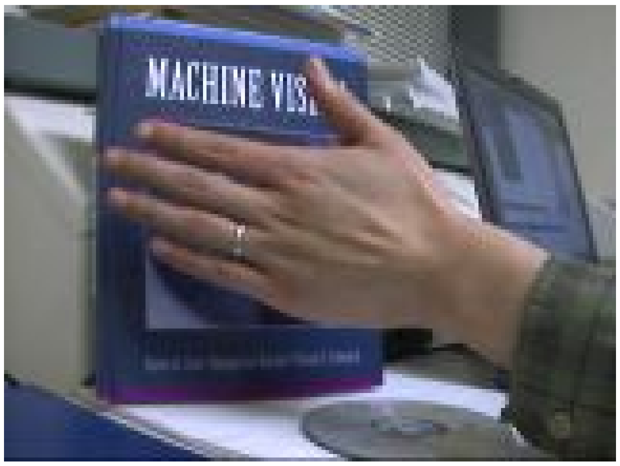
Kyle Qian

Canny Edge Detector

For this project we worked on implementing a canny edge detector, composed of many smaller parts. We also compared different implementations of the canny edge detectors and comparisons of their output. The images we used as testing are shown below:



For reference, the top left and top right images are Lenna and test1 respectively, and bottom left and bottom right images are joy1 and pointer1 respectively.

<u>The Algorithm</u>
This algorithm is broken up into several parts:

1. Gaussian Smoothing:
    - Use a gaussian filter to smooth the given grayscale image
2. Calculate Image Gradient:
    - Use a kernel (Sobel, Roberts, Prewitt, etc.) to produce the magnitude and direction of the edge map
3. Find High and Low Threshold:

   - Use a histogram to determine two thresholds for step (5), using a magic number percentageOfNonEdge

4. Suppressing Nonmaxima
   - Find local maxima of image gradient using nomaxima suppression techniques
5. Thresholding and Edge Linking
   - Use the thin edges generated by (4) and recursively link edges together from (3). The resultant image should include all high threshold edges, and all low threshold edges that are connected to a high threshold edge.

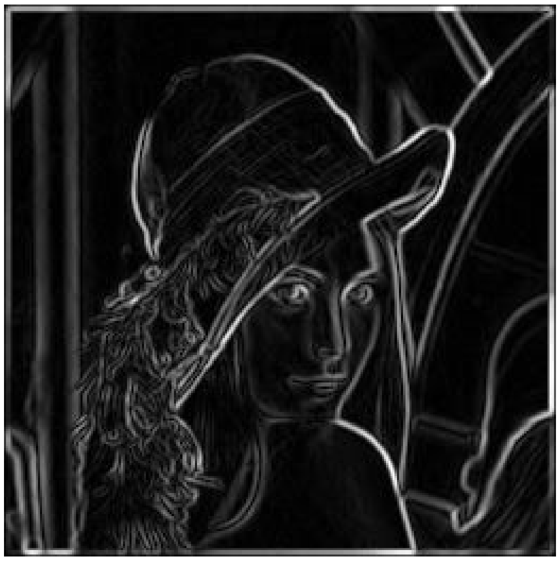
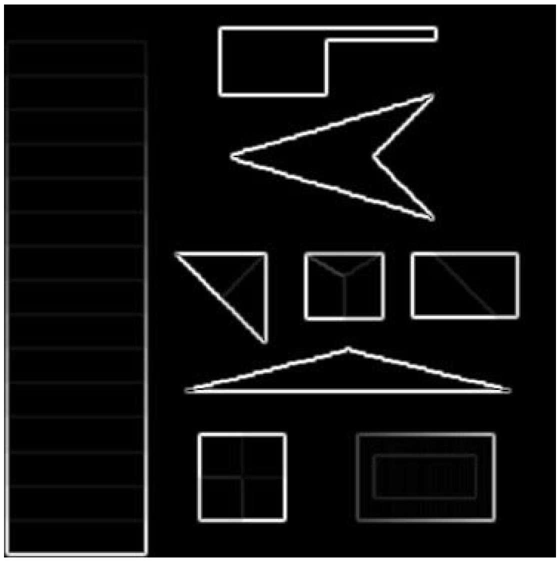
<u>Implementation Results</u>

- Gaussian Smoothing

Above we can see our 4 testing images with a [3x3] gaussian smoothing kernel with sigma = 1 convoluted with the original images. Edges have become much smoother and there appears to be much more blur in the image. We will see later what affects changing the size and sigma of the gaussian kernel will do.

