

Crowd Sourcing

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

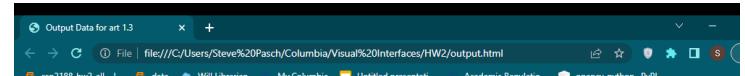
Step 1:

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.



Visualization Results Given Color

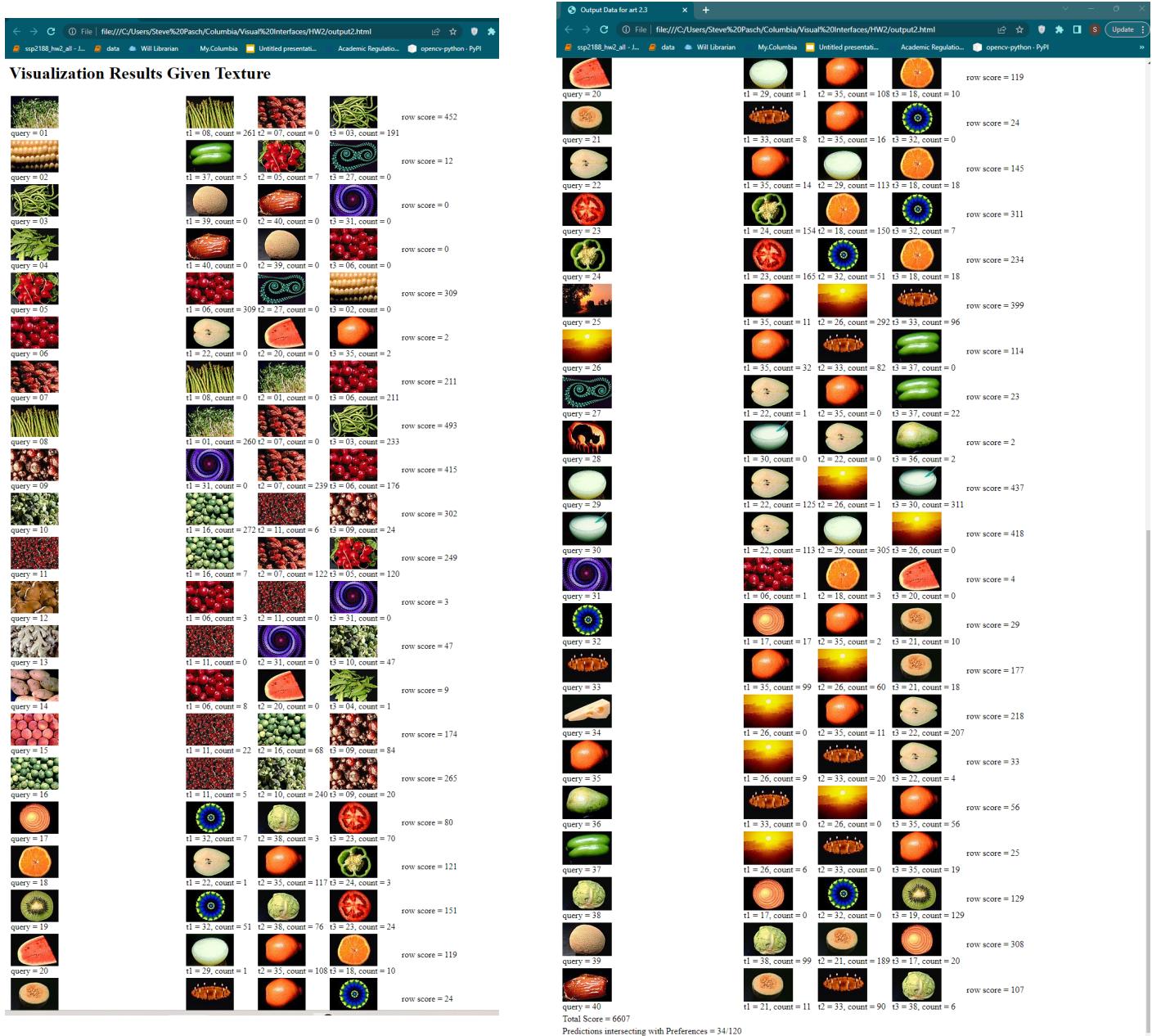
				row score = 540
query = 01	t1 = 10, count = 88	t2 = 03, count = 191	t3 = 08, count = 261	
				row score = 81
query = 02	t1 = 39, count = 30	t2 = 09, count = 21	t3 = 21, count = 30	
				row score = 388
query = 03	t1 = 04, count = 272	t2 = 36, count = 1	t3 = 01, count = 115	
				row score = 425
query = 04	t1 = 03, count = 295	t2 = 08, count = 121	t3 = 36, count = 9	
				row score = 376
query = 05	t1 = 06, count = 309	t2 = 23, count = 0	t3 = 11, count = 67	
				row score = 473
query = 06	t1 = 11, count = 72	t2 = 05, count = 259	t3 = 07, count = 142	
				row score = 284
query = 07	t1 = 09, count = 178	t2 = 40, count = 48	t3 = 11, count = 58	
				row score = 493
query = 08	t1 = 03, count = 233	t2 = 01, count = 260	t3 = 19, count = 0	
				row score = 300
query = 09	t1 = 07, count = 239	t2 = 11, count = 61	t3 = 02, count = 0	
				row score = 441
query = 10	t1 = 01, count = 136	t2 = 03, count = 33	t3 = 16, count = 272	
				row score = 451
query = 11	t1 = 09, count = 98	t2 = 07, count = 122	t3 = 06, count = 231	
				row score = 105
query = 12	t1 = 02, count = 30	t2 = 09, count = 75	t3 = 11, count = 0	
				row score = 251
query = 13	t1 = 10, count = 47	t2 = 01, count = 3	t3 = 14, count = 201	
				row score = 99
query = 14	t1 = 39, count = 30	t2 = 02, count = 13	t3 = 09, count = 56	
				row score = 72
query = 15	t1 = 40, count = 0	t2 = 07, count = 72	t3 = 35, count = 0	
				row score = 322
query = 16	t1 = 10, count = 240	t2 = 01, count = 39	t3 = 04, count = 43	
				row score = 380
query = 17	t1 = 25, count = 8	t2 = 18, count = 268	t3 = 21, count = 104	
				row score = 353
query = 18	t1 = 17, count = 236	t2 = 25, count = 0	t3 = 35, count = 117	
				row score = 371
query = 19	t1 = 38, count = 76	t2 = 24, count = 235	t3 = 36, count = 60	
				row score = 260
query = 20	t1 = 35, count = 108	t2 = 18, count = 10	t3 = 23, count = 142	
				row score = 105



