

Chapitre 7 : Extraction de primitives Partie 3 – Formes et frontières

Joël Lefebvre (UQÀM)

INF600F – Traitement d'images

Automne 2024

Annonces

- Évaluation des enseignements
 - Date limite : 2 décembre
 - <https://portailetudiant.uqam.ca>
- **Laboratoire** : Révision des ateliers
- **TP4** : remise avant le dimanche 15 décembre avant 23h59
- **Examen final**
 - Mercredi, 11 décembre entre 14h à 17h
 - 1 feuille de note manuscrite de format Lettre (8 ½ x 11)
 - **Plus de détails à venir**
- Cours de l'hiver
 - **INF5071** : Infographie
 - **INF889G** : Vision par ordinateur

Sommaire

- Partie 1 : Détection de coins
- Partie 2 : Détection de points caractéristiques (SIFT)
- Partie 3 : Détection de formes
 - Détection des régions et contours
 - Représentation des régions et contours
 - Caractérisation des régions
 - Caractérisation de la texture

Références

- Burger 2009, Vol2, Chapitre 2 : *Regions in Binary Images*
- Gonzalez2018, Chapitre 12 : *Feature Extraction*
- Russ2016, Chapitre 11 : *Characterizing Shape*
- Ranjbar, S. and Mitchell, J. R. (2017). *Biomedical Texture Analysis*, 223–245. <https://doi.org/10.1016/b978-0-12-812133-7.00008-9>

Détection des régions et contours

Extraction de primitives, Partie 3 – Formes et frontières

INF600F – Traitement d'images

Joël Lefebvre (UQÀM)

Régions et images binaires

- Image binaire : 2 valeurs d'intensité
- Représentent l'**avant-plan** (*foreground : FG*) et l'**arrière-plan** (*background : BG*)
- Scène naturelle : Plusieurs objets présents, pas seulement FG / BG
- **Objectif** : extraction et description des régions isolées

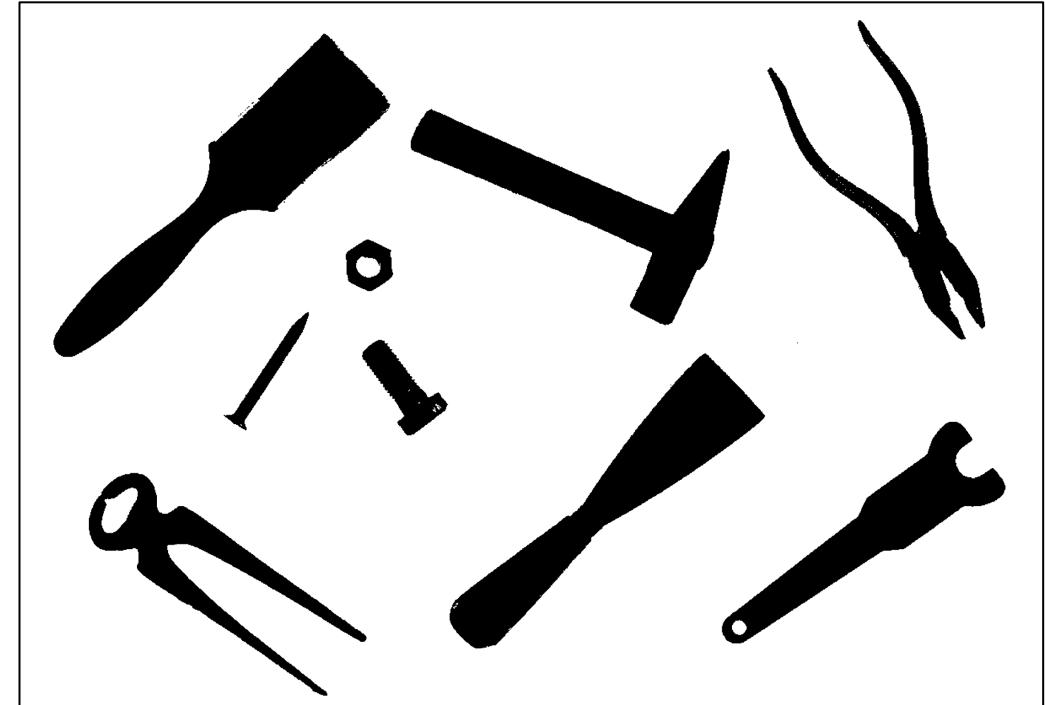
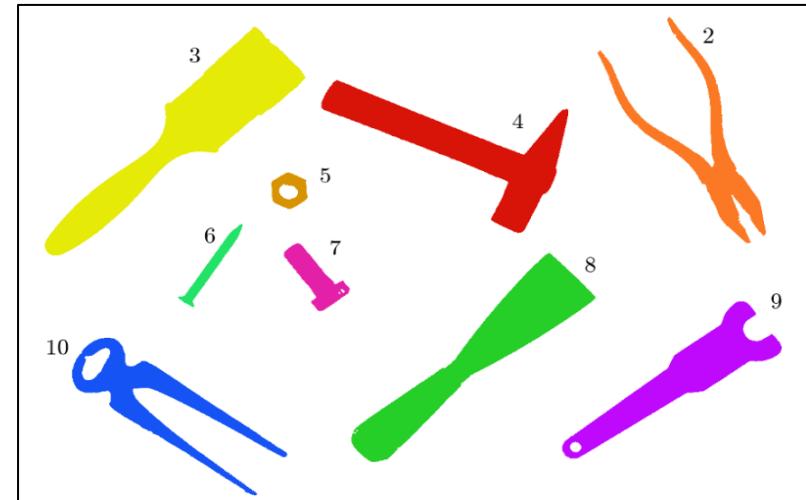


Image binaire à analyser en contraste inversé
(BG=Blanc, FG=Noir, Source : Burger Vol2,
Fig2.1)

Trouver les régions de l'image

- **Étiquetage des régions (*region labeling*)**
 - Déterminer les pixels pour chaque région
 - Déterminer le nombre de régions
 - Déterminer où sont situées les régions
- **3 méthodes**
 - Remplissage par inondation
 - Marquage séquentiel des régions
 - Méthode combinant l'étiquetage des régions / marquage des contours



Source : Burger, Vol2, Fig2.6

Étiquetage (*Labeling*)

- Assigner un nombre unique (étiquette ou *label*) pour identifier une région et ses pixels
- La région est formée de tous les pixels voisins
- On doit donc choisir un type de connectivité : N_4 ou N_8
- Convention pour les algorithmes d'étiquetage de région

$$I(u, v) = \begin{cases} 0 & \text{un pixel d'arrière-plan} \\ 1 & \text{un pixel d'avant-plan} \\ 2, 3, \dots & \text{une étiquette de région} \end{cases}$$

Étiquetage par inondation (*flood filling*)

- Rechercher des pixels d'avant-plan non étiquetés et remplir (*fill*) tous les pixels voisins de sa région.
- 3 façons de trouver le prochain pixel
 - Méthode récursive
 - Méthode itérative en profondeur (*depth-first*)
 - Méthode itérative en largeur (*breadth-first*)

```
1: REGIONLABELING( $I$ )
    $I$ : binary image;  $I(u, v) = 0$ : background,  $I(u, v) = 1$ : foreground
   The image  $I$  is labeled (destructively modified) and returned.

2:   Let  $m \leftarrow 2$                        $\triangleright$  value of the next label to be assigned
3:   for all image coordinates  $(u, v)$  do
4:     if  $I(u, v) = 1$  then
5:       FLOODFILL( $I, u, v, m$ )           $\triangleright$  use any version from Alg. 2.2
6:        $m \leftarrow m + 1$ .
7:   return the labeled image  $I$ .
```

Algorithme d'étiquetage de région
par inondation (*flood filling*)

Méthode 1 : Inondation récursive

```
1: FLOODFILL( $I, u, v, label$ )
```

▷ Recursive Version

```
2:   if ( $u, v$ ) is inside the image and  $I(u, v) = 1$  then
```

```
3:     Set  $I(u, v) \leftarrow label$ 
```

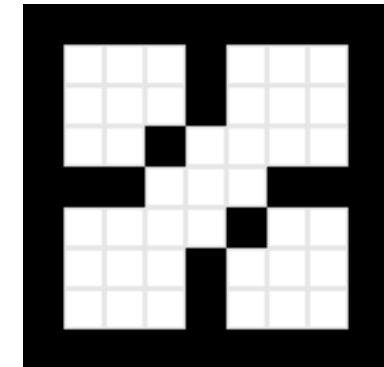
```
4:     FLOODFILL( $I, u+1, v, label$ )
```

```
5:     FLOODFILL( $I, u, v+1, label$ )
```

```
6:     FLOODFILL( $I, u, v-1, label$ )
```

```
7:     FLOODFILL( $I, u-1, v, label$ )
```

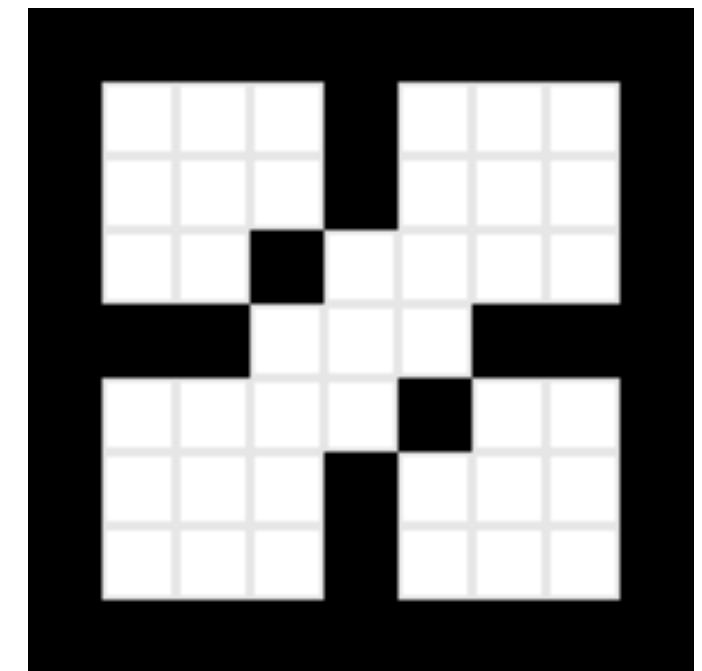
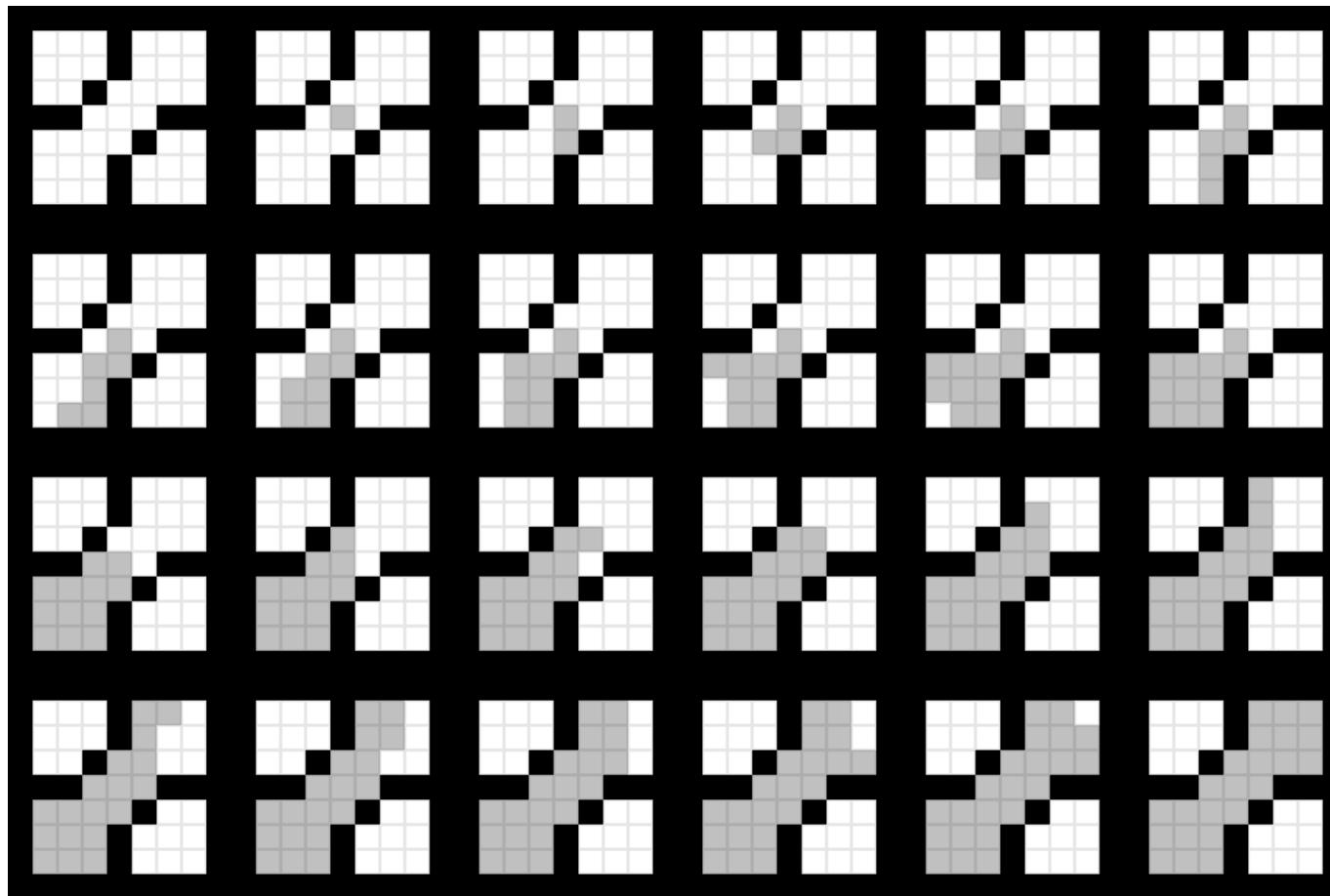
```
8:   return.
```



Source : [Wikimedia](#)

- Chaque région est représentée par un arbre
- La profondeur maximale de récursion est proportionnelle à la taille de la région
- La pile de mémoire (stack memory) est rapidement épuisée
- Pratique pour les très petites images seulement.

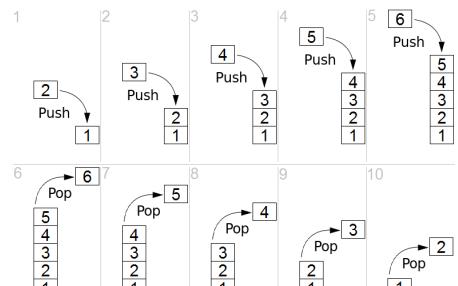
Exemple : Inondation récursive



Source : [Wikimedia](#)

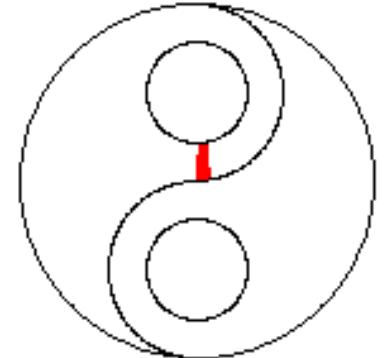
Méthode 2 : Inondation itérative par profondeur (*depth-first*)

- Reformulation de l'algorithme récursif en algorithme itératif
- Gestion d'une pile (*stack*) pour conserver les éléments « ouverts » (pixels adjacents mais pas encore visités)
- Dispose de plus de mémoire (*heap memory*)



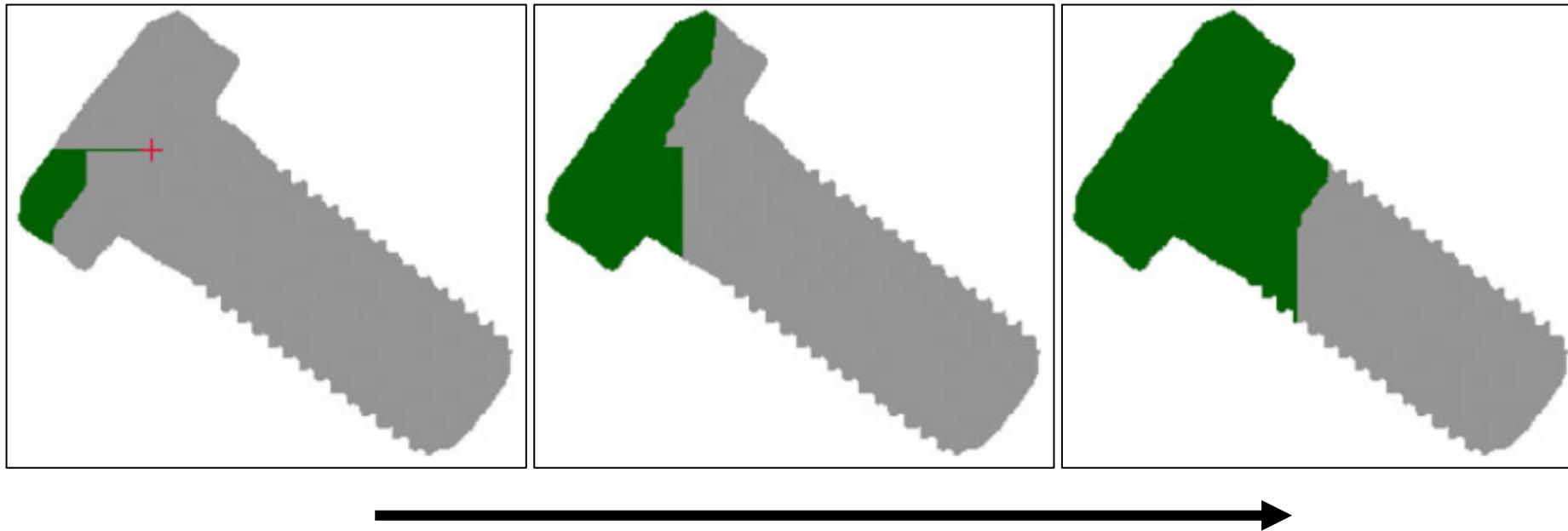
Pile ([Wikimedia](#))

```
9: FLOODFILL( $I, u, v, \text{label}$ ) ▷ Depth-First Version
10: Create an empty stack  $S$ 
11: Put the seed coordinate  $(u, v)$  onto the stack: PUSH( $S, (u, v)$ )
12: while  $S$  is not empty do
13:   Get the next coordinate from the top of the stack:
         $(x, y) \leftarrow \text{POP}(S)$ 
14:   if  $(x, y)$  is inside the image and  $I(x, y) = 1$  then
15:     Set  $I(x, y) \leftarrow \text{label}$ 
16:     PUSH( $S, (x+1, y)$ )
17:     PUSH( $S, (x, y+1)$ )
18:     PUSH( $S, (x, y-1)$ )
19:     PUSH( $S, (x-1, y)$ )
20:   return.
```



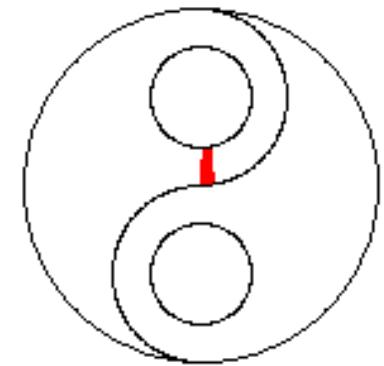
Source : [Wikimedia](#)

Exemple : Inondation itérative par profondeur (*depth-first*)



Source : Burger, Vol2, Fig2.2

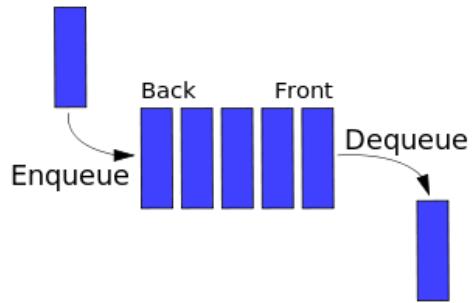
Taille maximale de la pile : 28 822 éléments



Source :
[Wikimedia](#)

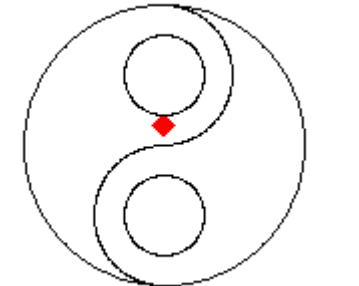
Méthode 3 : Inondation itérative par largeur (*breadth-first*)

- Les pixels sont traversés comme une vague s'éloignant du point initial
- La structure de données utilisée pour contenir les pixels non visités est une file (*queue*) plutôt qu'une pile (*stack*)



