This assignment was used to teach how to implement an edge detection algorithm. The program computes the convolution of the image between two Sobel filters separately, scaling the output, the magnitude of the image, and the orientation of the Grayscale version of the image. All of these are displayed to the user.

HW5 – Edge Detection

10/31/2021

Ryan Orf

CS4650: Digital Image Processing

For this assignment, students were challenged to implement an edge detection algorithm. Either an RGB or Grayscale image would be accepted as input, and depending on the image type, the program would loop either solely based on the row and column counts or based on the row, column, and channel counts. Using the convolution algorithm learned in class, the program calculates the convolution between Sobel filters Gx and Gy at each pixel, along with the magnitude of the image and the orientation of a Grayscale version of the image. I had much more success in using Python for this assignment, and I believe this assignment is a vast improvement over the previous.

Since this assignment does not require the use of statistical results, this section will focus on the experimentation and implementation. The program initially prompts the user for a file name. The program then opens the image in whichever format the image currently exists in as well as a Grayscale image no matter what. If no image data is present, the program will print an error message, and this loop will continue until the user provides a correct image file name located in the program's path. The Sobel filters Gx and Gy are initialized, and the programs will use an if statement to determine if the original input image is in RGB Color or Grayscale. The following process will be nearly identical from here, except for the RGB Color image will additionally loop through the number of channels.

The rows, columns, and potentially channels of the image are obtained, and variables imgIx, imgIy, imgEx, imgEy, and magnitude are initialized to zero. imgIx and imgIy will represent the image following the convolution with the related filter, imgEx and imgEy are simply imgIx and imgIy scaled using the given formula to display to the user, and magnitude will represent the magnitude. The image is then looped through using rows, columns, and potentially channels, and convolution and scaling are applied using the formulas presented in

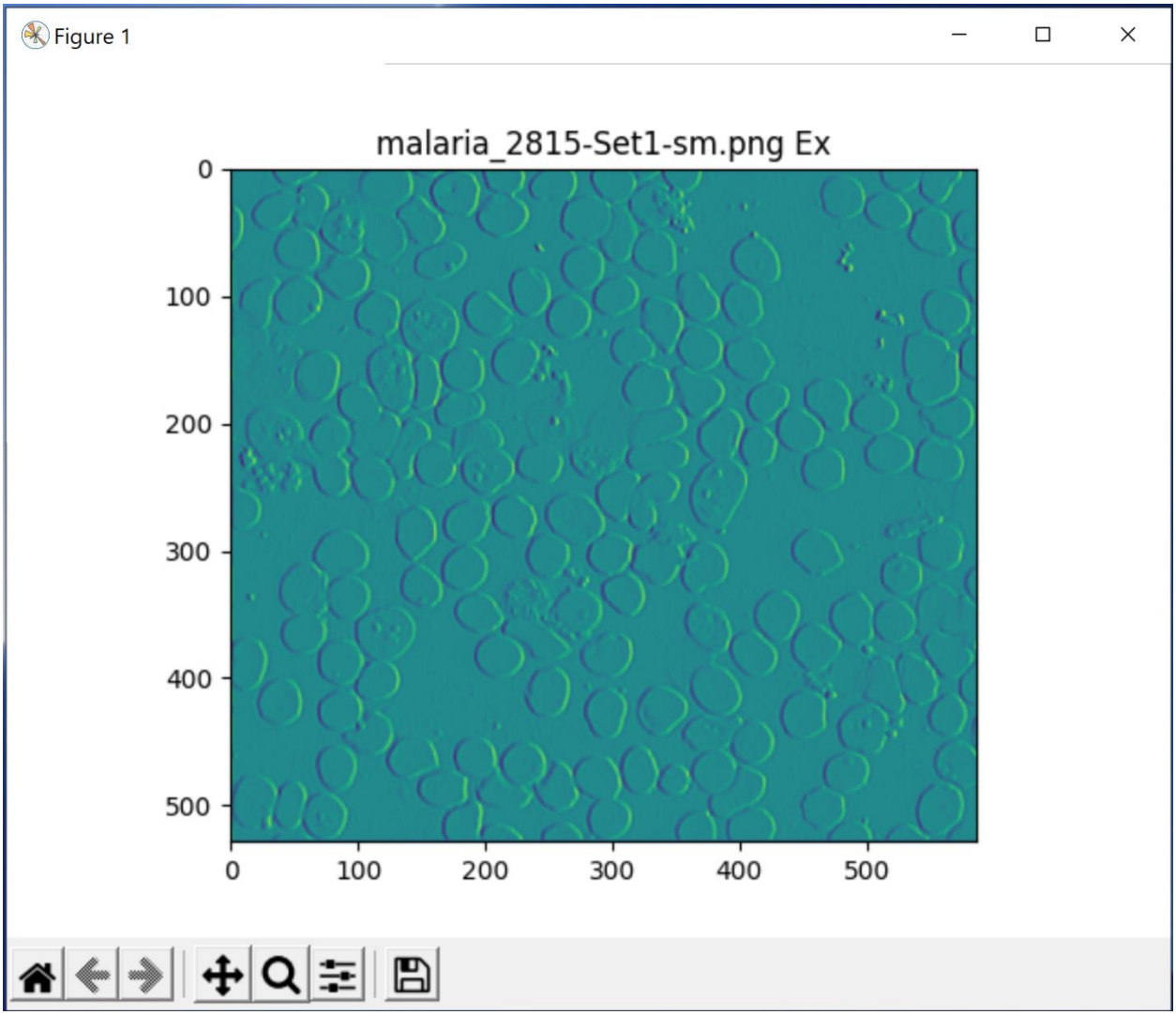
class. The magnitude is calculated using the formula provided (the square root of the sum of Ix squared and Iy squared), and, for color images, each channel is condensed together to form an output RGB image. Using plt functions, the image's Ex, Ey, and magnitudes are displayed to the user.