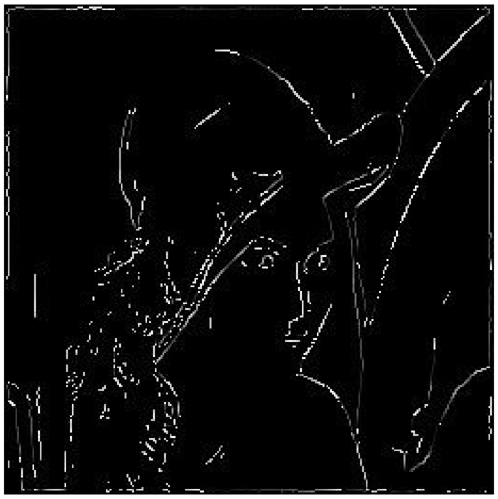
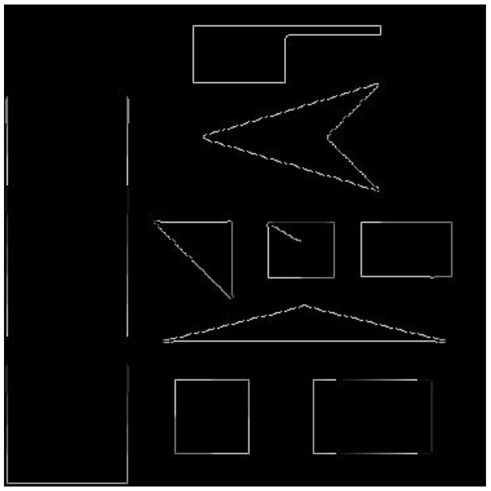
These gaussian smoothed images will be used in our image gradient, as they are much easier to work with.

