S1154007 賴宥瑋 計算機視覺作業 3

目錄

一、Create R-table

1. 利用 Canny 做 edge detection，並定義中心點為模板(Template.png)中心

```
edges = cv2.Canny(template, 100, 200) # 利用Canny進行邊緣檢測

center = (template.shape[1] // 2, template.shape[0] // 2) # 定義參考點p(Xc, Yc)
```

2. 計算 gradients 角度，以度數儲存，當 phi (Φ)用

```
# 計算gradients角度
grad_x = cv2.Sobel(edges, cv2.CV_64F, 1, 0, ksize=3)
grad_y = cv2.Sobel(edges, cv2.CV_64F, 0, 1, ksize=3)
angles = cv2.phase(grad_x, grad_y, angleInDegrees=True) # 以度數儲存
```

3. 初始化 R-table

```
# 初始化R-table
R_table = {}
```

4. 遍歷所有 edge points (白色) ，以 Φ 當 key，將 r 跟 alpha 計算完後，儲存到 R_table[Φ]

```
# 將edge point(白色)存起來
edge_points = np.argwhere(edges > 0)

# 遍歷所有edge point
for y, x in edge_points:
    phi = int(angles[y, x]) # key值(Φ)

    dx = x - center[0]      # 計算x到Xc的差
    dy = y - center[1]      # 計算y到Yc的差

    r = np.sqrt(dx**2 + dy**2) # 計算(Xc, Yc)到(x, y)的距離
    alpha = np.arctan2(dy, dx) # 計算alpha

    # 當phi沒在R-table時，創建一個屬於他的空向量
    if phi not in R_table:
        R_table[phi] = []

    # 將r跟alpha存在R-table
    R_table[phi].append((r, alpha))

return R_table
```

二、GHT

1. 利用 Canny 做 edge detection

```
edges = cv2.Canny(reference, 100, 200) # 利用Canny進行邊緣檢測
```

2. 計算 gradients 角度

```
# 計算gradients角度
grad_x = cv2.Sobel(edges, cv2.CV_64F, 1, 0, ksize=3)
grad_y = cv2.Sobel(edges, cv2.CV_64F, 0, 1, ksize=3)
angles = cv2.phase(grad_x, grad_y, angleInDegrees=True) # 以度數儲存
```

3. 計算角度區間的數量(360 個)，初始化 accumulator[h][w][角度區間的數量]=0

```
h, w = reference.shape # 原圖(reference)大小
theta_amount = (theta_range[1] - theta_range[0]) // theta_range[2] + 1 # 計算theta range間有多少角度
accumulator = np.zeros((h, w, theta_amount)) # 投票陣列，大小為 h * w * theta_amount，初始化為0
```

4. 遍歷所有 edge points (白色)，計算 x’ y’後得 Xc Yc(使用老師講義的公式)，為 accumulator[Yc][Xc][theta]投一票

```
# 將edge point(白色)存起來
edge_points = np.argwhere(edges > 0)

# 遍歷所有edge point
for y, x in edge_points:
    phi = int(angles[y, x])

    # 確認是否phi有在r_table
    if phi in r_table:
        for r, alpha in r_table[phi]: # 開始投票
            x_prime = r * math.cos(alpha) # x' = r*cos(α)
            y_prime = r * math.sin(alpha) # y' = r*cos(α)
            for theta_idx, theta in enumerate(range(*theta_range)):
                theta_rad = math.radians(theta) # 將角度轉成弧度

                # 計算 (x_c, y_c)
                # Xc = x - (x'cos(θ) - y'sin(θ))
                # Yc = y - (x'sin(θ) + y'cos(θ))
                x_c = int(x - (x_prime * math.cos(theta_rad) - y_prime * math.sin(theta_rad)))
                y_c = int(y - (x_prime * math.sin(theta_rad) + y_prime * math.cos(theta_rad)))

                # 若(Xc, Yc)在圖片內(合理位置)
                if 0 <= x_c < w and 0 <= y_c < h:
                    accumulator[y_c, x_c, theta_idx] += 1 # 為 accumulator[y_c, x_c, theta_idx]投票
```

5. 找到最適合的點(accumulator 最多的)，並回傳最佳點的座標(Xc, Yc)和角度

```
# 找到最適合的點(accumulator最多的)
best_idx = np.unravel_index(np.argmax(accumulator), accumulator.shape)

# 最佳點的座標(Xc, Yc)和角度
best_match = (best_idx[1], best_idx[0], best_idx[2])

return best_match
```

