

# Cours Traitement d'Images & Vision par Ordinateur

- Master 2 RFIA -

Présenté par : Dr Mohamed Ben Ali Y.

# L'image et la Vision?

## **Vision par ordinateur** ***Computer Vision***

Émule la vision humaine dans le but  
d'extraire de l'information ou de  
prendre une décision

# Introduction

Par *traitement d'images*, on désigne l'ensemble des *opérations sur les images numériques*, qui transforment une image en une autre image, ou en une autre primitive formelle.

La *vision par ordinateur* désigne la *compréhension* d'une scène ou d'un phénomène à partir d'informations « image », liant intimement *perception, comportement et contrôle*.

Les domaines traités vont du *traitement du signal* à l'*intelligence artificielle*, on ne saurait donc prétendre à l'exhaustivité, mais on vise plutôt l'exploration d'un certain nombre de techniques importantes et actuelles.

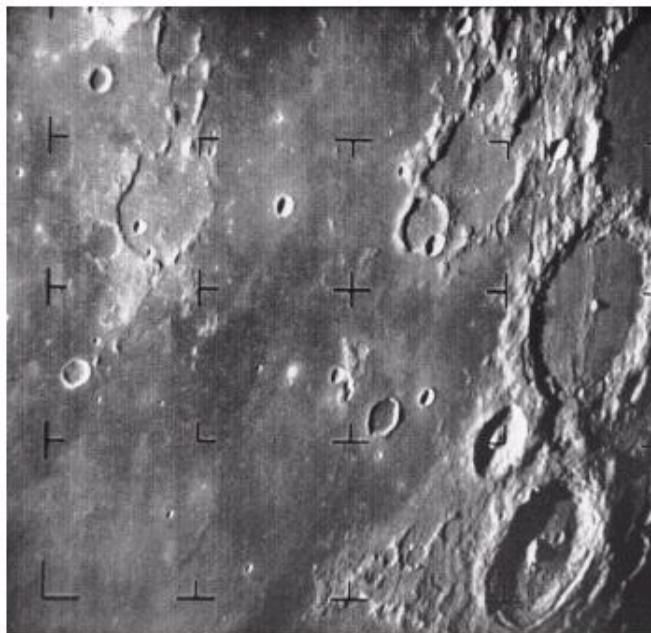
TI et vision sont des disciplines relativement jeunes (~années 60) et qui évoluent rapidement. Elles sont en plein expansion et donnent lieu chaque année à une profusion de travaux, académiques, technologiques, industriels.

Cette profusion s'explique par le caractère ardu du sujet : *complexité algorithmique* dû aux énormes volumes de données, caractère *mal posé* des problèmes et difficultés à formaliser une *faculté biologique « évidente »*.

D'autre part l'engouement pour ces disciplines s'explique par la multiplication permanente d'*applications* et d'*enjeux industriels* dans des domaines aussi variés que : médecine, télécommunications, automobile, météorologie, défense, jeux video, art, écologie...

# Un peu d'histoire

- Traitement d'image par ordinateur (1960)
  - Médecine, espace, astronomie



The first picture of the moon by a U.S. spacecraft. *Ranger 7* took this image on July 31, 1964 at 9:09 A.M. EDT, about 17 minutes before impacting the lunar surface. (Courtesy of NASA.)

# Un Peu d'Histoire ...

## Historique :

- 1950-1970 :

Images Rayons X

Amélioration de la qualité des images  
pour l'affichage

- 1970-1980 :

Extraction automatique d'informations  
seuillage, segmentation, extraction de contours  
morphologie mathématique

MAIS Problèmes de modélisation!

Puissance de calcul insuffisante

# Un Peu d'Histoire ...

## Historique :

- 1980- :

Image 2D  $\Rightarrow$  modèle 3D

Analyse du mouvement

Application : Robotique et  
guidage de véhicule

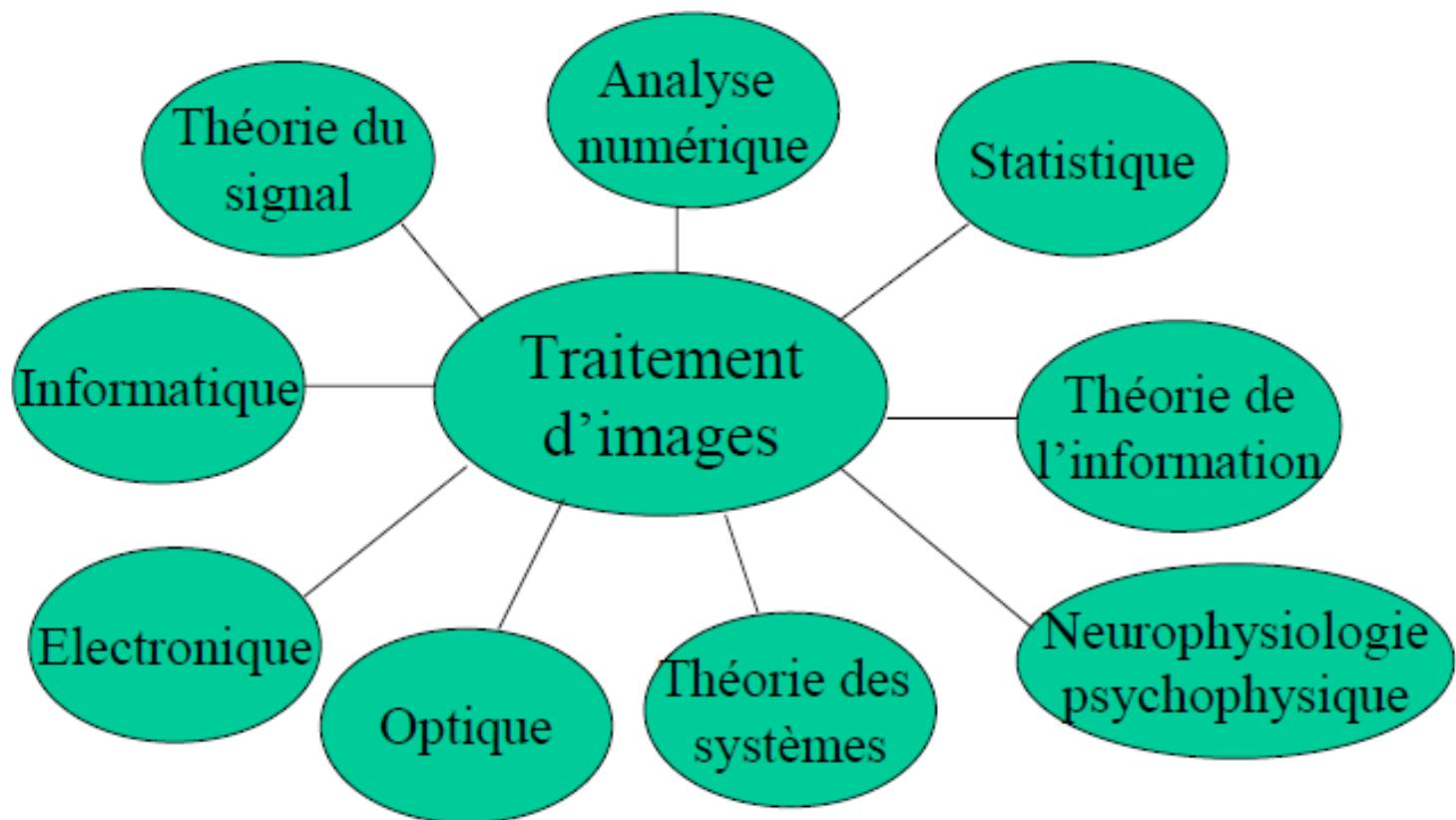
Vision active

- Perspectives :

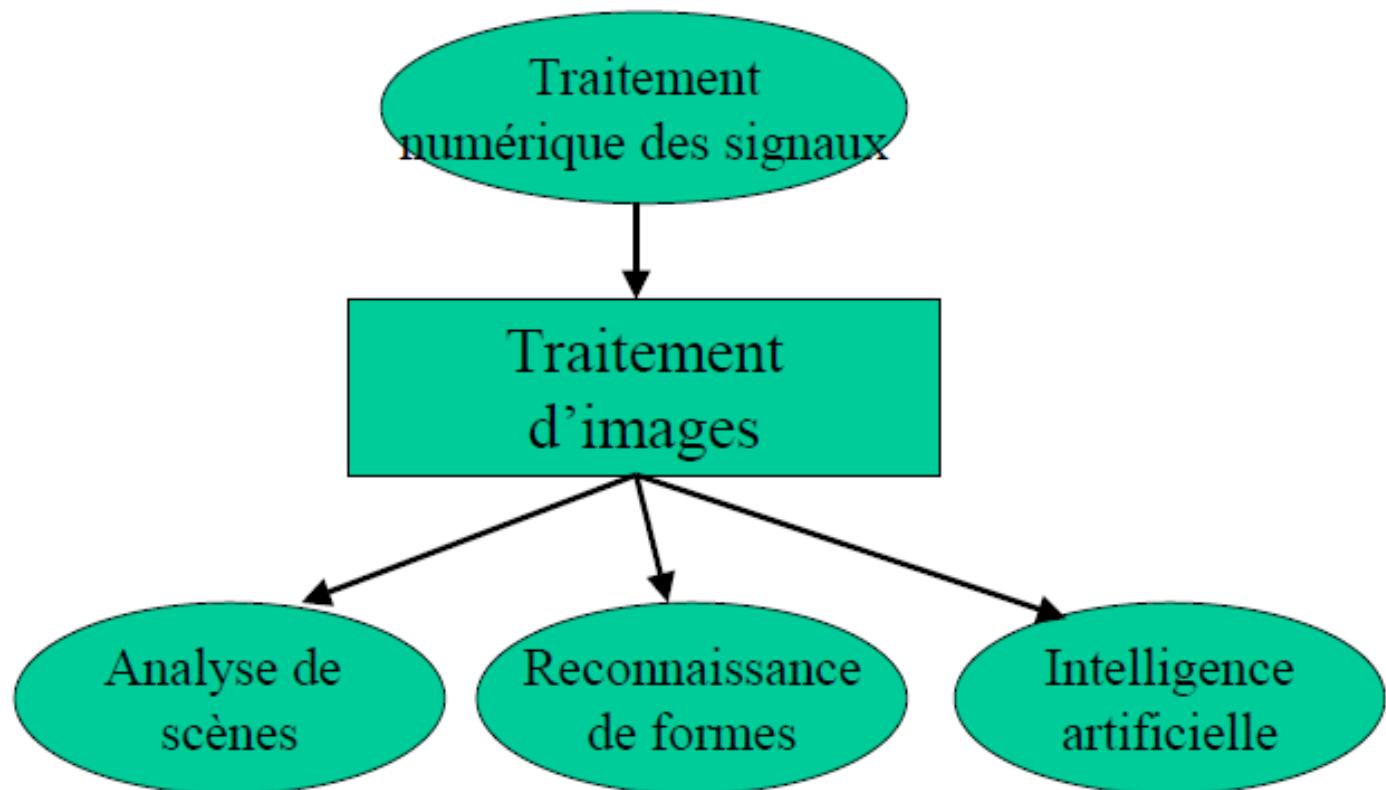
Acquisition

Analyse et interprétation de l'information  
image

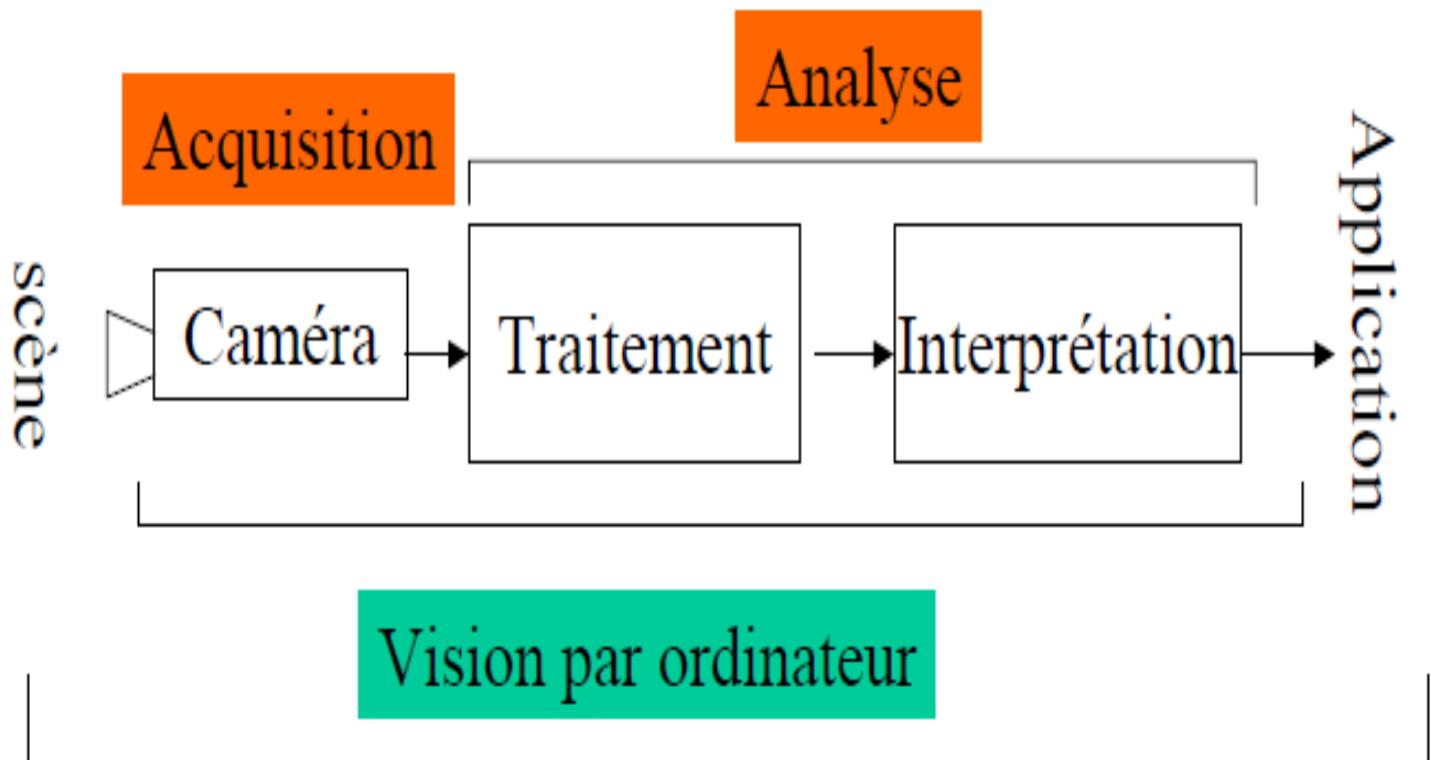
# Traitement d'images et autres disciplines



# Position du Traitement d'images



# Constitution d'un système de vision



# Les étapes de traitement numérique d'une image

- 3 étapes fondamentales :
  - **Acquisition** : scène physique  $\Rightarrow$  représentation numérique.
  - **Traitement** : Extraction de l'information pertinente par segmentation  $\Rightarrow$  description structurelle de l'image.
  - **Interprétation** : description structurelle  $\Rightarrow$  description sémantique.

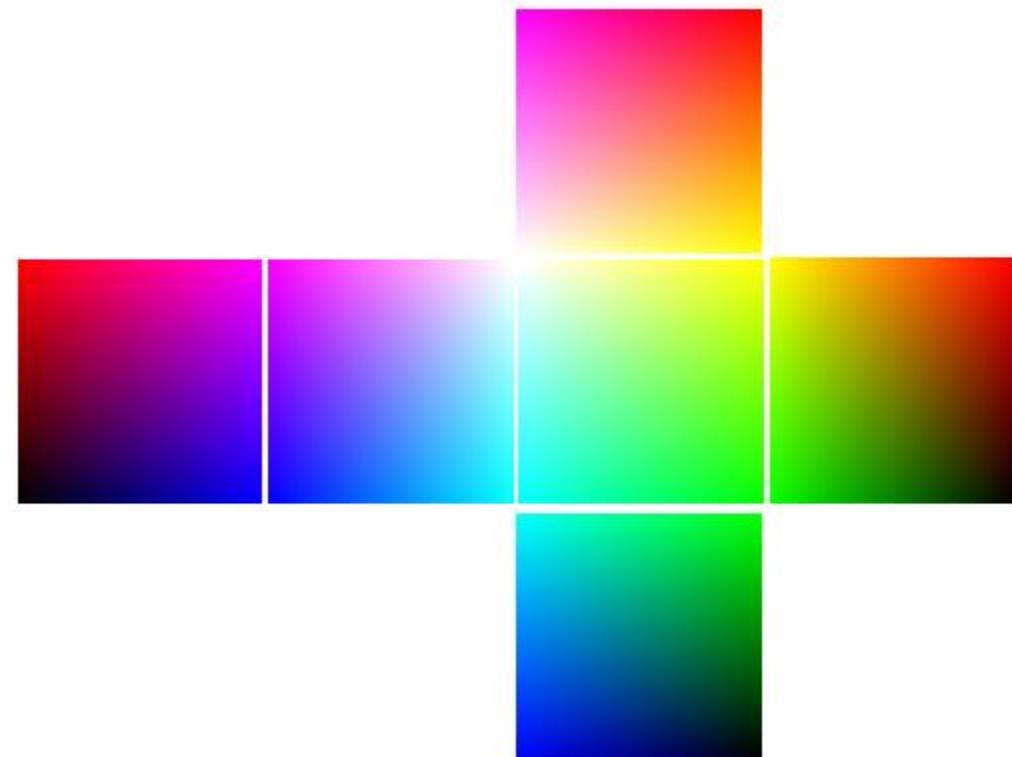
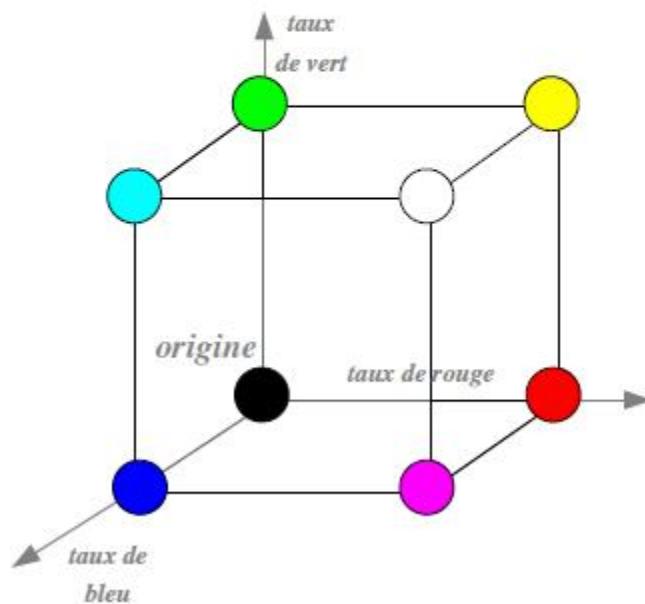
# Image Couleur

