Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

Programming 1 Report

Part 1

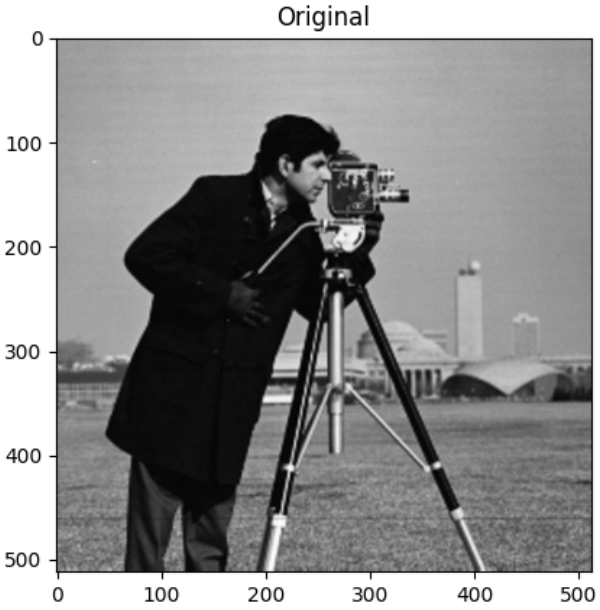
Introduction

For Part 1 of the assignment, I'm turning in one python code named "filters.py", which contains the codes for 3x3 and 5x5 Gaussian filters, Derivative of Gaussian (DoG) g_x and g_y filters, and the sobel filter. We were given fiter1_img.jpg and filter2_img.jpg, which were in grayscale, to run the filters on, and I have hard coded the image paths to be the same location as the code file. Running the code outputs the results for both images for all the filters. Each image is preprocessed by adding zero pads of width 1 for 3x3 Gaussian filter, DoG g_x filter, and DoG g_y filter and zero pads of width 2 for 5x5 Gaussian filter. The Gaussian filters are calculated by function gaussian(). It calculates the values by taking a 3x3 or 5x5 window from the padded image and performing element-wise multiplication of the selected window and the appropriate filter (3x3 or 5x5), and then summing up the result. The DoG calculations are done with DoG() function, and essentially work the same way as a 3x3 Gaussian except the filters used are g_x or g_y. The Sobel filter calculation is performed by sobel() function using the results from DoG g_x and DoG g_y filters.

(i) Gaussian Filter

Results

Image 1 Original:



Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

Image 1 Gaussian 3x3:

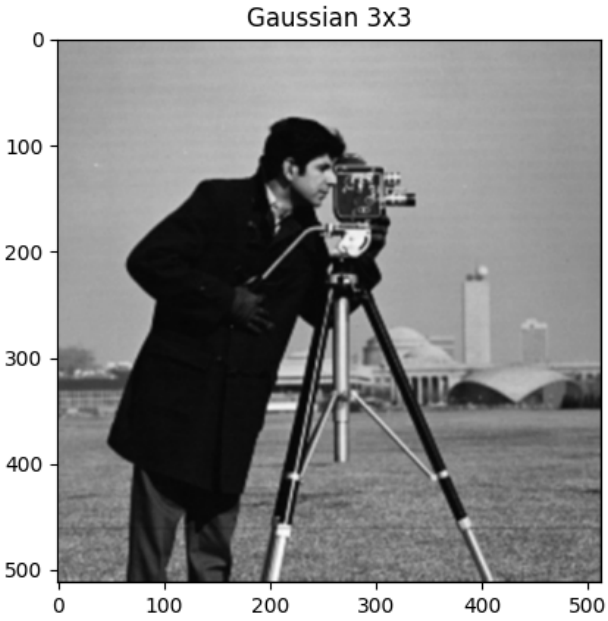
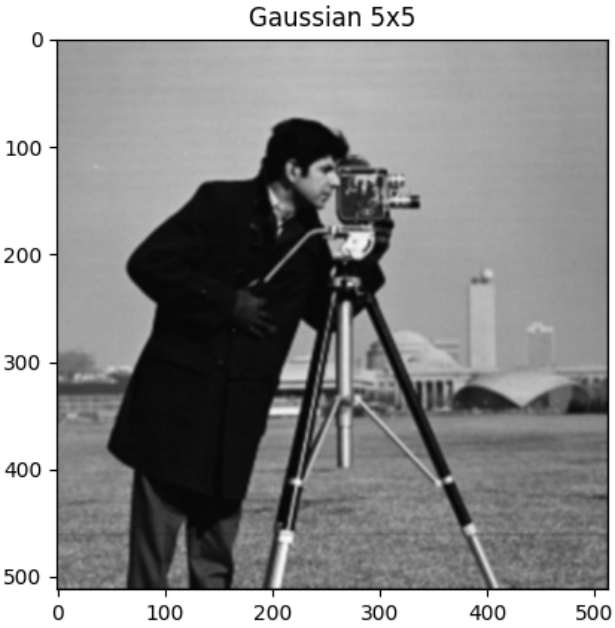


Image 1 Gaussian 5x5:



Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

Image 2 Original:

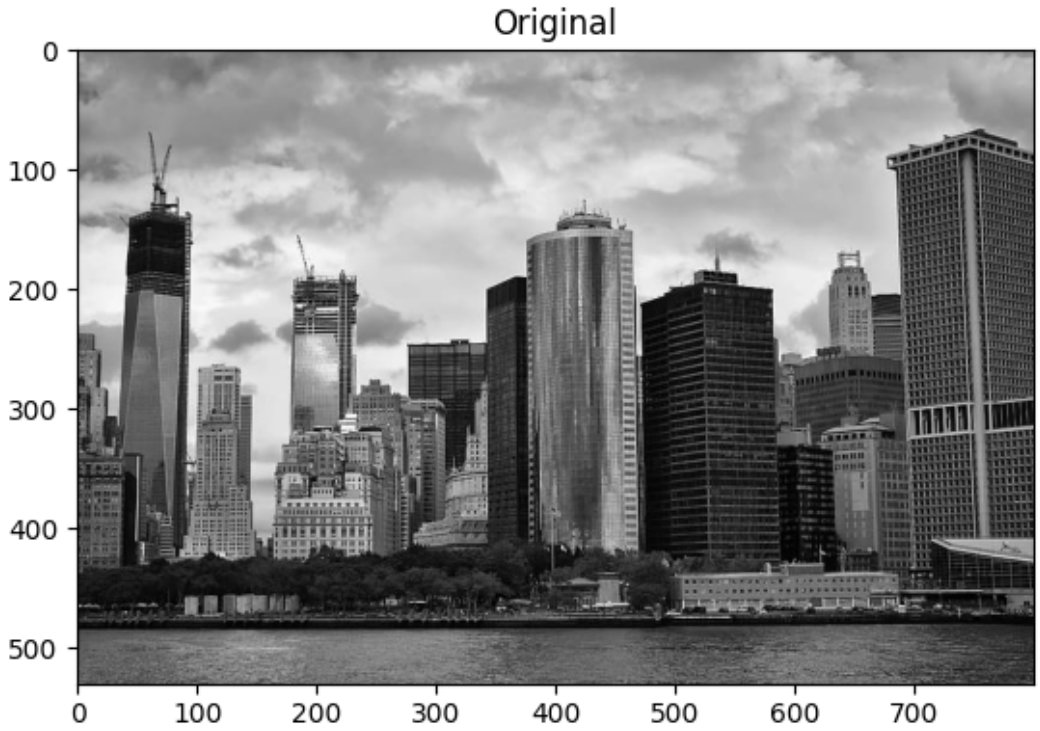
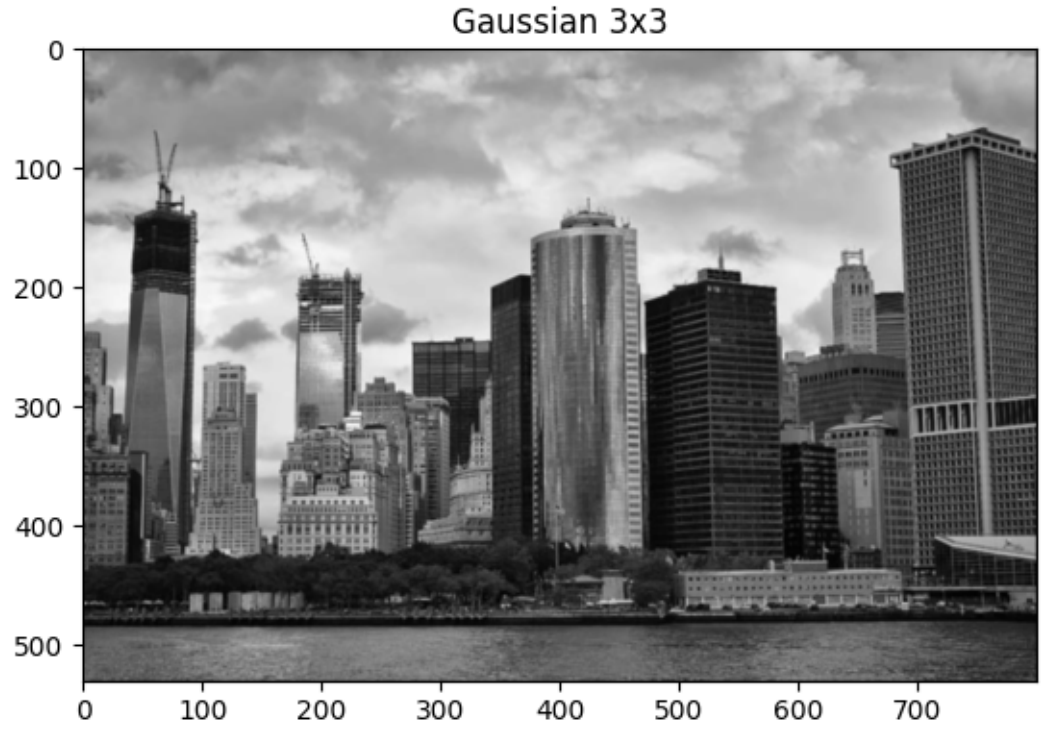
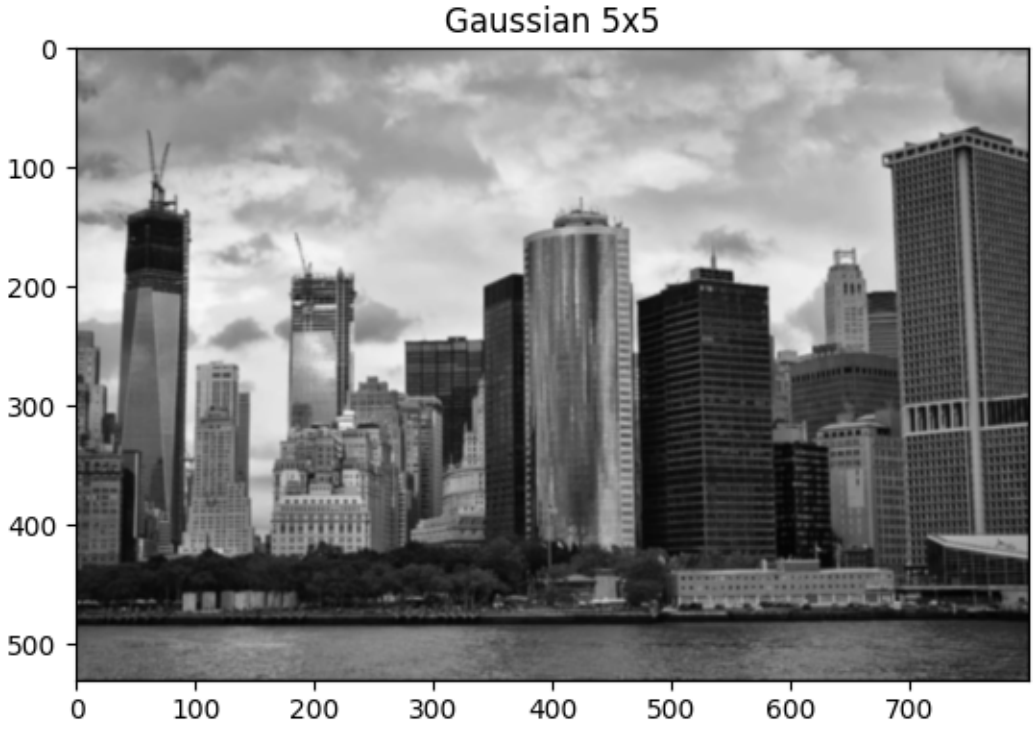


Image 2 Gaussian 3x3:



Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

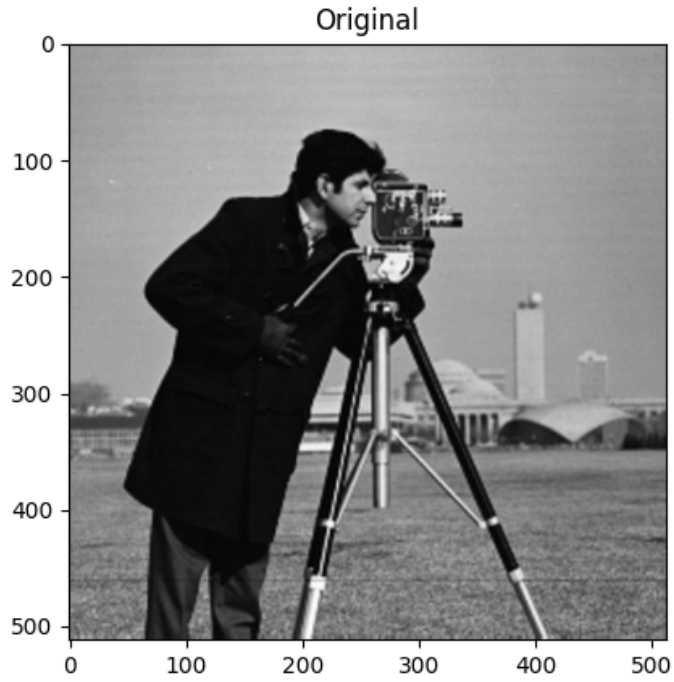
Image 2 Gaussian 5x5:



(ii) Derivative of Gaussian (DoG) Filter

Results

Image 1 Original:



Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

Image 1 DoG g_x:

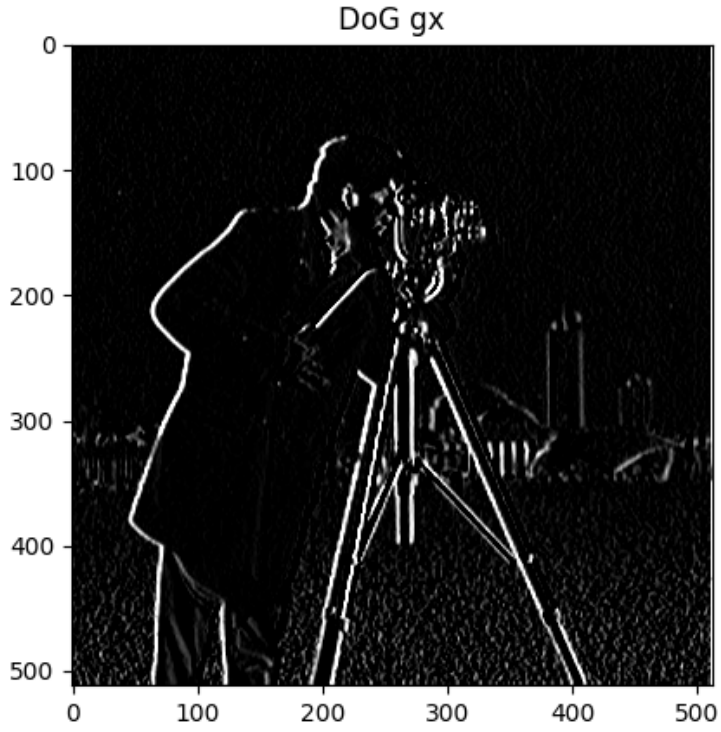
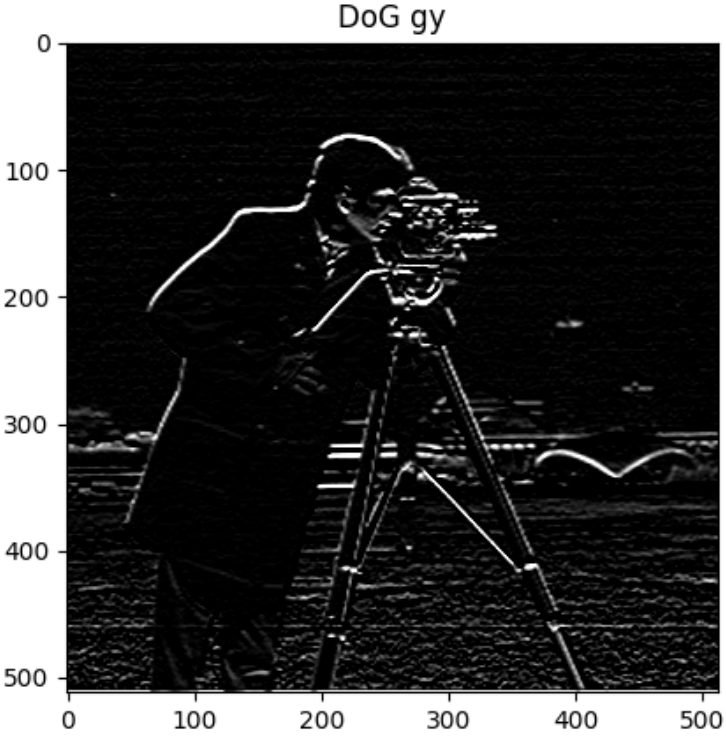


