

Introduction à l'imagerie numérique

3I022-fev2018

Détection de contours

Licence d'informatique

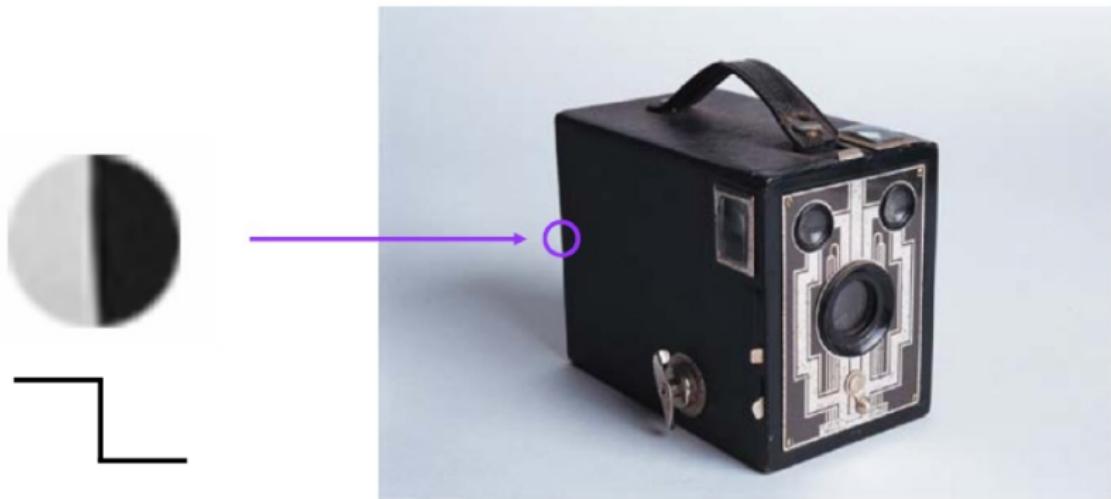


Février 2018

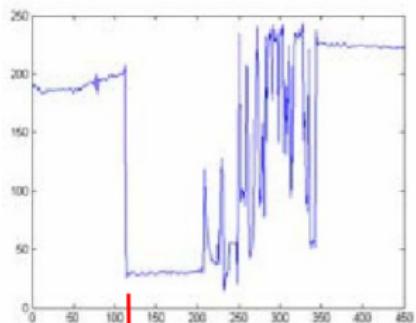
Détection contours (*edge detection*)

- ▶ Qu'est-ce qu'un contour ?
 - ▶ Définition spatiale
 - ▶ Définition fréquentielle
- ▶ Opérateurs dérivatifs du premier ordre
- ▶ Opérateurs dérivatifs du second ordre
- ▶ Détection de contours par filtrage optimal : méthode de Canny-Deriche
- ▶ Détection de points de contours
- ▶ Suivi de contours

Définition spatiale du contour



Définition spatiale du contour



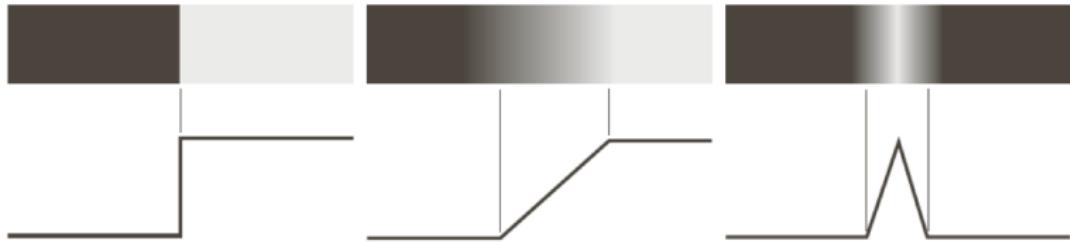
Définition spatiale du contour

- ▶ Un contour est une variation locale d'intensité dans l'image
- ▶ C'est une frontière qui sépare deux objets, ou discontinuité
- ▶ La réciproque n'est pas forcément vraie : une discontinuité dans l'image n'est pas forcément un contour (bruit, ...)
- ▶ Un contour peut être du à :
 - ▶ la profondeur de l'image ;
 - ▶ l'illumination ;
 - ▶ l'orientation des surfaces ;
 - ▶ la réflectance.