三、Main

1. 讀取圖片和度數，度數是為了輸出而已，不會參與任何步驟

```
# 讀入 Template.png 和 Reference.png
template_path  = 'Template162.png'        可以是任何角度
reference_path = 'Reference.png'
template       = cv2.imread(template_path , cv2.IMREAD_GRAYSCALE)
reference      = cv2.imread(reference_path, cv2.IMREAD_GRAYSCALE)
original_image = cv2.imread(reference_path)

# 讀入Template度數(為了輸出而已)
degree = ''.join([char for char in template_path if char.isdigit()])
```

2. 建立 R-table

```
# 建立 R-table
r_table = build_R_table(template)
```

3. 定義角度範圍並執行 GHT(Generalized Hough Transform)

```
# 度數範圍 (theta_min, theta_max, step)
theta_range = (0, 360, 1)  # 0 ~ 359 degree , step = 1

# 執行 GHT
best_point = generalized_hough_transform(reference, r_table, theta_range)
```

4. 儲存最佳點的座標(Xc, Yc)和角度，並將偵測到的方框標出

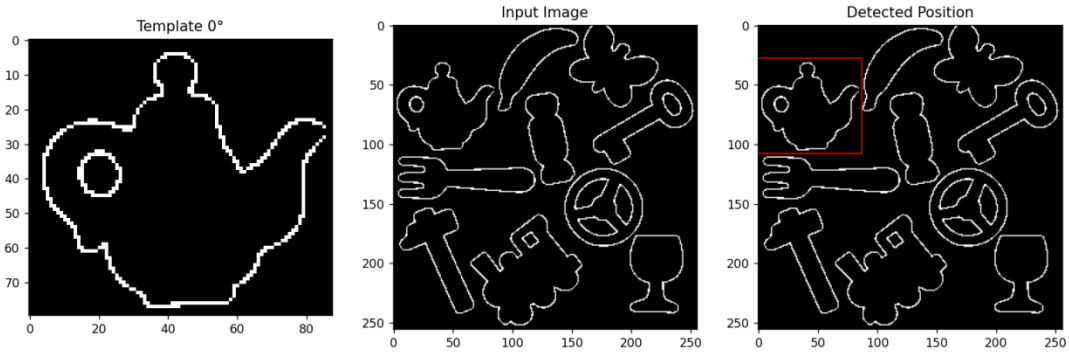
```
# 儲存最佳點的座標(Xc, Yc)和角度
best_x, best_y, best_theta = best_point

# 將偵測到的方框標出
template_h, template_w = template.shape                                  # 取得template的大小
top_left     = (best_x - template_w // 2, best_y - template_h // 2)   # 方框左上角座標
bottom_right = (best_x + template_w // 2, best_y + template_h // 2)   # 方框右下角座標
cv2.rectangle(original_image, top_left, bottom_right, (255, 0, 0), 1) # 為original_image繪製方框，紅色，寬度是1
```

5. 顯示結果

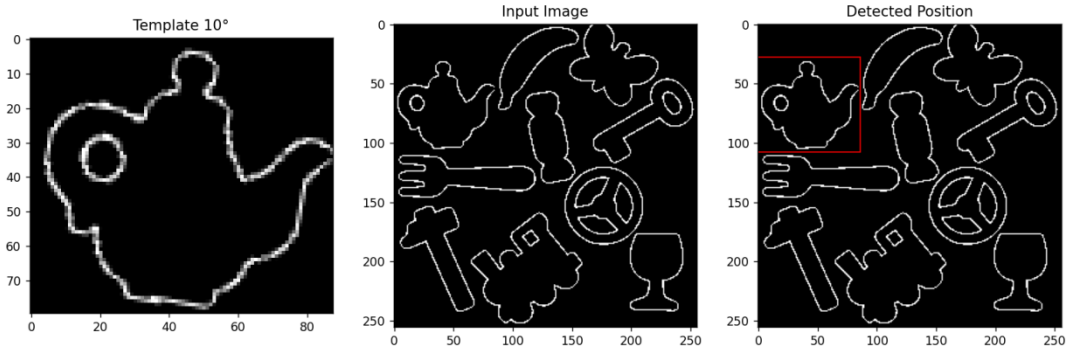
```
# 顯示結果
plt.figure(figsize=(16, 8))

plt.subplot(1, 3, 1)
plt.title(f"Template {degree}°")
plt.imshow(template, cmap='gray')

plt.subplot(1, 3, 2)
plt.title("Input Image")
plt.imshow(reference, cmap='gray')

plt.subplot(1, 3, 3)
plt.title(f"Detected Position")
plt.imshow(original_image)
plt.show()
```

