Multi-scale Imaging for the Study of Reproduction Summer school

2016 June 21-24 – Nouzilly, France

Part III – From Image Analysis to Pattern Recognition

(Classification)

Jean-Yves Ramel

ramel@univ-tours.fr

Laboratoire Informatique de Tours
Polytech'Tours

From Image Analysis to Pattern Recognition

- Brief Introduction

- Image segmentation
  - What does it mean?
  - Contour detection
  - Detection of Regions of Interest (ROI)

- Getting Binary Images
  - Why is it interesting?
  - Image Binarization
  - Connected component extraction and analysis

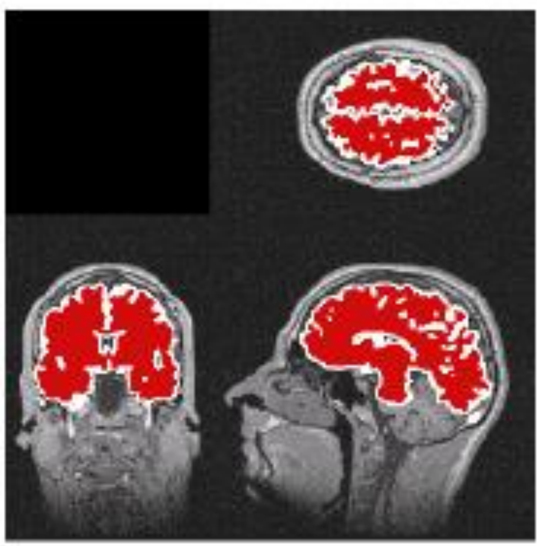
- From segmentation to recognition
  - What is machine learning? What is classification?
  - Image segmentation using pixel classification
  - ROI characterization and classification

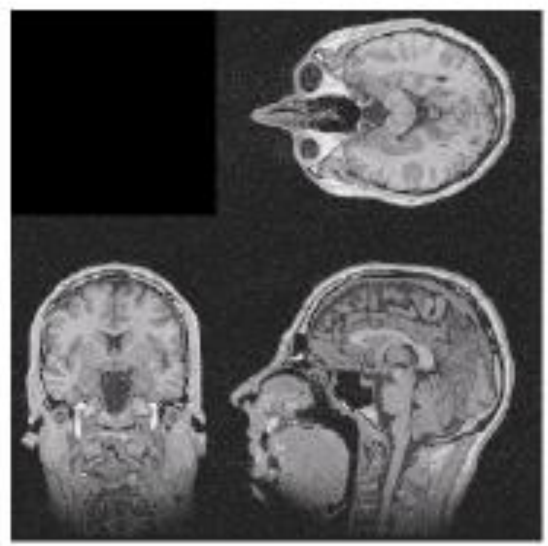
Segmentation vs Recognition

- **Image Segmentation** = Detection of Region of interest, localization and extraction of specific areas = image decomposition into homogeneous regions (according to a specific criteria)
- **Object recognition** = identification of objects / regions = class assignement (→ classifying)

The dilemma : Need to segment to recognize

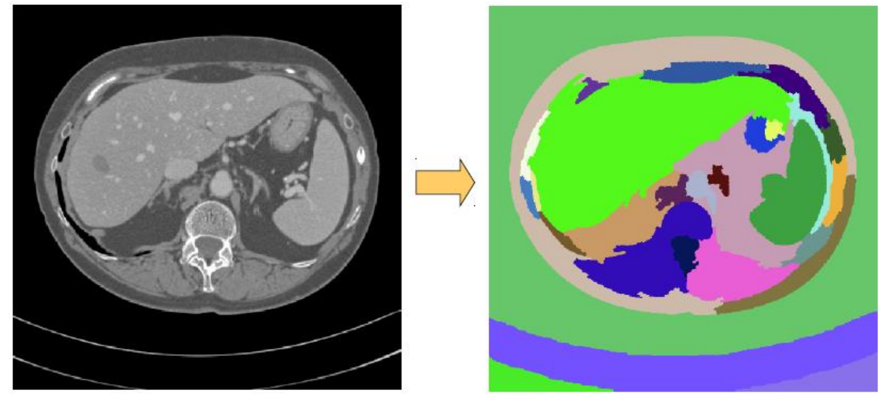
or need to recognize to better segment ???





What is segmentation ?

- Image decomposition into regions corresponding to meaningful objects
  - 1 object = 1 label ➔ Pixel tagging
  - 1 object = Homogeneity criteria ?
  - Same color = same region



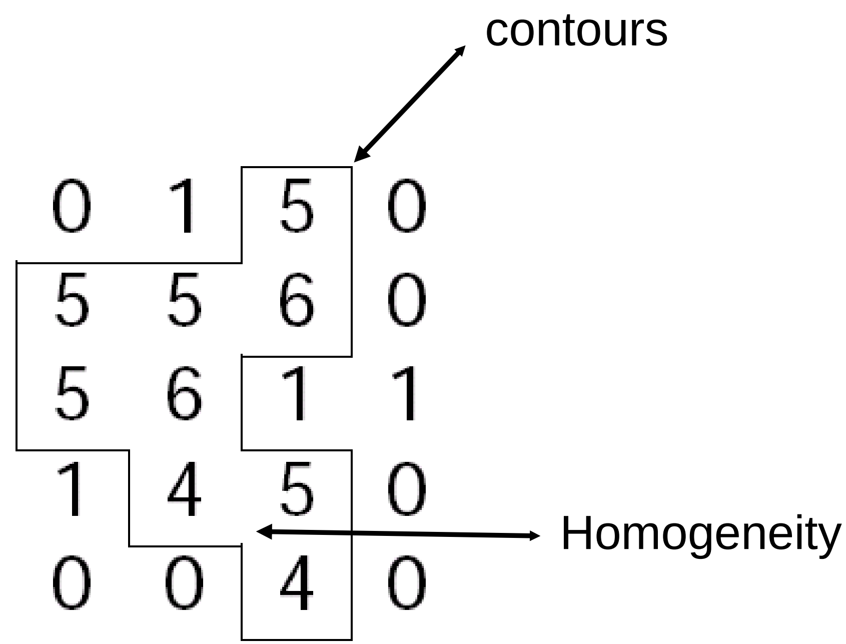
1 region = 1 label = 1 color
(for human visualisation)

- Easy for human
  - Use of a priori knowledge
  - Looking to the whole image for a more global interpretation
- 3 main categories of segmentation methods
  - Region based approach : associate similar pixels to obtain homogenous regions
  - Contour based approach: search for adjacent dissimilar pixels to obtain the frontiers between homogenous areas
  - Machine Learning based approaches

What is segmentation ?

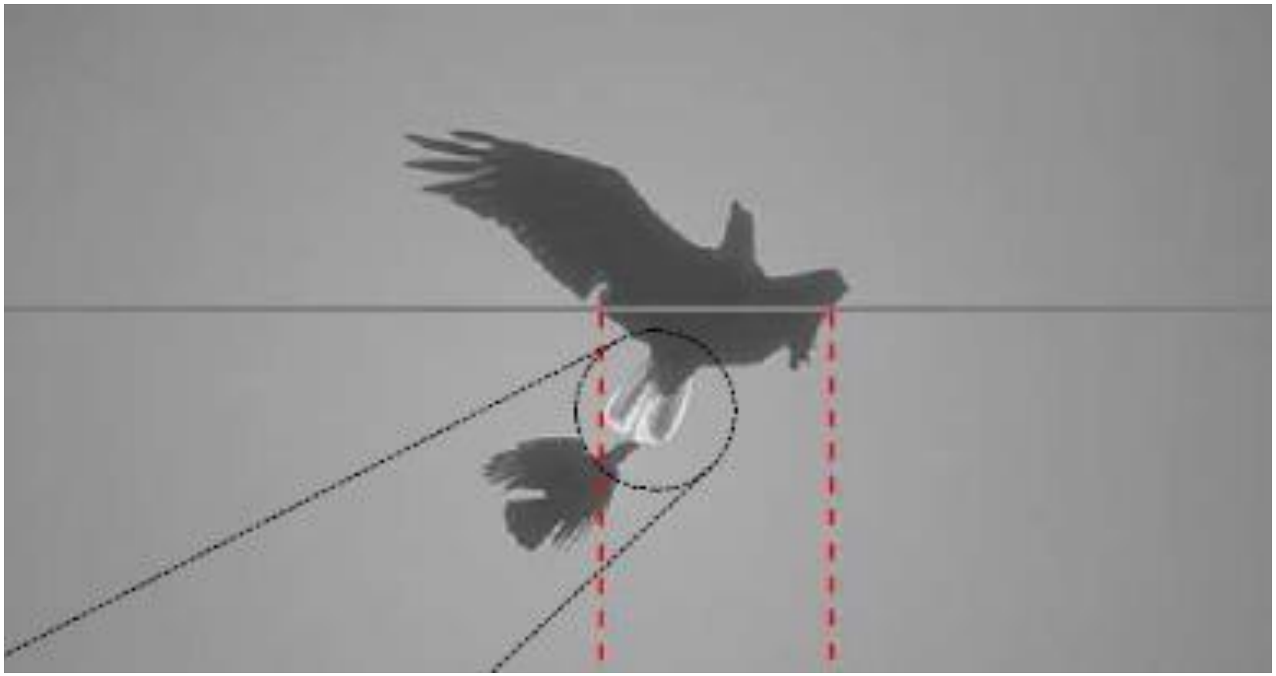
- From image to regions
- Region characterization
  - Histogram h(i) = (0; 1; 2; 3; 4; 5; 6) ➔ (7; 4; 0; 0; 2; 5; 2)
  - Number of pixels inside the region R = 9
  - Average of pixel intensities R = 5
  - Standard deviation R = 4/9
  - Length of the contour R = 18

Image ➔ Region R



Contour based approach

Question: How to detect the contours in an image ???



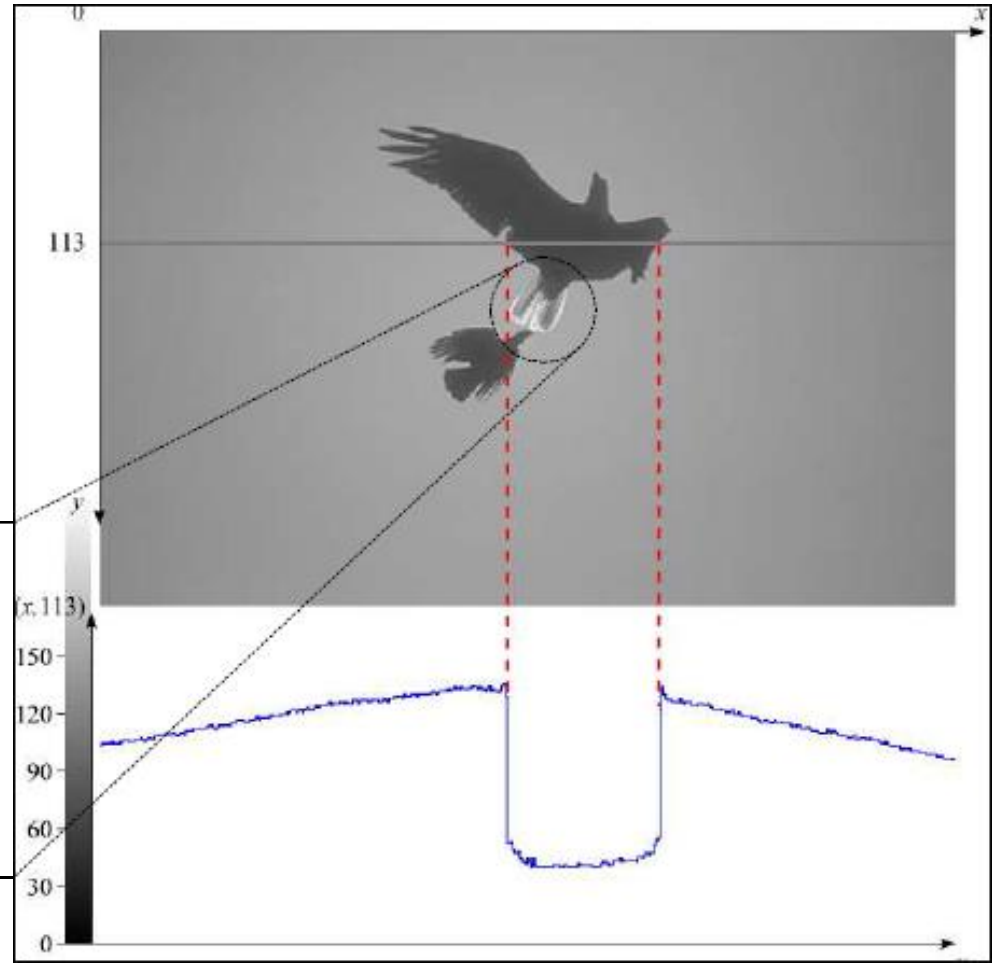
Notion of contours

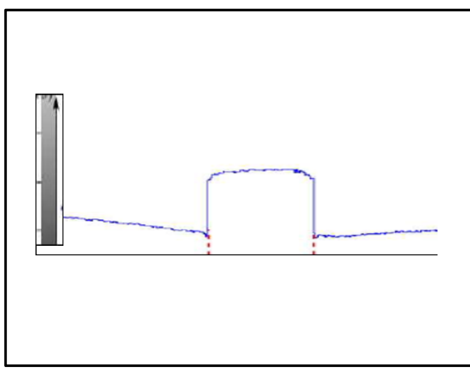
Definition

- Contour = Frontier between 2 objects or between an object and the background inside an image

Characterization of contours

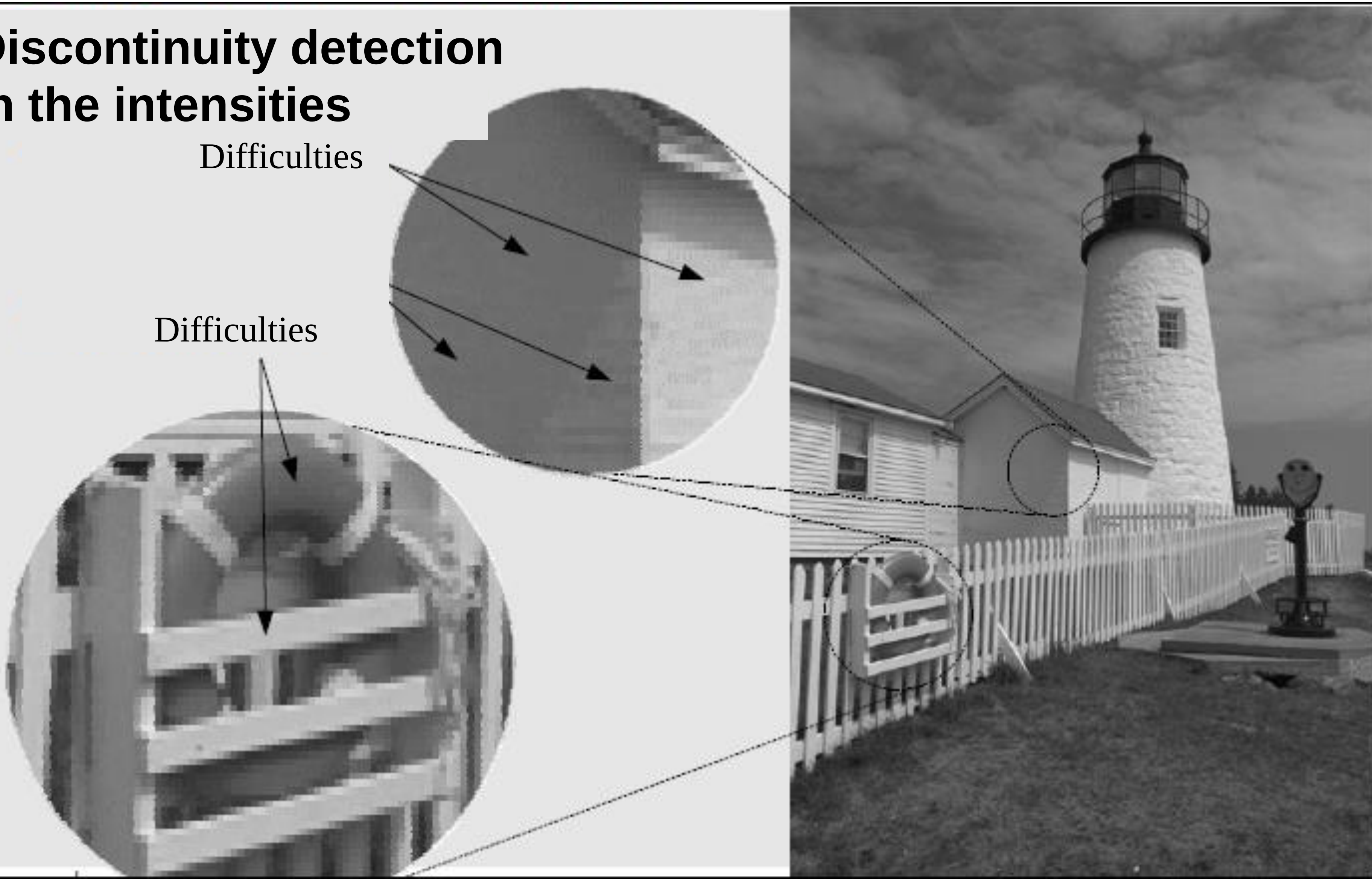
- Sudden variation of the intensity (discontinuity)
- *But d*iscontinuities do not always correspond to contours





Notion of contours

Discontinuity detection in the intensities

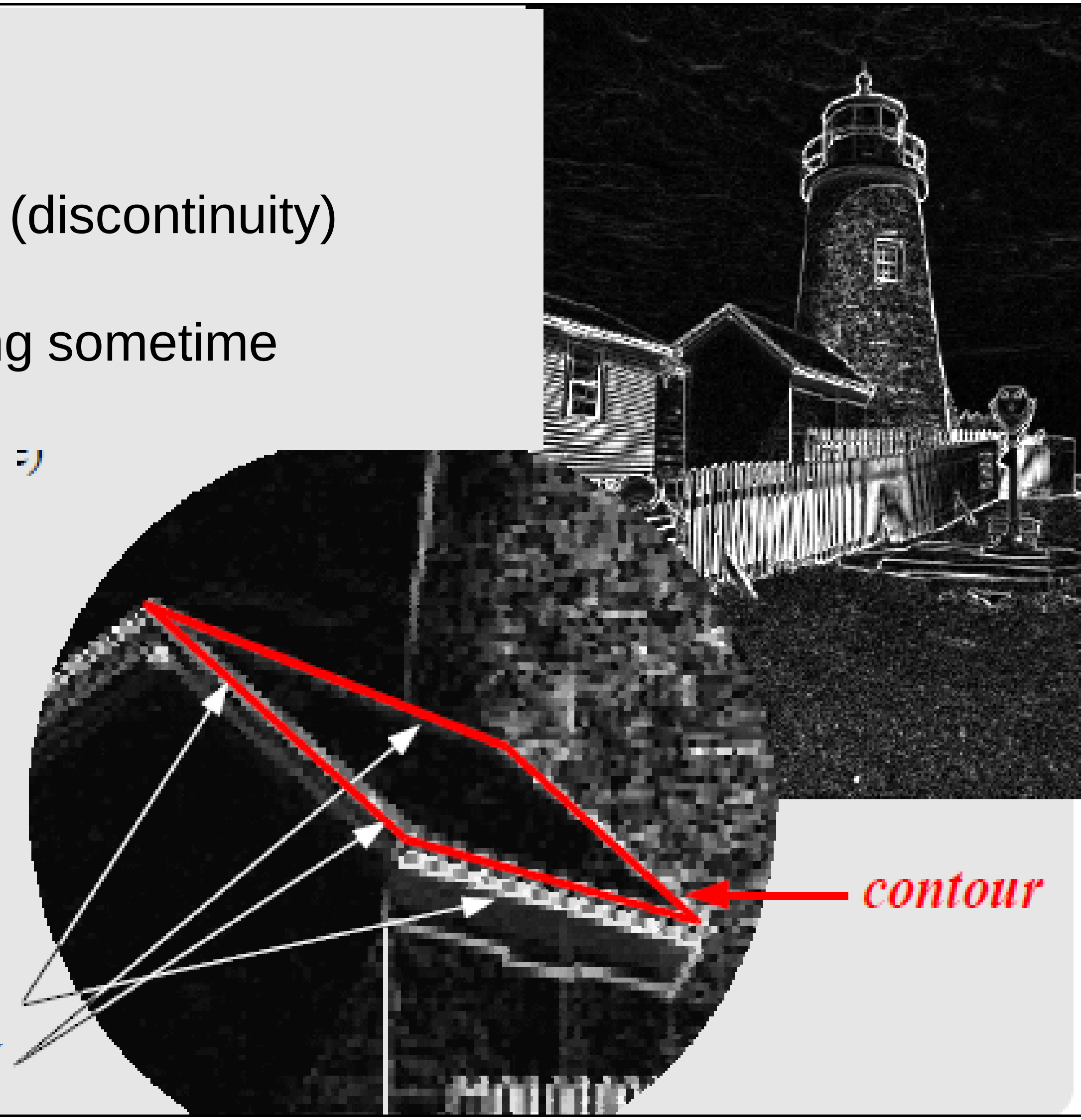


Notion of contours

Contour extraction

- **Candidate point detection**
  - According specific properties (discontinuity)
  - With a part of uncertainty
  - Disturbed by noise (smoothing sometime needed)

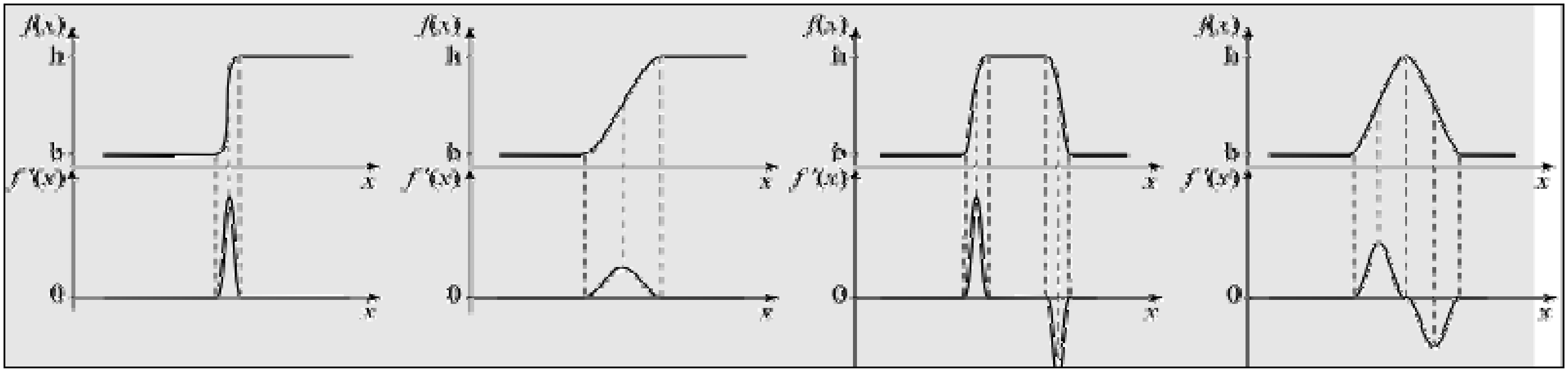
- **Contour construction**
  - Need a binary image
  - To try to link together the contour pixels (connexity analysis)
  - To obtain a real contour (curves, sequence, …)



