Automatic Pieces Detection and Quantification in Thermographic Images

Hairton Javier Sarmiento Yahuarcani

School of Engineering. Universitat Rovira I Virgili

AVRP: Artificial Vision and Pattern Recognition

Domènec Savi Puig Valls

Hatem Abdellatif Fatahallah Ibrahim Mahmoud

Saddam Abdulwahab

November 24th, 2025

Introduction

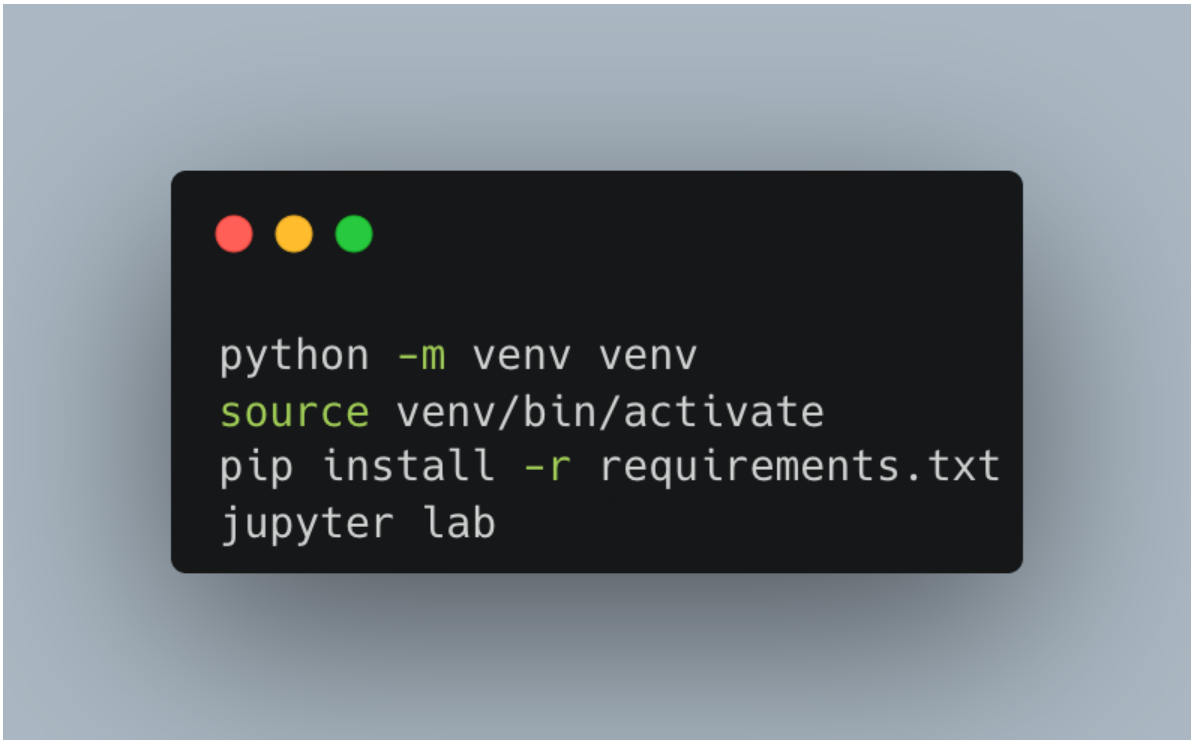
Computer vision is a field of AI, which basically teaches to a computer to see objects from images or videos. Applying and understanding different techniques, it is possible to have intuition of how this field has evolved and how it has been useful to solve problems at researching and private sector level. Nowadays, computer vision with the power of AI is even able to give outstanding solutions to different problems. However, the classical approach of computer vision is crucial to be understood. Hence, the knowledge can be used in a dynamic way, because it is not always the case where a ML model should solve a problem like using a forklift to move something that you can lift and move with your arms.

Problem Description and Code

In this assignment, it was provided 7 images that were taken using a thermographic camera. The goal of this work is to be able to count and show complete objects of the images. In this case circles. It is important to say that the images will need to preprocess the images and apply feature detection methods.

The code for this assignment was done in the following repo, you will find the documentation and the image too. Besides, you will find the same structure in the Moodle.

For running the code in your local, please run the following commands in the same folder where the code is. This will create an environment ready with the libraries to explore the code



Methodology

Preprocessing

Figure 1

Pre-processing images

```
image = cv2.imread('image1.png')
original_image = image.copy()  # Make a copy for displaying the original image

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Apply GaussianBlur to reduce noise and improve contour detection
blurred = cv2.GaussianBlur(gray, (5, 5), 0)
```

First, each image is converted from RGB (three channels) to grey scale (one channel). This allows to remove unnecessary noise and simplifies edge detection. Besides, color information is not important to find the circles of the images.

