Basic Image Processing

PPKE-ITK

Lecture 7.

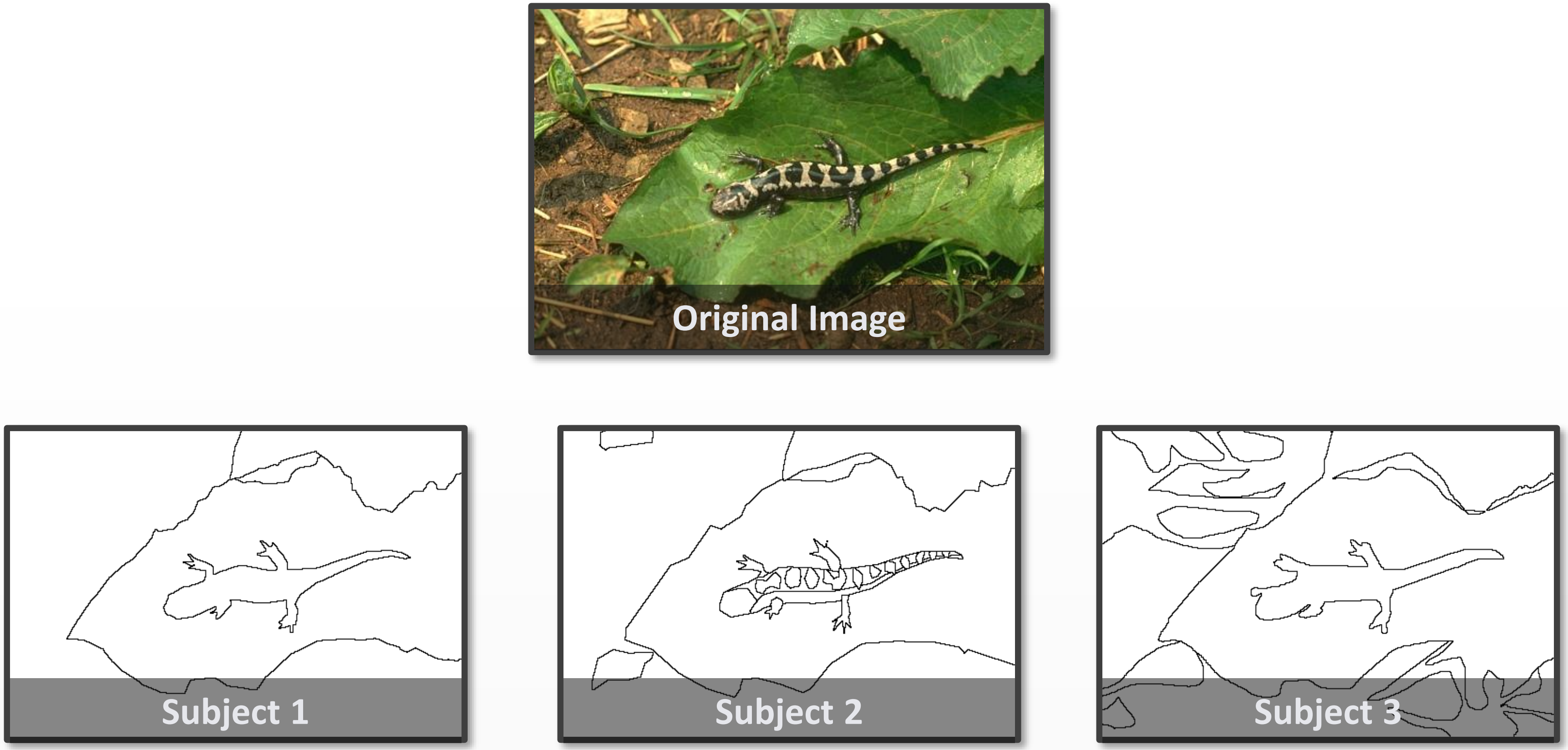
Image Segmentation

Image Segmentation

- What is on the image? – This is maybe the most important question we want to answer about an image.
- For a human observer it is a trivial task, for a machine it is still an unsolved problem.
- An important step toward our goal is to segment the image into meaningful parts.
- The objective is to group pixels together based on some common characteristics:
  - they belong to the same physical object
  - they have the same intensity level/color/texture
  - they belong to the background/foreground
  - …

Image Segmentation

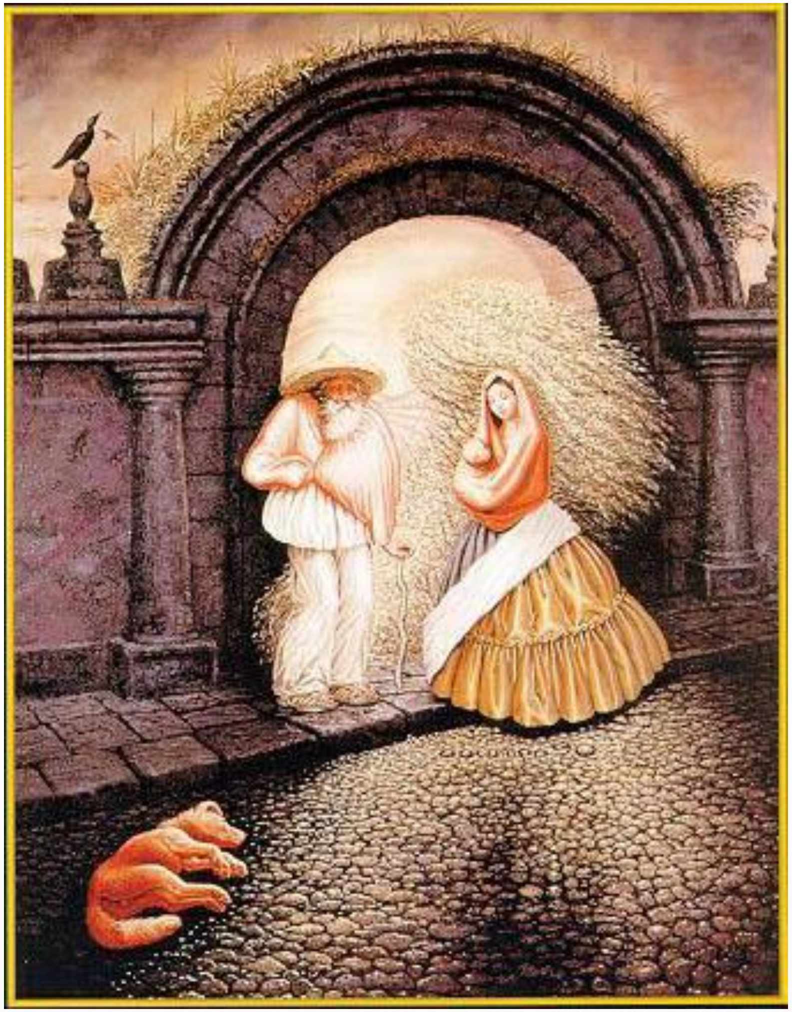
- Sometimes even humans cannot agree on a unique solution!



Sample from BSDS500 (Berkeley Segmentation Data Set and Benchmarks 500):
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html

Gestalt grouping

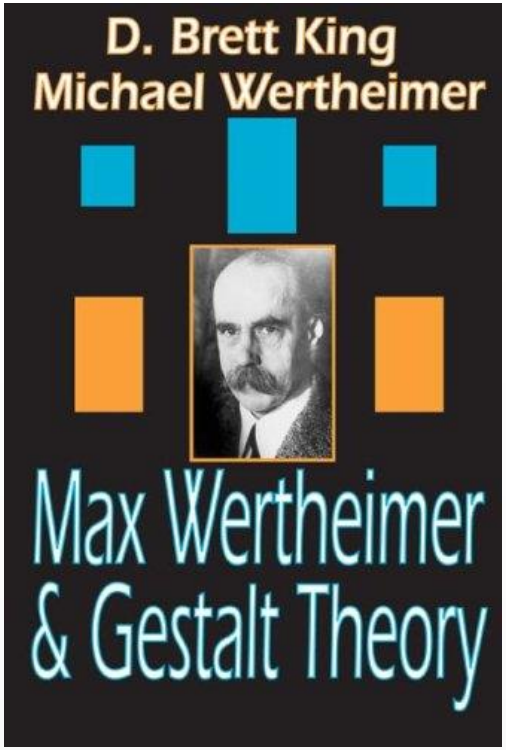
- Gestalt definition: a configuration or pattern of elements so unified as a whole that it cannot be described merely as a sum of its parts



Gestalt psychology or gestaltism

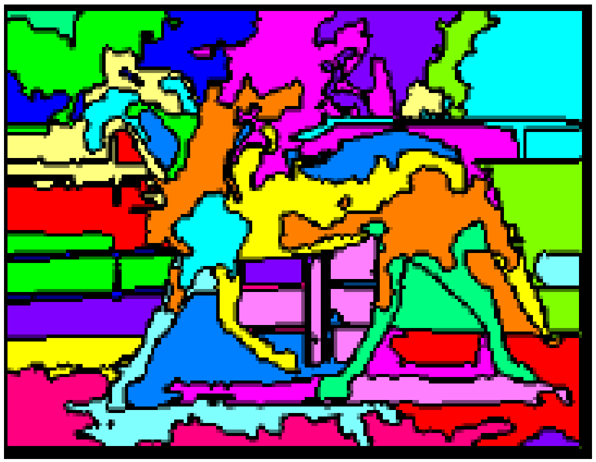
- German: *Gestalt* - "form" or "whole”
  - Berlin School, early 20th century
  - Kurt Koffka, Max Wertheimer, and Wolfgang Köhler

- View of brain:
  - whole is more than the sum of its parts
  - holistic
  - parallel
  - analog
  - self-organizing tendencies

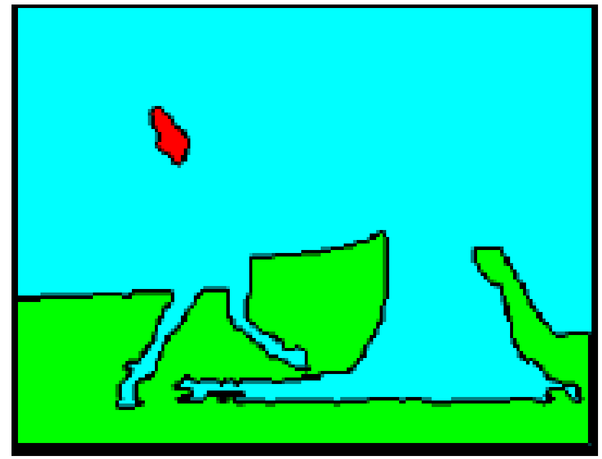


Slide from S. Saverese

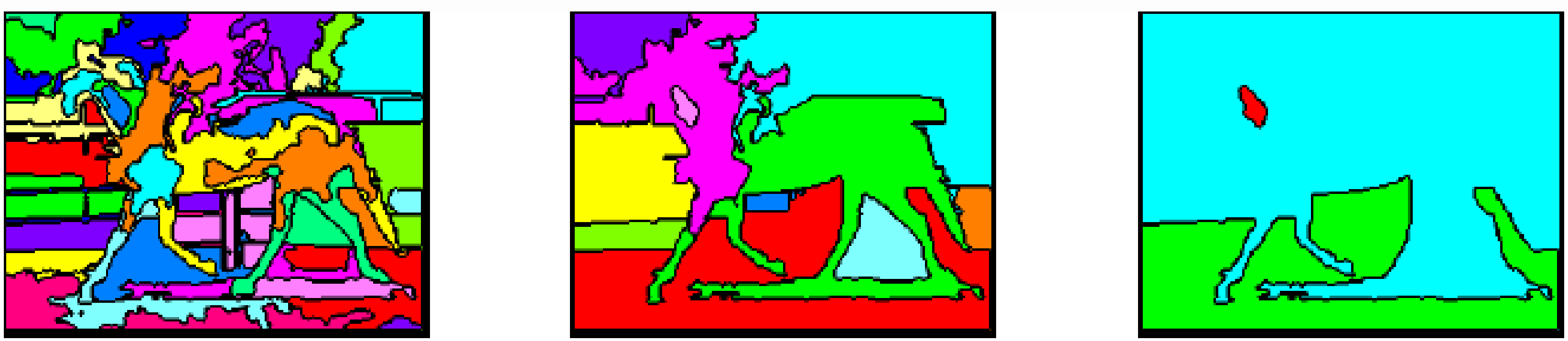
Types of segmentations



Oversegmentation



Undersegmentation



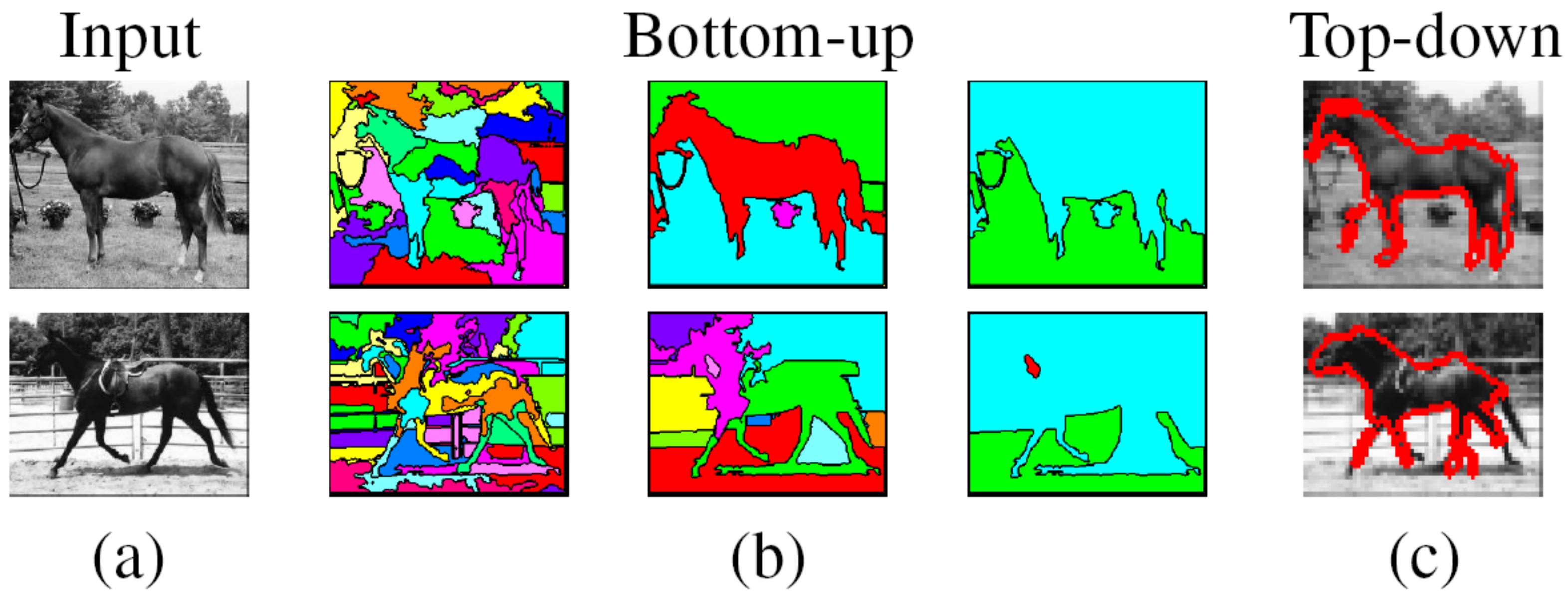
Multiple Segmentations

Image Segmentation

- The segmentation can be **knowledge-driven** (top-down) or **data-driven** (bottom-up).
- Knowledge driven segmentation methods builds prior knowledge into the segmentation algorithm:
  - Hard to implement
  - Cannot stand alone: need cues from bottom-up segmentation
- Data-driven methods builds on the raw pixel data:
  - they are easier to implement
  - they often fail on real life images
- There is the so-called **semantic gap** between the two approach.
- The complex, high level definitions of top-down methods are hard to embed efficiently into low level algorithms.

Major processes for segmentation

- Bottom-up: group tokens with similar features
- Top-down: group tokens that likely belong to the same object



[Levin and Weiss 2006]

K-means clustering using intensity alone and color alone

Image

Clusters on intensity

Clusters on color



Image Segmentation

- Intensity Level Based Segmentation
  - Otsu’s Method
- Region-based Segmentation
  - Region growing
  - Region Splitting and Merging
- Clustering in the Feature Space

Intensity Level Based Segmentation

- Thresholding
  - **Assumption**: the image parts (e.g. object and background) can be separated based on their intensity level.

$$s(n_1, n_2) = \begin{cases} \text{object} & x(n_1, n_2) < T \\ \text{background} & x(n_1, n_2) \geq T \end{cases}$$

…where $s(n_1,n_2)$ is the cluster of the (n_1,n_2) pixel of the x image and T is a threshold.

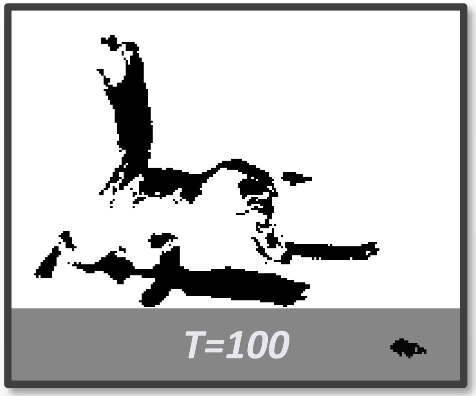
  - The main question is how to determine the threshold?

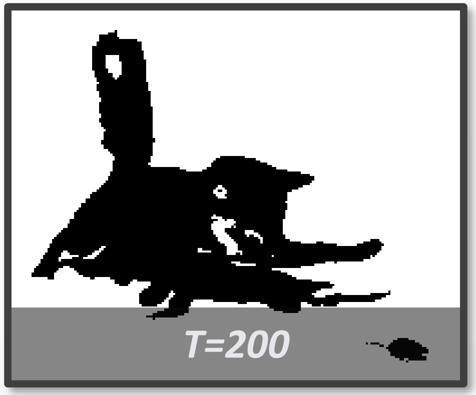
Intensity Level Based Segmentation

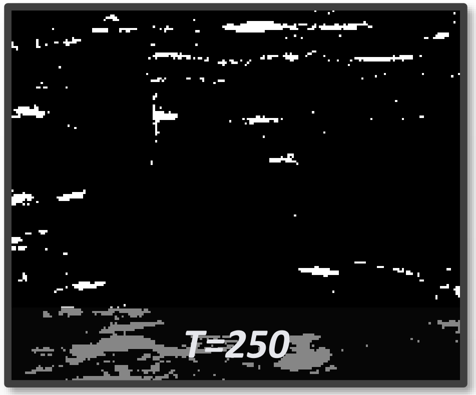
- Thresholding:
  - The main question is how to find the optimal threshold?


Original Image


Grayscale Image


T=100


T=200


T=250

Intensity Level Based Segmentation

- Otsu's method:
  - Automatically determines the optimal global threshold by minimizing the intra-class variance.
  - The intra-class variance is defined as follows:

$$\sigma_w^2(k) = \omega_1(k)\sigma_1^2(k) + \omega_2(k)\sigma_2^2(k)$$

where ω_i and σ_i are the probability and the variance of the two classes separated by the threshold k.

  - Otsu showed that ***minimizing the intra-class variance is the same as maximizing inter-class variance***:

$$\sigma_b^2(k) = \sigma^2 - \sigma_w^2(k) = \omega_1(k)\omega_2(k)(\mu_1(k) - \mu_2(k))^2$$

where μ_i are the means of the two classes separated by threshold k.

Nobuyuki Otsu (1979). "A threshold selection method from gray-level histograms". *IEEE Trans. Sys., Man., Cyber.* **9** (1): 62–66.

Intensity Level Based Segmentation

- Otsu's method:

$$\sigma_b^2(k) = \sigma^2 - \sigma_w^2(k) = \omega_1(k)\omega_2(k)\left(\mu_1(k) - \mu_2(k)\right)^2$$

  - To calculate ω_i and μ_i the normalized histogram of the image is used:

$$\omega_1(k) = \sum_{i=0}^{k} p_i \qquad \omega_2(k) = \sum_{i=k+1}^{L-1} p_i$$

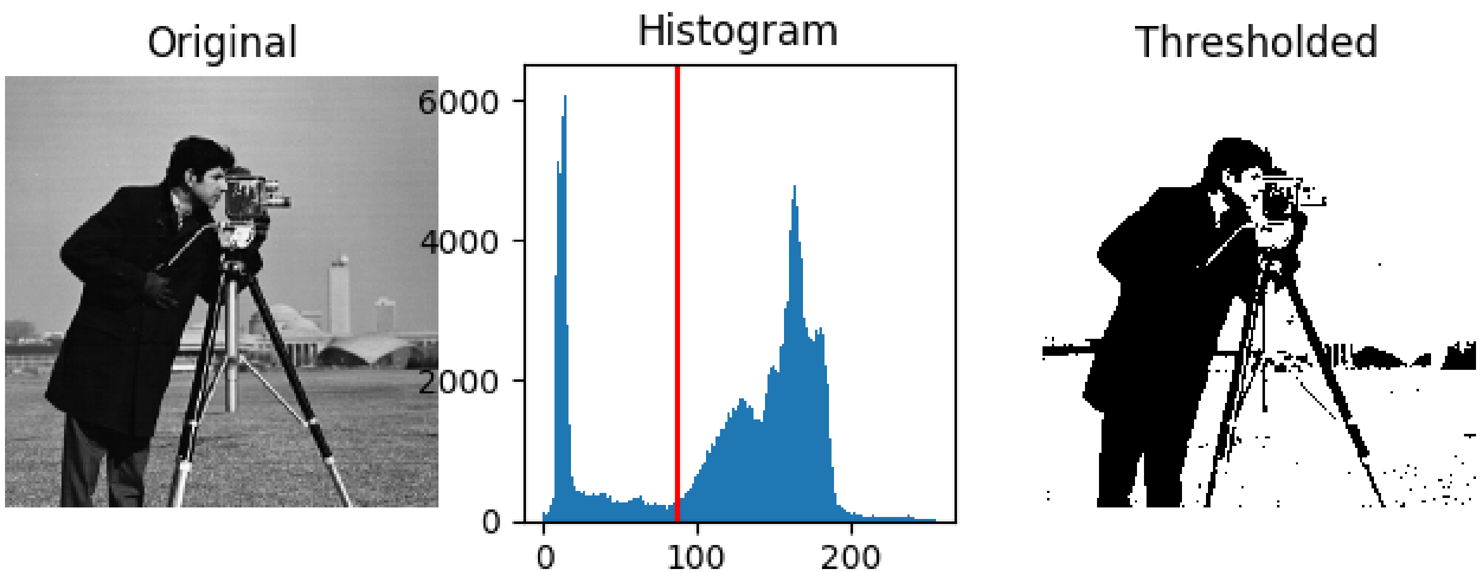
$$\mu_1(k) = \left(\sum_{i=0}^{k} i p_i\right) \Big/ \omega_1 \qquad \mu_2(k) = \left(\sum_{i=k+1}^{L-1} i p_i\right) \Big/ \omega_2$$

where p_i is the i-th entry in the normalized histogram of the image (probability of the i-th intensity level).

The Otsu threshold is the value that maximizes the inter-class variance.

* Nobuyuki Otsu (1979). "A threshold selection method from gray-level histograms". *IEEE Trans. Sys., Man., Cyber.* **9** (1): 62–66.

Result of Otsu's method



Result of Otsu's method

- Otsu's method:

Region-Based Segmentation Methods

- Let R be the entire image region, and $R_1, \dots, R_n$ are subregions.
- We want to find a segmentation that is..
  - Complete: $\bigcup_{i=1}^{n} R_i = R$
  - Points in the region R_i $(i = 1, \dots, n)$ are connected
  - The regions are disjoint: $R_i \cap R_j = \emptyset$ for $\forall i \neq j$
  - All the pixels in a region has common properties…
  - …that they don't share with pixels from other regions.

Region-Based Segmentation Methods

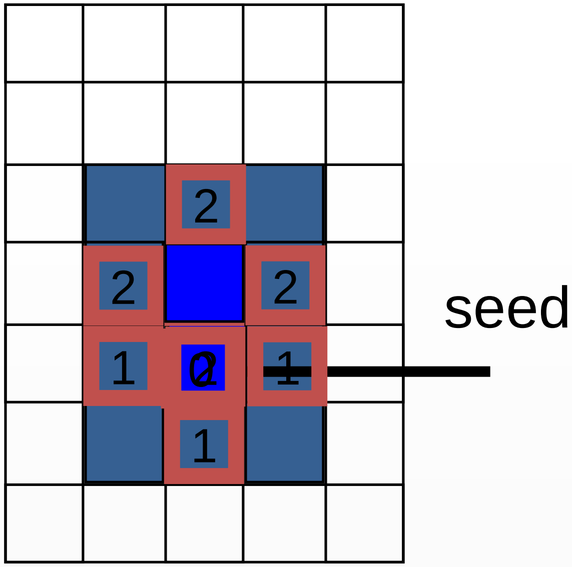
- Region growing:
  - The method is initialized with a set of **seed points** as regions
  - We start growing the regions by adding neighboring pixels to the region if they has **similar predefined properties** as the seed points.

  - The seeds can be selected based on prior information, or evenly, or random...
  - The similarity criteria is usually depending on the segmentation result we want. (Commonly used properties are the intensity level, color, texture, motion,…)

  - **Pros:** simple, works well on images with clear edges, prior knowledge can be easily utilized, robust to noise…
  - **Cons:** time consuming

Region growing

- We start growing the regions by adding neighboring pixels to the region if they has **similar predefined properties** as the seed points

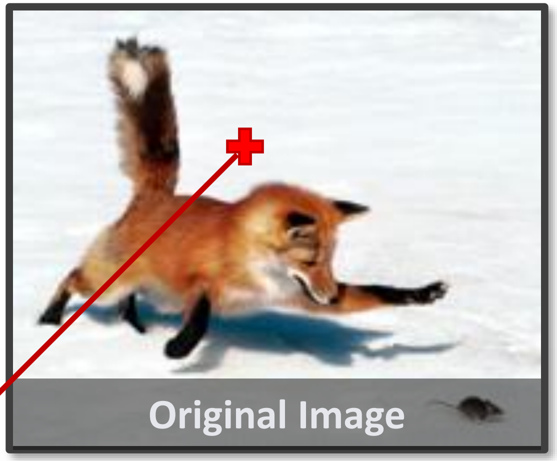



```
Flood(x, y) {
        if (pixel[x][y] is internal point ) {
                addToRegion(x, y);
                Flood(x, y-1);
                Flood(x, y+1);
                Flood(x-1, y);
                Flood(x+1, y);
        }
}
```

Internal point= dark blue

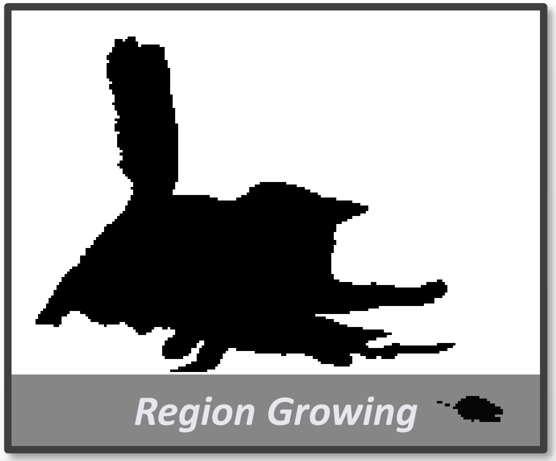
Region-Based Segmentation Methods

- Region growing:


Original Image

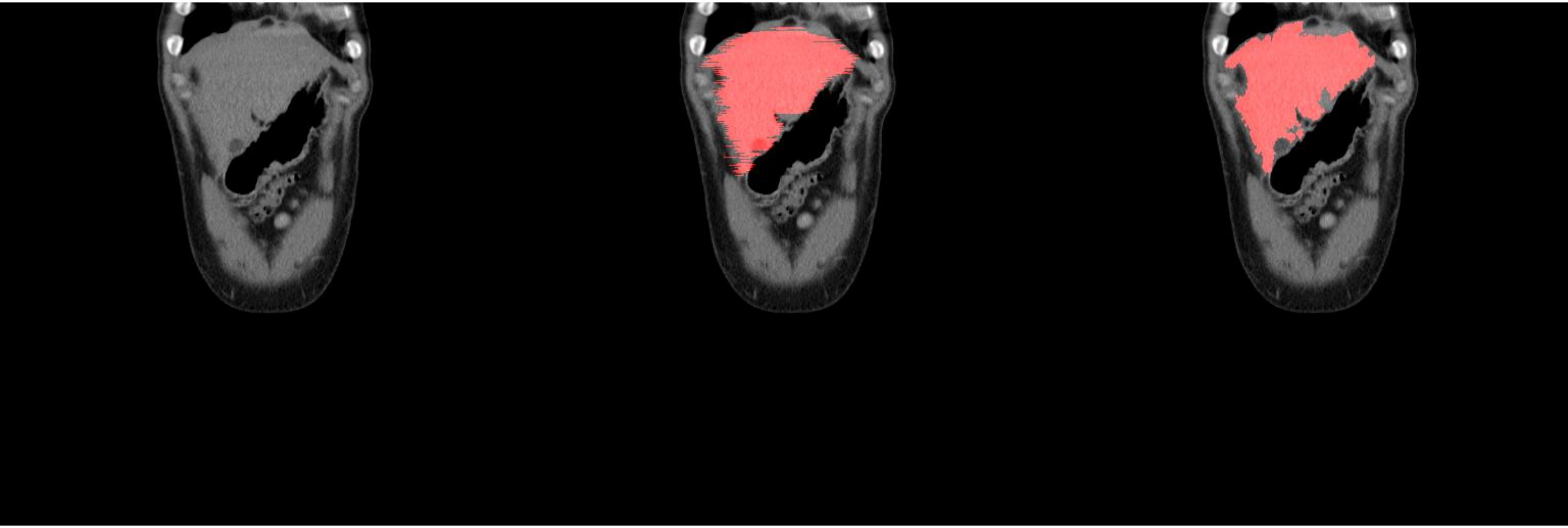

Grayscale Image

seed point


Region Growing

Region growing results on medical data

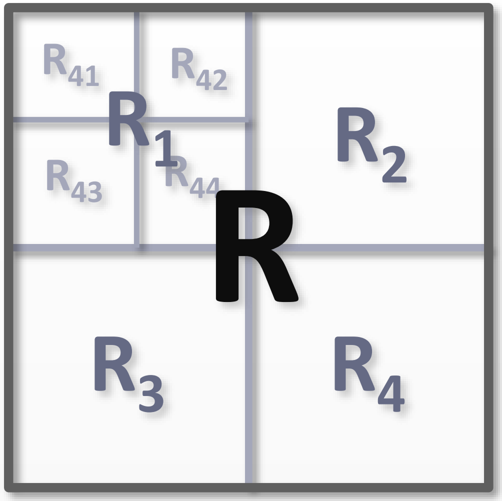
- Liver segmentation from CT

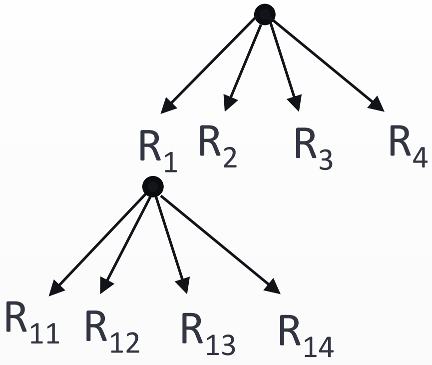


Manual Ground Truth Region growing result

Region-Based Segmentation Methods

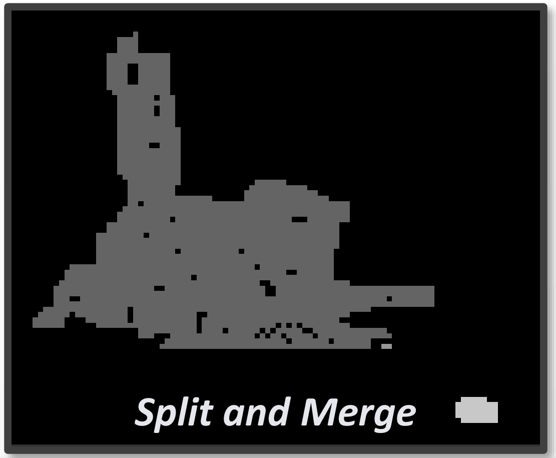
- Region splitting and merging:
  - Let ***R*** represent the entire image region and ***P*** be a predicate.
  - The splitting and merging steps are alternating:
    - We split the region R_i into 4 sub regions if $P(R_i)$ = false
    - We merge 2 neighboring regions R_i and R_j if $P(R_i \cup R_j)$ = true
  - The minimum region size has to be selected.





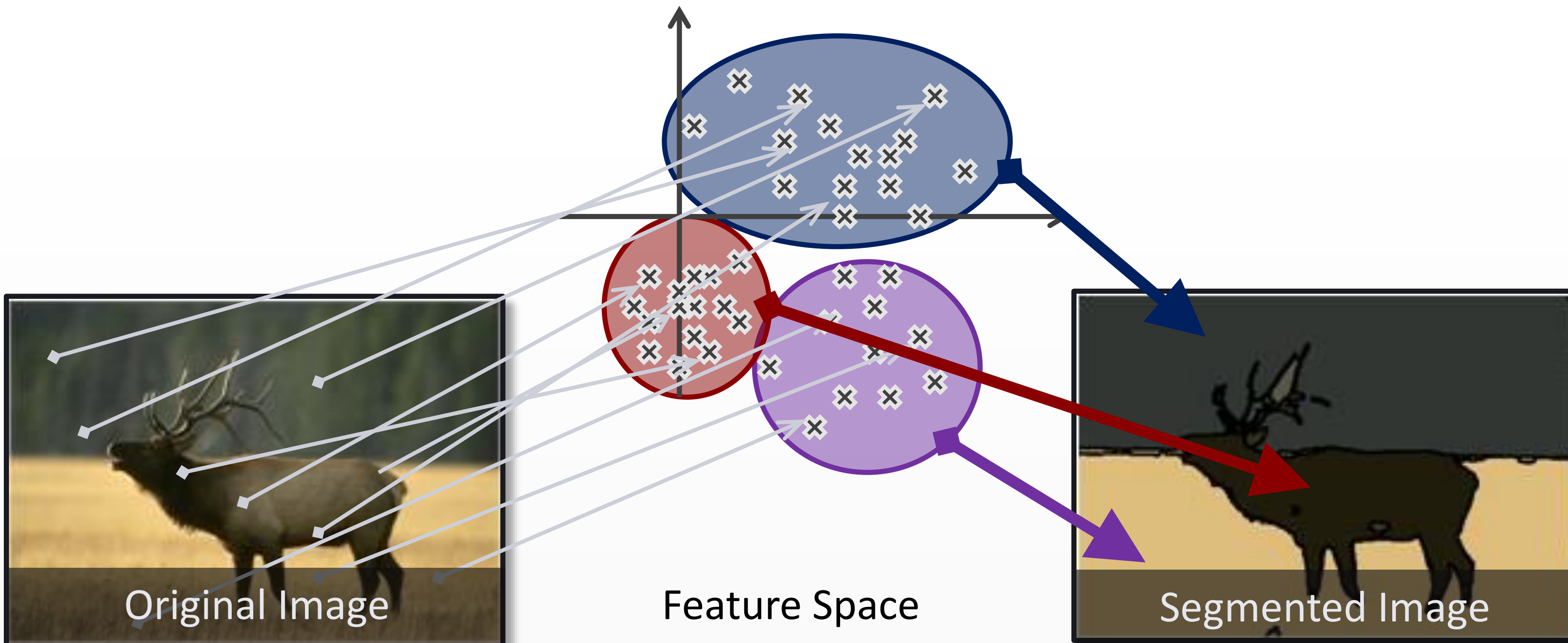
Region-Based Segmentation Methods

- Region splitting and merging:



Clustering in the Feature Space

- A clustering algorithm is used to find structure in the data.
- The pixels are represented in the feature space.
- Usual features: colors, pixel coordinates, texture descriptors,..