- ✓ Image couleur = 3 plans couleur.
- ✓ Pour la plupart des caméras : Rouge, Vert, Bleu (R,V,B).
- ✓ Chaque plan est codé comme une image niveaux de gris, avec des valeurs allant de 0 à 255.
- ✓ Lorsque R=V=B, la couleur associé est un niveau de gris.
- ✓ Pour passer d'une image couleur à une image niveau de gris, on réalise :

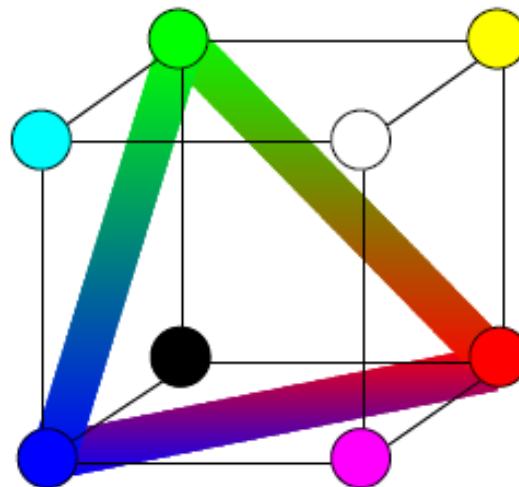
$$I(y,x) = \frac{R(y,x) + V(y,x) + B(y,x)}{3}$$

# L'espace RGB

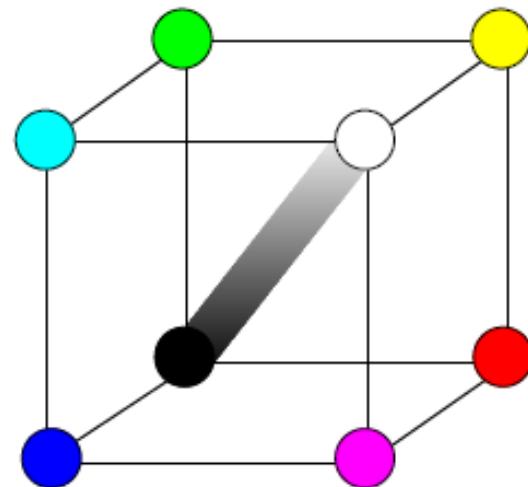
L'espace RGB est l'espace vectoriel engendré par les 3 composantes primaires (Rouge, Vert, Bleu). L'ensemble des couleurs produites se représente comme l'intérieur d'un cube :



# L'espace RGB



*Triangle chromatique*

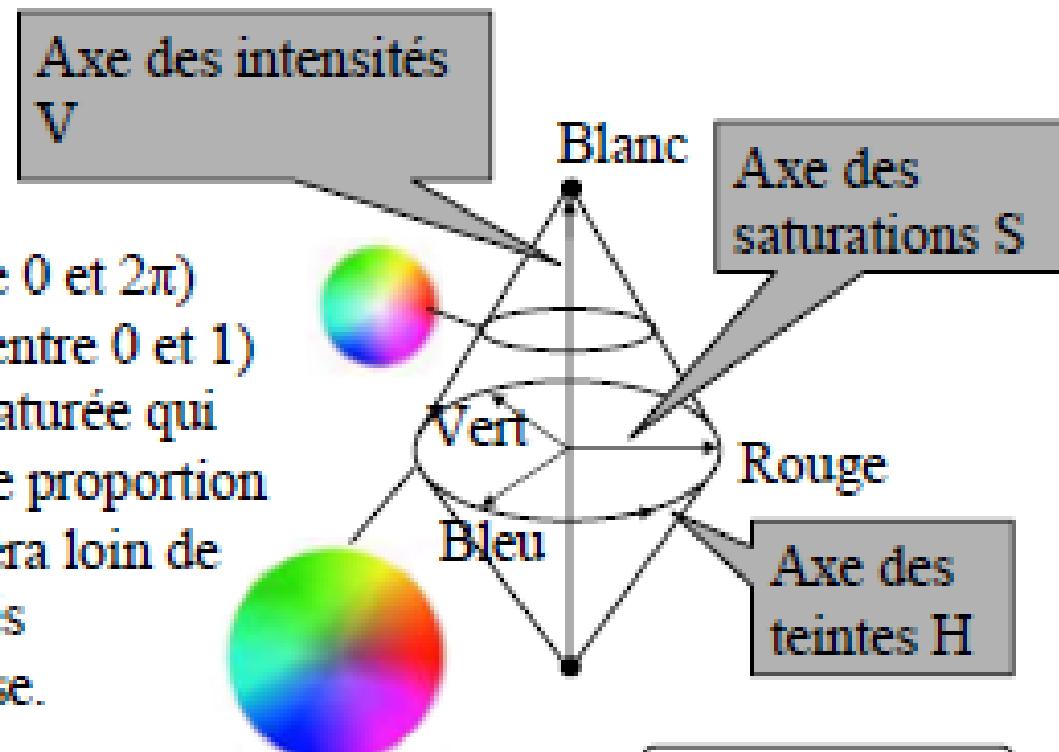


*Axe achromatique*

# Image Couleur

✓ De nombreux autres systèmes de représentation des couleurs existent parmi lesquels on peut citer le système HSV (Hue, Saturation, Value) :

H : teinte ( varie entre 0 et  $2\pi$ )  
S : saturation (varie entre 0 et 1)  
une couleur très saturée qui possède une faible proportion de blanc se trouvera loin de l'axe des intensités  
V : intensité lumineuse.



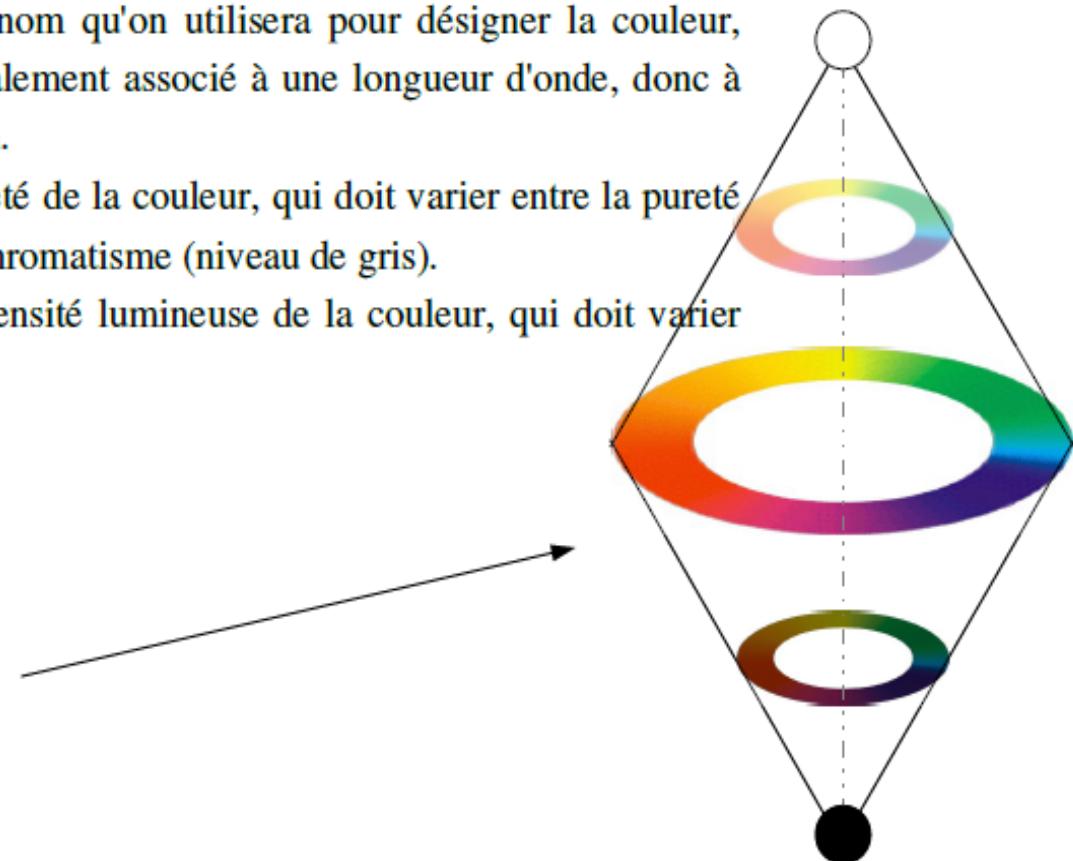
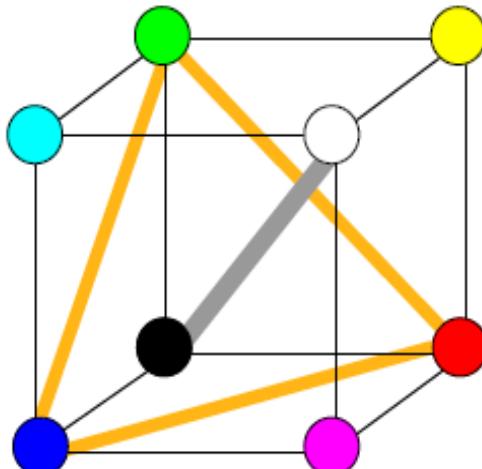
# L'espace HSV

Le principe de l'espace HSV est de caractériser les couleurs de façon plus intuitive, conformément à la perception naturelle des couleurs, en termes de :

**1 - teinte** : intuitivement, c'est le nom qu'on utilisera pour désigner la couleur, "vert", "mauve", "orange", etc. Idéalement associé à une longueur d'onde, donc à une position sur le cercle de Newton.

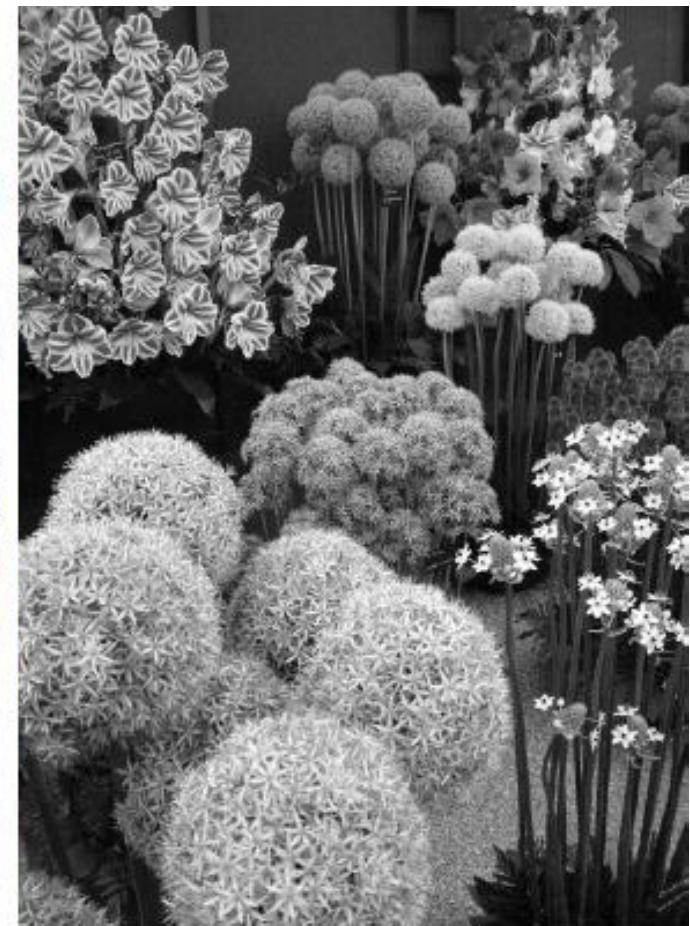
**2 - saturation** : c'est le taux de pureté de la couleur, qui doit varier entre la pureté maximale (couleur éclatante) et l'achromatisme (niveau de gris).

**3 - valeur** : c'est la mesure de l'intensité lumineuse de la couleur, qui doit varier entre le noir absolu et le blanc.





*Image couleur*



*Composante valeur*



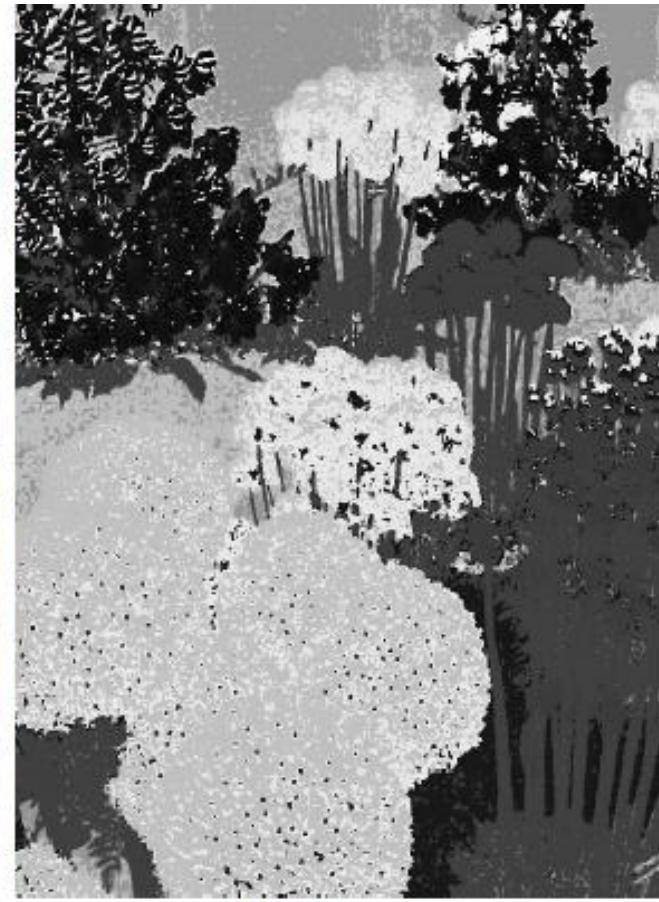
*Image couleur*



*Composante saturation*

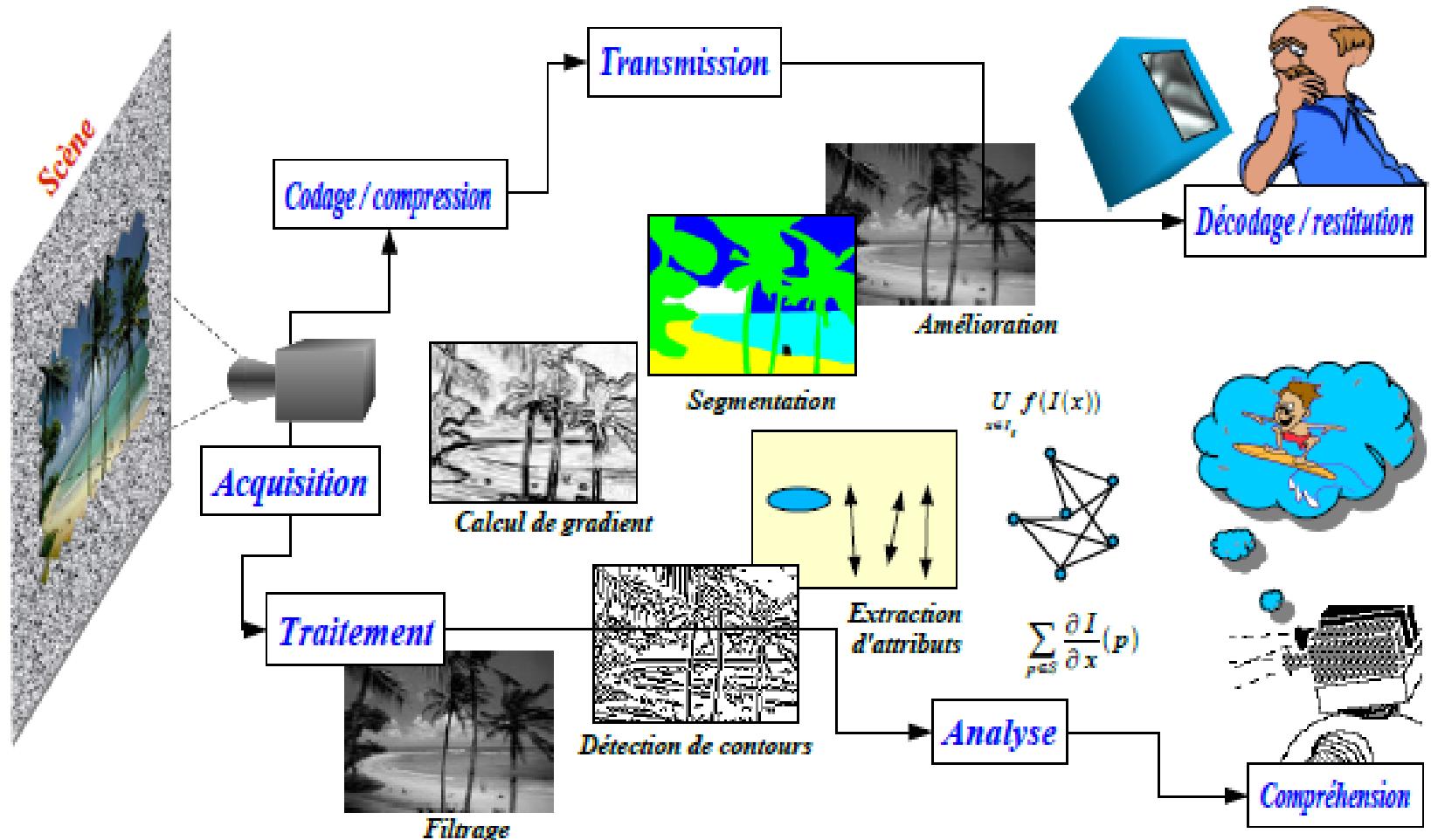


*Image couleur*



*Composante teinte*

# Système de Traitement d'images



# Plan de Travail

## *I] Introduction aux images numériques*

- *Vocabulaire*
- *Echantillonnage et quantification*
- *Outils fondamentaux*

## *II] Traitements à base d'histogramme*

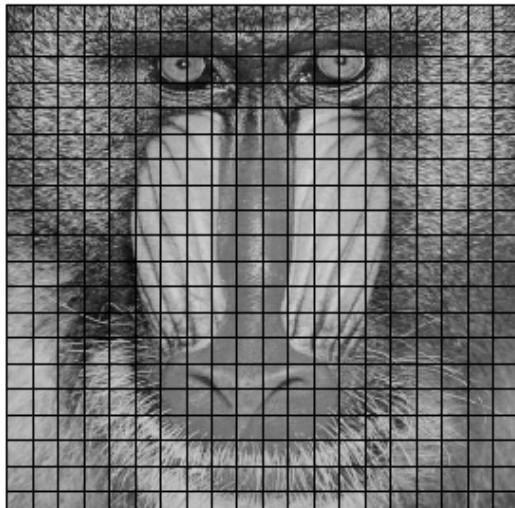
## *III] Filtres de lissage*

- *Filtrage dans le domaine de Fourier*
- *Filtrage par convolution*
- *Implantation des filtres linéaires*
- *Filtres non linéaires*

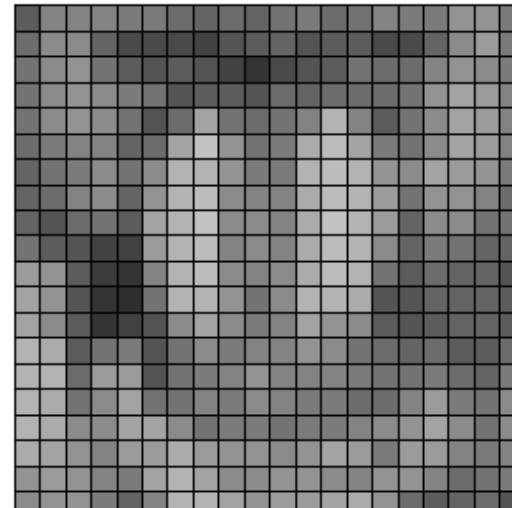
## *IV] Filtres dérivateurs*

- *Filtrage dans le domaine de Fourier*
- *Filtrage par convolution*

- Une image numérique est associée à un pavage de l'espace, en général rectangulaire. Chaque élément du pavage appelé *pixel* est désigné par ses coordonnées entières.
- L'échantillonnage est le procédé de discréétisation spatiale d'une image consistant à associer à chaque pixel une unique valeur.
- La quantification désigne la discréétisation tonale correspondant à la limitation du nombre de valeurs différentes que peut prendre chaque pixel.
- Une image numérique est donc une image échantillonnée et quantifiée.

