

Computer Vision — Homework 2

B05902118 陳盈如

September 29, 2020

1 A binary image (threshold at 128)

```
9  # binarize lena.bmp at 128 to get a binary image
10 binarize = np.zeros(image.shape, int)
11 for i in range(image_rows):
12     for j in range(image_cols):
13         if image[i][j] < 128:
14             binarize[i][j] = 0
15         else:
16             binarize[i][j] = 255
17 cv2.imwrite('binarize.jpg', binarize)
```

Actually, the picture is already grayscale, so we don't need to transfer the color into grayscale again. Then, identify the value. If it is less than 128, change it to black (pixel = 0), else, change it to white (pixel = 255).



(a) Original

(b) Result

Figure 1: binarize at threshold 128

2 A histogram

```
19 # draw a histogram
20 histogram = np.zeros(256, int)
21 index = np.arange(256)
22 for i in range(image_rows):
23     for j in range(image_cols):
24         histogram[image[i][j]] += 1
25 plt.bar(index, histogram)
26 plt.ylabel("Counts")
27 plt.xlabel('Gray Level')
28 plt.title('HISTOGRAM OF LENA.BMP')
29 plt.show()
```

Create an array and make it size 256. This array represents the number of each gray value from 0 to 255 on *lena.bmp*. Check each pixel's gray value on *lena.bmp* and make that gray value's count +1.