			row score = 260
query = 20	t1 = 35, count = 108	t2 = 18, count = 10	t3 = 23, count = 142
			row score = 105
query = 21	t1 = 34, count = 3	t2 = 28, count = 3	t3 = 17, count = 99
			row score = 108
query = 22	t1 = 34, count = 20	t2 = 36, count = 84	t3 = 38, count = 4
			row score = 77
query = 23	t1 = 35, count = 53	t2 = 40, count = 16	t3 = 28, count = 8
			row score = 299
query = 24	t1 = 36, count = 31	t2 = 38, count = 46	t3 = 19, count = 222
			row score = 201
query = 25	t1 = 17, count = 16	t2 = 18, count = 7	t3 = 28, count = 178
			row score = 16
query = 26	t1 = 07, count = 0	t2 = 40, count = 16	t3 = 11, count = 0
			row score = 267
query = 27	t1 = 32, count = 207	t2 = 24, count = 32	t3 = 30, count = 28
			row score = 380
query = 28	t1 = 33, count = 149	t2 = 21, count = 3	t3 = 25, count = 228
			row score = 338
query = 29	t1 = 30, count = 311	t2 = 36, count = 18	t3 = 38, count = 9
			row score = 344
query = 30	t1 = 36, count = 29	t2 = 29, count = 305	t3 = 38, count = 10
			row score = 42
query = 31	t1 = 25, count = 1	t2 = 28, count = 16	t3 = 33, count = 25
			row score = 198
query = 32	t1 = 27, count = 110	t2 = 24, count = 71	t3 = 33, count = 17
			row score = 253
query = 33	t1 = 28, count = 149	t2 = 40, count = 86	t3 = 21, count = 18
			row score = 110
query = 34	t1 = 21, count = 95	t2 = 28, count = 14	t3 = 38, count = 1
			row score = 263
query = 35	t1 = 18, count = 207	t2 = 20, count = 27	t3 = 23, count = 29
			row score = 170
query = 36	t1 = 38, count = 148	t2 = 24, count = 21	t3 = 03, count = 1
			row score = 345
query = 37	t1 = 24, count = 78	t2 = 30, count = 5	t3 = 36, count = 262
			row score = 414
query = 38	t1 = 36, count = 201	t2 = 24, count = 84	t3 = 19, count = 129
			row score = 18
query = 39	t1 = 02, count = 6	t2 = 08, count = 0	t3 = 34, count = 12
			row score = 194
query = 40	t1 = 07, count = 80	t2 = 23, count = 24	t3 = 33, count = 90
	Total Score = 10607		
	Predictions intersecting with Preferences = 40/120		