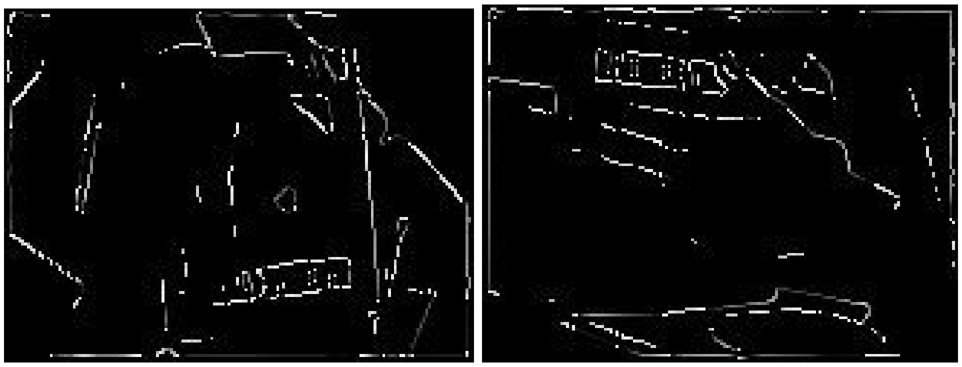
- Image Gradient

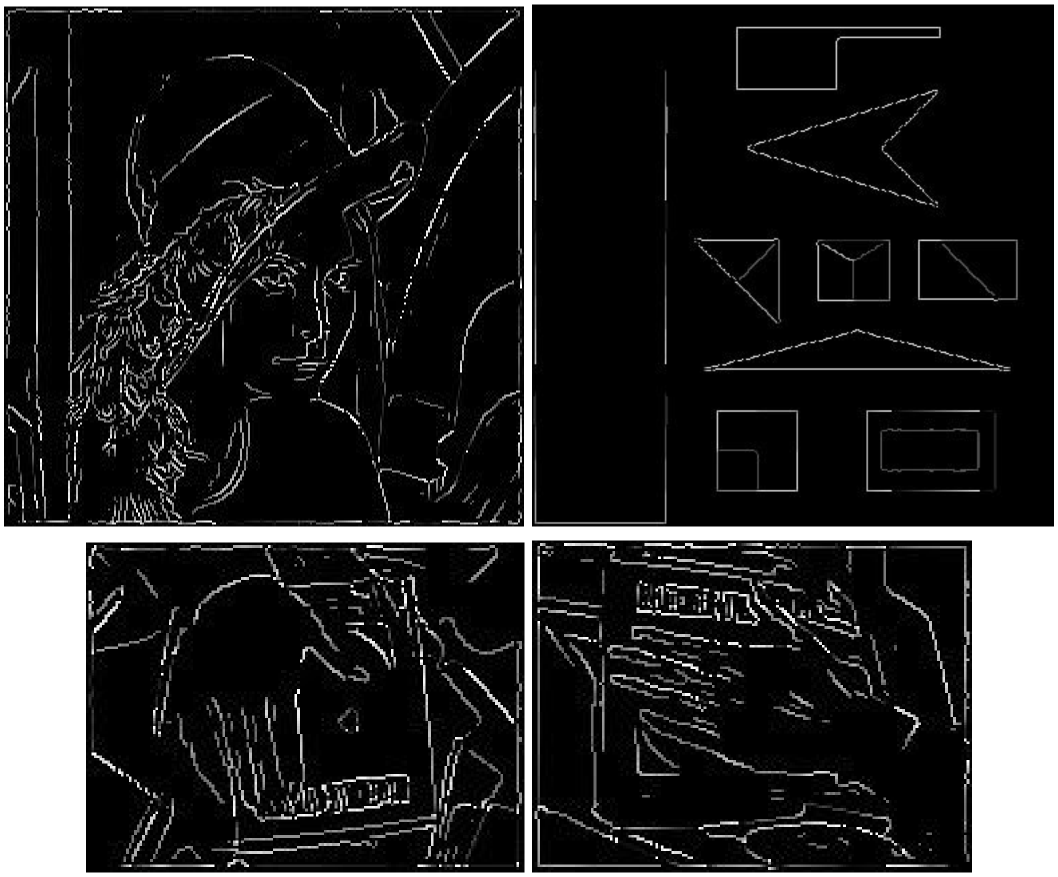
Above we can see our 4 testing images using a Sobel kernel to find the gradient of the image—depicted are the magnitudes of the image gradient. This is already a sort of edge detector, as we can see edges formed by differences in the gradient; However, it is quite messy for more complicated images like Lena. We use the magnitude and direction of the image gradient for finding high and low thresholds and suppressing nonmaxima. Later, we will see what different kernel types will produce.

