INTRODUCTION AU TRAITEMENT D'IMAGES

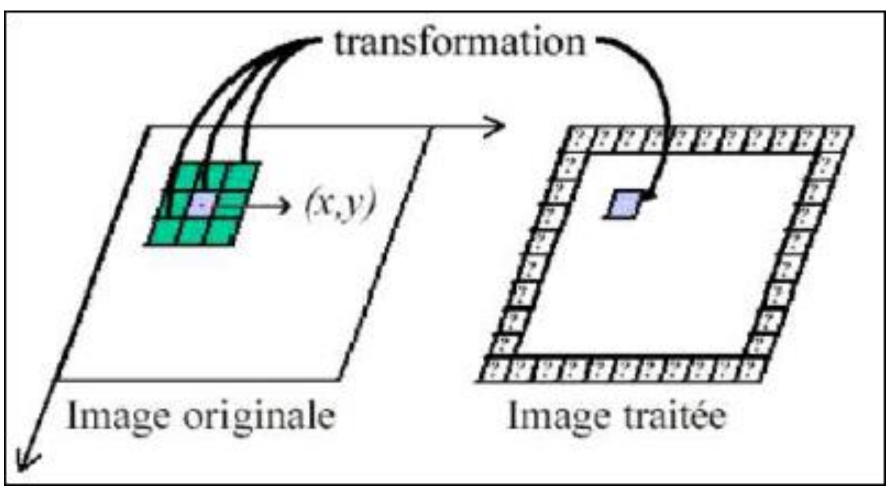
2eme Partie

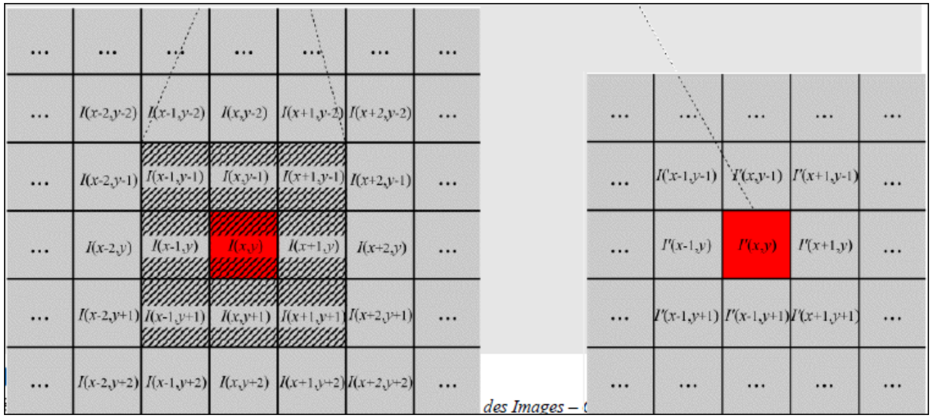
Donatello Conte

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



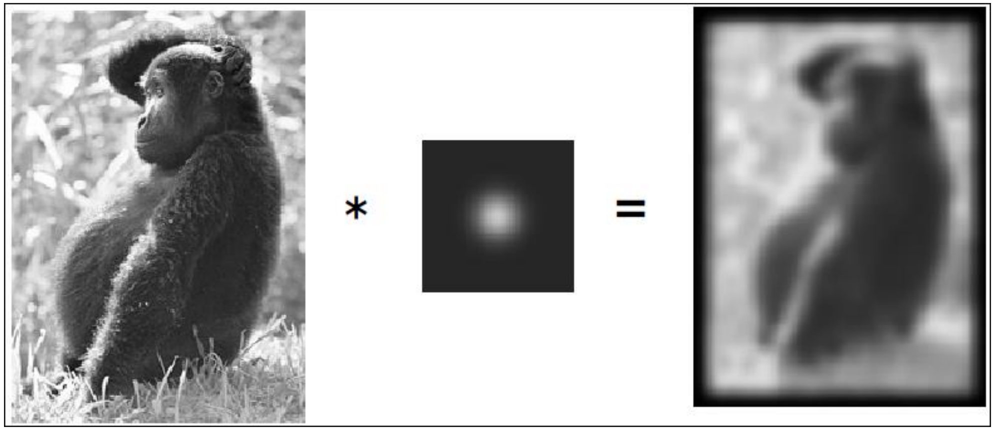


Convolution numérique

- 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, …

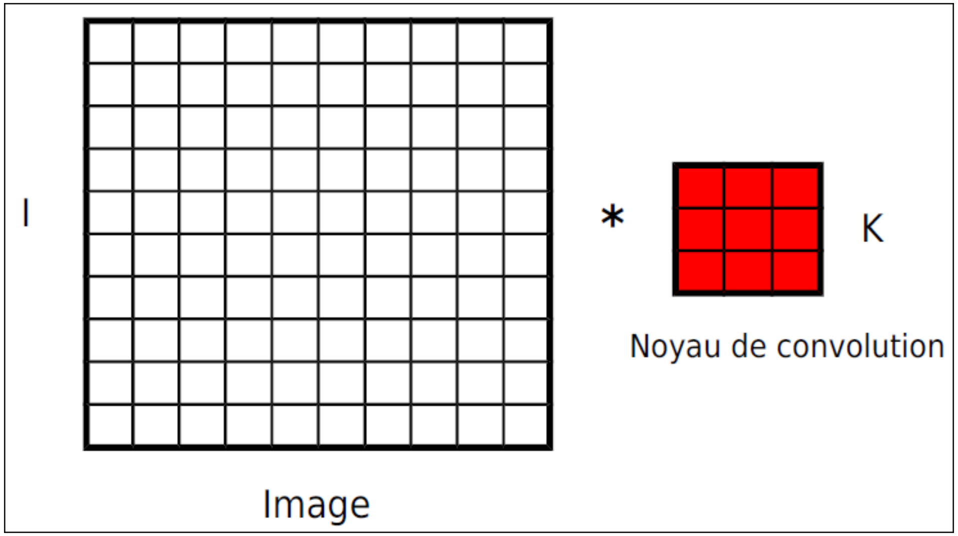


Convolution numérique

- **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

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i$$

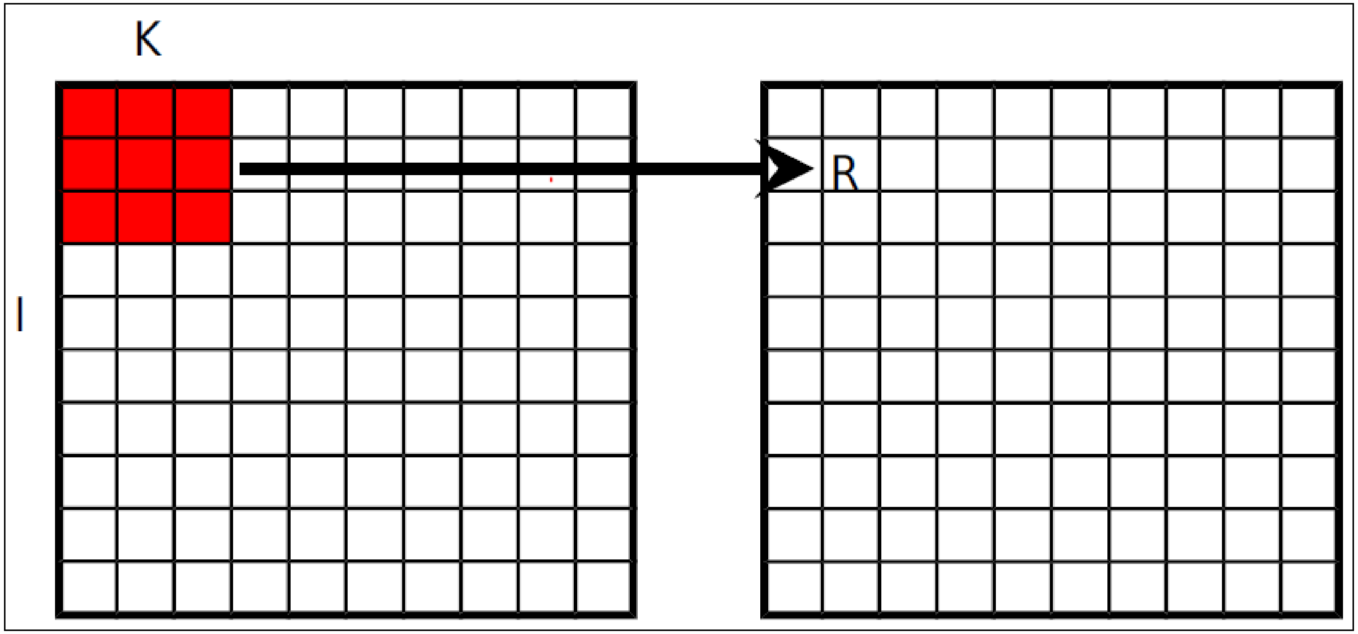
$$\begin{array}{ccccc} & & \vdots & & \\ & z1 & z2 & z3 & \\ \cdots & z4 & z5 & z6 & \cdots \\ & z7 & z8 & z9 & \\ & & \vdots & & \end{array}$$

