

Signaux, Sons et Images pour l'Informaticien: Morphomathématiques

Diane Lingrand

Polytech SI3

2016 - 2017

Outline

- Notions de base
- Dilatation
- Erosion
- Ouverture et Fermeture
- Hit and miss, Amincissement, Squelette

- Concerne initialement les images en noir et blanc (Matheron et Serra, 1965)
 - notion de fond / forme
- Etendu aux images en noir et blanc (Dougherty, 1978)
- Puis aux images en couleur

Notions de base

appartenance : blanc pour la forme, noir pour le fond

intersection de 2 ensembles : AND

union de 2 ensembles : OR

complémentaire : vidéo inverse

complémentaire de A par rapport à B :

$$A/B = A \setminus B = A \cap B^c = A / (A \cap B)$$

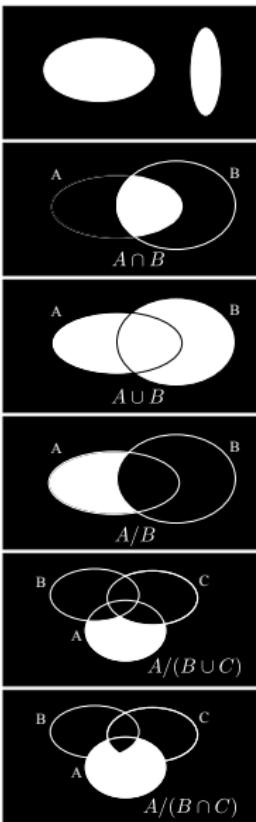
Lois de Morgan :

$$A / (B \cup C) = (A/B) \cap (A/C)$$

$$A / (B \cap C) = (A/B) \cup (A/C)$$

$$(A \cup B)^c = A^c \cap B^c$$

$$(A \cap B)^c = A^c \cup B^c$$



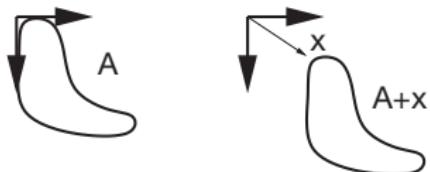
Addition de Minkowski

Réflexion d'un ensemble A : $-A = \{-a : a \in A\}$



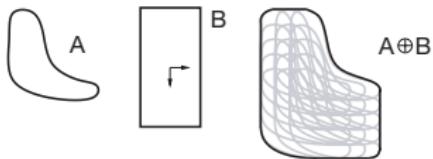
Ensemble symétrique : $-A = A$

Addition d'un élément : $B = A + x = \{a + x : a \in A\}$



Addition de Minkowski : commutative, appelée également "dilatation"

$$\begin{aligned} C &= A \oplus B = \bigcup (A + b : b \in B) \\ &= \bigcup (B + a : a \in A) = \{x : (-B + x) \cap A \neq \emptyset\} \end{aligned}$$



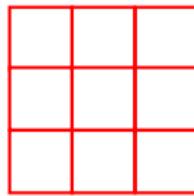
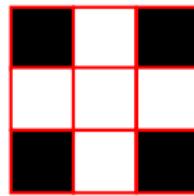
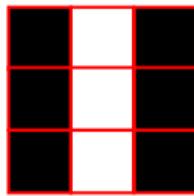
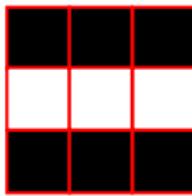
Eléments structurants

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

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

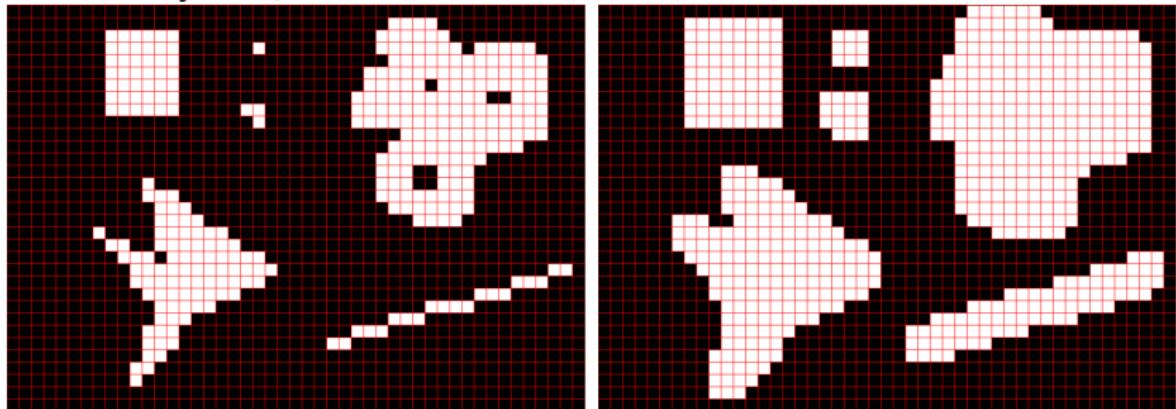
$$k_c = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$k_s = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Exemple de dilatation d'image en noir et blanc

avec le noyau k_s



Algorithme de dilatation d'image noir et blanc

```
double[][] dilate(int[][] ImageInput, int[][] k)
    uc <- k.length/2
    vc <- k[0].length/2
    parcours des pixels (x,y) de Image de uc à width-uc
    et de vc à height-vc
        ImageOutput[x][y] vaut noir
        pour i de -uc à +uc et pour j de -vc à +vc
            si k[i+uc][j+vc] vrai et ImageInput[x-i][y-j] blanc
                alors ImageOutput[x][y] vaut blanc
renvoie ImageOutput
```