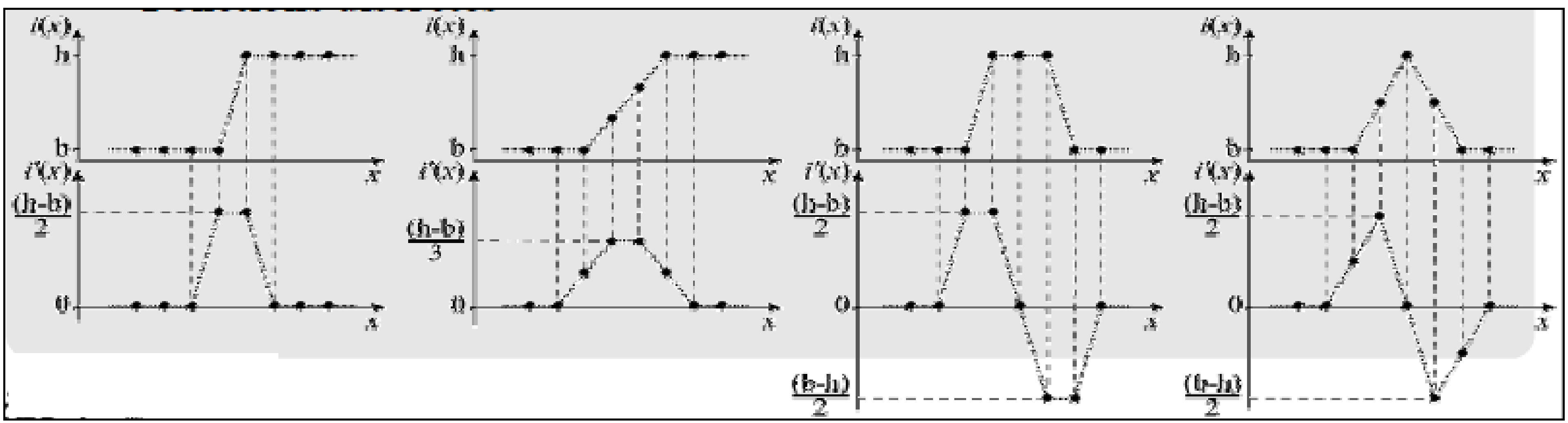
Contour detection

Bringing out contour areas : First derivative

- Continuous functions

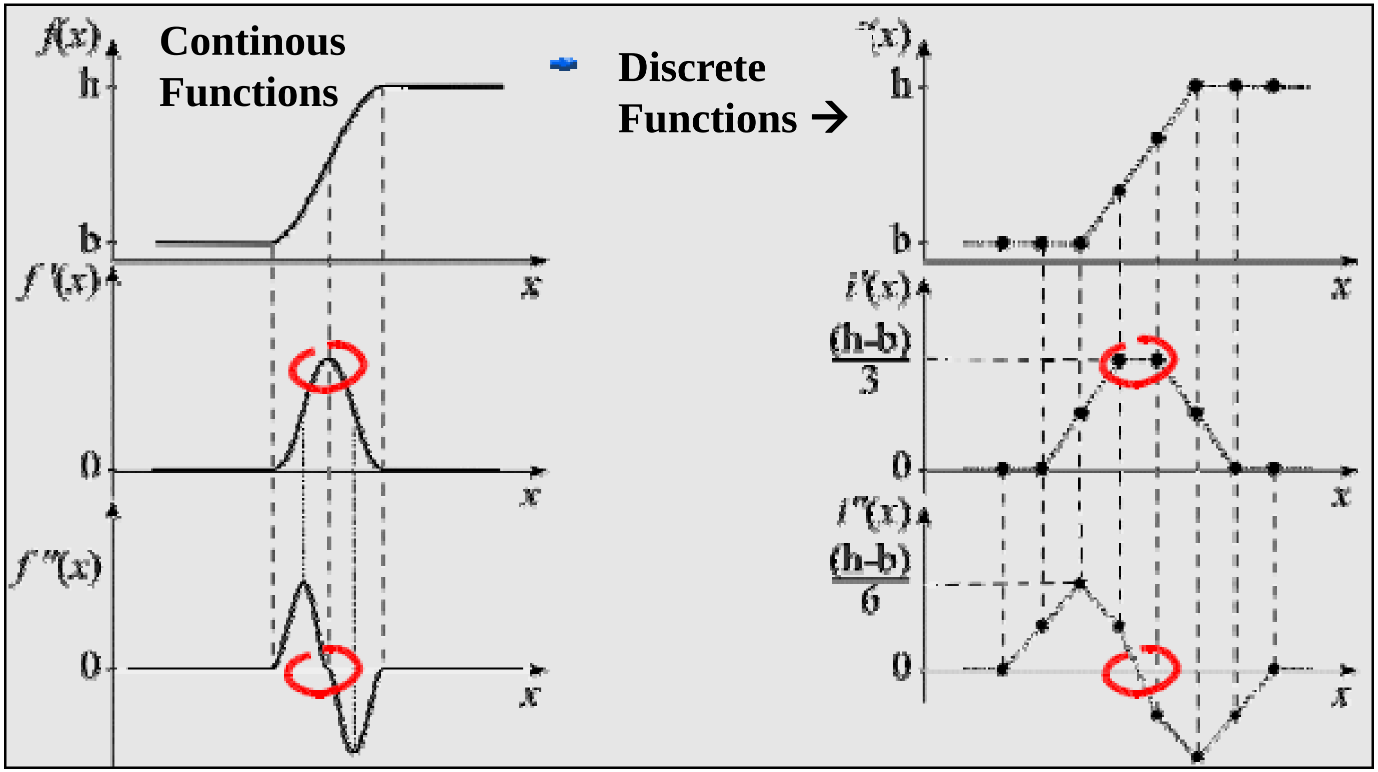


- Discrete functions



Contour detection

Bringing out contour areas : Second derivative



Contour point detection : utilization of a selection criteria

- First derivative : local maxima
- Second derivative : zero crossing

Contour detection & Gradient

Notion of gradient : First Derivative in 2D

- *The (discrete)* image *I* is defined as a set of quantified point of an underlying bi dimensional function *f*(*x*,*y*)

2D derivative of the underlying function

- We can compute a (partial) derivative of *f* in each principal directions ➔

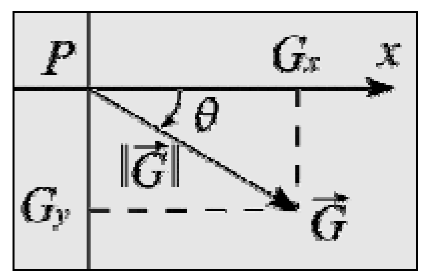
$$\frac{\partial f(x,y)}{\partial x} \quad \text{et} \quad \frac{\partial f(x,y)}{\partial y}$$

- Their combination corresponds to the gradient, with 2 dimensions ➔

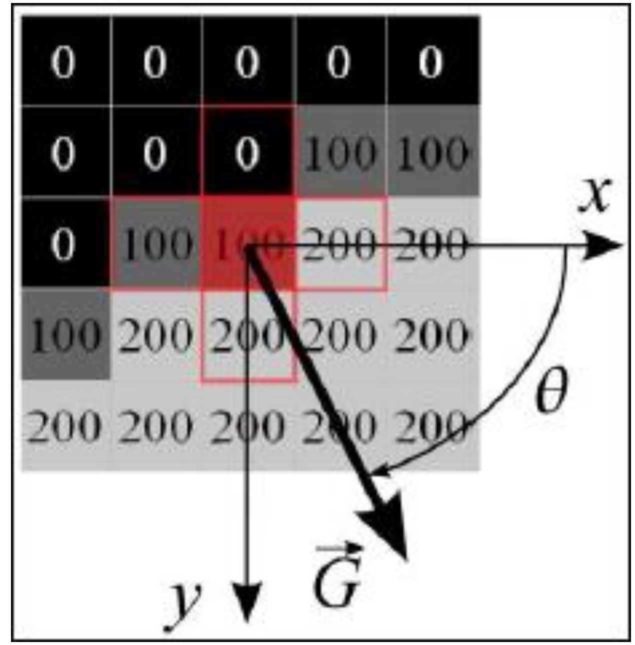
$$\vec{\nabla} f(x,y) = \begin{pmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{pmatrix}$$

$$\vec{G}(x,y) = \begin{pmatrix} G_x(x,y) \\ G_y(x,y) \end{pmatrix}$$

- This vector is characterized, in each point P, by
  - its norm (or module) $\|G\| = \text{sqrt}(Gx^2 + Gy^2)$
  - its direction $\Theta = \arctan(Gy / Gx)$



Contour detection & Gradient



First derivative in 2D (discrete)

Main properties of the gradient

- The magnitude of the vector represents the slope of the surface in the image around *P*
- High magnitude = strong variation around *P*
- The direction of the vector corresponds to the direction of the strongest slope in *P*
- The vector is oriented in the direction of the ascendant slope (ascendant order of gray level)

Relations between Gradient and contour

- Contour = important variation of gray levels
- The gradient vector is perpendicular with the contour

Dérivées premières : $G_x = 50,\ G_y = 100$

Norme du gradient : $\|\vec{G}\| = \sqrt{G_x^2 + G_y^2} = 112$

Autres formules parfois utilisées (plus simples) :

$\|\vec{G}\| = |G_x| + |G_y| = 150$ en norme L_1

$\|\vec{G}\| = max(|G_x|, |G_y|) = 100$ en norme L_∞

Direction du gradient : $\theta = \arctan(G_y / G_x) = 63°$

Associated masks :

$G_x = \frac{\partial f}{\partial x}$: $\frac{1}{2}$

+1	0	-1

$G_y = \frac{\partial f}{\partial y}$: $\frac{1}{2}$

+1
0
-1

Contour detection & Gradient

Principles of the smoothing+derivative masks

- The noise effects are amplified during derivation
- Necessity of a smoothing of the image
  - Neither, Pre-processing before the derivative
  - Or, at the same time, during the derivative
- Simultaneous smoothing and derivative :
  - Principle : smoothing in a perpendicular orientation from the derivative
    - Average in columns of the derivative computed on the lines;
    - Average in lines of the derivative computed on the column.
  - Smoothing/derivative filters can be constructed, less sensitive to noise :

$$\frac{1}{3}\begin{bmatrix}+1\\+1\\+1\end{bmatrix} \quad \frac{1}{2}\begin{bmatrix}+1 & 0 & -1\end{bmatrix} \rightarrow \frac{1}{6}\begin{bmatrix}+1 & 0 & -1\\+1 & 0 & -1\\+1 & 0 & -1\end{bmatrix}$$

  - Several such models have been defined : Prewitt, Sobel, …

Contour detection & Gradient

Prewitt filter: mean/derivative

- Vertical component of the gradient *Gx*

$$\frac{1}{3}\begin{bmatrix}+1\\+1\\+1\end{bmatrix}\quad\frac{1}{2}\begin{bmatrix}+1 & 0 & -1\end{bmatrix}\rightarrow\frac{1}{6}\begin{bmatrix}+1 & 0 & -1\\+1 & 0 & -1\\+1 & 0 & -1\end{bmatrix}$$

- Horizontal component of the gradient *Gy*

$$\frac{1}{3}\begin{bmatrix}+1 & +1 & +1\end{bmatrix}\quad\frac{1}{2}\begin{bmatrix}+1\\0\\-1\end{bmatrix}\rightarrow\frac{1}{6}\begin{bmatrix}+1 & +1 & +1\\0 & 0 & 0\\-1 & -1 & -1\end{bmatrix}$$

Sobel filter : Gaussian/derivative

- Vertical component of the gradient *Gx*

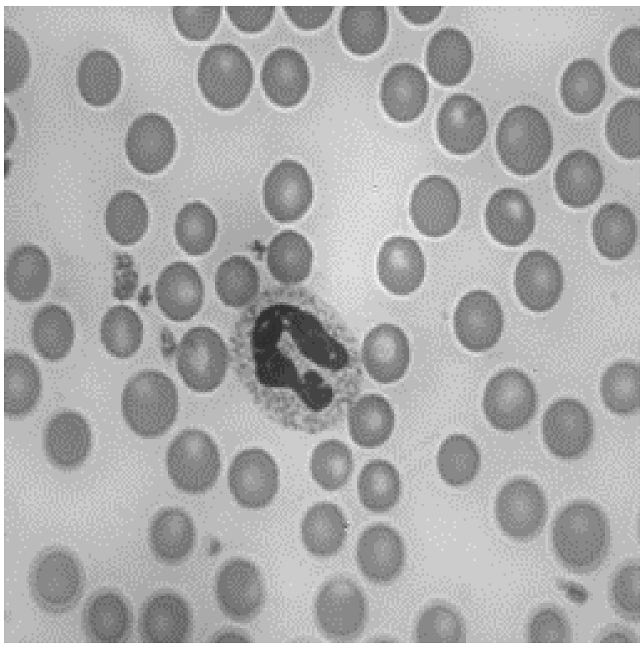
$$\frac{1}{4}\begin{bmatrix}+1\\+2\\+1\end{bmatrix}\quad\frac{1}{2}\begin{bmatrix}+1 & 0 & -1\end{bmatrix}\rightarrow\frac{1}{8}\begin{bmatrix}+1 & 0 & -1\\+2 & 0 & -2\\+1 & 0 & -1\end{bmatrix}$$

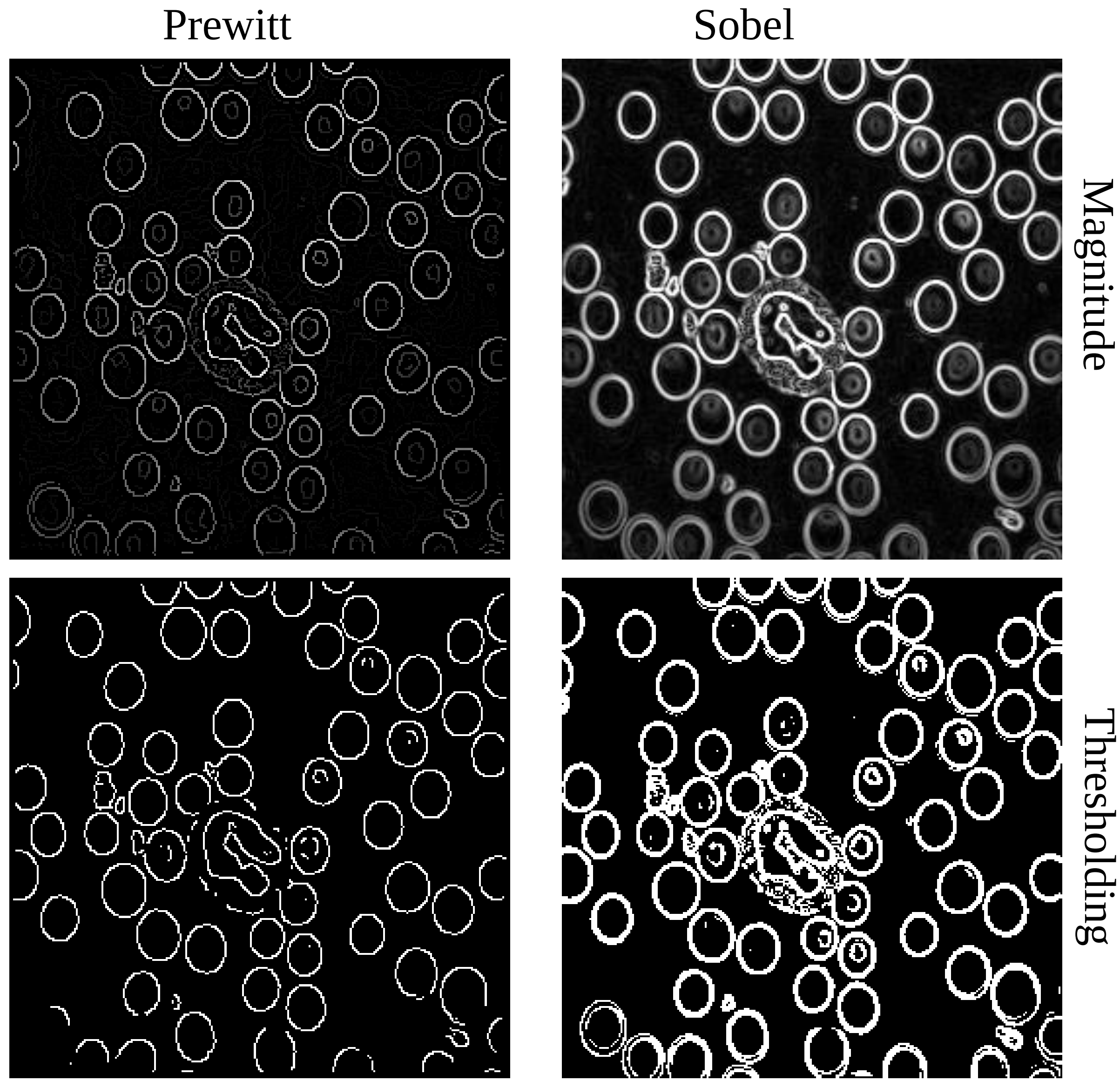
- Horizontal component of the gradient *Gy*

$$\frac{1}{4}\begin{bmatrix}+1 & +2 & +1\end{bmatrix}\quad\frac{1}{2}\begin{bmatrix}+1\\0\\-1\end{bmatrix}\rightarrow\frac{1}{8}\begin{bmatrix}+1 & +2 & +1\\0 & 0 & 0\\-1 & -2 & -1\end{bmatrix}$$

Contour detection & Gradient

Contour detection & Gradient





Contour detection & Laplacian

Definition of Laplacian (second derivative)

- Le Laplacian is defined by : $$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$
- It is a scalar signed value (not a sectorial on as the gradient)

Masques associés aux dérivées secondes :

pour $\frac{\partial^2}{\partial x^2}$: $\frac{1}{4}$

+1	-2	+1

pour $\frac{\partial^2}{\partial y^2}$: $\frac{1}{4}$

+1
-2
+1

Masques alternatifs (dérivées calculées sur les axes à 45°) :

pour $\frac{\partial^2}{\partial X^2}$: $\frac{1}{4}$

0	0	+1
0	-2	0
+1	0	0

pour $\frac{\partial^2}{\partial Y^2}$: $\frac{1}{4}$

+1	0	0
0	-2	0
0	0	+1

Approximations discrètes du Laplacien Δ :

$\frac{1}{8}$

0	+1	0
+1	-4	+1
0	+1	0

ou $\frac{1}{8}$

+1	0	+1
0	-4	0
+1	0	+1

ou encore $\frac{1}{16}$

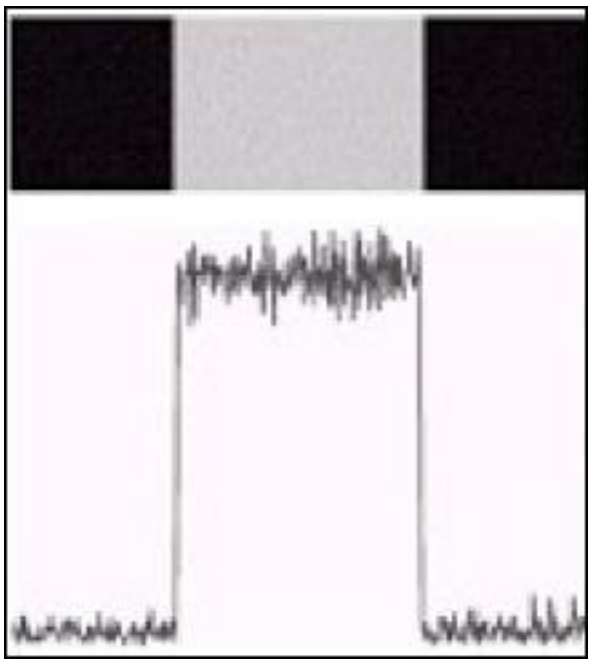
+1	+1	+1
+1	-8	+1
+1	+1	+1

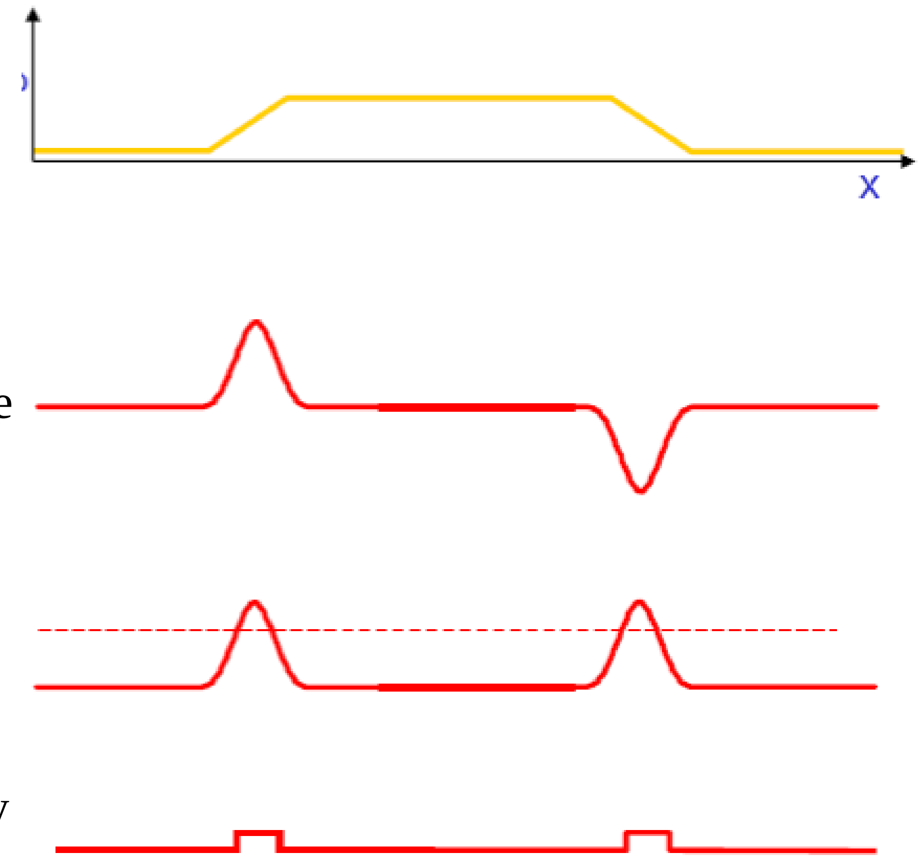
Contour detection & Laplacian

Sobel Laplacian

Contour detection & Thresholding

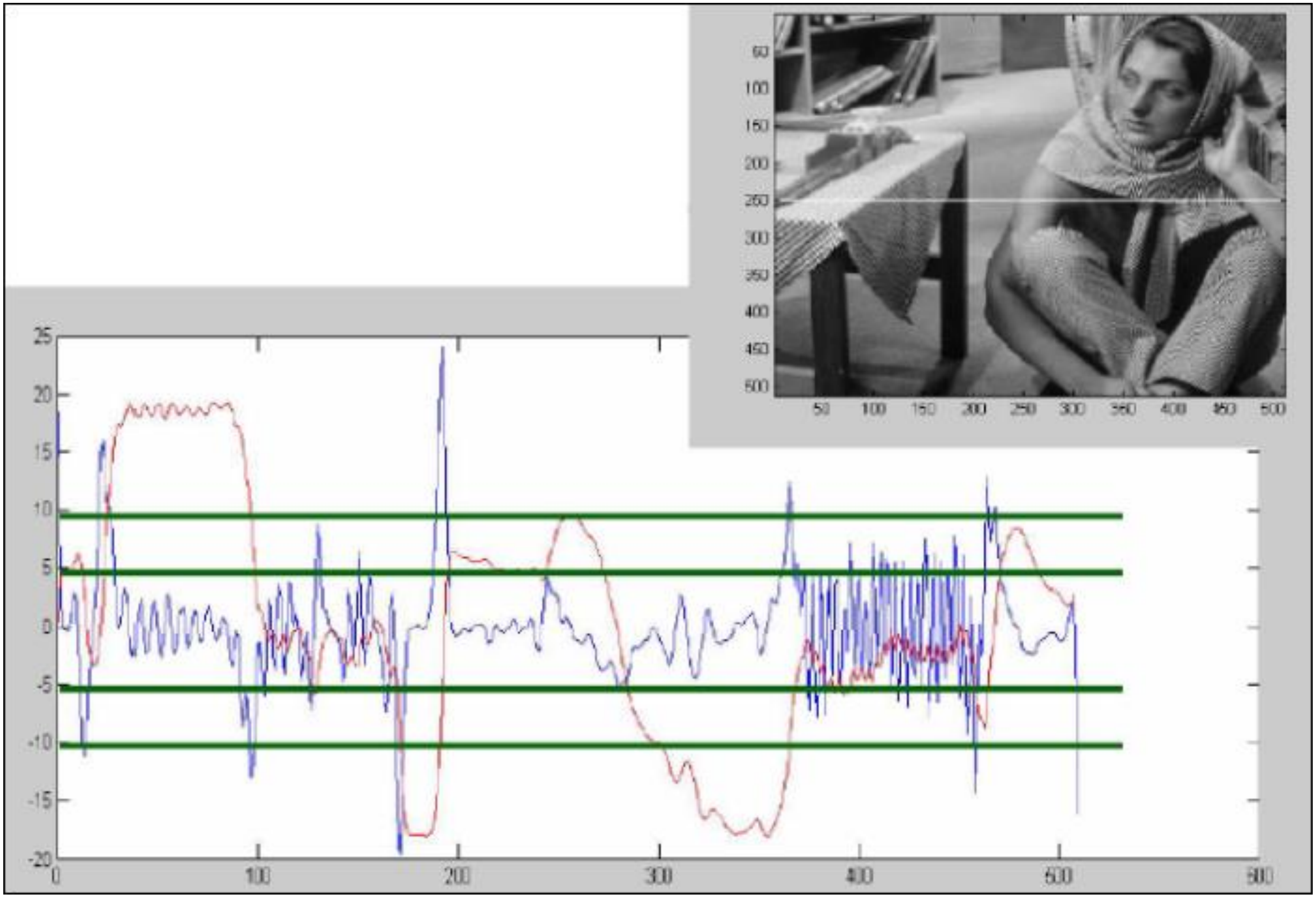
- Noise vs real Contours ➔ Local vs Global extrema





Contour detection and noise…

What about the choice of the local maxima to keep?



Optimal filtering: Canny

Filtering in several steps (not only one convolution)

Given :

- A contour model (step)
- A noise model (Gaussian white noise)

Cauterization of the performances in terms of :

- detection (mainly for low contours)
- localization (precision of the position)

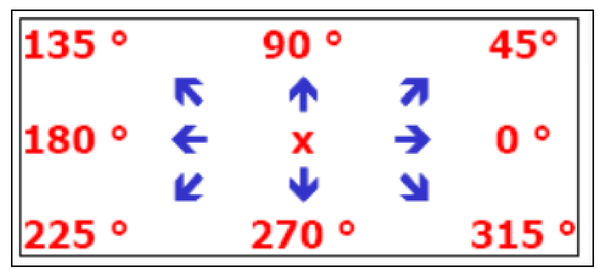
1. Apply a Gaussian filtering to remove noise

2. Compute gradient with Sobel in X and Y

- compute gradient magnitude $|G| = |Gx| + |Gy|$
-

3. Compute directions of the gradient

- Direction of gradient $\theta = \arctan(Gy / Gx)$
- Round the directions with multiples of $\pi/4$



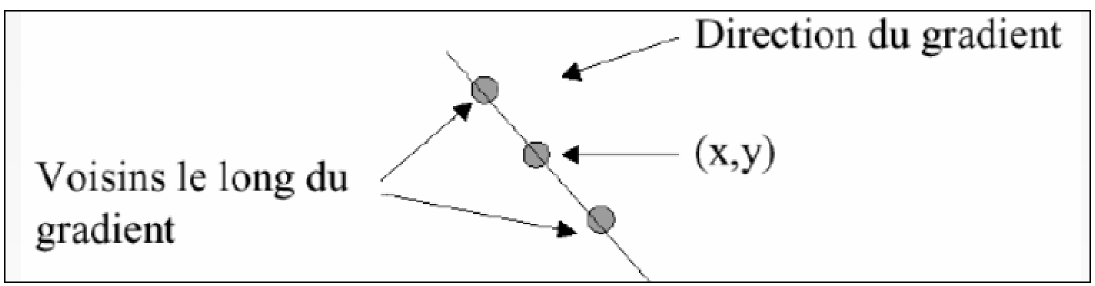
Optimal filtering: Canny

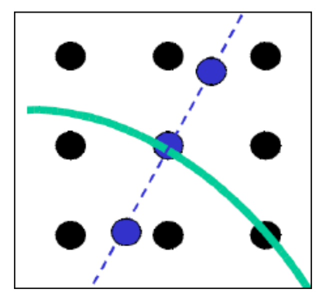
4. Removal of non-maxima:

- IF the magnitude of the gradient at the pixel (x, y) is lower than the magnitude of one of its 2 neighbors along the gradient direction, THEN set the magnitude of the pixel (x, y) to zero.

