CV HW2

R11922196 林佑鑫

(a) A binary image (threshold at 128)

Brief description, algorithm: 對影像中每個 pixel，若 pixel 數值大於等於 128，則在新圖上同 位置值爲 255，反之爲 0。
Parameters: None
Principal code fragment:

```
5   def binarize(img):
6       height, width = img.shape
7       binarize_img = np.zeros_like(img)
8
9       for h in range(height):
10          for w in range(width):
11              if((img[h, w] >= 128)):
12                  binarize_img[h, w] = 255
13              else:
14                  binarize_img[h, w] = 0
15      return binarize_img
```

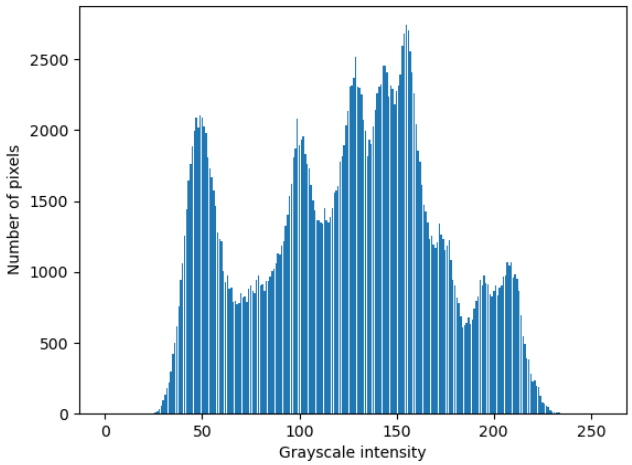
Resulting image:

(b) A histogram

Brief description, algorithm: iterate through 每個 pixels，並將各個 pixel 的值記錄在 256 維的 list 中，例如(1, 1)值爲 136，則 count[136]加一，最後畫出 count list。
Parameters: None
Principal code fragment:

```python
def plot_histogram(img):
    count = np.zeros(256, dtype=int)
    height, width = img.shape
    for h in range(height):
        for w in range(width):
            count[img[h, w]] += 1
    plt.bar(range(256), count)
    plt.xlabel('Grayscale intensity')
    plt.ylabel('Number of pixels')
    plt.savefig('b.png')
```

Resulting image:



(c) Connected components

Brief description, algorithm: 主要爲實作 Chapter 2 PPT 中第 36 頁的 iterative algorithm pseudocode。完成後以 mask 篩取各個 label 值，並以 500 個 pixels 爲 threshold，畫出大於 500 個 pixels 的 connected components 的 BBox, 重心。
Parameters: None
Principal code fragment:

```python
def plot_connected_components(img):
    # The iterative algorithm
    height, width = img.shape
    label = np.zeros((height + 2, width + 2), int)

    # Init
    tmp = 1
    for h in range(1, height+1):
        for w in range(1, width+1):
            if(img[h-1, w-1] == 255):
                label[h, w] = tmp
                tmp += 1
    # Iterative
    while(1):
        # Top-down pass
        change = False

        for h in range(1, height+1):
            for w in range(1, width+1):
                if(label[h, w] != 0):
                    # 4-connectivity
                    neighbors = [label[h-1, w], label[h+1,w], label[h, w-1], label[h, w+1], label[h, w]]
                    M = min([x for x in neighbors if x != 0])
                    if (M != label[h, w]):
                        change = True
                        label[h, w] = M

        # Bottom-up pass

        for h in reversed(range(1, height+1)):
            for w in reversed(range(1, width+1)):
                if(label[h, w] != 0):
                    # 4-connectivity
                    neighbors = [label[h-1, w], label[h+1,w], label[h, w-1], label[h, w+1], label[h, w]]
                    M = min([x for x in neighbors if x != 0])
                    if (M != label[h, w]):
                        change = True
                        label[h, w] = M

        if (not change):
            break
    label = label[1:-1, 1:-1]
    img = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)

    for uni_label in np.unique(label):
        if uni_label == 0:
            continue

        mask = np.where(label == uni_label, 1, 0)
        if np.sum(mask) < 500:
            continue

        mask = np.argwhere(mask != 0)
        min_h, min_w = np.min(mask, axis=0)
        max_h, max_w = np.max(mask, axis=0)

        centroid_h = (min_h + max_h) // 2
        centroid_w = (min_w + max_w) // 2

        img = cv2.rectangle(img, (min_w, min_h), (max_w, max_h), (255, 0, 0), 5)
        img = cv2.drawMarker(img, (centroid_w, centroid_h), color = (0, 0, 255), markerType = cv2.MARKER_CROSS, thickness = 2)

    cv2.imwrite('c.png', img)
```

Resulting image: