Lecture 04
Morphology and Segmention

2024-02-29

Sébastien Valade

Universidad Nacional Autónoma de México

Introduction

Previous lecture:

convolution: $f(x, y)$, $g(x, y)$, **w**: $\mathbb{N} \to \mathbb{R}$

where w = filter kernel

$\to$ (mostly) linear operators

Today:

morphology: $f(x, y)$, $g(x, y)$, **b**: $\mathbb{N} \to \{0, 1\}$

where b = structuring element

$\to$ non-linear operators

$\to$ concerned with connectivity and shape (close to set theory)

segmentation:

$\to$ labeling image pixels to partition an image into regions

Introduction

Previous lecture:

convolution: $f(x, y)$, $g(x, y)$, **w**: $\mathbb{N} \rightarrow \mathbb{R}$

where w = filter kernel

$\rightarrow$ (mostly) linear operators

Today:

morphology: $f(x, y)$, $g(x, y)$, **b**: $\mathbb{N} \rightarrow \{0, 1\}$

where b = structuring element

$\rightarrow$ non-linear operators

$\rightarrow$ concerned with connectivity and shape (close to set theory)

segmentation:

$\rightarrow$ labeling image pixels to partition an image into regions

Introduction

Previous lecture:

convolution: $f(x, y)$, $g(x, y)$, **w**: $\mathbb{N} \rightarrow \mathbb{R}$

where w = filter kernel

$\rightarrow$ (mostly) linear operators

Today:

morphology: $f(x, y)$, $g(x, y)$, **b**: $\mathbb{N} \rightarrow \{0, 1\}$

where b = structuring element

$\rightarrow$ non-linear operators

$\rightarrow$ concerned with connectivity and shape (close to set theory)

segmentation:

$\rightarrow$ labeling image pixels to partition an image into regions

- Initially proposed for binary images *(Matheron and Serra, 1964)*
  $\rightarrow$ later extended to gray-scale images, and later color images

- Binary images produced by simple thresholding are imperfect due to image noise, etc.
  $\Rightarrow$ **morphological image processing** attempts to remove these imperfections

- Main applications:
  - Image pre-processing (noise filtering, shape simplification)
  - Enhancing object structure (skeletonizing, convex hull, ...)
  - Segmentation
  - Quantitative description of objects (area, perimeter, ...)

- Initially proposed for binary images *(Matheron and Serra, 1964)*
  $\rightarrow$ later extended to gray-scale images, and later color images

- Binary images produced by simple thresholding are imperfect due to image noise, etc.
  $\Rightarrow$ **morphological image processing** attempts to remove these imperfections

- Main applications:
  - Image pre-processing (noise filtering, shape simplification)
  - Enhancing object structure (skeletonizing, convex hull, ...)
  - Segmentation
  - Quantitative description of objects (area, perimeter, ...)

2.1. Basic concepts

- Initially proposed for binary images *(Matheron and Serra, 1964)*
  $\rightarrow$ later extended to gray-scale images, and later color images

- Binary images produced by simple thresholding are imperfect due to image noise, etc.
  $\Rightarrow$ **morphological image processing** attempts to remove these imperfections

- Main applications:
  - Image pre-processing (noise filtering, shape simplification)
  - Enhancing object structure (skeletonizing, convex hull, ...)
  - Segmentation
  - Quantitative description of objects (area, perimeter, ...)

Morphological filtering mechanics are similar to spatial filtering using convolutions:

1) a kernel called a **structuring element** is used to determine filtering operation:

- the size is determined by the matrix dimensions
- the shape is determined by the pattern of 1 and 0 in the matrix
- the origin is usually the matrix center, although it can also off-centered or even outside it

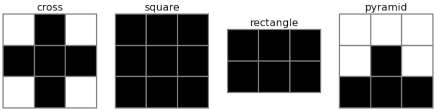
NB: like convolution kernels, it is common to have structuring elements of odd dimensions with the center as the origin.
NB: the shape, size, and orientation of the structuring element depends on application

Morphological filtering mechanics are similar to spatial filtering using convolutions:

1) a kernel called a **structuring element** is used to determine filtering operation:

- the size is determined by the matrix dimensions
- the shape is determined by the pattern of 1 and 0 in the matrix
- the origin is usually the matrix center, although it can also off-centered or even outside it

NB: like convolution kernels, it is common to have structuring elements of odd dimensions with the center as the origin.
NB: the shape, size, and orientation of the structuring element depends on application

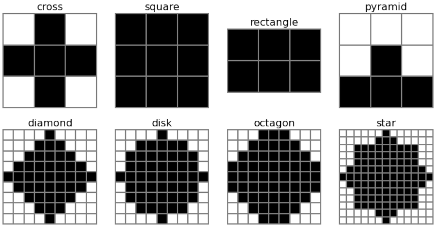


Morphological filtering mechanics are similar to spatial filtering using convolutions:

1) a kernel called a **structuring element** is used to determine filtering operation:

- the size is determined by the matrix dimensions
- the shape is determined by the pattern of 1 and 0 in the matrix
- the origin is usually the matrix center, although it can also off-centered or even outside it

NB: like convolution kernels, it is common to have structuring elements of odd dimensions with the center as the origin.
NB: the shape, size, and orientation of the structuring element depends on application

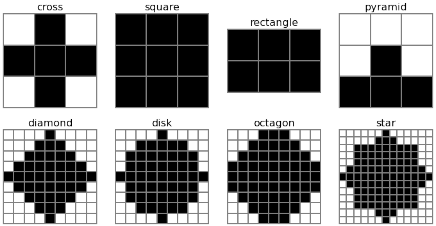


Morphological filtering mechanics are similar to spatial filtering using convolutions:

1) a kernel called a **structuring element** is used to determine filtering operation:

- the size is determined by the matrix dimensions
- the shape is determined by the pattern of 1 and 0 in the matrix
- the origin is usually the matrix center, although it can also off-centered or even outside it

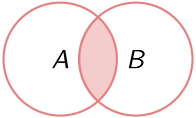
NB: like convolution kernels, it is common to have structuring elements of odd dimensions with the center as the origin.
NB: the shape, size, and orientation of the structuring element depends on application



2) the image is first **padded**, and the structuring element than **slides** across it

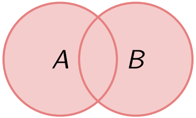
Morphological filters are essentially set operations

