GENERALIZED HOUGH TRANSFORM

S1154005 李育林

GHT流程

1. 對Reference和Template圖片都進行邊緣偵測處理

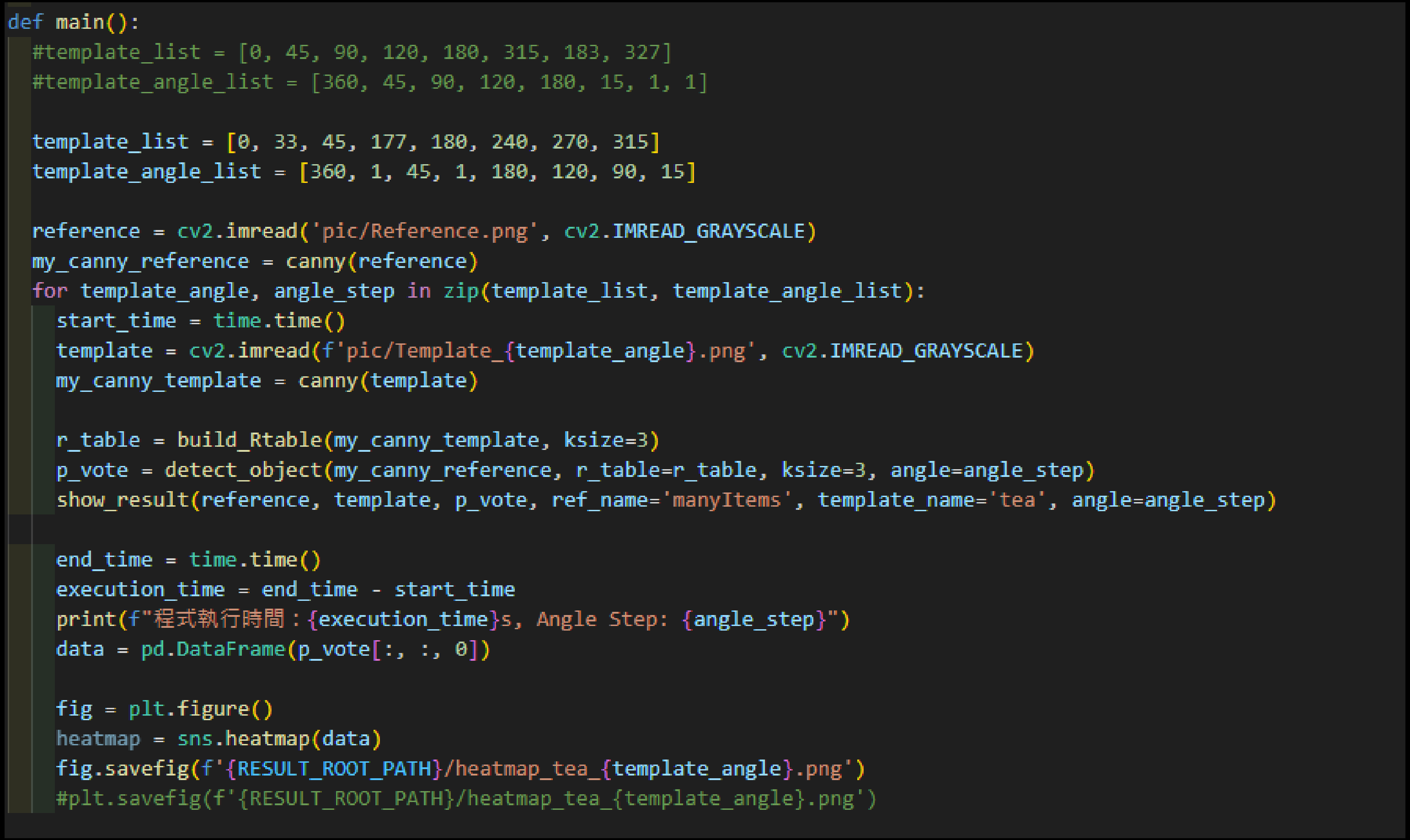
2. 對於Template圖片選擇參考點(Reference point)，並對邊緣edge points根據參考點計算其距離r和角度α，並將計算的資訊存入由edge points梯度爲index的R table

3. 對於想要偵測的圖片(Reference)，開始在每個edge points上使用R table反推回參考點，並記錄每個參考點座標出現次數

4. 最終選擇出現次數最多的座標，其就會是我們想要找的物件之位置

* 若要匹配不同角度，擇要在4.紀錄座標時多紀錄一個角度(P[x][y][θ])

MAIN FUNCTION

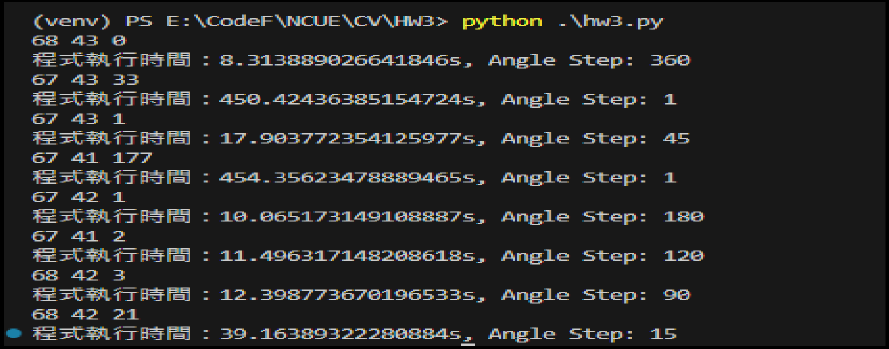


```python
def main():
    #template_list = [0, 45, 90, 120, 180, 315, 183, 327]
    #template_angle_list = [360, 45, 90, 120, 180, 15, 1, 1]

    template_list = [0, 33, 45, 177, 180, 240, 270, 315]
    template_angle_list = [360, 1, 45, 1, 180, 120, 90, 15]

    reference = cv2.imread('pic/Reference.png', cv2.IMREAD_GRAYSCALE)
    my_canny_reference = canny(reference)
    for template_angle, angle_step in zip(template_list, template_angle_list):
        start_time = time.time()
        template = cv2.imread(f'pic/Template_{template_angle}.png', cv2.IMREAD_GRAYSCALE)
        my_canny_template = canny(template)

        r_table = build_Rtable(my_canny_template, ksize=3)
        p_vote = detect_object(my_canny_reference, r_table=r_table, ksize=3, angle=angle_step)
        show_result(reference, template, p_vote, ref_name='manyItems', template_name='tea', angle=angle_step)

        end_time = time.time()
        execution_time = end_time - start_time
        print(f"程式執行時間：{execution_time}s, Angle Step: {angle_step}")
        data = pd.DataFrame(p_vote[:, :, 0])

        fig = plt.figure()
        heatmap = sns.heatmap(data)
        fig.savefig(f'{RESULT_ROOT_PATH}/heatmap_tea_{template_angle}.png')
        #plt.savefig(f'{RESULT_ROOT_PATH}/heatmap_tea_{template_angle}.png')
```