Algorithme de dilatation d'image en niveaux de gris

```
double[][] dilate(int [][] ImageInput, int [][] k)
    uc <- k.length/2
    vc <- k[0].length/2
    parcours des pixels (x,y) de Image de uc à width-uc
    et de vc à height-vc
        max initialisé à 0
        pour i de -uc à +uc et pour j de -vc à +vc
            si k[i+uc][j+vc] vrai et ImageInput[x-i][y-j] > max
                alors max <- ImageInput[x-i][y-j]
        ImageOutput[x][y] <- max
renvoie ImageOutput
```

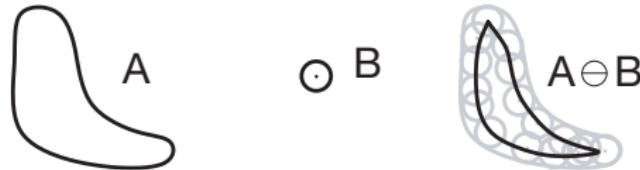
- Soustraction de Minkowski :

$$C = A - (-B) = \bigcap(A + b : b \in B)$$

- Erosion de A par B :

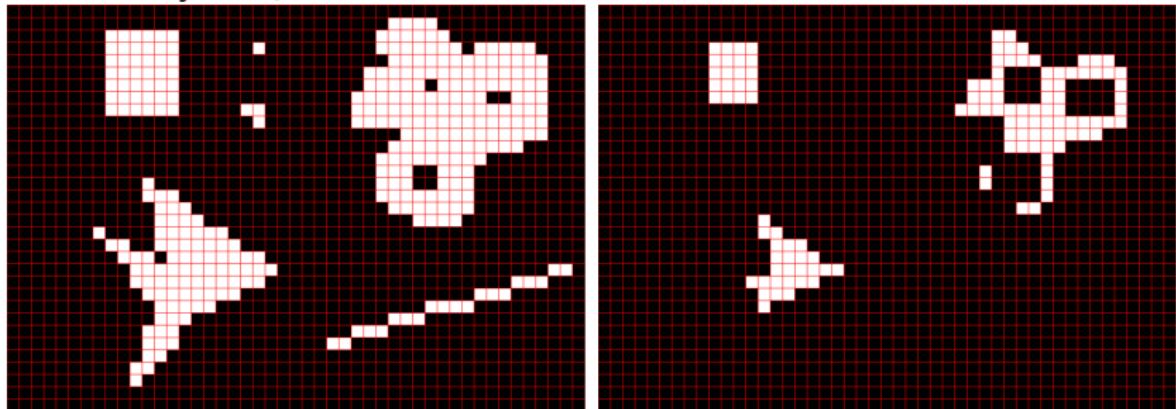
$$C = A \ominus B = \bigcap(A - b : b \in B) = \{x : B + x \subset A\}$$

- Il y a équivalence si B est symétrique.



Exemple d'érosion d'image en noir et blanc

avec le noyau k_s



Algorithme d'érosion d'image noir et blanc

```
double[][] erode(int[][] ImageInput, int[][] k)
    uc <- k.length/2
    vc <- k[0].length/2
    parcours des pixels (x,y) de Image de uc à width-uc
    et de vc à height-vc
        ImageOutput[x][y] vaut blanc
        pour i de -uc à +uc et pour j de -vc à +vc
            si k[i+uc][j+vc] vrai et ImageInput[x-i][y-j] noir
                alors ImageOutput[x][y] vaut noir
renvoie ImageOutput
```

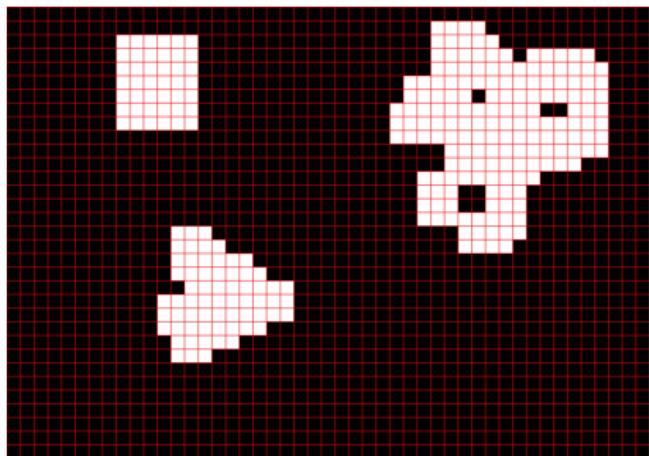
Algorithme d'érosion d'image en niveaux de gris

```
double[][] erode(int[][][] ImageInput, int[][] k)
    uc <- k.length/2
    vc <- k[0].length/2
    parcours des pixels (x,y) de Image de uc à width-uc
    et de vc à height-vc
        min <- 255
        pour i de -uc à +uc et pour j de -vc à +vc
            si k[i+uc][j+vc] vrai et ImageInput[x-i][y-j] < min
                alors min <- ImageInput[x-i][y-j]
        ImageOutput[x][y] <- min
renvoie ImageOutput
```