Source of the Images: http://ivrgwww.epfl.ch/supplementary_material/RK_CVPR09/

Clustering in the Feature Space

- Partitioning-Clustering Approach
  - Learning a partition on a data set to produce several non-empty clusters
  - Assume that the number of clusters, K, is given in advance
  - Given a *K*, find a partition of *K clusters* to optimize the chosen partitioning criterion (cost function)
  - In principle, optimal partition $S = \{S_1, \dots S_K\}$ achieved *via minimizing the sum of squared distance to its "representative object" in each cluster*

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x \in S_i} d^2(x, \mu_i)$$

  - global optimum: exhaustively search all partitions: **too expensive!**
  - a typical clustering analysis approach via **iteratively** partitioning training data set to learn a partition of the given data space

K-means Algorithm

- *K-means* algorithm (MacQueen'67): a heuristic method
  - Each cluster is represented by the centre of the cluster and the algorithm converges to stable centroids of clusters.
  - K-means algorithm is the simplest partitioning method for clustering analysis and widely used in data mining applications.
  - Each sample will belong to the cluster with the nearest mean.
- The objective is to minimize the within-cluster sum of squares:

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x \in S_i} d^2(x, \mu_i), \qquad d^2(x, \mu_i) = \|x - \mu_i\|^2 = \sum_{n=1}^{N} (x_n - \mu_{in})^2$$

  - where $x \in \mathbb{R}^N$ are the data samples, μ_i is the mean (prototype) of the points in the cluster S_i $(i = 1 \ldots K)$.

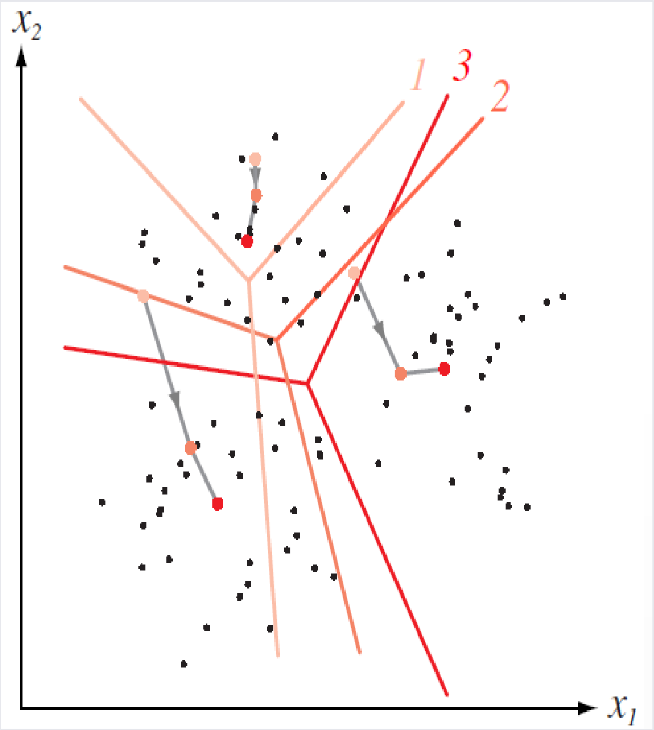
K-means Algorithm

- Given the cluster number *K*, the *K-means* algorithm is carried out in three steps after initialization:
  1) **Initialisation**: set the K cluster seed points (randomly)
  2) **Assignment step**: Assign each object to the cluster of the nearest seed point measured with a specific distance metric
  3) **Update step**: Compute new seed points as the centroids of the clusters of the current partition (the centroid is the centre, i.e., ***mean point***, of the cluster)

$$\mu_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x_j \in S_i^{(t)}} x_j$$

  4) **Go back** to Step 1), stop when no more new assignment (i.e., membership in each cluster no longer changes)

Understanding K-means

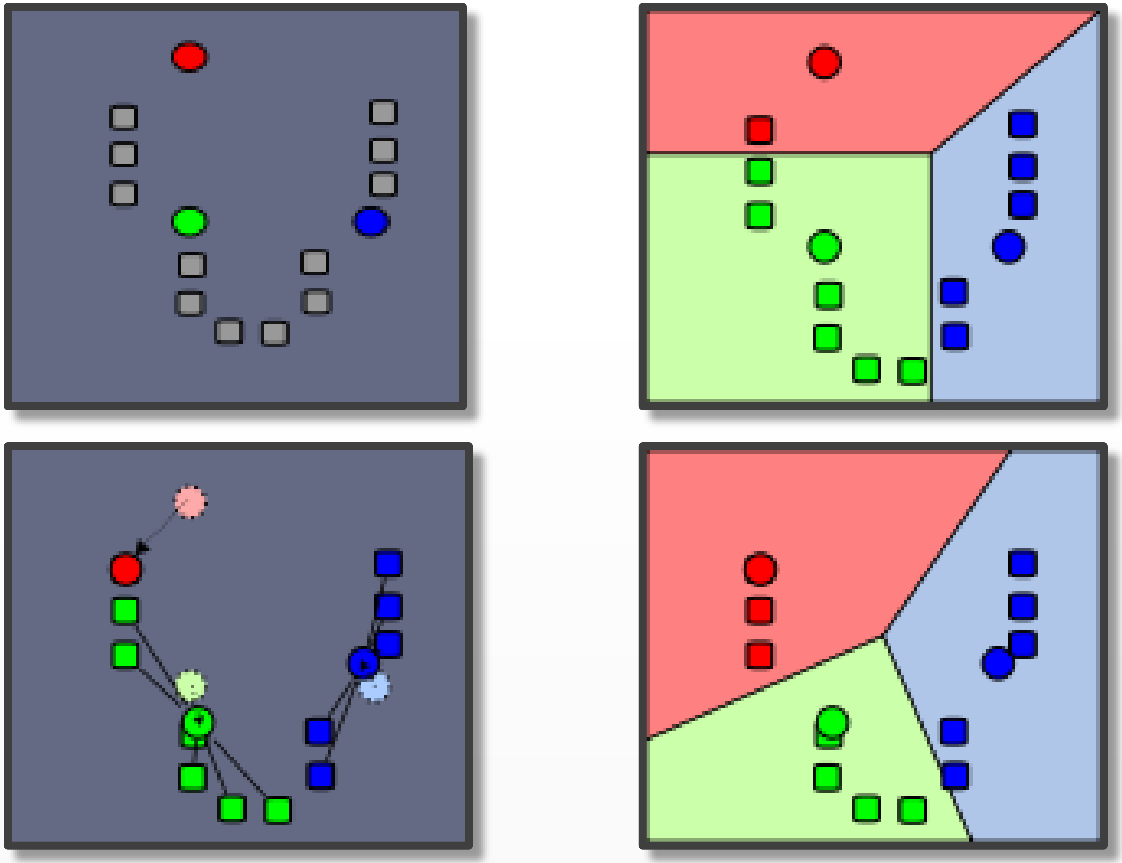


- How K-means partitions?
  - When K centroids are set/fixed, they partition the whole data space into K mutually exclusive subspaces to form a partition.
  - A partition amounts to a *Voronoi* diagram
  - Changing positions of centroids leads to a new partitioning.

- Efficient in computation ☺
  - $O(tKn)$, where n is number of objects (eg. pixels), K is number of clusters, and t is number of iterations. Normally, $K, t \ll n$.

K-Means Clustering

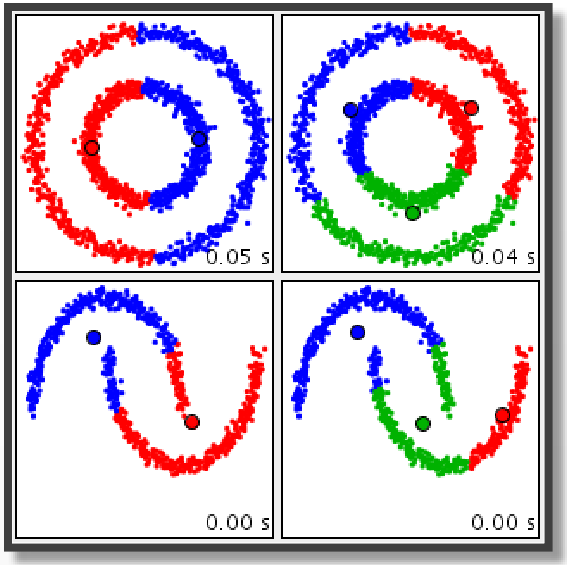
- Illustration of K-means iteration:

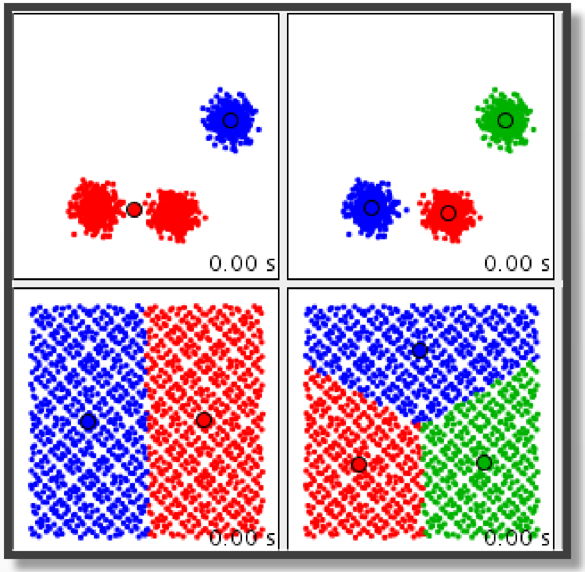


Source of the images: http://en.wikipedia.org/wiki/K-means_clustering

K-Means Clustering - Relevant Issues

- Limitation of K-means:
  - Number of clusters has to be known a priori priori (specify K in advance)
  - Sensitive to initial seed points, could stuck in a local minimum
  - Spherical clusters: not suitable for discovering clusters with non-convex shapes





Source of the images: http://commons.apache.org/proper/commons-math/userguide/ml.html

Relevant Issues

- Other issues
  - Unable to handle noisy data and outliers (K-Medoids algorithm)
  - Applicable only when mean is defined, then what about categorical data? (K-mode algorithm)
  - How to evaluate the K-mean performance?

Color-Based Image Segmentation Using K-means

- **Step 1**: Loading a color image of tissue stained with hemotoxylin and eosin (H&E)

H&E image

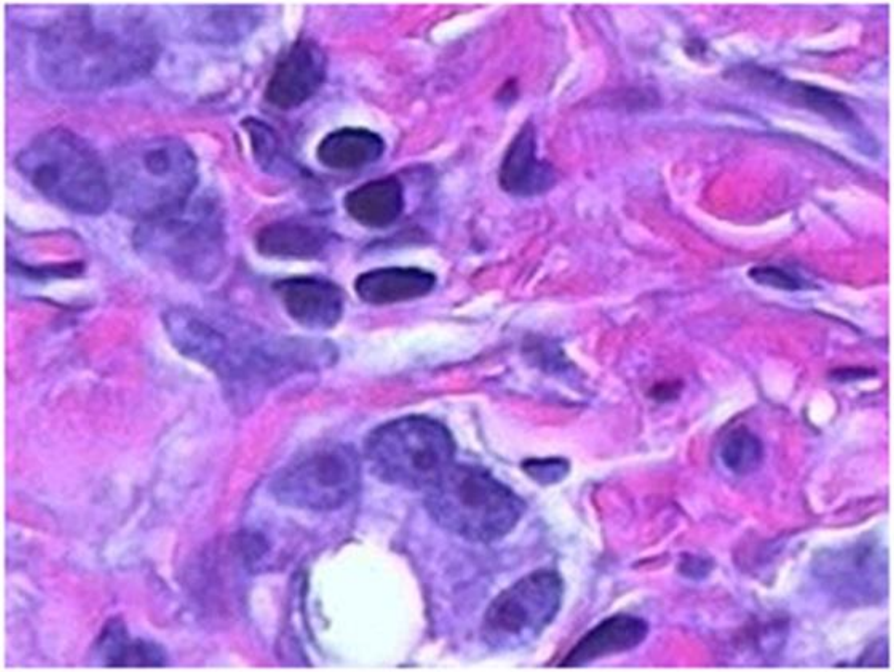
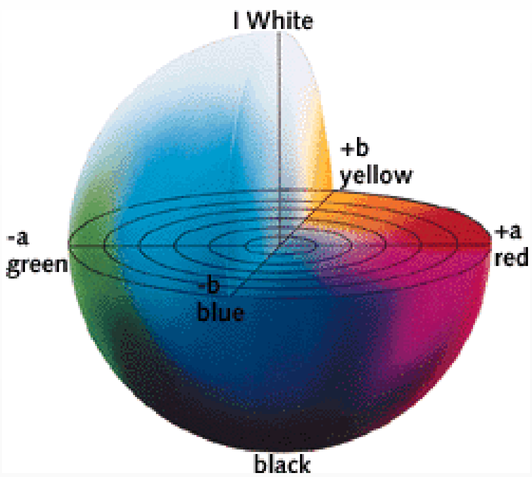


Image courtesy of Alan Partin, Johns Hopkins University

Color-Based Image Segmentation Using K-means

- **Step 2:** Convert the image from *RGB* color space to $CIE\ L^*a^*b^*$color space (ReCap from *Lecture 1*)
  - Unlike the RGB color model, $CIE\ L^*a^*b^*$ color is designed to approximate human vision: *brightness* and *color shade* components of the pixel values are encoded in different channels
  - There is a complicated transformation between *RGB* and $CIE\ L^*a^*b^*$
    - $(L^*a^*b^*)$= T*(R, G, B).*
    - *(R, G, B)* = T'$(L^*a^*b^*)$
  - The **brightness (L)** increases from the bottom to the top of the three-dimensional model.
  - **Color shades:** The a axis extends from green (-a) to red (+a) and the b axis from blue (-b) to yellow (+b).

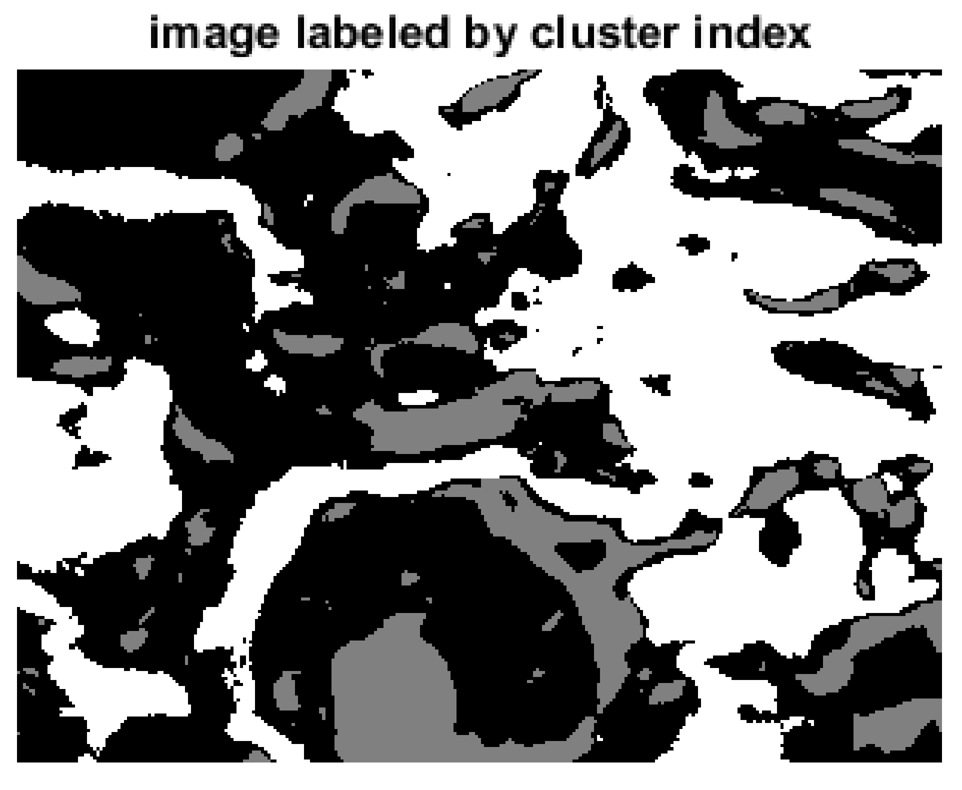


Color-Based Image Segmentation Using K-means

- **Step 3:** Undertake clustering analysis in the (a*, b*) color space with the *K-means* algorithm
  - During feature selection, L* feature is discarded. As a result, each pixel has a 2D feature vector $x = [\mathrm{a}*, \mathrm{b}*] \in \mathbb{R}^2$.
  - Applying the *K-means* algorithm to the image in the a*b* feature space where K = 3 (by applying the domain knowledge).