- High and Low thresholds

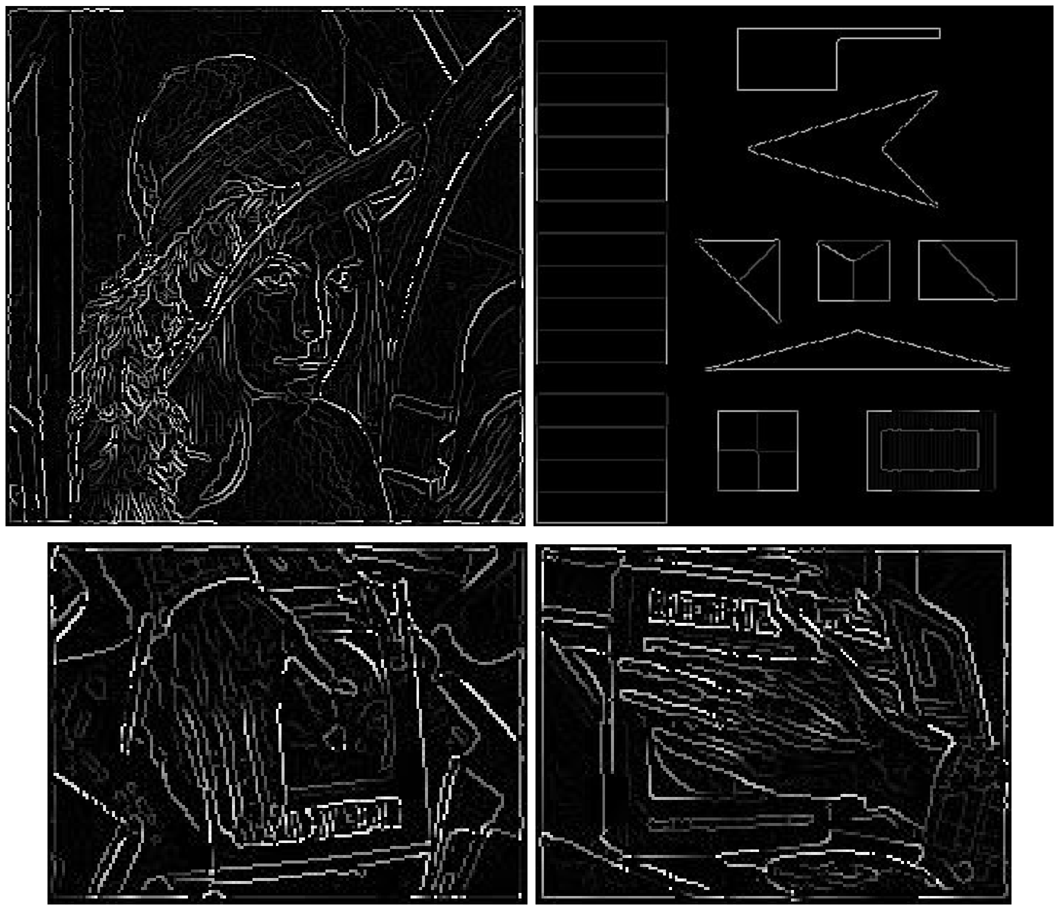


The above images are our test images under the high threshold with a percentageOfNonEdge = 0.95. This means that it will only show pixels with a value greater than 95% of all pixels from the suppressed nonmaxima results. As we can see, it only shows the most defined edges from our image gradient, having many broken lines and gaps.



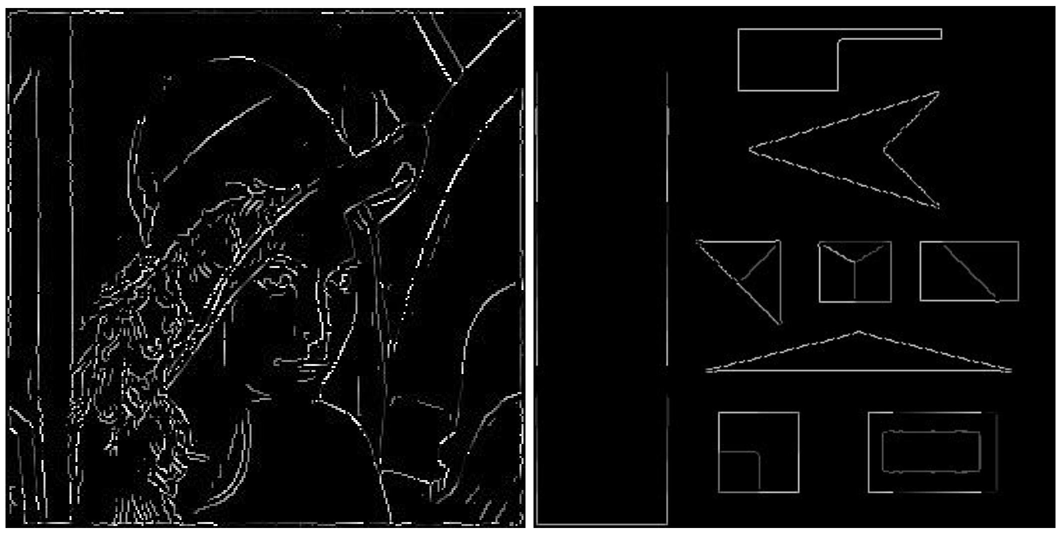
The above images are our test images under the low threshold with a percentageOfNonEdge = 0.95 and T_low = ½ T_high. This means that it will show pixels with a value greater than 47.5% of all pixels from the suppressed non maxima results. As we can see, it has many more edges than the high threshold image, but now is a lot more noisy than the high threshold image.

