HW 5 - Edge Detection

Brock Weekley

University of Missouri

Digital Image Processing

Assignment 5 - Edge Detection

10/28/2021

Abstract

There were four separate parts in this assignment that revolved around detecting edges. Using OpenCV for image I/O, we were tasked with applying a Sobel filter to an input image in grayscale or RGB. In the first part, we were to read an image and apply the Sobel filter with appropriate scaling. In the second part, we were to compute the magnitude of the resulting edge response. In the third part, we were tasked with computing the orientation of the edge response using arctan and plotting the result. In the final part for grad students, we were tasked with applying the Sobel filter convolution a second time and determining what the result meant.

Introduction

Throughout this assignment, our goal was to learn about edge detection and filter convolution with an image. Each part of this assignment was built on the previous parts. After applying the filter, magnitude could be computed, then orientation. I used Python for this assignment. I also used normal Windows window views of images in this assignment, rather than the default OpenCV ones, so image size may appear to differ based on the size of the window - but the image itself is unchanged. Images can be run as grayscale with the --gray flag, and setting the double constant at the top of the program will run the graduate part.

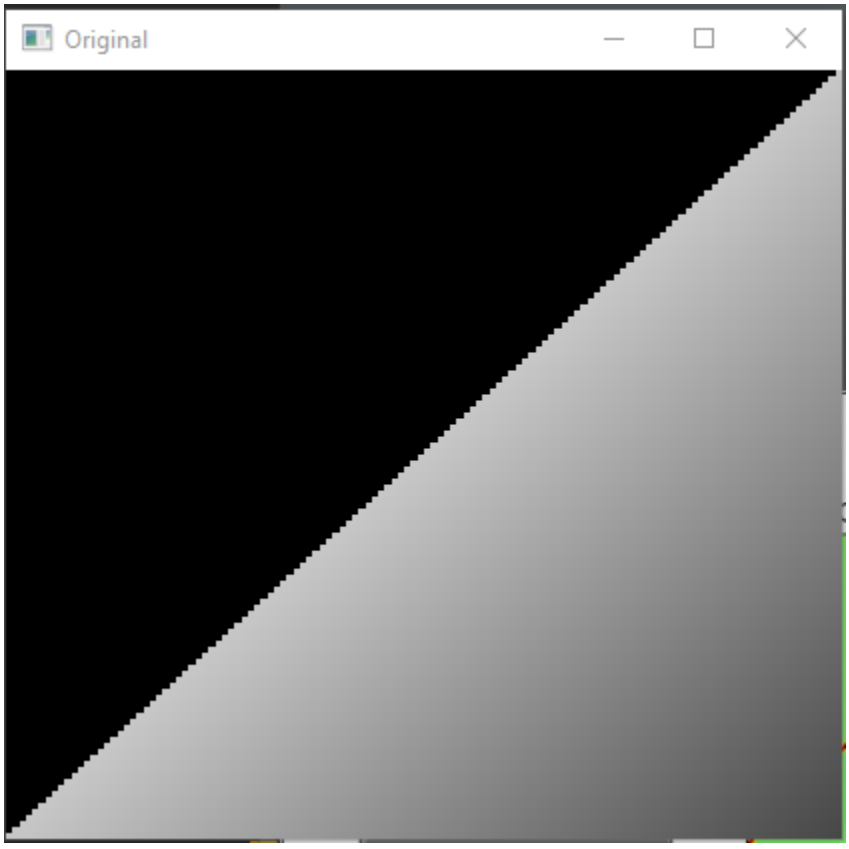


Image 1

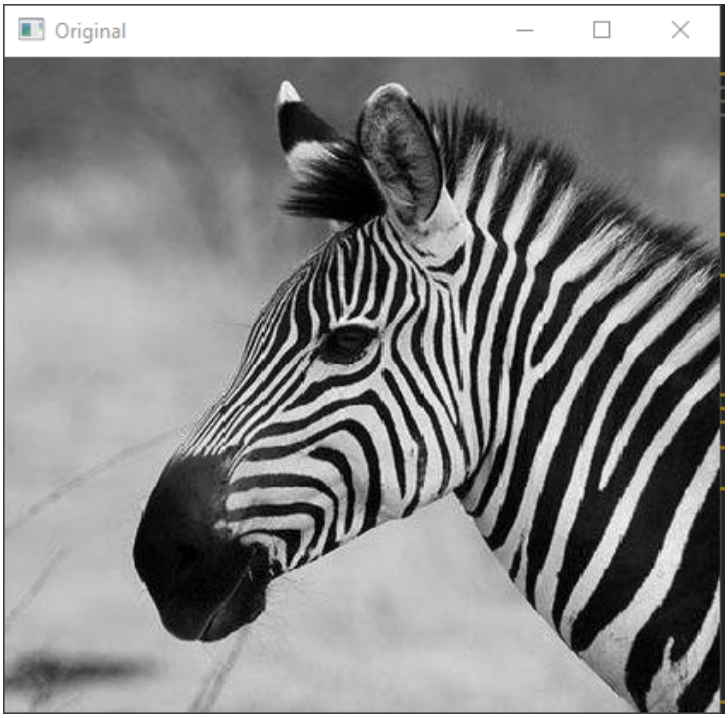


Image 2

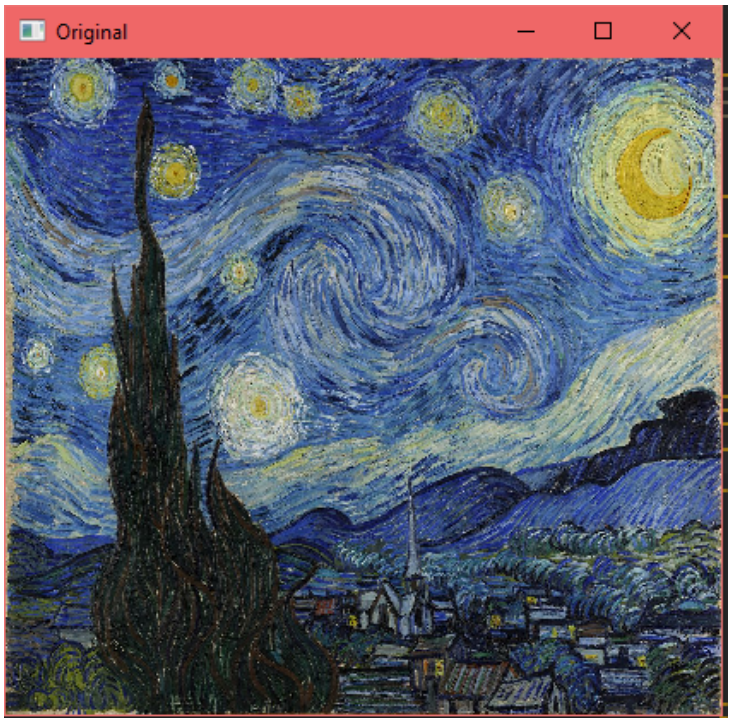
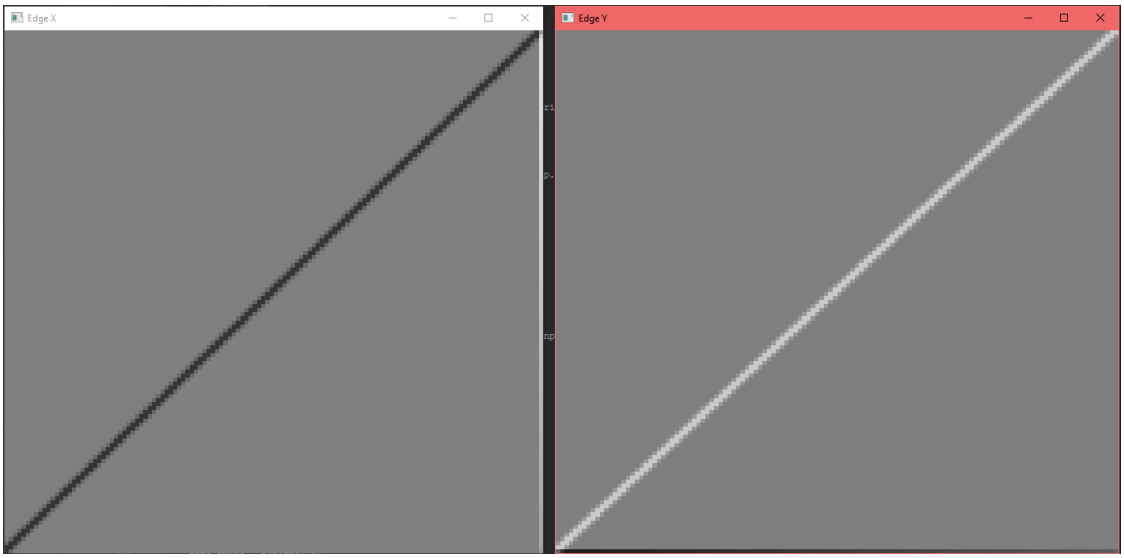
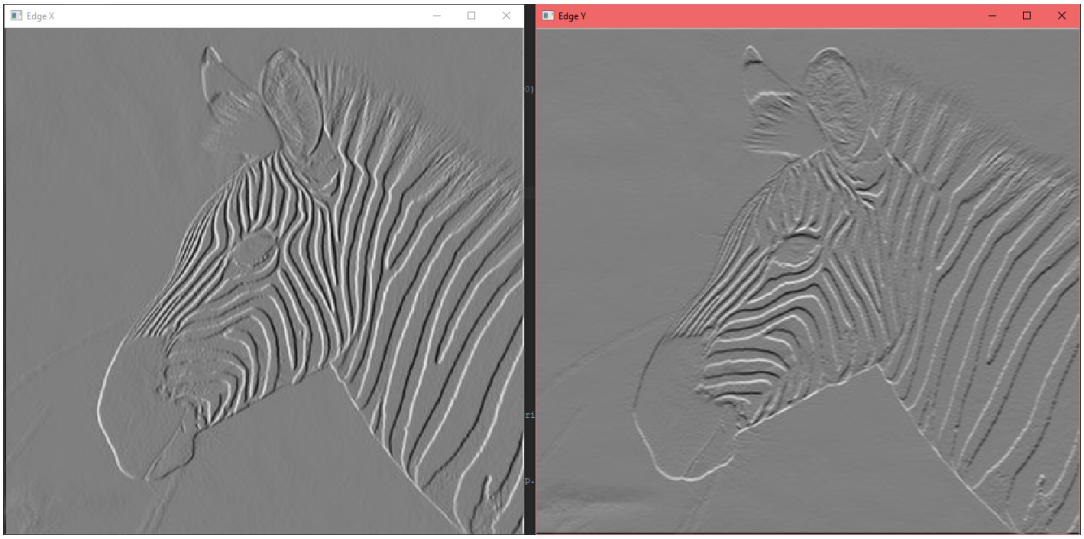


