Image Segmentation and Thresholding

Machine Vision

Dr. Mostapha Kalami Heris

m.k.heris@shu.ac.uk

School of Engineering and Built Environment

Sheffield Hallam University

Learning Outcomes

- **Define** the purpose and principles of image segmentation.
- **Differentiate** global, adaptive, and multilevel thresholding.
- **Apply** MATLAB functions (`graythresh` , `adaptthresh` , `imbinarize` , `multithresh`).
- **Analyze** limitations under uneven illumination and noise; propose remedies.
- **Evaluate** which method suits a given application scenario.
- **Explain** a chosen concept clearly to peers with an example.

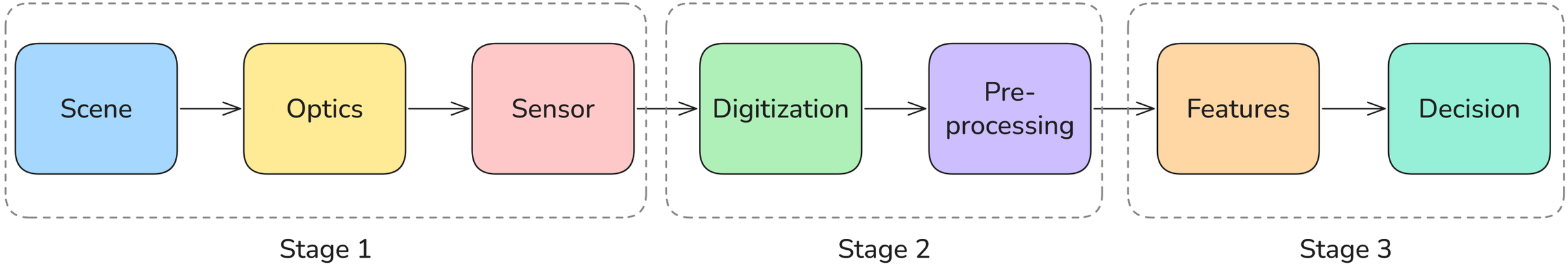
Road Map for Today's Lecture

- Introduction & Motivation
- Conceptual foundations
- Thresholding basics
- Global vs. Adaptive
- Otsu & histograms
- Multilevel thresholding (RGB options)
- Peer-teaching activity
- Summary & Wrap-up

Image Segmentation

Recall the Machine Vision Pipeline

We discussed the main stages of a typical machine vision system before.



Let's see where segmentation fits in this pipeline.

Where Segmentation Fits in the Pipeline

Typical flow

1. Image acquisition
2. Preprocessing (denoise, contrast)
3. **Segmentation** (pixel → region)
4. Features & measurements (`regionprops`)
5. Detection / decision / tracking

> Segmentation reduces millions of pixels to a handful of meaningful **regions**, enabling downstream reasoning.

Motivation: Why Segment Images?

If you had to teach a computer to identify the key objects here, what must it do **first**?

Sheffield Hallam University

What Is Image Segmentation?

Definition

Segmentation **partitions** an image into regions with similar properties (e.g., intensity or color).

The goal is to convert pixels into **meaningful regions** for analysis and decision making.

> Why **meaningful**? What do you think?

Why it matters

- Enables object **counting** and **measurement**
- Simplifies **detection** and **tracking**
- Critical for inspection, documents, and microscopy

Segmentation Methods

Segmentation Methods at a Glance

Families

- **Threshold-based:** global, adaptive, multilevel
- **Edge-based:** gradient, Canny, Laplacian
- **Region-based:** region growing, active contours
- **Clustering-based:** k-means, ISODATA
- **Learning-based:** U-Net, Mask R-CNN, SAM

Typical Cues

- Strong **intensity contrast** → thresholding
- Clear **boundaries** → edge-based
- Homogeneous **regions** → region-based
- Multi-feature **separation** → clustering
- Complex **scenes / semantics** → deep models

Applications Across Domains

Industrial Inspection

- Defect highlighting
- Part presence/absence
- Size measurement
- Surface quality
- Assembly verification

Document Analysis

- Text binarization
- Background removal
- OCR pre-processing
- Table detection
- Signature extraction

Microscopy / Medical

- Cell counting
- Tissue delineation
- Lesion detection
- Organ segmentation
- Tumor identification

Traffic Systems

- Lane markings
- Sign detection
- Road surface analysis
- Vehicle counting
- Pedestrian detection

Image Thresholding

What Is Image Thresholding?

