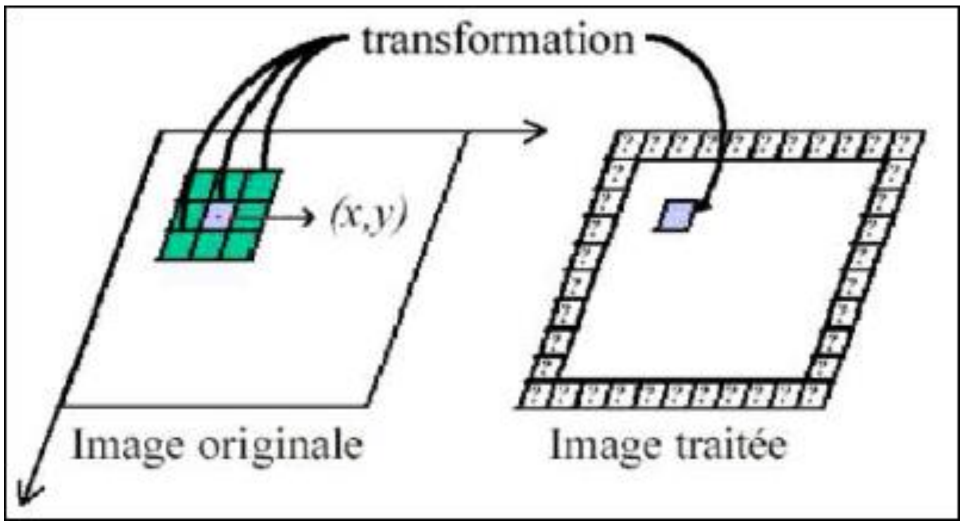
# Transformations locales : Filtrage, Détection de contours, …

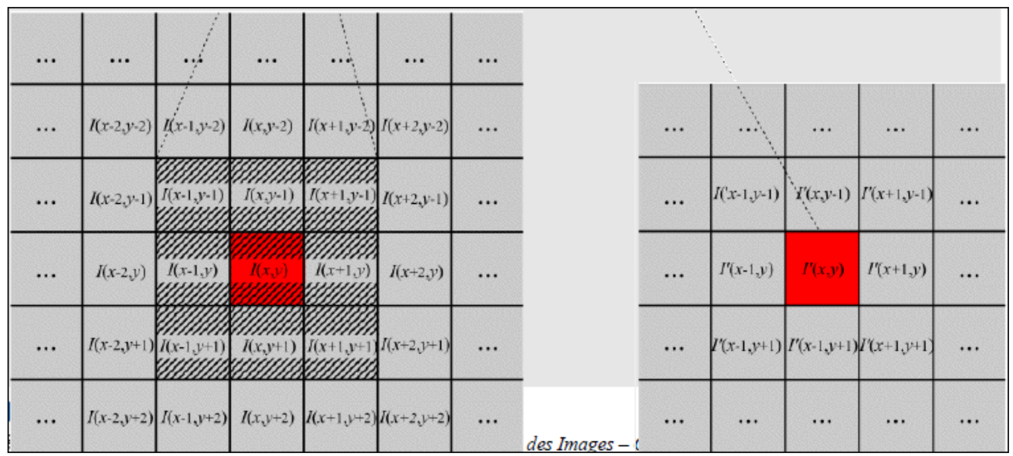
# Chapitre 2

# Transformations locales

## Principe

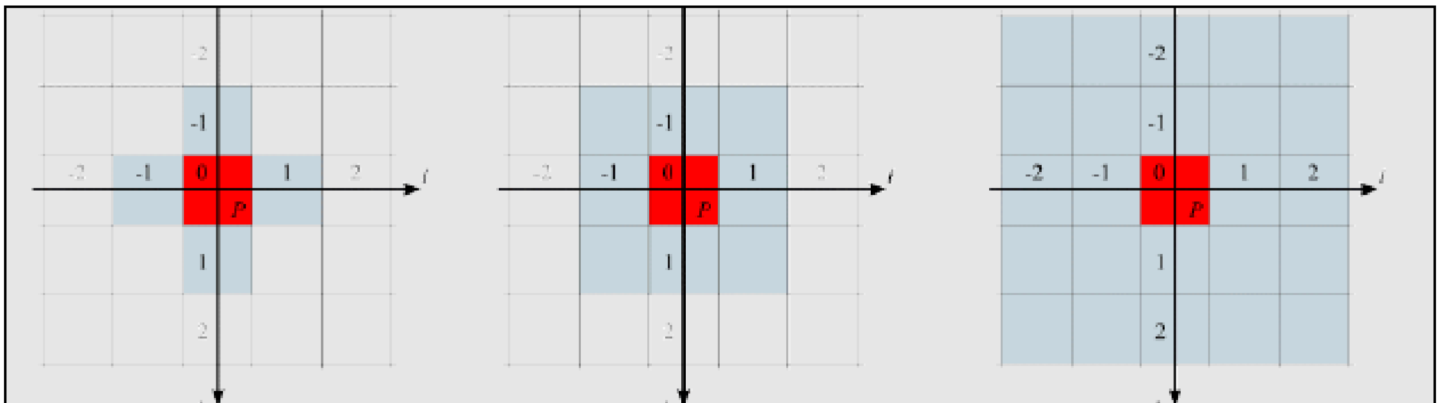
- Pour calculer la valeur du pixel de coordonnées (x,y) dans l'image résultat I', on utilise, dans l'image initiale, non seulement la valeur du pixel I(x,y) mais aussi celles des pixels situés dans un voisinage de ce dernier I(V (x,y))
- I' a même taille que I, mais des propriétés plus intéressantes



# Transformations locales

- **Voisinage** *V* **d'un pixel** ***P* :** *V* (*P*) est l'ensemble des pixels *Q* situés à moins d'une certaine distance de *P*.
  - *V* est centré en *P*
  - les pixels sont disposés selon une maille carrée.
  - La forme du voisinage (et le nombre de voisins) de *P* dépendent de la distance considérée
- **Voisinages les plus usités en traitement d'images :**



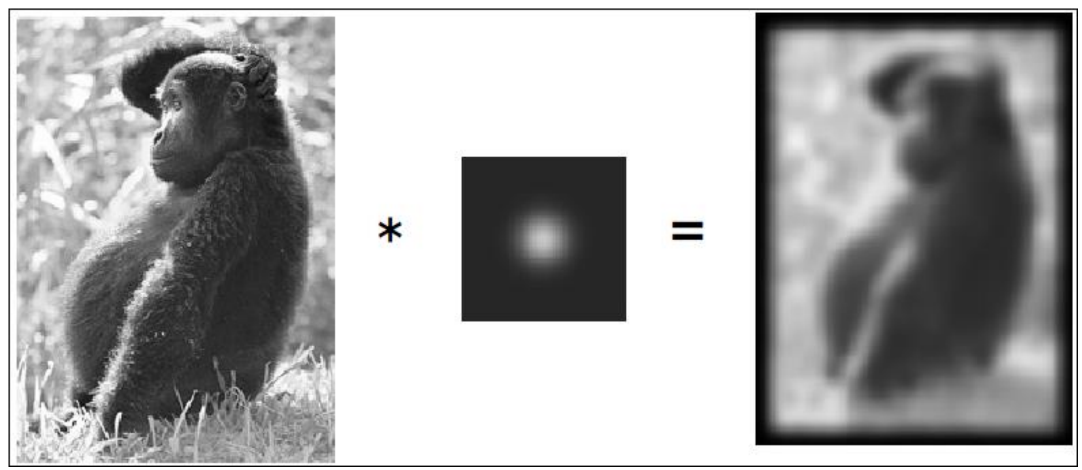

4-voisinage 8-voisinage *ou* « voisinage 3x3 » « voisinage 5x5 »

# Convolution numérique

- La convolution discrète est un outil permettant l'utilisation de **filtres linéaires** ou de filtres de déplacements invariants
- L 'équation générale de la convolution, notée g(x), de la fonction d'origine f(x) avec une fonction h(x) est :

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

- f(x) est la fonction d'origine et g(x) la fonction convoluée (résultat)
- Dans notre cas, une image est vue comme une fonction mathématique
- h(x) est appelé masque / noyau de convolution, fenêtre, kernel, …

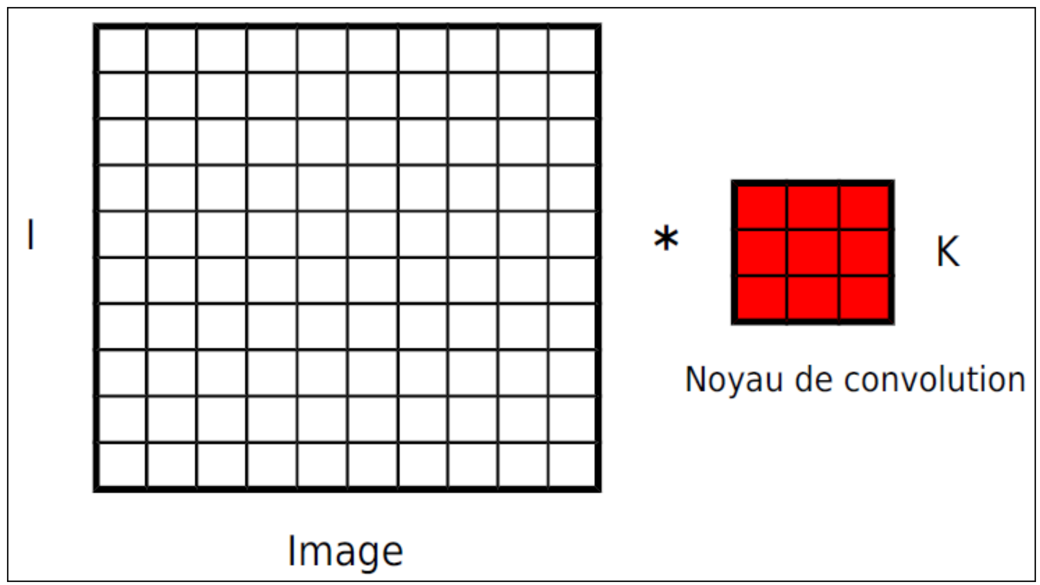


# Convolution numérique

- **Convolution d’une image par un filtre 2D :**

$$I'(i,j) = I(i,j) * filtre(i,j)$$
$$I'(i,j) = \sum_u \sum_v I(i-u, j-v) \cdot filtre(u,v)$$

- En pratique, la convolution numérique d'une image se fera par une **sommation de multiplications**
- Un filtre de convolution est une matrice généralement *(mais pas toujours)* de taille impaire et symétrique 3x3, 5x5, 7x7, …

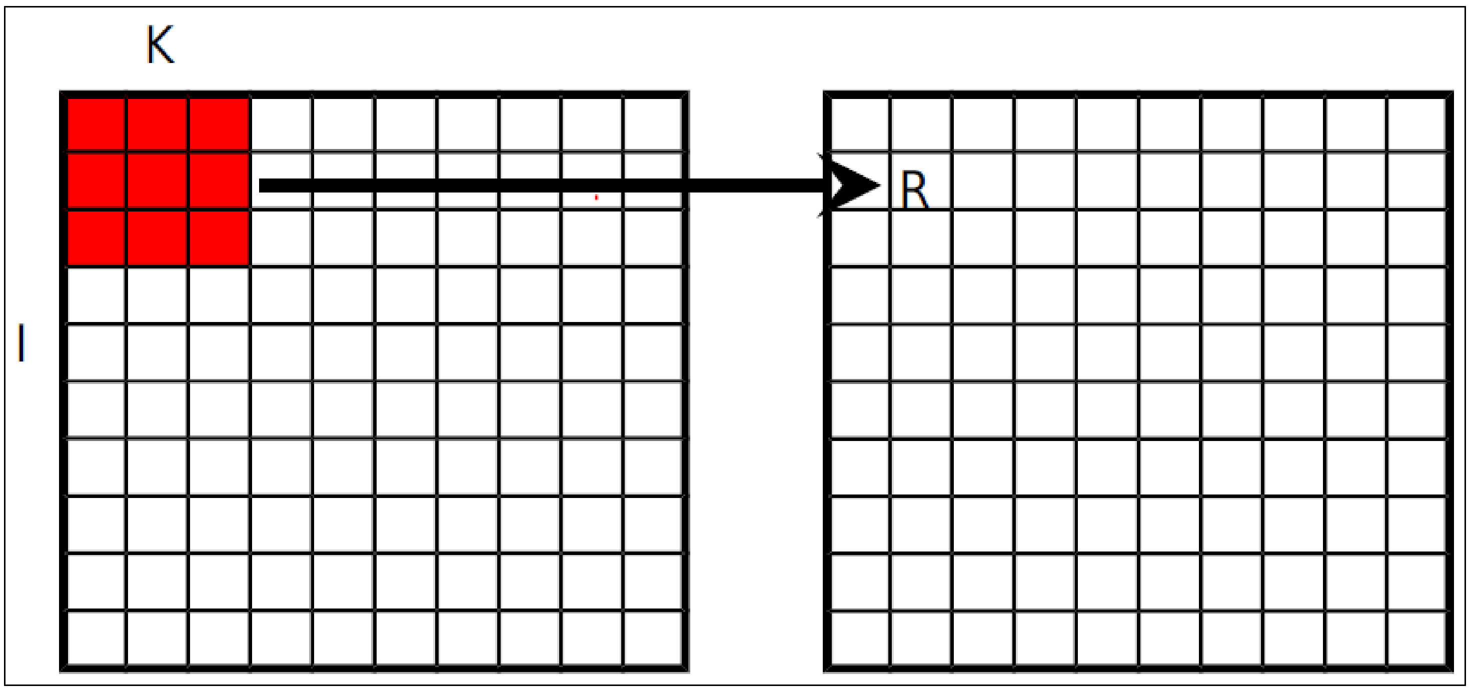


# Convolution numérique

- **Convolution d'une image par un filtre 2D :**

$$I'(i,j) = I(i,j) * filtre(i,j)$$

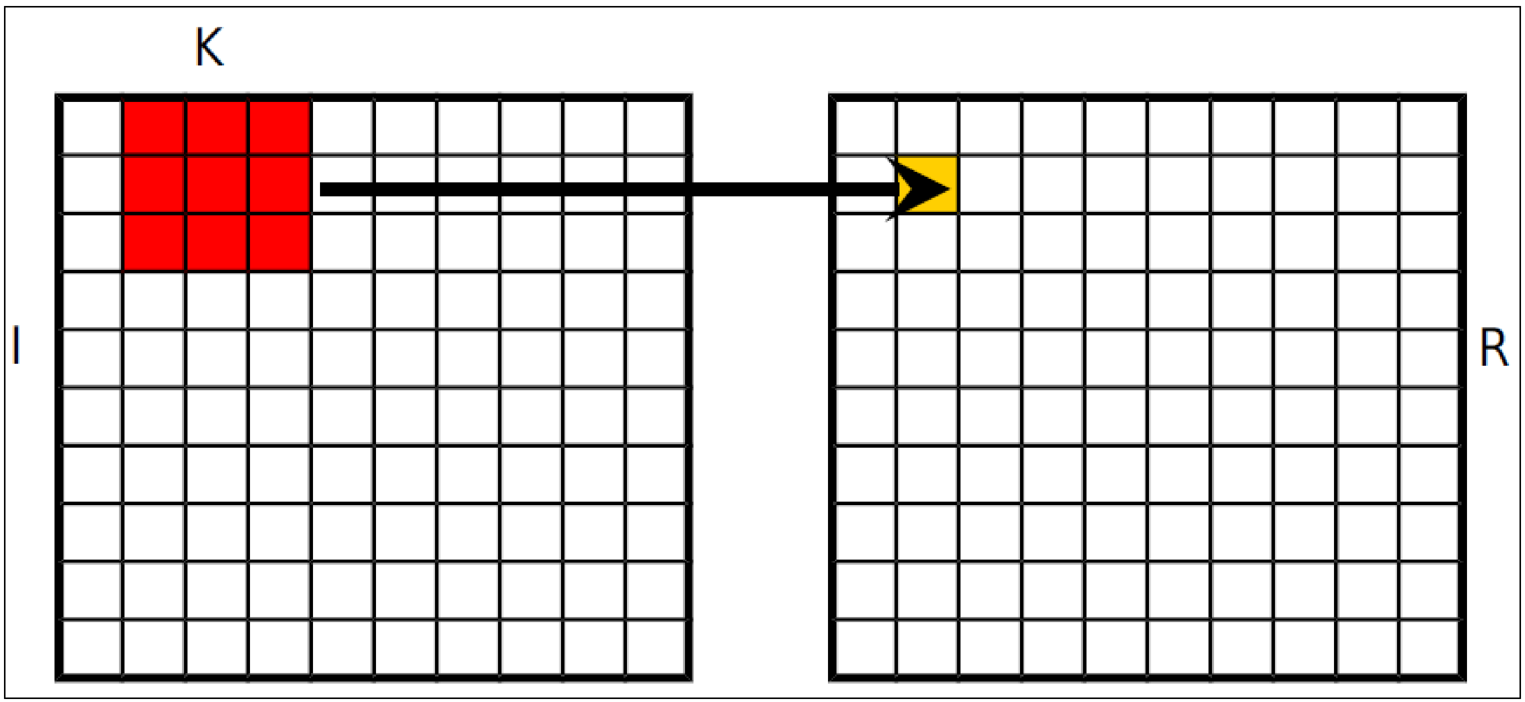
$$I'(i,j) = \sum_u \sum_v I(i-u, j-v) \cdot filtre(u,v)$$



# Convolution numérique

- **Convolution d'une image par un filtre 2D :**

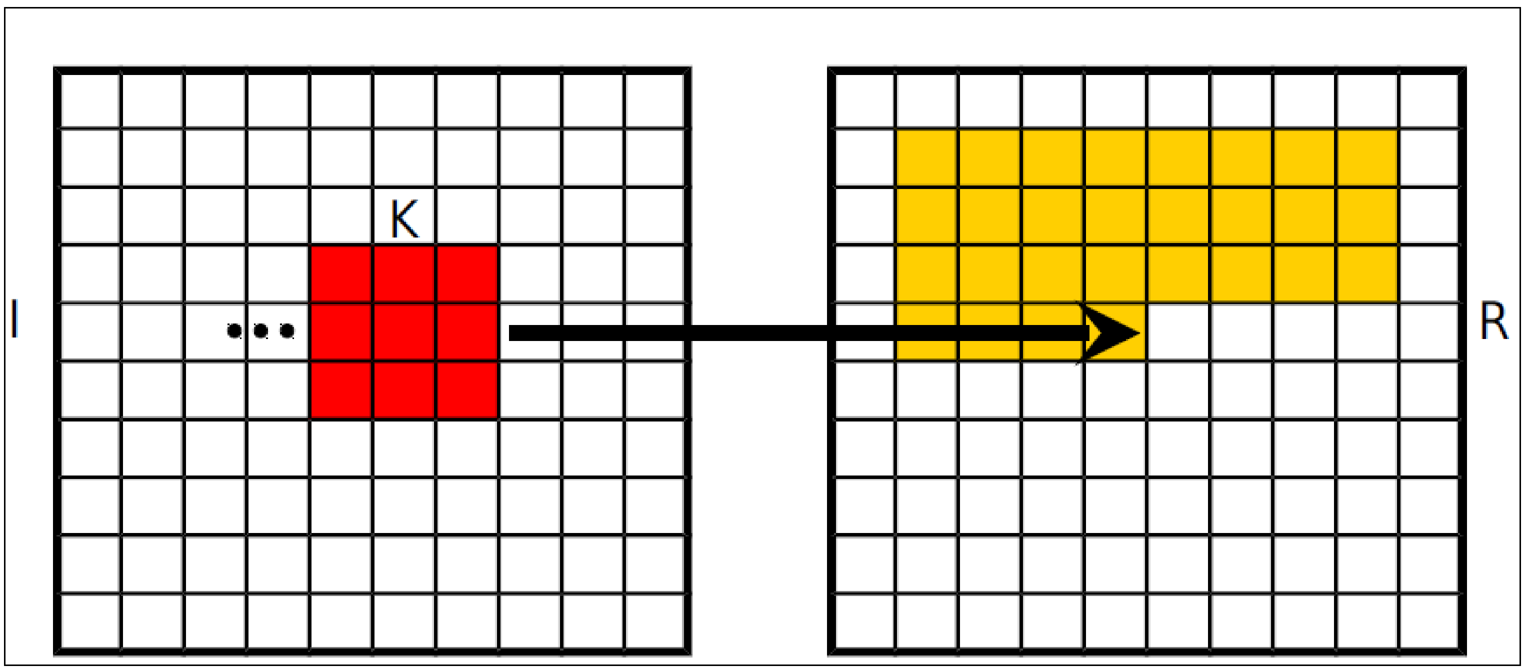
$$I'(i,j) = I(i,j) * filtre(i,j)$$
$$I'(i,j) = \sum_u \sum_v I(i-u, j-v) \cdot filtre(u,v)$$



