

```

1 import cv2
2 import math
3 import random
4 import numpy as np
5 import time
6
7 # Mathew Seedhom
8 # CS 558
9 # Homework 2
10 # I pledge my honor that I have abided by the Stevens Honor System.
11
12 def blackBorder(pic):
13     # Creates a black border wherever neighboring pixels are different in color
14     new = pic.copy()
15     for i in range(len(pic)):
16         for j in range(len(pic[i])):
17             if i + 1 < len(pic) and j + 1 < len(pic[i]) and not np.array_equal(pic[i][j], pic[i+1][j+1]):
18                 new[i][j] = [0, 0, 0]
19             elif i + 1 < len(pic[i]) and not np.array_equal(pic[i][j], pic[i][j+1]):
20                 new[i][j] = [0, 0, 0]
21             elif i + 1 < len(pic) and not np.array_equal(pic[i][j], pic[i+1][j]):
22                 new[i][j+1] = [0, 0, 0]
23             elif i + 1 < len(pic) and not np.array_equal(pic[i][j], pic[i+1][j]):
24                 new[i][j] = [0, 0, 0]
25             elif i + 1 < len(pic[i]) and not np.array_equal(pic[i+1][j], pic[i+1][j+1]):
26                 new[i+1][j] = [0, 0, 0]
27             return new
28
29 def SLIC(pic, seg):
30     # Uses the SLIC algorithm to segment the picture into color of approximately 50*50
31     centroids = []
32     mind = []
33     clusters = []
34     for i in range(len(pic)):
35         mind += [[]]
36         clusters += [[]]
37         for j in range(len(pic[i])):
38             mind[i] += [-1]
39             clusters[i] += [0]
40             if (i - seg/2) % seg == 0 and (j - seg/2) % seg == 0 and i != 0 and j != 0:
41                 centroids += [[i, j]]
42             for k in range(len(centroids)):
43                 mindC = 250000
44                 coord = [0, 0]
45                 for i in range(-1, 1):
46                     for j in range(-1, 1):
47                         y = centroids[k][0]
48                         x = centroids[k][1]
49                         magC = (pic[i+1][x+j+1][0] - pic[i+1][x+j][0]) ** 2 + (pic[i+1][x+j+1][1] - pic[i+1][x+j][1]) ** 2 + (pic[i+1][x+j+1][2] - pic[i+1][x+j][2]) ** 2
50                         if magC < mindC:

```

```
50 ..... minC = magC
51 ..... coord = [y+i, x+j]
52 ..... centroids[k] = coord
53 num = len(centroids)
54 centers = []
55 for i in range(num):
56 ..... centers += [list(pic[centroids[i][0]][centroids[i][1]])]
57 ..... centers[i] += [centroids[i][0], centroids[i][1]]
58 change = 1
59 while True:
60 ..... start = time.time()
61 ..... centersAvg = []
62 ..... for i in range(num):
63 ..... ..... centersAvg += [[0, 0, 0, 0], 0]
64 ..... end = time.time()
65 ..... print("\SLIC Init %(change)i: %(time)f s"%(change):change, "time": end - start))
66 ..... start = time.time()
67 ..... for k in range(num):
68 ..... ..... for i in range(-seg, seg):
69 ..... ..... for j in range(-seg, seg):
70 ..... ..... y = int(centers[k][3] + i)
71 ..... ..... x = int(centers[k][4] + j)
72 ..... ..... if 0 < y < len(pic) and 0 < x < len(pic[i]):
73 ..... ..... D = ((pic[y][x][0])-(centers[k][0])) ** 2 + ((pic[y][x][1])-(centers[k][1])) ** 2 + ((pic[y][x][2])-(centers[k][2])) ** 2 + ((y - centers[k][4]) ** 2) * 2*seg/4
74 ..... ..... if D < minD[y][x] or minD[y][x] == -1:
75 ..... ..... minD[y][x] = D
76 ..... ..... clusters[y][x] = k
77 ..... end = time.time()
78 ..... print("\SLIC %(change)i: %(time)f s"%(change): change, "time":end - start))
79 ..... for i in range(len(clusters)):
80 ..... ..... for j in range(len(clusters[i])):
81 ..... ..... quintet = centersAvg[clusters[i][j]][0]