Definition

Thresholding classifies pixels based on intensity: pixels above a certain value are *foreground*; below are *background*.

It is the simplest and fastest segmentation method.

Why it matters

- Computationally efficient for real-time systems
- Easy to implement and interpret
- Effective when foreground and background differ significantly in intensity
- Widely used in industrial inspection, document analysis, and medical imaging

When Does Thresholding Work?

Good Fit

- Foreground vs. background have **distinct intensities**
- **Uniform illumination** (or locally uniform)
- **Low noise** after light denoising
- You need **speed** and **interpretability**

Pitfalls to Watch

- **Illumination gradients / shadows**
- **Low contrast** or overlapping histograms
- **Specular highlights** and sensor noise
- **Touching objects** (need post-processing)

Quick Poll — What Hurts Thresholding Most?

Which factor most affects thresholding performance?

1. Illumination variation
2. Sensor noise
3. Object shape
4. Colorfulness

menti.com/aluop3n6fviy

Sheffield Hallam University

How Does Thresholding Work?

Thresholding: The Decision Rule

Binary decision

$$g(x,y) = \begin{cases} 1 & \text{if } I(x,y) \geq T \\ 0 & \text{if } I(x,y) < T \end{cases}$$

- Output: **mask**

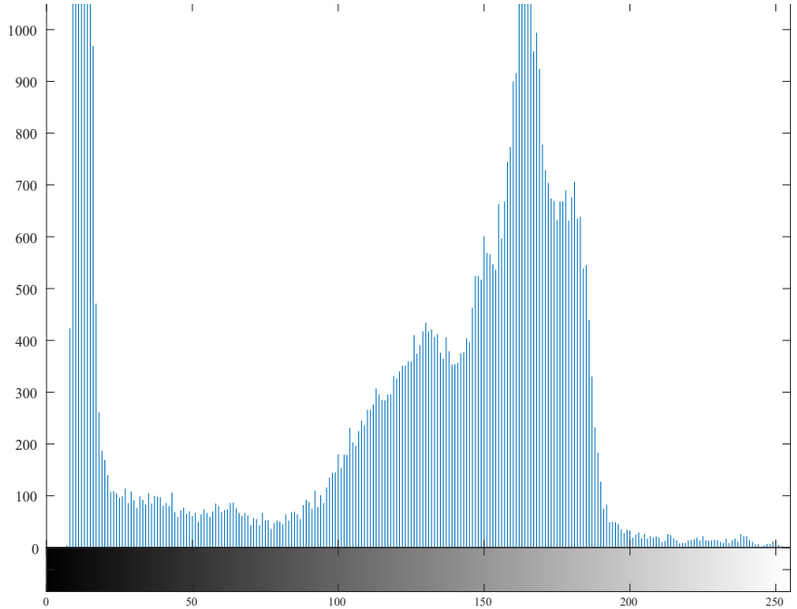
  (foreground=1, background=0)

- Choose T manually or automatically
- Works best when classes are **separable**

Quick intuition

- Pixels **well above** T → foreground
- Pixels **well below** T → background
- Pixels **near** T are ambiguous → sensitive to *noise & illumination*

Histogram Intuition



A bimodal histogram (from `cameraman.tif`).