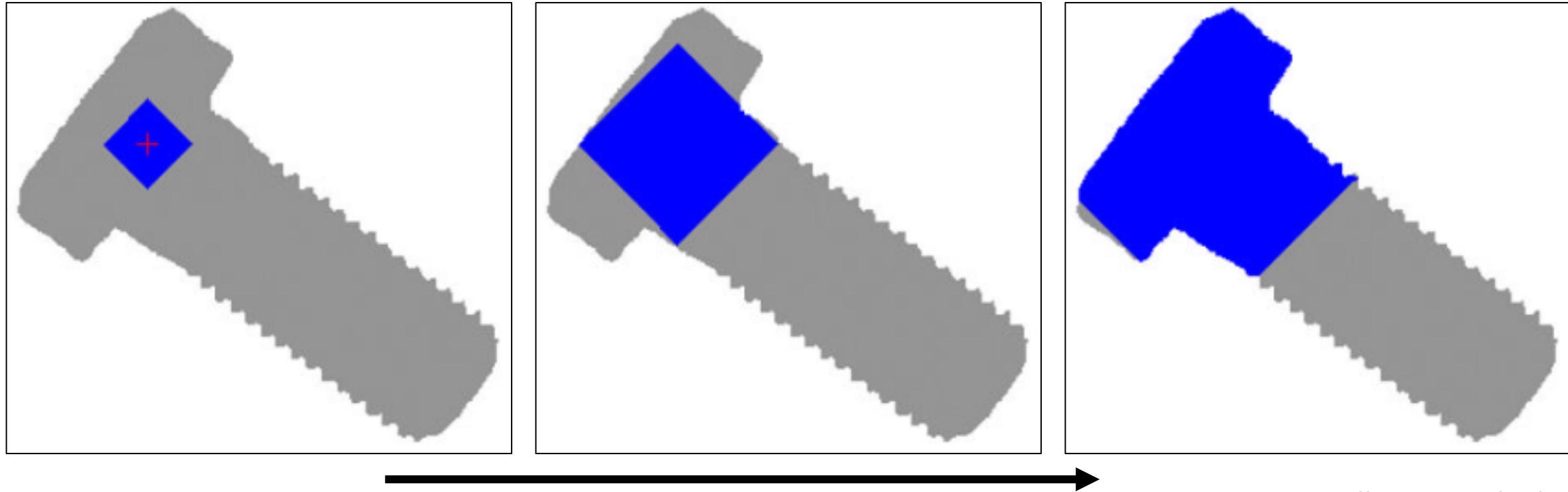
File ([Wikimedia](#))

```
21: FLOODFILL( $I, u, v, \text{label}$ ) ▷ Breadth-First Version
22: Create an empty queue  $Q$ 
23: Insert the seed coordinate  $(u, v)$  into the queue: ENQUEUE( $Q, (u, v)$ )
24: while  $Q$  is not empty do
25:   Get the next coordinate from the front of the queue:
         $(x, y) \leftarrow \text{DEQUEUE}(Q)$ 
26:   if  $(x, y)$  is inside the image and  $I(x, y) = 1$  then
27:     Set  $I(x, y) \leftarrow \text{label}$ 
28:     ENQUEUE( $Q, (x+1, y)$ )
29:     ENQUEUE( $Q, (x, y+1)$ )
30:     ENQUEUE( $Q, (x, y-1)$ )
31:     ENQUEUE( $Q, (x-1, y)$ )
32:   return.
```



Source : [Wikimedia](#)

Exemple : Inondation itérative par largeur (*breadth-first*)



Source : Burger, Vol2, Fig2.2

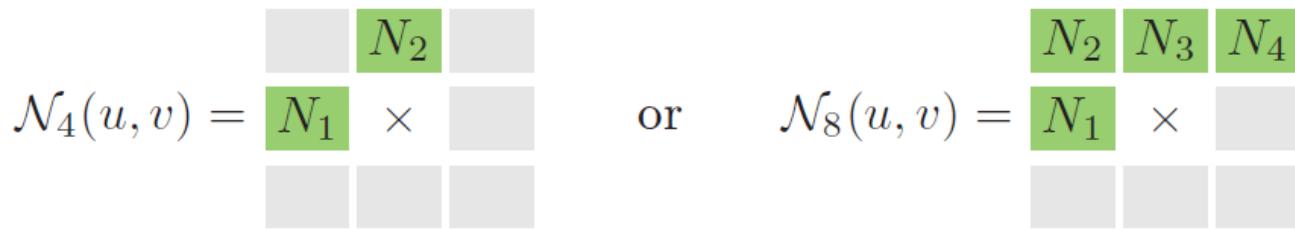
Taille maximale de
la file : 438 nœuds

Étiquetage séquentiel des régions

- Technique classique non récursive connue sous le nom de marquage de région (*region labeling*)
- Consiste en 3 étapes
 1. Étiquetage préliminaire des régions de l'image
 2. Correction des cas où plusieurs étiquettes sont assignées
 3. Ré-étiquetage
- Algorithme plus complexe que les précédents, mais utilisation modérée de la mémoire

Étape 1 : Étiquetage initial

- L'image est traversée séquentiellement de l'origine (en haut à gauche) jusqu'en bas à droite.
- Considérant le sens du parcours dans l'image, on peut utiliser deux définitions de voisinage
- Les collisions où ≥ 2 étiquettes sont présentes dans le voisinage sont enregistrées.



0	Background
1	Foreground

0 0
0 0 0 0 0 1 1 0 0 1 1 0 1 0 1 0
0 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 0
0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0
0 1 1 0 0 0 1 0 1 0 0 0 0 0 0 0
0 0

(b) only background neighbors new label (2)

0 0
0 0 0 0 0 0 1 1 0 0 1 1 0 1 0 1 0
0 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 0
0 0 0 0 1 0 1 0 0 0 0 0 0 1 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0
0 1 1 0 0 0 1 0 1 0 0 0 0 0 0 0
0 0

(c) exactly one neighbor label neighbor label is propagated

0 0
0 0 0 0 0 2 1 0 0 1 1 0 1 0
0 1 1 1 1 1 1 0 0 1 0 0 1 0
0 0 0 0 1 0 1 0 0 0 0 0 1 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 0
0 1 1 0 0 0 1 0 1 0 0 0 0 0 0
0 0

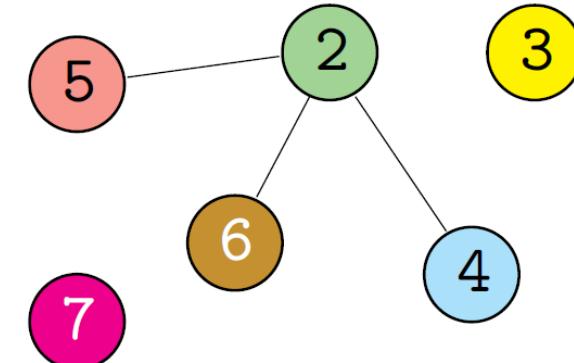
(d) two different neighbor labels one of the labels (2) is propagated

0 0
0 0 0 0 0 2 2 0 0 3 3 0 4 0
0 5 5 5 1 1 1 0 0 1 0 0 1 0
0 0 0 0 1 0 1 0 0 0 0 0 0 1 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 0
0 1 1 0 0 0 1 0 1 0 0 0 0 0 0
0 0

Étape 2 : Collision des étiquettes

- Détection des composantes connectées
- Représentation en graphe
 - Nœud : Étiquette d'une région
 - Lien : Collisions

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	2	2	0	0	3	3	0	4	0				
0	5	5	5	5	2	2	2	0	0	3	0	0	4	0				
0	0	0	0	0	2	0	2	0	0	0	0	0	4	0				
0	6	6	2	2	2	2	2	2	2	2	2	2	2	0				
0	0	0	0	0	2	2	2	2	2	2	2	2	2	0				
0	7	7	0	0	0	2	0	2	0	0	0	0	0	0				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				



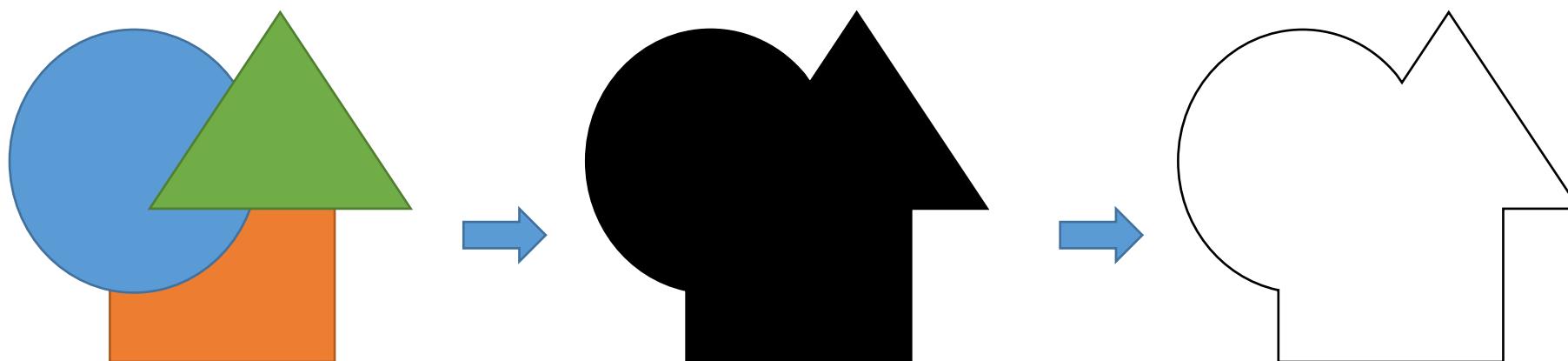
Étape 3 : Ré-étiquetage

- Les étiquettes de chaque composante connectée sont réassignées
 - **Exemple** : Choisir l'étiquette la plus petite dans la région



Contours des régions

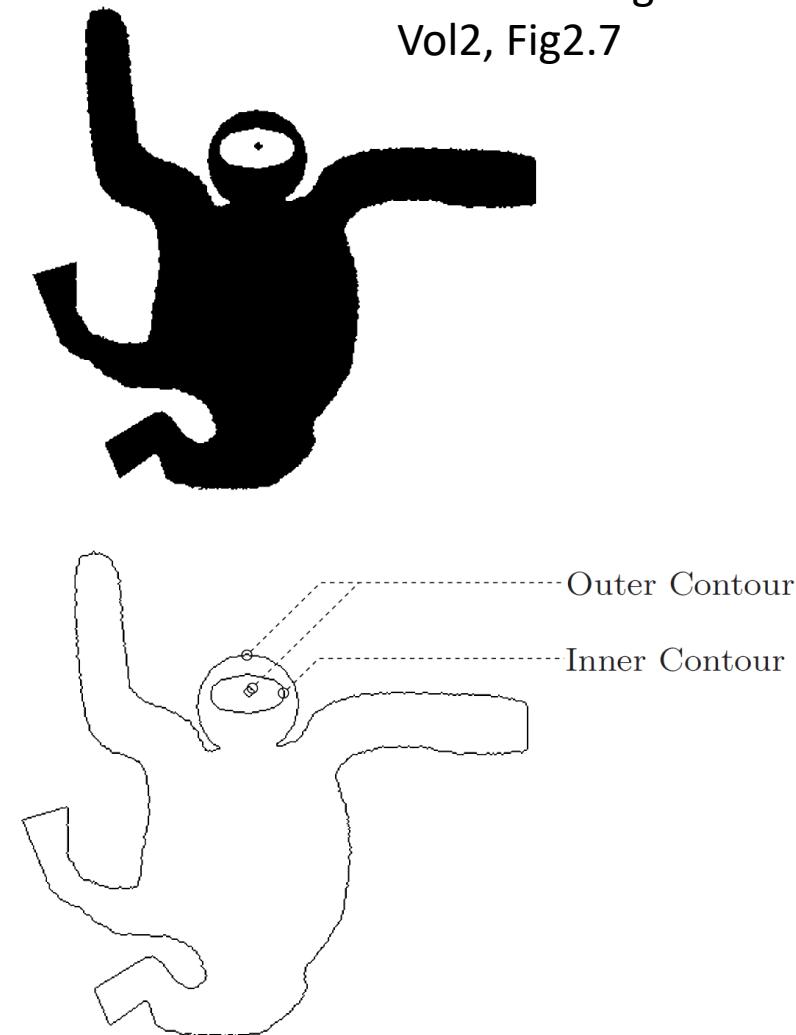
- Après l'étiquetage des régions, on peut détecter leurs contours
- Approche intuitive : suivre les frontières de la région
- Approche algorithmique plus complexe
- Problème classique en analyse d'images



Contours internes / externes

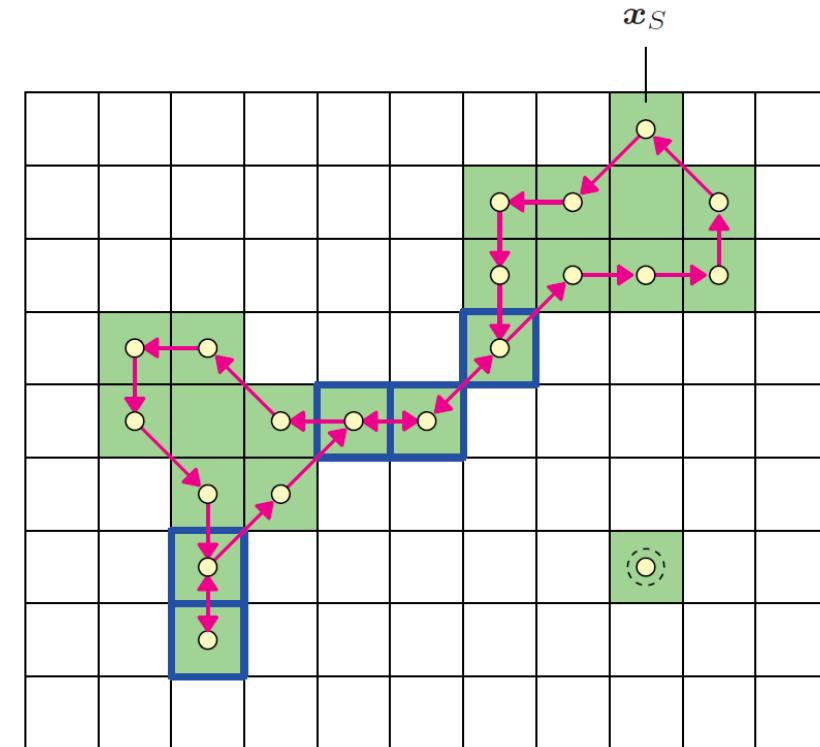
- Les pixels sur la frontière d'un objet binaire peuvent être identifiés en utilisant des **opérations morphologiques** et des différences d'images.
- Ce procédé « **marque** » les pixels de contour
- Étape supplémentaire : obtenir une **séquence ordonnée** des coordonnées des pixels de frontière
- Chaque région possède **un seul contour externe (outer)**, mais peut posséder **plusieurs contours internes (inner)**

Source : Burger
Vol2, Fig2.7



Détection des contours

- Méthode en 2 étapes
 - Identification des **régions connectées**
 - Pour chaque région, choisir un **pixel initial x_s** et **se déplacer sur la frontière** puis ajouter des pixels à la description du contour
- Pour les contours internes, on se déplace sur la frontière du trou
- Conceptuellement simple, mais **complexe dans l'implémentation** (ex : considérer les endroits où la région a une épaisseur de 1 pixel)



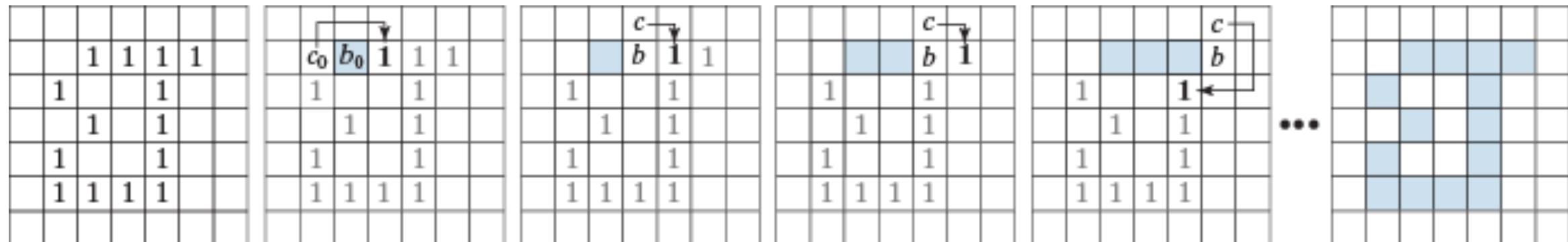
Source : Burger, Vol2, Fig2.8

Suivi de frontière (*Tracing*)

Algorithme de suivi de contour de Moore

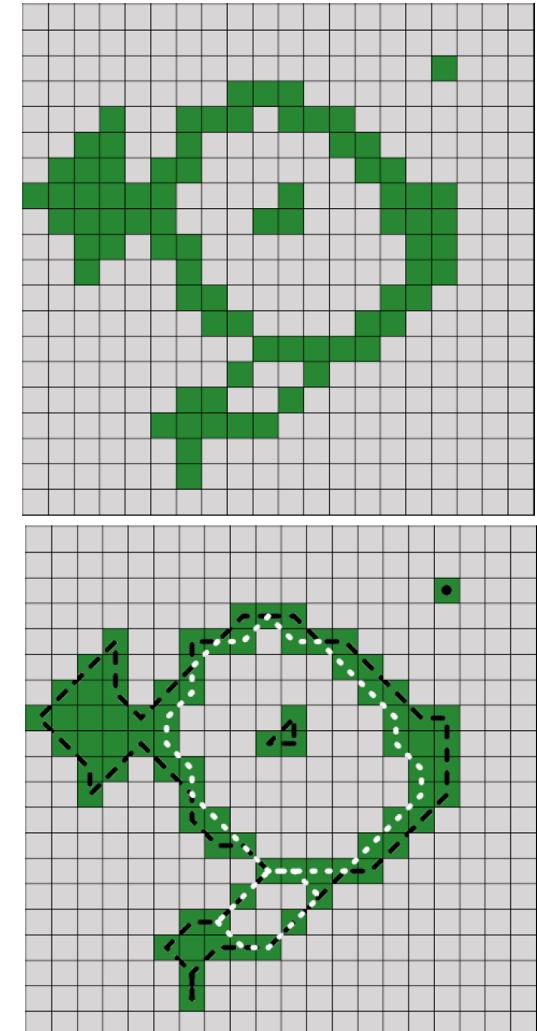
1. Choisir un **point de départ** $b = b_0$ (ex. le plus en haut à gauche) et un **point** $c = c_0$ à l'ouest de ce point.
2. Se déplacer dans le **sens horaire** sur le **voisinage** N_8 de b en partant de c . Le premier point d'avant-plan trouvé devient le nouveau b et le point d'arrière-plan le précédent devient le nouveau c .
3. Ajout de b à une liste des pixels de contour
4. Répéter jusqu'à ce que $b = b_0$

Gonzalez2018, Fig12.1



Détection combinée des régions et des contours

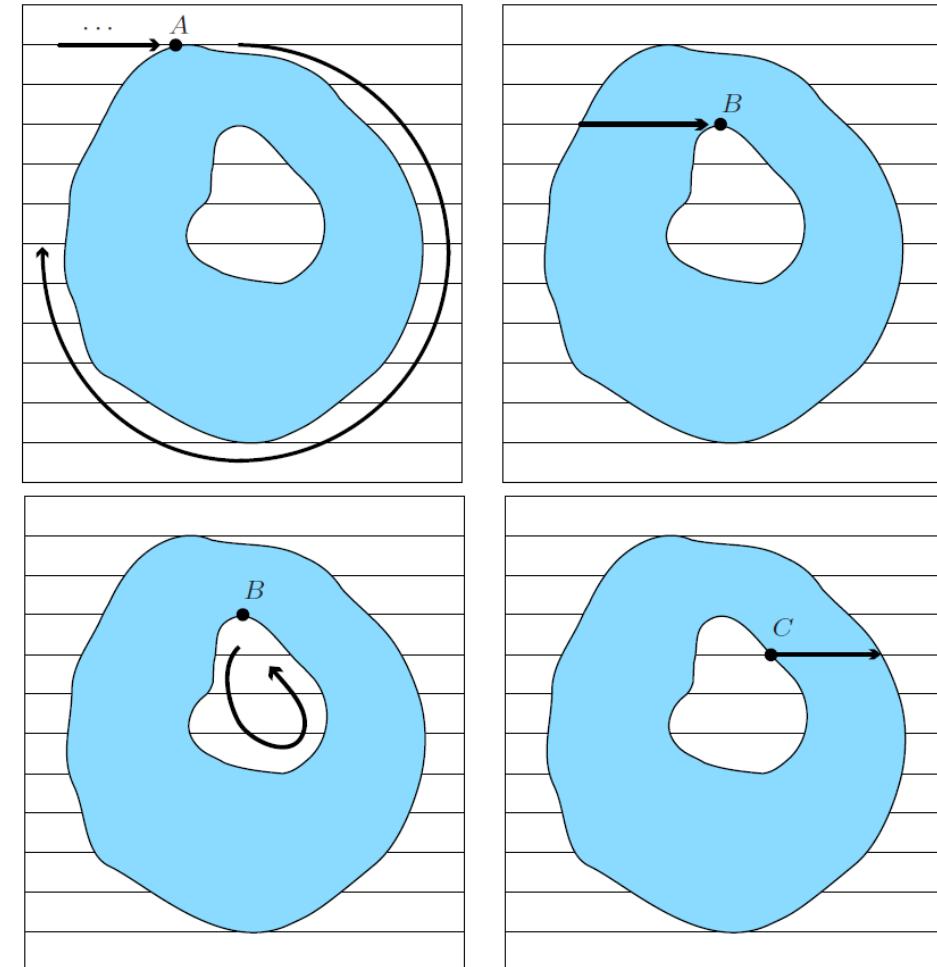
- **Combine** les concepts de **l'étiquetage** séquentiel des régions et le **tracage** des contours traditionnels en un seul algorithme
- Nécessite **un seul parcours** de l'image
- Identifie et étiquette les régions, et **trace les contours internes / externes.**
- Ne nécessite pas de structures de données complexes, et cette approche est très efficace