Image 1 DoG g_y:



Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

Image 2 Original:

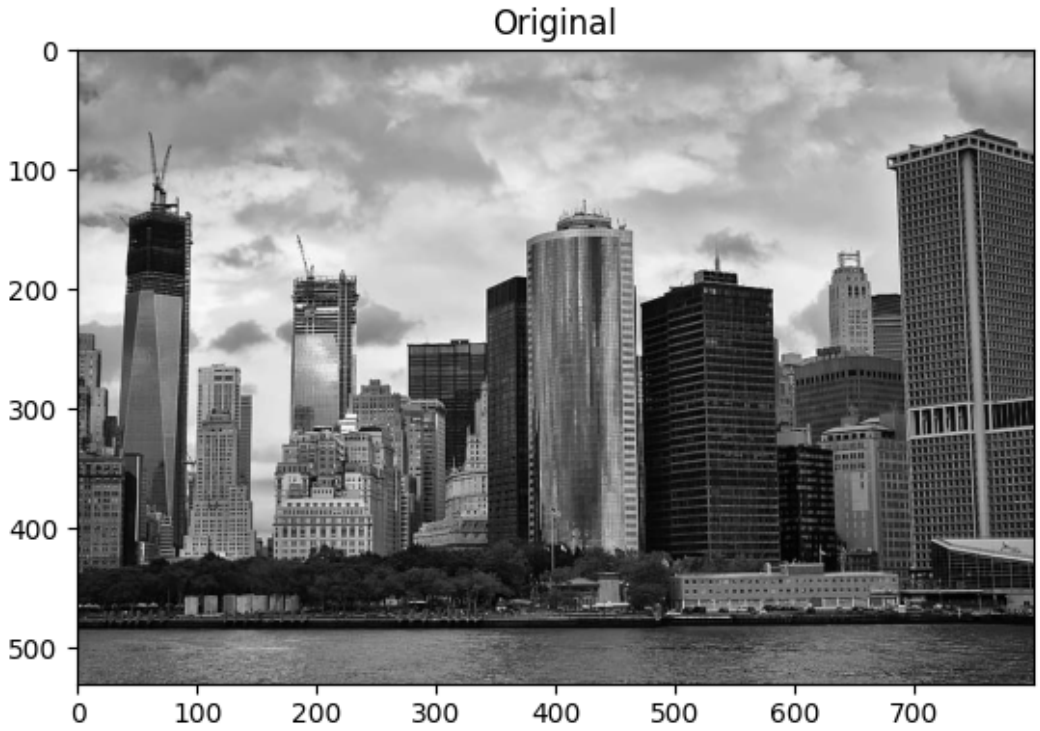
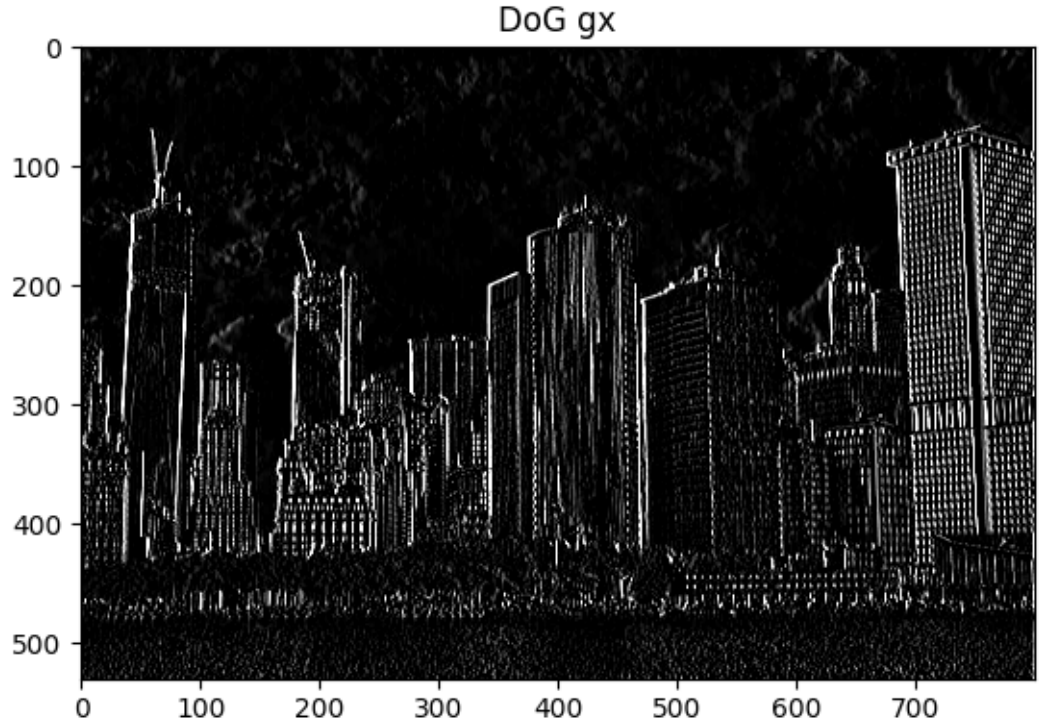
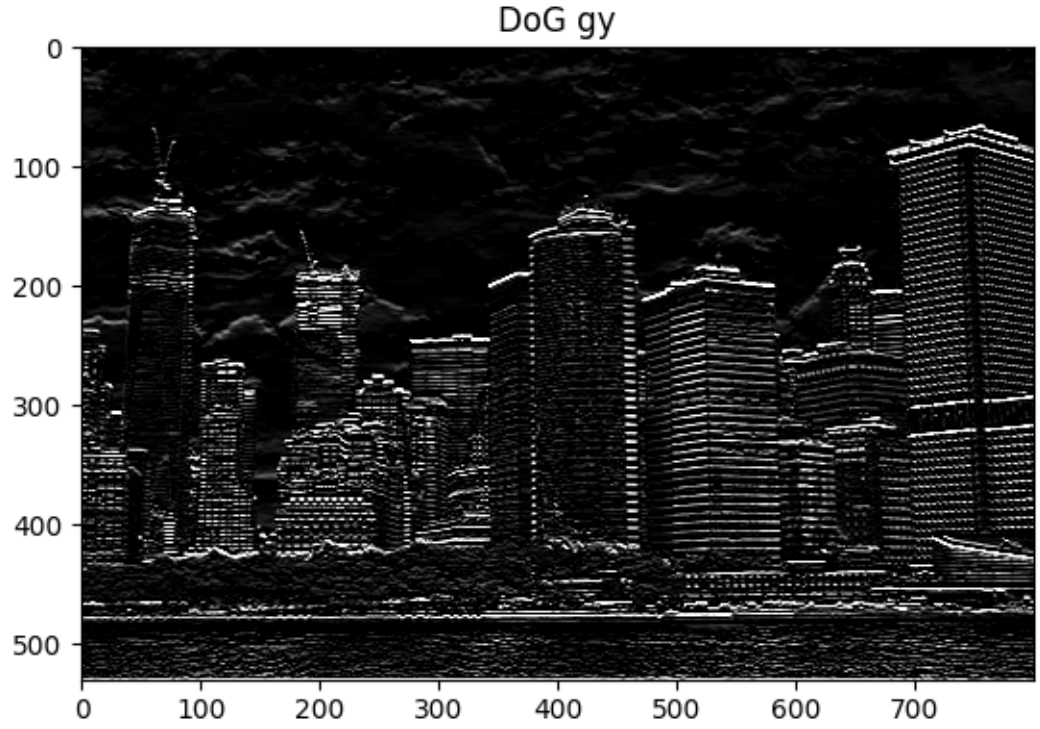


Image 2 DoG g_x:



Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

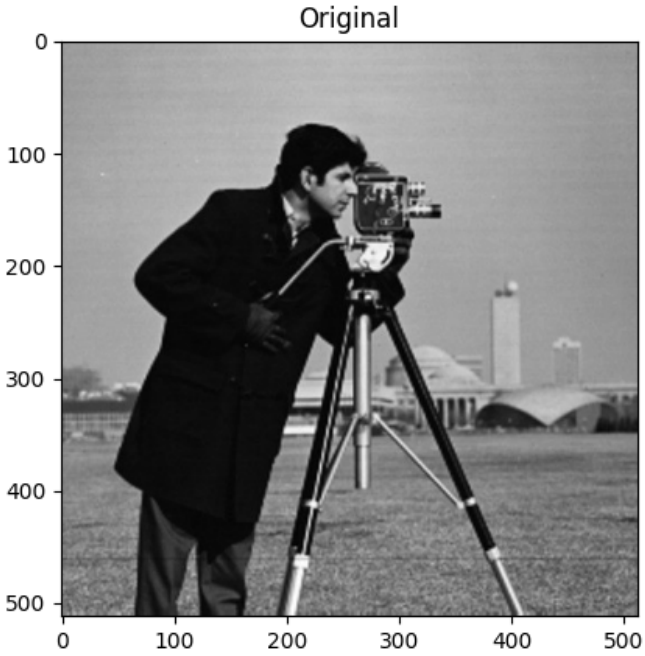
Image 2 DoG g_y:



(iii) Sobel Filter

Results

Image 1 Original:



Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

Image 1 Sobel:

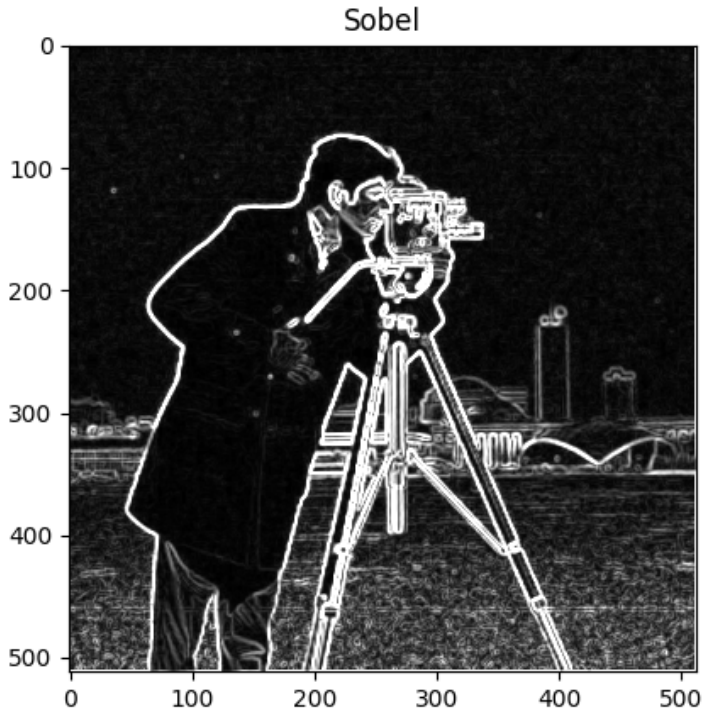


Image 2 Original:

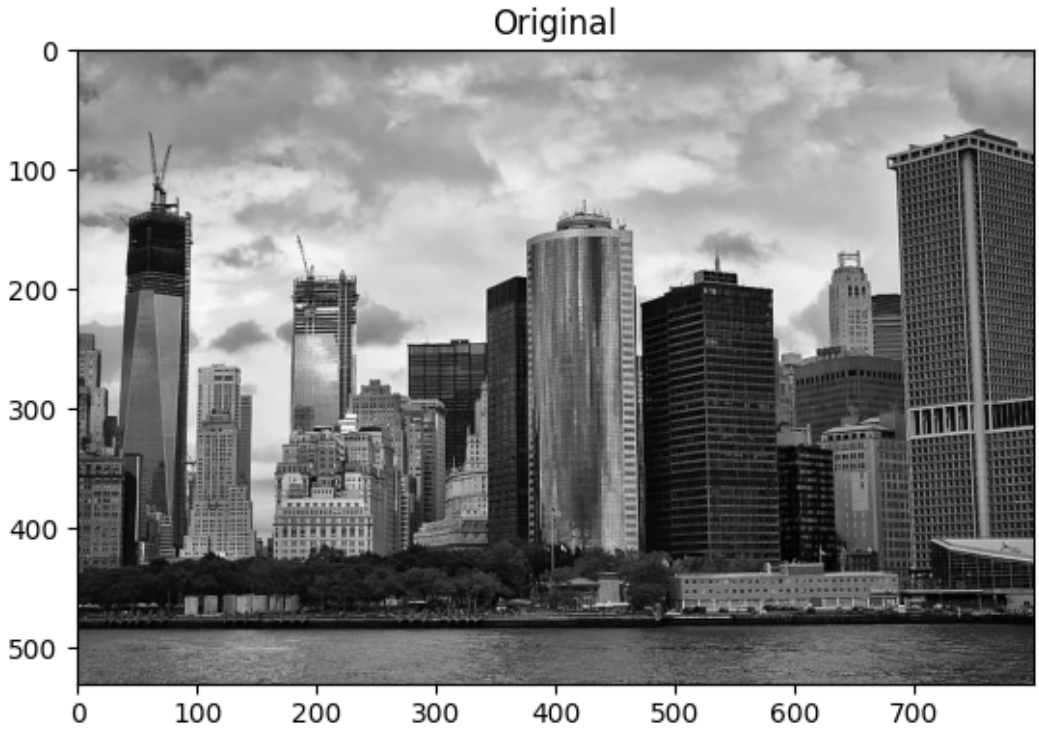
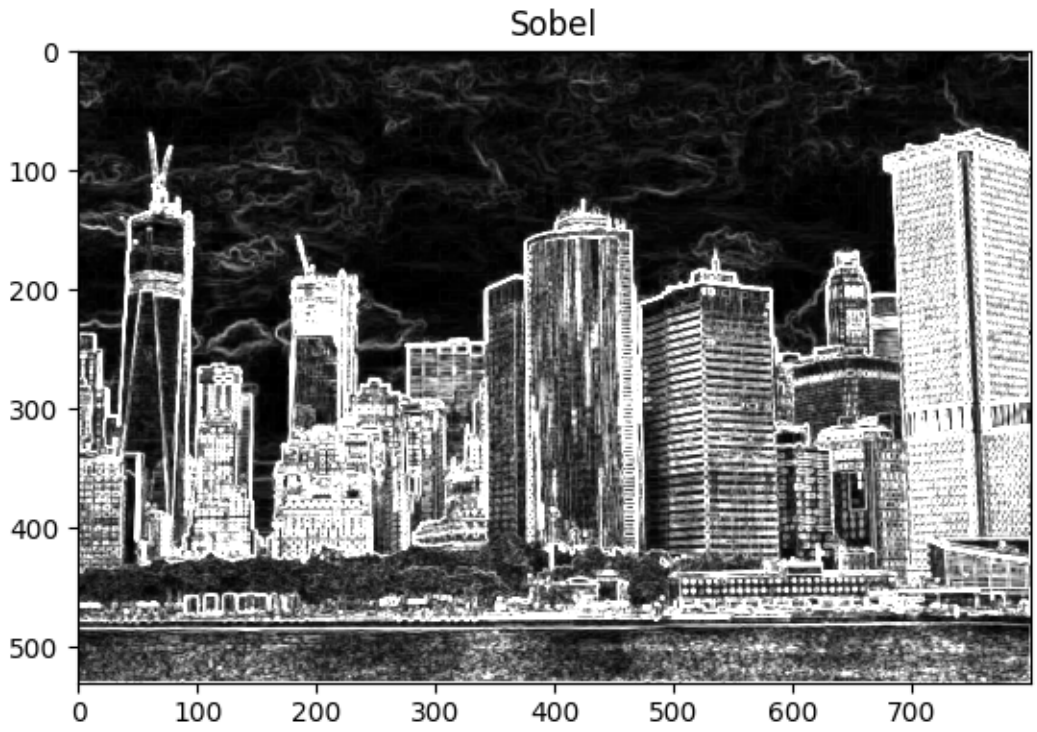


Image 2 Sobel:



Conclusion

5x5 filter seems to blur the image more than 3x3 filter. DoG g_x filter is good at detecting edges of vertical lines while DoG g_y filter is good at detecting edges of horizontal lines.

