Pol Pujol Santaella

Artificial Vision & Pattern Recognition

Assignment #1

Automatic Pieces detection and Quantification in thermographic images

UNIVERSITAT ROVIRA I VIRGILI

2025

INDEX

1. Introduction and System Objective 3
2. Methodology & Processing Pipeline 4
3. Algorithms & Techniques Employed 5
    3.1. Image Loading & Preprocessing 5
        3.1.1. Resize Image Aspect 5
        3.1.2. Preprocess_image 5
    3.2. Line Feature Detection 5
        3.2.1. Detect Lines 5
    3.3. Circle Detection 5
        3.3.1. Detect Circles 5
    3.4. Custom Geometric Filtering 5
        3.4.1. Get Complete Pieces 5
4. Visualization & Output 6
5. Conclusion 7

1. Introduction and System Objective

The objective of this system is to automatically detect, localize, and quantify complete circular pieces within thermographic images. These images often contain partially visible components or objects located near the borders, which can cause traditional detection methods to miscount or misinterpret the scene.

To address this issue, the system integrates classic image processing techniques with geometric filtering strategies. The core approach combines circle detection via the Hough Transform with robust line detection used to identify image borders or string thermal edges. By evaluating the geometric relationship between each circle and the detected lines, the system determines whether a piece is fully visible or truncated.

This methodology ensures reliable quantification of complete pieces, making the system suitable for inspection tasks in thermal sealing applications.

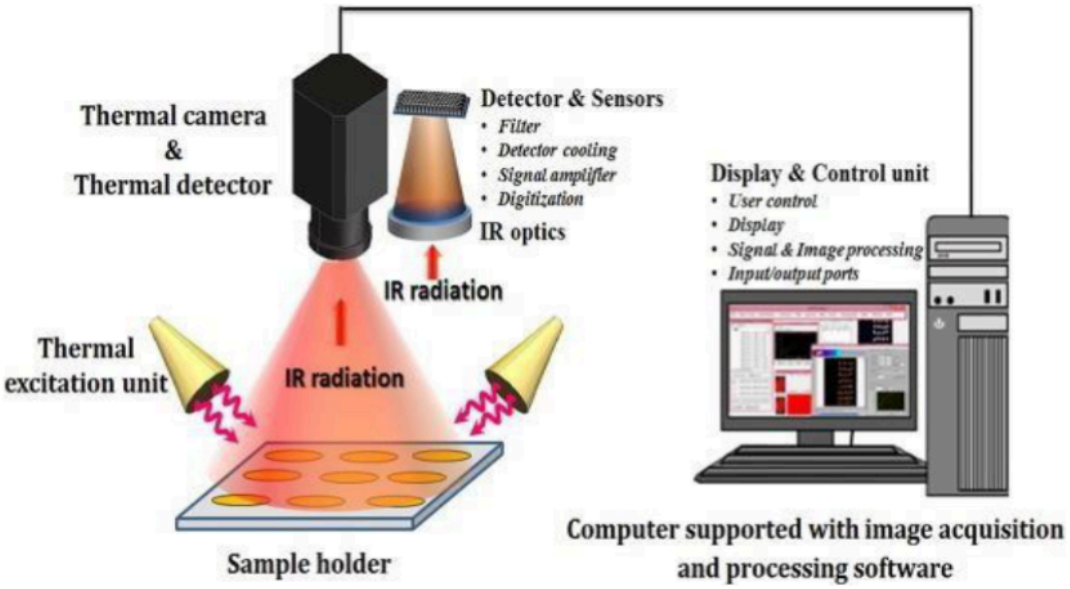


Figure 1: System overview.