Color-Based Image Segmentation Using K-means

- **Step 4:** Label every pixel in the image using the results from
  - K-means clustering (indicated by three different grey levels



Color-Based Image Segmentation Using K-means

- **Step 5:** Create Images that Segment the H&E Image by Color
  - Apply the label and the color information of each pixel to achieve separate color images corresponding to three clusters.

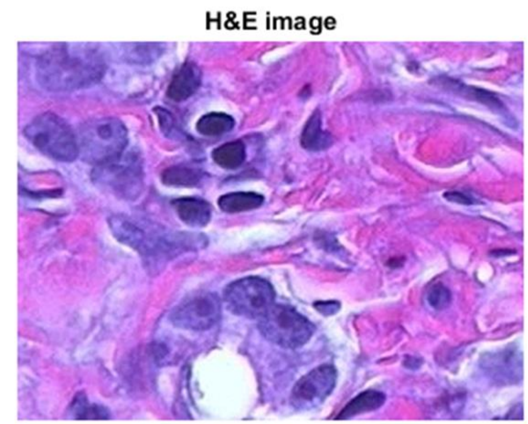
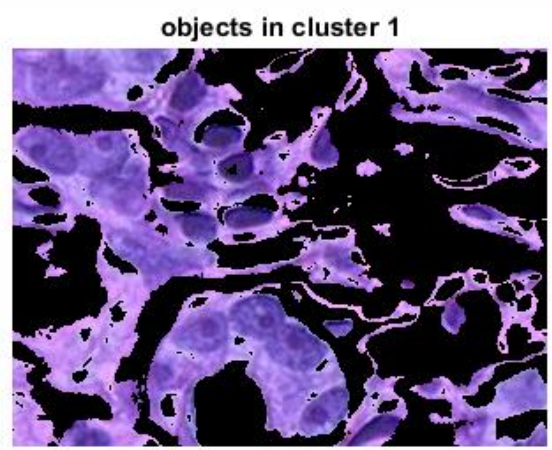
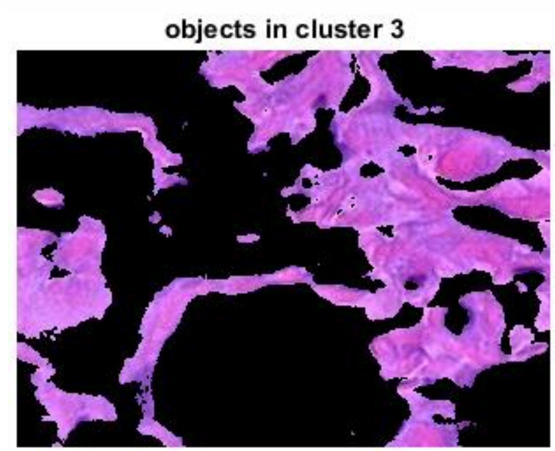


Image courtesy of Alan Partin, Johns Hopkins University



“blue” pixels

“white” pixels

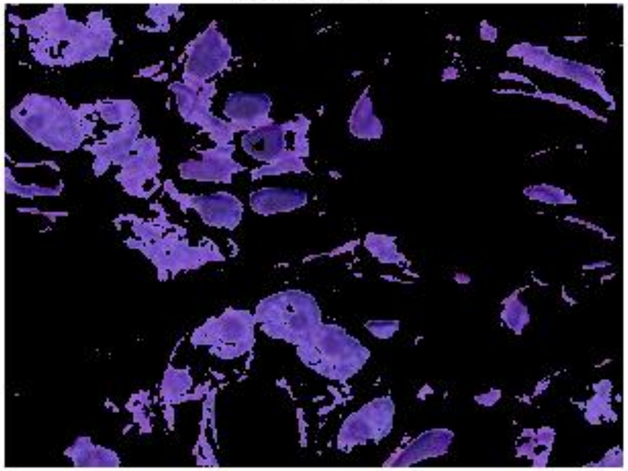


“pink” pixels

Color-Based Image Segmentation Using K-means

- **Step 6:** Segment the nuclei into a separate image with the L* feature
  - In cluster 1, there are dark and light blue objects (pixels). The dark blue objects (pixels) correspond to nuclei (with the domain knowledge).
  - L* feature specifies the brightness values of each colour.
  - With a threshold for L*, we achieve an image containing the nuclei only.

blue nuclei



Summary: K-means

- ***K*-means** algorithm is a simple yet popular method for clustering analysis
- Its performance is determined by initialisation and appropriate distance measure
- There are several **variants** of *K*-means to overcome its weaknesses
  - *K*-Medoids: resistance to **noise and/or outliers**
  - *K*-Modes: extension to **categorical data** clustering analysis
  - CLARA: extension to deal with **large data** sets
  - Mixture models (EM algorithm): handling **uncertainty** of clusters

Online tutorial: how to use **the *K*-means function in Matlab**
https://www.youtube.com/watch?v=aYzjenNNOcc

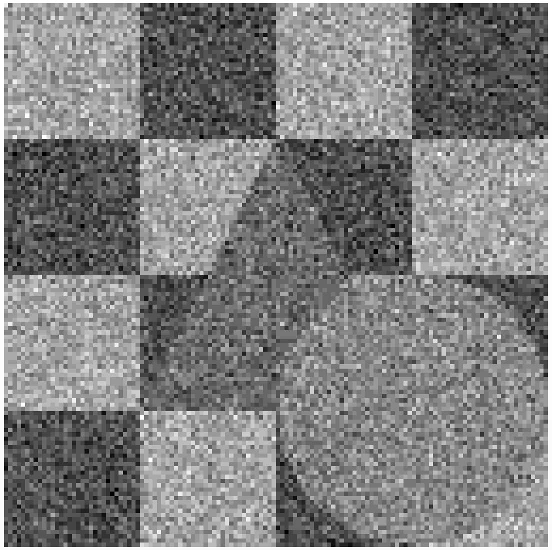
Discussed image segmentation example:
https://www.mathworks.com/help/images/examples/color-based-segmentation-using-k-means-clustering.html

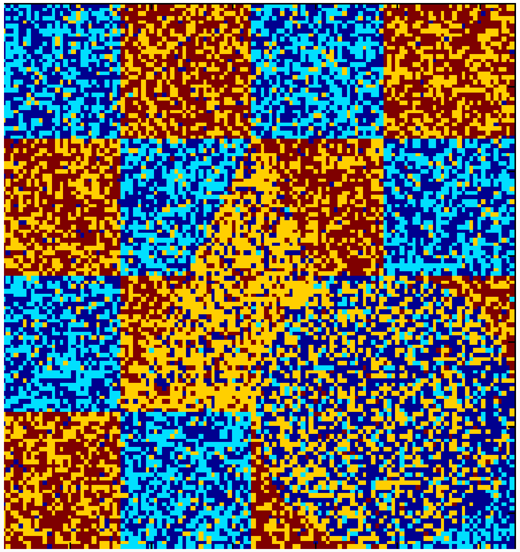
K-Means : some further results

- Segmentation in RGB color space can also work…

K-Means: some further results

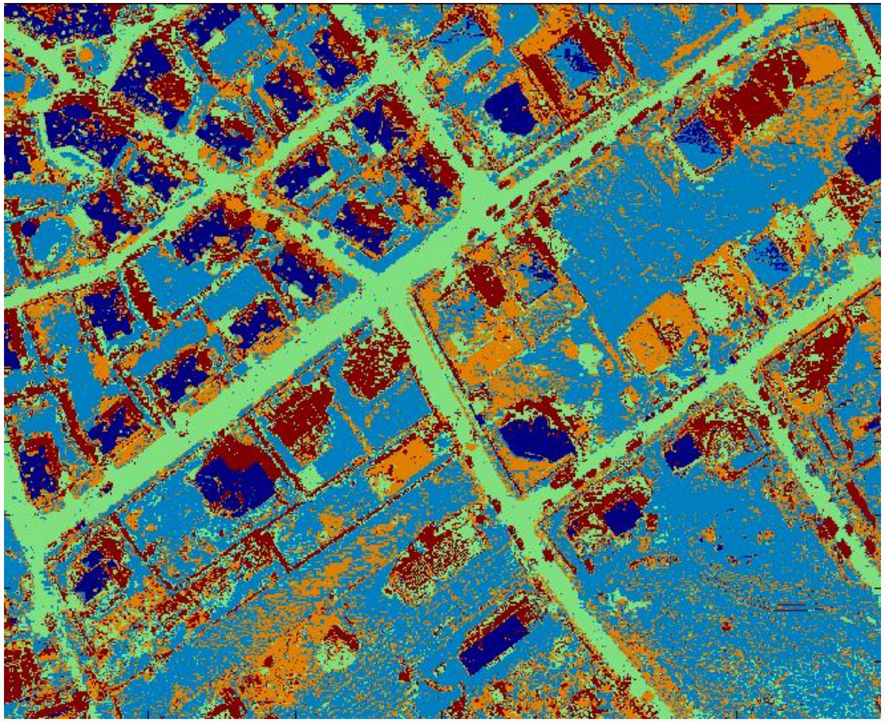
- Segmentation of a noisy grayscale image
  - Gaussian white noise, K=4



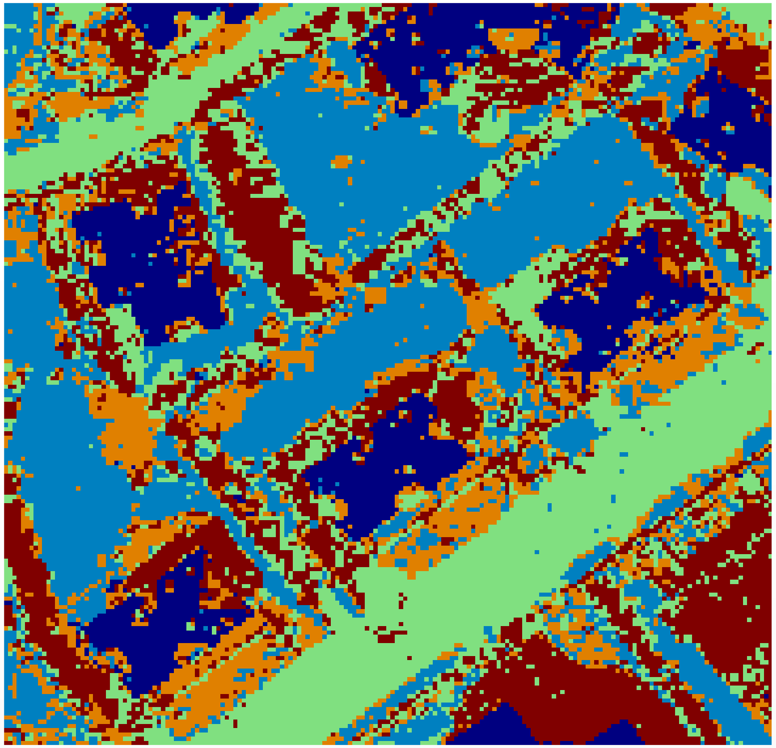


K-Means result

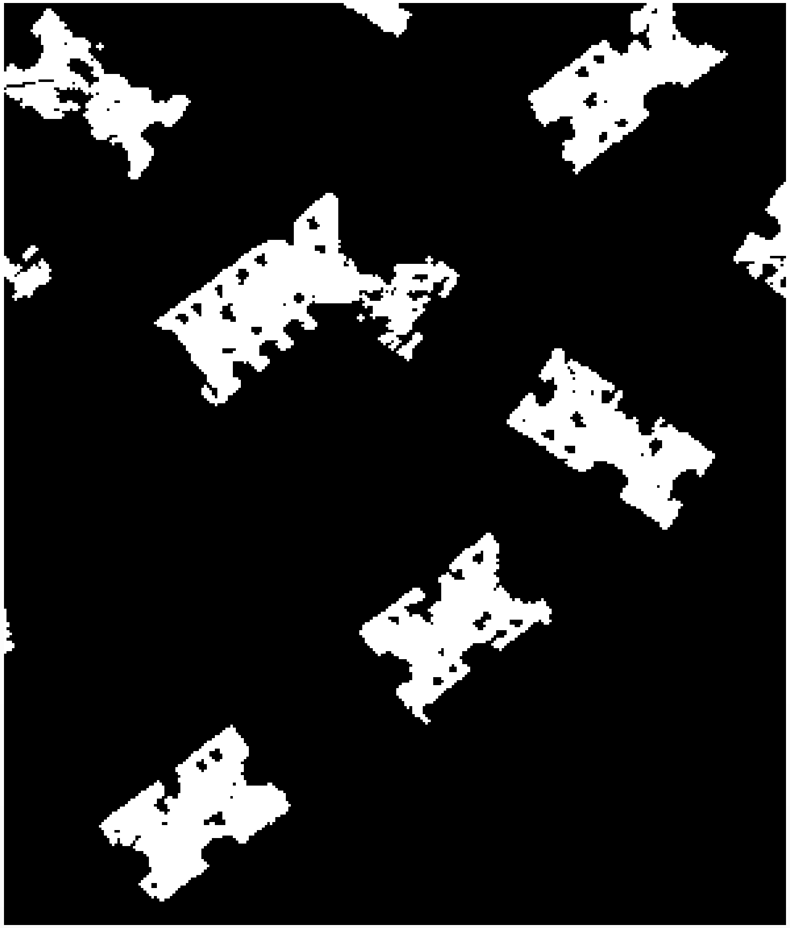
- Segmentation of an aerial image in $CIE\ L^*a^*b^*$, feature channels: (a^*b^*), $K = 5$



Per-pixel segmentation: noisy ouptut



Post processing: enhancing the regions by compactness and shape analysis



Morphology - overview

- Once segmentation is complete, morphological operations can be used to remove imperfections in the segmented image and provide information on the form and structure of the image
- In this section we will consider
  - What is morphology?
  - Simple morphological operations
  - Compound operations
  - Morphological algorithms

Slides for dilation/erosion credits: Dublin Institute of Technology

What Is Morphology?

- Morphological image processing (or **morphology**) describes a range of image processing techniques that deal with the shape (or morphology) of features in an image
- Morphological operations are typically applied to **remove imperfections** introduced during segmentation, and so typically operate on **bi-level images**

Morphological Operations: details
1, 0, Black, White?

- Throughout all of the following slides whether 0 and 1 refer to white or black is a little interchangeable
- All of the discussion that follows assumes segmentation has already taken place and that images are made up of 0s for background pixels and 1s for object pixels
- After this it doesn’t matter if 0 is black, white, yellow, green…….

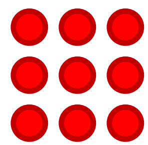
Morphological Operations

- Morphological operations are affecting the form, structure or shape of an object.
- They are used in pre- or postprocessing (filtering, thinning, and pruning) or for getting a representation or description of the shape of objects/regions (boundaries, skeletons convex hulls).
- Two basic operations:
  - **Dilation**: expands the object, fills in small holes and connects disjoint objects.
  - **Erosion**: shrinks objects by removing (eroding) their boundaries.
- The basic idea in binary morphology is to probe an image with a ***structuring element*** (a simple, pre-defined shape), drawing conclusions on how this shape fits or misses the shapes in the image.

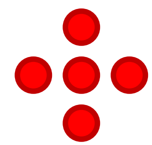
Morphological Operations

- Structuring element:

  e.g.:



8 neighbors



4 neighbors

- Dilation:
  - A shift-invariant operator, that expands the object, fills in small holes and connects disjoint objects.
  - Steps:
    - The structuring element is placed on each pixel on the image
    - If the pixel belongs to the foreground pixel, we do nothing
    - If the pixel belongs to the background, we change it to a foreground pixel if any pixel covered by the structuring element is a foreground pixel.