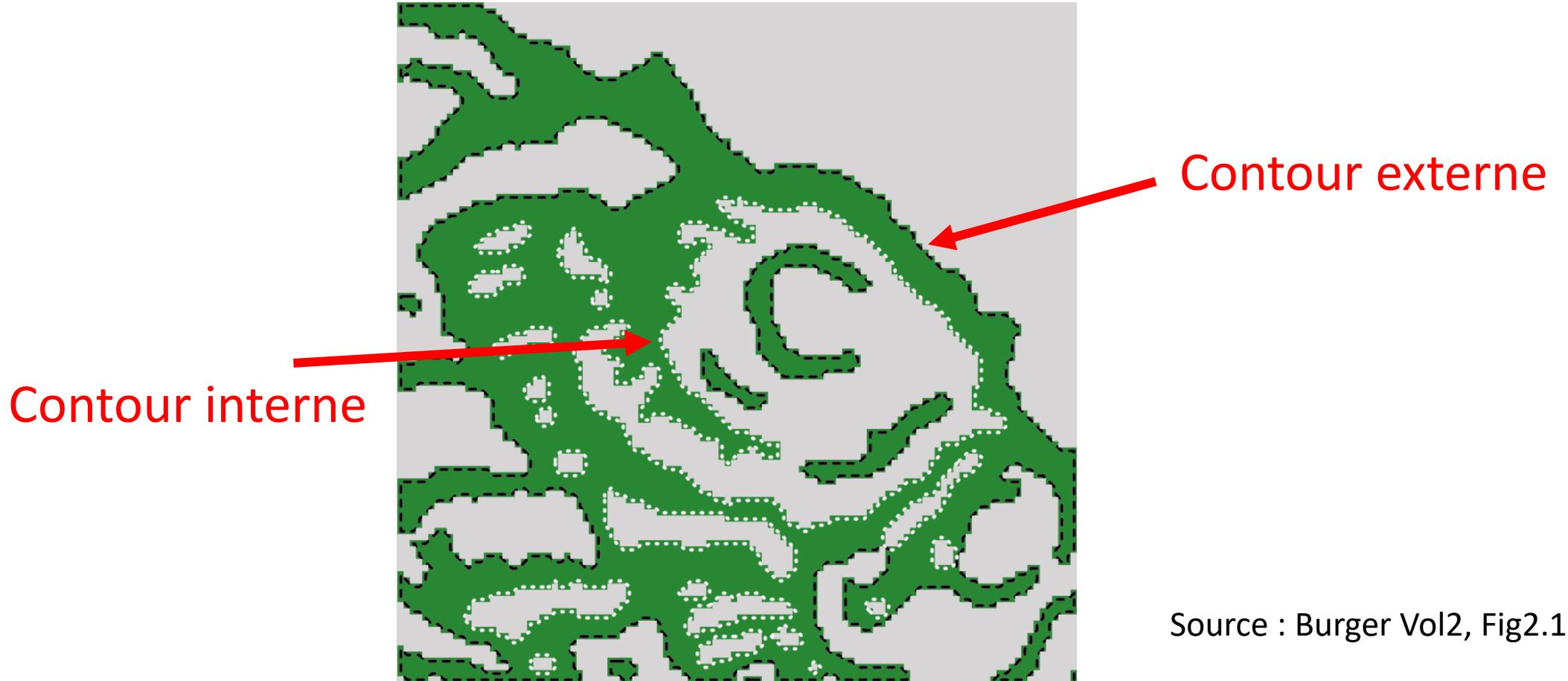
Source : Burger, Vol2, Fig2.10

Algorithme : Détection combinée des régions et des contours

- L'image binaire est **traversée** du coin supérieur gauche au coin inférieur droit
- Pour un point donné dans l'image
 - **Cas 1** : Transition d'un pixel d'arrière-plan vers un pixel d'avant-plan non visité.
 - **Cas 2** : Transition d'un pixel d'avant-plan vers un pixel d'arrière-plan non visité.
 - **Cas 3** : Le pixel d'avant-plan n'est pas sur un contour, donc l'étiquette du voisin de gauche est propagée au pixel courant



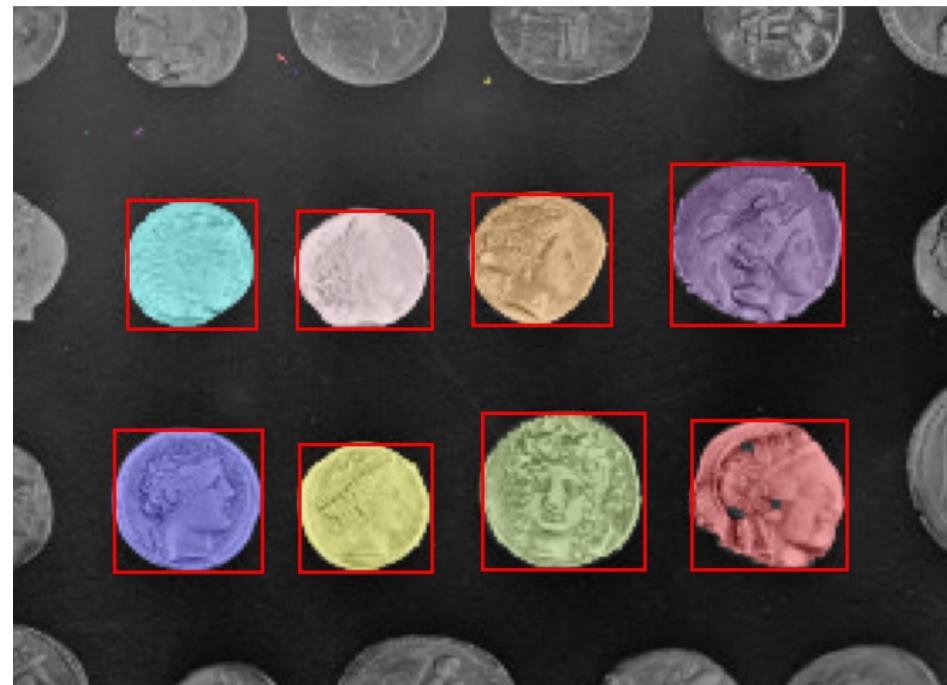
Exemple : Détection combinée des régions et des contours



Source : Burger Vol2, Fig2.11

Python et étiquetage de région

- **skimage.measure.label(label_image)**
- Utilise une approche hybride combinant recherche itérative et description du voisinage (Algorithme de Suzuki)
- Documentation ([URL](#))
- Démo Python ([URL](#))



Source : scikit-image
[URL](#)

Représentation des régions et contours

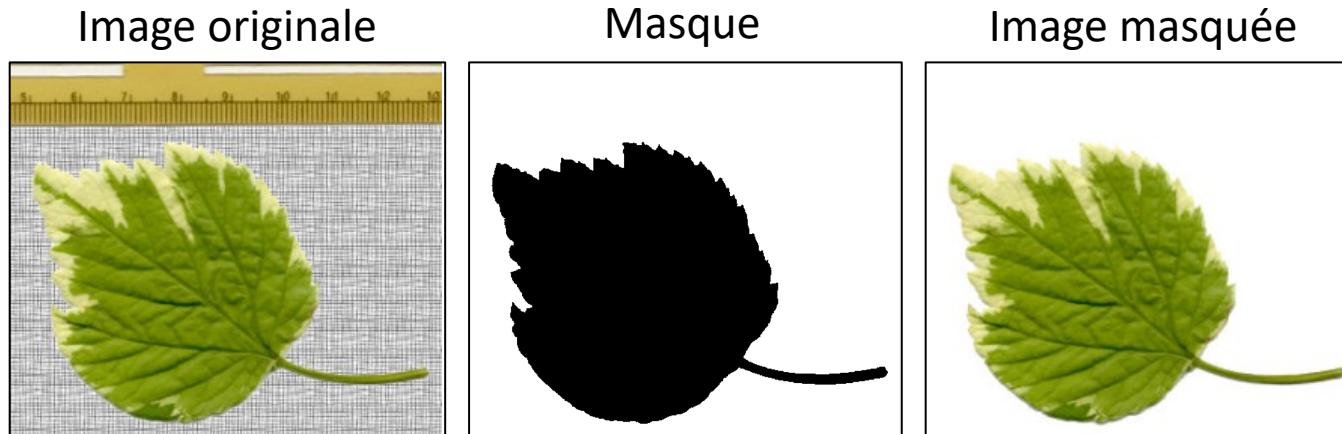
Extraction de primitives, Partie 3 – Formes et frontières

INF600F – Traitement d'images

Joël Lefebvre (UQÀM)

Représentation des régions de l'image

- **Représentation matricielle** d'une image
- Les régions d'une image peuvent être représentées par un masque logique
 - Les pixels associés à la **région** ont la **valeur True**
 - Les **autres pixels** ont la **valeur False**
- Ce type de représentation est parfois appelé « **bitmap** »
- **D'autres représentations sont possibles**



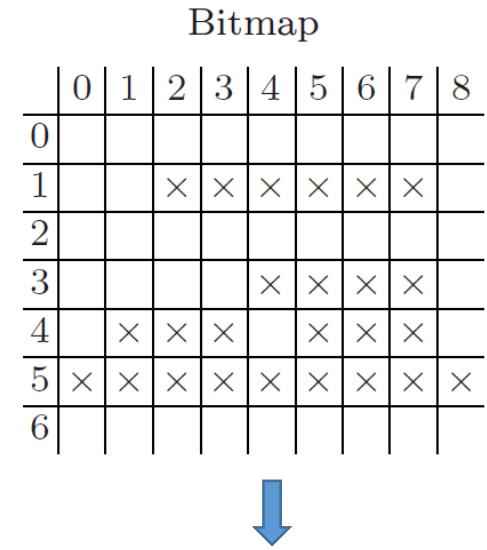
Source : Burger
Vol2, Fig2.12

Codage par plage (*Run length encoding*)

- **Représentation compacte** d'une région binaire.
- Une **plage (run)** est la longueur maximale d'une séquence contiguë de pixels ayant le même type.

$$Run_i = \langle \text{Ligne}_i, \text{Colonne}_i, \text{Longueur}_i \rangle$$

- Facile à implémenter et efficace.
- Méthode de **compression** de données **sans perte**
- Utilisé par les fax et par les formats d'images TIFF, GIF et JPEG



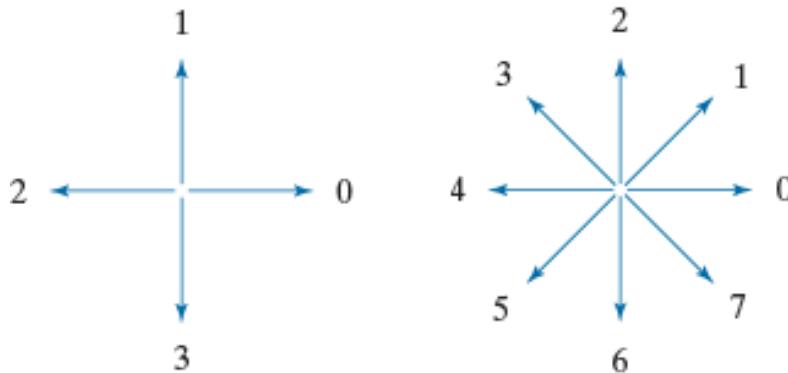
RLE

$\langle \text{row}, \text{column}, \text{length} \rangle$

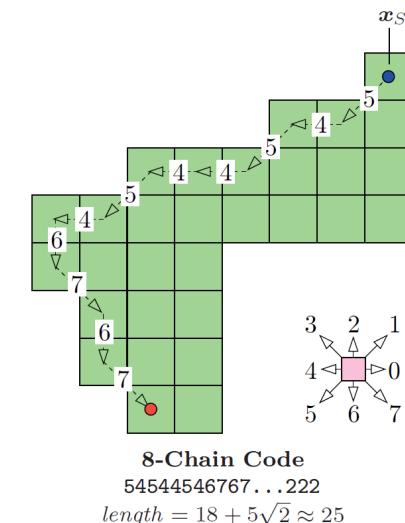
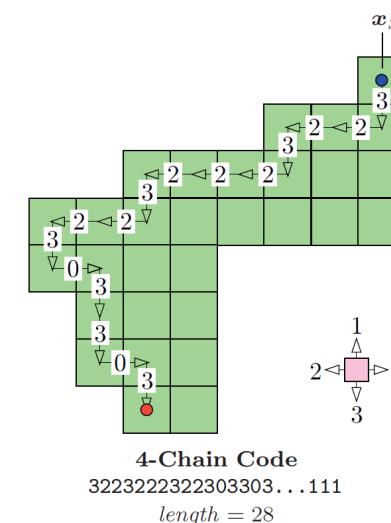
$\langle 1, 2, 6 \rangle$
 $\langle 3, 4, 4 \rangle$
 $\langle 4, 1, 3 \rangle$
 $\langle 4, 5, 3 \rangle$
 $\langle 5, 0, 9 \rangle$

Code à enchaînement (*chain code*)

- Les régions peuvent être représentées par leurs contours
- Le **code à enchaînement (*chain code* ou *Freeman codes*)** est une méthode classique pour encoder des contours
- Le contour commence en un point x_s et est représenté par une séquence des changements directionnels qu'il décrit sur l'image



Code de direction (Gonzalez, Fig12.3)



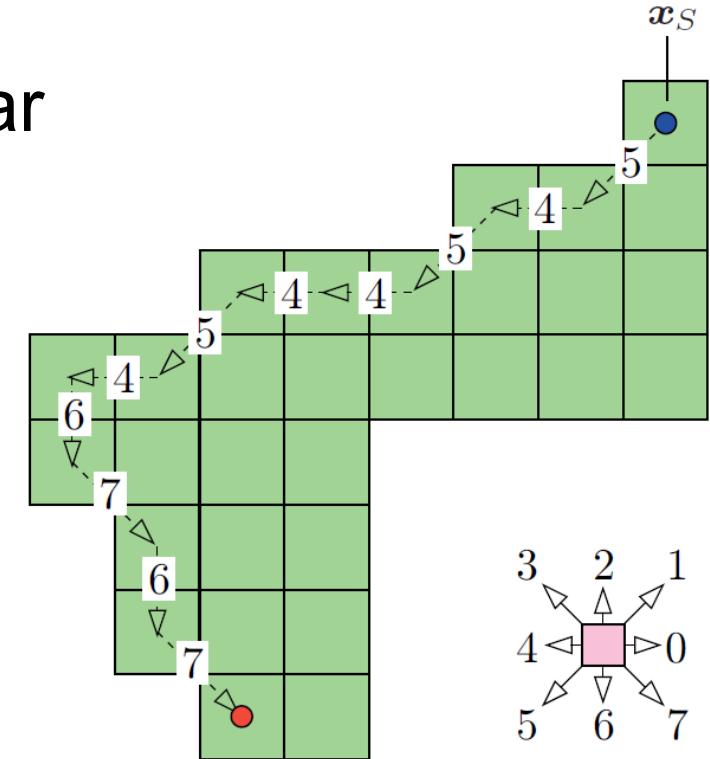
Source : Burger Vol2,
Fig2.14

Code à enchaînement absolu (1)

- Pour un contour fermé d'une région \mathcal{R} décrit par une séquence de points $c_{\mathcal{R}} = [x_0, x_1, \dots, x_{M-1}]$ avec $x_i = \langle u_i, v_i \rangle$
- Les éléments du code d'enchaînement $c'_{\mathcal{R}} = [c'_0, c'_1, \dots, c'_{M-1}]$ sont donnés par

$$c'_i = \text{Code}(\Delta u_i, \Delta v_i)$$

$$(\Delta u_i, \Delta v_i) = \begin{cases} (u_{i+1} - u_i, v_{i+1} - v_i) & \text{pour } 0 \leq i < M - 1 \\ (u_0 - u_i, v_0 - v_i) & \text{pour } i = M - 1 \end{cases}$$



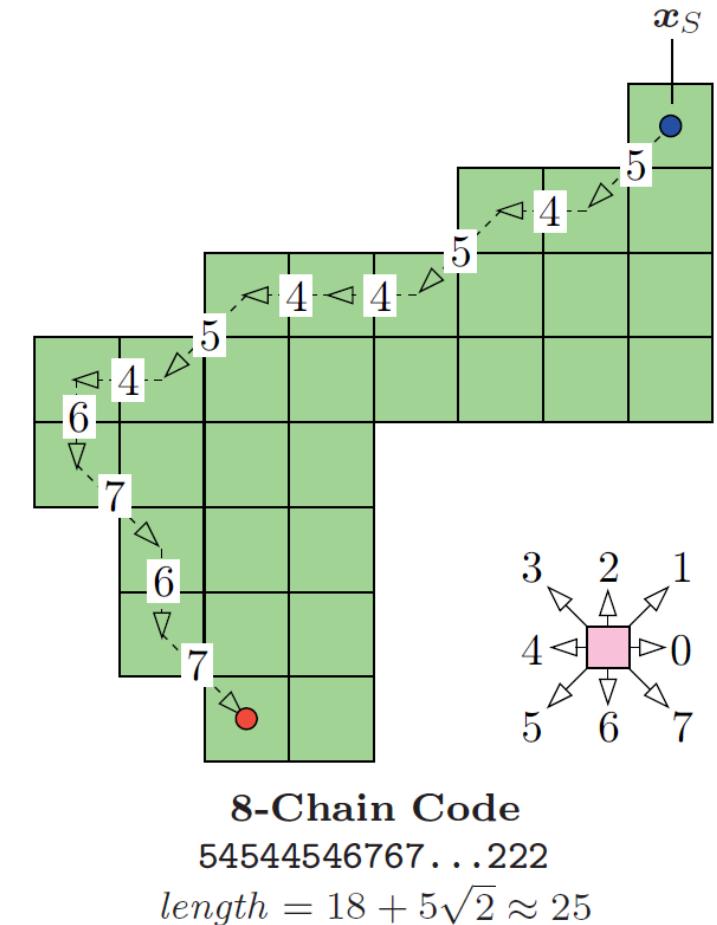
Code à enchaînement absolu (2)

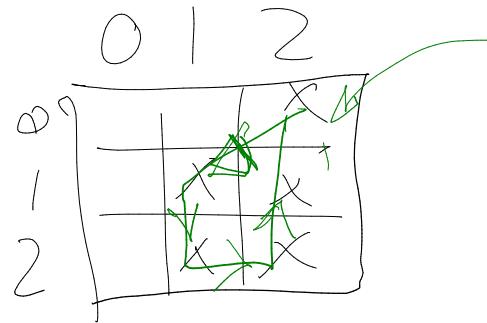
- En supposant un voisinage de type N_8

SQ donne Ca va être fourni

Δu	1	1	0	-1	-1	-1	0	1
Δv	0	1	1	1	0	-1	-1	-1
$\text{Code}(\Delta u, \Delta v)$	0	1	2	3	4	5	6	7

- Les codes à enchaînement sont compacts, car seule la coordonnée du premier point x_s doit être conservée
- Seuls 3 bits sont nécessaires pour représenter un code.