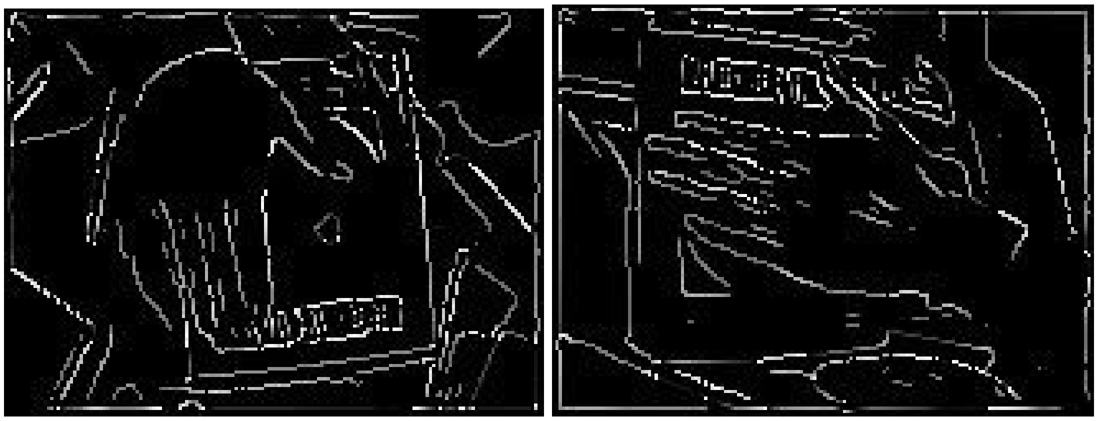
- Suppressing Nonmaxima



The above images are our test images with suppressed nonmaxima images from the Sobel image gradient. These were used in our previous low and high threshold results. As we can see, these images are very similar to our image gradient magnitude images, but with much crisper edges, as we now only include local maxima for each pixel.

- Thresholding and Edge Linking





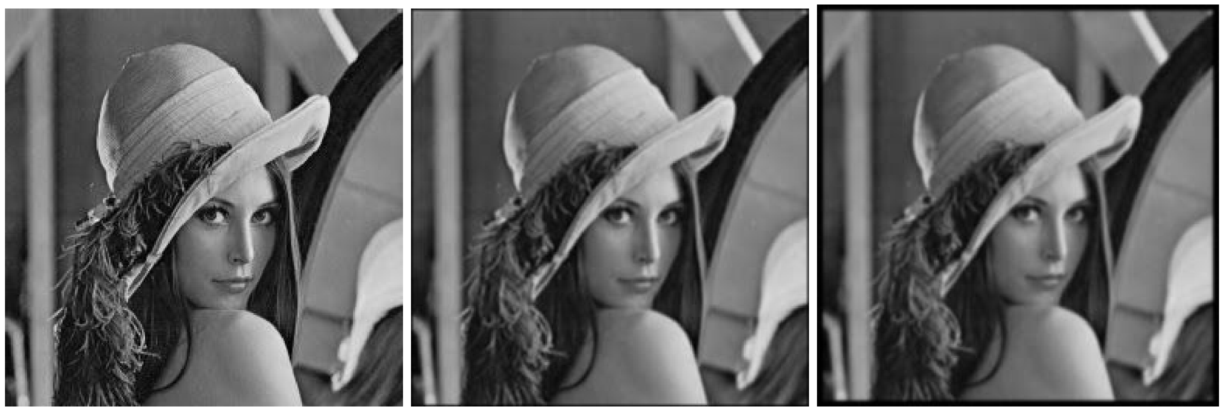
The above images are the final result of our canny edge detector on our test images. We get these by combining the high and low threshold images, but only including curves from the low threshold image if they are connected to an edge in the high threshold image. These have much clearer edges and greatly reduced noise compared to the image gradient magnitude images.

