Pouyan Firouzabadi
NetID: Spourm2

Machine Problem#2

1.

The approach used for this problem was quite different than Machine Problem #1. There were flaws and incompatibilities regarding Convolution method and the Gaussian kernel. The convolution was clipping all the data between 0-255 however while doing gradient we can have negative results. The gaussian kernels were dimming the pictures as I increased the sigma and kernel size. The gaussian kernel did not sum to 1, therefore, resulting in less values after each convolution. To fix this, I divided the final gaussian kernel by the sum of the whole kernel to get a proportional kernel so pictures don't change brightness.

2. **Packages** used were as same as MP#1. I only used OpenCV and Numpy for matrix manipulation. However, I had to upload the images as grayscale since uploading normally would result in three layered images. Which was not required.
3. **Description** of functions:

   Gaussian: As described before the kernel was made through a nested for loop which calculates the normal and gaussian exponential and populates a 2-d Array. After populating the array, all values get divided by the sum of the gaussian kernel. This results in a well-distributed gaussian kernel which prevents the picture from getting dimmer while convolving with kernel.

   Gradient: I used the Sobel operator to calculate this. Basically, convolving two kernels recognizing vertical changes and horizontal changes. The kernels basically took a derivative of the picture on the x and y axis. Then, taking the (magnitude) gradient of the two results F = sqrt(x^2 + y^2)

would give us the vague representation of edges (high changes in the img). Theta was considered to be a 2-d array of the gradient's direction.

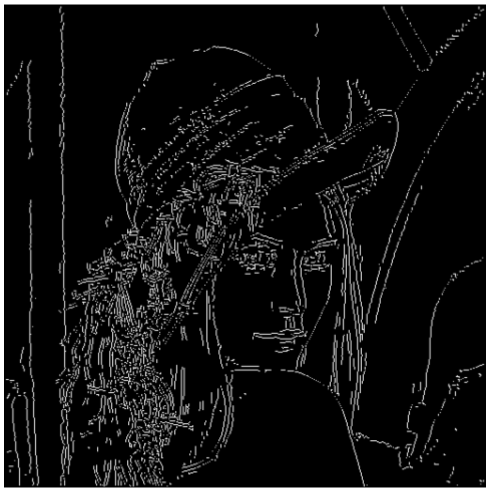
Suppressing Non-maxima: Suppressing technique used was to use the Theta array and the produced gradient image to look at surrounding maxims. Theta array is in radians therefore, to make it simpler I changed it all to 0-180 degree. This was done by simple math 180*radian/pi. Since we had negative radians, I added 180 to any value under 0. As described in the lectures suppression is by looking at the direction of Theta and look at the pixels that the gradient (Directional) passes. By looking at those neighbors called p,r we decide if the current pixel is the local maxima or not. If not, we set it as 0.

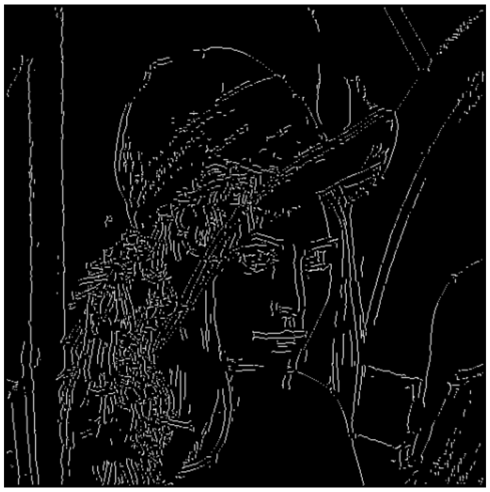
Edge Linking: **(Hysteresis)** This has not been done recursively as the lectures suggest. I have used two nested for loops to set values based on the set thresholds and link the strong to weak thresholds. First nested for loops takes care of finding values that are lower or higher than the thresholds. Values that are less than weak threshold are set to zero. Values in between set to weak*maximum value and Values higher than the strong are set to strong*maximum. Since we have only three values in our array we can compare the values. The second loop takes care of linking the edges. If an edge is Weak and one of the 8 neighbors is Strong we set that edge to be strong otherwise it is a 0.

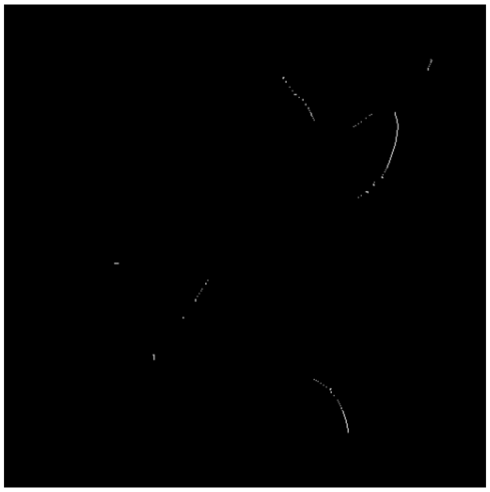
4. **Results: These are all after Edge Linking Hysteresis process.**

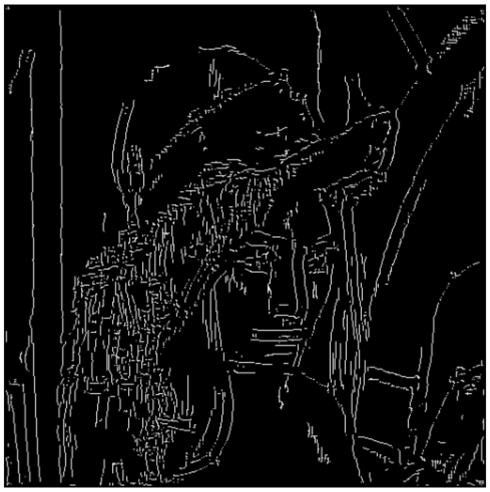
There were 4 test cases: I refer to pictures using the numbers.

