

# SEGMENTATION D'IMAGES

**Ingénieurs 2ème année**

2022-2023

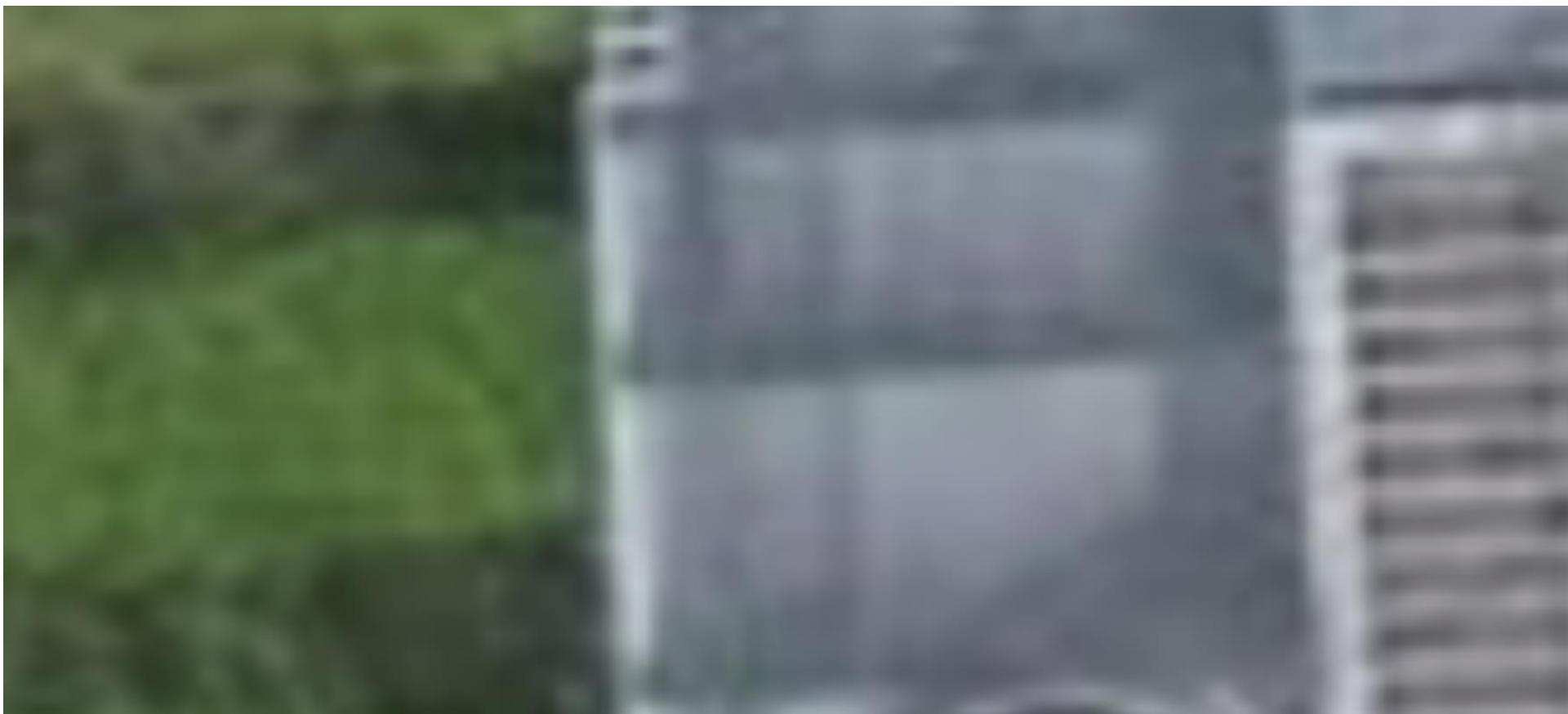
Arnaud LE BRIS – arnaud.le-bris@ign.fr

Clément MALLET – clement.mallet@ign.fr

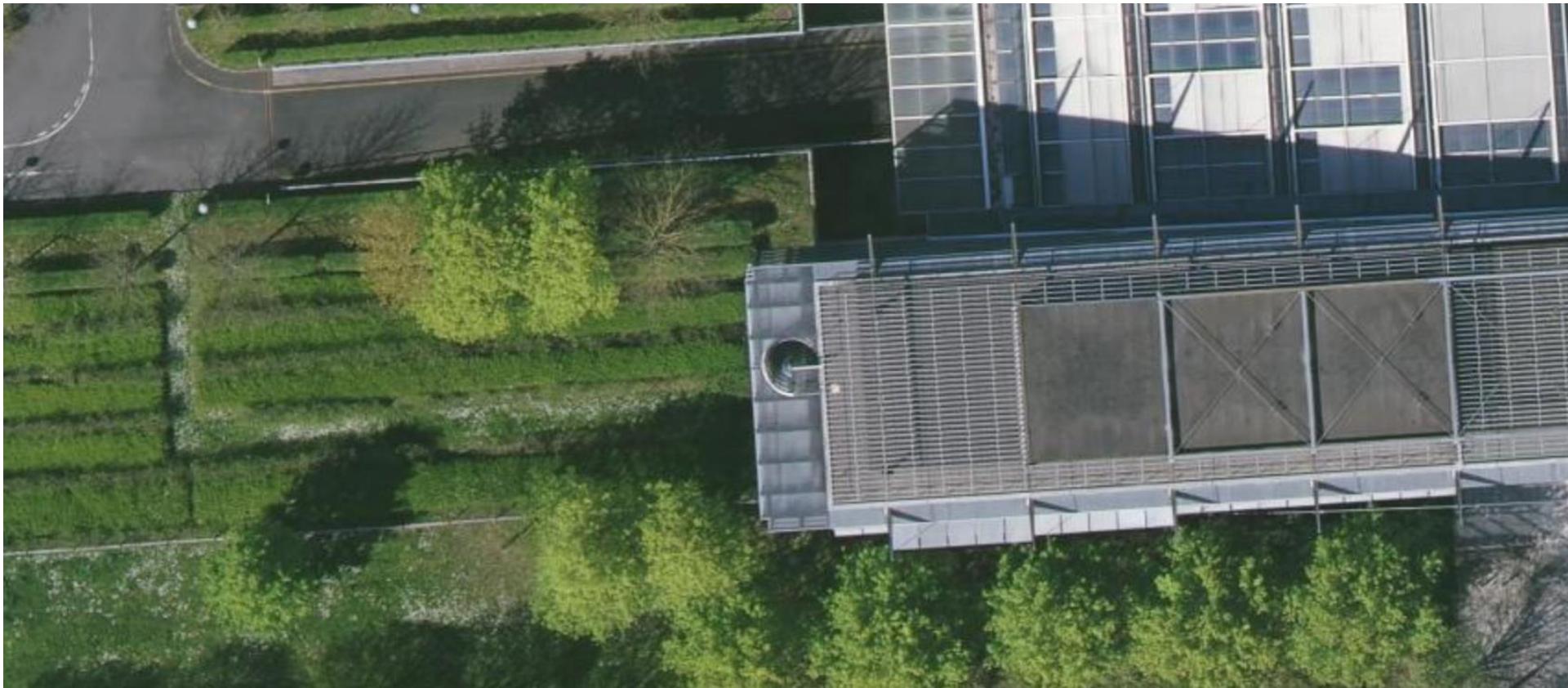
# Qu'est-ce que vous voyez ?

---

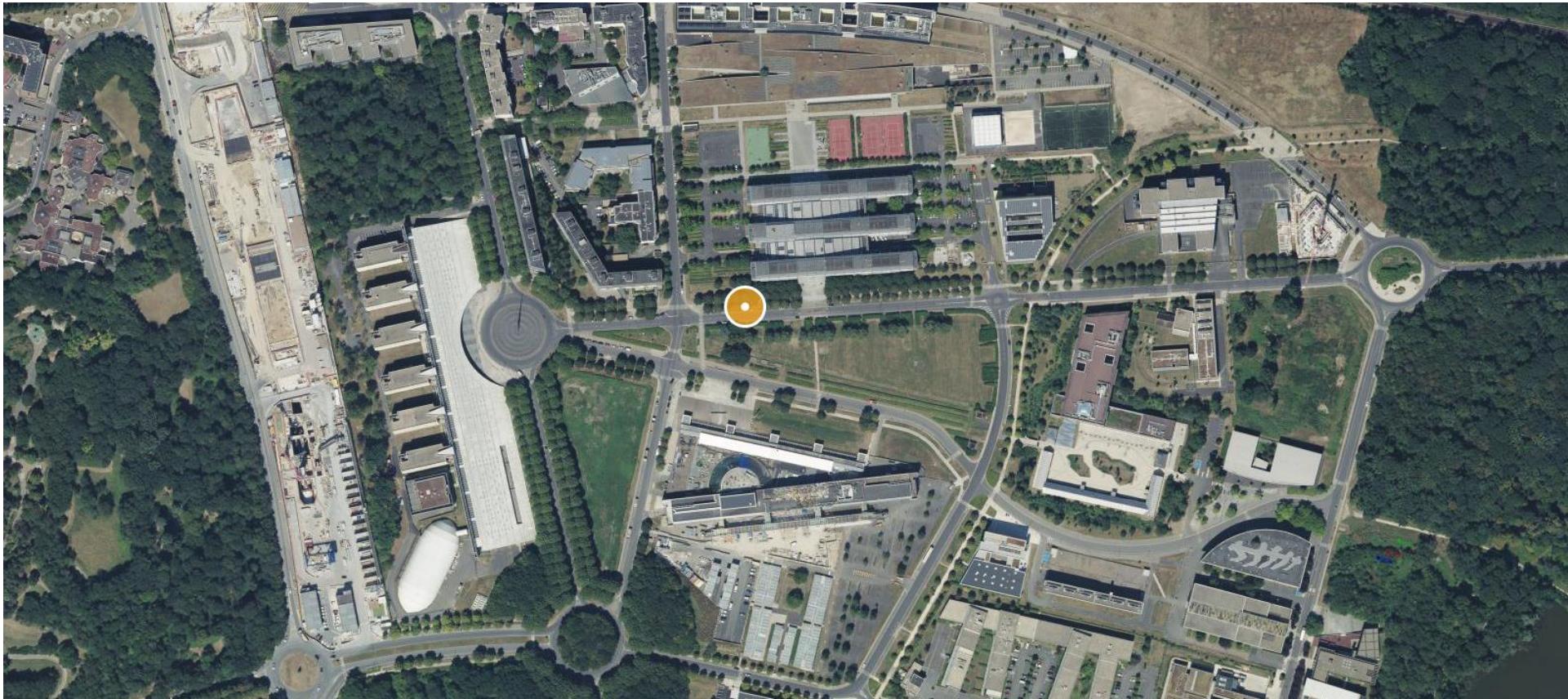
# Qu'est-ce que vous voyez ?



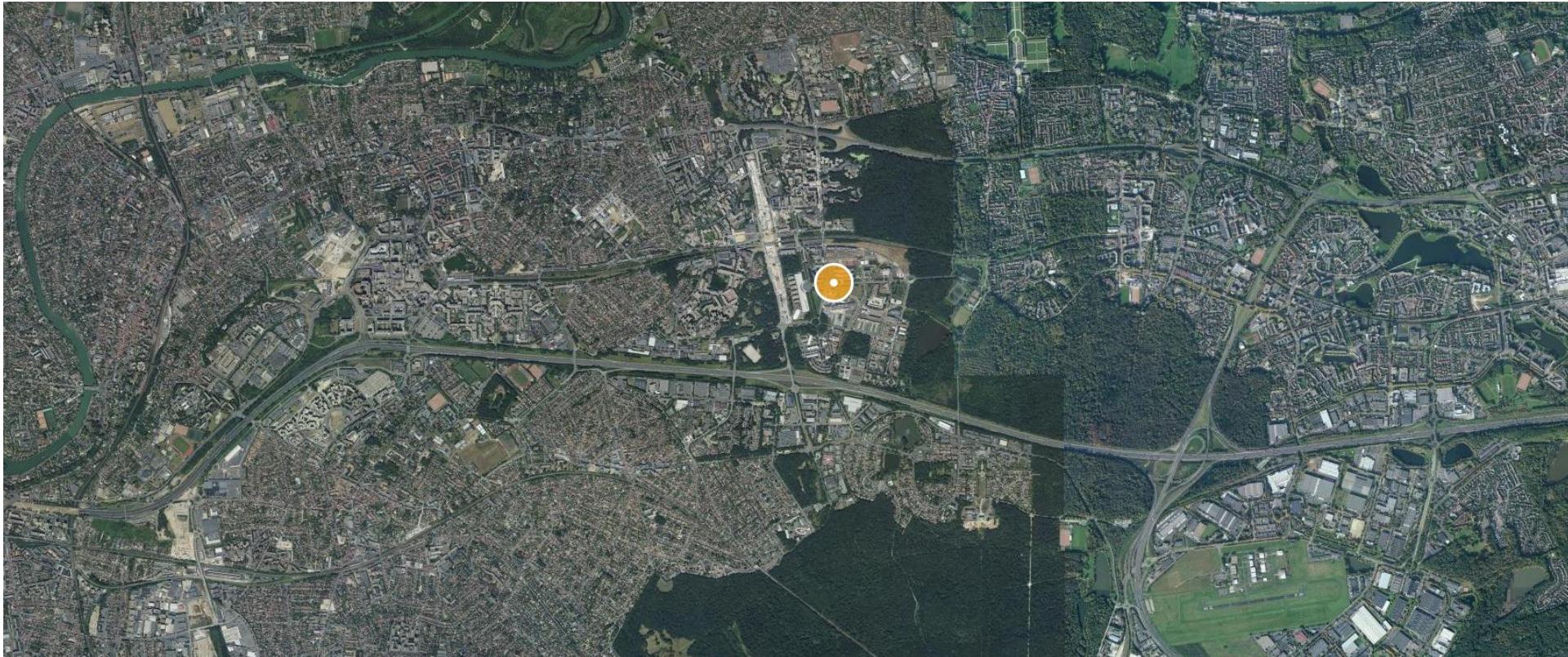
# Qu'est-ce que vous voyez ?