Analysis Results

- Different Gaussian smoothing functions:

The above images are the Lena test image with a [3x3] gaussian kernel, but with sigma values of 1, 3, and 5 respectively from left to right.

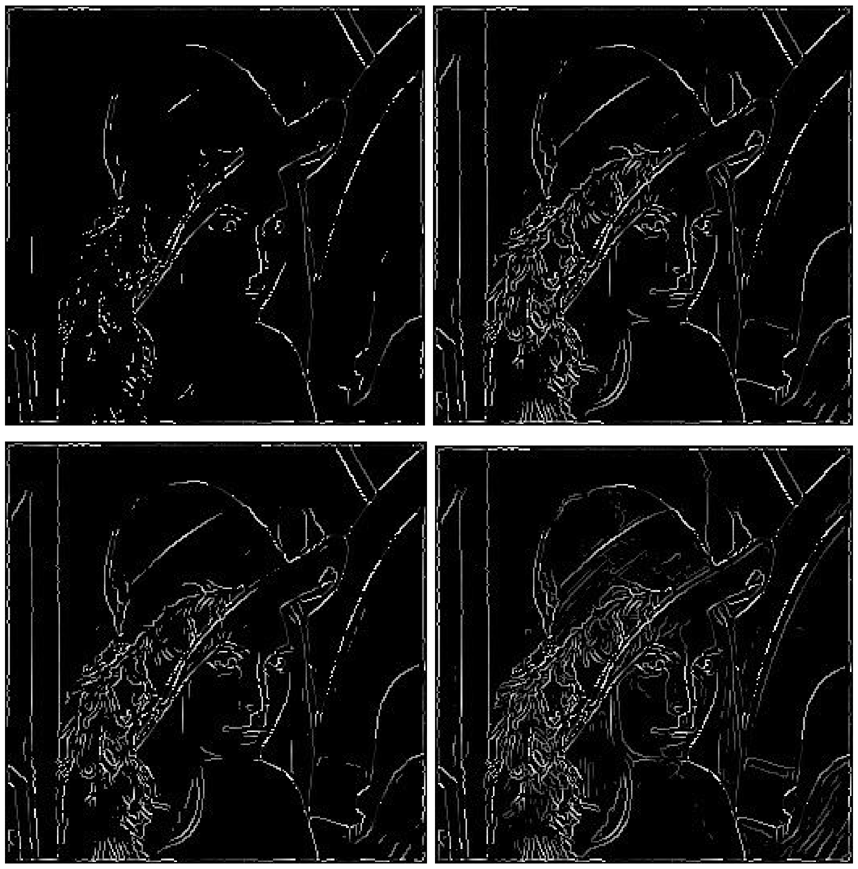
Increasing the sigma values means that our gaussian kernel will tighten its deviation to include less data. We can see that it only slightly increases the blur of each image— though not by much— as it focuses more on the center pixel for each convolution.

