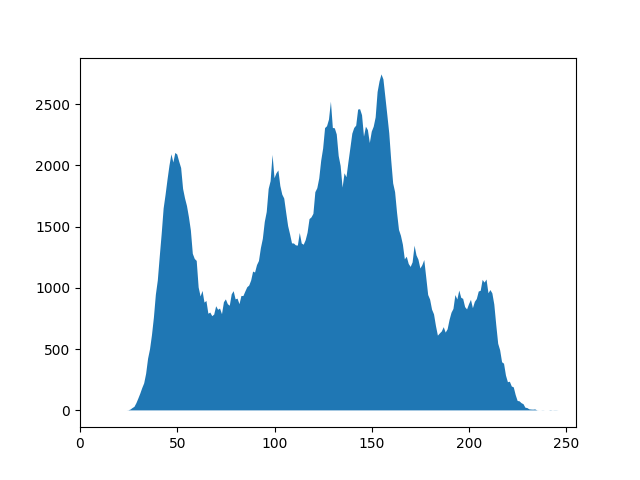
Homework 2

R13525009 羅筠笙

1. A binary image (threshold at 128)

Description: Set every pixel to binary with threshold at 128.

1. # Q1
2. img\_arr\_binary = img\_arr.copy()
3. for row in range(img\_size\_0):
4. for col in range(img\_size\_1):
5. if img\_arr\_binary[row, col] >= 128:
6. img\_arr\_binary[row, col] = 255
7. else:
8. img\_arr\_binary[row, col] = 0
9. A histogram

Description: Compute the frequency of each pixel intensity value (0–255) by iterating through the image, then store the counts in an array to plot a histogram that visualizes the distribution of pixel intensities.

1. # Q2
2. histogram = np.zeros(256)
3. for row in range(img\_size\_0):
4. for col in range(img\_size\_1):
5. histogram[img\_arr[row, col]] += 1
6. plt.fill(histogram)
7. plt.xlim(0, 255)
8. plt.ylim(0, max(histogram))
9. Connected components (regions with + at centroid, bounding box)

Description: Utilize Depth-First Search (DFS) to identify connected components within the binary image. For each component, neighbors are added using 4-connectivity (up, down, left, right).Filter out regions smaller than 500 pixels, and for the remaining regions, draw bounding boxes and mark the centroids

1. # Q3
2. # Find connected components using DFS
3. for i in range(img\_size\_0):
4. for j in range(img\_size\_1):
5. if img\_arr\_ccl[i, j] == 1:
6. # Initialize region properties
7. up = i
8. bottom = i
9. left = j
10. right = j
11. area = 1
12. stack = [(i, j)] # For DFS to track
13. rows = i
14. cols = j
15. while stack:
16. i1, j1 = stack.pop()
17. # Update the current pixel index
18. img\_arr\_ccl[i1, j1] = idx
19. # Update the region bounaries
20. up = min(up, i1)
21. bottom = max(bottom, i1)
22. left = min(left, j1)
23. right = max(right, j1)
24. area += 1
25. rows += i1
26. cols += j1
27. # Update and push valid neighbors (4-connectivity) to the stack
28. for x, y in [(i1-1, j1), (i1+1, j1), (i1, j1-1), (i1, j1+1)]: # up, bottom, left, right
29. if 0 <= x < img\_size\_0 and 0 <= y < img\_size\_1 and img\_arr\_ccl[x, y] == 1:
30. stack.append((x, y))
31. # Store region info (centroid row, centroid column, up, bottom, left, right, area)
32. region[idx] = (rows // area, cols // area, up, bottom, left, right, area)
33. # Update the idx to next region
34. idx += 1
35. # Filter regions with area greater than 500 pixels
36. answer = [val for val in region.values() if val[6] > 500]
37. img\_c = np.stack([img\_arr\_binary] \* 3, axis=-1).astype(np.uint8)
38. # Draw the bounding boxes and centroids
39. for i in answer:
40. cv2.rectangle(img\_c, (i[4], i[2]), (i[5], i[3]), (0, 0, 255), 3)
41. cv2.circle(img\_c, (i[1], i[0]), 5, (255, 0, 0), 3)