Ouverture / Fermeture

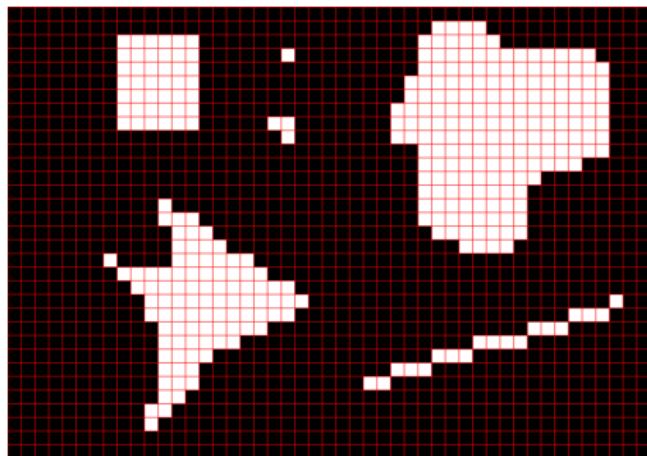
Ouverture (*opening*) : érosion puis dilatation

$$A \circ B = (A \ominus B) \oplus B$$



Fermeture (*closing*) : dilatation puis érosion

$$A \bullet B = (A \oplus B) \ominus B$$



L'application duale φ_d d'une application φ est définie par :

$$[\varphi_d(A)] = [\varphi(A^c)]^c$$

- dilatation et érosion

$$\begin{aligned} A \ominus B &= (A^c \oplus (-B))^c \\ A \oplus B &= (A^c \ominus (-B))^c \end{aligned}$$

- ouverture et fermeture :

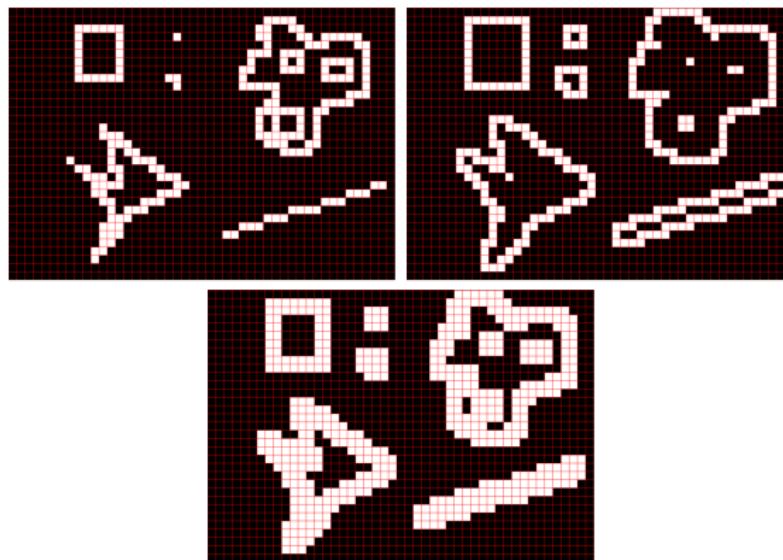
$$\begin{aligned} A \circ B &= (A^c \bullet B)^c \\ A \bullet B &= (A^c \circ B)^c \end{aligned}$$

Gradient morphologique

contour intérieur : $A/(A \ominus B)$

contour extérieur : $(A \oplus B)/A$

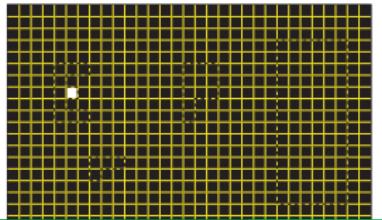
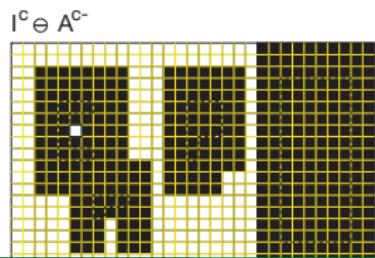
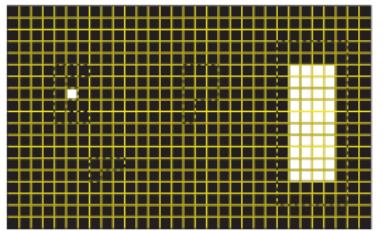
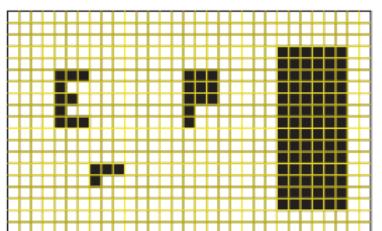
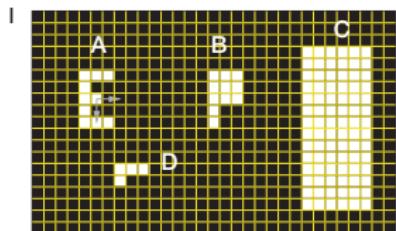
contour moyen (ou gradient morphologique) : $(A \oplus B)/(A \ominus B)$



Hit or Miss / Tout ou rien

$$I \otimes A = (I \ominus A) \cap (I^c \ominus A^{c-})$$

avec A^{c-} complémentaire local de A



- Erosion qui conserve les structures :

$$\text{thin}(I) = I / (I \otimes B)$$

- Choix pour B :

