National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

EE433: Digital Image Processing

Class: BSCS 7AB

Lab 3: Image histograms,Thresholding and connected component Analysis

Date: 3rd February, 2020

Time: 9.00Am to 12.00Pm & 2.00Pm to 5.00Pm

Instructor: Dr. Asif Ali

Lab3: Image histograms, thresholding and connected component analysis.

Objective:

The objective of this lab is:

- To understand image histograms as an analysis tool.
- Perform basic thresholding techniques to isolate background from foreground in simple images.
- To perform connected component labeling of a binary image.

Theory:

Histogram Processing:

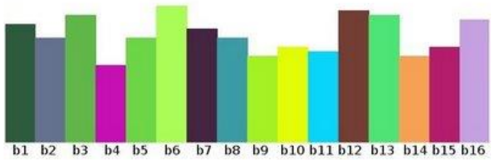
Histograms are collected counts of data organized into a set of predefined bins. When we say data we are not restricting it to be intensity values. The data collected can be whatever feature you find useful to describe your image. Let's see an example. Imagine that a Matrix contains information of an image (i.e. intensity in the range 0-255):

254	143	203	176	109	229	177	220	192	9	229	142	138	64	0	63	28	8	88	82
27	68	231	75	141	107	149	210	13	239	141	35	68	242	110	208	244	0	33	88
54	42	17	215	230	254	47	41	98	180	55	253	235	47	122	208	78	110	152	100
9	186	192	71	104	193	88	171	37	233	18	147	174	1	143	211	176	188	192	68
179	20	238	192	190	132	41	248	22	134	83	133	110	254	176	238	188	234	51	204
232	25	0	183	174	129	61	30	110	189	0	173	197	183	153	43	22	87	68	118
235	35	151	185	129	81	239	170	195	94	38	21	67	101	58	37	196	149	52	154
155	242	54	0	104	109	169	47	130	254	225	156	31	181	121	15	128	35	252	205
223	114	79	129	147	6	201	68	89	107	58	44	253	84	38	1	62	5	231	218
55	188	237	188	80	101	131	241	68	133	124	151	111	28	190	4	240	78	117	145
152	155	229	78	90	217	219	105	116	77	38	49	2	9	214	181	205	116	135	33
182	94	176	199	20	149	57	223	232	113	32	45	177	15	31	179	100	119	208	81
224	118	124	172	75	29	69	180	187	195	41	44	8	170	158	101	131	31	28	112
238	83	38	7	83	69	173	183	98	237	67	227	18	218	248	237	75	192	201	146
88	195	224	207	140	22	31	118	234	34	182	116	23	47	68	242	169	152	110	248
140	37	101	230	246	145	122	64	27	58	229	1	225	143	91	100	98	90	40	195
251	4	178	139	121	95	97	174	249	182	77	115	223	188	182	82	65	252	83	198
179	180	223	230	87	182	148	78	176	19	17	4	184	176	183	102	83	81	132	206
173	137	185	242	181	181	214	49	74	238	197	37	98	102	15	217	148	8	102	188
85	9	17	222	18	210	70	21	78	241	184	216	93	93	208	102	153	212	119	47

What happens if we want to count this data in an organized way? Since we know that the range of information value for this case is 256 values, we can segment our range in sub-parts (called bins) like:

$$[0, 255] = [0, 15] \cup [16, 31] \cup .... \cup [240, 255]$$
$$range = bin_1 \cup bin_2 \cup .... \cup bin_{n=15}$$

and we can keep count of the number of pixels that fall in the range of each bin_{i}. Applying this to the example above we get the image below (axis x represents the bins and axis y the number of pixels in each of them).



A histogram of an image can convey vital information and can be used to enhance quality of the image, apply threshold operation, linear and piece-wise transformations etc. For instance take a look at following images and their histograms:

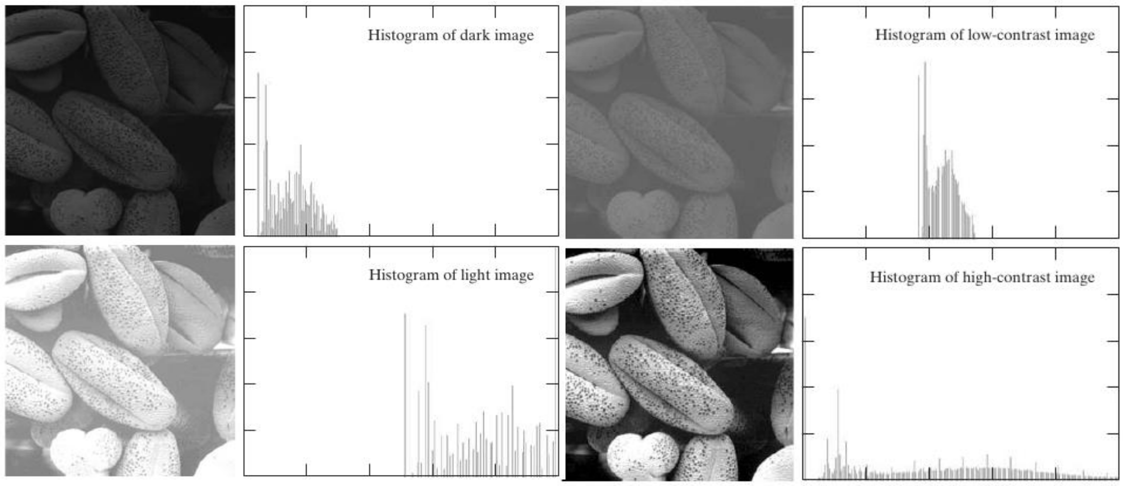


Figure 2: Four basic image types and their respective histograms.