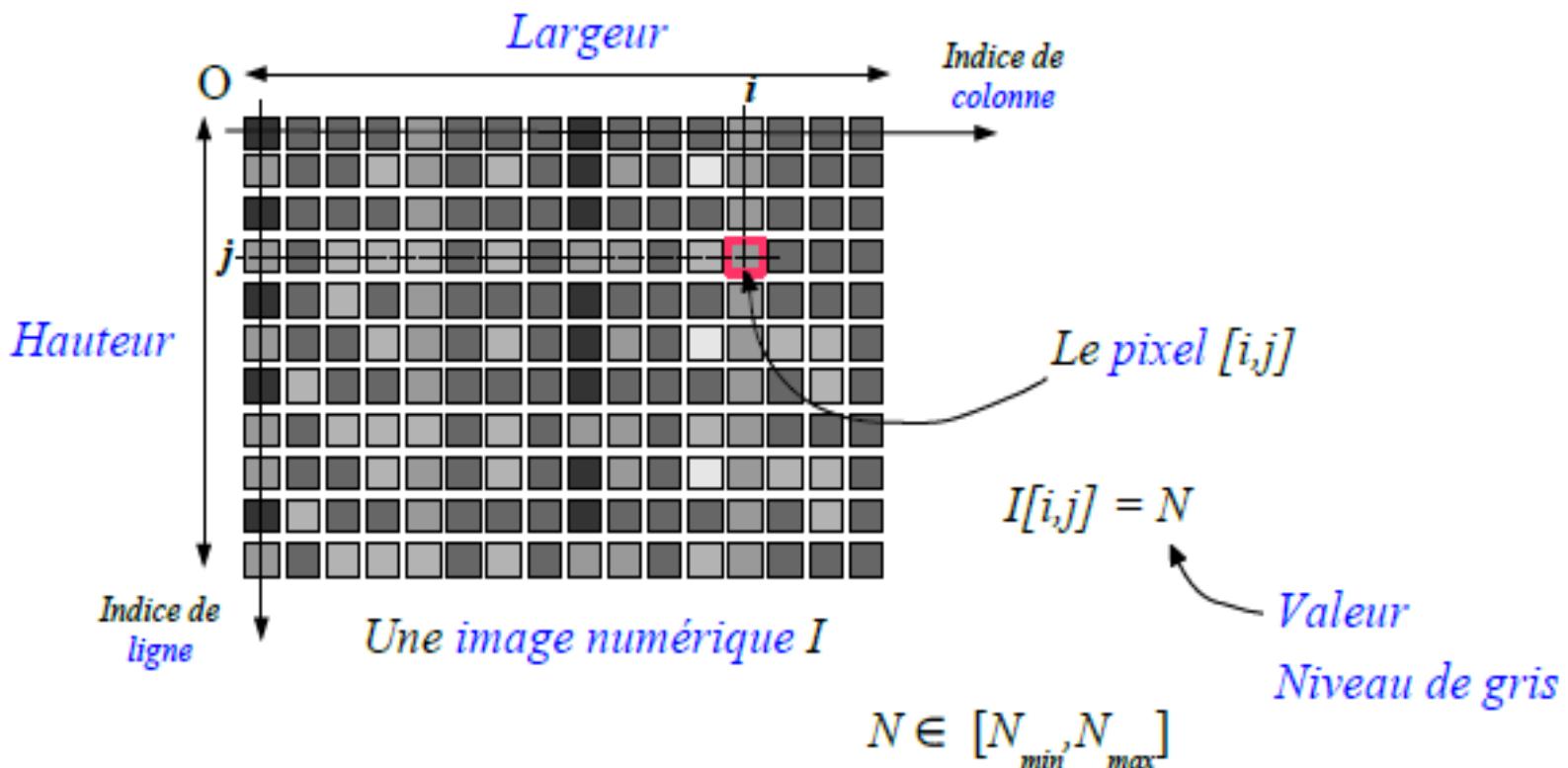


(1)



(2)

# Vocabulaire



$(N_{\max} - N_{\min})$  = nombre de niveaux de gris

$\log_2(N_{\max} - N_{\min})$  = dynamique

# Vocabulaire

*Résolution...*

*...spatiale :*



256x256



128x128



64x64



32x32

*...tonale :*

*Quantification*



6 bits



4 bits



3 bits



2 bits



1 bit

# Outils Fondamentaux

## a) L'histogramme

Outil de base pour l'étude des capteurs ou de la dynamique d'une scène, il est utilisé par certains opérateurs d'analyse. On retiendra cependant qu'il ne faut pas considérer l'histogramme comme une caractéristique fondamentale de l'image dans la mesure où on peut le transformer radicalement sans changer significativement l'image.

## b) La convolution

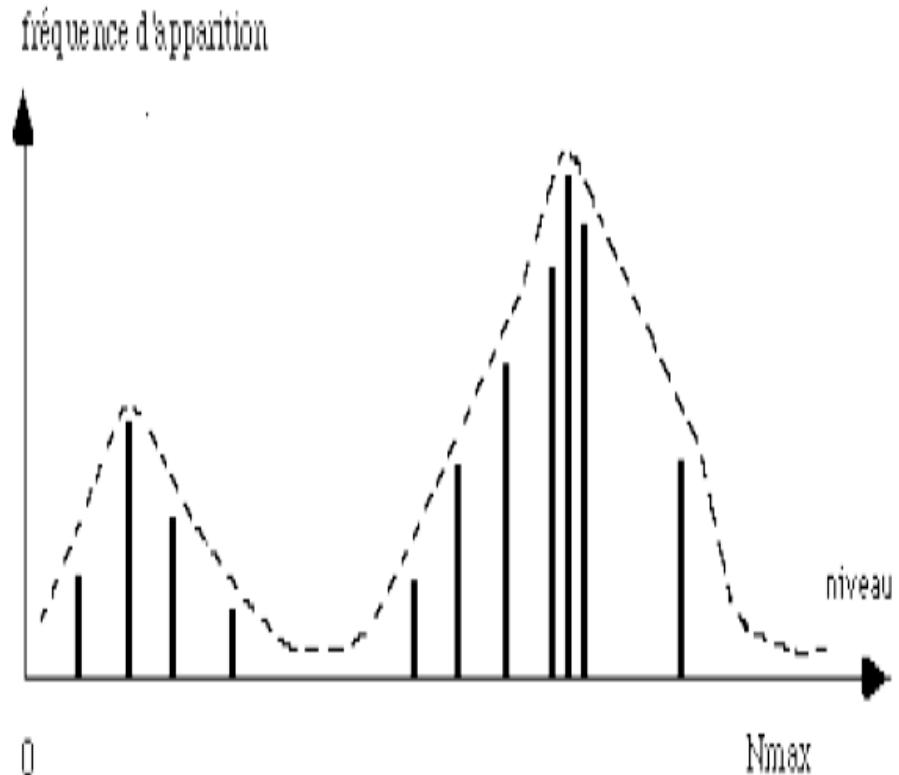
C'est l'opérateur de base du traitement linéaire des images. Apparue très tôt dans les premiers systèmes d'analyse d'images sous forme empirique et justifiée par des considérations d'implantation, ce n'est que plus tard qu'on a fourni des justifications physiques et fait le lien théorique avec les filtres et le traitement du signal.

## c) La transformée de Fourier

Outil fondamental d'analyse en traitement du signal, le pendant bidimensionnel de la TF et sa version discrète peut être appliquée avec profit aux images numériques. Si son utilisation en tant qu'outil analytique et algorithmique s'est estompée en traitement d'images au profit d'approches mieux adaptées à la localisation spatiale des fréquences (ondelettes), elle reste un outil théorique et pédagogique intéressant.

# Histogramme

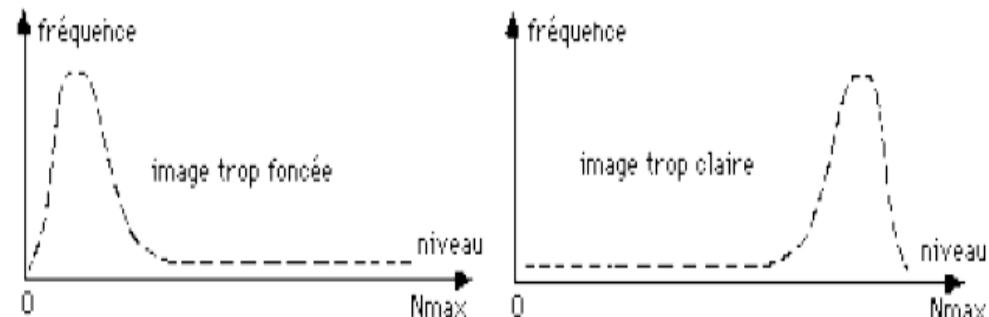
Soit une image comportant  $n$  lignes et  $p$  colonnes, donc  $n.p$  pixels. Chacun de ces pixels est codé sur  $q$  bits ( si  $q = 8$ , on a 256 niveaux). On peut effectuer une statistique sur les niveaux en comptant, pour chaque niveau, combien de pixels possèdent cette niveau. La représentation graphique de cette statistique est un histogramme par niveau .



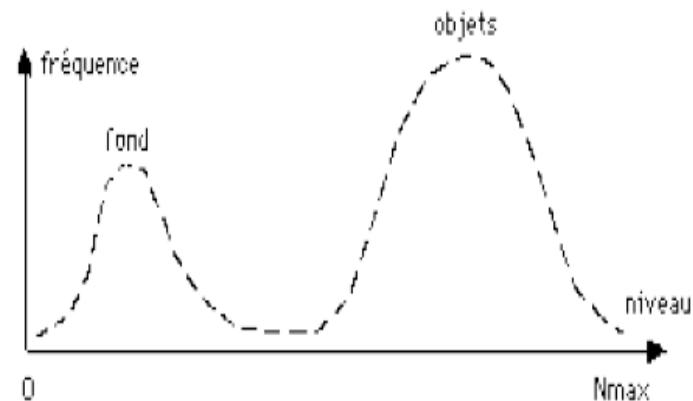
# Histogramme

L'histogramme permet d'obtenir des renseignements rapides sur une image.

On peut notamment faire la distinction entre une image trop foncée (niveaux en majorité près de 0) et une image trop claire (niveaux en majorité près de 255) comme indiqué ci-contre :

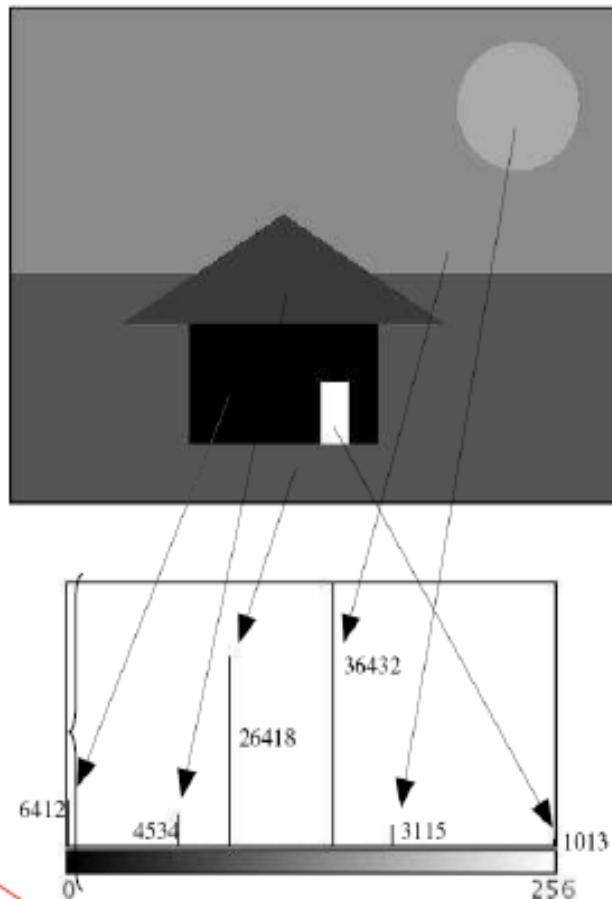


On peut aussi faire la différence entre le fond de l'image (arrière plan) et les objets intéressants de l'image (premier plan).



# Histogramme dans une image numérique

Une image de 322x242 pixels



Count: 77924  
Mean: 107.289  
Max: 255  
StdDev: 46.562  
Mode: 139 (36432)

Qq aspects quantitatifs de l'histogramme:

- 1 histogramme pour une gray image
- 3 histogrammes pour une color image
- On a autant de pics que d'objets
- La taille des pics est proportionnelle à la taille de l'objet concerné.
- Intervalle des valeurs qui prennent les niveaux de gris est appelé la dynamique des couleurs.

# Filtrage Linéaire

Parmi les filtres linéaires, 3 catégories de filtres sont envisagées qui se distinguent par des propriétés dans le domaine fréquentiel; nous verrons donc plus loin la raison des appellations utilisées :

- filtres "passe-bas" : ils sont utilisés pour atténuer les détails de l'image qui "tranchent" nettement avec le reste de l'image : bords, caractéristiques particulières notamment. Le résultat de l'application d'un filtre "passe-bas" sera une image "brouillée" ou "troublée".
- filtres "passe-haut" : contrairement aux précédents, ils sont utilisés pour atténuer les caractéristiques "neutres" et mettre en évidence les détails qui "tranchent".
- filtres "passe-bande" : ils ont peu d'intérêt pour le rehaussement d'image, mais par contre ils sont utilisés dans le cas de la restauration d'image.

Le principe du filtrage linéaire est de remplacer le niveau d'un pixel par une combinaison linéaire des niveaux des pixels environnants. Cette combinaison linéaire est usuellement représentée par un masque.

# Convolution

- La convolution discrète est un outil permettant la construction de **filtres linéaires** ou de **filtres de déplacements invariants**
- L'équation de convolution, notée  $g(x)$ , de la séquence  $f(x)$  avec une fonction  $h(x)$  est :

$$g(x) = f(x) * h(x) = \sum_{all k} h(x - k)f(k)$$

- $h(x)$  est appelée *masque de convolution, noyau de convolution, filtre, fenêtre, kernel, ...*
- En pratique, la convolution numérique d'une image se fera simplement par une **sommation de multiplications**.

# Masque de Convolution

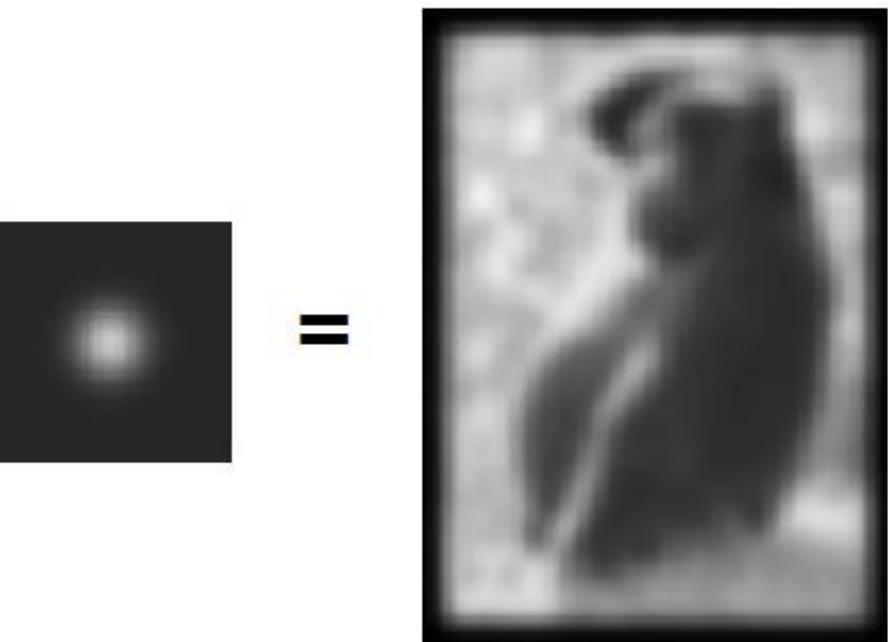
- Le masque de convolution est le plus souvent
  - Carré
  - De taille 3x3 ou 5x5 (ou plus, mais impair)
- Ce masque représente un filtre linéaire permettant de modifier l'image.
- La plupart du temps, on divisera le résultat de la convolution par la somme des coefficients du masque.
  - Pour éviter de modifier l'entropie de l'image, la somme des coefficients doit être égale à 1.
  - Dans le cas du Laplacien (plus loin), la somme sera égale à zéro

# Exemple de Convolution 2D



Image d'origine

$$\ast \quad \begin{matrix} \text{Filtre de convolution} \\ (\text{masque}) \end{matrix} \quad = \quad \begin{matrix} \text{Image convoluée} \\ (\text{résultat}) \end{matrix}$$



*Note : par convention pratique, la taille de l'image résultat est la même que celle de l'image d'origine*

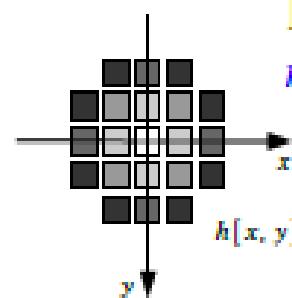
# Convolution

Soit  $I$  une image numérique.