① CR = $\left[(0,2), (1,1), (2,1), (2,2), (1,2) \right]$

Abs ② $C'_c = \left[(\Delta U, \Delta V), (-1, -1), (1, 0), (0, 1), (-1, 0), (-1, 1) \right]$

③ ABS = $\left[7, 0, 2, 4, 4 \right]$

$$C^{(1)} = \left[\begin{matrix} -7\%8, 2\%8, 4\%8, \\ 2 \end{matrix} \right]$$

] changement 7 avec 0
et modulo 8

Code à enchaînement différentiel

- La comparaison de deux régions est **difficile** avec les codes à enchaînement, car leur représentation **dépend du point initial x_s**
- **Exemple** : si la région subit une rotation de 90°, le code est différent
- **Amélioration** : le code à enchaînement **différentiel** utilise les changements de **direction** plutôt que les changements de **position**
- Conversion d'un **code absolu c'_R** en **code différentiel c''_R**

$$c''_i = \begin{cases} (c'_{i+1} - c'_i) \bmod 8 & \text{pour } 0 \leq i < M - 1 \\ (c'_0 - c'_i) \bmod 8 & \text{pour } i = M - 1 \end{cases}$$

Exemple : Code à enchaînement différentiel

- Code à enchaînement absolu

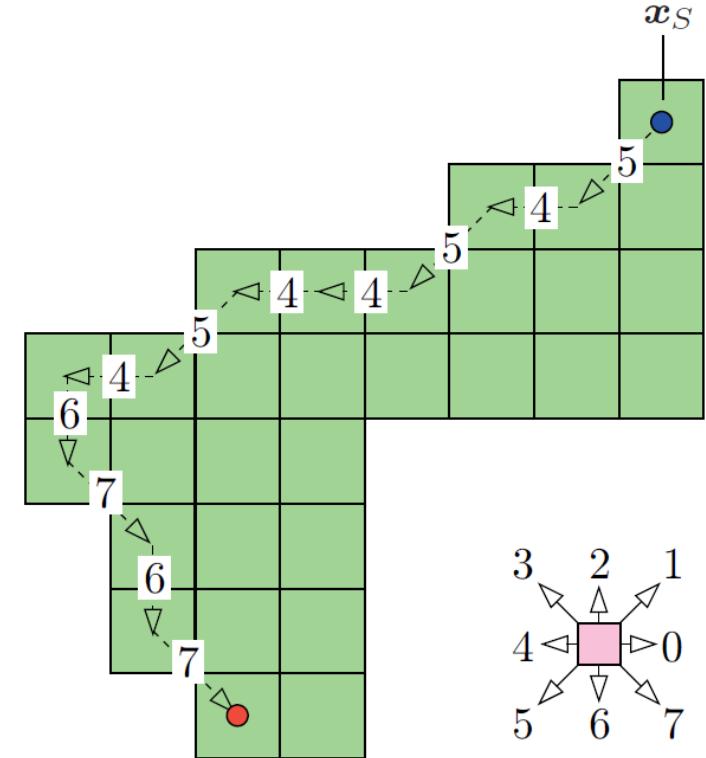
$$c'_R = [5, 4, 5, 4, 4, 5, 4, 6, 7, 6, 7, \dots, 2, 2, 2]$$

$$\begin{array}{|c|} \hline (5-4)\%8 \\ \hline (4-5)\%8 \\ \hline \end{array}$$

- Code à enchaînement différentiel

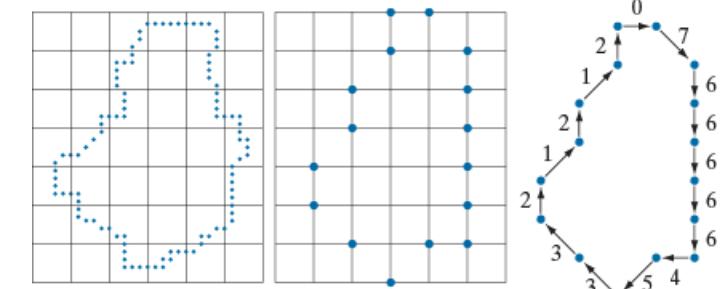
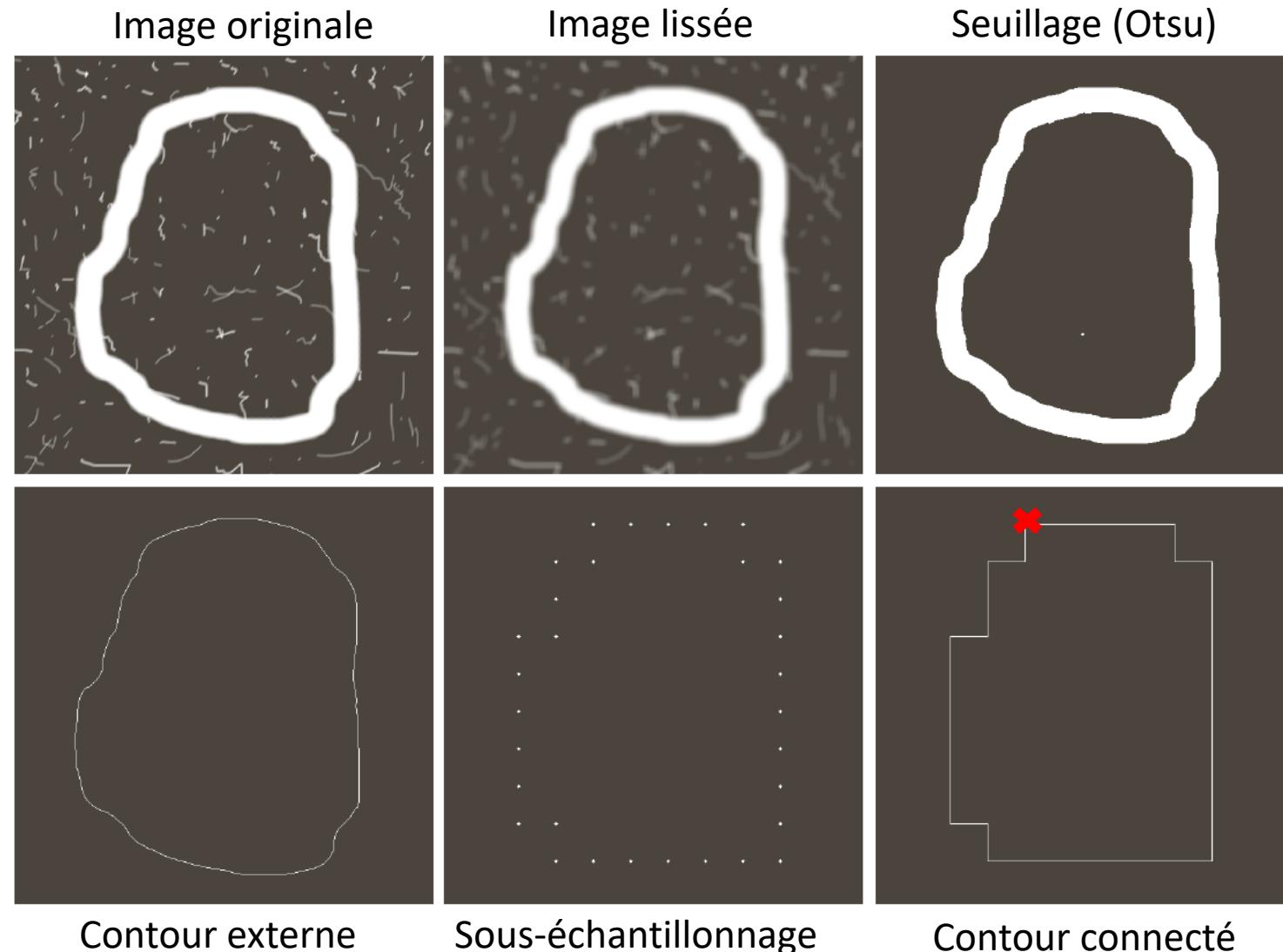
$$c''_R = [7, 1, 7, 0, 1, 7, 2, 7, 1, 1, \dots, 0, 0, 3]$$

- Pour reconstruire le contour, on doit connaître la position initiale x_s et la direction initiale c_0



8-Chain Code
54544546767...222
 $length = 18 + 5\sqrt{2} \approx 25$

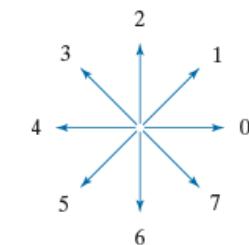
Exemple : Code de Freeman



Sous-échantillonnage du contour pour (1) réduire la longueur du code, et (2) réduire l'effet des petites variations

Point initial : (2,5)
Code absolu :
0 0 0 0 6 0 6 6 6
6 6 6 6 4 4 4 4 4 4 2 4 2 2 2 2
0 2 2 0 2

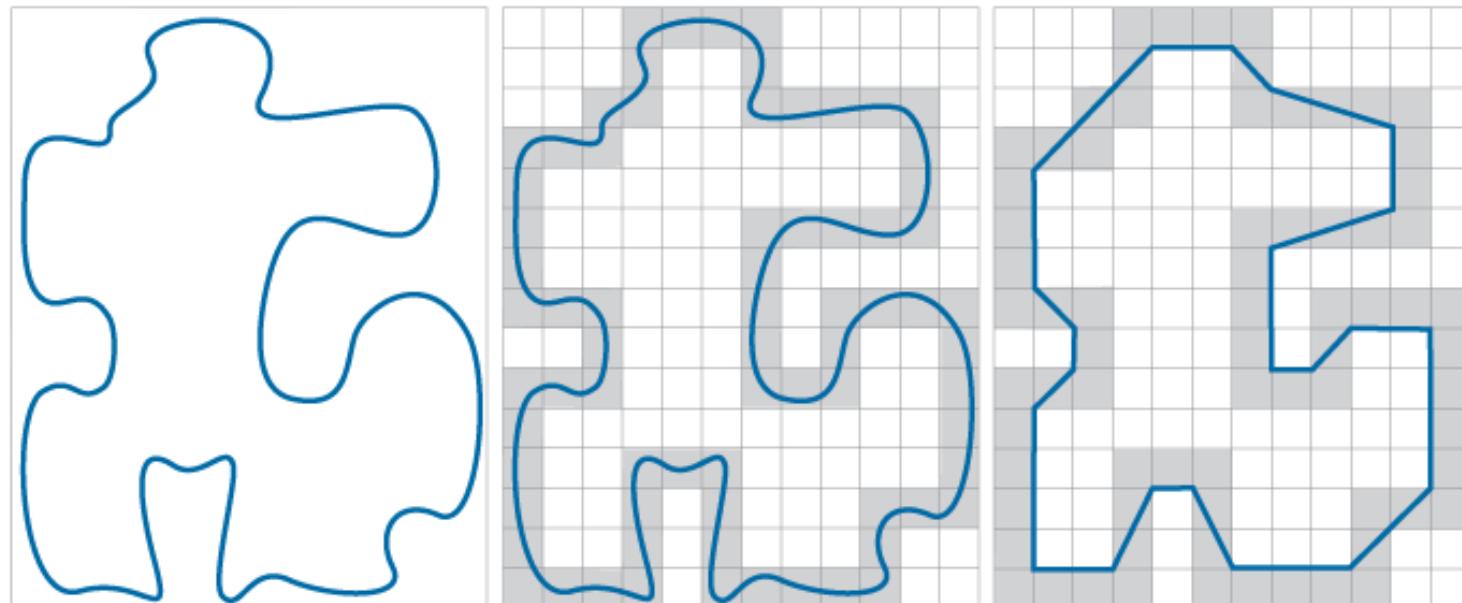
Gonzalez,
Fig12.5



Approximation des contours

Polygone de périmètre minimal

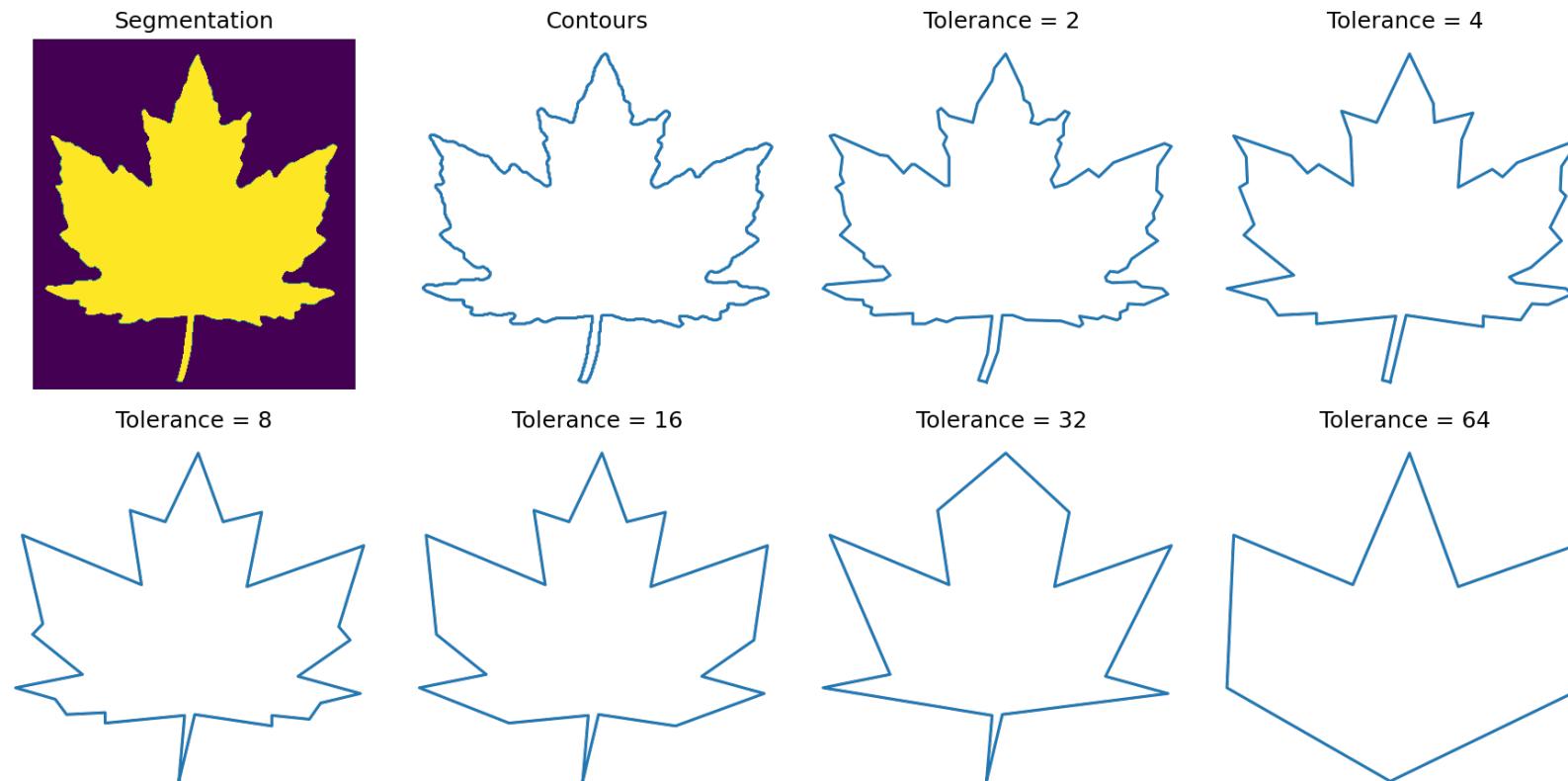
- *Minimum-perimeter polygon (MPP)*
- Un contour fermé peut être approximé par un polygone.
- But : Représenter l'essentiel de la forme avec le moins de segments possible.



Gonzalez, Fig2.7

Exemple : Approximation d'un contour par un polygone

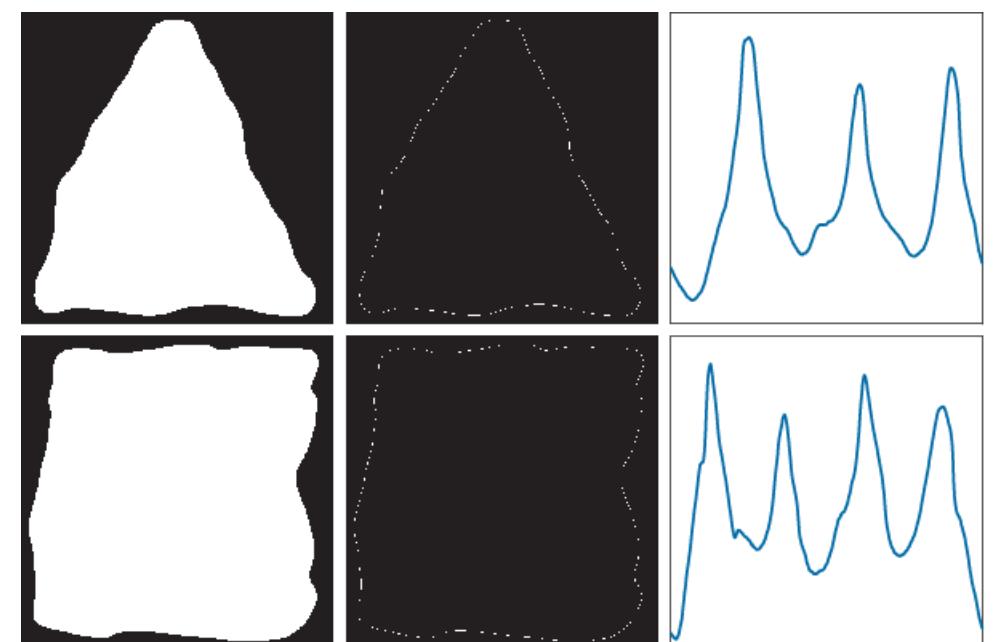
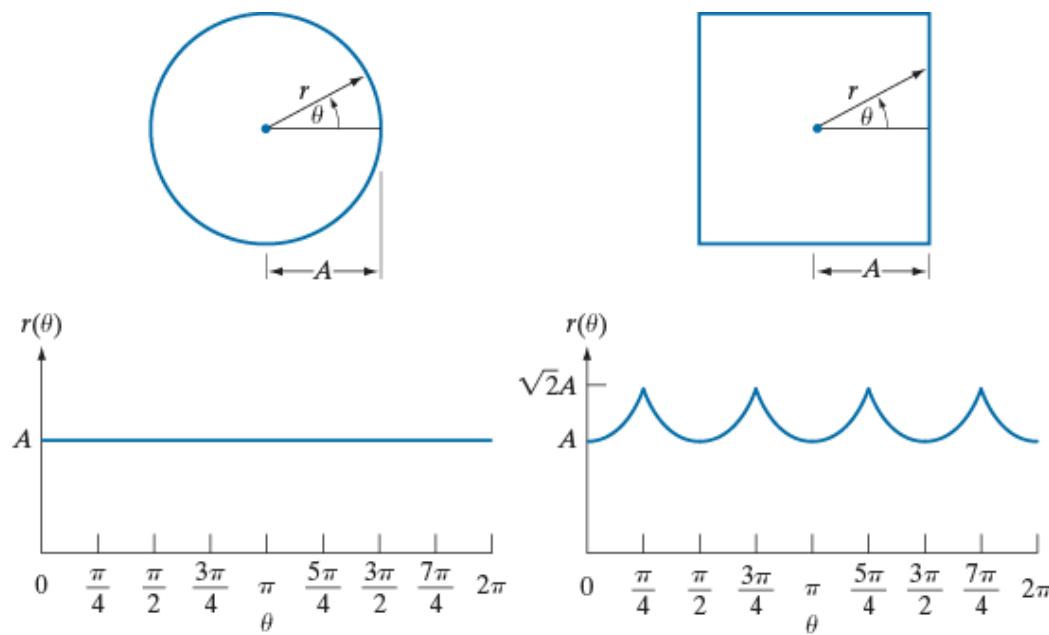
- **skimage.measure.approximate_polygon**
- Démo Python : [URL](#)



Signatures

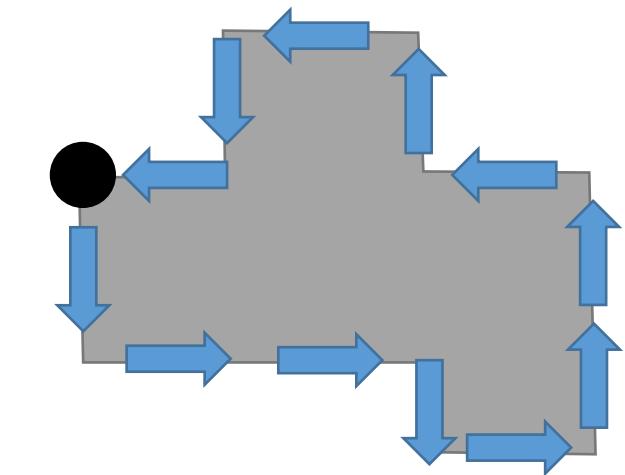
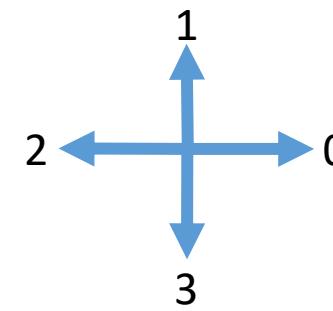
- Représentation fonctionnelle 1D d'un contour 2D
- Plusieurs façons de générer une signature
- Exemple : Distance du contour à son centroïde