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

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

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

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

Algorithme d'amincissement

- Notation pour les voisins d'un pixel

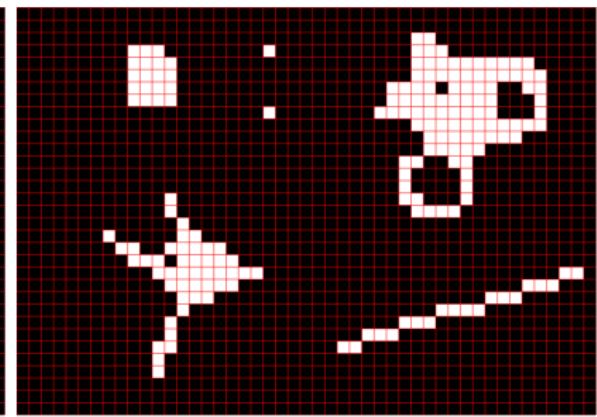
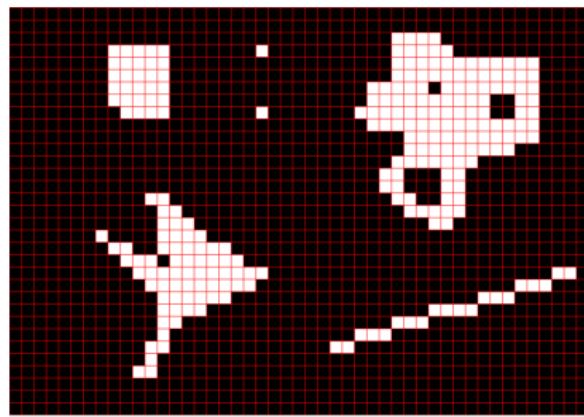
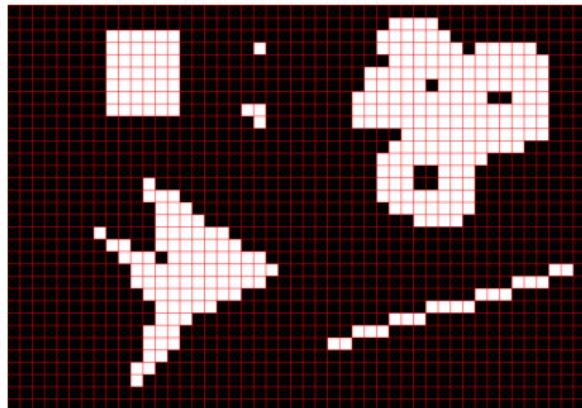
$v[5] =$ $(I[x - 1][y - 1] == 0)$	$v[4] =$ $(I[x][y - 1] == 0)$	$v[3] =$ $(I[x + 1][y - 1] == 0)$
$v[6] =$ $(I[x - 1][y] == 0)$	$I[x][y]$	$v[2] =$ $(I[x + 1][y] == 0)$
$v[7] =$ $(I[x - 1][y + 1] == 0)$	$v[0] =$ $(I[x][y + 1] == 0)$	$v[1] =$ $(I[x + 1][y + 1] == 0)$