w1	w2	w3
w4	w5	w6
w7	w8	w9

Convolution numérique

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

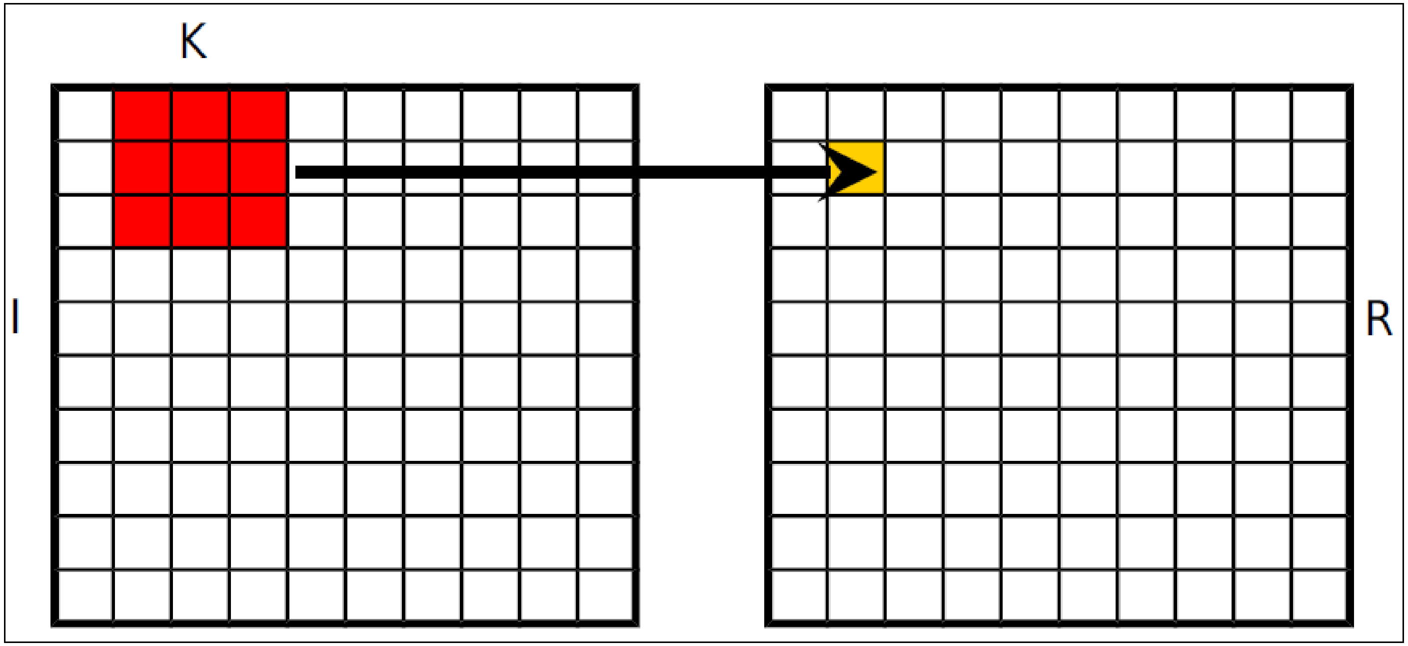
$$I'(i,j) = I(i,j) * filtre(i,j)$$
$$I'(i,j) = \sum_u \sum_v I(i-u, j-v) \cdot filtre(u,v)$$



