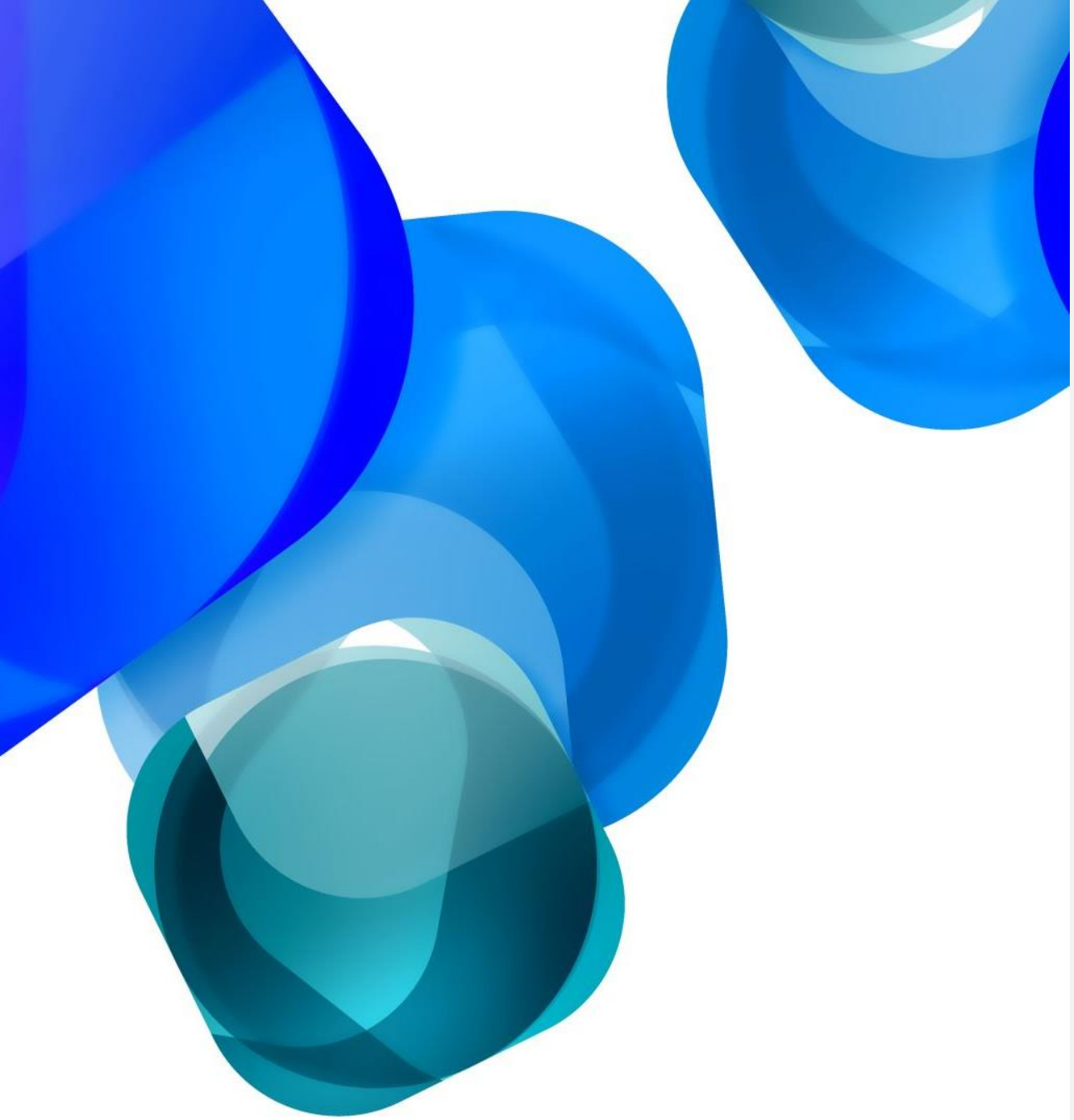


HW3
Object Detection

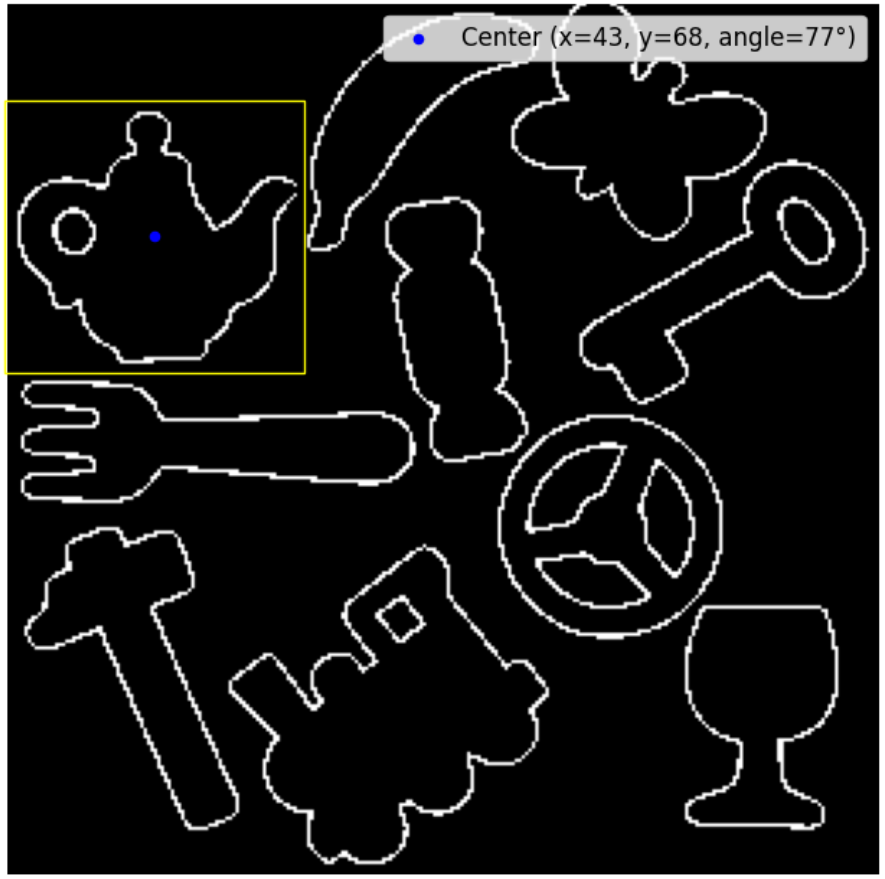
S1154018 資工三 王宇森

成果

- 可對任意整數角度進行辨識
- 每次辨識時間約為40秒

Template: img/template/Template_77.png

Reference: img/Refernce.png



Generalized Hough transform 整體流程

- 首先，將模板和參照分別進行Canny轉換，並計算r_table，再來進行「投票」，找到關鍵點並繪製結果，以下將詳細介紹過程

- 因為會大量的用到sin和cos計算，因此預先計算兩個長度為360的陣列，代表sin和cos在每個角度的值，並使用lru_cache，在重複使用時不必重新計算，來增加執行效率

```python
def generalized_hough_transform(reference, template, angle_step):
    # 1. 邊緣檢測
    template_edges = cv2.Canny(template, 50, 150)
    reference_edges = cv2.Canny(reference, 50, 150)
    # 2. 計算 R-table
    r_table = compute_r_table(template_edges)
    # 3. 創建accumulator, 所有角度的cos,sin值
    rows, cols = reference_edges.shape
    cos_values, sin_values = get_cos_sin_values(angle_step)
    accumulator = np.zeros((rows, cols, len(cos_values)), dtype=np.int32)

    # 4. 獲取reference所有邊緣點
    edge_points = np.argwhere(reference_edges == 255)
    # 5. 投票過程
    for y, x in edge_points:
        # 根據 R-table 投票
        for phi, vectors in r_table.items():
            for v in vectors:
```

```python
@lru_cache(None)
def get_cos_sin_values(angle_step):
    angles_deg = np.linspace(0, 360, 360 // angle_step, endpoint=False)
    angles_rad = np.deg2rad(angles_deg)
    cos_values = np.cos(angles_rad)
    sin_values = np.sin(angles_rad)
    return cos_values, sin_values
```

