Lecture 04
Morphology and Segmention

2024-09-04

Sébastien Valade

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

POR MI RAZA HABLARA EL ESPIRITU

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Introduction

Previous lecture:

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

where w = filter kernel

$\rightarrow$ (mostly) linear operators

Today:

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

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**: $\mathbb{N} \to \mathbb{R}$

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

Mathematical morphology

- Initially proposed for *binary images*, to *quantify minerals characteristics from thin cross sections* (Matheron and Serra, 1964)
  - → compute size distribution of spheres
  - → introduction of the concepts of *opening/closing*, *erosion/dilation*

- Later extended to *gray-scale images*, and later *color images*.

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

Mathematical morphology

- Initially proposed for *binary images*, to *quantify minerals characteristics from thin cross sections* (Matheron and Serra, 1964)
  - → compute size distribution of spheres
  - → introduction of the concepts of *opening/closing*, *erosion/dilation*
- Later extended to *gray-scale images*, and later *color images*.
- Main applications:
  - Image pre-processing (noise filtering, shape simplification)
  - Enhancing object structure (skeletonizing, convex hull, ...)
  - Segmentation
  - Quantitative description of objects (area, perimeter, ...)

Mathematical morphology

- Initially proposed for *binary images*, to *quantify minerals characteristics from thin cross sections* (Matheron and Serra, 1964)
  → compute size distribution of spheres
  → introduction of the concepts of *opening/closing*, *erosion/dilation*

- Later extended to *gray-scale images*, and later *color images*.

- 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

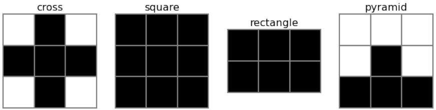
NB: like convolution kernels, it is common to have structuring elements of odd dimensions with the center as the origin.
NB: the shape, size, and orientation of the structuring element depends on application

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

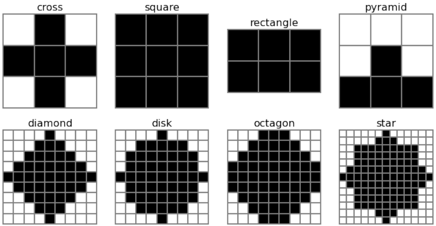


Morphological filtering mechanics are similar to spatial filtering using convolutions:

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

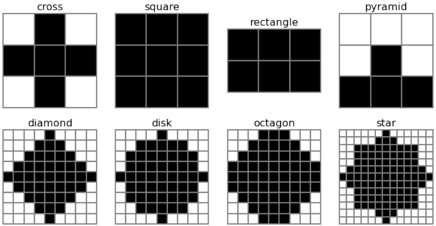


Morphological filtering mechanics are similar to spatial filtering using convolutions:

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

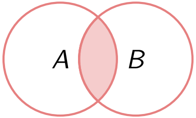
NB: like convolution kernels, it is common to have structuring elements of odd dimensions with the center as the origin.
NB: the shape, size, and orientation of the structuring element depends on application



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

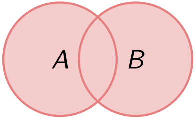
Morphological filters are essentially set operations

Intersection (AND)



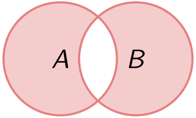
$A \cap B = \{x : x \in A \textit{ and } x \in B\}$

Union (OR)



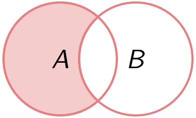
$A \cup B = \{x : x \in A \textit{ or } x \in B\}$

Symmetric difference (XOR)



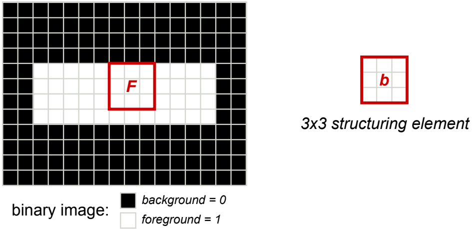
$\overline{A \cap B}$

Difference



$A - B$

Considering a **set of pixels F** of a binary image, and a **structuring element b**:



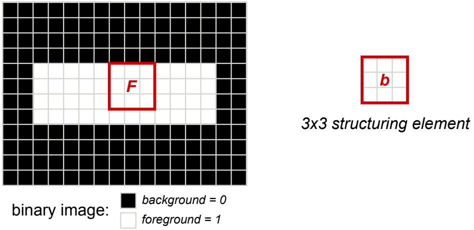
Primitive Morphological Operations:

- $\Rightarrow$ **dilation**: $F \oplus b$ $\rightarrow$ *growth* of foreground pixels
- $\Rightarrow$ **erosion**: $F \ominus b$ $\rightarrow$ *shrinkage* of foreground pixels

Composite Morphological Operations

- $\Rightarrow$ **closing**: $F \bullet b = (F \oplus b) \ominus b$ $\rightarrow$ concatenation of *dilation* and *erosion*
- $\Rightarrow$ **opening**: $F \circ b = (F \ominus b) \oplus b$ $\rightarrow$ concatenation of *erosion* and *dilation*

Considering a **set of pixels F** of a binary image, and a **structuring element b**:



Primitive Morphological Operations:

⇒ **dilation**: $F \oplus b$ → *growth* of foreground pixels

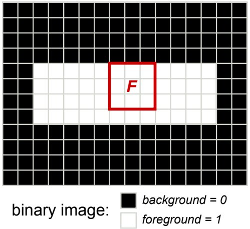
⇒ **erosion**: $F \ominus b$ → *shrinkage* of foreground pixels

Composite Morphological Operations

⇒ **closing**: $F \bullet b = (F \oplus b) \ominus b$ → concatenation of *dilation* and *erosion*

⇒ **opening**: $F \circ b = (F \ominus b) \oplus b$ → concatenation of *erosion* and *dilation*

Considering a **set of pixels F** of a binary image, and a **structuring element b**:



Primitive Morphological Operations:

$\Rightarrow$ **dilation**: $F \oplus b$ $\rightarrow$ *growth* of foreground pixels

$\Rightarrow$ **erosion**: $F \ominus b$ $\rightarrow$ *shrinkage* of foreground pixels

Composite Morphological Operations

$\Rightarrow$ **closing**: $F \bullet b = (F \oplus b) \ominus b$ $\rightarrow$ concatenation of *dilation* and *erosion*

$\Rightarrow$ **opening**: $F \circ b = (F \ominus b) \oplus b$ $\rightarrow$ concatenation of *erosion* and *dilation*

Primitive Morphological Operations

1. **Dilation**

$\Rightarrow$ *mathematical definition*: the dilation of set F withstructuring element b is a Minkowski addition:

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

$\Rightarrow$ *in words*: if $\geq$1 element of F intersects $\hat{b}$ then assign "1" to center of b, else assign "0"

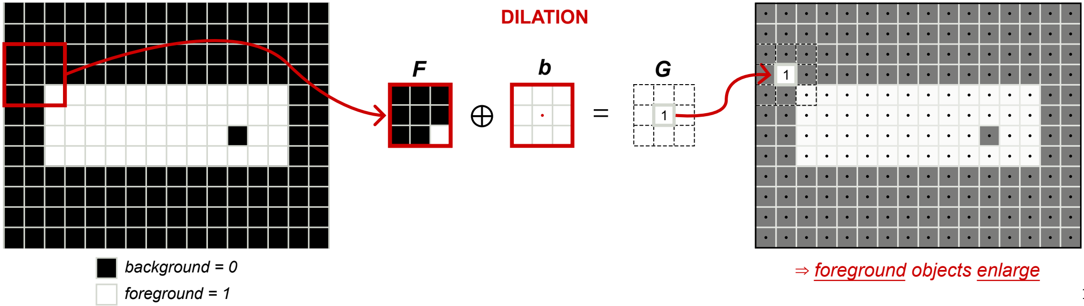
Primitive Morphological Operations

1. Dilation

⇒ *mathematical definition*: the dilation of set F withstructuring element b is a Minkowski addition:

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

⇒ *in words*: if $\geq$1 element of F intersects $\hat{b}$ then assign "1" to center of b, else assign "0"



⇒ *foreground objects enlarge*

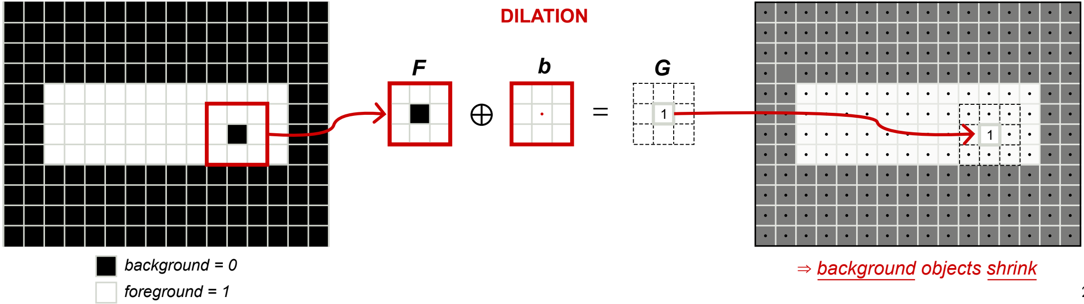
Primitive Morphological Operations

1. Dilation

⇒ *mathematical definition*: the dilation of set F withstructuring element b is a Minkowski addition:

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

⇒ *in words*: if ≥1 element of F intersects $\hat{b}$ then assign "1" to center of b, else assign "0"

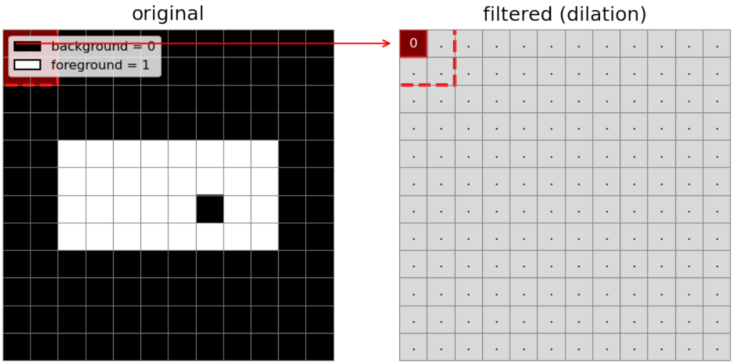


⇒ *background objects shrink*

Primitive Morphological Operations

1. **Dilation**

$\Rightarrow$ the structuring element *slides* across the entire image

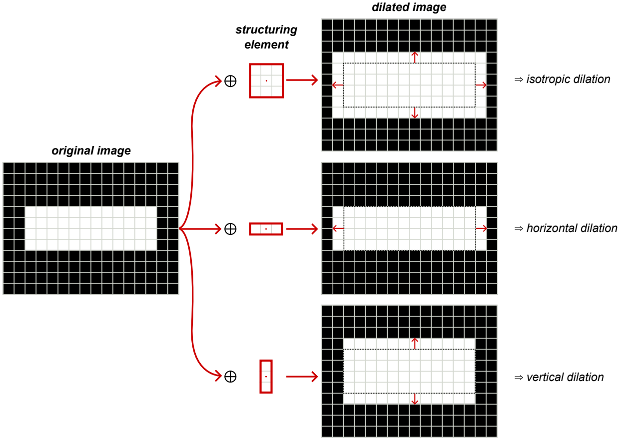


Note: this slide contains an animation, please view the PDF with a compatible viewer

Primitive Morphological Operations

1. **Dilation**

$\Rightarrow$ ***shape*** *of the structuring element* b determines the effect of the dilation