讀入reference、template圖片，並依次進行GHT流程尋找reference圖中的目標物件。

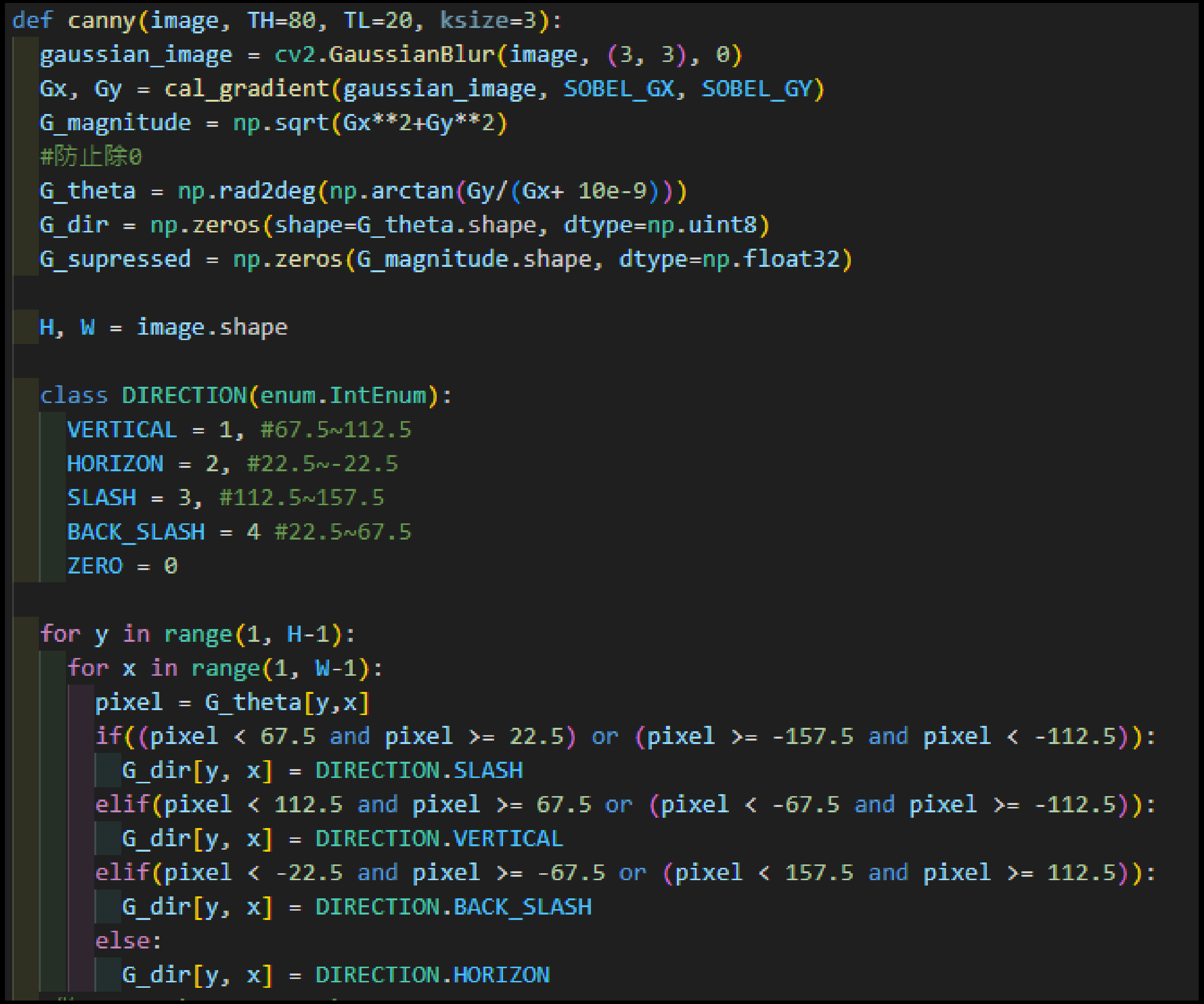
而在圖像旋轉部分，可以透過設定可能旋轉角度(angle)來讓後續可以偵測到目標圖片，其代表旋轉角度的精度，而精度越小計算時間也會越長

EX：一張270旋轉的template圖片，透過設定angle爲45可以正確偵測出[0, 45, 90, 135, 180, 225, 270, 315]等角度之圖

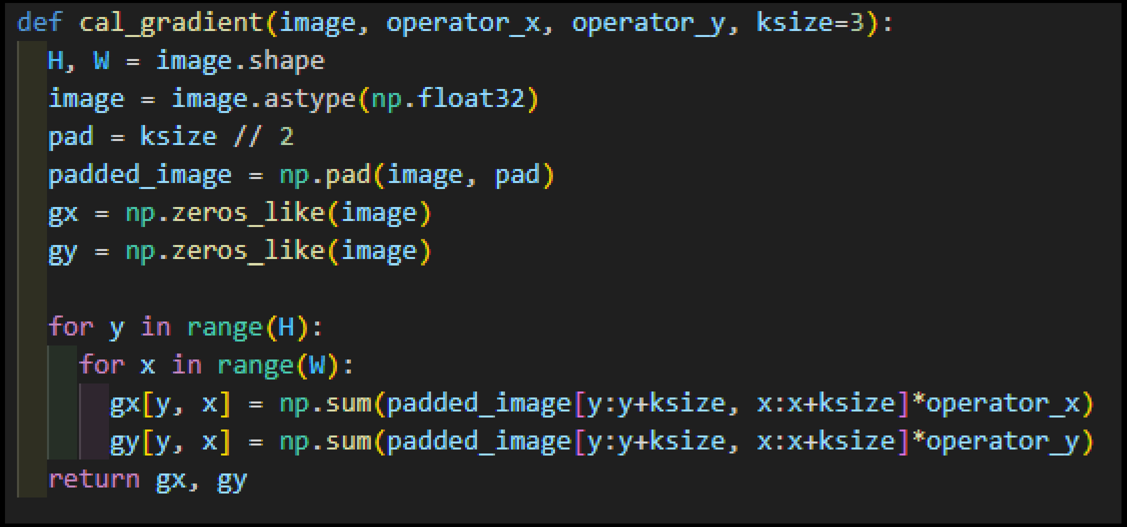


```
(venv) PS E:\CodeF\NCUE\CV\HW3> python .\hw3.py
68 43 0
程式執行時間：8.313889026641846s, Angle Step: 360
67 43 33
程式執行時間：450.42436385154724s, Angle Step: 1
67 43 1
程式執行時間：17.903772354125977s, Angle Step: 45
67 41 177
程式執行時間：454.35623478889465s, Angle Step: 1
67 42 1
程式執行時間：10.065173149108887s, Angle Step: 180
67 41 2
程式執行時間：11.496317148208618s, Angle Step: 120
68 42 3
程式執行時間：12.398773670196533s, Angle Step: 90
68 42 21
程式執行時間：39.16389322280884s, Angle Step: 15
```