# Convolution numérique

- **Convolution d'une image par un filtre 2D :**

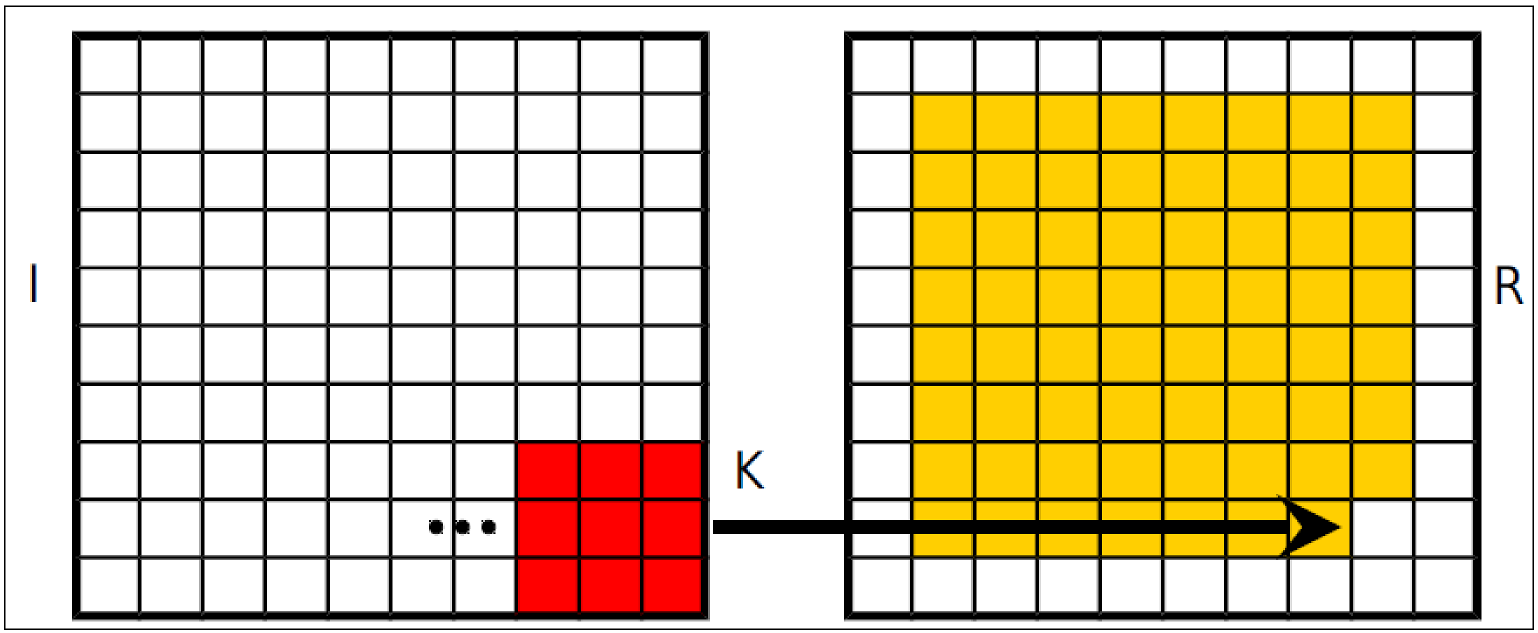
$$I'(i,j) = I(i,j) * filtre(i,j)$$
$$I'(i,j) = \sum_u \sum_v I(i-u, j-v) \cdot filtre(u,v)$$



# Convolution numérique

- **Convolution d'une image par un filtre 2D :**

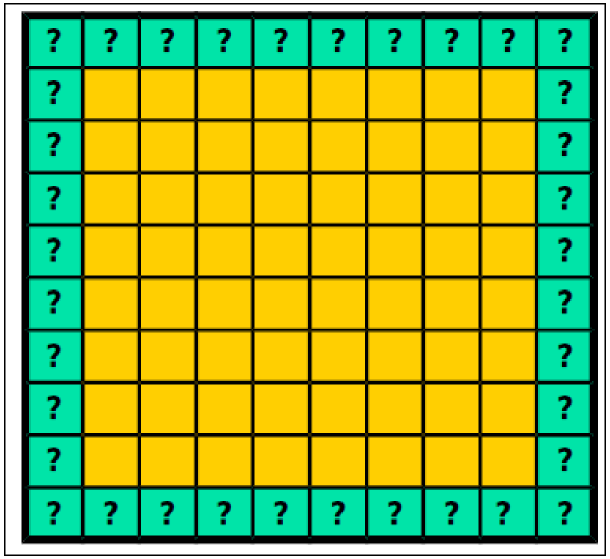
$$I'(i,j) = I(i,j) * filtre(i,j)$$
$$I'(i,j) = \sum_u \sum_v I(i-u, j-v) \cdot filtre(u,v)$$



# Convolution numérique

- Problème : Que faire avec les bords de l'image ?
  - Mettre à zéro (0)
  - Convolution partielle
  - Sur une portion du noyau
  - Réplication/interpolation des voisins après la convolution

  - Pas de solution miracle !

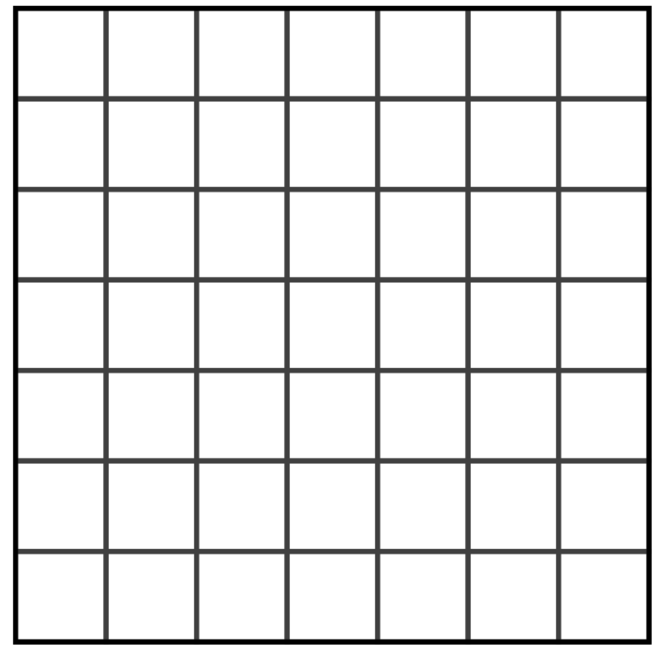


# Mais de tres bonnes propriétés?

- Commutative
  - A*B = B*A
- Associative
  - A*(B*C) = (A*B)*C
- Distributes over addition
  - A*(B+C) = A*B + A*C
- Plays well with scalars
  - x(A*B) = (xA)*B = A*(xB)

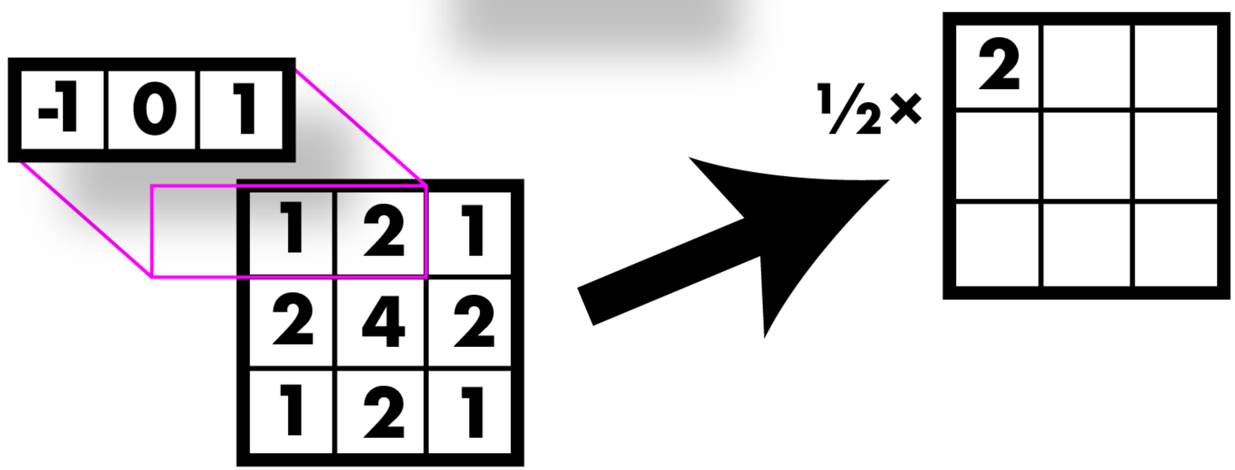
# Smooth first, then derivative

$$\frac{1}{2} \times \left( \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right) *$$

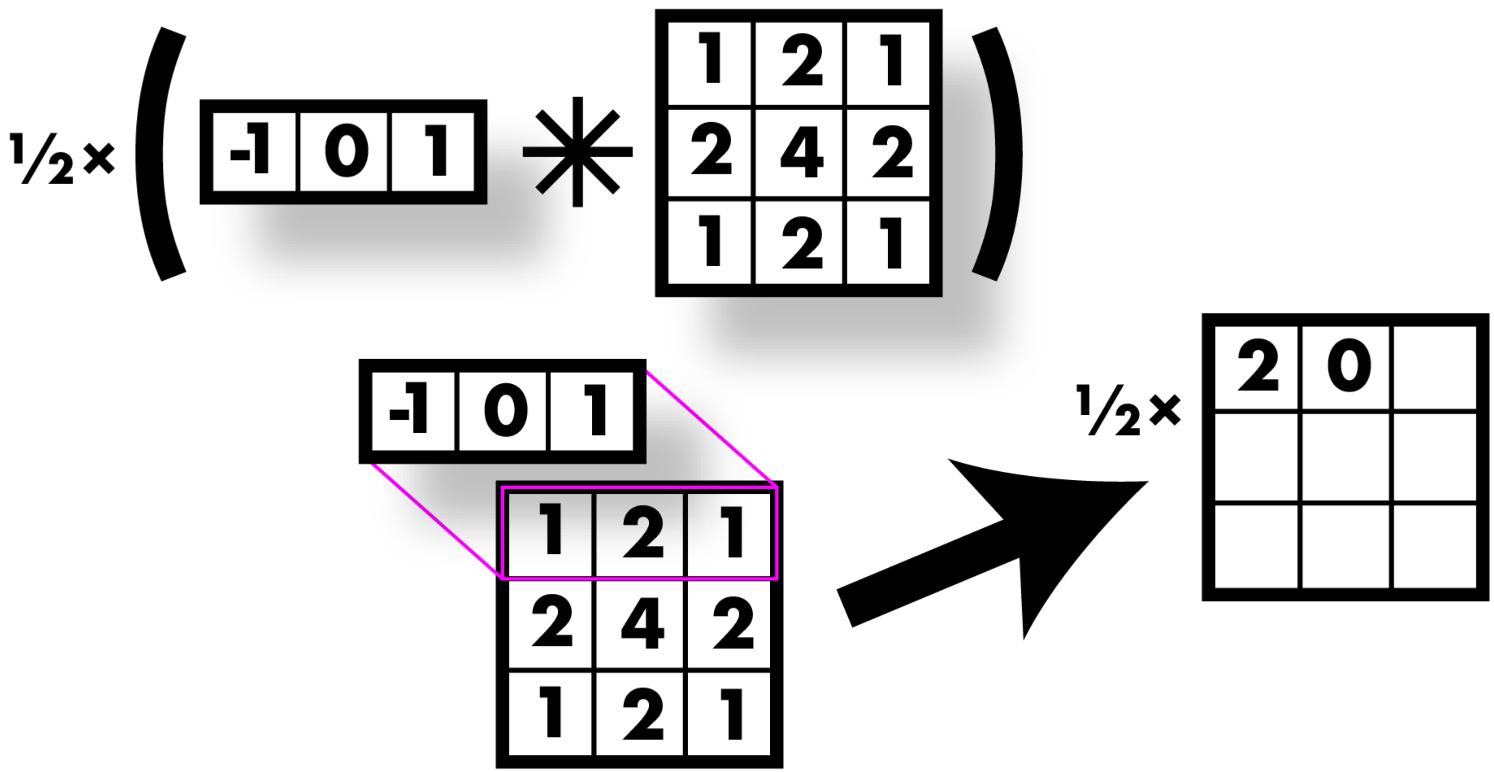


# Smooth first, then derivative

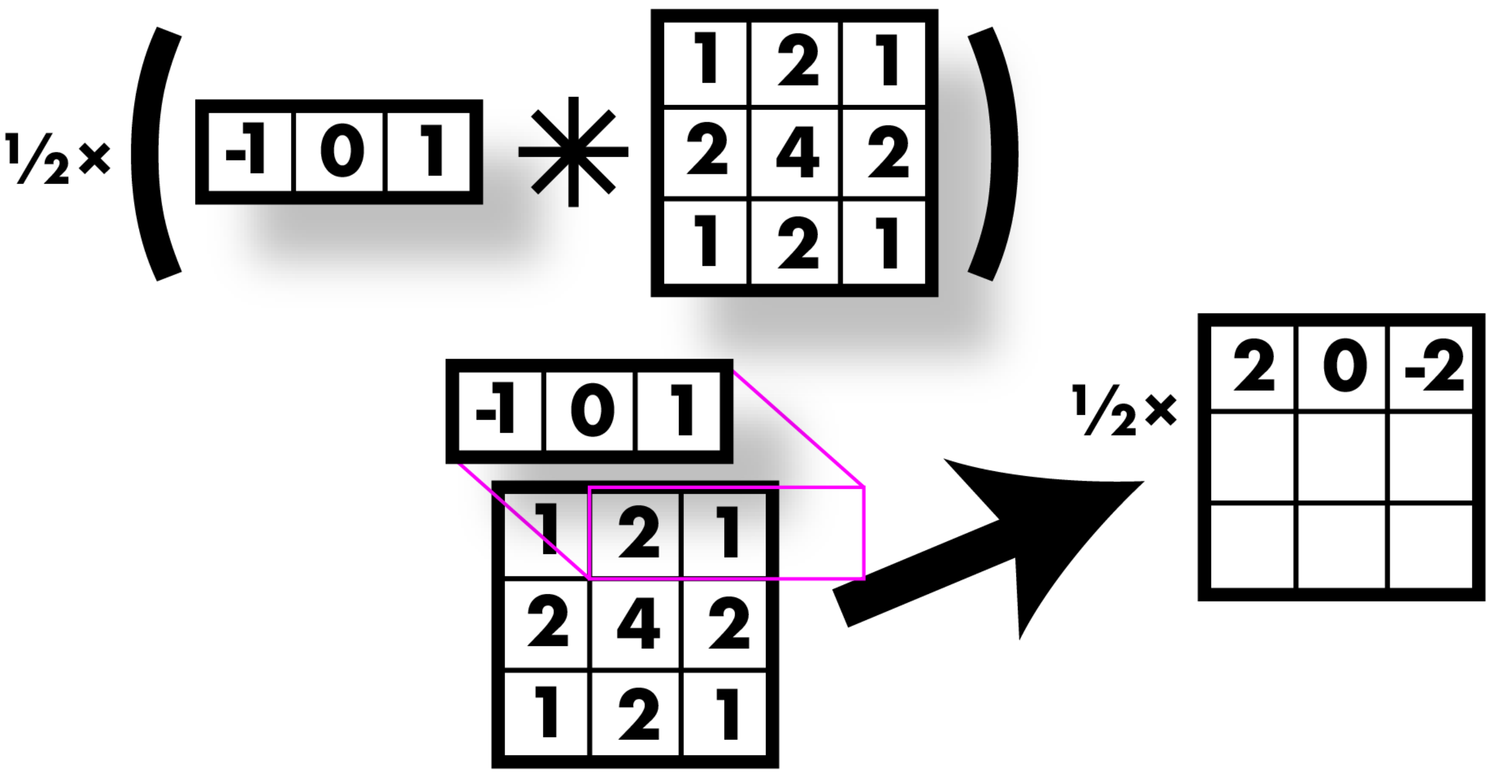
$$\tfrac{1}{2}\times\left(\begin{array}{|c|c|c|}\hline -1 & 0 & 1\\\hline\end{array} \ast \begin{array}{|c|c|c|}\hline 1 & 2 & 1\\\hline 2 & 4 & 2\\\hline 1 & 2 & 1\\\hline\end{array}\right)$$