Then, it is applied a gaussian smoothing to have a blur image. It is used a kernel (5, 5), which means that a 5x5 pixel is averaging values with a gaussian weight distribution. This helps the circle detection because the thermal images usually have random hot pixels, temperature texture, small visual artifacts, which may produce false circles.

Algorithms and Techniques Used

The Hough Circle Transform is a computer-vision technique used to automatically detect circular shapes in an image. It works by analysing the gradients (edges) of the image and voting for possible circle canters and radii in a mathematical “accumulator” space.

The process works as follows:

The algorithm uses gradient information (like Canny edge detection) to identify pixels that belong to potential circular edges. Voting in Parameter Space

For each edge pixel, the algorithm computes all possible circles that could pass through that pixel.

Each possible circle “votes” in an accumulator matrix for: its centre position (x, y) and its radius Finding Peaks

Locations with the highest number of votes correspond to actual circles in the image. These peaks become the detected circles returned by the algorithm.

This method is effective because it can detect circles even when edges are noisy, partially missing, or not perfectly bright.

Circle detection and Parameter tuning

To find the circle, the following parameters should be tunned:

Inverse Accumulator Resolution (dp)

Controls the resolution of the circle accumulator relative to the image.

dp = 1 → accumulator same size as image (more precise)

Higher dp → less resolution, faster but less accurate.

If circles are not detected → lower dp.

If too many false circles → increase dp.

Minimum Distance Between Centres (minDist)

Prevents multiple detections of the same circle.

Represents how far apart detected circle centers must be.

If circles are close → decrease minDist.

If duplicates appear → increase minDist.

High Threshold for Canny Edge Detection (param1)

Controls how edges are detected before circle voting.

Lower values → more edges (more sensitive).

Higher values → fewer edges (stricter).

If circles are missing → increase sensitivity by lowering param1.

If noise is detected as circles → increase param1.

Accumulator Threshold (Detection Sensitivity, param2)

This is the most important parameter.

High param2 → stricter detection (fewer circles, but more reliable)

Low param2 → more sensitive (detects weaker circles, but may produce false positives)

If no circles are detected → lower param2.

If too many false circles appear → raise param2.

Size Range of Circles (minRadius / maxRadius)

Restrict the search to circles within known size limits.

Essential when the object size is predictable (like in your thermographic pieces).

If small unwanted circles appear → increase minRadius.

If real circles disappear → increase maxRadius.

Results

Image 1

```python
# Detect circles using the Hough Circle Transform
# dp=1 → resolution of the accumulator equals the image resolution
# minDist=40 → minimum distance between detected circle centers
# param1=40 → upper threshold for the Canny edge detector
# param2=45 → threshold for center detection (higher = stricter)
# minRadius and maxRadius → limits for detected circle sizes
circles = cv2.HoughCircles(
    blurred, cv2.HOUGH_GRADIENT, dp=1, minDist=40,
    param1=40, param2=45, minRadius=55, maxRadius=145
)
```