一、邊緣偵測



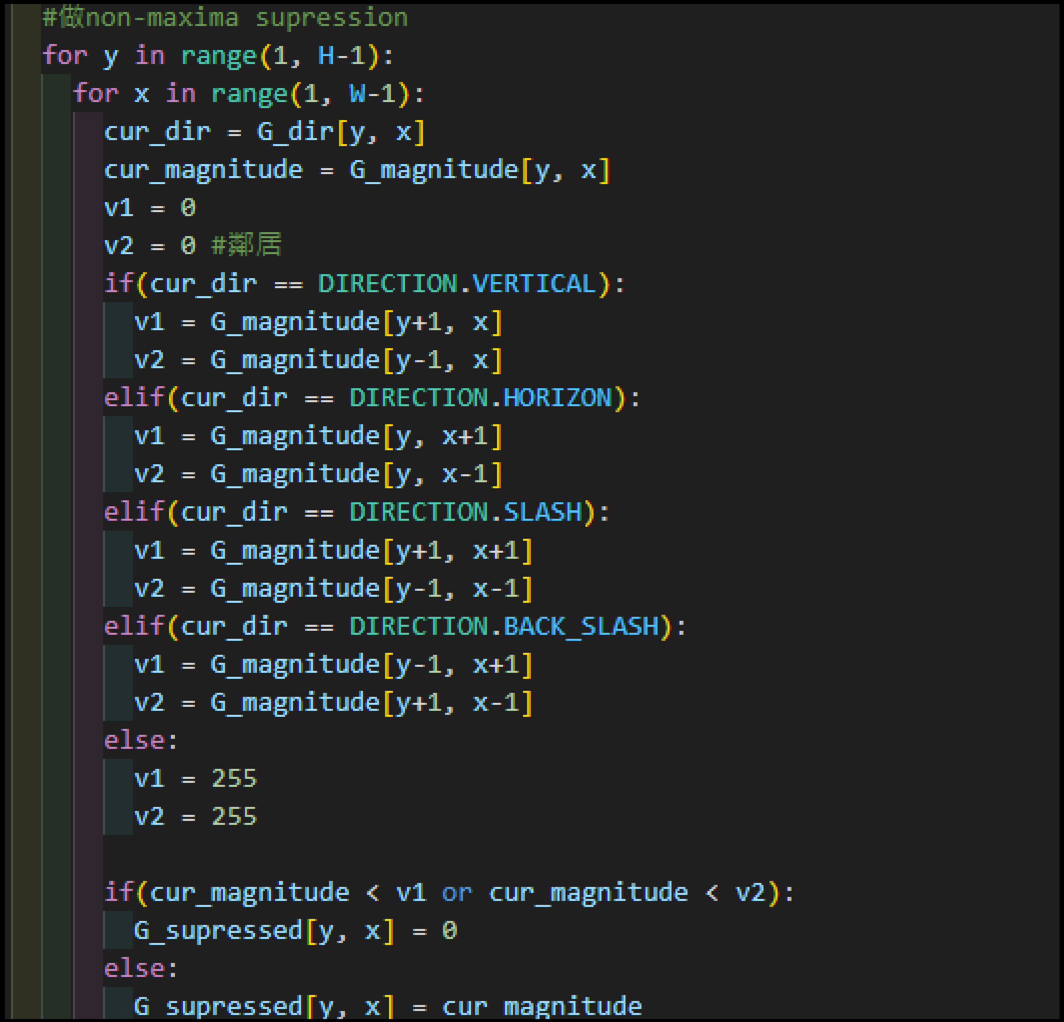
```python
def canny(image, TH=80, TL=20, ksize=3):
    gaussian_image = cv2.GaussianBlur(image, (3, 3), 0)
    Gx, Gy = cal_gradient(gaussian_image, SOBEL_GX, SOBEL_GY)
    G_magnitude = np.sqrt(Gx**2+Gy**2)
    #防止除0
    G_theta = np.rad2deg(np.arctan(Gy/(Gx+ 10e-9)))
    G_dir = np.zeros(shape=G_theta.shape, dtype=np.uint8)
    G_supressed = np.zeros(G_magnitude.shape, dtype=np.float32)

    H, W = image.shape

    class DIRECTION(enum.IntEnum):
        VERTICAL = 1, #67.5~112.5
        HORIZON = 2, #22.5~-22.5
        SLASH = 3, #112.5~157.5
        BACK_SLASH = 4 #22.5~67.5
        ZERO = 0

    for y in range(1, H-1):
        for x in range(1, W-1):
            pixel = G_theta[y,x]
            if((pixel < 67.5 and pixel >= 22.5) or (pixel >= -157.5 and pixel < -112.5)):
                G_dir[y, x] = DIRECTION.SLASH
            elif(pixel < 112.5 and pixel >= 67.5 or (pixel < -67.5 and pixel >= -112.5)):
                G_dir[y, x] = DIRECTION.VERTICAL
            elif(pixel < -22.5 and pixel >= -67.5 or (pixel < 157.5 and pixel >= 112.5)):
                G_dir[y, x] = DIRECTION.BACK_SLASH
            else:
                G_dir[y, x] = DIRECTION.HORIZON
```



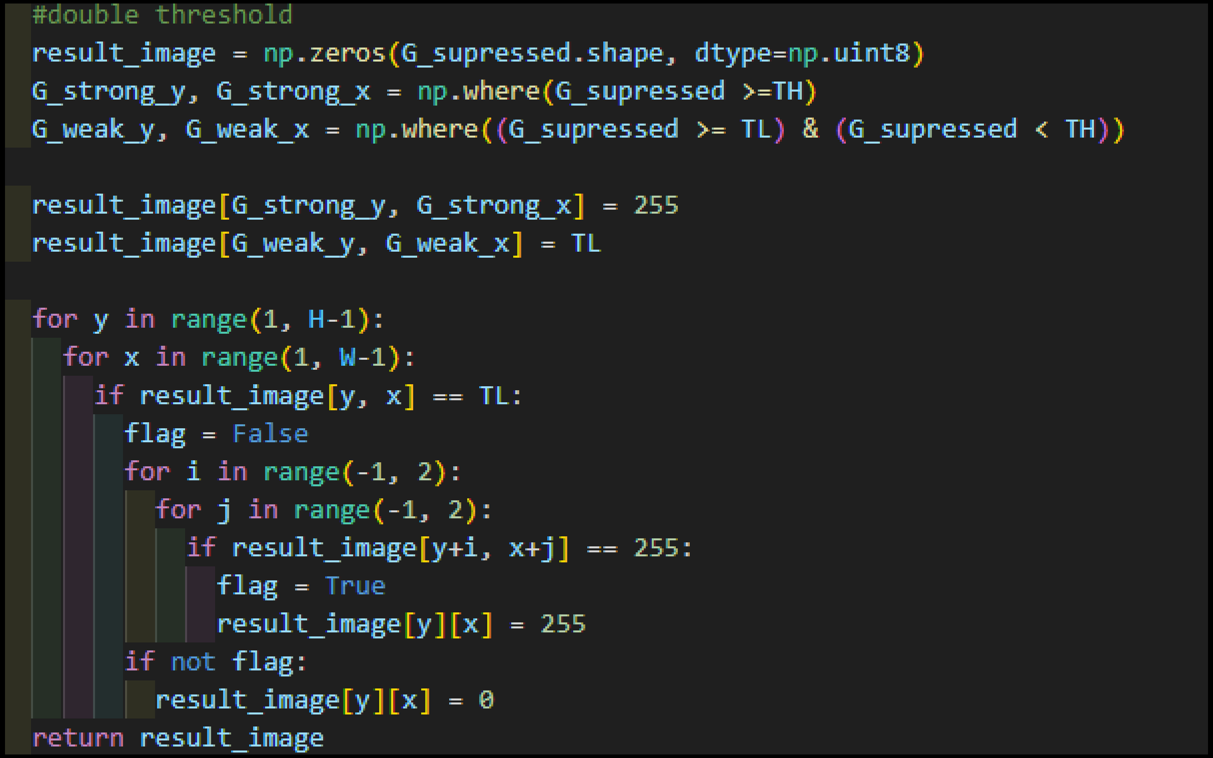
```python
def cal_gradient(image, operator_x, operator_y, ksize=3):
    H, W = image.shape
    image = image.astype(np.float32)
    pad = ksize // 2
    padded_image = np.pad(image, pad)
    gx = np.zeros_like(image)
    gy = np.zeros_like(image)

    for y in range(H):
        for x in range(W):
            gx[y, x] = np.sum(padded_image[y:y+ksize, x:x+ksize]*operator_x)
            gy[y, x] = np.sum(padded_image[y:y+ksize, x:x+ksize]*operator_y)
    return gx, gy
```