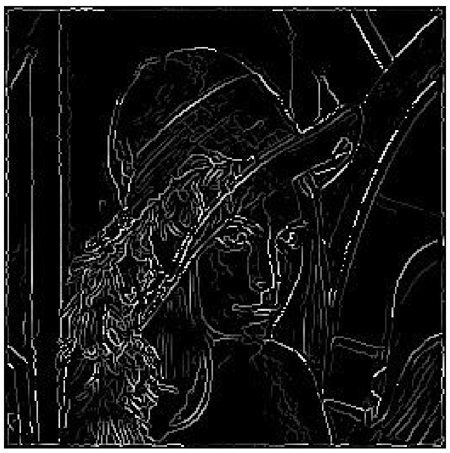
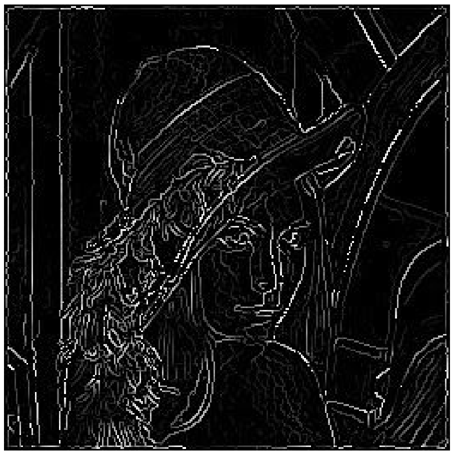


The above images are the Lena test image with [1x1], [3x3], [9x9] gaussian kernels respectively from left to right, with sigma values of 1 for each of them.

Increasing the size of the gaussian kernel means that each pixel will take more of its surroundings pixels into account during convolution. We can see this effect on the image, as the greater the size of the kernel the smoother the image gets.

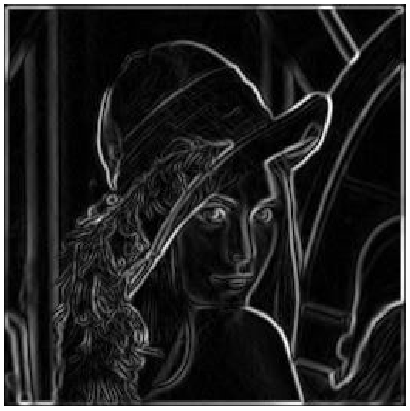
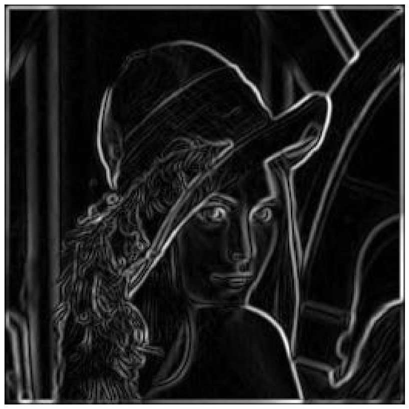
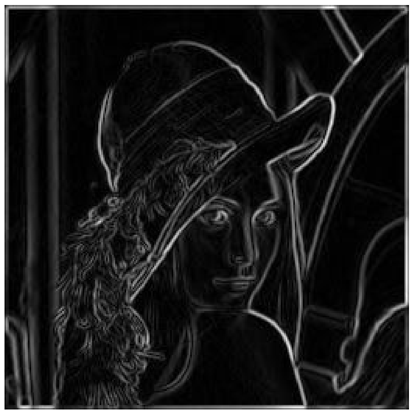
- Different percentageOfNonEdge



The above images are the Lena test image, with the left and right images representing high threshold and low threshold respectively, with changing PercentageOfNonEdge of 0.95, 0.8, and 0.5 from top to bottom respectively. As we can see, the more you increase the PercentageOfNonEdge, the more edges our high threshold image encompasses. This is what we expect, as we are lowering our high threshold to include more and more pixels. By the time we use 0.5 our high threshold and low threshold images are nearly identical.

- Comparing different edge detectors

The above images are the magnitude of the Lena test image with the Sobel, Prewitt and Robert, respectively from left to right, kernel image gradient convolutions. As we can see, the different edge detectors produce slightly different results. Sobel has a larger emphasis on the pixels directly on the left/right and up/down from the pixel being convoluted, so we can see that it has brighter strong edges. In contrast, Prewitt has an even kernel for each pixel's neighboring pixels. This gives it a more even spread of edge strength. Robert, being only a [2x2] kernel, includes fewer values per pixel, so it's edges are much darker compared to Sobel and Prewitt.