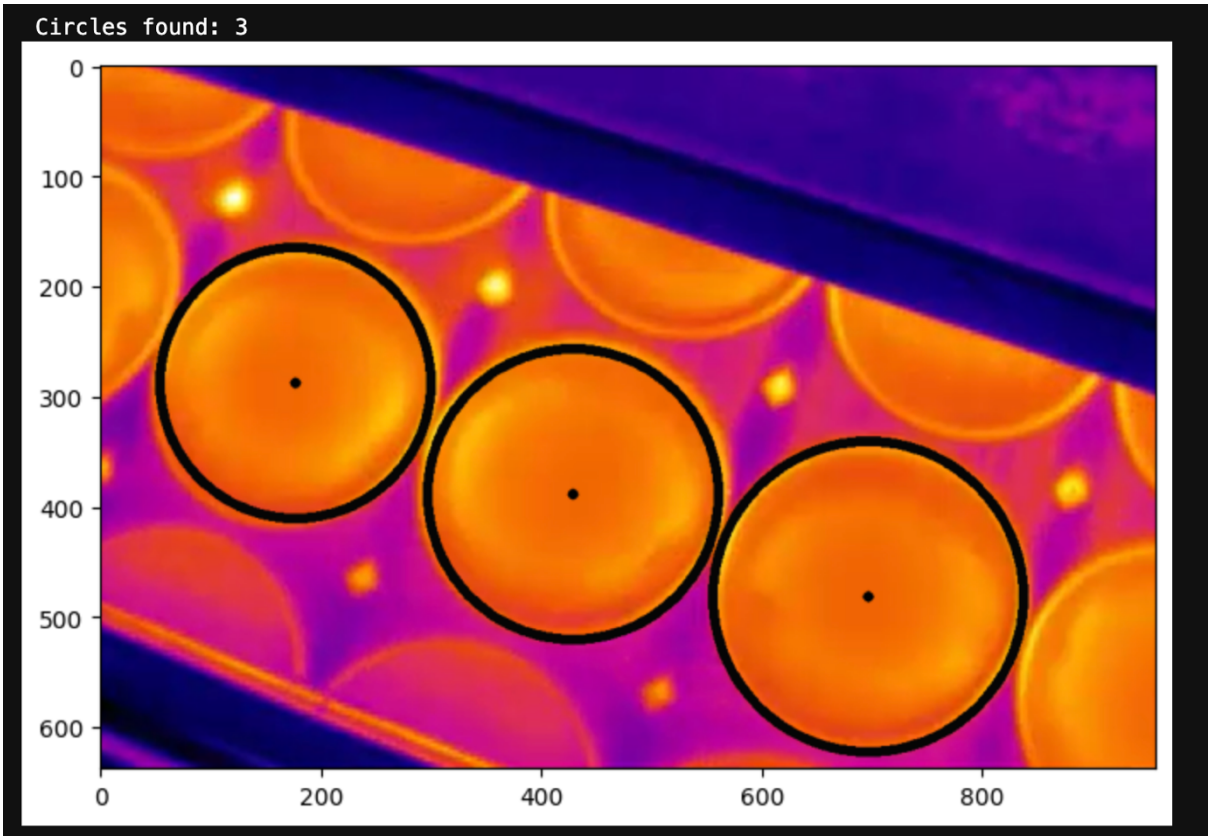


Image 2

```python
# Detect circles using the Hough Circle Transform
# dp=1 → same resolution as original image
# minDist=50 → minimum distance allowed between detected circles
# param1=100 → upper threshold for the Canny edge detector
# param2=80 → threshold for circle detection (higher → stricter detection)
# minRadius and maxRadius → expected size range of circles
circles = cv2.HoughCircles(
    blurred, cv2.HOUGH_GRADIENT, dp=1, minDist=50,
    param1=100, param2=80, minRadius=74, maxRadius=100
)
```

```
Circles found: 2
```