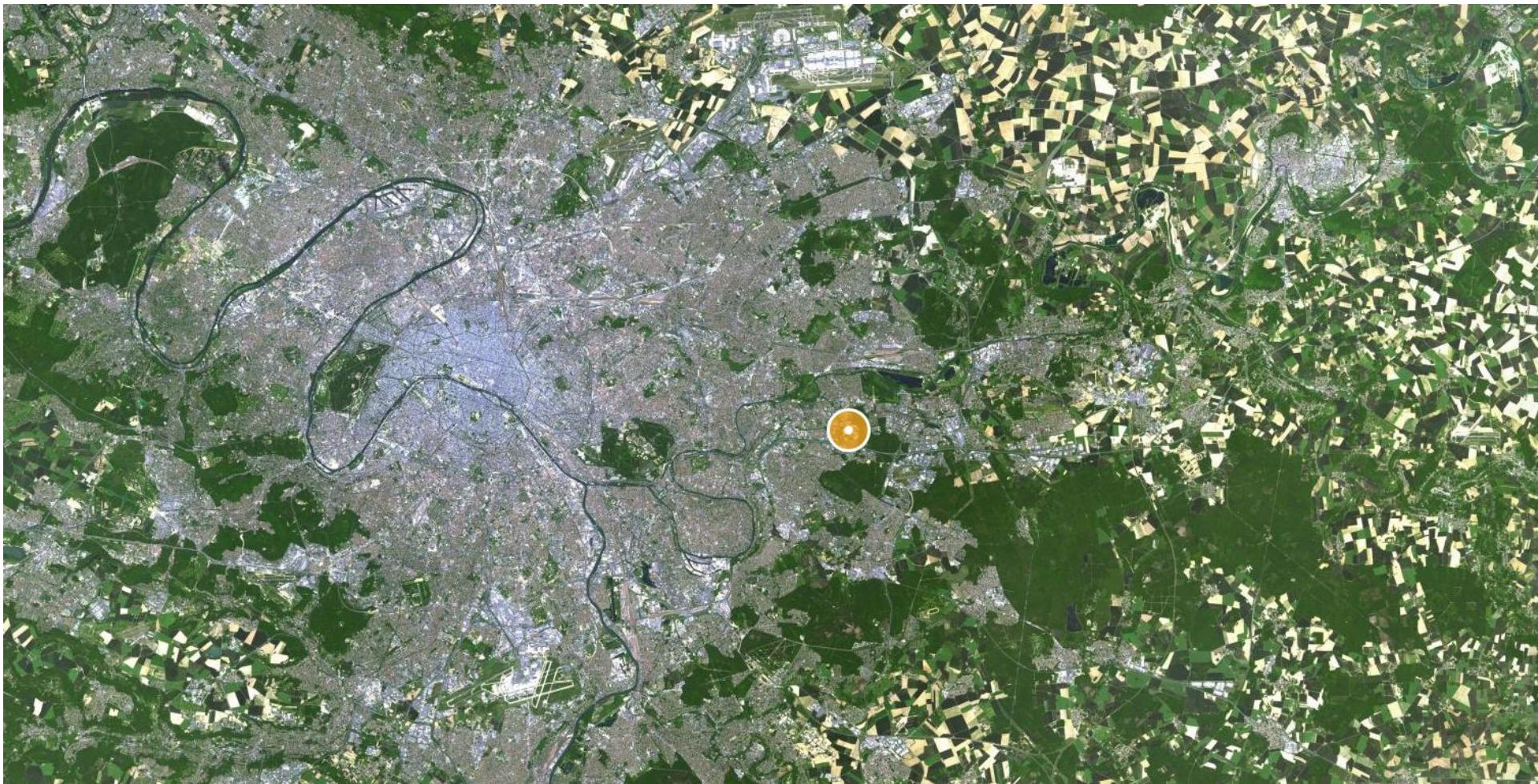
# Qu'est-ce que vous voyez ?



# Qu'est-ce que vous voyez ?



# Qu'est-ce que vous voyez ?



# MISE EN ROUTE TRANQUILLE

# **Qu'est-ce que vous savez déjà ?**

---

- En analyse de données ou Traitement d'Images
- De l'année dernière ?
- De cette année ?

# INTRODUCTION

# Définition

## ■ Segmenter une image/un nuage de points 3D:

Partition en régions homogènes de la donnée en entrée selon un ou plusieurs critères déterminés

- Couleur
- Texture
- Luminosité
- Profondeur
- Altitude

- ▶ Région = groupe de pixels **connexes**
- ▶ Bords ou contours d'une image = **frontières** entre ces régions
- ▶ Détection de contours = approche duale à la segmentation en régions.
- ▶ Segment = 1 ensemble de pixels ou de points 3D **regroupés spatialement**

# Définition

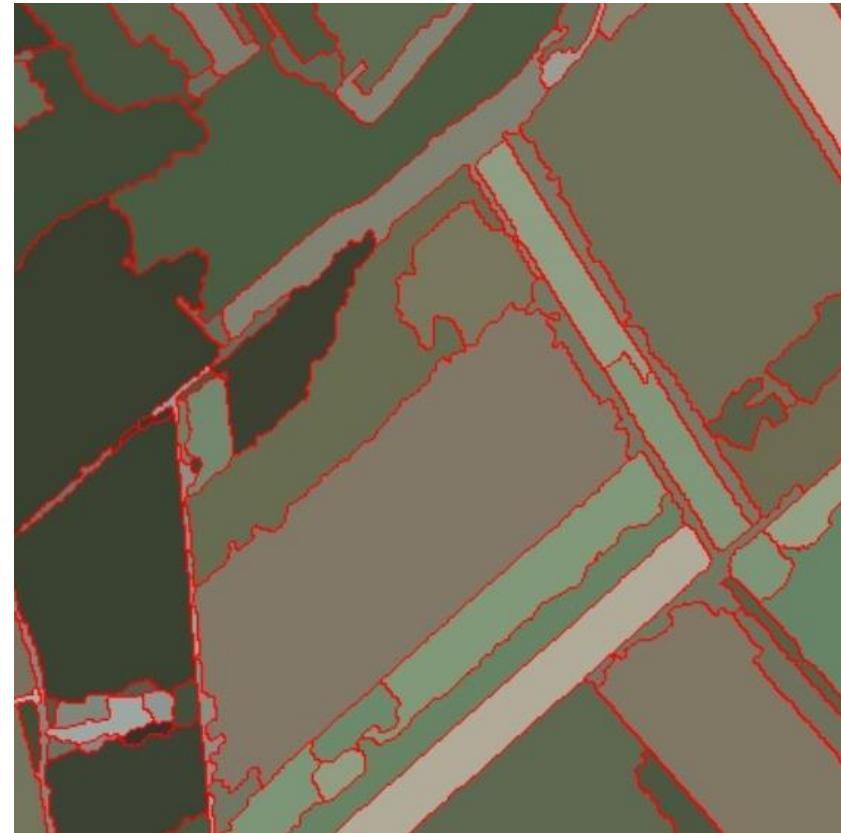
---

- Différence entre segmentation et classifications ?

- ▶ VS Classification supervisée ?
- ▶ VS Classification non supervisée ?

# A quoi ça sert ? (1)

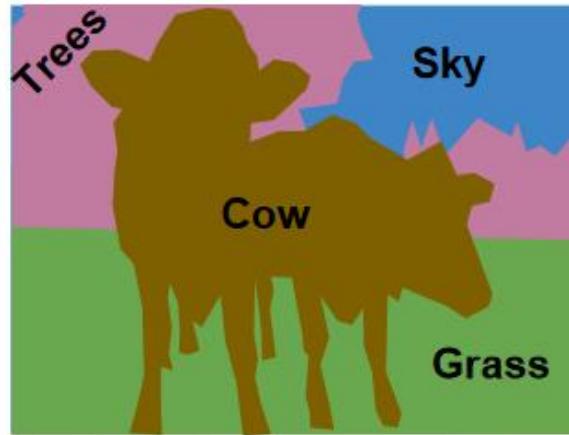
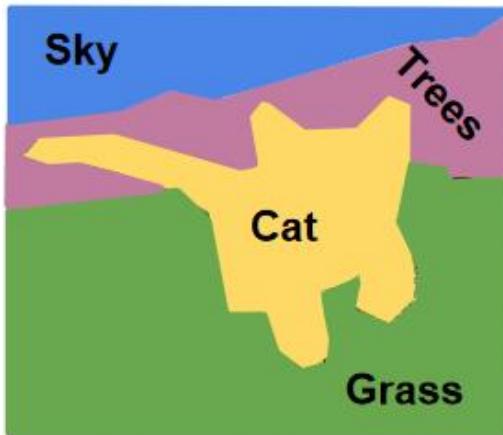
- A trouver les **frontières** des objets : délimitation précise.



Détection d'objets → classification plus facile

# A quoi ça sert ? (1)

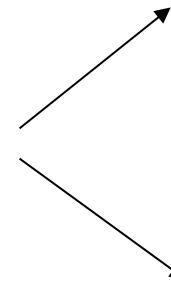
- A trouver les **frontières** des objets : délimitation précise.



Détection d'objets → classification plus facile

# A quoi ça sert ? (1)

► A trouver les **frontières** des objets : délimitation précise.



Détection d'objets → changement dans le rendu

# A quoi ça sert ? (1)

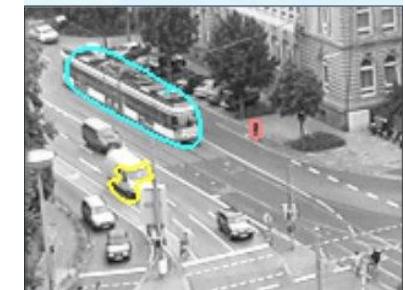
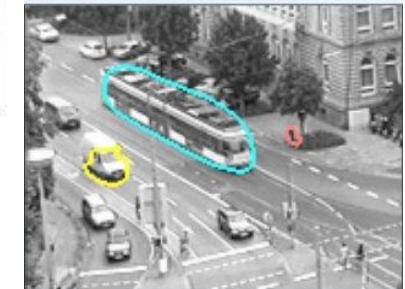
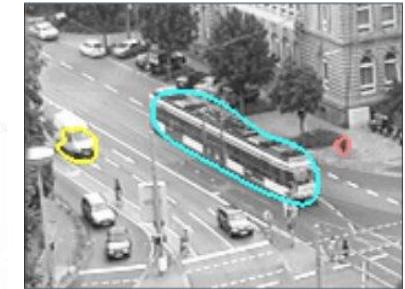
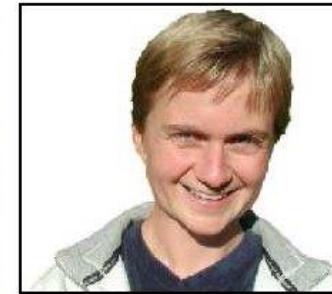
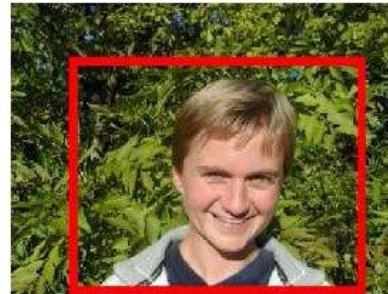
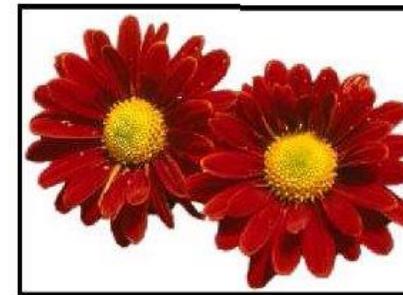
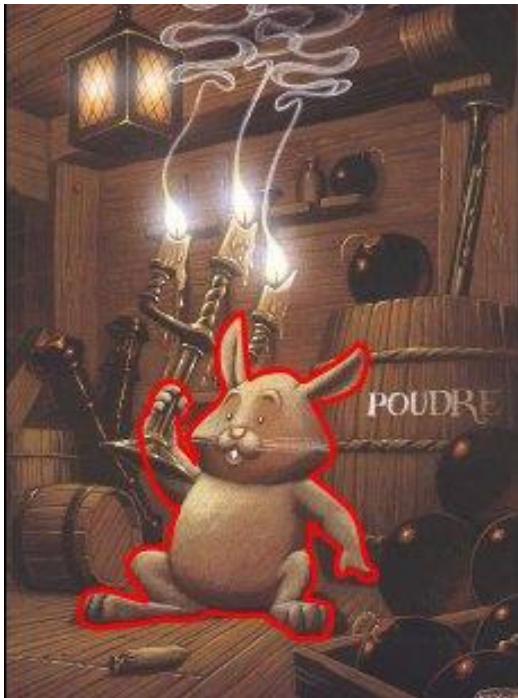
- A trouver les **frontières** des objets : délimitation précise.



Détection d'objets → changement dans le rendu

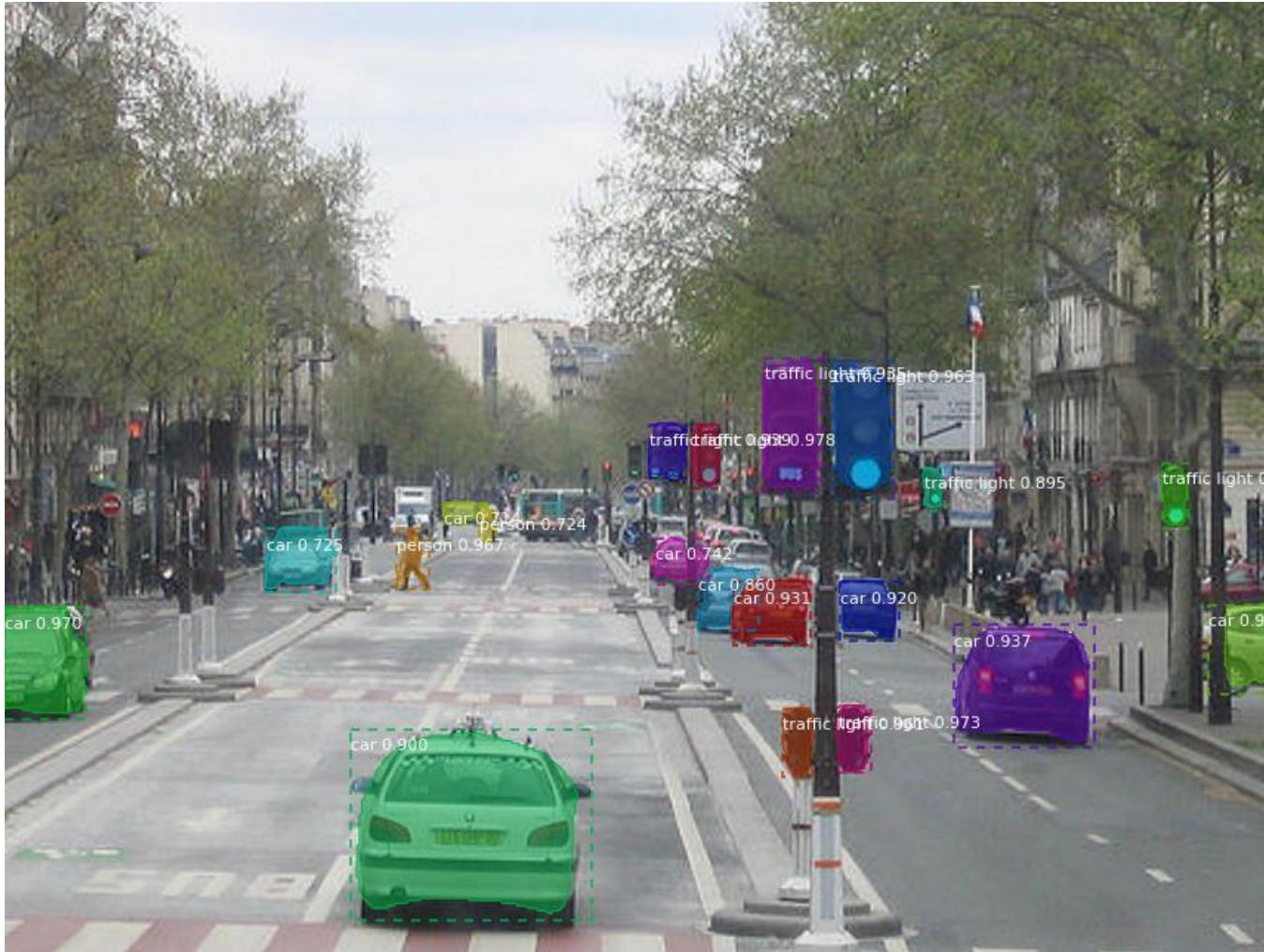
# A quoi ça sert ? (1)

► Détection/suivi d'objets : séparation avant/arrière



# A quoi ça sert ? (1)

- A ne pas confondre avec des détections d'objets grossières.