Intersection (AND)



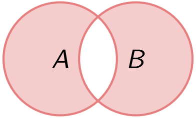
$A \cap B = \{x : x \in A \textit{ and } x \in B\}$

Union (OR)



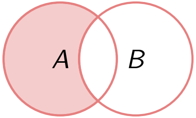
$A \cup B = \{x : x \in A \textit{ or } x \in B\}$

Symmetric difference (XOR)



$\overline{A \cap B}$

Difference



$A - B$

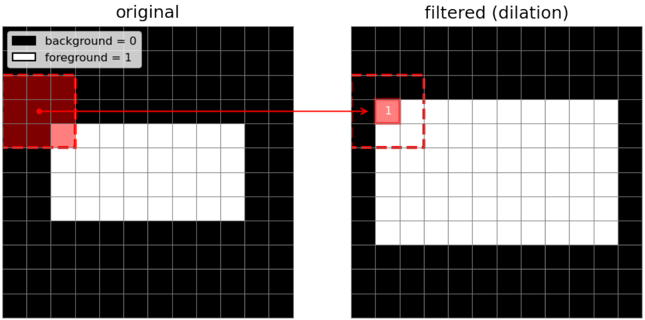
$\rightarrow$ Primary Morphological Operations are: **dilation** and **erosion**

$\rightarrow$ Concatenation of dilation and erosion result in higher level operations

$\Rightarrow$ Composite Morphological Operations: **closing** and **opening**

1. **Dilation**: the dilation of a set F with a structuring element b is defined as:

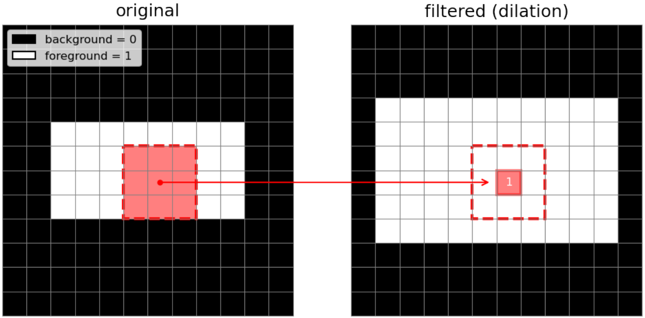
$$G = F \oplus b = \{x : \left(\hat{b}\right)_x \cap F \neq \emptyset\}$$



if >= 1 pixel within the mask = "1", the result is "1", otherwise "0"

1. **Dilation**: the dilation of a set F with a structuring element b is defined as:

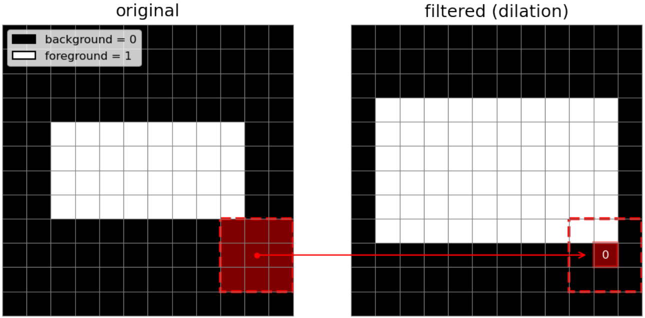
$$G = F \oplus b = \{x : \left(\hat{b}\right)_x \cap F \neq \emptyset\}$$



if >= 1 pixel within the mask = "1", the result is "1", otherwise "0"

1. **Dilation**: the dilation of a set F with a structuring element b is defined as:

$$G = F \oplus b = \{x : \left(\hat{b}\right)_x \cap F \neq \emptyset\}$$



if >= 1 pixel within the mask = "1", the result is "1", otherwise "0"

1. **Dilation**: the dilation of a set F with a structuring element b is defined as:

⇒ Foreground objects get larger
⇒ Background objects get smaller
⇒ Small gaps are closed

original

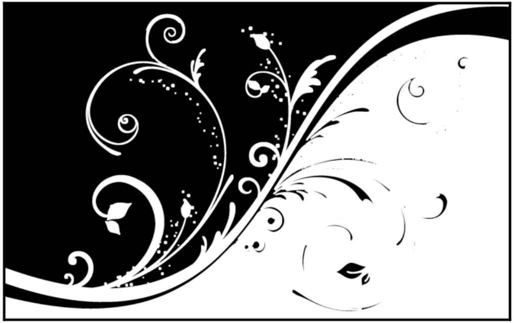
dilation (b=**3x3**)

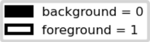
1. **Dilation**: the dilation of a set F with a structuring element b is defined as:

$\Rightarrow$ Foreground objects get larger
$\Rightarrow$ Background objects get smaller
$\Rightarrow$ Small gaps are closed

original

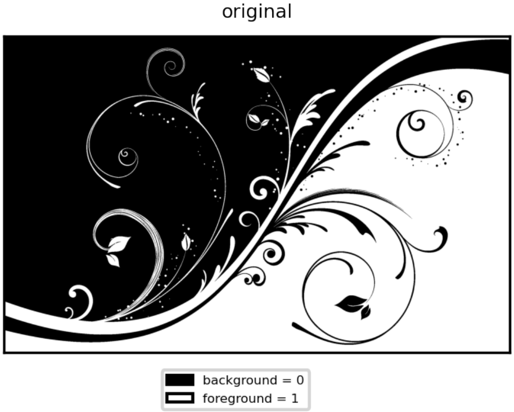
dilation (b=**7x7**)





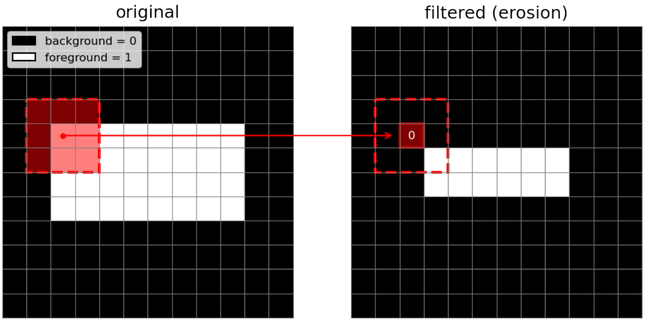
1. **Dilation**: the dilation of a set F with a structuring element b is defined as:

⇒ Foreground objects get larger
⇒ Background objects get smaller
⇒ Small gaps are closed



2. **Erosion**: the erosion of a set F with a structuring element b is defined as:

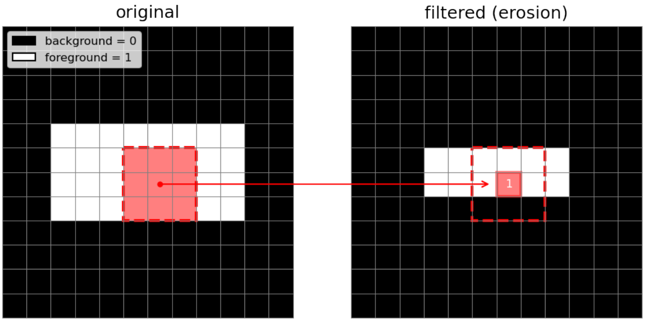
$$G = F \ominus b = \{x : (b)_x \subseteq F\}$$



if all pixel within the mask = "1" => the result is "1", otherwise "0"

2. **Erosion**: the erosion of a set F with a structuring element b is defined as:

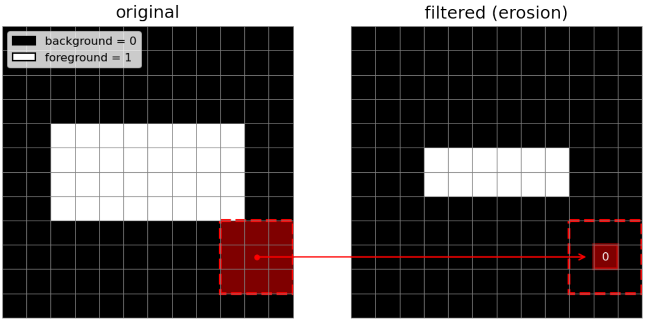
$$G = F \ominus b = \{x : (b)_x \subseteq F\}$$



if all pixel within the mask = "1" => the result is "1", otherwise "0"

2. **Erosion**: the erosion of a set F with a structuring element b is defined as:

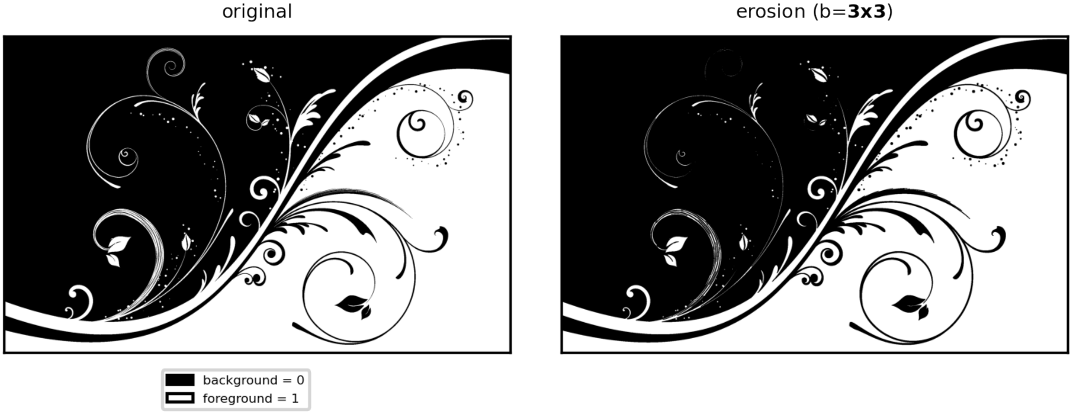
$$G = F \ominus b = \{x : (b)_x \subseteq F\}$$



if all pixel within the mask = "1" => the result is "1", otherwise "0"

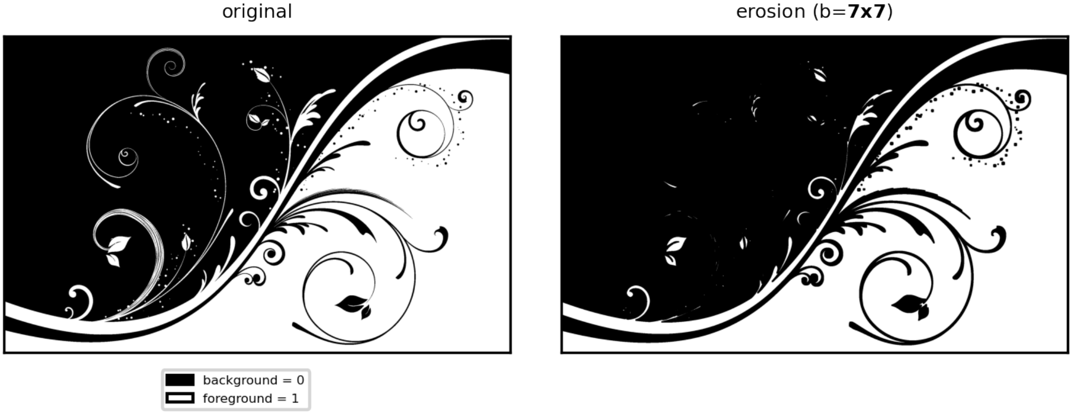
2. **Erosion**: the erosion of a set F with a structuring element b is defined as:

- $\Rightarrow$ Foreground objects get smaller, small objects disappear
- $\Rightarrow$ Background objects get larger
- $\Rightarrow$ Objects get separated



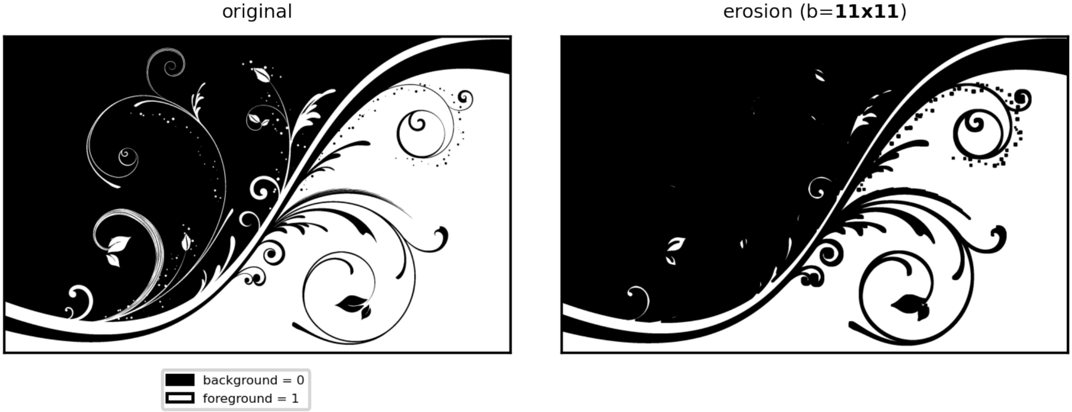
2. **Erosion**: the erosion of a set F with a structuring element b is defined as:

