Machine Vision

Measurement and Feature Extraction in Machine Vision Using MATLAB

Lab Activity Sheet 8

Author: Dr. Mostapha Kalami Heris

Introduction

Measurement and feature extraction are essential steps in any machine vision workflow. After segmentation, an image still contains groups of pixels that do not yet carry meaningful information. A system must convert these segmented regions into numerical descriptions so that it can analyze shapes, compare objects, detect defects, support automation, and make reliable decisions.
Accurate measurements play an important role in many applications, including industrial inspection, robotic manipulation, medical imaging, and scientific analysis. By transforming segmented regions into quantifiable features, a machine vision system can evaluate object size, shape, intensity patterns, and structural characteristics, and use this information to guide further processing or decision making.

What You Will Learn Today

By completing this lab, you will be able to:

1. label connected components in segmented images,
2. extract geometric properties such as area, centroid, bounding box, and axis lengths,
3. compute intensity-based and structural descriptors,
4. measure shape features such as perimeter, roundness, eccentricity, and Feret diameters,
5. form and compare feature vectors for different objects,
6. overlay measurements on images for visual interpretation,

7. assess the reliability and sensitivity of measurements,
8. design a simple workflow for feature-based analysis or grouping.

Quick Refresher

Before starting, remember the following points from earlier labs:

- segmentation produces a binary mask that separates foreground and background,
- morphological operations help remove noise and refine shapes,
- connected-component labeling identifies individual objects for measurement,
- image type and intensity range should be verified before processing,
- reliable measurement depends on clean masks and consistent segmentation quality.

Concepts and MATLAB Functions

Key Concepts

This lab builds on several ideas that support measurement and feature extraction in machine vision. These concepts explain how MATLAB computes region statistics and how these measurements help with tasks such as inspection, classification, and decision making.

- **Connected components:** groups of foreground pixels that form individual objects.
- **Region labeling:** assigning a unique label to each connected component.
- **Region measurements:** numerical descriptions of object size, shape, intensity, and structure.
- **Geometric descriptors:** area, perimeter, centroid, bounding box, axis lengths, and orientation.
- **Positional descriptors:** centroid coordinates, extrema points, and bounding box values.
- **Intensity descriptors:** mean, maximum, and minimum intensity values within each object.

- **Shape descriptors:** circularity, eccentricity, convexity, equivalent diameter, and Feret diameters.
- **Topological descriptors:** Euler number and number of holes.
- **Feature vectors:** groups of measurements used for comparison, grouping, or classification.
- **Measurement reliability:** sensitivity to noise, illumination, segmentation quality, and preprocessing choices.

Summary of MATLAB Documentation

MATLAB provides extensive documentation on region, pixel, and image-level measurements in the *Region and Image Properties* section. Students can explore:

```
mathworks.com/help/images/pixel-values-and-image-statistics.html
```

This resource explains common measurement tools and gives examples for binary and grayscale images. Key topics include:

- **Region properties:** area, centroid, orientation, axis lengths, convex hull, Euler number, and Feret diameters.
- **Pixel and path tools:** `impixel`, `improfile`, and contour-based functions for inspecting intensity patterns.
- **Image-level statistics:** histograms, global statistics such as `mean2` and `std2`, and distance transforms such as `bwdist` and `graydist`.

Students are encouraged to use these documentation pages during the lab to review examples and understand each measurement more deeply.

MATLAB Function Reference Table

Table 1 summarizes MATLAB functions used in this lab and their typical applications.

Dataset, Setup, and Safety Checklist

Suggested Images

You may use any suitable image for the tasks in this lab. MATLAB also provides several helpful example images, such as:

Table 1: Common MATLAB functions for region measurement and feature extraction.

Function	**Functionality**	**Application Use Case**
`regionprops`	Computes geometric, spatial, and intensity properties of labeled regions	Measuring object size, shape, orientation, and grayscale statistics.
`bwlabel`	Labels connected components in a binary image	Separating objects for further processing.
`bwconncomp`	Efficient connected-component detection	High-performance processing for images with many objects.
`bwarea`	Estimates the area of binary objects	Filtering or comparing objects based on size.
`bwperim`	Extracts object perimeters	Boundary-based analysis and circularity measurement.
`bwferet`	Computes Feret diameter measurements	Dimensional inspection and shape comparison.
`bweuler`	Computes Euler number (components minus holes)	Analyzing materials, biological samples, and porous structures.
`bwboundaries`	Extracts region boundaries	Contour visualization and outline-based analysis.
`insertShape`	Draws rectangles, circles, or lines on images	Displaying bounding boxes, axes, or measurement results.
`insertText`	Writes text labels on images	Annotating object indices or measurement values.
`imhist`	Computes histograms	Examining intensity distributions and selecting thresholds.
`improfile`	Extracts intensity values along a line	Analyzing intensity transitions or gradients.
`pdist2`	Computes pairwise distances between feature vectors	Clustering, similarity measurement, and simple classification.