Part 2

Introduction

For Part 2 of the assignment, I implemented a K-Means algorithm (k_means_v2.py) that can take a .txt file containing 2D points as input as well as images with 3 color channels as input. For 2D data, it outputs 2D plot using randomized colors for each cluster for part(i). For 3 color channel images, it outputs an image with each pixel colored the same color as its cluster for part(ii). The data set was provided by Dr. Rhodes for part (i). The data set contained 1500 2D points. Kmean_img1.jpg and Kmean_img2.jpg, which contained images in RGB colors, were given for part (ii). Currently the program is set to run the K-Means algorithm 3 times for 2D data so you could input K=2, K=3, and K=4 as required by the assignment, and it runs twice for each image so you could input K=5 and K=10. The algorithm asks for how many runs it should do

and how many clusters (k) it needs to form per run. For each run, *k* points are randomly chosen as initial means. Using the means, the algorithm calculates Euclidean distance between each points as assigns proper cluster in the assignment( ) function and recomputes the means using the recompute( ) function. The algorithm runs until the means no longer change. The program calculates the Sum of Squares Error (SSE) for each run, and outputs the means with minimum SSE.

(i) 2D K-Means

Each cluster was assigned a random color by the program for plotting. Means are marked as black points. I ran the algorithm 10 times per k-size. Initial mean points, final mean points, and SSE per run were recorded.

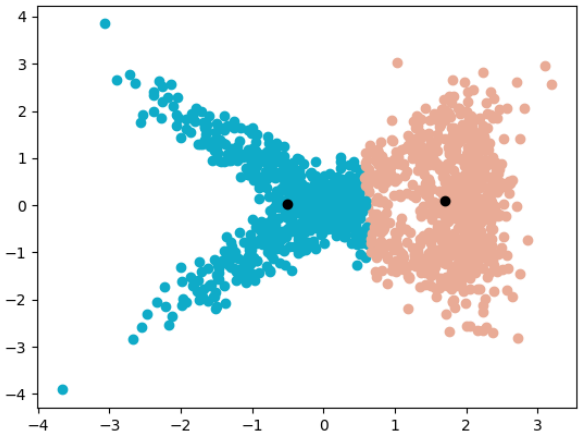
Results

The run with green highlight has the lowest SSE for the K. SSE was rounded to 2nd decimal place. I have included graphs of some of the runs for the different SSEs. The smallest SSE is highlighted in light green.

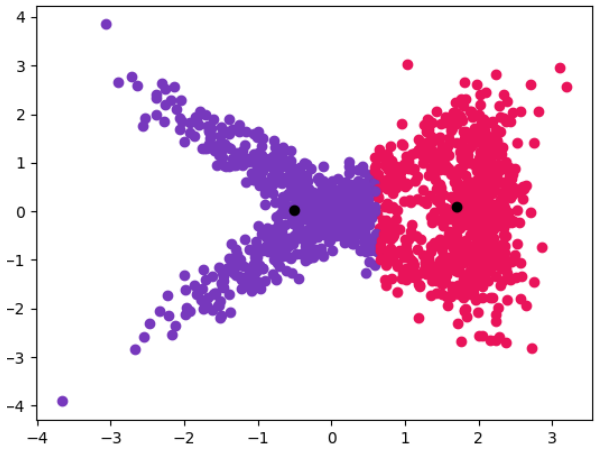
K = 2			
Runs	**Initial Means**	**Final Means**	**SSE**
1	[0.878127 0.924322] [-0.721023 -0.285133]	[1.69939044 0.09067278] [-0.50864058 0.02005367]	2164.44
2	[0.614224 -0.051647] [-0.339806 0.743384]	[1.69081902 0.08843396] [-0.51774696 0.02182924]	2171.04
3	[-0.892317 -0.711155] [0.053045 -0.277425]	[-0.51774696 0.02182924] [1.69081902 0.08843396]	2171.04
4	[2.086736 -0.704343] [0.223641 -0.290212]	[1.70232498 0.08960845] [-0.50568675 0.0213408]	2168.32
5	[2.284288 -0.001563] [-0.433158 -0.397562]	[1.70232498 0.08960845] [-0.50568675 0.0213408]	2168.32
6	[1.8857 1.003853] [-0.143903 0.214206]	[1.70232498 0.08960845] [-0.50568675 0.0213408]	2168.32
7	[1.423098 -1.677774] [-1.091898 -1.631381]	[1.69081902 0.08843396] [-0.51774696 0.02182924]	2171.04
8	[-0.359269 0.117875] [0.732056 -0.473954]	[-0.51774696 0.02182924] [1.69081902 0.08843396]	2171.04
9	[2.102896 1.074966] [-1.263716 1.151601]	[1.69081902 0.08843396] [-0.51774696 0.02182924]	2171.04
10	[-0.028479 0.12617] [0.643963 -0.42177]	[-0.50568675 0.0213408] [1.70232498 0.08960845]	2168.32

Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

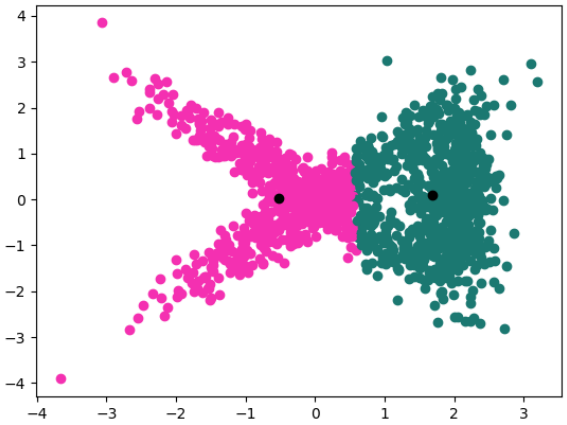
K=2 Run 1



K=2 Run 5



K=2 Run 7

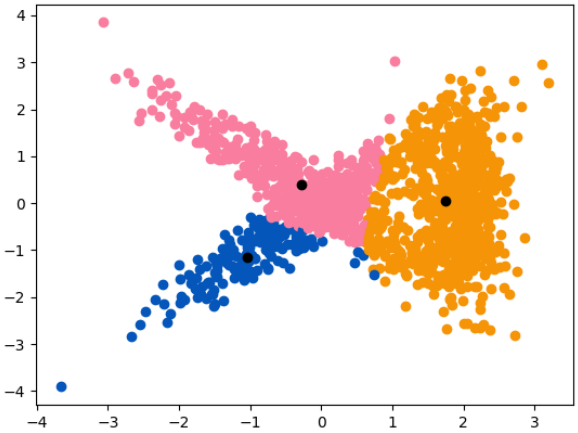


K = 3			
Runs	**Initial Means**	**Final Means**	**SSE**
1	[-1.189463 -1.125899] [-0.530223 -0.640602] [0.368226 0.107002]	[-1.03628358 -1.15076271] [-0.28450025 0.40417398] [1.75003436 0.05100056]	1603.77
2	[-0.108676 -0.398364] [1.070648 -0.979797] [-1.512506 1.545157]	[-0.16822732 -0.39443595] [1.79402114 0.17475807] [-1.17925695 1.23781304]	1915.94
3	[2.515471 -0.672047] [-0.266341 -0.506297] [1.931585 -0.234411]	[1.64650399 -0.85083626] [-0.52458245 0.03628108] [1.71541534 1.04614047]	1478.37
4	[0.054442 0.039398] [-0.695022 -0.920898] [2.399257 -0.129078]	[-0.28450025 0.40417398] [-1.03628358 -1.15076271] [1.75003436 0.05100056]	1603.77
5	[-2.209405 -2.148982] [1.21431 1.422195] [-0.063276 -0.382322]	[-1.03628358 -1.15076271] [1.75003436 0.05100056] [-0.28450025 0.40417398]	1603.77

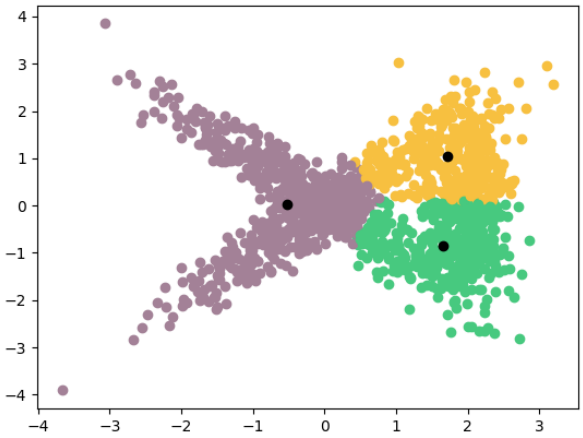
Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

6	[-0.327314 0.328952] [0.769671 0.849244] [2.103929 -0.142343]	[-0.52175499 0.03823938] [1.72186328 1.04772567] [1.64650399 -0.85083626]	1479.82
7	[1.245458 0.432252] [0.21732 -0.528494] [2.03124 0.01469]	[1.724763 1.04927424] [-0.52016677 0.03882836] [1.64650399 -0.85083626]	1479.54
8	[-1.336148 0.947007] [-0.15513 0.098193] [2.063661 0.355488]	[-1.17925695 1.23781304] [-0.16822732 -0.39443595] [1.79402114 0.17475807]	1915.94
9	[-0.397251 0.041922] [-0.325643 -0.074673] [-0.666205 -0.535584]	[-1.17925695 1.23781304] [1.79224925 0.17553264] [-0.16939125 -0.3961673]	1913.01
10	[-1.127723 1.032879] [0.091028 -0.030306] [0.066545 -0.335599]	[-1.17925695 1.23781304] [1.79224925 0.17553264] [-0.16939125 -0.3961673]	1913.01

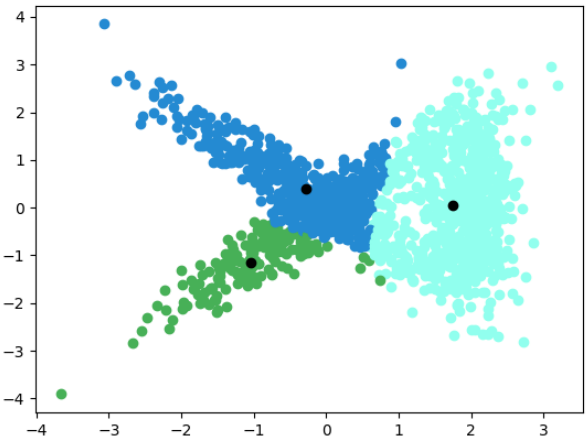
K=3 Run 1



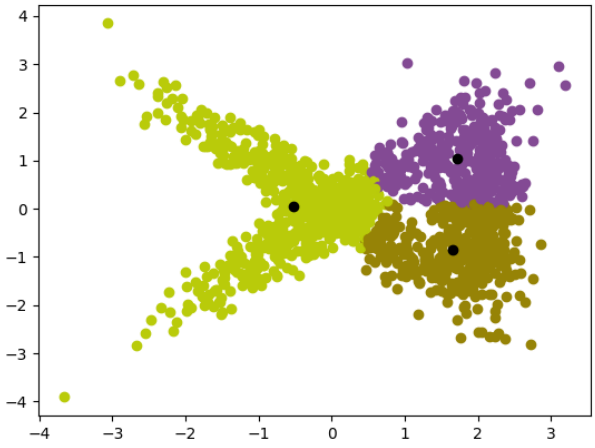
K=3 Run 3



K=3 Run 5

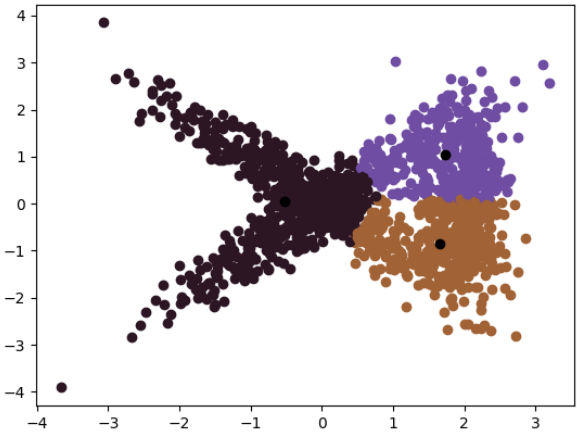


K=3 Run 6

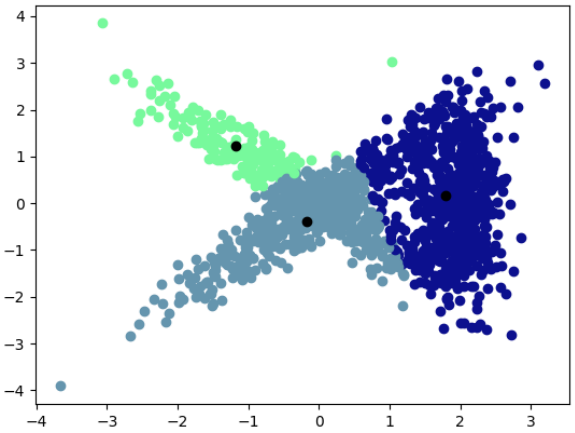


Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

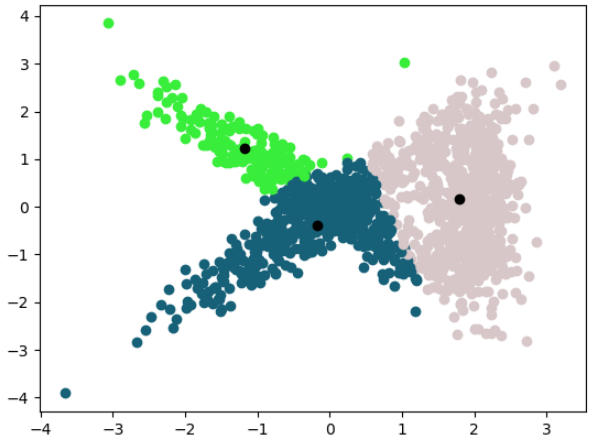
K=3 Run 7



K=3 Run 8



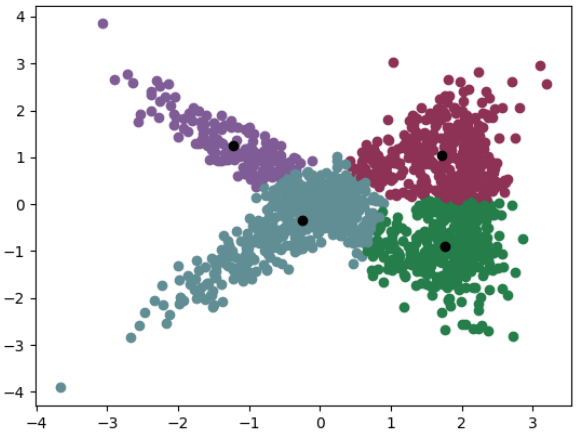
K=3 Run 10



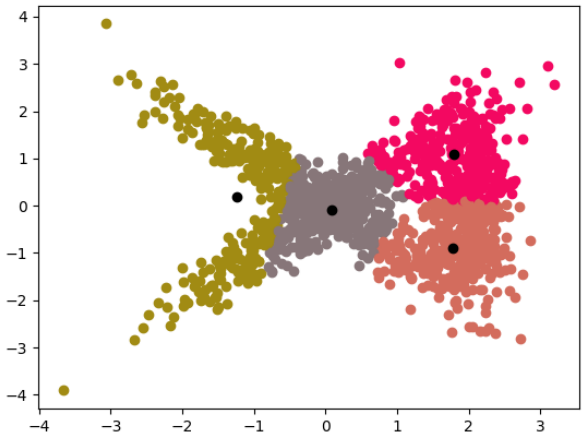
K = 4			
Runs	**Initial Means**	**Final Means**	**SSE**
1	[-0.493455 0.384125] [0.887540 -1.132606] [0.053233 -0.138308] [1.700506 0.000859]	[-1.2156474 1.24551518] [1.75641685 -0.89151568] [-0.25160605 -0.34303125] [1.7122852 1.04353302]	1422.15
2	[0.355589 -0.222219] [-1.677213 1.765818] [1.604107 1.898835] [2.399257 -0.129078]	[-0.25350305 -0.343343] [-1.2156474 1.24551518] [1.7122852 1.04353302] [1.7539538 -0.88945973]	1420.44
3	[-1.376366 1.295507] [2.040208 0.444612] [2.319132 -0.663918] [-0.327314 0.328952]	[-1.2156474 1.24551518] [1.7122852 1.04353302] [1.75641685 -0.89151568] [-0.25160605 -0.34303125]	1422.15