The process for calculating the image values Ix and Iy are repeated using the Grayscale version of the image if the image was initially RGB color. This is due to the fact that orientation only needs to be applied to Grayscale images, and the values are set to new variables grayImgIx and grayImgIy. An angle variable will be created to store the orientations at each pixel, the Grayscale image is once more looped through, and the orientation is calculated by using the inverse tangent equation given in class, along with a line to properly convert the output orientation from radians to degrees. If both grayImgIx and grayImgIy are 0, the angle is set to 0 since this would cause an error. Finally, the orientation is displayed to the user using "jet" as the cmap to simulate color LUT using HSV.

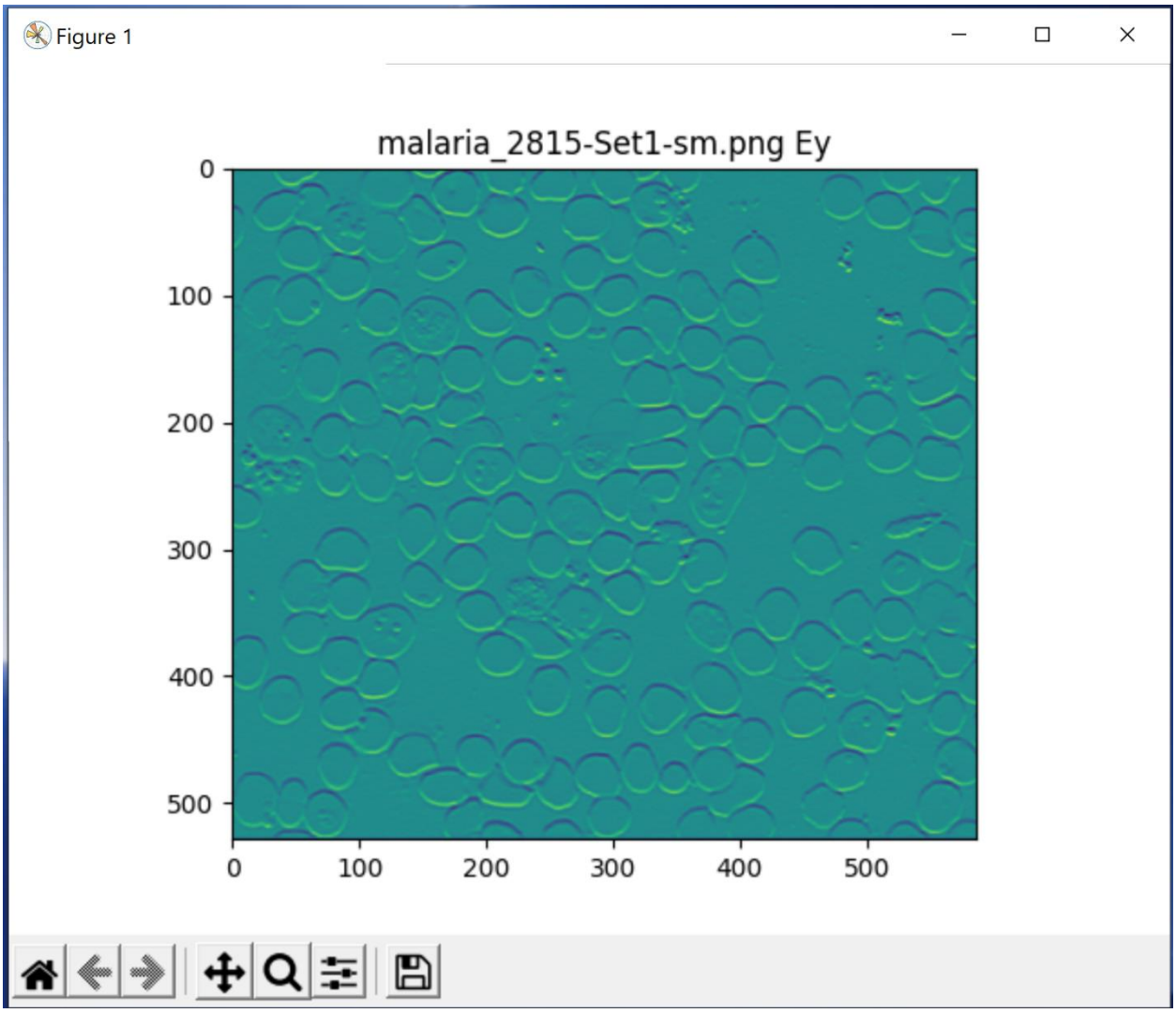
I felt that this assignment went much smoother than the previous assignment. I learned more about opencv and numpy in Python, which helped a lot. Logically, the