HW 3 Lab Report

Jacob Alongi

Part 1:  **Filtered Gradient**

* Load an image
* Convolve the image with a Gaussian



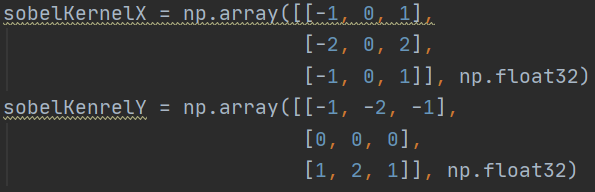
I did do this in the previous assignment calling a singular method is easier

(3,3) sets the kernel size, 0 is sigma

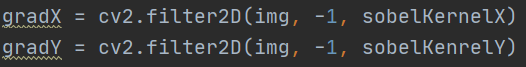


* Find the x and y components of the gradient Fx and Fy at each point

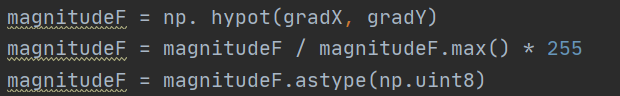
Using 2 Sobel kernels or 2 Prewitt kernels filter the image in the X and Y direction



This will give us the gradient for each direction



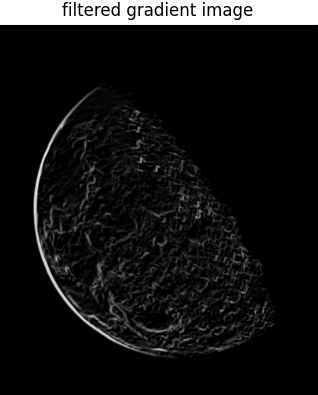
Taking the hypotenuse of the corresponding X and Y values will give us the magnitude



* Compute the edge strength F (the magnitude of the gradient) and edge orientation D = arctan(Fy/Fx) at each pixel

Using the Gradients computing the arctan of the magnitude will give us the orientation





Part 2: **Nonmaximum suppression:**

* Create a "thinned edge image" I(x,y) as follows:

First change the values to radians

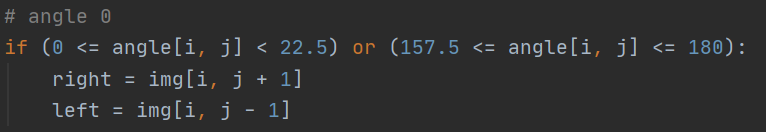
Make sure we don’t get negative



1. For each pixel find the direction D\* in (0, 45, 90, 135) that is closest to the orientation D at that pixel.

Even though there are 8 adjacent pixels, since we are getting the left and right pixel relative to the direction, we only need to take the first 4 quadrants as we will get the last 4 anyway.

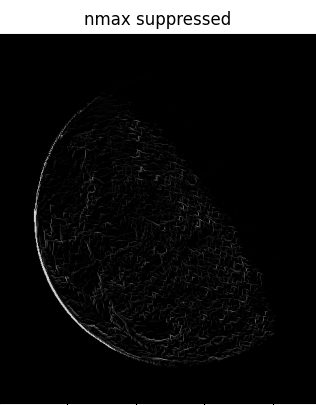
Since the Quadrants between 0 and 180 are 0, 45, 90, 135 we want the halfway angles between each angle to chose which direction to align



1. If the edge strength F(x,y) is smaller than at least one of its neighbors along D\*, set I(x,y) = 0, else set I(x,y) = F(x,y)

Above gets the left and right pixels relative to the 0 degree direction, after we check if either is stronger than the pixel, if they are than we set the pixel value to 0





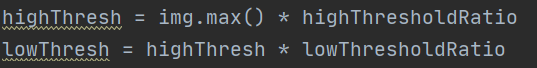
Part 3: **Hysteresis thresholding:**

* Repeatedly do the following:

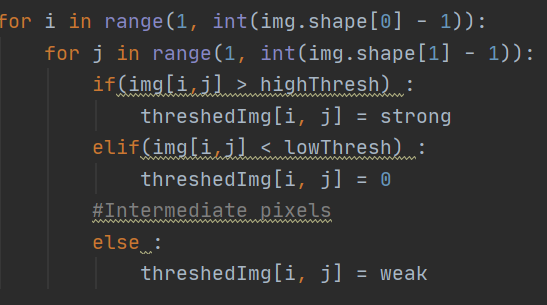
1. Locate the next unvisited pixel (x,y) such that I(x,y) > T\_h
2. Starting from (x,y), follow the chain of connected local maxima, in both directions, as long as I(x,y) > T\_l
3. Mark each pixel as it is visited

I actually Did not find good examples of implementing this way so Instead I broke it into 2 parts

* Set pixels to 3 values based on the thresholds calculated
  1. Threshold calculation



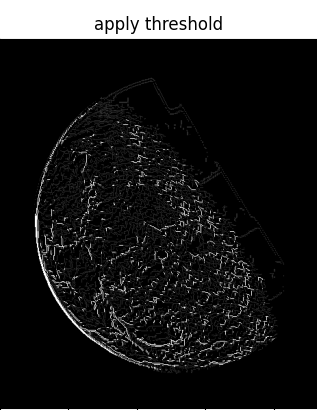
* 1. Set pixels to strong, weak, or 0

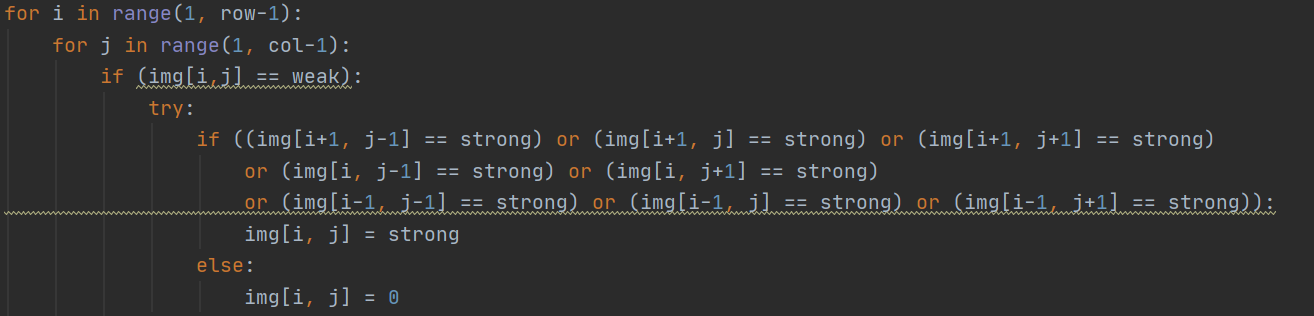


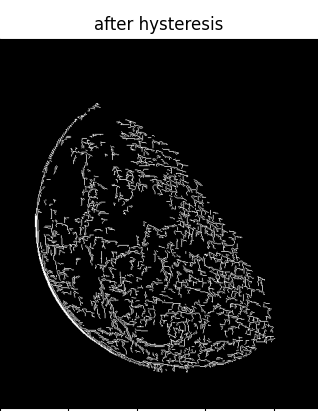
My values for strong turned the insanity value to 255

Weak to 45

Everything below low Threshold to 0

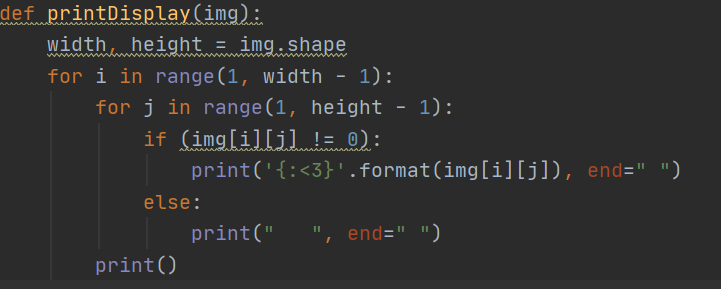


* Visit each pixel, check if it’s a weak pixel, if weak pixel has a strong neighbor its connected and set to strong, otherwise set it to 0
* 