四、Result

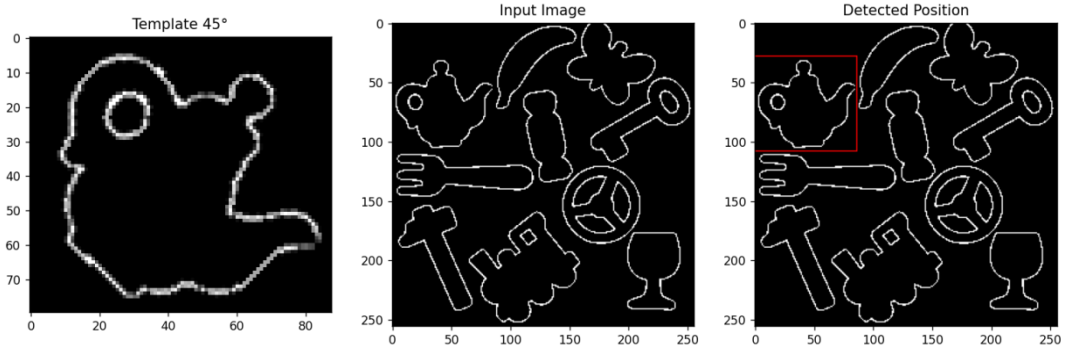
每個角度都可以偵測，只舉其中幾種，程式執行需花幾分鐘



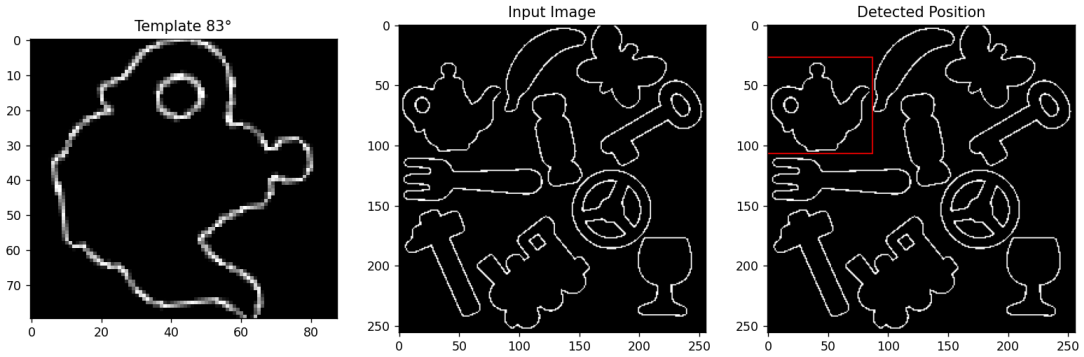
0 度，未經旋轉(原圖)



