Project 1

Part 1

The gaussian kernel function creates a kernel that is 3sigma by 3sigma. The myfilter function takes in a kernel and an image and applies that kernel to each pixel on the image. This is done by the weightedsum function, which takes the surrounding pixel values and sums them according to the weights in the kernel.

Filtering images using higher sigma gaussian kernels will result in blurrier images. You can see this pattern by looking at the differences between standard deviations of 3, 5, and 10.

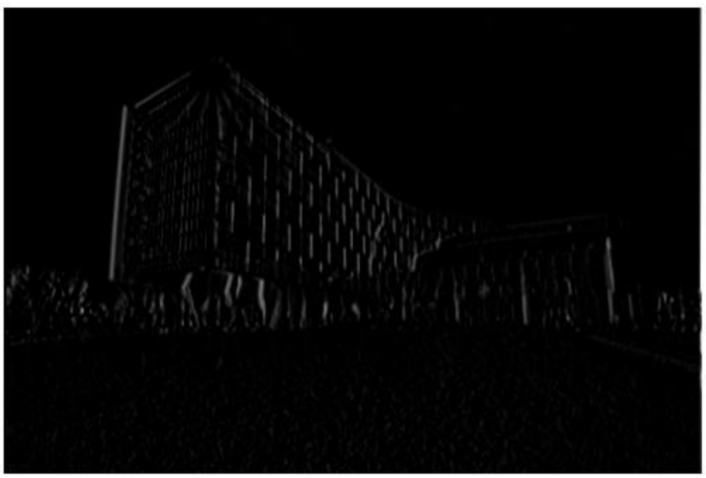
The h1 filter increases the contrast of the image. It also slightly decreases the brightness of the image.

The h2 filter is similar to the h1 filter, except it contrasts vertical lines, and decreases the brightness even further.

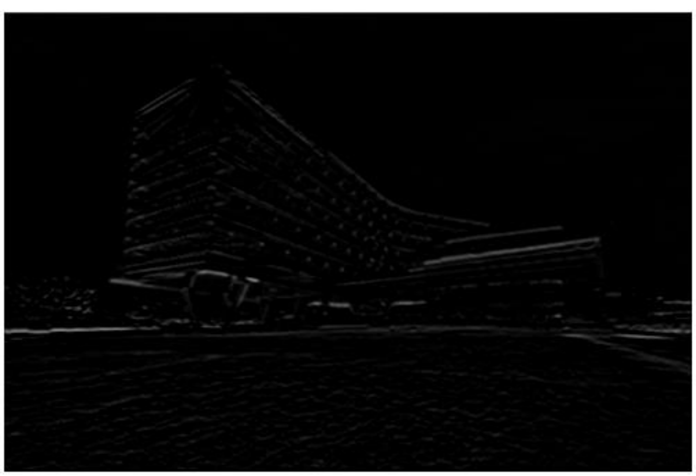
The h3 filter is the basically the same as the h2 filter, except that it contrasts horizontal lines.

Part 2

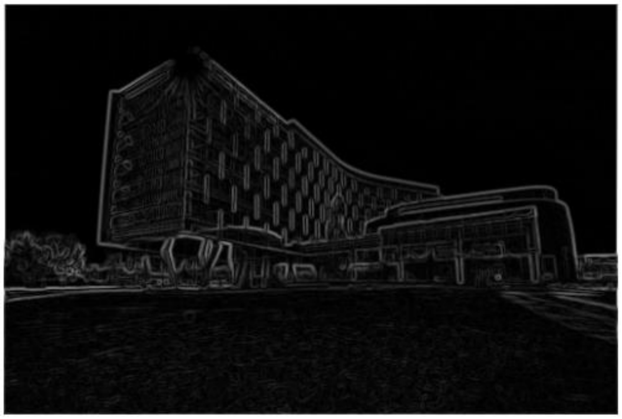
The partial derivatives Dx and Dy were gained by using the myfilter function with the given Sobel filters. The gradient magnitude was achieved by going through each element and applying the magnitude formula. The angle of the gradient was obtained by doing arctan(Dy/Dx) on each element. A problem that occurred was that Dx would sometimes be equal to 0, resulting in a divide-by-zero error. To work around this, if Dx equaled 0, then I set the angle to 90, since it could not be any other angle if Dx equaled 0. The derivative of the angle was acquired by going through the angle array and rounding it to 0, 45, 90, or 135 degrees. Since arctan has the range of [-pi/2, pi/2], it was only necessary to check for the intervals inside this range.



Dx with sigma=1



Dy with sigma = 1



Gradient magnitude with sigma = 1

The edge thinning process first adds a 1-unit wide padding around the image, since the algorithm examines pixels in a 3x3 box. For each pixel, the function created a 3x3 subarray of the pixels around it. The edge thinning process keeps an edge pixel if it has the highest gradient magnitude of two other nearby pixels. The position of the nearby pixels is determined by the angle of the center pixel.