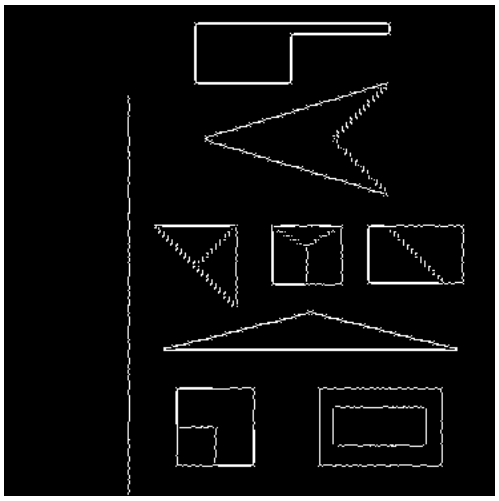
1)	Using Gauss kernel 3x3 sigma 1.	Lower Thresholds Weak= 0.05 / Strong= 0.1
2)	Using Gauss kernel 5x5 sigma 5.	Higher Thresholds Weak= 0.4 / Strong= 0.8
3)	Using Gauss kernel 5x5 sigma 5.	Lower Thresholds Weak= 0.05 / Strong= 0.1
4)	Using Gauss kernel 11x11 sigma 10.	Lower Thresholds Weak= 0.05 / Strong= 0.1

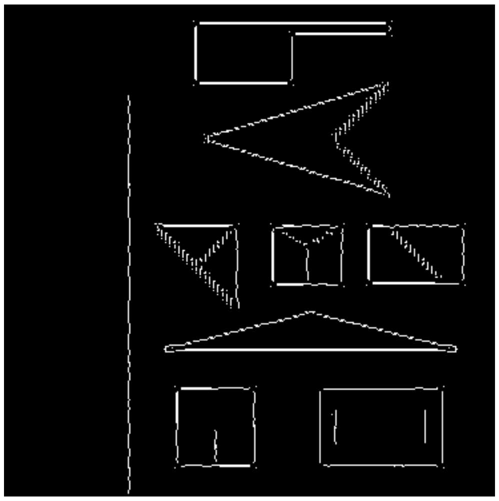
1.

3x3 sig1 Low threshold

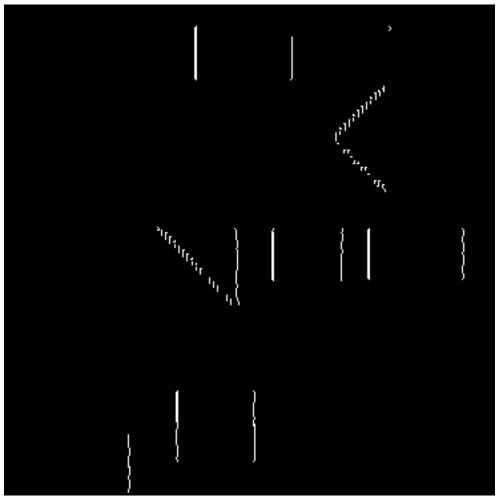
3.

5x5 sig5 Low threshold

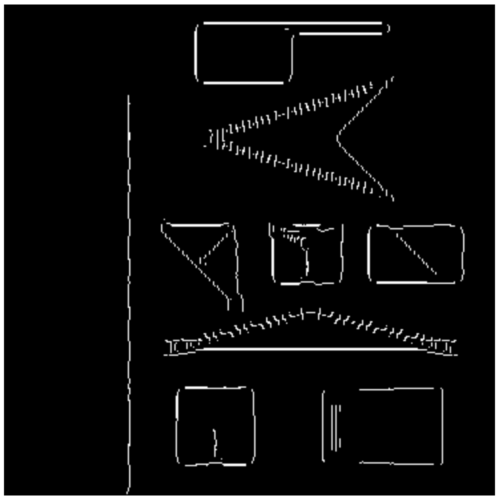
2.

5x5 sig5 High threshold

4.