Primitive Morphological Operations

1. **Dilation**

$\Rightarrow$ ***size** of the structuring element* b determines the effect of the dilation

original

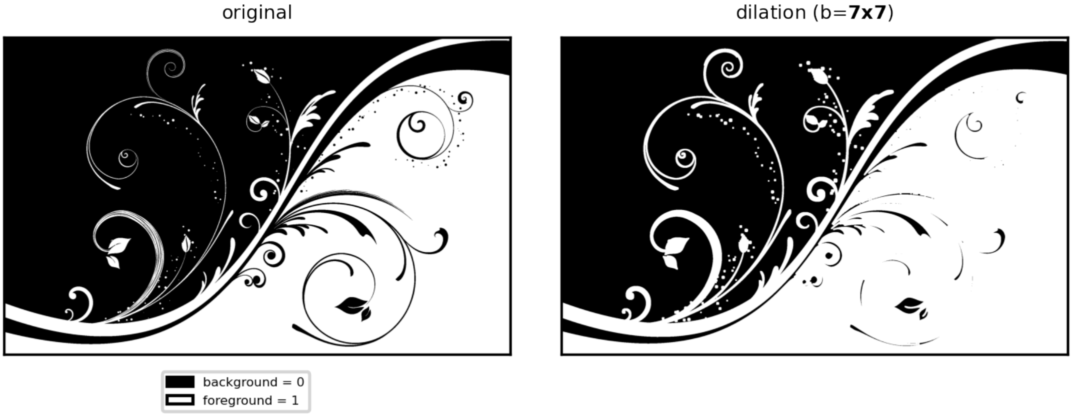
dilation (b=**3x3**)

background = 0
foreground = 1

Primitive Morphological Operations

1. **Dilation**

$\Rightarrow$ ***size** of the structuring element* b determines the effect of the dilation



Primitive Morphological Operations

1. **Dilation**

⇒ ***size** of the structuring element* b determines the effect of the dilation

original

dilation (b=**11x11**)

background = 0
foreground = 1

Primitive Morphological Operations

2. Erosion

⇒ *mathematical definition*: the erosion of set F with structuring element b is defined as:

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

⇒ *in words*: if all elements of F within b are 1, then assign “1” to center of b, else assign “0”

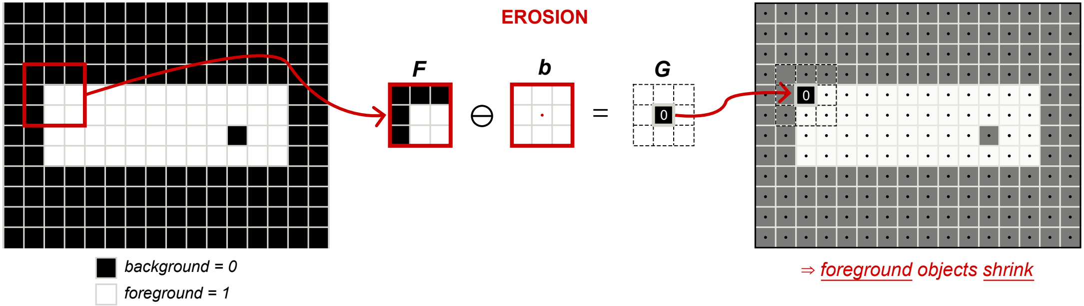
Primitive Morphological Operations

2. Erosion

⇒ *mathematical definition*: the erosion of set F with structuring element b is defined as:

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

⇒ *in words*: if all elements of F within b are 1, then assign "1" to center of b, else assign "0"



⇒ foreground objects shrink

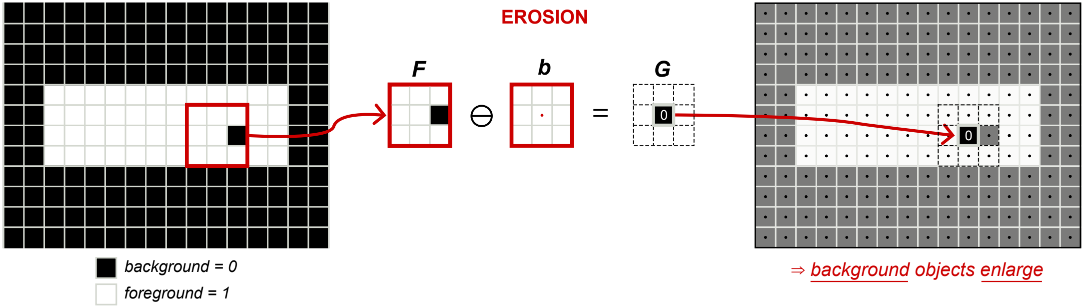
Primitive Morphological Operations

2. Erosion

⇒ *mathematical definition*: the erosion of set F with structuring element b is defined as:

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

⇒ *in words*: if all elements of F within b are 1, then assign "1" to center of b, else assign "0"

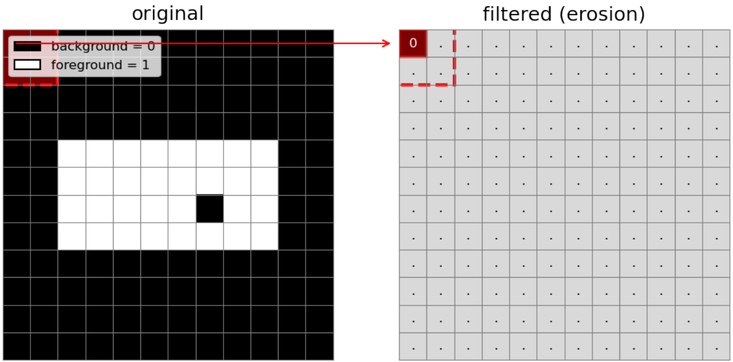


⇒ *background objects enlarge*

Primitive Morphological Operations

2. Erosion

⇒ the structuring element *slides* across the entire image



Note: this slide contains an animation, please view the PDF with a compatible viewer