Morphological Operations

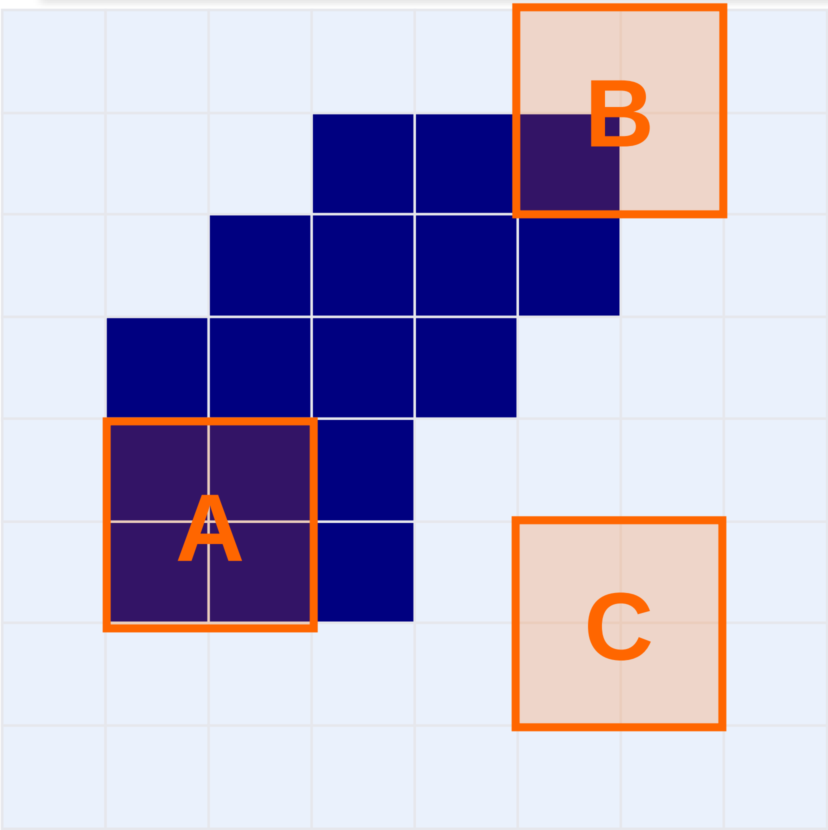
- Erosion:
  - A shift-invariant operator, that erodes away the boundaries of regions of foreground pixels. Thus areas of foreground pixels shrink in size, and holes within those areas become larger.
  - Steps:
    - The structuring element is placed on each pixel on the image
    - If the pixel is a background pixel, we do nothing
    - If the pixel is a foreground pixel, we change this pixel to a background if any pixel covered by the structuring element is a background pixel.
- Erosion on the image has the same effect as dilatation on the inverse image.
- **Opening**: Erosion + Dilation
- **Closing**: Dilation + Erosion

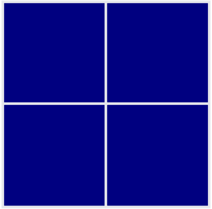
Quick Example

Image after segmentation

Image after segmentation and morphological processing

Structuring Elements, Hits & Fits





Fit: All *on pixels* in the structuring element cover *on pixels* in the image

Hit: Any *on pixel* in the structuring element covers an *on pixel* in the image

All morphological processing operations are based on these simple ideas

Structuring Elements

- Structuring elements can be any size and make any shape
- However, for simplicity we will use rectangular structuring elements with their origin at the middle pixel.
  - 1s represent the *on pixels* of the structuring element

1	1	1
1	**1**	1
1	1	1

0	1	0
1	**1**	1
0	1	0

0	0	1	0	0
0	1	1	1	0
1	1	**1**	1	1
0	1	1	1	0
0	0	1	0	0

Fitting & Hitting

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	1 B	1	1	1	0	0 C	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	1	1	1	0	0	0
0	0	1	1	1	1	1	1 A	1	1	1	0
0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0

1	1	1
1	1	1
1	1	1

Structuring Element 1

0	1	0
1	1	1
0	1	0

Structuring Element 2

Fundamental Operations

- Fundamentally morphological image processing is very like spatial filtering
- The structuring element is moved across every pixel in the original image to give a pixel in a new processed image
- The value of this new pixel depends on the operation performed
- There are two basic morphological operations: **erosion** and **dilation**

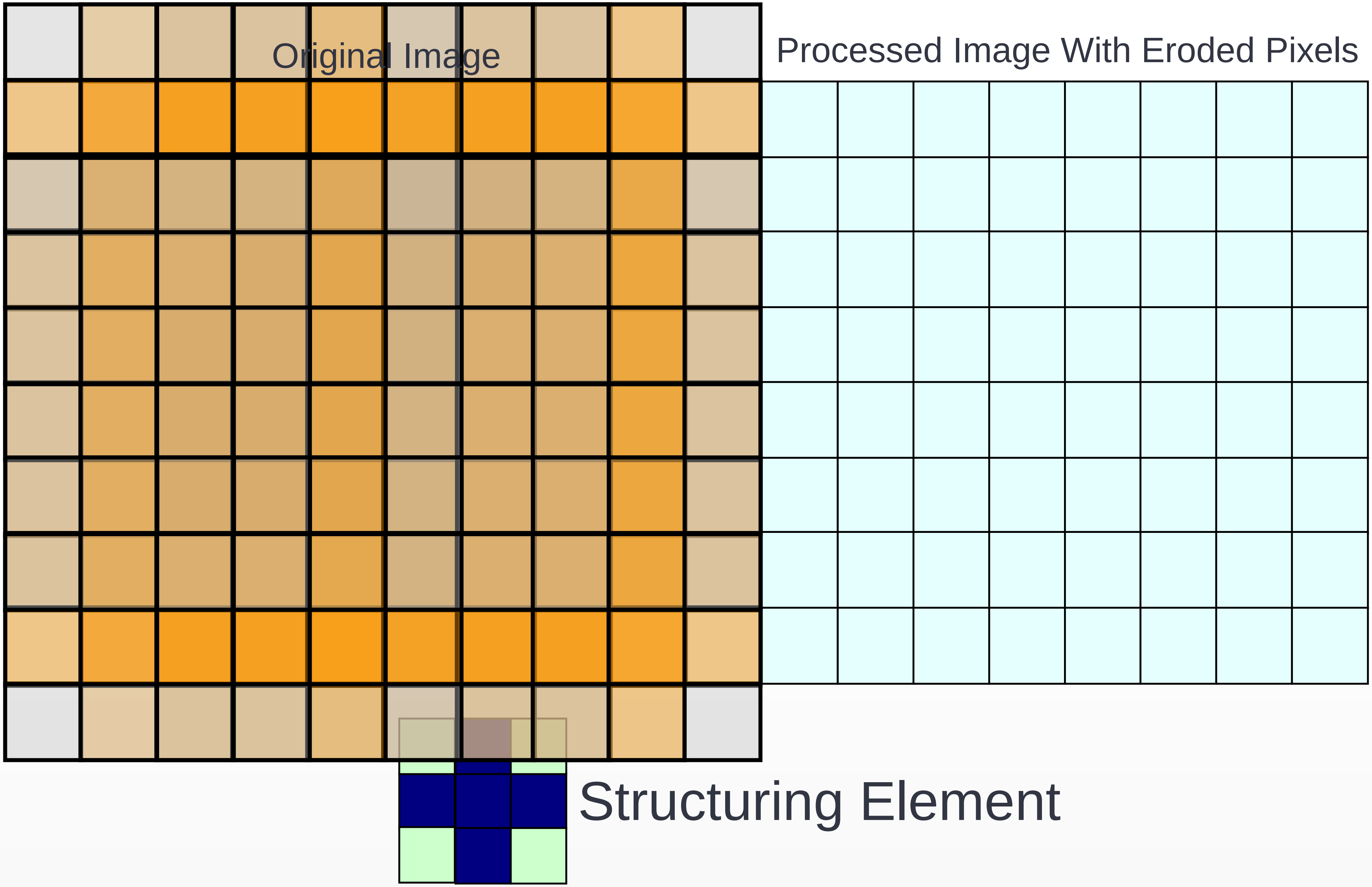
Erosion

- **Erosion** of image f by structuring element s is given by $f \ominus s$

The structuring element s is positioned with its origin at (x, y) and the new pixel value is determined using the rule:

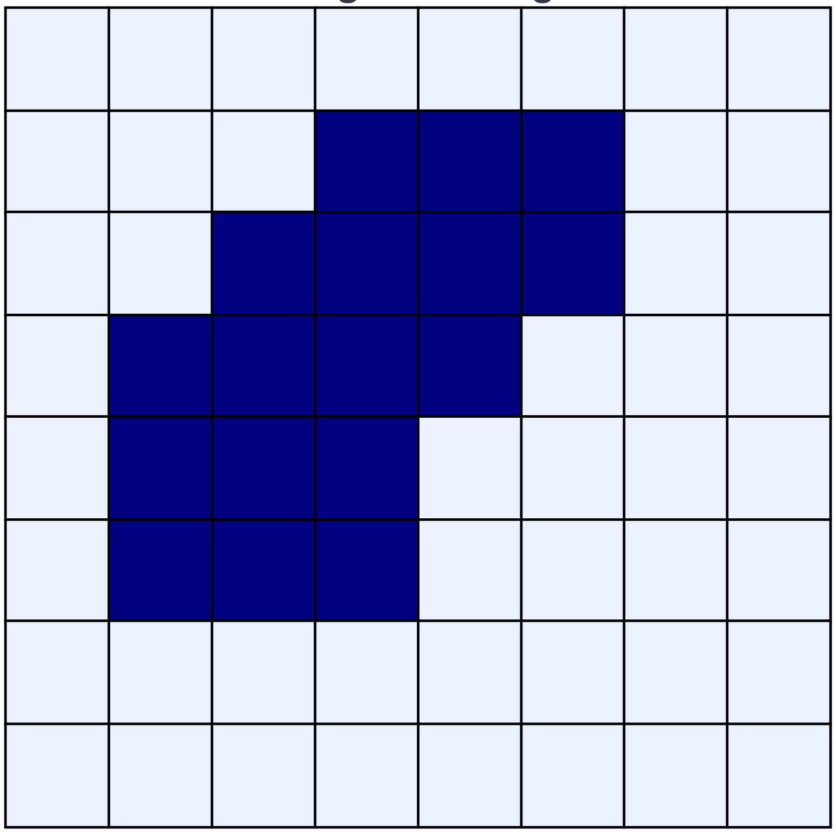
$$g(x, y) = \begin{cases} 1 & \text{if } s \text{ fits } f \\ 0 & \text{otherwise} \end{cases}$$

Erosion Example

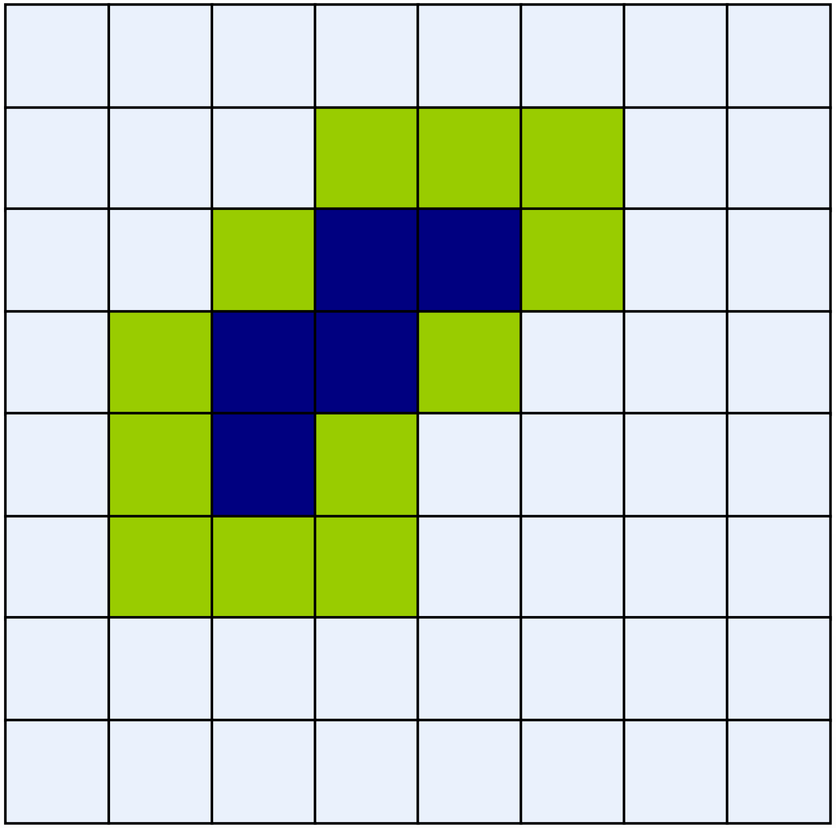


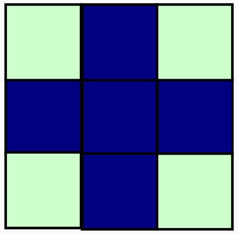
Erosion Example

Original Image



Processed Image With Eroded Pixels





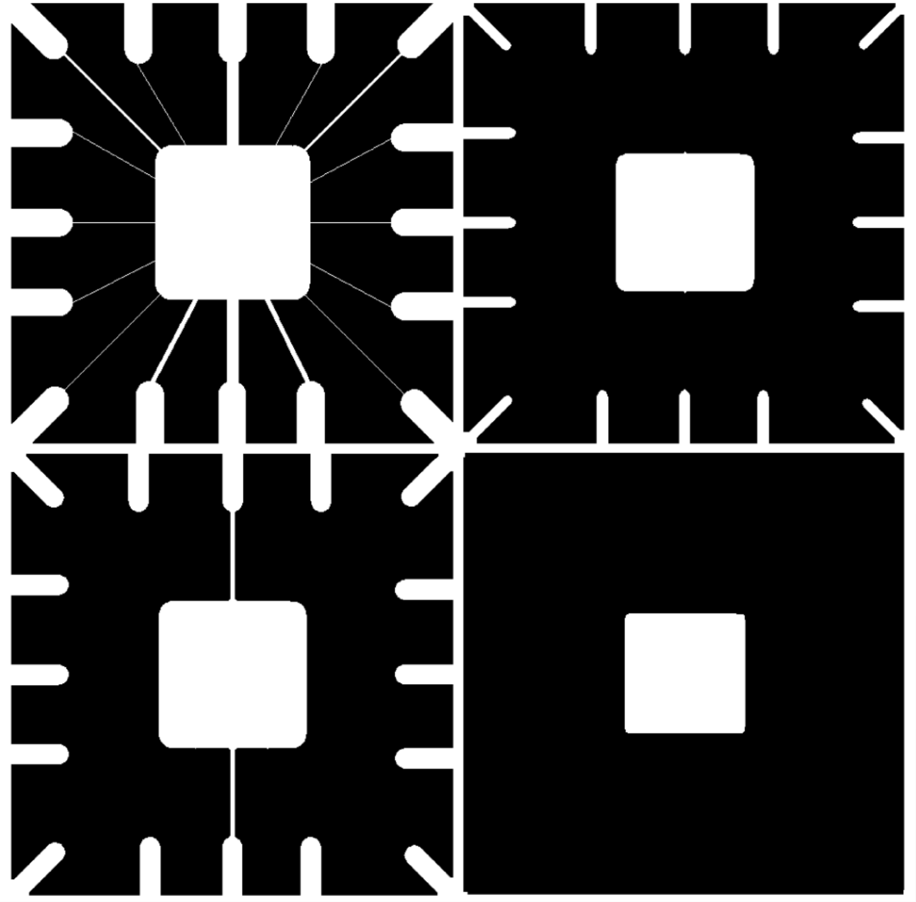
Structuring Element

Erosion Example 1

Watch out: In these examples a 1 refers to a black pixel!

Erosion Example 2

Original image



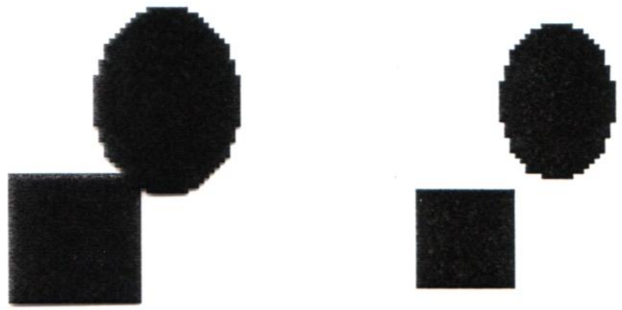
After erosion with a disc of radius 10

After erosion with a disc of radius 5

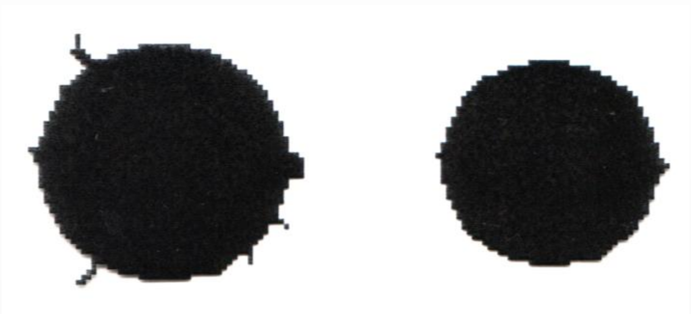
After erosion with a disc of radius 20

What Is Erosion For?