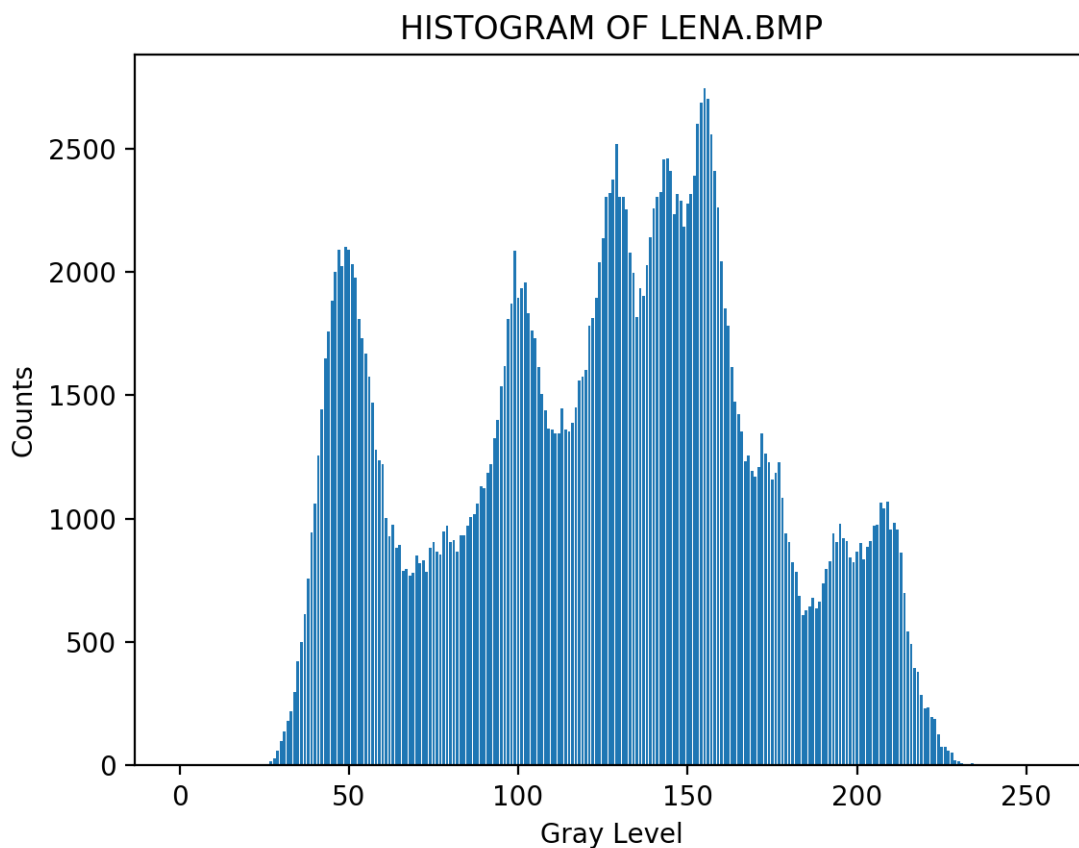


Figure 2: Histogram

3 Connected components (regions with + at centroid, bounding box)

```

40 for i in range(image3_rows):
41     row_counts = 0
42     startCOL = 0
43     endCOL = 0
44     for j in range(image3_cols):
45
46         if image3[i][j] > 128: # pixel 0 黑色, pixel != 0 白色
47             if in_run == 0:
48                 in_run = 1
49                 startCOL = j
50                 row_counts += 1
51             if j == image3_cols - 1:
52                 in_run = 0
53                 endCOL = j
54                 run_data.append([i, startCOL, endCOL, 0, endCOL - startCOL + 1])
55             else:
56                 if in_run != 0:
57                     in_run = 0
58                     endCOL = j - 1
59                     run_data.append([i, startCOL, endCOL, 0, endCOL - startCOL + 1])
60
61     if row_counts == 0:
62         counts_inRow.append([0, 0])
63     else:
64         length = len(run_data)
65         counts_inRow.append([length - row_counts + 1, length])

```

The first loop is responsible for recording two lists, *counts_inRow* and *run_data*.

counts_inRow: [0]: start of the run, [1]: end of the run

run_data: [0]: this run on which row, [1]: this run starting from which column, [2]: ending from which column, [3]: label, [4]: the number of pixels in this run

```

70 for L in range(image3_rows):
71     Start_inL = counts_inRow[L][0]
72     End_inL = counts_inRow[L][1]
73     if L == 0:
74         preStart_inL = 0
75         preEnd_inL = 0
76     else:
77         preStart_inL = counts_inRow[L - 1][0]
78         preEnd_inL = counts_inRow[L - 1][1]
79     while (Start_inL <= End_inL and preStart_inL <= preEnd_inL and preStart_inL != 0):
80
81         if run_data[Start_inL - 1][2] < run_data[preStart_inL - 1][1]:
82             Start_inL = 1
83             elif run_data[preStart_inL - 1][2] < run_data[Start_inL - 1][1]:
84                 preStart_inL += 1
85             else:
86                 PLabel = run_data[Start_inL - 1][3]
87                 preLabel = run_data[preStart_inL - 1][3]
88                 if PLabel == 0:
89                     run_data[Start_inL - 1][3] = preLabel
90                     label[preLabel].append(Start_inL - 1)
91                     label[PLabel].append(preStart_inL - 1)
92                 elif PLabel != 0 and preLabel != PLabel:
93
94                     if PLabel < preLabel:
95                         run_data[preStart_inL - 1][3] = PLabel
96                         label[preLabel].remove(preStart_inL - 1)
97                         label[PLabel].append(preStart_inL - 1)
98                     else:
99                         run_data[Start_inL - 1][3] = preLabel
100                         label[PLabel].remove(Start_inL - 1)
101                         label[preLabel].append(Start_inL - 1)
102                 preStart_inL += 1
103             elif run_data[preStart_inL - 1][2] > run_data[Start_inL - 1][2]:
104                 Start_inL += 1
105             elif run_data[preStart_inL - 1][2] == run_data[Start_inL - 1][2]:
106                 preStart_inL += 1
107             Start_inL += 1
108     Start_inL = counts_inRow[L][0]
109     while (Start_inL <= End_inL):
110         PLabel = run_data[Start_inL - 1][3]
111         LenOfLabel = len(label)
112         if PLabel == 0:
113             run_data[Start_inL - 1][3] = LenOfLabel
114             label.append([Start_inL - 1])
115             Start_inL += 1