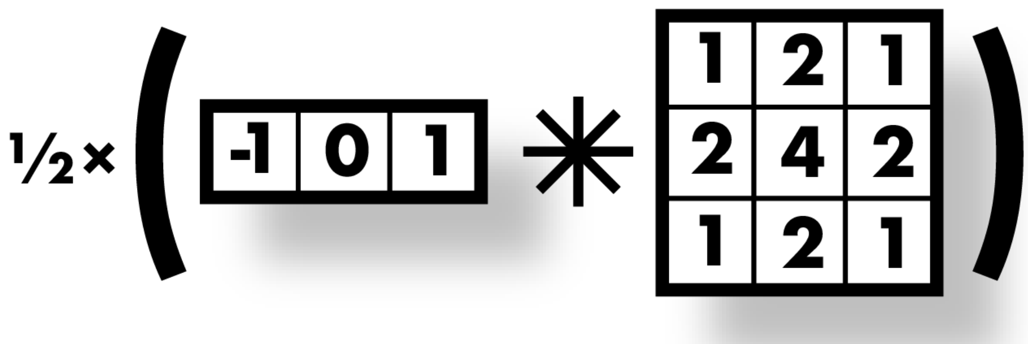
# Smooth first, then derivative

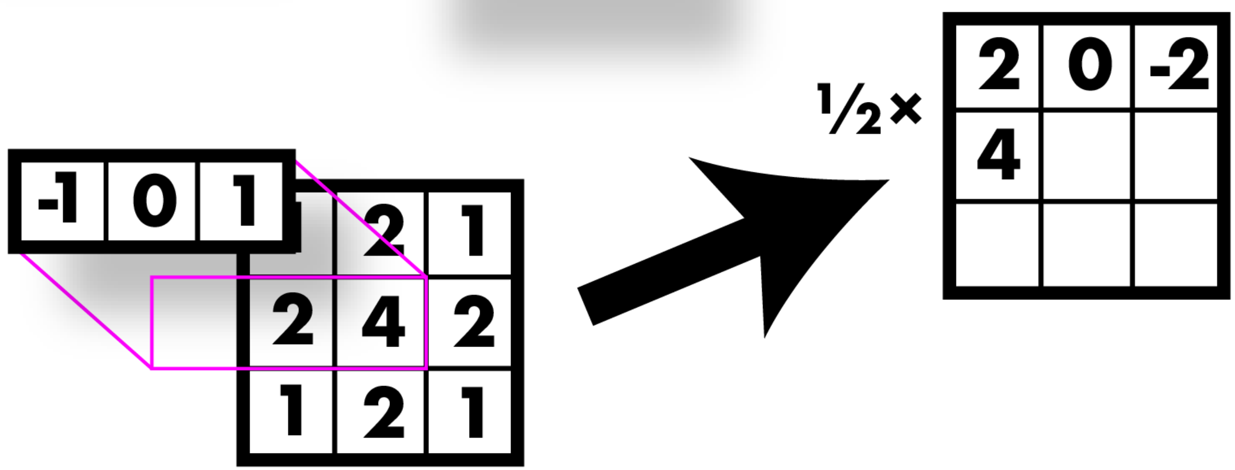


# Smooth first, then derivative

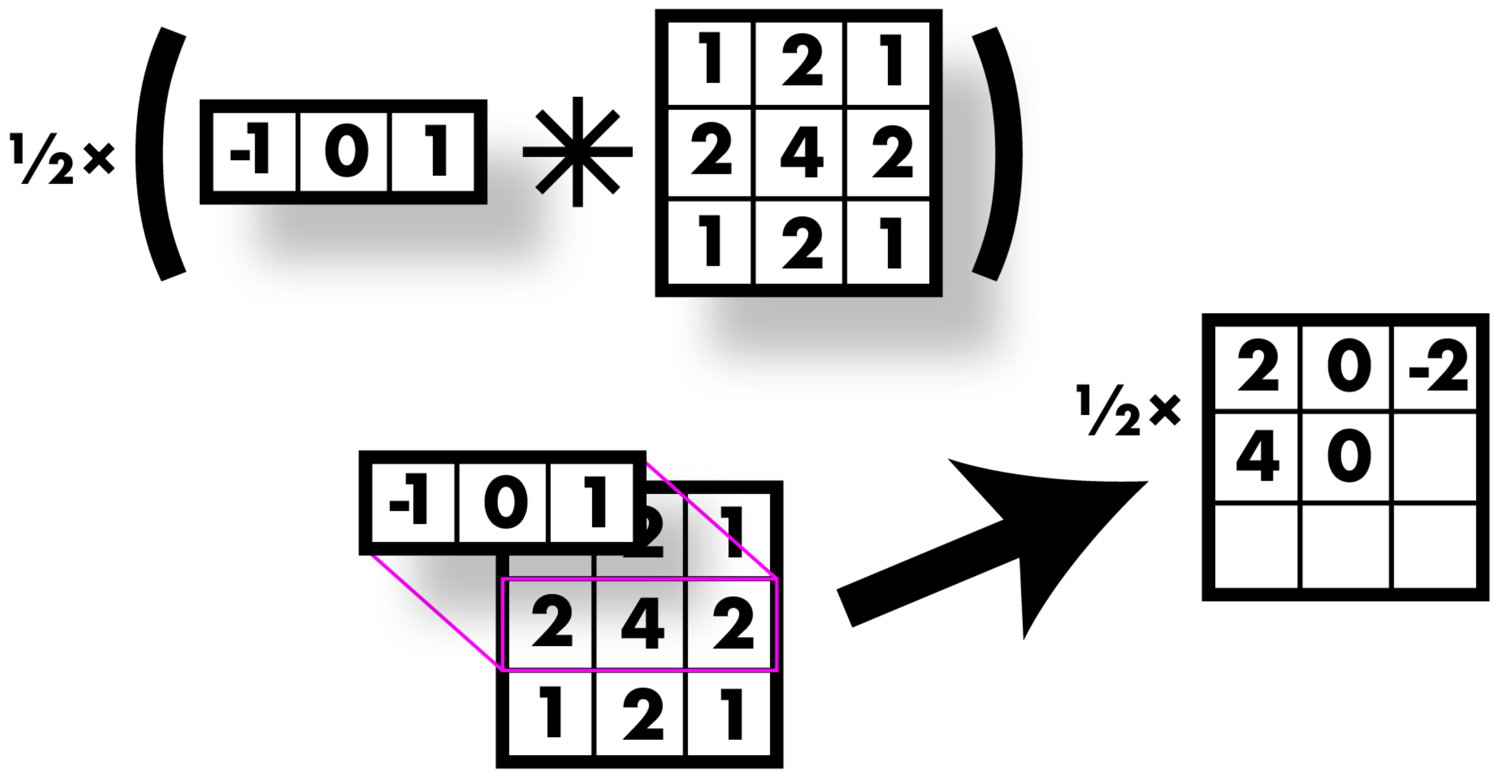


# Smooth first, then derivative

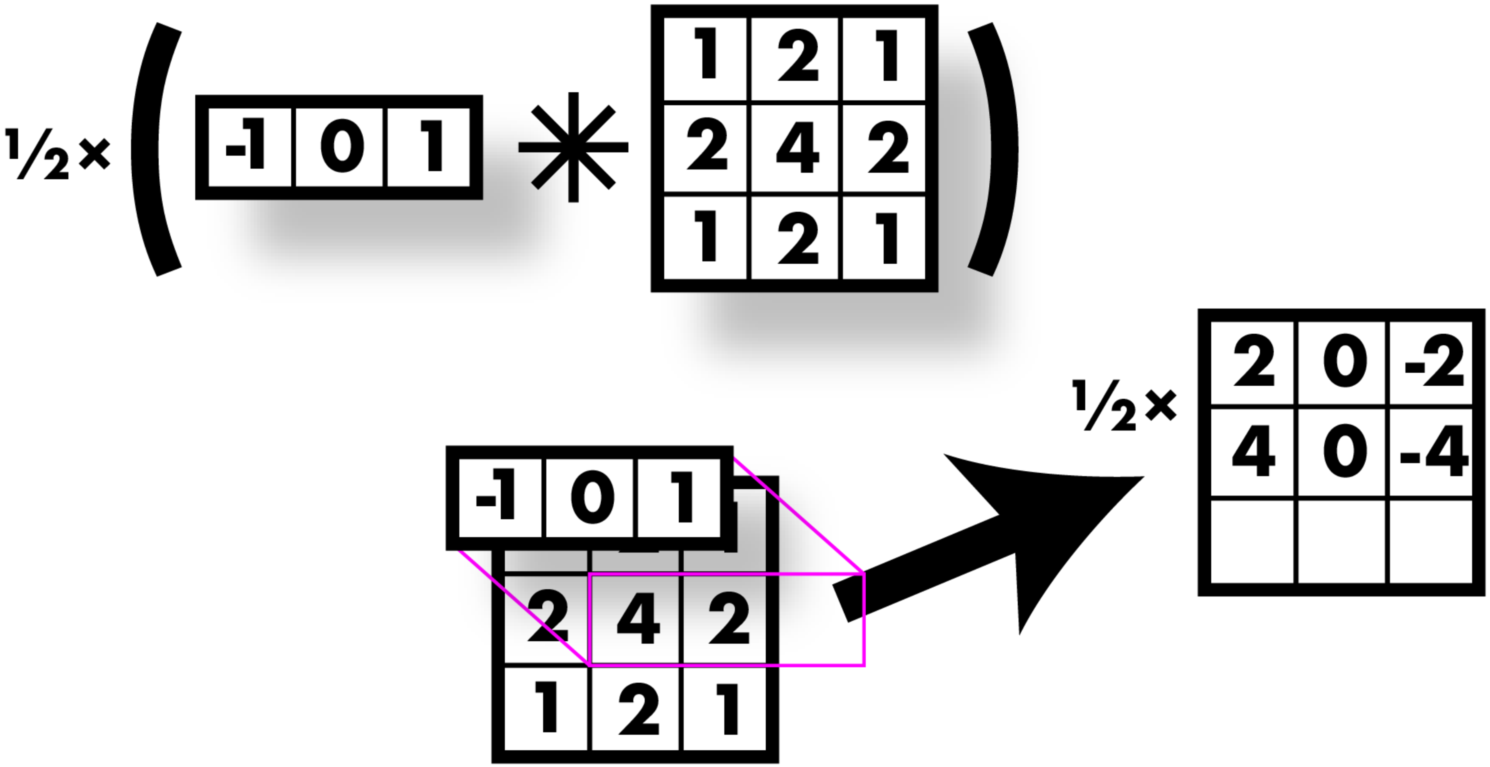


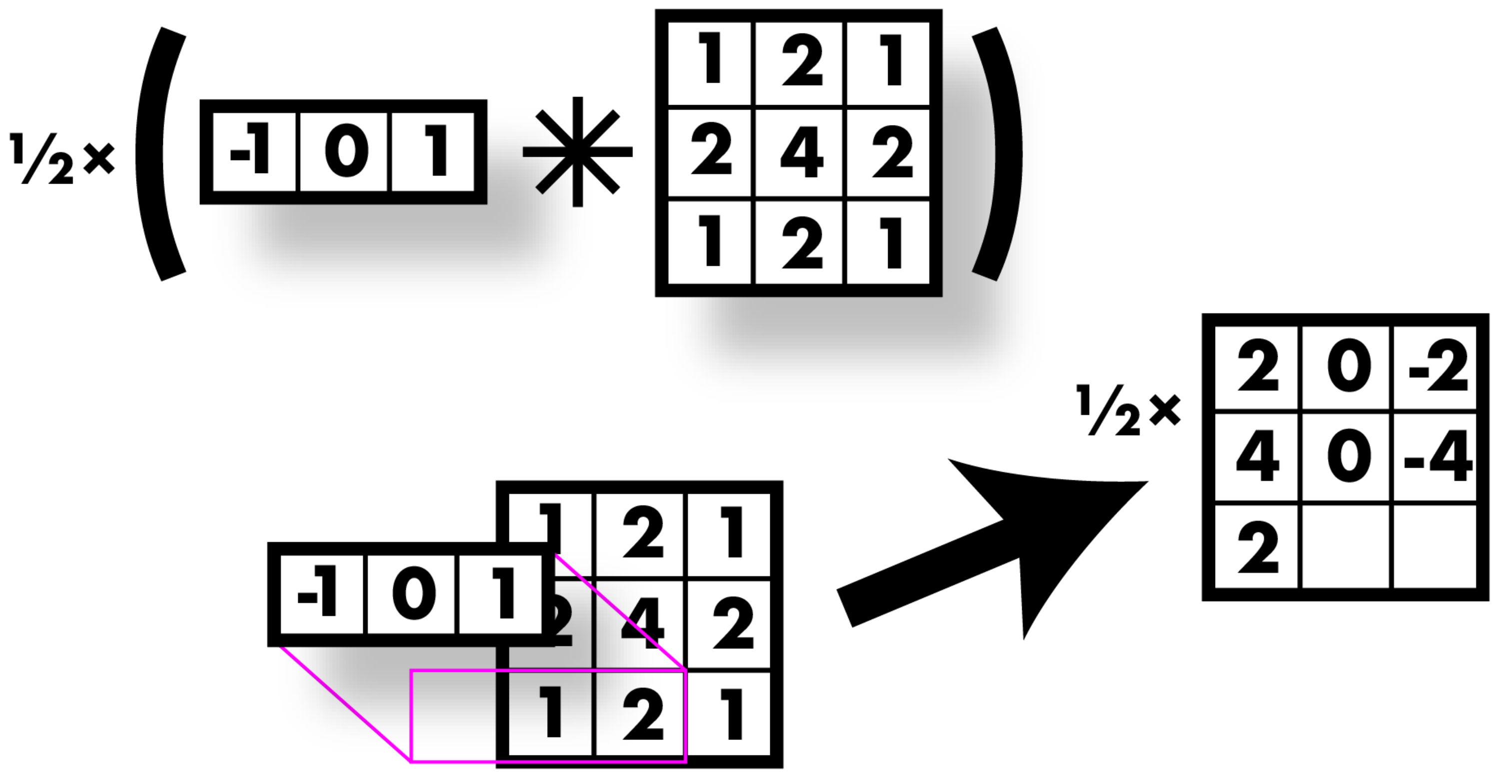
# Smooth first, then derivative



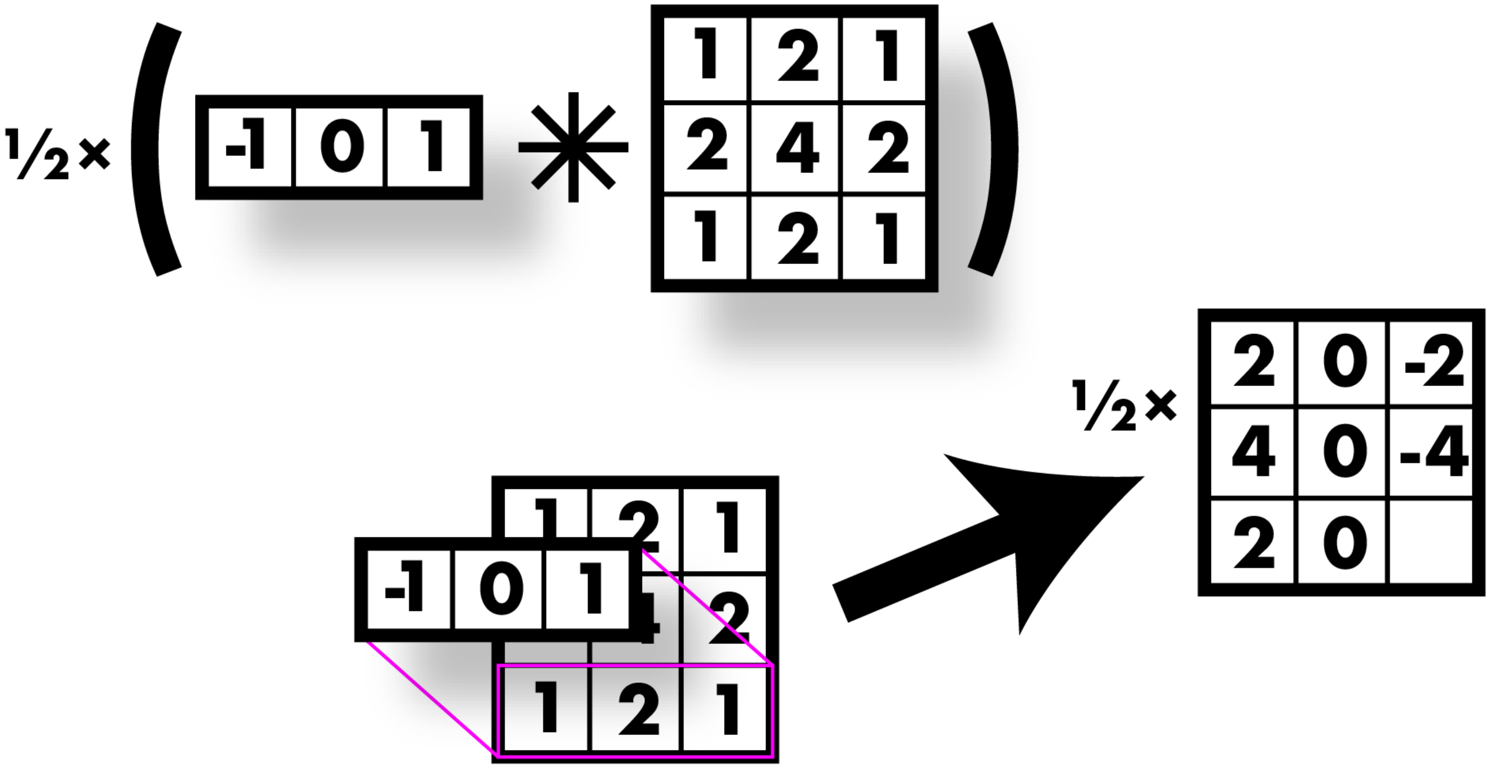
# Smooth first, then derivative



# Smooth first, then derivative

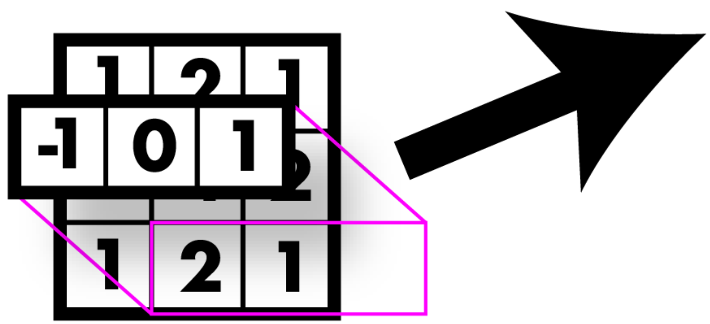


# Smooth first, then derivative



# Smooth first, then derivative

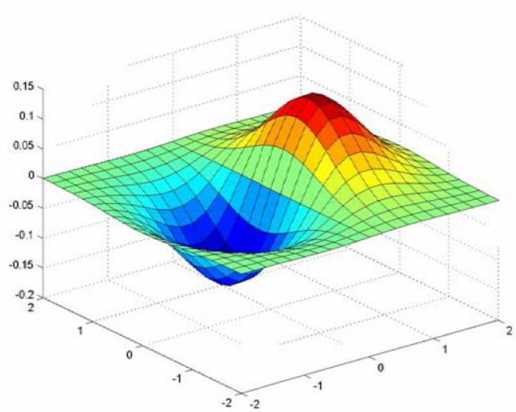
$$\tfrac{1}{2}\times\left(\boxed{\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline\end{array}} * \boxed{\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline\end{array}}\right)$$

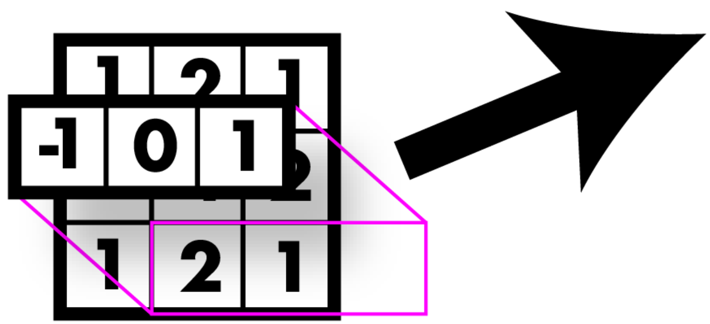



$$\tfrac{1}{2}\times\begin{array}{|c|c|c|}\hline 2 & 0 & -2 \\ \hline 4 & 0 & -4 \\ \hline 2 & 0 & -2 \\ \hline\end{array}$$

# Sobel filter! Smooth & derivative

$$\frac{1}{2}\times\left(\begin{bmatrix}-1 & 0 & 1\end{bmatrix} * \begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}\right)$$



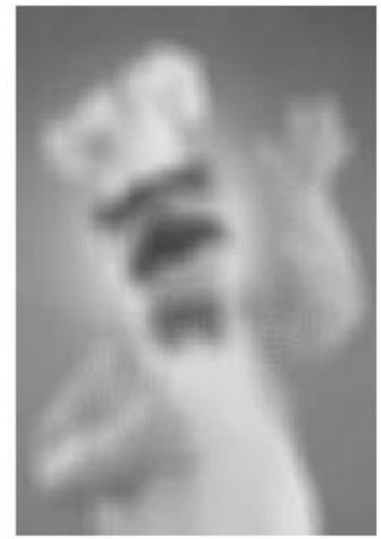

$$\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix}$$

# Convolution numérique

- **Caractéristiques du masque de convolution**
  - Souvent carré et de taille impaire (3x3, 5x5, ...) pour être centré sans ambigüité sur le pixel d'analyse
  - Souvent à valeurs symétriques par rapport à l'élément central
- **Normalisation**
  - Soit S la somme des coefficients du masque
  - Si l'on veut conserver la luminance de l'image, on doit avoir S = 1.
  - On doit donc diviser les coefficients par | S |.
- **Obtention de valeurs dans [0, 255]**
  - Les coefficients peuvent être négatifs, et le résultat de la convolution également
  - Un décalage est donc parfois nécessaire (après calcul du résultat) pour obtenir des valeurs entre 0 et 255

# Convolution numérique

- **Il existe différents types de filtres, avec différents effets**
- **Un filtre est caractérisé par**
  - sa taille (nombre de pixels voisins considérés)
  - son contenu (opération réalisée sur les pixels voisins considérés)