Convolution numérique

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

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

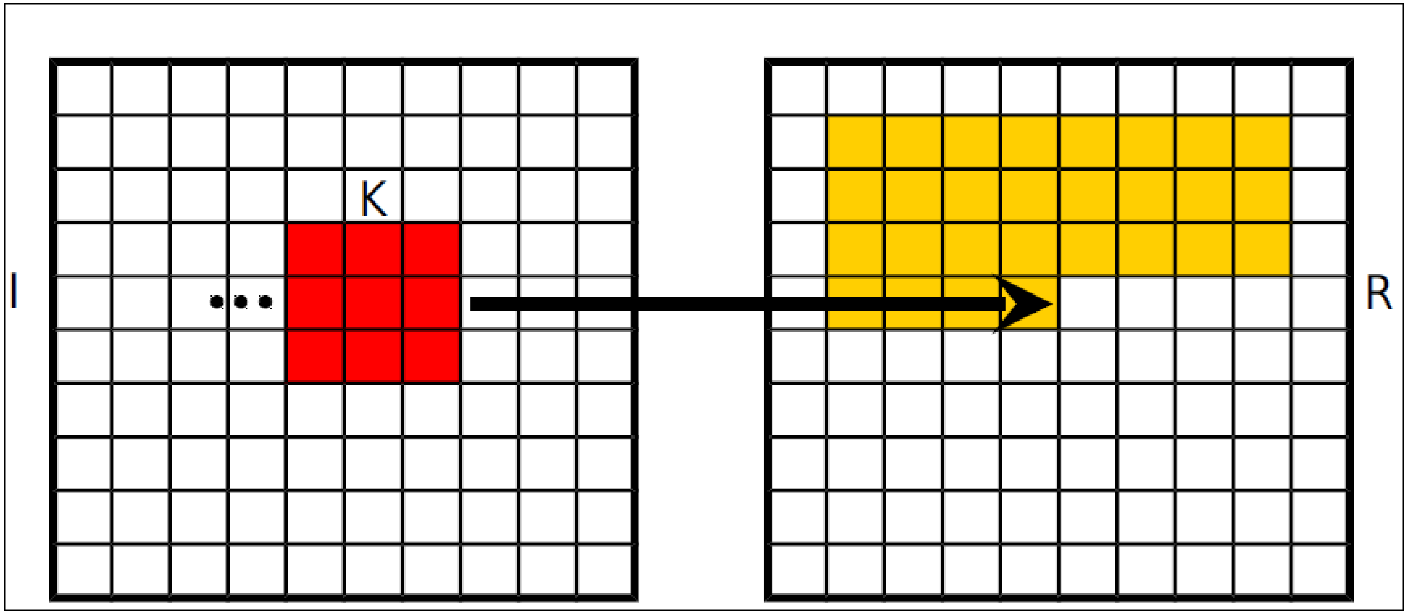


Convolution numérique

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

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

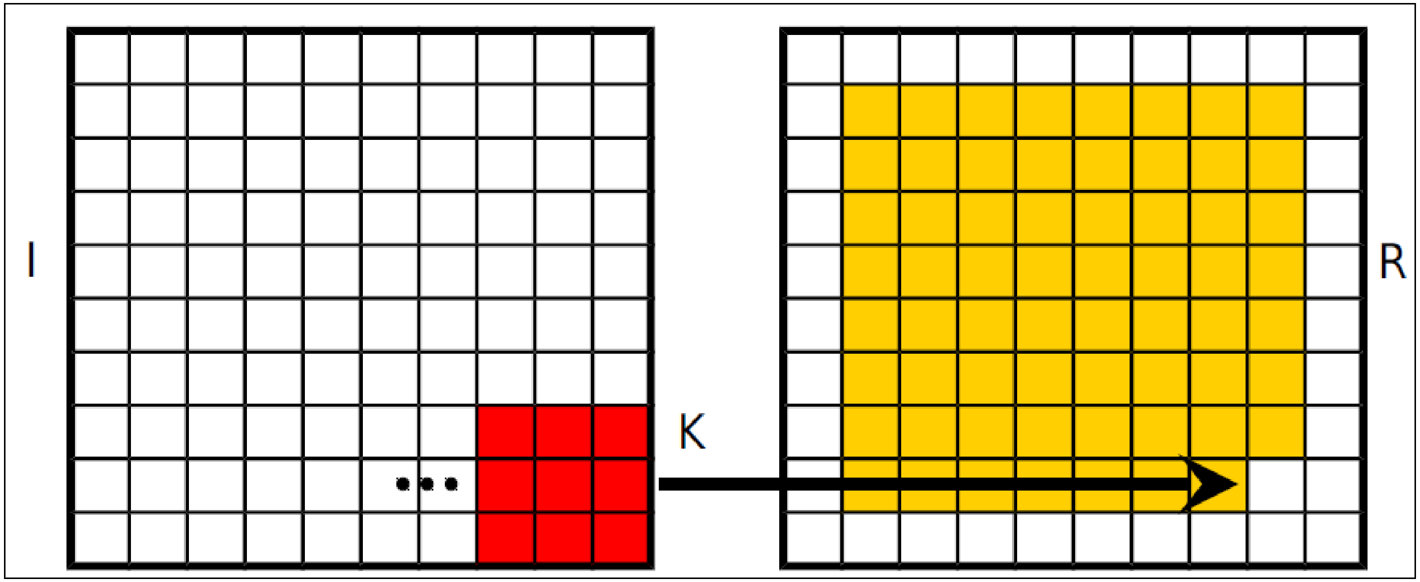
$$I'(i,j) = \sum_u \sum_v I(i-u, j-v) \cdot filtre(u,v)$$



Convolution numérique

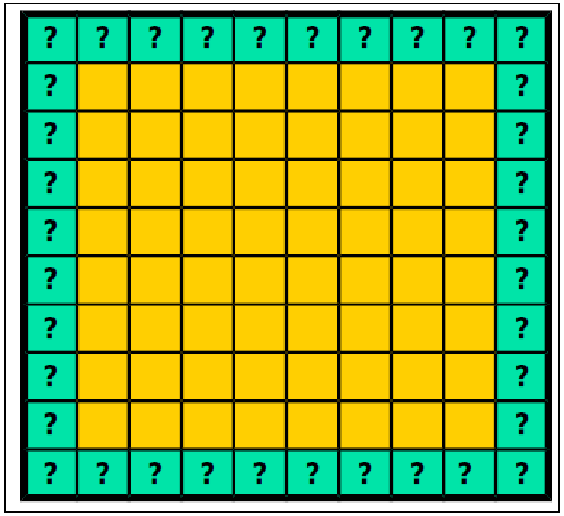
- **Convolution d'une image par un filtre 2D :**

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



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 !

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

Les filtres de lissage

- Les filtres de lissage sont utilises pour réduire le bruit ou supprimer certains détails de l'image
- Ils existent plusieurs filtres de lissage, linéaires et non linéaires
- Pour un lissage plus important on applique plusieurs fois le même filtre, ou on augmente le masque de convolution

Le filtre moyenneur

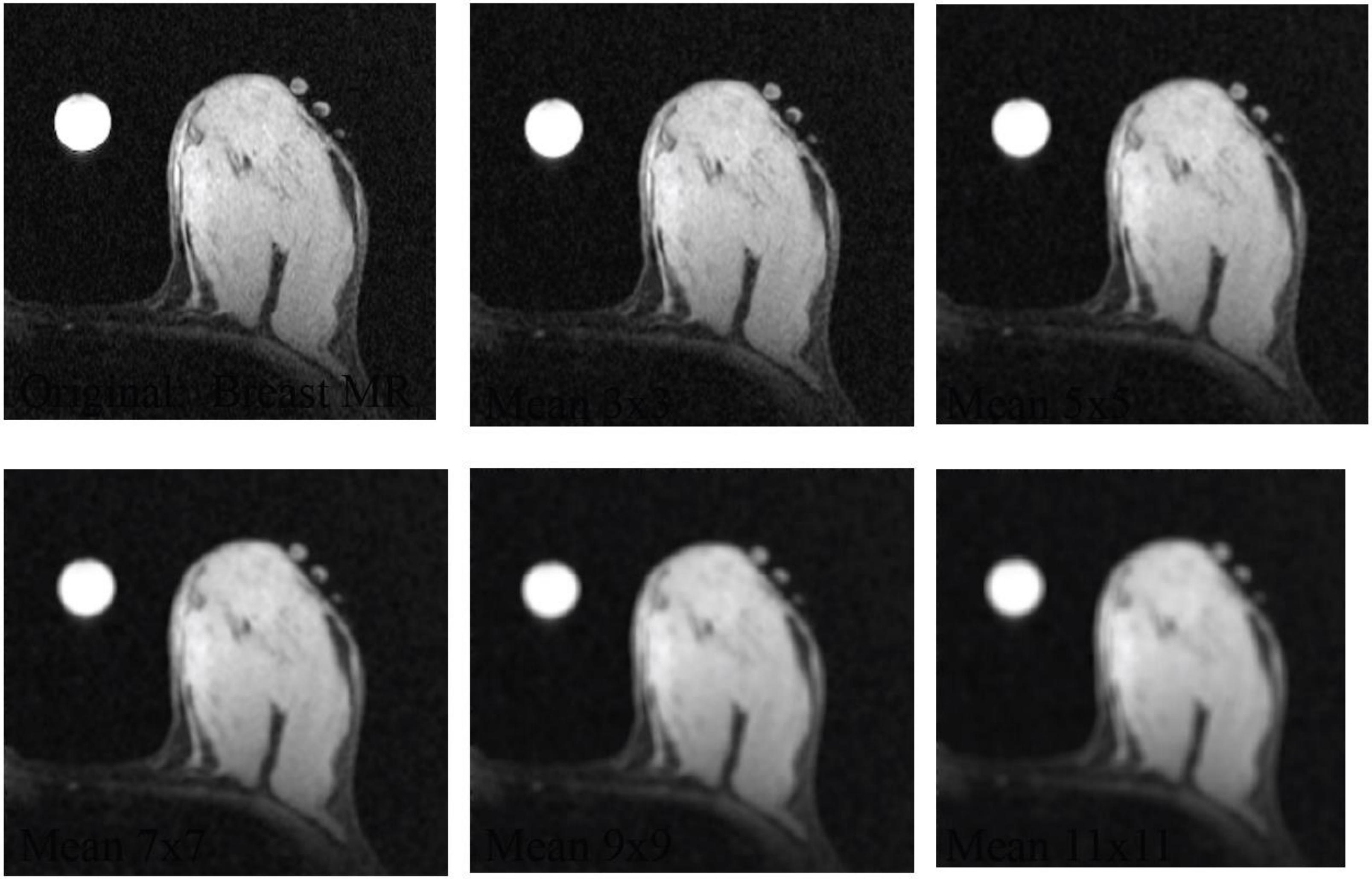
- Le plus simple des filtres de lissage est le filtre moyenneur
- La valeur de sortie du pixel correspond a la moyenne des valeurs de son voisinage
- L'opération correspond a la convolution avec un masque dont les valeurs sont toutes égales et la somme est 1

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

$$\frac{1}{9}\times\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$

$$\frac{1}{25}\times\begin{array}{|c|c|c|c|c|}\hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline\end{array}$$