11x11 sig10 low threshold

1. 
3x3 sig1 Low threshold

3. 
5x5 sig5 Low threshold

2. 
5x5 sig5 High threshold

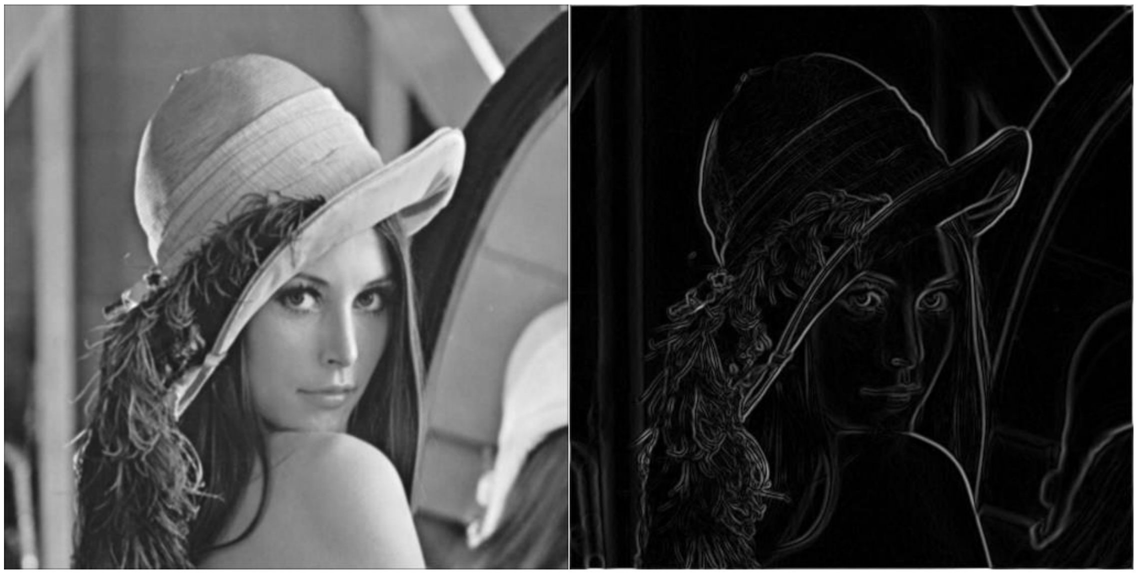
4. 
11x11 sig10 low threshold

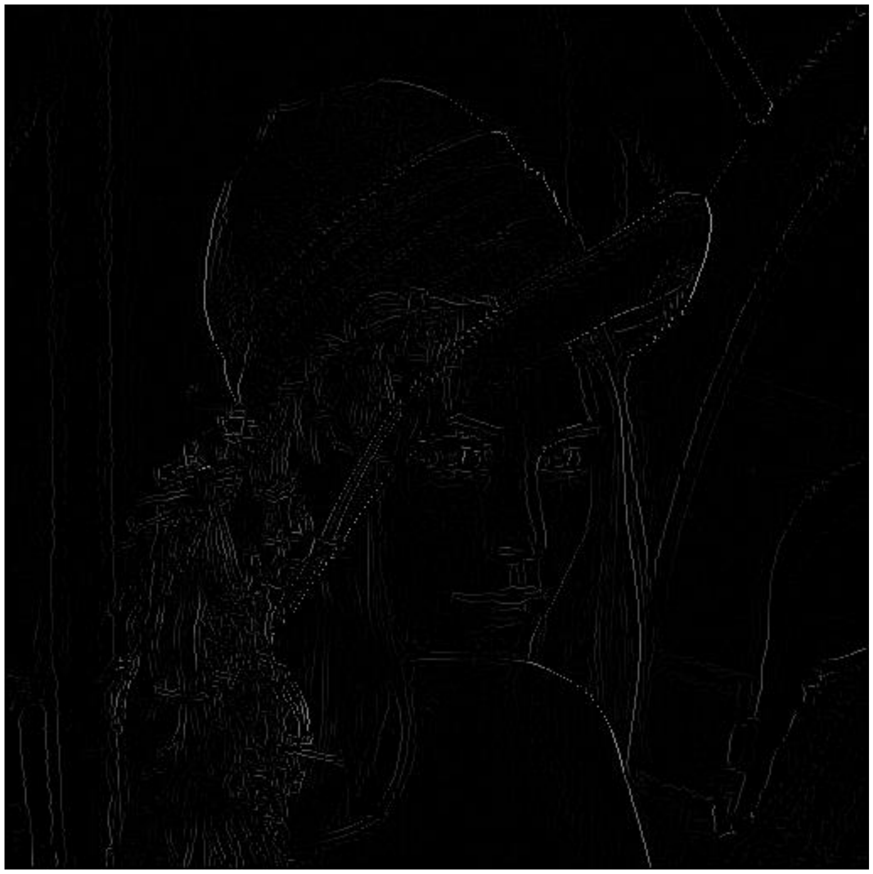
Process of getting the results are shown below for each result:

- **Gaussian kernel convolved.**
- **Finding image gradient and Theta**
- **Suppressing non maxima**

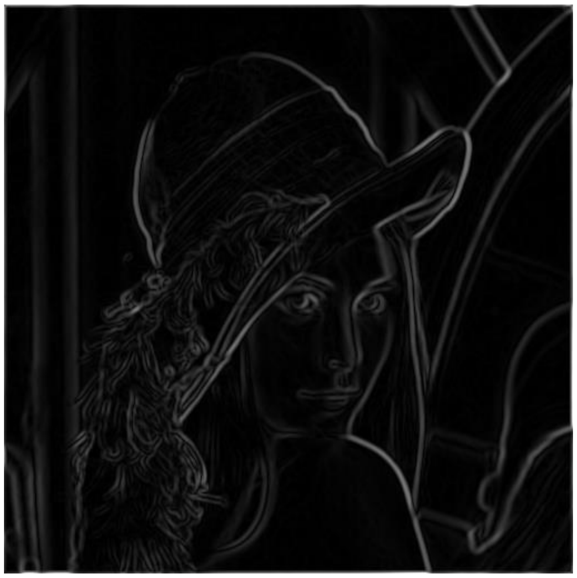
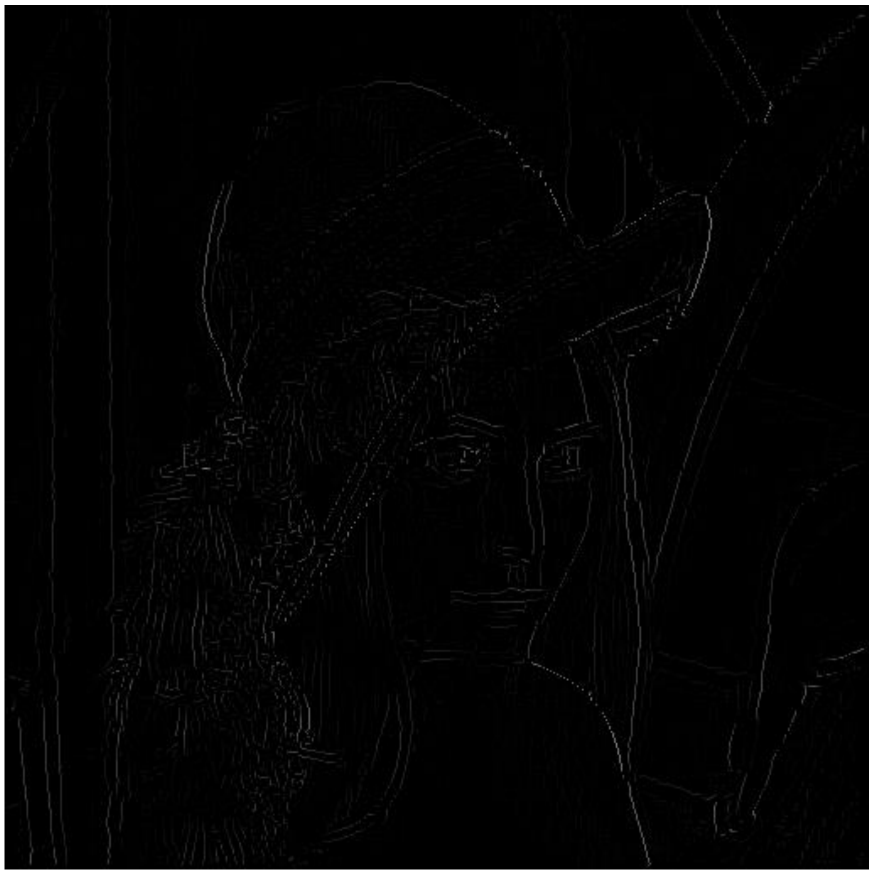
LENA _ GRAY :