Erosion can split apart joined objects



Erosion can strip away extrusions



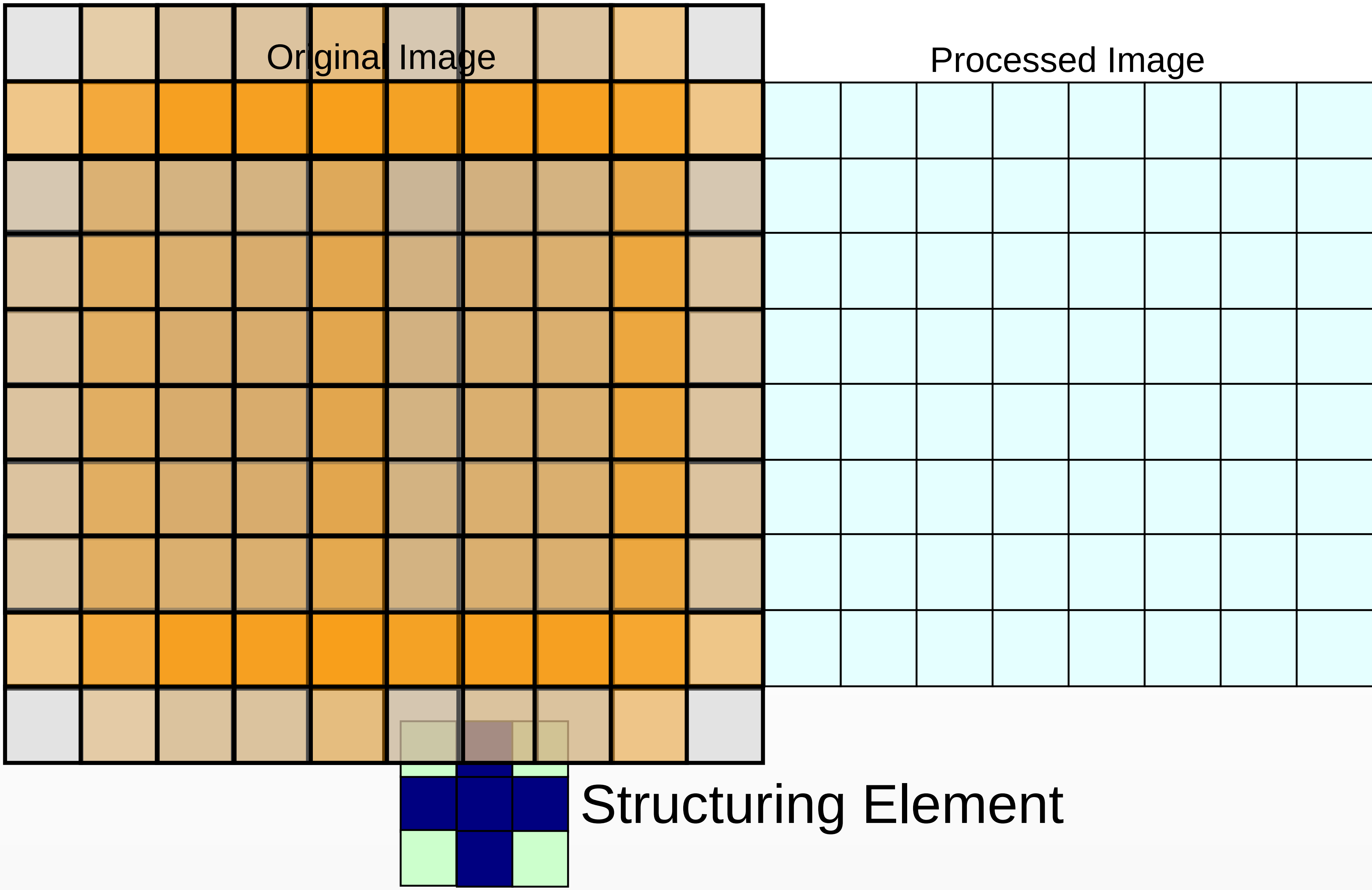
Watch out: Erosion shrinks objects

Dilation

- **Dilation** of image f by structuring element s is given by $f \oplus s$
- The structuring element s is positioned with its origin at (x, y) and the new pixel value is determined using the rule:

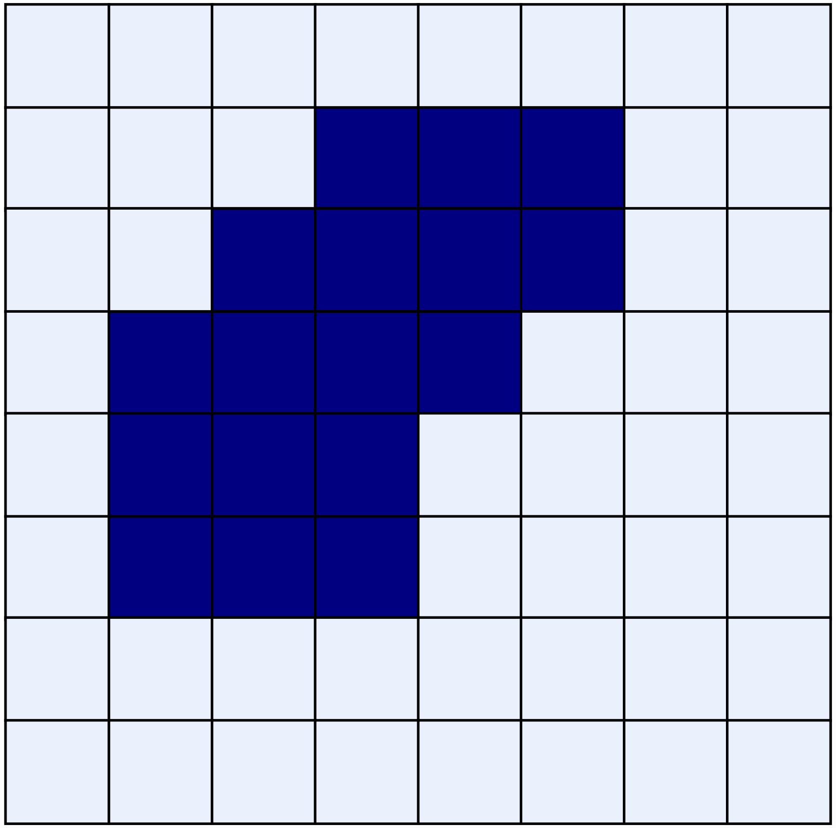
$$g(x,y) = \begin{cases} 1 \text{ if } s \text{ hits } f \\ 0 \text{ otherwise} \end{cases}$$

Dilation Example

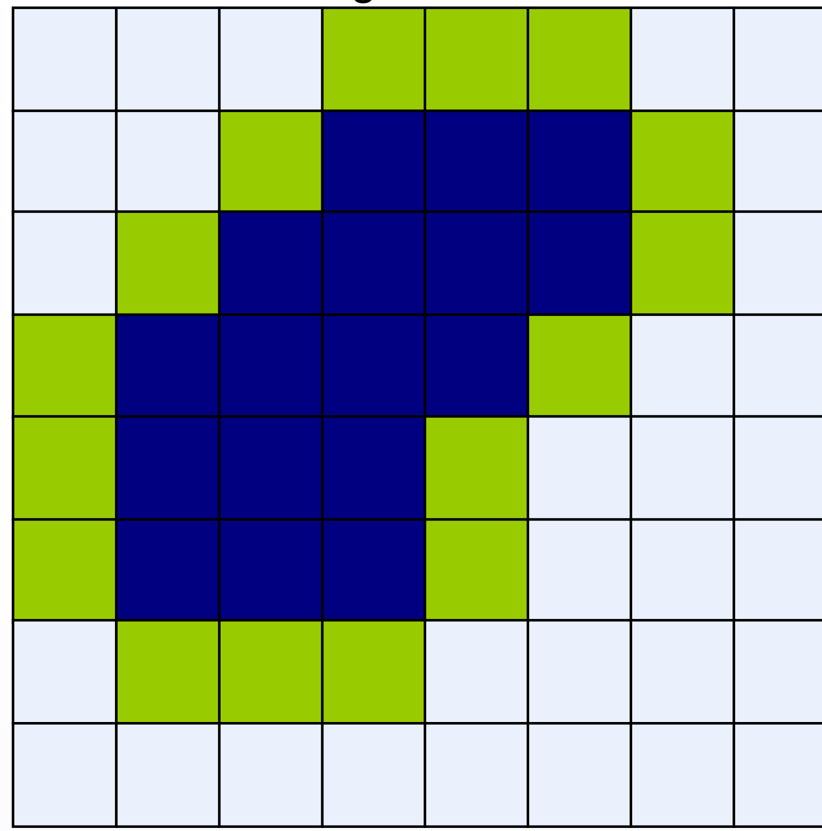


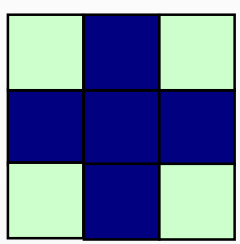
Dilation Example

Original Image



Processed Image With Dilated Pixels





Structuring Element

Dilation Example 1

Original image

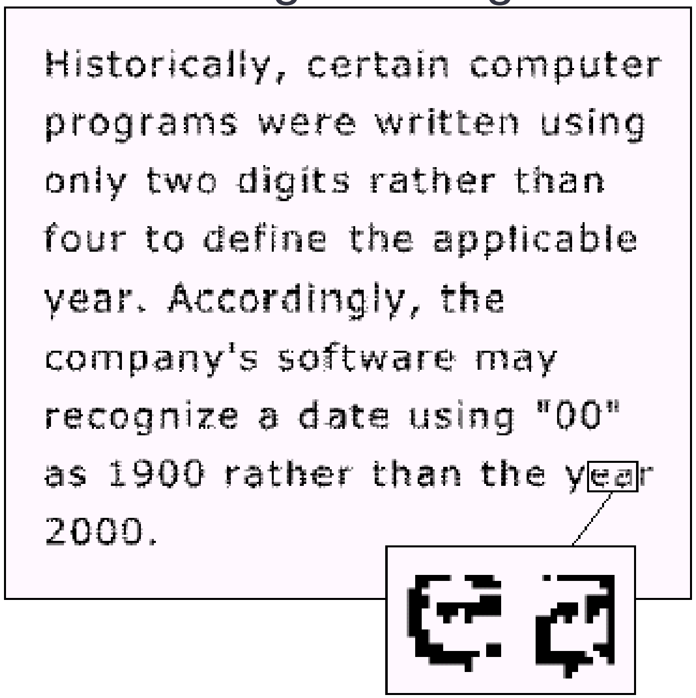
Dilation by 3*3 square structuring element

Dilation by 5*5 square structuring element

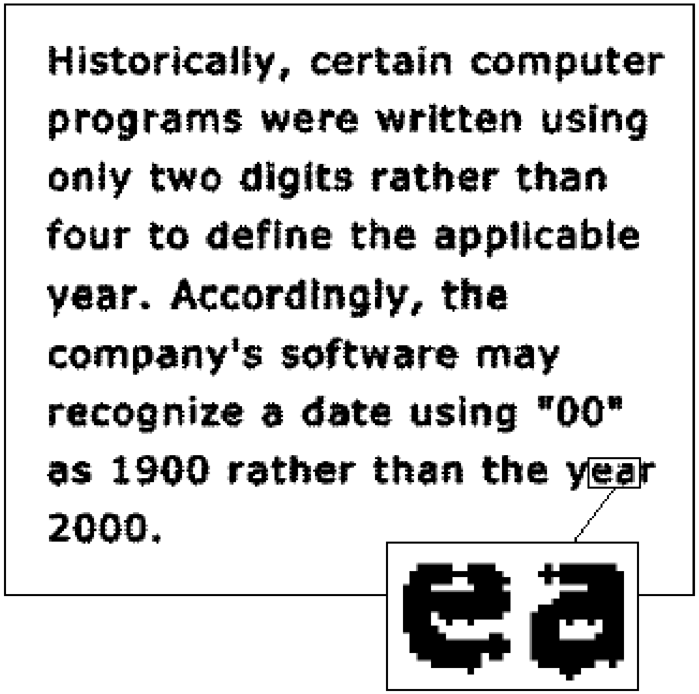
Watch out: In these examples a 1 refers to a black pixel!

Dilation Example 2

Original image



After dilation



Structuring element

0	1	0
1	1	1
0	1	0

What Is Dilation For?

Dilation can repair breaks

Dilation can repair intrusions

Watch out: Dilation enlarges objects

Compound Operations

More interesting morphological operations can be performed by performing combinations of erosions and dilations

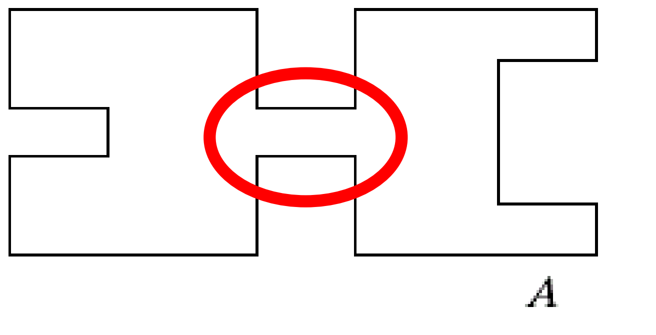
The most widely used of these *compound operations* are:

- Opening
- Closing

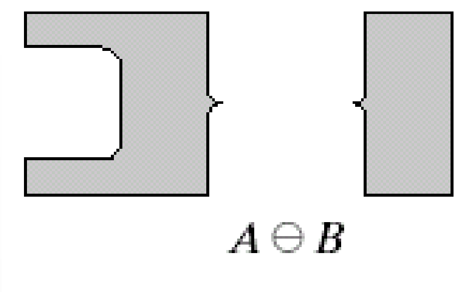
Opening

- The opening of image f by structuring element s, denoted $f \circ s$ is simply an erosion followed by a dilation

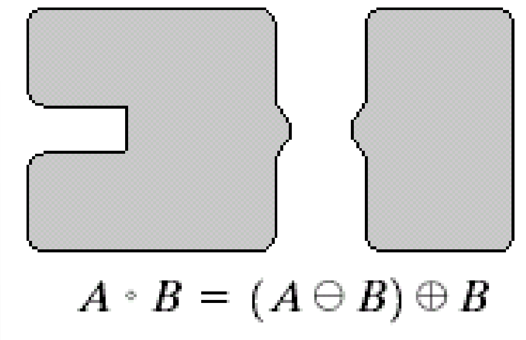
$$f \circ s = (f \ominus s) \oplus s$$



Original shape



After erosion



After dilation (opening)

Note: a disc shaped structuring element is used

Opening Example

Original Image

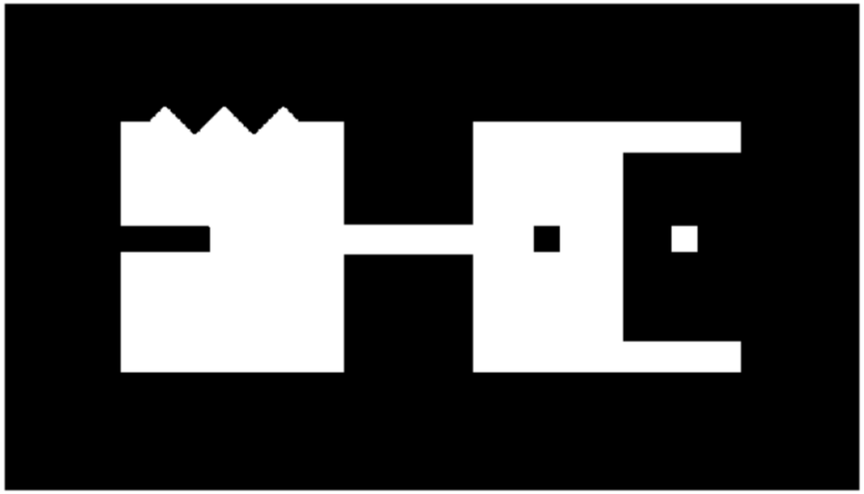
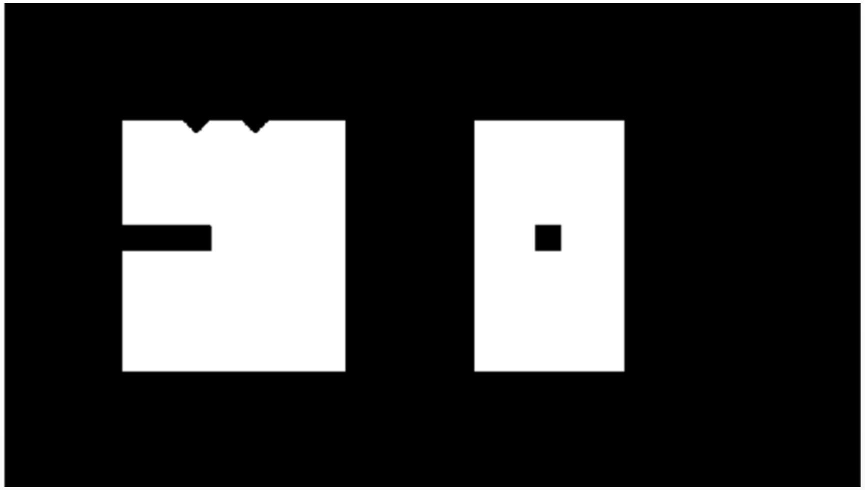
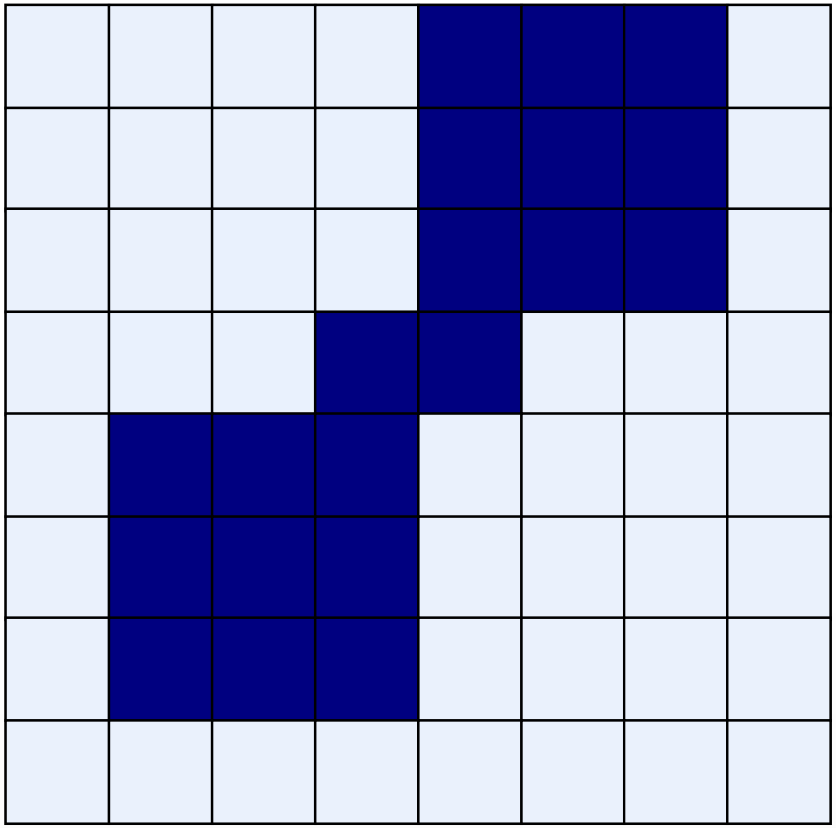