- En 2 passes

Amincissement en 2 passes

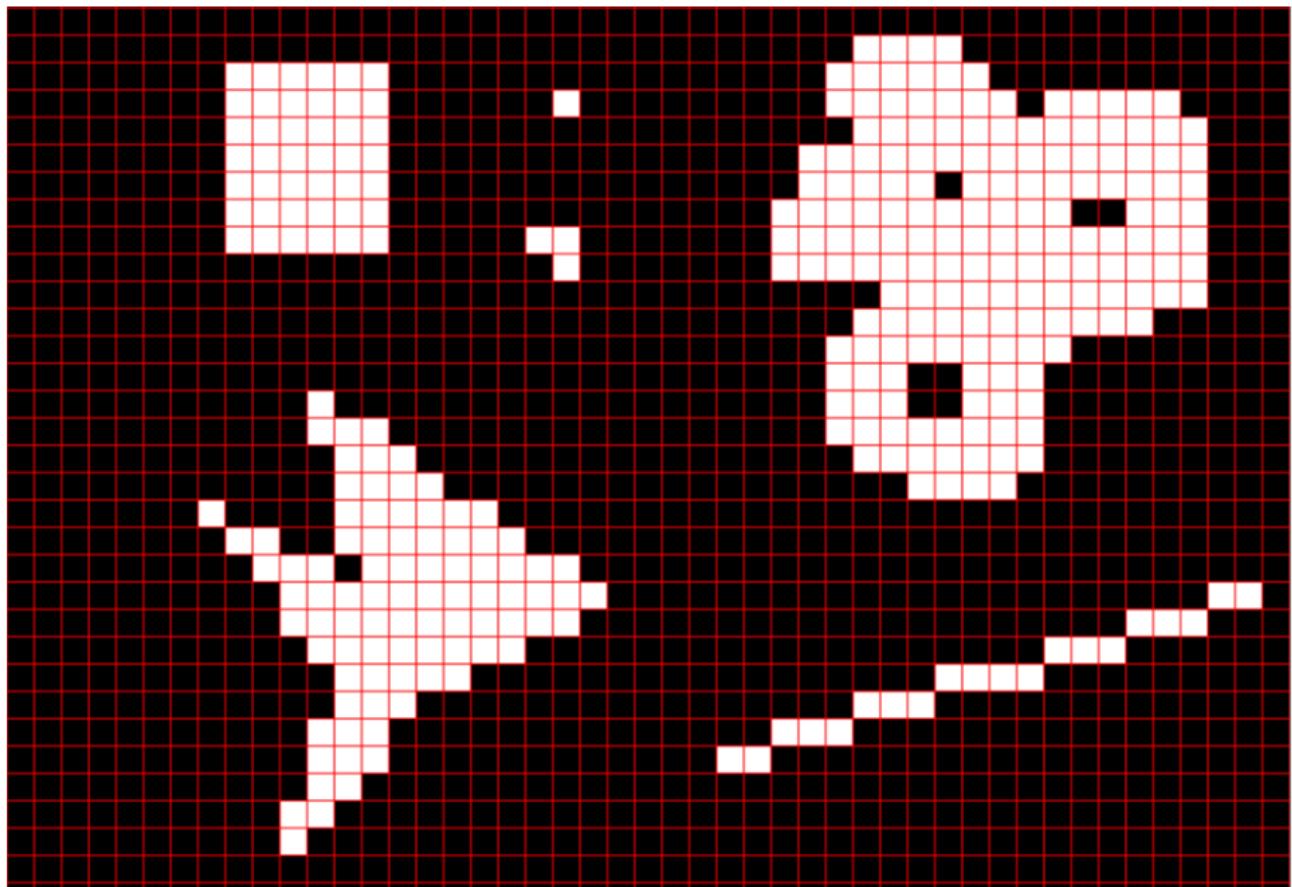
- Première passe : conditions à remplir pour qu'un pixel passe de la forme au fond :
 - ➊ $I[x][y] \neq 0$ (sinon, le pixel est déjà à la couleur du fond)
 - ➋ nombre de voisins ≥ 2 (conserver les squelettes d'épaisseur inf. à 2)
 - ➌ nombre de voisins ≤ 6 (être en bordure d'un masque 3x3)
 - ➍ nombre de transitions false-true égal à 1 (éviter la fragmentation)
 - ➎ $v[0] \parallel v[2] \parallel v[4]$ vaut true
 - ➏ $v[2] \parallel v[4] \parallel v[6]$ vaut true (conservation des lignes connectées)
- Seconde passe : conditions à remplir pour qu'un pixel passe de la forme au fond :
 - ➊ $I[x][y] \neq 0$
 - ➋ nombre de voisins ≥ 2
 - ➌ nombre de voisins ≤ 6
 - ➍ nombre de transitions false-true égale à 1
 - ➎ $v[0] \parallel v[2] \parallel v[6]$ vaut true
 - ➏ $v[0] \parallel v[4] \parallel v[6]$ vaut true

Illustration des 2 passes :

