Computational Photography 6475
Jacob Fund
Summer 2015

For the first 2 functions I simply subtracted a shifted image back one with the numpy.roll function. I had to make sure the pixels could be negative so I made the image arrays floats. Then I removed the last row or column depending on the function since there was one less entry.

For the gradient function, I needed to make the kernel compute all pixels. Since I could not use a filtering function like imfilter, I just ran two loops around the rows and columns of the image and used the numpy.tensordot which sums the kernel with the window in the image for the particular pixel. Then I normalized it by the size of the kernel. I also normalized the gradient functions sum over 8. This Doesn't really matter, but makes the numbers in the image smaller and more appropriate for the Sobel kernel that I will use in the next part.

Figure 2 Input

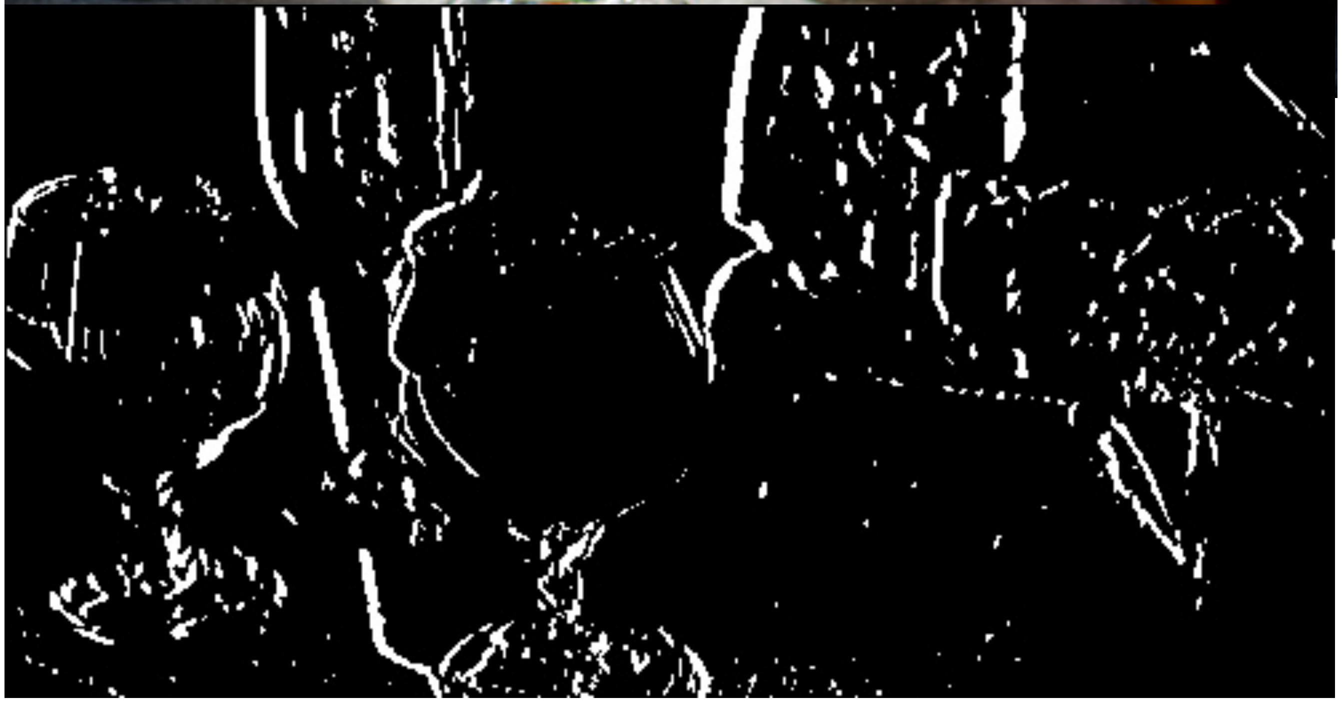

Figure 1 Output

To get the edge image I used the convertToBlackAndWhite method but I hcanged the threshold to a much smaller amount. Specifically, 7 instead of 128. The kernel I used was similar to the Sobel Kernel. That is [-1 0 1],[-2 0 2], [-1 0 1].