1) **3 by 3 gaussian kernel sigma-1 with low percentage thresholds 0.05 and 0.1**

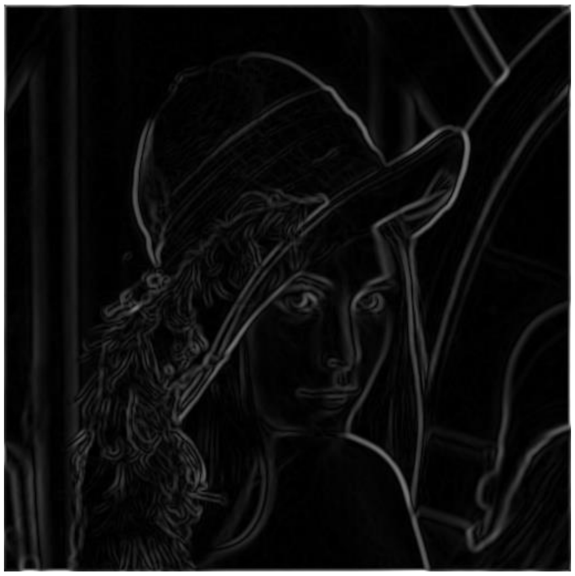


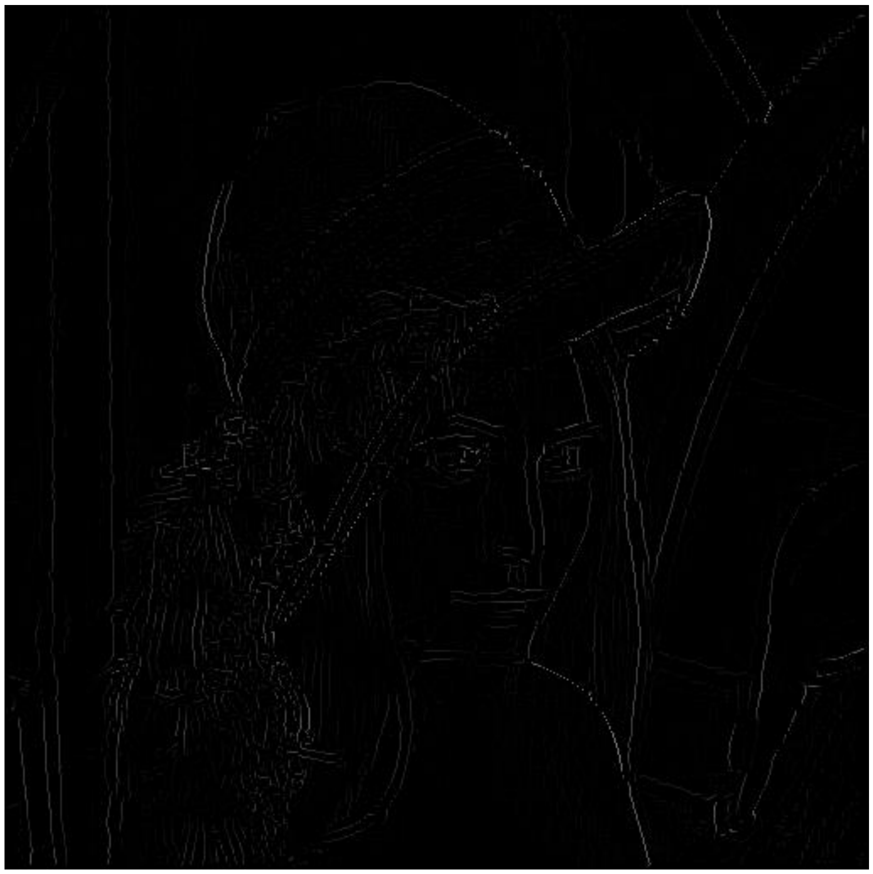


2) 5 by 5 gaussian kernel sigma-5 with low percentage thresholds 0.05 and 0.1

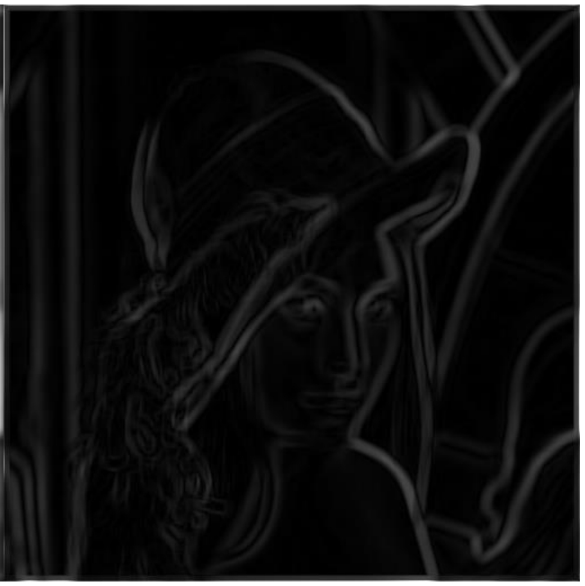
