

Segmentation Methods

Thresholding techniques

Global, adaptive, Otsu's method

Region growing

Seed-based similar pixel grouping

Watershed algorithm

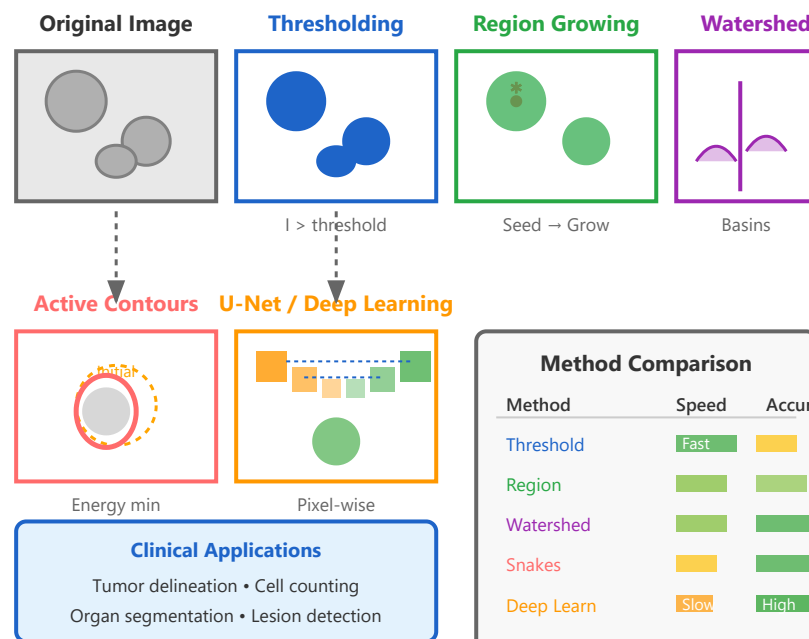
Treating image as topographic surface

Active contours

Energy-minimizing snakes

Machine learning methods

U-Net, Mask R-CNN for segmentation



► Overview

Thresholding is the simplest segmentation method that converts grayscale images into binary images by comparing pixel intensities against a threshold value. This technique is particularly effective for images with clear contrast between objects and background.

► Types of Thresholding

- **Global Thresholding:** Single threshold value applied to entire image ($T = \text{constant}$)
- **Adaptive Thresholding:** Threshold varies across image regions based on local statistics
- **Otsu's Method:** Automatically determines optimal threshold by minimizing intra-class variance
- **Multi-level Thresholding:** Multiple thresholds for segmenting into more than two classes

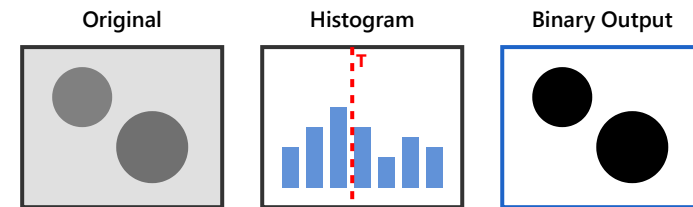
Binary Thresholding Formula:

$$g(x, y) = 1 \text{ if } f(x, y) > T$$
$$g(x, y) = 0 \text{ if } f(x, y) \leq T$$

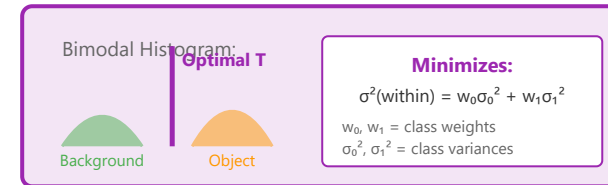
Where: $f(x, y)$ = input intensity, T = threshold,
 $g(x, y)$ = output

► Clinical Applications

Thresholding Process



Otsu's Automatic Threshold Selection



✓ Advantages

- Extremely fast computation
- Simple implementation
- Low memory requirements
- Works well with high contrast

✗ Limitations

- Sensitive to noise and lighting
- Fails with overlapping histograms
- Manual threshold selection needed
- No spatial information used

- Bone segmentation in X-ray images
- Cell nuclei detection in microscopy
- Lung nodule detection in CT scans
- Blood vessel segmentation in angiography

2 Region Growing

► Overview

Region growing is a pixel-based segmentation method that starts from seed points and iteratively adds neighboring pixels with similar properties. The algorithm groups connected pixels that satisfy a homogeneity criterion, making it ideal for segmenting objects with uniform intensity or texture.

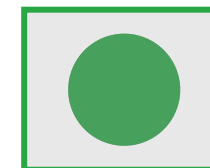
► Algorithm Steps

- **Step 1:** Select initial seed points (manually or automatically)
- **Step 2:** Define similarity criterion (intensity, color, texture)

Region Growing Process

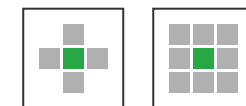


Step 4: Complete!