邊緣偵測使用之前在HW2已經寫好的canny+sobel計算梯度，去除不必要的圖片資訊，僅保留邊界方便後續R table的建立

一、邊緣偵測

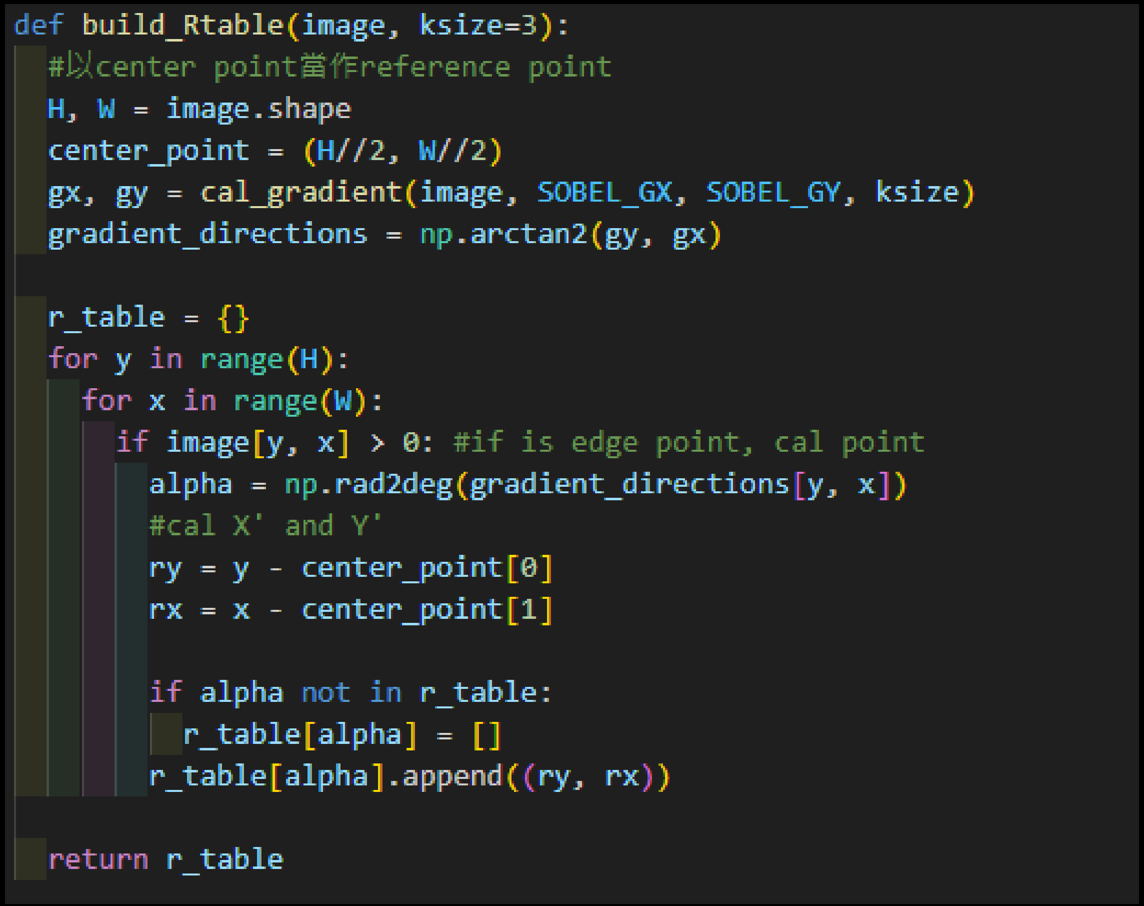


```
#做non-maxima supression
for y in range(1, H-1):
    for x in range(1, W-1):
        cur_dir = G_dir[y, x]
        cur_magnitude = G_magnitude[y, x]
        v1 = 0
        v2 = 0 #鄰居
        if(cur_dir == DIRECTION.VERTICAL):
            v1 = G_magnitude[y+1, x]
            v2 = G_magnitude[y-1, x]
        elif(cur_dir == DIRECTION.HORIZON):
            v1 = G_magnitude[y, x+1]
            v2 = G_magnitude[y, x-1]
        elif(cur_dir == DIRECTION.SLASH):
            v1 = G_magnitude[y+1, x+1]
            v2 = G_magnitude[y-1, x-1]
        elif(cur_dir == DIRECTION.BACK_SLASH):
            v1 = G_magnitude[y-1, x+1]
            v2 = G_magnitude[y+1, x-1]
        else:
            v1 = 255
            v2 = 255

        if(cur_magnitude < v1 or cur_magnitude < v2):
            G_supressed[y, x] = 0
        else:
            G_supressed[y, x] = cur_magnitude
```



```
#double threshold
result_image = np.zeros(G_supressed.shape, dtype=np.uint8)
G_strong_y, G_strong_x = np.where(G_supressed >=TH)
G_weak_y, G_weak_x = np.where((G_supressed >= TL) & (G_supressed < TH))

result_image[G_strong_y, G_strong_x] = 255
result_image[G_weak_y, G_weak_x] = TL

for y in range(1, H-1):
    for x in range(1, W-1):
        if result_image[y, x] == TL:
            flag = False
            for i in range(-1, 2):
                for j in range(-1, 2):
                    if result_image[y+i, x+j] == 255:
                        flag = True
                        result_image[y][x] = 255
            if not flag:
                result_image[y][x] = 0
return result_image
```

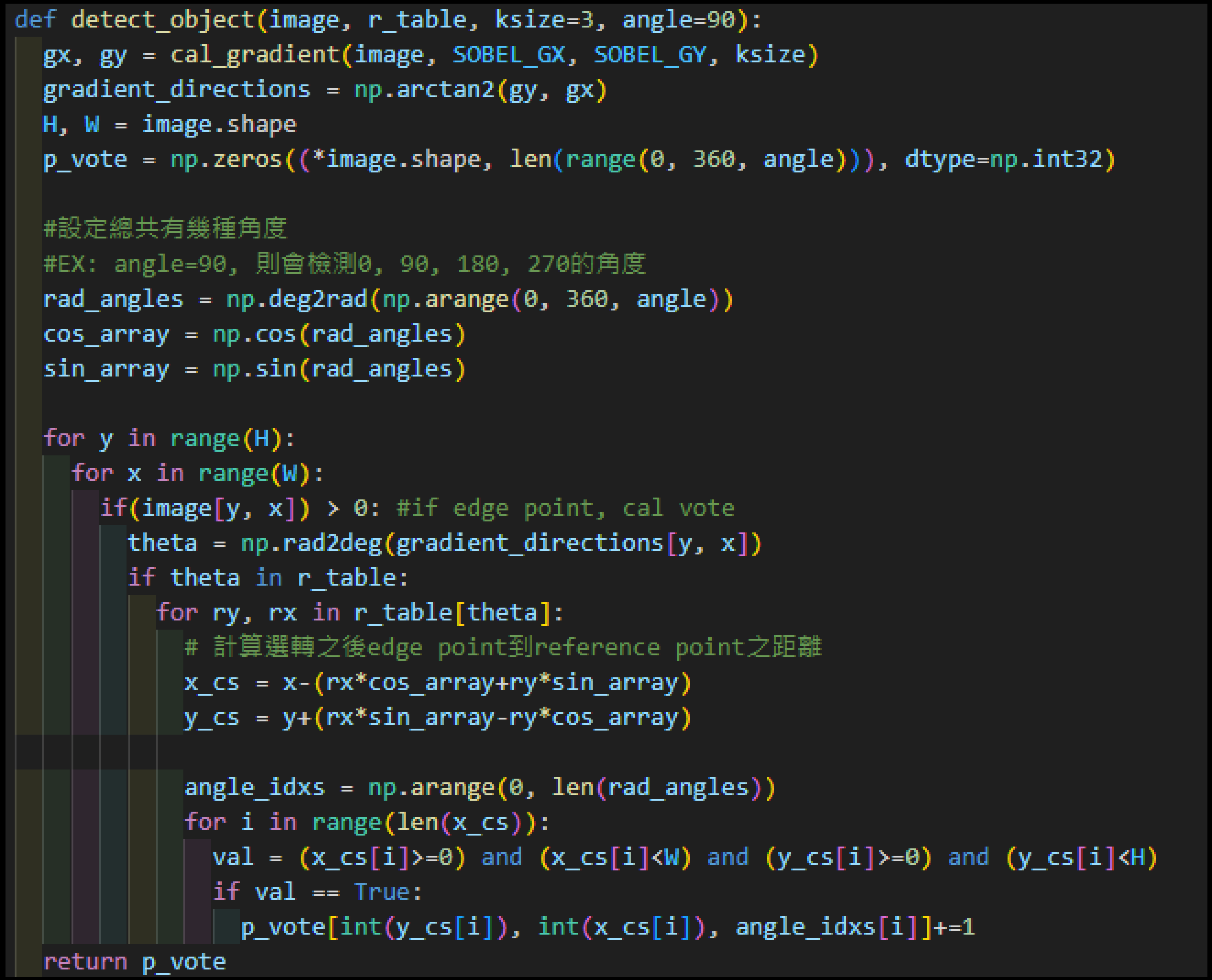
二、建立R TABLE



```
def build_Rtable(image, ksize=3):
  #以center point當作reference point
  H, W = image.shape
  center_point = (H//2, W//2)
  gx, gy = cal_gradient(image, SOBEL_GX, SOBEL_GY, ksize)
  gradient_directions = np.arctan2(gy, gx)

  r_table = {}
  for y in range(H):
    for x in range(W):
      if image[y, x] > 0: #if is edge point, cal point
        alpha = np.rad2deg(gradient_directions[y, x])
        #cal X' and Y'
        ry = y - center_point[0]
        rx = x - center_point[1]

        if alpha not in r_table:
          r_table[alpha] = []
        r_table[alpha].append((ry, rx))

  return r_table
```