# A quoi ça sert ? (2)

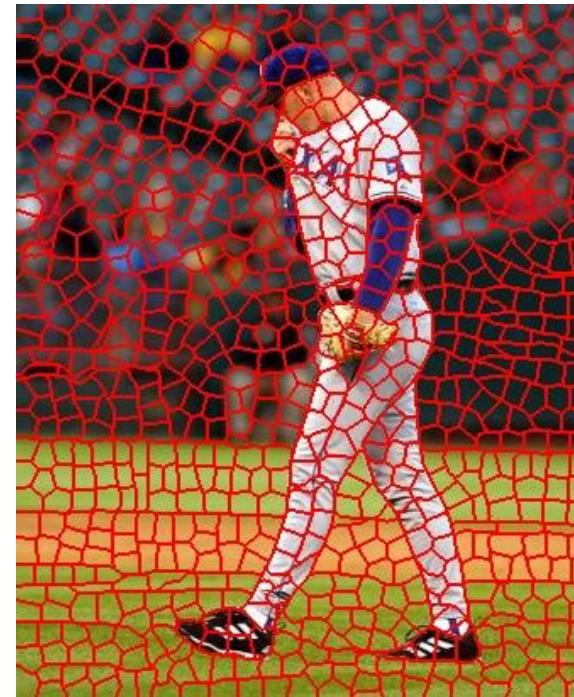
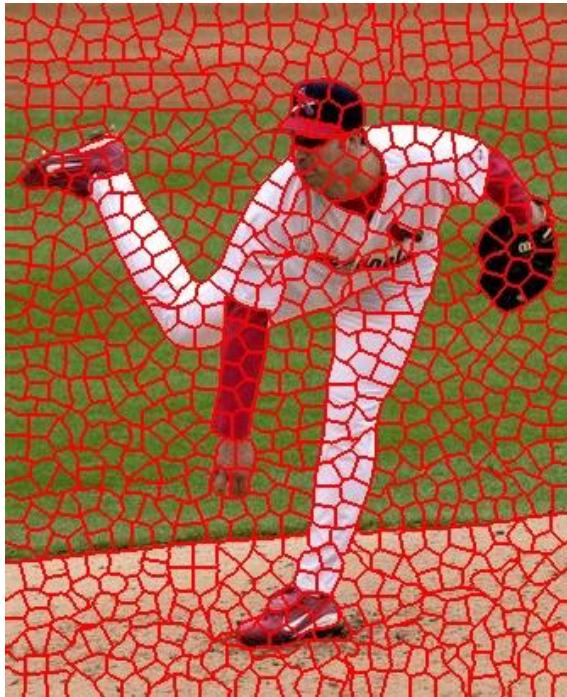
- A découper une image en petites régions



**Superpixels** → classification plus facile

# A quoi ça sert ? (2)

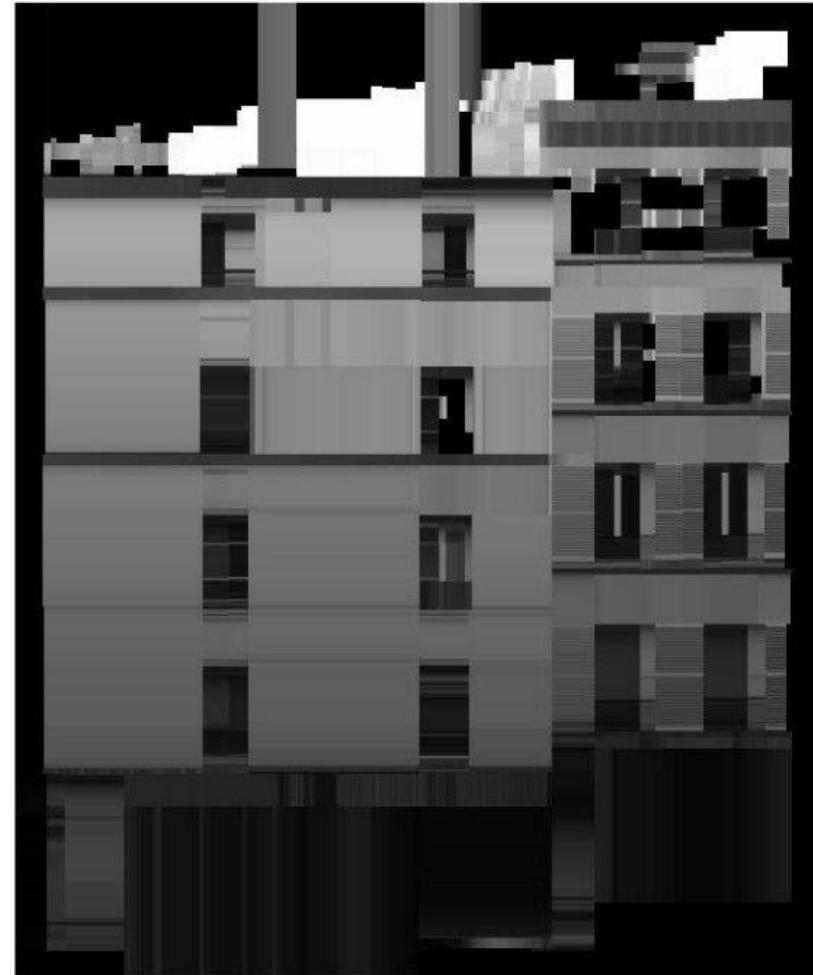
## ► Calcul d'attributs pour la classification : notion de voisinage



**Superpixels** → classification plus facile

# A quoi ça sert ? (2)

## ► Représentation compacte de la donnée

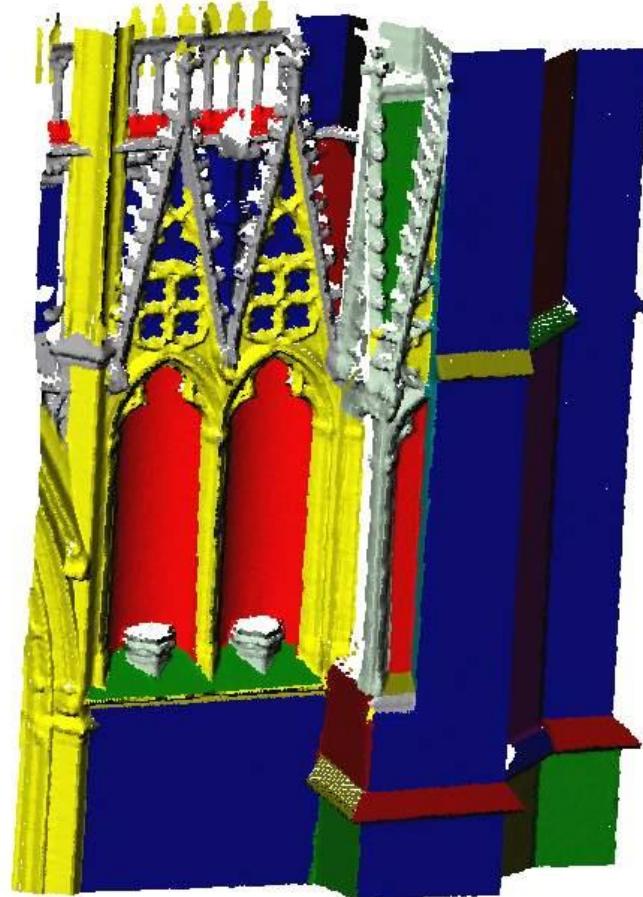


# A quoi ça sert ? (2)

► Analyse orientée-objet, fusion de données



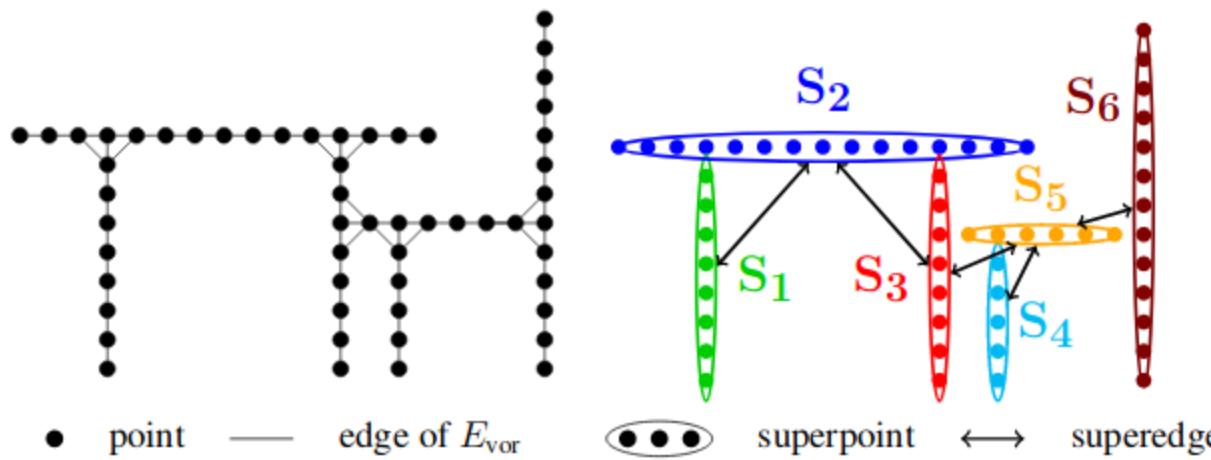
1. Segmentation image 2D



2. Reconstruction lidar 3D

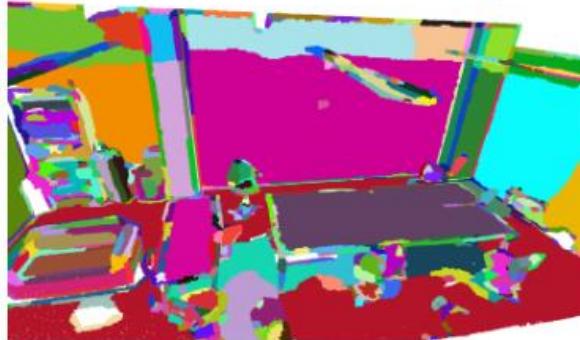
# A quoi ça sert ? (3)

► Analyse orientée-objet : les relations de voisinage entre segments apportent une information supplémentaire pour la classification



# A quoi ça sert ? (3)

► Analyse orientée-objet : les relations de voisinage entre segments apportent une information supplémentaire pour la classification



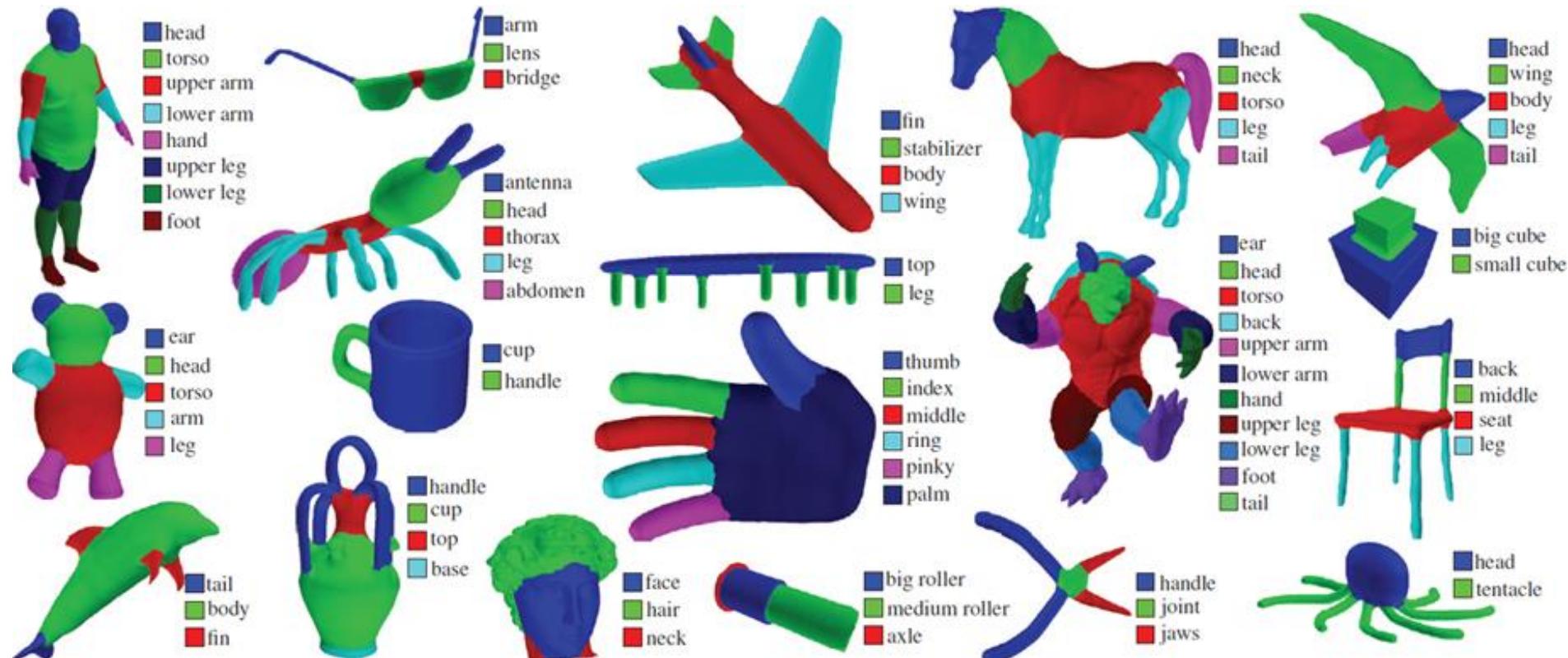
1. Nuage de points 3D+RGB

2. Segmentation 3D

3. Classification 3D

# A quoi ça sert ? (3)

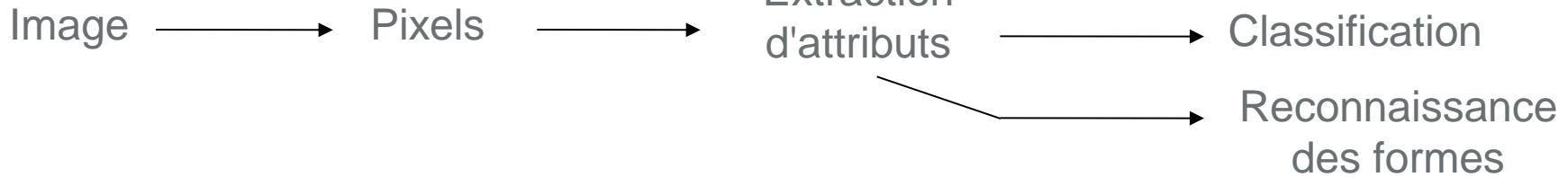
► Analyse orientée-objet : les relations de voisinage entre segments apportent une information supplémentaire pour la classification



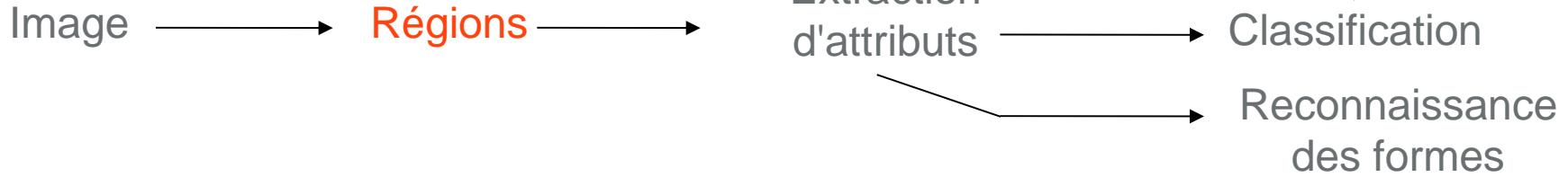
Analyse de maillage 3D : attribut initial → courbure