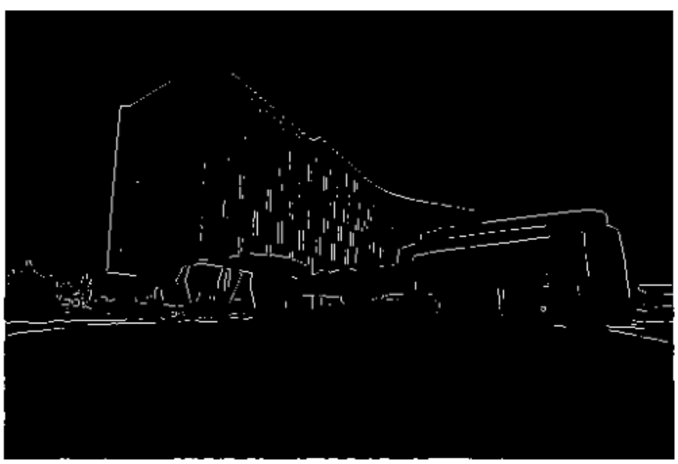
The hysteresis thresholding process first sets a pixel to black if it is below the t_low threshold, and white if it is above the t_high threshold. Otherwise, the algorithm uses the scipy label function to find and label all connected features. It then finds the edge that the pixel is connected to, and goes through each point of the edge and tries to find a pixel that is above the t_high threshold. If it does find one, then it sets the edge pixels all to white.

Final image with sigma=1, t_low=0.5, t_high=1

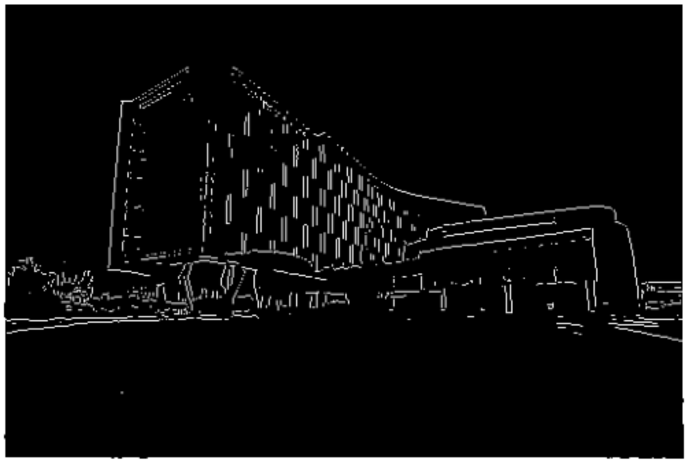
Analysis

The sigma value has a significant effect on the resulting image. Since a higher sigma value results in a blurrier image, it becomes harder to detect places of sharp contrast or high derivatives, and that is where the edge pixels are. Selecting a high sigma value decreases the quality of the final image. Very low values for t_low and t_high are needed for a decent final image.

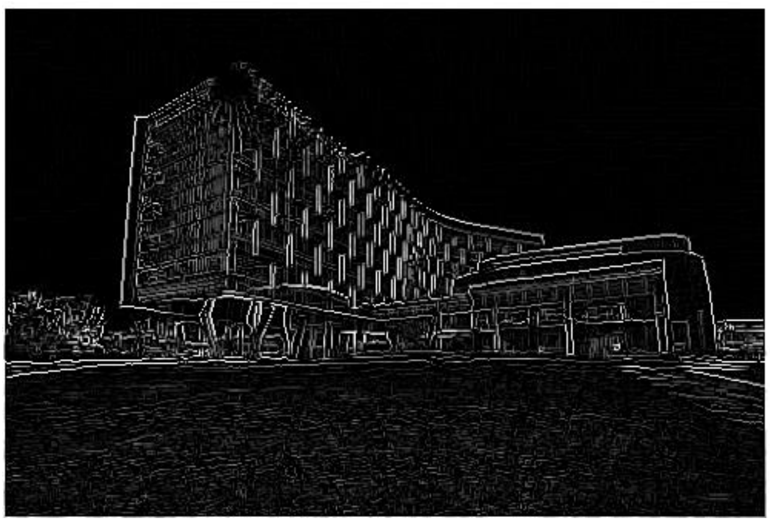
The threshold values have a significant effect on the amount of noise in the final image. Increasing the t_low value removes a lot of noise, especially the edges that are on the grass. However, it also removes a lot of the window edges that are close to the glare. Decreasing the t_high values allows more of the window edges to be captured. This is because the window edges are separated from other edges and will be eliminated if their gradient isn't high enough. Decreasing both t_low to 0.3 and t_high to 0.5 while using a sigma of 1 was the most ideal set of values.



Final image with sigma=1, t_low=0.3, t_high=1



Final image with sigma=1, t_low=0.5, t_high=0.7



Final image with sigma=1, t_low=0.3, t_high=0.5

Final image with sigma=3, t_low=0.1, t_high=0.2