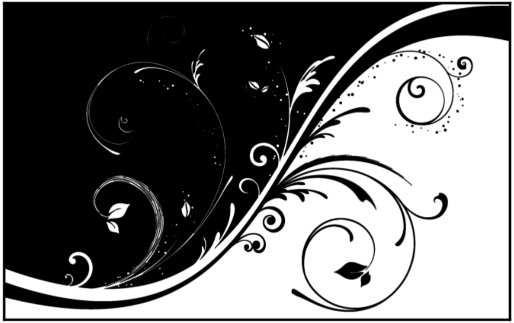
Primitive Morphological Operations

2. Erosion

$\Rightarrow$ ***size** of the structuring element* b determines the effect of the erosion

original

erosion (b=**3x3**)

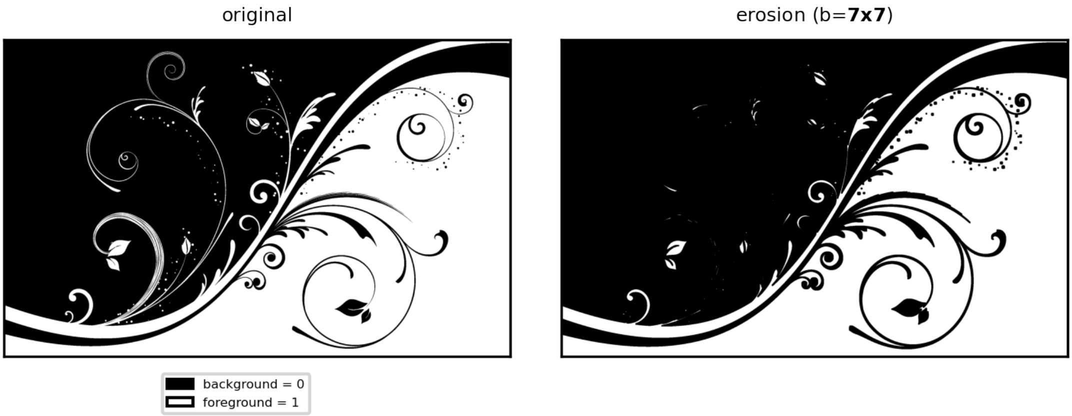


background = 0
foreground = 1

Primitive Morphological Operations

2. **Erosion**

⇒ ***size** of the structuring element* b determines the effect of the erosion



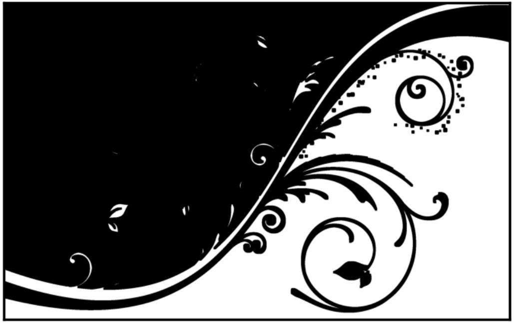
Primitive Morphological Operations

2. Erosion

$\Rightarrow$ ***size** of the structuring element* b determines the effect of the erosion

original

erosion (b=**11x11**)



background = 0
foreground = 1

$\rightarrow$ **Primitive morphological operations** (**dilation** & **erosion**) results are "coarse"

$\Rightarrow$ **Composite morphological operations** are useful to avoid some pitfalls:

- **closing**: $F \bullet b = (F \oplus b) \ominus b$ $\rightarrow$ concatenation of *dilation* and *erosion*
- **opening**: $F \circ b = (F \ominus b) \oplus b$ $\rightarrow$ concatenation of *erosion* and *dilation*

Composite Morphological Operations

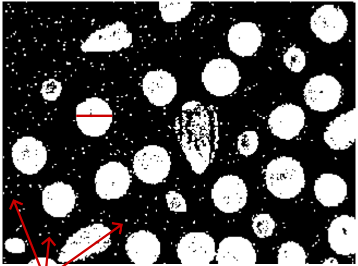
1. Opening

Problem: erosion removes unwanted small foreground objects, BUT other foreground objects shrink

Solution: after erosion, apply dilation with the same structuring element ⇒ opening

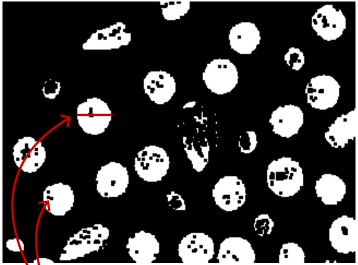
$$G = F \circ b = (F \ominus b) \oplus b$$

original



unwanted small foreground object

EROSION (b=3x3)



☑ small foreground objects removed

☒ foreground object shrunk, holes created

Composite Morphological Operations

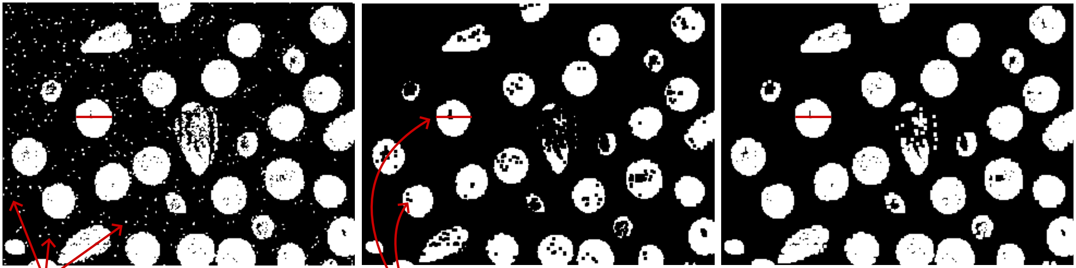
1. Opening

Problem: erosion removes unwanted small foreground objects, BUT other foreground objects shrink

Solution: after erosion, apply dilation with the same structuring element ⇒ **opening**