Image After Opening

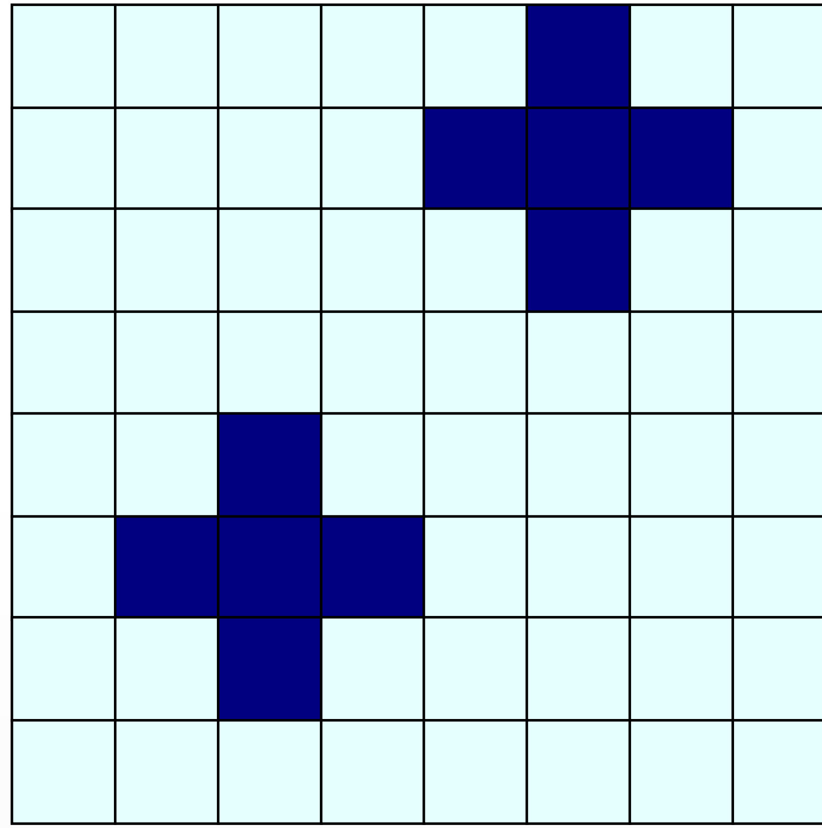


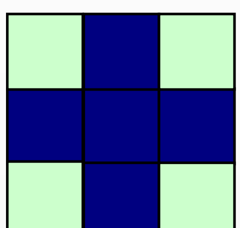
Opening Example

Original Image



Processed Image



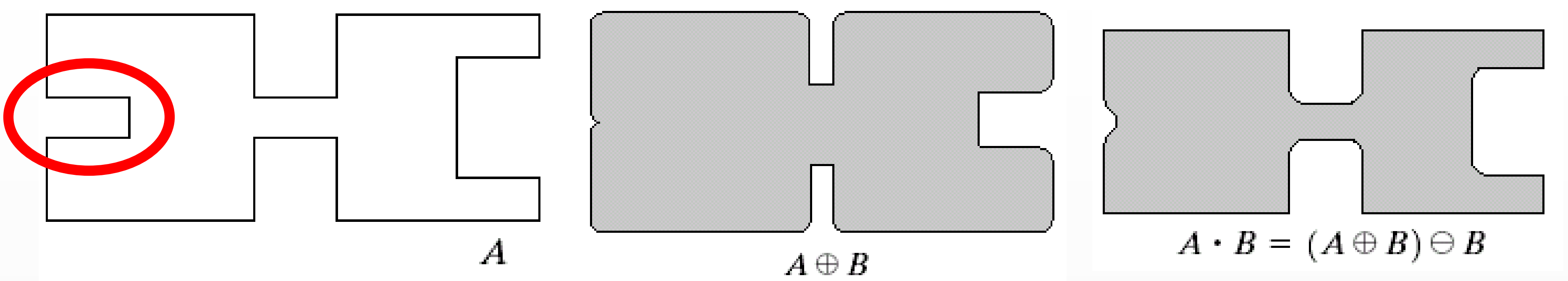


Structuring Element

Closing

- The closing of image f by structuring element s, denoted $f \bullet s$ is simply a dilation followed by an erosion

$$f \bullet s = (f \oplus s) \ominus s$$



Original shape | After dilation | After erosion (closing)

Note: a disc shaped structuring element is used

Closing Example

Original Image

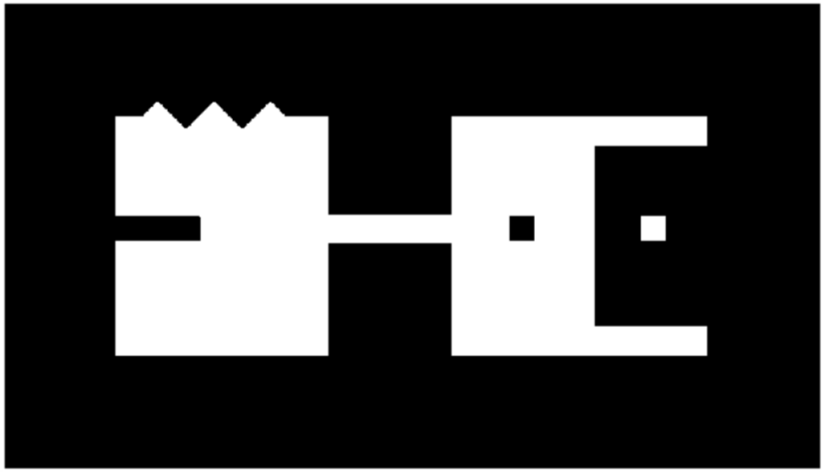
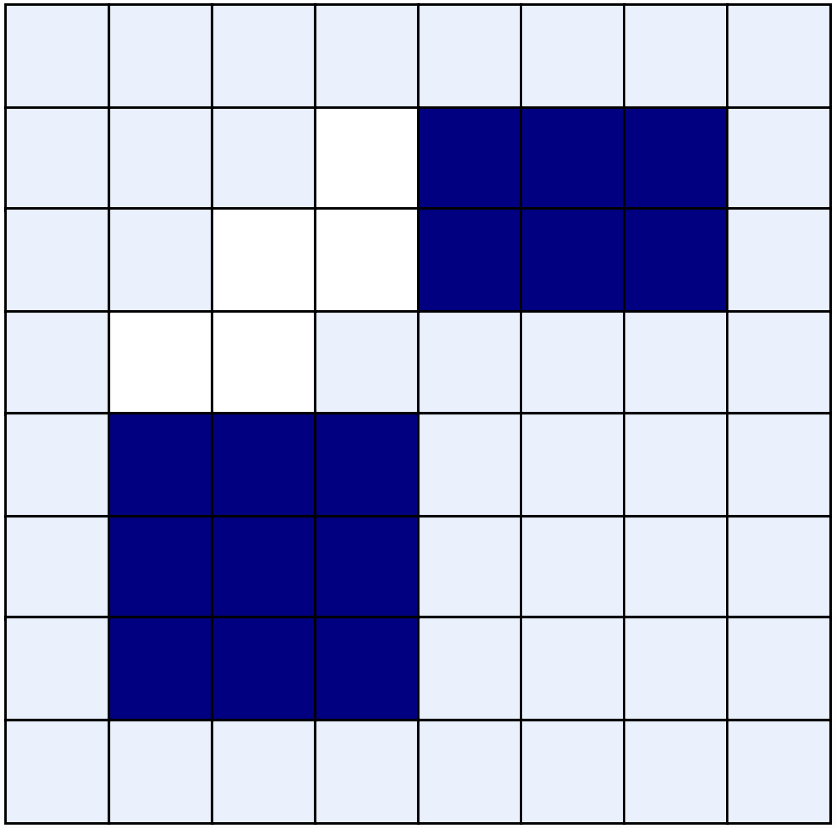


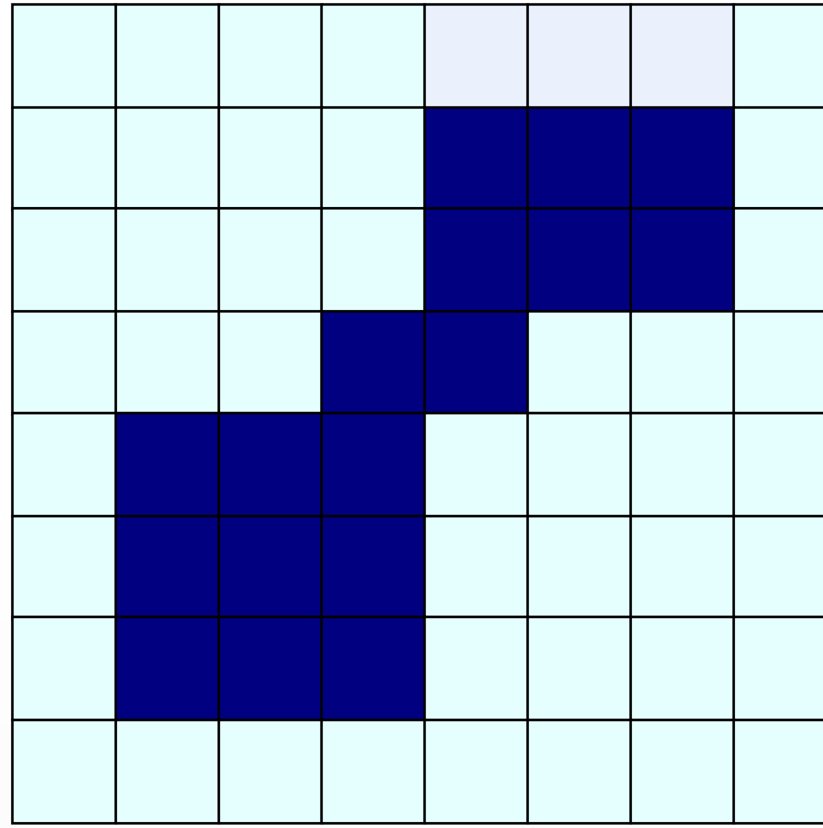
Image After Closing

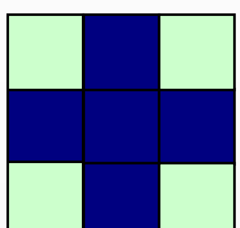
Closing Example

Original Image



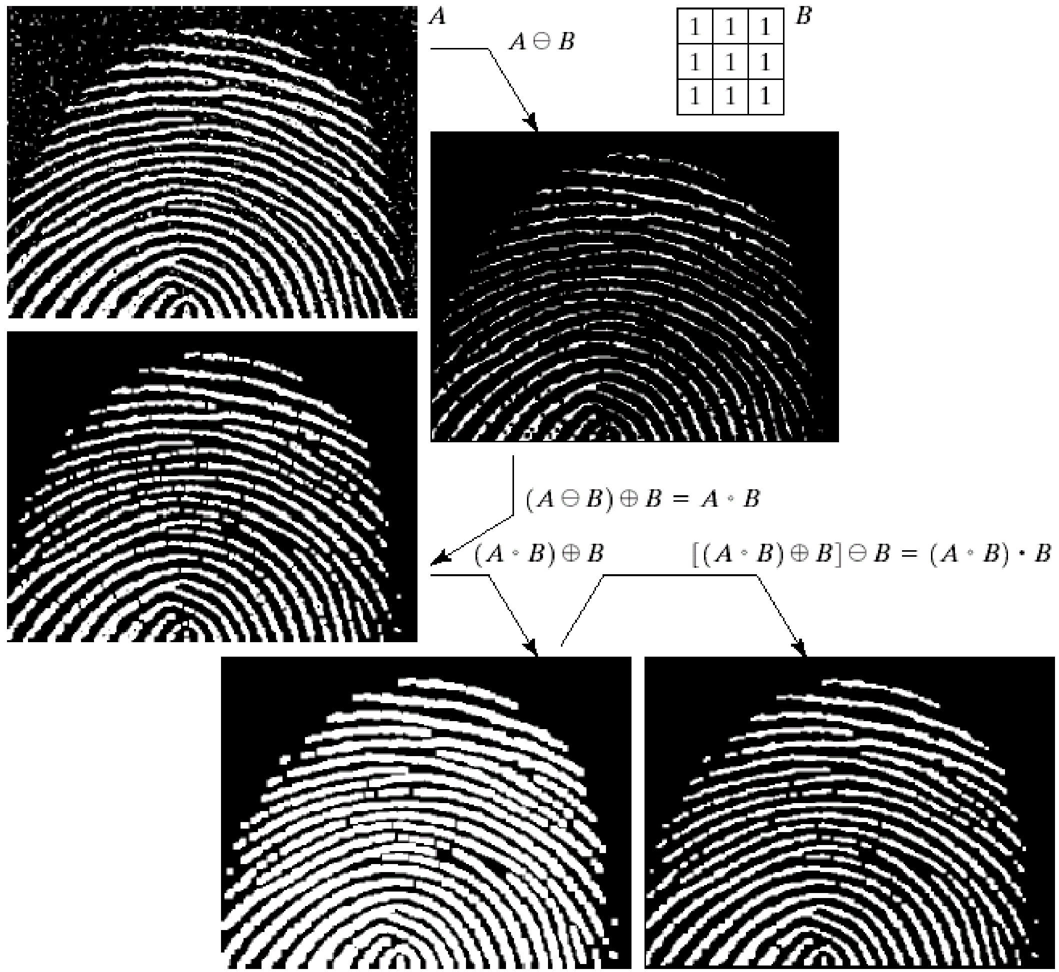
Processed Image





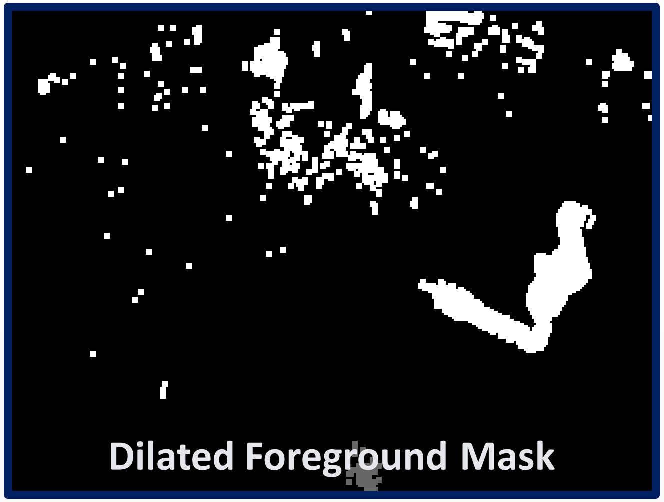
Structuring Element

Morphological Processing Example



Morphological Operations

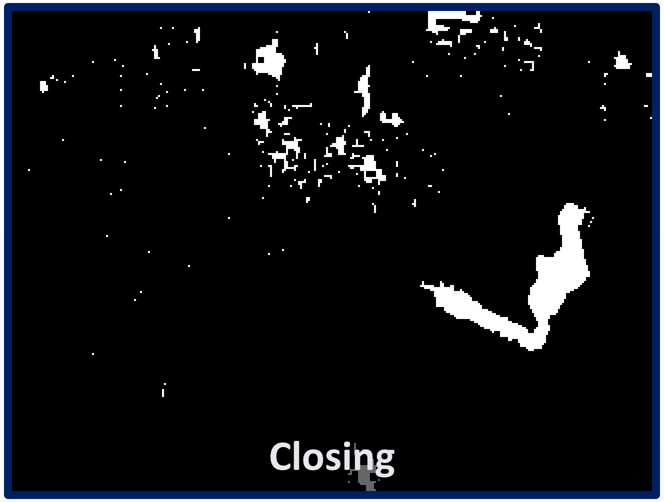
Foreground Mask of MoG (*T=20*)

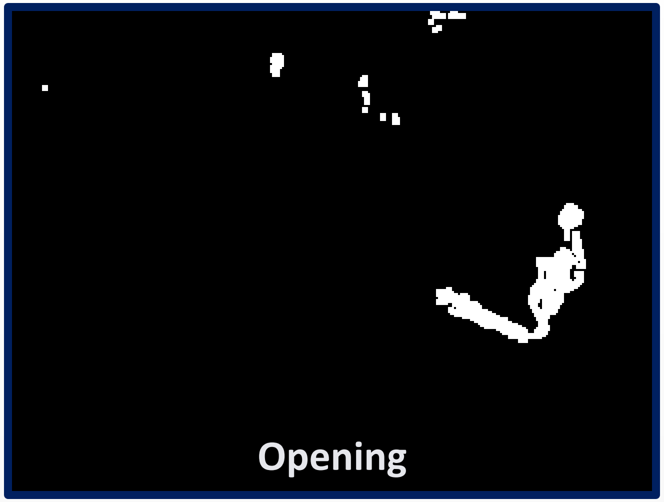


Dilated Foreground Mask

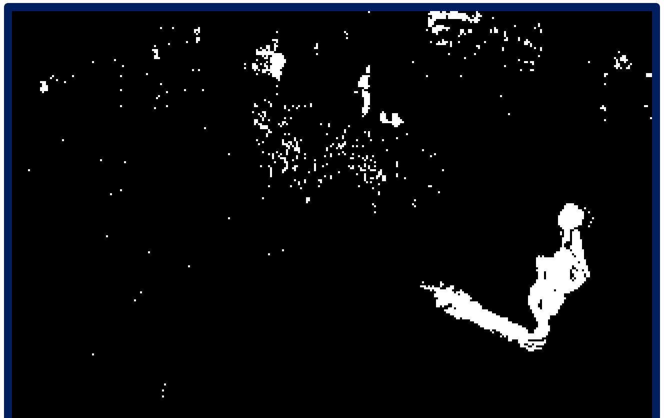
Eroded Foreground Mask

Morphological Operations





Morphological Operations



Foreground Mask of MoG (*T=20*)