You can notice that with the help of histograms, you can classify images in four basic image types.

Threshold:

Thresholding is a simplest method to segment an image. In the simplest form (a.k.a. *binary thresholding*), an threshold **T** is applied to a histogram and all gray level intensities above **T** are set to *high_level* (usually 255) and lower or equal inensities are set to *low_value* (usually 0). There exists various versions of this basic thresholding technique which can be selected as per requirement. Have a look at [OpenCV's thresholding](#).

Connected Component Analysis or Labeling enables us to detect different objects from a binary image. Once different objects have been detected, we can perform a number of operations on them: from counting the number of total objects to counting the number of objects that are similar, from finding out the biggest object of the bunch to finding out the smallest and from finding out the closest pair of objects to finding out the farthest etc. Connected Component labeling procedure is as follows:

- Process the image from left to right, top to bottom:

If the next pixel to process is 1

i.) If only one of its neighbors (top or left) is 1, copy its label.

ii.) If both are 1 and have the same label, copy it.

iii.) If they have different labels

– Copy the label from the left.

– Update the equivalence table.

iv.) Otherwise, assign a new label.

- Re-label with the smallest of equivalent labels

Algorithm 1: simplified two pass algorithm.

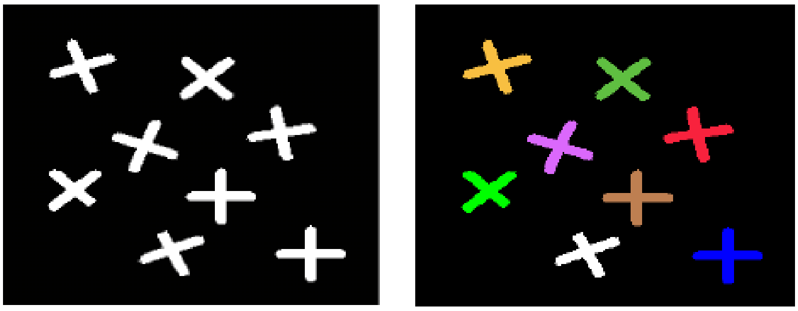


Figure 3: Labeling of each connected component to a different color (for visualization only)

Lab Tasks:

Task 1:

1. Use python notebook and opencv's imread function and load an image (e.g. dark.tif)
2. Calculate the histogram of the image manually i.e. traverse the entire image and calculate the number of pixels corresponding to each intensity value.
3. Calculate the histogram of the image using cv2.calcHist.
4. Plot the histogram of the image and observe/validate the aforesaid information in the theory part.
5. Save the histogram plot for later viewing and comparison.
6. Repeat all steps 1-4 for all provided images.

Task 2:

- Read “coins.jpg” image in python notebook and store that in “img” variable.
- Plot histogram of that image.
- Identify which threshold value would be suitable to isolate black background from foreground objects.
- Use that threshold value to create a binary image which contains foreground pixels as 255 and background pixels as 0 and save that binary image as “mask” image
- Apply logical OR operator between “mask” and “img” variable.
- You have just created a very basic image segmentation technique!
- Apply the same steps to “bookpage.jpg” and visualize how hard it is to isolate text from the page!

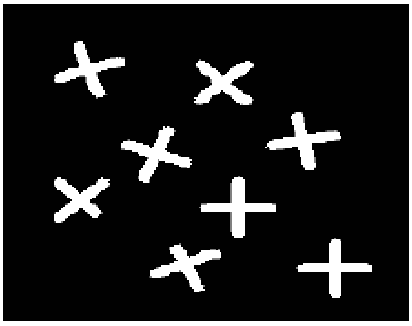
Task 3:

- Read “cc.png” from directory.
- Apply connected component labelling using 4 connectivity and count total number of objects in the list. (**which means write your own function from scratch!**)

    - HINT1: the image is a binary image with background (black portion) has numeric value of 1 while the white objects have numeric values of 255
    - HINT2: you can use **two-pass algorithm** from the Algorithm 1 (which doesn't use a specialized data structure and simply use two for loops!)
- Use Opencv's **cv2.connectedComponents** function and admire the underlying workings.

Note: - Before getting to test your code on an image it is highly recommended to test it on a 2d-array. It will help you in following ways:-

- As a good measure of cross check i.e. you can quickly figure out which condition is going wrong because you have a standard output to compare it with and you can apply corrective measures where you find conditions going wrong.
- Debugging is easier with a smaller size 2d-arrray than a huge size image.



<u>Submission Guidelines:-</u>

<u>Deliverables:</u>
You have to submit a word document containing all the codes and screen shots of output of all codes. Do not submit any zip folder containing code and screenshots separately.
<u>Penalty for copying code from internet:-</u>
You are strictly not permitted to copy code from internet because this lab will be evaluated on the base of quiz/viva (in next lab) so anyone who has poor understanding of procedure overall will get a penalty in lab submission part as well.