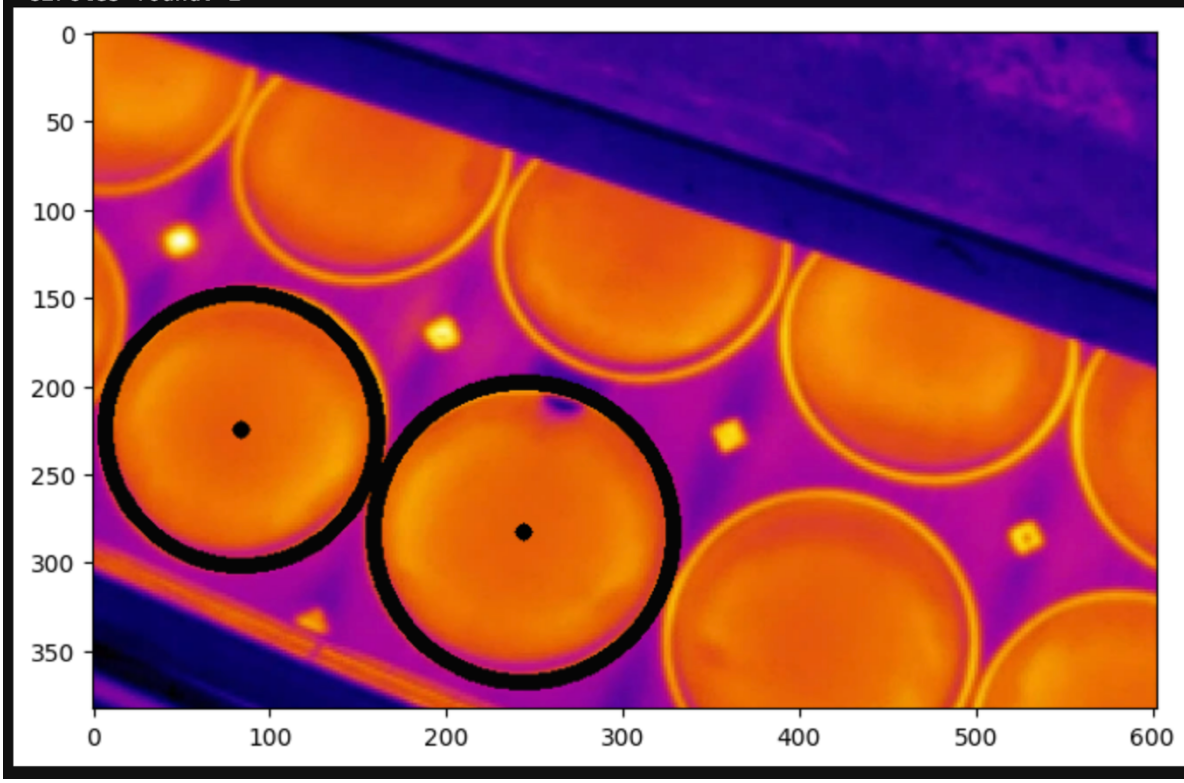


Image 3

```python
# Detect circles using the Hough Circle Transform
# dp=1          -> accumulator resolution equals image resolution
# minDist=50    -> minimum distance between detected circle centers
# param1=55     -> upper threshold for Canny edge detection
# param2=90     -> threshold for circle detection (higher means stricter)
# minRadius     -> smallest allowed radius
# maxRadius     -> largest allowed radius
circles = cv2.HoughCircles(
    blurred, cv2.HOUGH_GRADIENT, dp=1, minDist=50,
    param1=55, param2=90, minRadius=100, maxRadius=250
)
```