⇒ Foreground objects get smaller, small objects disappear
⇒ Background objects get larger
⇒ Objects get separated



2. **Erosion**: the erosion of a set F with a structuring element b is defined as:

$\Rightarrow$ Foreground objects get smaller, small objects disappear
$\Rightarrow$ Background objects get larger
$\Rightarrow$ Objects get separated



Concatenation of **dilation** and **erosion** result in higher level operations: **closing**, **opening**

1. **Opening**:

   Problem: erosion causes deletion of small objects, BUT other objects shrink
   Solution: after *erosion*, apply *dilation* with the same structuring element $\Rightarrow$ **opening**

   $$G = F \circ b = (F \ominus b) \oplus b$$

   Usage example: removing small isolated "bright spots" *(EX: volcanic SO2 detection from Sentinel-5P as foreground (mask=1))*

Concatenation of **dilation** and **erosion** result in higher level operations: **closing**, **opening**

1. **Opening**:

Problem: erosion causes deletion of small objects, BUT other objects shrink
Solution: after *erosion*, apply *dilation* with the same structuring element $\Rightarrow$ **opening**

$$G = F \circ b = (F \ominus b) \oplus b$$

Usage example: removing small isolated “bright spots” *(EX: volcanic SO2 detection from Sentinel-5P as foreground (mask=1))*

Concatenation of **dilation** and **erosion** result in higher level operations: **closing**, **opening**

1. **Opening**:

   Problem: erosion causes deletion of small objects, BUT other objects shrink
   Solution: after *erosion*, apply *dilation* with the same structuring element $\Rightarrow$ **opening**

$$G = F \circ b = (F \ominus b) \oplus b$$

   Usage example: removing small isolated "bright spots" *(EX: volcanic SO2 detection from Sentinel-5P as foreground (mask=1))*

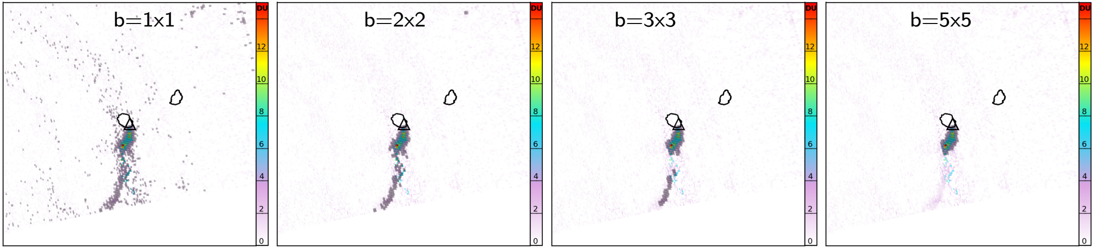
Concatenation of **dilation** and **erosion** result in higher level operations: **closing**, **opening**

1. **Opening**:

Problem: erosion causes deletion of small objects, BUT other objects shrink
Solution: after *erosion*, apply *dilation* with the same structuring element $\Rightarrow$ **opening**

$$G = F \circ b = (F \ominus b) \oplus b$$

Usage example: removing small isolated "bright spots" *(EX: volcanic SO2 detection from Sentinel-5P as foreground (mask=1))*



Concatenation of **dilation** and **erosion** result in higher level operations: **closing**, **opening**

2. **Closing**:

Problem: dilation closes small holes and fractions, BUT objects get enlarged
Solution: after *dilation*, apply *erosion* with the same structuring element $\Rightarrow$ **closing**

$$G = F \bullet b = (F \oplus b) \ominus b$$

Usage example: removing small isolated “dark spots” (binary mask value = 0)

Concatenation of **dilation** and **erosion** result in higher level operations: **closing**, **opening**

2. **Closing**:

Problem: dilation closes small holes and fractions, BUT objects get enlarged
Solution: after *dilation*, apply *erosion* with the same structuring element $\Rightarrow$ **closing**

$$G = F \bullet b = (F \oplus b) \ominus b$$

Usage example: removing small isolated “dark spots” (binary mask value = 0)

Concatenation of **dilation** and **erosion** result in higher level operations: **closing**, **opening**

2. **Closing**:

Problem: dilation closes small holes and fractions, BUT objects get enlarged
Solution: after *dilation*, apply *erosion* with the same structuring element $\Rightarrow$ **closing**

$$G = F \bullet b = (F \oplus b) \ominus b$$

Usage example: removing small isolated "dark spots" (binary mask value $= 0$)

Concatenation of **dilation** and **erosion** result in higher level operations: **closing**, **opening**

2. Closing:

Problem: dilation closes small holes and fractions, BUT objects get enlarged
Solution: after *dilation*, apply *erosion* with the same structuring element $\Rightarrow$ **closing**

$$G = F \bullet b = (F \oplus b) \ominus b$$

Usage example: removing small isolated "dark spots" (binary mask value = 0)

original | binarized | closing b=3x3 | closing b=7x7

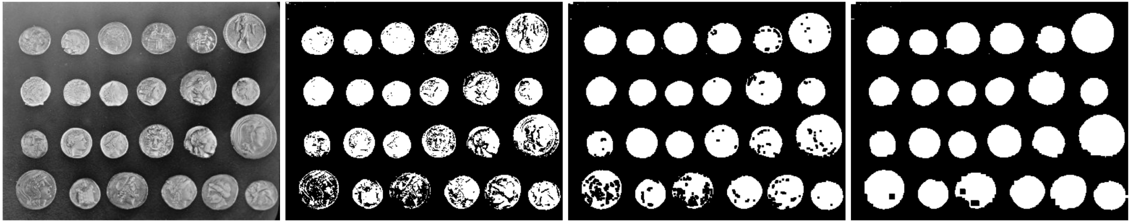


Image segmentation = labeling image pixels to partition an image into regions

- Histogram-based segmentation
  $\Rightarrow$ based on thresholding of pixel values
  - ex: manual thresholding
  - ex: automatic thresholding (e.g., Otsu)
  - ex: k-means clustering
- Edge-based segmentation
  $\Rightarrow$ based on local contrast $\rightarrow$ uses gradients rather than the grey values
- Region-based segmentation
  $\Rightarrow$ based on image gradients and region properties
  - ex: Watershed transform
  - ex: Random Walker
  - ex: Flood Fill
- Many other!
  - ex: Graph-cuts
  - ex: Active Contours, Region Growing, Weighted Pyramid Linking, Mean-Shift, etc.