# Chaîne de traitements

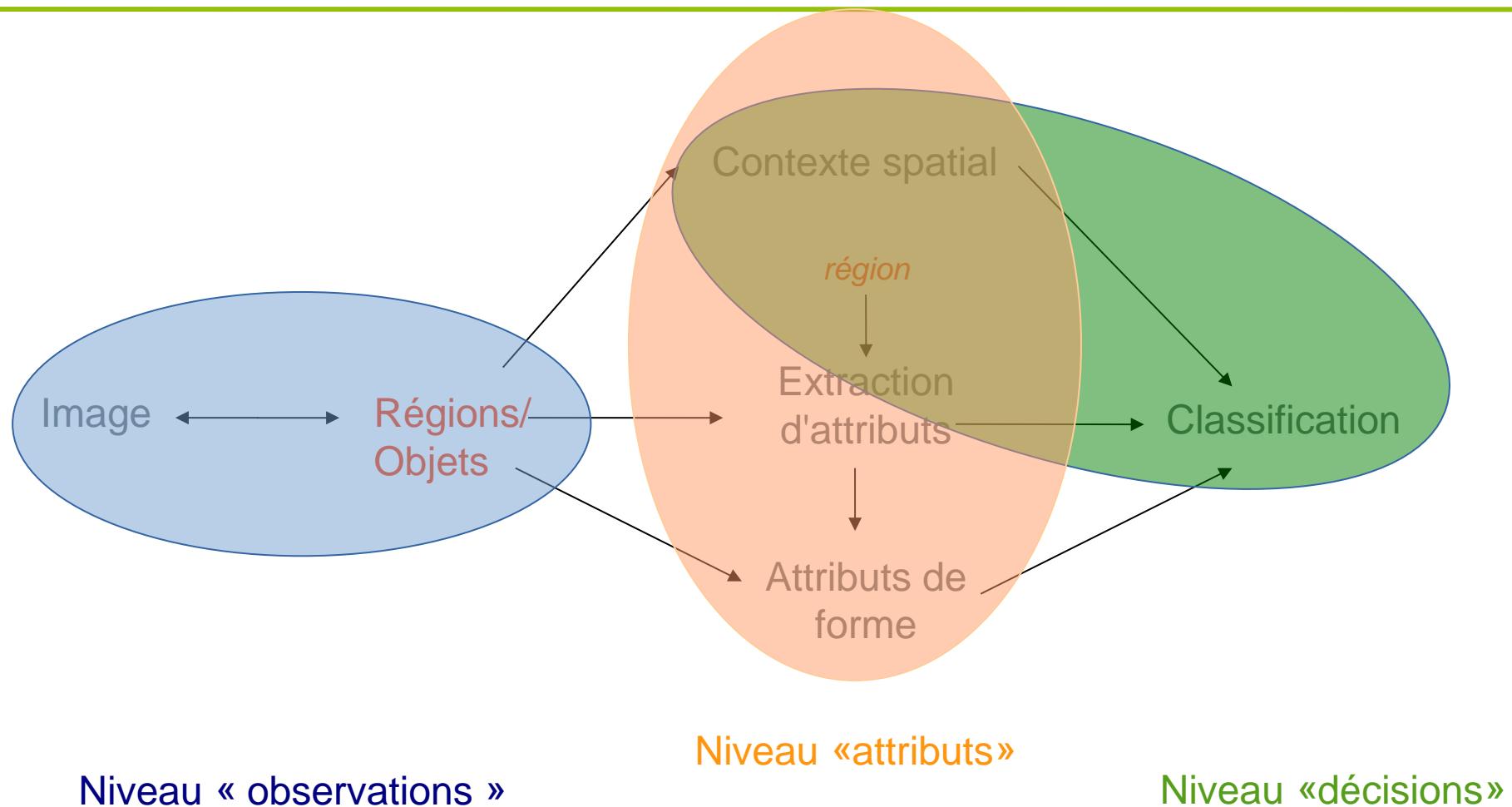
## Sans segmentation



## Avec segmentation



# Chaîne de traitements



# Segmentation en télédétection

- Simplification de l'image :

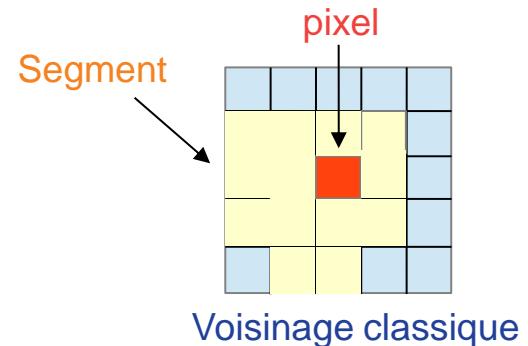
- Compression
- Moins d'instances à classer (30-100x moins)

- Accession à d'autres informations :

- Paramètres de forme de la région → nouveaux attributs (moyenne, écart-types)
- Meilleure définition d'un voisinage pour chaque pixel

- Régularisation :

- Des observations
- Des attributs (texture, couleur)
- Des décisions (classifications moins bruitées)



# Difficulté

- **Problème mal posé**

Notion de bonne segmentation/détection de contours dépend fortement du type d'images/nuages de points 3D à traiter et des applications envisagées.

- **Problème difficile**

Piège : définit un prédicat d'homogénéité

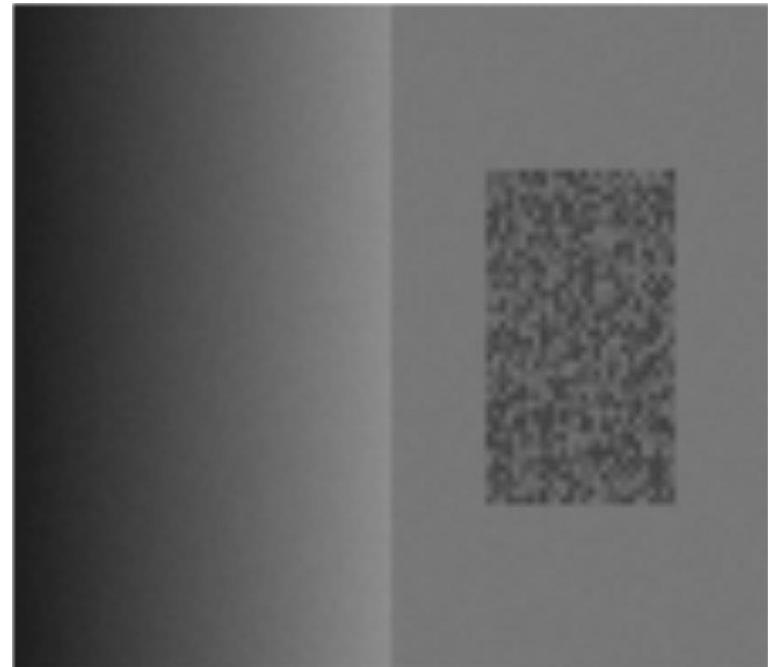
Nature de l'image : bruit, éclairage, complexité

- images optiques aériennes/satellites : Haute Résolution ↔ Très Haute Résolution
- panchromatiques ↔ hyperspectrales, radar, lidar etc.

# Difficulté

- Problème facile pour un être humain

- Connaissances
- Déductions
- Image vue dans sa totalité  
→ Photo-interprétation



Combien de régions ?

Où sont les plus forts gradients radiométriques ?

# Difficulté en conditions réelles



# Définition théorique

- **Segmentation** = partition d'un ensemble  $E$  en un ensemble  $k$  de régions  $R_k$  satisfaisant :
  - Tout élément de  $E$  appartient à une région :  $E = \bigcup_k R_k$
  - Aucun pixel n'appartient à plusieurs régions,
  - Région = ensemble connexe de pixels (cohérence spatiale),
  - Attributs des régions cohérents selon un prédicat donné  $P$  :
    - Pour tout  $R_k$ ,  $P(R_k)$  est vrai
    - Pour deux régions  $R_i, R_j$  adjacentes,  $P(R_i \cup R_j)$  est faux.

# Trois grands types d'approches

## ► Approches « régions »

- Ensemble connexe de pixels/points 3D ayant des propriétés communes

## ► Approches « contours »

- Recherche des points frontières entre 2 zones homogènes mais distinctes

## ► Approches mixtes

- Fusion pour tirer avantage des deux approches
- C'est ce qui marche le mieux (*deep learning*)

# Approches régions

- Quatre grandes familles :

(1) Méthodes par classification

→ Analyse de données / attributs

(2) Méthodes par transformation de régions

→ Division/fusion selon un (des) critère(s)

(3) Méthodes morphologiques

(4) Méthodes structurelles par graphes

## **2. ANALYSE DE DONNÉES**

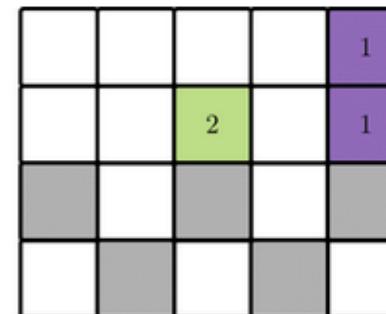
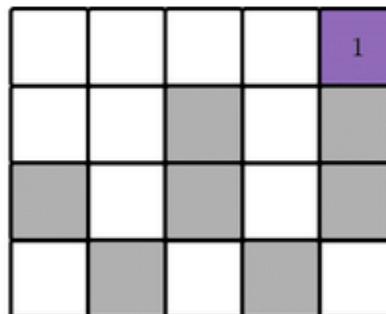
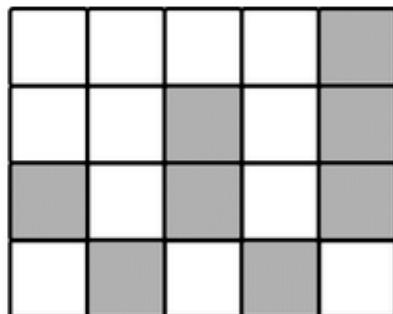
# Méthodes par analyse de données

## ► Méthode la plus simple :

- Lancer un algorithme de classification non-supervisée
  - Standard : k-moyennes/*k-means*
- Regrouper les régions en composantes connexes

## ► Et si on ne connaît rien en classification ?

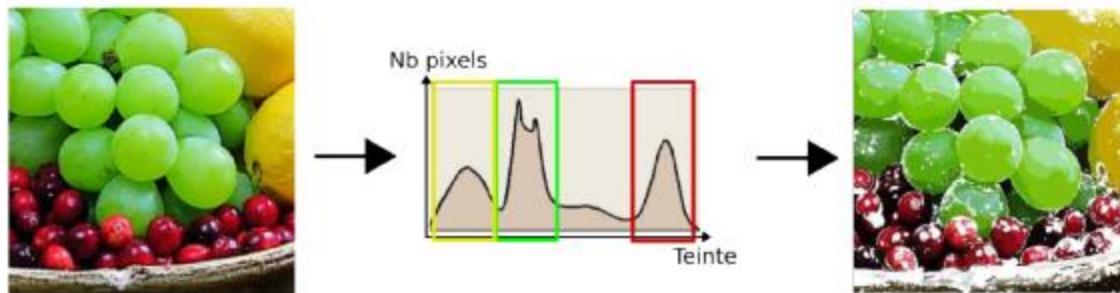
- Choisir un seuil sur un attribut pour séparer les pixels en 2 groupes
- (Recommencer avec un autre sur un des deux sous-ensembles si besoin)
- Regrouper les régions en composantes connexes



# Approches sur histogrammes

## ► Principe :

- Déetecter les modes d'un histogramme (traitement du signal)
- Calcul/choix de l'attribut le plus discriminant

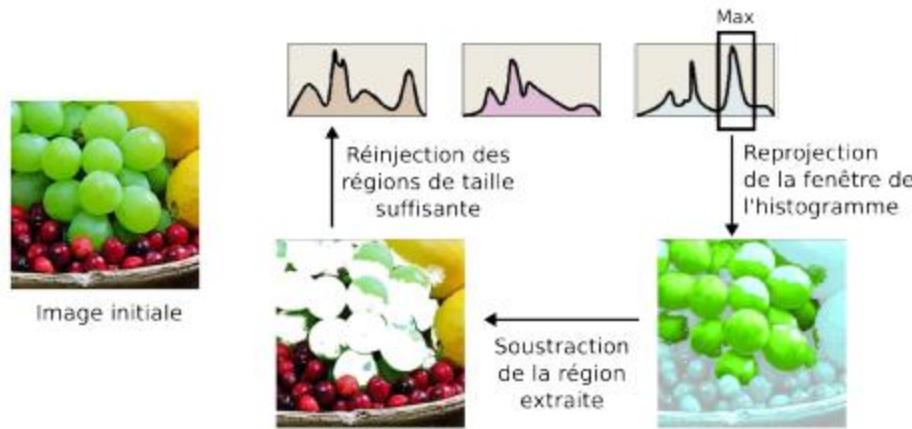


- Avantages et inconvénients :
  - + Simples et rapides
  - + Adaptées à plusieurs dimensions
  - Pas de critère de cohérence spatiale

# Approches sur histogrammes

## ■ Avec plusieurs histogrammes:

- Une possibilité : être récursif
- Élément décrit par plusieurs canaux



Quel est le problème ?

# Approches sur histogrammes

## ▪ Seuillages d'histogrammes:

- *Binarisation* :
  - Recherche du seuil optimal unique
  - Obtention d'une image binaire, et pas d'une segmentation
- *Seuillage multi-niveaux*
  - N seuils optimaux avec n pas forcément connu → méthodes statistiques
- *Seuillage local* : calcul d'un seuil optimal à partir de caractéristiques locales : corrélation entre pixels, distributions de probabilité

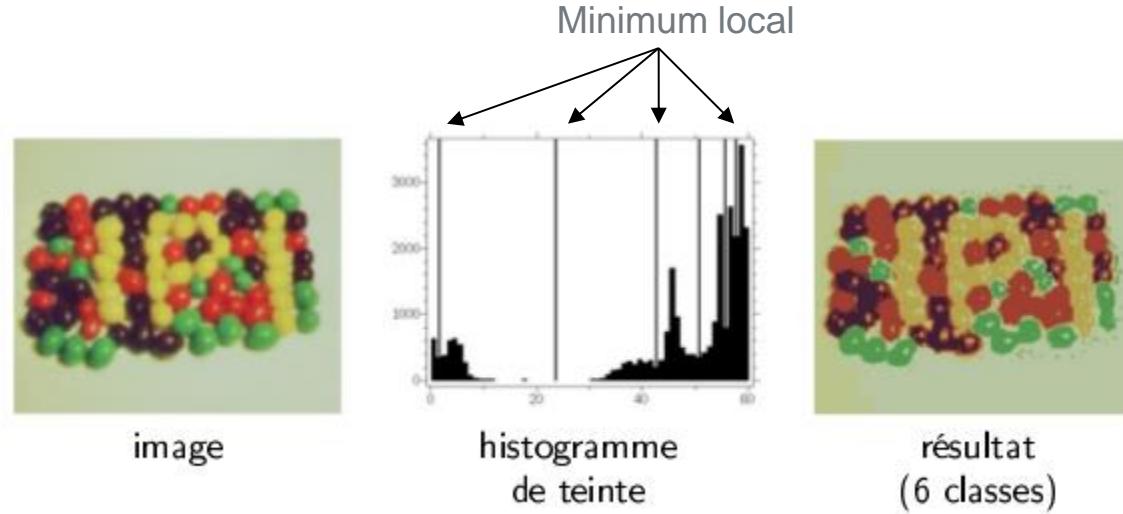
# Approches sur histogrammes

- Seuillage binaire : je coupe à une valeur donnée



# Approches sur histogrammes

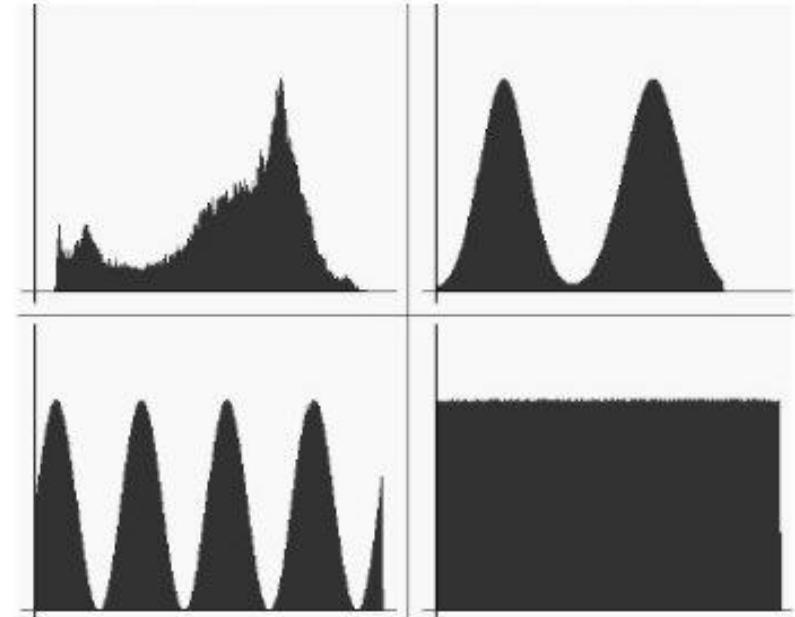
## ■ Seuillage multi-niveaux:



- Plus de seuils : sur-segmentation
- Moins de seuils : sous-segmentation
  - Devient difficile quand les moyennes se rapprochent.