- `coins.png`
- `rice.png`
- `shapes.png`
- `text.png`
- `pout.tif`
- `cameraman.tif`

You are encouraged to use your own images if they are appropriate for segmentation and measurement.

Setup

Before you begin, complete the steps below:

- open MATLAB and create a working folder for this lab,
- place any external images you plan to use into the working folder,
- confirm that the Image Processing Toolbox is available.

Safety and Reliability Checklist

To ensure accurate and consistent measurements:

- check the image type using `class` and convert to grayscale if needed,
- verify the dynamic range to avoid clipping or incorrect scaling,
- clean segmentation masks using methods such as `bwareaopen` or `imopen`,
- make sure objects are well separated before taking measurements,
- remove noise or small fragments that may affect results,
- avoid measuring touching objects unless the activity requires it,
- be aware that illumination changes can influence intensity-based features,
- remember that all measurements are in pixels unless calibration is provided.

Activity 1: Measuring Basic Geometric Properties

Objective. Extract and interpret fundamental geometric properties of segmented circular objects. You will measure area, centroid, equivalent diameter, perimeter, and circularity, and display or print these values.

Description. In this activity, you will start with a segmented image such as `coins.png`. After labeling the connected components, you will measure several geometric descriptors. Because coins are approximately circular and have no meaningful orientation, the focus is on area, centroid, perimeter, and equivalent diameter. You will also compute *circularity*, which describes how closely an object resembles a perfect circle.

A common circularity metric is:

$$\text{Circularity}_{\text{simple}} = \frac{4\pi \cdot \text{Area}}{\text{Perimeter}^2}.$$

This value is close to 1 for a perfect circle and decreases as the shape becomes less round.
MATLAB refines this formula to reduce digital boundary bias:

$$\text{Circularity} = \left(\frac{4\pi \cdot \text{Area}}{\text{Perimeter}^2}\right)\left(1 - \frac{0.5}{r}\right)^2, \quad r = \frac{\text{Perimeter}}{2\pi} + 0.5.$$

Circular objects have values close to 1, while irregular or elongated objects produce lower values.
If possible, overlay the measurements directly on the image. **If annotation is difficult, print the results in the MATLAB command window.**
Tasks.

1. Load a sample image such as `coins.png`.
2. Convert the image to grayscale if needed.
3. Segment the image using `imbinarize` or `adaptthresh`.
4. Clean the mask using `bwareaopen` or morphological operations.
5. Label connected components using `bwlabel` or `bwconncomp`.
6. Use `regionprops` to extract:
   - `Area`
   - `Centroid`
   - `BoundingBox`
   - `Perimeter`
   - `EquivDiameter`
   - `Circularity`
7. Convert the label matrix to a color image using `label2rgb`.
8. Add bounding boxes and labels using `insertShape` and `insertText`.
9. Annotate measurements on the image *or* print values to the console.
10. Interpret the results by identifying the most circular objects and noting how segmentation influences measurements.

Starter Code.

```
I = imread('coins.png');
G = im2gray(I);

BW = imbinarize(G);
BW = bwareaopen(BW, 30);

[L, num] = bwlabel(BW);

props = regionprops(L, G, ...
   'Area','Centroid','BoundingBox', ...
   'Perimeter','EquivDiameter','Circularity');

RGB = label2rgb(L, 'jet', 'k');

figure; imshow(RGB); title('Labeled Objects');

out = I;
for k = 1:num
    bb = props(k).BoundingBox;
    c = props(k).Centroid;
    out = insertShape(out,'Rectangle',bb, ...
        'Color','yellow','LineWidth',2);
    out = insertText(out,c,sprintf('%d',k), ...
        'BoxOpacity',0.6,'FontSize',12);

    % Add annotations or print results:
    % txt = sprintf('D=%.1f  C=%.2f', ...
    %     props(k).EquivDiameter, props(k).Circularity);
    % out = insertText(out, c + [0 20], txt, ...
    %     'BoxOpacity',0.6,'FontSize',11);
end

figure; imshow(out); title('Objects with Measurements');

% Option B: print results if annotation is difficult
% T = struct2table(props);
% disp(T)
```

Additional Tasks.

- Print a summary table showing area, perimeter, equivalent diameter, and circularity.
- Identify the objects with the highest and lowest circularity.
- Examine how different thresholding methods affect the measurements.

Further Exploration.

- Use an image with noncircular objects and measure orientation or axis lengths.
- Compare diameter and area measurements for different objects.
- Apply different segmentation methods and observe the impact on measurements.
- Summarize all measurements using `struct2table`.

Reflection Questions.

- Which geometric properties are most useful for circular objects?
- How close are the objects to perfect circles based on circularity?
- Which measurements change the most when segmentation varies?
- Why is equivalent diameter helpful for round objects?

Activity 2: Exploring Shape and Intensity Descriptors

Objective. Extract and interpret additional shape descriptors and intensity-based features. Compare objects using several measurements and assess how preprocessing and segmentation changes affect measurement stability.
Description. In this activity, you will extend the measurements from Activity 1 by examining more advanced descriptors. These include eccentricity, equivalent diameter, perimeter, circularity, Feret diameters, Euler number, and intensity statistics. You will observe how each feature describes different aspects of object geometry and internal structure. You will also evaluate how preprocessing choices influence the consistency of these measurements.
Tasks.