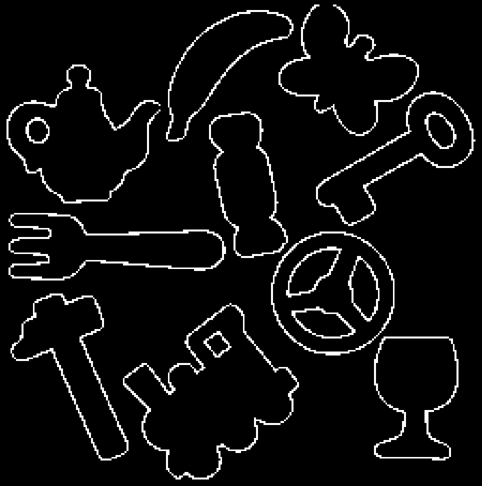
對template圖片建立R table儲存預計偵測物體對於參考點的特徵(距離+角度)，有了這些資訊就能幫助我們後續去檢測物件時從邊緣點反推參考點

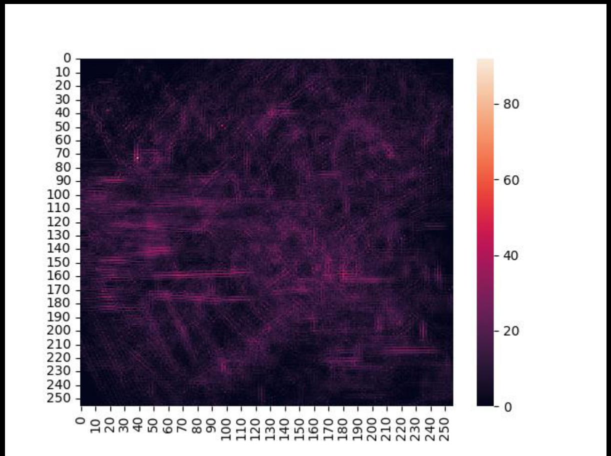
三、根據R TABLE做物件檢測



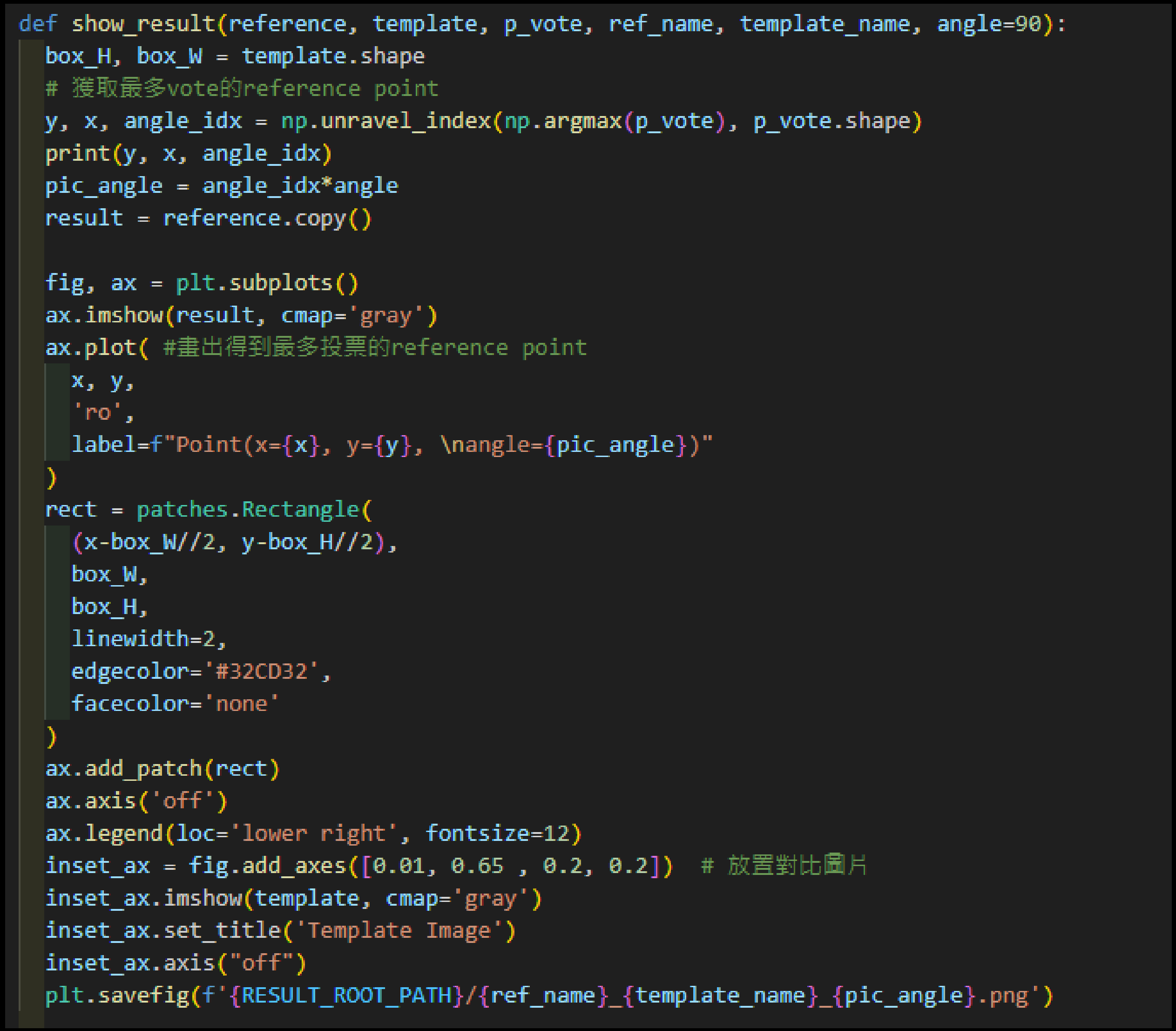
```
def detect_object(image, r_table, ksize=3, angle=90):
    gx, gy = cal_gradient(image, SOBEL_GX, SOBEL_GY, ksize)
    gradient_directions = np.arctan2(gy, gx)
    H, W = image.shape
    p_vote = np.zeros((*image.shape, len(range(0, 360, angle))), dtype=np.int32)

    #設定總共有幾種角度
    #EX: angle=90， 則會檢測0， 90， 180， 270的角度
    rad_angles = np.deg2rad(np.arange(0, 360, angle))
    cos_array = np.cos(rad_angles)
    sin_array = np.sin(rad_angles)

    for y in range(H):
        for x in range(W):
            if(image[y, x]) > 0: #if edge point, cal vote
                theta = np.rad2deg(gradient_directions[y, x])
                if theta in r_table:
                    for ry, rx in r_table[theta]:
                        # 計算選轉之後edge point到reference point之距離
                        x_cs = x-(rx*cos_array+ry*sin_array)
                        y_cs = y+(rx*sin_array-ry*cos_array)

                        angle_idxs = np.arange(0, len(rad_angles))
                        for i in range(len(x_cs)):
                            val = (x_cs[i]>=0) and (x_cs[i]<W) and (y_cs[i]>=0) and (y_cs[i]<H)
                            if val == True:
                                p_vote[int(y_cs[i]), int(x_cs[i]), angle_idxs[i]]+=1
    return p_vote
```