Gonzalez,
Fig12.10-11



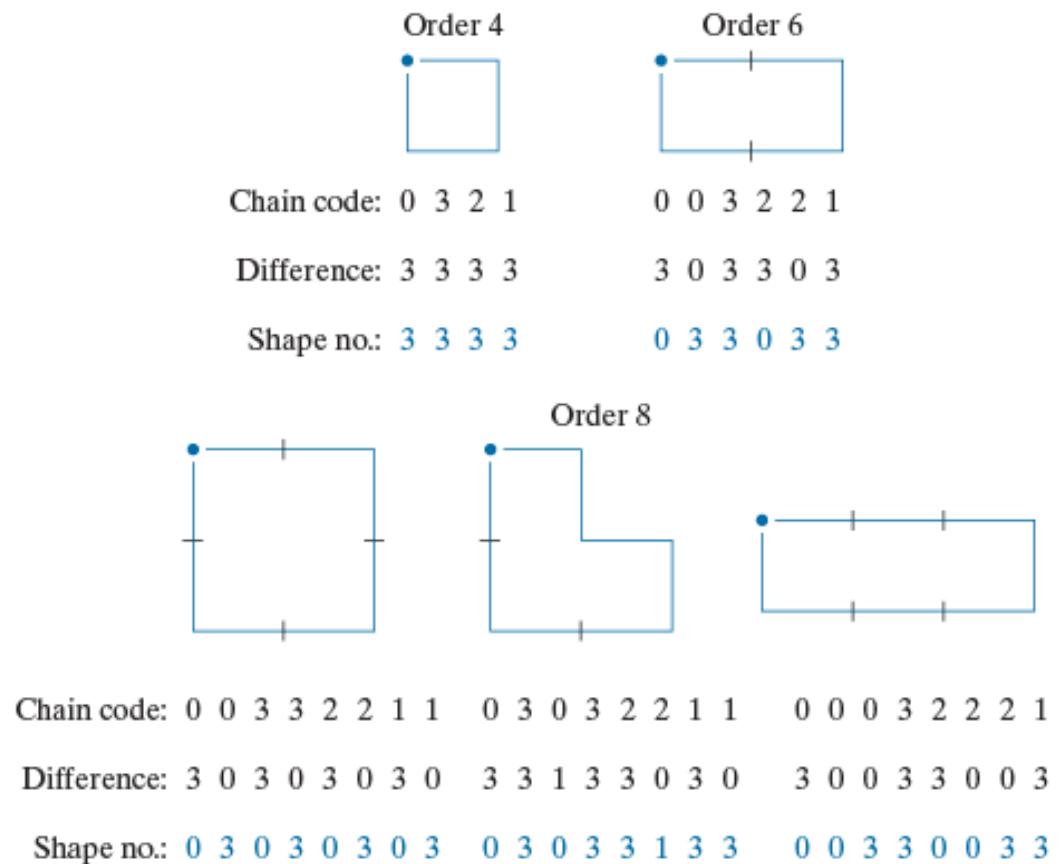
Nombre de forme (*Shape numbers*)

- Le code à enchaînement différentiel dépend du point initial x_s
- Pour comparer deux codes c_1'' et c_2'' , on doit d'abord s'assurer que le même point initial x_s a été utilisé
- On obtient le **nombre de forme** (*shape number*) par rotation cyclique du code différentiel et en choisissant le code ayant le nombre le plus petit (ordre lexicographique)
- **Exemple**
 - **Code absolu** : 3003011211232
 - **Code différentiel** : 10311013113
 - **Nombre de forme** : 013113130311



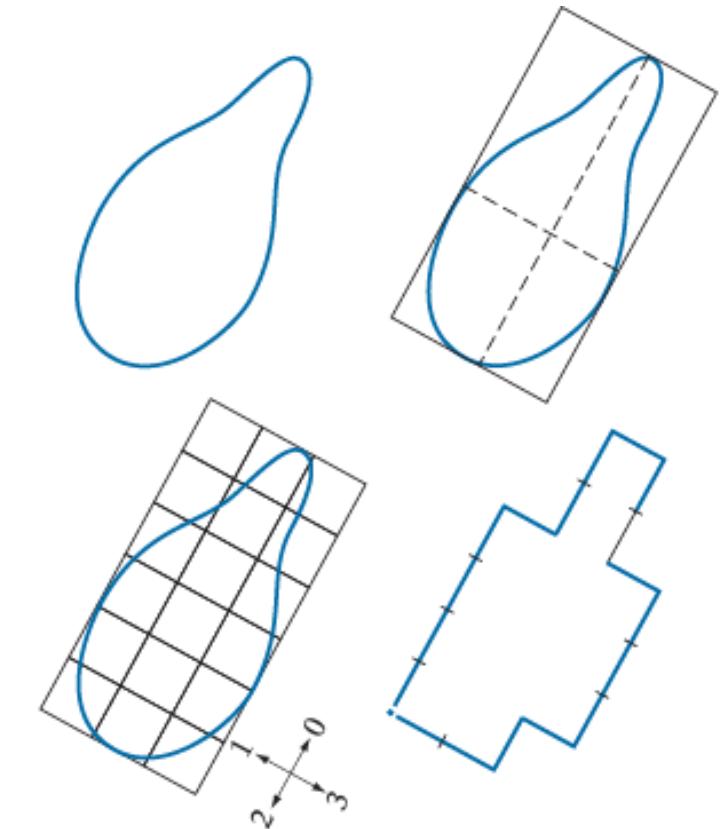
Ordre du nombre de forme

- L'ordre n d'un nombre de forme indique le **nombre de bits** nécessaire pour représenter la forme
- La quantité de formes différentes pouvant être représentées par un nombre de forme d'ordre n est **limitée**.
- **Exemple** : 1 seule forme d'ordre 4, 1 seule forme d'ordre 6, 3 formes d'ordre 8



Exemple : Calcul du nombre de forme d'ordre n

- Choisir un ordre n (ici $n = 18$)
- Trouver le rectangle de base enveloppant la forme (à voir)
- Sous-diviser le rectangle de base pour obtenir le même ordre n (ici : $3 \times 6 = 18$)
- Ré-échantillonner le contour sur la grille définie par le rectangle
- Calculer le code de Freeman et le nombre de forme



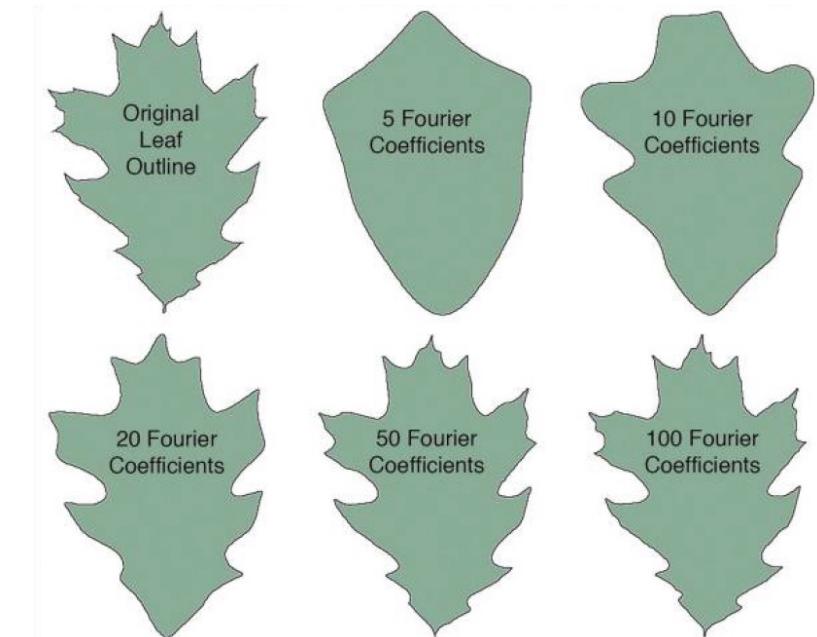
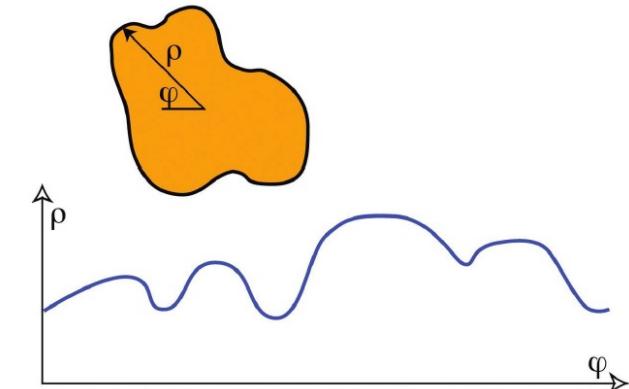
Chain code: 0 0 0 0 3 0 0 3 2 2 3 2 2 2 1 2 1 1

Difference: 3 0 0 0 3 1 0 3 3 0 1 3 0 0 3 1 3 0

Shape no.: 0 0 0 3 1 0 3 3 0 1 3 0 0 3 1 3 0 3

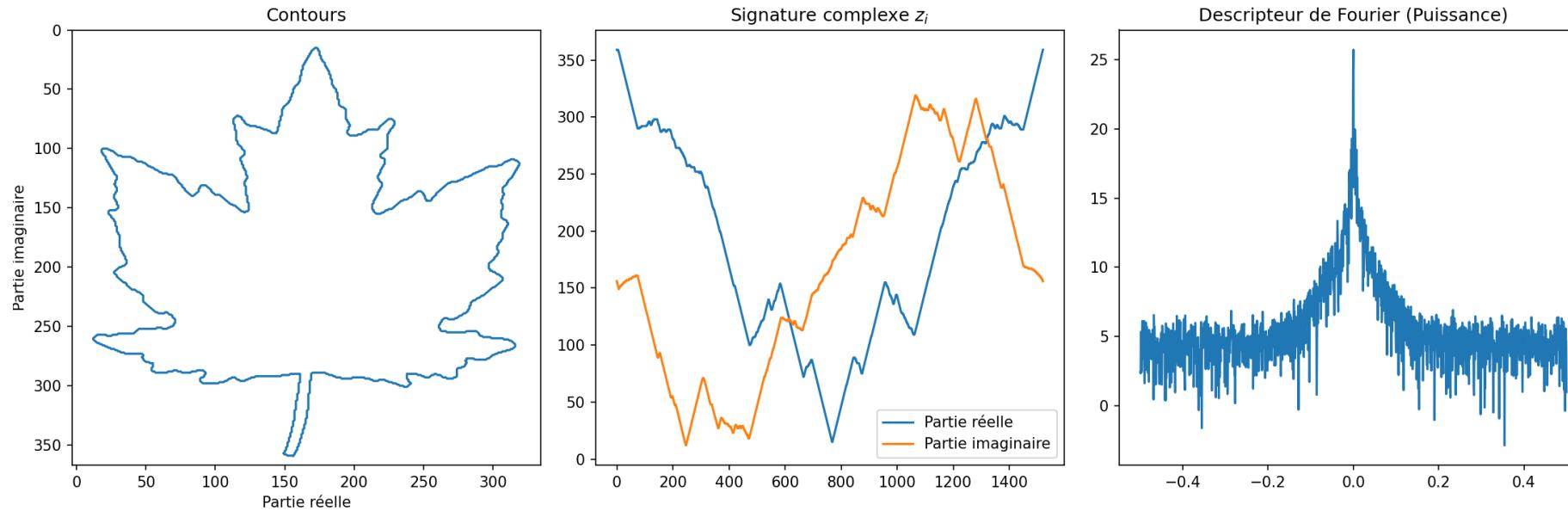
Descripteur de Fourier

- Interprétation d'un contour 2D $c_R = [x_0, x_1, \dots, x_{M-1}]$ avec $x_i = (u_i, v_i)$ en tant qu'une séquence de valeurs $[z_0, z_1, \dots, z_{M-1}]$ dans le plan complexe
$$z_i = (u_i + jv_i) \in \mathbb{C}$$
- On peut convertir cette séquence en fonction périodique 1D
- Les coefficients du spectre de Fourier 1D de cette fonction décrivent la forme du contour dans l'espace fréquentiel
- Les basses fréquences décrivent la forme grossière, et les hautes fréquences décrivent les détails du contour

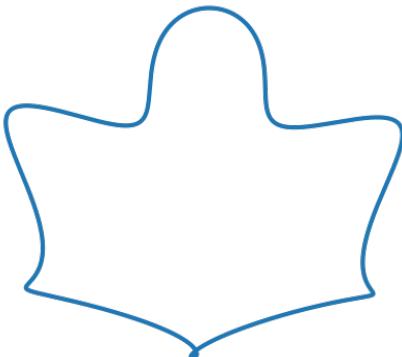


Source : Russ Fig.11.60 et 11.61

Exemple : Descripteur de Fourier



1% des coefficients



2% des coefficients



5% des coefficients



10% des coefficients



20% des coefficients



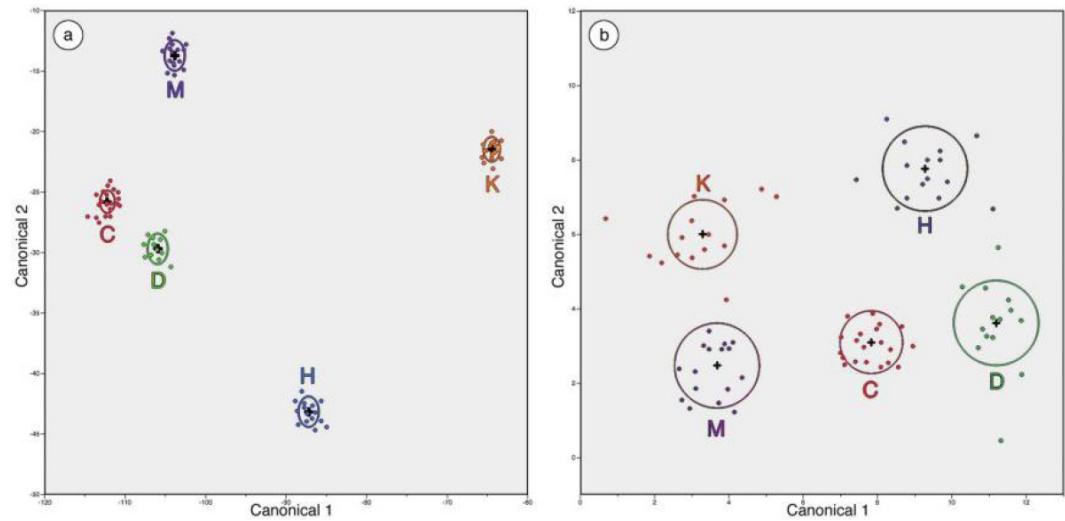
Exemple : Classification de d'artéfacts archéologiques (*Pointes de flèche*)

- Les coefficients du descripteur de Fourier peuvent être utilisés pour classifier les formes
- Ici, l'amplitude des 20 premiers coefficients a été utilisée
- De bons résultats peuvent être obtenus avec 3 coefficients.

$$C_1 = 66 \cdot FFT_4 + 98 \cdot FFT_6 - 74 \cdot FFT_{10}$$

$$C_2 = -12 \cdot FFT_4 + 54 \cdot FFT_6 - 309 \cdot FFT_{10}$$

$$C_3 = 25 \cdot FFT_4 + 95 \cdot FFT_6 - 124 \cdot FFT_{10}$$



Russ, Fig11,67-68

Caractérisation des régions

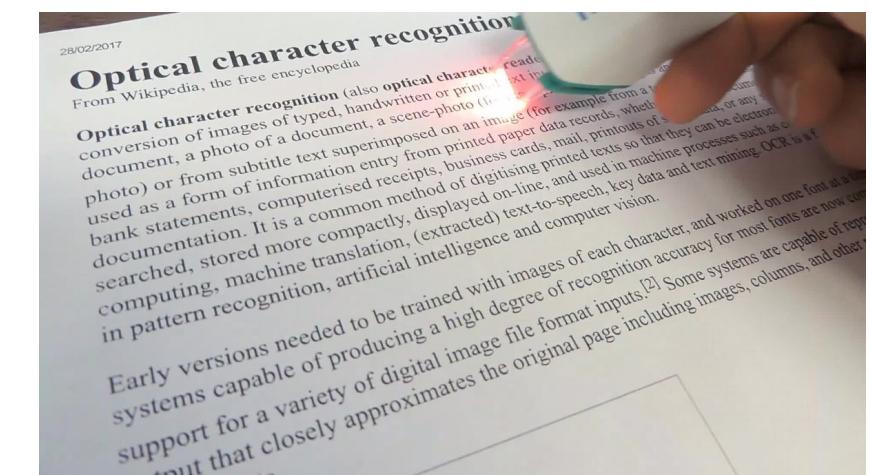
Extraction de primitives, Partie 3 – Formes et frontières

INF600F – Traitement d'images

Joël Lefebvre (UQÀM)

Propriétés d'une région binaire

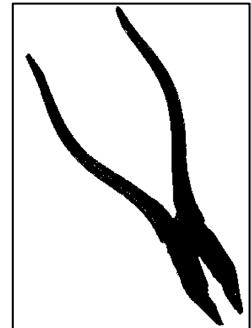
- **But** : Description d'une forme pour diverses tâches (ex. : reconnaissance de formes, détection d'objet, classification ...)
- Exemple d'applications
 - *Optical character recognition (OCR)*
 - Comptage automatique des globules rouges
 - Contrôle de qualité d'une pièce sur une chaîne de montage
- Plusieurs types de propriétés possibles
 - Forme
 - Géométrie
 - Projections
 - Topologie



Wikimedia

Caractéristique de forme (*Shape features*)

- Une caractéristique d'une région est un nombre spécifique ou une **mesure** qualitative calculable à partir des valeurs et des pixels appartenant à la forme.
- **Exemples** : taille, aire, ...
- Plusieurs caractéristiques sont combinées en un seul **vecteur caractéristique** (*feature vector*)
- C'est la « **signature** » de la **région** qui peut être utilisée pour sa classification ou pour une comparaison avec d'autres régions
- Les meilleures caractéristiques sont robustes face à certains changements (ex. : translation, rotation, mise à l'échelle)



Périmètre
Aire
Circularité
Centroïde
Excentricité
...