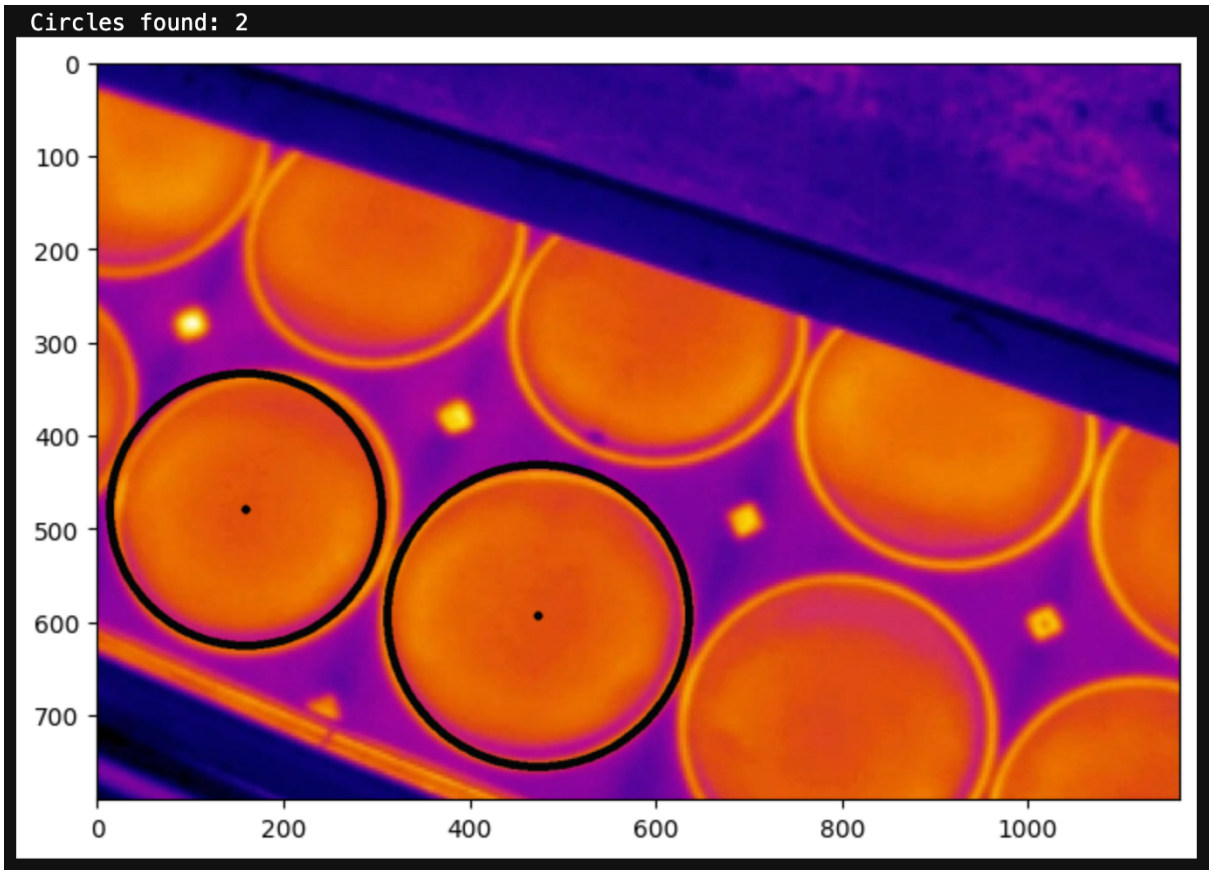


Image 4

```
# Detect circles using the Hough Circle Transform
# dp = 1        -> accumulator resolution equal to the image resolution
# minDist = 50  -> minimum distance between detected circle centers
# param1 = 55   -> upper threshold for Canny edge detection
# param2 = 90   -> threshold for circle detection (higher = stricter)
# minRadius, maxRadius -> expected range of circle radii
circles = cv2.HoughCircles(
    blurred, cv2.HOUGH_GRADIENT, dp=1, minDist=50,
    param1=55, param2=90, minRadius=100, maxRadius=250
)
```