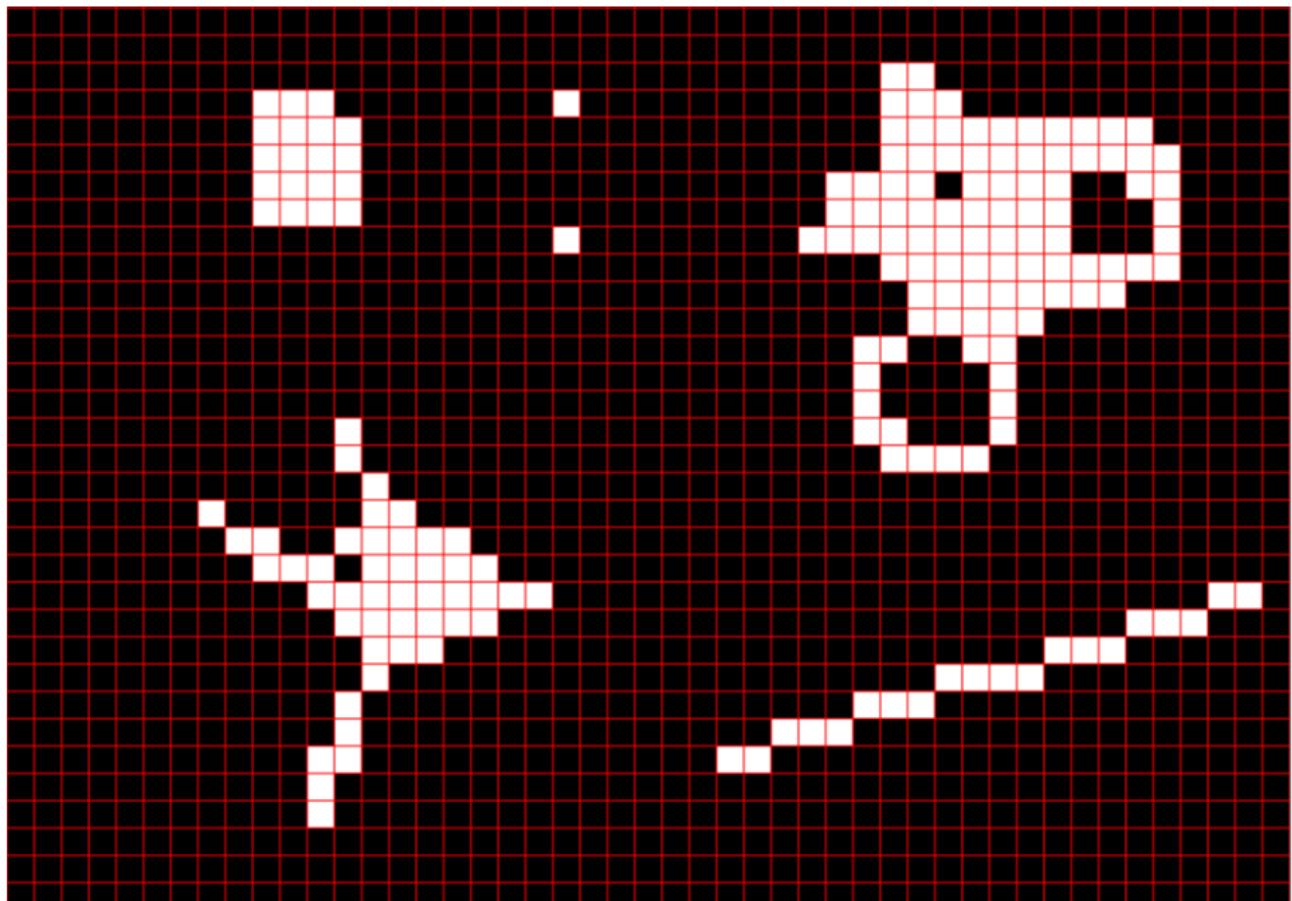


- Squelette d'une forme : axe médian (trace des centres des disques maximaux entièrement contenus cette forme).
- Calcul d'un squelette : par amincissements successifs.
- Intérêt : reconnaissance de caractères, détection de réseaux routiers, planification de trajectoires.

