Crowd Sourcing

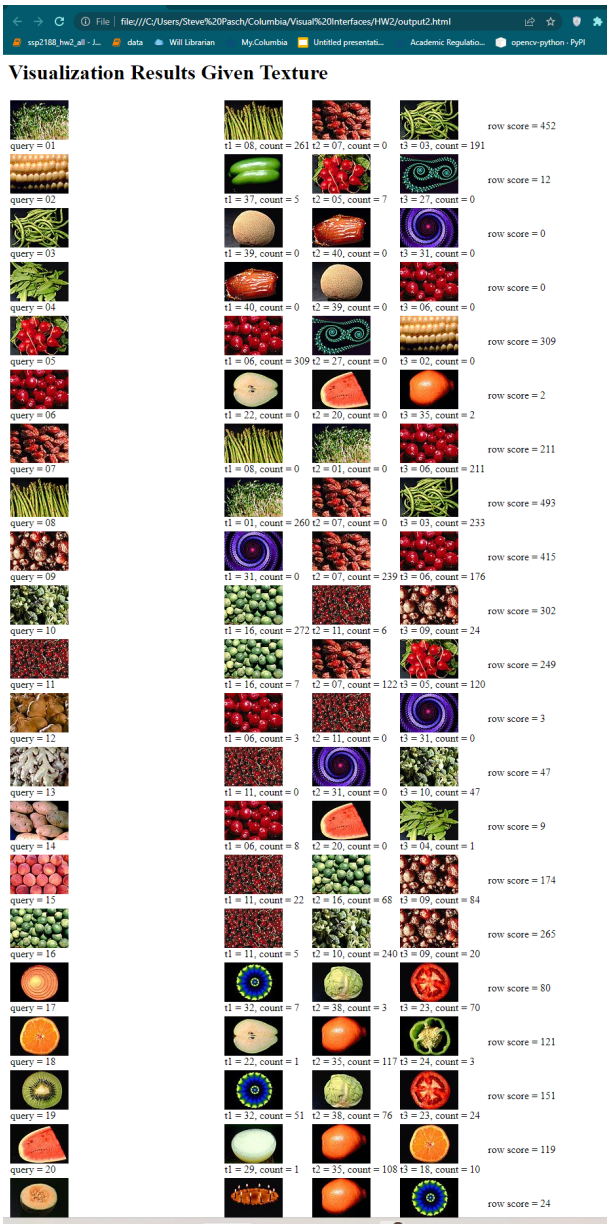
I included screenshots of my results, but if you run the program, it will produce 5 .html files

Step 1:

To find a good three-dimensional histogram bin definition, I took what the professor said to heart about blue being less distinguishable as opposed to color like green, and red being somewhere in between. So I chose a ratio of 6 bins, 8 bins, and 2 bins for the r,g,b color channels, respectively. I saw good results with a high green and a low blue. Red, I tweaked it until I got a higher overall score.

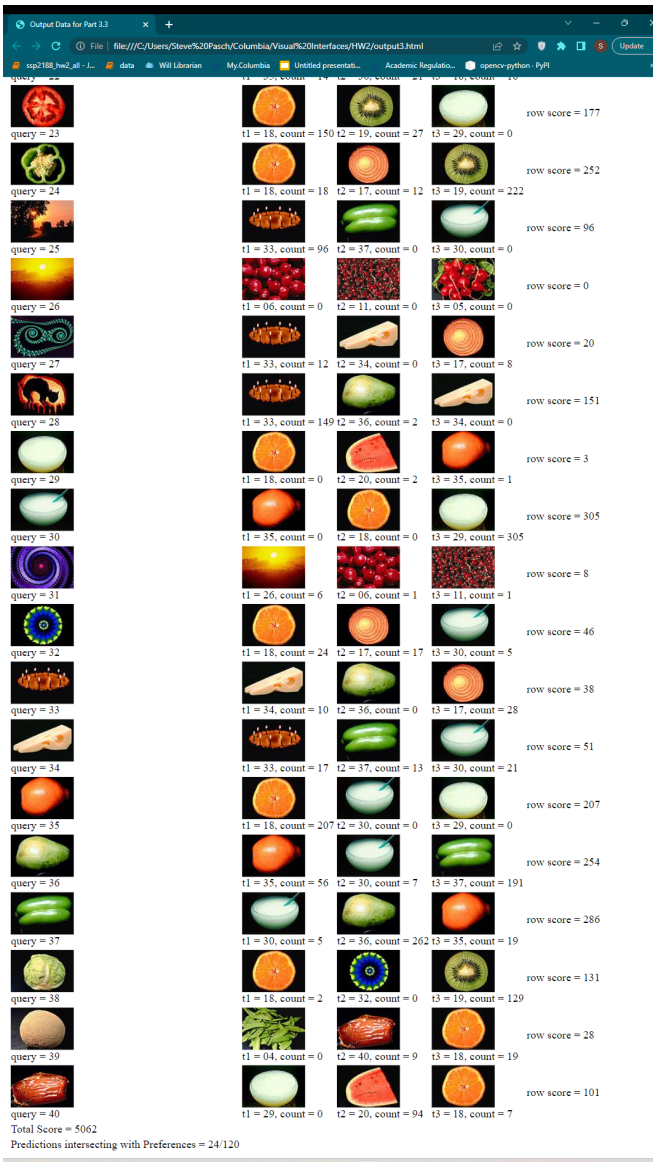
Step 2

When creating the one-dimensional histograms, I found that I could use the open cv normalize() function on each image before using the same L1 distance function from part 1 to get distance values between 0-1.