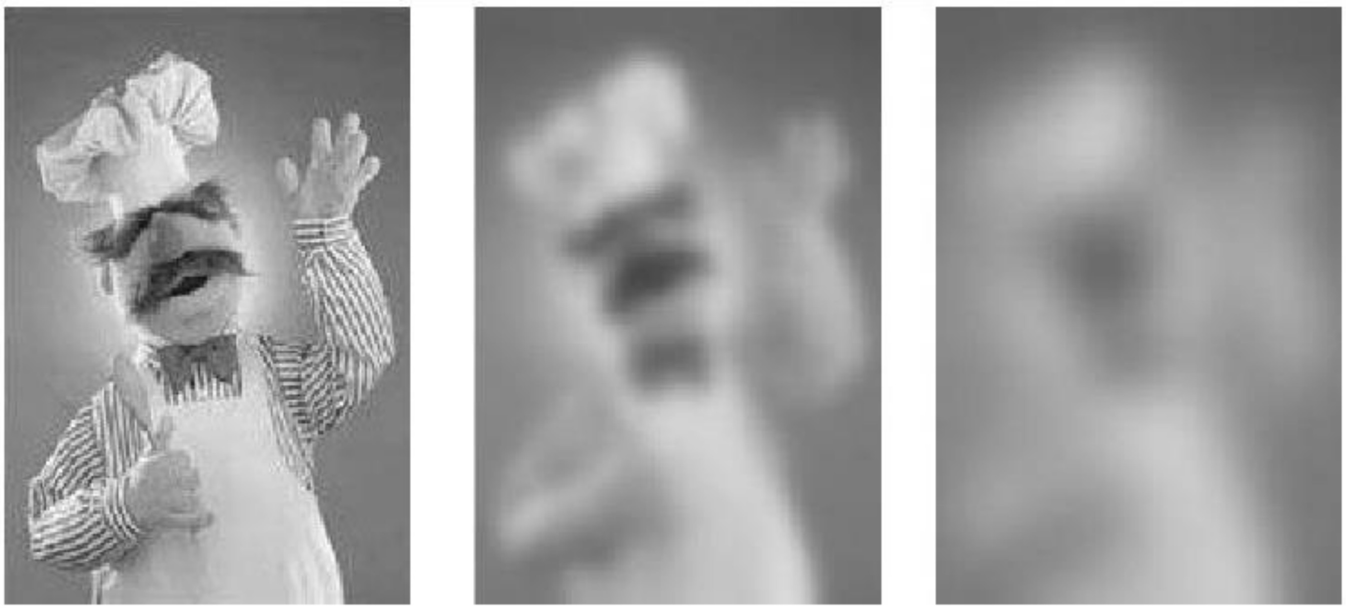
- **Types de filtrage spatial :**
  - **Filtres passe-bas ou de lissage**
    - Effet : lissage de l'image (élimine petites fluctuations)
    - Avantage : atténuation du bruit
    - Inconvénient : atténuation des détails, flou

  - **Filtres passe-haut ou de contours**
    - Effet : accentuation des détails de l'image
    - Avantage : mise en évidence des contours/détails
    - Inconvénient : accentuation du bruit

# 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**

$$\frac{1}{K^2}\begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \dots & \dots & \dots & \dots \\ 1 & 1 & \dots & 1 \end{pmatrix}$$



Original — Moyenne 5x5 — Moyenne 11x11

$$\begin{bmatrix} 0 & \frac{1}{5} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ 0 & \frac{1}{5} & 0 \end{bmatrix} \quad \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

connexité 4 — connexité 8

# Le filtre binomial

- Filtre dont tous les coefficients **ne sont pas égaux**

  – Moyenne pondérée des voisins ➔ Approximation du filtre de Gauss

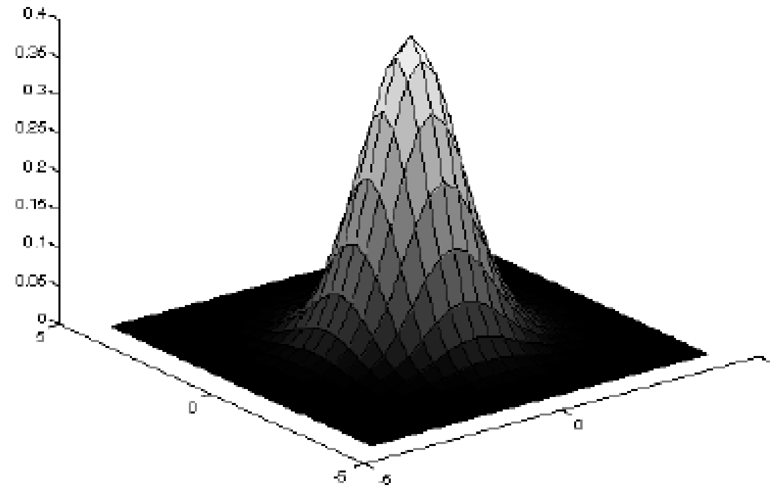
$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$\frac{1}{64}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

# Le filtre Gaussien

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

$$h(x,y)=\frac{1}{2\pi\sigma^2}\exp\left(\frac{-(x^2+y^2)}{2\sigma^2}\right)$$

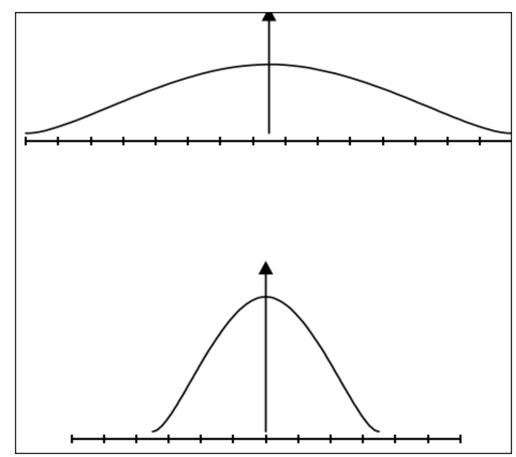



Avantages :

- Filtre paramétrable (sigma = ? , µ = 0)
- Adaptation au problème

Inconvénients :

- Difficulté du choix de taille de la fenêtre et valeur de sigma
- Complexité (calcul flottant vs entier)



# Le filtre Gaussien

- Mise en place du masque
  - (x,y) = (0,0) ← centre du masque
  - Avec la formule précédente
  - Normalisation pour obtenir des valeurs entières

Paramètres :

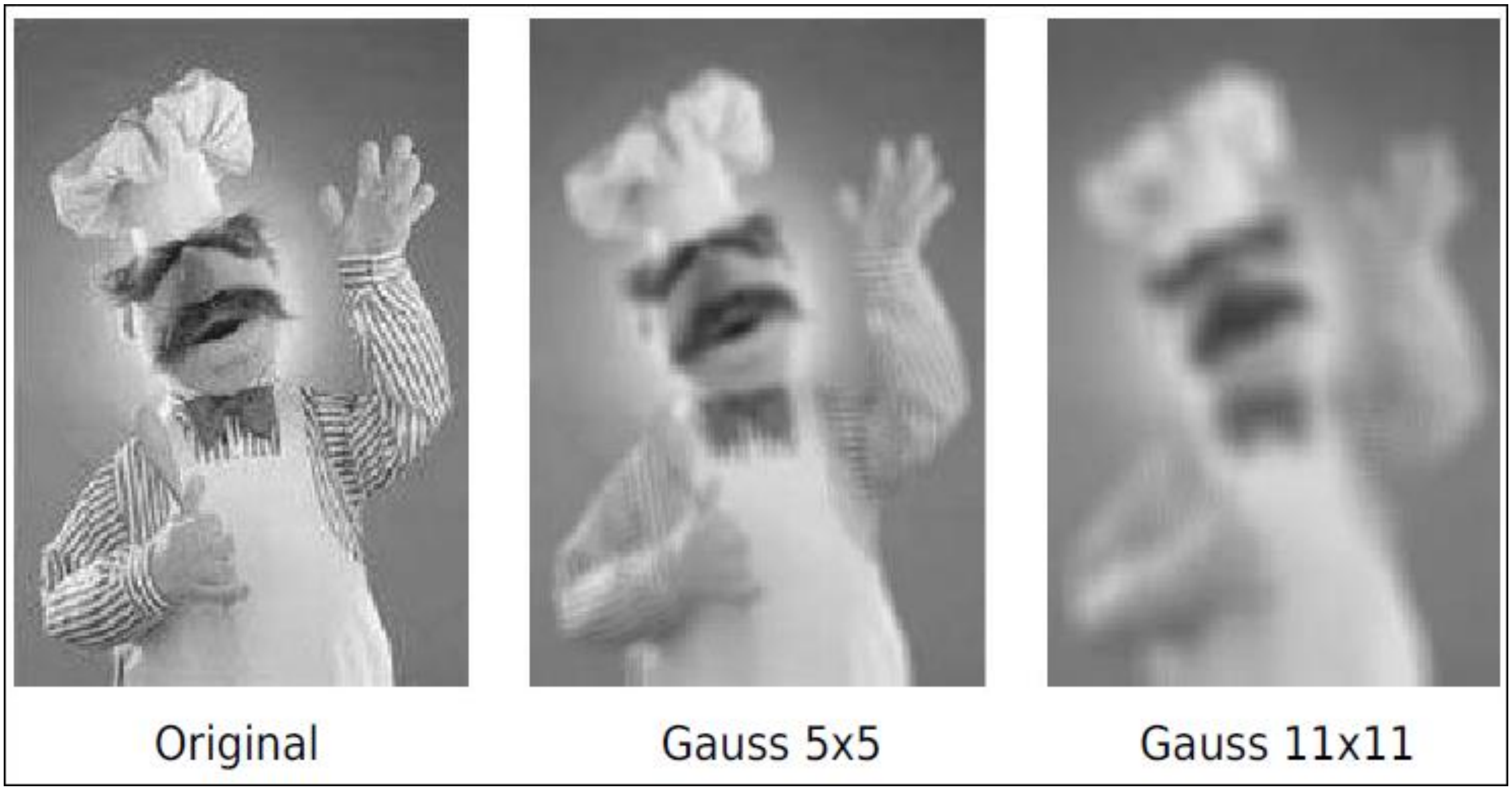
Taille du masque ($2\sigma+1 \times 2\sigma+1$).

$\mu = 0$ $\sigma = 1.0$ , $\sigma = 1.6$

$$\frac{1}{1344}\begin{pmatrix} 4 & 18 & 29 & 18 & 4 \\ 18 & 80 & 132 & 80 & 18 \\ 29 & 132 & 218 & 132 & 29 \\ 18 & 80 & 132 & 80 & 18 \\ 4 & 18 & 29 & 18 & 4 \end{pmatrix} \qquad \frac{1}{1279}\begin{pmatrix} 2 & 7 & 12 & 14 & 12 & 7 & 2 \\ 7 & 18 & 32 & 38 & 32 & 18 & 7 \\ 12 & 32 & 57 & 69 & 57 & 32 & 12 \\ 14 & 38 & 69 & 84 & 69 & 38 & 14 \\ 12 & 32 & 57 & 69 & 57 & 32 & 12 \\ 7 & 18 & 32 & 38 & 32 & 18 & 7 \\ 2 & 7 & 12 & 14 & 12 & 7 & 2 \end{pmatrix}$$

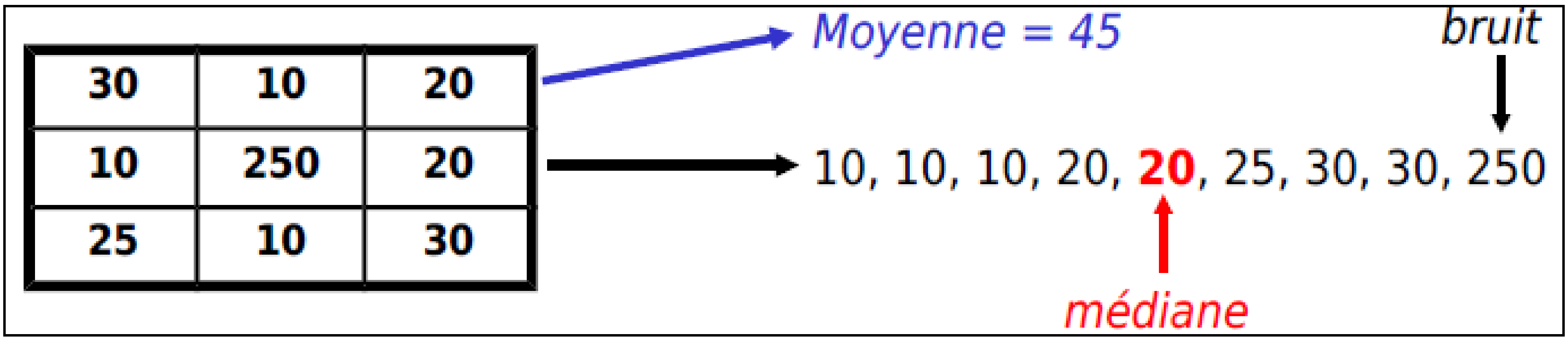
# Le filtre Gaussien

- Le filtre gaussien donnera un meilleur lissage et une meilleure réduction du bruit que le filtre moyenneur



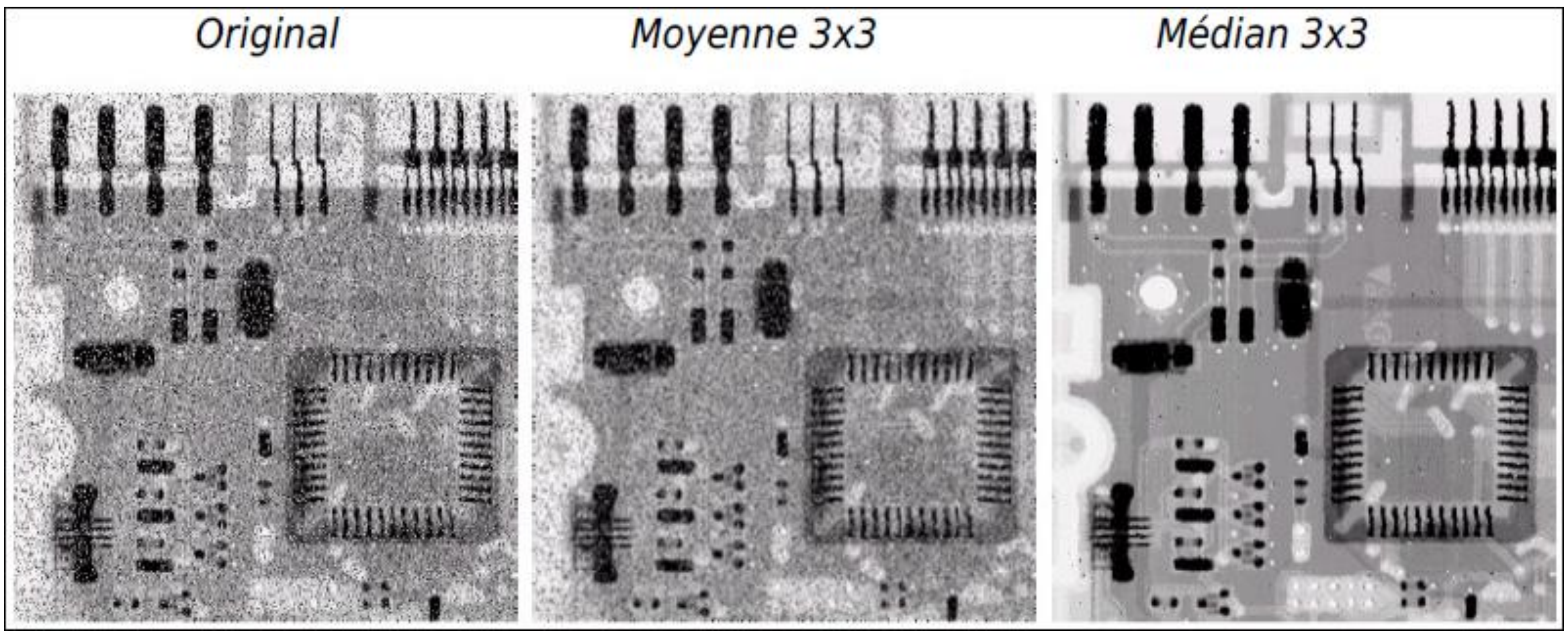
# Le filtre Médian *(non-linéaire)*

- 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



# Le filtre Médian *(non-linéaire)*

- **Propriétés du filtre médian** :
  - Non création de niveaux de gris
  - Invariance par étirement de contraste
  - Préservation des marches et rampes rectilignes
  - Érosion des connexités (notamment des disques)
  - Non convergence de la répétition (possibilité de limite alternée)
  - Élimination du bruit impulsif



# Filtrage non linéaire : filtres de rang

- Généralisation du filtre médian
- Principe : au lieu du niveau médian, on choisit le niveau d'un rang donné

- Exemple (rang 5) :

| | | |
|---|---|---|
| 125 | 99 | 210 |
| 47 | 5 | 22 |
| 243 | 72 | 186 |

- Trier les 9 valeurs : 5 < 22 < 47 < 72 < 99 < 125 < 186 < 210 < 243
- Sélectionner la 5$^{ème}$ : 99

# Filtrage non linéaire : filtre Min-Max

**Filtrage « Conservative smoothing »**

- Principe : garantit que la valeur de tout pixel appartient à l'intervalle des valeurs de ses voisins
- Débruitage efficace
- Préserve encore mieux les contours que le filtre médian
- Calcul :

$$I'(x,y)=\begin{cases} I(x,y) & \text{si } i_{min}\leqslant I(x,y)\leqslant i_{max} \\ i_{min} & \text{si } I(x,y)<i_{min} \\ i_{max} & \text{si } I(x,y)>i_{max} \end{cases}$$

Exemple :

| | | |
|---|---|---|
| 124 | 126 | 127 |
| 120 | 150 | 125 |
| 115 | 119 | 123 |

*imin*=115, *imax*=127

moyenne = 125
moyenne8 = 123
médiane = 124
(min-)max = 127

# Petit contrôle …

- Quels sont les principales méthodes de traitement d’images exploitant l’histogramme ?
- Qu’est ce qu’une opération de convolution sur une image ?
- Qu’est ce qu’un filtrage passe-bas ? Donner des exemples ?
- Qu’est ce qu’un filtrage passe-haut ? Donner des exemples ?