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

original | **EROSION** (b=3x3) | **OPENING** (b=3x3)



unwanted small foreground object

☑ small foreground objects removed
☒ foreground object shrunk, holes created

☑ small foreground objects removed
☒ holes and object shrinking reduced

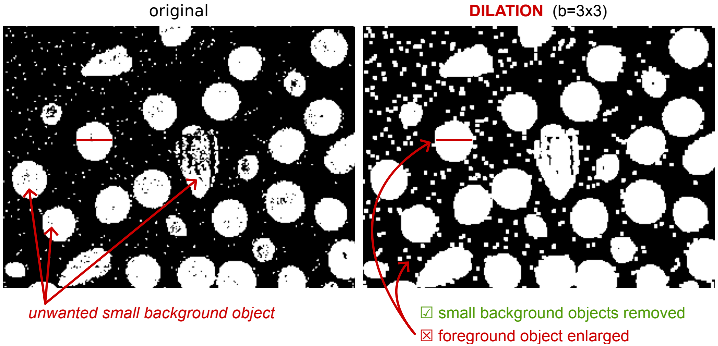
Composite Morphological Operations

2. Closing

Problem: dilation closes small background objects (holes), BUT foreground objects get enlarged

Solution: after dilation, apply erosion with the same structuring element $\Rightarrow$ closing

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



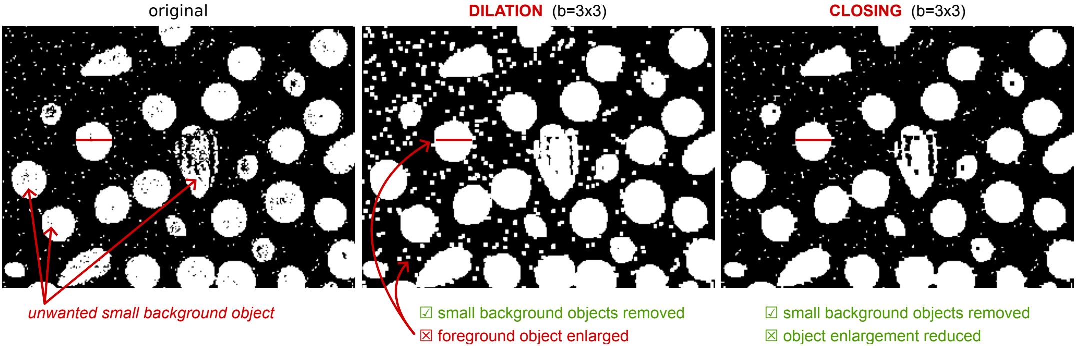
Composite Morphological Operations

2. Closing

Problem: dilation closes small background objects (holes), BUT foreground objects get enlarged

Solution: after dilation, apply erosion with the same structuring element ⇒ **closing**

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



3. Rank filters

<u>Rank filters</u> = *nonlinear spatial filters* whose response is based on *<u>ordering</u>* (*<u>ranking</u>*) the pixels contained in the region encompassed by the neighborhood **b**, and replacing the value of the center pixel with the value determined by the ranking result *(Gonzalez and Woods, 2018)*

⇒ Rank filters are a generalization of flat dilation/erosion: in lieu of min or max value in window, use the p-th ranked value

- → get *<u>minimum</u>* value in the neighborhood (0th percentile) ⇔ *<u>erosion</u>*
  - ⇒ *useful for finding the darkest points in an image or for eroding light regions adjacent to dark areas*
- → get *<u>maximum</u>* value in the neighborhood (100th percentile) ⇔ *<u>dilation</u>*
  - ⇒ *useful for finding the brightest points in an image or for eroding dark regions adjacent to bright areas*
- → get *<u>median</u>* value in the neighborhood (50th percentile)
  - ⇒ *very effective for salt-and-pepper noise reduction*

3. Rank filters

Rank filters = *nonlinear spatial filters* whose response is based on *ordering* (*ranking*) the pixels contained in the region encompassed by the neighborhood **b**, and replacing the value of the center pixel with the value determined by the ranking result *(Gonzalez and Woods, 2018)*

⇒ Rank filters are a generalization of flat dilation/erosion: in lieu of min or max value in window, use the p-th ranked value

- → get *minimum* value in the neighborhood (0th percentile) ⇔ *erosion*
  - ⇒ *useful for finding the darkest points in an image or for eroding light regions adjacent to dark areas*
- → get *maximum* value in the neighborhood (100th percentile) ⇔ *dilation*
  - ⇒ *useful for finding the brightest points in an image or for eroding dark regions adjacent to bright areas*
- → get *median* value in the neighborhood (50th percentile)
  - ⇒ *very effective for salt-and-pepper noise reduction*

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

original image

$\longrightarrow$

segmented image

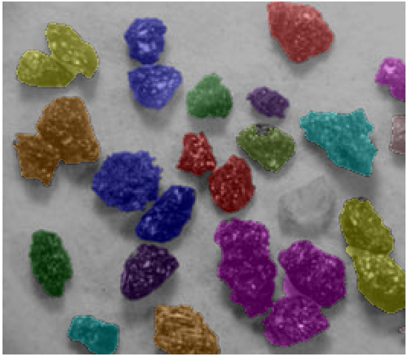


Image segmentation = labeling image pixels to partition an image into regions

- 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 region properties
  - ex: Watershed transform
  - ex: Random Walker
  - ex: Flood Fill
- Many other!
  - ex: Graph-cuts, 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 region properties
  ex: Watershed transform
  ex: Random Walker
  ex: Flood Fill

- Many other!
  ex: Graph-cuts, 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 region properties
  ex: Watershed transform
  ex: Random Walker
  ex: Flood Fill

- Many other!
  ex: Graph-cuts, 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

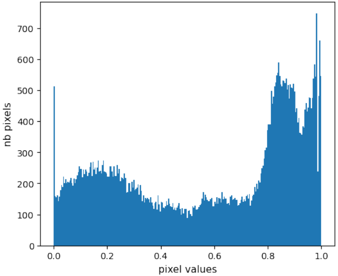
- Edge-based segmentation
  $\Rightarrow$ based on local contrast $\rightarrow$ uses gradients rather than the grey values