2. Methodology & Processing Pipeline

The complete workflow is implemented as a sequential pipeline applied to each image in the dataset:

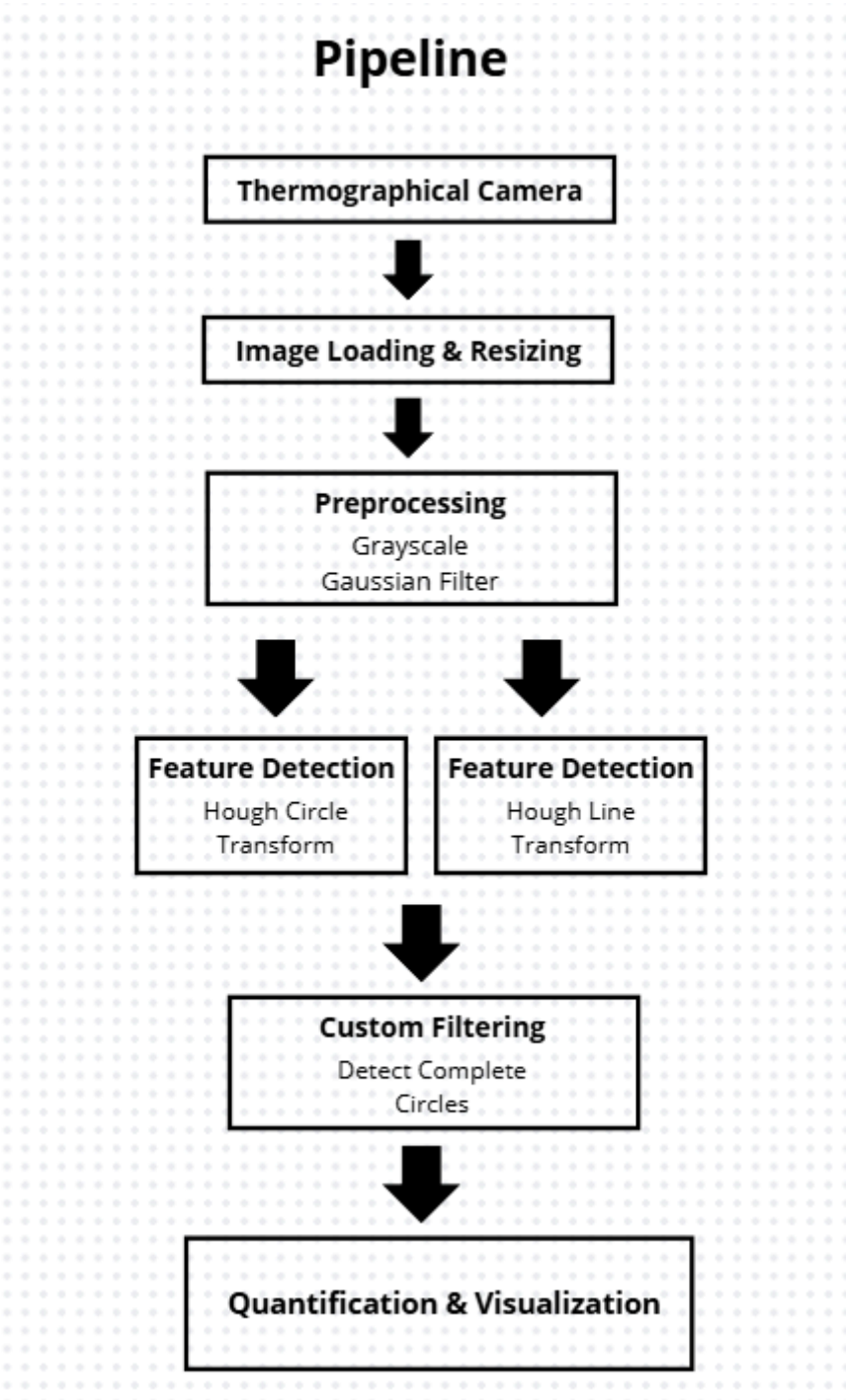
1. **Image Loading and Resizing**
   All images are resized to a uniform width to standardize the expected scale of features across the dataset.

2. **Preprocessing**
   The images undergo grayscale conversion and Gaussian smoothing to reduce noise and enhance structural edges.

3. **Feature Detection**
   Two types of features are extracted in parallel:
   - Circular structures via Hough Circle Transform
   - Linear structures via filtered Sobel response followers by Hough Line Transform

4. **Custom Filtering**
   Circles intersected by any detected line (including synthetic borders) are considered incomplete and removed.

5. **Quantification and Visualization**
   The final output displays original images, preprocessed images, detected lines, circles detected, and then the final result with total complete circles detected.