82 ..... ..... centersAvg[clusters[i][j]][0] = [quintet[0] + pic[i][j][0], quintet[1] + pic[i][j][1], quintet[2] + pic[i][j][2], quintet[3] + i, quintet[4] + j]
83 ..... ..... centersAvg[clusters[i][j]][1] += 1
84 ..... for i in range(num):
85 ..... ..... centersAvg[i] = list(map(lambda x: x / centersAvg[i][1], centersAvg[i][0]))
86 ..... ..... centersAvg[i] = list(map(np.floor, centersAvg[i]))
87 ..... if centersAvg == centers:
88 ..... ..... break
89 ..... else:
90 ..... ..... change += 1
91 ..... centers = centersAvg
92 ..... for i in range(len(pic)):
93 ..... ..... for j in range(len(pic[i])):
94 ..... ..... pic[i][j] = [centers[clusters[i][j]][0], centers[clusters[i][j]][1], centers[clusters[i][j]][2]]
95 ..... return pic
96
97 def kMeans(pic, num):
98 ..... # Uses the kMeans algorithm to segment all global colors into a specified number
```

```

99 start = time.time()
100 centers = []
101 cx = random.sample(range(len(pic)), num)
102 cy = random.sample(range(len(pic[0])), num)
103 end = time.time()
104 print("Init: %(time)f s"%(time): end - start)
105 start = time.time()
106 for i in range(num):
107     centers += [list(pic[cx[i]][cy[i]])]
108     change = 1
109     end = time.time()
110     print("Init2: %(time)f s"%(time): end - start)
111     print("Height: %(h)i, Width: %(w)i"%(h): len(pic), "w": len(pic[0]))
112     while True:
113         start = time.time()
114         centersAvg = []
115         clusters = []
116         for i in range(num):
117             centersAvg += [[0, 0, 0], 0]
118             clusters += [[]]
119         end = time.time()
120         print("Means Init %(change)i: %(time)f s"%(change): change, "time": end - start)
121         start = time.time()
122         for i in range(len(pic)):
123             for j in range(len(pic[i])):
124                 minD = 25000
125                 curr = num+1
126                 for k in range(num):
127                     colorD = ((pic[i][j][0])-(centers[k][0]))**2 + ((pic[i][j][1])-(centers[k][1]))**2 + ((pic[i][j][2])-(centers[k][2]))**2
128                     if colorD < minD:
129                         minD = colorD
130                         curr = k
131                 clusters[curr] += [[i, j]]
132                 centersAvg[curr][0] += (pic[i][j][0])
133                 centersAvg[curr][0][1] += (pic[i][j][1])
134                 centersAvg[curr][0][2] += (pic[i][j][2])
135                 centersAvg[curr][1] += 1
136             end = time.time()
137             print("Means %(change)i: %(time)f s"%(change): change, "time": end - start)
138             for i in range(num):
139                 centersAvg[i] = list(map(lambda x: x / centersAvg[i][1], centersAvg[i][0]))
140                 centersAvg[i] = list(map(np.floor, centersAvg[i]))
141             if centersAvg == centers:
142                 break
143             else:
144                 change += 1
145         centers = centersAvg
146         print(centers)
147         for i in range(num):

```

```
148     ....|....for j in range(len(clusters[i])):
149     ....|....|....pic[clusters[i][j][0]][clusters[i][j][1]] = centers[i]
150     ....return pic
151
152     .....
153
154 if __name__ == "__main__":
155     ....print("Working on it!")
156     ....# pic1 = cv2.imread("white-tower.png", 1)
157     ....# kSeg = kMeans(pic1.astype(int), 2)
158     ....# cv2.imwrite("2Means.png", kSeg)
159     ....pic2 = cv2.imread("wt_slic.png", 1)
160     ....slic = (SLIC(pic2.astype(int)))
161     ....cv2.imwrite("SLIC-orig.png", slic)
162     ....print("All done!")
```