# Approches sur histogrammes

- **Problème** : une image peut avoir un histogramme quelconque sans que sa signification en soit affectée



# **3. TRANSFORMATION DE RÉGIONS**

# Principe

- **Méthodes agrégatives**
  - On regroupe les deux ensembles/régions les plus proches
  - Au sens d'une énergie
  - Tant qu'on est satisfait (à définir) : homogénéité et/ou taille
- **Méthodes par division** :
  - On divise une région en plusieurs sous-régions
  - Avec des algorithmes binaires de type seuillage d'histogramme
  - Tant qu'on est satisfait (à définir) : homogénéité et/ou taille

# Croissance de région

- Idée :

- On part d'un point amorce (graine)
- On l'étend en ajoutant les éléments de la frontière satisfaisant un critère d'homogénéité



- Nécessite :
  - Sélection de graine(s)
  - Critère d'agrégation
  - (Souvent) un critère d'arrêt.

# Croissance de région

- Choix des germes: régions à faible contraste (minimum local)
- Agrégation :
  - $| \text{valeur\_pixel} - \text{moyenne(région)} | < \text{seuil}/n\sigma^2$
  - Région où  $\text{var}(R_i \cup R_j) < \text{seuil}$
  - Région où  $\text{var}(R_i \cup R_j) - \text{var}(R_i) - \text{var}(R_j)$  minimale
  - Faible saut de valeur\_pixel le long du bord commun de deux régions
  - Prédicat dont on réduit la sévérité progressivement
- Arrêt : seuil sur la variance
- Inconvénients :
  - Pas de partition de l'image
  - Performances moyennes si texture/bruit
  - Heuristique.

# Croissance de région

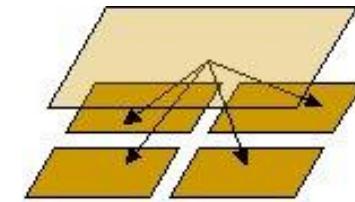
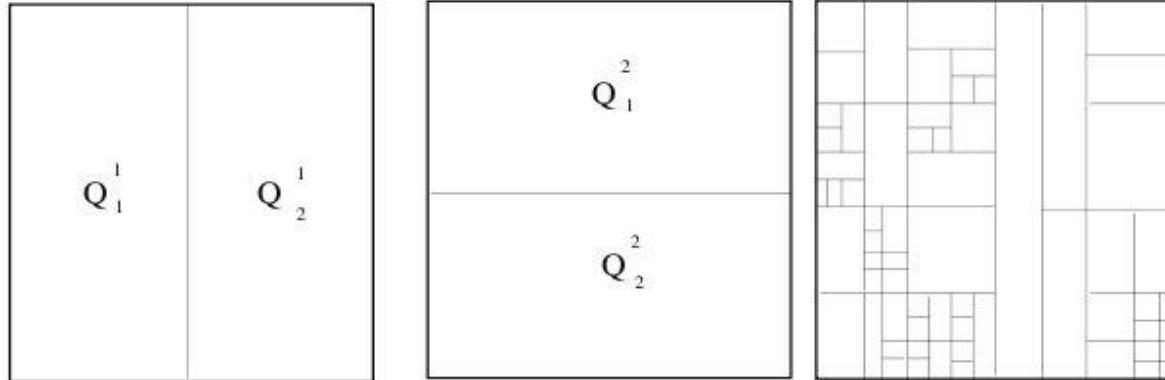
- Si les frontières sont mal définies, résultat mauvais.



# Partage de régions

- **Idée** : approche Top-Down

- Subdivisions successives de l'image entière tant que le prédictat n'est pas vérifié → création d'un arbre.



- **Inconvénients :**

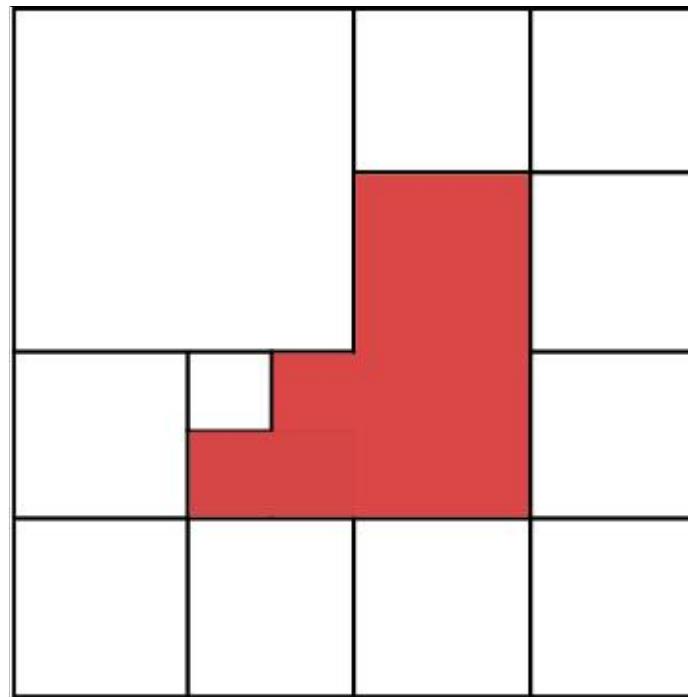
- Stratégies rigides
- Performances moyennes si texture/bruit
- Heuristique.

# Regroupement de régions

- **Idée** : approche Bottom-Up
  - Regroupement des régions à partir d'une partition fine: pex 2x2 pixels ou une sur-segmentation initiale.
- Critère de fusion :
  - Tests statistiques
  - Frontière peu contrastée
  - Amélioration d'un facteur de forme (compacité)
- Inconvénients :
  - Dépend de l'ordre de fusion des régions

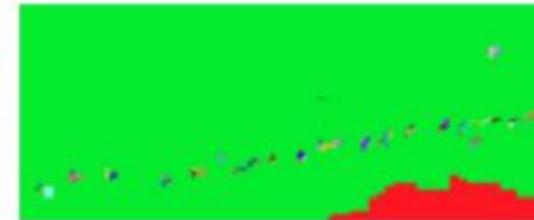
# Méthode hybride : partitionnement/regroupement

- Idée : approche Top-Down puis Bottom-Up (Split-and-Merge)



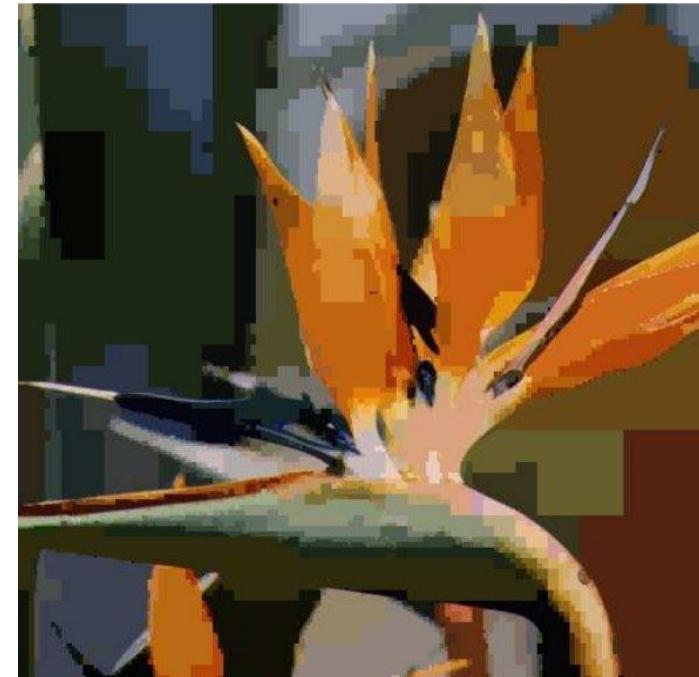
# Méthode hybride : partitionnement/regroupement

- Avantage :
  - Méthode hybride locale/globale : compensation des faiblesses respectives des 2 méthodes
- Inconvénient :
  - Découpage un peu carré.



# Méthode hybride : partitionnement/regroupement

- **Avantage :**
  - Méthode hybride locale/globale : compensation des faiblesses respectives des 2 méthodes
- **Inconvénient :**
  - Découpage un peu carré.



# **4. METHODES MORPHOLOGIQUES**

# Ligne de Partage des Eaux (LPE)

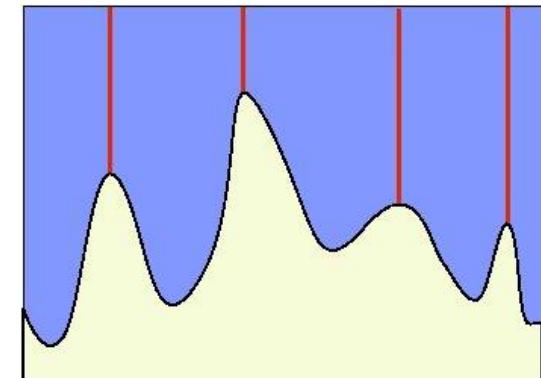
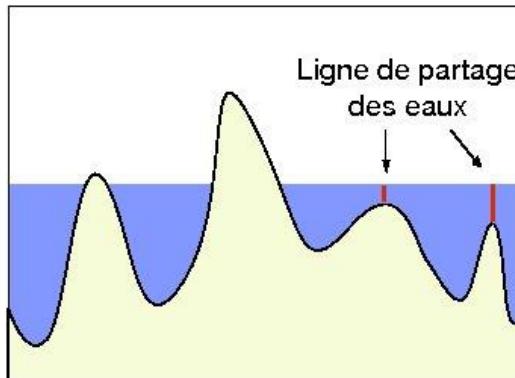
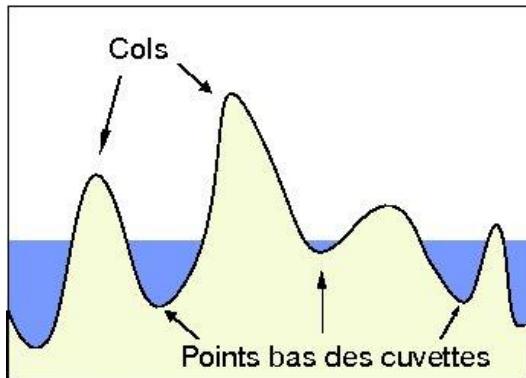
- Analogie :                   image → surface topographie  
                                     niveaux de gris → altitude
- Application à ce relief des notions de **LPE** et de bassins versants
- Idée 1 :
  - Étude comportement écoulements d'eau imaginaires sur relief.
  - Goutte d'eau → finit dans un minimum local.
  - Bassin versant = zone d'influence d'un minimum.
  - LPE = ligne séparant 2 bassins = position instable d'une goutte.

LPE = contours – Bassins versants = régions

# Ligne de Partage des Eaux (LPE)

## ■ Idée 2 : on inonde !

- On inonde la surface à partir des minima locaux, l'eau montant à vitesse constante et uniforme dans les bassins versants,
- Quand les eaux issues de 2 minima différents se rencontrent, on monte
- Une digue pour qu'elles ne se mélangent pas,
- A la fin de l'immersion, l'ensemble des digues constituent la ligne de partage des eaux.



# La LPE en pratique

## ■ Algorithme :

- Calcul du module de gradient
- Filtrage (pourquoi ces 2 étapes ?)
- Parcours des pixels par ordre inverse de leurs hauteurs
- Pour chaque niveau de hauteur :
  - Extension des bassins existants
  - Détection de nouveaux bassins
  - Formation de LPE par rencontre de bassins.

# La LPE en pratique

*[http://bigwww.epfl.ch/demo/ip/demos/watershed](http://bigwww.epfl.ch/demo/ip/demos/watershed/)*  
/

# La LPE en pratique

## ▪ Propriétés:

- Sur-segmentation
- Sensible au bruit
- Peu complexe :  $O(n)$
- Approche globale : on ne peut savoir si un point appartient à la LPE par analyse de son voisinage local

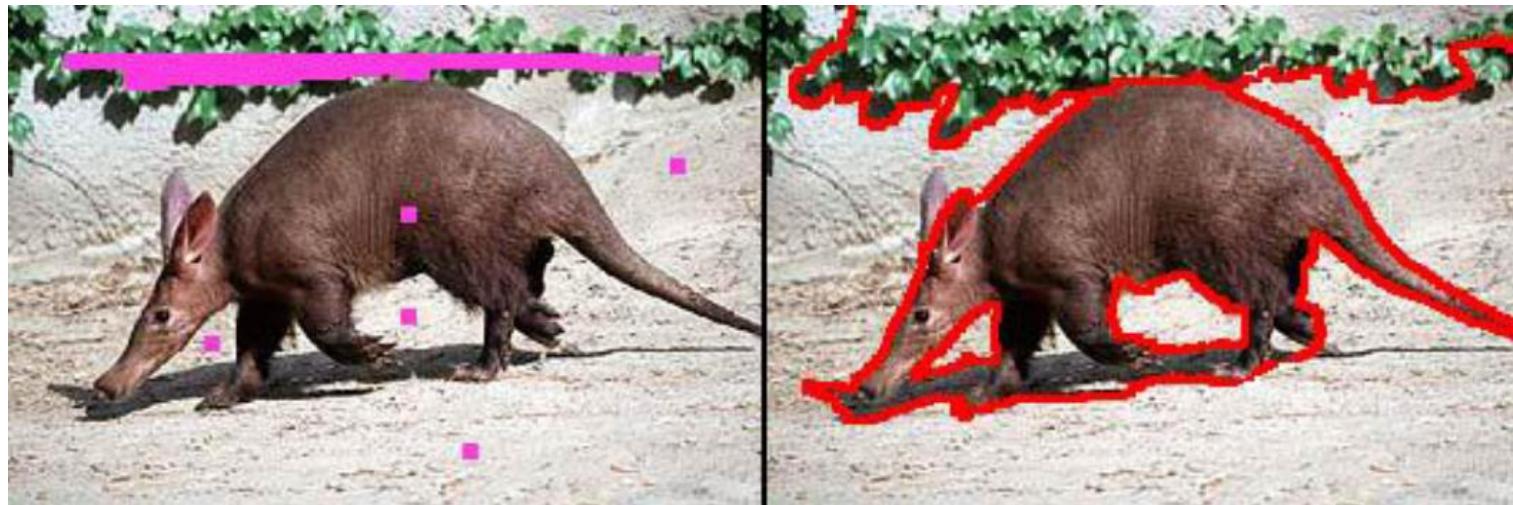
## ▪ Améliorations :

- Filtrage des minima non significatifs
- Transformation de données : inverser les hauteurs, rehaussement de contraste
- Utilisation de marqueurs → choix du nombre de zones.