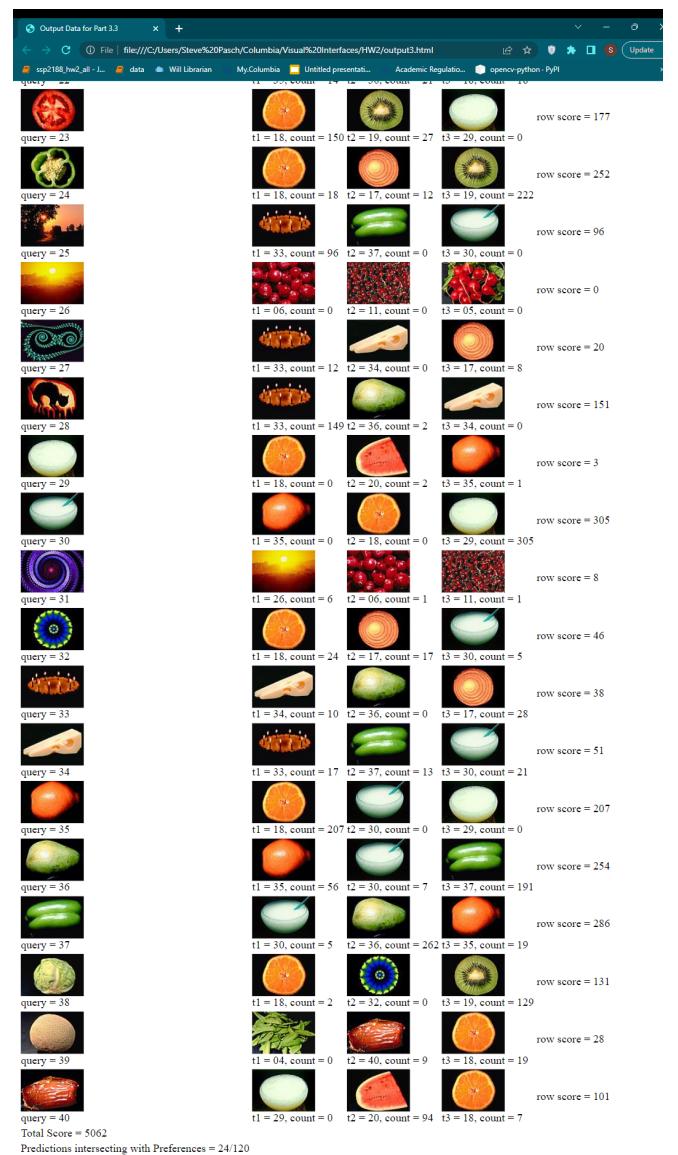
Step 2

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.