Pipeline
Thermographical Camera
Image Loading & Resizing
Preprocessing
Grayscale
Gaussian Filter
Feature Detection
Hough Circle Transform
Feature Detection
Hough Line Transform
Custom Filtering
Detect Complete Circles
Quantification & Visualization

3. Algorithms & Techniques Employed

3.1. Image Loading & Preprocessing

3.1.1. Resize Image Aspect

Functions: *resize_image_aspect / load_images*

The input image is resized to a fixed width of 800 pixels. The height is scaled proportionally to avoid geometric distortion. The *cv2.INTER_AREA* interpolation method is used, which is optimal for downsampling and preserves detail efficiently.

```
def resize_image_aspect(image, target_width):

    h, w = image.shape[:2]
    new_h = int(target_width * h / w)
    return cv2.resize(image, (target_width, new_h), interpolation=cv2.INTER_AREA)

def load_images(path):

    files = sorted([f for f in os.listdir(path) if f.lower().endswith(('.png'))])
    images = []

    for name in files:
        img = cv2.imread(os.path.join(path, name))
        if img is None:
            print(f"Images unable to be loaded: {name}")
            continue
        images.append(resize_image_aspect(img, TARGET_WIDTH))

    if images:
        h, w = images[0].shape[:2]
        print(f"Loaded {len(images)} images. Resized to {w}x{h}")
    else:
        print("No images loaded.")

    return images
```

3.1.2. Preprocess Image
Function: *preprocess_image*

The preprocessing phase consists of:

- Converting the image from BGR to grayscale
- Applying a 5x5 Gaussian blur (sigma = 0) to reduce high-frequency noise and improve gradient-based detection stability.

```python
def preprocess_image(img):

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)
    return blurred
```

3.2. Line Feature Detection

3.2.1. Detect Lines
Function: *detect_lines*

Line detection focuses on identifying significant horizontal and vertical structures, particularly the image boundaries. The process includes:

1. **Convolution with Sobel Y-like Kernel**
   A custom 3x3 kernel is applied to emphasize horizontal lines by detecting strong vertical gradients.

y-direction kernel

-1	-2	-1
0	0	0
1	2	1

2. **Hough Lines Probabilistic Transform (cv2.HoughLinesP)**
   Parameters:
   - ρ: 1 (1-pixel distance resolution)
   - θ: π / 180 (1-degree angle resolution)
   - threshold: 50
   - minLineLength: 400.

   These settings ensure only major continuous lines are detected

3. **Synthetic Borders**
   To ensure border consistency, artificial left and bottom boundaries are appended to the detected lines list.

```python
def detect_lines(img, kernel):

    conv = cv2.filter2D(img, -1, kernel)
    h, w = conv.shape
    lines = [
        np.array([0, h, w, h]), # Bottom border
        np.array([0, 0, 0, h]) # Left border
    ]

    detected_lines = cv2.HoughLinesP(conv, 1, np.pi / 180, 50, None, 400, 0)

    if detected_lines is not None:
        lines.extend([l[0] for l in detected_lines])

    return lines
```

3.3. Circle Detection

3.3.1. Detect Circles

Function: *detect_circles*

The detection of circular components uses the Hough Gradient method *(cv2.HOUGH_GRADIENT)*, which combines edge information and gradient direction to efficiently estimate circle centers and radii.

Parameter	Value	Rationale
minDist	200	Prevents duplicate circle detections
param1	50	Upper Canny threshold for edge detection
param2	45	Accumulator threshold for circle confirmation
minRadius / maxRadius	100/200	Expected size range of pieces

The parameter tuning focuses on ensuring robustness and minimizing the detection of partial arcs.

```
HOUGH_PARAMS = {
    'dp': 1.0,
    'minDist': 200,
    'param1': 50,
    'param2': 45,
    'minRadius': 100,
    'maxRadius': 200
}

def detect_circles(img, params):

    return cv2.HoughCircles(
        img,
        cv2.HOUGH_GRADIENT,
        dp=params['dp'],
        minDist=params['minDist'],
        param1=params['param1'],
        param2=params['param2'],
        minRadius=params['minRadius'],
        maxRadius=params['maxRadius']
    )
```

3.4. Custom Geometric Filtering

3.4.1. Identifying Complete Pieces

This is the core algorithm for distinguishing complete from incomplete pieces, utilizing the line features detected in Section 3.2.