Image segmentation = labeling image pixels to partition an image into regions

- Histogram-based segmentation
  ⇒ based on thresholding of pixel values
  - ex: manual thresholding
  - ex: automatic thresholding (e.g., Otsu)
  - ex: k-means clustering
- Edge-based segmentation
  ⇒ based on local contrast → uses gradients rather than the grey values
- Region-based segmentation
  ⇒ based on image gradients and region properties
  - ex: Watershed transform
  - ex: Random Walker
  - ex: Flood Fill
- Many other!
  - ex: Graph-cuts
  - ex: Active Contours, Region Growing, Weighted Pyramid Linking, Mean-Shift, etc.

Image segmentation = labeling image pixels to partition an image into regions

- Histogram-based segmentation
  ⇒ based on thresholding of pixel values
    ex: manual thresholding
    ex: automatic thresholding (e.g., Otsu)
    ex: k-means clustering

- Edge-based segmentation
  ⇒ based on local contrast → uses gradients rather than the grey values

- Region-based segmentation
  ⇒ based on image gradients and region properties
    ex: Watershed transform
    ex: Random Walker
    ex: Flood Fill

- Many other!
    ex: Graph-cuts
    ex: Active Contours, Region Growing, Weighted Pyramid Linking, Mean-Shift, etc.

Image segmentation = labeling image pixels to partition an image into regions

- Histogram-based segmentation
  $\Rightarrow$ based on thresholding of pixel values
  - ex: manual thresholding
  - ex: automatic thresholding (e.g., Otsu)
  - ex: k-means clustering

- Edge-based segmentation
  $\Rightarrow$ based on local contrast $\rightarrow$ uses gradients rather than the grey values

- Region-based segmentation
  $\Rightarrow$ based on image gradients and region properties
  - ex: Watershed transform
  - ex: Random Walker
  - ex: Flood Fill

- Many other!
  - ex: Graph-cuts
  - ex: Active Contours, Region Growing, Weighted Pyramid Linking, Mean-Shift, etc.

Histogram-based segmentation

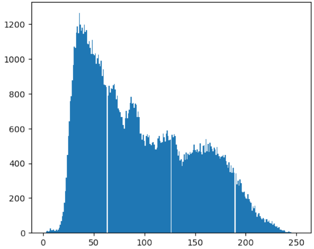
$\Rightarrow$ based on thresholding pixel values

Histogram-based segmentation

⇒ based on thresholding pixel values

- global thresholding
  - manual
  - automatic (e.g. Otsu's method)

    (threshold calculated to separate pixels into two classes, minimizing intra-class intensity variance)

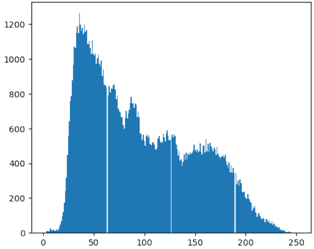


Histogram-based segmentation

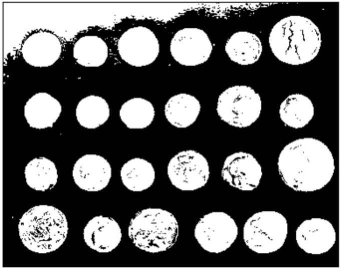
⇒ based on thresholding pixel values

- global thresholding
  - manual
  - automatic (e.g. Otsu's method)
  (threshold calculated to separate pixels into two classes, minimizing intra-class intensity variance)

original



automatic threshold (Otsu thresh=107)

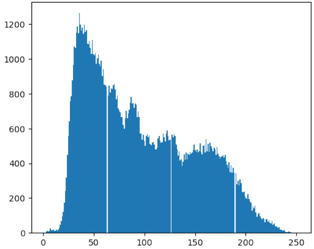


Histogram-based segmentation

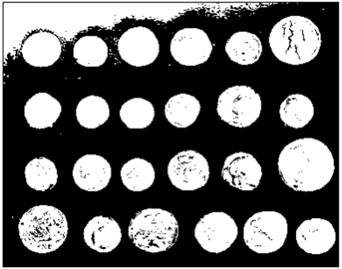
⇒ based on thresholding pixel values

- global thresholding
  - manual
  - automatic (e.g. Otsu's method)
  (threshold calculated to separate pixels into two classes, minimizing intra-class intensity variance)

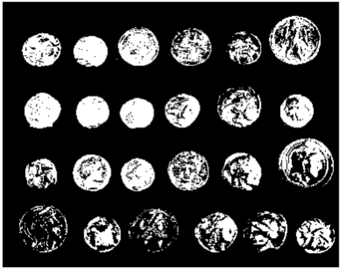
original



automatic threshold (Otsu thresh=107)



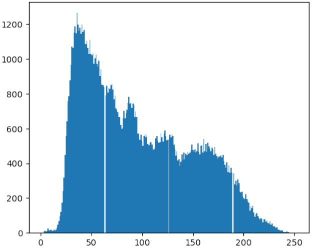
manual threshold (thresh=150)