The Fg mask after a more complicated sequence of erosin and dilation

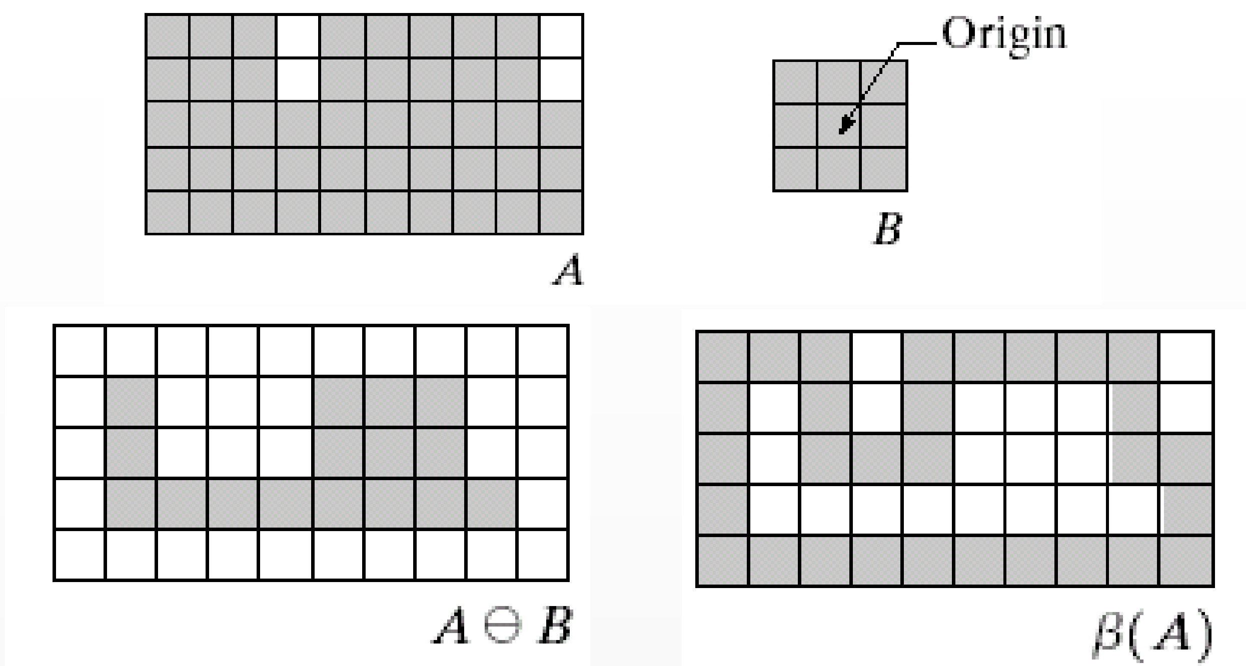
Morphological Algorithms

- Using the simple technique we have looked at so far we can begin to consider some more interesting morphological algorithms
- We will look at:
  - Boundary extraction
  - Region filling
- There are lots of others as well though:
  - Extraction of connected components
  - Thinning/thickening
  - Skeletonisation

Boundary Extraction

- Extracting the boundary (or outline) of an object is often extremely useful
- The boundary can be given simply as

$$\beta(A) = A - (A \ominus B)$$



Boundary Extraction Example

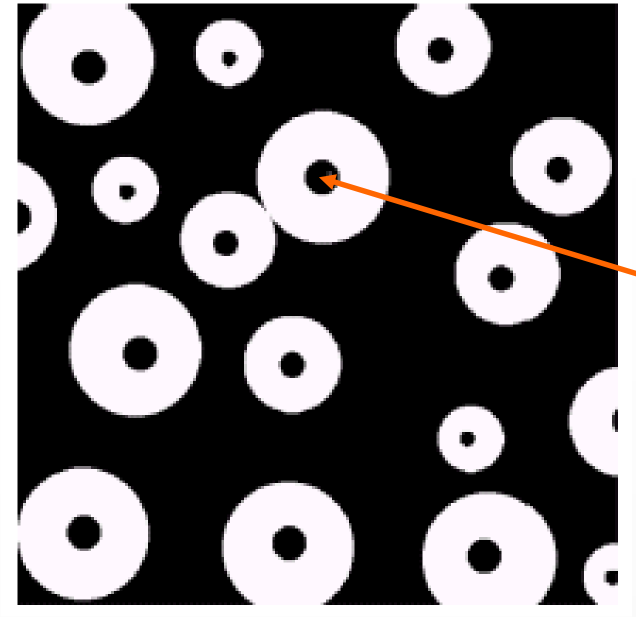
- A simple image and the result of performing boundary extraction using a square 3×3 structuring element

Original Image

Extracted Boundary

Region Filling

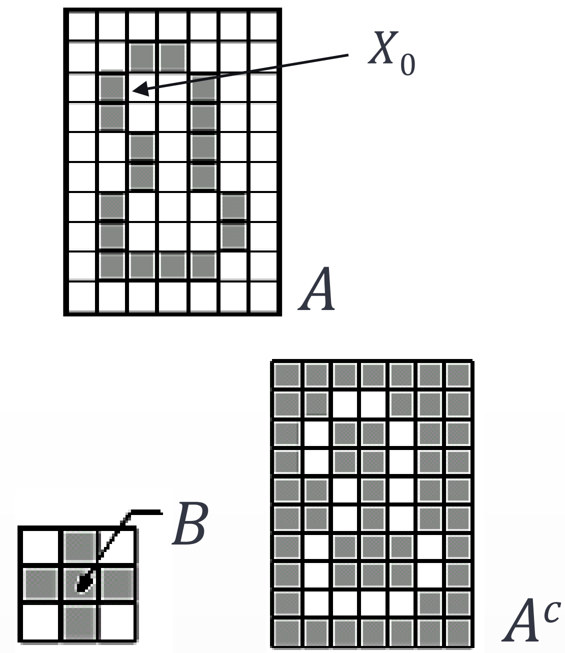
- Given a pixel inside a boundary, *region filling* attempts to fill that boundary with object pixels (1s)



Given a point inside here, can we fill the whole circle?

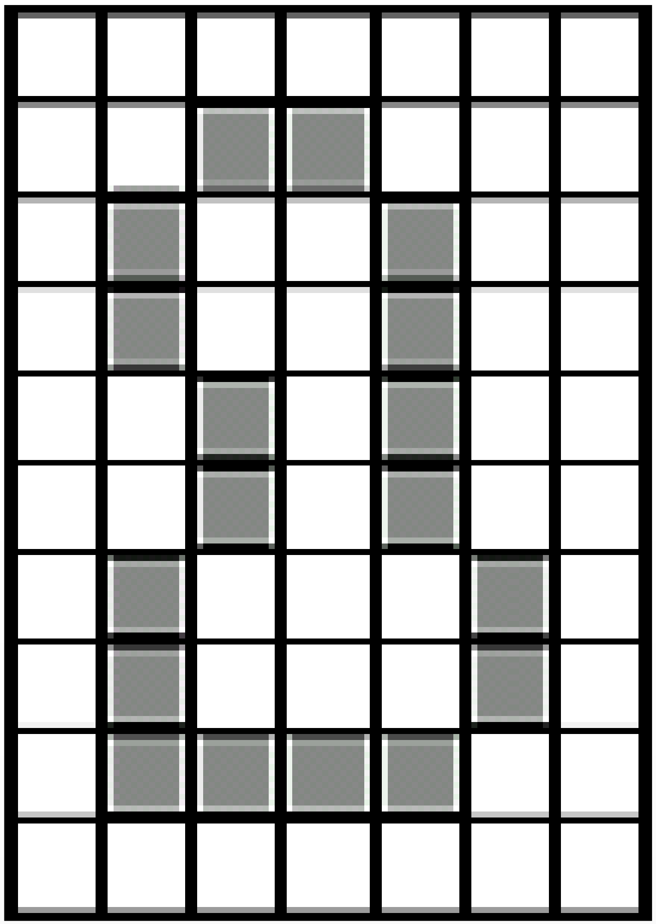
Region Filling

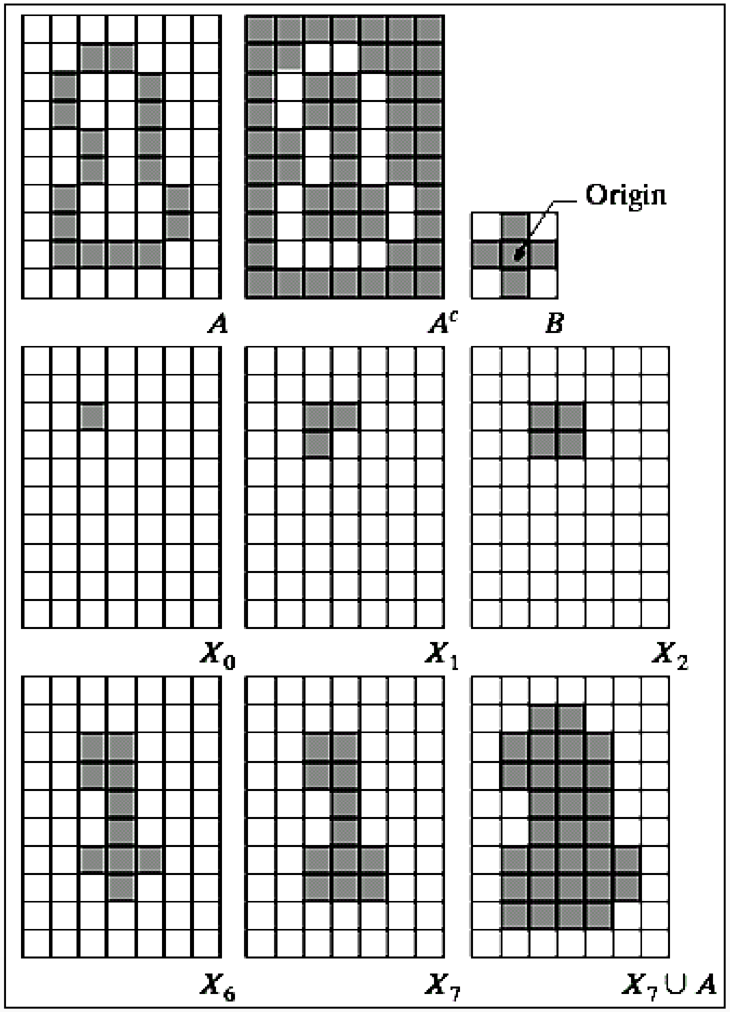
- The key equation for region filling is:
  $X_k = (X_{k-1} \oplus B) \cap A^c$, where
  - A is the original (boundary) image,
  - X_0 is simply the starting point (single pixel) inside the boundary,
  - B is a simple structuring element and
  - A^c is the complement of A



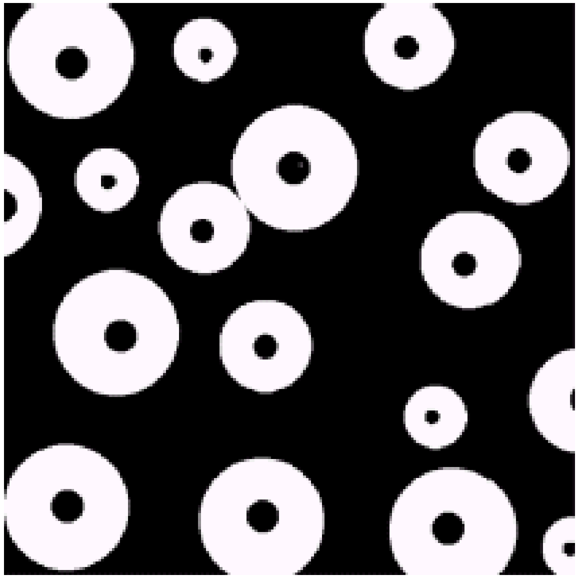
- This equation is applied repeatedly until X_k is equal to X_{k-1}
- Finally the result is unioned with the original boundary

Region Filling Step By Step

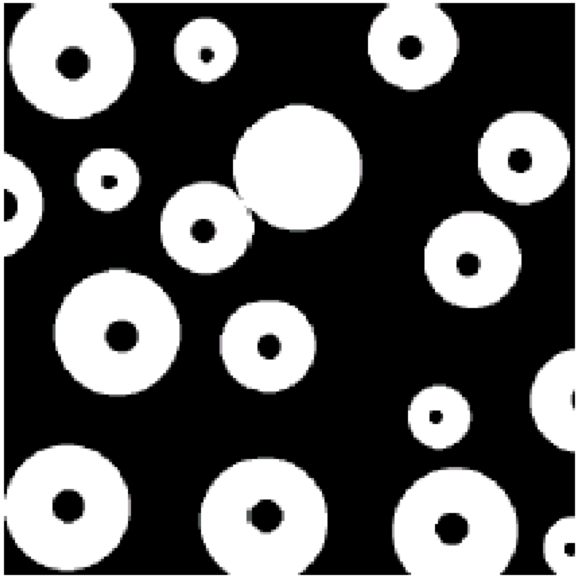




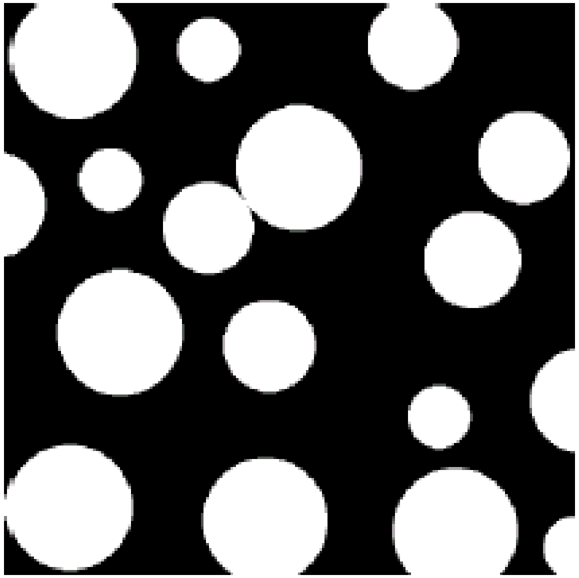
Region Filling Example



Original Image



One Region Filled



All Regions Filled

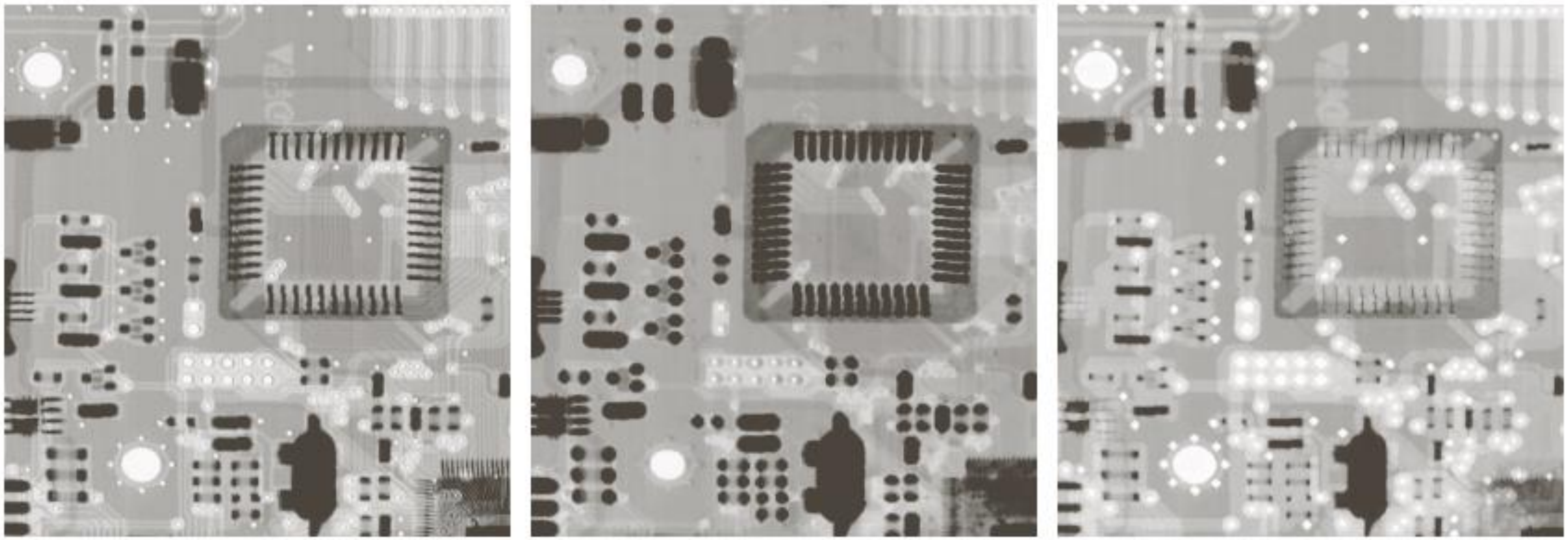
Grayscale morphology

- Gray-Scale Morphology: Erosion and Dilation by Flat Structuring

$$[f - b](x, y) = \min_{(s,t)\in b}\{f(x+s, y+t)\}$$

$$[f \oplus b](x, y) = \max_{(s,t)\in b}\{f(x-s, y-t)\}$$

Grayscale morphology



Erosion Dilation

Summary-morphology

- The purpose of morphological processing is primarily to remove imperfections added during segmentation
- The basic operations are *erosion* and *dilation*
- Using the basic operations we can perform *opening* and *closing*
- More advanced morphological operation can then be implemented using combinations of all of these