Why histograms help

- Show **how many** pixels occupy each intensity
- **Bimodal** case → a natural split (valley)
- **Overlapping** peaks → harder decision; expect errors
- **Shifted** histogram (lighting changes) → global T may fail

MATLAB Demo: Manual Global Threshold

This example uses a fixed threshold $T = 0.5$ on a grayscale image of coins.

```MATLAB
I  = imread('coins.png');     % grayscale
T  = 0.5;                     % manual threshold in [0,1]
BW = imbinarize(I, T);
imshowpair(I, BW, 'montage')
title('Original | Global Threshold (T=0.5)')
```

What to look for

- Clean separation of coins vs. background
- Missed edges if contrast is low
- Holes inside objects → fix later via morphology

MATLAB Demo: Adaptive Thresholding

We use adaptive thresholding to handle uneven illumination.

```MATLAB
I   = imread('printedtext.png');
T   = adaptthresh(I, 0.4, 'ForegroundPolarity','dark');
BW  = imbinarize(I, T);
imshowpair(I, BW, 'montage')
title('Original | Adaptive Threshold (sensitivity=0.4)')
```

Key parameter: `sensitivity`, which is between 0 and 1.

Higher → more pixels labeled as foreground.

Why adaptive?

- Computes $T(x, y)$ from local statistics
- Robust to **shadows** and **gradients**
- Slightly higher compute cost, but often worth it

Typical use

- Document binarization
- Text or symbols under uneven lighting

Global vs. Adaptive: Quick Comparison

Global

- ✅ Fast and simple
- ✅ Interpretable, single T
- ⚠️ Fails under illumination changes
- ⚠️ Sensitive to low contrast & noise

Adaptive

- ✅ Handles local lighting variation
- ✅ Better on textured backgrounds
- ⚠️ More parameters (window, sensitivity)
- ⚠️ Slightly higher computation

Rule of thumb:

Try **global** first; if histogram/illumination look problematic, switch to **adaptive** or pre-process using *CLAHE*.

Otsu’s Method & Histogram-Based Segmentation

Why Otsu?

The challenge

Choosing a good global threshold T **automatically**.

Otsu's idea

Pick T^* that maximizes **between-class variance** of the two groups (background vs. foreground).

Key formula

Let a candidate threshold t split pixels into

$C_0 : I \leq t$ and $C_1 : I > t$.

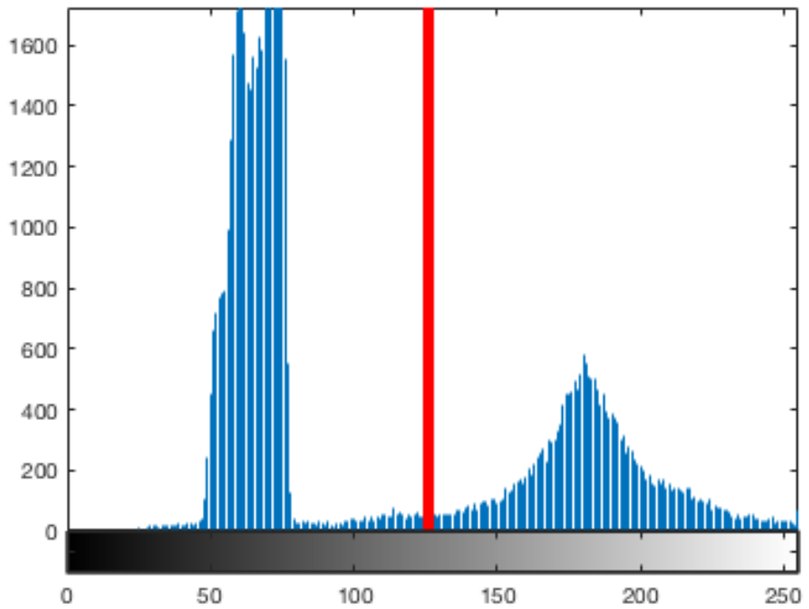
- Class probabilities: $\omega_0(t)$ and $\omega_1(t)$
- Class means: $\mu_0(t)$ and $\mu_1(t)$

$$\sigma_B^2(t) = \omega_0(t)\,\omega_1(t)\,\big(\mu_1(t) - \mu_0(t)\big)^2$$

Choose $T^* = \arg\max_t \sigma_B^2(t)$.