Caractéristiques géométriques

- Soit une région R d'une image binaire
- Elle peut être interprétée comme une distribution 2D de points d'avant-plan $x_i = (u_i, v_i)$ dans le plan discret \mathbb{Z}^2

$$R = \{x_0, x_1, \dots, x_{N-1}\} = \{(u_0, v_0), (u_1, v_1), \dots, (u_{N-1}, v_{N-1})\}$$

- La plupart des propriétés géométriques sont définies ainsi
- Les pixels n'ont pas besoin d'être connectés

C. Géométrique : Périmètre

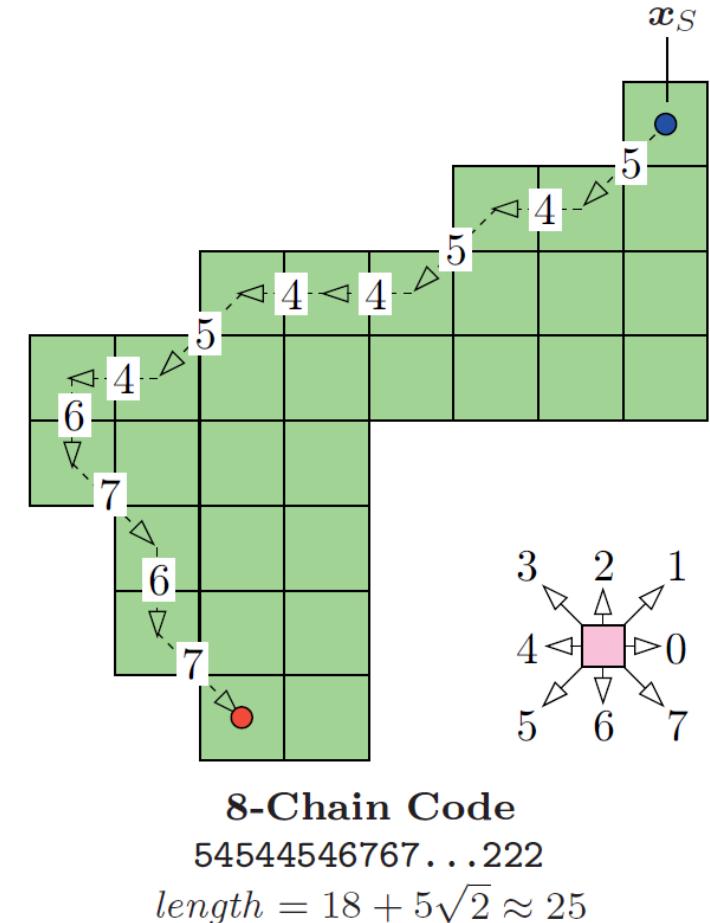
- Le **périmètre** (ou circonférence) d'une région R est la longueur de son contour extérieur
- La valeur dépend du type de voisinage (N_4 , N_8) utilisé
- Pour un code d'enchaînement utilisant N_8 $c'_R = [c'_0, c'_1, \dots, c'_{M-1}]$

$$\text{Périmètre}(R) = \sum_{i=0}^{M-1} \text{longueur}(c'_i)$$

$$\text{longueur}(c) = \begin{cases} 1 & \text{pour } c = 0, 2, 4, 6 \\ \sqrt{2} & \text{pour } c = 1, 3, 5, 7 \end{cases}$$

- Le périmètre est généralement surestimé par cette méthode. On peut donc corriger par le facteur empirique 0.95

$$P(R) \approx \text{Périmère}_{\text{corr}}(R) = 0.95 \cdot \text{Périmètre}(R)$$



C. Géométrique : Aire

- L'aire est estimée en comptant le nombre de pixels dans la région

$$A(R) = |R| = N$$

- Pour une région connectée sans trous, on peut approximer l'aire à partir de son contour fermé
- Soit un contour fermé défini par M points $(x_0, x_1, \dots, x_{M-1})$, où $x_i = (u_i, v_i)$, l'aire est donnée par la formule de Gauss pour un polygone

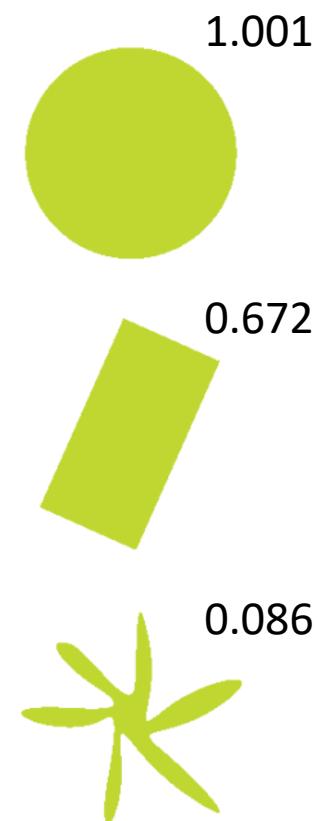
$$A(R) \approx \frac{1}{2} \cdot \left| \sum_{i=0}^{M-1} (u_i \cdot v_{(i+1)\bmod M} - u_{(i+1)\bmod M} \cdot v_i) \right|$$

C. Géométrique : Compacité et circularité

- **Compacité** : Relation entre l'aire d'une région et son périmètre
- Le périmètre P grandit linéairement, alors que l'aire A grandit quadratiquement pour un changement d'échelle de la forme
- Le ratio A/P^2 devrait être le même, peu importe l'échelle.
- Invariant face aux translations, aux rotations et aux **mises à l'échelle**

$$\text{Circularité}(R) = 4\pi \frac{A(R)}{P^2(R)}$$

- Indique la ressemblance avec un cercle

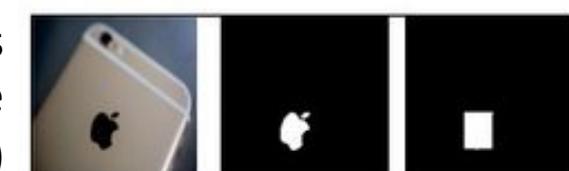
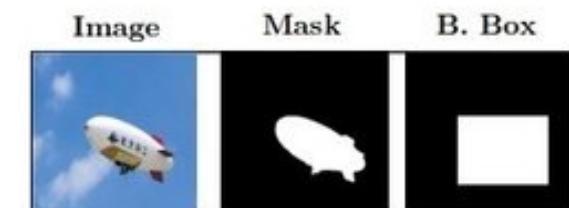
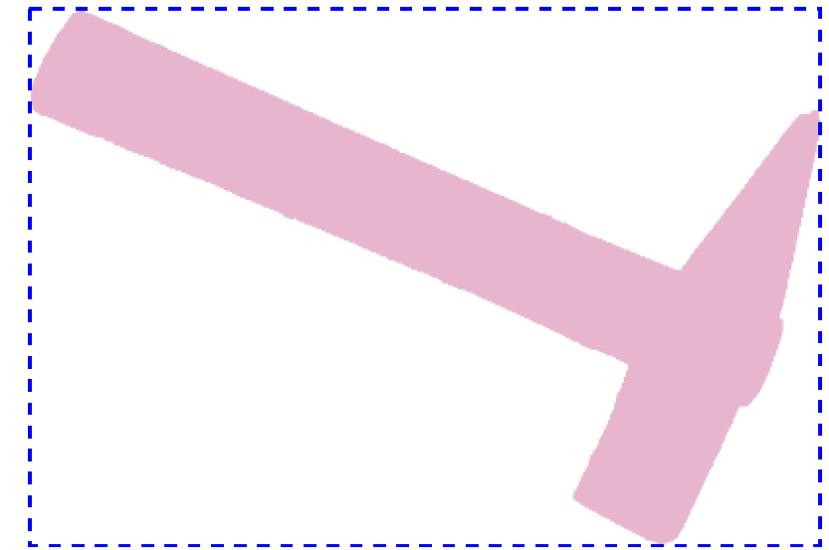


C. Géométrique : boîte englobante

- Plus petit rectangle dont les côtés sont alignés avec les axes de l'image contenant entièrement la région R

$$\text{BoundingBox}(R) = \langle u_{min}, u_{max}, v_{min}, v_{max} \rangle$$

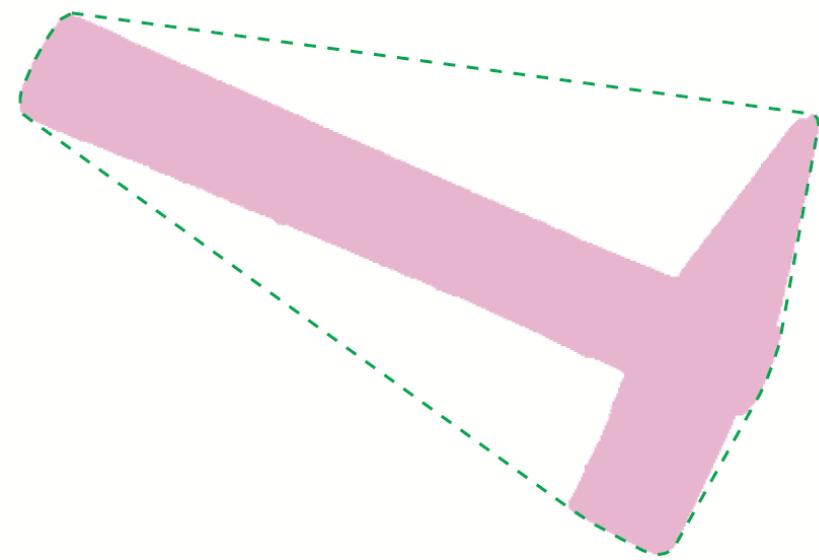
- Souvent utilisé pour évaluer les performances de détection d'objet



Exemple de boîtes englobantes générées pour le jeu de données FSS-1000 ([URL](#))

C. Géométrique : Enveloppe convexe

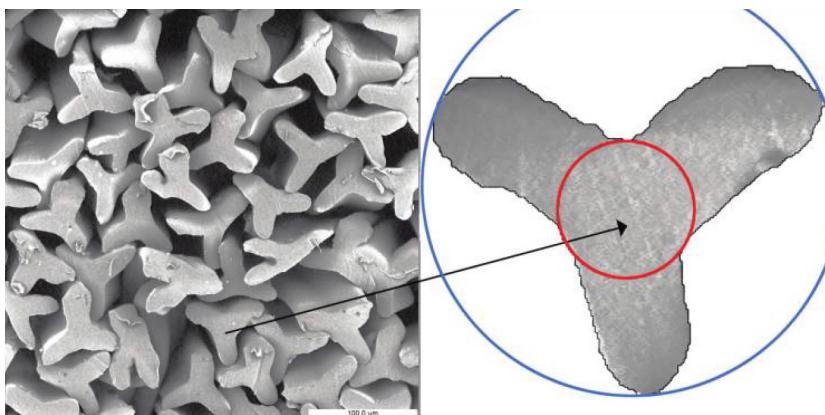
- Plus petit polygone convexe contenant entièrement la région R
- Utile pour estimer la convexité et la densité d'une région.
- **Convexité** : relation entre la longueur de l'enveloppe convexe et le périmètre original de la région
- **Densité** : ratio entre l'aire d'une région et l'aire de son enveloppe convexe.
- **Diamètre** : distance maximale entre deux nœuds de l'enveloppe convexe.



Source : Burger Vol2, Fig2.16

Quelques descripteurs de formes

- Plusieurs ratios adimensionnels calculés à partir de la longueur, du périmètre, de l'aire, ...
- Le choix de la propriété dépend de l'application

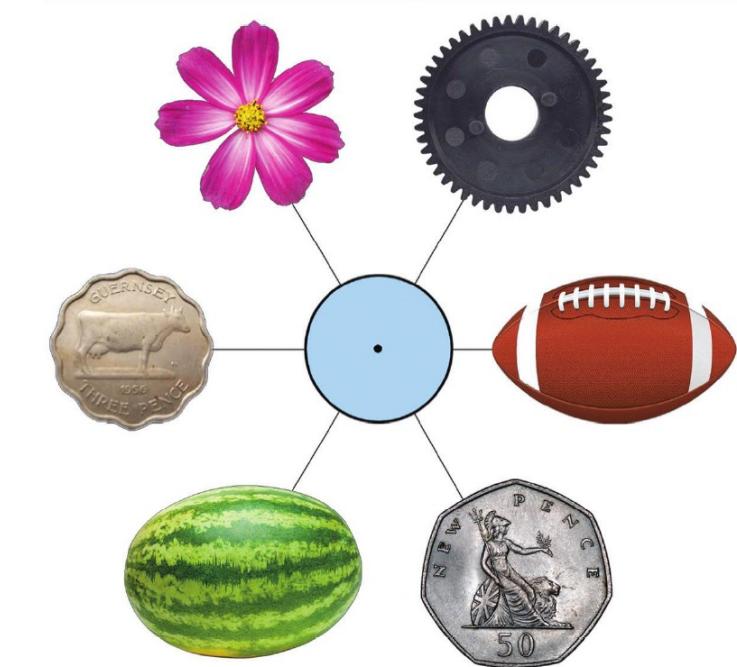


Exemple de ratio de rayons (Microscopie électronique à balayage de fibres de nylon tri-lobate, Russ Fig11.9)

Nom	Équation
Facteur de forme (<i>Form factor</i>)	$4\pi \cdot \frac{\text{Aire}}{\text{Périmètre}^2}$
Rondeur (<i>Roundness</i>)	$\frac{4 \cdot \text{Aire}}{\pi \cdot \text{Diamètre maximal}^2}$
Ratio d'aspect	$\frac{\text{Épaisseur maximale}}{\text{Épaisseur minimale}}$
Convexité	$\frac{\text{Périmètre convexe}}{\text{Périmètre}}$
Solidité	$\frac{\text{Aire}}{\text{Aire convexe}}$
Ratio des rayons	$\frac{\text{Diamètre interne}}{\text{Diamètre externe}}$
Compacité	$\frac{\sqrt{\frac{4}{\pi i} \cdot \text{Aire}}}{\text{Diamètre maximal}}$
Élongation	$\frac{\text{Longueur du filament}}{\text{Largeur du filament}}$
Courbure (<i>curl</i>)	$\frac{\text{Longueur}}{\text{Longueur du filament}}$

Exemple : Circularité

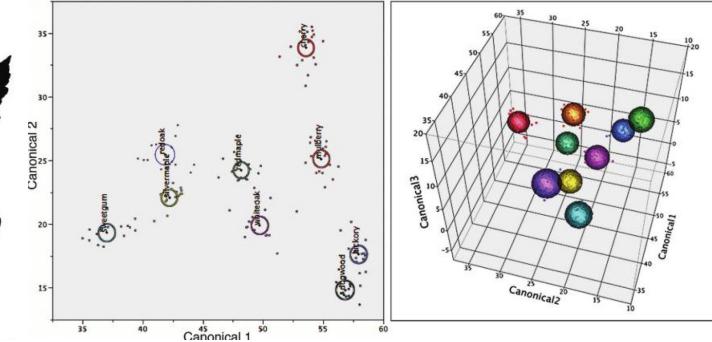
Forme	Facteur de forme	Circularité	Ratio d'aspect	Convexité	Solidité	Ratio des rayons	Compacité
Fleur	0,257	0,729	1,053	0,581	<u>0,799</u>	<u>0,381</u>	0,854
Engrenage	<u>0,256</u>	0,921	<u>1,004</u>	<u>0,526</u>	0,929	0,909	0,960
Football	0,838	<u>0,567</u>	1,653	0,965	1,000	0,605	<u>0,753</u>
Pièce	0,917	0,979	1,006	0,965	1,000	0,954	0,990
Melon	0,896	0,740	1,369	0,965	1,000	0,732	0,860
Pièce courbée	0,969	0,925	1,06	0,948	0,980	0,924	0,962
Cercle	0,934	0,987	1,013	0,968	1,000	0,987	0,994



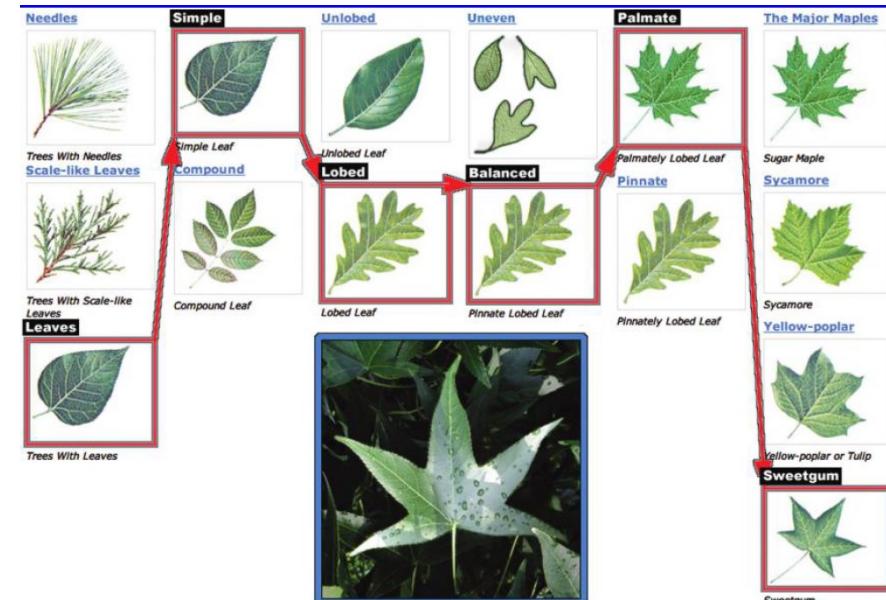
Russ Fig 11,18

Exemple : Classification des feuilles basée sur leur forme

- Classification en utilisant 3 paramètres de formes adimensionnels
 - $C1 = -0,156 \cdot \text{FormFactor} + 58,19 \cdot \text{Solidity} + 1,956 \cdot \text{RadiusRatio}$
 - $C2 = -46,44 \cdot \text{FormFactor} + 36,22 \cdot \text{Solidity} + 20,97 \cdot \text{RadiusRatio}$
 - $C3 = +5,488 \cdot \text{FormFactor} - 9,248 \cdot \text{Solidity} + 21,86 \cdot \text{RadiusRatio}$

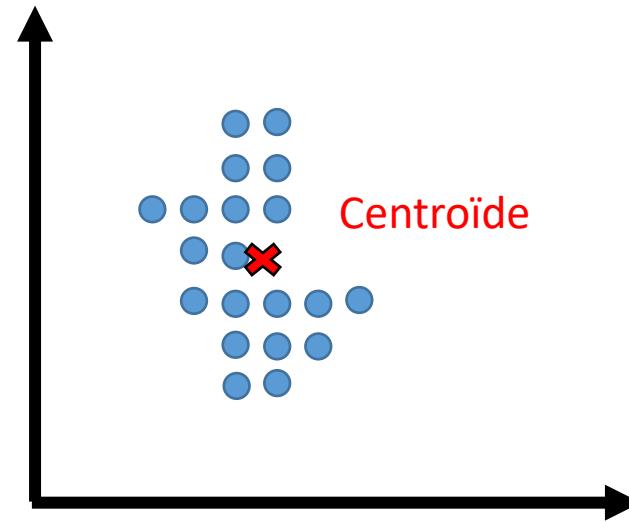


- Approche similaire des guides d'identification des feuilles



Propriétés de formes statistiques

- Considérons une région R en tant que collection de coordonnées de pixels distribués dans un espace 2D
- Les statistiques sur la distribution des points peuvent être calculées avant ou après la segmentation
- Concepts importants
 - Centroïde
 - Moments centraux



Centroïde (centre de gravité)

- **Analogie** : Si la région était un morceau de carton ayant la même forme, son centroïde est la position sur cet objet où on pourrait le tenir en équilibre avec un seul doigt.
- Obtenu par la moyenne arithmétique de toutes les positions de la région

$$\bar{x} = \frac{1}{|R|} \cdot \sum_{(u,v) \in R} u$$

$$\bar{y} = \frac{1}{|R|} \cdot \sum_{(u,v) \in R} v$$



Source : [Wikimedia](#)

Moment

- La formule pour le centroïde d'une région est un cas spécial du concept statistique plus général de **moment**

$$m_{pq} = \sum_{(u,v) \in R} I(u, v) \cdot u^p v^q$$

- Cette équation décrit le **moment (ordinaire)** d'ordre p, q pour la fonction discrète (image) $I(u, v) \in R$.
- Pour une image binaire, $I(u, v) \in \{0,1\}$, et on peut simplifier

$$m_{pq} = \sum_{(u,v) \in R} u^p v^q$$

Moments, aire et centroïde

- L'aire d'une région binaire est son **moment d'ordre zéro**

$$A(R) = |R| = \sum_{(u,v) \in R} 1 = \sum_{(u,v) \in R} u^0 v^0 = m_{00}(R)$$

- Le centroïde est relié au moment d'ordre 1

$$\bar{x} = \frac{1}{|R|} \cdot \sum_{(u,v) \in R} u^1 v^0 = \frac{m_{10}(R)}{m_{00}(R)}$$

$$\bar{y} = \frac{1}{|R|} \cdot \sum_{(u,v) \in R} u^0 v^1 = \frac{m_{01}(R)}{m_{00}(R)}$$

- Ces moments représentent des propriétés concrètes de la région.
 - m_{00} est une information importante pour décrire la région,
 - le centroïde permet d'obtenir une estimation exacte et fiable de la position de la région

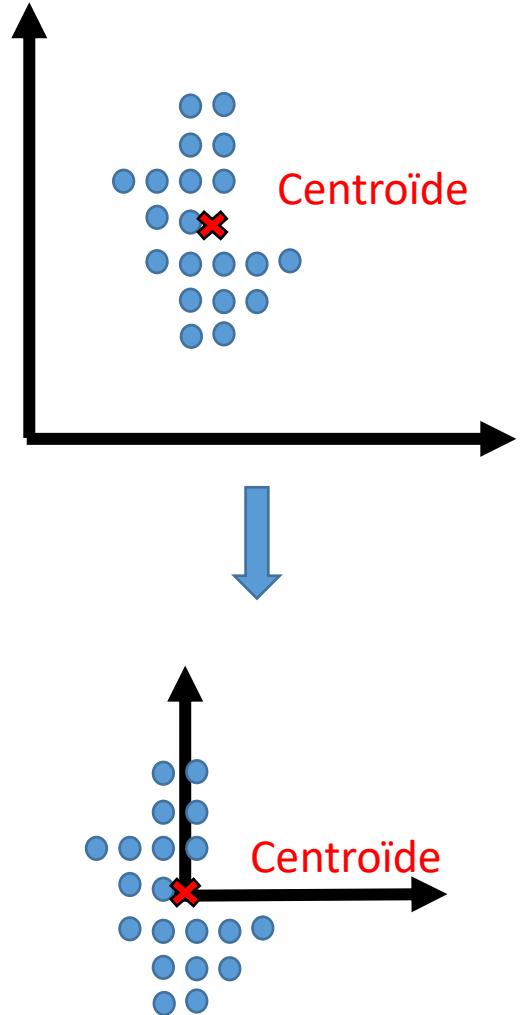
Moments centraux

- Le centroïde d'une région peut être utilisé pour obtenir des caractéristiques de région invariante à la translation
- On déplace l'origine du système de coordonnées sur le centroïde pour obtenir les **moments centraux d'ordre p, q**

$$\mu_{pq}(R) = \sum_{(u,v) \in R} I(u, v) \cdot (u - \bar{x})^p \cdot (v - \bar{y})^q$$

