Morphological Operations & Image Cleaning

Machine Vision

Dr. Mostapha Kalami Heris

m.k.heris@shu.ac.uk

School of Engineering and Built Environment

Sheffield Hallam University

Learning Outcomes

- **Apply** morphological operations (erosion, dilation, opening, closing) to improve images.
- **Analyze** how structuring element shape and size influence morphological outcomes.
- **Evaluate** appropriate operations for noise removal, object separation, etc.
- **Implement** main MATLAB tools (`strel` , `imopen` , `imclose` , `imtophat` , `imsubtract`).
- **Create** an image-cleaning workflow combining morphology and background correction.

Road Map for Today

1. Morphology: The Big Idea
2. Binary vs. Grayscale Morphology
3. Structuring Elements and Their Effects
4. MATLAB Primer for Structuring Elements
5. Core Operations
6. Applications and Background Correction

Motivation

Observing the Result of Segmentation

This is the segmented fingerprint produced by a simple thresholding method.

What We Aim to Obtain

A clean representation of the fingerprint ridges, free from artifacts.

Motivational Question

We want to clean the segmented fingerprint while preserving the ridge structure.

How can we remove the artifacts without damaging the important details?

Take a moment and think: What kind of processing could achieve this?

Foundations of Morphology

Morphology: The Big Idea

Morphology examines the **shape and structure** of objects in an image.

It uses a small shape called a **structuring element (SE)** to **probe** the image.

- Acts on geometry rather than intensity values alone.
- Nonlinear and set-based (unlike linear filters).
- Useful for cleaning, measuring, and understanding object form.

Why Morphology?

- Remove small specks or isolated pixels.
- Smooth or regularize object boundaries.
- Bridge small gaps or connect nearby parts.
- Separate touching objects.
- Correct uneven backgrounds.
- Prepare images for measurement and classification.

Binary vs. Grayscale Morphology

Binary morphology works on logical (0/1) images.

Operations depend on whether the SE fits entirely within a foreground region.

Grayscale morphology applies min/max operations within the SE neighborhood.

It can modify brightness and local contrast without thresholding.

Best for cleaning thresholded masks or binary segmentations.

Useful for background correction, enhancement, and illumination balancing.

Structuring Elements (SEs)

Structuring Element: The Design Knob

A **structuring element (SE)** defines how morphology interacts with image structures.

- **Shape:** square, disk, line, or custom pattern
- **Size:** controls the scale of influence
- **Origin:** determines alignment and directionality

Matching SE to the Task

- **Square / Disk:** general noise removal and smoothing
- **Line(θ):** remove or bridge oriented artifacts
- **Ball / Non-flat:** approximate background variations in grayscale images
- **Custom:** tailored to specific shapes or patterns

Choose SEs based on the geometry of the unwanted or desired features.

MATLAB Primer: Structuring Elements (1/2)

The `strel` function creates SEs for morphological operations.

```MATLAB
% Common flat SEs
se1 = strel('square', 3);
se2 = strel('disk', 5);          % radius in pixels
se3 = strel('line', 11, 90);     % length and angle (degrees)
```

> Start small and increase the size gradually. Large SEs can oversmooth or distort object boundaries.

MATLAB Primer: Structuring Elements (2/2)

There are also non-flat SEs for grayscale morphology.

```MATLAB
% Non-flat / offset SEs (grayscale)
se_bg = offsetstrel('ball', 15, 4); % radius, height

% Check the neighborhood pattern
nbhd = se1.Neighborhood;
```

> Non-flat SEs help model gradual intensity changes, useful for background or illumination correction.

Quick Check

Which SE would you choose to remove **vertical scratches**?

A. `square(3)`

B. `disk(5)`

C. `line(9, 90°)`

Quick Check

Which SE would you choose to remove **vertical scratches**?

A. `square(3)`

B. `disk(5)`

C. `line(9, 90°)` ✅

> **Answer: C** — Line SE aligned with the artifact orientation.

Core Morphological Operations

Core Morphological Operations

Erosion

Shrinks bright regions; removes small bright specks.

Dilation

Grows bright regions; fills small gaps and bridges narrow breaks.

Opening

Erosion followed by Dilation; removes small bright noise.

Closing

Dilation followed by Erosion; fills small dark holes.

Erosion: Concept

Erosion replaces each pixel with the minimum value under the SE.