Sheffield Hallam University

Otsu: Histogram Intuition



Two peaks with a valley. Otsu picks the threshold that best separates the classes.

Interpretation

- If peaks are **distinct**, σ_B^2 is large → confident split
- If peaks **overlap**, σ_B^2 is smaller → less reliable split
- If the histogram is **unimodal** (e.g., strong shadow), global Otsu will struggle

MATLAB Demo: `graythresh` (Automatic Global T)

```MATLAB
I = imread('coins.png');      % grayscale
[level, EM] = graythresh(I); % Otsu threshold & effectiveness
BW = imbinarize(I, level);

imshowpair(I, BW, 'montage')
title(sprintf('Otsu: level=%.4f, EM=%.3f', level, EM))
```

EM (effectiveness metric) is a measure of separability of the two classes.

The value is between 0 and 1; and higher values indicate better separation.

Notes

- `graythresh` returns **normalized** level in range $[0, 1]$
- `imbinarize` with no second arg also uses Otsu internally

MATLAB Demo: `otsuthresh` from Histogram

```MATLAB
I = imread('coins.png');
[counts, x] = imhist(I, 16);     % coarse histogram
T = otsuthresh(counts);          % Otsu from counts [0,1]
BW = imbinarize(I, T);

subplot(1,2,1), stem(x, counts), title('Histogram')
subplot(1,2,2), imshow(BW), title('Binarized Image')
```

> Coarser bins can smooth noise in counts; try 16, 32, 64 to compare stability.

When to use this path

- You want explicit control over **binning**
- You already have histogram counts (e.g., streaming pipeline)
- Teaching the **mechanics** of Otsu from histogram statistics

Otsu's Effectiveness Metric (EM)

EM Range	Typical Reading	Likely Action
≥ 0.75	Strong bimodality	Global Otsu usually OK
0.5–0.75	Moderate separation	Check results; minor cleanup
0.3–0.5	Weak separation	Consider **adaptive** or pre-processing
< 0.3	Poor separation	Use adaptive, contrast enhancement, or different method

Rule of thumb: Always confirm visually and with task metrics (e.g., count accuracy).

Limits & Mitigations

Where Otsu struggles

- **Unimodal** or heavily **overlapped** histograms
- **Shadows / illumination gradients**
- **Specular highlights** / sensor noise
- **Touching objects** (need separation)

Practical fixes

- **Adaptive thresholding** for uneven lighting
- **Contrast enhancement** before Otsu
- **Morphology** for cleanup
- **Multilevel** thresholds when more than two intensity classes exist

Multilevel Thresholding & Hybrid Segmentation

Why Multilevel Thresholding?

Concept

Instead of a single threshold T, choose **multiple thresholds** $\{T_1, T_2, \ldots, T_K\}$ to split intensities into $K + 1$ bands.

When useful

- Images with **several** meaningful intensity modes
- Need to separate **background / midtones / highlights**
- Pre-step for label-based analysis or region refinement

Outcome

- Output is a **label map** (0,1,2,...), not just binary
- Can be visualized via `label2rgb`
- Often paired with simple **morphology** (optional)

Sheffield Hallam University

MATLAB Demo: `multithresh` + `imquantize`

```MATLAB
I = imread('foggysf2.jpg');
G = rgb2gray(I);                    % work in grayscale first

thresh  = multithresh(G, 2);        % 2 thresholds → 3 classes
labels  = imquantize(G, thresh); % label map
RGBseg  = label2rgb(labels);        % color each region

subplot(1,3,1); imshow(I);       title('Original');
subplot(1,3,2); imshow(G);       title('Grayscale');
subplot(1,3,3); imshow(RGBseg); title('Multilevel Segmentation');
```