Connectivity

4-connected: 8-connected:



Decision: $|I - \mu| < T$?

YES → Add to region

NO → Skip pixel

- **Step 3:** Examine neighboring pixels (4-connected or 8-connected)
- **Step 4:** Add similar neighbors to region and update region statistics
- **Step 5:** Repeat until no more pixels can be added

Homogeneity Criterion:

$$|I(x,y) - \mu_{\text{region}}| < T$$

Where:

$I(x,y)$ = pixel intensity

μ_{region} = mean intensity of current region

T = similarity threshold

Key Parameter: The similarity threshold T critically affects results. Too low = under-segmentation, too high = over-segmentation.

✓ **Advantages**

- Produces connected regions
- Works with complex shapes
- Incorporates spatial information
- Can handle multiple seeds

✗ **Limitations**

- Sensitive to seed placement
- Requires user interaction
- Sensitive to noise
- Can leak into adjacent regions

3

Watershed Algorithm

► **Overview**

The watershed algorithm treats the image as a topographic surface where pixel intensities represent elevation. Water "floods" from regional minima, and watershed lines form where different regions meet. This method excels at separating touching or overlapping objects, making it invaluable for cell counting and particle analysis.

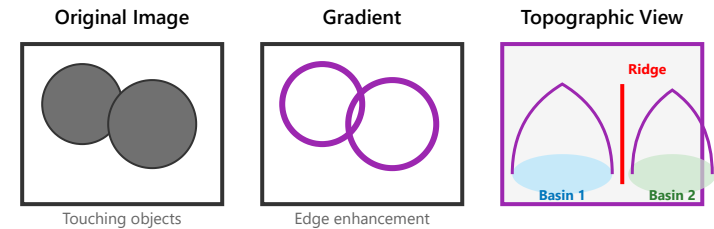
► Key Concepts

- **Topographic Interpretation:** Image treated as 3D landscape (x, y, intensity)
- **Catchment Basins:** Regions where water flows to same minimum
- **Watershed Lines:** Boundaries between adjacent basins (ridges)
- **Flooding Process:** Gradual immersion from minima upward

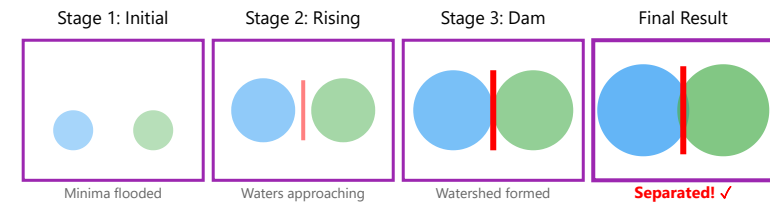
► Implementation Steps

- **Preprocessing:** Compute gradient magnitude of image
- **Marker Extraction:** Identify regional minima or place markers
- **Flooding:** Simulate water rising from each minimum
- **Dam Building:** Create barriers where waters from different basins meet
- **Segmentation:** Each basin becomes a segmented region

Watershed Segmentation



Flooding Process



Marker-Controlled Watershed



✓ Advantages

- Separates touching objects
- Produces closed contours
- Good for particle analysis
- Handles complex topologies

✗ Limitations

- Prone to over-segmentation
- Sensitive to noise
- Requires preprocessing
- Parameter tuning needed

Over-segmentation Problem: Classic watershed often produces too many regions due to noise and local minima. Solution: Marker-controlled watershed with preprocessing.

4 Active Contours (Snakes)

► Overview

Active contours, or "snakes," are deformable curves that move and adapt their shape to fit object boundaries by minimizing an energy functional. The contour evolves iteratively, balancing internal forces (smoothness) and external forces (image features like edges).

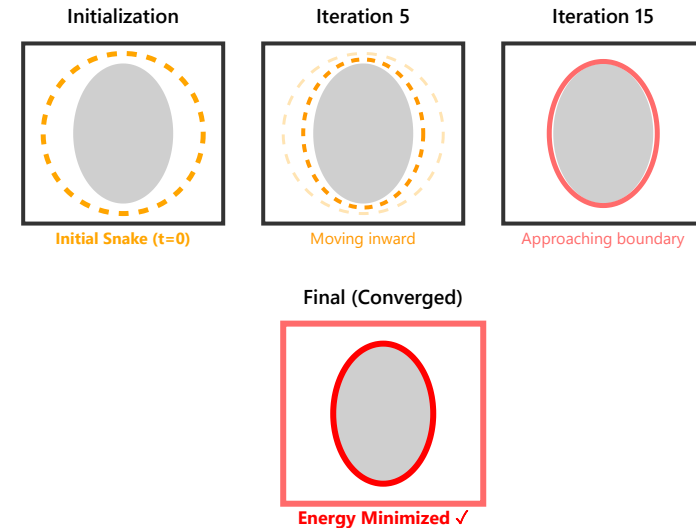
► Energy Functional

The snake minimizes total energy $E = E_{\text{internal}} + E_{\text{external}}$

Energy Components:

$$E_{\text{total}} = E_{\text{internal}} + E_{\text{external}}$$

Active Contour Evolution



Energy Forces

Internal Forces

Continuity (α):
Curvature (β):

