial('sobel')';
fxIm = imfilter(myIm,Fx);
Fy = -fspecial('sobel');
fyIm = imfilter(myIm,Fy);
magIm = sqrt(fxIm.^2 + fyIm.^2);
tIm = magIm > T;
imagesc(tIm);
pause;

T = 20;
Fx = -fspecial('sobel')';
fxIm = imfilter(myIm,Fx);
Fy = -fspecial('sobel');
fyIm = imfilter(myIm,Fy);
magIm = sqrt(fxIm.^2 + fyIm.^2);
tIm = magIm > T;
imagesc(tIm);
pause;

T = 100;
Fx = -fspecial('sobel')';
fxIm = imfilter(myIm,Fx);
Fy = -fspecial('sobel');
fyIm = imfilter(myIm,Fy);
magIm = sqrt(fxIm.^2 + fyIm.^2);
tIm = magIm > T;
imagesc(tIm);
pause;

%% Problem 6
Im = rgb2gray(myim);
edge(Im,'canny');

%% Problem 2 Function
function [Gx, Gy] = gaussDeriv2D(sigma)
    MaskSize = 2 * ceil(sigma * 3) + 1;
    for r = 1:MaskSize
        for c = 1:MaskSize
            x = c-ceil(3*sigma)-1;
            y = r-ceil(3*sigma)-1;
            Gx(r,c) = x * exp(-1 * (x^2 + y^2)/(2 * sigma^2)) / (2 * pi *
sigma^4);
```

```
            Gy(r,c) = y * exp(-1 * (x^2 + y^2)/(2 * sigma^2)) / (2 * pi *
sigma^4);
        end
    end
end
```