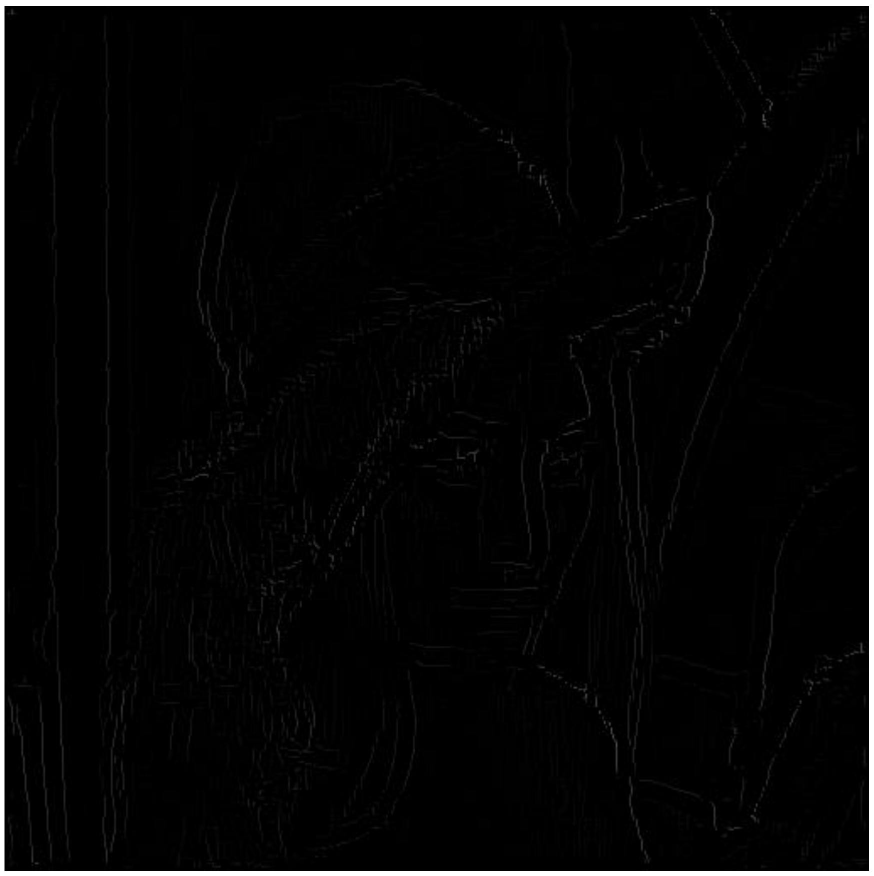



3) **5 by 5 gaussian kernel sigma-5 with High percentage thresholds 0.4 and 0.8**



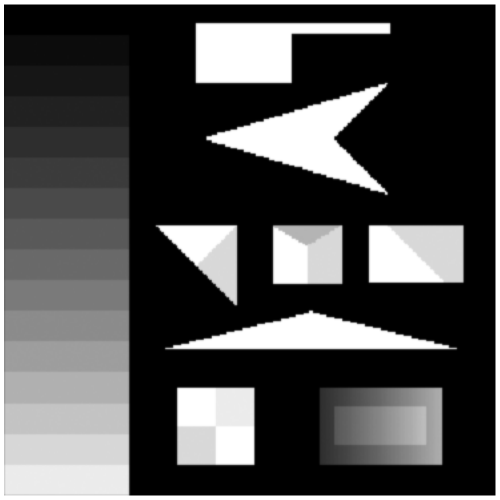


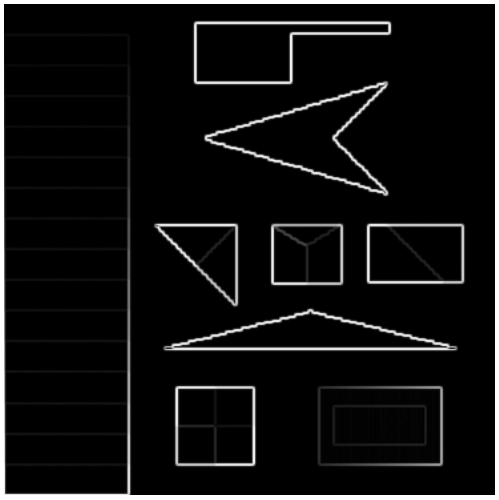
4) **11 by 11 gaussian kernel sigma-10 with Low percentage thresholds 0.05 and 0.1**