Mathematical Definition: $(I \ominus S)(x, y) = \min_{(i,j) \in S} I(x + i, y + j)$

Effects of Erosion:

- Shrinks bright regions; removes small bright specks.
- Can break thin bridges and separate lightly touching parts.
- Sensitive to SE size and shape.

Erosion: MATLAB Code

This example uses a disk SE of radius 3 pixels.

```MATLAB
I = imread('coins.png');          % Load image
se = strel('disk', 3);            % Create SE
J  = imerode(I, se);              % Erode image
imshowpair(I, J, 'montage');      % Display side-by-side
```

💡 **Tip.** If objects fragment, reduce SE size or try a different shape.

Dilation: Concept

Dilation replaces each pixel with the maximum value under the SE.

Mathematical Definition: $(I \oplus S)(x, y) = \max_{(i,j) \in S} I(x - i,\ y - j)$

Effects of Dilation:

- Grows bright regions; fills small gaps and narrow breaks.
- Bridges close components; can close tiny pinholes.
- Strongly influenced by SE size, shape, and orientation.

Dilation: MATLAB Code

This example uses a disk SE of radius 3 pixels.

```MATLAB
I = imread('coins.png');          % Load image
se = strel('disk', 3);            % Create SE
J  = imdilate(I, se);             % Dilate image
imshowpair(I, J, 'montage');      % Display side-by-side
```

💡 **Tip.** If small gaps persist, increase SE slightly or try a line SE aligned with the gap direction.

Opening: Concept

Opening is erosion followed by dilation.

Mathematical Definition: $(I \circ S) = (I \ominus S) \oplus S$

Effects of Opening:

- Removes small bright noise while preserving larger shapes.
- Smooths outer boundaries without expanding regions.
- Useful after thresholding to clean salt noise.

Opening: MATLAB Code

Example on a binary mask `BW` using a disk SE of radius 3 pixels.

```MATLAB
% Assume BW is a logical mask from an earlier step
se  = strel('disk', 3);
BWc = imopen(BW, se);
imshowpair(BW, BWc, 'montage');
```

> 💡 **Tip.** If useful detail is removed, reduce SE size or switch to a better matching shape.

Closing: Concept

Closing is dilation followed by erosion.

Mathematical Definition: $(I \bullet S) = (I \oplus S) \ominus S$

Effects of Closing:

- Fills small dark holes and bridges narrow gaps.
- Smooths inner boundaries of objects.
- Complements opening for balanced cleanup.

Closing: MATLAB Code

Example on a binary mask `BW` using a disk SE of radius 3 pixels.

```MATLAB
% Assume BW is a logical mask from an earlier step
se  = strel('disk', 3);
BWf = imclose(BW, se);
imshowpair(BW, BWf, 'montage');
```

> 💡 **Tip.** If regions start merging unintentionally, decrease SE size or use a less aggressive shape.

Quick Comparison

Erosion: shrink, remove small bright specks, separate thin bridges.

Dilation: grow, fill tiny gaps, connect nearby parts.

Opening: remove small bright noise; smooth outer boundaries.

Closing: fill small dark holes; bridge narrow gaps; smooth inner boundaries.

Mini Example: Basic Cleaning Workflow

This sequence removes small bright noise and then fills small dark holes.

> 💡 **Tip.** Start with the smallest SE that works. Increase gradually to preserve shape and measurements.

```MATLAB
% 1. Threshold (example)
BW  = imbinarize(I);

% 2. Remove small bright noise
se  = strel('disk', 2);
BW1 = imopen(BW, se);

% 3. Fill small holes / bridge tiny gaps
BW2 = imclose(BW1, se);

imshowpair(BW, BW2, 'montage');
```

Quick Check

Which operation best addresses **small dark holes** and **narrow gaps** inside objects?

A. Opening

B. Closing

C. Erosion

Quick Check

Which operation best addresses **small dark holes** and **narrow gaps** inside objects?

A. Opening

B. Closing ✅

C. Erosion

> **Answer: B** — Closing (dilation → erosion) fills small dark holes and can bridge narrow gaps.

Image Cleaning & Background Correction

Noise Removal with Morphology

Common cases:

- **Salt noise** (isolated bright pixels) → use **opening**.
- **Pepper noise** (small dark holes) → use **closing**.
- **Mixed noise** → combine opening and closing (order depends on which is more severe).

Choose SE size just large enough to remove the artifacts without damaging object shape.