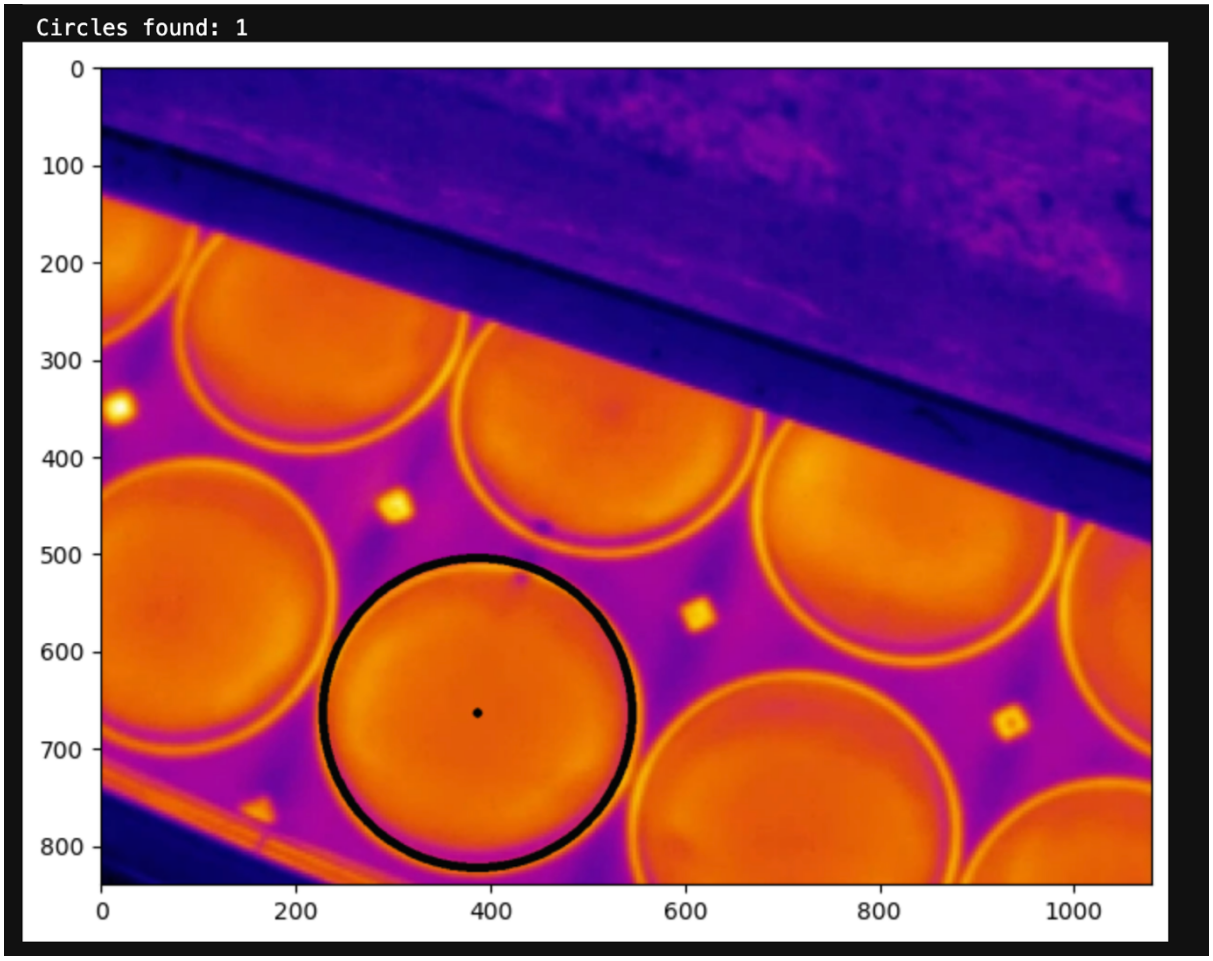


Image 5

```
# Detect circles using the Hough Circle Transform
# dp=1        -> accumulator resolution equals image resolution
# minDist=50  -> minimum distance between detected circle centers
# param1=50   -> upper threshold for Canny edge detection
# param2=70   -> threshold for deciding if a detection is a valid circle
# minRadius   -> smallest circle radius to look for
# maxRadius   -> largest circle radius to look for
circles = cv2.HoughCircles(
    blurred, cv2.HOUGH_GRADIENT, dp=1, minDist=50,
    param1=50, param2=70, minRadius=60, maxRadius=250
)
```