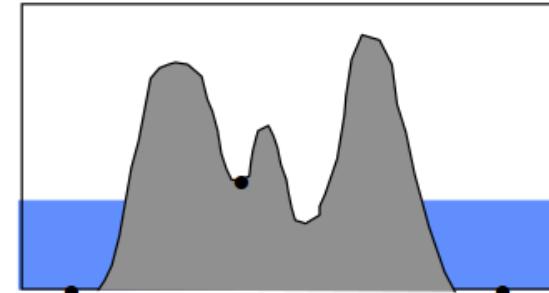
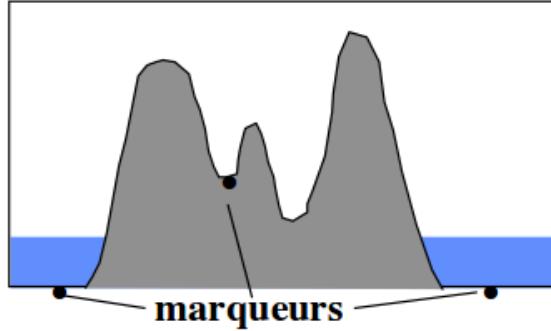
# Utilisation de marqueurs

## ■ Pour limiter la sur-segmentation

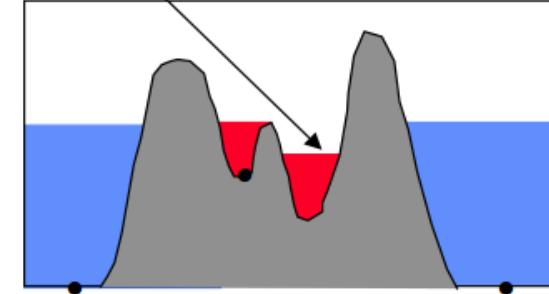
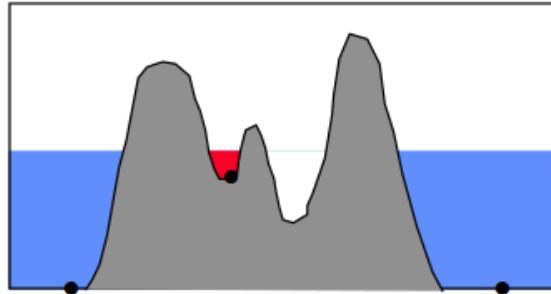
- Ne considérer que certains minima
- Marqueurs = zone de minima.
- Inondations des minima non significatifs
- Marqueurs automatiques ou manuels
- Comment les choisiriez les vous ?



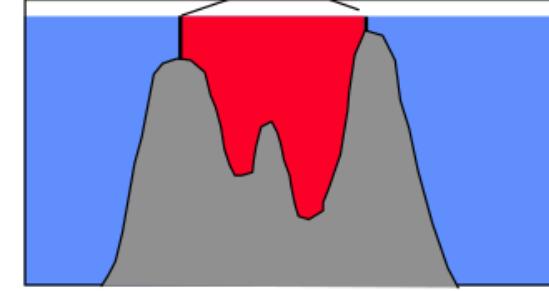
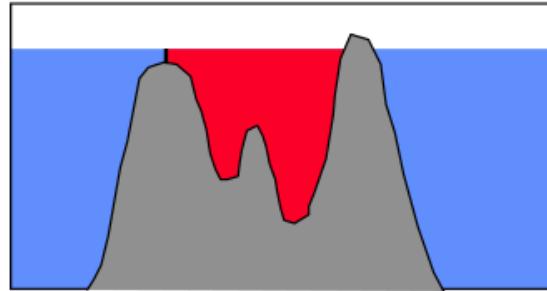
# Utilisation de marqueurs



bassin inondé à partir d'un bassin marqué



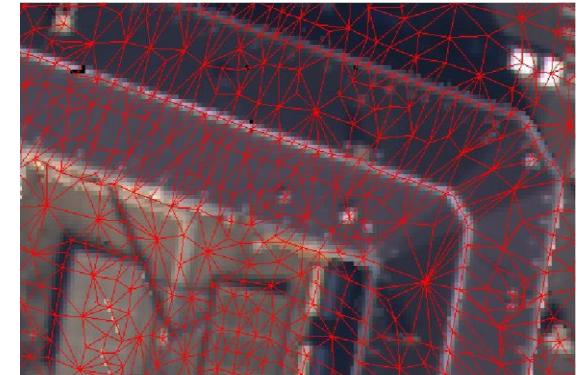
LPE



## **5. MÉTHODES PAR GRAPHE**

# Construction du graphe

- **Image** : graphe indirect pondéré dont
  - les nœuds sont les pixels ou les groupes de pixels (sur-segmentation)
  - Les arêtes les relations d'adjacence
- On cherche à séparer les nœuds en des ensembles disjoints
  - Grande similarité dans un ensemble
  - Faible entre eux
- On parle de *Graphe d'Adjacence de Régions.*
  - *Partition optimale : problème NP-complet*



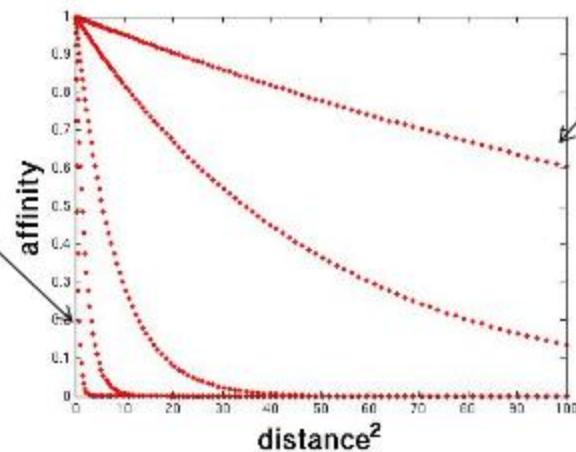
# Construction du graphe

- Poids des arêtes = mesure de différence ou d'affinité entre pixels

**Fortes valeurs** pour les pixels avec le même attribut et proches l'un de l'autre

$$aff(x, y) = \exp\left\{-\left(\frac{1}{2\sigma_d^2}\right)(\|x - y\|^2)\right\}$$

Petit sigma :  
grouper les points  
très proches

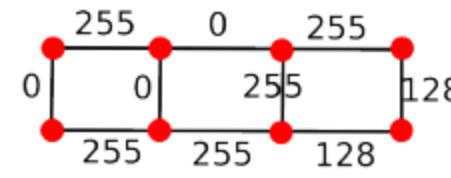


Grand sigma :  
grouper les points  
distants

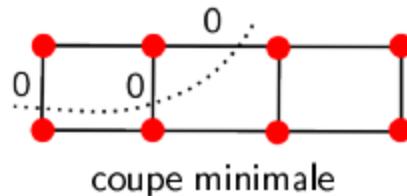
# Coupe minimale



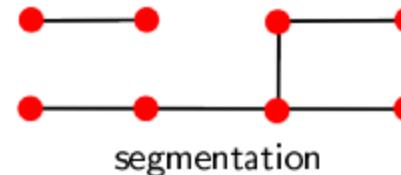
image



graphe des pixels



coupe minimale

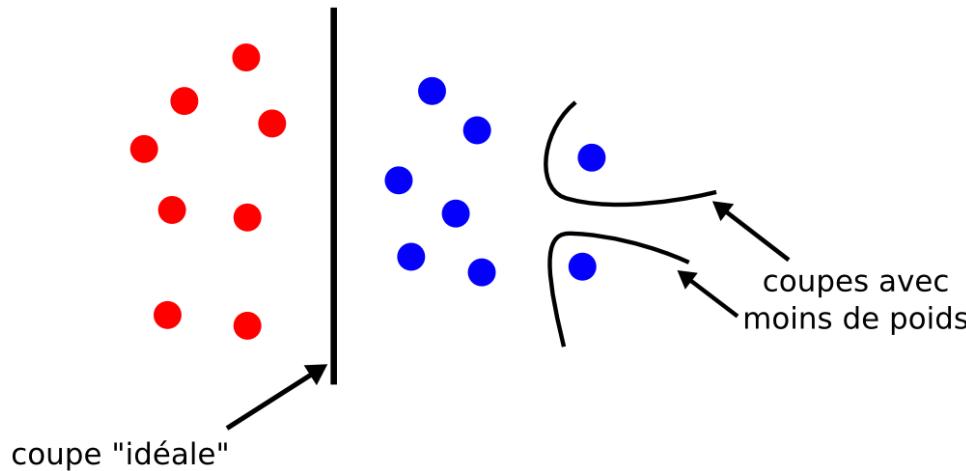


segmentation

A votre avis, quel est le problème ?

# Coupe minimale et normalisée

- Elle favorise la coupe de petites régions.
  - Le poids de la coupe est proportionnel au nombre d'arêtes.

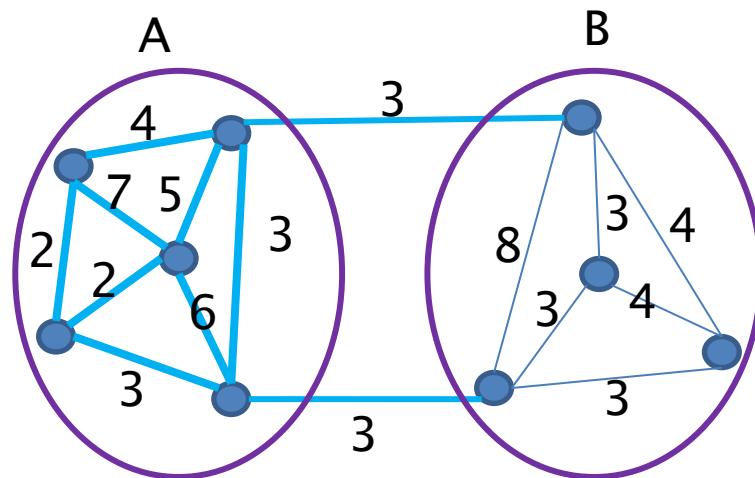


- Suppression de l'influence du nombre d'arcs dans la coupe :  
*coupe normalisée (Normalized Cuts)*

# Coupe minimale et normalisée

Mesure de l'association à l'intérieur d'un groupe

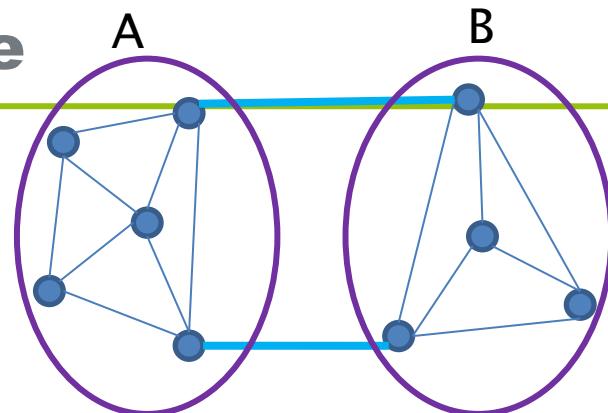
$$assoc(A, V) = \sum_{x \in A, y \in V} w(x, y)$$



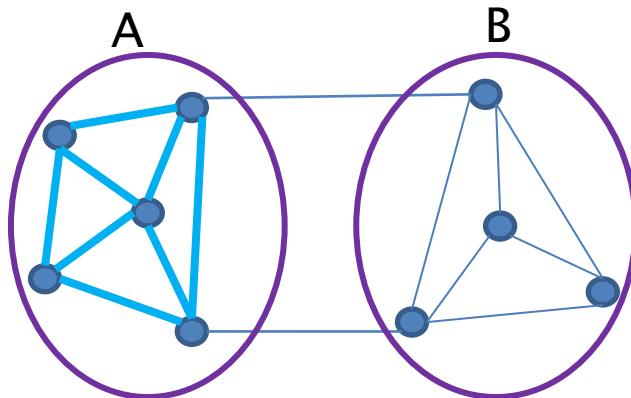
$$assoc(A, V) = 4 + 5 + 7 + 3 + 3 + 6 + 2 + 2 + 3 + 3 = \boxed{38}$$

## Coupe minimale et normalisée

$$cut(A, B) = \sum_{x \in A, y \in B} w(x, y)$$



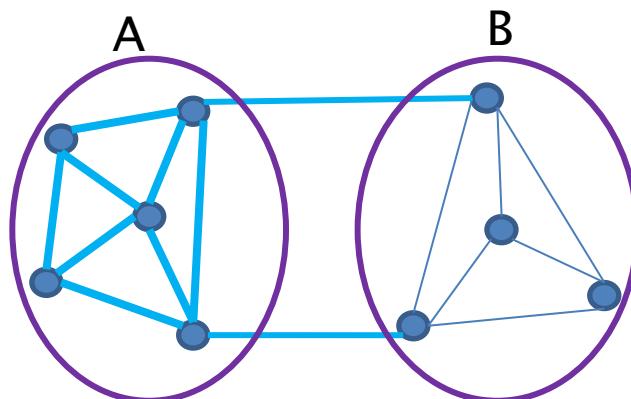
$$assoc(A, A) = \sum_{u_1, u_2 \in A} w(u_1, u_2)$$



$$assoc(A, V) = assoc(A, A) + Cut(A, B)$$

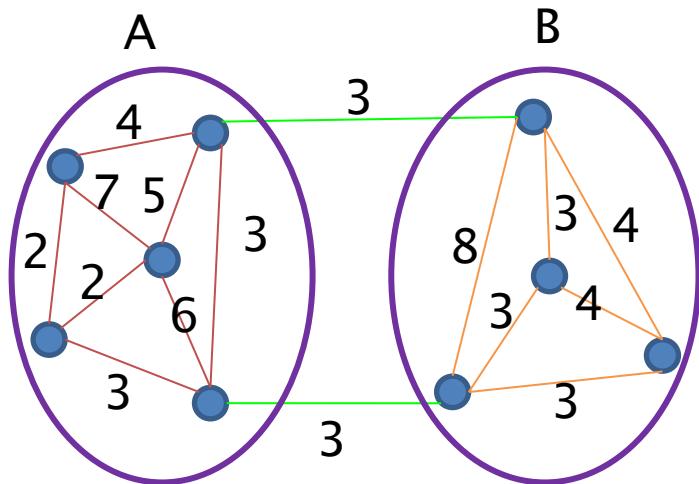
↓

$$assoc(A, V) = \sum_{u \in A, t \in V} w(u, t)$$



# Coupe minimale et normalisée

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}$$



$$\frac{6}{29+6} + \frac{6}{25+6}$$

Que l'on souhaite minimiser....

