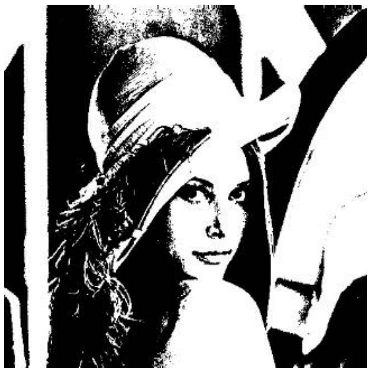
Computer Vision Homework 2

B07902078 資工三 沈韋辰

Language: Python3.8.3

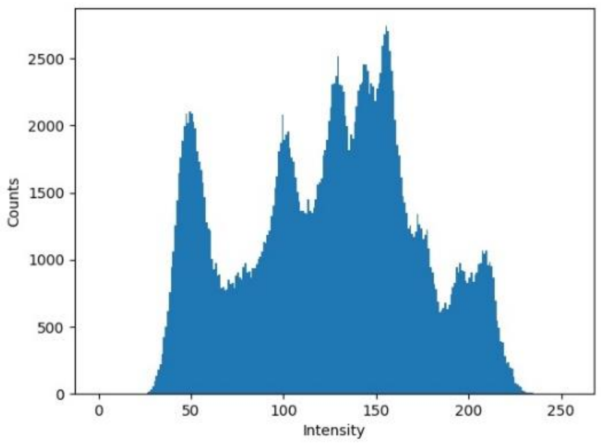
Modules: OpenCV(cv2), numPy, matplotlib.pyplot

(a) Generate a binary image (threshold at 128):



Algorithm: Traversing all the pixel. If its intensity is higher than or equal to 128, then raise it to 255 (brightest); otherwise, reduce it to 0 (darkest).

(b) Generate a histogram:



Algorithm: Accumulating the pixels with the same intensity and draw it by the module *matplotlib.pyplot*

(c) Find the connected components with centroid and bounding box.

In my method, I divide this problem to four steps:

Step.1

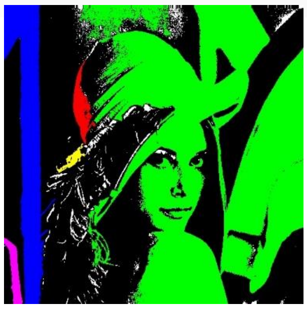
We need to turn lena.bmp into binary image. Just take the result of problem (a), let the intensity 255 be ‘1’ and 0 be ‘0’.

Step.2

In this homework, I adopt an algorithm which is similar to **flooding algorithm** with **4-connected** because I think the algorithms taught in class are not such intuitive and a little complicated. This method traverses all the pixels, if it hasn't been labeled, then we put a barrel of "water" (label) on it and it will flood to 4 directions of its neighbors. If the neighbor has been labeled, the flood stops there; otherwise, we label the neighbor and continue the flood. We use a queue to maintain whom is the next one to be labeled and another list to record if the scale of this flood is over 500 pixels. This algorithm takes **O(RC)** time. (for size of image is RxC.) Each pixel just needs to check status, check neighbor status and labeled, so I think it is still a good way just like the algorithms taught in the lecture but easier to realize.

Step.3

After finishing labeling, we traverse the image again for measuring the boundary and centroid of each component. By the way, I also color a demo image to make sure if the algorithm worked.



Step.4

Finally, draw the boundary boxes and centroids by *cv2.rectangle and cv2.circle*.