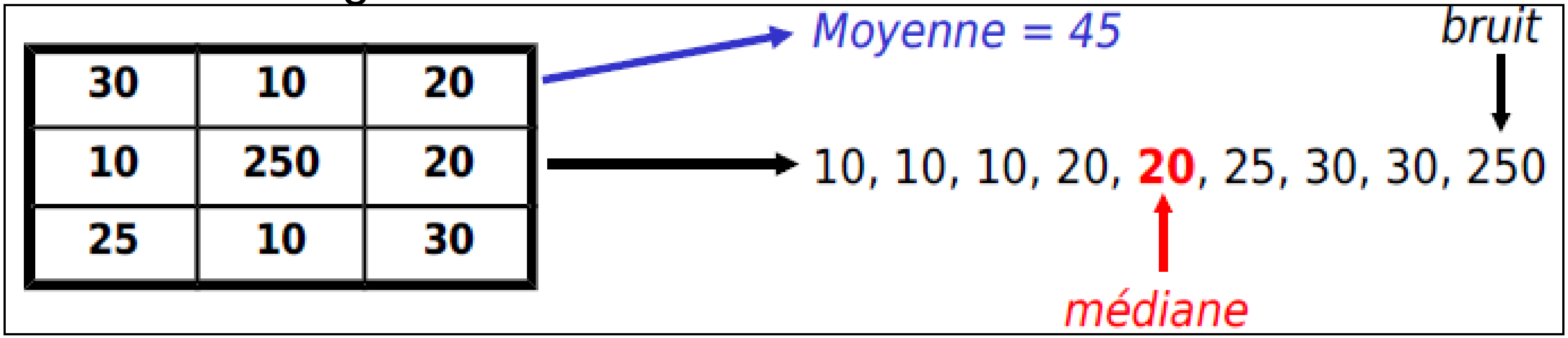
Le filtre moyenneur



Le bruit sel et poivre

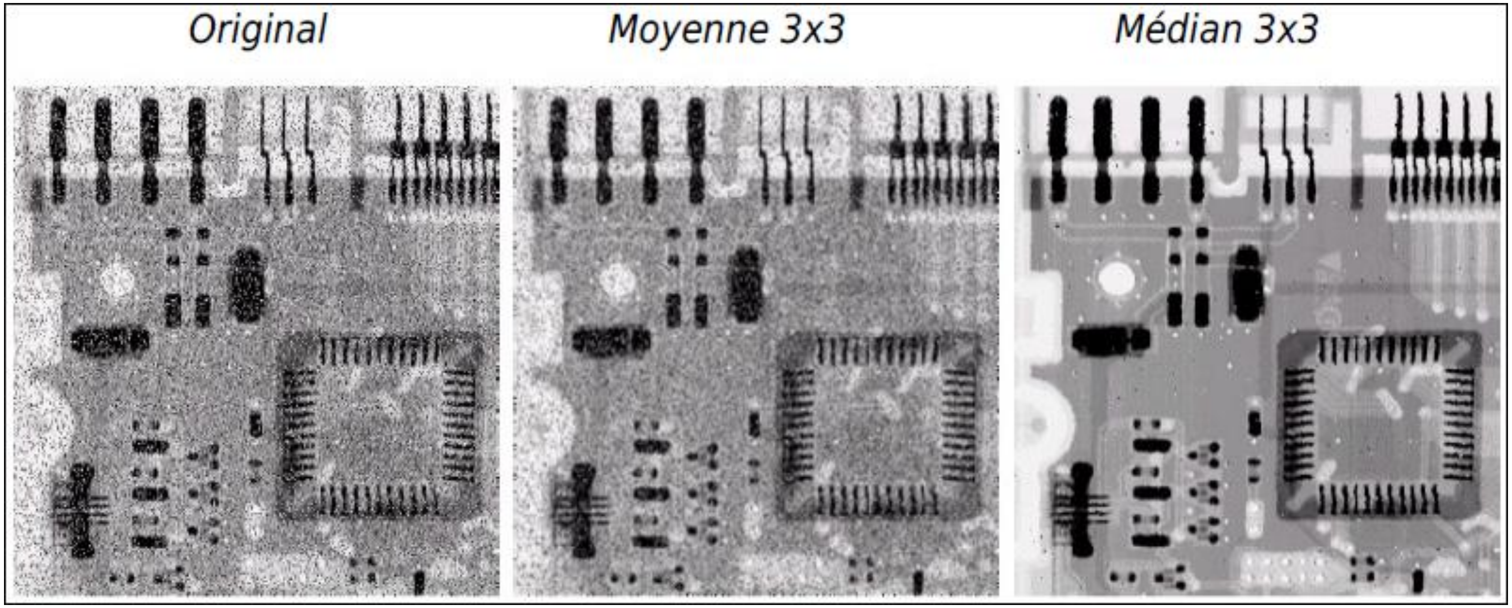
Le filtre Médian *(non-linéaire)*

- Pour nettoyer certains types de bruit dans une image, il existe mieux que le filtre moyenneur
- 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

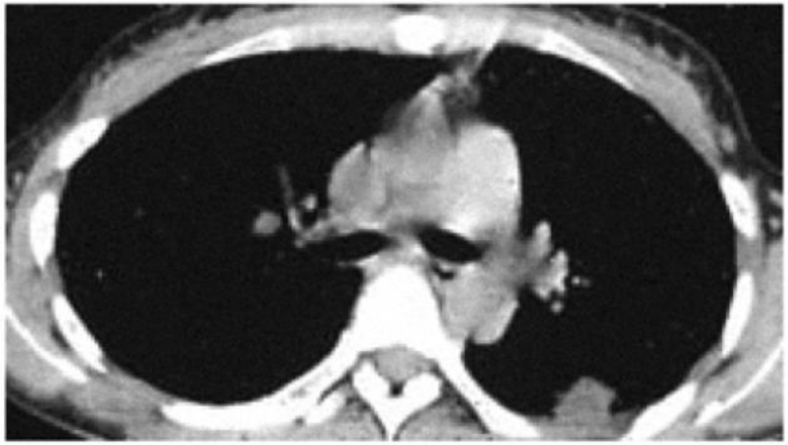


Le filtre Médian *(non-linéaire)*

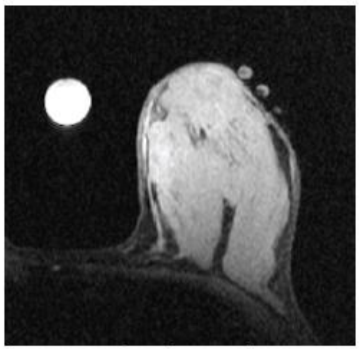
- **Propriétés du filtre médian** :
  - Non création de niveaux de gris
  - Invariance par étirement de contraste
  - Préservation des marches et rampes rectilignes
  - Érosion des connexités (notamment des disques)
  - Élimination du bruit impulsif