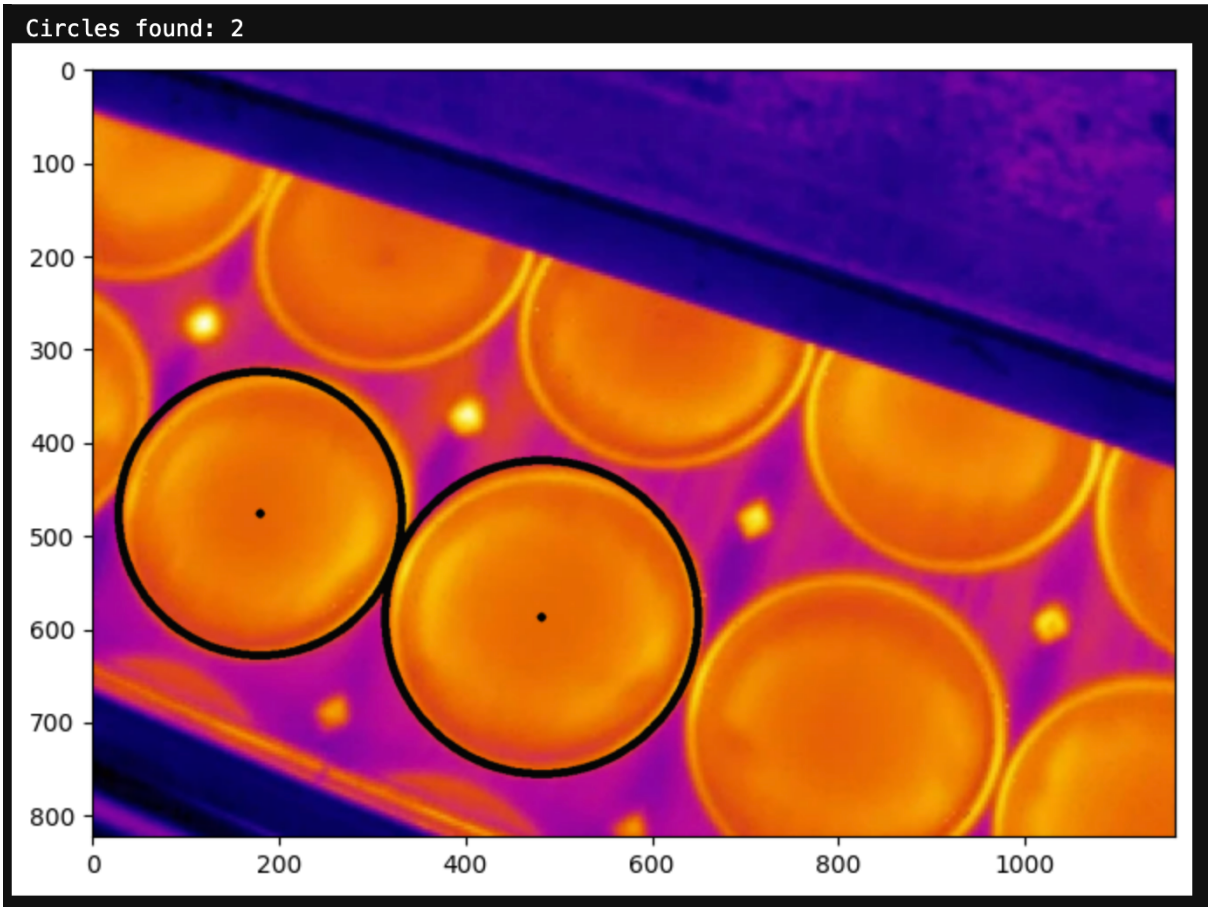


Image 6

```
# Detect circles using Hough Circle Transform
# dp=1        -> accumulator resolution same as input image
# minDist=50  -> minimum distance between detected circle centers
# param1=50   -> high threshold for Canny edge detection
# param2=90   -> detection threshold (higher = stricter)
# minRadius   -> smallest radius to detect
# maxRadius   -> largest radius to detect
circles = cv2.HoughCircles(
    blurred, cv2.HOUGH_GRADIENT, dp=1, minDist=50,
    param1=50, param2=90, minRadius=60, maxRadius=250
)
```

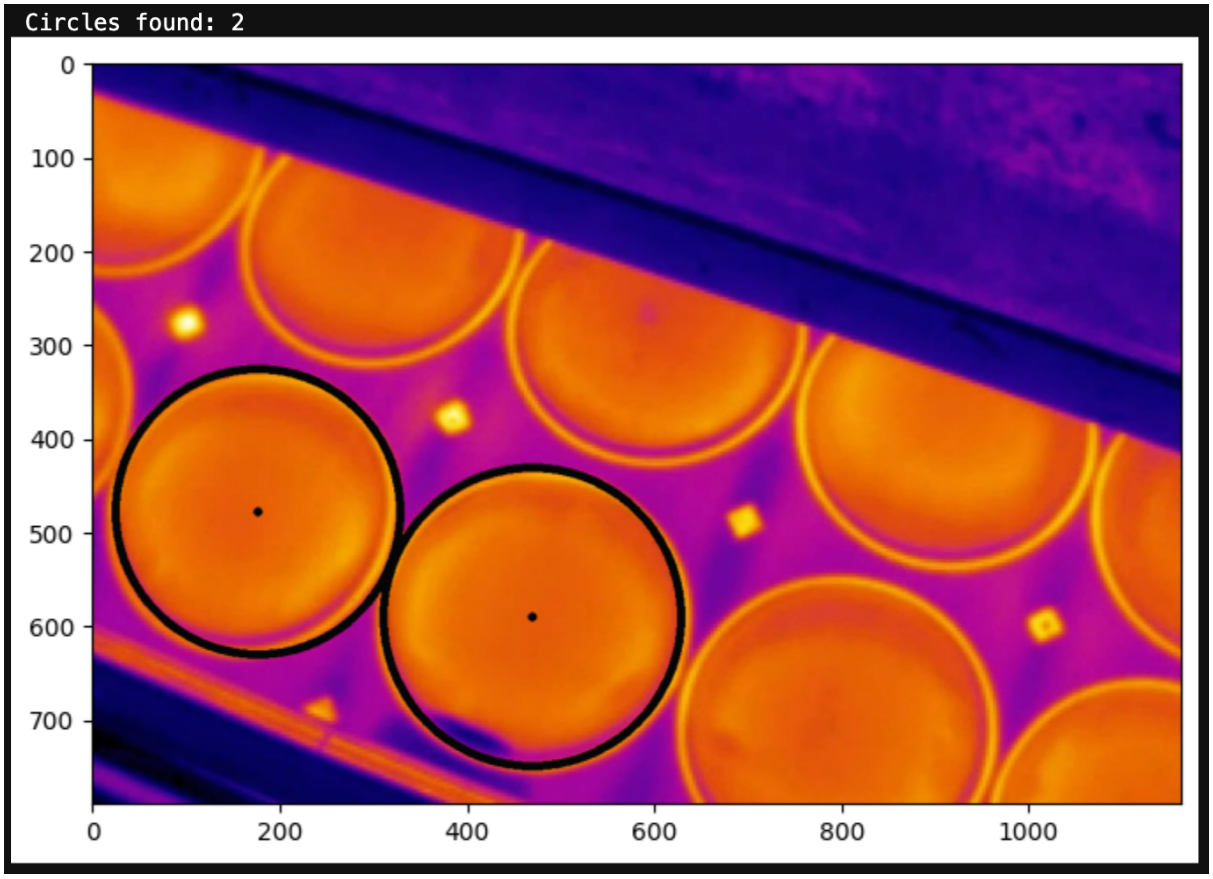
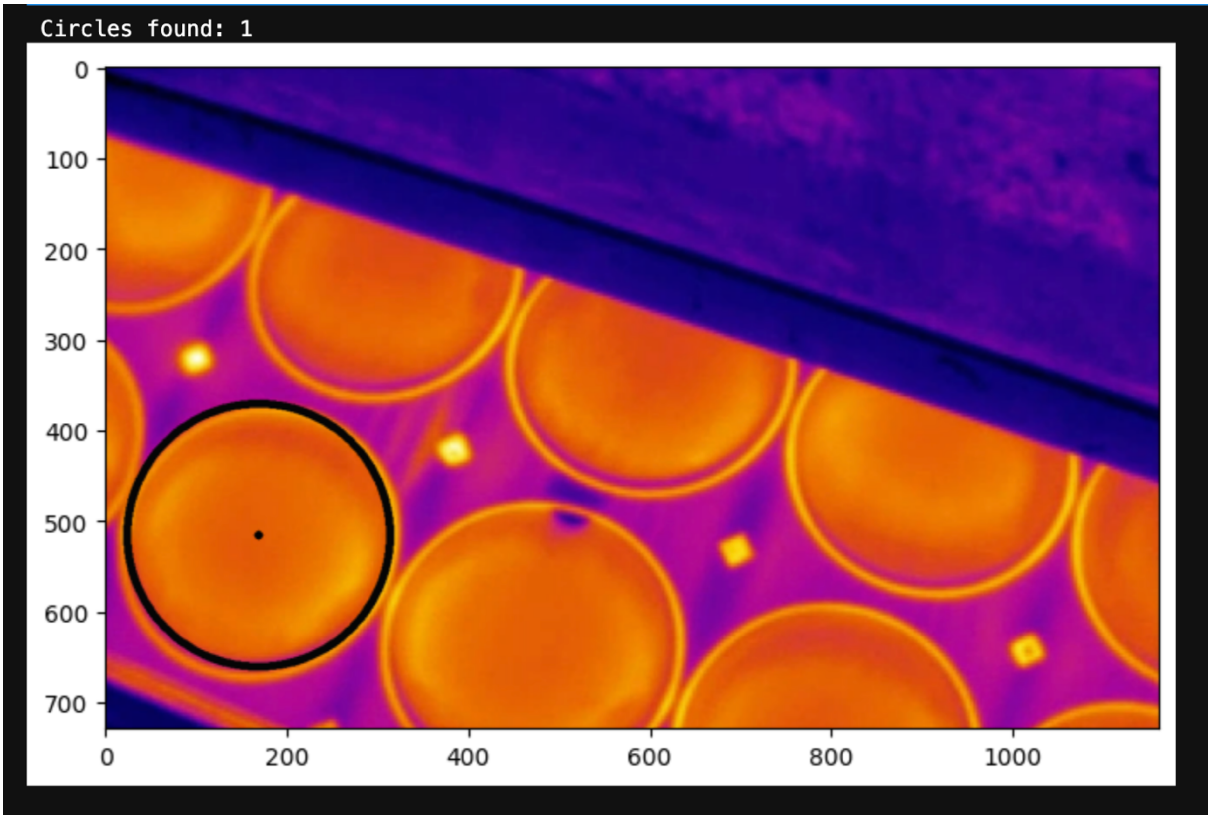


Image 7

```python
# Detect circles using Hough Circle Transform
# dp=1        -> accumulator resolution same as input image
# minDist=50  -> minimum distance between detected circle centers
# param1=50   -> high threshold for Canny edge detection
# param2=90   -> detection threshold (higher = stricter)
# minRadius   -> smallest radius to detect
# maxRadius   -> largest radius to detect
circles = cv2.HoughCircles(
    blurred, cv2.HOUGH_GRADIENT, dp=1, minDist=70, param1=50,
    param2=90, minRadius=80, maxRadius=250
)
```



Conclusions

- **Precise Piece Detection:** The developed system effectively utilized feature detection methods to separate and mark individual geometric pieces(circles) within the thermal imagery.
- **Count:** The system successfully determined the count of complete geometric pieces (circles) in the input images, completing the specific task requirement. This capability is critical for quality control and comprehensive analysis of the components.
- **Methodology Validation:** The effectiveness of the detection system was demonstrated through visual outputs on the provided sample thermographic images, confirming the robustness and accuracy of the employed algorithms and methodology but at the same time it is demonstrated how one individual or general solution does not exist. It is necessary always a different configuration for each image.