# Tous le monde sait appliquer un masque de convolution ?

Soit le filtre défini par le masque de convolution 2D suivant :

$$H = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- Calculer l'image résultant de l'application de ce filtre sur chacune des images suivantes :

$I_1 =$

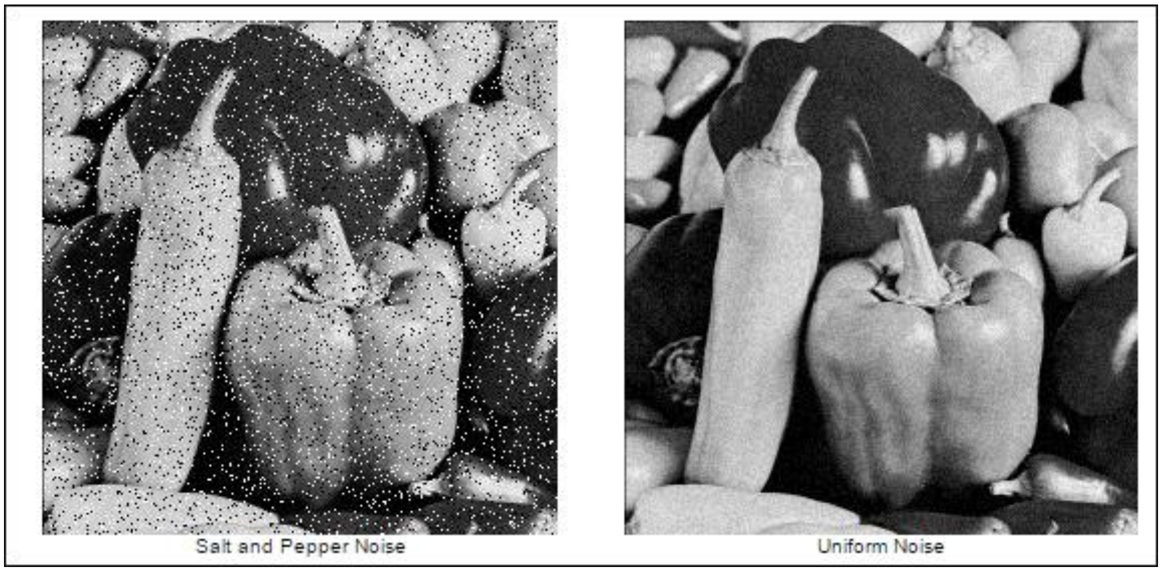
| y\x | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 180 | 180 | 180 |
| 1 | 0 | 0 | 0 | 0 | 180 | 180 | 180 |
| 2 | 0 | 0 | 0 | 0 | 180 | 180 | 180 |
| 3 | 0 | 0 | 0 | 0 | 180 | 180 | 180 |
| 4 | 0 | 0 | 0 | 0 | 180 | 180 | 180 |
| 5 | 0 | 0 | 0 | 0 | 180 | 180 | 180 |
| 6 | 0 | 0 | 0 | 0 | 180 | 180 | 180 |

$I_2 =$

| y\x | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 180 | 180 | 180 | 0 | 0 |
| 3 | 0 | 0 | 180 | 180 | 180 | 0 | 0 |
| 4 | 0 | 0 | 180 | 180 | 180 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

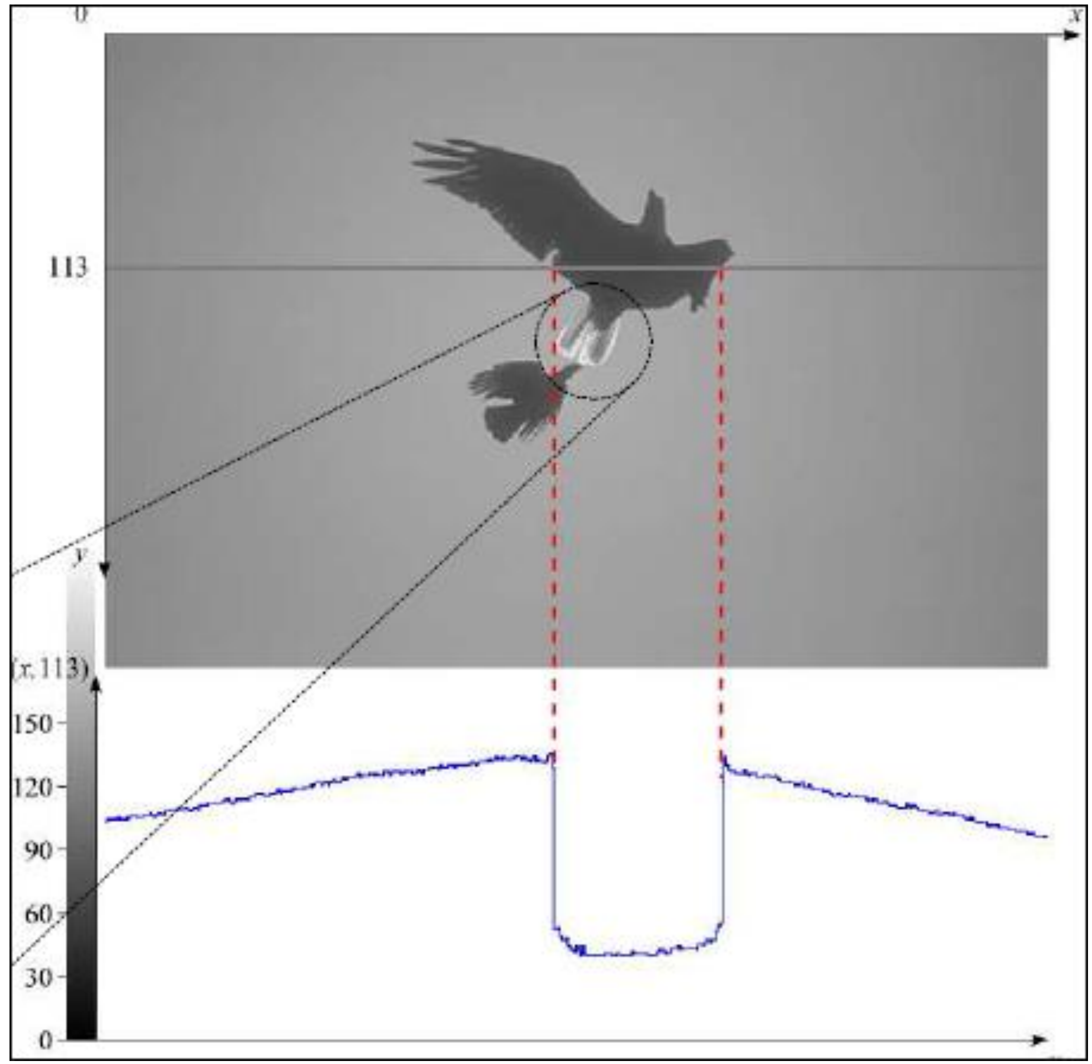
# Type de bruits ➔ Type de traitement ?

Choix du filtrage à effectuer en fonction du type de bruit présent dans l'images




# Notion de contours

Question : Comment détecter les frontières des régions / objets ?



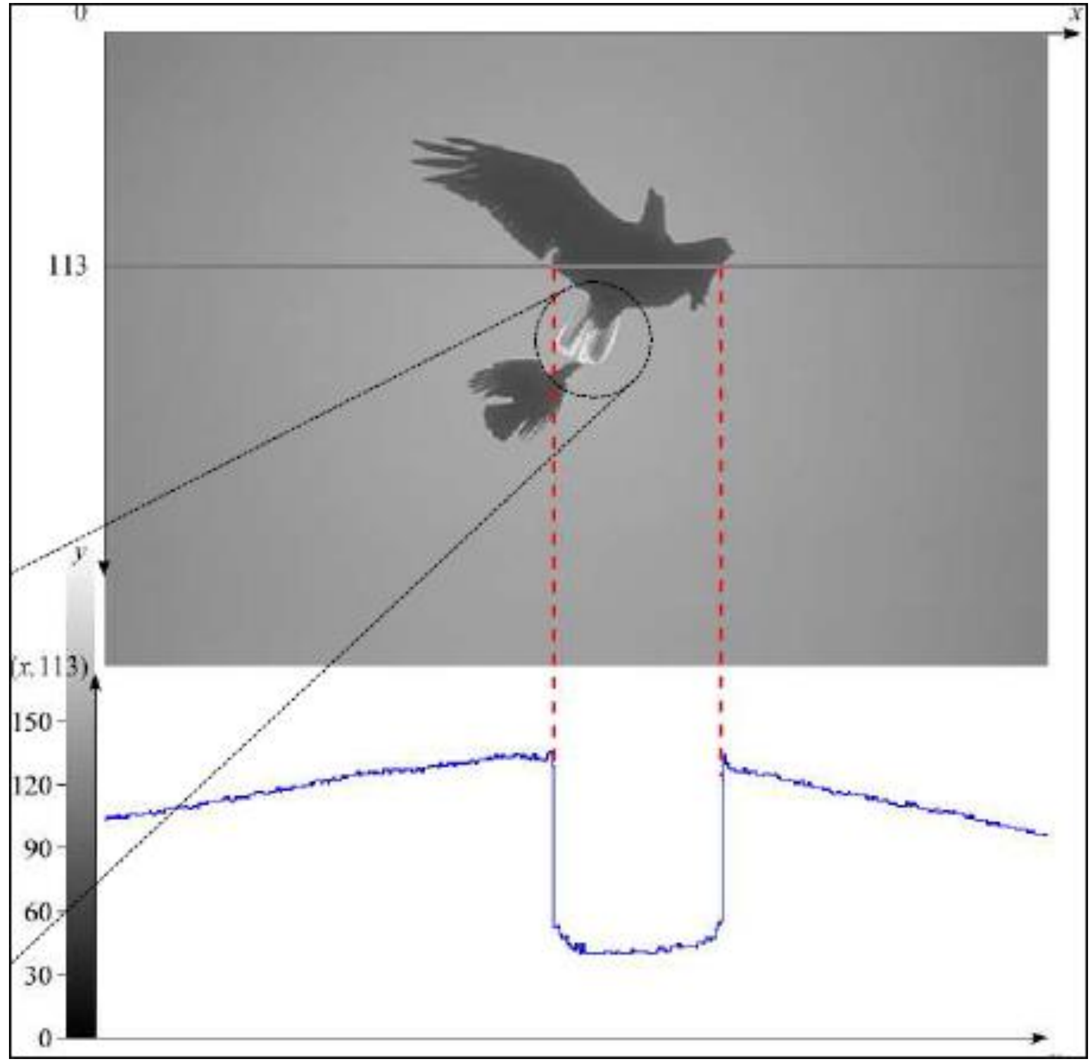
# Notion de contours

## Définition

- Frontière qui sépare 2 objets (ou un objet du fond) dans une image

## Caractérisation des zones de contours

- Variation brusque de l'intensité (discontinuité)
- *Remarque* : toute zone de discontinuité ne caractérise pas forcément un contour

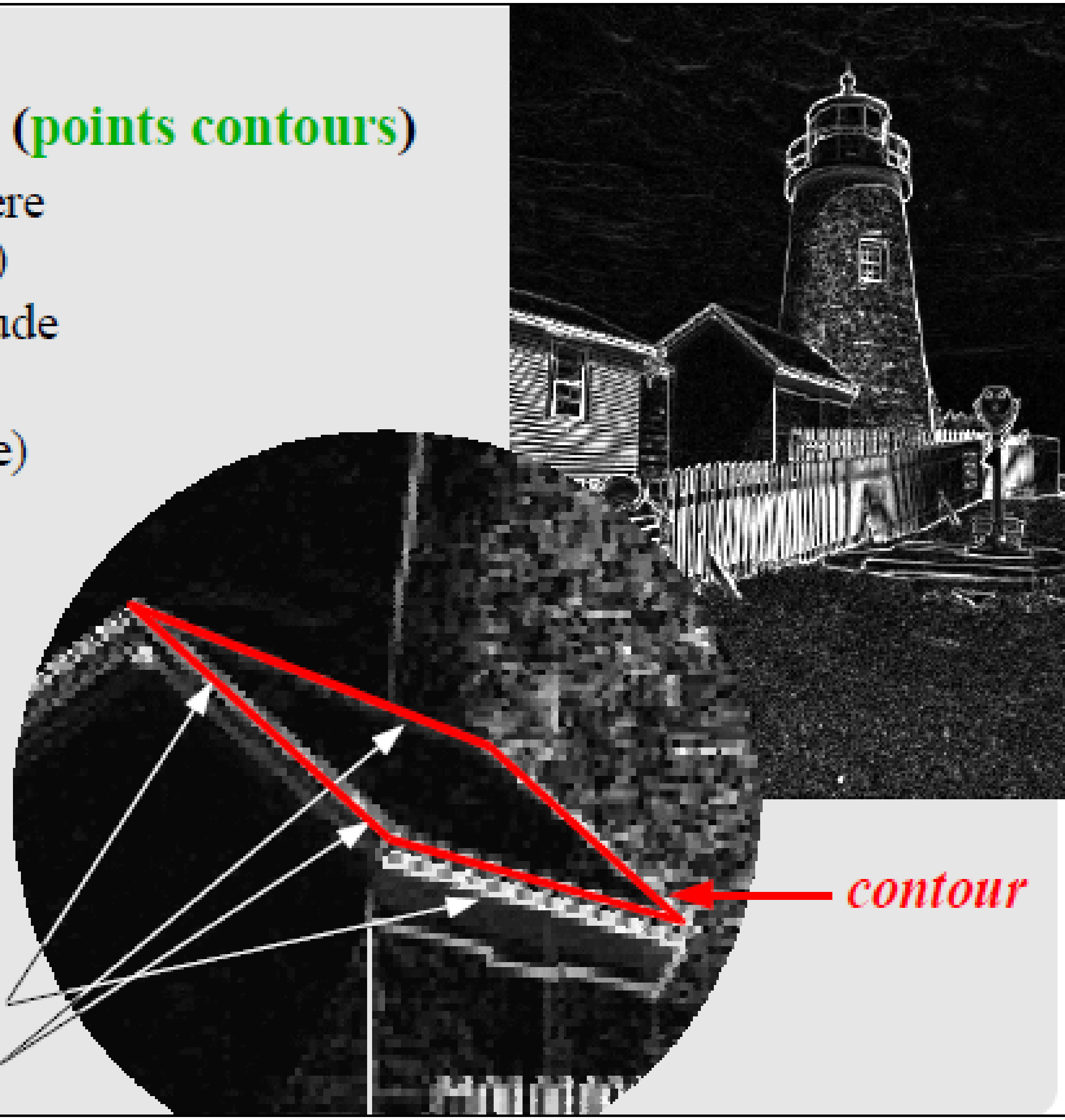


# Notion de contours

- **Exemple de zones de discontinuité de l'intensité lumineuse**

# Notion de contours

- « Détection » de « contours » ?
  - **Détection des pixels candidats (points contours)**
    - grâce à une propriété particulière (*ex.* discontinuité de l'intensité)
    - avec un certain degré de certitude
    - perturbée par le bruit (⇒ lissage préalable nécessaire)
  - **Formation des contours**
    - relier les points contours (par analyse de connexité ou autre)
    - obtention de <u>contours</u> à proprement parler (courbes = chaînes fermées de pixels)

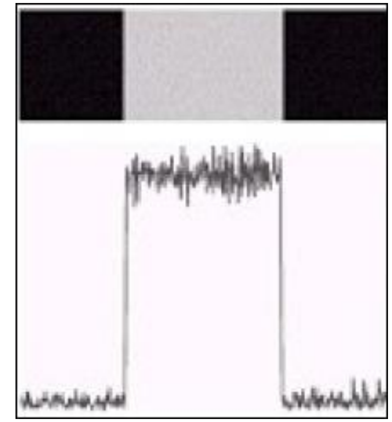


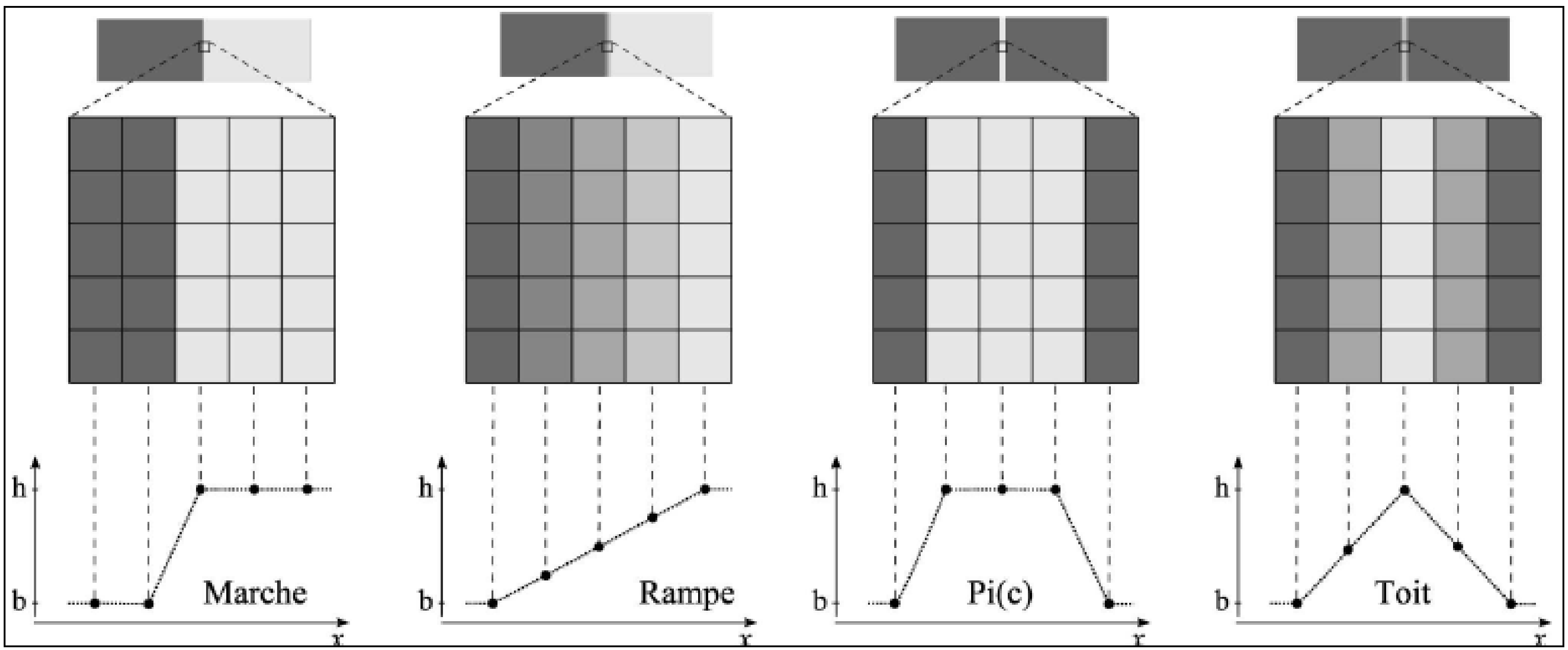
# Notion de contours

## Caractéristiques d'une zone de contour

- Transition entre deux niveaux très différents.
- Paramètres : largeur, hauteur (contraste)
- Types de profils (théoriques) :