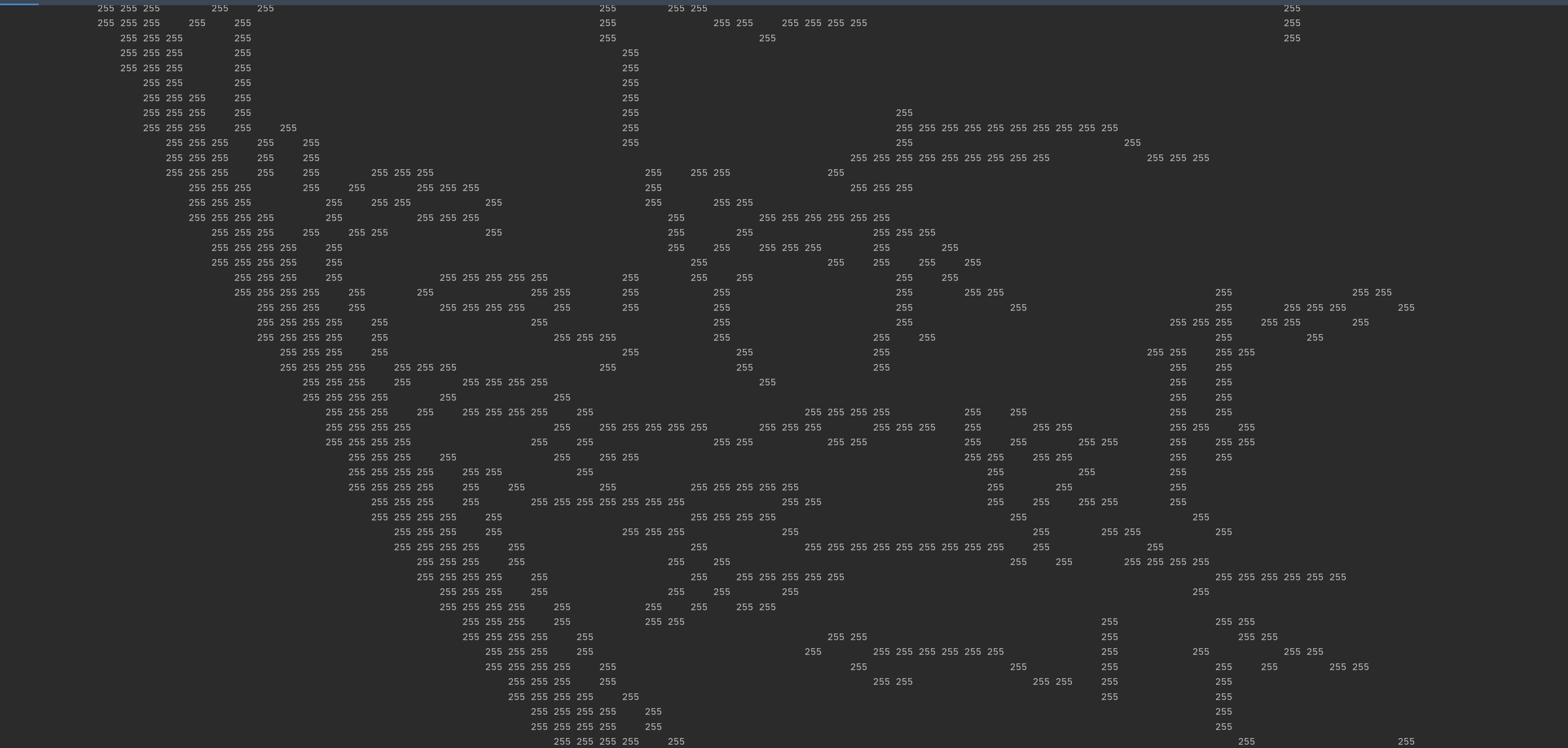


**Issues:**

Understanding what was going on at each step was the biggest issue, there were plenty of bad examples on the internet. The book is awful and 2 weeks worth of class was put into 1 video with very minimal explanation on how to do things. I had to create a printDisplay to show each pixel in its location after each step to make sure the values were corresponding correctly to what I was supposed to do.



Displayed the pixels as this

  
as you can see the pixels values were left justified so that they would show up in their corresponding spot. This worked for every step allowing me to see the values, I had to initially do a lot of testing with a smaller image so that I could see the results on the screen

**Code:**