Notes

- `multithresh(G,N)` finds (N) thresholds using an Otsu extension
- Use `imquantize` to convert thresholds into **region labels**
- `label2rgb` provides quick visualization for discussion

Sheffield Hallam University

RGB Images: Whole vs. Per-Plane Thresholding

Strategies

- **Whole-image** multilevel on grayscale

  → simple & fast

- **Per-plane** thresholds (R,G,B or HSV)

  → better for **color-coded** structure

- **Hue-only** banding for strong chroma

  signals (e.g., colored capsules)

MATLAB

```
I = imread('peppers.png');

% Whole image thresholds (grayscale)
G = rgb2gray(I);
t = multithresh(G, 2);
L1 = imquantize(G, t);
imshow(label2rgb(L1));
title('Labels from grayscale thresholds');

% Per-plane thresholds
tR = multithresh(I(:,:,1), 2);
tG = multithresh(I(:,:,2), 2);
tB = multithresh(I(:,:,3), 2);
```

Peer-Teaching Activity

Activity: Teach One Concept to Your Group

Explain **one** of today's segmentation concepts to others in your group.

Steps

1. Form groups of **3–5**.
2. Each person **chooses a different concept** from the list (next slide).
3. You have **2 minutes** to prepare.
4. Then you have **1–2 minutes** to explain it to group.

Why are we doing this?

- Explaining = **higher-order learning**
- You will see how concepts **connect**
- You will practice "choosing the right method"

Focus on: *What is it? When do we use it? Which MATLAB function(s) implement it?*

Choose One Concept to Explain

- Image Segmentation
- Basic Rule of Thresholding
- Global Thresholding
- Adaptive/Local Thresholding
- Otsu's Method
- Multilevel Thresholding
- Segmentation Applications

Your mini-teaching (1–2 minutes)

1. **Name it**
2. **Say when we use it** (conditions, data)
3. **Say why not another method**
4. **Mention MATLAB** function(s), if applicable

Summary & Wrap-Up

Today's Key Messages

1. **Segmentation = pixels → regions**
   - It is a **data reduction** step.
   - Everything after (detection, measurement) depends on it.
2. **Thresholding is the simplest segmentation**
   - One rule, very fast, very readable.
3. **Illumination decides the method**
   - Uniform → **global**
   - Non-uniform → **adaptive**
   - Bimodal histogram → **Otsu**
   - Several intensity classes → **multithresh**

Sheffield Hallam University

Check Against Learning Outcomes

- Can you **define** image segmentation and say *why* we use it?
- Can you **compare** global vs. adaptive thresholding?
- Can you **apply** `graythresh`, `adaptthresh`, `imbinarize`, `multithresh` in MATLAB?
- Can you **analyze** when Otsu fails (illumination, unimodal histogram)?
- Can you **explain** one of these to someone else?

If one of these is still unclear, write it down now. We will address it in the next practical / tutorial.

References

- Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing* (4th ed.). Pearson.
- Gonzalez, R. C., Woods, R. E., & Eddins, S. L. (2020). *Digital Image Processing Using MATLAB* (3rd ed.). Gatesmark Publishing.
- Forsyth, D. A., & Ponce, J. (2011). *Computer Vision: A Modern Approach* (2nd ed.). Pearson.
- Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.
- MathWorks. (n.d.). *Image Processing Toolbox – MATLAB*. Retrieved from https://mathworks.com/help/images/index.html

Exit Ticket

Before you leave, please complete the online form and provide your feedback.

1. **One clear concept** from today
2. **One confusing concept**
3. **One thing to try in MATLAB** this week

Scan the QR code or visit the link below.

🔗 forms.office.com/e/YvWnAr66yJ

Sheffield Hallam University

Questions & Support

- You are encouraged to **ask questions anytime** 💡
  - During the lecture
  - In lab sessions
  - By email → m.k.heris@shu.ac.uk
- **No question is too simple** — asking questions:
  - Helps you learn faster
  - Builds confidence
  - Improves understanding for the whole class

Sheffield Hallam University