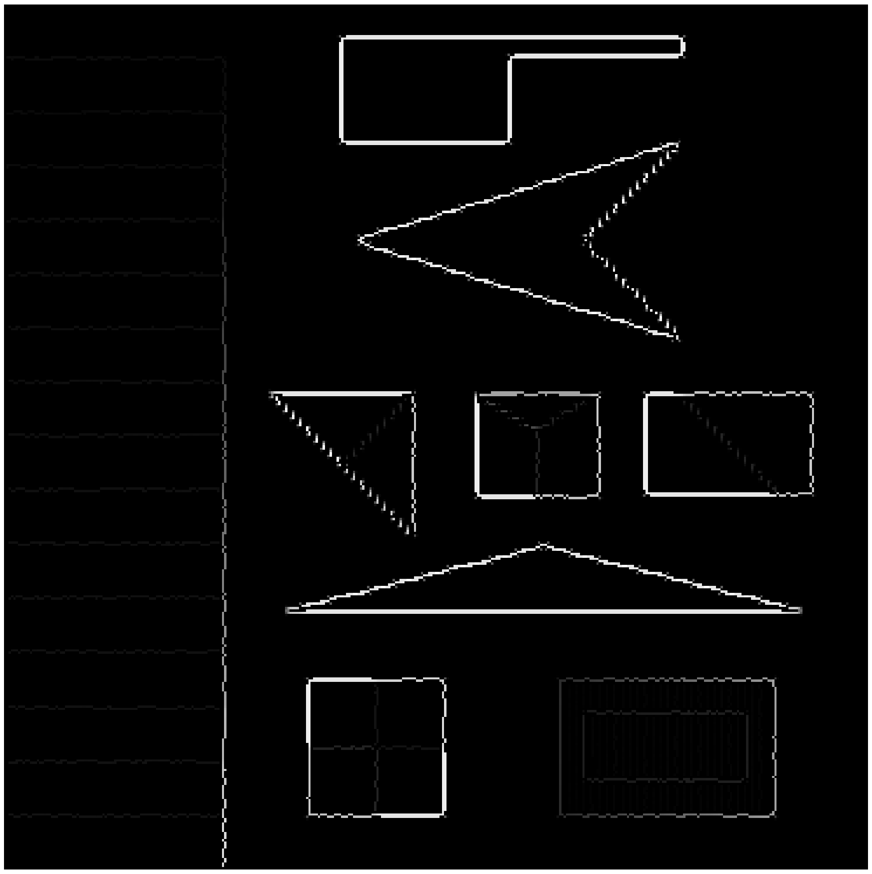



TEST :

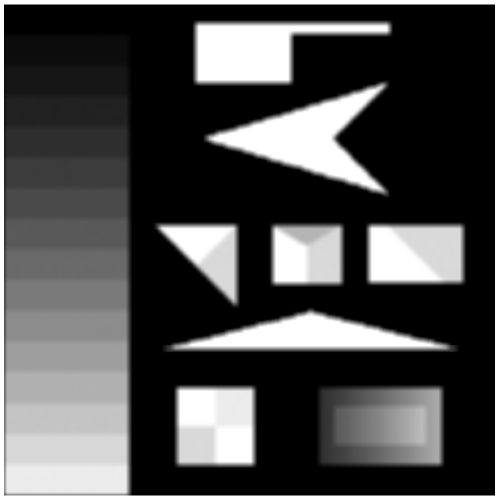
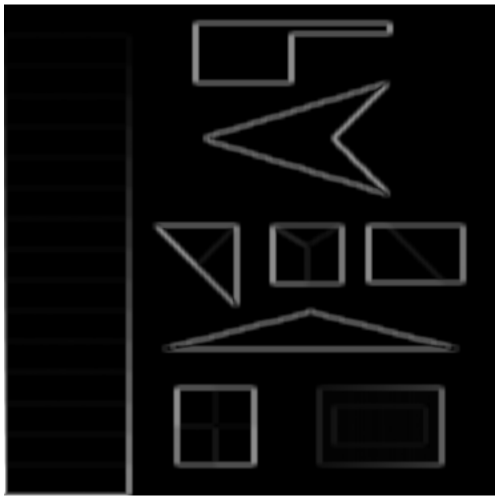
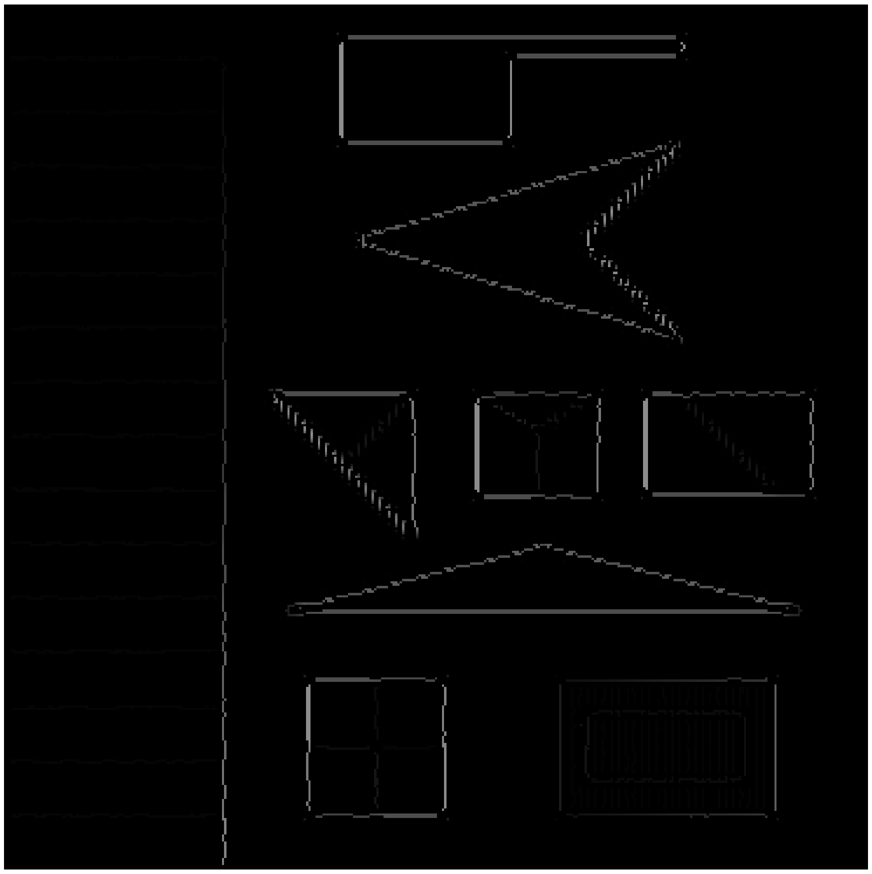
1) 3 by 3 gaussian kernel sigma-1 with low percentage thresholds 0.05 and 0.1