5. Thresholding of the contours (hysteresis) :

- Need 2 thresholds : One high Sh & one low Sb.
- For each pixel, analyse the magnitude of the gradient :
    1. IF magnitude(x, y) < Sb THEN put pixel to 0 (∈ / contour)
    2. IF magnitude(x, y) > Sh THEN set pixel ∈ contour
    3. IF Sb ≤ magnitude(x, y) ≤ Sh AND pixel is connoted to another pixel already accepted as contour THEN set pixel ∈ contour.





Contour detection, summary

- The detection of contour points is based on derivative computation (gradient or Laplacian) of the image
- The computation of the derivative is approximated using convolution masks
  - Advantages : very fast, very local.
  - Drawbacks :very sensitive to noise (Laplacian). Need denoising before
- The smoothing+derivative filter are less sensitive but also less precise

- All these filters allow only to estimate the probability for each pixel to be part of a contours (candidate points)

- Then, we have to :
  - Decide if either the pixel is really part of a contour or not, according a threshold for example
  - Analyze the selected point to build real contour chains

Region based approach

- Question : How to agglomerate pixels into regions ?
- Question : How to split images into regions ?

Region Growing

Algorithm:

- Select seed points
- Examines neighboring pixels of initial seed points
- Determines whether the pixel neighbors should be added to the region
- The process is iterated on

Seed *Growing* *Final region*

The Similarity threshold value

- The criteria of similarities or so called homogeneity are important. It usually depends on the original image and the segmentation result we want.
- Some criteria often used are grayscale (average intensity or variance), color, and texture or shape.

Region Growing

The suitable selection of seed points is important

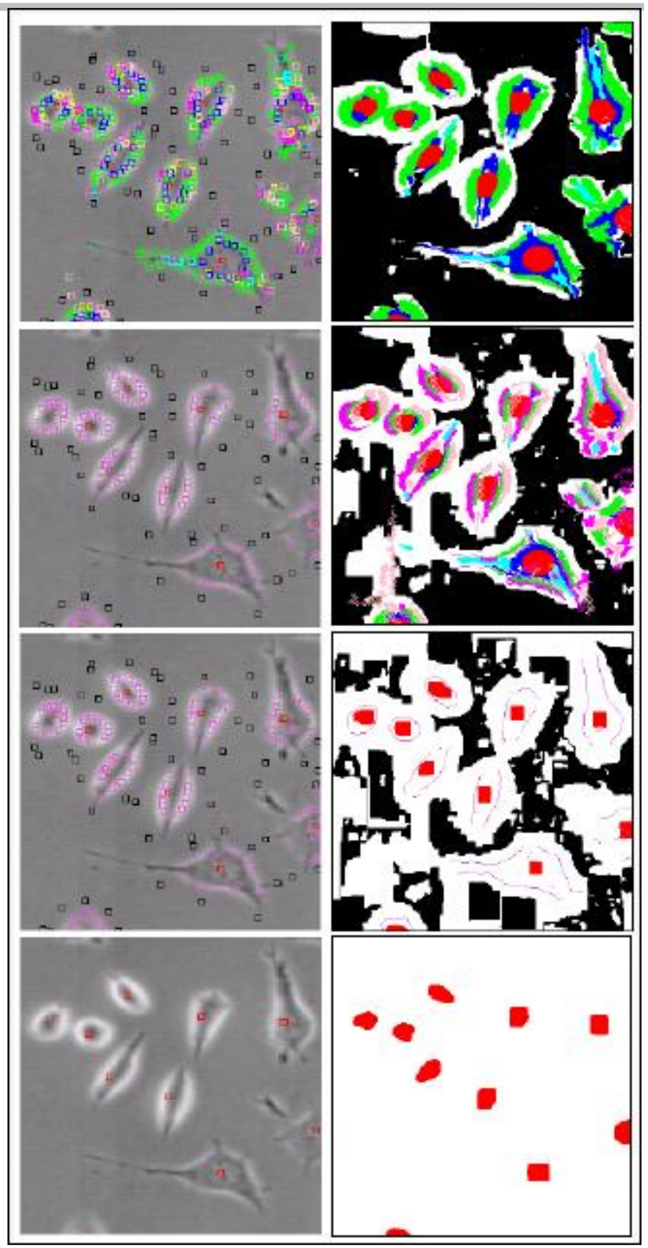
- The selection of seed points is depending on the users
- Some automatic/random selection methods can be used (avoiding high gradient pixels)

More information on the image is better

- Number of seed points?
- Homogeneity criteria?.

A minimum area threshold

- No region should be smaller than this threshold in the segmented image



Region Growing

Advantages

- Simple
- We can determine the seed points and the criteria we want to use
- We can choose the multiple criteria at the same time

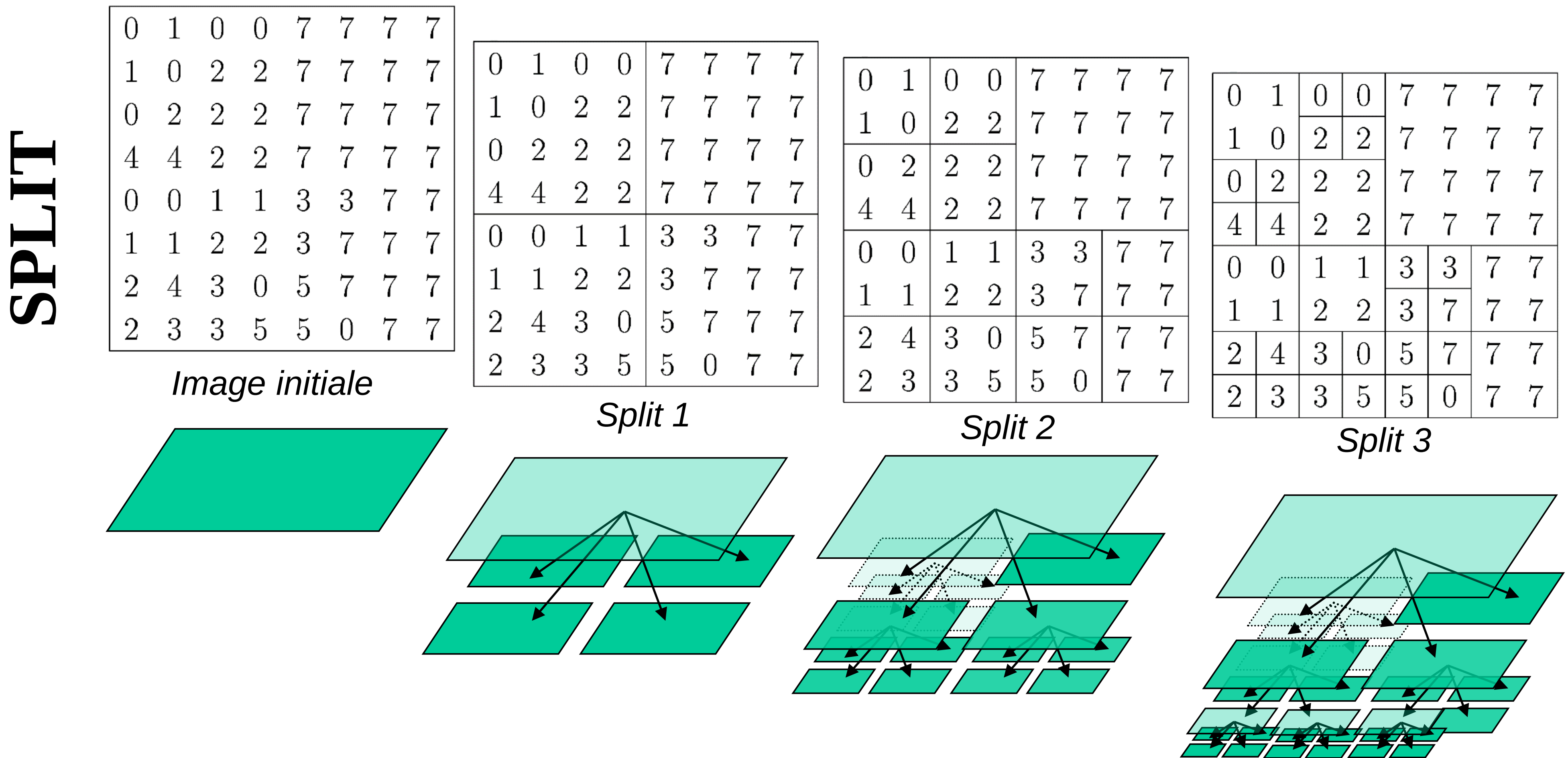
Disadvantages

- It is a local method with no global view of the problem
- Sensitive to noise
- Seed dependent
- Computationally expensive
- A continuous path of points a continuous small color gradient may exist which connects any two points in the image

Split and Merge

Principle :

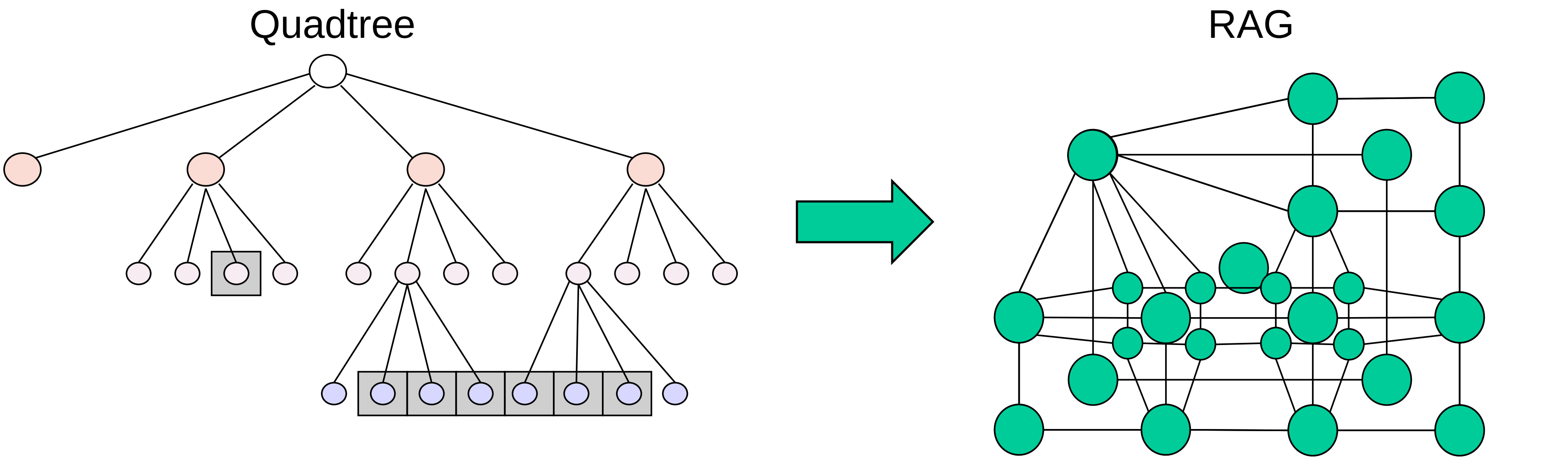
- Divide recursively the image into 4 sub-regions (SPLIT)
- Merge homogenous adjacent regions (MERGE)



Split and Merge

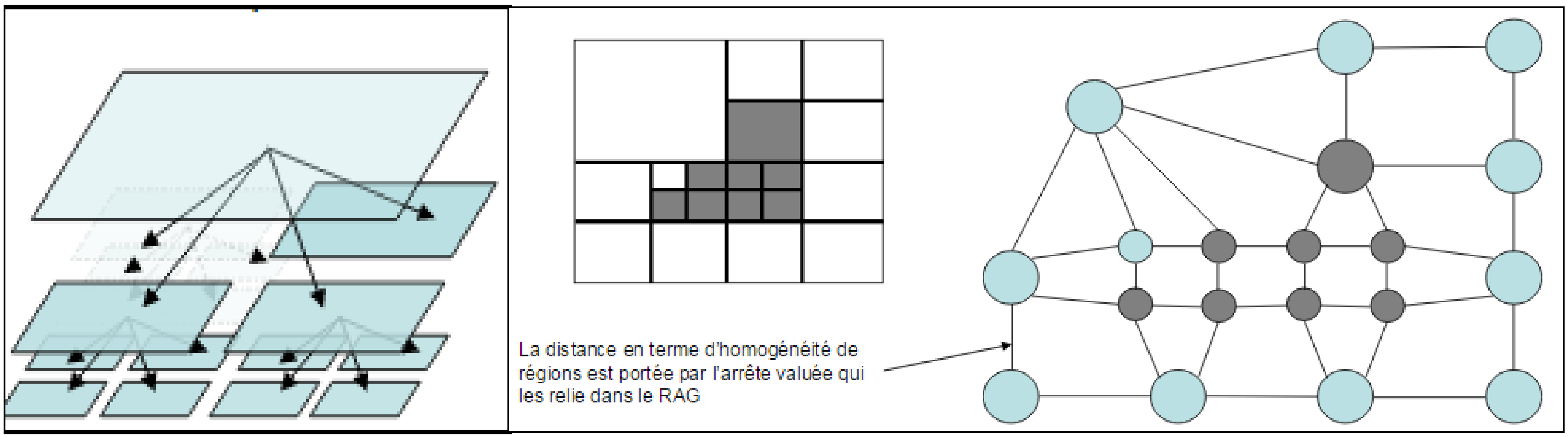
MERGE

- A Region Adjacent Graph is constructed
- Vertices attributes : homogeneity measure between graph
- Iterative merging of the nodes



Split and Merge

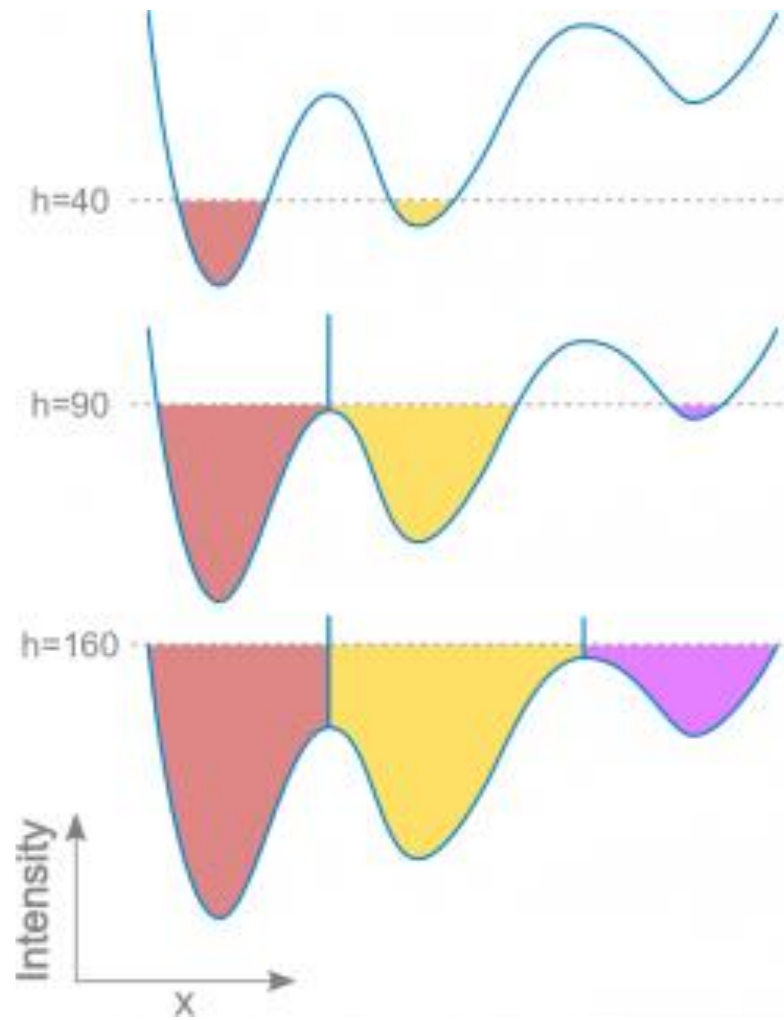
- It is the dual approach of the region growing method
- Time consuming



Classical Watershed

Watershed by flooding

- Considering the input image as topographic surface
- Placing a water source in each regional minimum of its relief
- Flooding from the sources → Dams are placed where the different water sources meet
- The watersheds are the zones dividing adjacent catchment basins
- The first image points that are reached by water are the points at the lowest grayscale value *h{min}*
- All image pixels are progressively reached up to the highest level *h{max}*



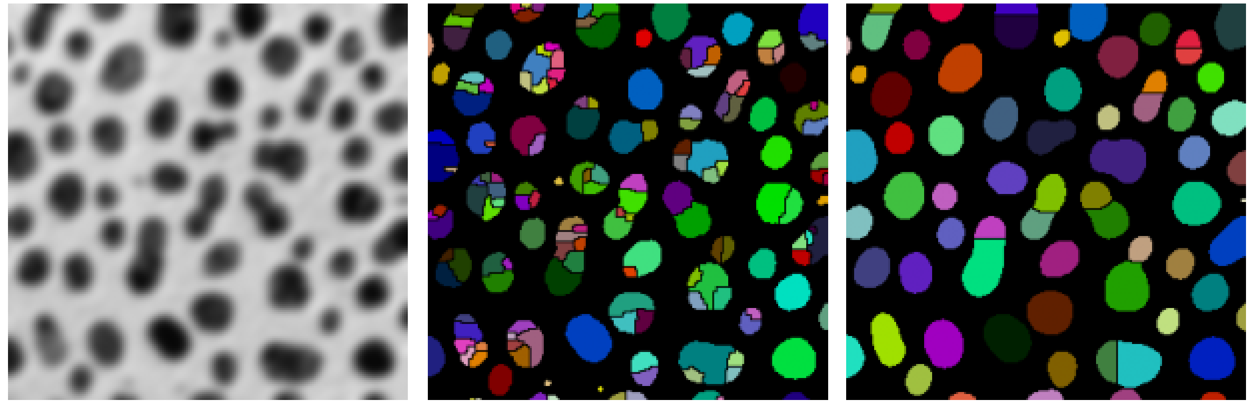
Classical Watershed

Advantages

- High precision

Drawbacks

- High memory consumption
- Sensitive to noise
- Over-segmentation when many regional minima exist ➔ Post processing needed



Gaussian-blurred blobs image used as input (radius = 3).

Watershed segmentation on original image (Min h = 0, Max h = 150)

Watershed segmentation on Gaussian-blurred original image (radius = 3, Min h = 0, Max h = 150)

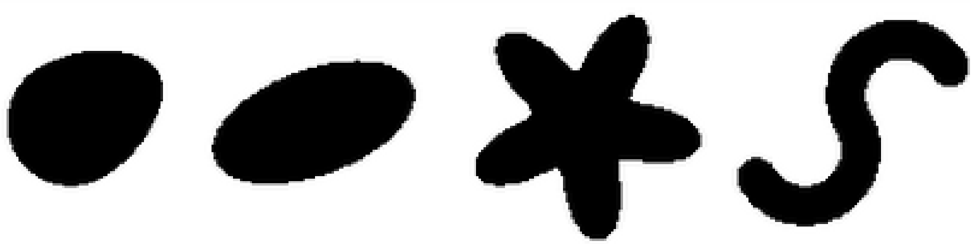
Binary Watershed ➔ Cf Binary Image

Binary Image Processing

Why so much attention?

- **Problem simplification (0 & 1)**
  - Background = white / Shapes = black
  - 1 object ➔ 1 region / 1 contour ➔ 1 black component (blob)
  - Analysis, characterization of objects from B&W data
  - Counting, measure, shape analysis are easier

- **Mathematic morphology and discrete geometry**
  - Many possible operations on binary images

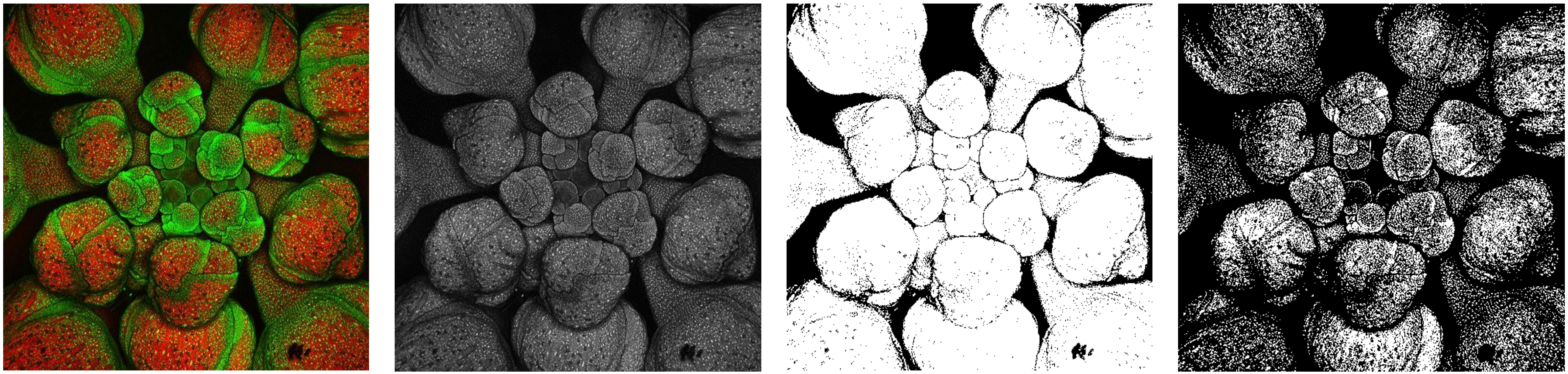


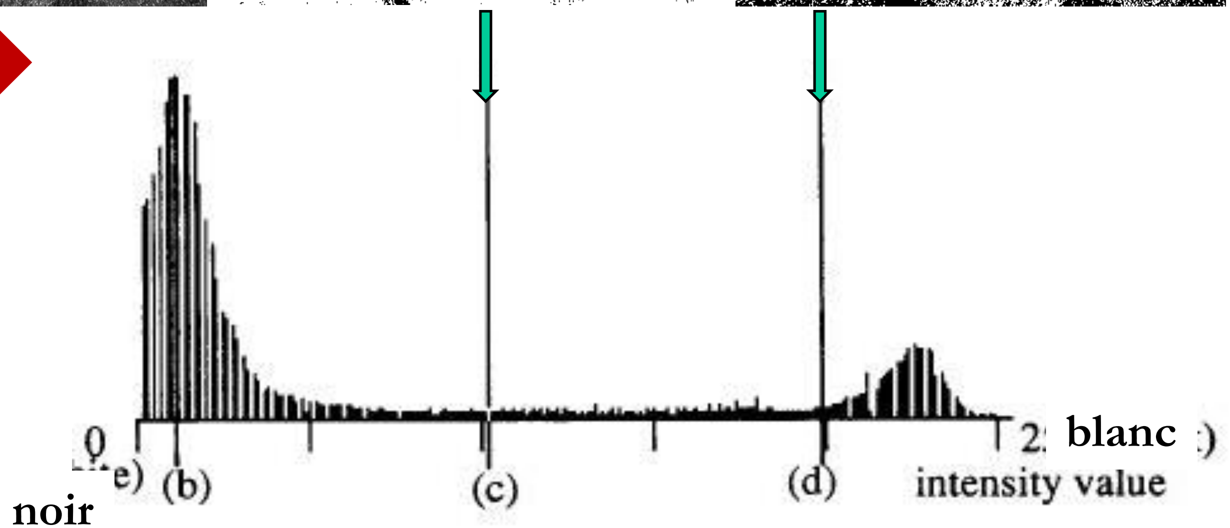
Binarisation = Thresholding

- **Manual Thresholding ➔** The easiest and the most often used
- There is a relation between the gray level and the probability of belonging to a shape

Binarisation with a global threshold

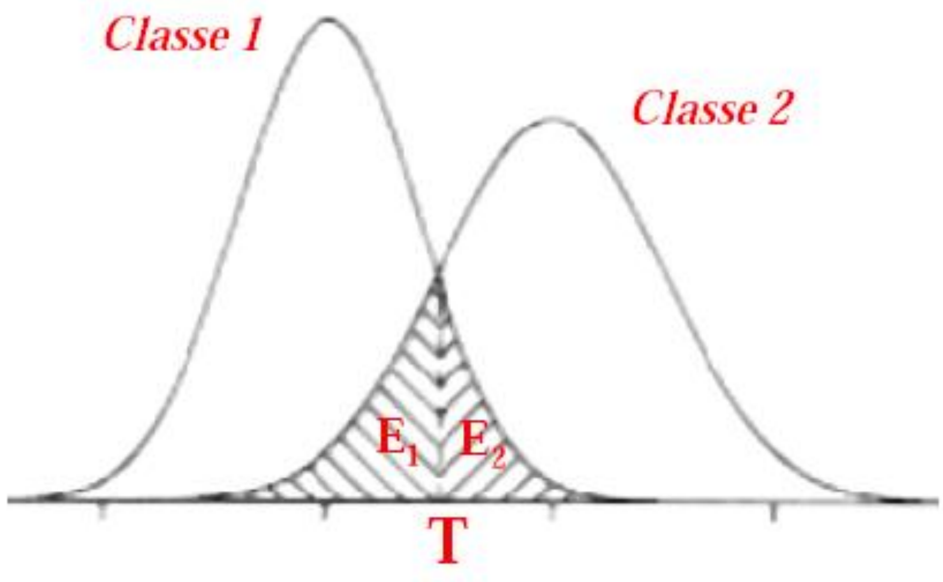
- How to choose a good threshold ➔ so many methods
  - It depends of the images and the objectives …





