

電腦視覺 作業二



指導老師 傅楸善

學生 蔡宇晴

學號 r08945050

Part1.

Binary Image

原圖	
Binary Image	
程式碼	<pre>def binary_img(): img = cv2.imread("./lena.bmp",cv2.IMREAD_GRAYSCALE) w,h = img.shape for i in range(w): for j in range(h): if img[i,j]>127: img[i,j] = 255 else: img[i,j] = 0 cv2.imwrite("./lena.jpg",img) binary_img()</pre>

想法	雙 for 迴圈，遍歷所有點後，把閾值上下的點，像素值各自分成 255 和 0。
----	--

Part2.

原圖	
直方圖	
程式碼	<pre>def histogram(): img = cv2.imread("./lena.bmp", cv2.IMREAD_GRAYSCALE) w,h = img.shape dict1 = {} for i in range(255): dict1[i] = 0 for row in range(w): for col in range(h): dict1[img[row,col]] += 1 X = [i for i in range(255)] x = [pixel for pixel in dict1.values()] plt.title('Lena Histogram') plt.xlabel("Bin") plt.ylabel("# of pixel") plt.bar(X, x, alpha=0.5, width=1, facecolor='red', edgecolor='black', label='two', lw=1) plt.show()</pre>
想法	把灰階圖遍歷後，記下所有 0 至 255 對應的像素點數量，在把彼此的關係顯示成直方圖。

Part3.

<p>原圖</p>	
<p>標記後的 圖 (使用 四連通)</p>	

程式碼

```
# 左上右下的 kernel
kernel_down = np.array([[0, 1, 0],
                        [1, 1, 0],
                        [0, 0, 0]], dtype=np.bool)
kernel_up = np.array([[0, 0, 0],
                      [0, 1, 1],
                      [0, 1, 0]], dtype=np.bool)

# np.pad(圖, ((上, 下), (左, 右)), 方法)
img_pad = np.pad(img, ((1, 1), (1, 1)), 'constant')

count = 1
# 填入 counter
for i in range(h):
    for j in range(w):
        if img_pad[i+1, j] != 0 and img_pad[i, j+1] != 0 and img_pad[i+1, j+1] != 0:
            img_counter[i, j] = count
            count += 1

# np.pad(圖, ((上, 下), (左, 右)), 方法)
img_counter_pad = np.pad(img_counter, ((1, 1), (1, 1)), 'constant')
```

```
# 上往下
iteration = 5
for iter in range(iteration): # iteration
    # print(f"top down, iter = {iter}")
    for i in range(h):
        for j in range(w):
            cut = img_counter_pad[i:i+3, j:j+3]
            test = cut * kernel_down

            non_zero_pos = np.where(test != 0)
            non_zero_val = test[non_zero_pos]

            for y, x in zip(non_zero_pos[0], non_zero_pos[1]):
                img_counter_pad[i+y, j+x] = np.min(non_zero_val)

# 下往上
print(f"bottom up, iter = {iter}")
for i in range(h-1, -1, -1):
    for j in range(w-1, -1, -1):
        cut = img_counter_pad[i:i + 3, j:j + 3]
        test = cut * kernel_up

        non_zero_pos = np.where(test != 0)
        non_zero_val = test[non_zero_pos]

        for y, x in zip(non_zero_pos[0], non_zero_pos[1]):
            img_counter_pad[i + y, j + x] = np.min(non_zero_val)
```

```
index500 = []
# 將 500 以下的填 0
for c in np.unique(img_counter_pad):
    if np.sum(img_counter_pad == c) < 500:
        img_counter_pad[img_counter_pad == c] = 0
    elif c != 0:
        # print(f"c = {c}, # = {np.sum(img_counter_pad == c)}")
        index500.append(c)
```

	<pre> def mark(img, c): x_min = np.min(np.where(img == c)[1]) - 1 x_max = np.max(np.where(img == c)[1]) + 1 y_min = np.min(np.where(img == c)[0]) - 1 y_max = np.max(np.where(img == c)[0]) + 1 center = (np.mean(np.where(img == c)[1]) - 1, np.mean(np.where(img == c)[0]) - 1) return (x_min, y_min), (x_max, y_max), center def draw_cross(img, center, length_=5): x, y = center x = int(x) y = int(y) img[y-1:y+1, x - length: x + length + 1] = np.array([0, 0, 255]) img[y - length: y + length + 1, x-1:x+1] = np.array([0, 0, 255]) return img for i in range(1, len(index500) + 1): min_loc, max_loc, center = mark(img_counter_pad, i) # print(min_loc, max_loc, center) cv2.rectangle(img, min_loc, max_loc, (255, 0, 0), 2) img = draw_cross(img, center) </pre>
<p>想法</p>	<p>首先，先開 2 個 filter,以便分別確認左邊和上方以及右邊和下方的像素點。</p> <p>步驟 1.</p> <p>遍歷所有點的時候，在所在位置(l, j)有值的時候，同時確認左邊及上面，當三者皆有值的時候，所在位置(l, j)用 counter 標記。</p> <p>步驟 2.</p> <p>再一次確認左邊及上方，把所在位置(l, j)和其他兩點比較，取出最小值，並賦值給(l, j)。</p> <p>接著，由右下至左上的遍歷開始執行，此時確認右邊及下方還有(l, j)本身的值。將最小值賦值給(l, j)。以上流程重複 5 次，直到不再變化。</p> <p>步驟 3.</p> <p>從各自物件中的像素點中，得到所有像素點的座標和座標極大值，座標極小值，進而可以得到重心還有矩形的範圍。</p>