Algorithm: The method iterates through every detected circle and every detected line (including image borders). If any line segment passes “inside” the boundary of a circle, that circle is marked as incomplete.

1. **Perpendicular Distance Calculation:** The distance d from the circle center (x, y) to the infinite line defined by the segment points (x1, y1) and (x2, y2) is calculated using the standard point-to-line distance formula:

$$d = \frac{|(y_2 - y_1)x - (x_2 - x_1)y + x_2y_1 - y_2x_1|}{\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}}$$

2. **Filtering Condition:** A circle of radius r is considered incomplete if the distance d to a line meets the following condition:

$$d - r < -10$$

This means the line segment is at a distance d from the center, where d is less than the radius r minus a 10-pixel buffer. This condition implies the line passes through a region that should be part of the complete circle body, effectively signifying that the circle is “cut” by a strong edge (either the image boundary or an internal thermal feature)

```
def line_point_distance(x, y, x1, y1, x2, y2):

    denom = np.hypot(y2 - y1, x2 - x1)
    if denom == 0:
        return float('inf')
    num = abs((y2 - y1) * x - (x2 - x1) * y + x2 * y1 - y2 * x1)
    return num / denom

def get_complete_pieces(circles, lines):

    if circles is None:
        return 0, np.empty((0, 3), dtype=np.uint16)

    circles = np.uint16(np.around(circles))
    complete = []

    for x, y, r in circles[0]:
        is_complete = True
        for x1, y1, x2, y2 in lines:
            if line_point_distance(x, y, x1, y1, x2, y2) - r < -10:
                is_complete = False
                break
        if is_complete:
            complete.append([x, y, r])
    return len(complete), np.array(complete)
```

4. Visualization & Output

Function: *draw_circles*

The final visualization step highlights the results for analysis:

- **Lines**: All line segments (detected and synthetic borders) used for filtering are drawn in Green.
- **Complete Circles:** The boundary of each filtered, complete circle is drawn in Black, and the center is marked in Red.
- **Quantification:** The total count of complete pieces is displayed in the top-left corner of the output image.

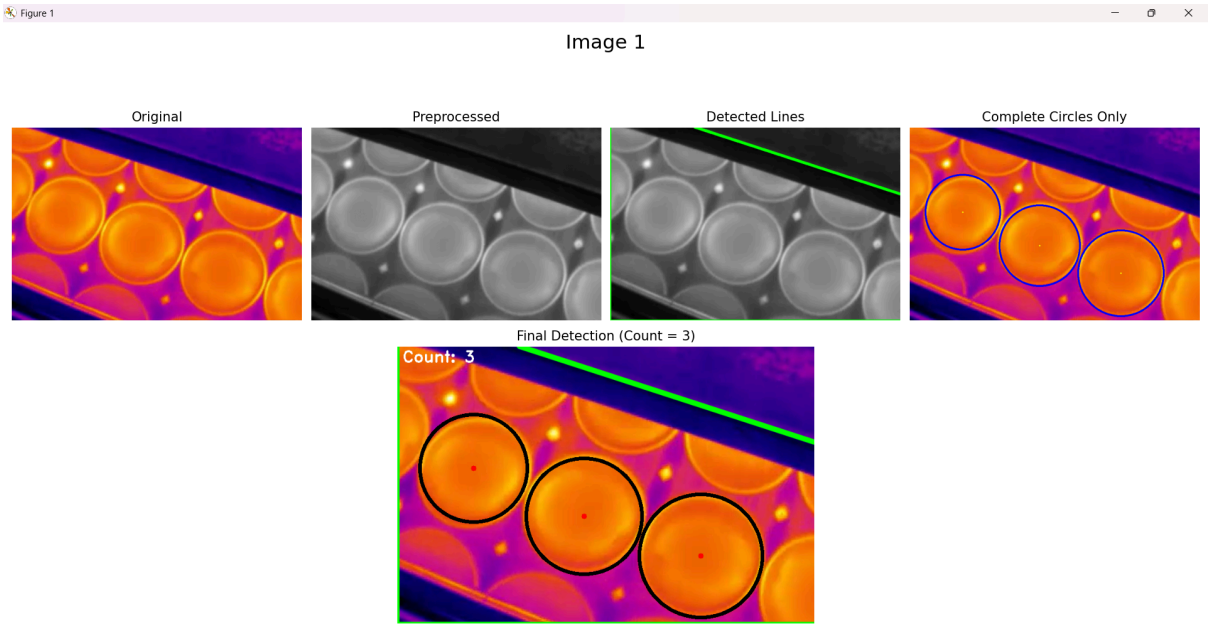
The main function presents a comprehensive visual output showcasing the full pipeline: Original Image, Preprocessed Image, Detected Lines, Detected Complete Circles, and the Final Result.