計算R_Table

- R_table以dict的方式儲存，將template的中心點定為參照點，並將Canny後所有的邊和該中心點進行計算dx,dy的arctan值，即為r_table的key，並將[dx,dy]加入r_table對應的value（list），以做後續投票使用

```python
def compute_r_table(template_edges):
    rows, cols = template_edges.shape
    r_table = {}
    yc = rows / 2
    xc = cols / 2
    for y in range(rows):
        for x in range(cols):
            if template_edges[y, x] > 0:
                dx = x - xc
                dy = y - yc
                angle = math.atan2(dy, dx)
                # r = math.hypot(dx, dy) # sqrt(x*x + y*y)
                angle = math.degrees(angle)
                # converts an angle from radians to degrees.
                if angle not in r_table:
                    r_table[angle] = []
                r_table[angle].append([dx, dy])
    return r_table
```

GHT-投票過程

- 先將reference的邊緣點全部抓出。
- 將每個點和剛剛的R_table的所有[x,y]向量進行sin,cos的轉換得到一個新的點，並在該點的投票+1(除了x,y，還要記下角度，所以總共有cols*rows*360個候選點)，找出票數最高的點後回傳，並用matplotlib顯示結果。

```
# 4. 獲取reference所有邊緣點
edge_points = np.argwhere(reference_edges == 255)
# 5. 投票過程
for y, x in edge_points:
    # 根據 R-table 投票
    for phi, vectors in r_table.items():
        for v in vectors:
            dx, dy = v
            # 計算投票的 xc 和 yc 值
            # (直接乘以sin&cos陣列，產生length為360//angle_step的陣列)
            xc = x - dx * cos_values + dy * sin_values
            yc = y - dx * sin_values - dy * cos_values
            # 轉換為整數並確保在圖像範圍內
            xc = np.round(xc).astype(int)
            yc = np.round(yc).astype(int)
            # 篩選出有效的座標
            valid = (0 <= xc) & (xc < cols) & (0 <= yc) & (yc < rows)
            # 更新累加器
            valid_x_c = xc[valid]
            valid_y_c = yc[valid]
            valid_angle_idx = np.arange(len(cos_values))[valid]
            accumulator[valid_y_c, valid_x_c, valid_angle_idx] += 1
x, y, angle_idx = np.unravel_index(np.argmax(accumulator), accumulator.shape)
angle = angle_idx * angle_step
return x, y, angle
```

測試方法

- 為了方便使用者測試任一角，請使用者輸入想要測試的角度，有函式能將原本0度的template旋轉指定的角度，再進行辨識。
- 其中angle_step參數，代表上述投票時考量的角度間距，預設為1代表0-359都會考慮到，若設為90則只會考慮0,90,180,270

```
def main():
    reference_path = 'img/Refernce.png'
    reference = cv2.imread(reference_path, cv2.IMREAD_GRAYSCALE)
    angle_step = 1
    angle =  int( input('輸入想要辨識的角度：') )
    angle = angle % 360
    print(angle)
    template_path = 'img/template/Template_' + str(angle) + '.png'
    if not os.path.exists(template_path):
        rotate_template(angle)
    template = cv2.imread(template_path, cv2.IMREAD_GRAYSCALE)
    x, y, angle = generalized_hough_transform(reference, template, angle_step)
    display_result(reference, x, y, angle, box_size=template.shape,
                   ref=reference_path, tmp=template_path, template_img=template)
```