assignments have all made sense. However, when it comes to implementation, it is not the easiest when unfamiliar with the technology at hand, and I felt much more comfortable using Python on this assignment. I am very confident in my RGB color convolutions, my magnitudes, and my orientations. I am unsure about the Grayscale convolutions; however, I believe it should be working correctly given that I simply altered the RGB color convolutions code. I would also like to note that for the Grayscale image I used, the image was technically an RGB color image with only gray values, so I converted the input image to Grayscale. If there is an issue, perhaps this could be a cause. However, I can say that I thought this assignment looked difficult, but after some time of

implementing, I found this assignment almost enjoyable. In order to demonstrate my work, please find below sample images of the program running.

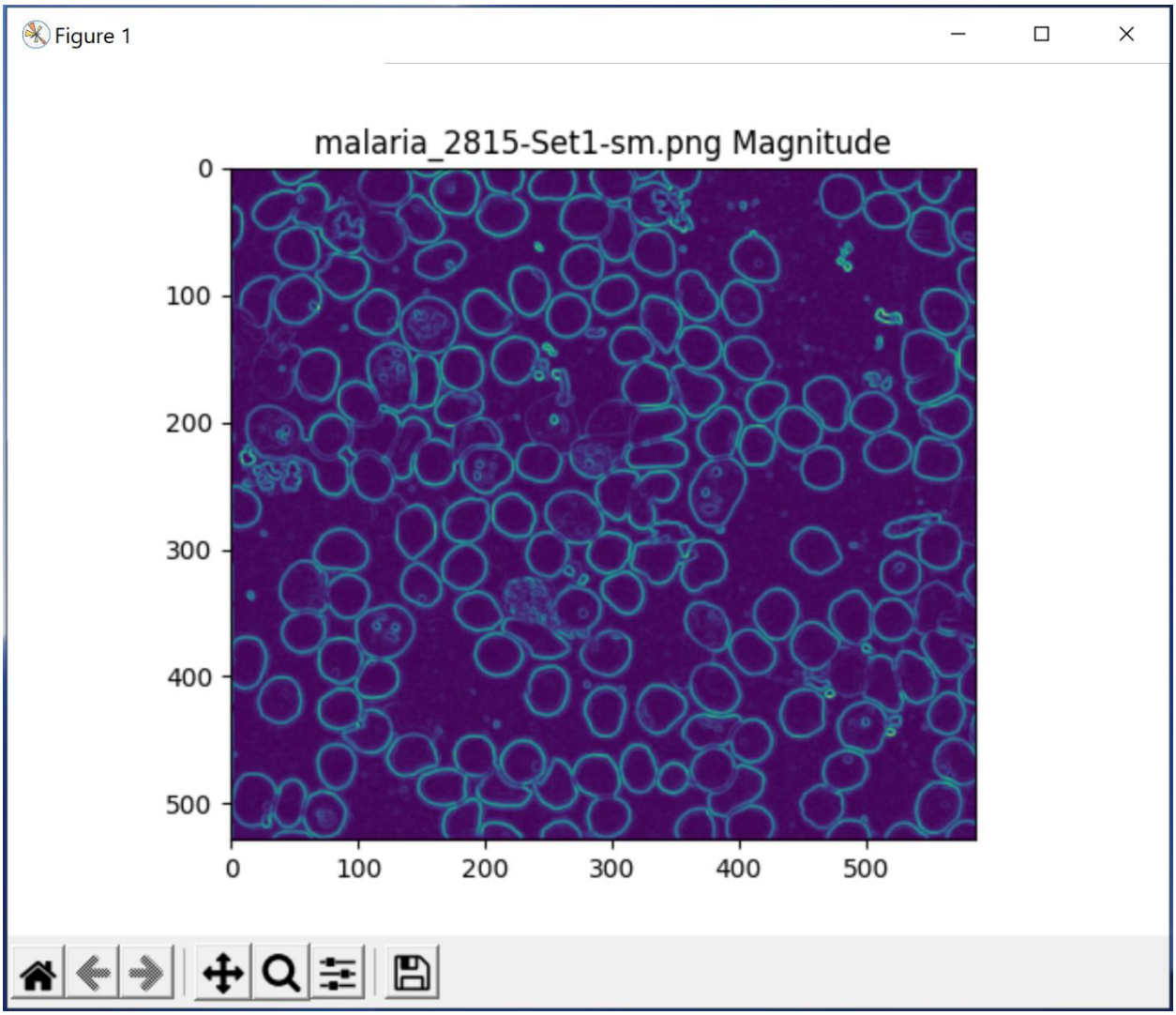
Scaled Ex Sobel Convolution of RGB Image:



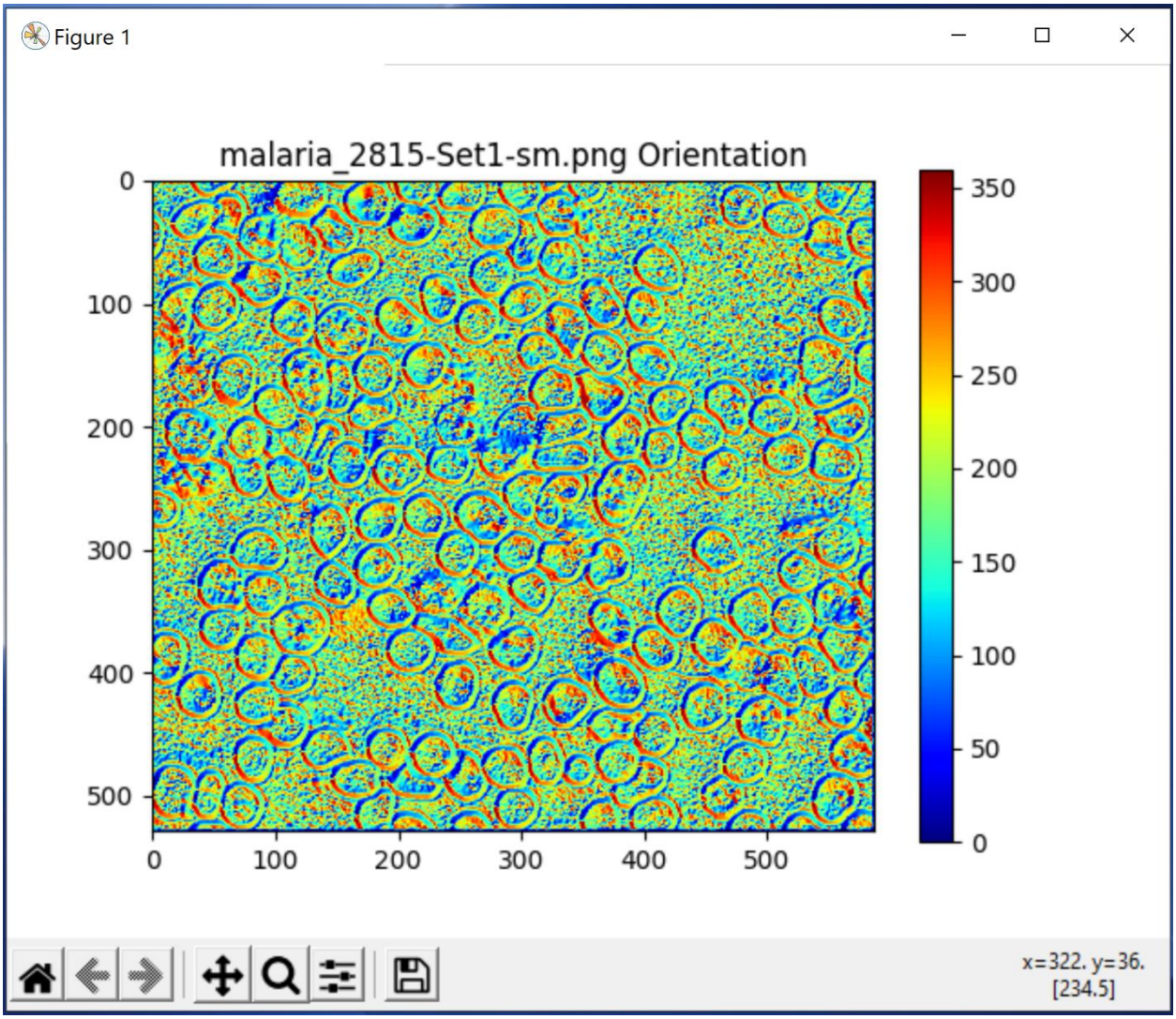
Scaled Ey Sobel Convolution of RGB Image:



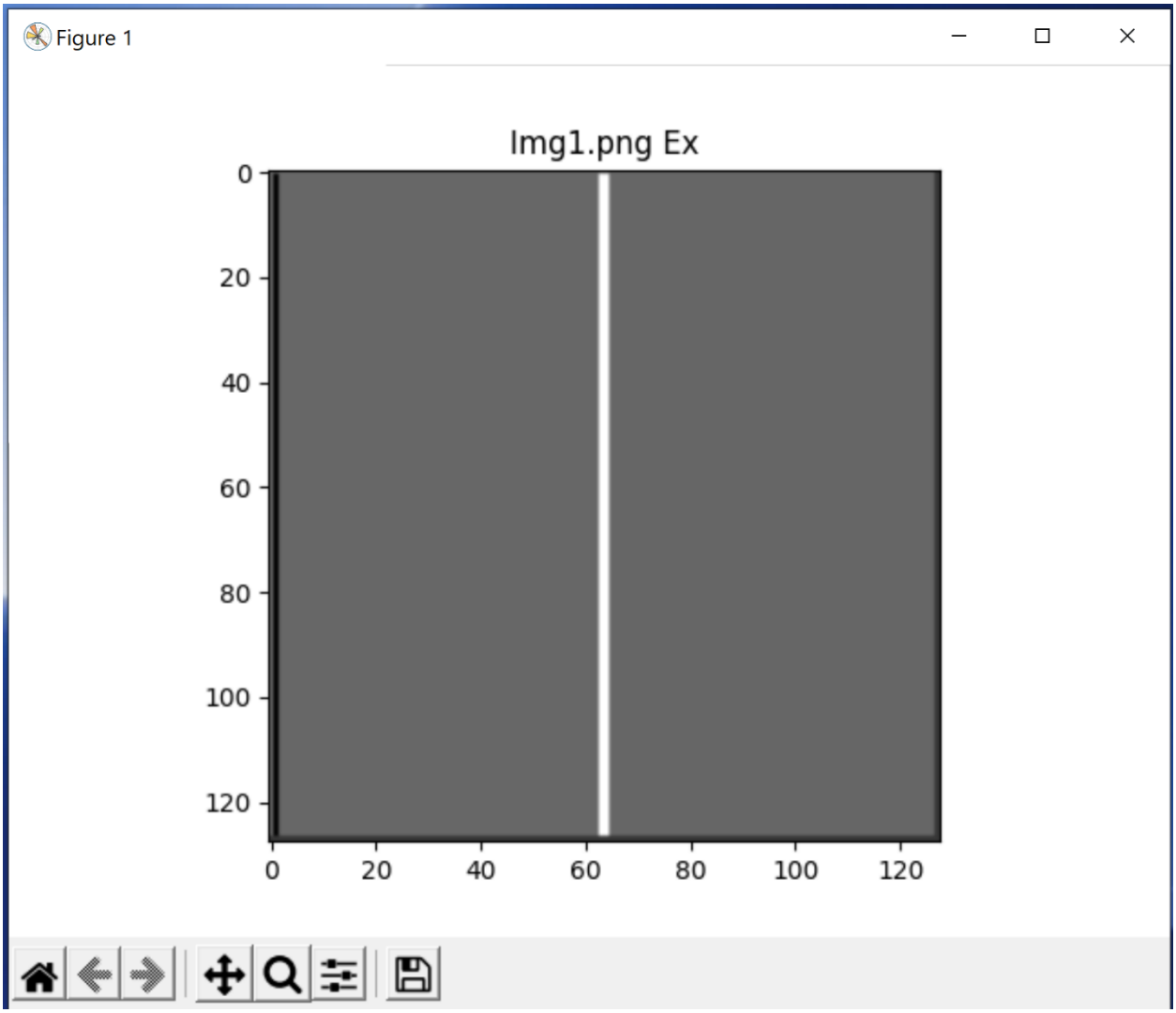
Magnitude of RGB Image:



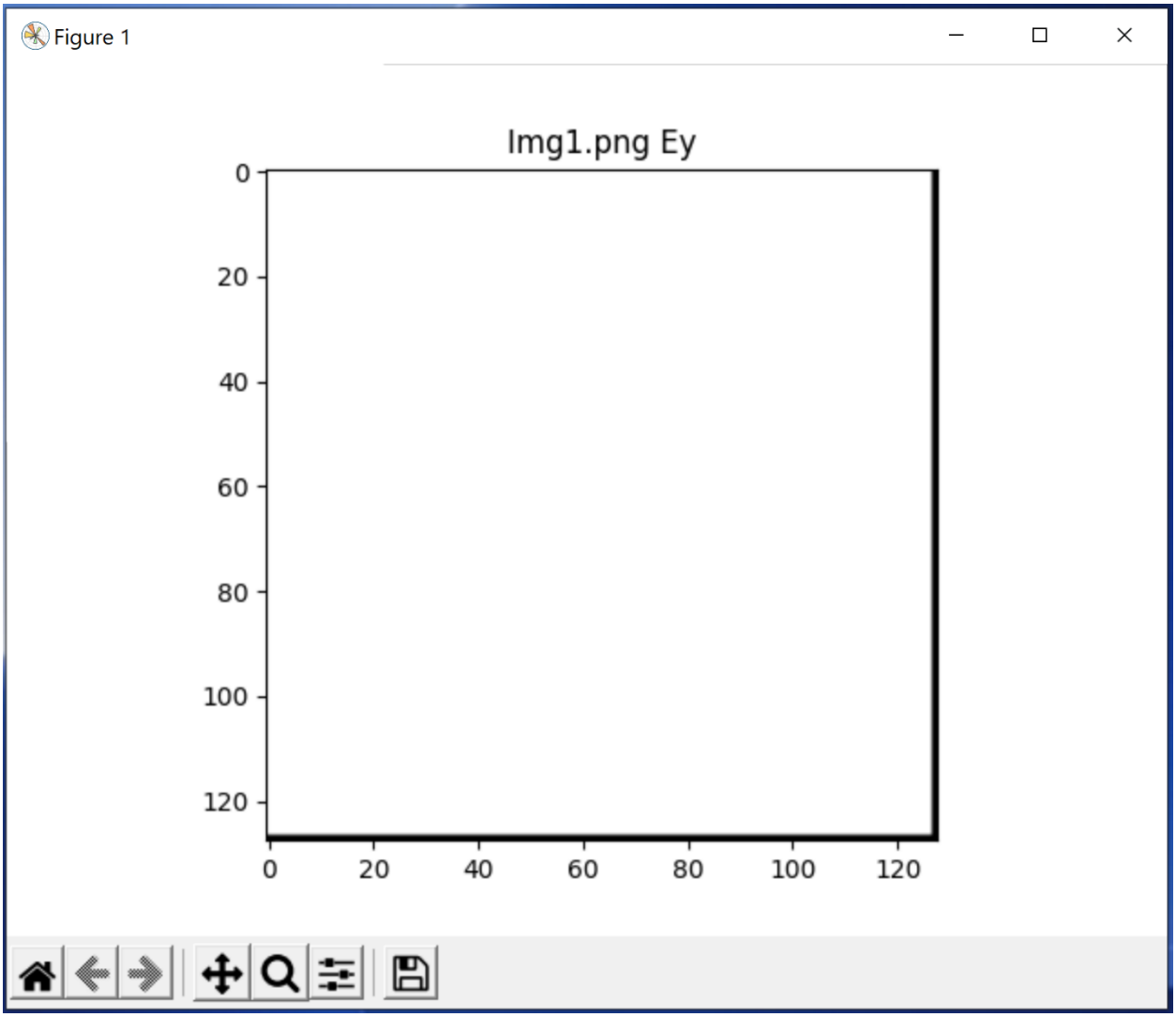
Orientation of the Grayscale of the RGB Image:



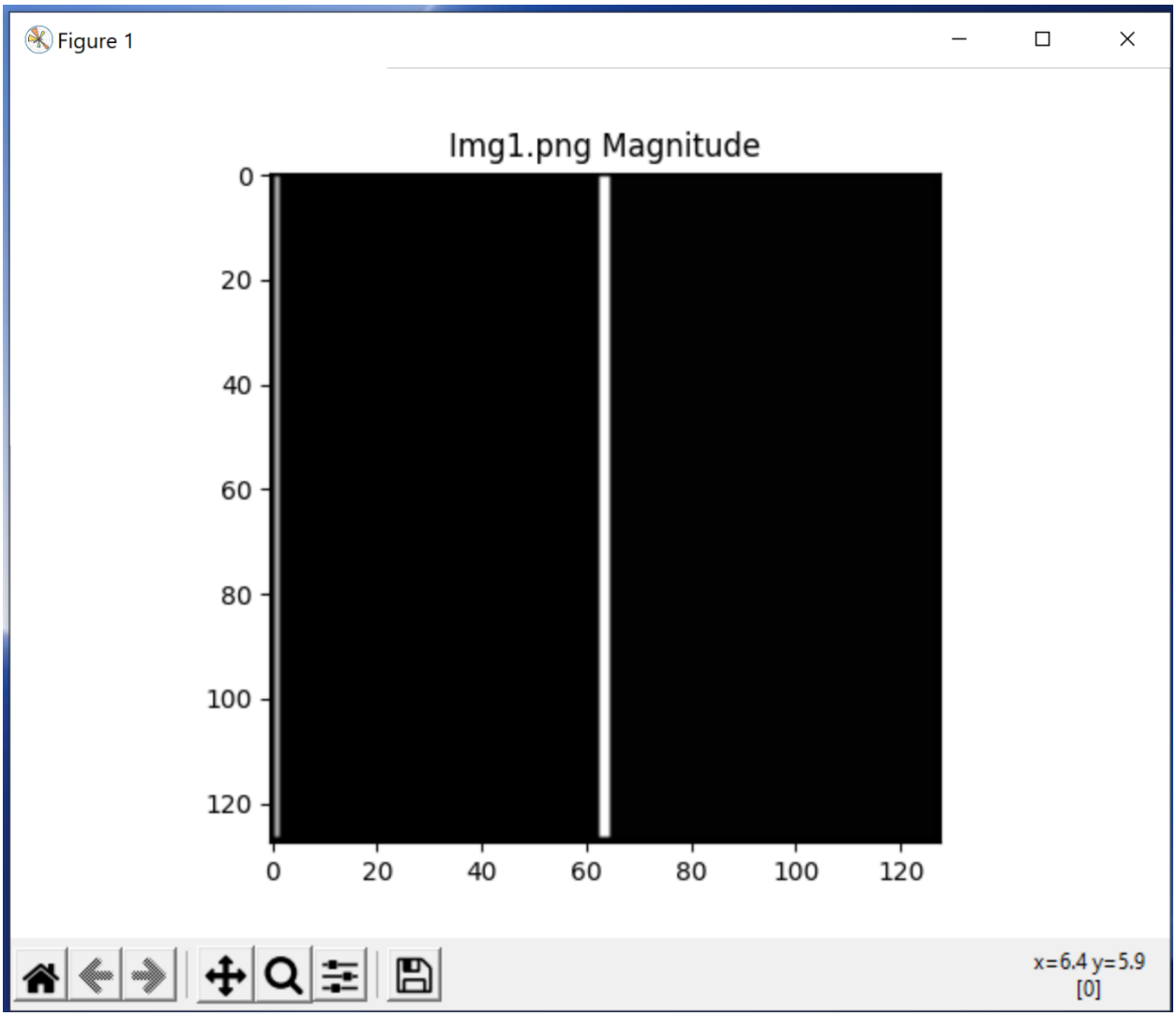
Scaled Ex Sobel Convolution of Grayscale Image:



Scaled Ey Sobel Convolution of Grayscale Image:



Magnitude of Grayscale Image:



Orientation of Grayscale Image:

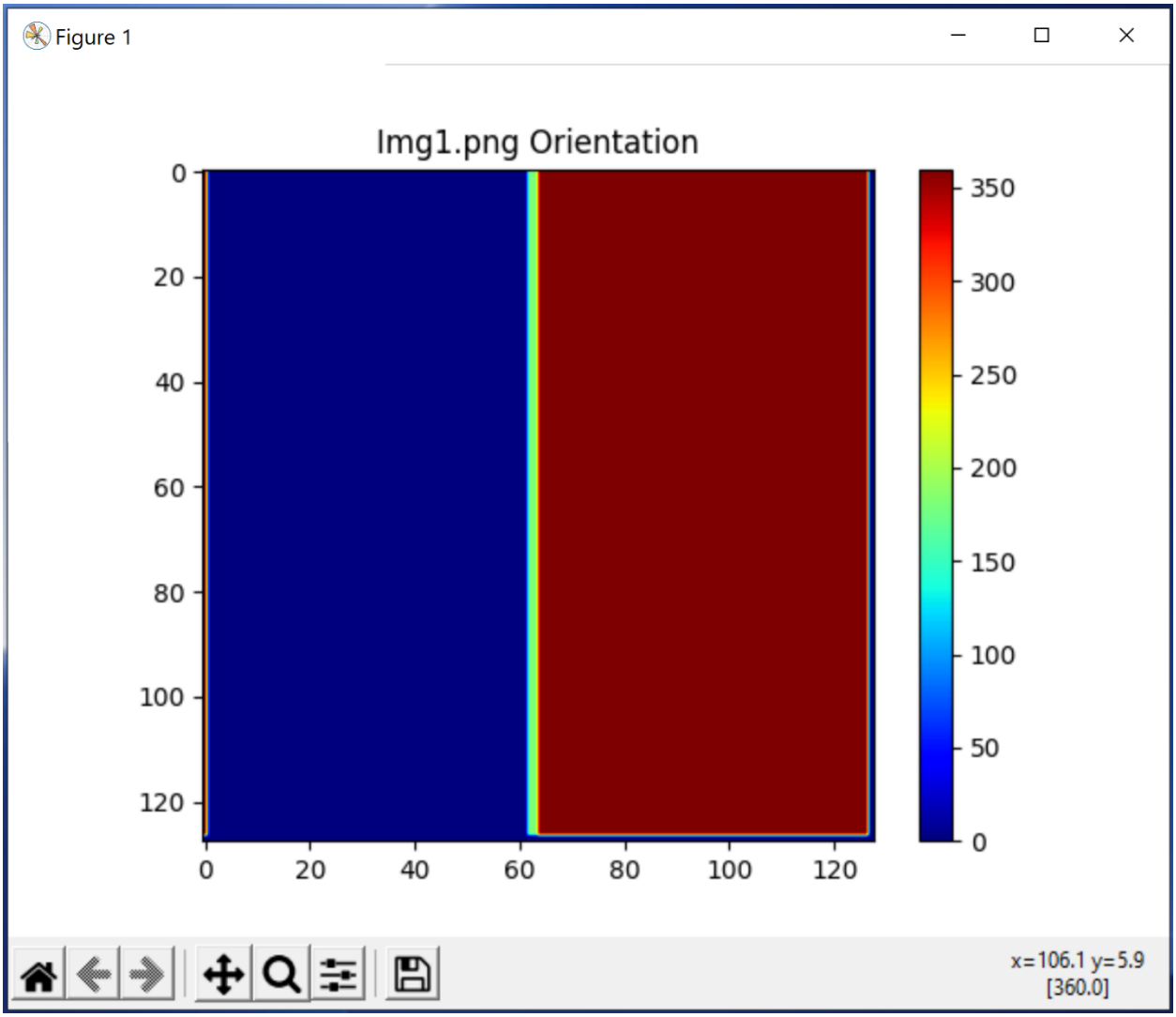


Figure 1
Img1.png Orientation
0
20
40
60
80
100
120
350
300
250
200
150
100
50
0
x=106.1 y=5.9
[360.0]