Définition spatiale du contour

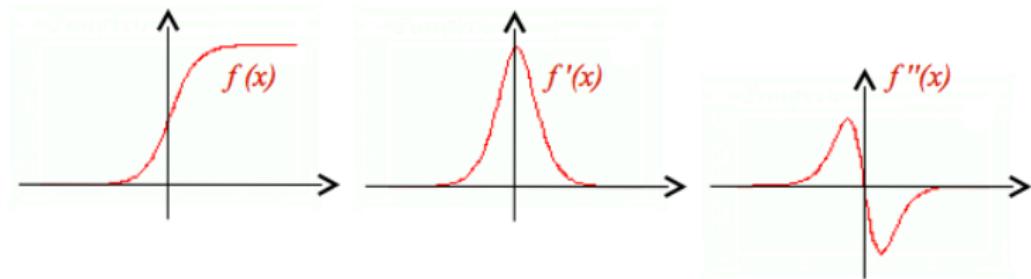
▶ Types de contours

- ▶ Contour “saut d'amplitude” : entre deux régions ayant des intensités moyennes différentes
- ▶ Contour “en rampe” : variation d'intensité avec pallier
- ▶ Contour “en toit” : variation locale d'intensité, présentant un maximum ou un minimum



Définitions du contour

- ▶ Propriétés de dérivation d'un contour

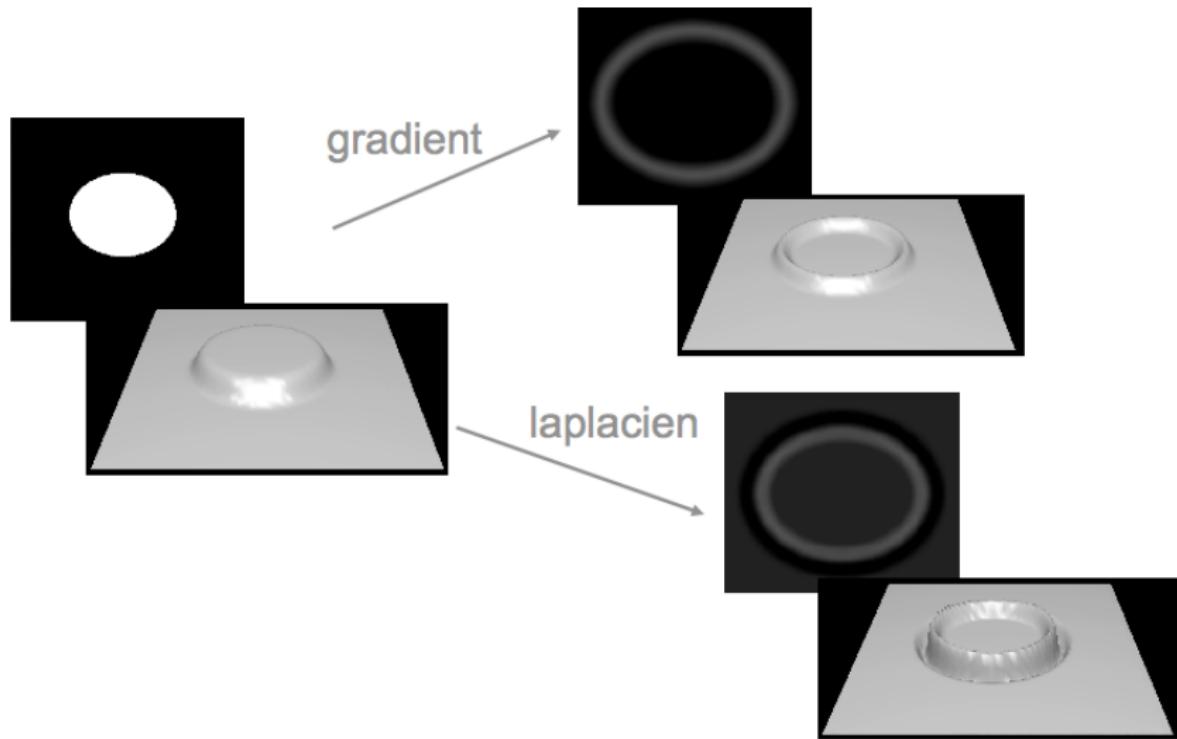


- ▶ Définition fréquentielle
 - ▶ Variation brutale : haute fréquence dans le spectre d'amplitude
 - ▶ Nécessité d'utiliser un filtre travaillant sur les hautes fréquences uniquement
 - ▶ Filtre passe-haut : préserve et met en avant les HF tout en éliminant les BF

Détection de contours

- ▶ Dérivée première d'un contour
 - ▶ Opérateurs dérivatifs du premier ordre
 - ▶ Passage par un maximum au niveau du contour
 - ▶ Opérateur gradient
 - ▶ Sa direction (perpendiculaire au contour) est celle dans laquelle la dérivée de f est maximale
 - ▶ Ce maximum est donné par son module
- ▶ Dérivée seconde d'un contour
 - ▶ Opérateurs dérivatifs du second ordre
 - ▶ Passage par zéro au niveau du contour
 - ▶ Opérateur laplacien (approximation)