4	[2.017044 -1.322949] [0.648025 -0.332394] [-0.05066 -0.21711] [-0.710463 -0.285516]	[1.77342197 -0.89297734] [1.7927188 1.07997065] [0.08444769 -0.08358771] [-1.2312406 0.19797265]	1294.18
5	[0.535483 0.754891] [2.407111 -0.221771] [1.698821 1.403323] [1.940652 -0.948329]	[-0.52151121 0.0399479] [1.92277512 0.12343697] [1.62397952 1.36063094] [1.52988536 -1.12578616]	1325.85
6	[-2.537806 1.926895] [-1.753323 -1.598156] [1.164641 -0.787343] [0.079181 0.214384]	[-1.22984207 1.27082787] [-1.15179381 -1.19421484] [1.87505863 0.15439879] [0.19648694 -0.07463654]	1324.90
7	[0.98681 -0.942877] [-1.019407 -1.467897] [1.347919 -0.104028] [2.13213 0.143807]	[1.77621649 -0.89333204] [-1.22418634 0.19907] [0.08952139 -0.08736224] [1.7927188 1.07997065]	1295.37
8	[2.111994 1.445771] [2.290955 1.636436] [0.037179 -0.100798] [1.353029 0.875881]	[1.80799649 0.16842367] [1.69468589 1.42498345] [-0.52370607 0.04208018] [1.55498075 -1.12929263]	1303.90
9	[1.52024 0.996018] [-0.493455 0.384125] [1.452016 -1.088161] [-1.739437 -2.014993]	[1.78129547 1.06147901] [-0.32247339 0.40120375] [1.64199088 -0.85954018] [-1.08453109 -1.15569143]	810.56
10	[2.284288 -0.001563] [0.298869 -0.069559] [0.119117 0.460156] [2.349029 -0.7230180]	[1.7122852 1.04353302] [-0.25160605 -0.34303125] [-1.2156474 1.24551518] [1.75641685 -0.89151568]	1422.15

K=4 Run 2

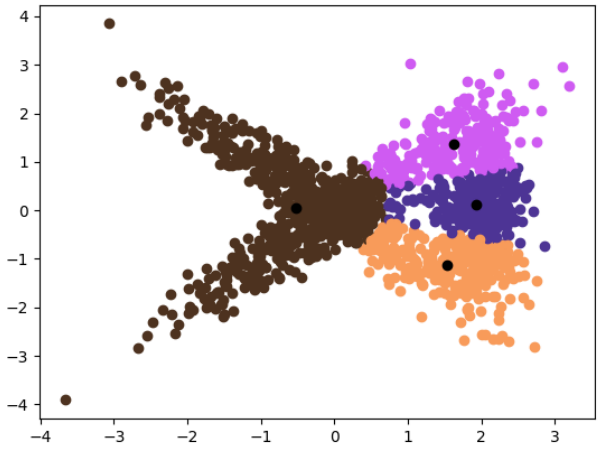


K=4 Run 4

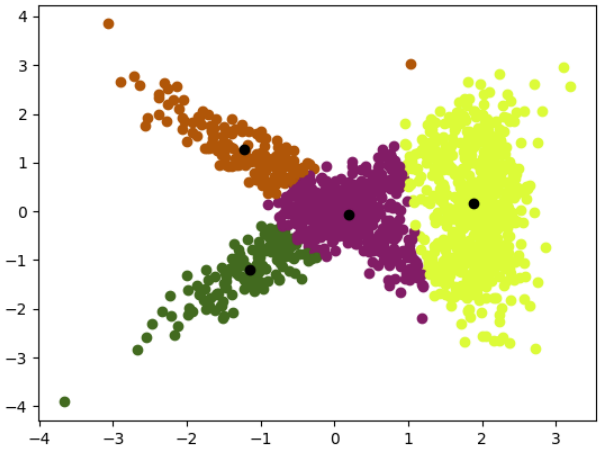


Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

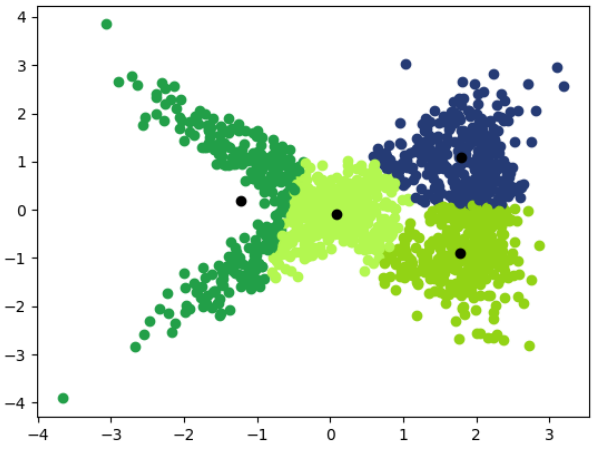
K=4 Run 5



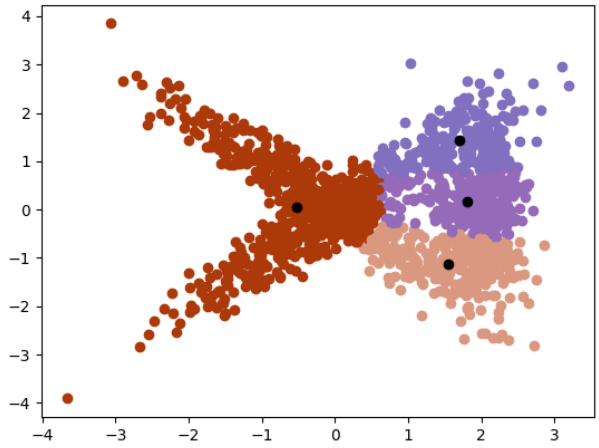
K=4 Run 6



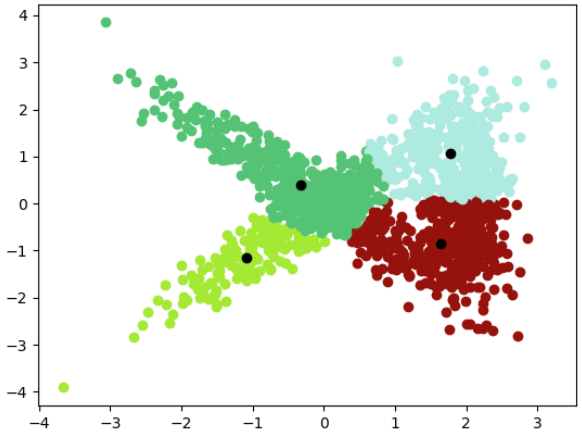
K=4 Run 7



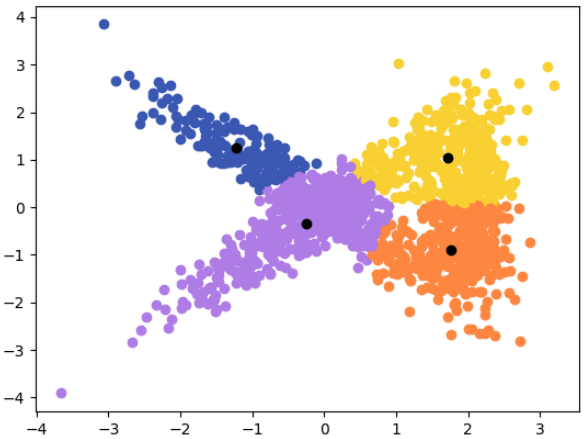
K=4 Run 8



K=4 Run 9



K=4 Run 10



(ii) 3D K-Means

Each cluster was assigned the RBG color of the final centroid means for image output. Since processing the images took a lot longer, I ran the algorithm 5 times per k-size. Initial mean points, final mean points, and SSE per run were recorded. The smallest SSE is highlighted in light green.

Image 1			
K = 5			
Runs	**Initial Means**	**Final Means**	**SSE**
1	[179. 193. 105.] [182. 190. 207.] [152. 175. 53.] [77. 157. 144.] [44. 117. 95.]	[172. 157. 125.] [192. 199. 218.] [161. 159. 59.] [37. 104. 101.] [17. 56. 46.]	1621934633.0
2	[143. 173. 160.] [161. 179. 72.] [155. 174. 63.] [174. 128. 111.] [0. 76. 76.]	[194. 201. 223.] [173. 166. 140.] [164. 178. 59.] [158. 115. 75.] [23. 74. 66.]	1492307877.0
3	[158. 179. 176.] [195. 200. 245.] [159. 184. 74.] [179. 162. 141.] [205. 197. 208.]	[158. 116. 77.] [194. 201. 225.] [23. 74. 66.] [164. 178. 59.] [174. 168. 143.]	1487645936.0
4	[166. 127. 83.] [175. 193. 224.] [189. 195. 140.] [191. 200. 233.] [162. 147. 85.]	[23. 74. 66.] [174. 168. 143.] [158. 116. 77.] [194. 201. 225.] [164. 178. 59.]	1487662280.0
5	[168. 114. 79.] [158. 181. 96.] [188. 187. 167.] [167. 138. 101.] [187. 171. 154.]	[23. 74. 66.] [164. 178. 59.] [194. 201. 225.] [158. 116. 77.] [174. 168. 143.]	1487633248.05
K = 10			
Runs	**Initial Means**	**Final Means**	**SSE**
1	[179. 210. 233.] [3. 70. 49.] [215. 222. 237.] [189. 194. 193.] [144. 170. 110.] [187. 174. 158.] [22. 53. 54.]	[183. 189. 199.] [22. 80. 71.] [178. 177. 157.] [162. 123. 87.] [164. 178. 58.] [147. 94. 40.] [15. 45. 35.]	514635681.0

	[90. 136. 170.] [185. 198. 212.] [206. 218. 252.]	[44. 117. 119.] [170. 157. 118.] [200. 208. 238.]	
2	[156. 179. 35.] [166. 116. 86.] [166. 174. 81.] [176. 137. 115.] [184. 183. 163.] [165. 182. 77.] [204. 210. 247.] [176. 134. 121.] [196. 202. 243.] [168. 130. 96.]	[162. 183. 42.] [22. 74. 67.] [156. 167. 73.] [158. 117. 81.] [171. 174. 142.] [176. 184. 74.] [199. 207. 236.] [171. 138. 115.] [182. 186. 189.] [145. 91. 33.]	820647838.0
3	[186. 188. 106.] [153. 178. 104.] [0. 119. 119.] [161. 142. 97.] [180. 197. 123.] [216. 225. 239.] [187. 190. 188.] [174. 147. 120.] [171. 190. 71.] [169. 169. 79.]	[165. 183. 121.] [34. 104. 101.] [17. 56. 46.] [147. 93. 39.] [175. 144. 125.] [198. 206. 235.] [181. 183. 181.] [161. 122. 86.] [164. 183. 43.] [164. 173. 71.]	645225917.0
4	[184. 196. 230.] [177. 193. 129.] [163. 125. 91.] [30. 85. 70.] [161. 103. 61.] [143. 172. 117.] [179. 166. 150.] [175. 131. 118.] [112. 101. 71.] [39. 80. 72.]	[198. 206. 235.] [172. 164. 131.] [164. 174. 74.] [15. 45. 34.] [148. 96. 44.] [164. 183. 44.] [181. 184. 183.] [164. 126. 93.] [43. 117. 118.] [22. 79. 70.]	591313007.0
5	[140. 170. 81.] [173. 191. 208.] [182. 194. 196.] [158. 186. 39.] [218. 220. 230.] [210. 196. 202.] [170. 186. 49.] [176. 168. 67.] [157. 179. 43.] [191. 191. 173.]	[22. 73. 66.] [180. 176. 162.] [158. 180. 120.] [163. 183. 43.] [200. 208. 238.] [183. 190. 201.] [165. 174. 72.] [158. 118. 81.] [145. 91. 34.] [173. 140. 117.]	831716740.0

Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

Image 1 Original:

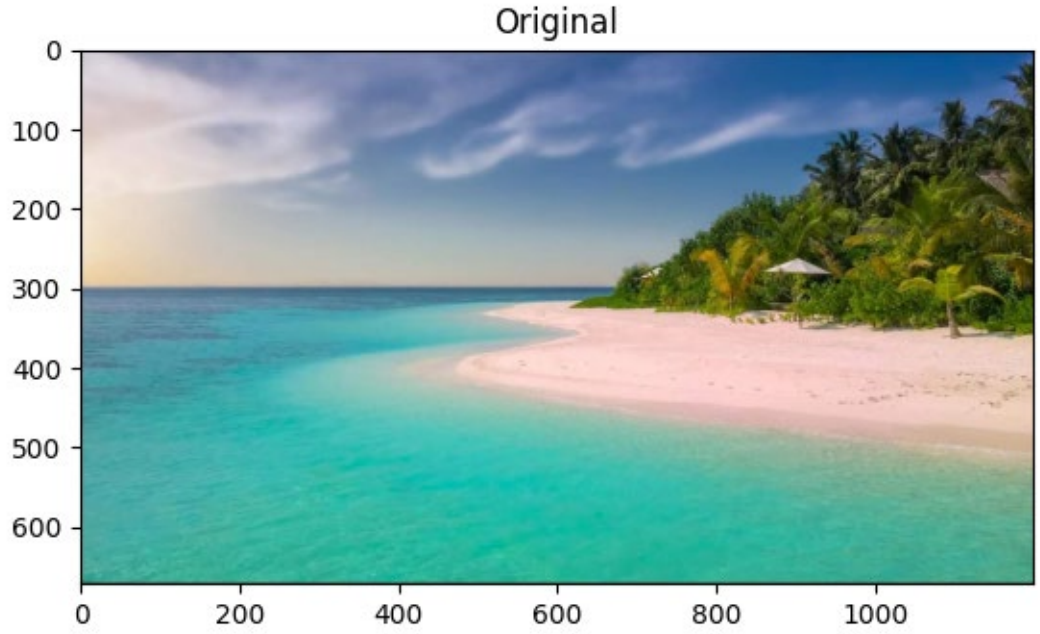
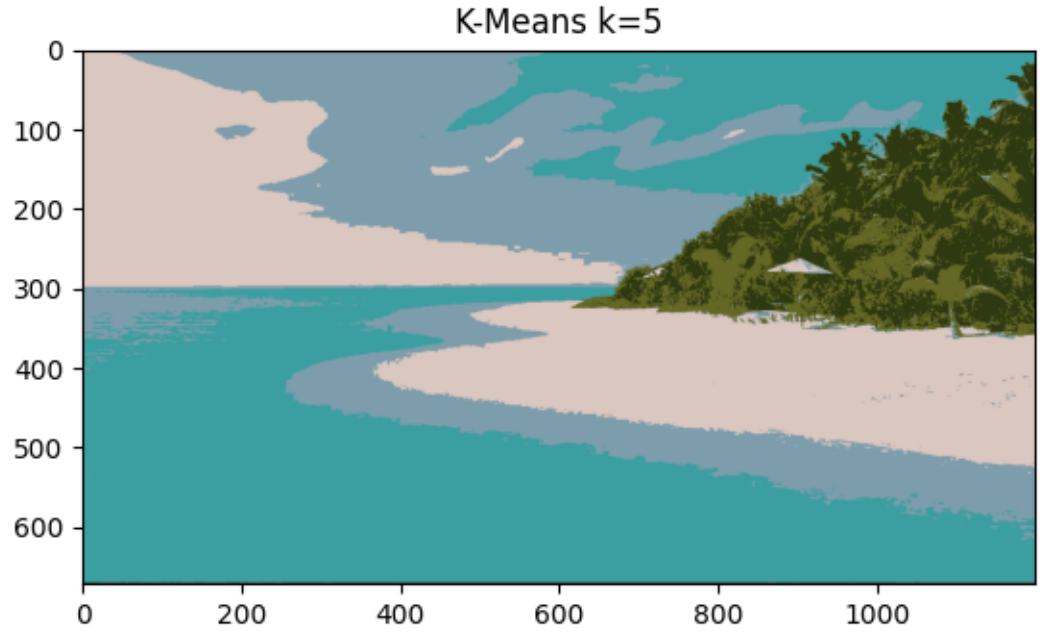