Soit  $h$  une fonction de  $[x_1, x_2] \times [y_1, y_2]$  à valeurs réelles.

La *convolution* de  $I$  par  $h$  est définie par :

$$(I * h)[x, y] = \sum_{i=x_1}^{x_2} \sum_{j=y_1}^{y_2} h[i, j] \cdot I[x-i, y-j]$$



La fonction  $h$  est dite *noyau de convolution*

Propriétés de la convolution :

**COMMUTATIVITÉ**  $h * g = g * h$

**ASSOCIATIVITÉ**  $(h * g) * k = h * (g * k) = h * g * k$

**DISTRIBUTIVITÉ/+**  $h * (g + k) = (h * g) + (h * k)$

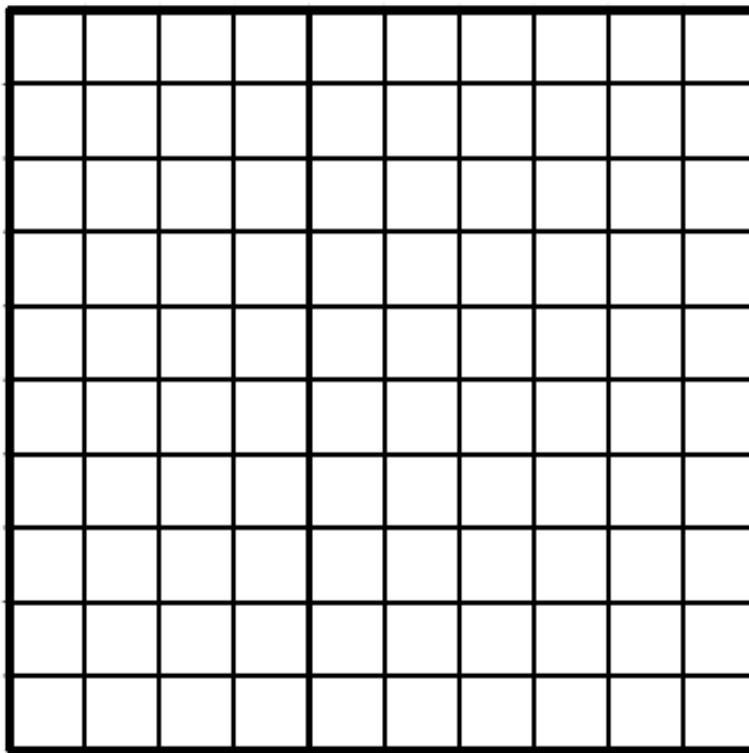
Les nouvelles valeurs du pixel sont calculées par *produit scalaire* entre le noyau de convolution et le *voisinage* correspondant du pixel.

# Convolution de Fonctions Discrètes

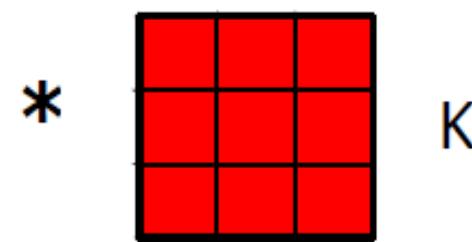
Elément neutre de la convolution

0	0	0
0	1	0
0	0	0

# Convolution numérique

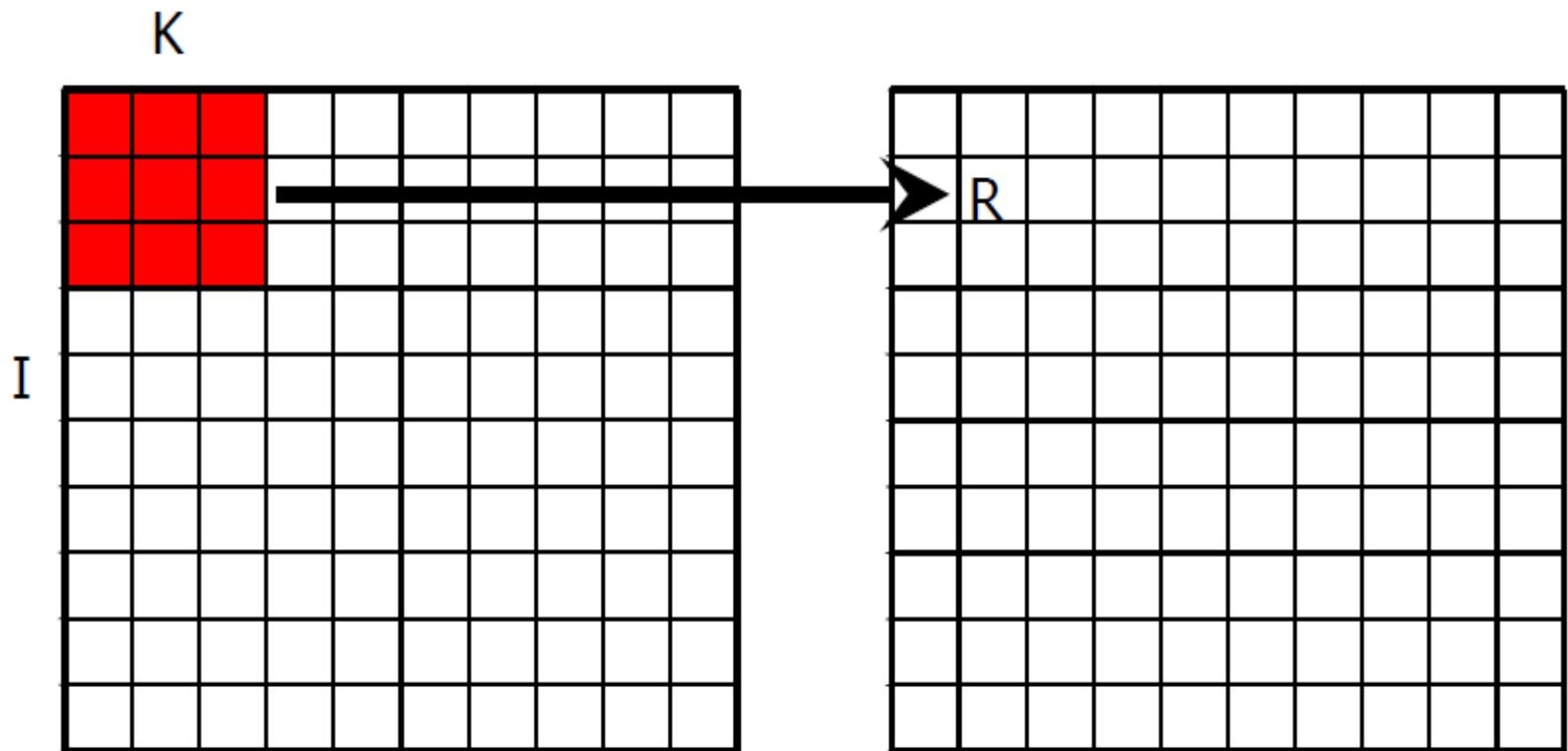


Image



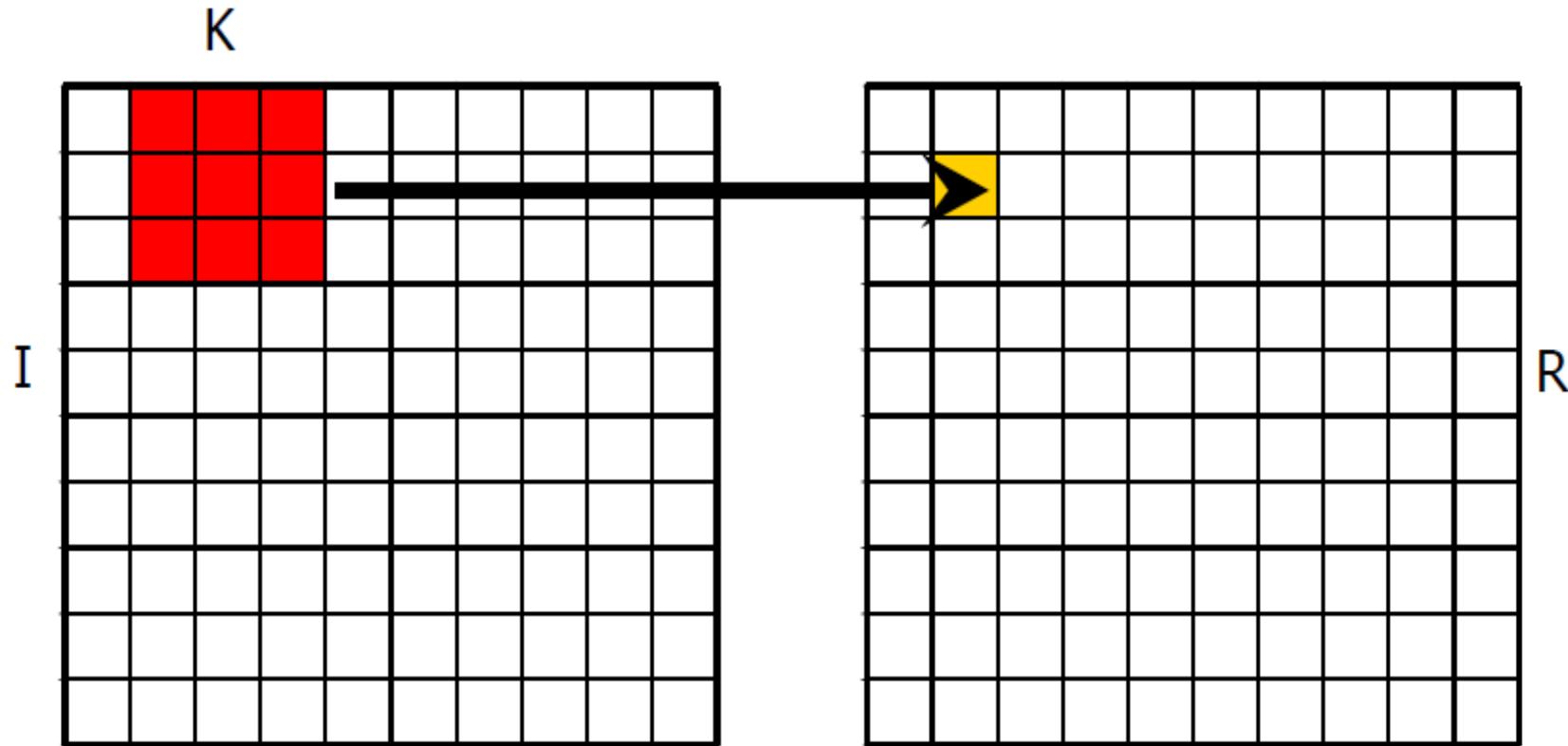
Noyau de convolution

# Convolution numérique



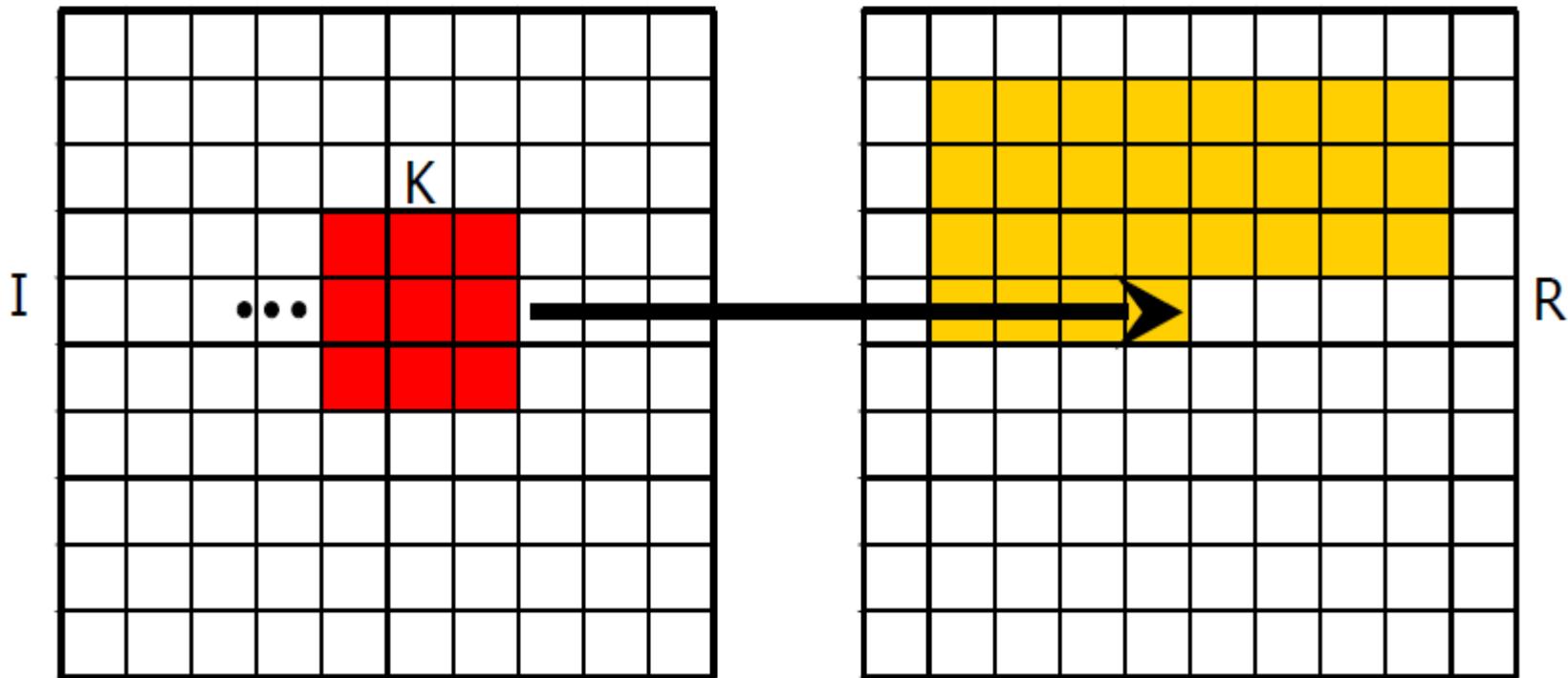
$$\begin{aligned} R(1,1) = & I(0,0) K(0,0) + I(1,0) K(1,0) + I(2,0) K(2,0) \\ & + I(0,1) K(0,1) + I(1,1) K(1,1) + I(2,1) K(2,1) \\ & + I(0,2) K(0,2) + I(1,2) K(1,2) + I(2,2) K(2,2) \end{aligned}$$

# Convolution numérique $R=I*K$



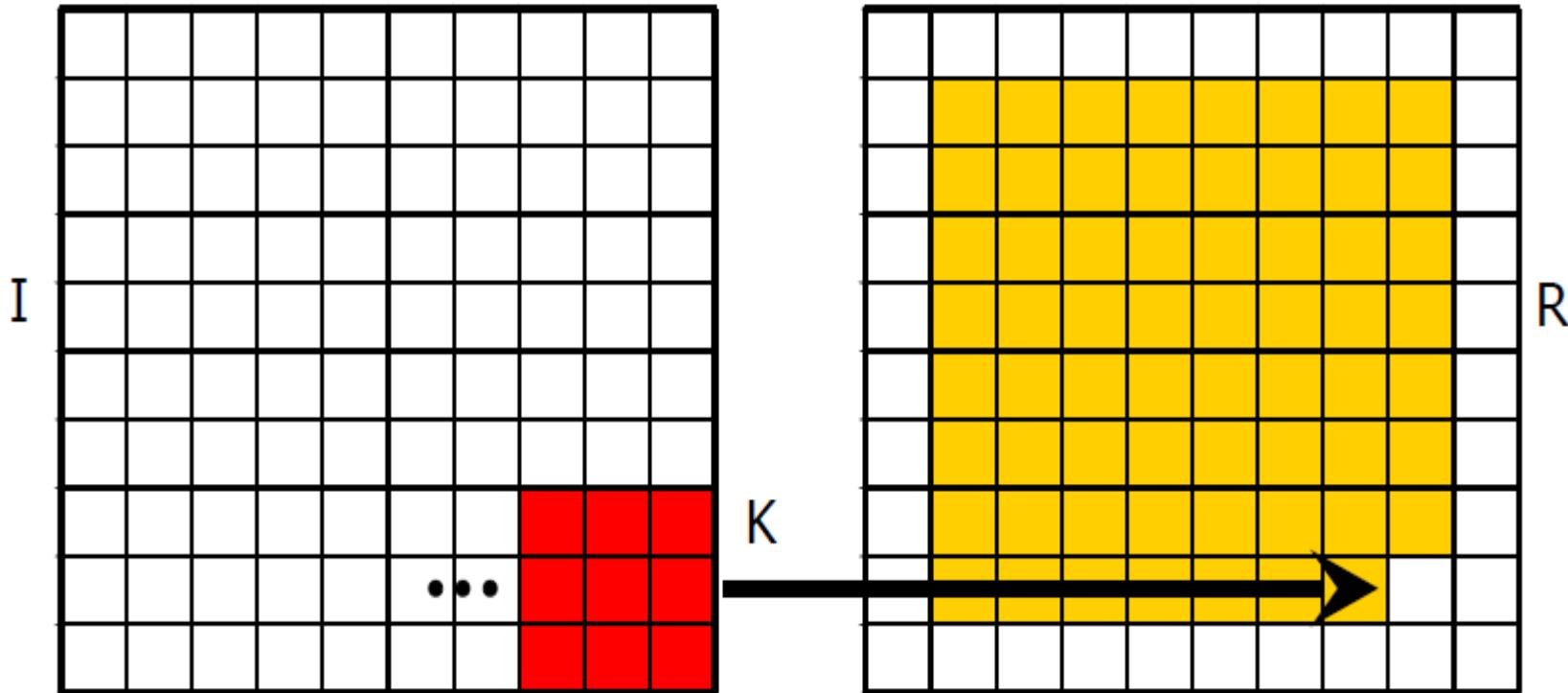
$$\begin{aligned} R(2,1) = & I(1,0) K(0,0) + I(2,0) K(1,0) + I(3,0) K(2,0) \\ & + I(1,1) K(0,1) + I(2,1) K(1,1) + I(3,1) K(2,1) \\ & + I(1,2) K(0,2) + I(2,2) K(1,2) + I(3,2) K(2,2) \end{aligned}$$

# Convolution numérique $R=I*K$



$$\begin{aligned} R(x,y) = & I(x-1,y-1) K(0,0) + I(x, y-1) K(1,0) + I(x+1, y-1) K(2,0) \\ & + I(x-1,y) K(0,1) + I(x,y) K(1,1) + I(x+1,y) K(2,1) \\ & + I(x-1,y+1) K(0,2) + I(x,y+1) K(1,2) + I(x+1,y+1) K(2,2) \end{aligned}$$

# Convolution numérique $R=I*K$



$$\begin{aligned} R(N-2, M-2) = & I(N-3, M-3) K(0,0) + I(N-2, M-3) K(0,1) + I(N-1, M-3) K(0,3) \\ & + I(N-3, M-2) K(1,0) + I(N-2, M-2) K(1,1) + I(N-1, M-2) K(1,2) \\ & + I(N-3, M-1) K(2,0) + I(N-2, M-1) K(2,1) + I(N-1, M-1) K(2,2) \end{aligned}$$

# Convolution numérique

- Problème : Que faire avec les bords de l'image ?
  - Mettre à zéro (0)
  - Convolution partielle
    - Sur une portion du noyau
  - ... (pas de solution miracle)

?	?	?	?	?	?	?	?	?	?
?									?
?									?
?									?
?									?
?									?
?									?
?									?
?	?	?	?	?	?	?	?	?	?