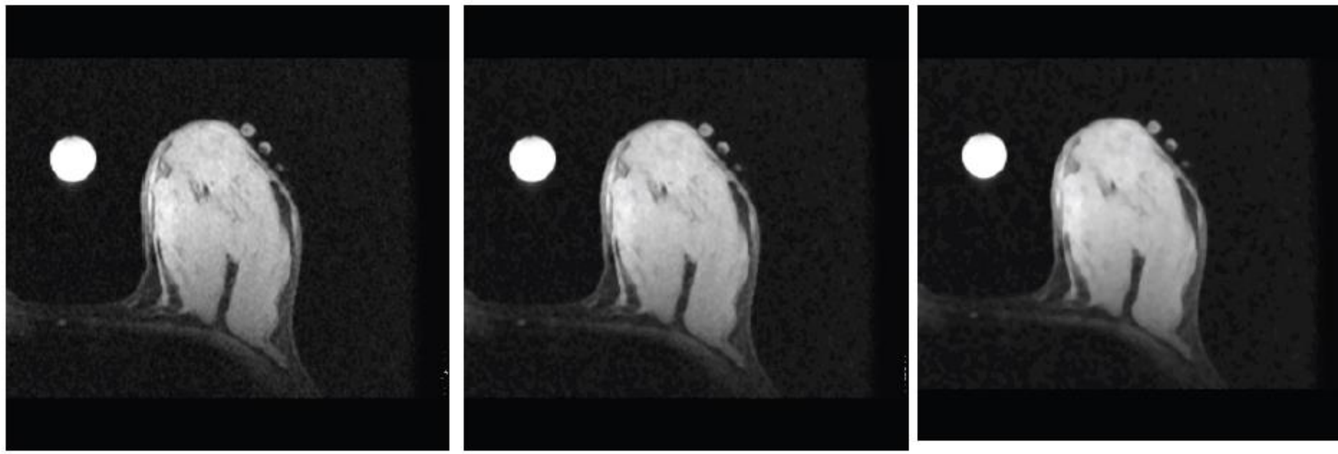
Le filtre Médian *(non-linéaire)*

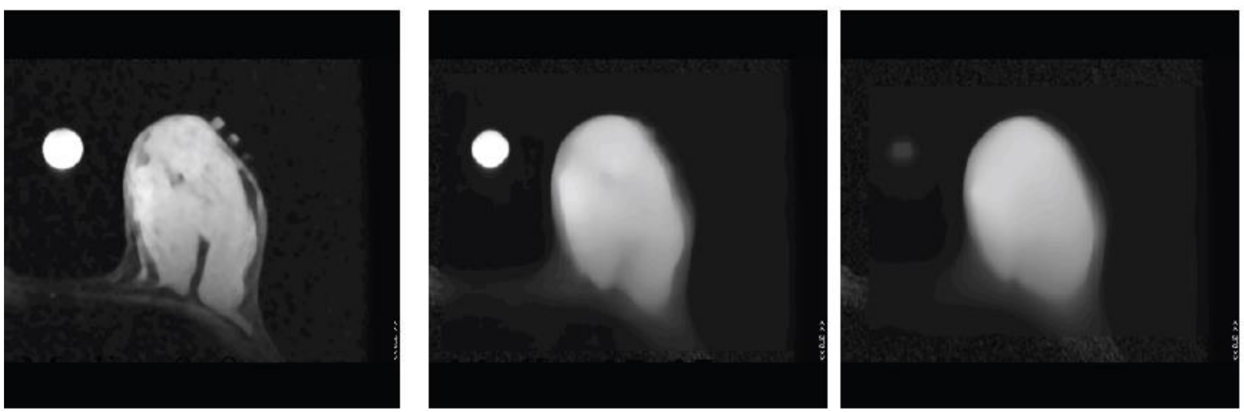


Le filtre Médian *(non-linéaire)*



original





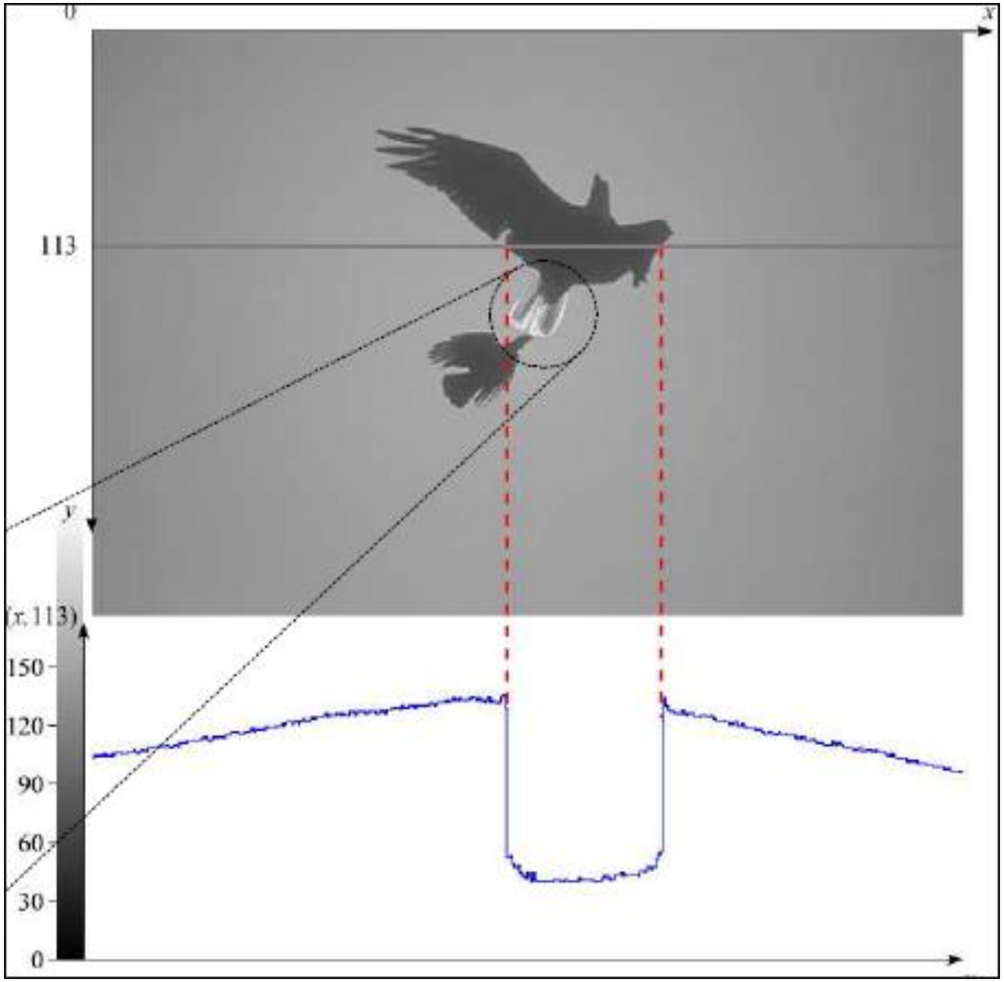
Approche frontières : Notion de contours