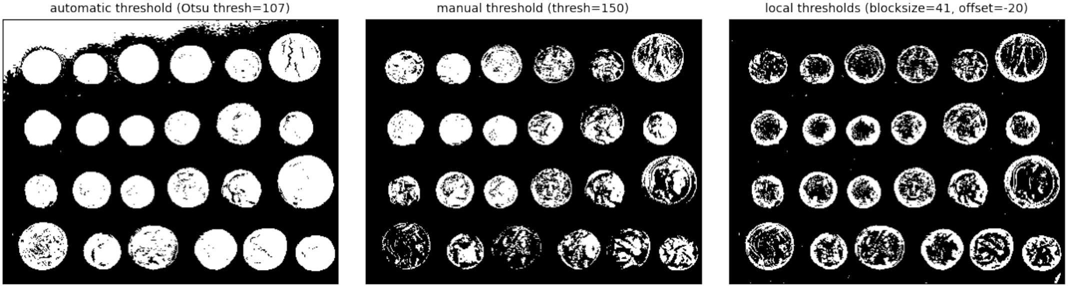


Histogram-based segmentation

⇒ based on thresholding pixel values

- global thresholding
  - manual
  - automatic (e.g. Otsu's method)
  (threshold calculated to separate pixels into two classes, minimizing intra-class intensity variance)
- local thresholding (adaptive)
  (thresholds calculated based on pixel local neighborhood)



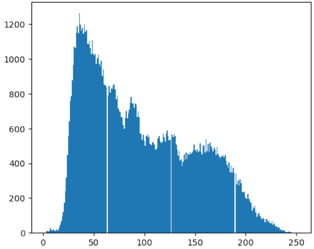


Histogram-based segmentation

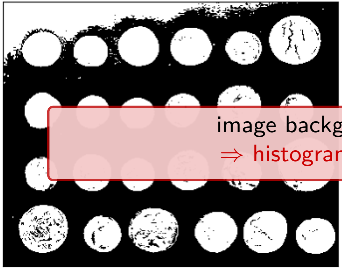
⇒ based on thresholding pixel values

- global thresholding
  - manual
  - automatic (e.g. Otsu's method)
  (threshold calculated to separate pixels into two classes, minimizing intra-class intensity variance)
- local thresholding (adaptive)
  (thresholds calculated based on pixel local neighborhood)

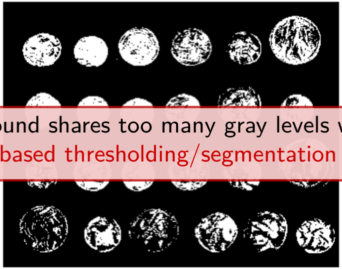
original



automatic threshold (Otsu thresh=107)



manual threshold (thresh=150)

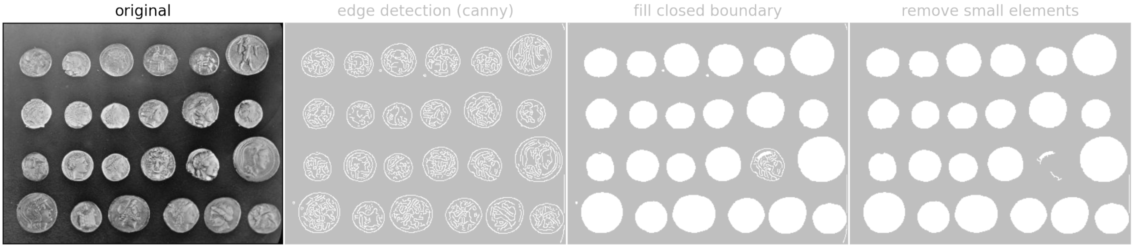


local thresholds (blocksize=41, offset=-20)

image background shares too many gray levels with the coins
⇒ histogram-based thresholding/segmentation is insufficient

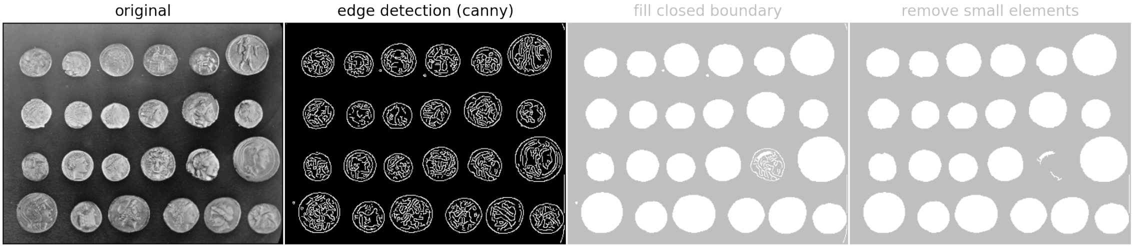
Edge-based segmentation

⇒ based on image gradients



Edge-based segmentation

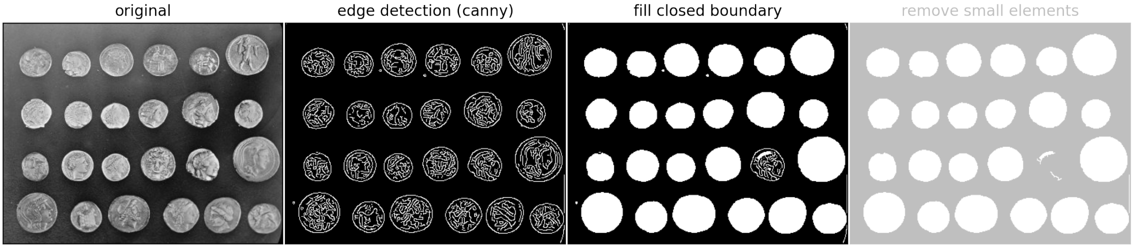
$\Rightarrow$ based on image gradients



1. apply Canny edge detection algorithm (involves gradient detection using e.g. Sobel operator)

Edge-based segmentation

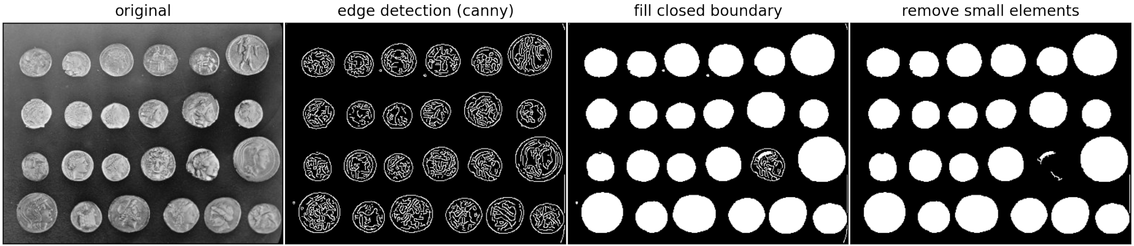
$\Rightarrow$ based on image gradients



1. apply Canny edge detection algorithm (involves gradient detection using e.g. Sobel operator)
2. apply mathematical morphology to fill inner part of the coins