Bruit ! ➔

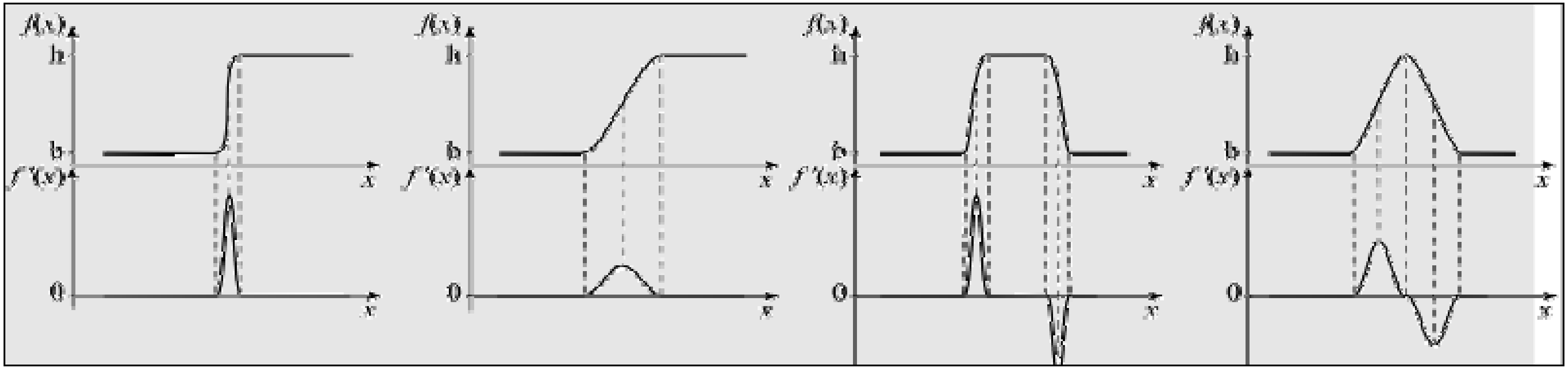




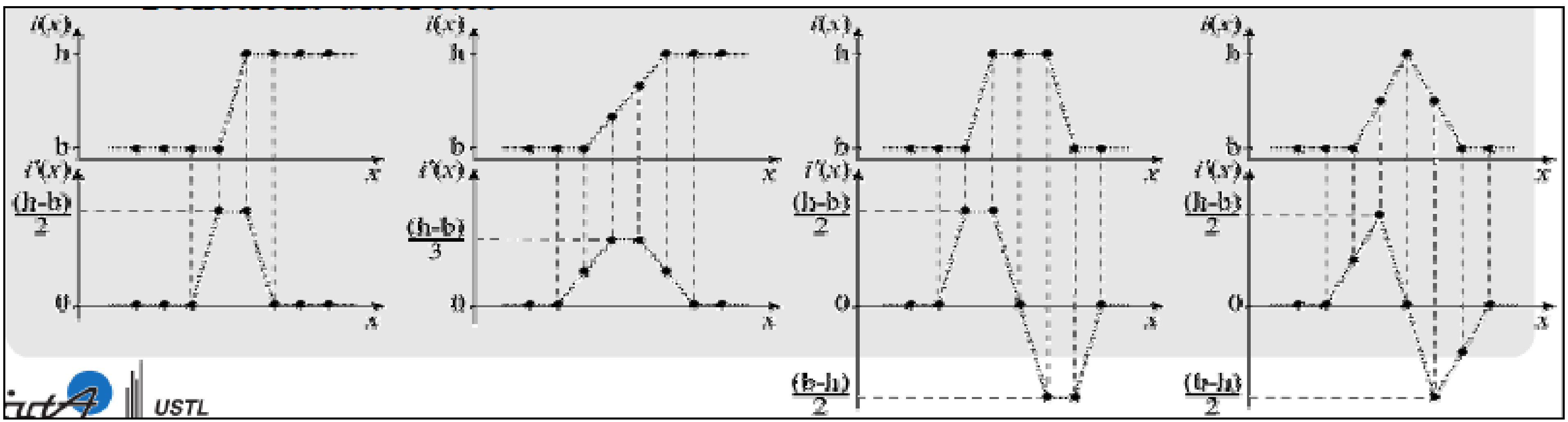
# Détection de contours

## Mise en évidence des zones de contours : dérivée première

- Fonctions continues




- Fonctions discrètes

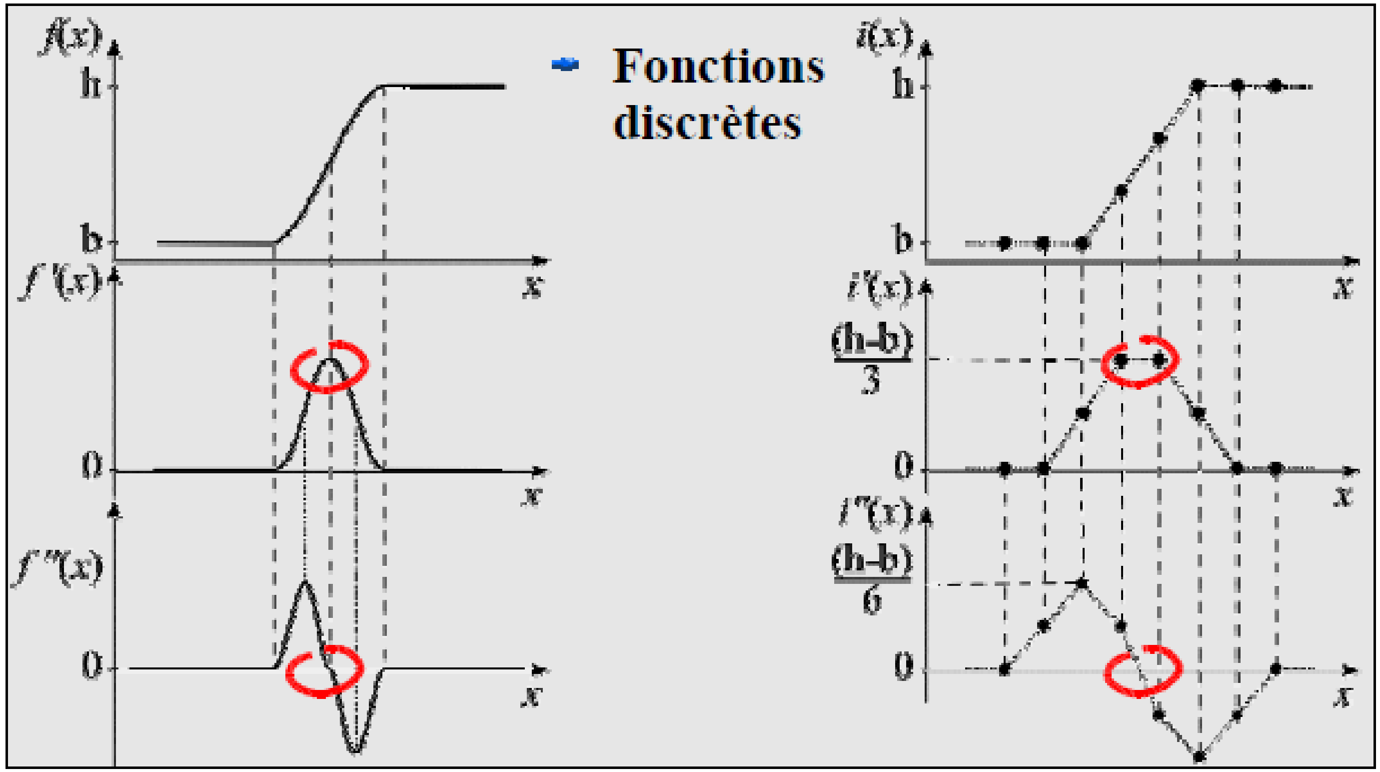



USTL

# Détection de contours

## Mise en évidence des zones de contours : dérivée seconde

Fonctions continues ➔

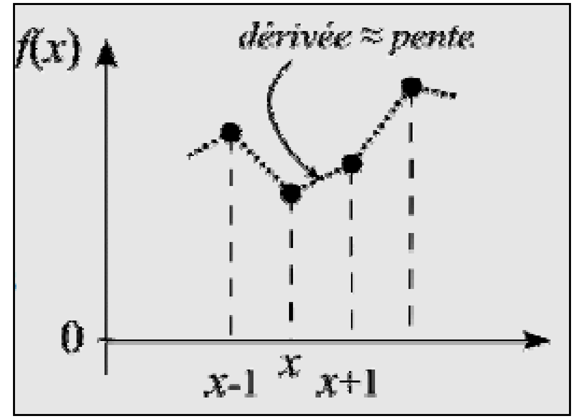



Détection des points contours : utilisation d'un critère de décision

- Dérivée première : maxima locaux
- Dérivée seconde : passages par zéro

# Détection de contours et gradient

## Notion de gradient : Dérivée première en 1D




- Dérivée d'une fonction 1D continue :

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- Approximations de la dérivée d'une fonction discrète 1D par différences locales

$$f'(x) \approx f(x+1) - f(x)$$

ou $$f'(x) \approx f(x) - f(x-1)$$

ou $$f'(x) = \frac{1}{2}\left(f'(x^+) + f'(x^-)\right) \approx \frac{f(x+1) - f(x-1)}{2}$$

- Masques de convolution 1D correspondant

| | |
|---|---|
| +1 | -1 |

ou

| | |
|---|---|
| +1 | -1 |

ou $\frac{1}{2}$

| | | |
|---|---|---|
| +1 | 0 | -1 |

meilleure approximation

# Détection de contours et gradient

## Notion de gradient : Dérivée première en 2D

- *L*'image (discrète) *I* est définie comme un ensemble de points d'échantillonnage de la fonction bidimensionnelle sous-jacente $f(x,y)$

## Dérivée 2D de la fonction sous-jacente

- On peut calculer une dérivée (partielle) de *f* dans chaque direction principale ➔

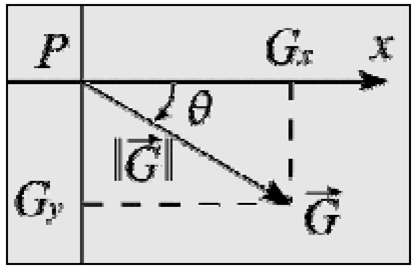
$$\frac{\partial f(x,y)}{\partial x} \quad \text{et} \quad \frac{\partial f(x,y)}{\partial y}$$

- Leur combinaison forme le vecteur gradient, à 2 composantes ➔

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

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

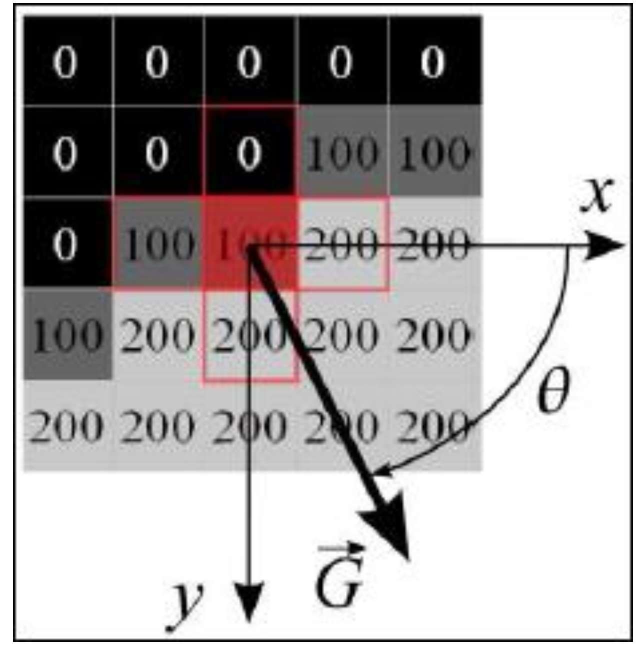
- Ce vecteur est caractérisé, en chaque point *P*, par
  - une norme (ou module) $\|G\| = \mathrm{sqrt}(Gx^2 + Gy^2)$
  - une direction $\Theta = \arctan(Gy / Gx)$



# Détection de contours et gradient

**Dérivée première en 2D (cas discret)**

- **Propriétés fondamentales du vecteur gradient**
  - Le module du vecteur gradient représente la pente de la surface image en *P* :
  - module élevé = forte variation au voisinage de *P*
  - La direction du vecteur gradient correspond à celle de la plus grande pente en *P*
  - Le vecteur est orienté dans le sens de la montée (*i.e.* niveaux de gris croissants)




- **Relation entre gradient et contour**
  - Contour = forte variation locale NdG = $\|\vec{G}\|$ élevé
  - Le vecteur gradient $\vec{G}$ est perpendiculaire au contour

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

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

**Autres formules parfois utilisées (plus simples) :**

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

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

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

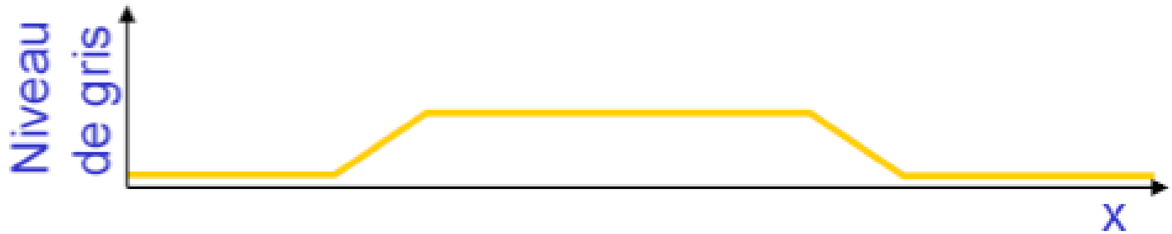
- **Masques associés**

$$G_x = \frac{\partial f}{\partial x} \;:\; \frac{1}{2}\begin{bmatrix} +1 & 0 & -1 \end{bmatrix}$$

$$G_y = \frac{\partial f}{\partial y} \;:\; \frac{1}{2}\begin{bmatrix} +1 \\ 0 \\ -1 \end{bmatrix}$$

# Détection de contours et seuillage

Image 1D f(x)

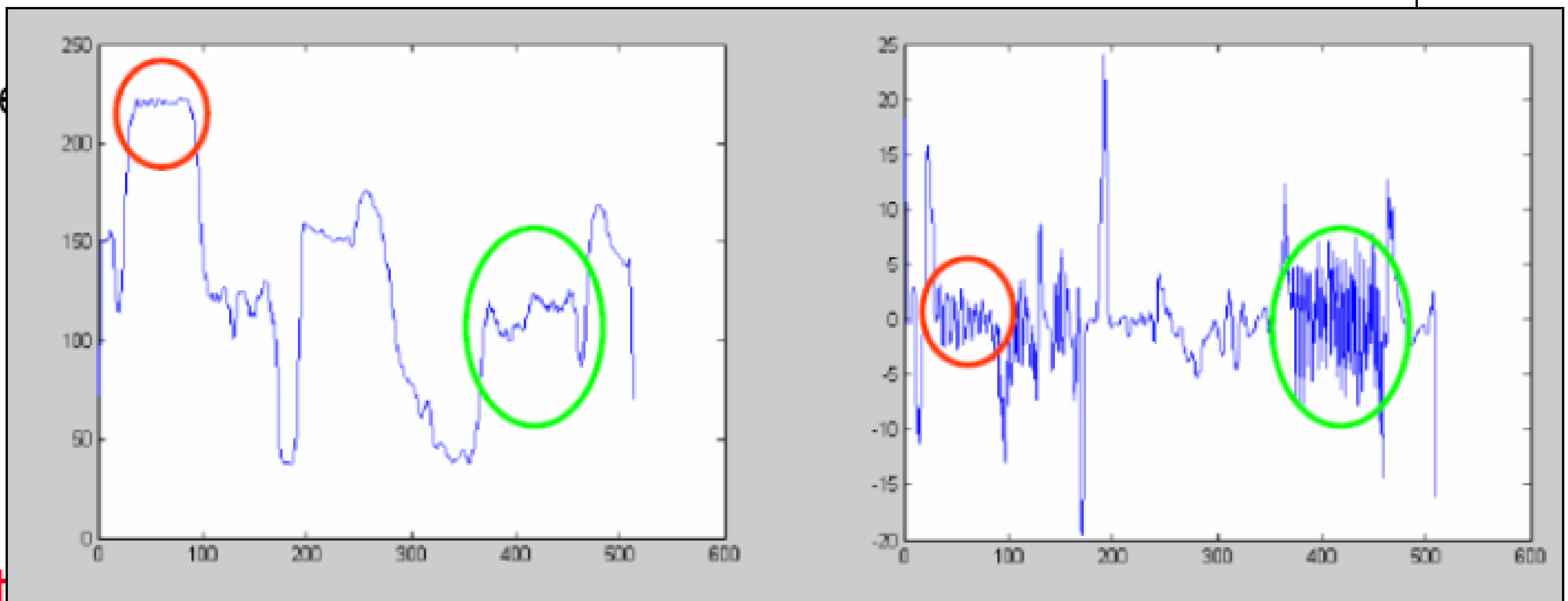



1ère dérive

|f'(x)|

Pixels cont

|f'(x)| >




signal lissé

dérivée première

# Détection de contours et gradient

**Principes des filtres de lissage/dérivation**

- Les effets du bruit sont amplifiés lors de la dérivation.
- Nécessité de lisser l'image
  - soit par un pré-traitement, avant dérivation
  - soit lors de la dérivation même
- Dérivation et lissage simultanés :
  - Principe : lissage dans la direction perpendiculaire à la dérivation
    - moyenne en colonnes de la dérivée calculée sur les lignes ;
    - moyenne en lignes de la dérivée calculée sur les colonnes.
  - On obtient des filtres de lissage/dérivation, moins sensibles au bruit

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

  - Plusieurs modèles ont été définis : Prewitt, Sobel, …

# Détection de contours et gradient

## Filtre de Prewitt : moyennage/dérivation

- Calcul de la composante horizontale du gradient *Gx*

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

- Calcul de la composante horizontale du gradient *Gy*

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

## Filtre de Sobel : Gaussien/dérivation

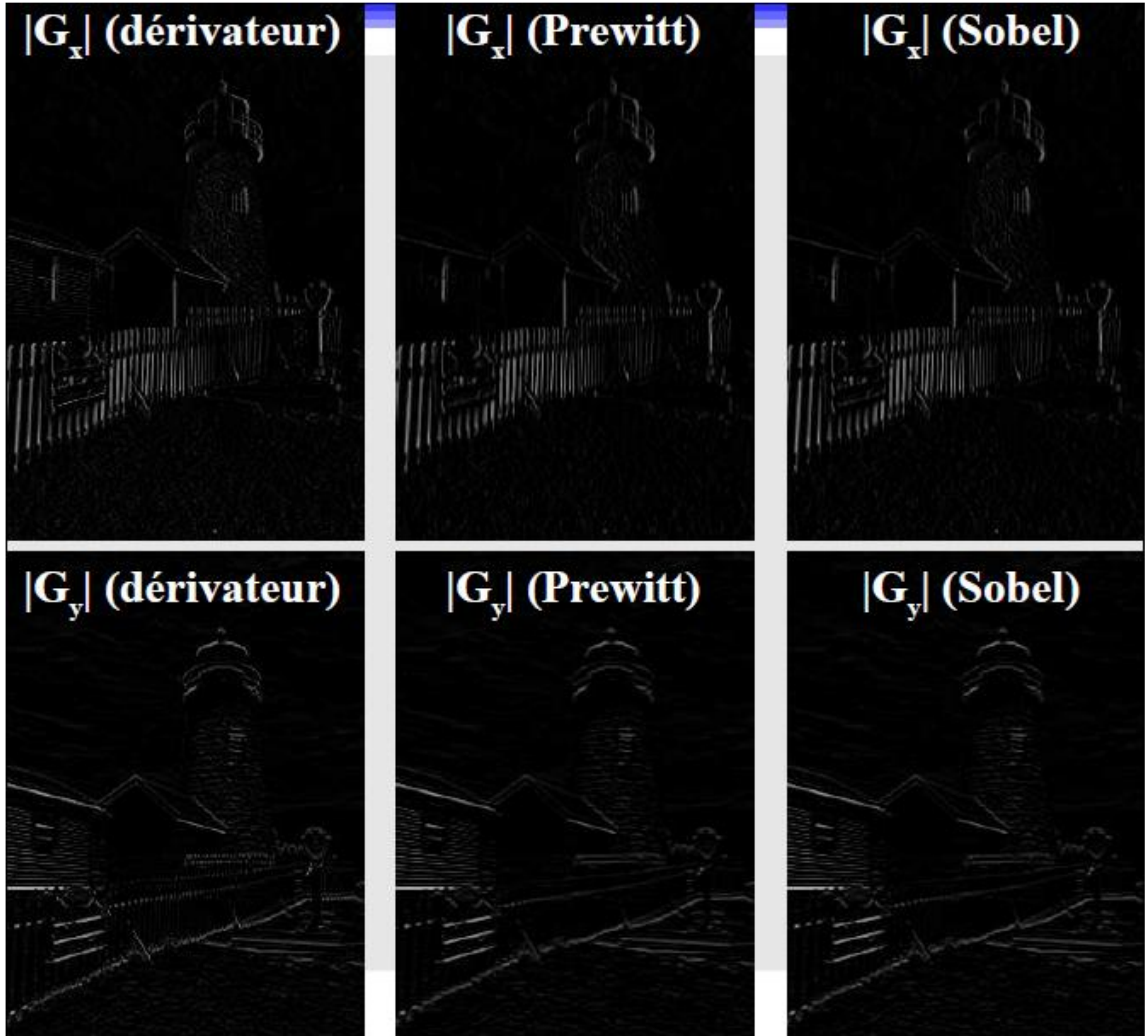
- Calcul de la composante horizontale du gradient *Gx*

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

- Calcul de la composante horizontale du gradient *Gy*

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

# Détection de contours et gradient

# Détection de contours et Laplacien

## Définition du Laplacien (dérivée seconde)

- Le Laplacien est défini par : $\Delta f = \dfrac{\partial^2 f}{\partial x^2} + \dfrac{\partial^2 f}{\partial y^2}$

- C'est une grandeur scalaire signée (et non vectorielle comme le gradient)