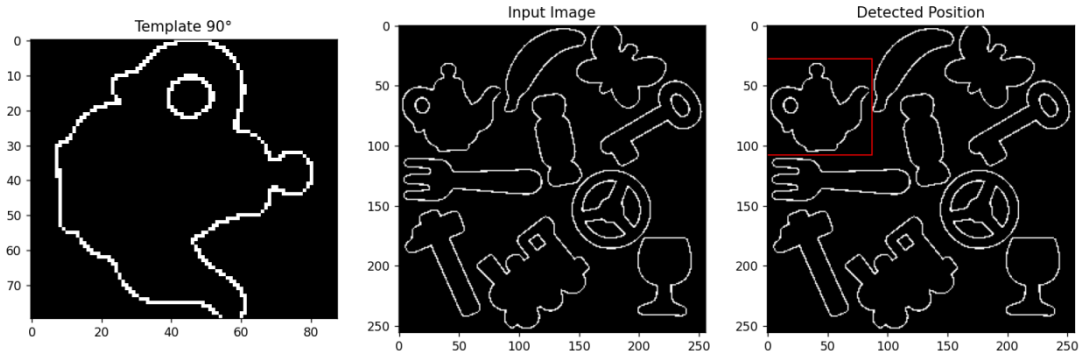
旋轉 10 度



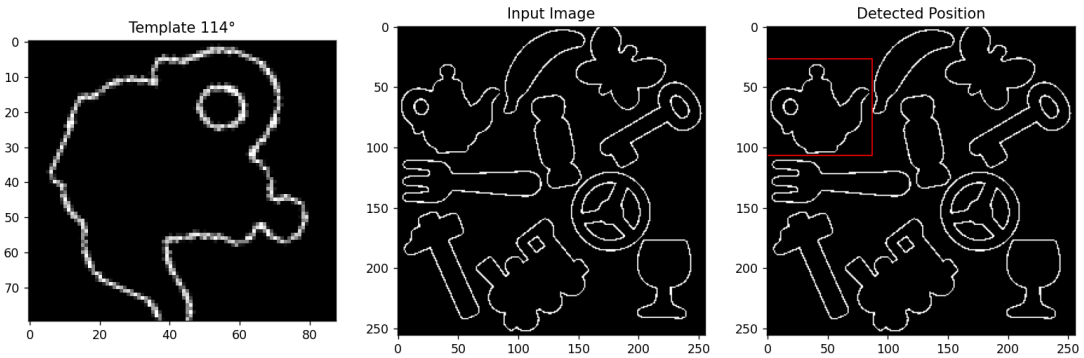
旋轉 45 度



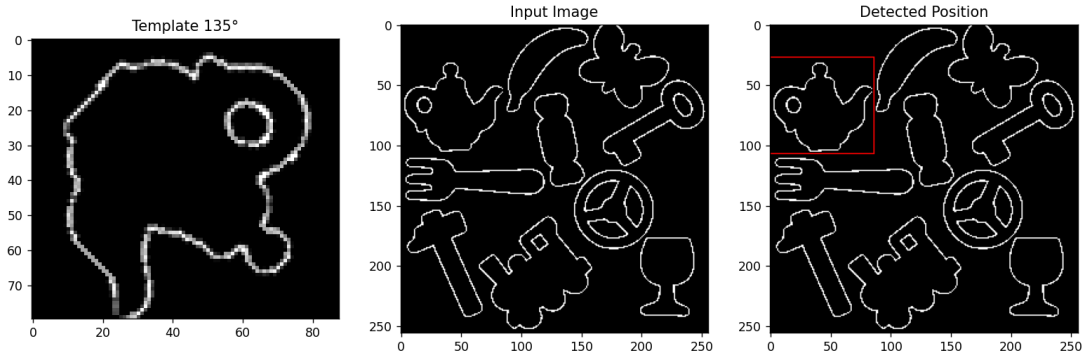
旋轉 83 度



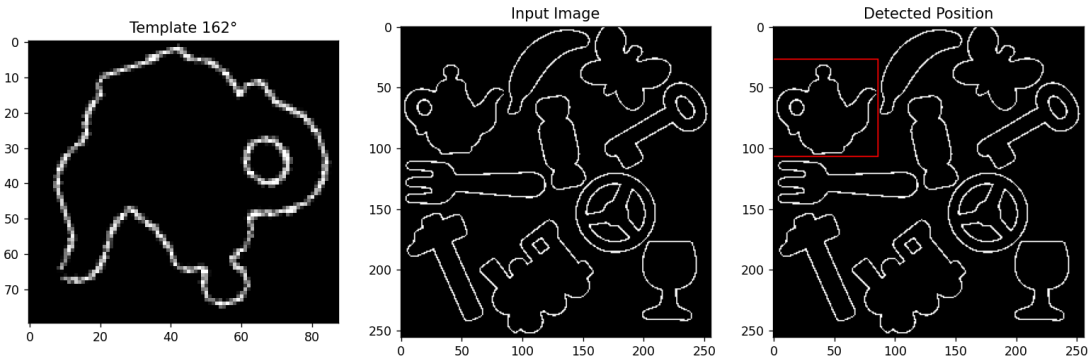
旋轉 90 度