5. Outcomes

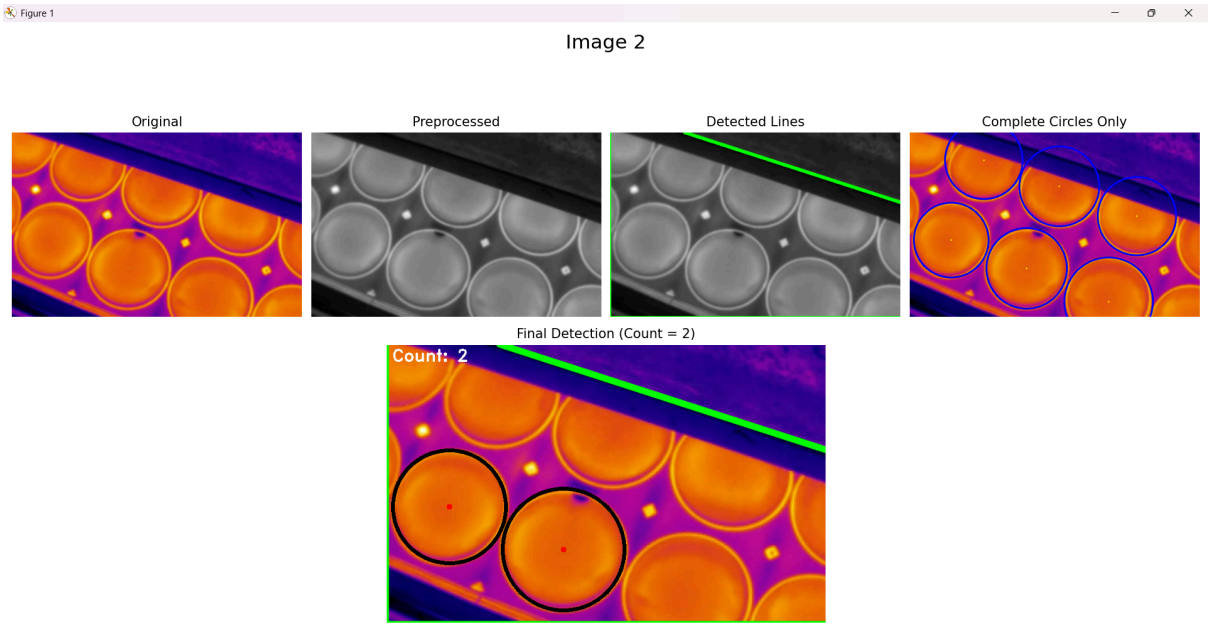
The method was applied to seven thermographic images, producing reliable detections across varying thermal gradients and object positions. In all cases, the geometric filtering strategy successfully removes circles truncated by image borders or strong internal edges.

The results for each image correspond to consistent and correct counts of complete pieces.