Step 3

I chose an m value of m = 20. Using matplotlib.pyplot's imshow(), I viewed the different samples of the binary images at various m values. Values higher than 20 did not give me enough of a good border to clearly disambiguate the shape. Values less began to have a fuzzy noise of binary pixels outside the border. I attribute this to light reflecting off the objects and onto the black background. This ambient light began being picked up between 10-20. Below 10, even the black would register. m=20 allowed me to have a good border without much noise. Black was, therefore m < 20.

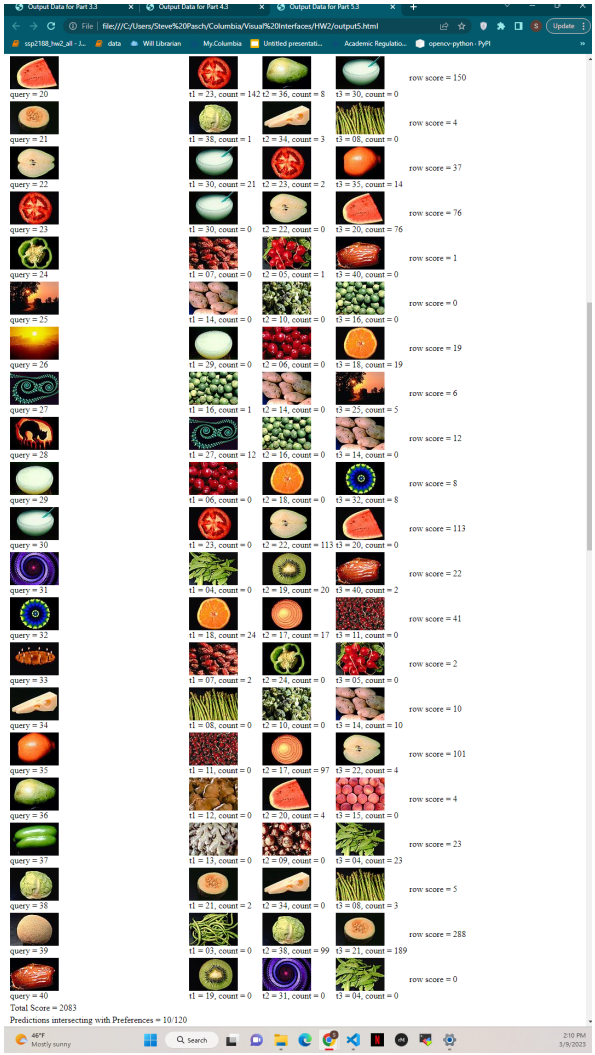


Step 4

When investigating symmetry, I faced the issue of gaps within my objects. For example, the dark rings of the onion registered as “black.” Which before did not bother me because I was focusing on the border. But now that I was matching pixels, I wanted my objects to be completely in the foreground. Thus, I lowered the threshold value to m = 10. This produced some ambient noise around the objects but brought the full objects into the foreground. I was concerned less about the noise in this case because, comparatively, I gained much more symmetrical counts within the object than I lost to symmetrical background on the outside of the object. This choice to lower the threshold resulted in a better total score. My method of finding the symmetry was to XNOR twin bits and record all resulting 1’s as a count.

Step 5

Here I took a ratio of success in the previous trials and used that to come up with a simplex vector of []. My results were a little confusing to the eye. There is a bug in there where i40 is simulated extra times. My plan if I made better use of my time was to find the top 3 of each query in the Crowd.txt data. Then run each parameter over the results and rank the first pick as 5pts to the best parameter that describes it, the second pick as 3pts, and the third as 1pt. Then use the ratio of earned points between each parameter to find a good simplex. In theory, it would perform very similarly to the class average.



Step 6

Did not complete step 6

Sources:
Histogram Making:
https://docs.opencv.org/3.4/d6/dc7/group__imgproc__hist.html#ga4b2b5fd75503ff9e6844cc4dcdaed35d

For writing the output to an html file for viewing.
https://www.geeksforgeeks.org/how-to-write-to-an-html-file-in-python/

How to create A binary image from greyscale:
https://www.tutorialspoint.com/opencv-python-how-to-convert-a-colored-image-to-a-binary-image