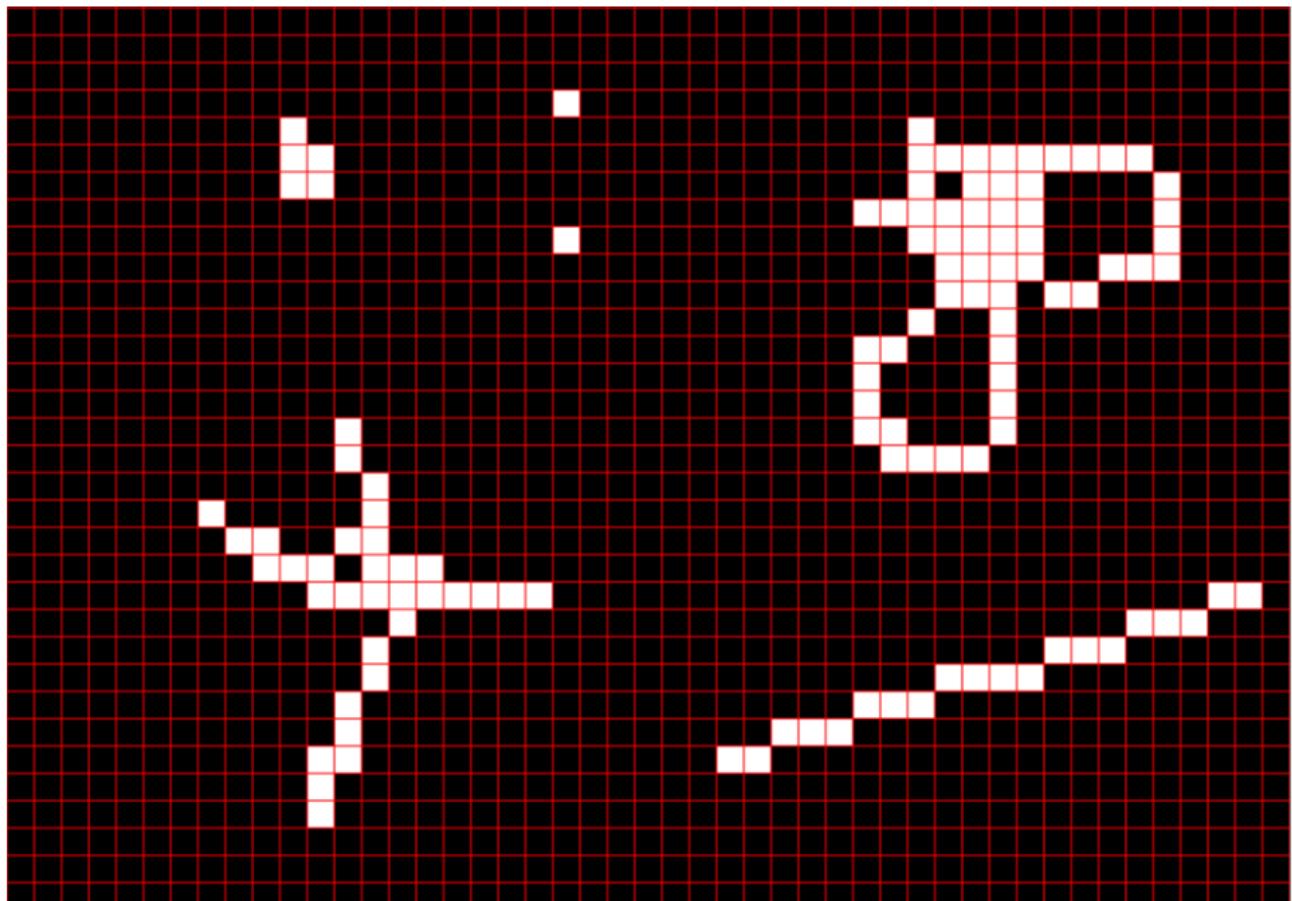
Squelette en exemple



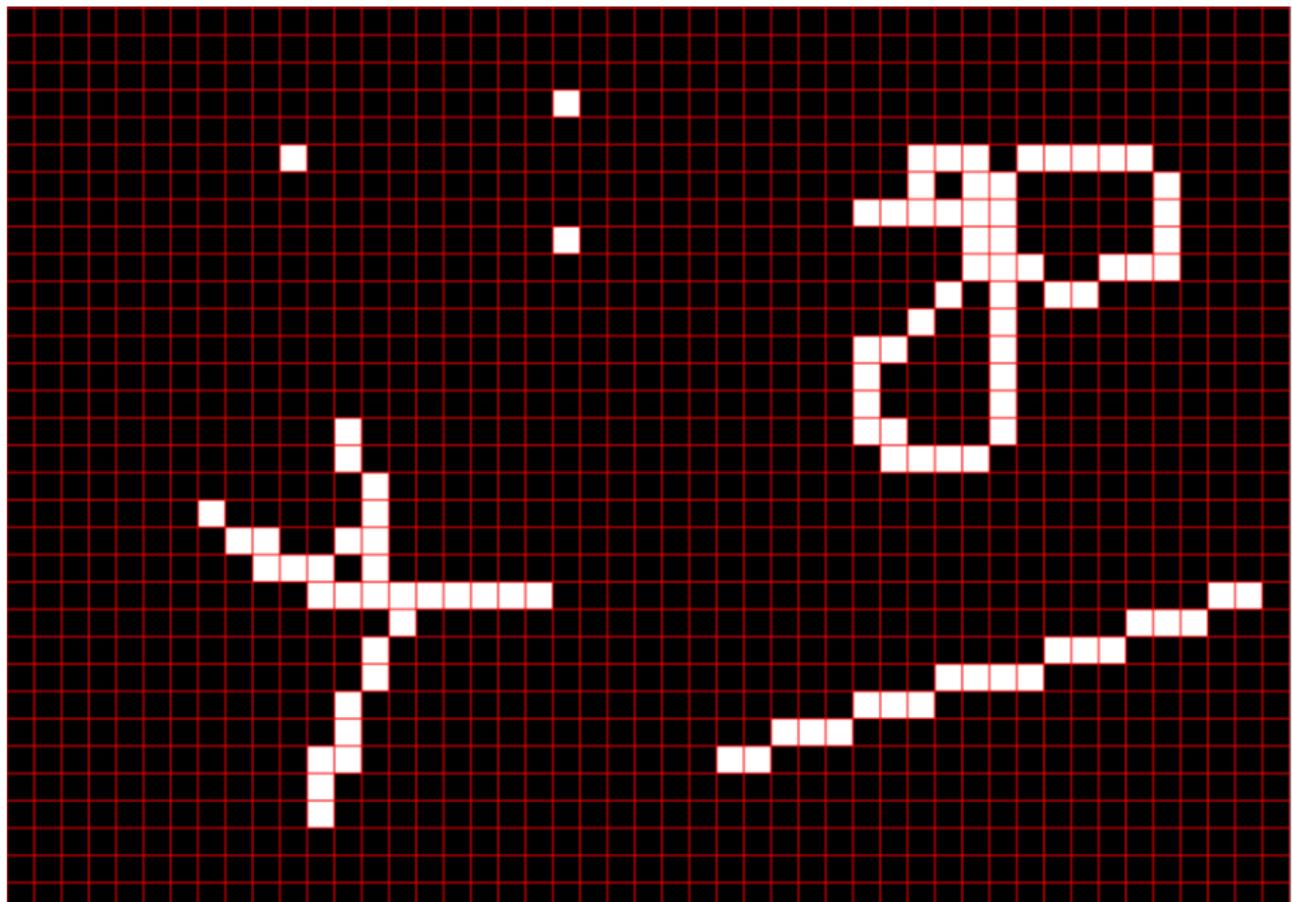
Squelette en exemple



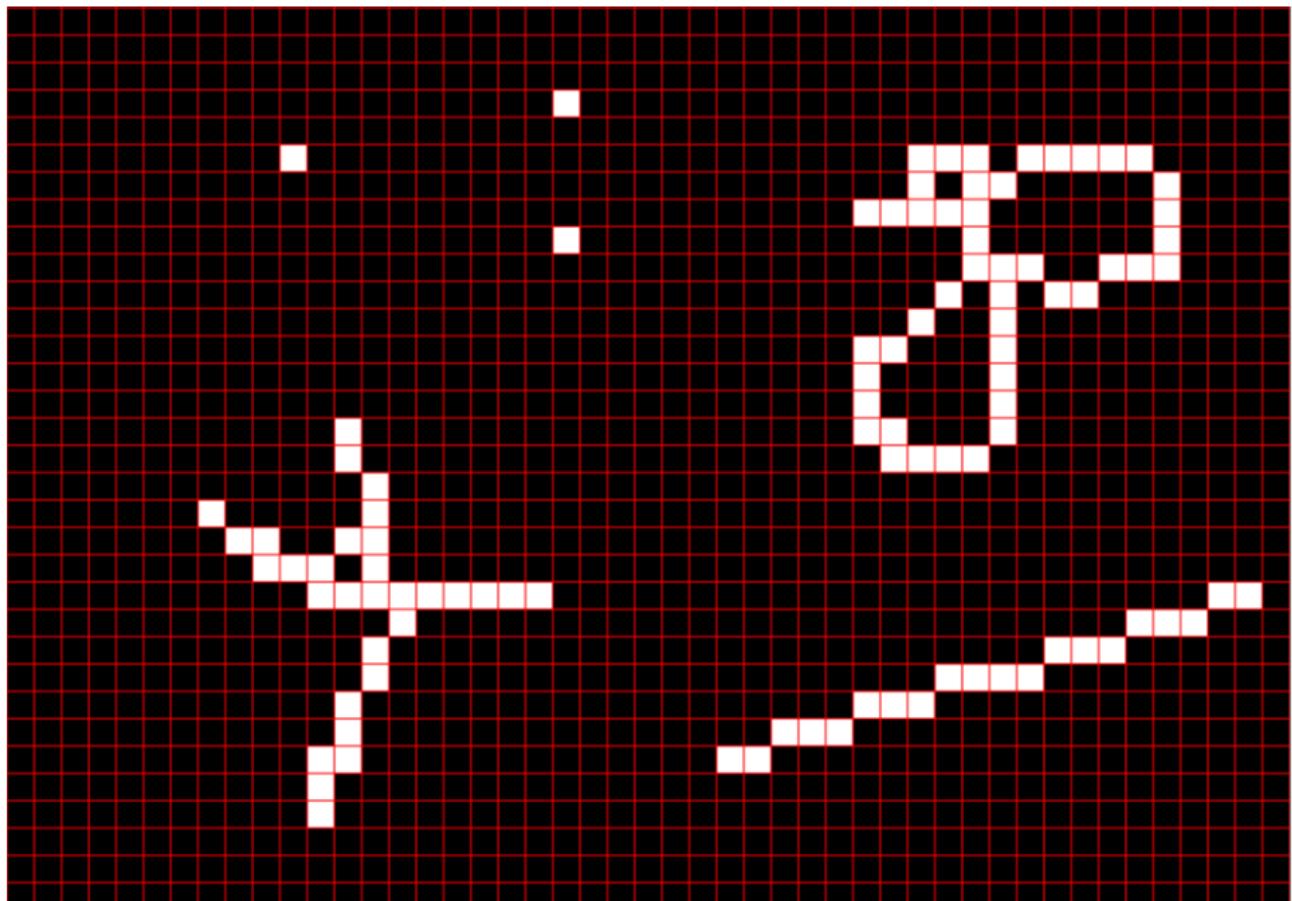
Squelette en exemple



Squelette en exemple



Squelette en exemple



