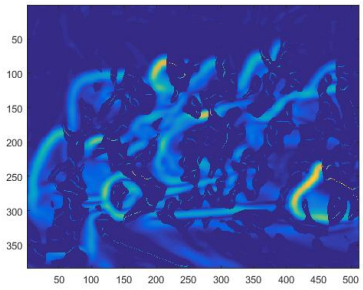
HW07 Niyati Shah – Canny Edge Detector

To get a good output with the Canny edge detector algorithm, it is a good idea to smoothen the image out by using a filter like Gaussian so that we any unwanted noise is removed from the given image.

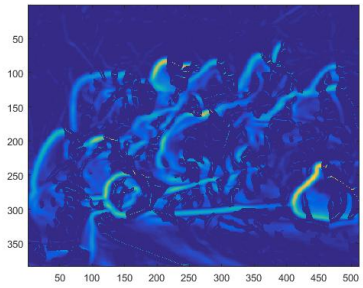
As part of the assignment, I have used the peppers.png image to work the Canny edge detector. I observed how different filter sizes and sigma values affected the edge detection.

Original image:

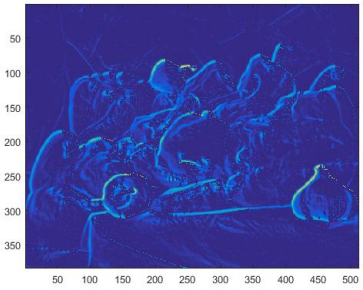
Test 1: Gaussian filter size: 40



sigma: 5

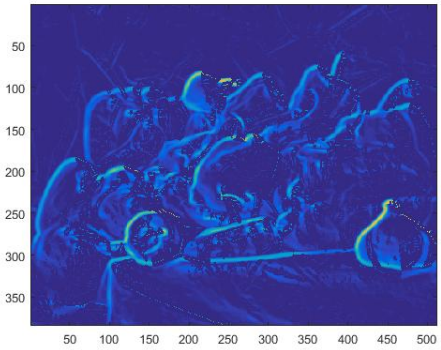


sigma: 3

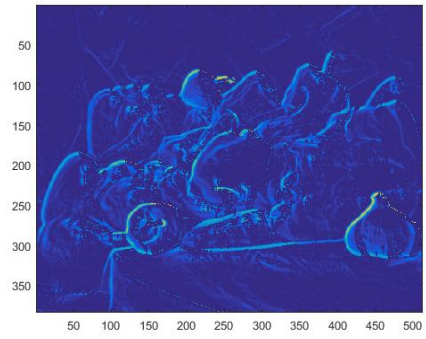


sigma: 1

Test 2: Gaussian filter size: 5

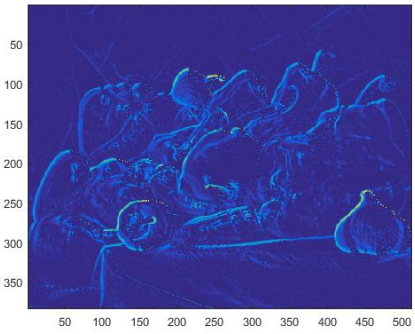


sigma: 5

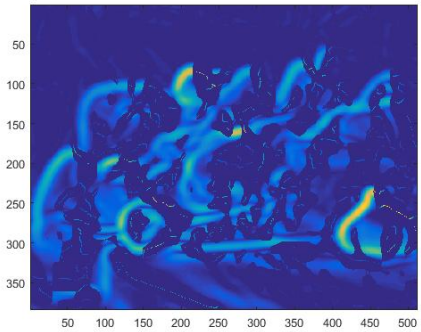


sigma: 1

Test 3: Gaussian filter size: 30



sigma: 1



sigma: 5

As we can see from the above tests where I have applied Gaussian filter of different size and a different sigma. Whenever we had a high sigma value, we got thicker edges and blurry/smudgy image. Lower the sigma value more precise edges were found after applying the canny edge detector minus the hysteresis.

Learning:
It is seen that Matlab takes all negative angles as angles after 180 degrees. So angles from 0 to -180 are also equivalent and automatically taken as from 180 to 360. So I have directly taken angles after 180 degrees and not converted them to negative as Matlab does that.

As the values were too small after performing the non maximal suppression I changed the image from rgb to grey just for the writeup to show contrast and show what exactly happens with the image.

The actual rgb images with the non maximal suppression are shown below.