Définition

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

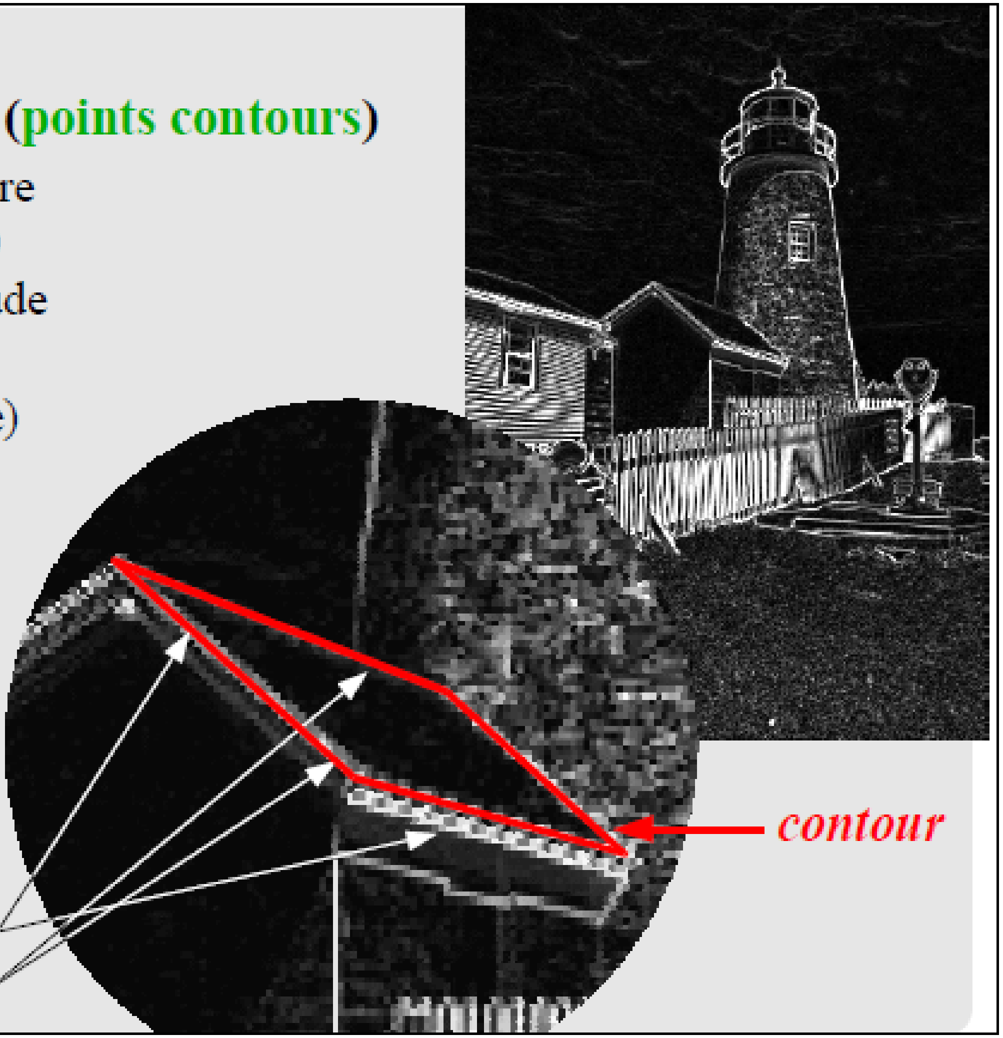
Caractérisation des zones de contours

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



Notion de contours

- « **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)