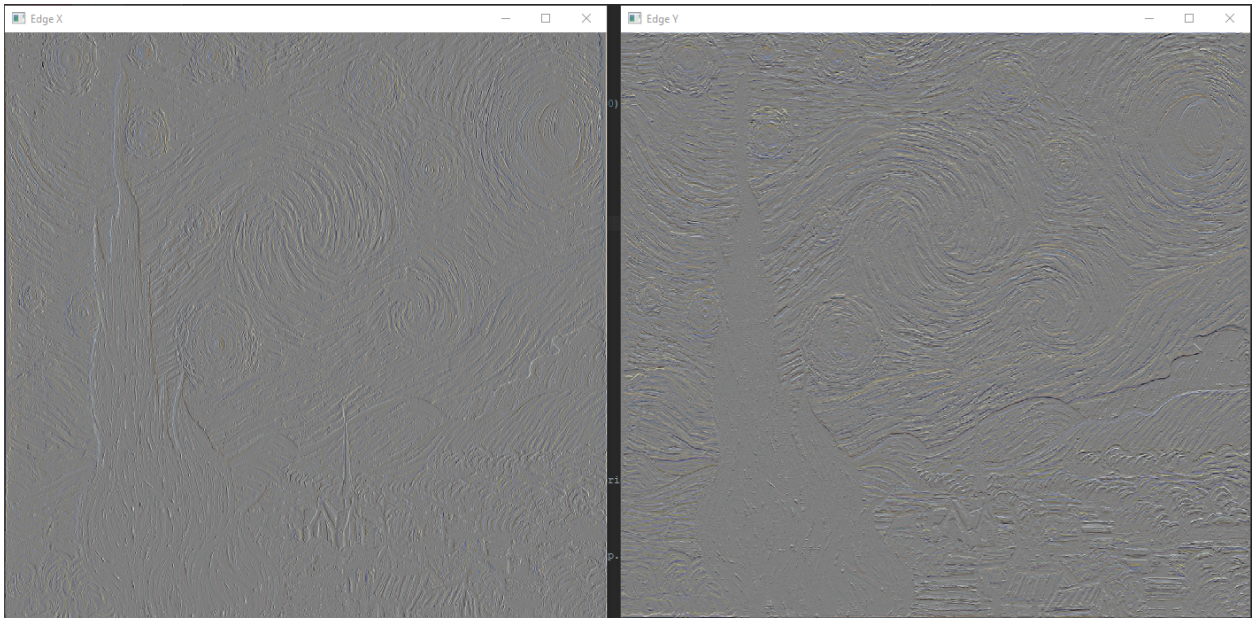
Image 3 (RGB)

Sobel Filter

The Sobel filter is applied to the image twice (Gx and Gy) in order to detect the edges on both the x and y axes of the image. I applied the filter to the image using convolution, flipping the filter appropriately (and scaling) before applying the filter to the image. After doing this, I also scaled the edge detector output before returning it using Ix + 255 / 510 * 255. For RGB images, this process is applied to each channel before the result is combined back into one image. The result is a view of the vertical and horizontal edges:

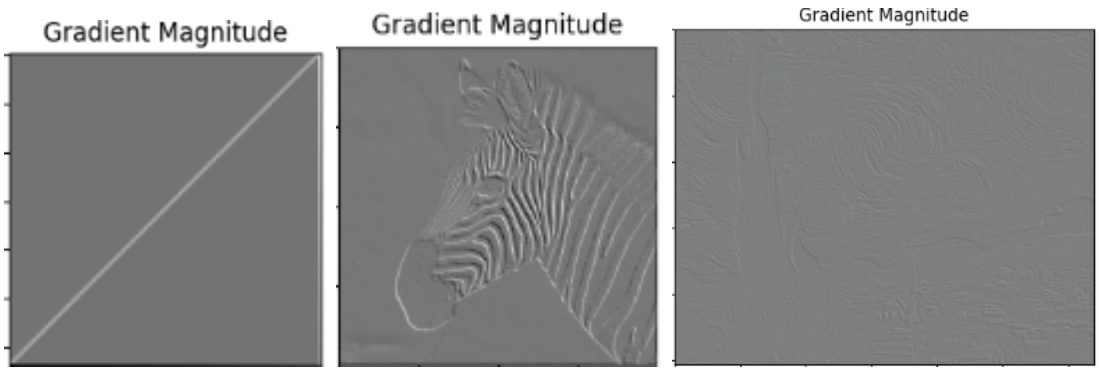






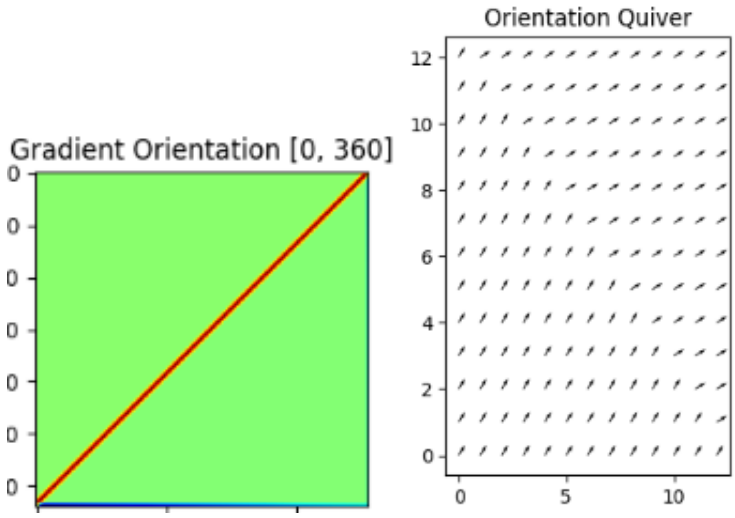
Gradient Magnitude

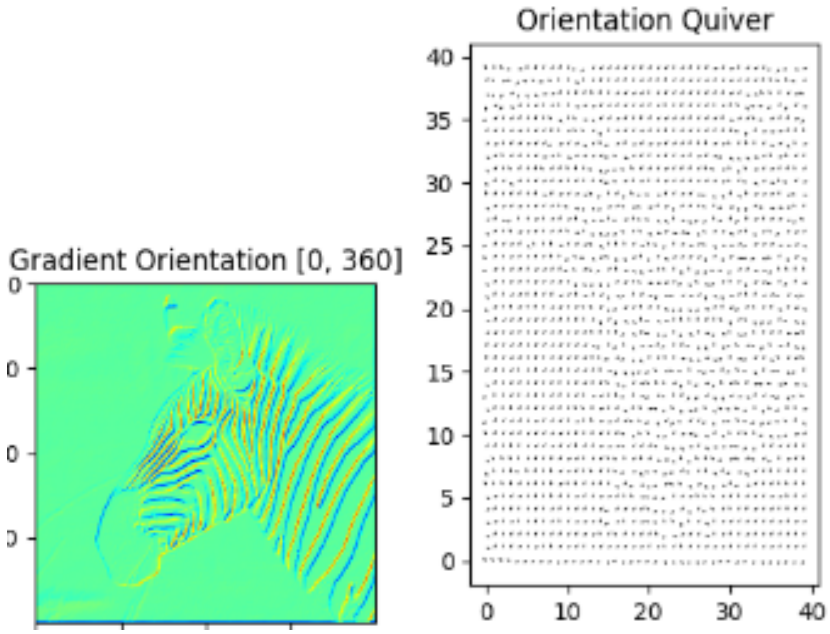
The gradient magnitude of the various images was computed using the formula sqrt(Ix^2 + Iy^2) - or the resulting convolutions of x and y squared, summed, and square rooted using numpy. This was then scaled as well using magnitude / (255 * sqrt(2)) * 255. This is the only result shown for RGB images as instructed. RGB magnitudes are computed on each channel before combination.