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

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

Histogram-based segmentation

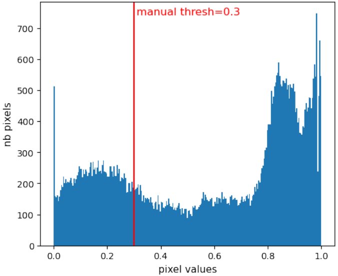
$\Rightarrow$ based on thresholding pixel values



Histogram-based segmentation

$\Rightarrow$ based on thresholding pixel values

- global thresholding
  - manual

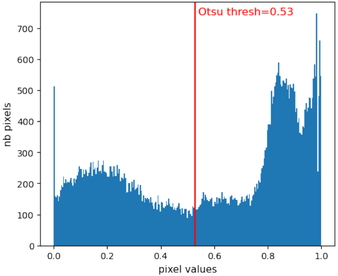


manual threshold (thresh=0.3)

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)



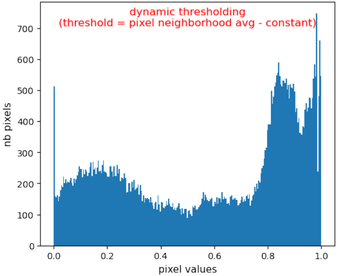
manual threshold (thresh=0.3)

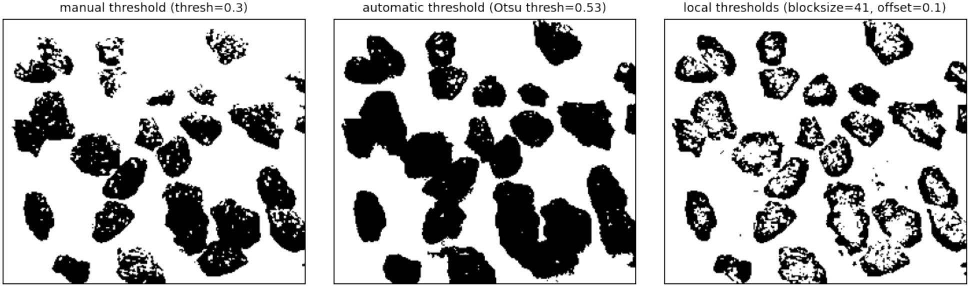
automatic threshold (Otsu thresh=0.53)

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)

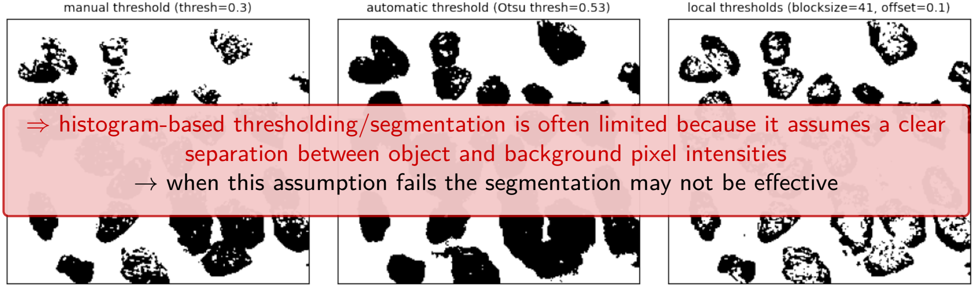




Histogram-based segmentation

$\Rightarrow$ 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)

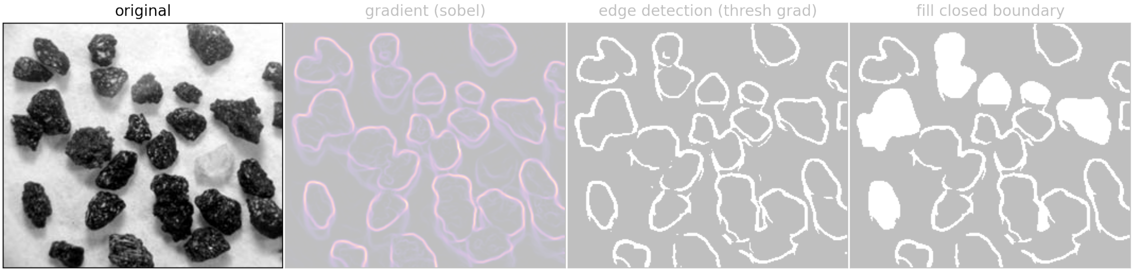


$\Rightarrow$ histogram-based thresholding/segmentation is often limited because it assumes a clear separation between object and background pixel intensities