```

(a) Top-down pass

```

118 for L in range(image3_rows - 1, -1, -1):
119     Start_inL = counts_inRow[L][0]
120     End_inL = counts_inRow[L][1]
121     if L == image3_rows - 1:
122         preStart_inL = 0
123         preEnd_inL = 0
124     else:
125         preStart_inL = counts_inRow[L + 1][0]
126         preEnd_inL = counts_inRow[L + 1][1]
127     while (Start_inL <= End_inL and preStart_inL <= preEnd_inL and preStart_inL != 0):
128
129         if run_data[Start_inL - 1][2] < run_data[preStart_inL - 1][1]:
130             Start_inL = 1
131             elif run_data[preStart_inL - 1][2] < run_data[Start_inL - 1][1]:
132                 preStart_inL += 1
133             else:
134                 PLabel = run_data[Start_inL - 1][3]
135                 preLabel = run_data[preStart_inL - 1][3]
136                 if PLabel != preLabel:
137                     if PLabel < preLabel:
138                         Len_of_Label = len(label[preLabel])
139                         for i in range(Len_of_Label):
140                             run_num = label[preLabel][i]
141                             run_data[run_num][3] = PLabel
142                             label[PLabel] += label[preLabel]
143                             for i in range(Len_of_Label):
144                                 label[preLabel].pop()
145                     else:
146                         Len_of_Label = len(label[PLabel])
147                         for i in range(Len_of_Label):
148                             run_num = label[PLabel][i]
149                             run_data[run_num][3] = preLabel
150                             label[preLabel] += label[PLabel]
151                             for i in range(Len_of_Label):
152                                 label[PLabel].pop()
153                 if run_data[Start_inL - 1][2] > run_data[preStart_inL - 1][2]:
154                     preStart_inL += 1
155                 elif run_data[preStart_inL - 1][2] > run_data[Start_inL - 1][2]:
156                     Start_inL += 1
157                 elif run_data[preStart_inL - 1][2] == run_data[Start_inL - 1][2]:
158                     preStart_inL += 1
159                     Start_inL += 1
160     Start_inL = counts_inRow[L][0]

```

(b) Bottom-up pass

Figure 3: Run Length Implementation

Implement the third method on PowerPoint – Run Length Implementation. Maintain *label* list at the same time when tracing top-down pass and bottom-up pass steps.

label: Its index represents label number, and content represents the orders of the runs, which are in this label. For example: *label*=[[*nothing*], [0, 2, 3], [1]]. It means run #0, run #2, run #3 have the same label, label#1, and run #1 has label#2.

```

165 for i in range(1, Len):
166     pixel = 0
167     if len(label[i]) == 0:
168         right_pos.append([])
169         left_pos.append([])
170     for j in range(len(label[i])):
171         pixel += run_data[label[i][j]][4]
172         if j == 0:
173             right_pos.append(run_data[label[i][j]][2])
174             left_pos.append(run_data[label[i][j]][1])
175         else:
176             if right_pos[i] < run_data[label[i][j]][2]:
177                 right_pos[i] = run_data[label[i][j]][2]
178             if left_pos[i] > run_data[label[i][j]][1]:
179                 left_pos[i] = run_data[label[i][j]][1]
180
181     len_pixel.append(pixel)
182     count = 0
183     final_image = cv2.imread('binarize.jpg')
184     for i in range(Len):
185         label[i].sort()
186         Length = len(label[i])
187         x_cen = 0
188         y_cen = 0
189         if (len_pixel[i] >= 500):
190             count += 1
191             x = left_pos[i]
192             x_end = right_pos[i]
193             y = run_data[label[i][0]][0]
194             y_end = run_data[label[i][Length - 1]][0]
195             cv2.rectangle(final_image, (x, y), (x_end, y_end), (255, 255, 0), 1)
196             for j in range(Length):
197                 run_order = label[i][j]
198                 x_cen_plus = ((run_data[run_order][1] + run_data[run_order][2]) / 2) * run_data[run_order][4]
199                 y_cen_plus = run_data[run_order][0] * run_data[run_order][4]
200                 x_cen += x_cen_plus
201                 y_cen += y_cen_plus
202             x_cen /= len_pixel[i]
203             y_cen /= len_pixel[i]
204             x_cen = int(x_cen)
205             y_cen = int(y_cen)
206             cv2.line(final_image, (x_cen - 6, y_cen), (x_cen + 6, y_cen), (0, 0, 255), thickness=2)
207             cv2.line(final_image, (x_cen, y_cen - 6), (x_cen, y_cen + 6), (0, 0, 255), thickness=2)

```

Threshold at 500, figure out the label whose area is larger than 500, use a bounding box to frame the smallest area of it, and mark the centroid of this area.



Figure 4: Result