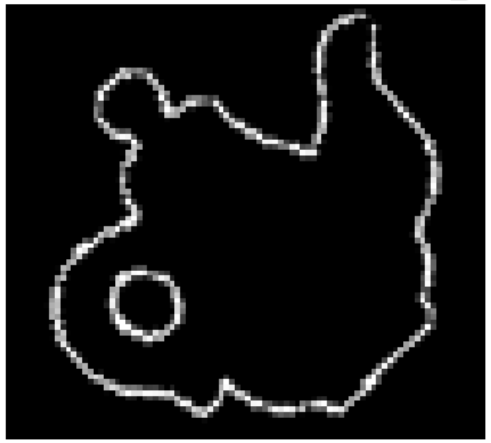
旋轉圖片

- 搭配CV2內建函式實作，方便測試

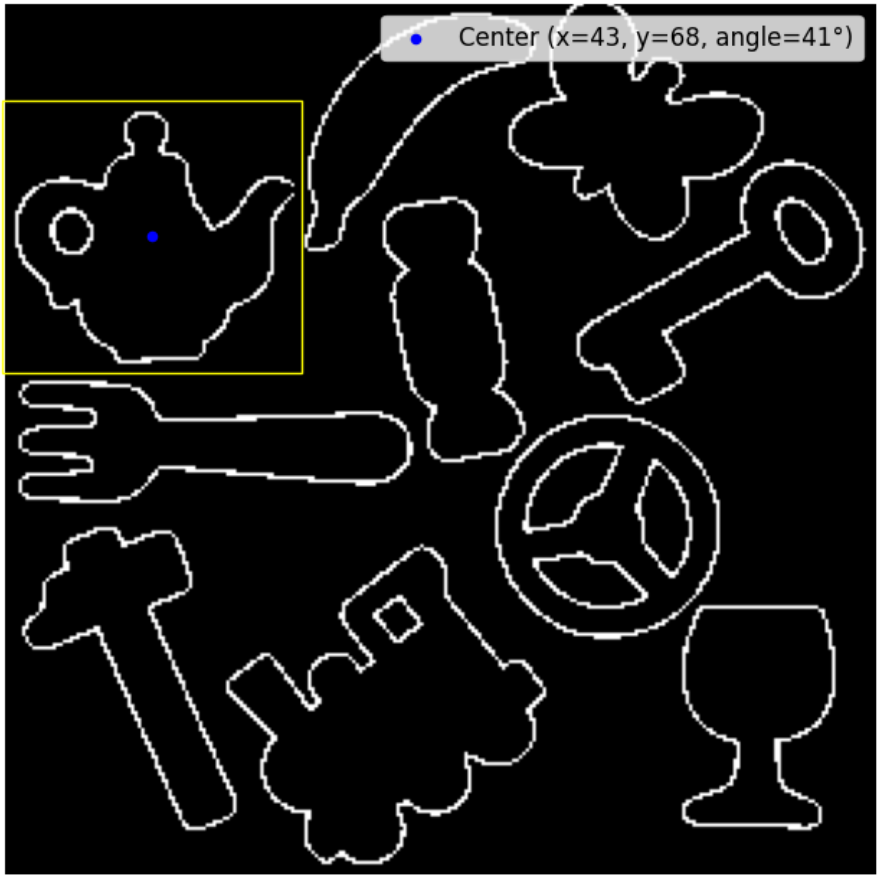
```python
def rotate_template(angle):
    template = cv2.imread('img/template/Template.png', cv2.IMREAD_GRAYSCALE)
    template_path = 'img/template/Template.png'
    base_name = os.path.splitext(os.path.basename(template_path))[0]  #img/template/Template
    output_dir = os.path.dirname(template_path)  # img/template
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)
    h, w = template.shape
    center = (w // 2, h // 2)

    rotation_matrix = cv2.getRotationMatrix2D(center, angle, 1.0)  # 1.0是縮放比例
    rotated_image = cv2.warpAffine(template, rotation_matrix, (w, h))

    output_filename = f"{base_name}_{angle}.png"
    output_path = os.path.join(output_dir, output_filename)
    cv2.imwrite(output_path, rotated_image)
```