使用R table之資訊，去對reference圖像做從邊緣點反推參考點的步驟，我們會得到各種座標的參考點和角度，對於計算得到的參考點去做計數，最後在所有參考點中count值最高的就是我們的目標物件位置



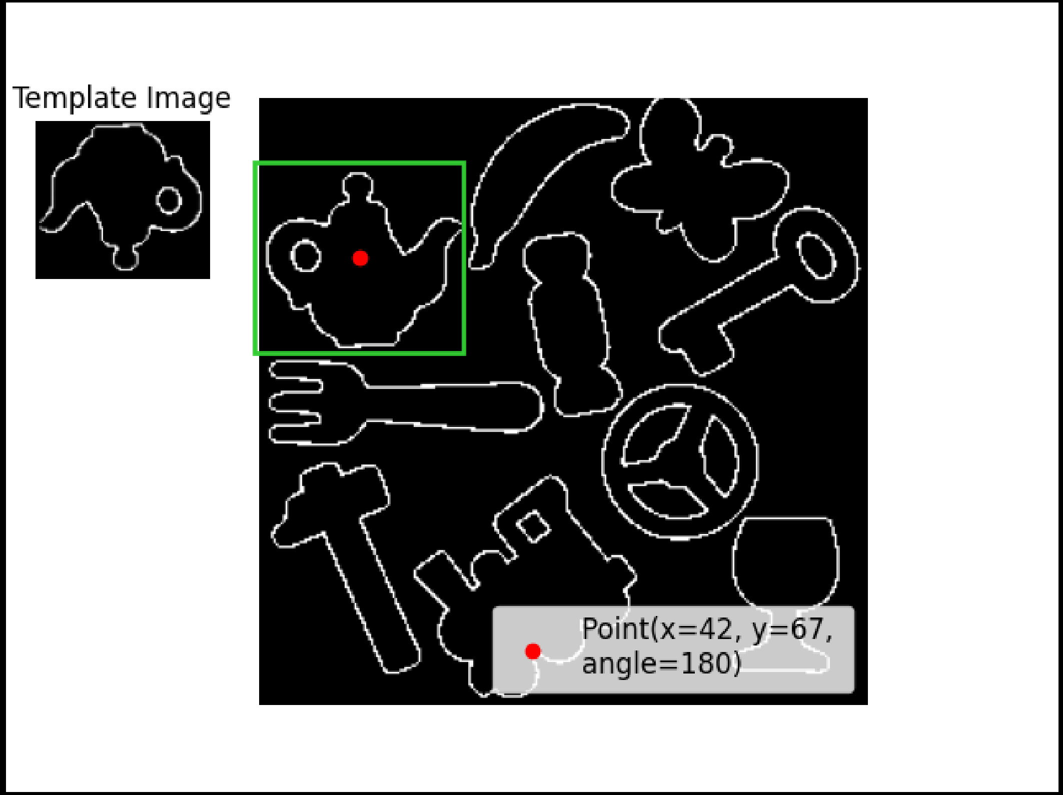


四、儲存察看結果

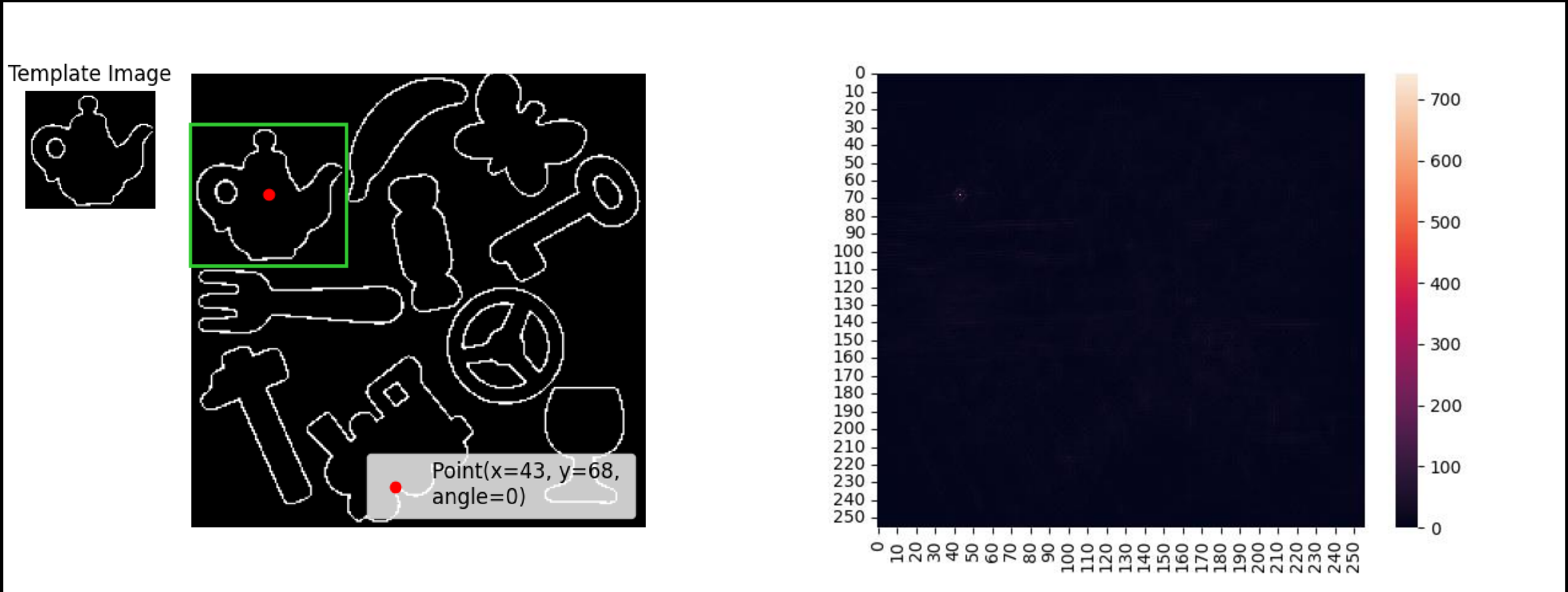


```python
def show_result(reference, template, p_vote, ref_name, template_name, angle=90):
    box_H, box_W = template.shape
    # 獲取最多vote的reference point
    y, x, angle_idx = np.unravel_index(np.argmax(p_vote), p_vote.shape)
    print(y, x, angle_idx)
    pic_angle = angle_idx*angle
    result = reference.copy()

    fig, ax = plt.subplots()
    ax.imshow(result, cmap='gray')
    ax.plot( #畫出得到最多投票的reference point
        x, y,
        'ro',
        label=f"Point(x={x}, y={y}, \nangle={pic_angle})"
    )
    rect = patches.Rectangle(
        (x-box_W//2, y-box_H//2),
        box_W,
        box_H,
        linewidth=2,
        edgecolor='#32CD32',
        facecolor='none'
    )
    ax.add_patch(rect)
    ax.axis('off')
    ax.legend(loc='lower right', fontsize=12)
    inset_ax = fig.add_axes([0.01, 0.65 , 0.2, 0.2])  # 放置對比圖片
    inset_ax.imshow(template, cmap='gray')
    inset_ax.set_title('Template Image')
    inset_ax.axis("off")
    plt.savefig(f'{RESULT_ROOT_PATH}/{ref_name}_{template_name}_{pic_angle}.png')
```