- Start with the labeled mask from Activity 1 or choose a new image.

- Use `regionprops` to measure key shape properties:
  - `Perimeter` (boundary length),
  - `Eccentricity` (elongation measure),
  - `Circularity` (roundness),
  - `EquivDiameter` (diameter of an equal-area circle).
- Compute Feret diameters using `bwferet`. Record:
  - minimum caliper diameter,
  - maximum caliper diameter,
  - mean Feret diameter.
- Compute topological properties using `bweuler`. Interpret the Euler number in terms of connectivity and holes.
- Extract intensity statistics using `regionprops` with the grayscale image:
  - mean intensity,
  - maximum intensity,
  - minimum intensity,
  - standard deviation (if appropriate).
- Visualize boundaries using `bwboundaries` and overlay them on the original image with `plot`. Check whether the boundary shape matches the measurements.
- Create a small comparison table that includes at least: area, perimeter, circularity, eccentricity, and one Feret measurement.
- Repeat measurements after changing the segmentation process. For example:
  - apply smoothing before thresholding,
  - adjust the thresholding method,
  - change the minimum size in `bwareaopen`,
  - apply morphological opening or closing.

  Compare the new results with your earlier measurements.

Reflection Questions.

- Which features change the most when segmentation is modified?

- Which measurements remain stable across different preprocessing steps?
- Do Feret diameters reveal shape information not apparent from area or perimeter alone?
- How do smoothing or morphological operations influence perimeter and circularity?
- Which intensity-based features help distinguish objects with similar shapes?

Activity 3: Feature-Based Analysis and Grouping (Optional)

Objective. Design a small feature extraction and grouping workflow. You will select features, form feature vectors, and explore similarities or differences among objects.
Description. In this activity, you will create a measurement pipeline using an image of your choice. You will extract multiple features from each object, combine them into feature vectors, and explore how different measurements help distinguish objects. This task requires you to make your own design decisions, justify your feature choices, and interpret the patterns you observe.
What You Will Do.

- **Choose an image** with several objects that differ in size, shape, or appearance. Examples include coins, geometric shapes, grains, symbols, or items in a tray.
- **Perform segmentation and cleaning** using tools such as `imbinarize`, `bwareaopen`, `imopen`, or threshold adjustments. Ensure that:
  - objects are well separated,
  - boundaries are clean enough for stable measurements,
  - noise and small artifacts are removed.
- **Extract meaningful features** using `regionprops` and other functions. Measure at least five of the following:
  - area,
  - eccentricity,
  - equivalent diameter,
  - Feret diameters using `bwferet`,
  - major and minor axis lengths,
  - mean or maximum intensity,

  - bounding box height-to-width ratio,
  - perimeter and circularity.

  Add additional features if they support your analysis.

- **Form feature vectors** for each object. A sample feature vector may look like:

$$f = [\text{Area}, \text{Eccentricity}, \text{MeanIntensity}, \text{FeretMax}, \text{BB\_Ratio}]$$

  Select the elements based on the needs of your application.

- **Create visualizations** to explore relationships between features. Possible options include:
  - 2-D scatter plots such as area versus eccentricity,
  - 3-D scatter plots such as area versus eccentricity versus FeretMax,
  - color-coded scatter plots where color represents intensity,
  - pairwise comparison plots using `plotmatrix`.
- **Identify patterns or groups**. Look for:
  - objects with similar feature values,
  - unusual or defective objects,
  - natural separations based on shape or intensity.
- **Discuss your findings**. Reflect on which features were most helpful, which were less useful, and how feature choices affected grouping or clustering.

Open-Ended Challenges.

- Detect objects that differ from expected sizes, shapes, or intensities.
- Identify defective or irregular objects using distance measures such as `pdist2`.
- Produce 2-D or 3-D feature plots to visualize clusters or outliers.
- Investigate how segmentation errors influence feature variability.
- Try reducing the number of features and observe how grouping performance changes.

Summary and Applications

In this lab, you explored how segmented regions can be transformed into meaningful numerical information. You measured geometric properties such as area, centroid, axis lengths, perimeter, and bounding boxes. You also worked with more advanced descriptors, including eccentricity, equivalent diameter, Feret diameters, Euler number, and intensity-based features. Together, these measurements provide a detailed description of each object and support higher-level analysis.
Feature extraction is a core component of modern machine vision. It enables systems to examine object size, shape, structure, and internal intensity patterns, and it forms the basis for many practical applications, including:

- industrial inspection and defect detection,
- robotic manipulation and alignment,
- automated sorting and classification,
- medical image measurement and analysis,
- scientific imaging and quantitative evaluation.

By completing this lab, you developed skills in extracting, interpreting, and visualizing a wide range of object features. These skills form a strong foundation for more advanced tasks such as classification, object recognition, and quality assessment in machine vision systems.