# Filtrage Spatiale : 2 Types

- **Filtres passe-bas**

- Atténue le bruit et les détails (basses fréquences)  
→ lissage



- **Filtres passe-haut**

- Accentue les détails et les contours (hautes fréquences)  
→ accentuation

# Filtre Moyenneur

- Le filtre moyenneur
  - Permet de lisser l'image (*smoothing*)
  - Remplace chaque pixel par la valeur moyenne de ses voisins
  - Réduit le bruit
  - Réduit les détails non-important
  - Brouille ou rend floue l'image (*blur edges*)
- Filtre dont tous les coefficients sont égaux
- Exemple de filtres moyenneurs :

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

ou

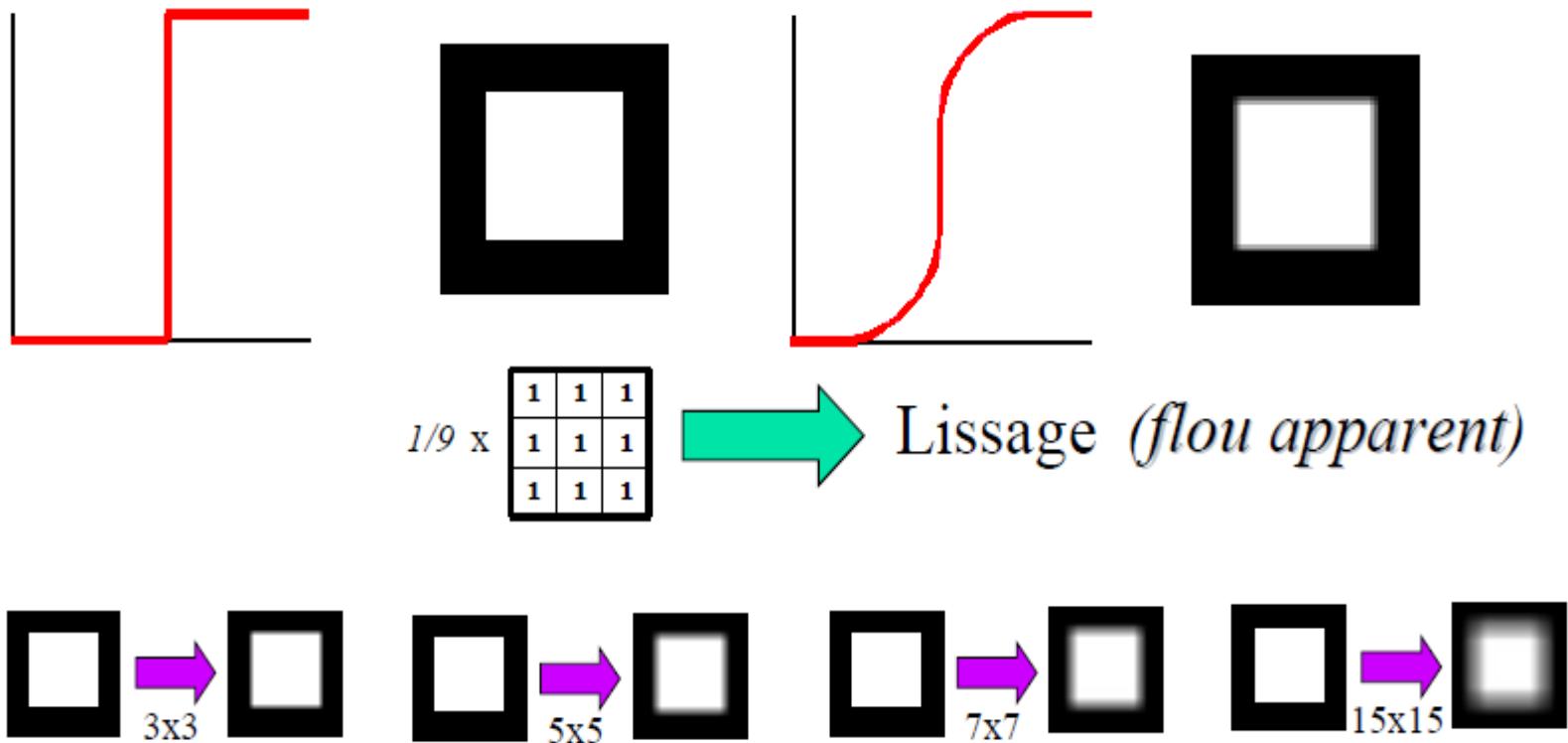
1/9	1	1	1
1	1	1	1
1	1	1	1

3x3

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

5x5

# Filtre Moyenneur



Plus le filtre grossit , plus le lissage devient important et plus le flou s'accentue !

# Exemples de Filtres Moyenneurs



Original



Moyenne 5x5



Moyenne 11x11

# Filtres Linéaires

Les coefficients de tous ces filtres sont positifs; ils contribuent à "adoucir" l'image en remplaçant le niveau de chaque pixel par une moyenne arithmétique pondérée des niveaux des pixels voisins. Il s'agit de filtres "passe-bas". On peut imaginer des filtres linéaires dont les masques présentent des coefficients positifs au centre et des coefficients négatifs pour les pixels voisins. Il s'agit alors de filtres "durcissants" qui mettront en valeur des détails qui "tranchent", donc de filtres "passe-haut". Ainsi sera le

---

filtre dont le masque est défini ci-dessous :

$$h_4 = \begin{vmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{vmatrix}$$

Pour "durcir" l'image, on peut aussi utiliser des méthodes plus élaborées; par exemple, on peut d'abord utiliser un filtre "passe-bas" ce qui conduit à une image  $g_b(x,y)$ , puis effectuer la soustraction de l'image  $g_b(x,y)$  de l'image originale  $I(x,y)$ .

# Filtre Gaussien

Dans le filtrage linéaire classique, on a pu remarquer que les éléments du masque sont des poids statistiques. On prendra ici comme poids statistiques la valeur de la fonction de Gauss "bi-dimensionnelle" :

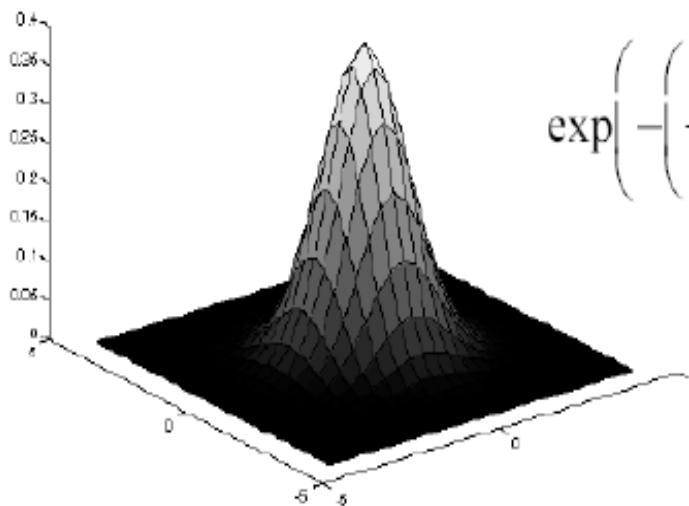
$$G(x,y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

mais en limitant  $x$  et  $y$  à la plage  $[-3\sigma, +3\sigma]$  puisque, pratiquement, les valeurs significatives sont dans cette plage. Si on prend une taille  $2n+1=5$  ( $\sigma=5/6$ ), le masque est donc :

$$G = \begin{pmatrix} 0,0007 & 0,0063 & 0,0129 & 0,0063 & 0,0007 \\ 0,0063 & 0,0543 & 0,1116 & 0,0543 & 0,0063 \\ 0,0129 & 0,1116 & 0,2292 & 0,1116 & 0,0129 \\ 0,0063 & 0,0543 & 0,1116 & 0,0543 & 0,0063 \\ 0,0007 & 0,0063 & 0,0129 & 0,0063 & 0,0007 \end{pmatrix}$$

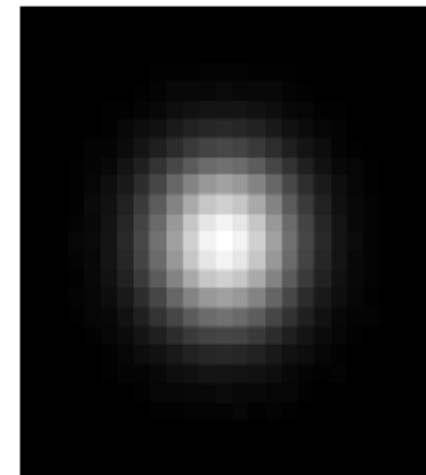
( la somme de tous les coefficients est très proche de 1).

# Filtre Gaussien



*Fonction gaussienne en 3D*

$$\exp\left(-\left(\frac{x^2 + y^2}{2\sigma^2}\right)\right)$$



*Image d'une gaussienne*

*Le filtre gaussien donnera un meilleure lissage et une meilleure réduction du bruit que le filtre moyenne*

$$\frac{1}{98} \times \begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 6 & 8 & 6 & 2 \\ 3 & 8 & 10 & 8 & 3 \\ 2 & 6 & 8 & 6 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$$

# Filtre Gaussien



original



$\sigma = 0.6$



$\sigma = 1$



$\sigma = 3$



$\sigma = 5$



$\sigma = 10$

# Exemples de Filtres Gaussiens



Original



Gauss 5x5

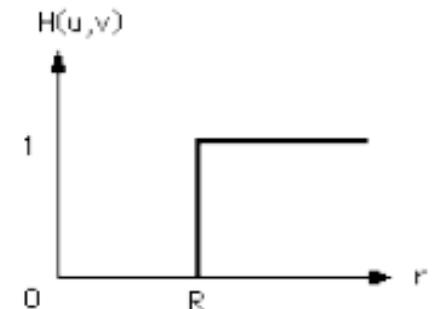


Gauss 11x11

# Filtre Passe-haut

Les filtres "passe-haut" atténuent les basses fréquences. Le filtre idéal est défini par :

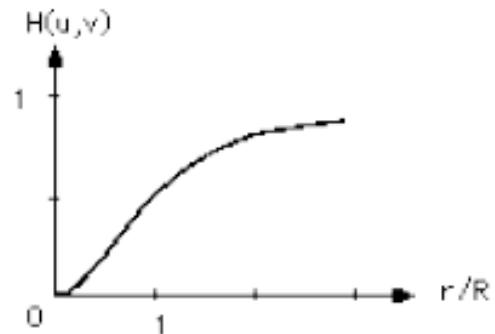
$$\begin{aligned} H(u,v) &= 0 \text{ si } r < R \\ H(u,v) &= 1 \text{ si } r > R \\ \text{avec } r^2 &= u^2 + v^2 \end{aligned}$$



Le filtre de Butterworth est moins brutal :

$$H(u,v) = \frac{1}{1 + \alpha \left(\frac{R}{r}\right)^{2n}}$$

$$\text{avec } \alpha = 1 \quad \text{ou} \quad \alpha = \sqrt{2} - 1$$



# Rehaussement de Contraste

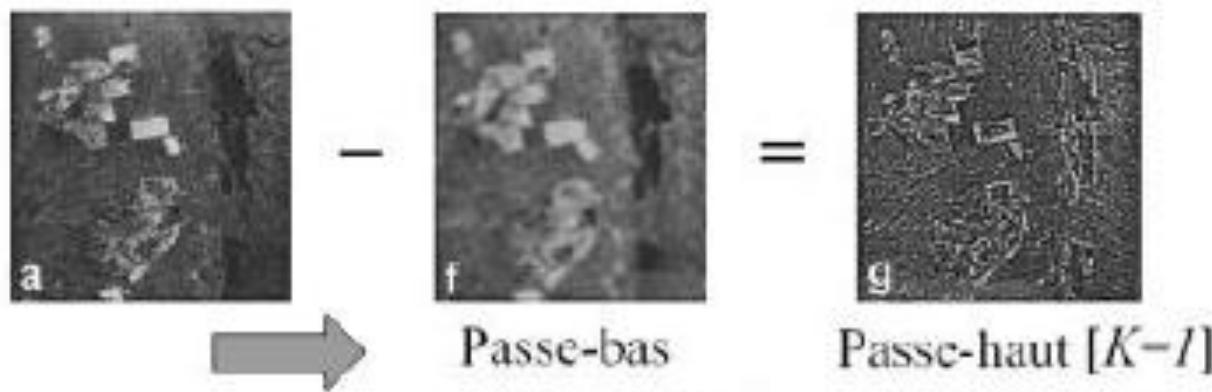
## Filtre passe-haut



Principe: DIFFERENCE

- ❖ Laplacien
- ❖ DéTECTEURS de contours
  - ♦ Sobel
  - ♦ Prewitt
  - ♦ ....

# Filtre passe-haut : principe



$$\frac{1}{49} X$$

1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

$$\frac{1}{49} X$$

-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	48	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1

# Filtre Laplacien

- ❖ Exemples Laplacien (filtre passe haut):

0	1	0
1	-4	1
0	1	0

...

1	1	1
1	-8	1
1	1	1



# Rehaussement

- Exemples Laplacien (filtre passe haut):

$$\begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 4 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 0 & -1 & 0 \\ \hline -1 & 8 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

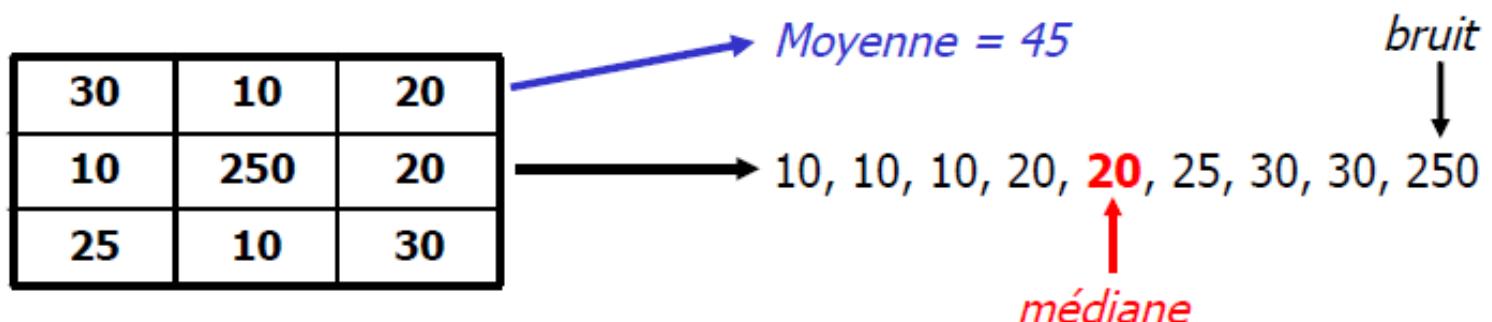


# Filtres Non-Linéaires

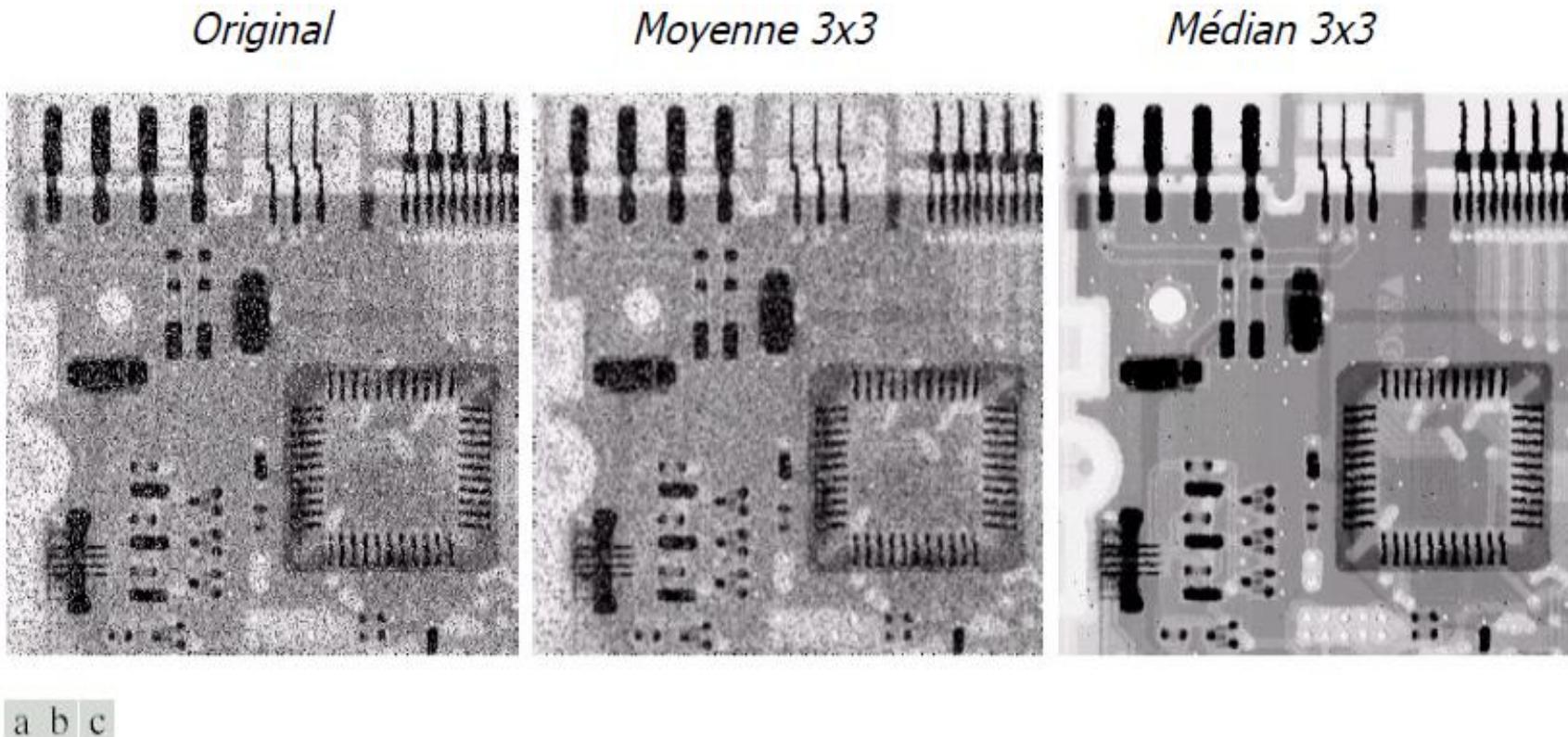
## (Autres que Convolution)