Image 1 K=5 Run 1:



Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

Image 1 K=5 Run 2:

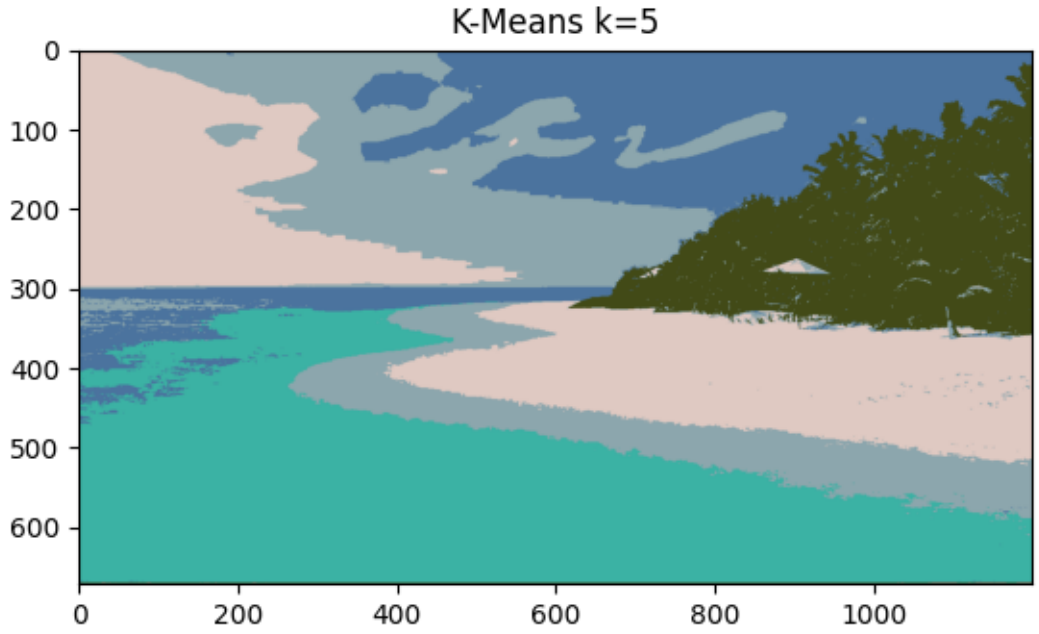
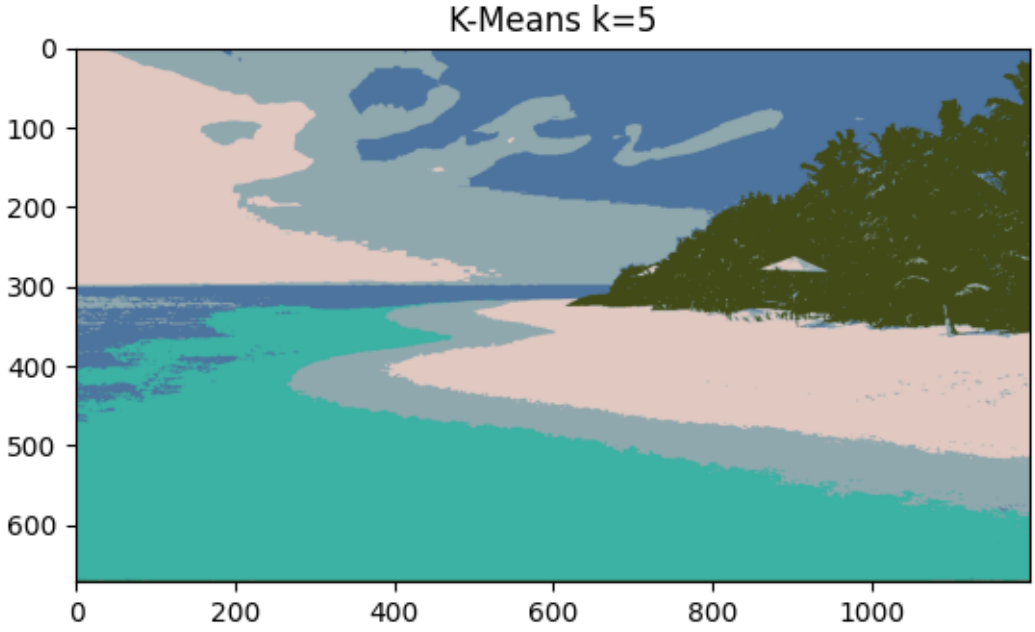


Image 1 K=5 Run 3:



Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

Image 1 K=5 Run 4:

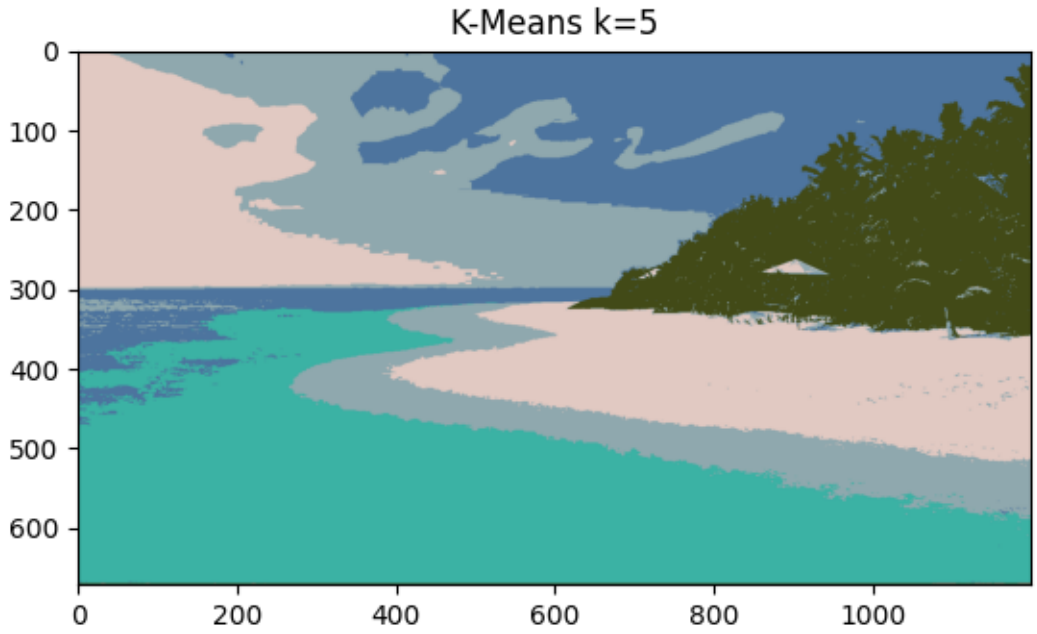
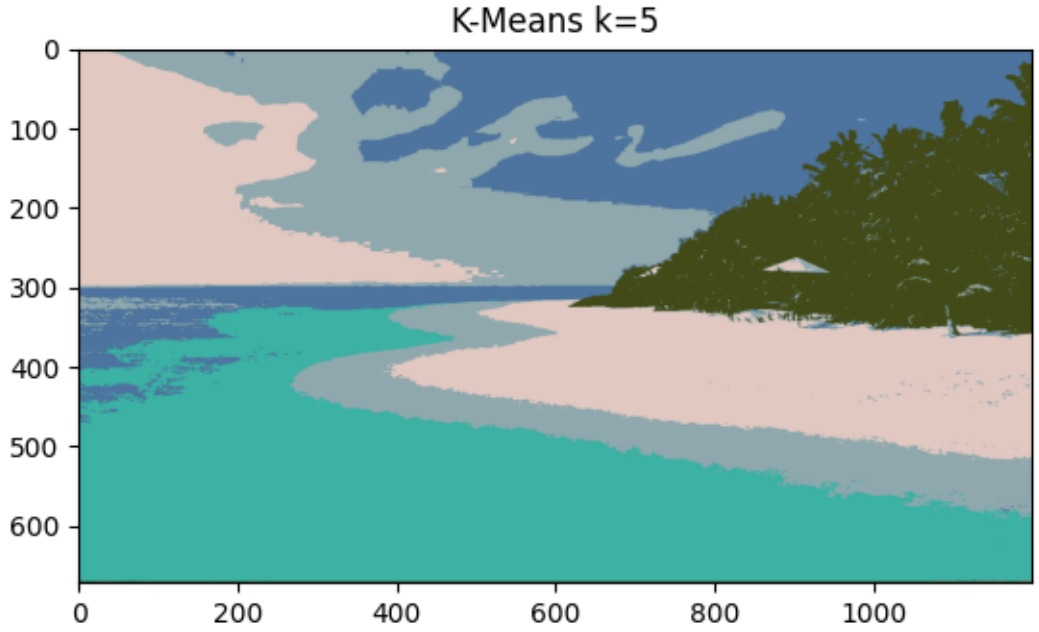


Image 1 K=5 Run 5:



Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

Image 1 K=10 Run 1:

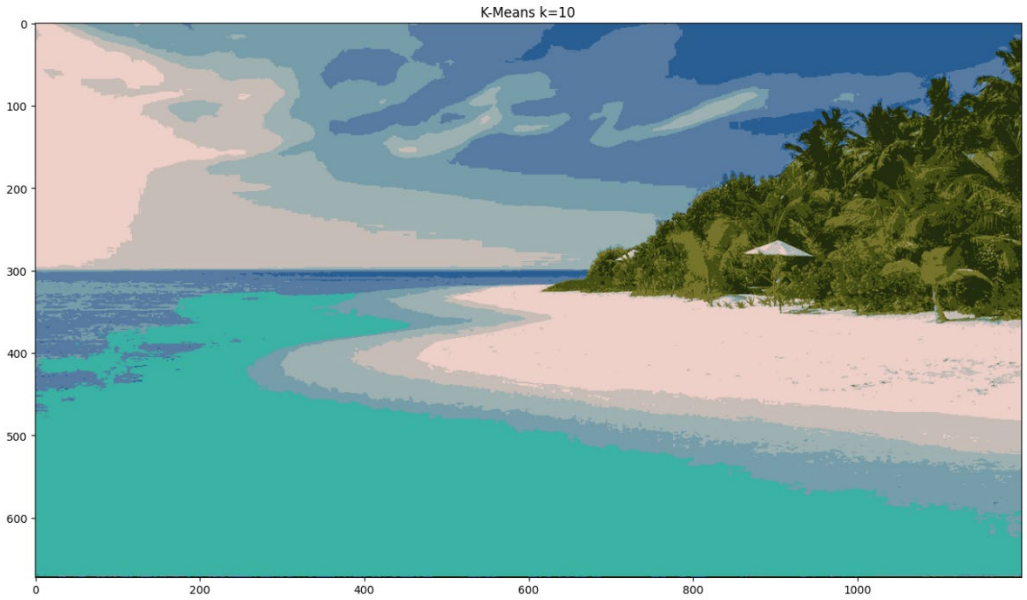
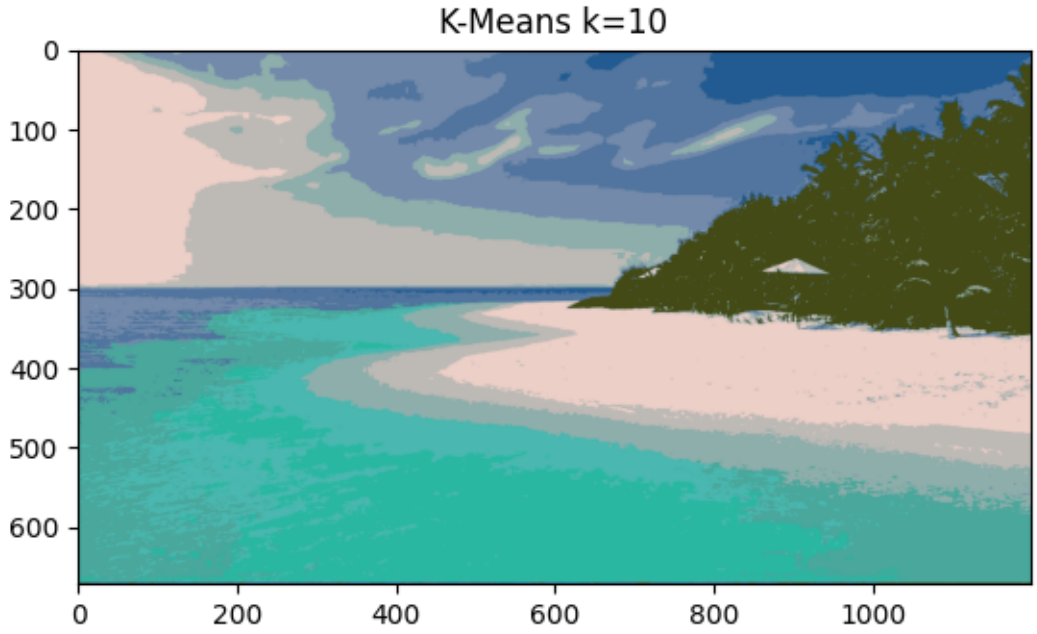


Image 1 K=10 Run 2:



Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

Image 1 K=10 Run 3:

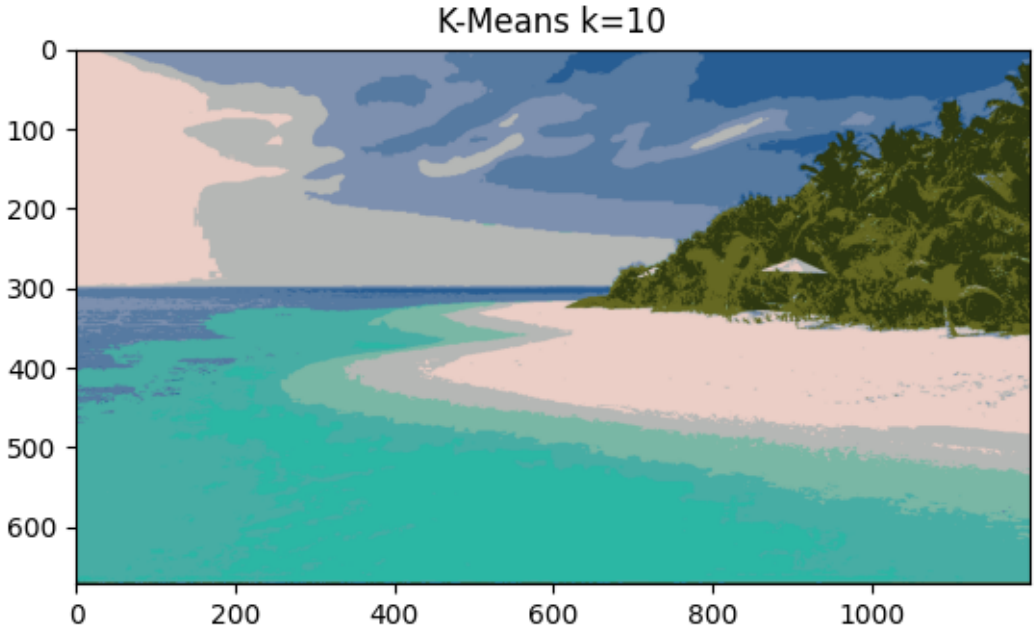
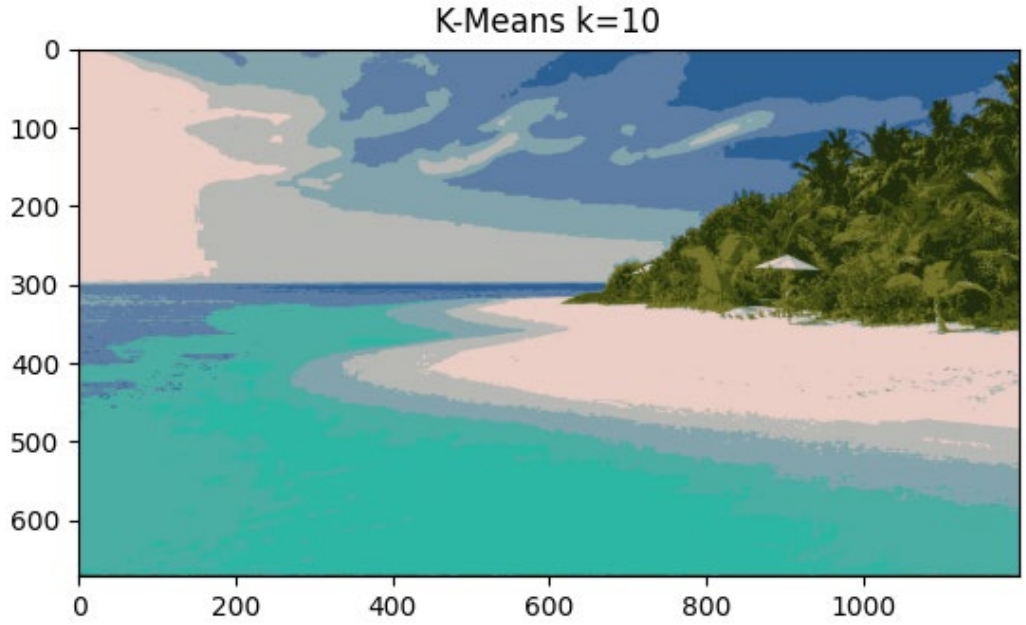