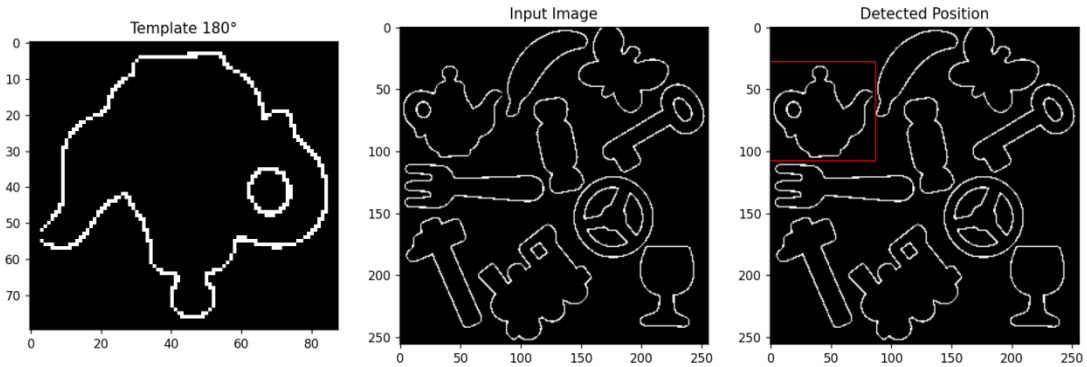
旋轉 114 度



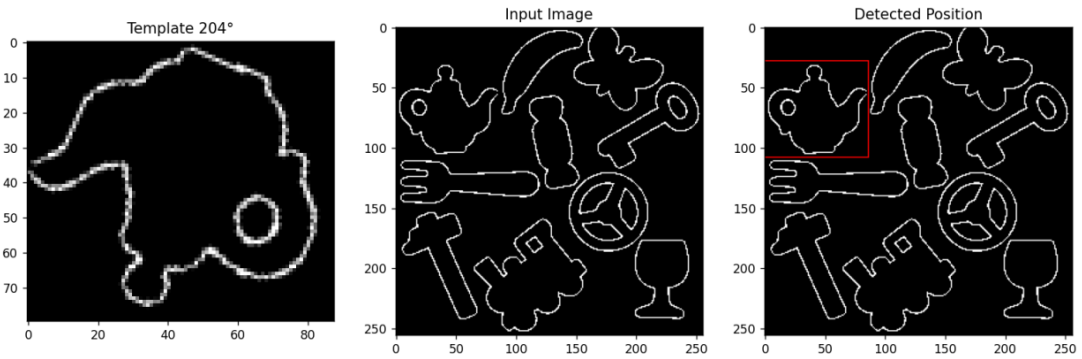
旋轉 135 度



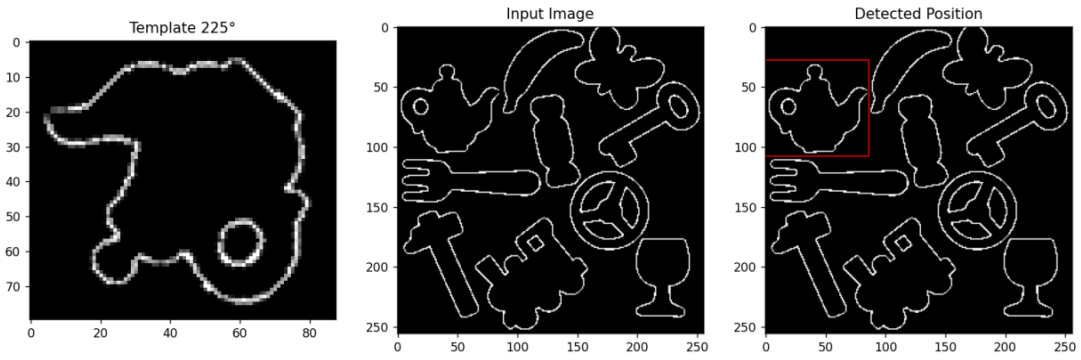
旋轉 162 度



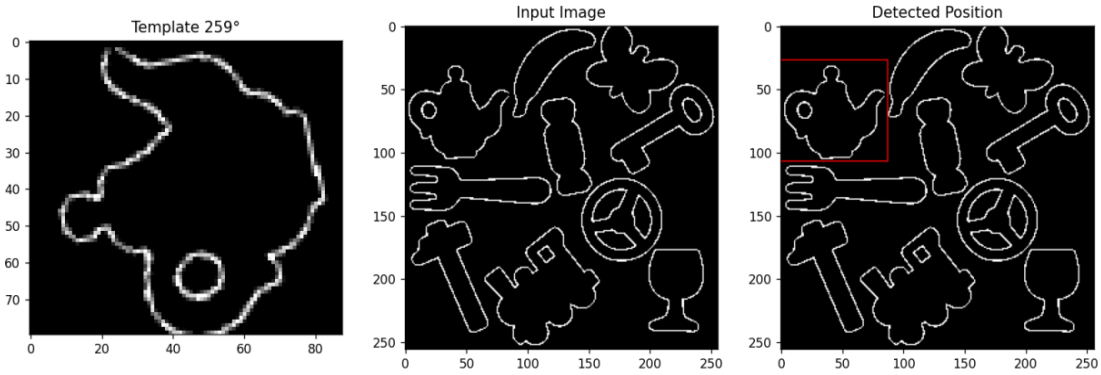
