Function Documentation:`find_and_sort_contours`

1 Description

The `find_and_sort_contours` function processes a grayscale image to identify, group, and sort contours based on their area and perimeter. The function also calculates distances of contour centers from a given reference point and sorts the contours accordingly.

2 Function Definition

```
def find_and_sort_contours(gray_image, min_thresh, max_thresh, last_center):
    # Apply range threshold
    range_threshold_image = cv2.inRange(gray_image, min_thresh, max_thresh)
    range_threshold_image = cv2.medianBlur(range_threshold_image, 9)

    # Fill in holes
    kernel_size = 9
    kernel = np.ones((kernel_size, kernel_size), np.uint8)

    # Apply the closing operation
    range_threshold_image = cv2.morphologyEx(range_threshold_image, cv2.MORPH_CLOSE,
    ↪  kernel)

    # Find contours
    contours, _ = cv2.findContours(range_threshold_image, cv2.RETR_TREE,
    ↪  cv2.CHAIN_APPROX_SIMPLE)
    sorted_contours = sorted(contours, key=cv2.contourArea, reverse=True)

    # sort contours
    grouped_contours = []
    weights = []
    index = 0
    while index < len(sorted_contours):
        contour = sorted_contours[index]

        contour_list = []
        contour_list.append(contour)

        # calculate the center of the contour
        M = cv2.moments(contour)
        # Calculate the center (centroid) of the contour
        if M["m00"] != 0:
            centerX = int(M["m10"] / M["m00"])
            centerY = int(M["m01"] / M["m00"])
        else:
```

```
            centerX, centerY = 0, 0  # Set default values if m00 is zero to avoid division
            ↪ by zero


        # Calculate area
        area = cv2.contourArea(contour)

        # Calculate perimeter
        if area > 0:
            perimeter = cv2.arcLength(contour, True)  # True indicates the contour is
            ↪ closed

            index_next = index + 1
            while index_next < len(sorted_contours):
                contour_next = sorted_contours[index_next]
                area2 = cv2.contourArea(contour_next)
                if area2 > 0:
                    perimeter2 = cv2.arcLength(contour_next, True)  # True indicates the
                    ↪ contour is closed
                    if is_contour_inside(contour_next, contour):
                        if args['--include_holes']:
                            perimeter += perimeter2  # True indicates the contour is
                            ↪ closed
                            area -= area2
                            contour_list.append(contour_next)
                        sorted_contours.pop(index_next)
                    else:
                        index_next += 1
                else:
                    sorted_contours.pop(index_next)

            grouped_contours.append([np.array([centerX, centerY]), area, perimeter,
            ↪ contour_list])

            # calculate distance from center/area, g
            weights.append(np.sqrt((centerX-last_center[0])**2 +
            ↪ (centerY-last_center[1])**2)/area)
            index += 1
        else:
            sorted_contours.pop(index)

    sorted_indices = np.argsort(np.array(weights))
    last_center = grouped_contours[sorted_indices[0]][0]

    return sorted_contours, grouped_contours, sorted_indices, last_center, weights
```

3 Function Explanation

3.1 Step-by-Step Breakdown

Function 1: Apply Range Threshold

Apply a range threshold to the grayscale image to create a binary image where pixels within the specified threshold range are set to 255 (white), and others are set to 0 (black). Apply median blur to reduce noise.

```
range_threshold_image = cv2.inRange(gray_image, min_thresh, max_thresh)
range_threshold_image = cv2.medianBlur(range_threshold_image, 9)
```

Explanation: The function thresholds the grayscale image to segment features of interest and reduces noise using a median blur.

Function 2: Fill in Holes

Use morphological operations to fill in small holes in the thresholded image and apply a closing operation with a specified kernel size.

```
kernel_size = 9
kernel = np.ones((kernel_size, kernel_size), np.uint8)
range_threshold_image = cv2.morphologyEx(range_threshold_image, cv2.MORPH_CLOSE, kernel)
```

Explanation: Morphological closing is used to clean up the thresholded image by closing small gaps and holes.

Function 3: Find and Sort Contours

Find contours in the processed image, sort them by area in descending order, and group contours based on containment.

```
contours, _ = cv2.findContours(range_threshold_image, cv2.RETR_TREE,
↪  cv2.CHAIN_APPROX_SIMPLE)
sorted_contours = sorted(contours, key=cv2.contourArea, reverse=True)
```

Explanation: Contours are identified and sorted by their area to prioritize larger contours.

Function 4: Group Contours

Group contours based on their containment and calculate their center, area, and perimeter. Store weights based on distance from a reference center and area.

```
grouped_contours = []
weights = []
index = 0
while index < len(sorted_contours):
    contour = sorted_contours[index]
    contour_list = []
    contour_list.append(contour)
    M = cv2.moments(contour)
    if M["m00"] != 0:
        centerX = int(M["m10"] / M["m00"])
        centerY = int(M["m01"] / M["m00"])
    else:
        centerX, centerY = 0, 0
    area = cv2.contourArea(contour)
    if area > 0:
        perimeter = cv2.arcLength(contour, True)
        index_next = index + 1
        while index_next < len(sorted_contours):
            contour_next = sorted_contours[index_next]
            area2 = cv2.contourArea(contour_next)
            if area2 > 0:
                perimeter2 = cv2.arcLength(contour_next, True)
                if is_contour_inside(contour_next, contour):
                    if args['--include_holes']:
                        perimeter += perimeter2
                        area -= area2
                        contour_list.append(contour_next)
                    sorted_contours.pop(index_next)
                else:
                    index_next += 1
            else:
                sorted_contours.pop(index_next)
        grouped_contours.append([np.array([centerX, centerY]), area, perimeter,
        ↪  contour_list])
        weights.append(np.sqrt((centerX-last_center[0])**2 +
        ↪  (centerY-last_center[1])**2)/area)
        index += 1
    else:
        sorted_contours.pop(index)
```

Explanation: Contours are grouped if one is contained within another. The function calculates the center, area, and perimeter for each group. Weights are computed based on the distance from a reference point and the contour area.

Function 5: Sort by Weights

Sort the grouped contours based on their calculated weights and update the last center.

```
sorted_indices = np.argsort(np.array(weights))
last_center = grouped_contours[sorted_indices[0]][0]
```

Explanation: Contours are sorted based on their weights (distance-to-area ratio). The 'last_center' is updated to the center of the highest-weight contour.

4 Conclusion

The `find_and_sort_contours` function processes a grayscale image to find, group, and sort contours by their area and perimeter. It also calculates and uses weights based on distance and area to determine the significance of each contour.