# Coupe minimale et normalisée

Minimize  $\frac{\text{cut}(A,B)}{\text{assoc}(A,V)} + \frac{\text{cut}(A,B)}{\text{assoc}(B,V)}$



= total connection from nodes  
in A to all nodes in graph (V)

Maximize  $\left( \frac{\text{assoc}(A,A)}{\text{assoc}(A,V)} \right) + \left( \frac{\text{assoc}(B,B)}{\text{assoc}(B,V)} \right)$

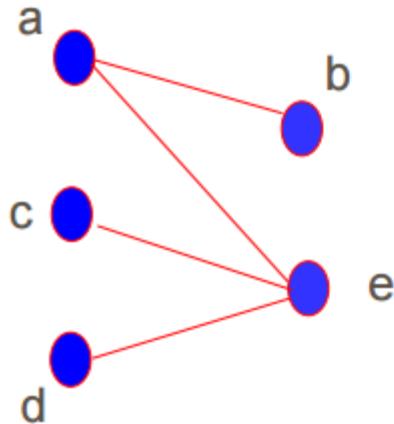
Minimiser le coût de séparation = maximiser l'association

$$Ncut(A,B) = 2 - Nassoc(A,B).$$

Problème NP-complet → on va chercher à atteindre une solution proche

# Coupe minimale et normalisée

Création d'une matrice d'adjacence



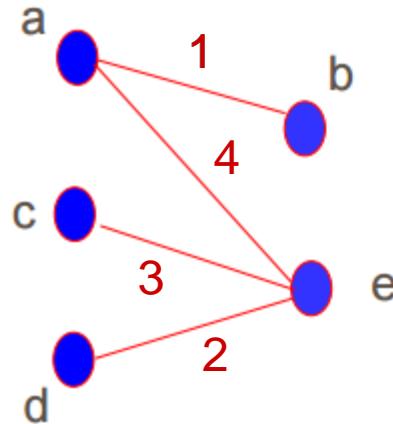
Une ligne par nœud

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Matrice d'adjacence

# Coupe minimale et normalisée

Création d'une matrice d'adjacence : ajout de **poids**



Une ligne par nœud

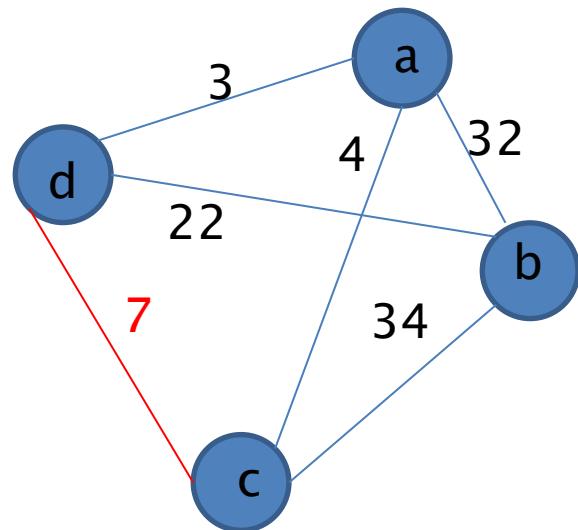
A red arrow points from the graph to the matrix below.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 4 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 2 \\ 4 & 0 & 3 & 2 & 0 \end{bmatrix}$$

Matrice d'adjacence W

# Coupe minimale et normalisée

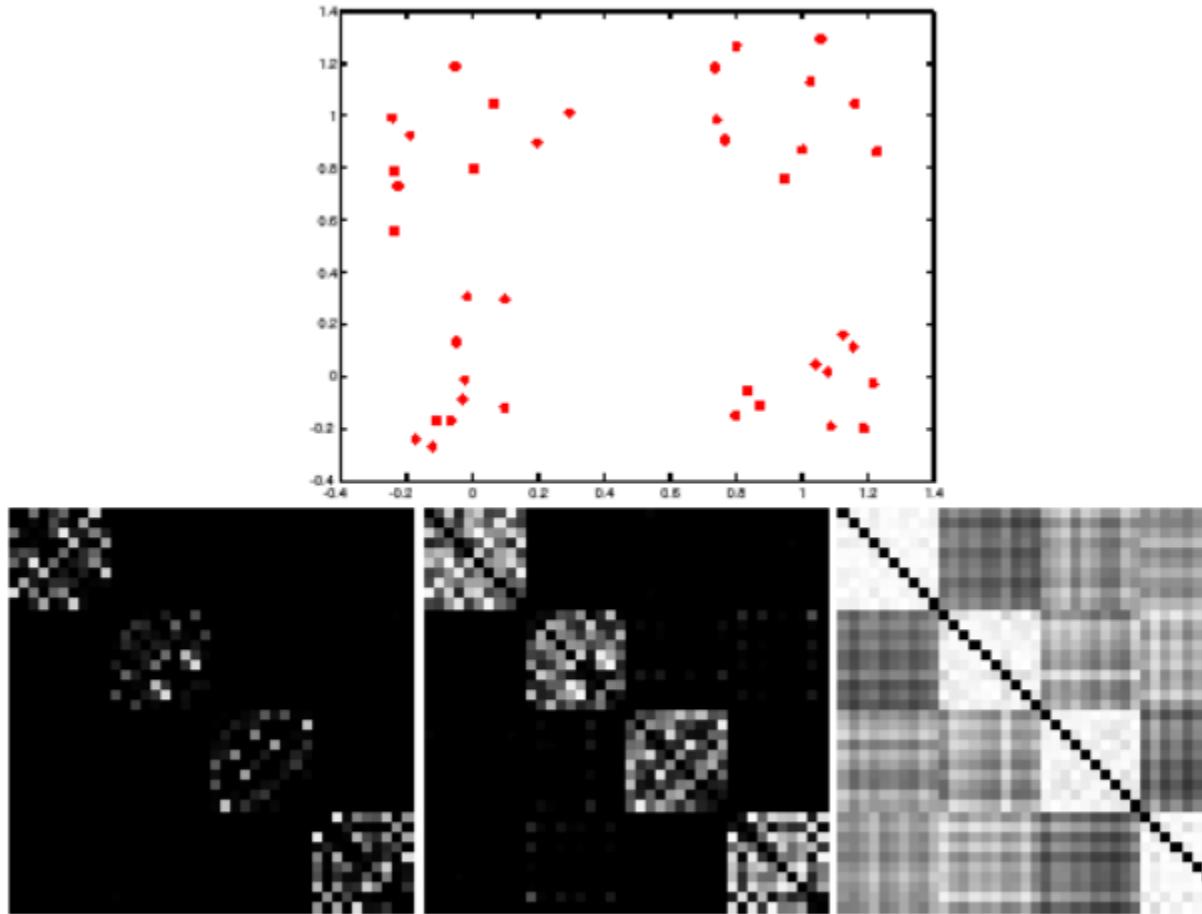
Création d'une matrice d'adjacence : ajout de **poids**



$$W \Rightarrow$$

$$\begin{pmatrix} 0 & 32 & 4 & 3 \\ 32 & 0 & 34 & 22 \\ 4 & 34 & 0 & 7 \\ 3 & 22 & 7 & 0 \end{pmatrix}$$

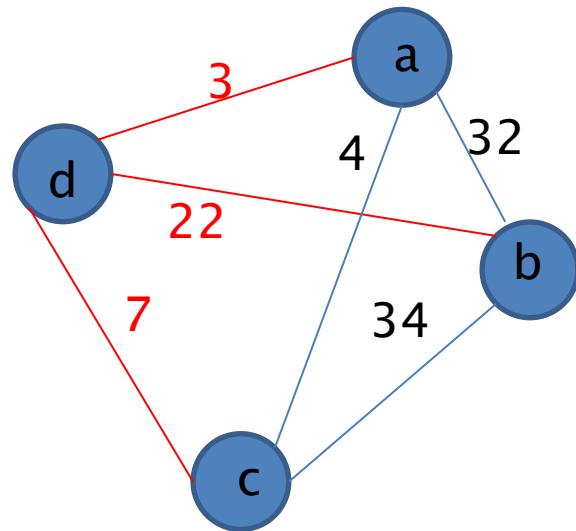
# Matrice d'affinité



Sigma de plus en plus grand

# Coupe minimale et normalisée

Création d'une matrice de degrés D



$$D \Rightarrow$$

$$\begin{matrix} & a & b & c & d \\ a & 39 & 0 & 0 & 0 \\ b & 0 & 88 & 0 & 0 \\ c & 0 & 0 & 41 & 0 \\ d & 0 & 0 & 0 & 32 \end{matrix}$$

# Coupe minimale et normalisée

Création d'une matrice laplacienne  $L = D - W$

$$\begin{pmatrix} 39 & 0 & 0 & 0 \\ 0 & 88 & 0 & 0 \\ 0 & 0 & 41 & 0 \\ 0 & 0 & 0 & 32 \end{pmatrix} - \begin{pmatrix} 0 & 32 & 4 & 3 \\ 32 & 0 & 34 & 22 \\ 4 & 34 & 0 & 7 \\ 3 & 22 & 7 & 0 \end{pmatrix} L \Rightarrow \begin{pmatrix} 39 & -32 & -4 & -3 \\ -32 & 88 & -34 & -22 \\ -4 & -34 & 41 & -7 \\ -3 & -22 & 7 & 32 \end{pmatrix}$$

1. Matrice symétrique, les vecteurs propres sont orthogonaux les uns les autres.
2. Les valeurs propres sont toutes non négatives.

# Coupe minimale et normalisée

Création d'une matrice laplacienne  $L = D - W$

La  $n$  valeurs propres positives, la plus petite est 0 et correspond au vecteur propre composé de 1.

$$L = \begin{pmatrix} 39 & -32 & -4 & -3 \\ -32 & 88 & -34 & -22 \\ -4 & -34 & 41 & -7 \\ -3 & -22 & -7 & 32 \end{pmatrix}$$

$$L \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = [(39 - 32 - 4 - 3)(-32 + 88 - 34 - 22)(-4 - 34 + 41 - 7)(-3 - 22 - 7 + 32)] \\ = [0000]$$

# Coupe minimale et normalisée

Il est montré qu'une solution approchée est obtenue en résolvant une équation de la forme :

$$(\mathbf{D} - \mathbf{W})\mathbf{y} = \lambda \mathbf{D}\mathbf{y}.$$

La solution  $\mathbf{y}$  est un vecteur propre de  $(\mathbf{D} - \mathbf{W}) = \mathbf{L}$ .

## Vecteur propre :

- Vecteur caractéristique d'une matrice ;
- Spécifie une segmentation selon les valeurs de ses composantes.
- → Points similaires = mêmes composantes de vecteurs propres

# Coupe minimale et normalisée

Le plus petit vecteur propre est toujours 0 :

–  $A=G$  ;  $B = \{ \}$  =>  $\text{CoupeNormalisee}(A,B)=0 \Rightarrow$  une seule région.

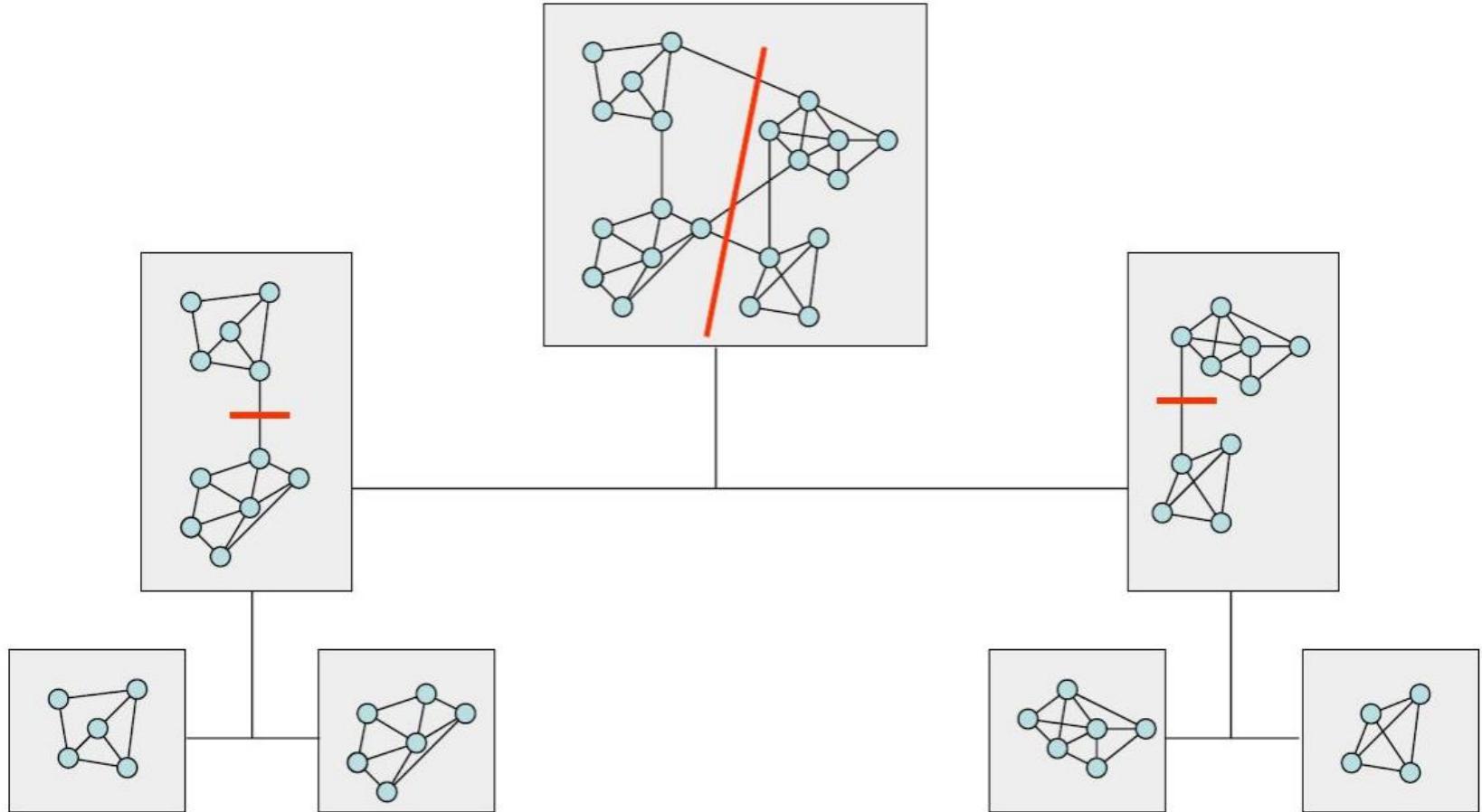
Le **2nd plus petit vecteur propre** est la valeur réelle  $y$  qui minimise la coupe normalisée.

Le **3ème** est la valeur réelle  $y$  qui découpe de manière optimale les 2 premières régions.

Etc.