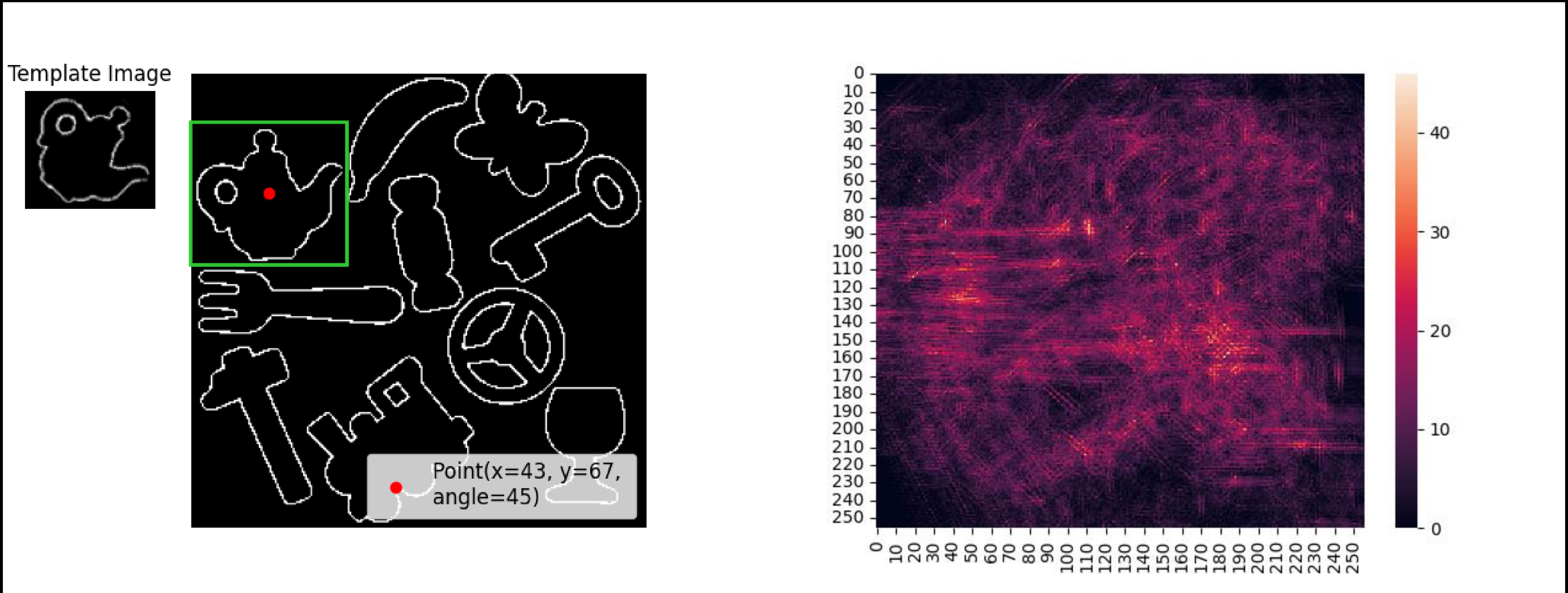
找到我們目標物體的最佳位置後，將其標示出來並將物體框出來



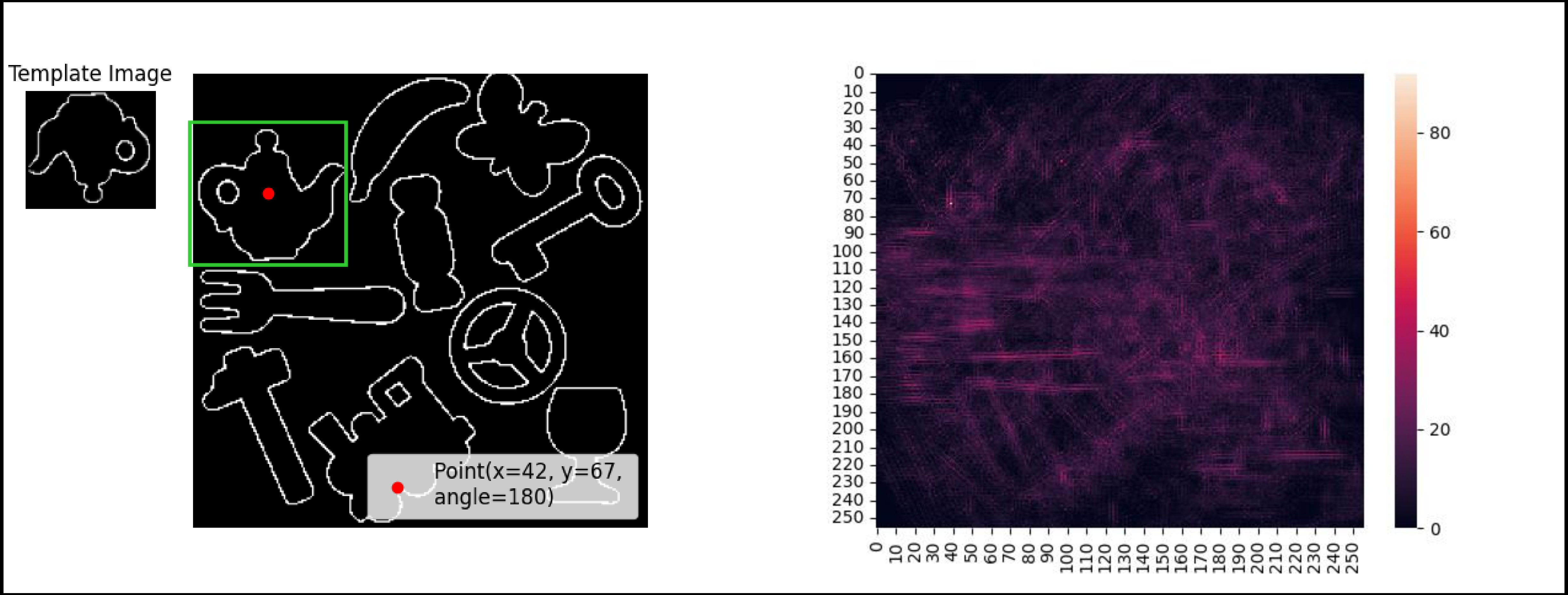
成果(無旋轉)


Template Image
Point(x=43, y=68, angle=0)
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
700
600
500
400
300
200
100
0

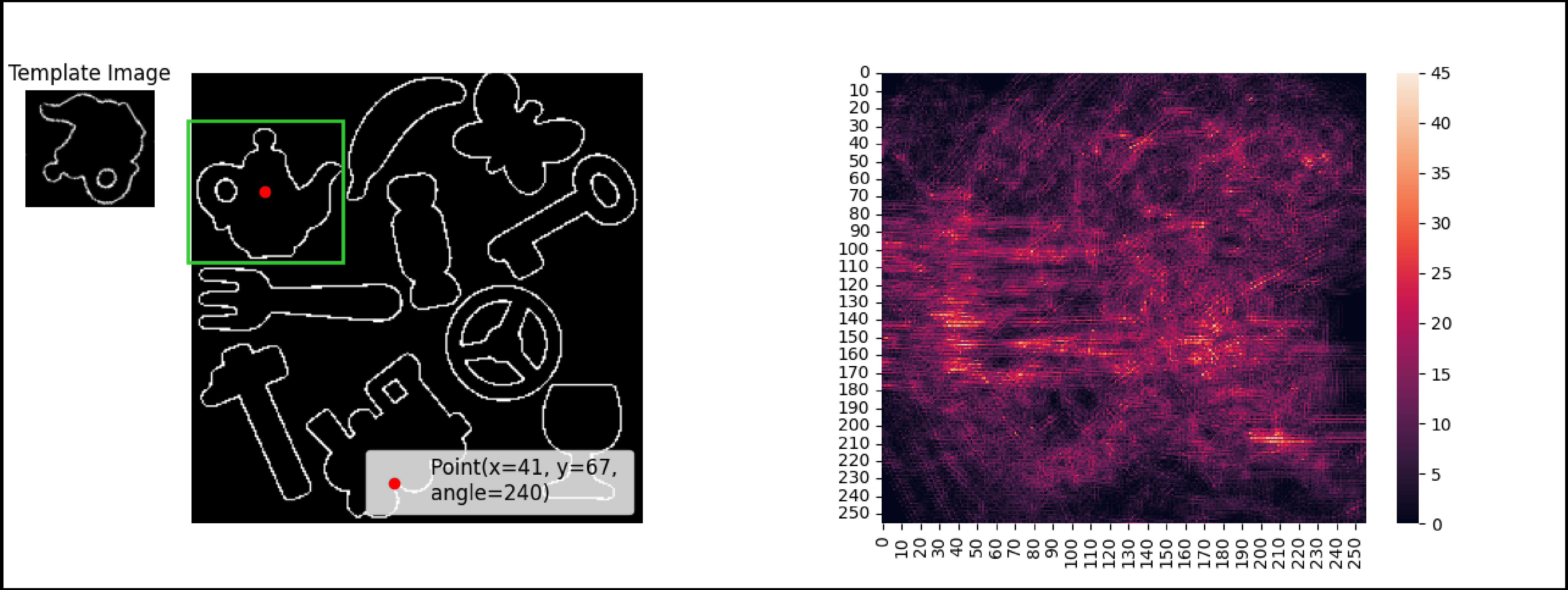
成果(45 °)