- **Masques associés aux dérivées secondes :**

  pour $\dfrac{\partial^2}{\partial x^2}$ : $\dfrac{1}{4}\begin{bmatrix} +1 & -2 & +1 \end{bmatrix}$ pour $\dfrac{\partial^2}{\partial y^2}$ : $\dfrac{1}{4}\begin{bmatrix} +1 \\ -2 \\ +1 \end{bmatrix}$

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

  pour $\dfrac{\partial^2}{\partial X^2}$ : $\dfrac{1}{4}\begin{bmatrix} 0 & 0 & +1 \\ 0 & -2 & 0 \\ +1 & 0 & 0 \end{bmatrix}$ pour $\dfrac{\partial^2}{\partial Y^2}$ : $\dfrac{1}{4}\begin{bmatrix} +1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & +1 \end{bmatrix}$

- **Approximations discrètes du Laplacien $\Delta$ :**

  $\dfrac{1}{8}\begin{bmatrix} 0 & +1 & 0 \\ +1 & -4 & +1 \\ 0 & +1 & 0 \end{bmatrix}$ ou $\dfrac{1}{8}\begin{bmatrix} +1 & 0 & +1 \\ 0 & -4 & 0 \\ +1 & 0 & +1 \end{bmatrix}$ ou encore $\dfrac{1}{16}\begin{bmatrix} +1 & +1 & +1 \\ +1 & -8 & +1 \\ +1 & +1 & +1 \end{bmatrix}$

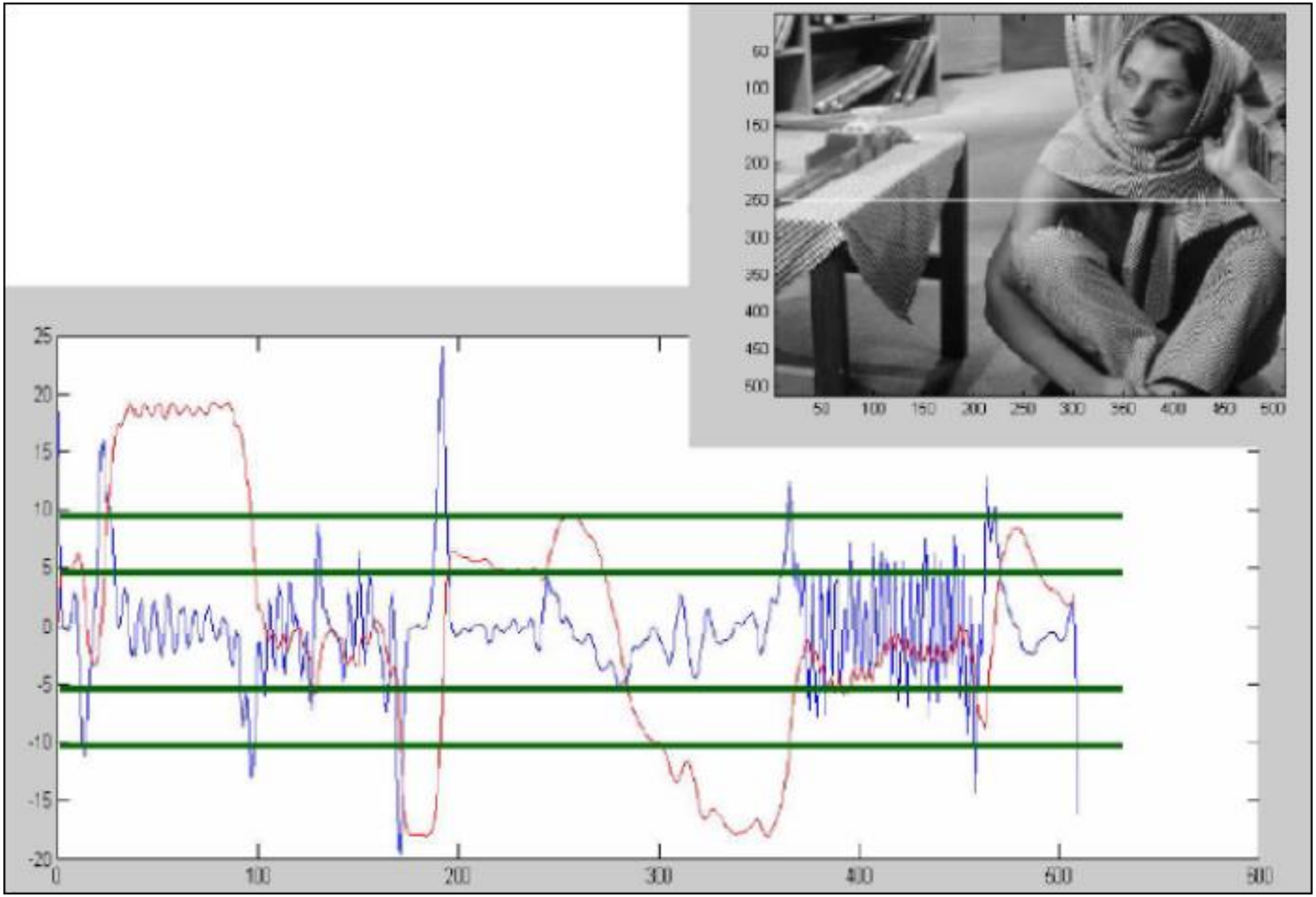
- Pour lutter contre le bruit, combinaison avec filtre gaussien possible
  - Laplacian of Gaussian ➔ LoG

# Détection de contours et Laplacien

Sobel Laplacien

# Détection de contours et bruit…

Optimisation du choix des maxima locaux à conserver ?



# Filtrage optimal : Canny

**Filtre en plusieurs étapes (pas seulement une convolution)**

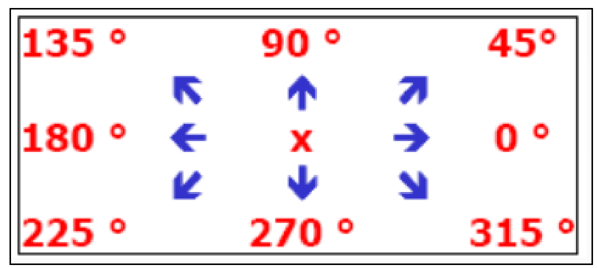
Etant donnés :

- un modèle de contour (marche)
- un modèle de bruit (blanc gaussien)

Caractériser les performances en termes de :

- détection (surtout pour les contours faibles)
- localisation (contour détecté proche du contour réel)

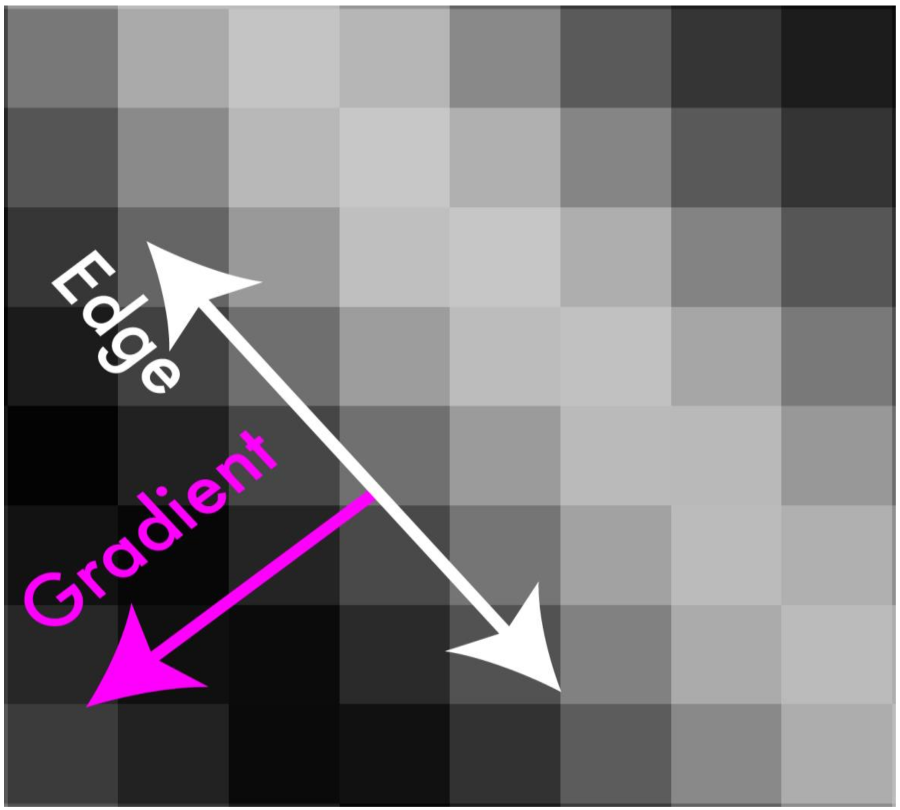
1. Appliquer un filtre Gaussien passe-bas pour enlever le bruit
2. Calculer l'intensité du gradient dans l'image par Sobel en X et Y
   - Calcul de la norme $|G| = |G_x| + |G_y|$
3. Calculer les directions du gradient dans l'image
   - Direction du gradient $\theta = \arctan(G_y / G_x)$
   - Arrondi des directions par multiples de $\pi/4$



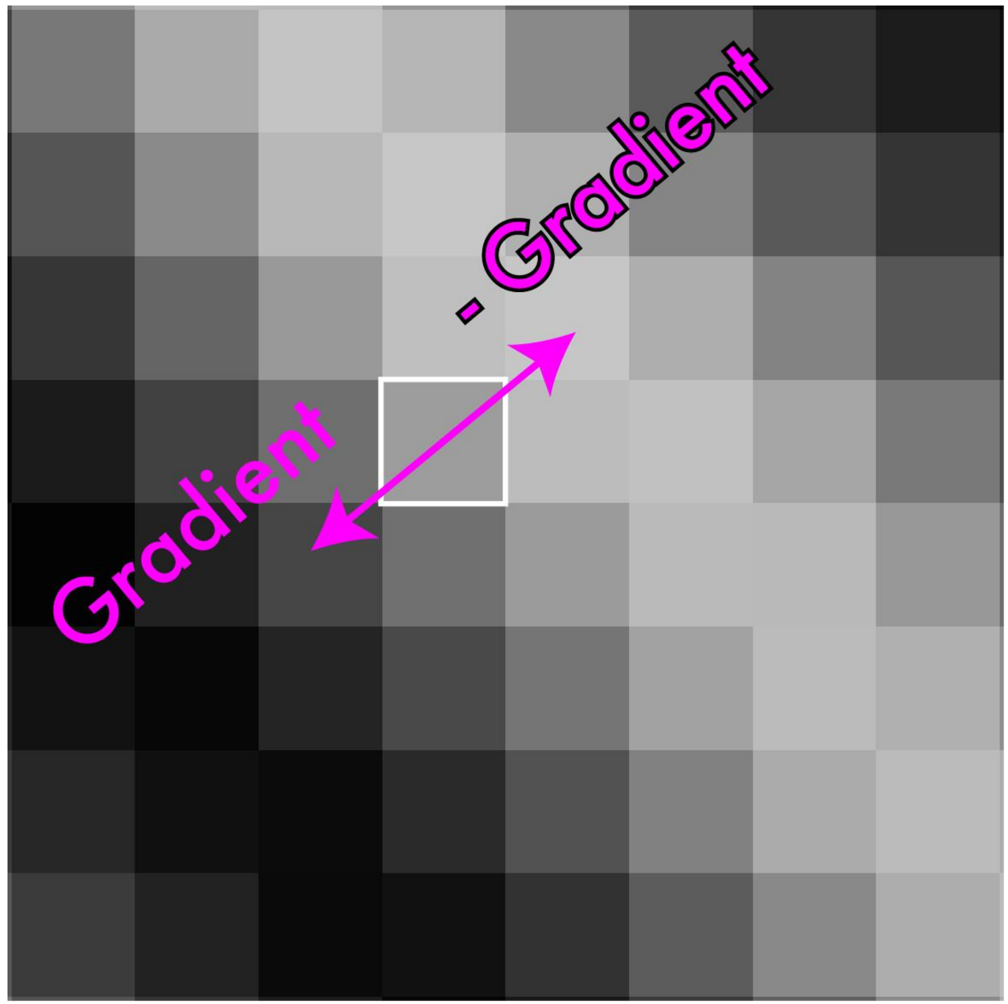
# Filtrage optimal : Canny

4.Suppression des non-maxima :

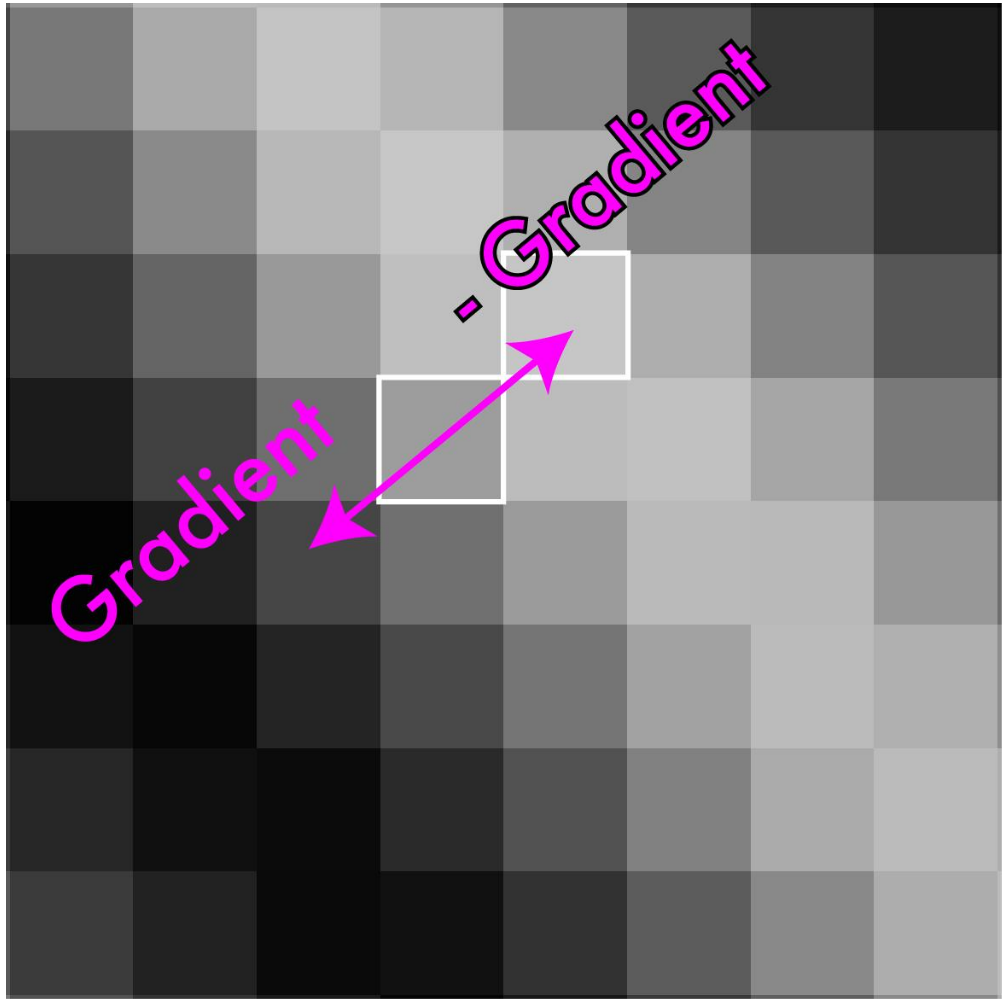
- Si la norme du gradient en un pixel (x, y) est inférieure à la norme du gradient d'un de ses 2 voisins le long de la direction du gradient, alors mettre la norme pour le pixel (x, y) à zéro.



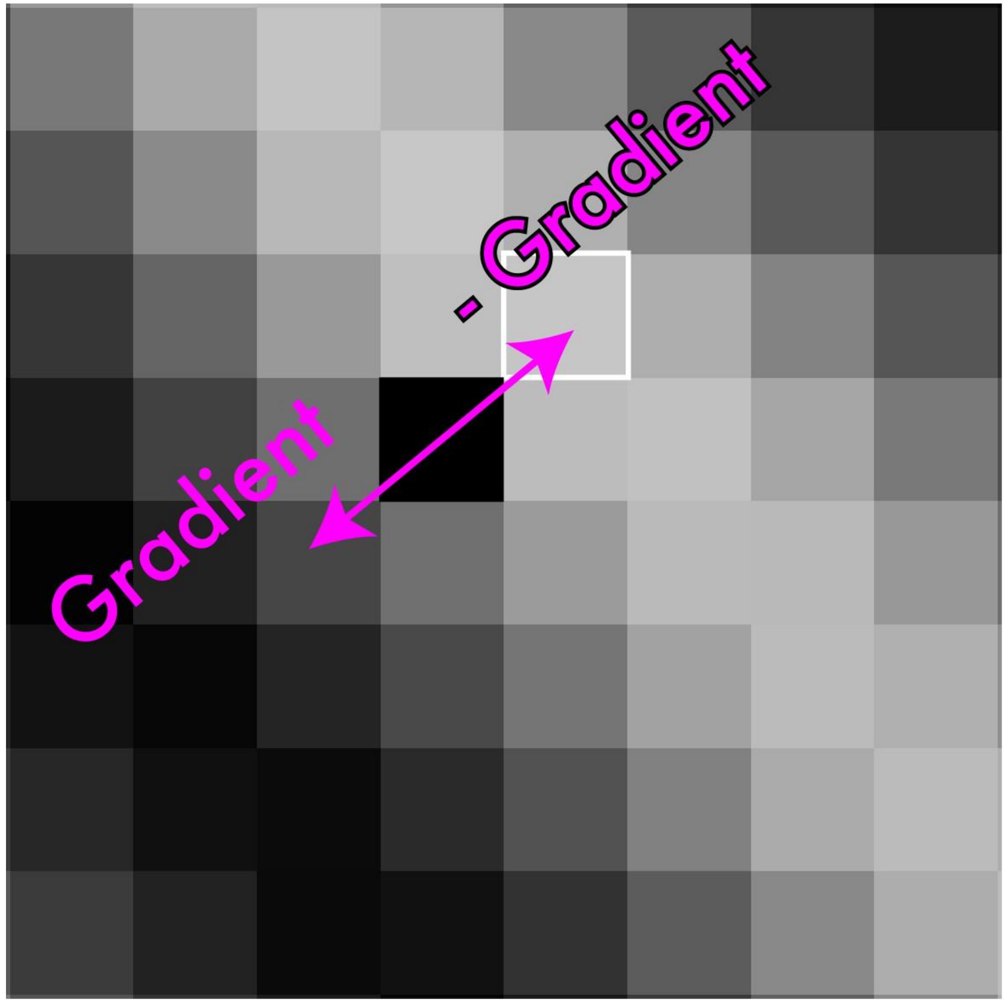
# Non-maximum suppression



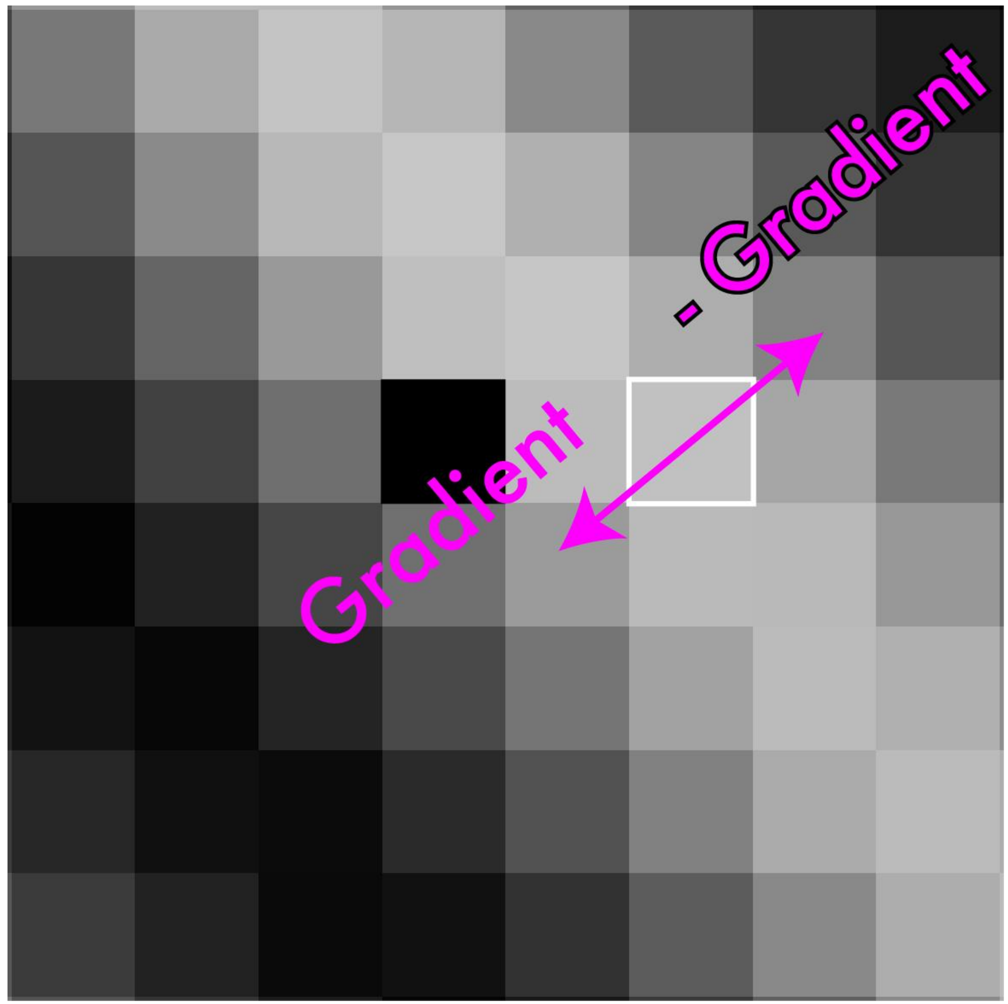
# Non-maximum suppression



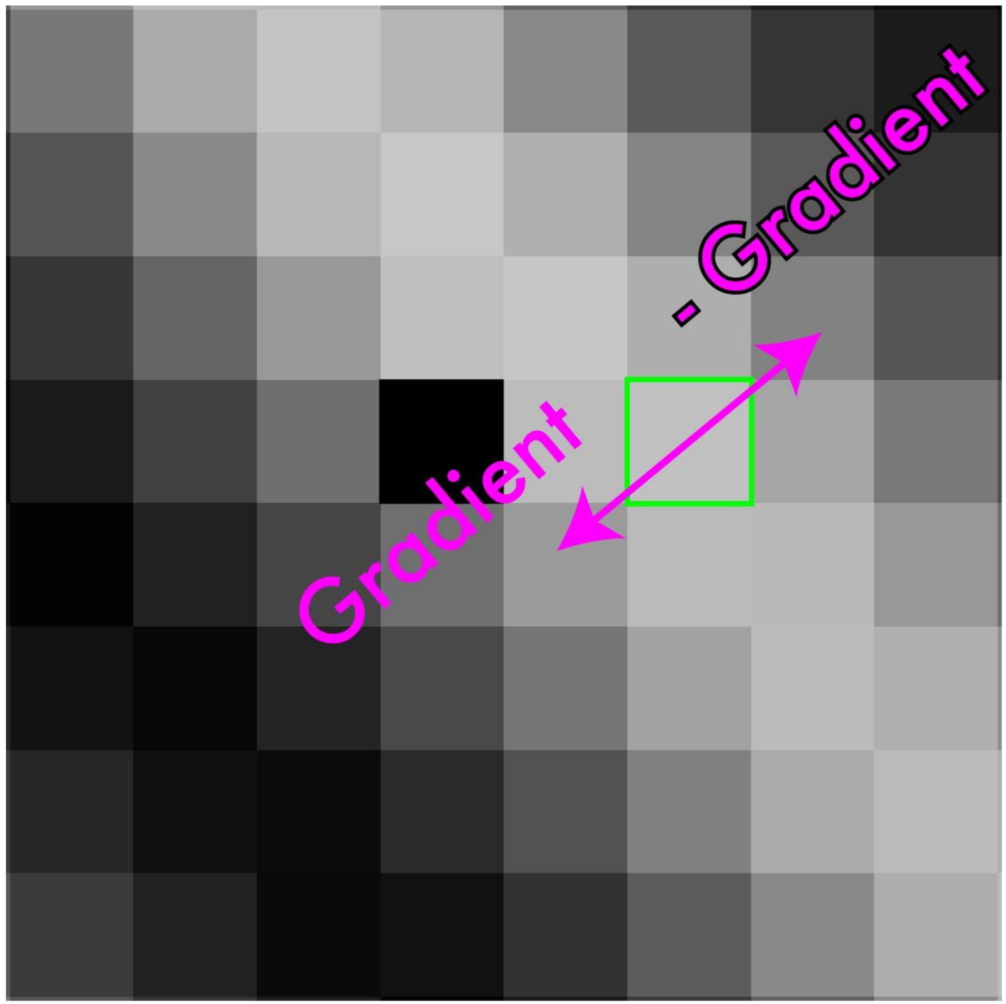
# Non-maximum suppression



# Non-maximum suppression



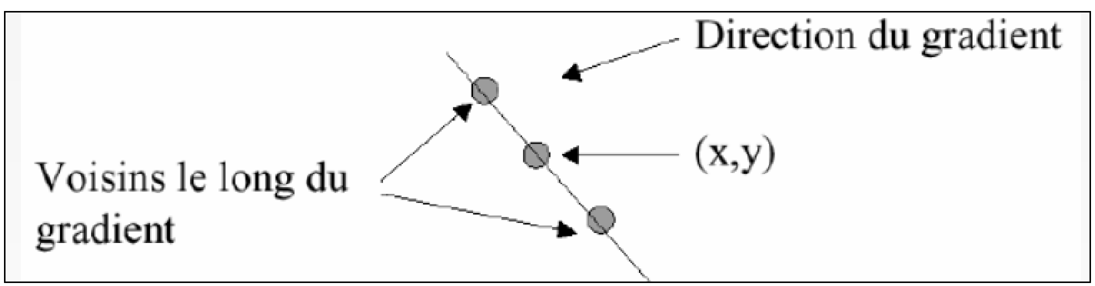
# Non-maximum suppression

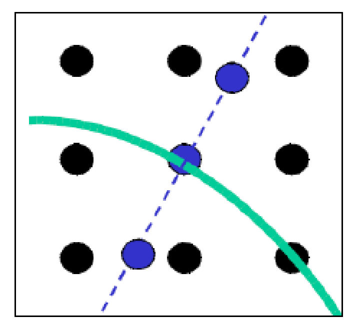


# Filtrage optimal : Canny

5. Seuillage des contours (hystérésis) :

- Utilise deux seuils : un seuil haut Sh et un seuil bas Sb.
- Pour chaque pixel, analyse de la norme du gradient :
  1. Si norme(x, y) < Sb alors le pixel est mis à 0 (∉ contour)
  2. Si norme(x, y) > Sh alors le pixel ∈ contour
  3. Si Sb ≤ norme(x, y) ≤ Sh alors le pixel ∈ contour s'il est connecté à un autre pixel déjà accepté comme contour.



# Détection de contours

**Résumé**

- La détection des points contours est basée sur les dérivées premières (gradient) ou secondes (Laplacien) 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, en particulier le Laplacien. Ils nécessitent donc l'emploi de filtres de lissage débruiteurs, en pré-traitement
- Les filtres de lissage/dérivation sont moins précis que le filtre de dérivation « pur », mais plus robustes. Ils privilégient donc la détection des points contours par rapport à leur localisation
- Ces filtres permettent seulement d'**estimer la « probabilité » qu'un pixel soit un point contour candidat**
- Il reste ensuite à :
  - décider si un pixel est *effectivement* un point contour, par exemple au **moyen d'un seuillage**
  - utiliser les points contours pour former les contours proprement dits.

# Un autre detecteur de contours

- Une Image est une fonction:
  - Qui est composée de basses et de hautes frequences
  - CF Transformée de Fourier

- Contours = Hautes frequences

- Il peut etre interessant / possible de detecter des contours de taille (frequence) spécifique

# Difference of Gaussian (DoG)

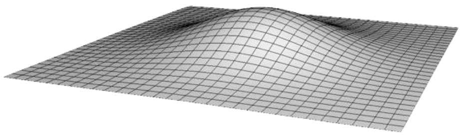
- Filtre Gaussien = filtre passe bas = g
- Detection des composantes avec une frequence $f < \sigma$
- $(g*I)$ produit une image des basses frequences
- $I - (g*I)$ produit une image des hautes frequences
- $g(\sigma 1)*I - g(\sigma 2)*I$ = filtre de frequences specifiques

- De plus, on montre que :

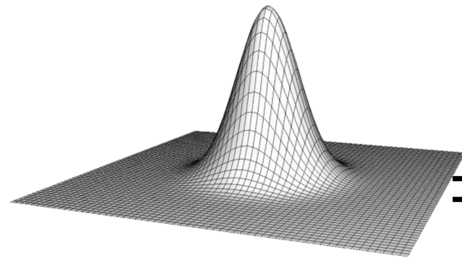
$$g(\sigma 1)*I - g(\sigma 2)*I = [g(\sigma 1) - g(\sigma 2)]*I$$

# Difference of Gaussian (DoG)

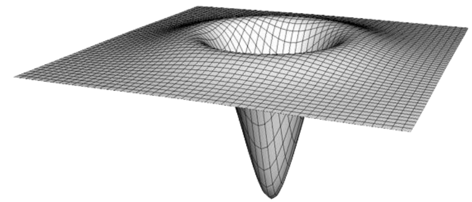
- $g(\sigma 1)*I - g(\sigma 2)*I = [g(\sigma 1) - g(\sigma 2)]*I$



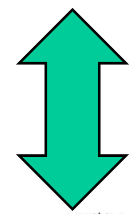
-

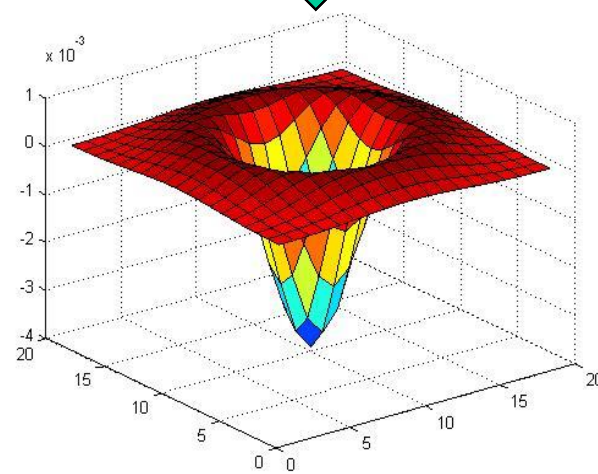


=



$\sigma = 2$ $\sigma = 1$

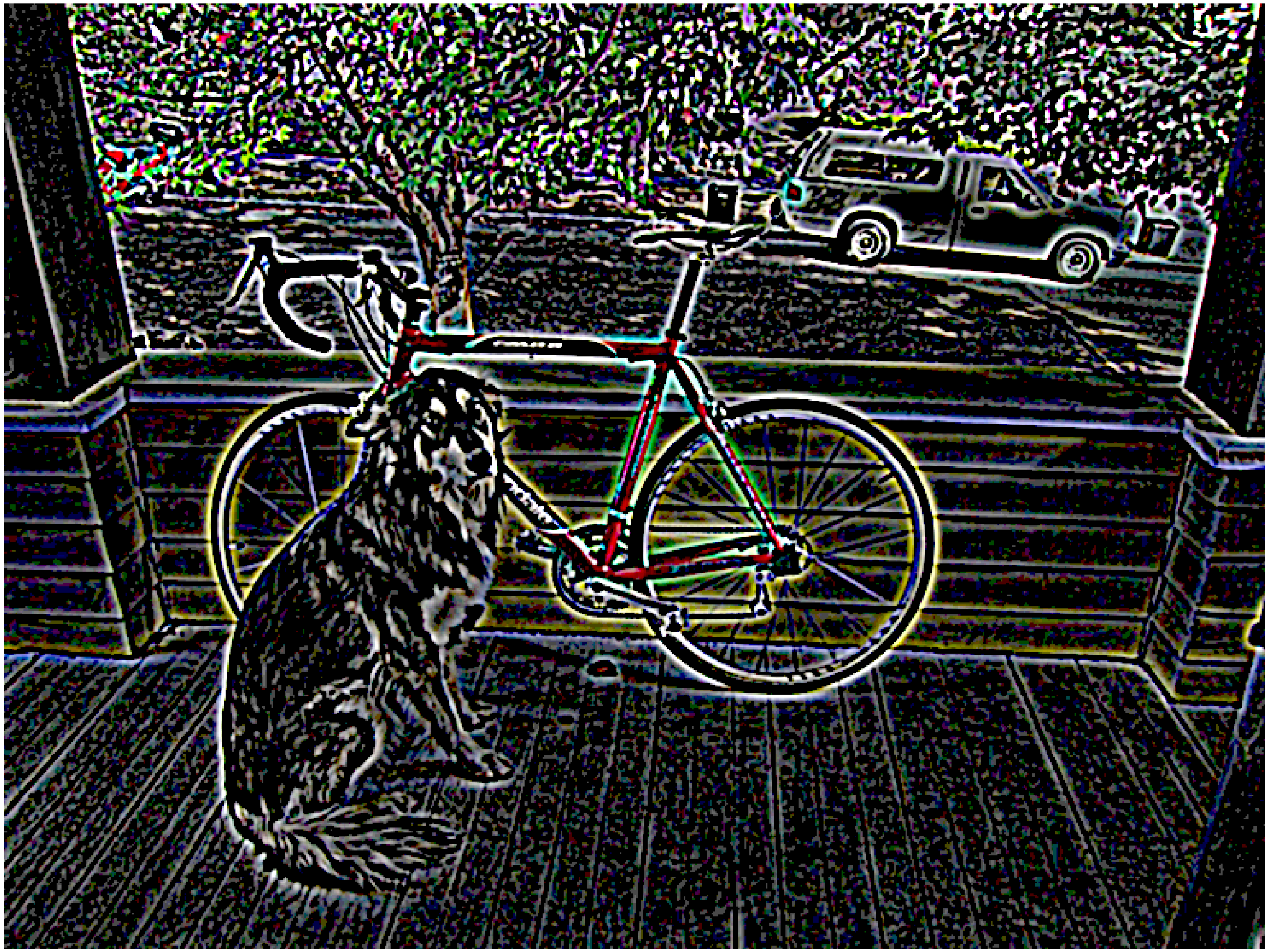




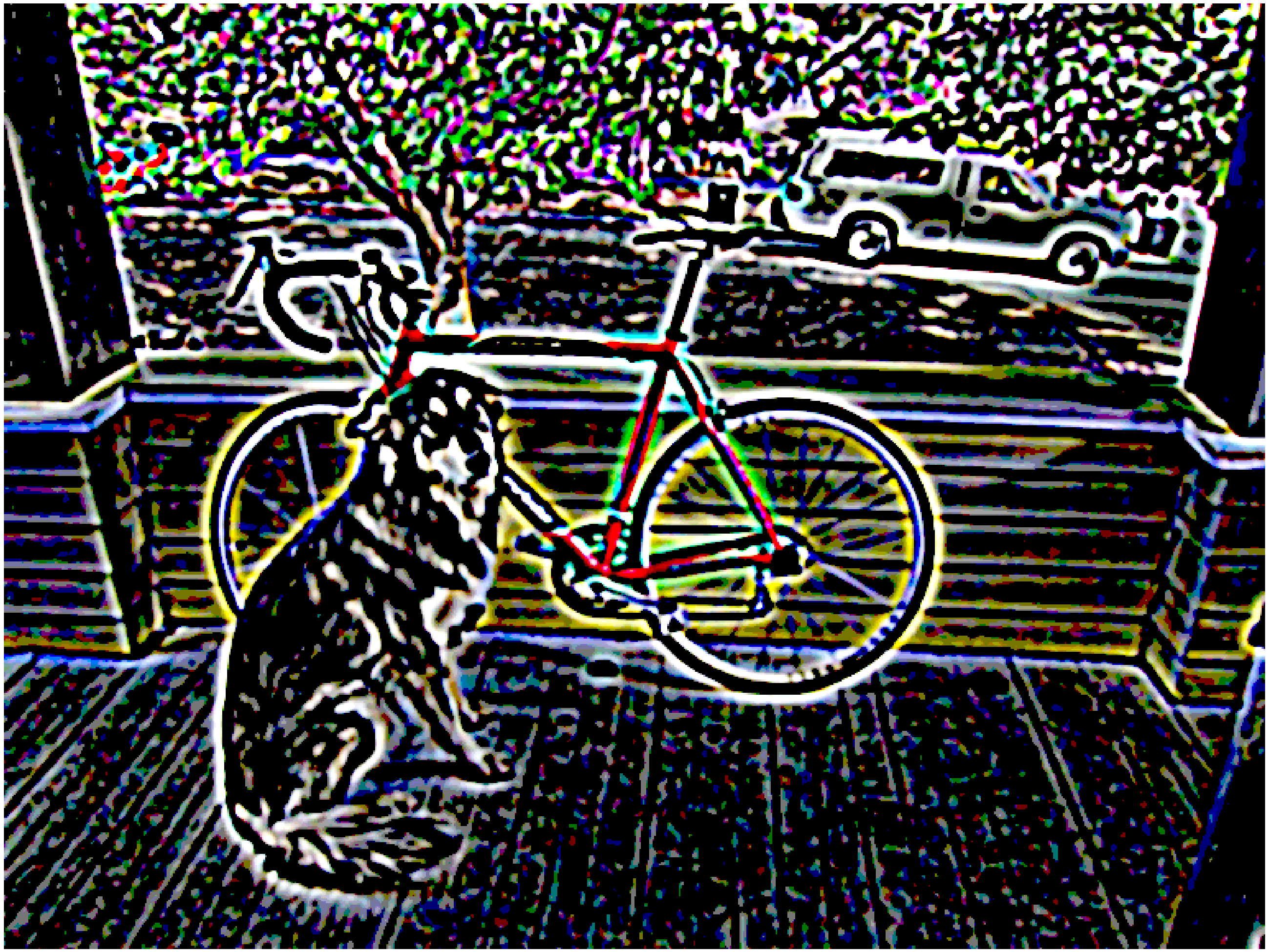

Remarque : Cette technique est proche de la technique LoG

# DoG (1 - 0)

# DoG (2 - 1)



# DoG (3 - 2)



# DoG (4 - 3)

# Tous le monde sait extraire des contours ?

- Appliquer les masques de dérivation en $x$ ($H_x$) et en $y$ ($H_y$) pour calculer, en chaque pixel de l'image ci-dessous, les dérivées selon les 2 directions principales :

$I_3 =$

| y\x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 200 | 200 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 200 | 200 | 200 | 200 | 0 | 0 |
| 4 | 0 | 0 | 200 | 200 | 200 | 200 | 200 | 0 | 0 |
| 5 | 0 | 0 | 200 | 200 | 200 | 200 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 200 | 200 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- Calculer et la norme et la direction du gradient en chaque pixel de cette image.
- Représenter le vecteur gradient aux pixels où sa norme est non nulle.
- Représenter l'image de la norme du gradient après suppression des non-maxima locaux.
- Quels type et valeur de seuil proposez-vous pour obtenir finalement les contours ?

# Tous le monde sait extraire des contours ?

- Expliquez les filtres ci-dessous

$$H_x = \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}, \quad H_y = \frac{1}{2}\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}, \quad H_{y1} = \frac{1}{2}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \quad H_{y2} = \frac{1}{2}\begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Quelles informations peuvent être apportées / obtenues à partir des contours ?