$\rightarrow$ when this assumption fails the segmentation may not be effective

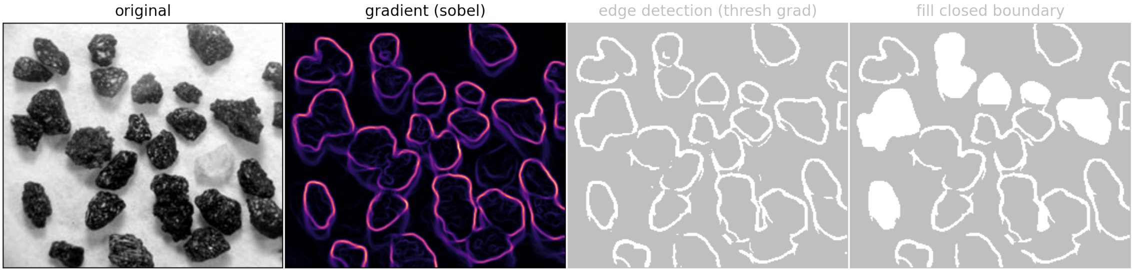
Edge-based segmentation

⇒ based on image gradients



Edge-based segmentation

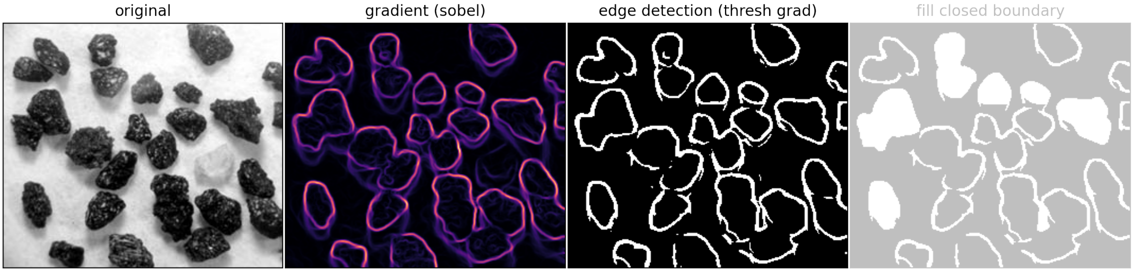
$\Rightarrow$ based on image gradients



1. compute image gradient magnitude using Sobel filter

Edge-based segmentation

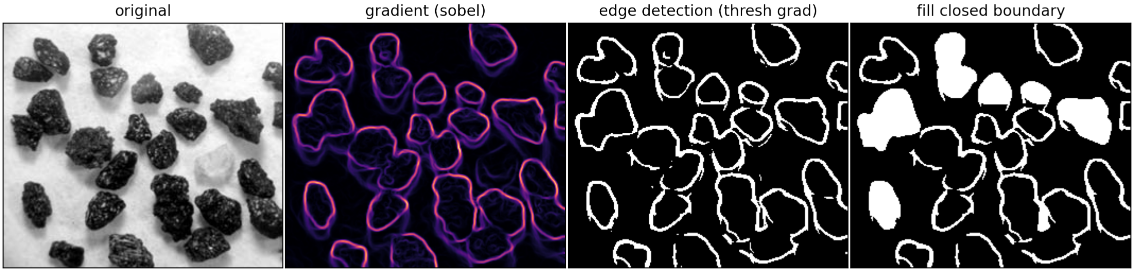
$\Rightarrow$ based on image gradients



1. compute image gradient magnitude using Sobel filter
2. threshold gradient magnitude to obtain edge map

Edge-based segmentation

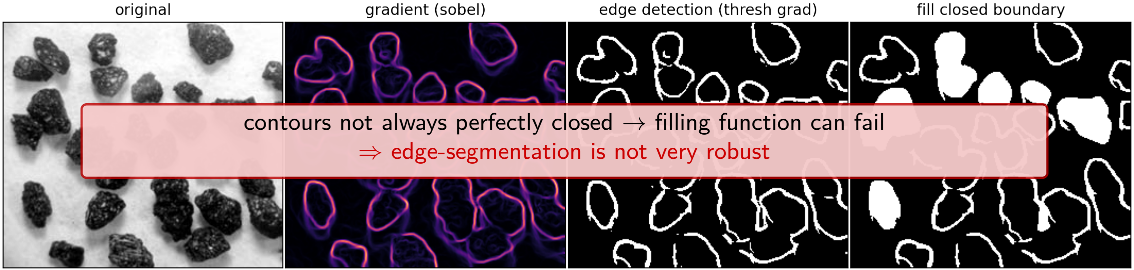
$\Rightarrow$ based on image gradients



1. compute image gradient magnitude using Sobel filter
2. threshold gradient magnitude to obtain edge map
3. apply mathematical morphology to fill inner part of the coins and remove objects smaller than a threshold

Edge-based segmentation

⇒ based on image gradients



contours not always perfectly closed → filling function can fail
⇒ edge-segmentation is not very robust

1. compute image gradient magnitude using Sobel filter
2. threshold gradient magnitude to obtain edge map
3. apply mathematical morphology to fill inner part of the coins and remove objects smaller than a threshold

Region-based segmentation

$\Rightarrow$ accounts for region properties (pixel-neighborhood)

Popular algorithms:

- Watershed transform
- Flood Fill
- Random Walker

Region-based segmentation

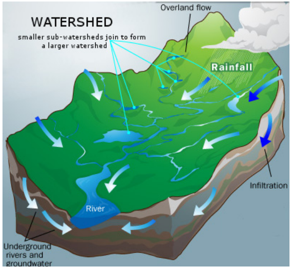
⇒ accounts for region properties (pixel-neighborhood)

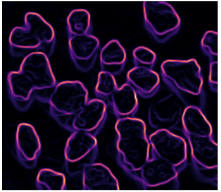
Watershed transform:

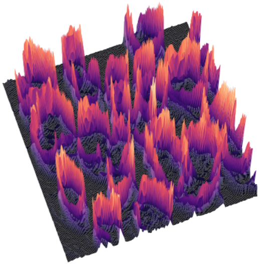
⇒ region-growing approach that fills "basins" in the image

⇒ 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



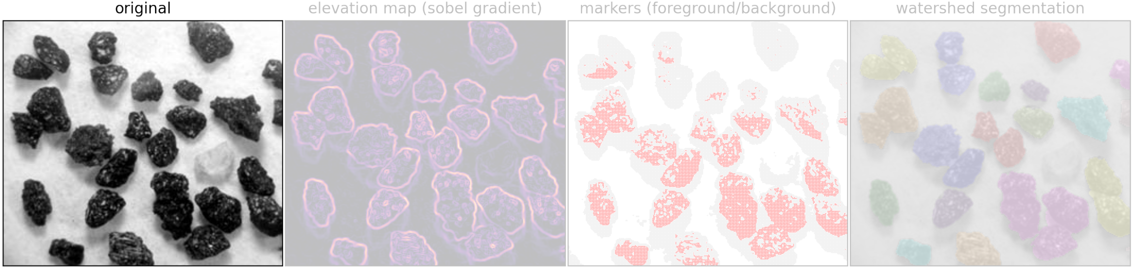




Region-based segmentation

⇒ accounts for region properties (pixel-neighborhood)