# Filtre Médian

- Pour nettoyer le bruit dans une image, il existe mieux que le **filtre moyenneur** ou le **filtre gaussien**
- Il s'agit du **filtre médian**
- C'est un filtre non-linéaire, qui ne peut pas s'implémenter comme un produit de convolution
- On remplace la valeur d'un pixel par la valeur médiane dans son voisinage NxN

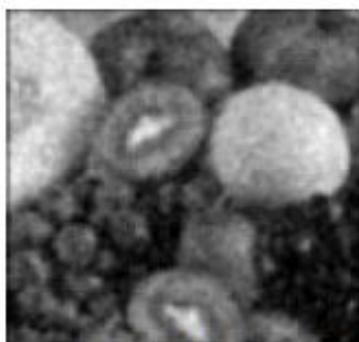


# Exemples

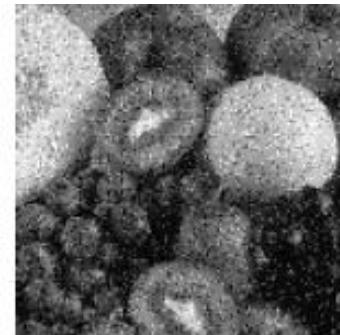


**FIGURE 3.37** (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a  $3 \times 3$  averaging mask. (c) Noise reduction with a  $3 \times 3$  median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

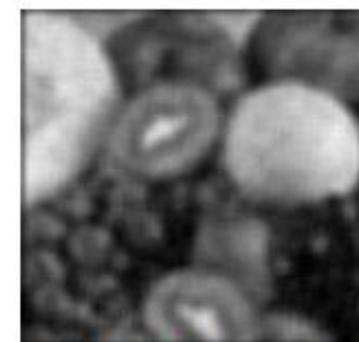
# Nettoyage de bruit dans une Image



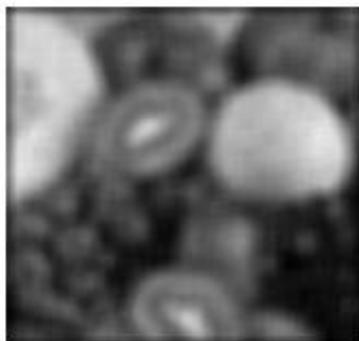
3 X 3 Moyenne



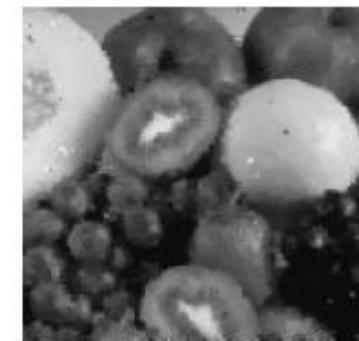
Bruit "poivre et sel"



5 X 5 Moyenne



7 X 7 Moyenne



Filtre médian



Image initiale



Bruit Poivre & Sel

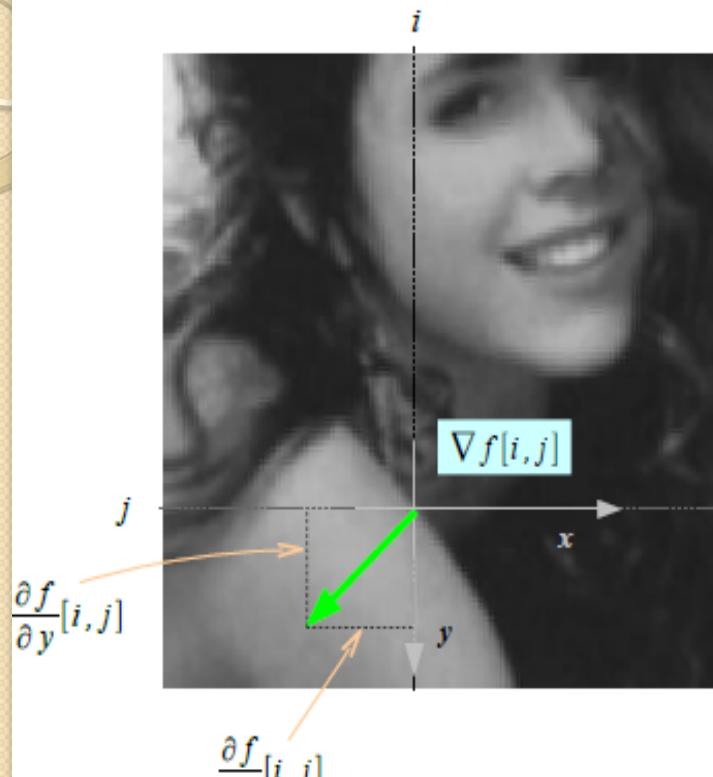


Moyenne V8



Médian V8

# Filtres Dérivateurs



Les *variations locales d'intensité* constituent une source primordiale d'information en traitement d'images. Elles sont mesurées par le *gradient*, fonction *vectorielle* des pixels  $[i, j]$  :

$$\nabla f[i, j] = \left( \frac{\partial f}{\partial x}[i, j], \frac{\partial f}{\partial y}[i, j] \right)$$

D'autres grandeurs différentielles sont utilisées en traitement d'images, comme le *laplacien*, fonction *scalaire* de  $[i, j]$  :

$$\Delta f[i, j] = \frac{\partial^2 f}{\partial x^2}[i, j] + \frac{\partial^2 f}{\partial y^2}[i, j]$$

ou encore le *hessien*, fonction *matricielle* de  $[i, j]$  :

$$H_f[i, j] = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2}[i, j] & \frac{\partial^2 f}{\partial x \partial y}[i, j] \\ \frac{\partial^2 f}{\partial x \partial y}[i, j] & \frac{\partial^2 f}{\partial y^2}[i, j] \end{pmatrix}$$

Le problème du calcul des filtres déivateurs dans les images numériques est l'*approximation* de ces grandeurs différentielles dans notre espace discret ; on s'intéresse aussi à leur *utilisation* : réhaussement, détection de contours,...

# Filtre Laplacien

## (Dérivation seconde de l'image)

Formellement, la définition du laplacien de la fonction  $I(x,y)$  est

$$\Delta I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Par discréétisation, on obtient la relation approchée  $\Delta I = 4 I(x,y) - I(x,y+1) - I(x-1,y) - I(x+1,y) - I(x,y-1)$  ce qui conduit au masque :

$$\Delta = \begin{vmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{vmatrix}$$

La présence d'un contour est indiquée par le passage du laplacien par zéro. Toutefois, le laplacien est plus souvent utilisé pour reconnaître à quelle région appartient un pixel.

# Seuillage

# Introduction

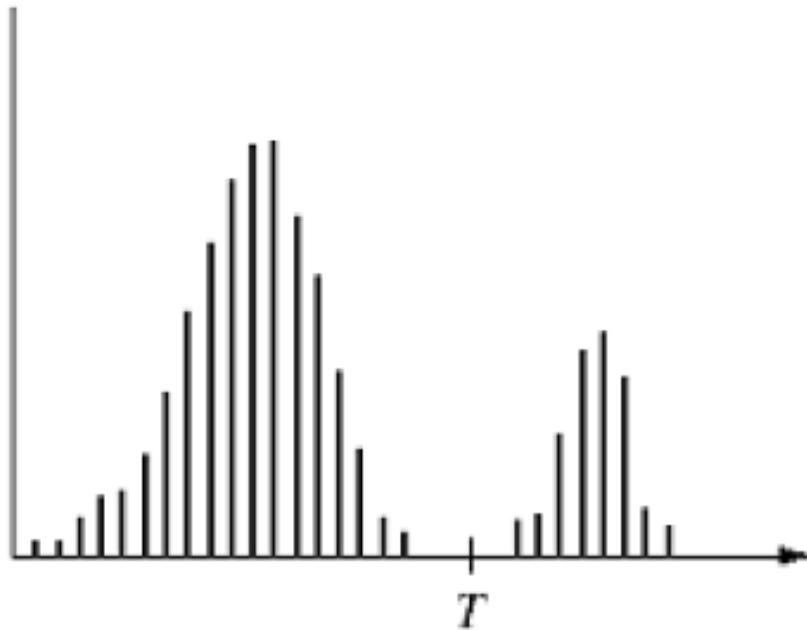
- Le seuillage est une méthode simple et très populaire pour le traitement des images numériques
- Ce n'est pas une méthode de segmentation en régions
  - *Approche pixel (pas région ni contour)*
  - *Mais on l'utilise souvent en segmentation (avec post-traitements)*
- Le seuillage peut être
  - Global : un seuil pour toute l'image
  - Local : un seuil pour une portion de l'image
  - Adaptatif : un seuil s'ajustant selon les parties de l'image

# Principe de base

- Seuillage de base (2 classes) :
  - Si  $valeur(pixel) \geq seuil$  alors  $valeur(pixel) = 1$
  - Si  $valeur(pixel) < seuil$  alors  $valeur(pixel) = 0$
- Le résultat du seuillage est une image binaire
  - 0 ou 1 (*qu'on transforme parfois en 0:255 pour l'affichage*)
- **Problème** : choix du seuil !

# Seuillage d'histogramme simple

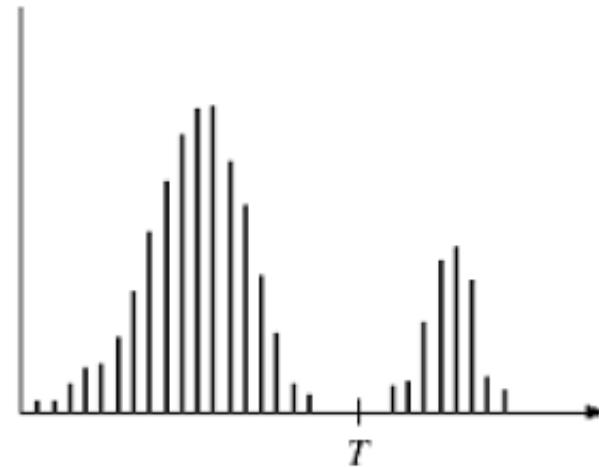
$$g(x, y) = \begin{cases} 1 & \text{si } f(x, y) \geq T \\ 0 & \text{si } f(x, y) < T \end{cases}$$



# Comment définir le seuil?

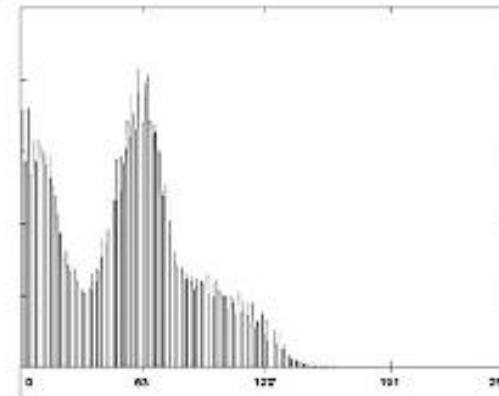
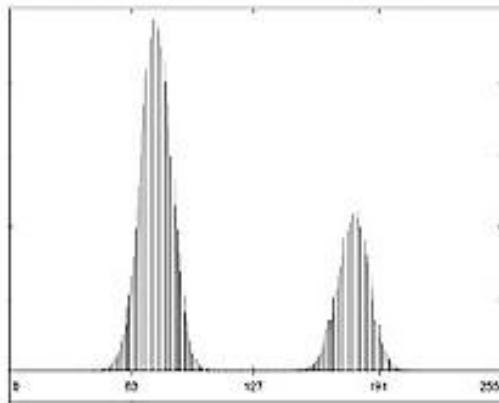
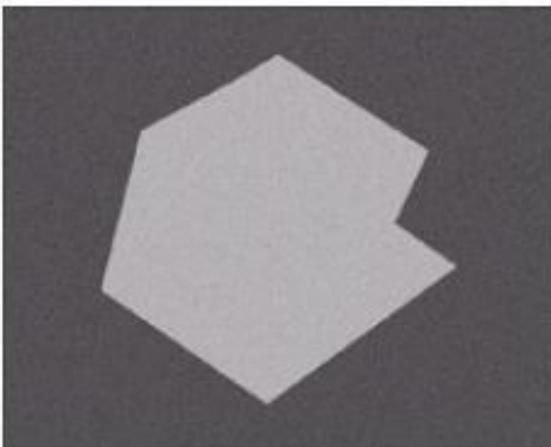
- Comment trouver le bon seuil ( $T$ ) ?
  - Une valeur obtenue par tests
  - La valeur moyenne des tons de gris
  - La valeur médiane entre le ton maximum et le ton minimum
  - Une valeur qui balance les deux sections de l'histogramme

*Il existe des algorithmes automatiques pour trouver le seuil !*



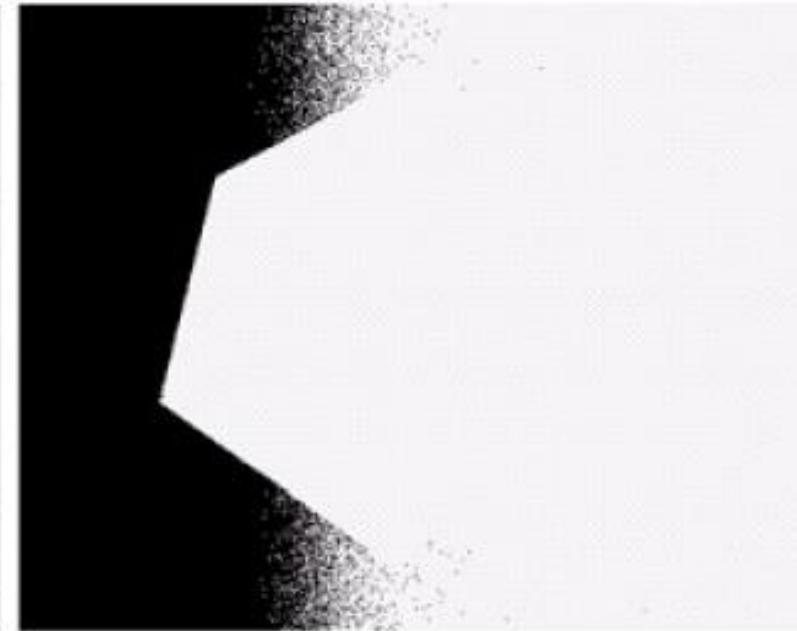
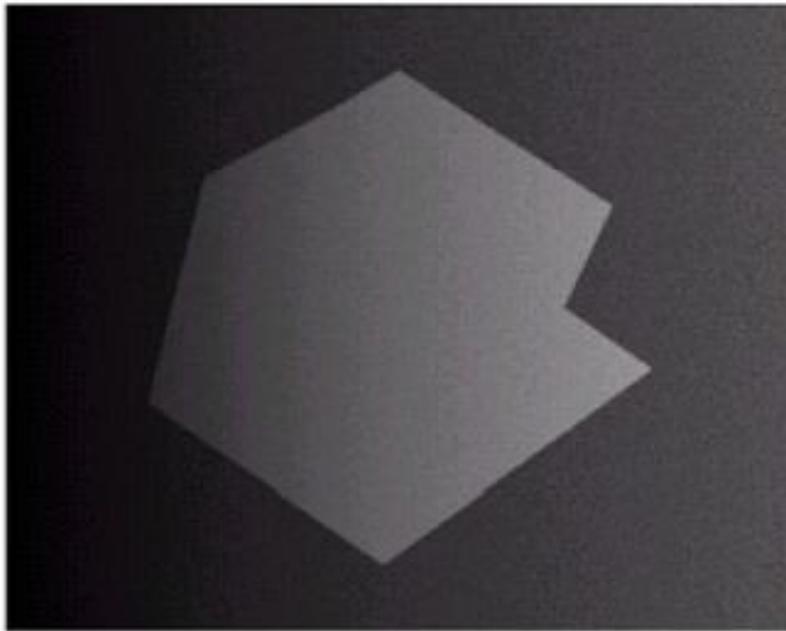
# Seuillage global -problème

*Problème d'éclairage ?*



# Seuillage global -problème

- **Problème** : Le seuillage global ne peut traiter ce cas
- **Solution** : seuillage local adaptatif