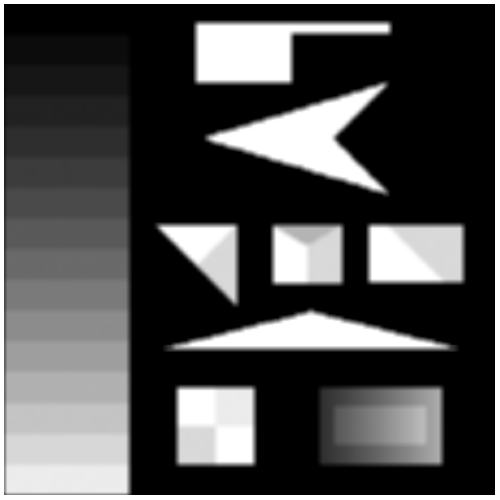


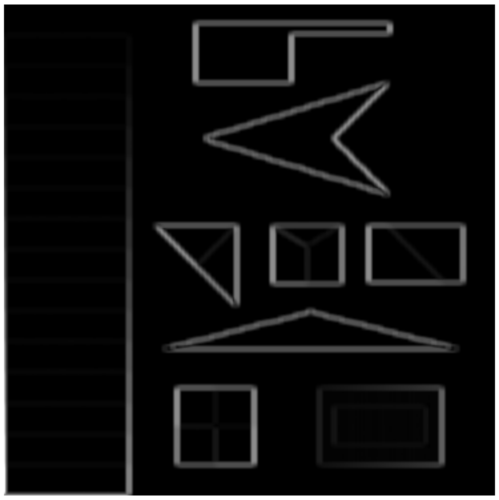


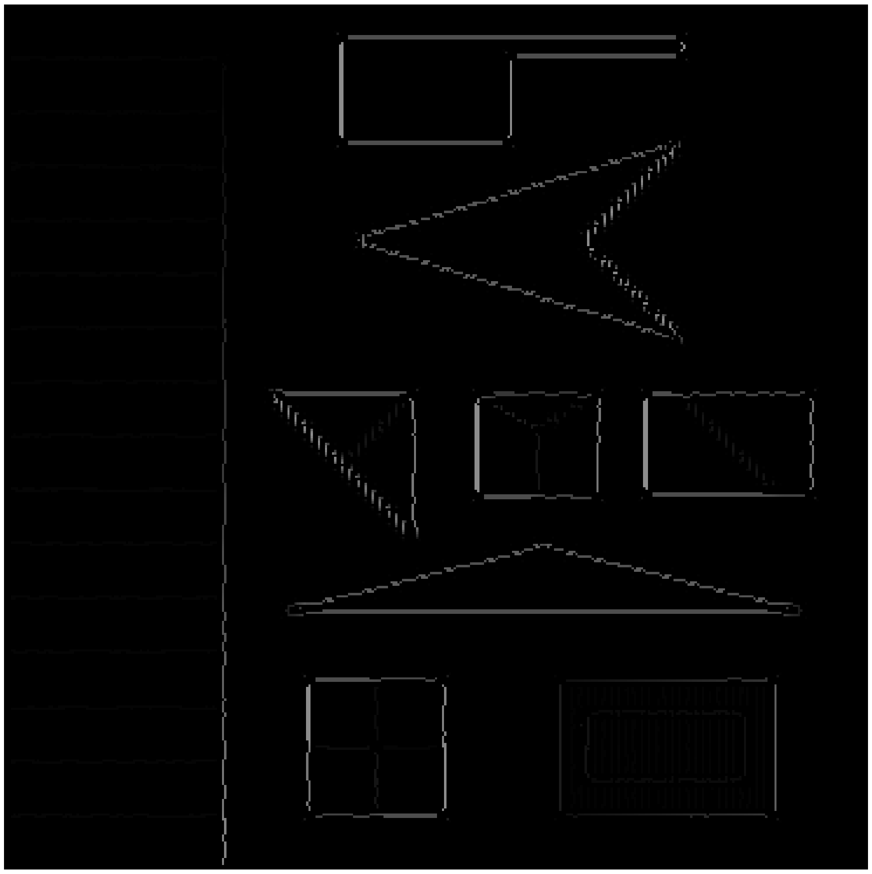
2) 5 by 5 gaussian kernel sigma-5 with low percentage thresholds 0.05 and 0.1