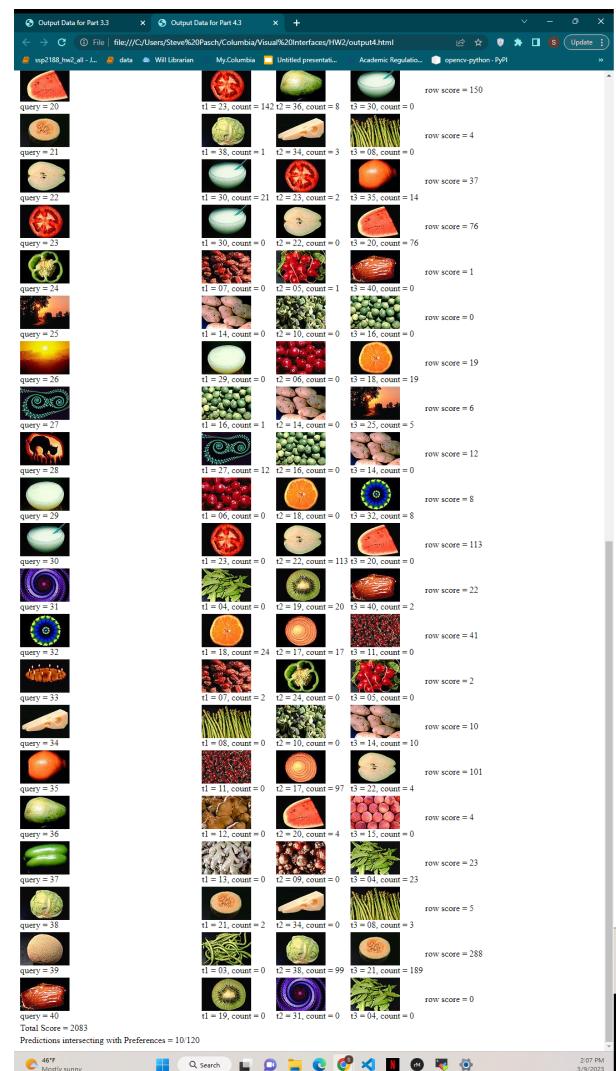
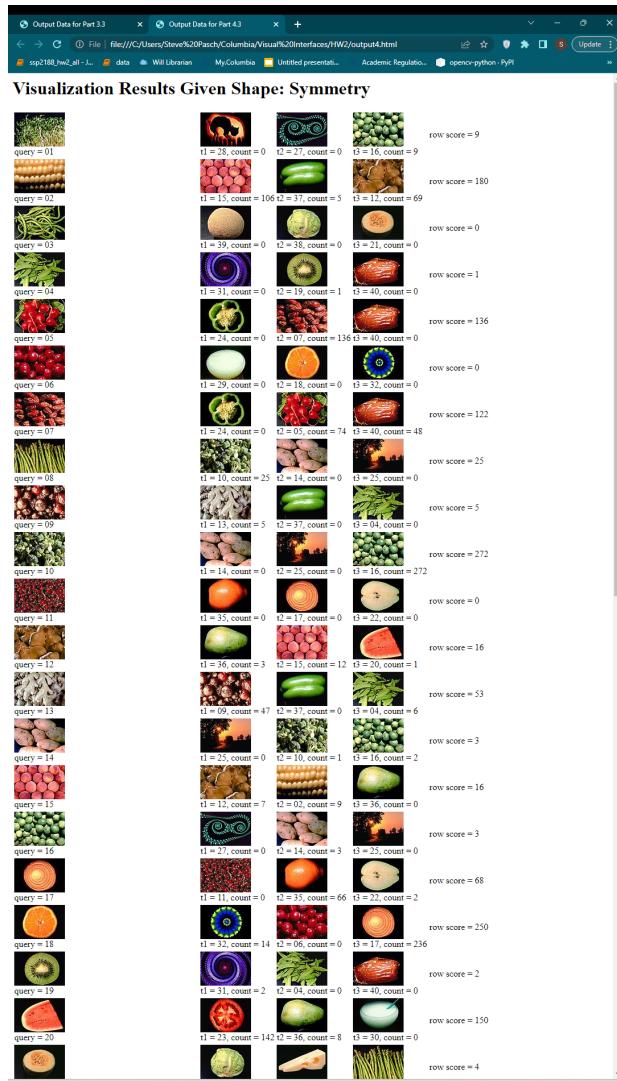
Step 3

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.