Gradient Orientation

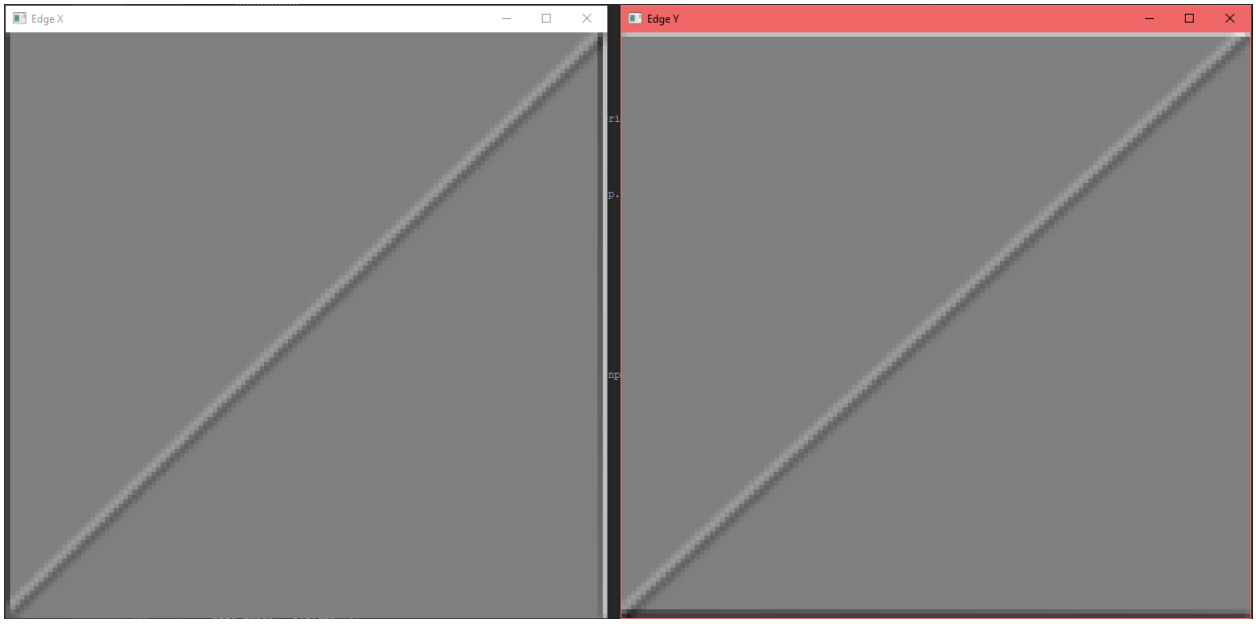
The gradient orientation is computed using the arctan of the vertical edge result and the horizontal edge result. The resulting radians array is converted to degrees, and if the degree is negative, 360 is added. This results in counterclockwise positive degree values in the array. Using matplotlib, the orientations are displayed as an HSV image and every ten pixels are displayed in the quiver plot, as instructed.