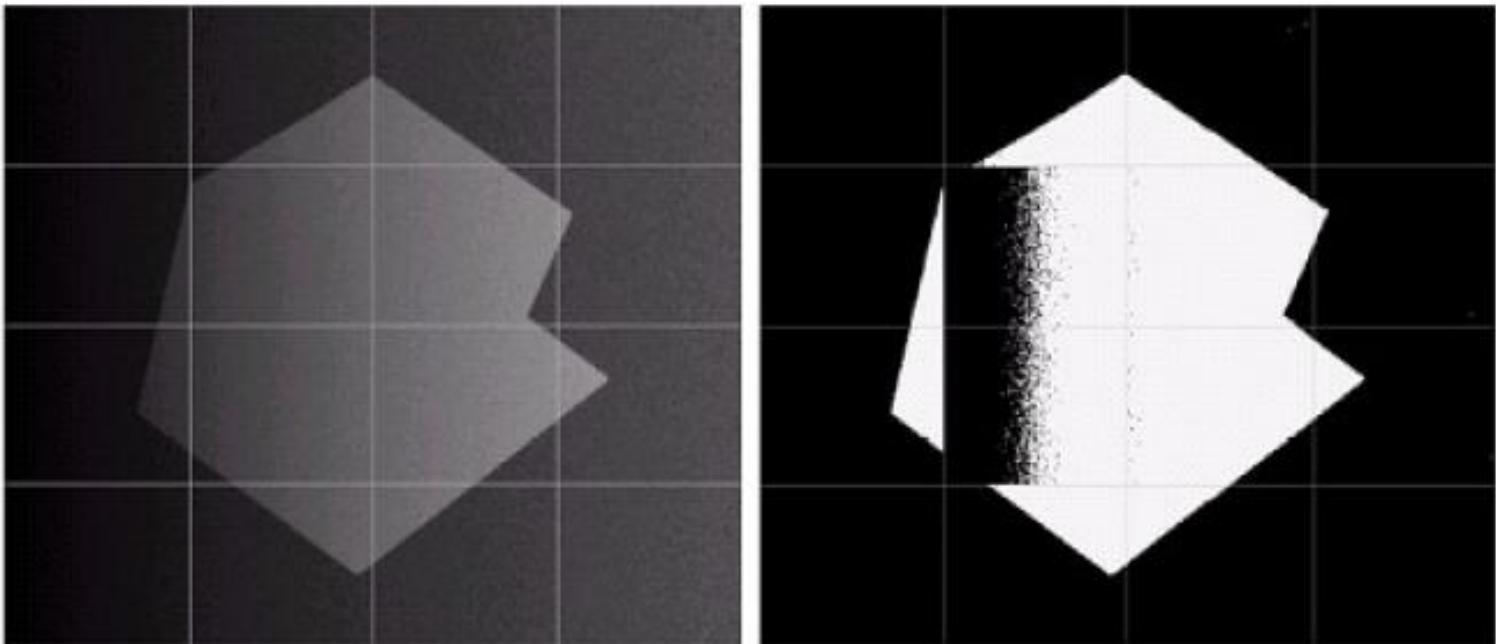


# Exemple de seuillage adaptatif

- Nous avons besoin de séparer l'image en sous images, et de traiter chacune avec son propre seuil
- Le choix de la dimension des sous-images est important
- Avant de traiter chaque sous-image, nous vérifions la variance des tons de gris pour décider s'il existe un besoin de segmentation
  - *Exemple : pas besoin si variance < 100*

# Exemple de seuillage adaptatif

- On divise l'image en sous-images
- On seuille chaque sous-image indépendamment
- Les 4 sous images de coins ne sont pas traitées car  $variance < 100$



# Segmentation

# Introduction

La segmentation des images consiste à regrouper les pixels de ces images qui partagent un même propriété pour former des régions connexes

Deux approches :

- **Approche « contours »** : les régions sont délimitées par les contours des objets qu'elles représentent (séparation)
- **Approche « régions »** : les régions sont déterminées en fonction de leurs propriétés intrinsèques (agrégation de pixels fonction d'un critère d'homogénéité)

# Introduction

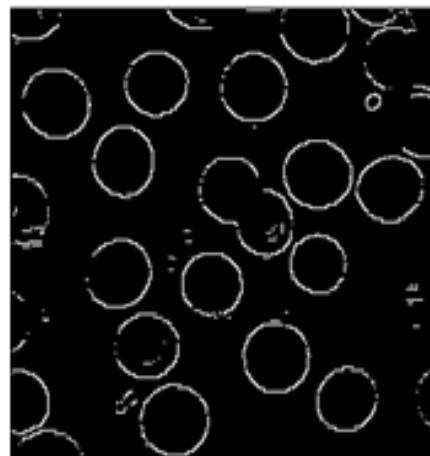
- La segmentation est normalement basée sur:
  - les discontinuités
    - les arrêtes, les changements abruptes, ...
  - les similitudes (zones homogènes)
    - couleurs, textures, intensités, ...
- La segmentation est le découpage d'une image en différentes régions et/ou contours.

# Introduction

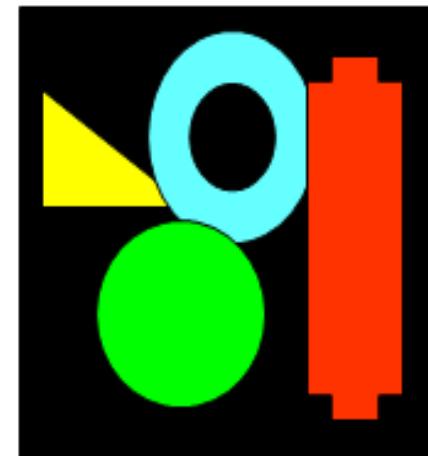
Segmentation



Recherche de frontières  
(approches « contours »)



Recherche de régions  
(approches « régions »)



# Segmentation en Régions

- Méthodes descendantes (division)
- Méthodes ascendantes (agrégation de pixels ou fusion de régions)
- Méthodes mixtes (division- fusion)

# Segmentation par division (split)

Il s'agit de considérer une région comme un ensemble de points possédant la même propriété (homogénéité). Les critères d'homogénéité peuvent être la couleur ou le niveau de gris, la texture, ....

Elle est basée sur la technique du Quad Tree : l'image est supposée carrée : on la divise en 4 quadrants, puis chaque quadrant en 4 sous-quadrants et ainsi de suite.

L'algorithme de division est simple et récursif :

```
Separation(zone)
Si critere(zone)=VRAI alors
    classer
sinon
    diviser zone en 4 sous-zones Z1,Z2,Z3,Z4
    Pour i=1 à 4 faire :
        Separation(Zi)
    fin Pour
fin Si
```

# Segmentation par division (split)

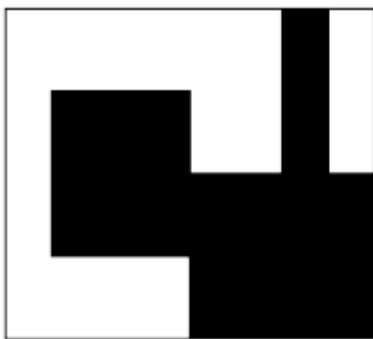
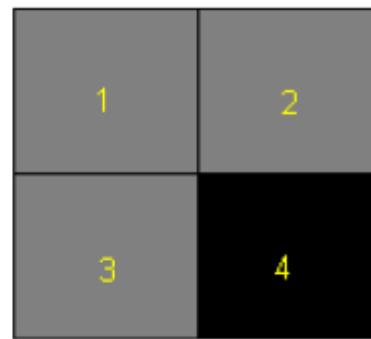
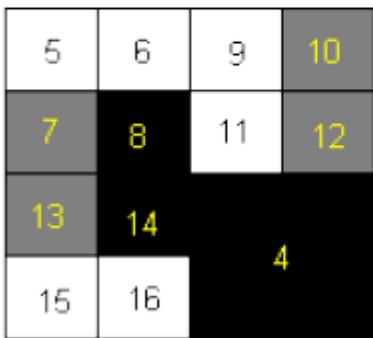


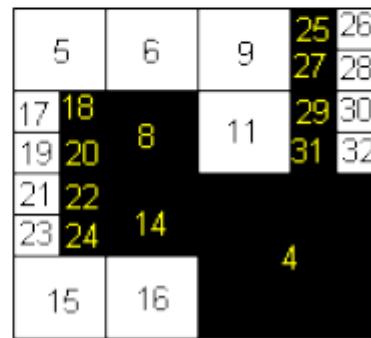
image originale



1er niveau de séparation



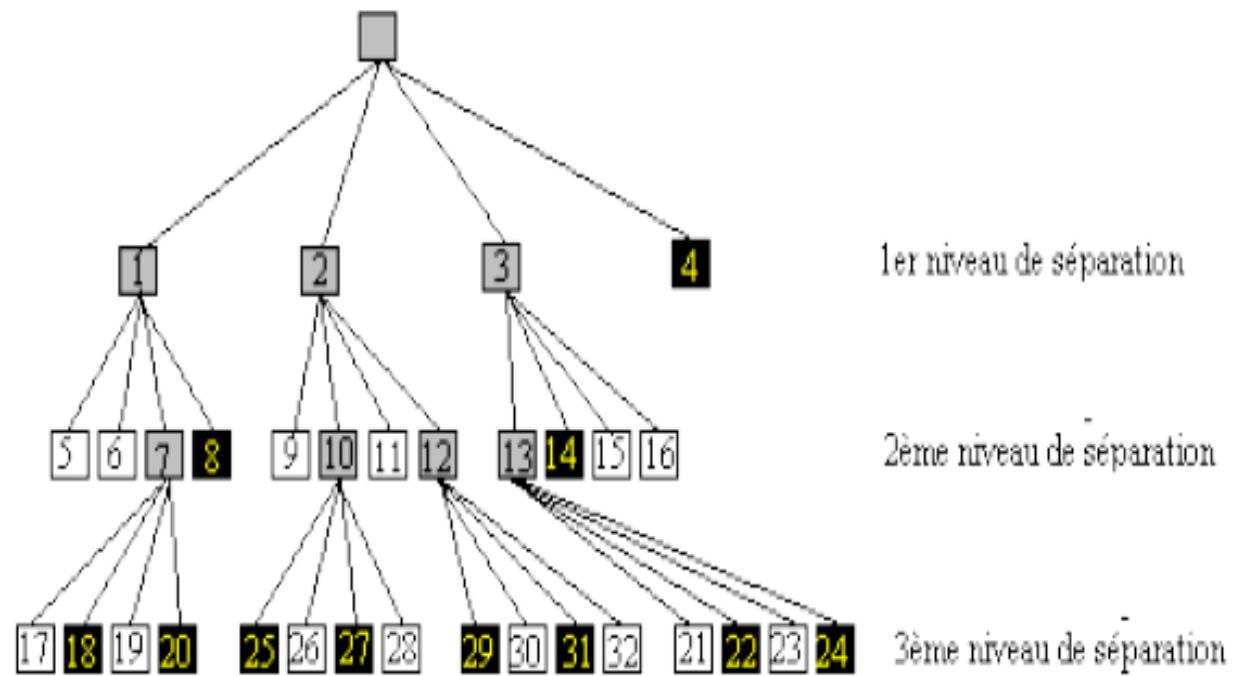
2ème niveau de séparation



3ème niveau de séparation

# Segmentation par division (split)

et le quad tree est :



# Segmentation par division (split).... Exemple2

0	1	0	0	7	7	7	7
1	0	2	2	7	7	7	7
0	2	2	2	7	7	7	7
4	4	2	2	7	7	7	7
0	0	1	1	3	3	7	7
1	1	2	2	3	7	7	7
2	4	3	0	5	7	7	7
2	3	3	5	5	0	7	7

Image initiale

0	1	0	0	7	7	7	7
1	0	2	2	7	7	7	7
0	2	2	2	7	7	7	7
4	4	2	2	7	7	7	7
0	0	1	1	3	3	7	7
1	1	2	2	3	7	7	7
2	4	3	0	5	7	7	7
2	3	3	5	5	0	7	7

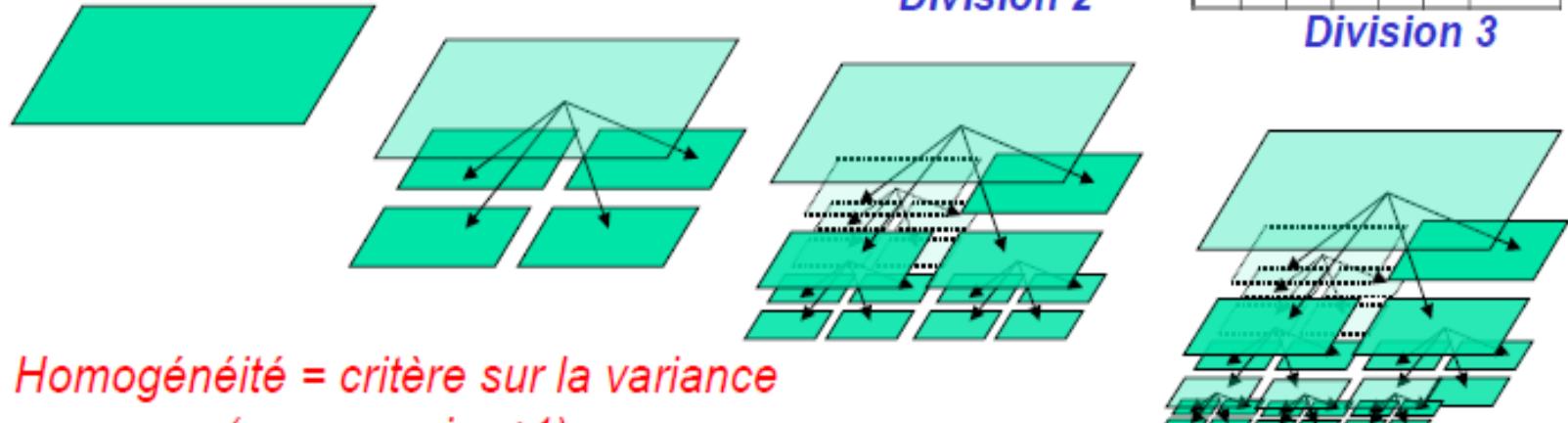
Division 1

0	1	0	0	7	7	7	7
1	0	2	2	7	7	7	7
0	2	2	2	7	7	7	7
4	4	2	2	7	7	7	7
0	0	1	1	3	3	7	7
1	1	2	2	3	7	7	7
2	4	3	0	5	7	7	7
2	3	3	5	5	0	7	7

Division 2

0	1	0	0	7	7	7	7
1	0	2	2	7	7	7	7
0	2	2	2	7	7	7	7
4	4	2	2	7	7	7	7
0	0	1	1	3	3	7	7
1	1	2	2	3	7	7	7
2	4	3	0	5	7	7	7
2	3	3	5	5	0	7	7

Division 3



Homogénéité = critère sur la variance  
(ou  $\max-\min \leq 1$ )

# Segmentation par Fusion (Merge)

Alors que la méthode précédente consistait à diviser l'image, les méthodes par fusion appliquent le principe inverse : on fait croître une région en y incorporant les régions voisines ayant les mêmes critères d'homogénéité (fusion de région). La règle de fusion est double : les deux régions correspondent au même critère et elles sont adjacentes.

La méthode de la coloration de tâches consiste à promener sur l'image une fenêtre de trois points. En chaque point on applique l'algorithme indiqué ci-dessous où Couleur est un niveau de couleur attribué aux pixels satisfaisant le même critère.

## Algorithme de coloration de tâches

Pour chaque pixel  $I(i, j)$  Faire

Si Critère  $(i, j) = \text{Critère } (i-1, j)$

Alors Couleur  $(i, j) \leftarrow \text{Couleur } (i-1, j)$

Sinon Si Critère  $(i, j) = \text{Critère } (i, j-1)$

Alors Couleur  $(i, j) \leftarrow \text{Couleur } (i, j-1)$

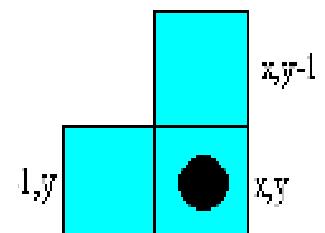
Sinon Couleur  $(i, j) \leftarrow \text{NouvelleCouleur}$

Si  $(\text{Critère } (i, j) = \text{Critère } (i-1, j))$

ET  $(\text{Critère } (i, j) = \text{Critère } (i, j-1))$

Alors Fusionner les régions en donnant la même couleur aux pixels  $I(i, j)$ ,  $I(i-1, j)$  et  $I(i, j-1)$

FinPour



# Segmentation par croissance de régions

- On débute avec un pixel, et on « ajouter » les pixels voisins qui répondent à un critère d'appartenance :
  - *Variance faible*
  - *Niveau de gris répondant un seuil*
  - ...
- Les pixels initiaux sont appelés « germes », « amorces » ou « semences »
  - *Choix des pixels initiaux automatiques ou semi-automatiques*
- La région « grandit » à partir de son germe
  - *Besoin d'une critère (ou prédicat) pour choisir les pixels à ajouter*

# Segmentation par croissance de régions

- On part d'un germe (*seed*) et on l'étend en ajoutant les pixels voisins qui satisfont le critère d'homogénéité
- Le germe peut être choisi soit par un humain, soit de manière automatique en évitant les zones de fort contraste (*gradient important*)



# Autres approches de Segmentation

- Segmentation par partage des eaux ...
- Segmentation CSC (Color Structure Code)..
- Segmentation par contours actifs (snakes)...
- Segmentation par clustering ...
- Autres...