Automatic thresholding: OTSU

- Knowing there are only 2 classes of color in the image
  - We assume that the gray level distribution is composed by 2 Gaussians
  - We look for a threshold *t* that will minimize the intra-class variance



Automatic thresholding: OTSU

- For each possible value of ***t***, the intra class variance is computed

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

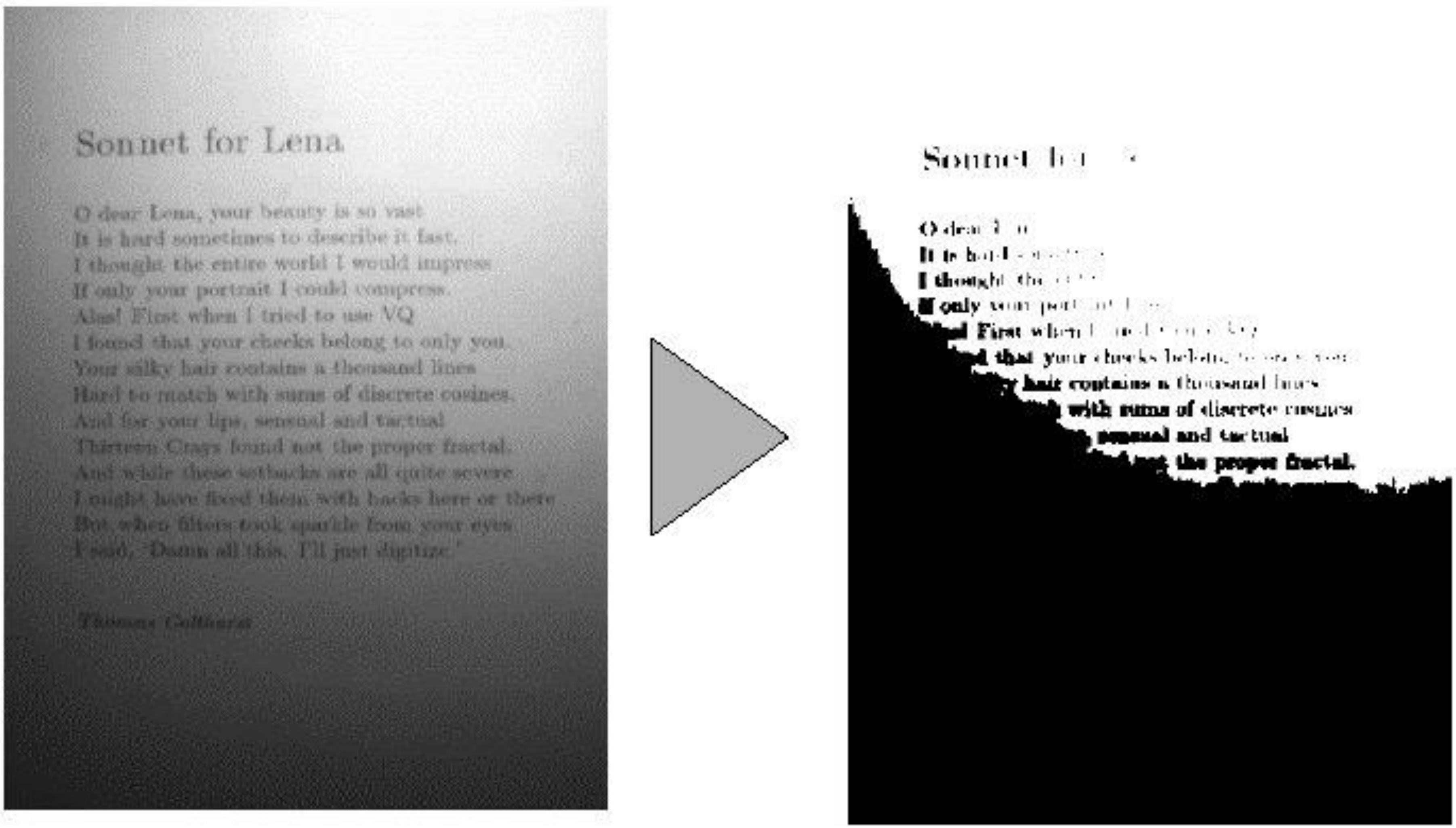
- The optimal threshold is the one for which σ^2_w is minimum
  - $\boldsymbol{w_i}$ is the probability (weight) of the class *i*
  - σ^2_i is the variance of the gray levels of the class *i*
- A more efficiant formulation uses the inter-class variance σ^2_b
  - probability and mean of the classes can be updated iteratively

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

1. Calculer l'histogramme et les probabilités de chaque niveau d'intensité
2. Définir les $\omega_i(0)$ et $\mu_i(0)$ initiaux
3. Parcourir tous les seuils possibles $t = 1 \ldots$ intensité max
   1. Mettre à jour ω_i et μ_i
   2. Calculer $\sigma_b^2(t)$
4. Le seuil désiré correspond au $\sigma_b^2(t)$ maximum.

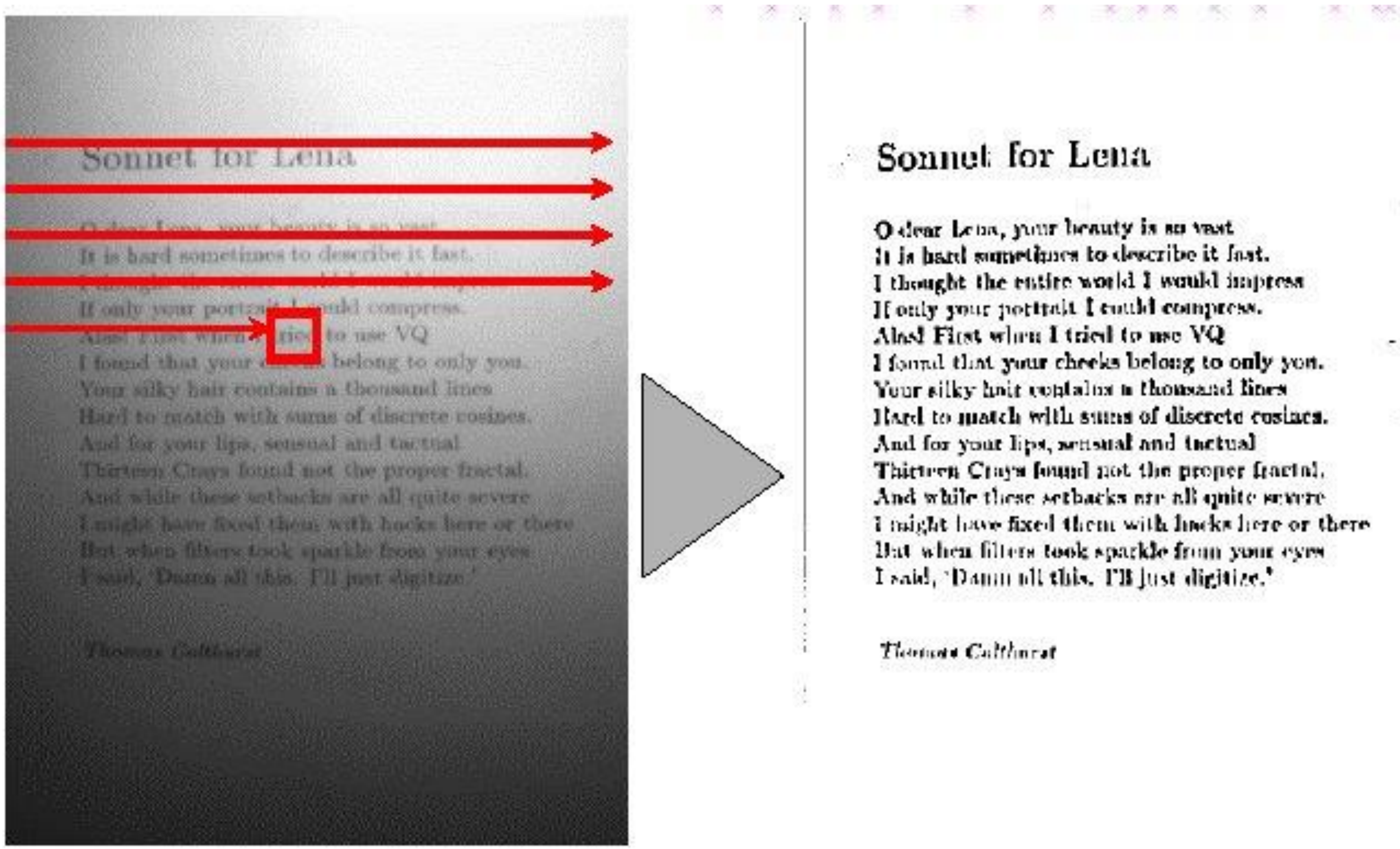
Problem with Global Thresholding

- **A solution ?**



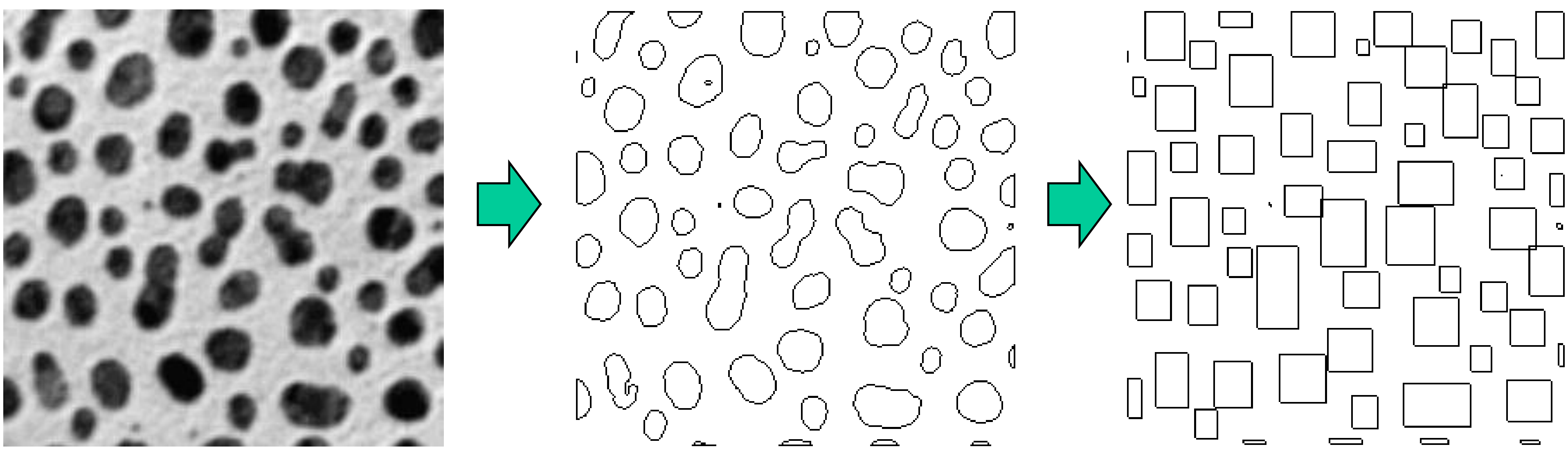
Adaptive local Thresholding

- A local threshold is defined and used for each pixel according to it neighborhood (sometime difficult to select)
- **Niblack :** $S = m + ks^2$ avec k= -0,2 | *m* : mean et s : standard deviation



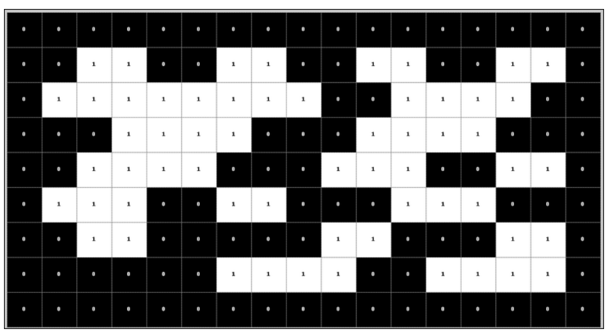
Background-Foreground segmentation

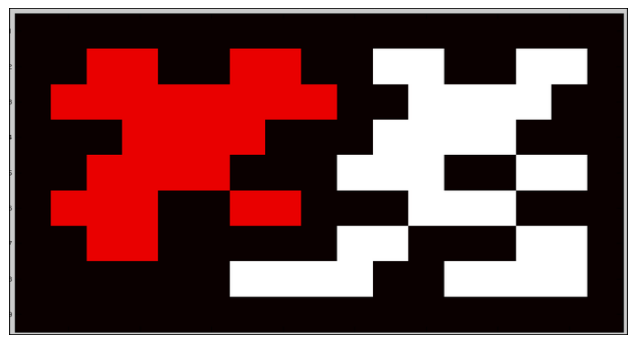
- **How to analyze black shapes on a white background?**
  - Shape localization
  - Counting
  - Describing, charaterizing
  - Classifying



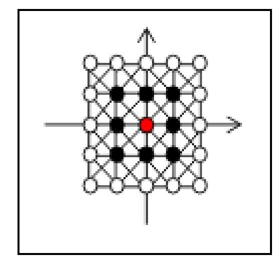
Connected component extraction

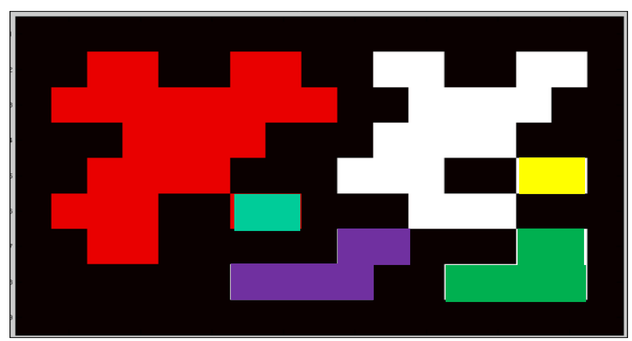
- **Notion of Connected component**
  - A set S of pixels is a 4-connected component if and only if, for all pairs of pixels P and Q, a 4-path $p_1, p_2, p_3, \ldots p_n$ with $p_1 = P$ and $p_n = Q$ and all $p_i \in S$.
  - A set S of pixels is a 8-connected component if and only if, for all pairs of pixels P and Q, a 8-path $p_1, p_2, p_3, \ldots p_n$ with $p_1 = P$ and $p_n = Q$ and all $p_i \in S$.



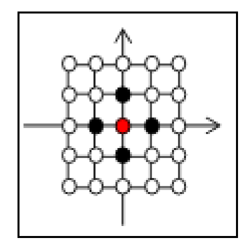


8-connexity





4-connexity



Connected component extraction

- **A two pass algorithm**

```
. . . . . . . . . . . . . . .
. . . * * . . . * * * . . . .
. . . * * . . . * * * . . . .
. . * * * * . * * * * . . . .
. . . * * * * * * * * . . . .
. . . . . * * * * . . . . . .
. . . . . . * * * . . * . . .
. . . . . . * * * . . * * . .
. . * * . . * * * . . * * . .
. . * * . . . . . . . . . . .
. . . . . . . . . . . . . . .
```

(b) Original binary image.

```
. . . . . . . . . . . . . . .
. . . 1 1 . . . 2 2 2 . . . .
. . . 1 1 . . . 2 2 2 . . . .
. . 1 1 1 1 . 2 2 2 2 . . . .
. . . 1 1 1 ? * * * * . . . .
. . . . . * * * * . . . . . .
. . . . . . * * * . . * . . .
. . . . . . * * * . . * * . .
. . * * . . * * * . . * * . .
. . * * . . . . . . . . . . .
. . . . . . . . . . . . . . .
```

(c) Labeling in progress.

```
. . . . . . . . . . . . . . .
. . . 1 1 . . . 2 2 2 . . . .
. . . 1 1 . . . 2 2 2 . . . .
. . 1 1 1 1 . 2 2 2 2 . . . .
. . . 1 1 1 1 1 1 1 1 . . . .
. . . . . 1 1 1 1 . . . . . .
. . . . . . 1 1 1 . . 3 . . .
. . . . . . 1 1 1 . . 3 3 . .
. . 4 4 . . 1 1 1 . . 3 3 . .
. . 4 4 . . . . . . . . . . .
. . . . . . . . . . . . . . .
```

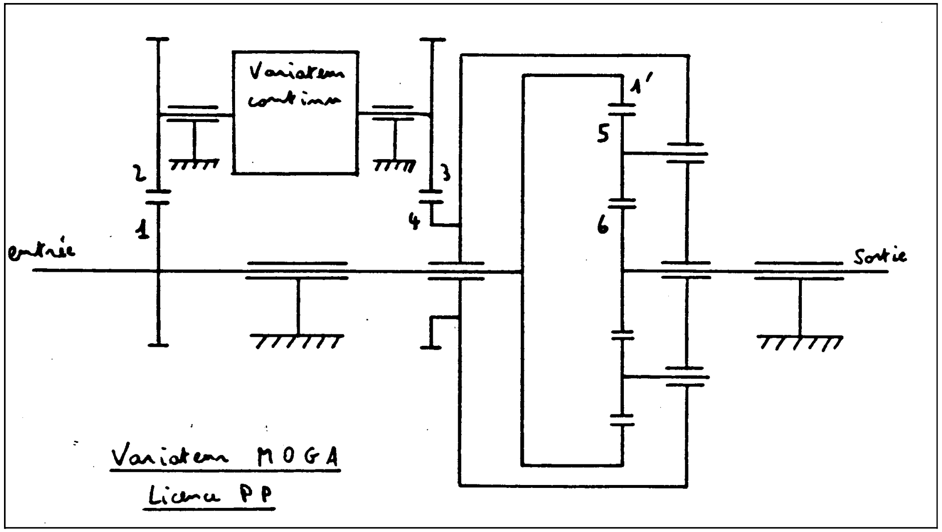
(d) Scanning completion.

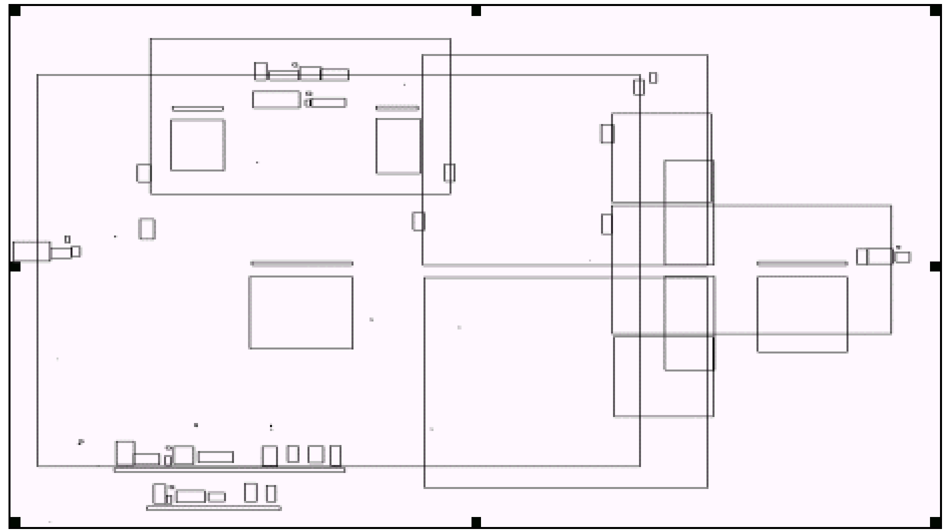
```
. . . . . . . . . . . . . . .
. . . 1 1 . . . 1 1 1 . . . .
. . . 1 1 . . . 1 1 1 . . . .
. . 1 1 1 1 . 1 1 1 1 . . . .
. . . 1 1 1 1 1 1 1 1 . . . .
. . . . . 1 1 1 1 . . . . . .
. . . . . . 1 1 1 . . 2 . . .
. . . . . . 1 1 1 . . 2 2 . .
. . 3 3 . . 1 1 1 . . 2 2 . .
. . 3 3 . . . . . . . . . . .
. . . . . . . . . . . . . . .
```

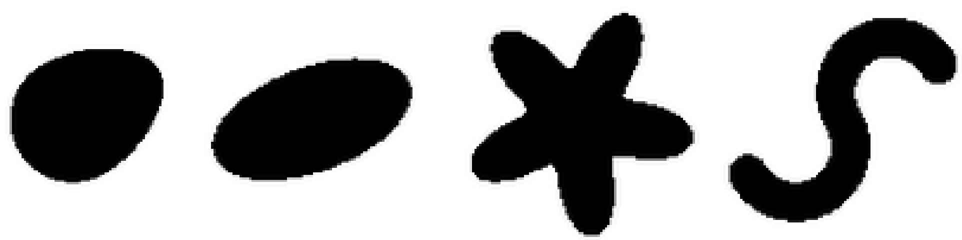
(e) After label unification and reassignment.

Connected component Analysis

- Studying their size and shape



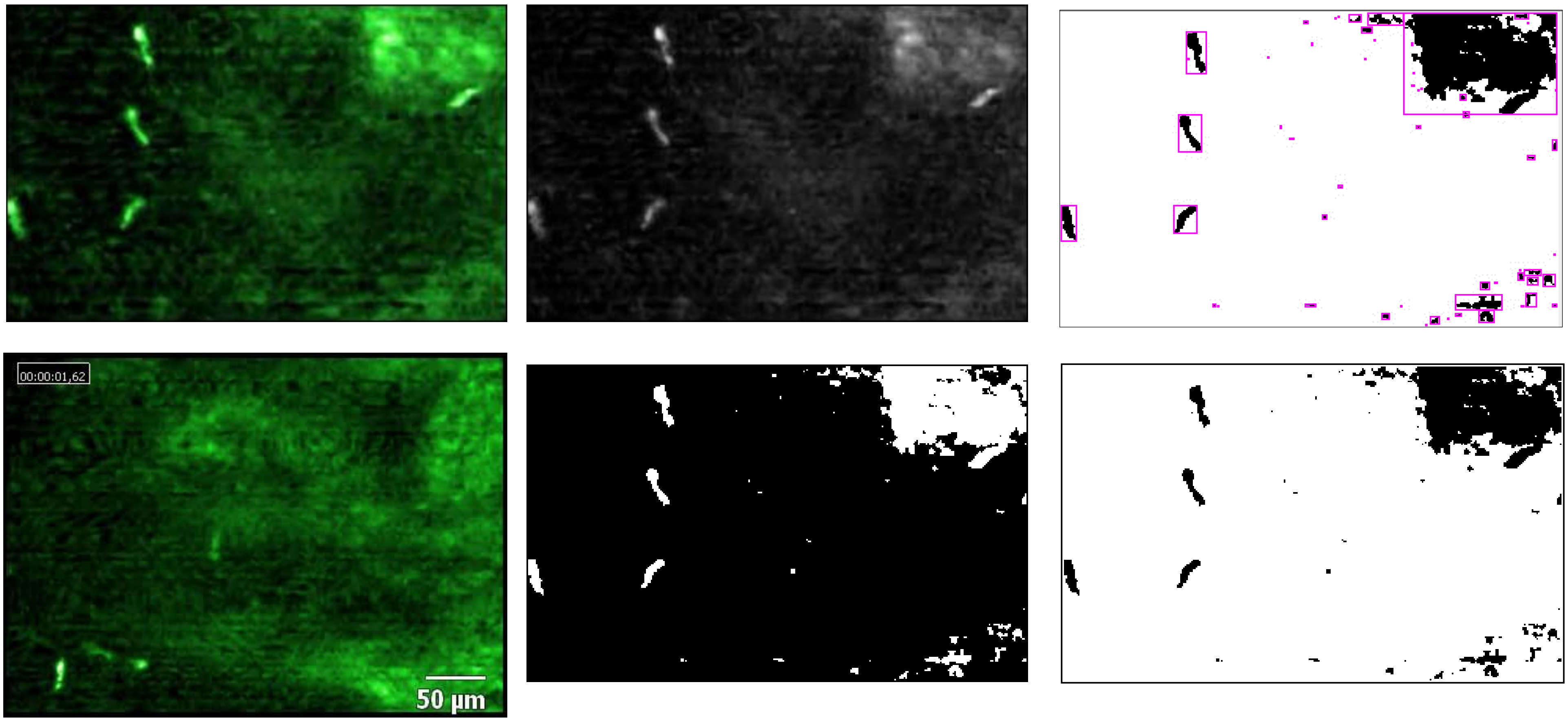




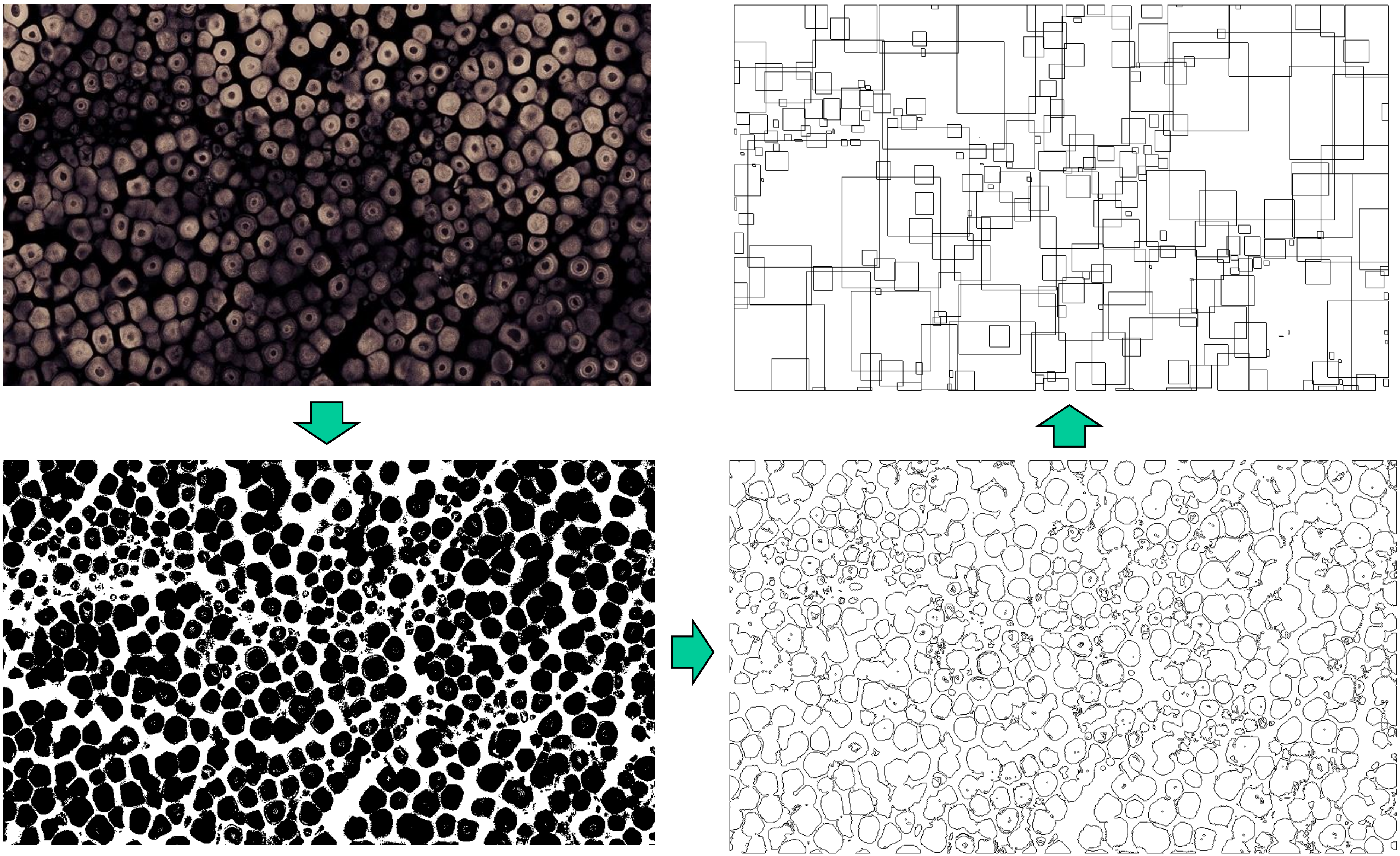
Circularité	0.966	0.834	0.395	0.280
Compacité	0.973	0.961	0.685	0.556
Allongement	1.29	1.99	1.00	2.63
Tortuosité	1.06	1.06	1.04	1.26

Connected component Analysis

- Exploitation of the connected components
  - How ?
  - Image analysis sequence ?