其他成果

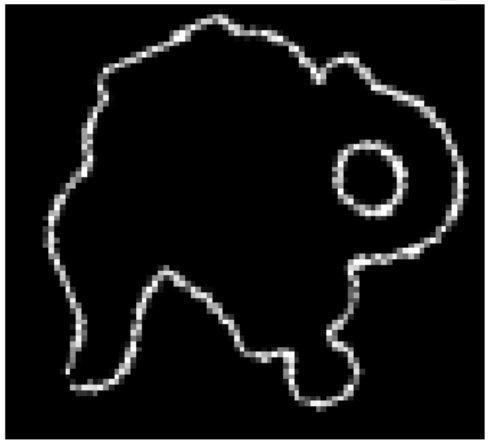
Template: img/template/Template_41.png



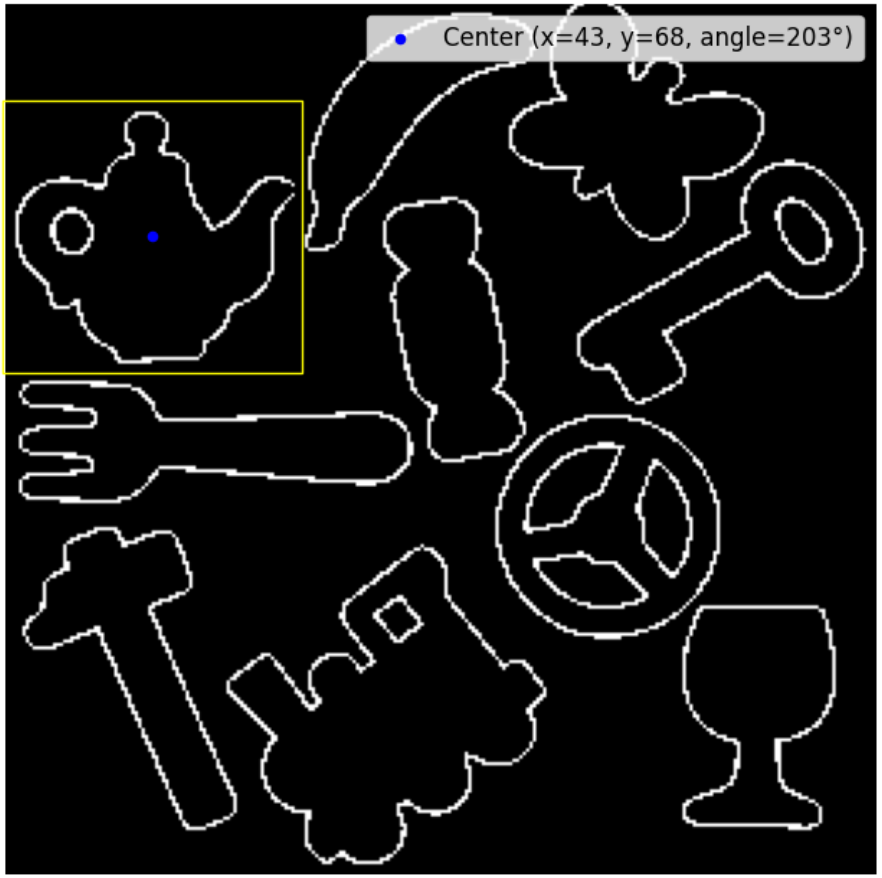
Reference: img/Refernce.png



Template: img/template/Template_203.png



Reference: img/Refernce.png



其他心得

- 在實作時，我不斷地在嘗試如何減少計算時間，例如一開始提到的用lru_cache來儲存sin,cos陣列外，亦將計算時的遍歷次數減少，像是投票時從0-359度各計算一次，變成直接用numpy的陣列乘法來操作，搭配valid找出有效點進行投票，雖然演算法複雜度沒有實質降低，但運作的時間從原本的2分鐘左右降到40秒，是很好的成果。