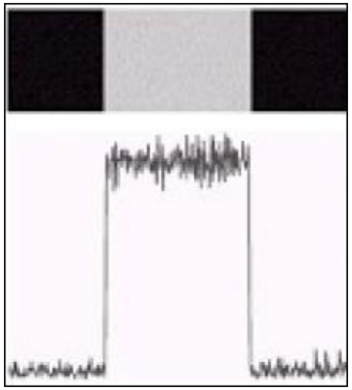


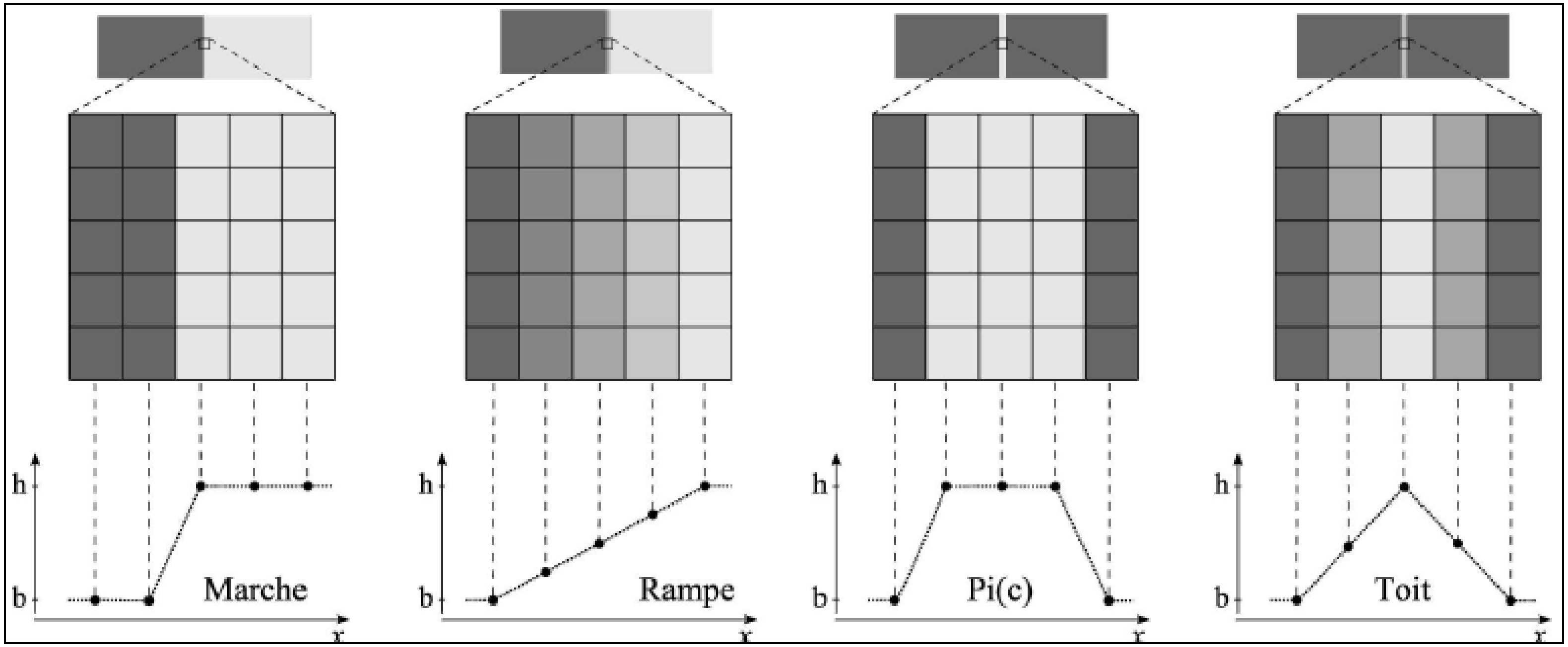
Notion de contours

Caractéristiques d'une zone de contour

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

Bruit ! ➔

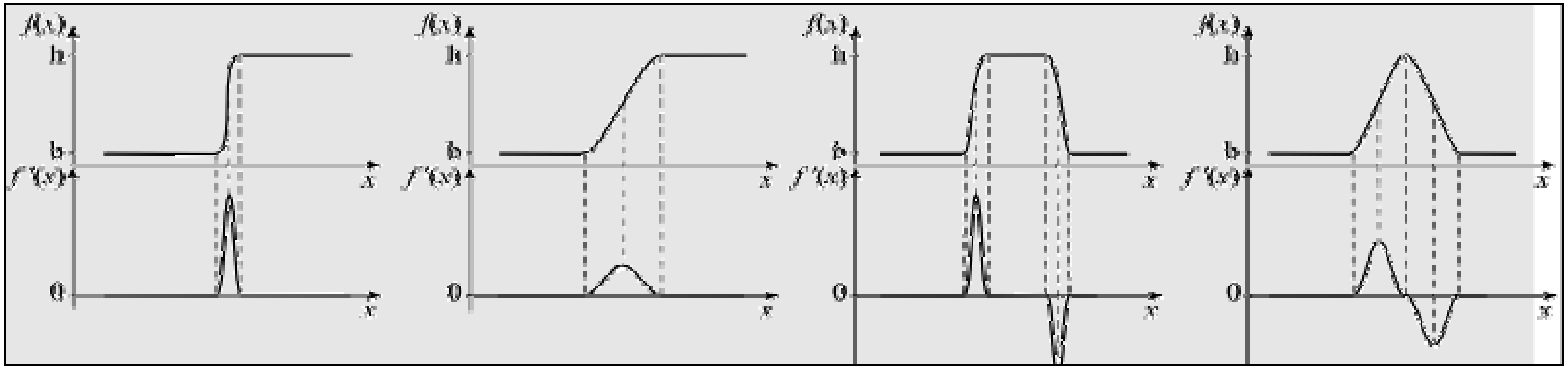




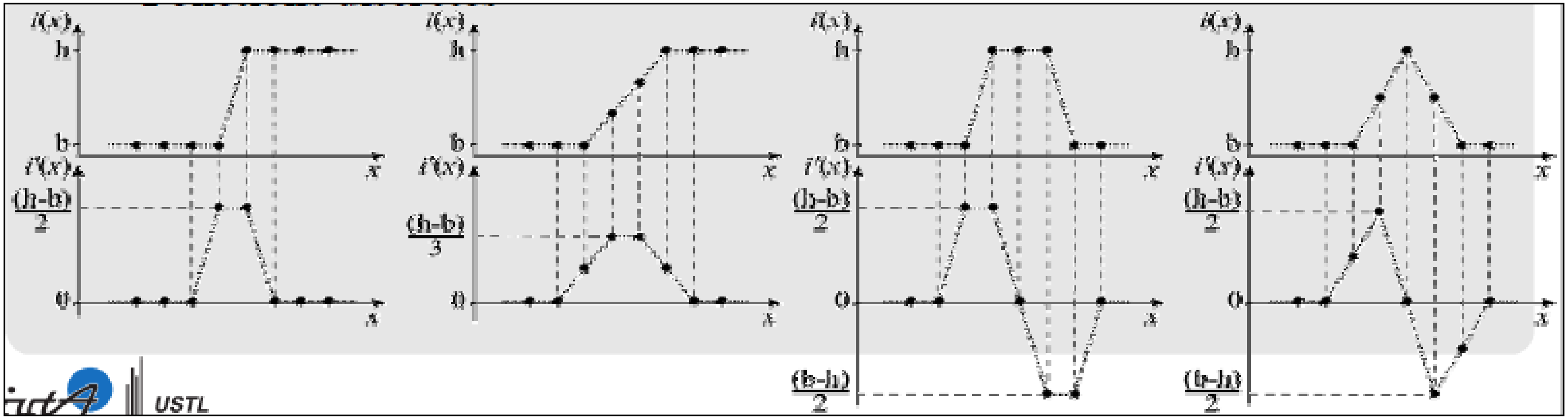
Détection de contours

Mise en évidence des zones de contours : dérivée première

- Fonctions continues



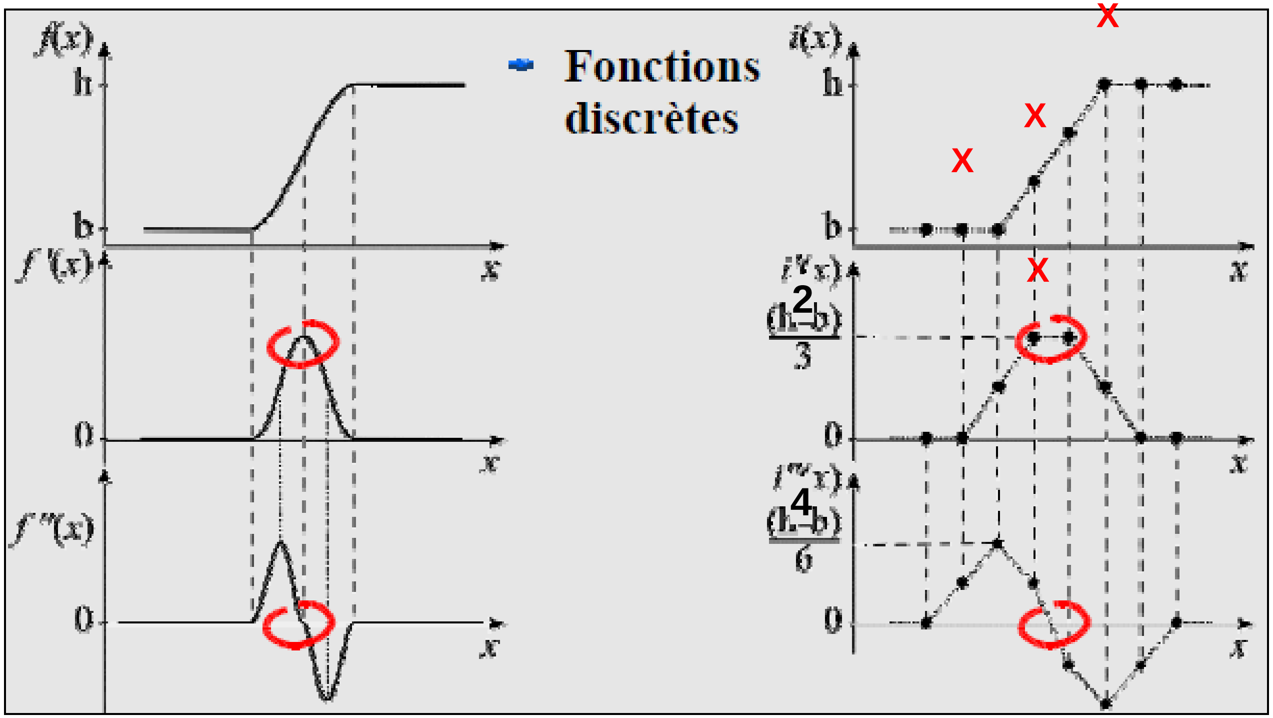
- Fonctions discrètes



Détection de contours

Mise en évidence des zones de contours : dérivée seconde

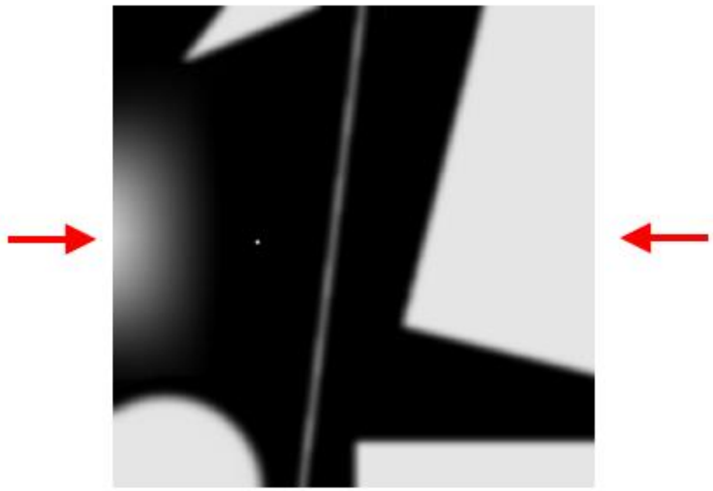
Fonctions continues ➔

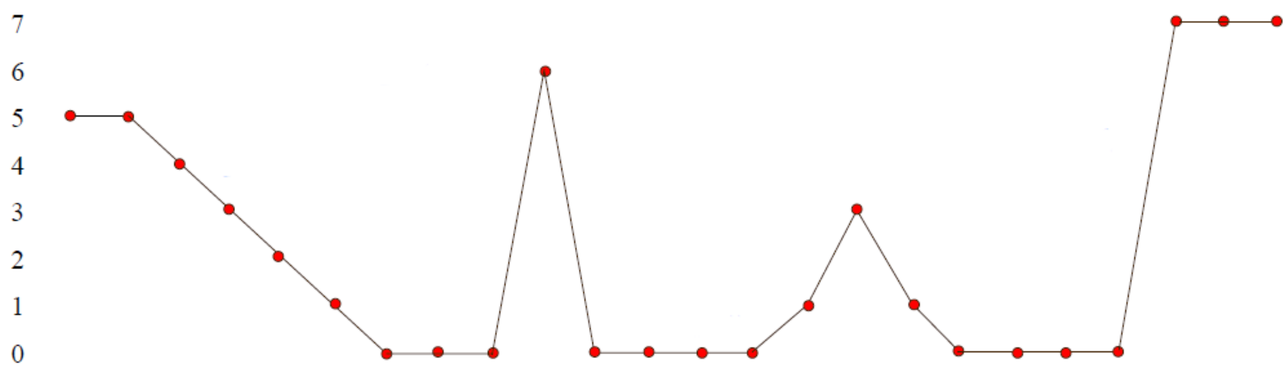


Détection des points contours : utilisation d'un critère de décision

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

Dérivées : example





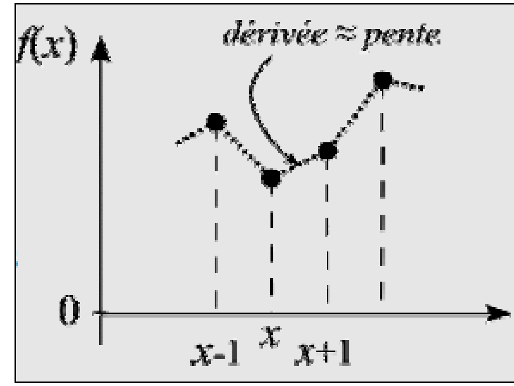
f	5	5	4	3	2	1	0	0	0	6	0	0	0	0	1	3	1	0	0	0	0	7	7	7	--
f'	--	-1	-1	-1	-1	-1	0	0	6	-6	0	0	0	1	2	-2	-1	0	0	0	7	0	0	--	--
f''	--	-1	0	0	0	0	1	0	6	-12	6	0	0	1	1	-4	1	1	0	0	7	-7	0	--	--