Connected Components and connexity !



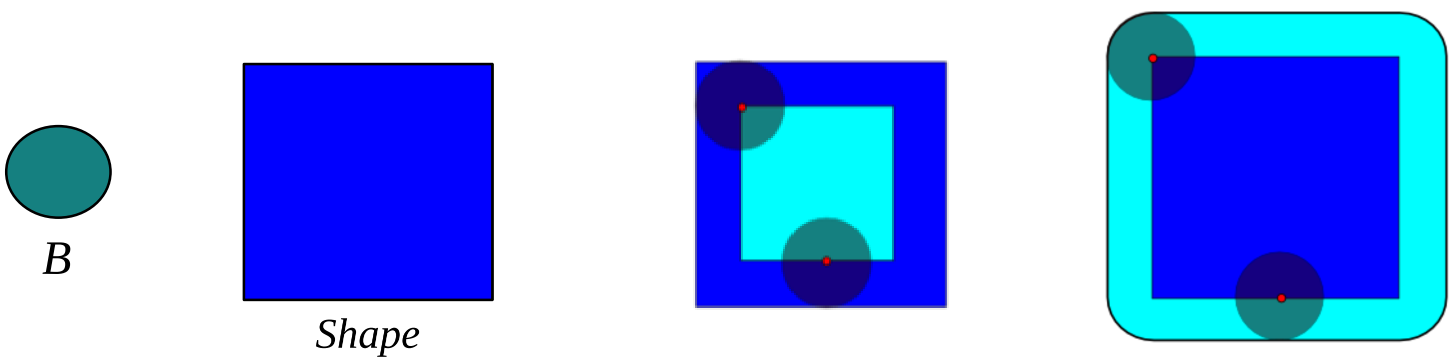
Mathematical / Binary morphology

Binary morphology and structuring element

- The basic idea in binary morphology is to probe an image with a simple, pre-defined shape, drawing conclusions on how this shape fits or misses the shapes in the image.
- This simple "probe" is called the structuring element, and is itself a binary image

Some examples of widely used structuring elements (denoted by *B*):

- In R^2, B is an open disk of radius r, centered at the origin.
- In Z^2, B is a 3x3 square, that is, B={(-1,-1), (-1,0), (-1,1), (0,-1), (0,0), (0,1), (1,-1), (1,0), (1,1)}.
- In Z^2, B is the "cross" given by: B={(-1,0), (0,-1), (0,0), (0,1), (1,0)}.



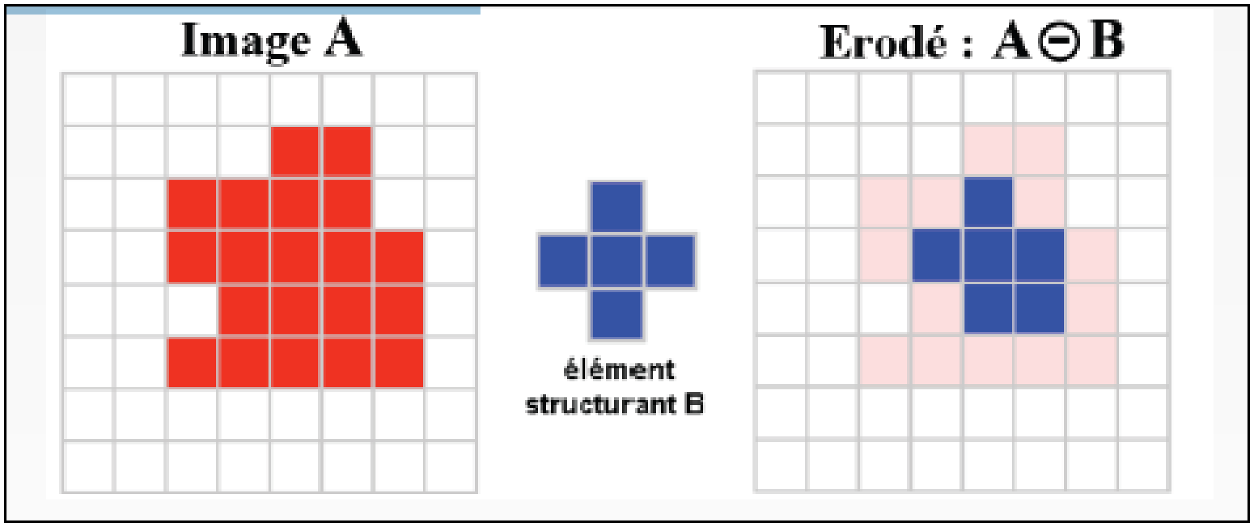
Basic operations: Erosion

Let B_x be the center of the structuring element B that is put on the pixel X of the image

Algorithm :

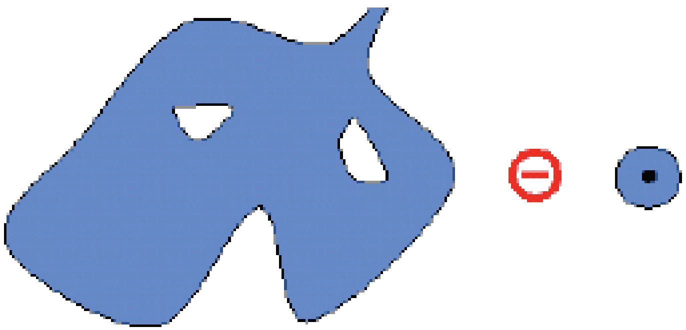
- B_x is positioned on each pixel X of the object A
- IF all pixels of B are inside the object A THEN

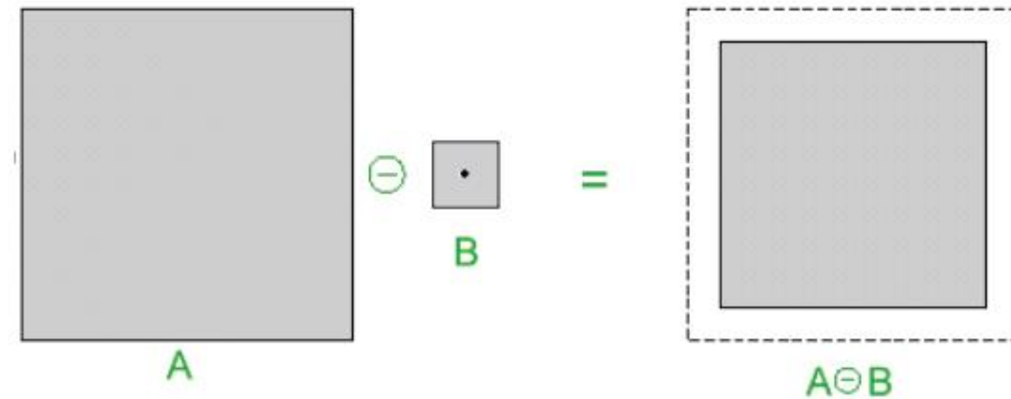
  B_x is set to kept (as part of the eroded object)

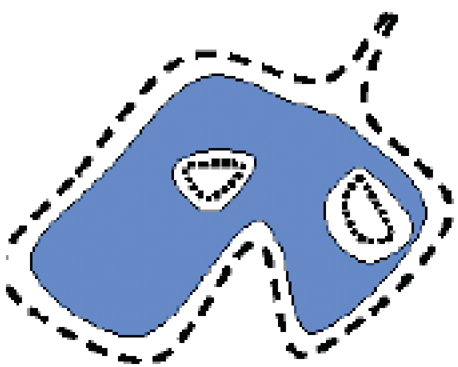


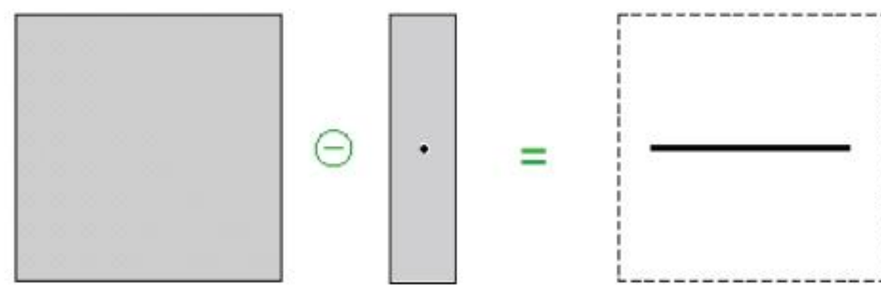
Basic operations: Erosion

- Examples









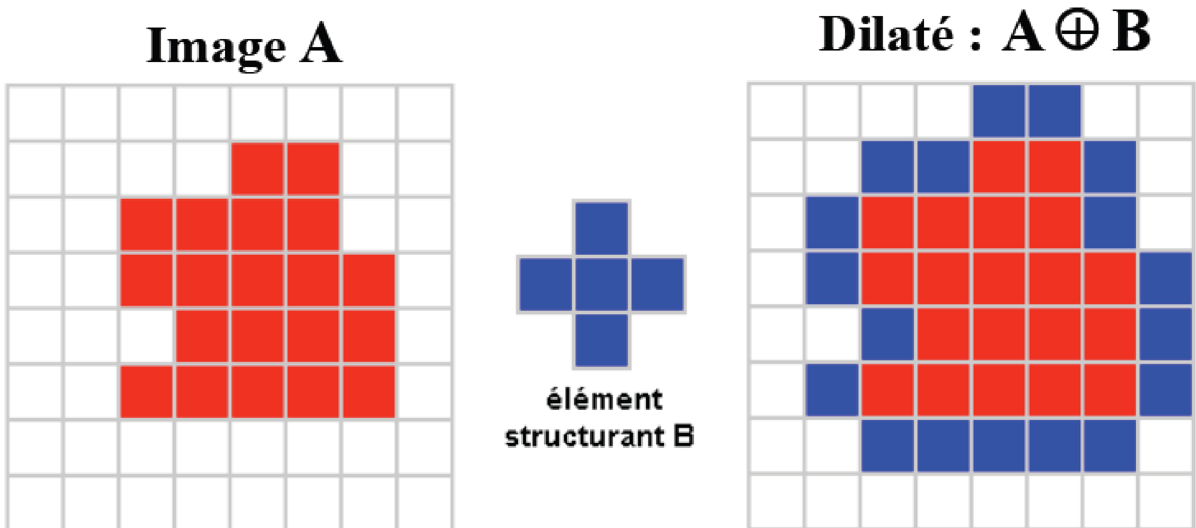
Basic operations: Dilatation

Let B_x be the center of the structuring element B that is put on the pixel X of the image

Algorithm :

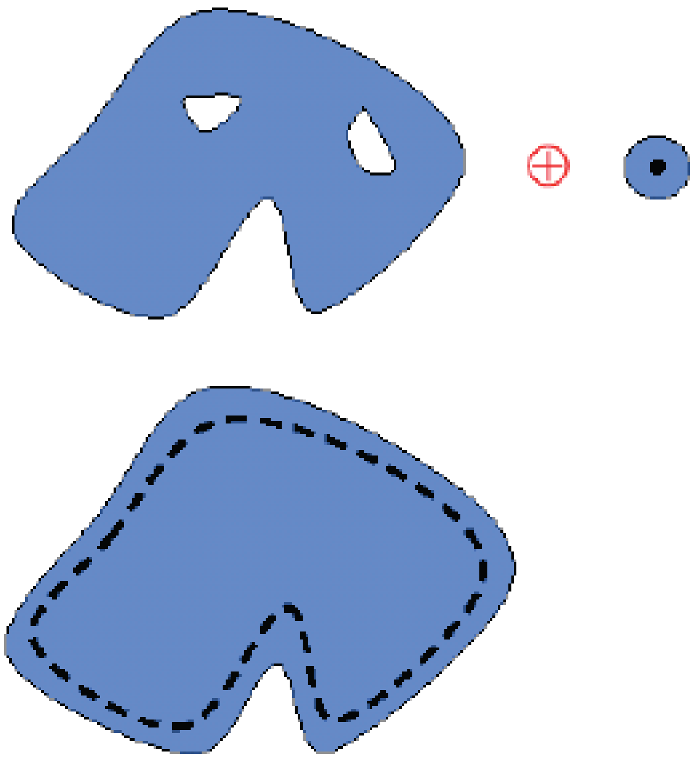
- B_x is positioned on each pixel X of the object A
- IF $B \cap A \neq \varnothing$ THEN

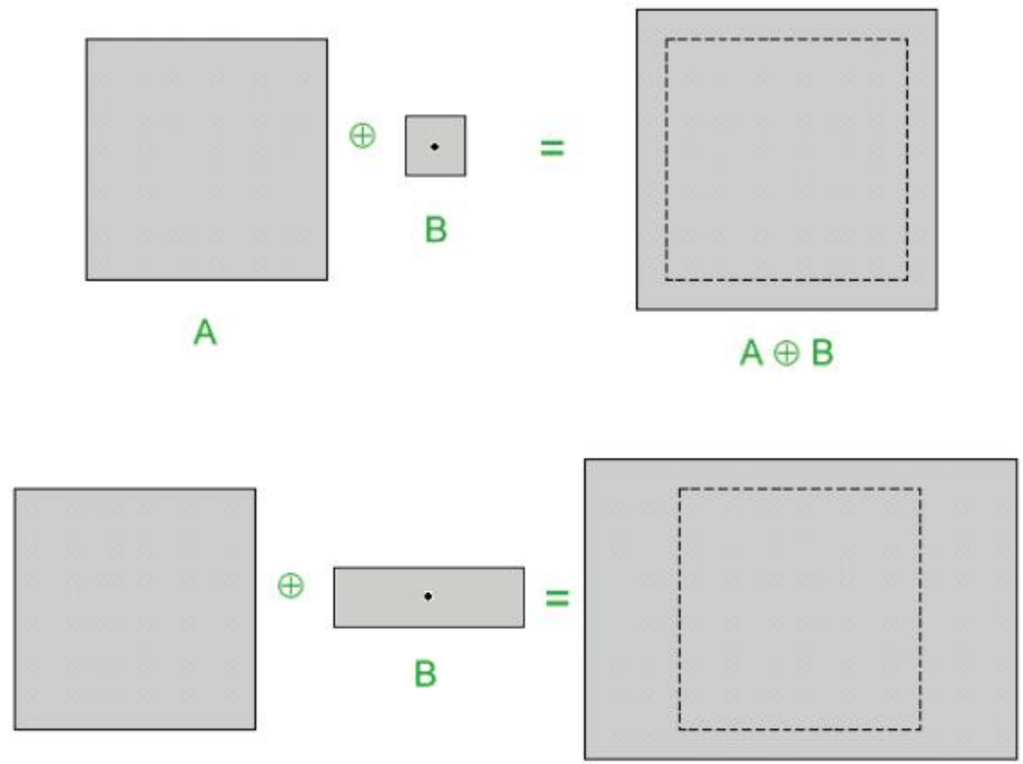
  B_x is set to kept (as part of the dilated object)