Image 1 K=10 Run 4:



Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

Image 1 K=10 Run 5:

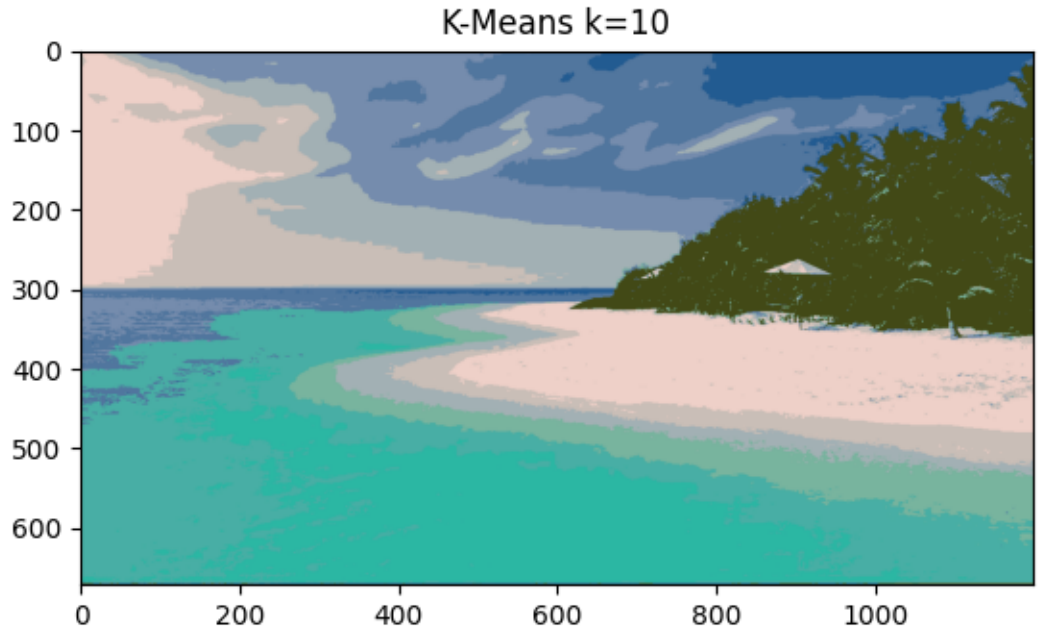
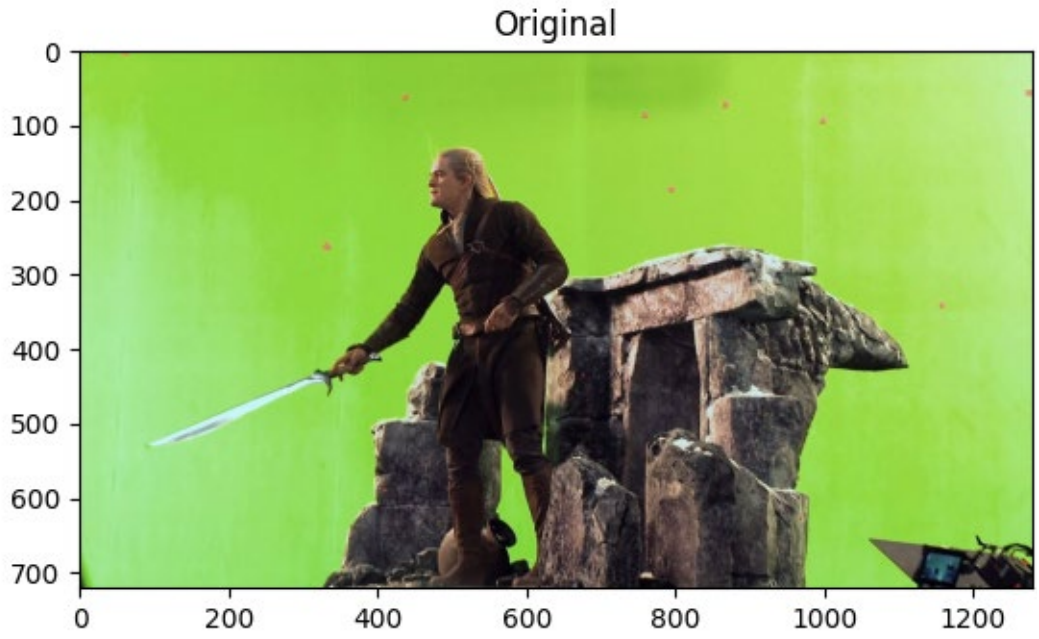


Image 2			
K = 5			
Runs	**Initial Means**	**Final Means**	**SSE**
1	[74. 222. 150.] [63. 217. 146.] [48. 49. 69.] [99. 227. 150.] [64. 216. 143.]	[101. 224. 161.] [65. 214. 148.] [32. 30. 37.] [145. 222. 176.] [80. 90. 101.]	1107207149.0
2	[125. 227. 155.] [62. 213. 150.] [106. 225. 146.] [111. 229. 176.] [68. 73. 74.]	[146. 221. 176.] [65. 214. 148.] [80. 90. 101.] [102. 225. 161.] [32. 30. 37.]	1113314489.0
3	[204. 222. 245.] [37. 29. 36.] [91. 222. 155.] [137. 231. 160.] [57. 208. 139.]	[80. 90. 101.] [32. 30. 37.] [101. 224. 161.] [145. 222. 176.] [65. 214. 148.]	1107189605.0
4	[63. 215. 142.] [53. 56. 61.] [94. 223. 148.] [56. 200. 141.] [62. 214. 148.]	[65. 214. 148.] [32. 30. 37.] [145. 222. 176.] [80. 90. 101.] [101. 224. 161.]	1107207149.0
5	[60. 215. 146.] [166. 172. 191.]	[65. 214. 148.] [145. 222. 176.]	1110324363.0

	[58. 65. 82.] [60. 215. 141.] [65. 214. 150.]	[32. 30. 37.] [80. 90. 101.] [101. 225. 161.]	
K = 10			
Runs	**Initial Means**	**Final Means**	**SSE**
1	[43. 43. 55.] [39. 46. 66.] [60. 215. 140.] [135. 230. 156.] [91. 224. 167.] [49. 40. 43.] [96. 225. 164.] [75. 220. 151.] [62. 209. 151.] [60. 217. 143.]	[48. 48. 55.] [76. 86. 96.] [56. 204. 139.] [186. 204. 205.] [100. 225. 160.] [22. 19. 24.] [134. 231. 169.] [74. 219. 156.] [116. 135. 152.] [62. 215. 144.]	447315358.0
2	[20. 15. 24.] [63. 216. 141.] [57. 51. 56.] [33. 28. 30.] [89. 220. 147.] [24. 16. 16.] [63. 215. 142.] [50. 61. 93.] [97. 223. 147.] [57. 62. 61.]	[28. 26. 33.] [60. 212. 143.] [66. 72. 80.] [46. 46. 53.] [109. 227. 162.] [16. 12. 17.] [77. 220. 156.] [131. 152. 172.] [150. 230. 177.] [90. 105. 117.]	401330030.0
3	[102. 227. 155.] [28. 24. 29.] [36. 36. 50.] [62. 216. 139.] [57. 201. 142.] [26. 28. 28.] [103. 225. 154.] [62. 216. 145.] [112. 231. 180.] [51. 47. 52.]	[99. 225. 160.] [19. 16. 21.] [60. 63. 71.] [59. 211. 142.] [123. 143. 162.] [39. 38. 44.] [133. 231. 168.] [72. 218. 154.] [190. 208. 207.] [84. 97. 108.]	377208353.0
4	[72. 217. 143.] [74. 70. 75.] [27. 26. 42.] [65. 214. 150.] [108. 228. 164.] [117. 126. 136.] [147. 233. 161.] [20. 16. 22.] [112. 130. 129.] [71. 220. 164.]	[99. 225. 160.] [63. 68. 76.] [41. 40. 47.] [59. 211. 142.] [133. 231. 168.] [126. 147. 166.] [192. 210. 207.] [20. 17. 22.] [88. 102. 113.] [72. 218. 154.]	370725690.0

5	[74. 221. 169.] [123. 230. 161.] [62. 217. 143.] [54. 66. 68.] [21. 17. 22.] [81. 221. 158.] [21. 14. 19.] [135. 231. 161.] [89. 102. 116.] [127. 229. 158.]	[72. 218. 155.] [100. 226. 172.] [59. 211. 142.] [75. 84. 94.] [47. 47. 54.] [100. 224. 150.] [21. 18. 24.] [184. 203. 204.] [113. 133. 149.] [135. 231. 168.]	463148683.0

Image 2 Original:



Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

Image 2 K=5 Run 1:

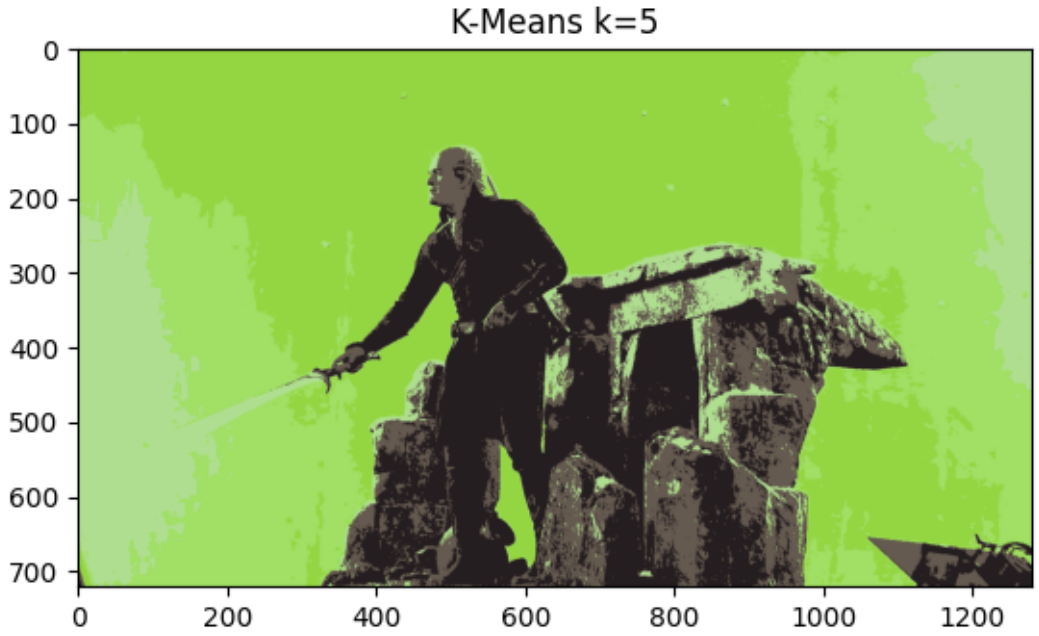


Image 2 K=5 Run 2:

Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

Image 2 K=5 Run 3:

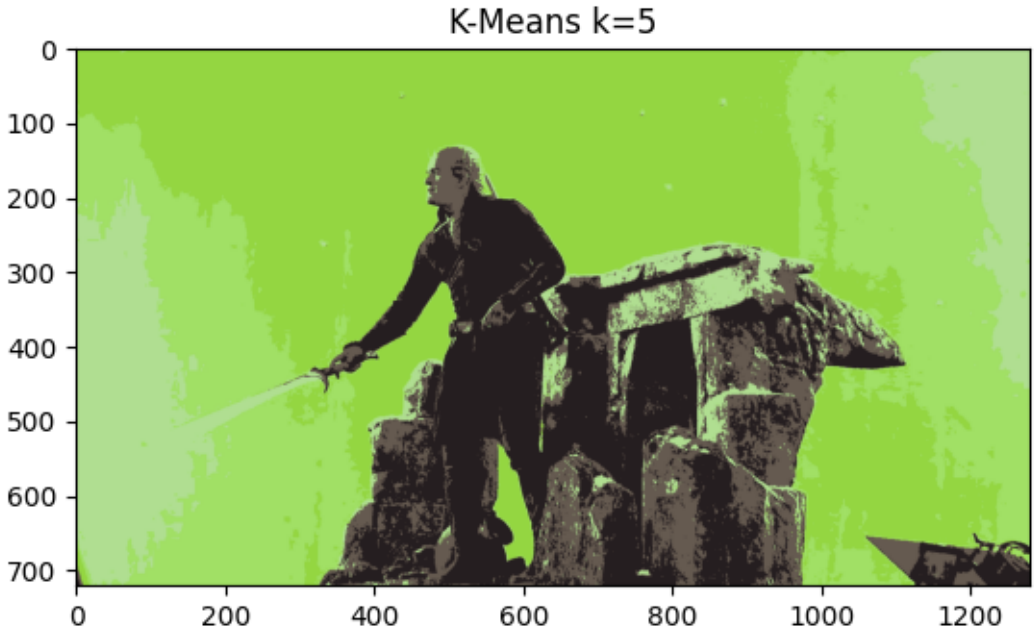


Image 2 K=5 Run 4:

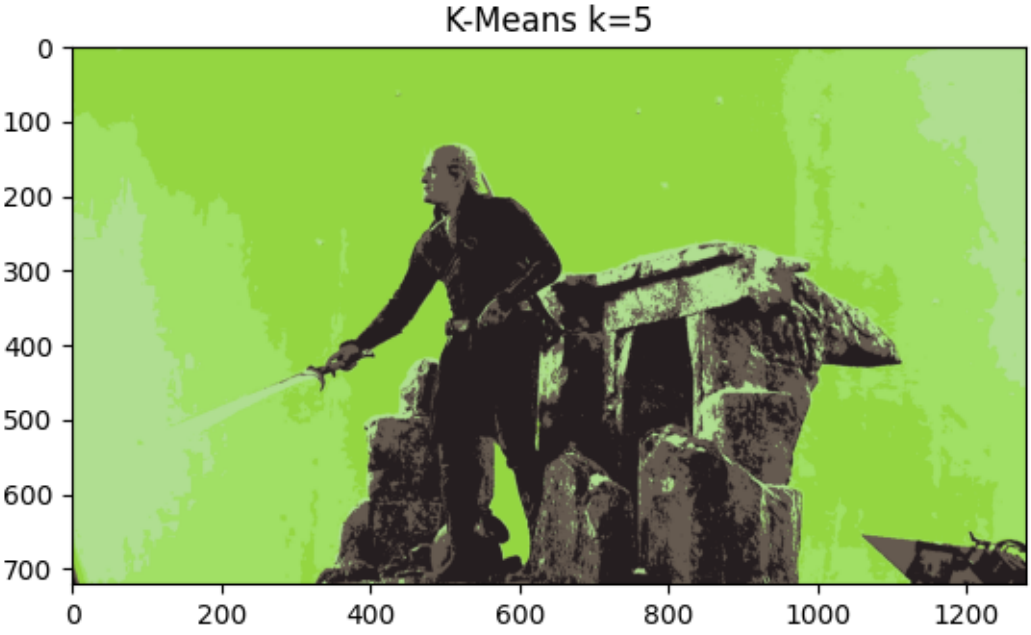


Image 2 K=5 Run 5:

Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

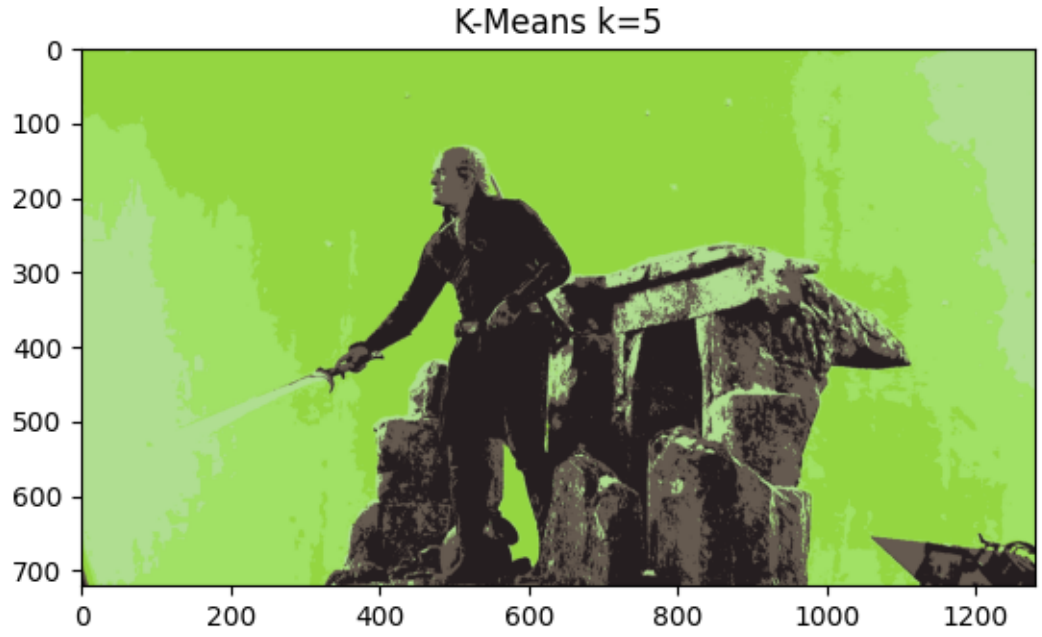
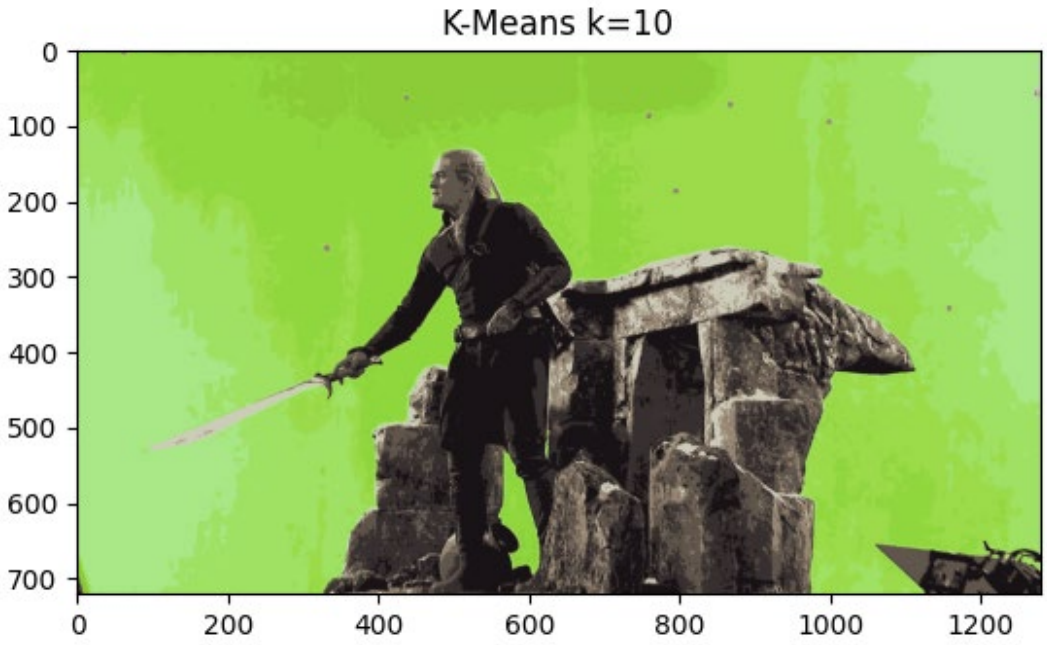


Image 2 K=10 Run 1:



Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

Image 2 K=10 Run 2:

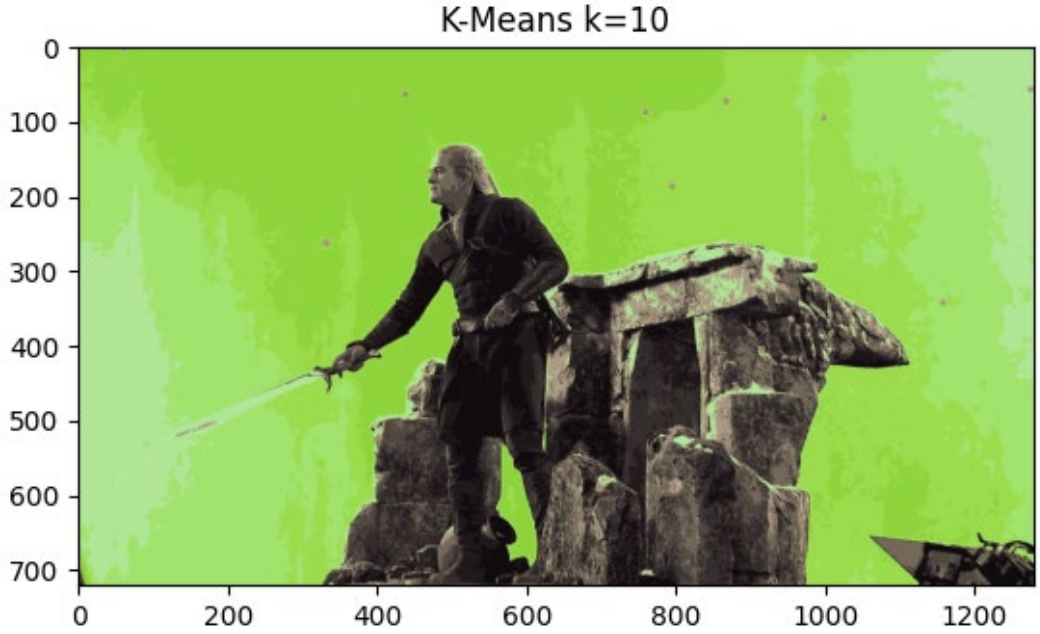
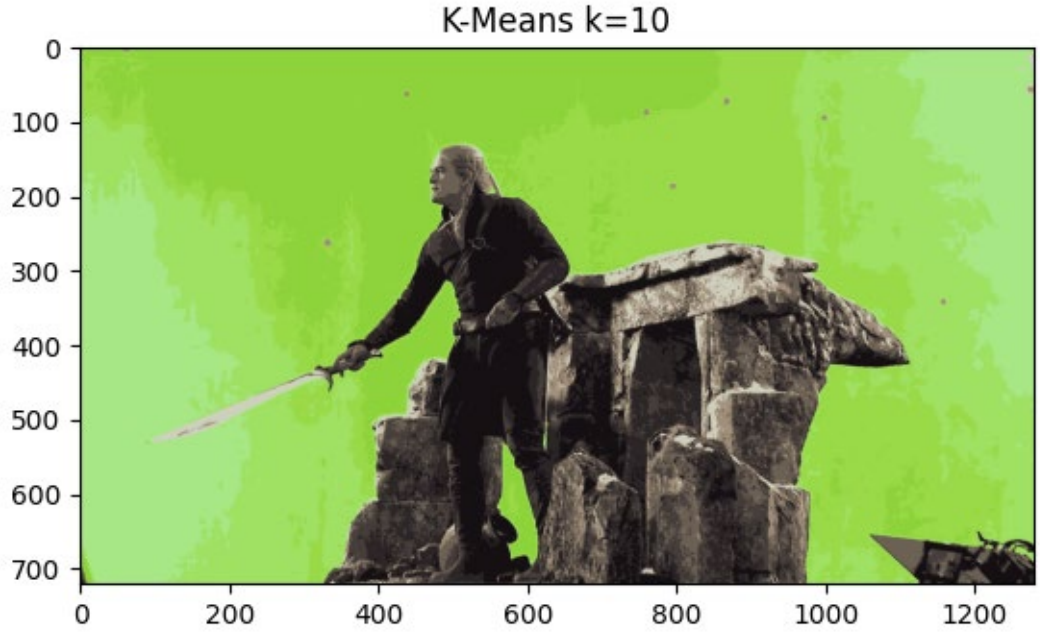


Image 2 K=10 Run 3:



Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

Image 2 K=10 Run 4:

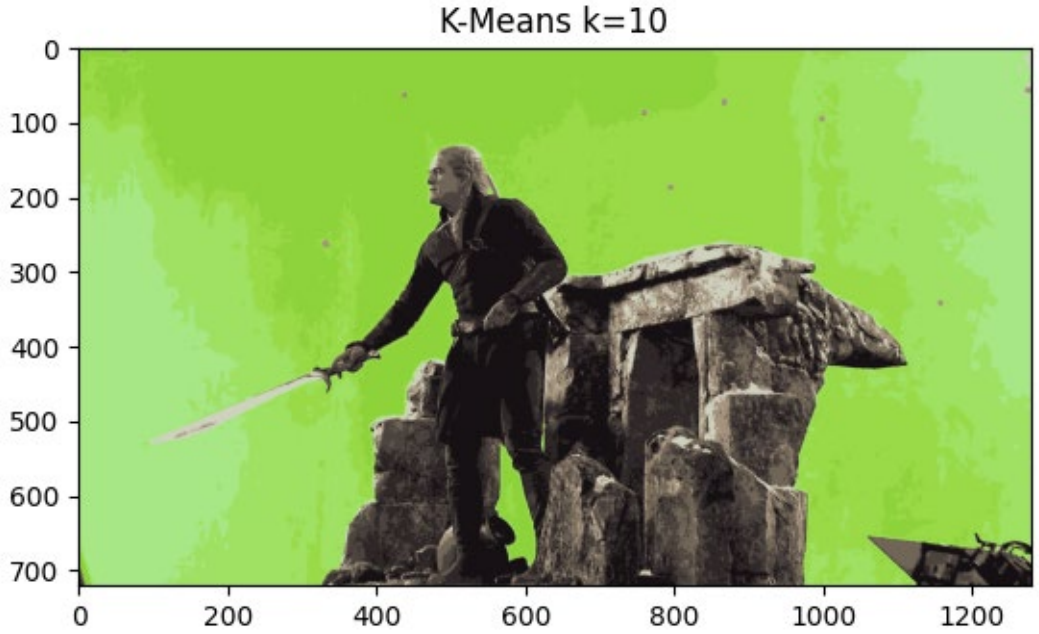
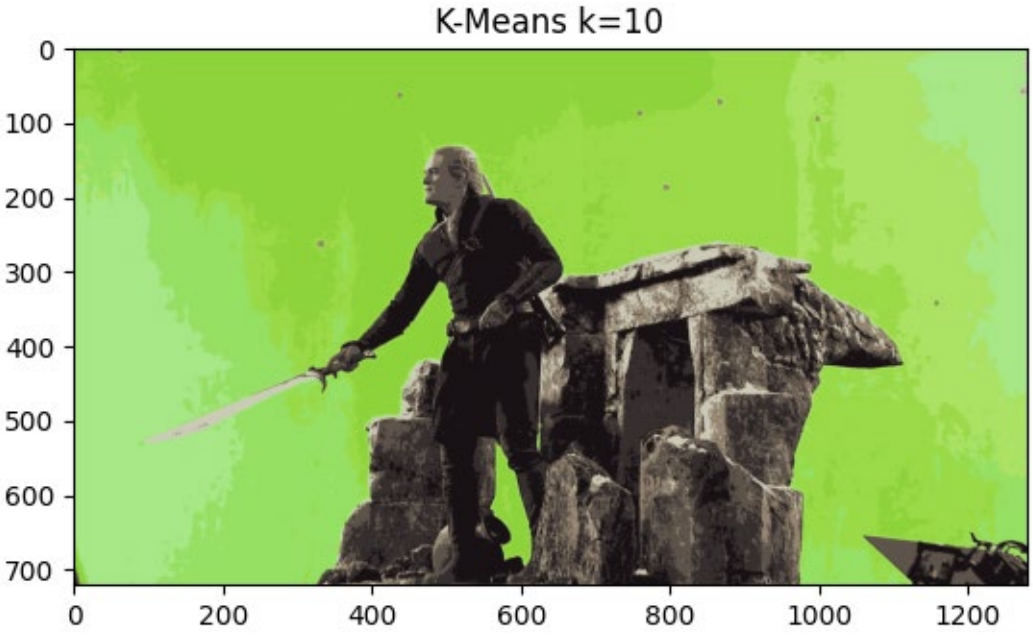


Image 2 K=10 Run 5:



Conclusion

The random initial means chosen have a large impact on the end results as you can see in the differences in SSE values and resultant graphs/images. Generally, higher the number of k, lower the SSE.

Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

Part 3

Introduction

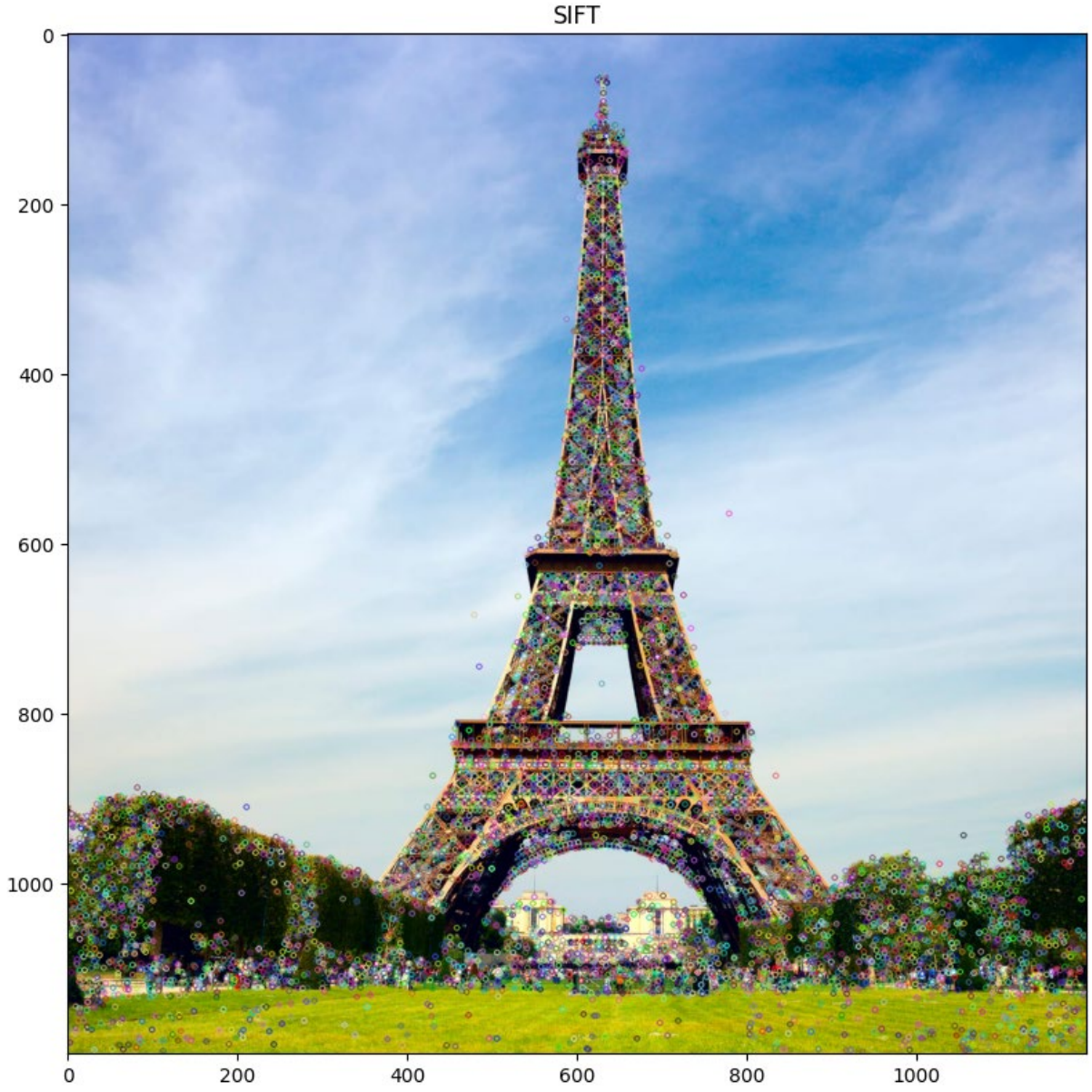
For Part 3 of the assignment, the code is in sift.py. Running the program displays the two images with keypoints and the match result. SIFT descriptor was implemented using the OpenCV library. We were given two images, SIFT1_img.jpg and SIFT2_img.jpg as our inputs. The code outputs images with keypoints marked on each of the images for the first portion of the assignment. Then, nearest-neighbor for each keypoint in image 1 to that of image 2 were calculated using minimum L2 distance between the SIFT descriptor vectors using the match( ) function. Afterwards, top 10% of the key points are kept using top_10( ) function and the matches are drawn as lines between the two images using the OpenCV library's drawMatches( ) function.