Edge-based segmentation

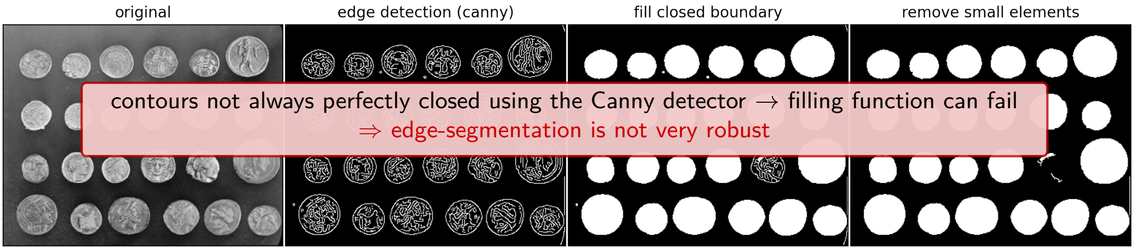
$\Rightarrow$ based on image gradients



1. apply Canny edge detection algorithm (involves gradient detection using e.g. Sobel operator)
2. apply mathematical morphology to fill inner part of the coins
3. remove objects smaller than a threshold

Edge-based segmentation

$\Rightarrow$ based on image gradients



contours not always perfectly closed using the Canny detector $\rightarrow$ filling function can fail
$\Rightarrow$ edge-segmentation is not very robust

1. apply Canny edge detection algorithm (involves gradient detection using e.g. Sobel operator)
2. apply mathematical morphology to fill inner part of the coins
3. remove objects smaller than a threshold

Region-based segmentation: *watershed transform*

$\Rightarrow$ region-growing approach that fills "basins" in the image

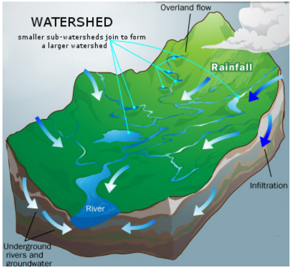
Region-based segmentation: *watershed transform*

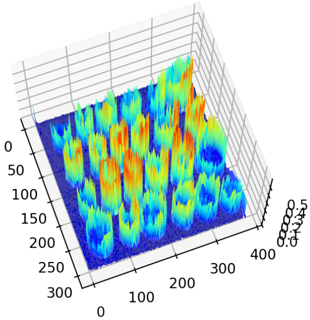
⇒ region-growing approach that fills "basins" in the image

⇒ the name "watershed" comes from an analogy with hydrology:

→ the *watershed transform* "floods" a "topographic" representation of the image

→ flooding starts from "markers", in order to determine the catchment basins of these markers





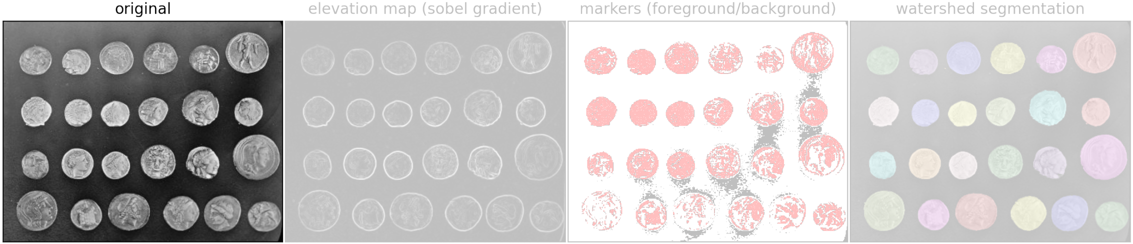
Region-based segmentation: *watershed transform*

⇒ region-growing approach that fills "basins" in the image

⇒ the name "watershed" comes from an analogy with hydrology:

→ the *watershed transform* "floods" a "topographic" representation of the image

→ flooding starts from "markers", in order to determine the catchment basins of these markers

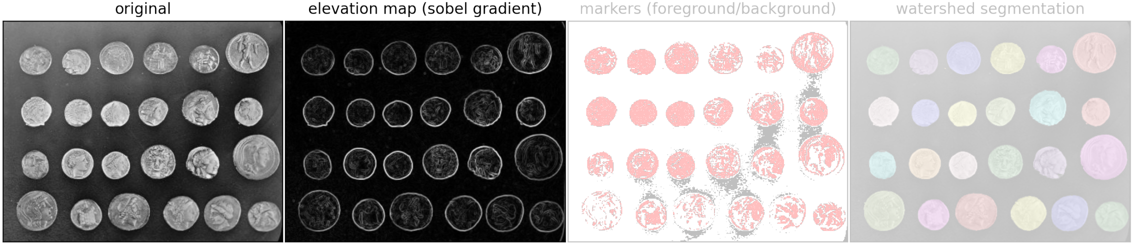


Region-based segmentation: *watershed transform*

⇒ region-growing approach that fills “basins” in the image

⇒ the name ”watershed” comes from an analogy with hydrology:

- → the *watershed transform* ”floods” a ”topographic” representation of the image
- → flooding starts from ”markers”, in order to determine the catchment basins of these markers



1. build “elevation map” from image gradient amplitude (using the Sobel operator)

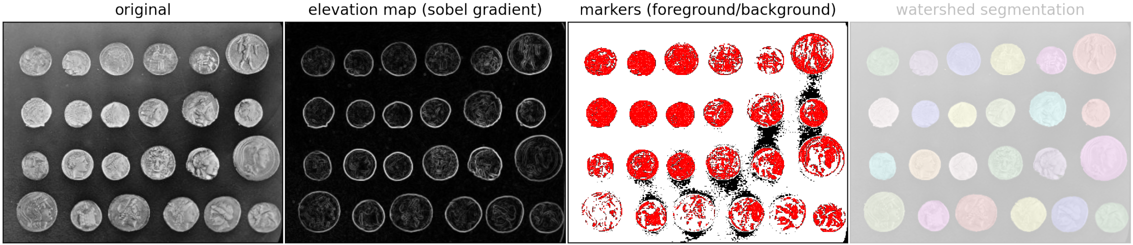
Region-based segmentation: *watershed transform*

⇒ region-growing approach that fills "basins" in the image

⇒ the name "watershed" comes from an analogy with hydrology:

→ the *watershed transform* "floods" a "topographic" representation of the image

→ flooding starts from "markers", in order to determine the catchment basins of these markers



1. build "elevation map" from image gradient amplitude (using the Sobel operator)
2. define markers for background / foreground (here based on the extreme parts of the histogram)

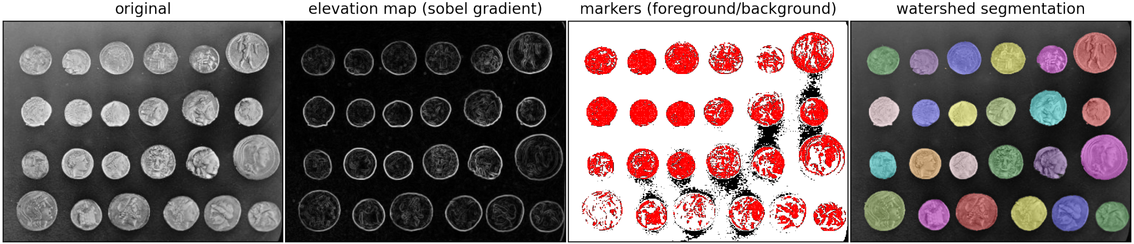
Region-based segmentation: *watershed transform*

⇒ region-growing approach that fills "basins" in the image

⇒ the name "watershed" comes from an analogy with hydrology:

→ the *watershed transform* "floods" a "topographic" representation of the image

→ flooding starts from "markers", in order to determine the catchment basins of these markers



1. build "elevation map" from image gradient amplitude (using the Sobel operator)
2. define markers for background / foreground (here based on the extreme parts of the histogram)
3. apply watershed transform (and colorize segmented elements)

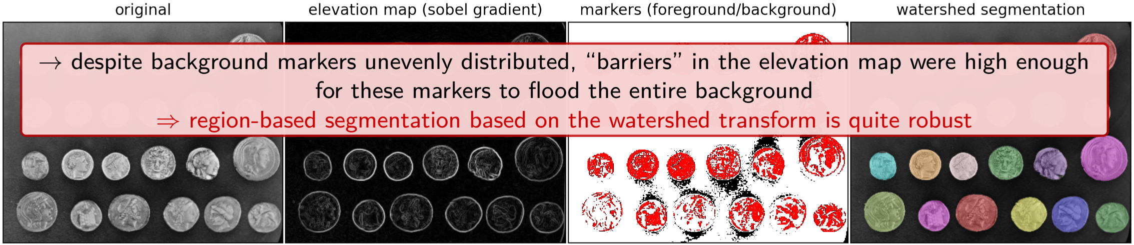
Region-based segmentation: *watershed transform*

⇒ region-growing approach that fills “basins” in the image

⇒ the name ”watershed” comes from an analogy with hydrology:

→ the *watershed transform* ”floods” a ”topographic” representation of the image

→ flooding starts from ”markers”, in order to determine the catchment basins of these markers

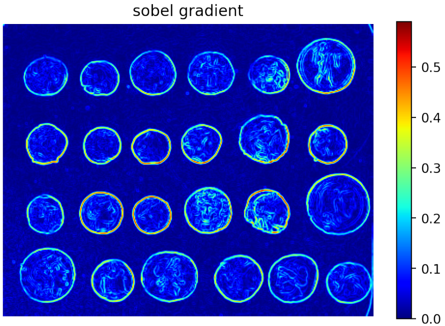


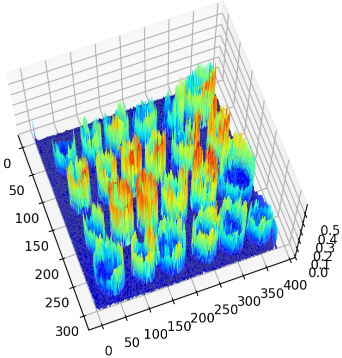
→ despite background markers unevenly distributed, “barriers” in the elevation map were high enough for these markers to flood the entire background

⇒ region-based segmentation based on the watershed transform is quite robust

1. build “elevation map” from image gradient amplitude (using the Sobel operator)
2. define markers for background / foreground (here based on the extreme parts of the histogram)
3. apply watershed transform (and colorize segmented elements)

Region-based segmentation: *watershed transform*





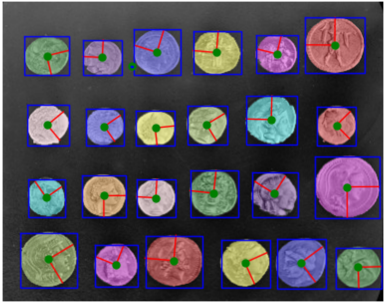
1. Start with lowest "altitude" (Gradient amplitude)
2. Increase the "water level" each time by 1
3. Merge all connected pixel with same/less level

Analyze segmented image

The segmented elements can be analysed indidually to:

→ provide statistics on their shape, distribution, orientation, etc.

(e.g. fields in a satellite image, crystal/bubble shape distribution in a rock sample, etc.)

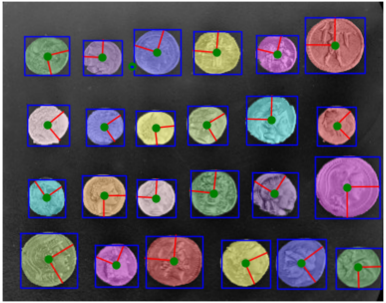


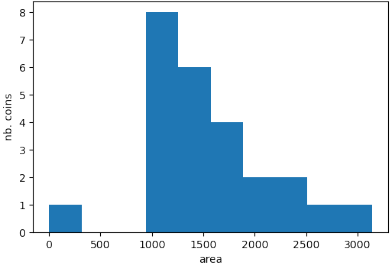
Analyze segmented image

The segmented elements can be analysed indidually to:

$\rightarrow$ provide statistics on their shape, distribution, orientation, etc.

(e.g. fields in a satellite image, crystal/bubble shape distribution in a rock sample, etc.)





Analyze segmented image

Exercises:

1. Exercise 1:
   $\Rightarrow$ histogram-based segmentation of Popocatépetl
2. Exercise 2:
   $\Rightarrow$ analyze a thermal infrared image of a lava lake
   $\rightarrow$ segment the crustal plates from the incandescent cracks and analyze