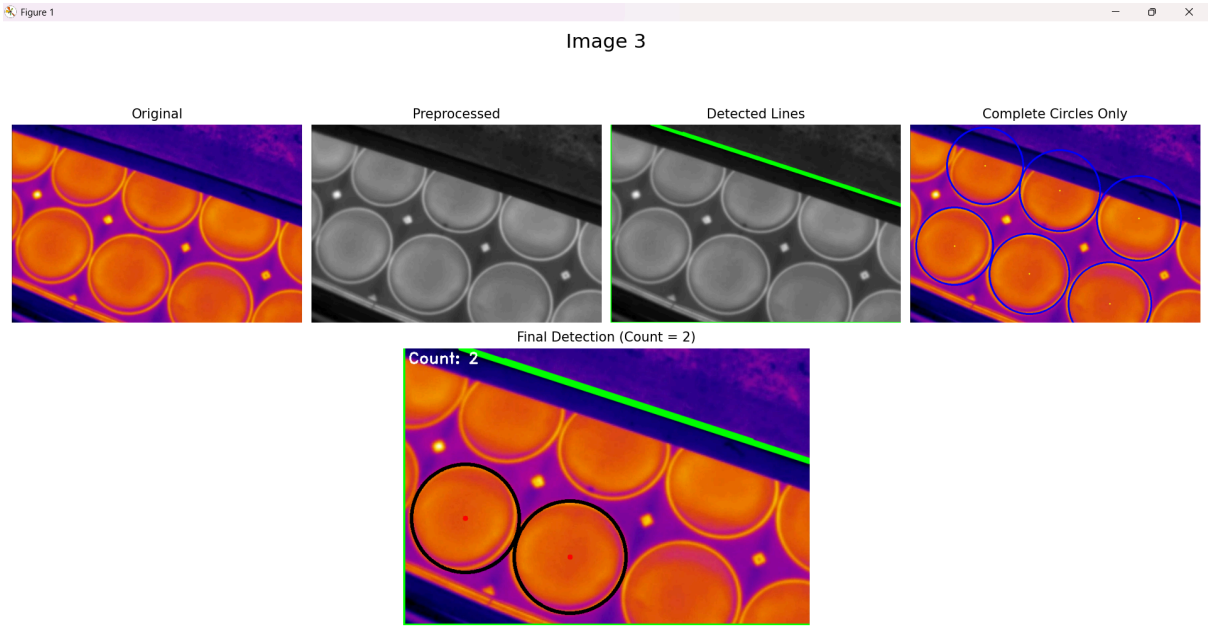
‘image1.png’



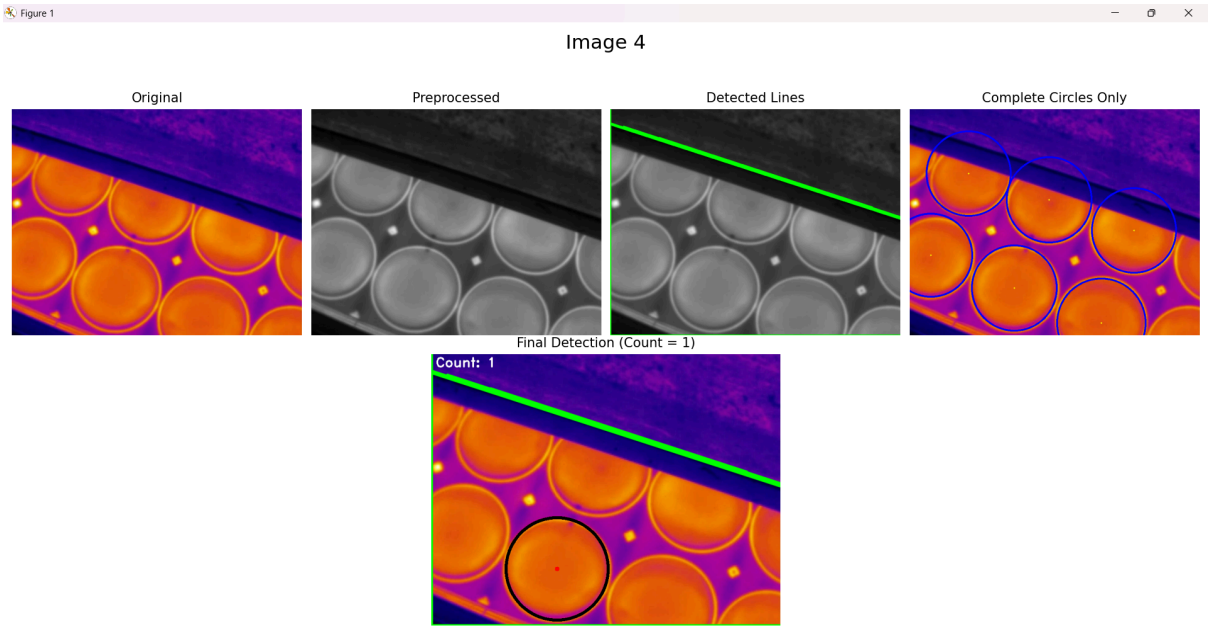
‘image2.png’