External Forces

Edge Attraction

$E_{\text{internal}} = \alpha \cdot E_{\text{continuity}} + \beta \cdot E_{\text{curvature}}$

- Continuity: keeps contour smooth
- Curvature: controls bending

$E_{\text{external}} = E_{\text{image}} + E_{\text{constraint}}$

- Image: attracts to edges/features
- Constraint: user-defined forces

► Types of Active Contours

- **Parametric Snakes:** Explicit contour representation (Kass et al., 1988)
- **Geometric Active Contours:** Level set methods, implicit representation
- **Gradient Vector Flow (GVF):** Extended capture range for initialization
- **Chan-Vese Model:** Region-based, works without edges

Key Advantage: Unlike edge-based methods, active contours produce smooth, closed boundaries and can incorporate prior shape knowledge.

✓ Advantages

- Smooth, closed boundaries
- Can incorporate prior knowledge
- Subpixel accuracy
- Handles topology changes

✗ Limitations

- Requires good initialization
- Computationally intensive
- Sensitive to parameters α, β
- Can get stuck in local minima

► Overview

Modern segmentation leverages deep learning, particularly convolutional neural networks (CNNs), to learn complex feature representations directly from data. These methods have revolutionized medical image analysis, achieving state-of-the-art performance on challenging segmentation tasks without manual feature engineering.

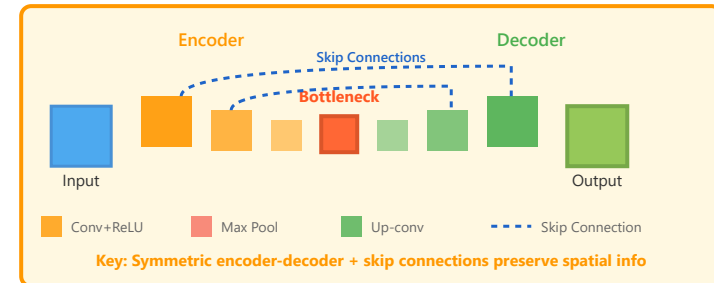
► Key Architectures

- **U-Net:** Encoder-decoder with skip connections for biomedical segmentation
- **Mask R-CNN:** Extends Faster R-CNN for instance segmentation
- **FCN:** Fully Convolutional Network for end-to-end pixel-wise classification
- **DeepLab:** Uses atrous convolution and CRF for semantic segmentation
- **Transformers:** Vision transformers (ViT), Swin-Unet for long-range dependencies

U-Net Revolution: Introduced in 2015 for biomedical segmentation, U-Net remains the gold standard due to its

Deep Learning Segmentation

U-Net Architecture



Performance Metrics

Dice Coefficient:

$$\frac{2|A \cap B|}{|A| + |B|}$$

Typical: 0.85-0.95

IoU (Jaccard):

$$\frac{|A \cap B|}{|A \cup B|}$$

Typical: 0.75-0.90

Pixel Accuracy:

$$\frac{\text{Correct}}{\text{Total}}$$

Typical: >95%

✓ Advantages

- State-of-the-art accuracy

✗ Limitations

- Requires large labeled datasets

ability to work with small training datasets and produce precise localization.

► Training Requirements

- **Annotated Data:** Pixel-level labels required (time-consuming)
- **Data Augmentation:** Rotation, flipping, elastic deformation
- **Loss Functions:** Cross-entropy, Dice loss, focal loss
- **Hardware:** GPU acceleration essential for training

- Learns complex features
- Handles diverse images
- End-to-end training
- Transfer learning possible

- Computationally expensive
- Black box interpretation
- Needs GPU hardware
- Long training time

Method Selection Guide



Need Speed?

→ Thresholding

Real-time processing, simple images with clear contrast



Maximum Accuracy?

→ Deep Learning

When you have labeled training data and GPU resources



Touching Objects?

→ Watershed

Ideal for cell counting, particle analysis, overlapping objects



Smooth Boundaries?



Homogeneous Regions?

→ **Active Contours**

Perfect for organ segmentation, tracking applications

→ **Region Growing**

When user interaction is acceptable and objects are uniform



Hybrid Approach Often Best!

Combine multiple methods for optimal results. Example: Preprocessing with thresholding → Watershed refinement → Deep learning validation