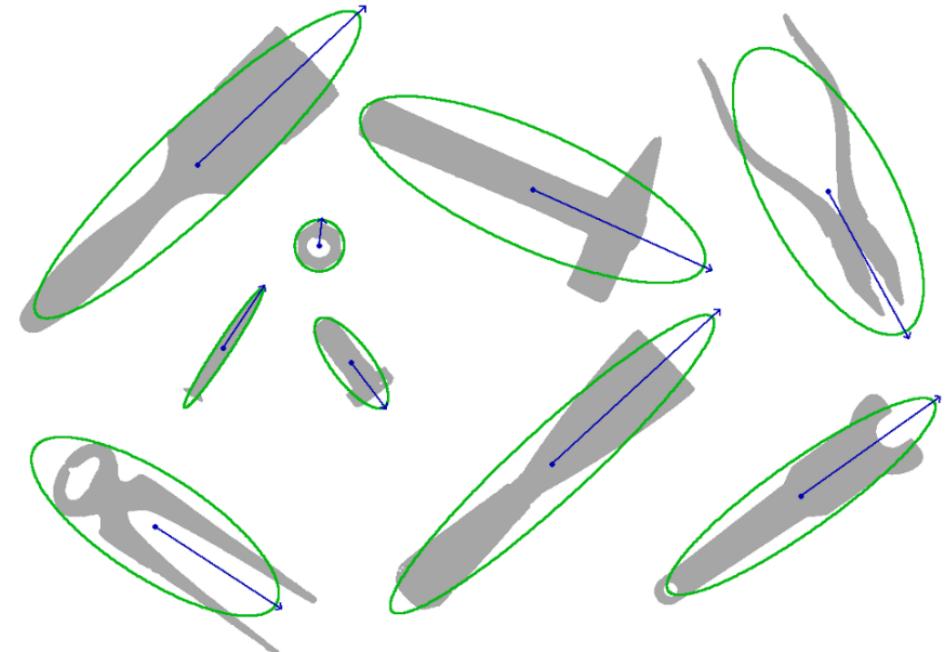
- Pour une image binaire ($I(u, v) = 1$ dans la région R)

$$\mu_{pq}(R) = \sum_{(u,v) \in R} (u - \bar{x})^p \cdot (v - \bar{y})^q$$



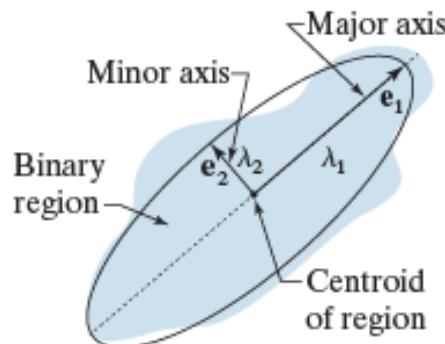
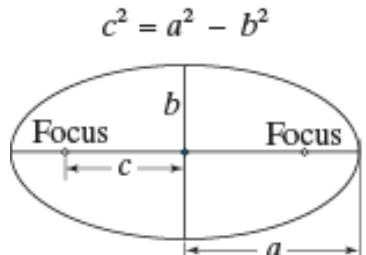
Propriétés géométriques basées sur le moment

- Les moments peuvent être utilisés directement pour classifier les régions
- On peut aussi les utiliser pour extraire des propriétés géométriques intéressantes
- Exemples : **Orientation** et **excentricité**



Burger Vol2, Fig2.19

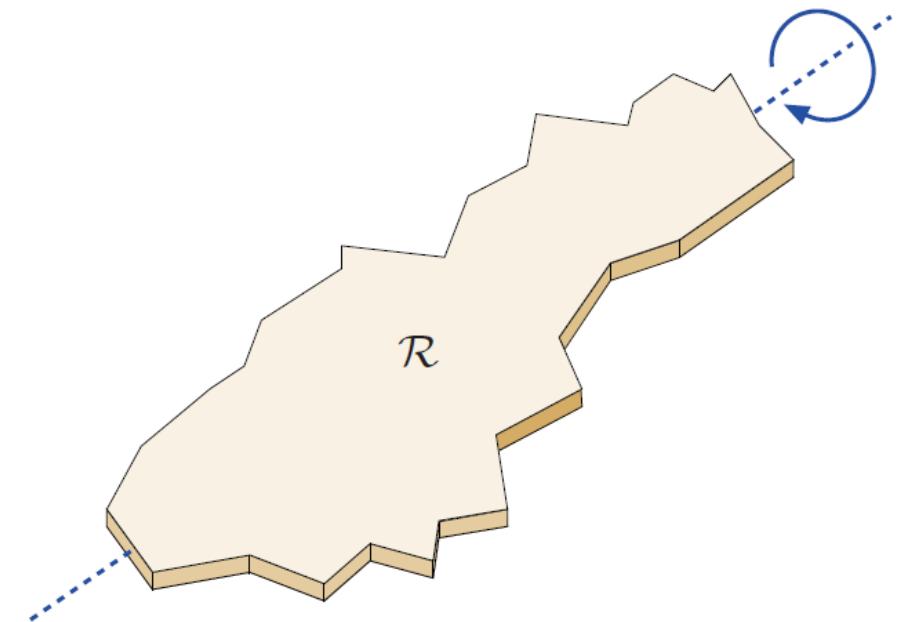
Gonzalez
Fig12.21



$e_1 \lambda_1$ and $e_2 \lambda_2$ are the eigenvectors and corresponding eigenvalues of the covariance matrix of the coordinates of the region

Orientation d'une région

- Décrit la direction de l'axe principal, c.-à-d. l'axe qui traverse le centroïde et qui est parallèle à la partie la plus large de la région.
- Parfois nommé l'axe principal de rotation
- **Analogie :** toupie



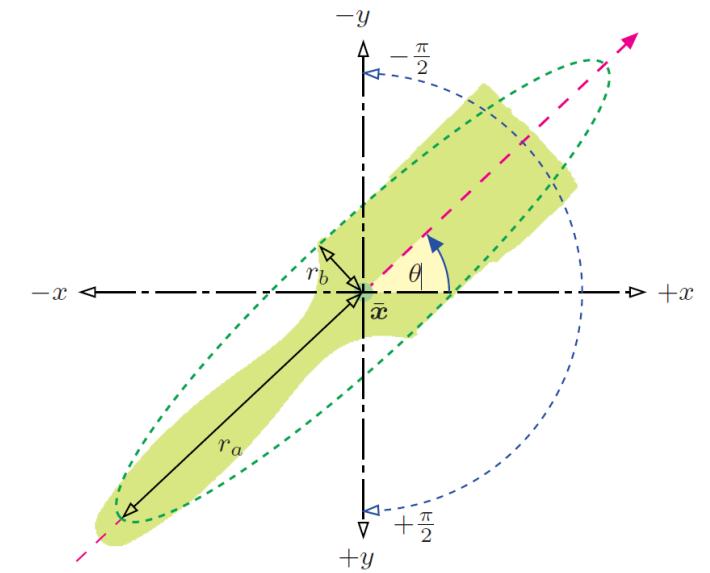
Source : [Spoon & Tamago](#)

Calcul de l'orientation principale de la région

- Si la région possède une orientation ($\mu_{20}(R) \neq \mu_{02}(R)$), alors la direction θ_R de l'axe principal peut être calculée à l'aide des moments centraux μ_{pq}

$$\theta_R = \frac{1}{2} \tan^{-1} \left(\frac{2 \cdot \mu_{11}(R)}{\mu_{20}(R) - \mu_{02}(R)} \right)$$

- L'orientation sera entre $[-\frac{\pi}{2}, \frac{\pi}{2}]$
- Les orientations calculées à l'aide des moments de la région sont en général très précises.



Burger Vol2, Fig2.18

Calcul du vecteur d'orientation de la région

- **Tâche fréquente** : Visualiser l'orientation principale d'une région à l'aide d'une ligne ou d'une flèche dont l'origine est sur le centroïde de la région $\bar{x} = (\bar{x}, \bar{y})$
- Le **vecteur d'orientation** $x_d = (x_d, y_d)$ peut être calculé directement à partir des moments

$$x_d = \cos(\theta_R) = \begin{cases} 0 & \text{pour } A = B = 0 \\ \left[\frac{1}{2} \left(1 + \frac{B}{\sqrt{A^2 + B^2}} \right) \right]^{1/2} & \text{sinon} \end{cases}$$

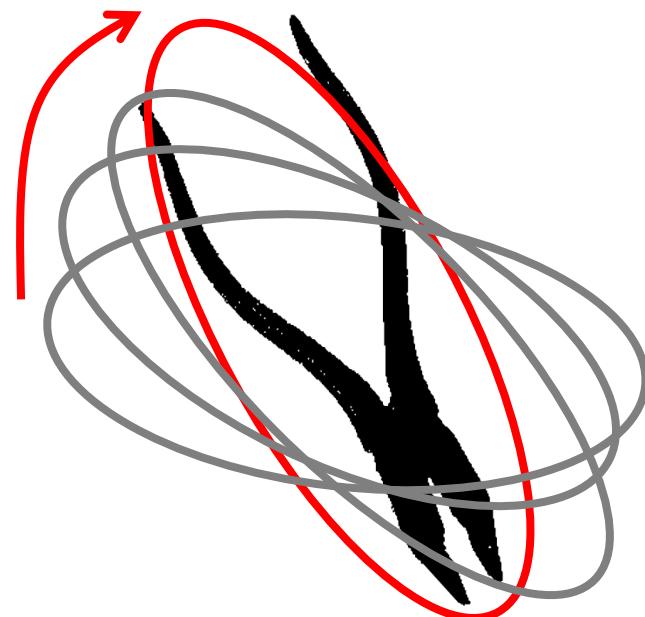
$$y_d = \cos(\theta_R) = \begin{cases} 0 & \text{pour } A = B = 0 \\ \left[\frac{1}{2} \left(1 - \frac{B}{\sqrt{A^2 + B^2}} \right) \right]^{1/2} & \text{pour } A \geq 0 \\ - \left[\frac{1}{2} \left(1 - \frac{B}{\sqrt{A^2 + B^2}} \right) \right]^{1/2} & \text{pour } A < 0 \end{cases}$$

$$A = 2\mu_{11}(R)$$

$$B = \mu_{20}(R) - \mu_{02}(R)$$

Excentricité de la région

- Mesure « l'élongation » de la région
- **Approche naïve** : tourner la région pour trouver l'ellipse englobante ayant le plus grand ratio d'aspect possible
- **Meilleure approche** : Utiliser les moments de la région pour obtenir une estimation stable et précise de l'excentricité sans nécessité de recherche itérative ou d'optimisation



$$Ecc(R) = \frac{\mu_{20} + \mu_{02} + \sqrt{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}}{\mu_{20} + \mu_{02} - \sqrt{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}}$$

Moments invariants (*Moments de Hu*)

- Les moments centraux normalisés ne sont pas affectés par des translations et des mises à l'échelle uniforme de la région, mais en général les rotations vont modifier leurs valeurs.
- Solution à ce problème : **Moments de Hu** calculés à partir de combinaisons de moments centraux normalisés
- Ces moments sont invariants face aux translations, rotations et mise à l'échelle
- **skimage.measure.moments_hu**

$$\begin{aligned}H_1 &= \bar{\mu}_{20} + \bar{\mu}_{02}, \\H_2 &= (\bar{\mu}_{20} - \bar{\mu}_{02})^2 + 4\bar{\mu}_{11}^2, \\H_3 &= (\bar{\mu}_{30} - 3\bar{\mu}_{12})^2 + (3\bar{\mu}_{21} - \bar{\mu}_{03})^2, \\H_4 &= (\bar{\mu}_{30} + \bar{\mu}_{12})^2 + (\bar{\mu}_{21} + \bar{\mu}_{03})^2, \\H_5 &= (\bar{\mu}_{30} - 3\bar{\mu}_{12}) \cdot (\bar{\mu}_{30} + \bar{\mu}_{12}) \cdot [(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - 3(\bar{\mu}_{21} + \bar{\mu}_{03})^2] \\&\quad + (3\bar{\mu}_{21} - \bar{\mu}_{03}) \cdot (\bar{\mu}_{21} + \bar{\mu}_{03}) \cdot [3(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - (\bar{\mu}_{21} + \bar{\mu}_{03})^2], \\H_6 &= (\bar{\mu}_{20} - \bar{\mu}_{02}) \cdot [(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - (\bar{\mu}_{21} + \bar{\mu}_{03})^2] \\&\quad + 4\bar{\mu}_{11} \cdot (\bar{\mu}_{30} + \bar{\mu}_{12}) \cdot (\bar{\mu}_{21} + \bar{\mu}_{03}), \\H_7 &= (3\bar{\mu}_{21} - \bar{\mu}_{03}) \cdot (\bar{\mu}_{30} + \bar{\mu}_{12}) \cdot [(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - 3(\bar{\mu}_{21} + \bar{\mu}_{03})^2] \\&\quad + (3\bar{\mu}_{12} - \bar{\mu}_{30}) \cdot (\bar{\mu}_{21} + \bar{\mu}_{03}) \cdot [3(\bar{\mu}_{30} + \bar{\mu}_{12})^2 - (\bar{\mu}_{21} + \bar{\mu}_{03})^2].\end{aligned}$$

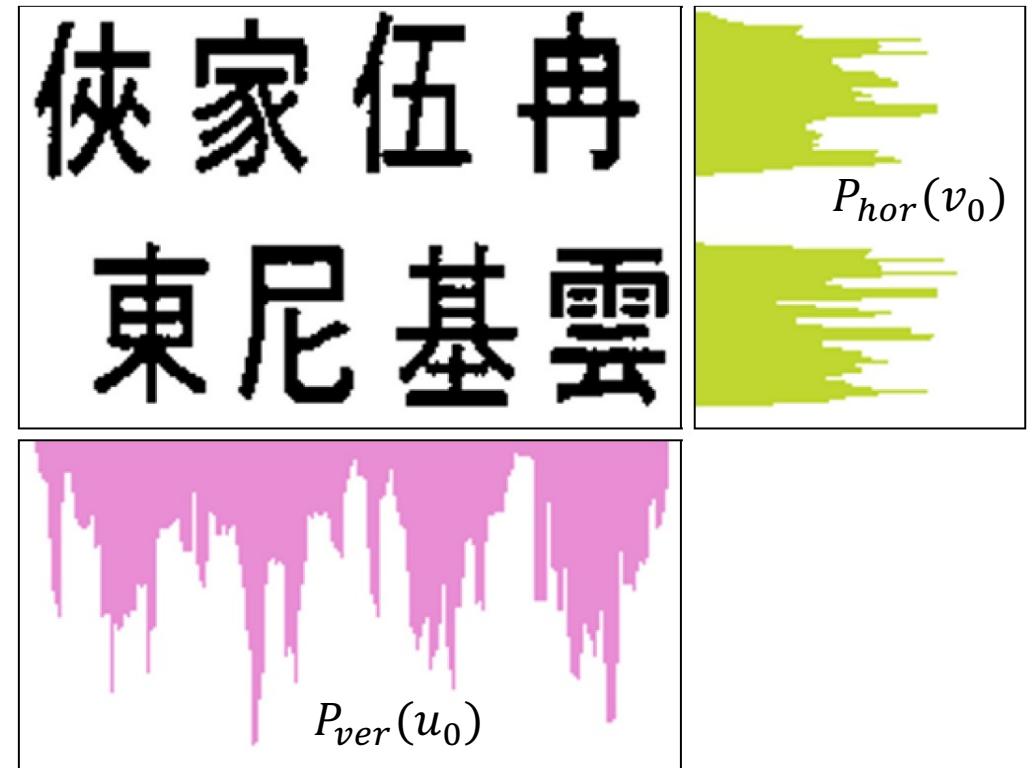
Projections

- Les projections d'images sont une représentation 1D du contenu de l'image

$$P_{hor}(v_0) = \sum_{u=0}^{M-1} I(u, v_0)$$

$$P_{ver}(u_0) = \sum_{v=0}^{N-1} I(u_0, v)$$

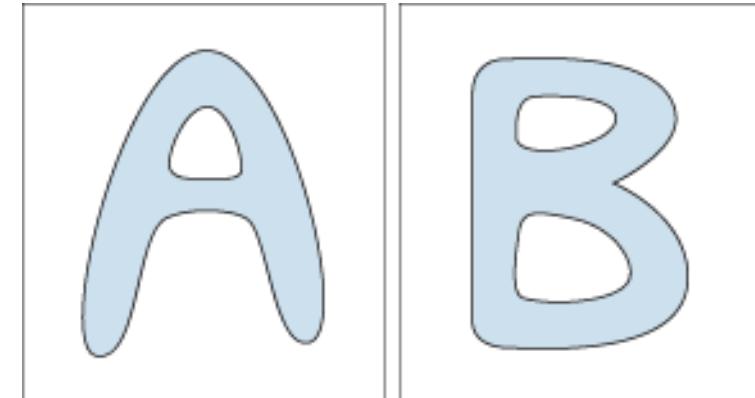
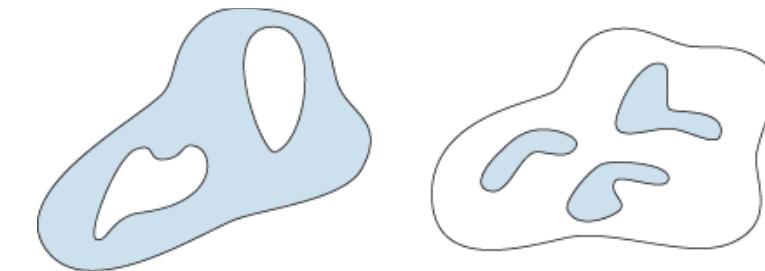
- Souvent utilisé pour analyser rapidement la structure d'une image et isoler ses composants
- En pratique, les projections sont réalisées selon les mêmes directions que l'axe principal de la région.
- Résulte en un vecteur de description invariant à la rotation (sa « **signature** »)



Propriétés de topologie

- Ne décrit pas la forme d'une région en termes continus, mais représente plutôt les propriétés de sa structure.
- Généralement invariant face à des transformations extrêmes de l'image
- Quelques propriétés
 - $N_R(R)$: Nombre de régions
 - $N_L(R)$: Nombre de trous
 - N_E : **Nombre d'Euler** ($N_R(R) - N_L(R)$). Exemple :
 $N_E(8) = -1$
- Les propriétés topologiques sont souvent combinées à des propriétés numériques pour classifier des images.