PART 2

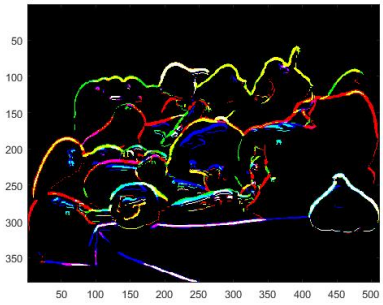
To apply the hysteresis, the following conditions regarding the pixel magnitude, need to be checked.
Case 1: E(x,y) < low threshold, the pixel is rejected (pixel =0)
Case 2: If E(x,y) > high threshold, the pixel is accepted (pixel =1)
Case 3: If low threshold < E(x,y) < high threshold then check the neighboring pixel to see if is an accepted pixel or not. If yes then the pixel is accepted or else rejected.

To apply the code I have taken the third condition as value 2, so once the initial acceptance and rejection is done we can check if the accepted pixel is part of a continuous edge or not. This is not done in the earlier loop because we check all neighboring 8 pixels which means 4 pixels from these are still not determined if they are accepted or rejected and this can lead to pixels not part of a proper edge to show up.

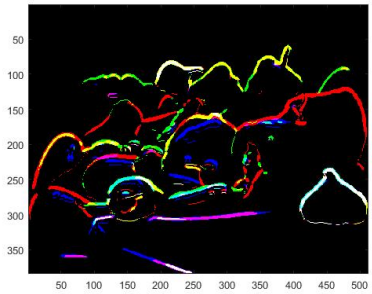
The Guassian filter also effects the hysteresis part. With a high sigma we get less edges as the image has become to blurred to find a good continuous edge while a lower value of sigma made the edges seem perfect. Also a lot of edges which has a thinner edge and magnitude were lost as seen in test 1 sigma 5, test 2 sigma 5 images below.

Earlier part had the true magnitude of edges as part of the edge detection. This means that after performing the operations, a lot of value were removed if they did not meet the condition. Many edges had a discontinuous line and a lot of lined that were not actual edge were detected.
After applying the hysteresis part, the edges so formed contained only the true continuous edge with the exception of a few outliers. The lines formed are also continuous. The resulting image also has a thinner edge as compared to the the earlier part.

Test 1: Gaussian filter size: 5 :

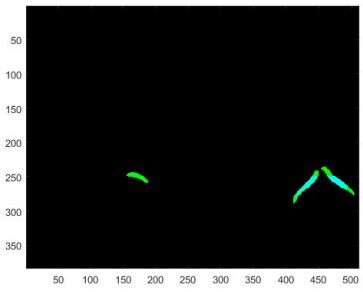


sigma: 1



sigma: 5

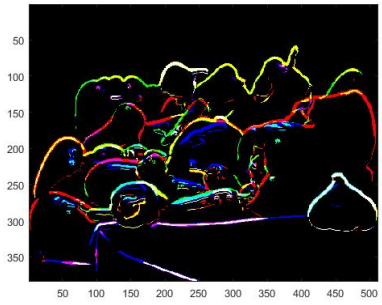
Test 2: Gaussian filter size: 40 :



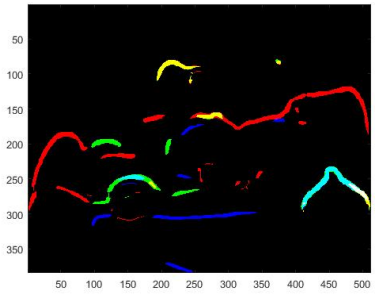
sigma: 5

sigma: 1

Test 2: Gaussian filter size: 30 :



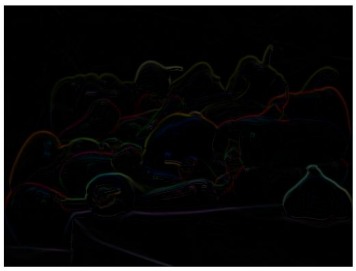
sigma: 1

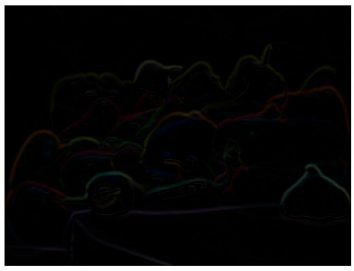


sigma: 5

Part 1 image in full rgb

Test 1: Gaussian filter size: 5


sigma: 1

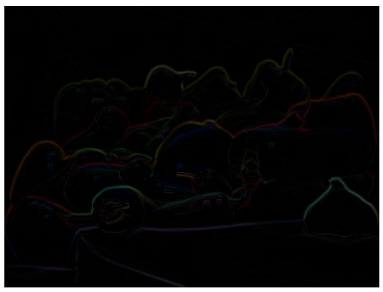

sigma: 5

Test 2: Gaussian filter size: 40 :


sigma: 1


sigma: 5

Test 3: Gaussian filter size: 30 :


sigma: 1


sigma: 5