On s'arrête avec un seuil sur les valeurs propres → **on choisit le nombre de segments.**

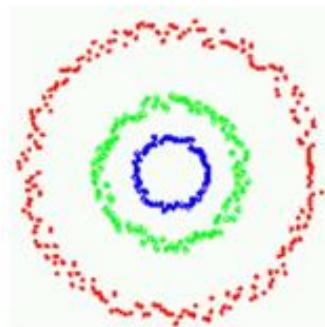
# Coupe minimale et normalisée



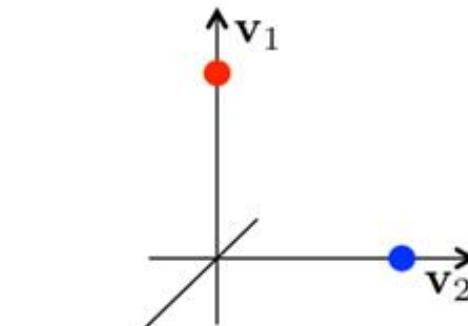
# Coupe minimale et normalisée

Ce que l'on voudrait :

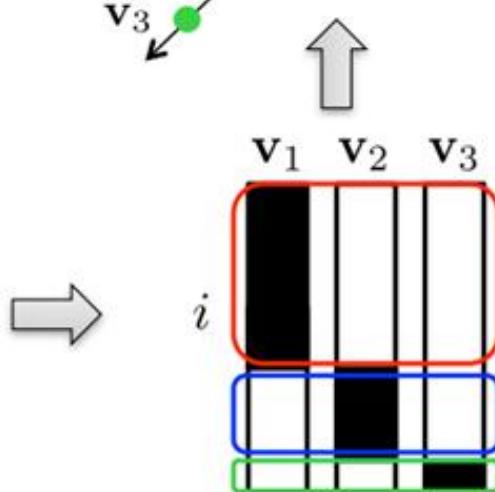
Disconnected subgraphs



$$L = \begin{bmatrix} & & & 0 \\ & & & \\ & & & \\ 0 & & & \\ & & & \\ & & & \end{bmatrix}$$



Points are easy to cluster in embedded space e.g. using k-means

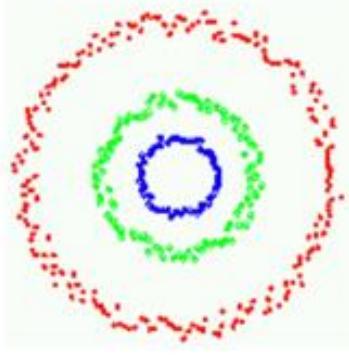


Embedding of point i  
[ $v_1(i), v_2(i), v_3(i)$ ]

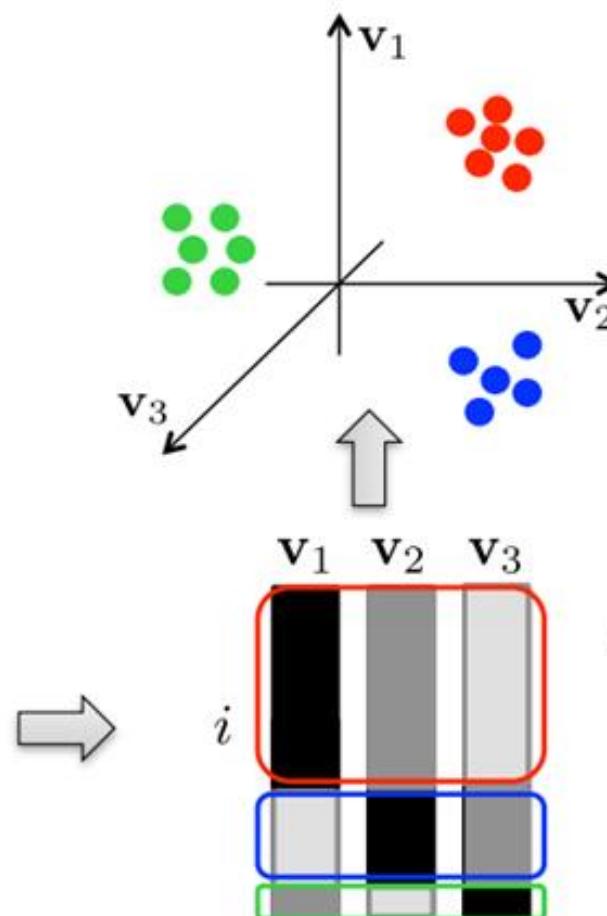
# Coupe minimale et normalisée

Ce que l'on a :

Disconnected subgraphs



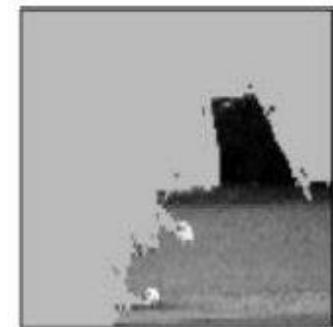
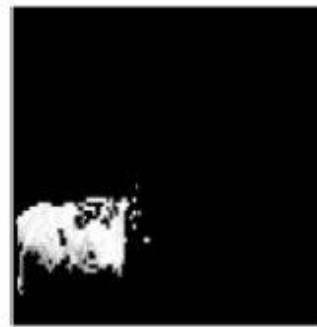
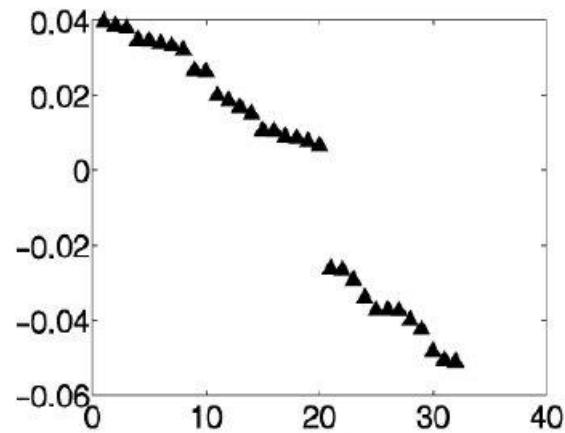
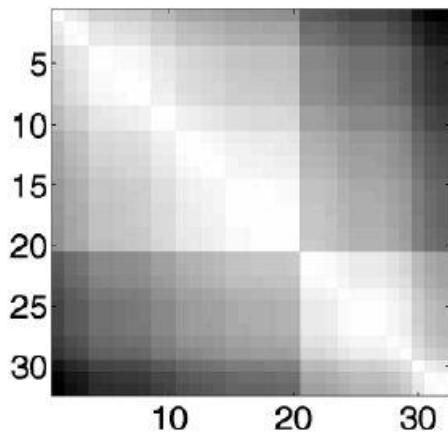
$$\mathbf{L} = \begin{matrix} & & & \varepsilon \\ & & \varepsilon & \\ & \varepsilon & & \\ & & & \varepsilon \end{matrix}$$



Points are easy to cluster in embedded space e.g. using k-means

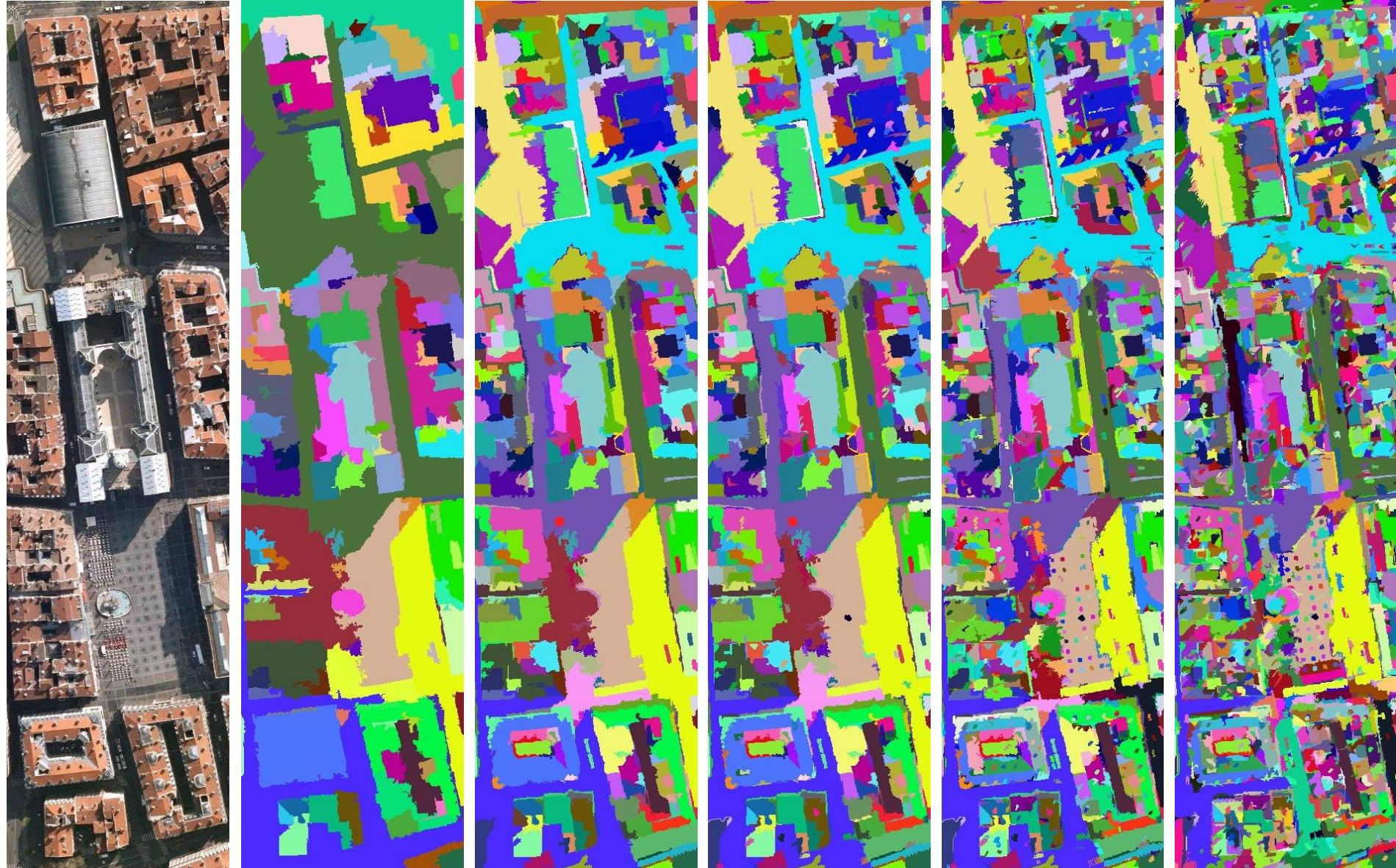
Embedding of point  $i$   
[ $v_1(i), v_2(i), v_3(i)$ ]

# Coupe minimale et normalisée



## **6. ÉVALUATION(S)**

# Laquelle est la meilleure ?



# Quelques méthodes

- Auto-qualification : sans a priori
  - Uniformité intra-région
  - Contraste inter-régions
- Qualification intrinsèque : connaissant la méthode de segmentation
  - Par exemple ?
- Qualification extrinsèque : mesure(s) de différence
  - Nécessité d'une vérité terrain
  - Comparaison régions ou contours
    - Nombre ou position des pixels mal classés
    - Nombre d'objets
    - Calculs d'attributs dans des régions d'intérêt.

# Qualification d'une segmentation

- Erreur locale (local refinement error LRE) :

- 2 segmentations  $S$  et  $S'$
- $C$  : ensemble de pixels regroupés avec  $x_i$
- Mesure non symétrique

$$\text{LRE}(S, S', x_i) = \frac{|C(S, x_i) \setminus C(S', x_i)|}{|C(S, x_i)|}$$

- Erreur globale (global consistency error GCE) :

- Mesure pour toute l'image ( $N$  pixels)

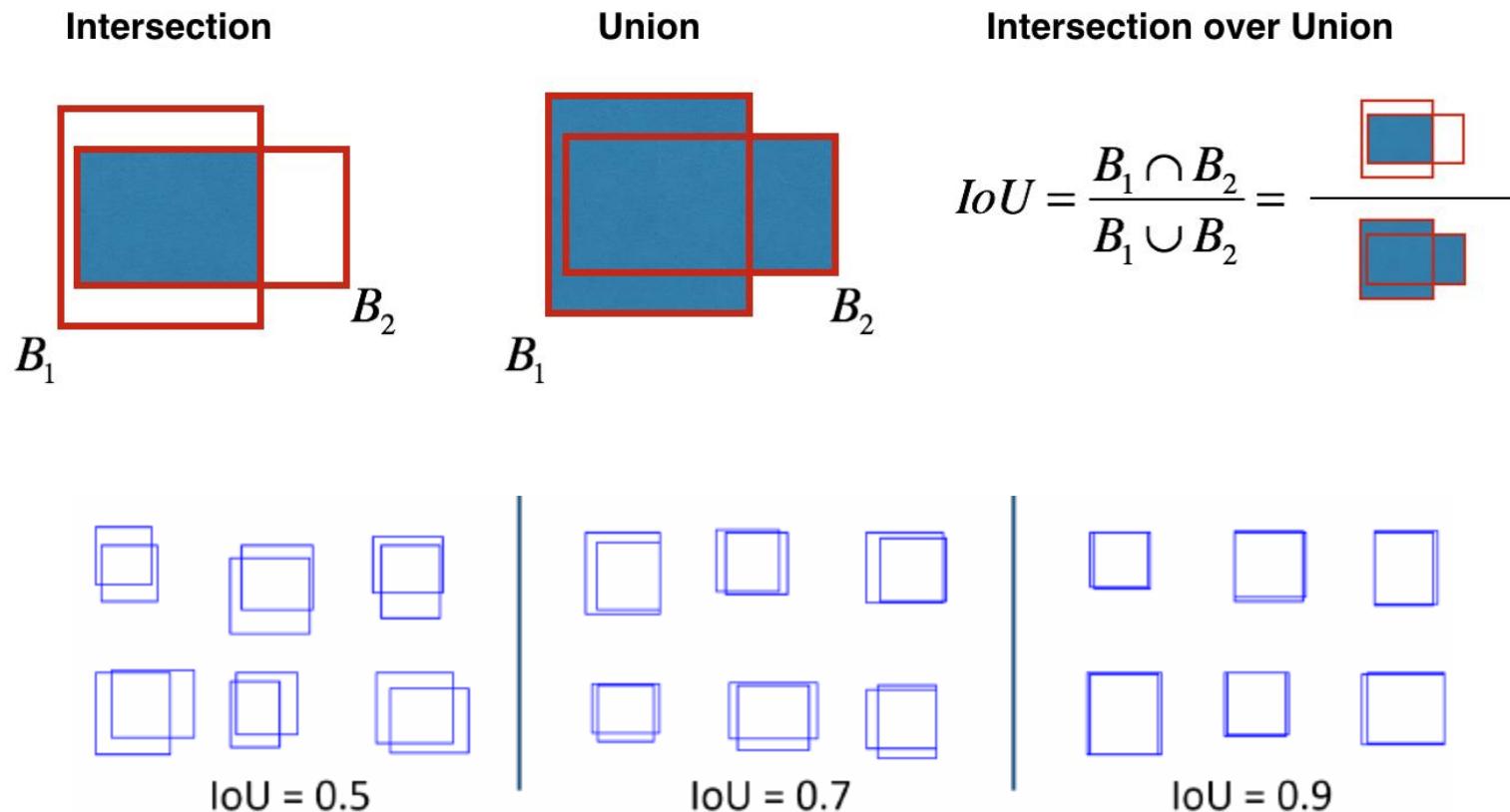
$$\begin{aligned} \text{GCE}(S, S') = \\ \frac{1}{N} \min \left\{ \sum_i \text{LRE}(S, S', x_i), \sum_i \text{LRE}(S', S, x_i) \right\} \end{aligned}$$

$\text{GCE}=0 \rightarrow$  pas d'erreur

$\text{GCE}=1 \rightarrow$  erreur maximale

# Qualification d'une segmentation

- Intersection-over-union : comparaison entre la région obtenue et celle de la vérité terrain.



## 7. BONUS

# Informations supplémentaires

- Les méthodes image de (sur)segmentation standard disponibles facilement :
  - **Watershed** : n'importe où
  - **Mean-Shift** : Orfeo Tool Box
  - **SLIC** : <http://ivrg.epfl.ch/research/superpixels>
  - **SEEDS** : <http://www.mvdblive.org/seeds/>
  - **Felzenszwalb & Huttenlocher** : <http://cs.brown.edu/~pff/segment/>
  - **QuickShift** : <http://www.vlfeat.org/install-shell.html>
  - **Turbopixels** : [http://www.cs.toronto.edu/~babalex/closure\\_code.tgz](http://www.cs.toronto.edu/~babalex/closure_code.tgz)
  - **Normalized Cuts** : <http://coewww.rutgers.edu/riul/research/code/EDISON/>
  - **Superpixels** : <http://www2.cs.sfu.ca/~mori/research/superpixels/>
- Des articles scientifiques qui font les synthèses : pour/contre, langage informatique etc.