Région avec 2 trous, et une région avec 3 composantes connectées (Gonzalez Fig12.25)



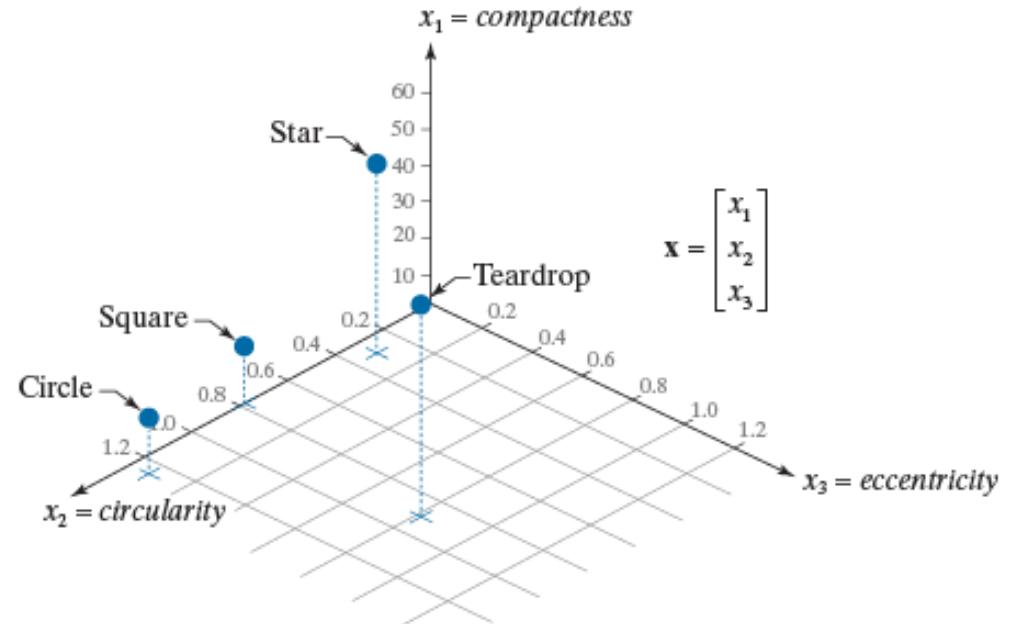
A : $N_E = 0$, B : $N_E = -1$
Gonzalez Fig12.26

Comparaison des caractéristiques

Quelques caractéristiques pour des régions simples

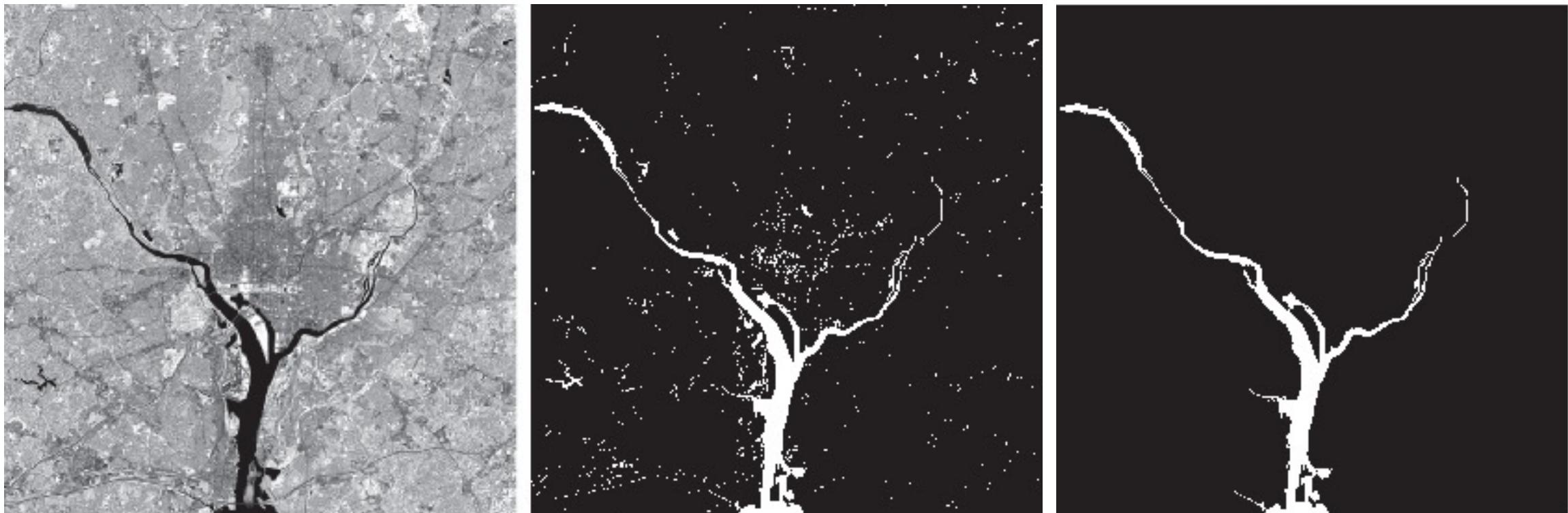
Descriptor				
Compactness	10.1701	42.2442	15.9836	13.2308
Circularity	1.2356	0.2975	0.7862	0.9478
Eccentricity	0.0411	0.0636	0	0.8117

Espace des caractéristiques 3D
(feature space)



Utiliser les caractéristiques d'aire

- Exemple : Détection du plus grand objet dans l'image



Gonzalez Fig2.28

Python et propriétés des régions

- Plusieurs fonctions pour mesurer des propriétés de régions sont réunies dans le module **skimage.measure** ([URL](#))

Fonction	Description
skimage.measure.euler_number (image[, ...])	Calculate the Euler characteristic in binary image.
skimage.measure.label (label_image[, ...])	Label connected regions of an integer array.
skimage.measure.moments (image[, order])	Calculate all raw image moments up to a certain order.
skimage.measure.moments_central (image[, ...])	Calculate all central image moments up to a certain order.
skimage.measure.moments_coords (coords[, order])	Calculate all raw image moments up to a certain order.
skimage.measure.moments_coords_central (coords)	Calculate all central image moments up to a certain order.
skimage.measure.moments_hu (nu)	Calculate Hu's set of image moments (2D-only).
skimage.measure.moments_normalized (mu[, order])	Calculate all normalized central image moments up to a certain order.
skimage.measure.perimeter (image[, neighbourhood])	Calculate total perimeter of all objects in binary image.
skimage.measure.regionprops (label_image[, ...])	Measure properties of labeled image regions.
skimage.measure.regionprops_table (label_image)	Compute image properties and return them as a pandas-compatible table.

Caractérisation de la texture

Extraction de primitives, Partie 3 – Formes et frontières

INF600F – Traitement d'images

Joël Lefebvre (UQÀM)

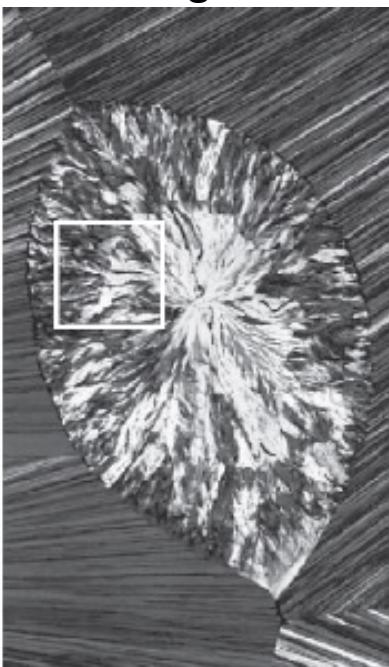
Caractérisation de la texture

- Approche importante pour décrire une région
- La texture permet de mesurer la douceur, la grossièreté et la régularité des patrons d'intensités présentes dans une région

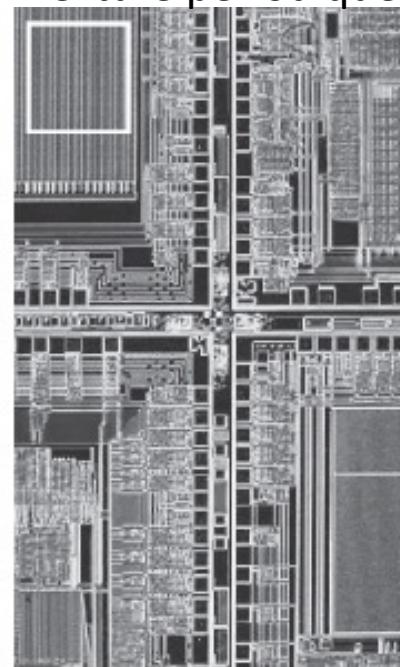
Texture lisse



Texture grossière



Texture périodique



Gonzalez, Fig12.29

Approche statistique pour la description de la texture

- Approche la plus simple : utiliser les moments statistiques de l'intensité d'une image ou de d'une région
- Soit z une valeur d'intensité, et $p(z_i), i = 0, 1, 2, \dots, L - 1$ l'histogramme normalisé pour L niveaux d'intensités
- Le n^e moment de z par rapport à la moyenne est

$$\mu_n(z) = \sum_{i=0}^{L-1} (z_i - m)^n p(z_i)$$

Avec $m = \sum_{i=0}^{L-1} z_i p(z_i)$ l'intensité moyenne de la région

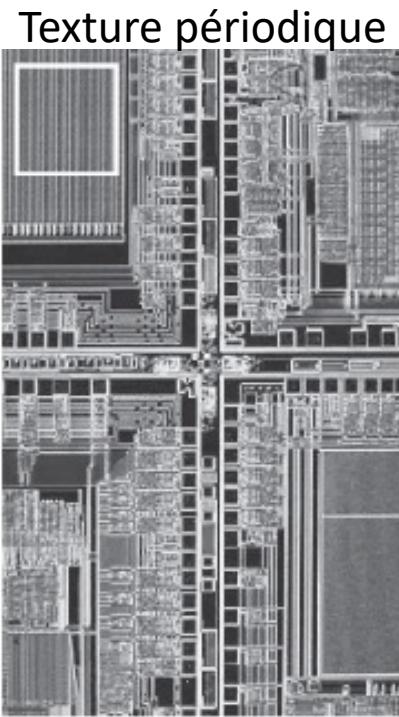
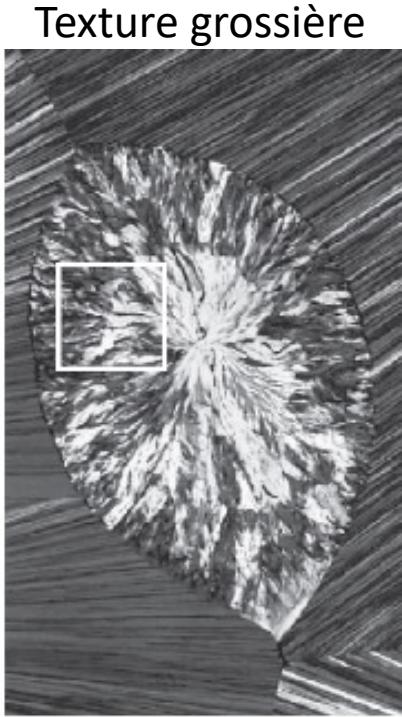
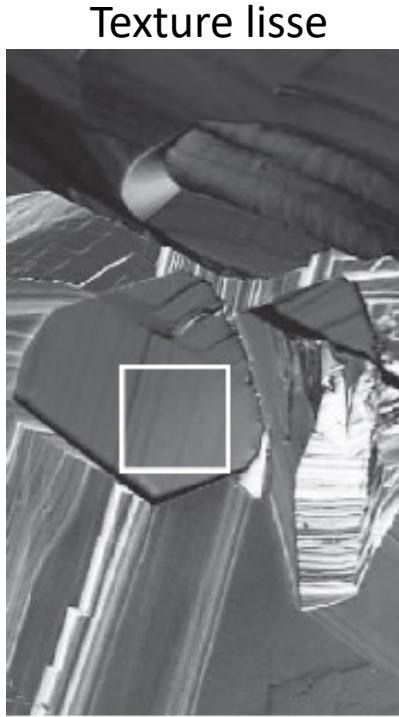
Texture et moments statistiques

- **2^e moment** $\mu_2(z)$ (variance) mesure le contraste d'intensité et peut être utilisé pour décrire la douceur des intensités relatives

$$R(z) = 1 - \frac{1}{1 + \mu_2(z)}$$

- Le **3^e moment** $\mu_3(z)$ mesure l'asymétrie de l'histogramme
- Le **4^e moment** $\mu_4(z)$ mesure sa platitude (*flatness*)
- Les 5^e et 6^e moment sont aussi utiles pour décrire la texture
- **Uniformité** : $U(z) = \sum_{i=0}^{L-1} p^2(z_i)$
- **Entropie moyenne** : $e(z) = - \sum_{i=0}^{L-1} p(z_i) \log_2 p(z_i)$ (Variabilité)

Exemple : Descripteur de texture basé sur l'histogramme



Texture	Lisse	Grossière	Périodique
Moyenne	82,64	143,56	99,72
Écart-type	11,79	74,63	33,73
R (normalisé)	0,002	0,079	0,017
3 ^e moment	-0,105	-0,151	0,750
Uniformité	0,026	0,005	0,013
Entropie	5,434	7,783	6,674

Matrice de cooccurrence

- Les mesures de texture basées sur l'histogramme ne contiennent aucune information sur les relations spatiales entre les intensités
- Pour obtenir plus d'informations, on peut extraire la position relative entre 2 pixels grâce à la **matrice de cooccurrence G**
- Compte le nombre d'occurrences de paires d'intensités pour deux pixels espacés d'une distance prédefinie $Q = (\Delta x, \Delta y)$

Pour calculer une matrice de cooccurrence G , on doit d'abord choisir un nombre de niveaux d'intensité L et un déplacement Q entre 2 pixels.
Ici, $L = 8$ et $Q = (1,0)$

	1	2	3	4	5	6	7	8
1	1	2	0	0	0	1	1	0
2	0	0	0	0	1	1	0	0
3	0	1	0	1	0	0	0	0
4	0	0	1	0	1	0	0	0
5	2	0	1	0	1	0	0	0
6	1	3	0	0	0	0	0	1
7	0	0	0	0	1	1	0	2
8	1	0	0	0	0	2	2	1

Image f Co-occurrence matrix \mathbf{G}

Gonzalez, Fig12.30

Analyse d'une matrice de cooccurrence

- Plusieurs descripteurs existent pour analyser G
- On doit d'abord normaliser la matrice : $p_{ij} = g_{ij}/n$

Descripteur	Explication	Équation
Probabilité maximum	Mesure la réponse maximale de G (entre 0 et 1)	$\max_{i,j}(p_{ij})$
Corrélation	Mesure le niveau de corrélation d'un pixel avec son voisin pour toute l'image	$-\sum_i \sum_j \frac{(i - m_r)(j - m_c)p_{ij}}{\sigma_r \sigma_c}$
Contraste	Mesure le contraste d'intensité entre le pixel et son voisin pour toute l'image	$-\sum_i \sum_j (i - j)^2 p_{ij}$
Uniformité (Énergie)	L'uniformité = 1 pour une image constante	$-\sum_i \sum_j p_{ij}^2$
Homogénéité	Mesure la proximité spatiale à la diagonale de la distribution des éléments de G	$-\sum_i \sum_j \frac{p_{ij}}{1 + i - j }$
Entropie	Mesure le caractère aléatoire des éléments de G	$-\sum_i \sum_j p_{ij} \log_2 p_{ij}$

$$m_r = \sum_i i \sum_j p_{ij}$$

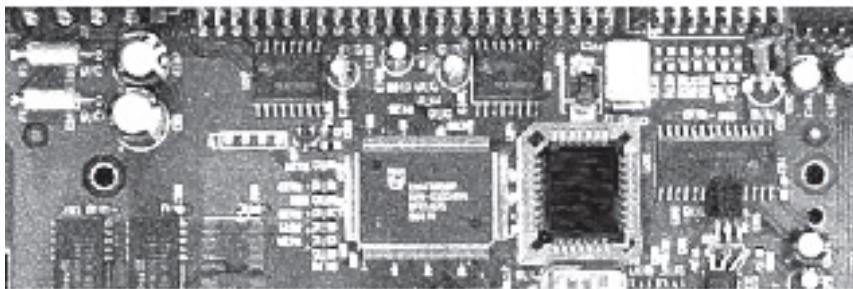
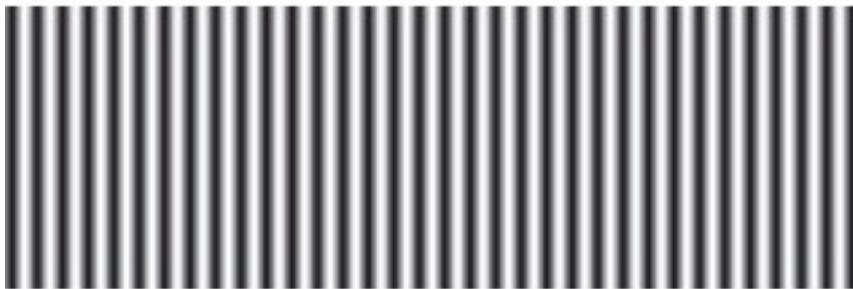
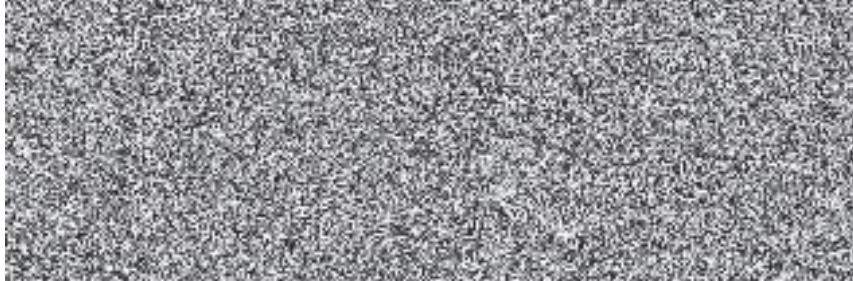
$$m_c = \sum_i j \sum_j p_{ij}$$

$$\sigma_r^2 = \sum_i (i - m_r)^2 \sum_j p_{ij}$$

$$\sigma_c^2 = \sum_i (j - m_c)^2 \sum_j p_{ij}$$

Exemple : Descripteur de texture basé sur la matrice de cooccurrence

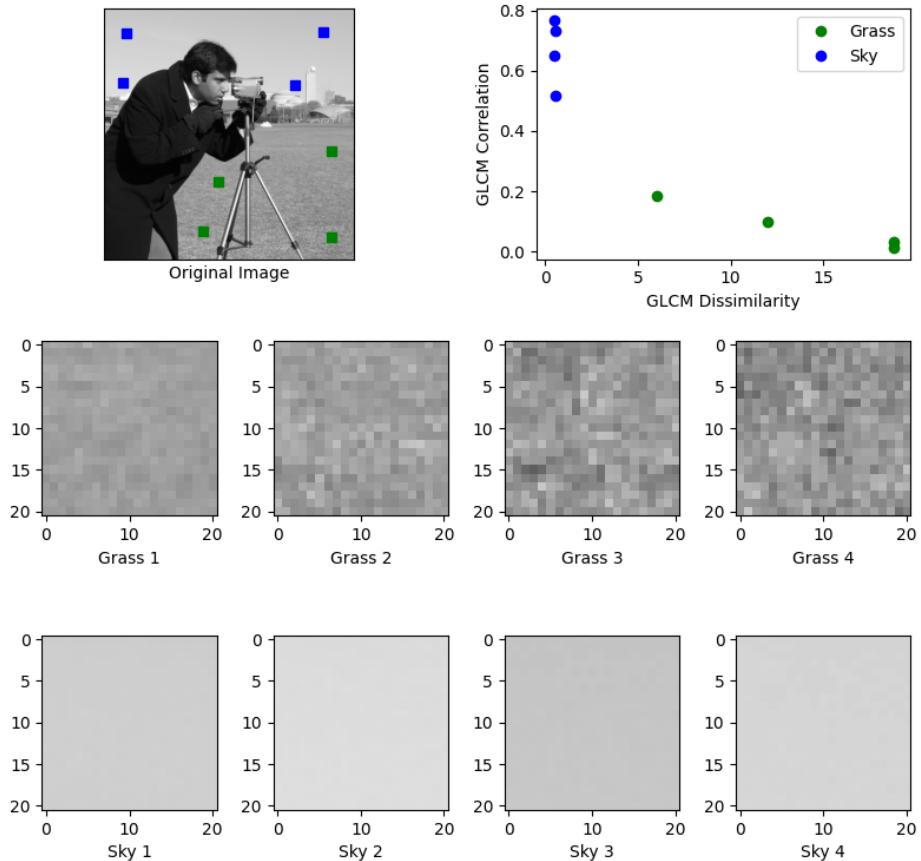
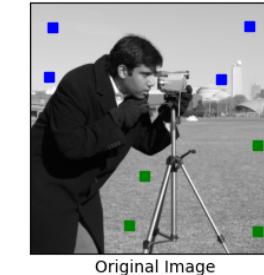
$L=256, Q=(1,0)$



Matrice de cooccurrence normalisée	Probabilité maximum	Corrélation	Contraste	Uniformité	Homogénéité	Entropie
G_1/n_1	0,00006	-0,0005	10838	0,00002	0,0366	15,75
G_2/n_2	0,01500	0,9650	00570	0,01230	0,0824	6,43
G_3/n_3	0,06860	0,8798	01356	0,00480	0,2048	13,58

Analyse de texture et Python

- **skimage.feature.greycomatrix :**
Calcule la matrice de cooccurrence
(niveaux de gris) ([URL](#))
- **skimage.feature.greycoprops :**
Calcule les propriétés de la matrice de cooccurrence ([URL](#))
- **Démo Python : GLCM Texture Features** ([URL](#))



Application : *Radiomics* en cancérologie

- ***Radiomics*** : l'extraction et l'analyse d'un **grand nombre de caractéristiques quantitatives** dans des images médicales afin d'améliorer la **caractérisation** des maladies et la **prédiction** de l'effet des traitements