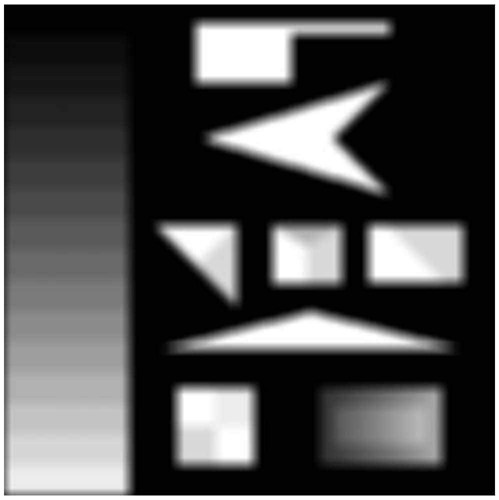
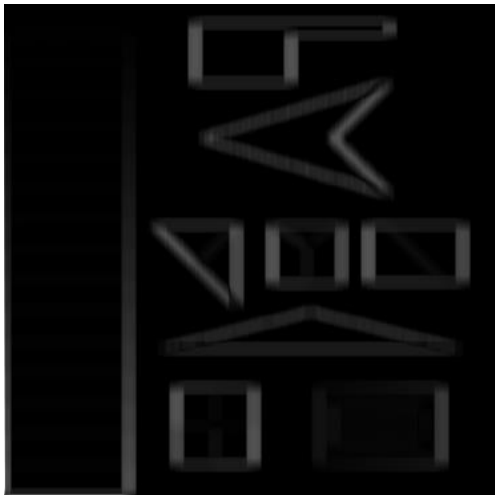
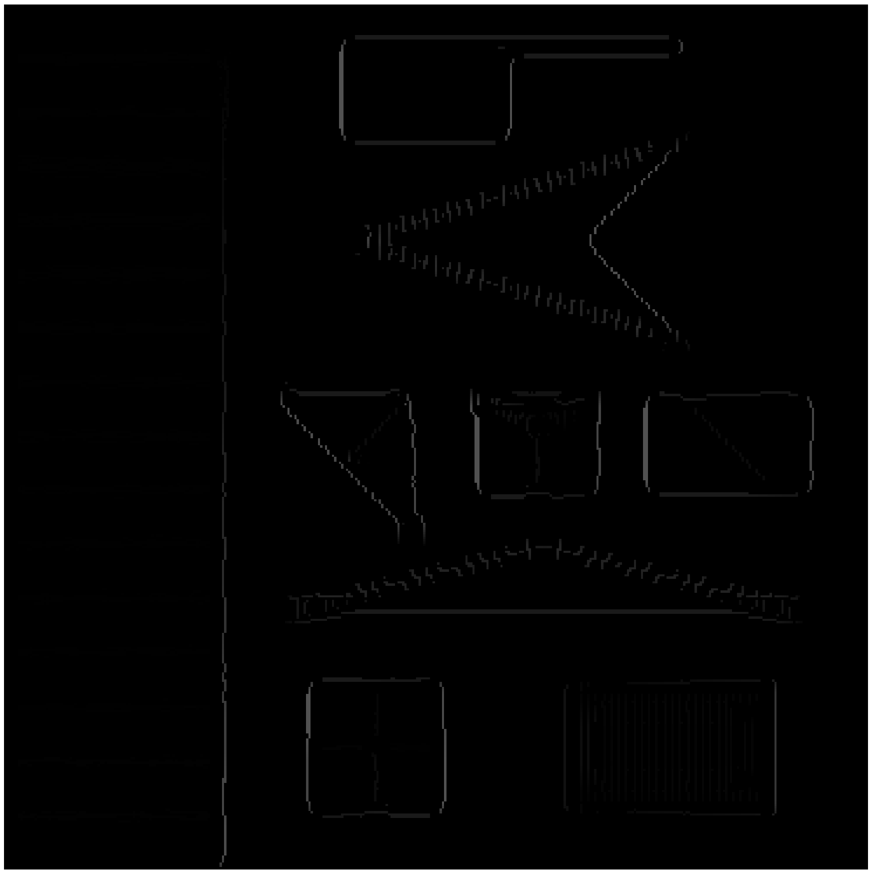
3) **5 by 5 gaussian kernel sigma-5 with High percentage thresholds 0.4 and 0.8**







4) **11 by 11 gaussian kernel sigma-10 with Low percentage thresholds 0.05 and 0.1**

Expectations and Reality:

1. Gaussian sigma has a major effect on the blurring process due to it distributing the data more. Therefore, the data doesn't just act like median or mean blurring. As it is shown in the results for **Lena.png** we get way better results when we use higher sigma values.
   The size of the kernel doesn't make much of difference when it is very large since it just blurs the same data with the same sigma value.
   What surprised me was the fact that the same kernel sizes and values were used for **Test.png** and the kernel with the smallest value had the best result. The reason for this was that Test.png had more detailed edges which disappeared by smoothing the image too much. However, Lena had more obvious edges by smoothing out the picture due to many noise.
2. The gradient had negative values which got translated into positive when trying to display the image. At first, these values were being clipped to 0-255 which made the result not be as detailed. The theta also wasn't being calculated right due to many division by 0.
3. Suppressing the maxima was straightforward, however, all the angles had to be divided equally since we cannot just look at 0 degree for r and p. The angles had to be in every 22.5 degrees range on both +,- sides in order to be able to cover all 180 degrees. Which is why to capture a gradient at 0 degree we look at any degree between 0-22.5 and 157.5-180.
4. Edge linking and thresholds. The only surprising result was when manipulating the values for strong and weak edges. When a very High value of probability was used, the

graph was not as detailed. Since many edges might not be as thick and strong as believed. Therefore, I used a very much smaller value. However, I still kept the ratio between the weak and strong as half. The result wouldn't differ by much when different ratios were chosen. It was different when the values were bigger.

Other surprising results by changing values were higher Sigma and Lower Threshold doesn't always mean better results. As saw in the Lena and Test images, when the image has many noise and obvious edges, smoothing and blurring helps detect more edges. When image is clean and has more detailed edges with low contrast, smoothing doesn't help as much.

As my result show, using higher thresholds disregards many details, therefore a proportional ratio which capture most data is preferred.

The final results were much more detailed after doing the edge linking due to less threshold.