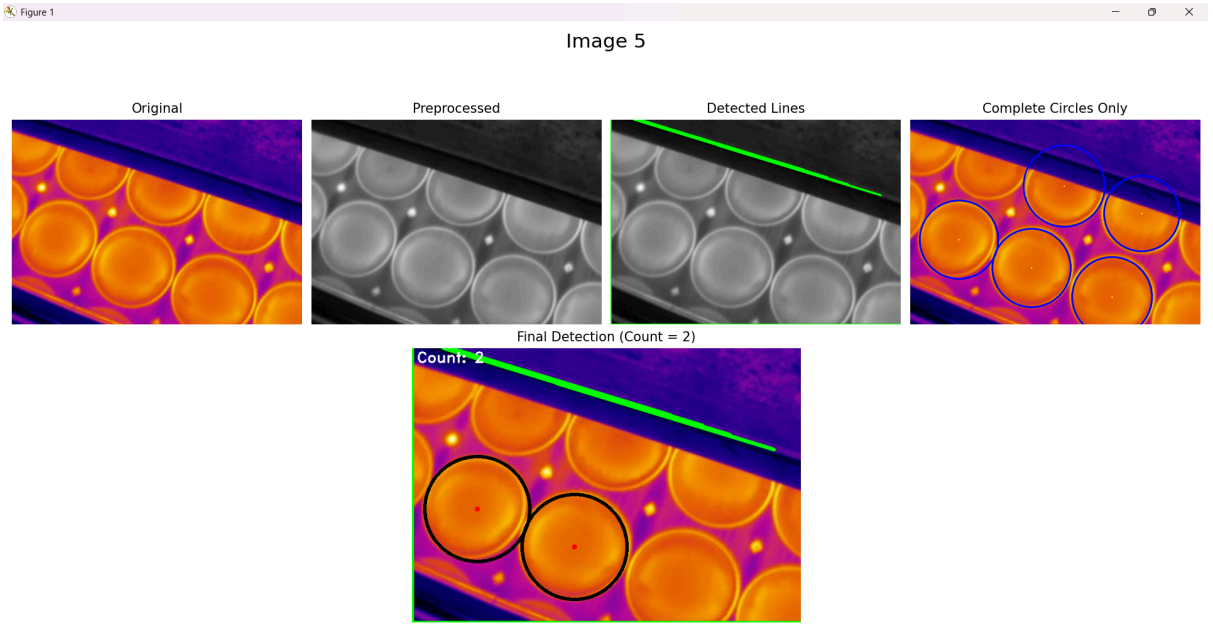
‘image3.png’



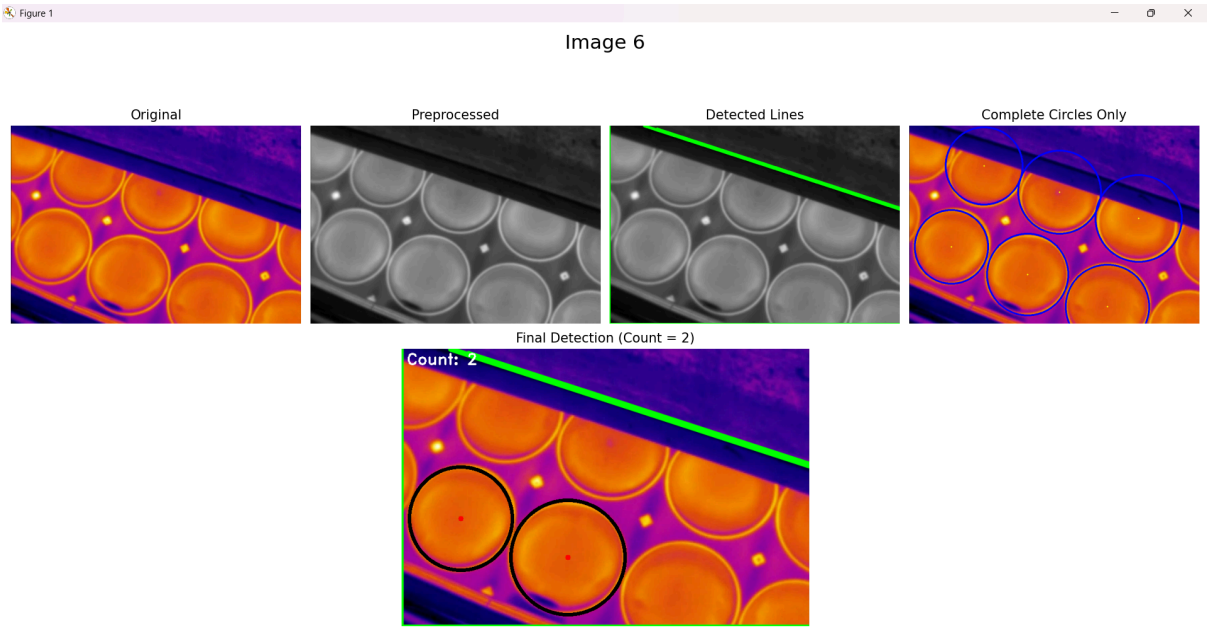
‘image4.png’