Results

I'm including original images with SIFT features as well as resized images with SIFT features for better view of the keypoints marked. Only the match between the two original images are provided.

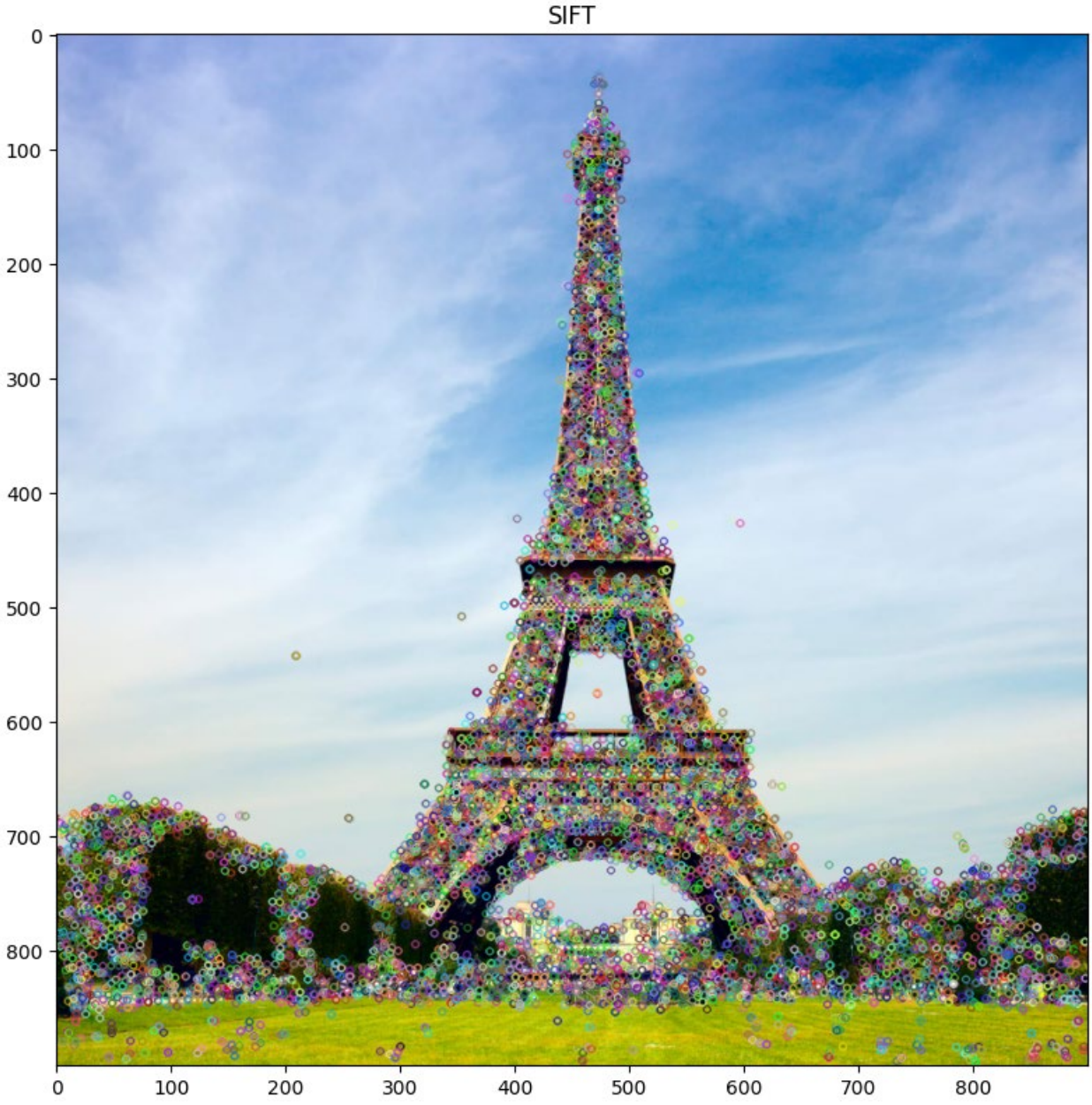
Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

All SIFT keypoints detected on original image 1:



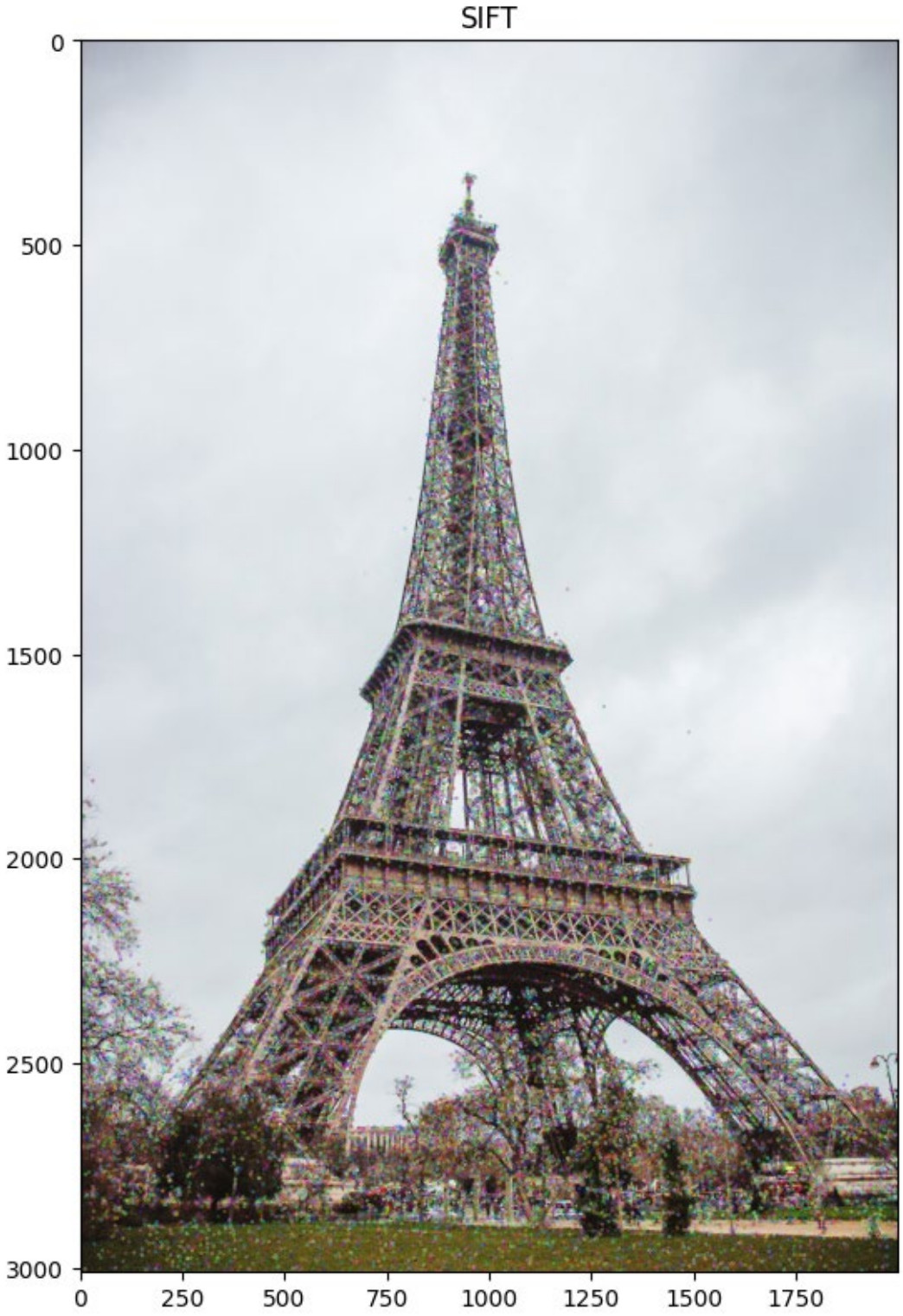
Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

All SIFT keypoints detected on resized (75%) image 1 for better view:



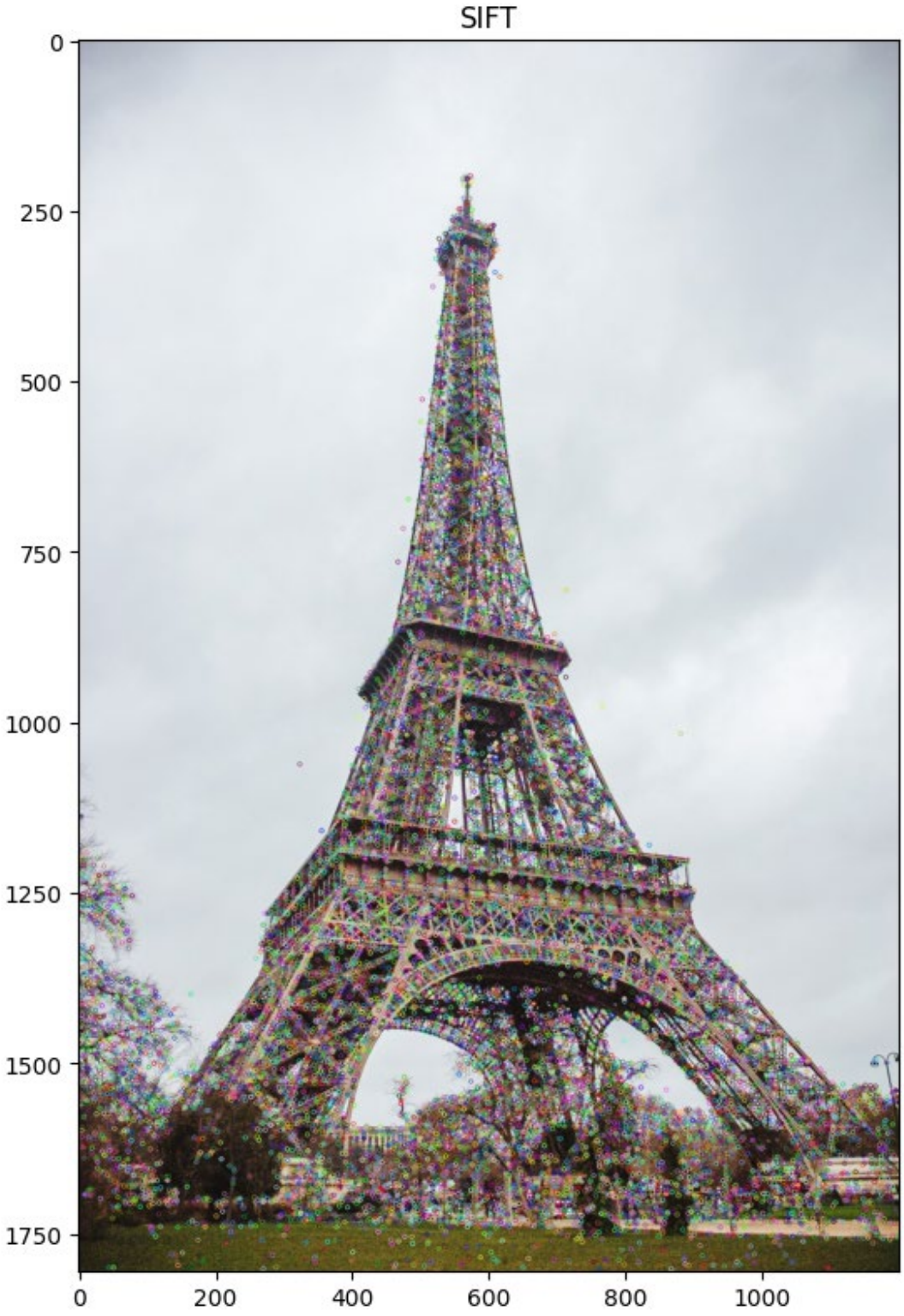
Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

All SIFT keypoints detected on original image 2:



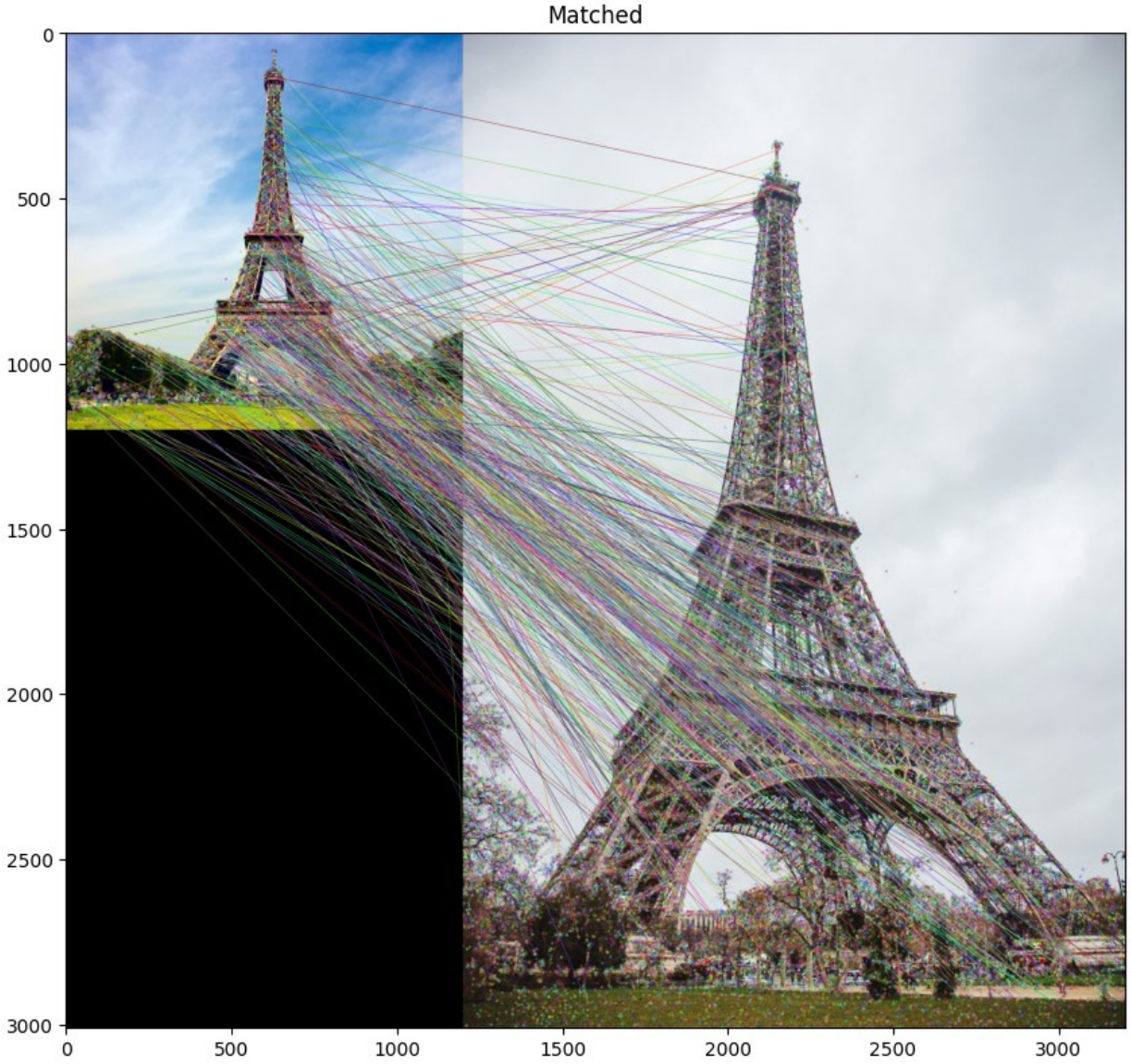
Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

All SIFT keypoints detected on resized (60%) image 2 for better view:



Cera Oh
CS510 CV and DL
Summer 2023
7/10/2023

Top 10% Keypoint Match between the two original image:



Conclusion

SIFT keypoints and its descriptors (128 element vector per keypoint) are easy to calculate and visualize using the OpenCV library. Understanding how they are related to come up with the match function took a while but ended but being easier than I thought.