Object Separation: Concept

- Touching objects may be connected by thin "necks" or small bridges.
- **Erosion** can break these narrow connections.
- After erosion, **labeling** can count or analyze separate objects.
- Optional: distance-based methods (e.g., watershed) for more complex overlaps.

Goal: separate objects just enough for measurement, without losing important structure.

Object Separation: MATLAB Example

```MATLAB
I  = imread('coins.png');      % Example grayscale image
BW = imbinarize(I);            % Simple thresholding

se   = strel('disk', 2);       % Modest erosion
BW_e = imerode(BW, se);

[L, num] = bwlabel(BW_e);      % Label separated objects
stats    = regionprops(L, 'Area', 'Centroid');
```

💡 **Tip.** If objects disappear, reduce SE size.

If they remain connected, increase SE slightly or use a line SE aligned with the neck.

Uneven Illumination: Problem & Idea

- Global thresholding can fail under strong lighting gradients.
- Background may be brighter on one side, darker on the other.
- Idea: **estimate the background** with a large SE, then remove it.

Top-hat transform (bright objects):

$$I_{\text{tophat}} = I - (I \circ S)$$

Background model via opening:

$$I_{\text{bg}} = I \circ S, \quad I_{\text{corr}} = I - I_{\text{bg}}$$

Background Correction: MATLAB Code

```MATLAB
I = imread('rice.png');          % Example with uneven illumination

% 1. Choose a large SE relative to object size
se_bg = strel('disk', 15);

% 2a. Top-hat for bright objects
I_th = imtophat(I, se_bg);

% 2b. Explicit background model
I_bg = imopen(I, se_bg);
I_cs = imsubtract(I, I_bg);

% 3. Threshold corrected image
BW = imbinarize(I_cs);
```

💡 **Tip.** SE for background should be **larger than the objects** so it follows the illumination, not the objects.

Summary and Conclusion

Key Takeaways

- Morphology uses **structuring elements (SEs)** to analyze and modify image structure.
- **Erosion** and **dilation** form the basis of all morphological operations.
- **Opening** removes small bright noise; **closing** fills small dark holes.
- SE **shape and size** strongly influence the outcome of every operation.
- Morphology helps with **noise removal**, **object separation**, and **illumination correction**.

Choosing the Right Operation

- Use **opening** for small bright specks or isolated noise.
- Use **closing** for dark holes or narrow gaps.
- Use **erosion** to break thin bridges; **dilation** to connect nearby parts.
- For uneven lighting, apply **top-hat** or **open + subtract** background correction.
- Always match SE size and direction to the **geometry of the problem**.

Practical Workflow

A typical image-cleaning pipeline:

1. Background correction (if needed)
2. Thresholding or segmentation
3. Morphological cleaning (opening/closing)
4. Object separation (erosion, labeling)
5. Measurement and analysis

Final Thoughts

- Morphological operations are powerful tools for image enhancement and analysis.
- Experiment with different SEs and sequences to find the best approach for your images.
- Combine morphology with other techniques for robust image processing workflows.
- Practice is key to mastering morphological image processing!
- Happy imaging!

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

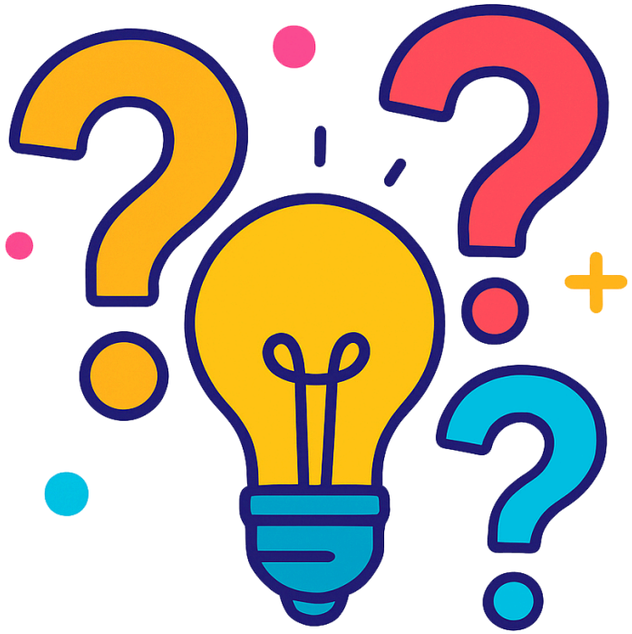
Scan the QR code or visit the link below.

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