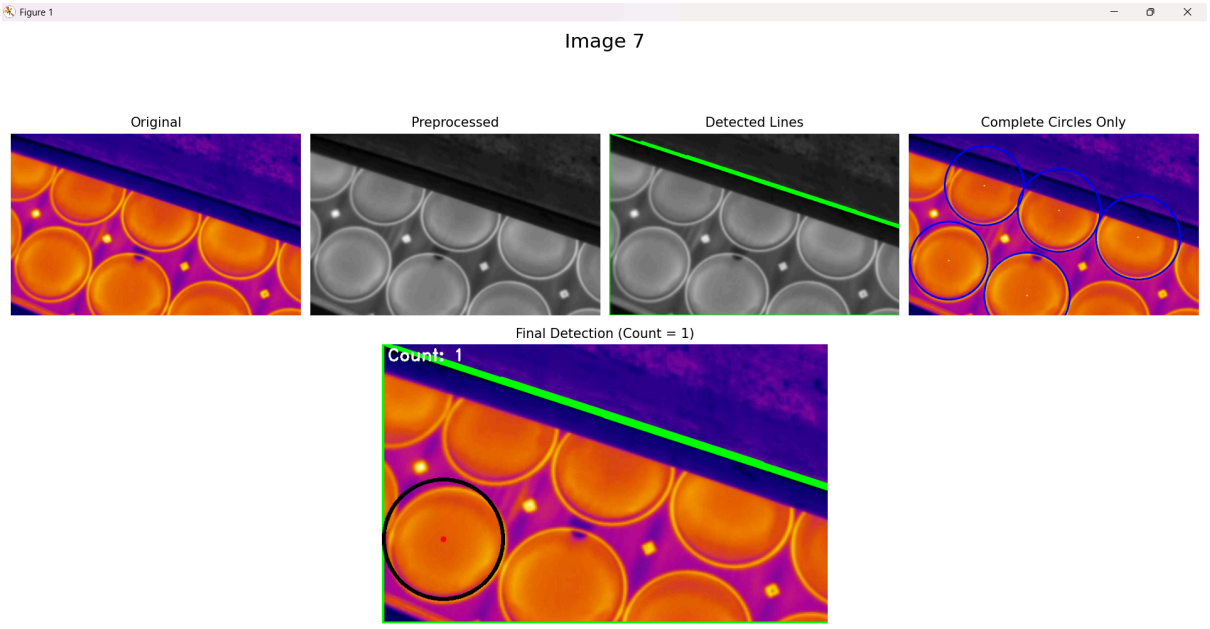
‘image5.png’



‘image6.png’



‘image7.png’



6. Conclusion

The implemented system demonstrates a robust approach for identifying and quantifying complete circular pieces in thermographic imagery. By combining gradient-based preprocessing, Hough feature detection, and geometric filtering, the pipeline minimizes false detections and ensures accurate classification of complete components.

This approach can be extended to other industrial inspection tasks requiring shape-based detection under noisy or thermally distorted conditions.