Template Image
Point(x=43, y=67,
angle=45)
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
40
30
20
10
0

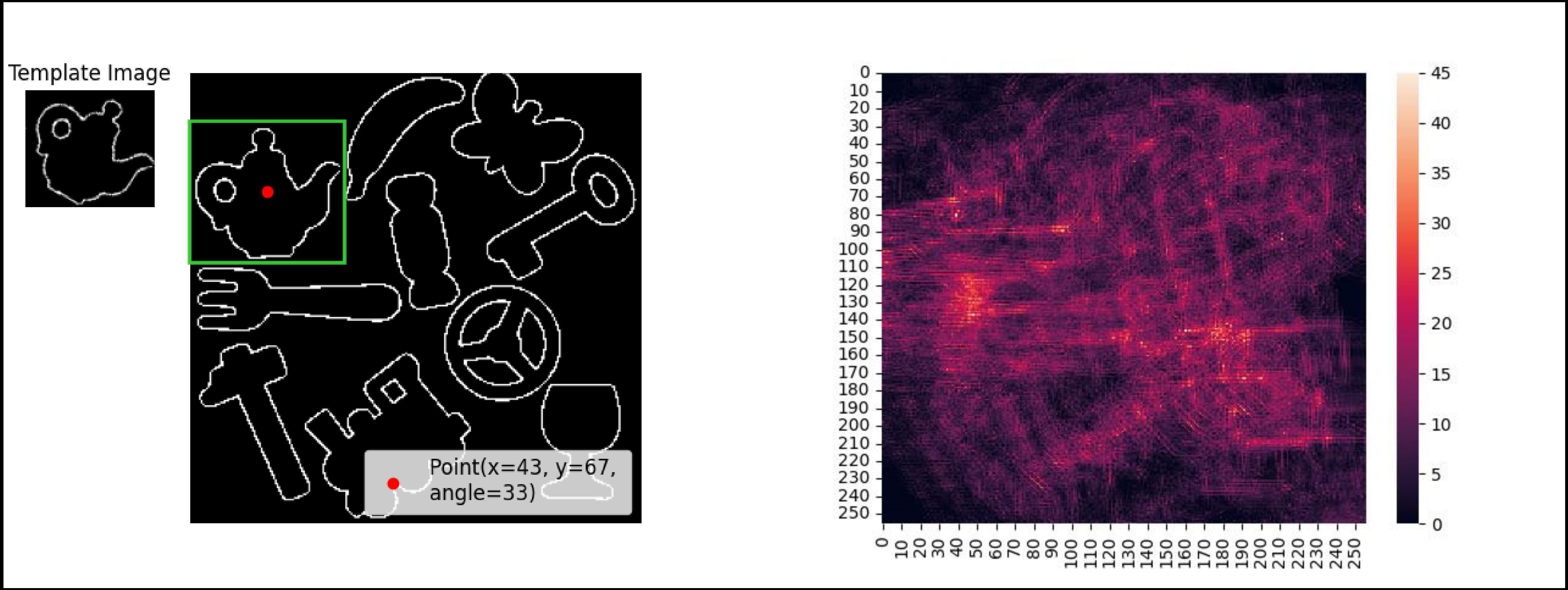
成果(180 °)


Template Image
Point(x=42, y=67,
angle=180)
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
80
60
40
20
0

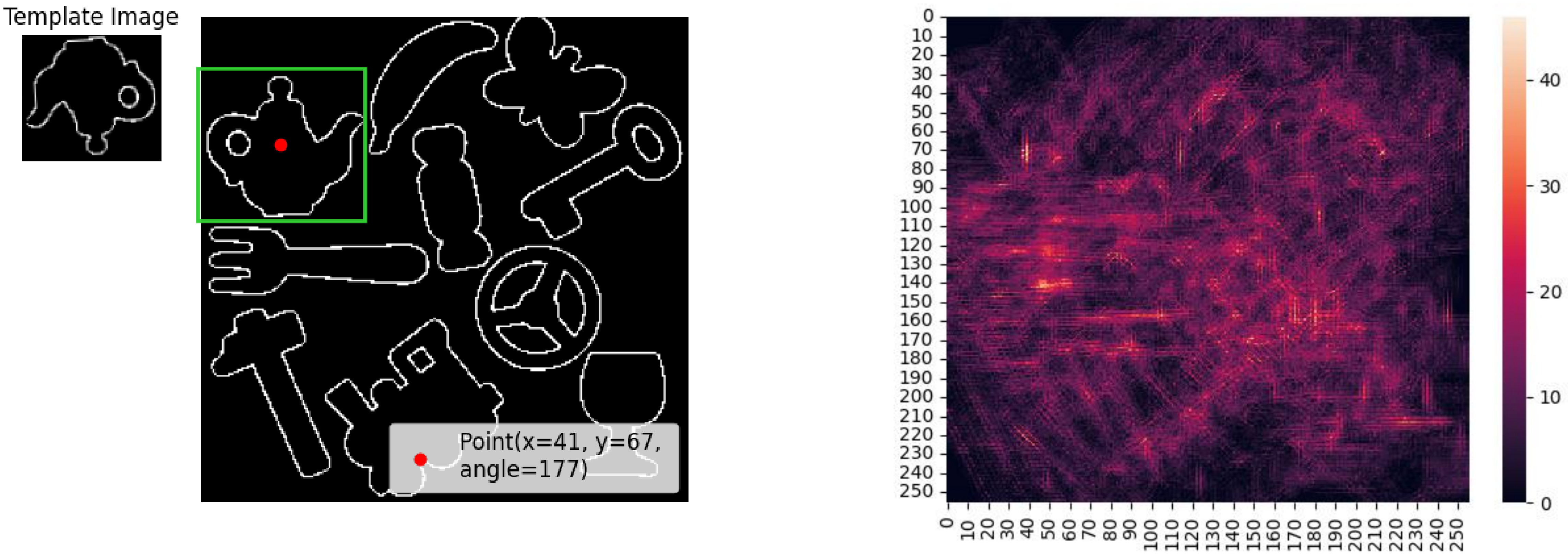
成果(240°)


Template Image
Point(x=41, y=67, angle=240)
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
45
40
35
30
25
20
15
10
5
0

成果(33°特殊角度)


Template Image
Point(x=43, y=67,
angle=33)
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
0
5
10
15
20
25
30
35
40
45

成果(177°特殊角度)


Template Image
Point(x=41, y=67,
angle=177)
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
0
10
20
30
40