# Segmentation -conseils

- La segmentation d'une image cause encore aujourd'hui beaucoup de problèmes
  - *Aucune méthode ne fonctionne pour toutes les images*
  - *Pas de garantie, pas de recette miracle !*
- Le **pré-traitement** des images, la **sélection de capteurs** et **sources d'énergie** appropriées, et la **prise contrôlée des images** rendent cette étape plus facile et plus efficace
- La segmentation aide beaucoup pour la reconnaissance, mais elle n'est pas obligatoire dans tous les cas

# Segmentation -conseils

- Evaluer le résultat d'une segmentation n'est pas facile
  - *Il dépend de l'application*
  - *Il dépend de ce qu'on veut*
  - *Il est subjectif et varie d'une personne à l'autre*
- Un des principaux problème est de définir le but de la segmentation
  - ***Qu'est-ce qu'on recherche exactement dans l'image ?***
    - *Eléments globaux de l'image ou détails fin de la composition ?*
    - *Présence d'un humain ou détails du visage ?*
- Il faut toujours se poser la question de ce que l'on veut faire avec la segmentation
  - *Cela permet de définir le degré de précision nécessaire*

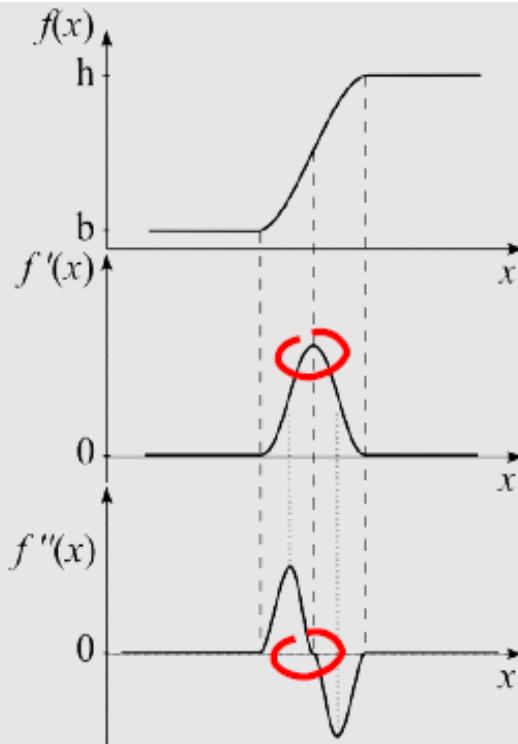
# Limites de la segmentation

*La segmentation ne peut pas trouver tous les objets de l'image tel que nous les interprétons !*

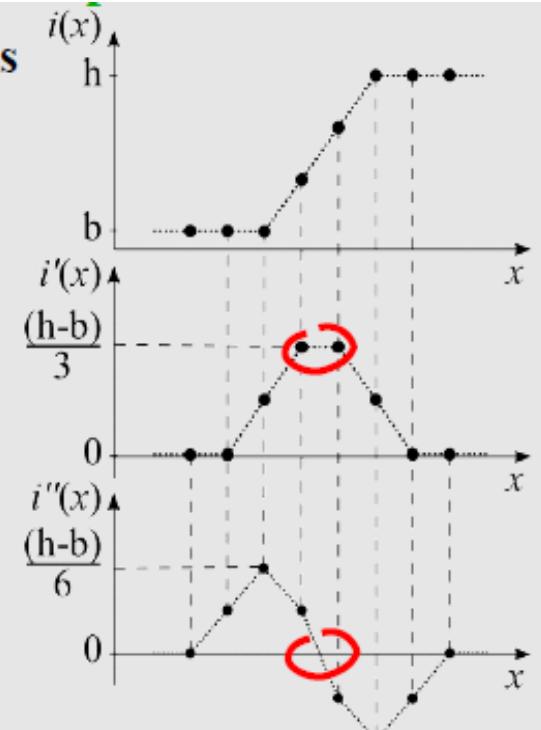


# Caractérisation des points contours

- Fonctions continues



- Fonctions discrètes



- Détection des points contours : utilisation d'un critère de décision
  - Dérivée première : **maxima locaux**
  - Dérivée seconde : passages par **zéro**

# Rappel sur le gradient

## • Dérivées premières en 2D et vecteur gradient

- On calcule une dérivée (partielle) de la fonction image  $f$  dans chaque direction principale.

- Le **vecteur gradient** est alors :

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

- En chaque point  $(x, y)$ , le vecteur gradient est caractérisé par :

- sa **norme** (ou module)  $\|\vec{\nabla} f(x, y)\| = \sqrt{\left(\frac{\partial f}{\partial x}(x, y)\right)^2 + \left(\frac{\partial f}{\partial y}(x, y)\right)^2}$

- sa **direction**  $\theta(x, y) = \arctan\left(\frac{\partial f}{\partial y}(x, y)/\frac{\partial f}{\partial x}(x, y)\right)$

# Détection des points contours

## • Résumé.

- La détection des points contours est basée sur les dérivées premières (gradient) ou secondes de la fonction sous-jacente à l'image.
- Le calcul de ces dérivées est approché au moyen de filtres de convolution.
  - Avantages : grande rapidité de calcul, aspect local.
  - Inconvénients : ces filtres sont très sensibles au bruit. Ils nécessitent donc l'emploi de filtres de lissage débruiteurs, souvent intégrés aux filtres de dérivation.
- Les filtres de lissage/dérivation sont moins précis que le filtre de dérivation « pur », mais plus robustes. Ils privilégiuent donc la détection des points contours par rapport à leur localisation.

# Détection des points contours

- **Vers la détection des contours.**

- Ces filtres permettent seulement d'estimer la « probabilité » qu'un pixel soit un **point contour candidat**. Il reste donc à :
  - décider si un pixel est *effectivement* un point contour, par exemple au moyen d'un seuillage :
    - si  $\|\vec{\nabla} f(x, y)\| > S$ , le pixel  $P(x, y)$  est un point contour candidat ;
    - si  $\|\vec{\nabla} f(x, y)\| \leq S$ , le pixel  $P(x, y)$  n'est pas point contour candidat.

- utiliser les points contours pour former les contours proprement dits. Cela nécessite des étapes supplémentaires, car les contours formés par ces points sont :
  - épais

- « bruités »

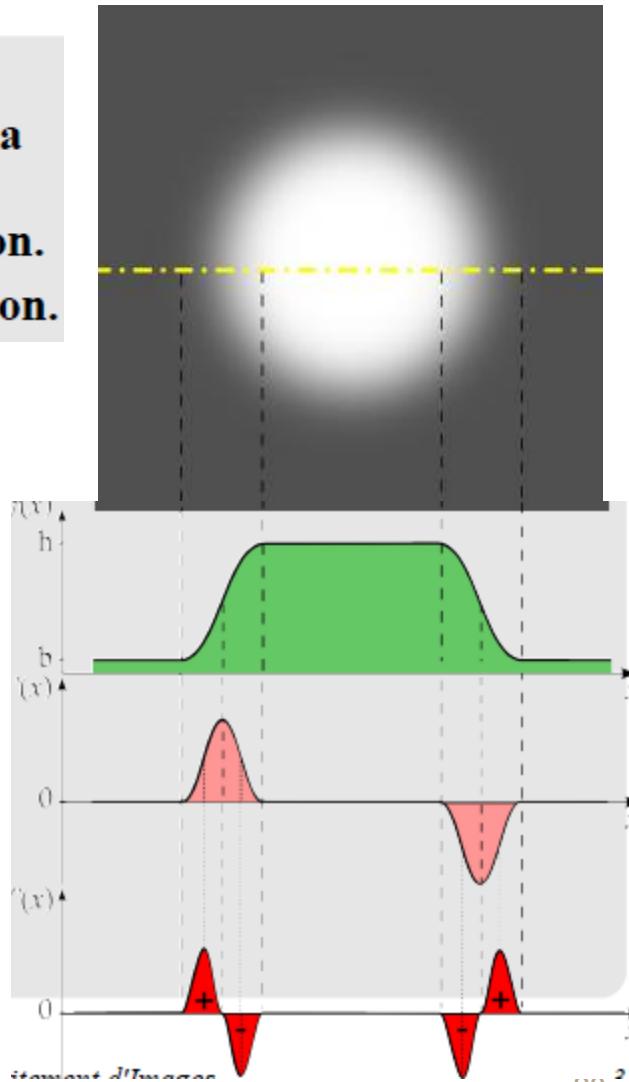
- interrompus  
(non fermés)



# Justification de l'approche par dérivée seconde

- Critique des approches par dérivées premières
  - Nécessitent une détection des maxima dans la direction du gradient, car l'épaisseur du « contour » dépend de la largeur de transition.
  - Difficulté de localiser le contour avec précision.

- Utilisation de la dérivée seconde
  - La dérivée seconde d'une fonction mesure sa courbure locale.
  - Utilisation dans la détection des contours :
    - Aux points contours, la dérivée seconde est nulle.
    - Plus précisément, les points contours sont caractérisés par un **passage par zéro** (ang. « zero crossing ») de la dérivée seconde.

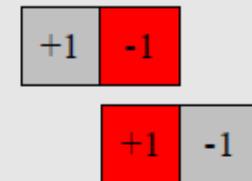


- Dérivées premières discrètes (*rappel*)

- **Masques de Roberts**

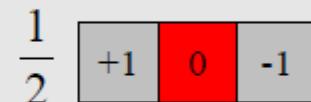
$$\frac{\partial f}{\partial x}(x_0, y_0) \approx f(x_0+1, y_0) - f(x_0, y_0)$$

ou  $\approx f(x_0, y_0) - f(x_0-1, y_0)$



- **Masques de Roberts2**

$$\frac{\partial f}{\partial x}(x_0, y_0) \approx \frac{f(x_0+1, y_0) - f(x_0-1, y_0)}{2}$$



- Dérivées secondes discrètes

- Meilleure approximation au centre : appliquer



$$\frac{\partial^2 f}{\partial x^2}(x_0, y_0) \approx \frac{\partial}{\partial x}[f(x_0+1, y_0) - f(x_0, y_0)]$$

$$\approx [f(x_0+1, y_0) - f(x_0, y_0)] - [f(x_0, y_0) - f(x_0-1, y_0)]$$

$$\approx f(x_0+1, y_0) + f(x_0-1, y_0) - 2f(x_0, y_0)$$

- Masque associé à la dérivée seconde selon  $x$  :



# Détection de Contours: Approche gradient

## Masque de Prewitt

Les masques dérivateur sont maintenant :

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

$$Gy = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

Lissage des lignes ou des colonnes

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

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

Dérivée des lignes ou des colonnes

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

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

On combine à la fois un filtrage et une dérivée  
→ moins sensible au bruit que le calcul direct des dérivées.

# Détection de Contours: Approche gradient

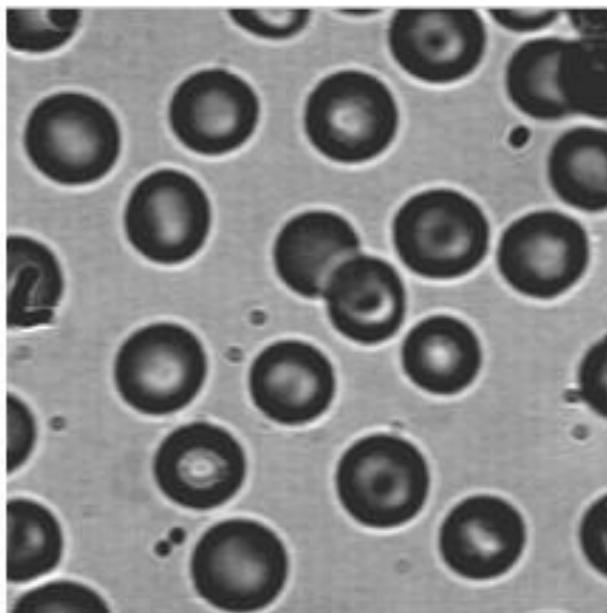


Image originale

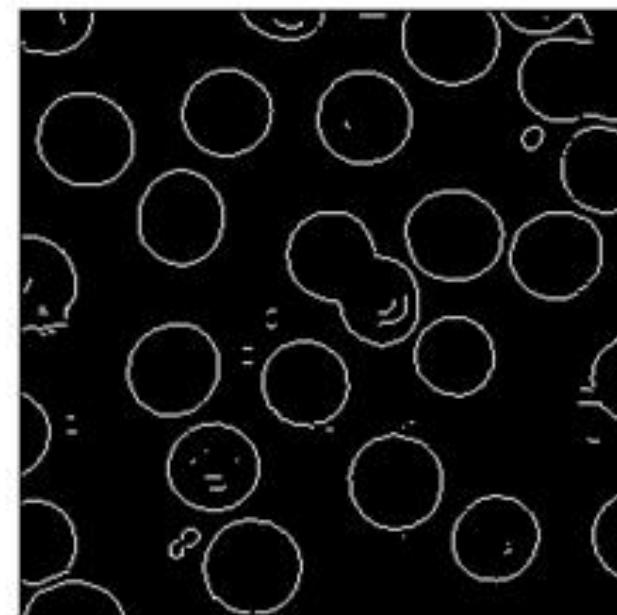


Image des contours  
(opérateur Prewitt)

# Détection de Contours: Approche gradient

## Masque de Sobel

Même principal mais autre filtrage préalable

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * [-1 \quad 0 \quad 1]$$

$$Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} * [1 \quad 2 \quad 1]$$

# Détection de Contours: Approche gradient

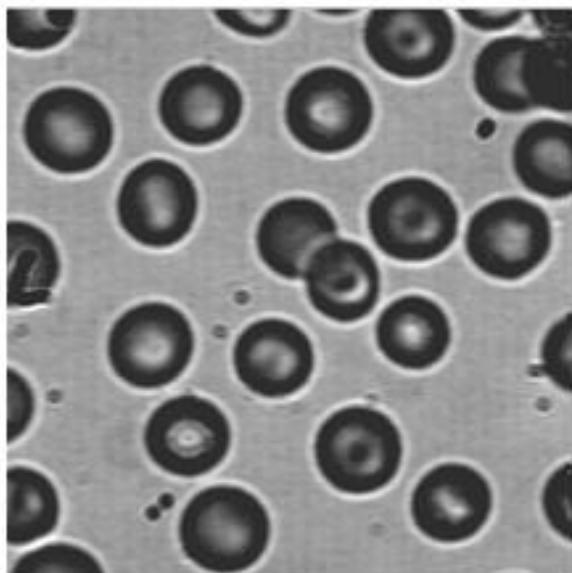


Image originale

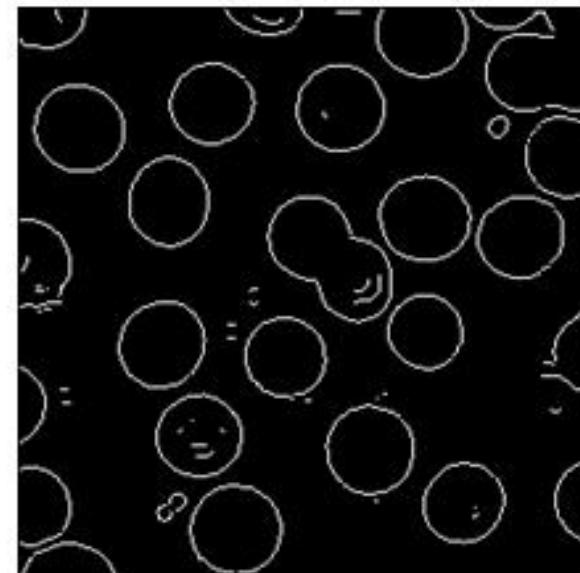


Image des contours  
(opérateur Sobel)

# Détection de Contours: Approche gradient

Autres Filtrage célèbre: Filtre de Canny ...

# Comparaison Gradient/Laplacien

## • Gradient

### • Avantages

- Fournit l'orientation du contour
- Bonne localisation malgré le lissage
- La suppression des non-maxima locaux fournit des contours fins

### • Inconvénients

- Assez sensible au bruit  
⇒ nécessite un lissage
- Le seuillage fournit des contours non fermés.

## • Les deux approches

## • Laplacien

### • Avantages

- Proche du système visuel humain
- La détection des passages par 0 fournit des réseaux de lignes **fermées**

### • Inconvénients

- Grande sensibilité au bruit  
⇒ nécessite un lissage fort  
⇒ affecte la localisation
- Pas d'info sur l'orientation du contour
- Le seuillage des passages par 0 crée des lacunes (« ouvertures ») dans les contours

# Comparaison Gradient/Lapalcien

- Exemples de détection des points contours



Canny



Marr-Hildreth ( $\sigma=2$ )

- Conclusion

- Pas d'opérateur parfait pour détecter les contours
- On obtient en pratique des contours incomplets (ouverts)
  - **détection** incorrecte : pixels superflus, pixels manquants
  - **localisation** incorrecte : erreurs dans la position des points contours, l'orientation
- La détection des points contours n'est que la première étape dans la chaîne de segmentation

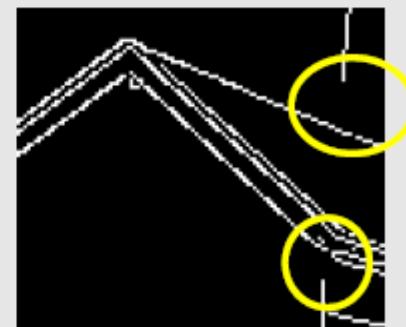
# Généralités sur les post-traitements

- **La détection des points contours fournit**

- une carte binaire des *points contours* ;
  - souvent, des contours ouverts  
(composantes de l'image non séparées en objets topologiquement distincts).

Causes :

- bruit
  - faible contraste
  - occultations



- **Nécessité de fermer les contours**

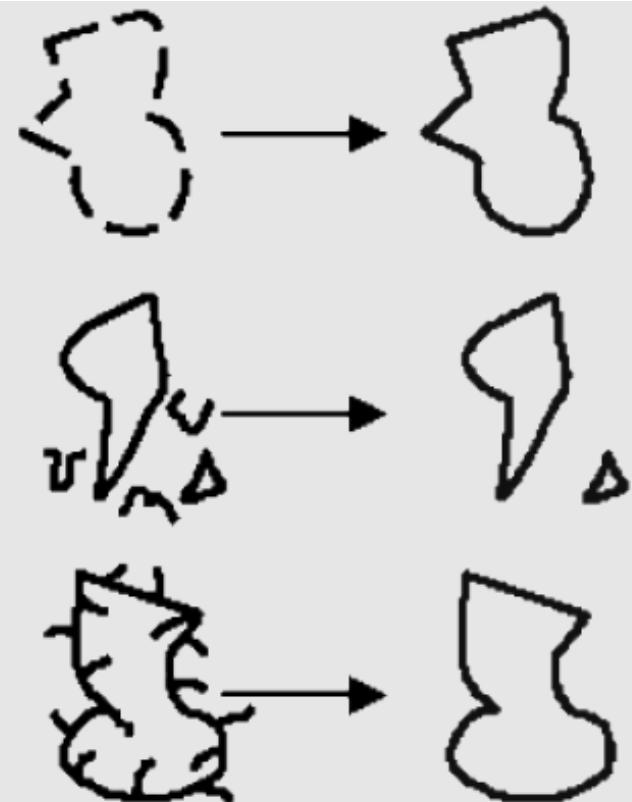
- **Pour obtenir des régions fermées**

- définies comme des composantes connexes maximales n'incluant pas de points contours ;
    - interprétables comme projections des objets de la scène.

- **Très important pour l'exploitation en segmentation d'image.**

# Généralités sur les post-traitements

- **Fermeture des contours par extrapolation**  
Ajout de points non détectés
- **Suppression des contours non fermés / non significatifs**
- **Suppression des « branches pendantes » des contours fermés (ébarbulage)**  
Rejet de points détectés par erreur
- **Codage ou chaînage des contours**



# Codage en chaînage des contours

- **Principe**

- Décrire un contour au moyen d'une structure de donnée.

- **Exemple simple : codage de Freeman**

- On code les changements de direction d'un pixel de contour à son voisin.
  - La séquence des codes locaux constitue le codage du contour. Ex. à partir de  $P$  :
  - $\{ 6, 0, 7, 0, 2, 0, 1, 1, 2, 2, 4, 2, 4, 5, 6, 4, 6, 4, 5 \}$
  - Réduction de la chaîne de contour

- **Autres approches possibles**

- Approches *globales* : recherche des contours complets  $\neq$  seulement points.
  - Exemple : transformée de **Hough** (déttection de contours continus en classes de formes : lignes, mais aussi cercles et ellipses).