Dérivées : remarques

- On peut conclure que la dérivée première produit des contours épais, alors que la dérivée seconde donne des contours fins mais "en double"
- En correspondance du point isolé, la dérivée seconde a des valeurs plus importantes que celles de la dérivée première
- La dérivée seconde est capable de souligner les petits détails (bruit compris !)
- Sur le bord a droite on peut remarquer les valeurs doubles de la dérivée seconde ; cette propriété est exploité pour détecter les contours des objets dans les images

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

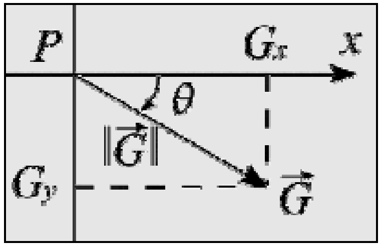
- On peut calculer une dérivée (partielle) de *f* dans chaque direction principale ➔

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

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

$$\vec{\nabla} f(x,y) = \begin{pmatrix} \dfrac{\partial f(x,y)}{\partial x} \\ \dfrac{\partial f(x,y)}{\partial y} \end{pmatrix} \qquad \vec{G}(x,y) = \begin{pmatrix} G_x(x,y) \\ G_y(x,y) \end{pmatrix}$$

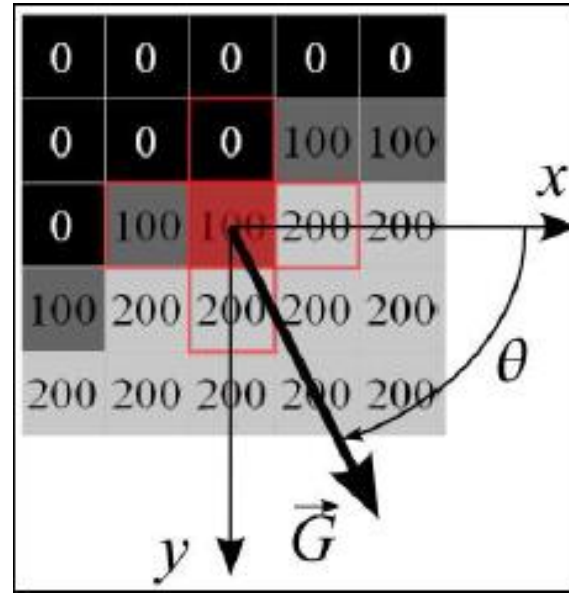
- Ce vecteur est caractérisé, en chaque point *P*, par
  - une norme (ou module) $\|G\|=\mathrm{sqrt}(Gx^2+Gy^2)$
  - une direction $\Theta = \arctan(Gy / Gx)$



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

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

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

- **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 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

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

  - 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

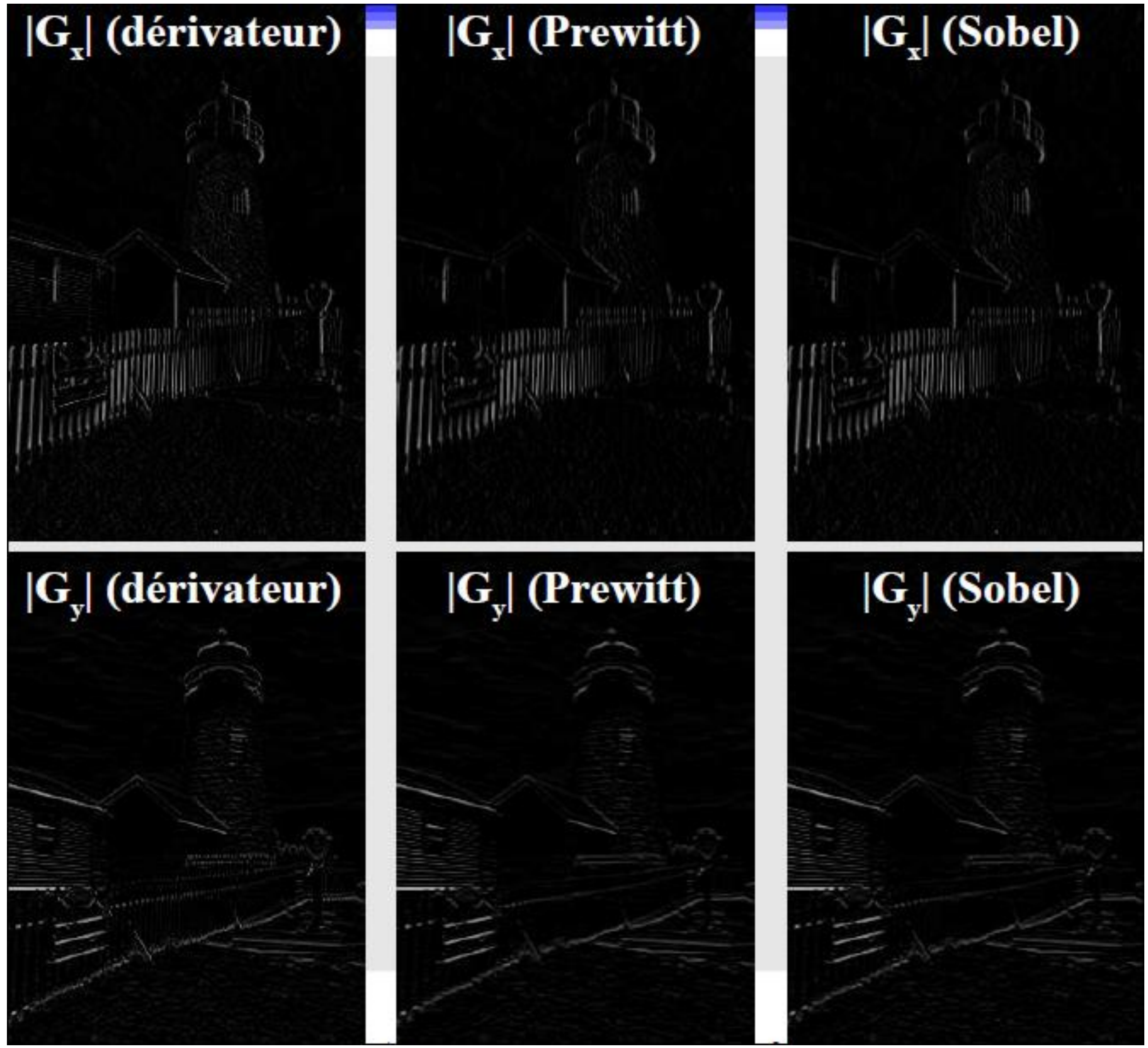
- Calcul de la composante horizontale du gradient *Gx*

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

- 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 $\dfrac{\partial^2}{\partial x^2}$: $\dfrac{1}{4}$

+1	-2	+1

pour $\dfrac{\partial^2}{\partial y^2}$: $\dfrac{1}{4}$

+1
-2
+1

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

pour $\dfrac{\partial^2}{\partial X^2}$: $\dfrac{1}{4}$

0	0	+1
0	-2	0
+1	0	0

pour $\dfrac{\partial^2}{\partial Y^2}$: $\dfrac{1}{4}$

+1	0	0
0	-2	0
0	0	+1

Approximations discrètes du Laplacien Δ :

$\dfrac{1}{8}$

0	+1	0
+1	-4	+1
0	+1	0

ou $\dfrac{1}{8}$

+1	0	+1
0	-4	0
+1	0	+1

ou encore $\dfrac{1}{16}$

+1	+1	+1
+1	-8	+1
+1	+1	+1

Détection de contours et Laplacien

Sobel Laplacien

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.