Basic operations: Dilatation

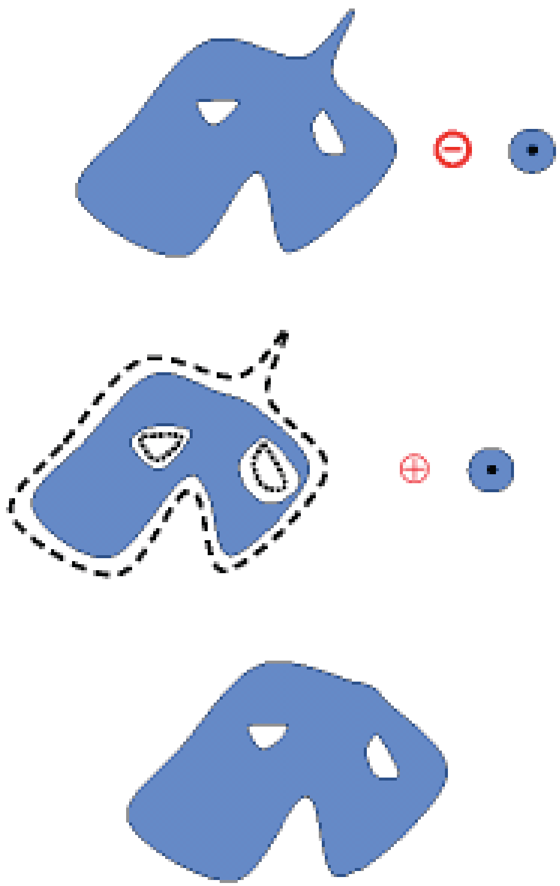
- Examples

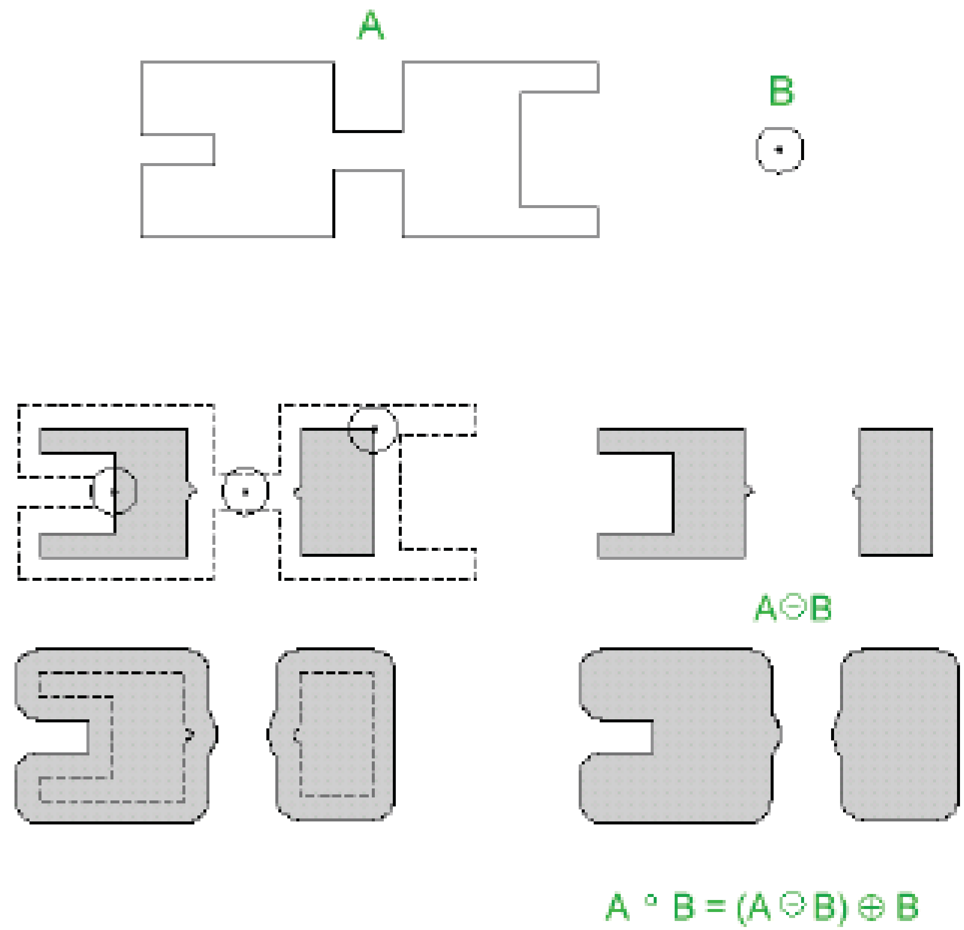




Basic operations: Opening

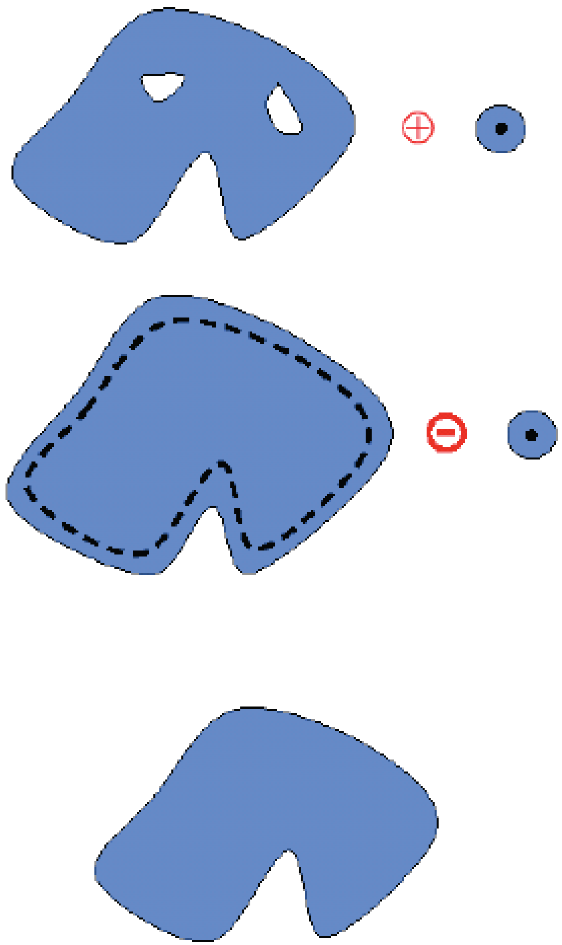
- A v B = (A - B) + B

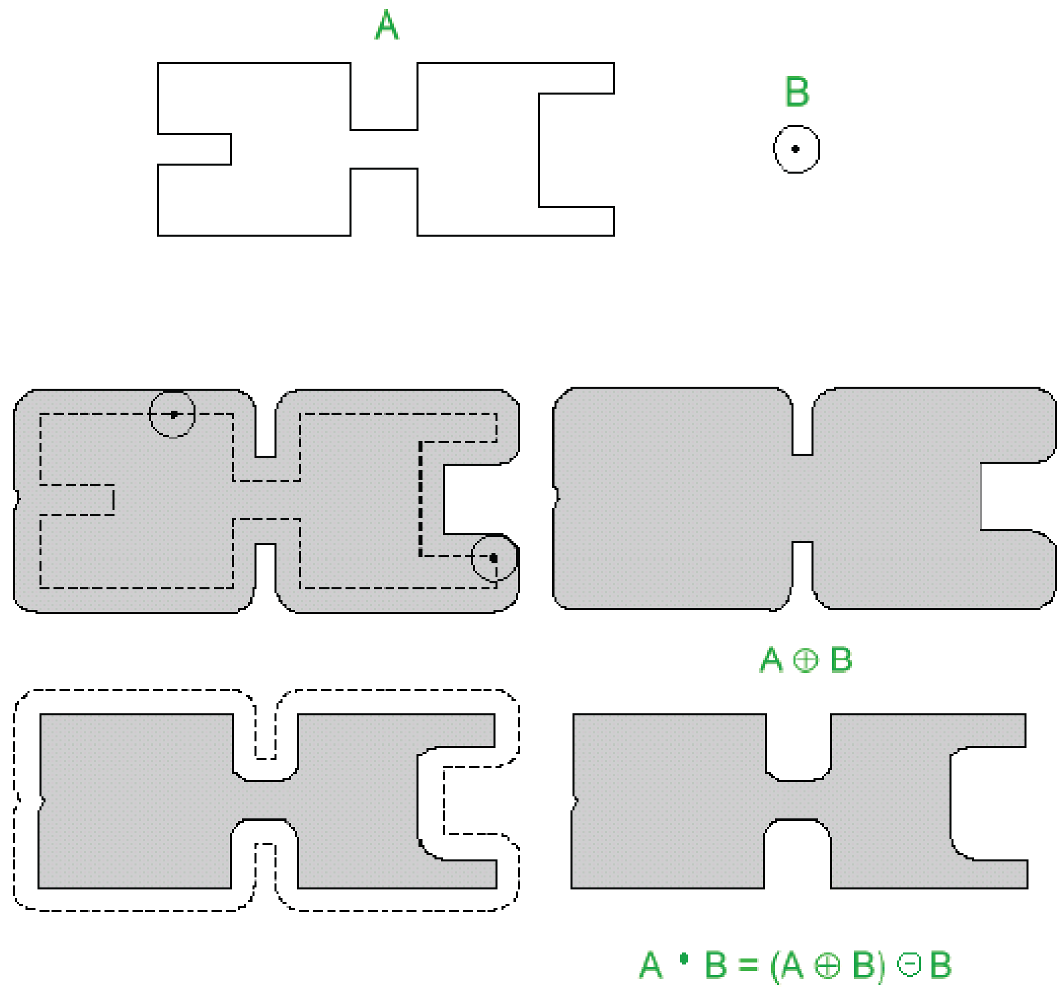




Basic operations: Closing

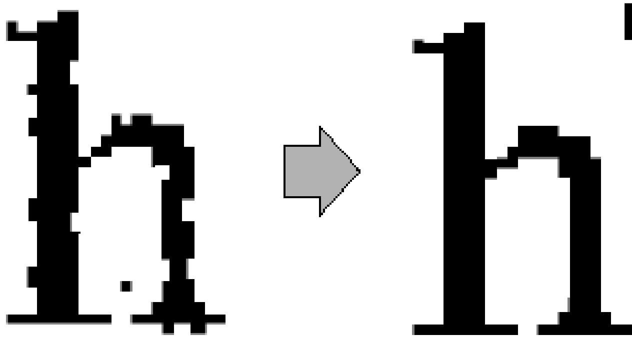
- A B = (A + B) - B





Basic operations: quality improvement

Using binary patterns

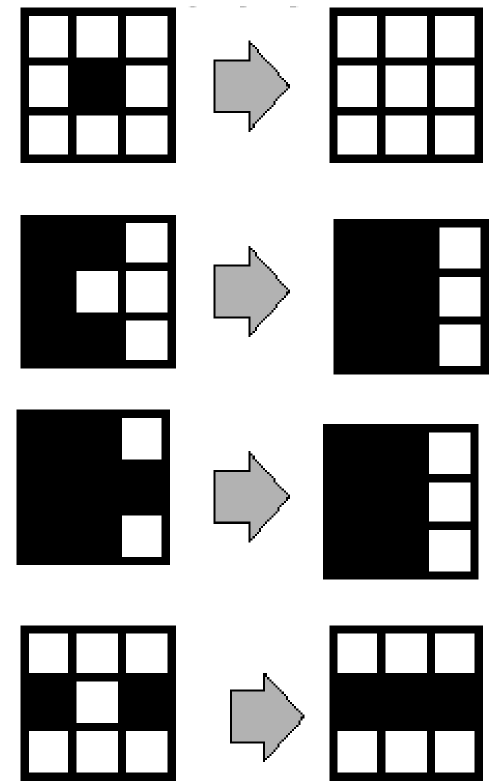


IF THEN

Noise removal

Hole filling

Connexity reparation



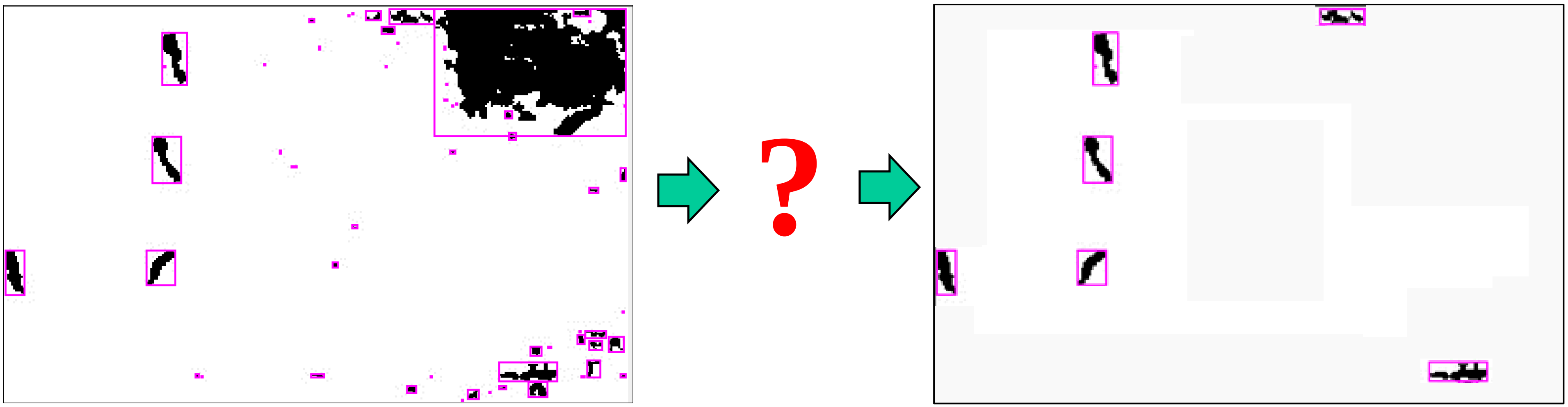
Ect…

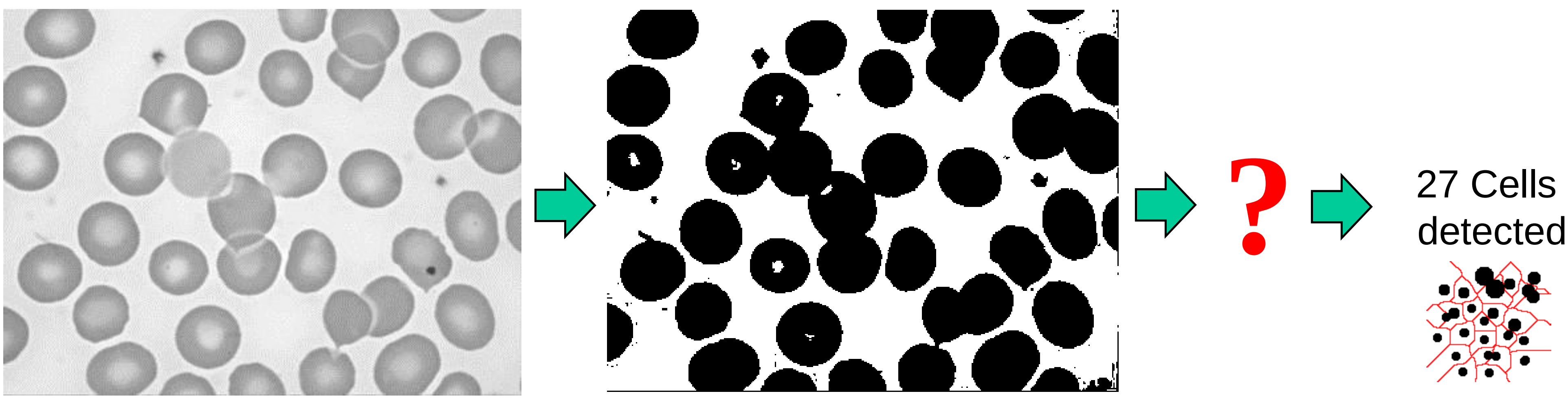
Basic operations: quality improvement

Basic operations: quality improvement

Basic operations: quality improvement

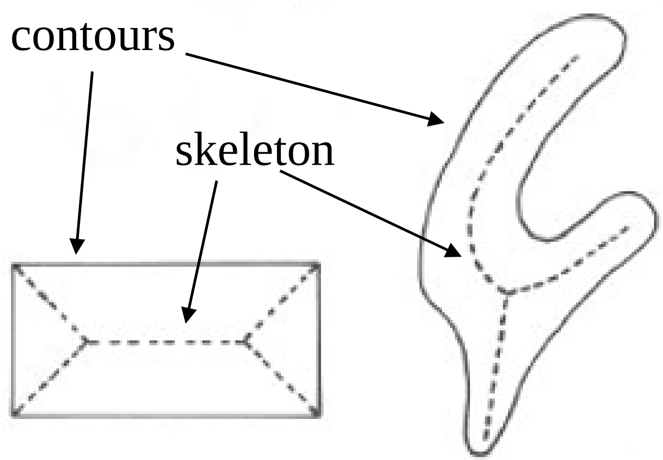
- Image analysis sequence ?

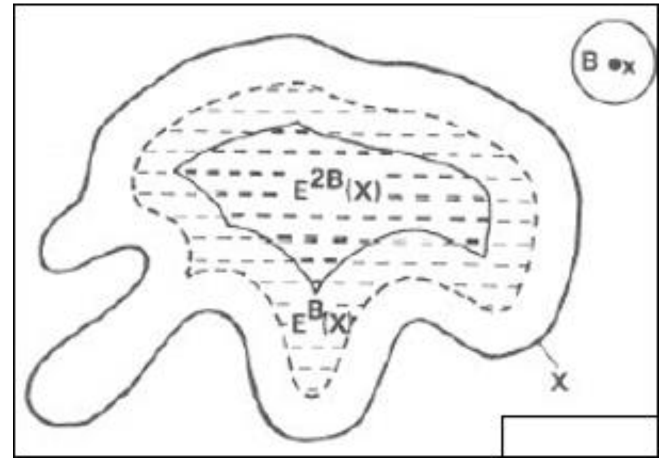


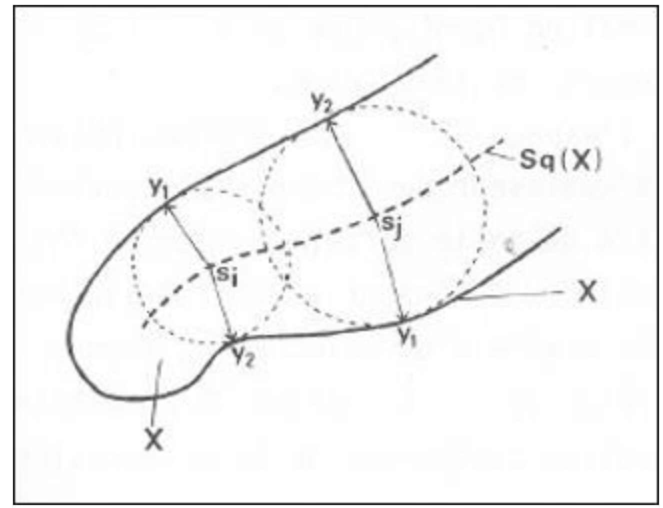


More sophisticate: Skeleton

- Skeleton is defined by the set of points located at equal distance from the border of the shape
- Union of the centers of the maximal spheres that can be included into the shape
- Computed by successive erosions

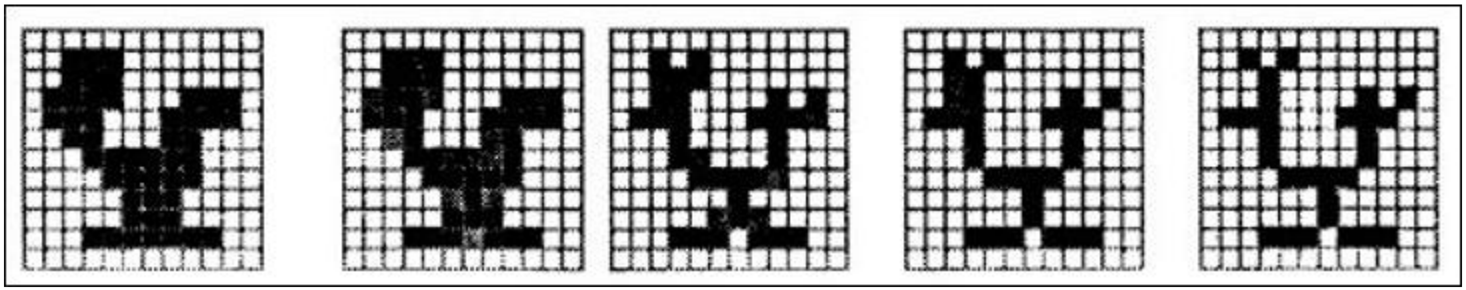




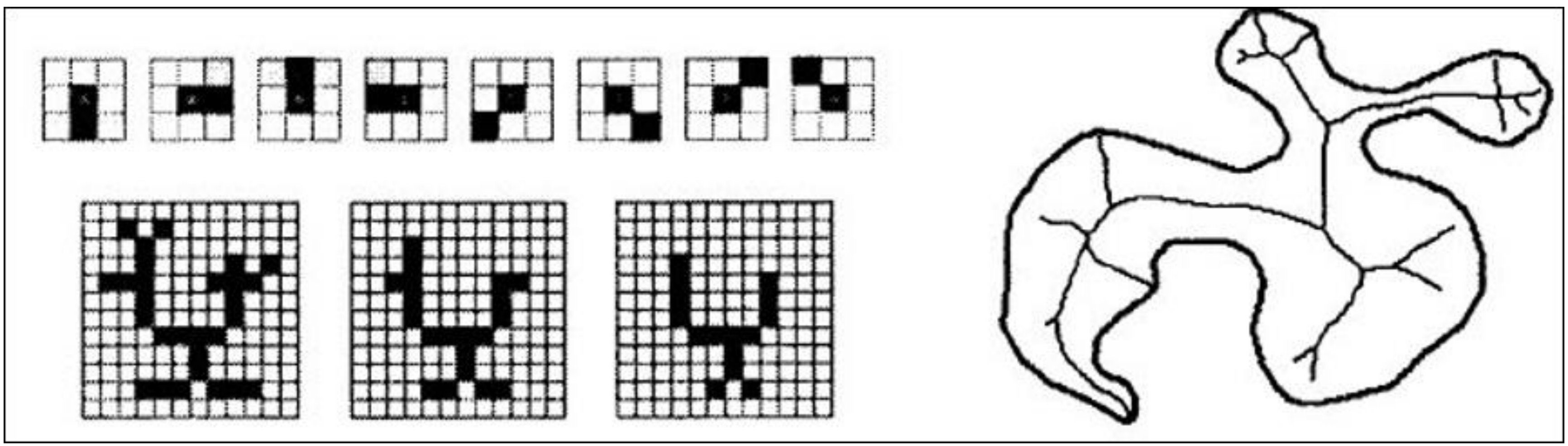


More sophisticate: Skeleton

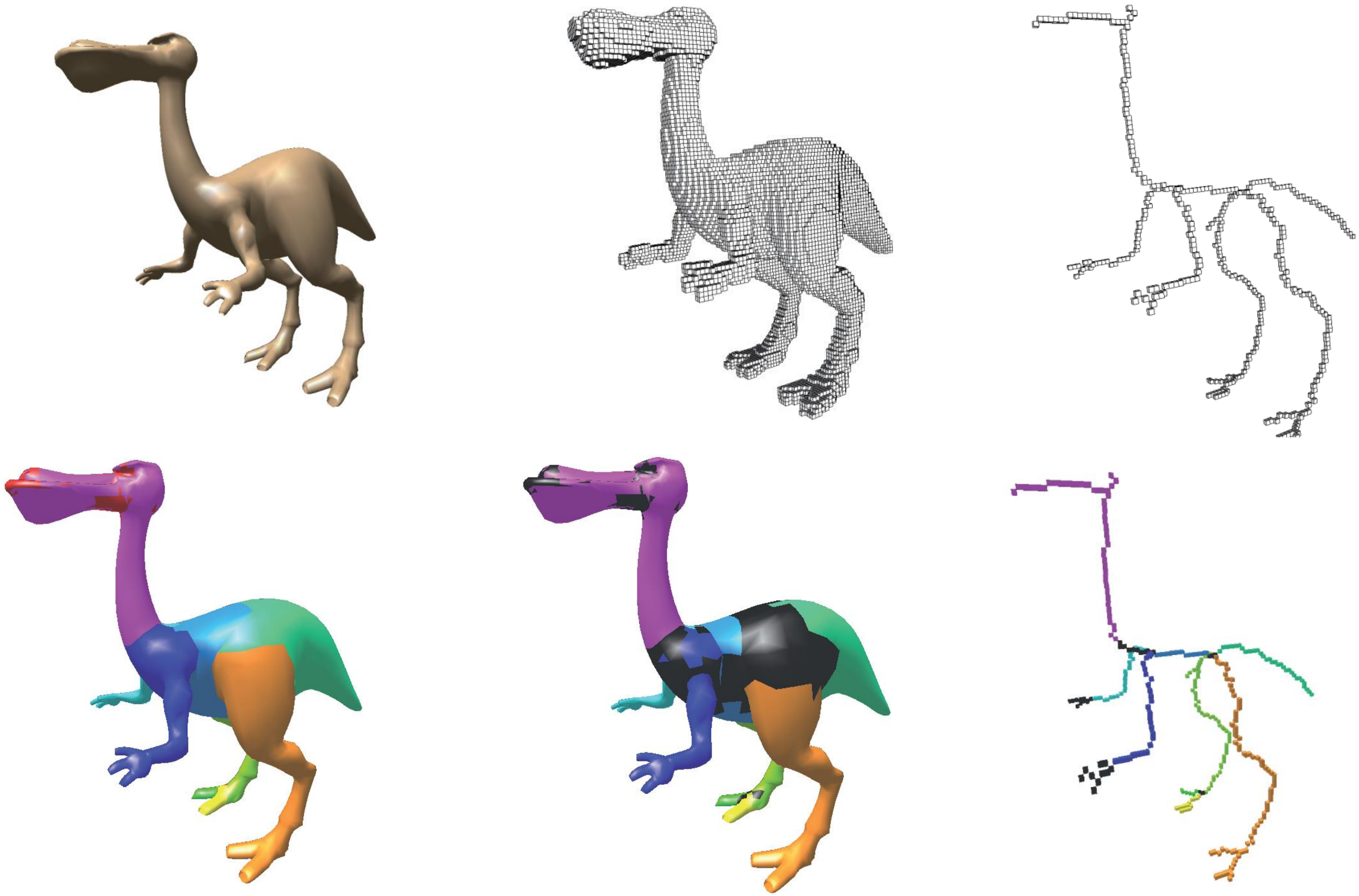
- Computation of the skeleton by successive erosions with different masks



- Small branches have to be removed by using specific masks

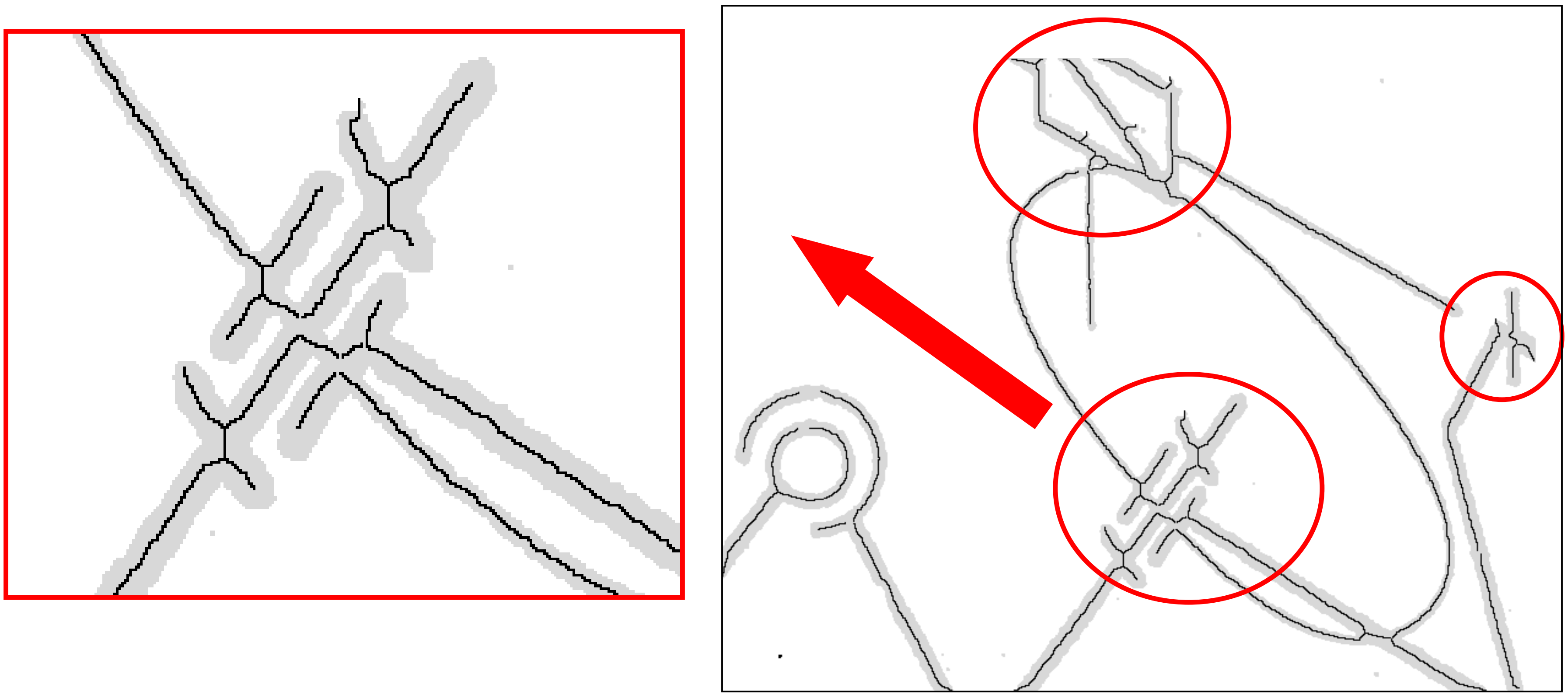


Possible also in 3D



But not so easy to analyse . . .

- Possible wrong representations of junctions and crossing (noise)
- Noise, barbules, …

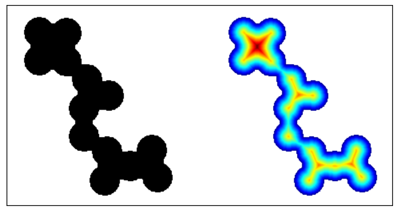


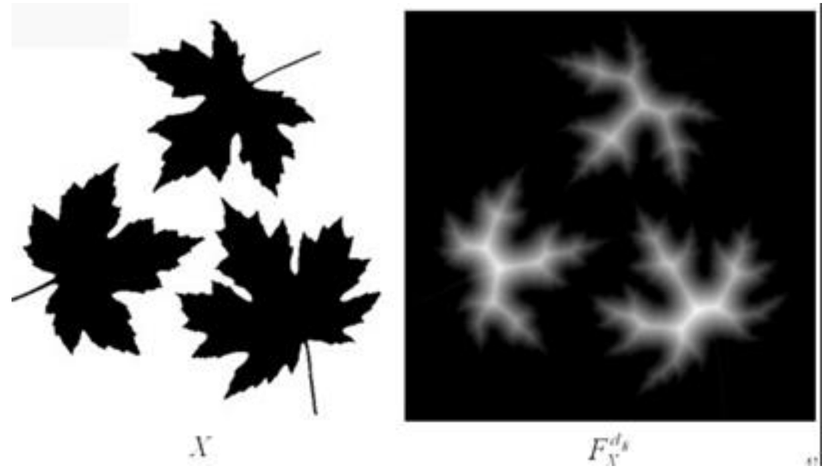
Distance map and Watershed

Euclidian Distance Map (EDM)

- Function that associated to each pixel the distance to the background pixels

$$F_X^d : \mathbf{Z}^2 \to \mathbf{N}, \quad p \mapsto d(p, X^c)$$

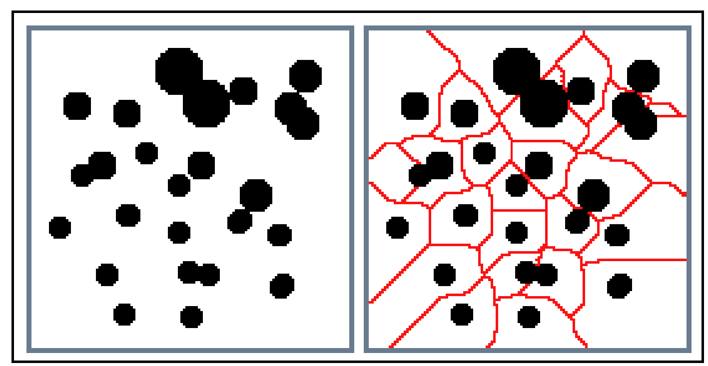




Binary Watershed based on EDM

- Based on EDM, finds the ultimate eroded points (UEPs)
- Dilate each of the UEPs as far as possible
  - either until the edge of the particle is reached,
  - or the edge touches a region of another (growing) UEP.