旋轉 180 度



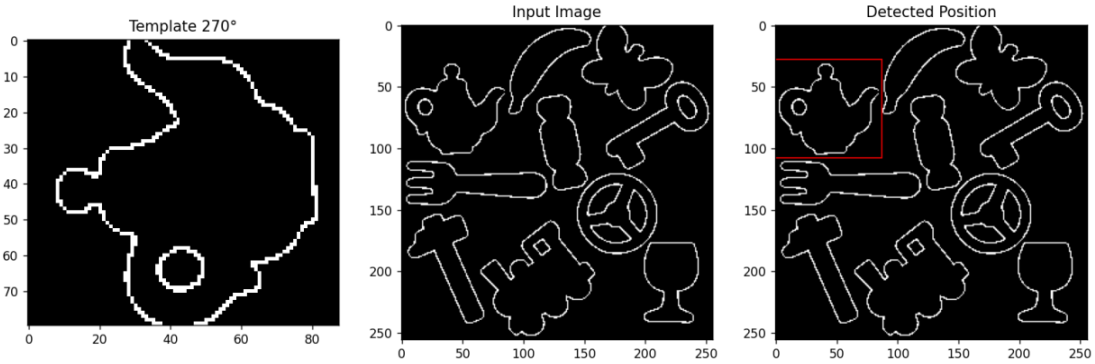
旋轉 204 度



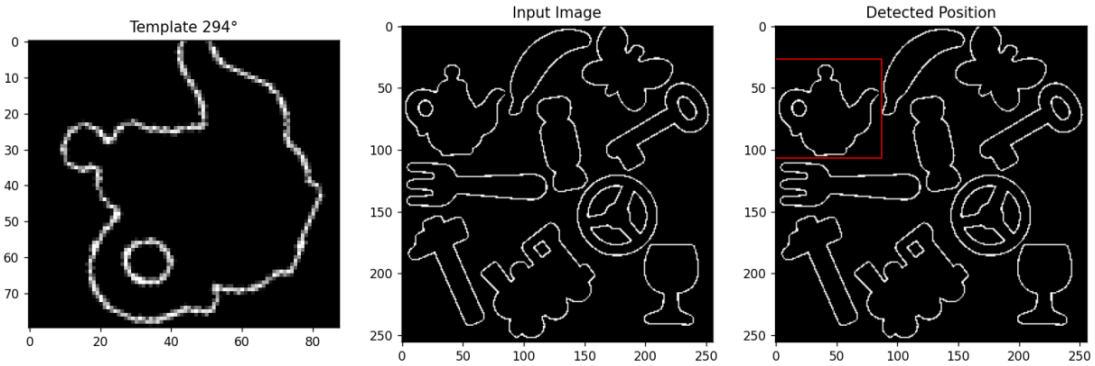
旋轉 225 度



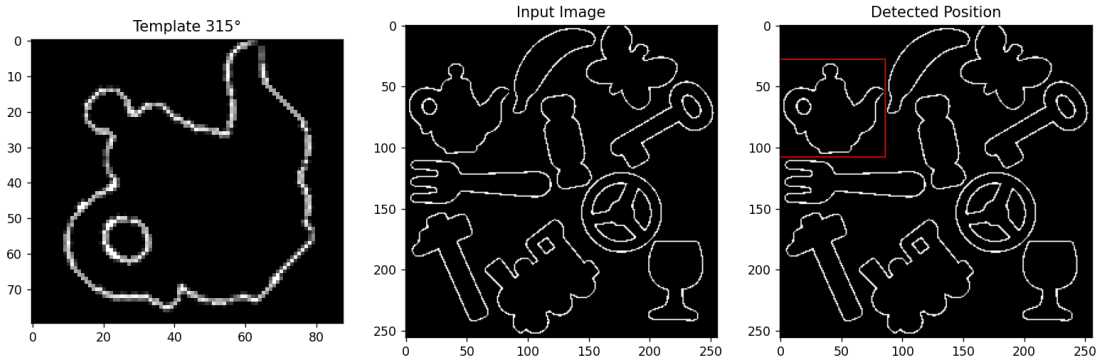
旋轉 259 度



旋轉 270 度



旋轉 294 度



旋轉 315 度