Homework 2

R13525009 羅筠筌

a. A binary image (threshold at 128)

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

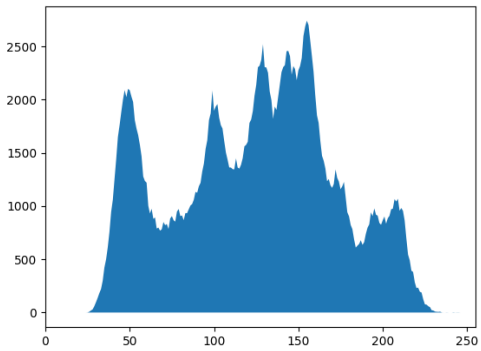
```python
# Q1
img_arr_binary = img_arr.copy()

for row in range(img_size_0):
    for col in range(img_size_1):
        if img_arr_binary[row, col] >= 128:
            img_arr_binary[row, col] = 255
        else:
            img_arr_binary[row, col] = 0

```

b. 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.




```python
# Q2
histogram = np.zeros(256)

for row in range(img_size_0):
    for col in range(img_size_1):
        histogram[img_arr[row, col]] += 1

plt.fill(histogram)
plt.xlim(0, 255)
plt.ylim(0, max(histogram))
```

c. 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

```
# Q3
# Find connected components using DFS
for i in range(img_size_0):
    for j in range(img_size_1):
        if img_arr_ccl[i, j] == 1:
            # Initialize region properties
            up = i
            bottom = i
            left = j
            right = j
            area = 1
            stack = [(i, j)]     # For DFS to track
            rows = i
            cols = j
            while stack:
                i1, j1 = stack.pop()
                # Update the current pixel index
                img_arr_ccl[i1, j1] = idx
                # Update the region bounaries
                up = min(up, i1)
                bottom = max(bottom, i1)
                left = min(left, j1)
                right = max(right, j1)
                area += 1
                rows += i1
                cols += j1
                # Update and push valid neighbors (4-connectivity) to the stack
                for x, y in [(i1-1, j1), (i1+1, j1), (i1, j1-1), (i1, j1+1)]:    # up, bottom, left, right
                    if 0 <= x < img_size_0 and 0 <= y < img_size_1 and img_arr_ccl[x, y] == 1:
                        stack.append((x, y))
            # Store region info (centroid row, centroid column, up, bottom, left, right, area)
            region[idx] = (rows // area, cols // area, up, bottom, left, right, area)
            # Update the idx to next region
            idx += 1

# Filter regions with area greater than 500 pixels
answer = [val for val in region.values() if val[6] > 500]

```

```
img_c = np.stack([img_arr_binary] * 3, axis=-1).astype(np.uint8)

# Draw the bounding boxes and centroids
for i in answer:
    cv2.rectangle(img_c, (i[4], i[2]), (i[5], i[3]), (0, 0, 255), 3)
    cv2.circle(img_c, (i[1], i[0]), 5, (255, 0, 0), 3)
```