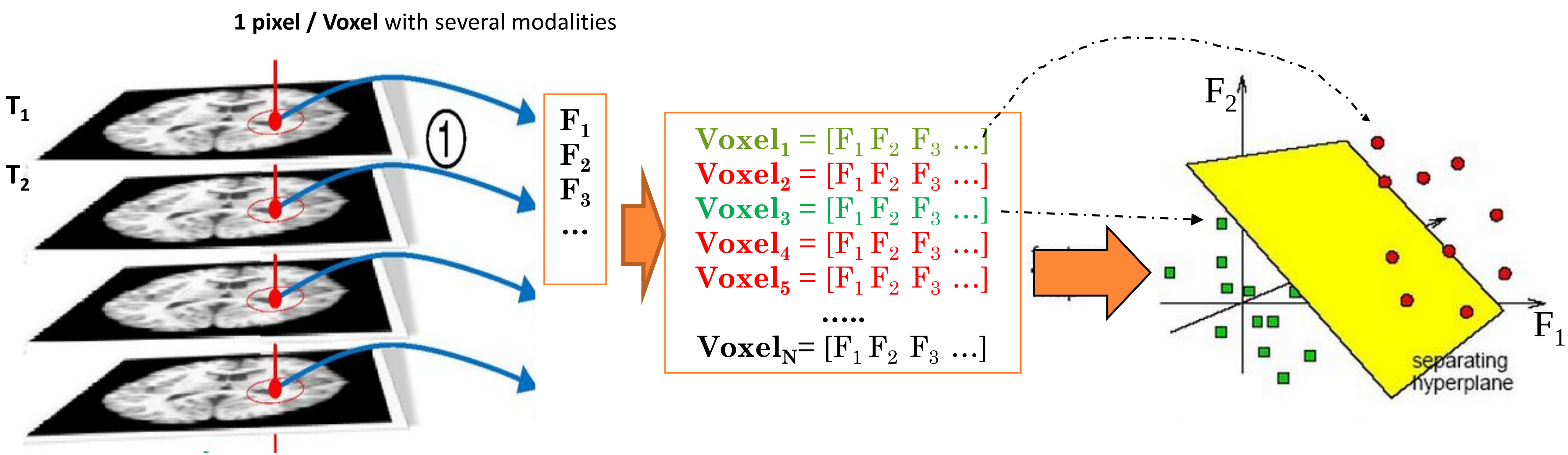
- Automatically cutting particles that touch
- Work best for smooth convex objects that don't overlap too much.



Machine learning based approaches

Principles

- Images are no more seen as image, **BUT**
- Pixels become objects → Each object has to be described by features
- 1 object ⇨ p features ⇨ 1 Vector = 1 Point $\in R^p$
- Data analysis, Statistics and Machine Learning tools become usable



Machine learning based approaches

Feature selection for object description

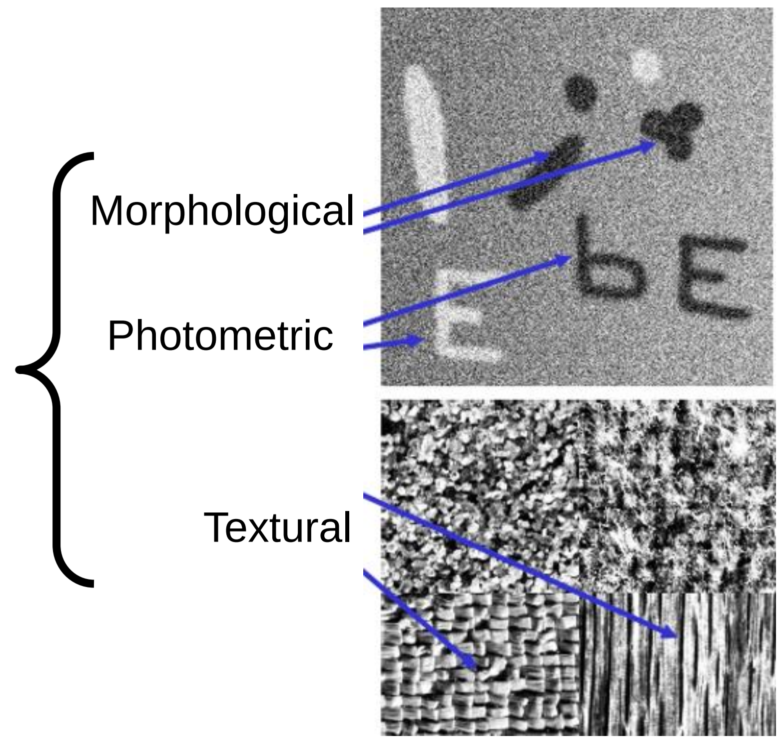
Initial data

⇩

Feature selection

⇩

Classification



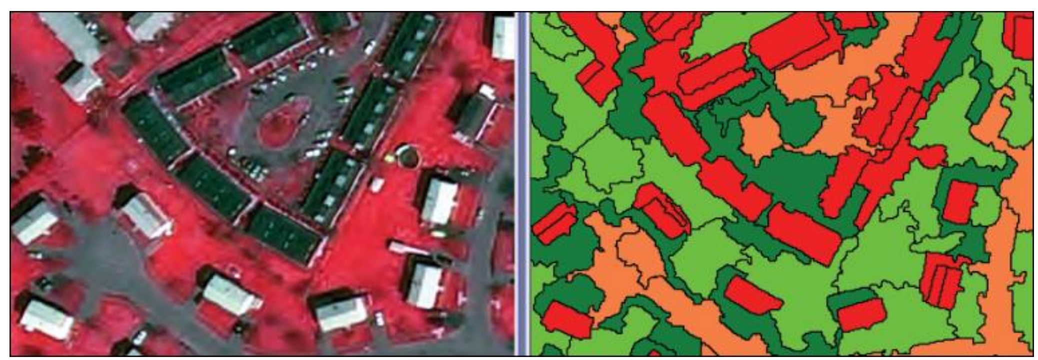
Classification

- Supervised Classification

A training/learning set is available to define **a classifier**

- Unsupervised Classification

We just have the feature vectors
⇨ Grouping of similar vectors to build homogenous clusters

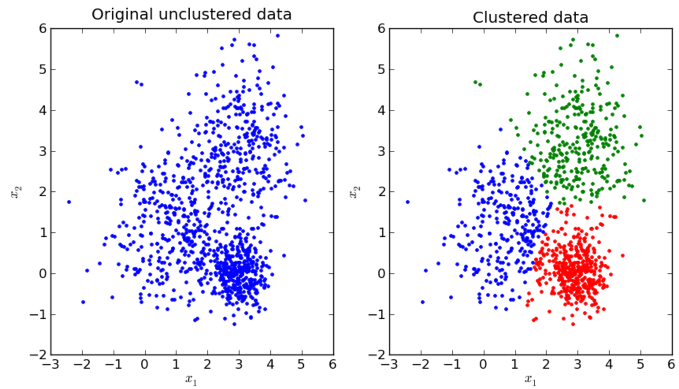


Machine learning based approaches

Unsupervised classification : k-means clustering

- No tagged data available ⇨ learning impossible
- We look for k classes starting from k centers (G_i)

Objectif : minimising the intra-class variance



Algorithm:

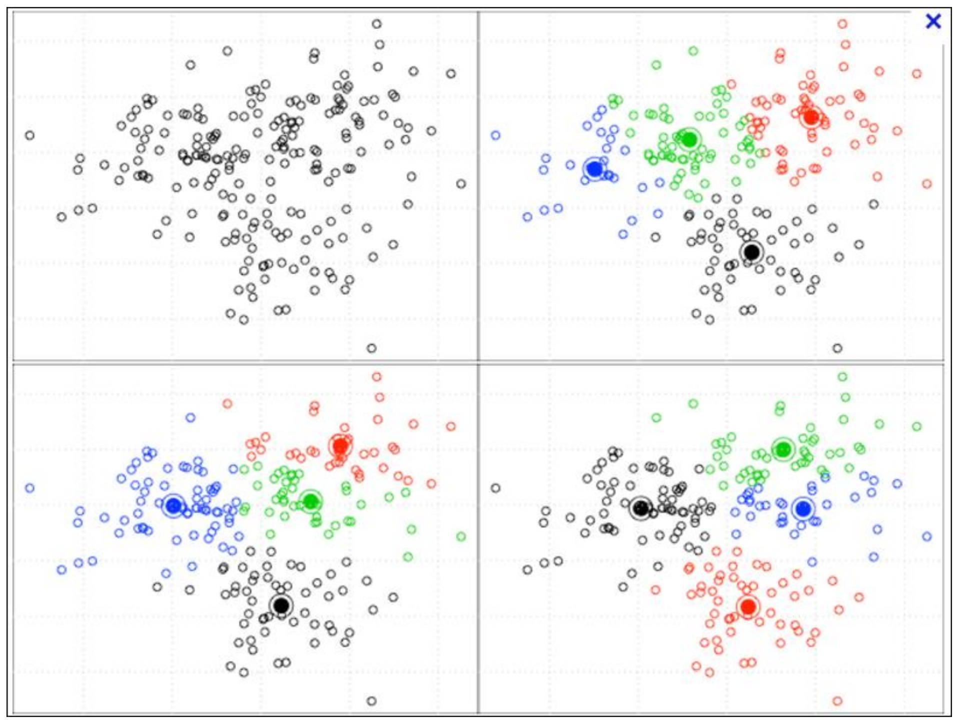
1 - choose K centers randomly

2 – repeat :

a/ Allocate each x to the closest center Gi

b/ Compute the new Gi until

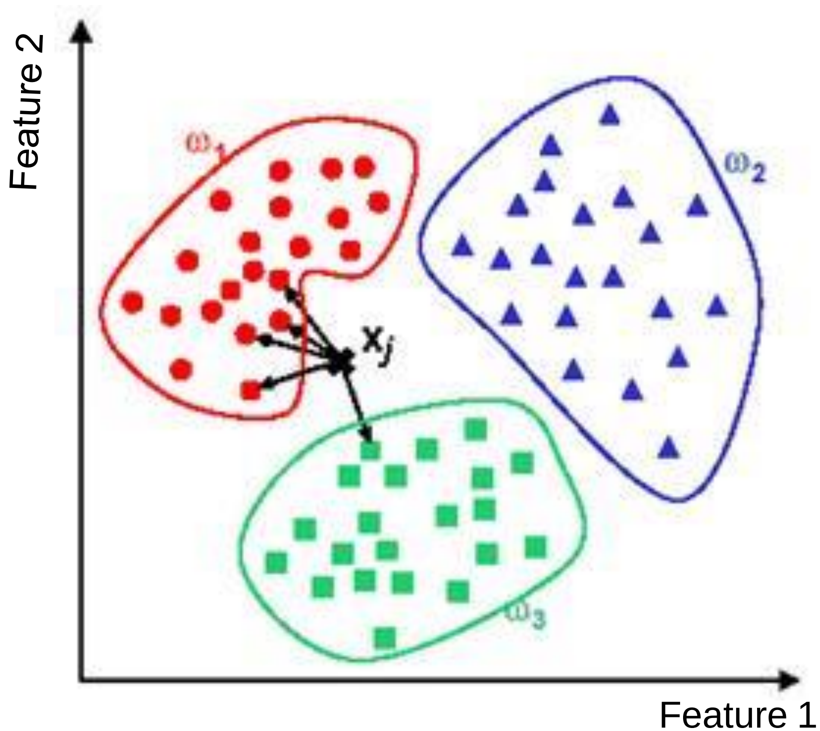
stabilization



Machine learning based approaches

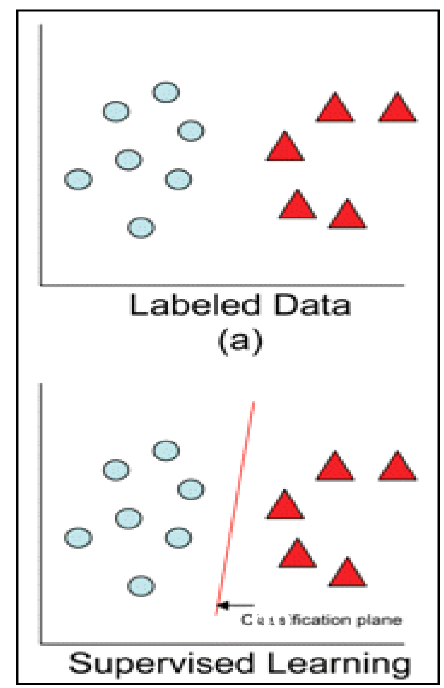
Supervised classification : k-Nearest-Neighbors (kNN)

- We have a training set with feature vectors tagged with the corresponding classes (w_i)
- The unknown vector X_j is classified with/inside the most represented class among its k nearest neighbors



Many sophisticate techniques for classifier definition

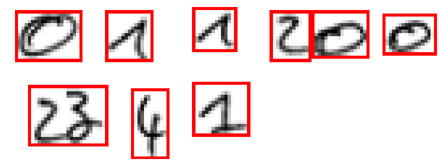
➔



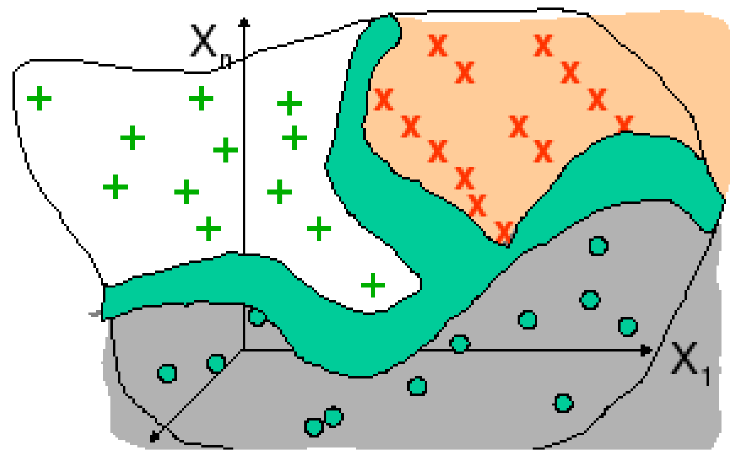
A full sample (simplified)

1. Image acquisition

2. Pre-processing :
   - Filtering, noise removal, scale selection, ...
   - Segmentation of the ROI (objects)



3. Feature extraction : building a vectorial representation of the objects
   - V(2:3;□2; 1000; 50; :::; 45) 
   - V(□3; 10:2; 0; 20; :::;□4; 5) 



4. Classification : select a label for each object from the vectorial representation

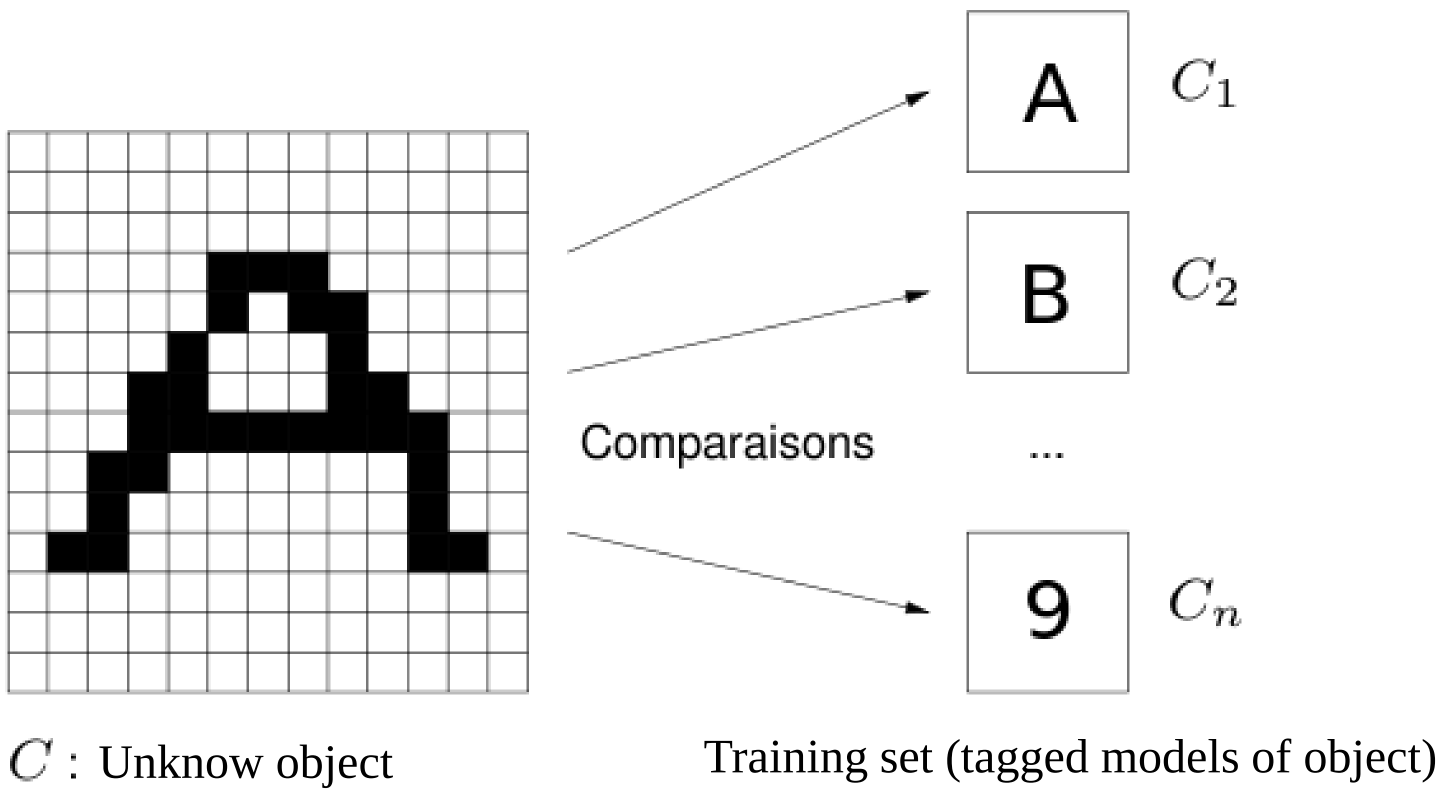
5. Post-processing : contextual verification to correct some errors

.

Feature selection (simplified)

Pixels can be considzered as features

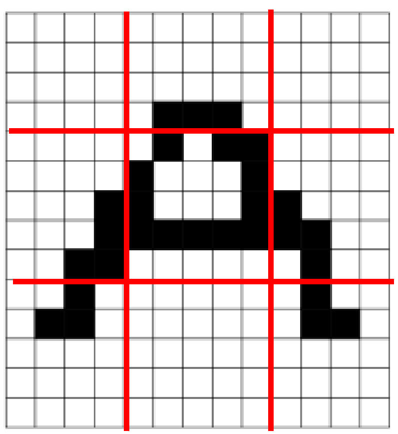
- distance(C; C_i) = $\sum$ij | P (i; j) - P_i(i; j) |



Feature selection (simplified)

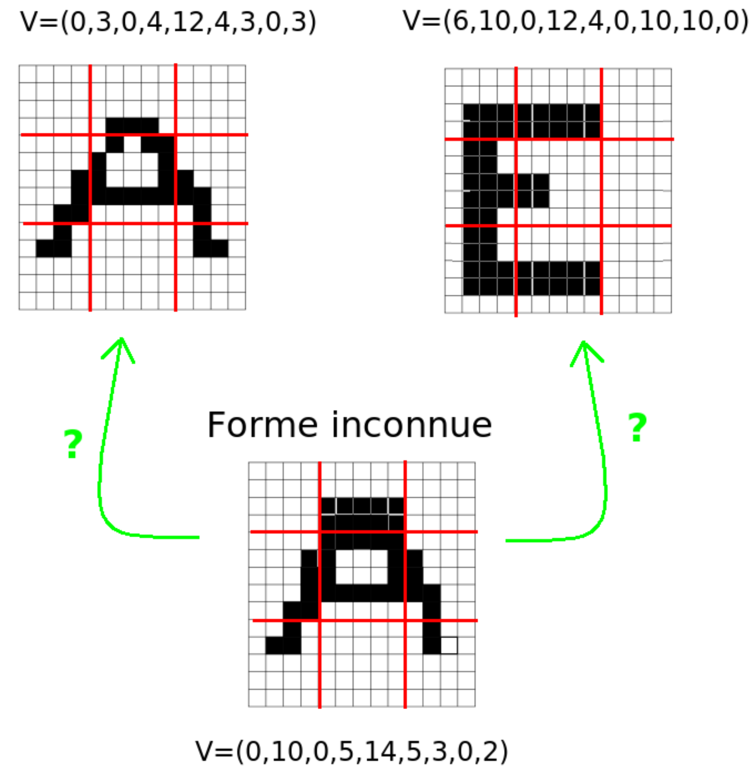
Zoning

- The image is splitted in n blocks
- For each block, some features are computed (number of black pixels)
- A new feature vector is obtained : $V = (Nb_1; Nb_2; :::; Nb_n)$



V=(0,3,0,4,12,4,3,0,3)

Classification (simplified)



The unknown object ? is identified as a « A » because min(D(A; ?);D(B; ?)) = D(A; ?)

$$D(A,?) = \sqrt{(0-0)^2 + (3-10)^2 + (0-0)^2 + (4-5)^2 + \ldots + (3-2)^2}$$

$$D(A,?) = 7,48 \text{ et } D(B,?) = 19,05$$

That All for today…

- Thank you

Approche basée Graphes

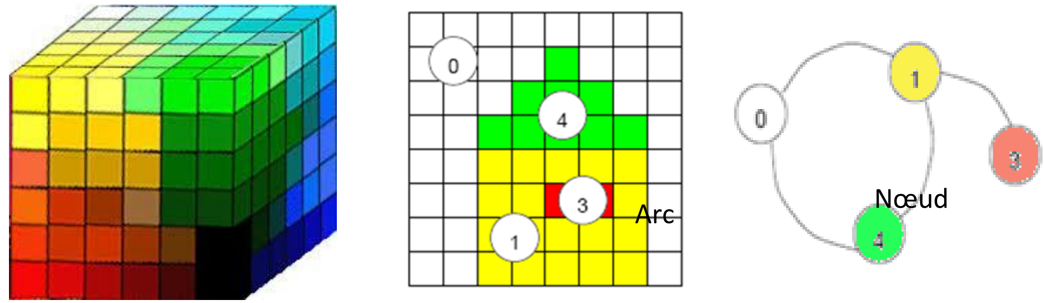
Modélisation du contenu de l’image

⇨ Structuration des données à l’aide d’un graphe d’adjacence

Nœud = Région

Attributs = Descripteurs de la région

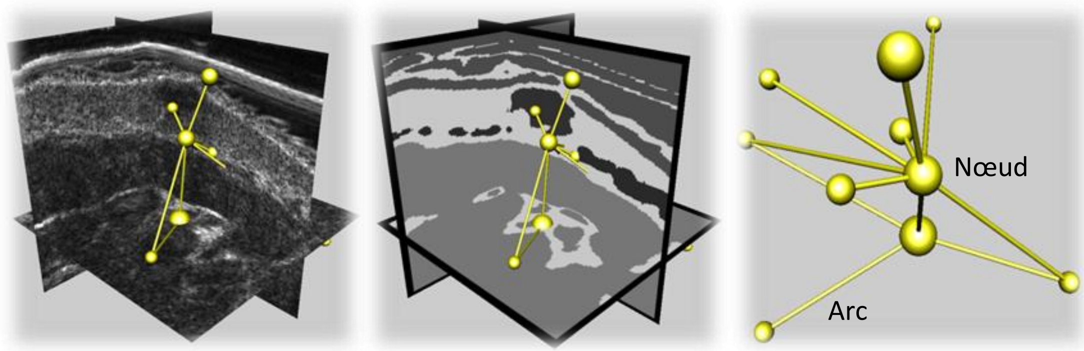
- Liste des voxels de la région
- Position : G(x,y,z)
- Liste de caractéristiques F_j (les moyennes)
- Forme, couleur, modalités, …
- Label, annotation, …



Arc = Relation entre Régions

Attributs = Descripteurs des relations

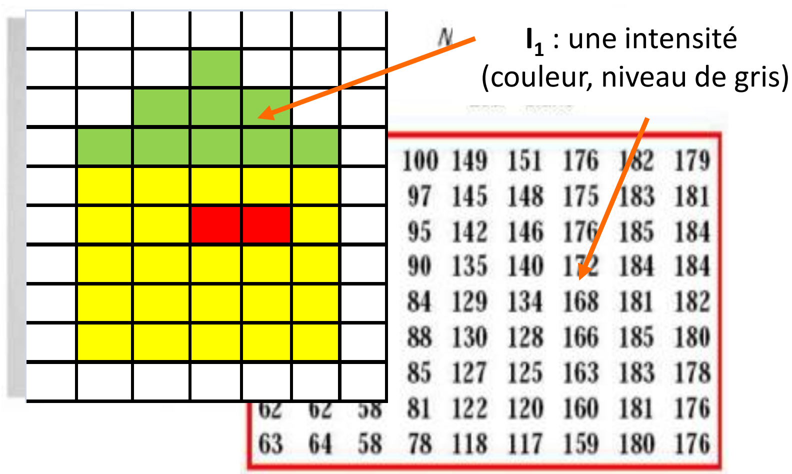
- Les 2 nœuds liés
- Aire de la surface de contact
- Type de relation, distance G_1, G_2
- Taux de ressemblance, …
- Label, annotation, …