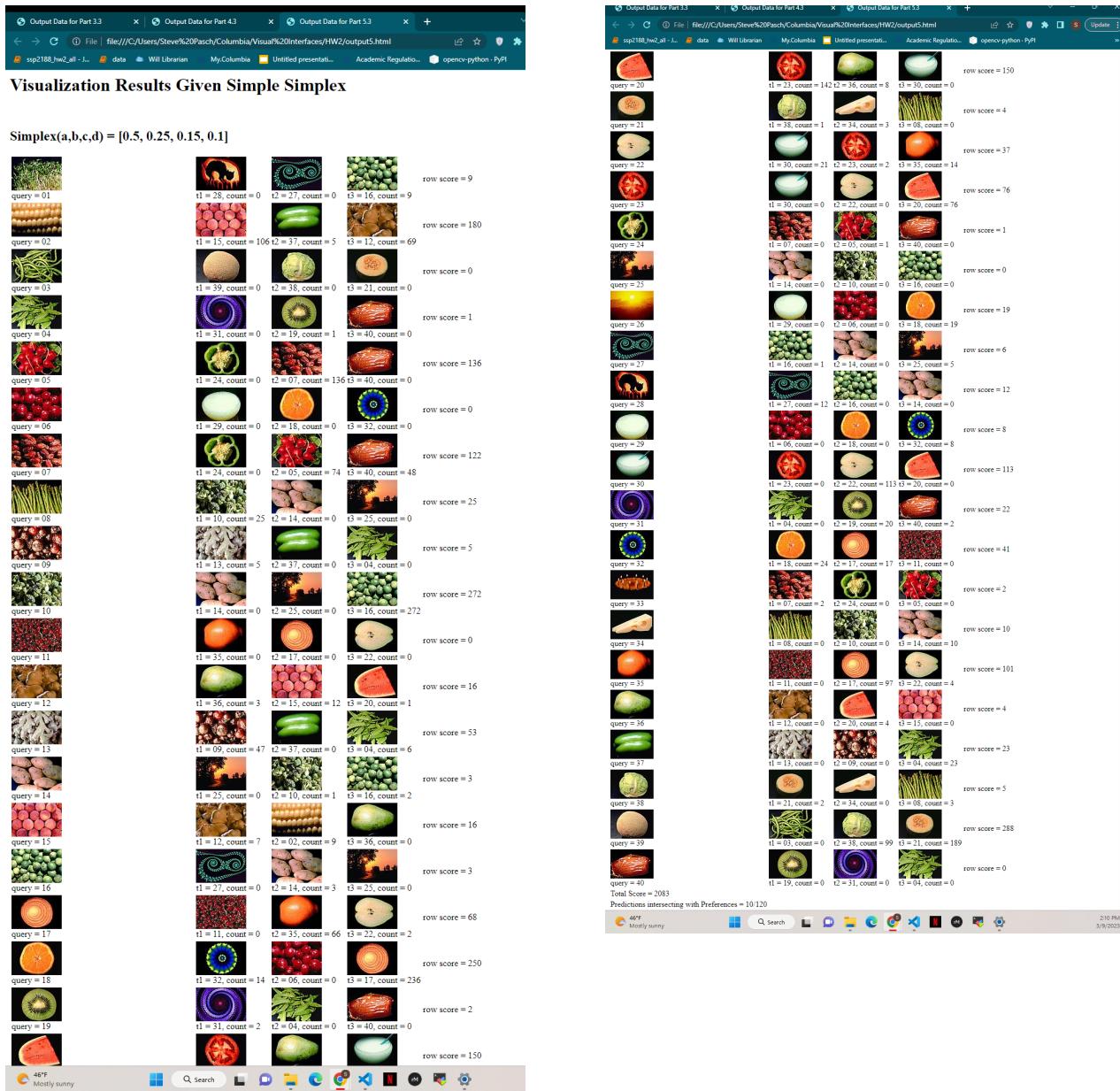
Step 4

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>