Squelette de contour



lissage 0.5

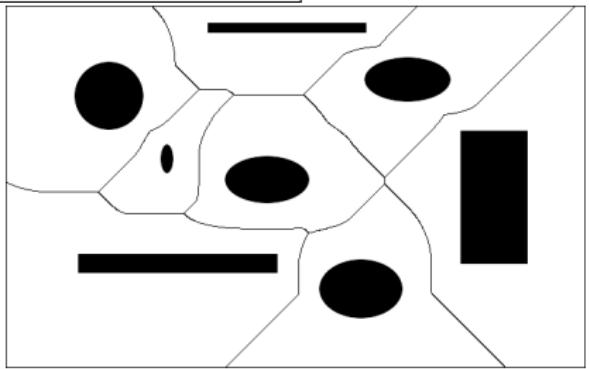
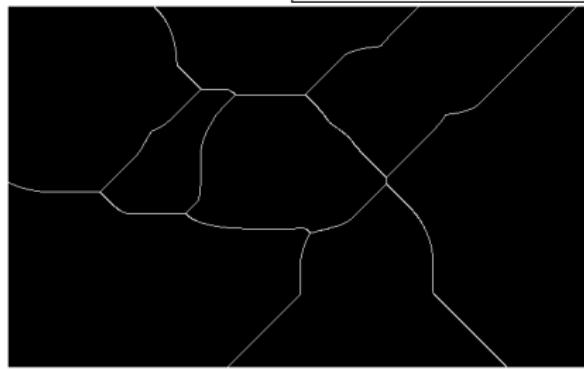
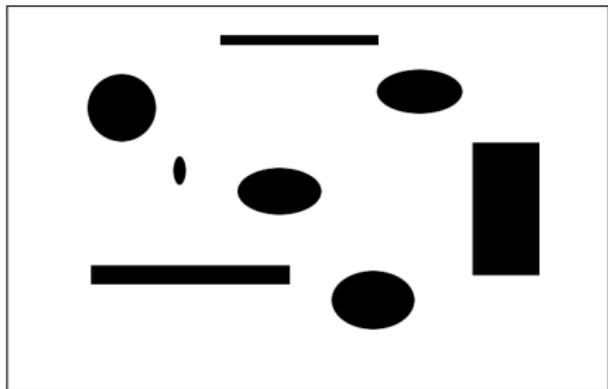


seuillage



squelette

Squelette sans obstacle



Squelette de squelette