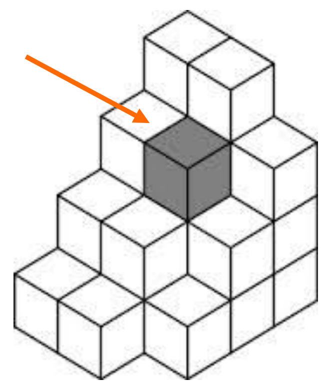
Approche basée Graphes

Modélisation du contenu des images

⇨ **Caractérisation des contenus**



Approche classique



1 voxel = 1 valeur

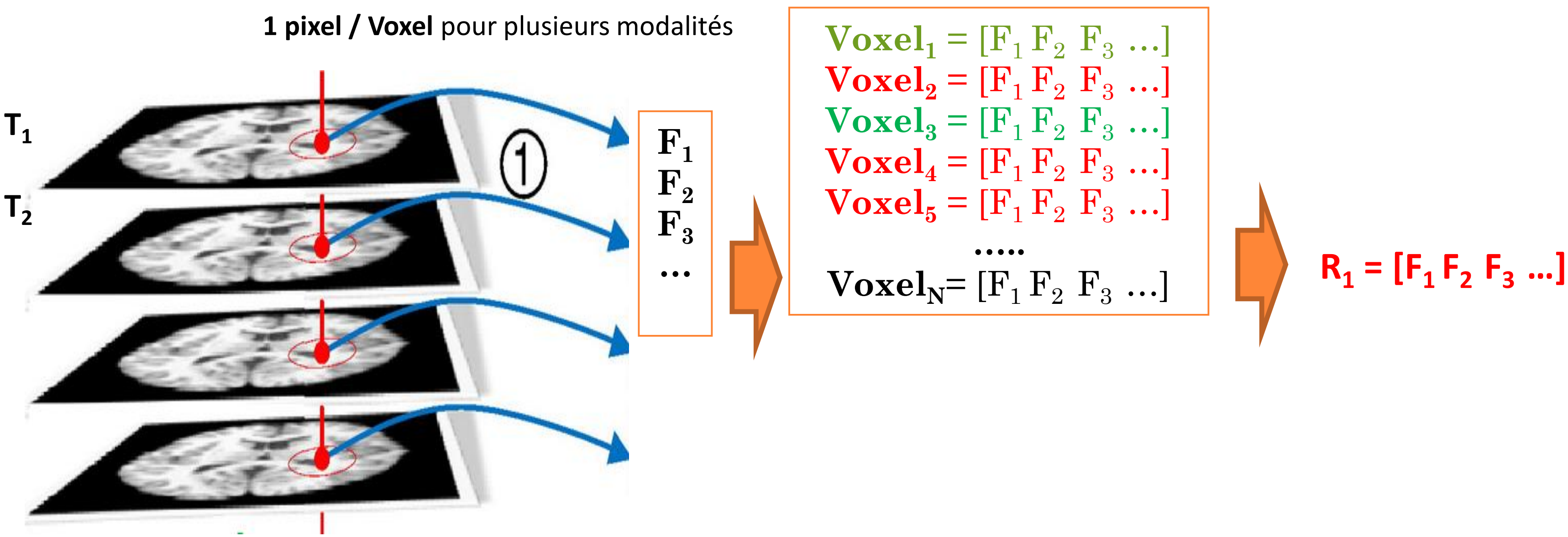
Caractérisation des voxels

- Recalage des acquisitions multimodales (T1, T2, …)
- 1 voxel = 1 liste de caractéristiques F_j plutôt qu'une couleur
- 1 voxel = 1 individu ⇨ **Voxel$_i$** = [F_1 F_2 F_3 …]

Approche basée Graphes

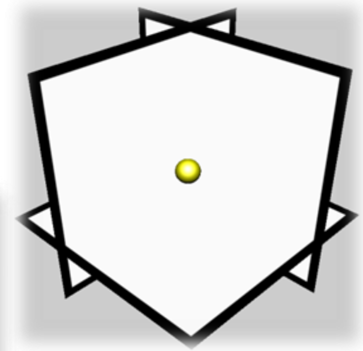
Caractérisation des régions

- Segmentation = regroupement des individus similaires en classes
- Perte de l'information de connexité (vs approche région, contours)
- Caractéristiques Région = Moyenne des caractéristiques des Voxels appartenant à la région

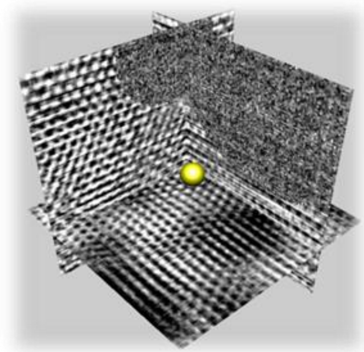


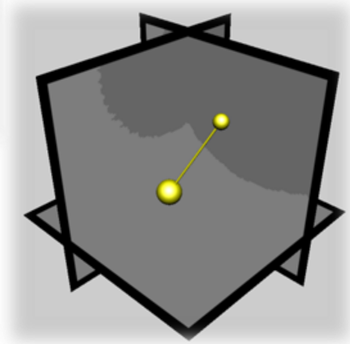
Approche basée Graphes

Segmentation interactive du contenu de l'image
⇨ Transformation successive du graphe (RAG)



Opération de Division





Fonction *Seg* correspond à un *K-means*

- Rapidité d'exécution
- Faible coût en mémoire
- Partitionnement efficace

Nouveau graphe :

$$G_{k+1} = Kmeans(U_{op}.V, G_k, F, U_{Seg})$$

U_{Seg} : *nombre de régions*

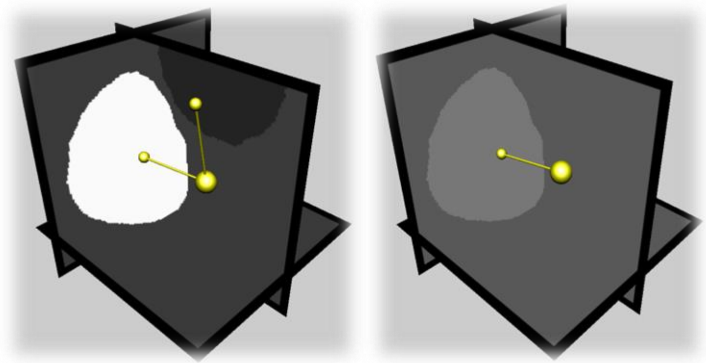
F : *caractéristiques choisies par l'utilisateur (attributs des nœuds)*

$U_{op}.V$: *identifiant du nœud à diviser*

Approche basée Graphes

Segmentation interactive du contenu de l'image

➔ Transformations successives du graphe (RAG)



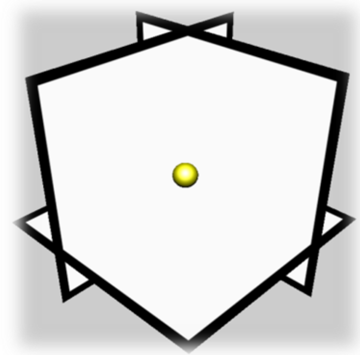
Opération de Fusion

Simple fonction de fusion *Merge*

- Nouveau graphe : $G_{k+1} = Merge(U_{op}.E, G_k)$

- Initialisation des attributs de V_{new} durant la fusion de V_1 et V_2

$$V_{new}.T_i = \frac{(V_1.T_i)(V_1.NV) + (V_2.T_i)(V_2.NV)}{V_1.NV + V_2.NV}$$



NV : nombre de voxels identifié par un nœud
$T = \{\bar{F}, \bar{G}\}$

Approche basée Graphes

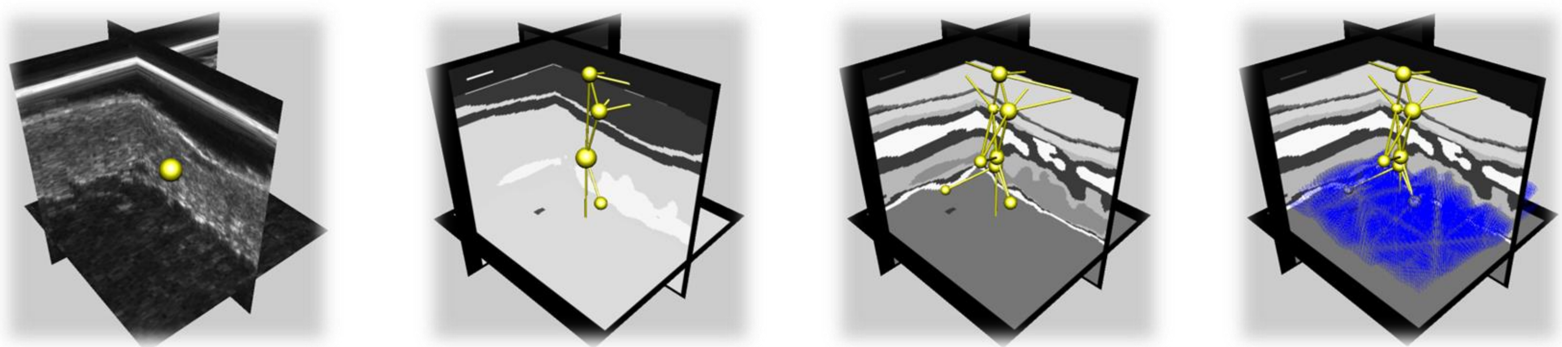
Segmentation interactive du contenu de l'image

➔ Transformations successives du graphe (RAG)

Initialisation

- GAR initialisé à partir d'un unique nœud = 1 région unique
- 1ère étape du processus de segmentation : Division
- Libre choix des opérations par la suite (Fusion, Division, Etiquetage, …)

Construction de scénarios de segmentation interactive

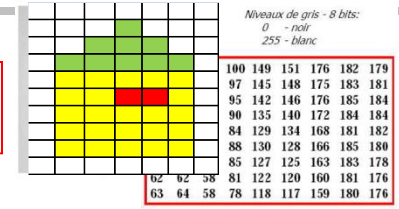


Autres opérations possibles : Etiquetage, simplification, suppression, …

ANALYSE D'IMAGES

Acquisition

Images, signaux 1D 2D 3D

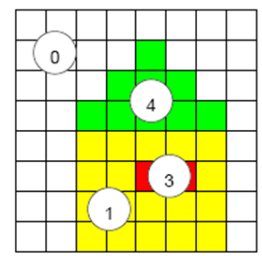


Prétraitements

Images, signaux améliorés

Segmentation / Description / Représentation

Régions d'intérêt, primitives de description



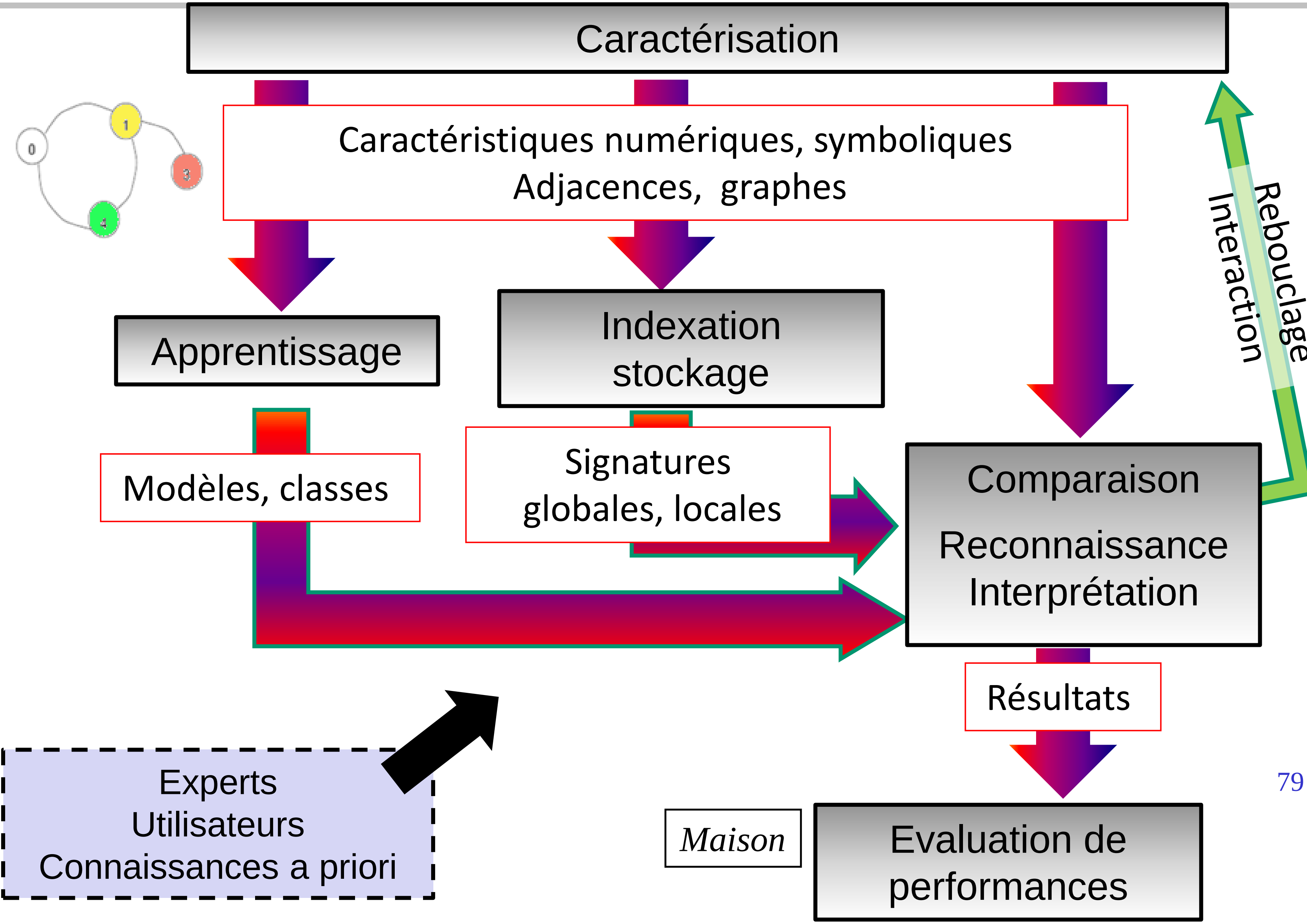
Caractérisation

Experts

Utilisateurs

Connaissances a priori

Reconnaissance des Formes



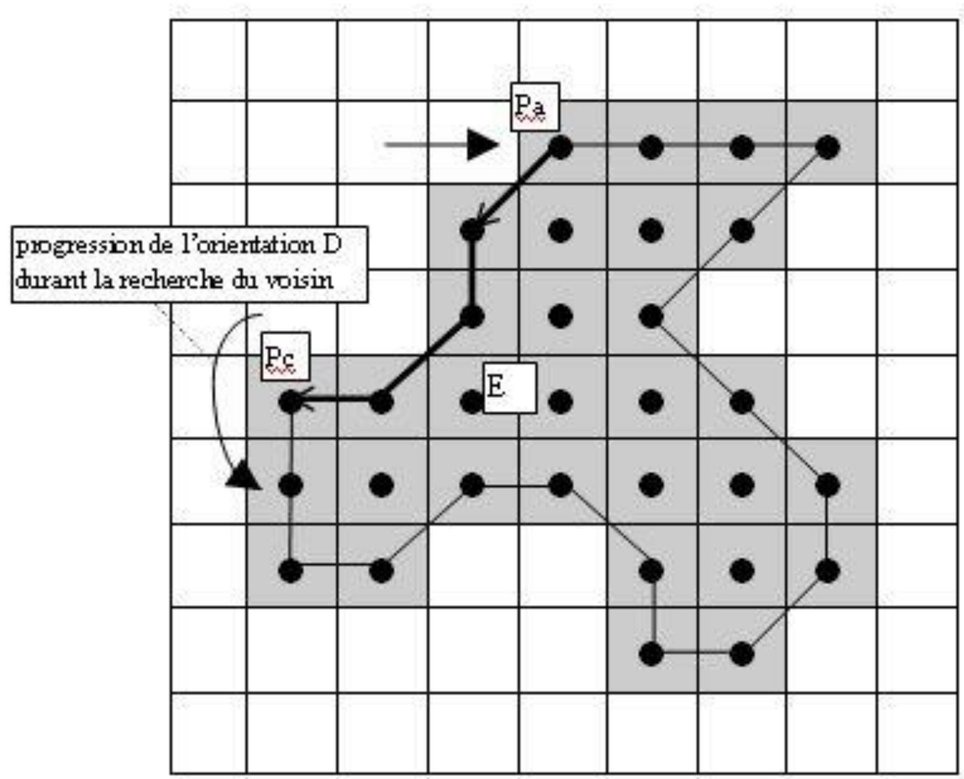
Contours et composantes connexes

- Suivi de contour →

3	2	1
4	Pc	0
5	6	7

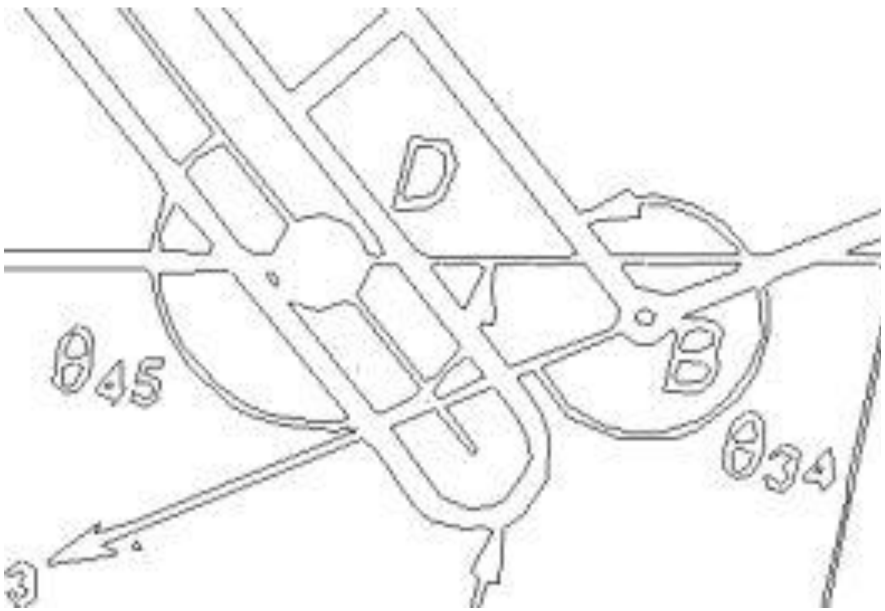
Image naturelle (NdG) !!!

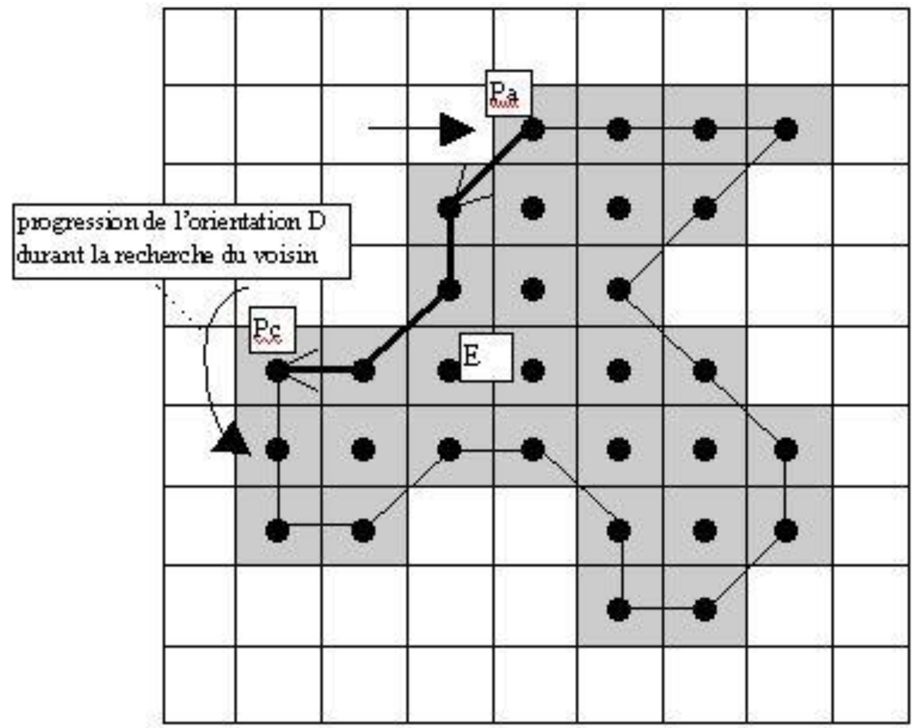
Image binaire



Approximation polygonale → Vectorisation

- Des contours/squelettes aux vecteurs





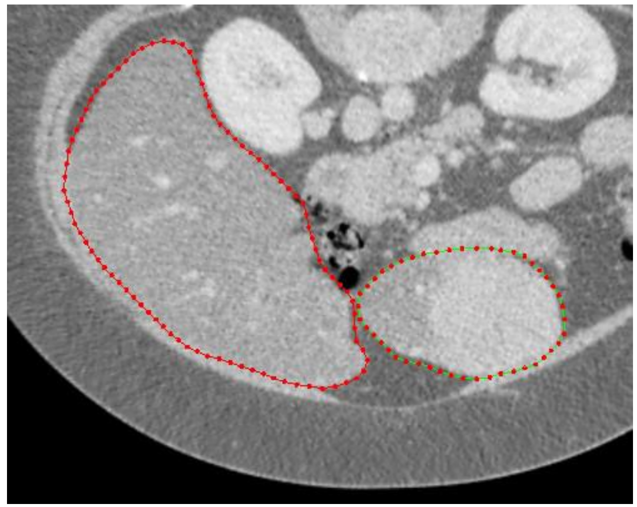
Contours actifs

- **Contour actif :** un ensemble de **n points mobiles**
- **Initialisation** manuelle **du** contour
- **Approche itérative** cherchant à repositionner les points de contours “le mieux possible”
- “Le mieux possible” ➔ minimisation d’une fonction d’énergie
- Tous le problème est de définir la **bonne fonction** à minimiser
- **Souvent,**
  - $E_T = E_{interne} + E_{externe}$ ➔ $E_{Totale} = \underbrace{-\int |\nabla I(p(s))|^2 ds}_{E_{Externe}} + \underbrace{\int \left(\alpha(s)|p'(s)|^2 + \beta(s)|p''(s)|^2\right) ds}_{E_{Interne}}$
  - $E_{externe}$ est minimale quand le contour est bien positionné sur le contour (fort gradient)

$$E_{Externe}(p_{n,k}) = \gamma \left| \underbrace{\nabla G_\sigma}_{Gradient:\,Canny\,edge\,detector} * I(p_{n,k}) \right|$$

  - $E_{interne}$ est minimale quand le contour a une forme adaptée (lisse)

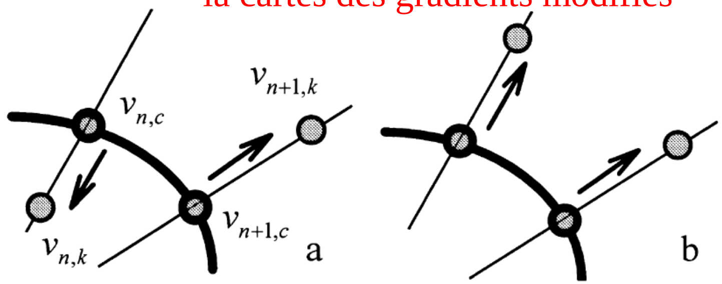
$$E_{Interne}(p_i) = \frac{1}{2}\left(\alpha_i |p_i - p_{i-1}|^2 + \beta_i |p_{i+1} - 2p_i + p_{i+1}|^2\right)$$

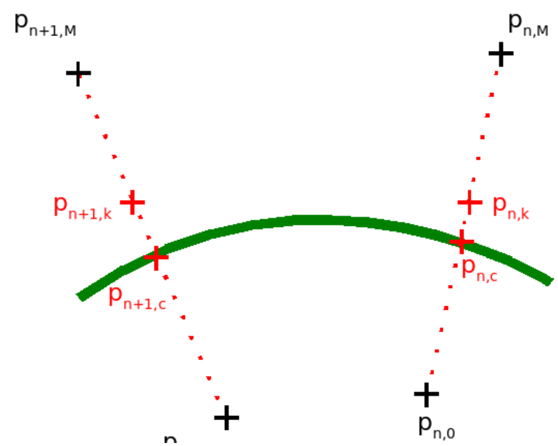


Contours actifs

- Evolution du contour…

Comparaison des différences d'intensités dans la cartes des gradients modifiés



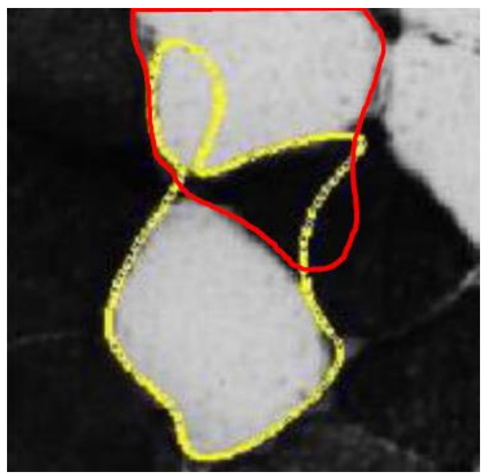


$$E_{Externe}(p_{n,k}) = \gamma \left| \underbrace{\nabla G_\sigma}_{Gradient : Canny\, edge\, detector} * I(p_{n,k}) \right|$$

$$E_{Interne} = \alpha \left| (p_{n+1,k} - p_{n+1,c}) - (p_{n,k} - p_{n,c}) \right|$$

Comparaison des déplacements

- De nombreuses propositions d'améliorations
  - Surfaces actives 3D, levelset, …
  - Incorporation de mécanisme **d'apprentissage et classification (RF)**
  - Ajout d'autres énergies (a priori de **forme, texture**, …)
- Limitations
  - Choix des paramètres, initialisation manuelle, …
  - Changement de topologie (nombre de contours)
  - Temps de calcul



Contours actifs : modèle explicite

- Avantages
  - Adaptation aux déformations des objets (tissus biologiques)
  - Quelques itérations suffisent
- Désavantages du modèle classique :
  - Le contour initial ne peut pas être sélectionné automatiquement (sauf cas simple)
  - Le contour initial doit être proche du contour final
  - Le modèle classique n’est pas utilisable dans le cas de la présence de texture
  - Le modèle classique peut être perturbé en présence de bruit
  - La minimisation d'énergie demande l’inversion de matrices de grandes tailles à chaque itération