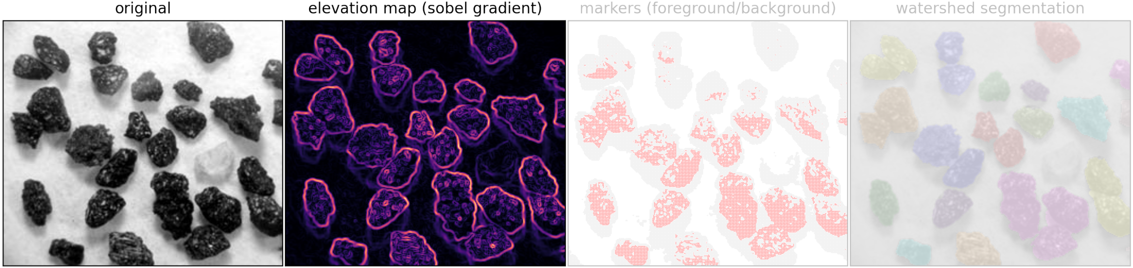
⇒ **Watershed transform**:



Region-based segmentation

⇒ accounts for region properties (pixel-neighborhood)

⇒ **Watershed transform**:

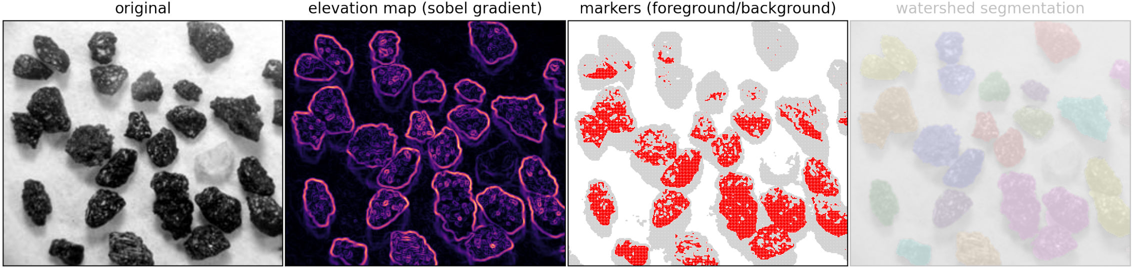


1. build "elevation map" from image gradient amplitude (using the Sobel operator)

Region-based segmentation

⇒ accounts for region properties (pixel-neighborhood)

⇒ **Watershed transform**:

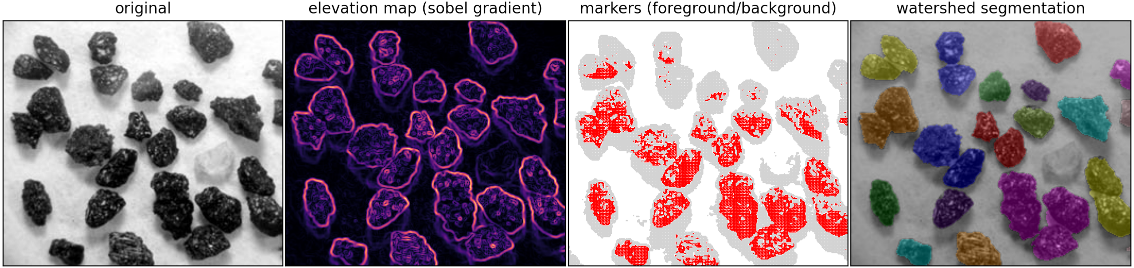


1. build "elevation map" from image gradient amplitude (using the Sobel operator)
2. define markers for background (red) & foreground (white) (here based on the extreme parts of the histogram)

Region-based segmentation

⇒ accounts for region properties (pixel-neighborhood)

⇒ **Watershed transform**:

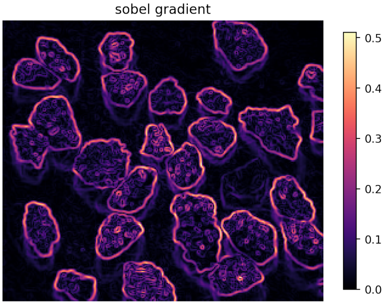


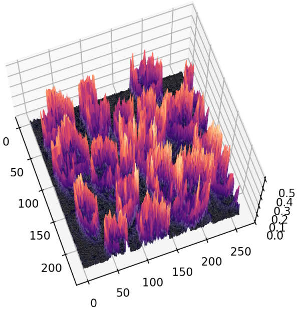
1. build “elevation map” from image gradient amplitude (using the Sobel operator)
2. define markers for background (red) & foreground (white) (here based on the extreme parts of the histogram)
3. apply watershed transform (and colorize segmented elements)

Region-based segmentation

⇒ accounts for region properties (pixel-neighborhood)

⇒ **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

Image Segmentation

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.)

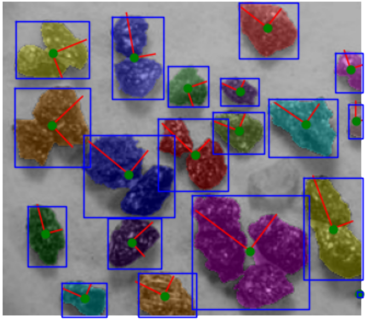
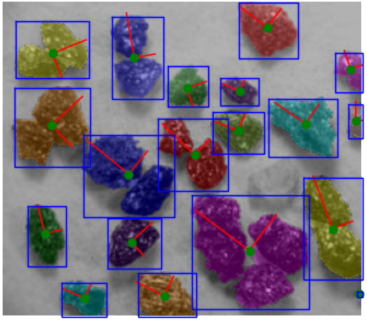


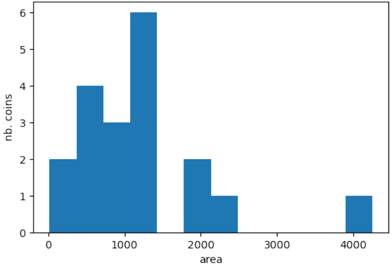
Image Segmentation

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.)





Exercises

Exercise:

$\Rightarrow$ analyze a thermal infrared image of a lava lake

$\rightarrow$ segment the crustal plates from the incandescent cracks and analyze