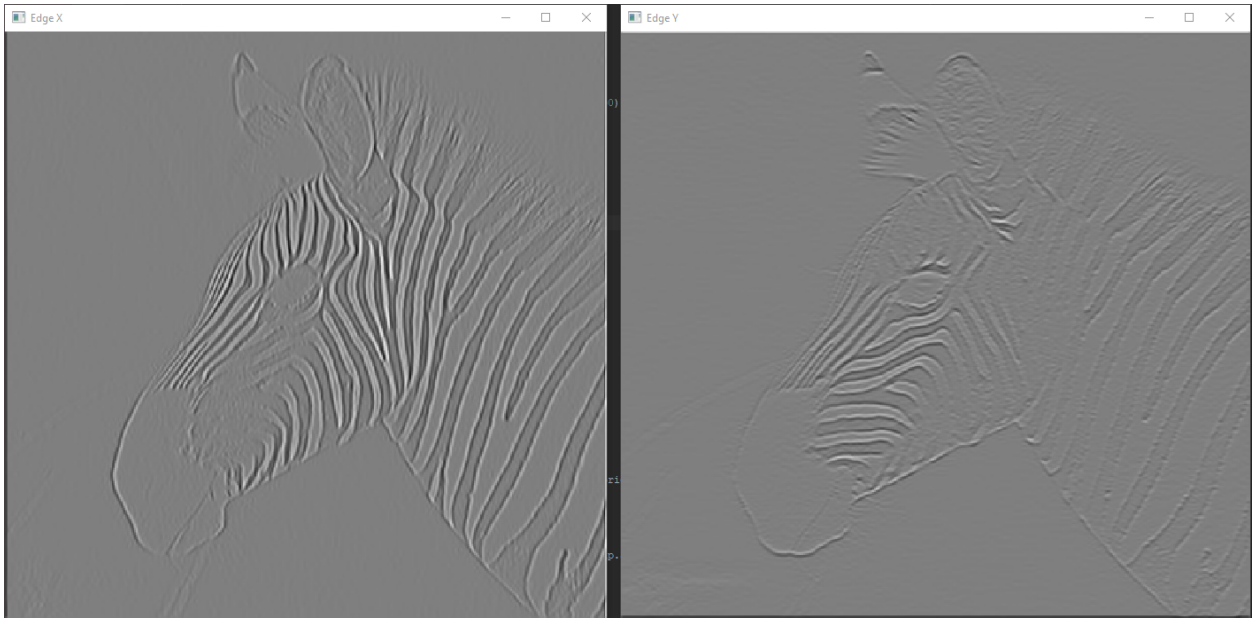


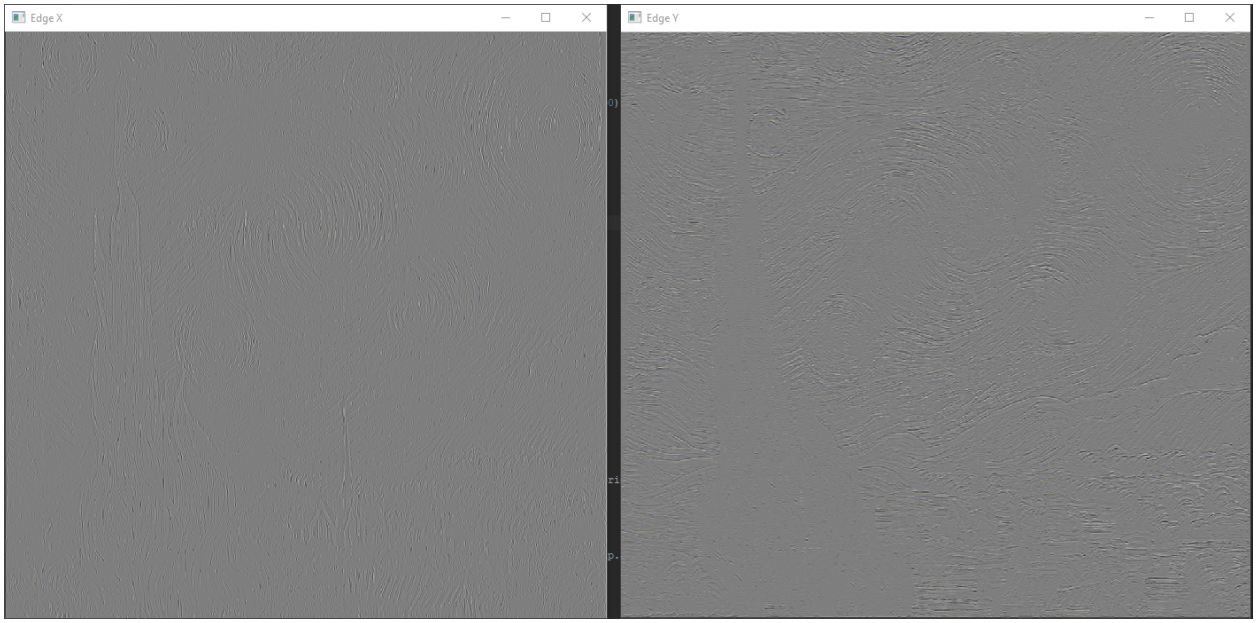


Double Filtering

When the Sobel filter is applied twice, the resulting magnitude image shows lines instead of edges. Computing the second-order derivative means that first a second-order filter must be produced, this can either be done by convolving twice in the program, or by having the original filter input be the second-order filter. I originally had a second-order filter as the input, but ended up convolving twice in my program instead. The result is very interesting for all three images, especially the zebra, as you can see the stripes as the most prominent part of the image.




Edge X
Edge Y


Edge X
Edge Y

Conclusion

In conclusion, the goal for each part of the program was achieved, including the graduate part. In testing each part with prebuilt library functions (SciPy convolve2d), my results matched exactly. The program works as expected for RGB or grayscale images. Throughout this assignment, I have gained a new familiarity with edge detection, filters, and convolution especially. I struggled with convolution at the start, but after lots of help was able to figure it out. I also struggled with the quiver plot, which was a result of me not flattening the input arrays. Images are output with imshow and each imshow has a waitKey after it. Magnitude and orientation are output with matplotlib plot. I used PyCharm to complete this assignment and built the project specifically for Python 3.7. The program takes an input image as a file path for the first argument, then the optional gray flag. Overall, the results of this assignment were very interesting and are likely very useful in many different computer vision applications.