Méthodes analytiques pour la détection de contour



Opérateurs dérivatifs du premier ordre : gradient ∇

- ▶ Mesure des dérivées de l'image dans deux directions (G_x et G_y)

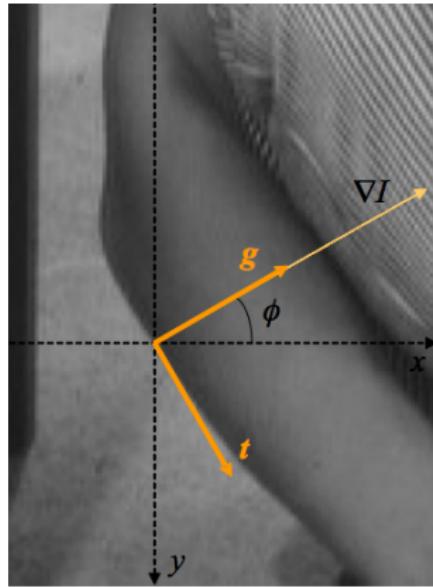
$$\nabla I = \begin{pmatrix} \frac{\partial I(x,y)}{\partial x} \\ \frac{\partial I(x,y)}{\partial y} \end{pmatrix} = \begin{pmatrix} G_x \\ G_y \end{pmatrix}$$

- ▶ Calcul du module G du gradient : carte des transitions de l'image

$$G = \sqrt{G_x^2 + G_y^2}$$

- ▶ Calcul de l'orientation ϕ du gradient (perpendiculaire à celle φ du contour)

$$\phi = \tan^{-1} \left(\frac{G_y}{G_x} \right) \quad \text{et} \quad \varphi = \frac{\pi}{2} + \phi$$



Détection de contours avec le gradient

- ▶ Calculer le gradient en chaque point de l'image
- ▶ Créer l'image du module (i.e. norme) du gradient
- ▶ Extraire les maxima locaux
- ▶ Seuiller l'image des maxima locaux pour ne garder que les contours significatifs et fins (post-traitements)

Approximation du gradient : cas discret

- ▶ Les dérivées directionnelles suivant les directions horizontales et verticales au point (x, y) sont approchées par les différences finies

$$\begin{aligned}G_x &= I(x+1, y) - I(x, y) \\G_y &= I(x, y+1) - I(x, y)\end{aligned}$$

- ▶ La norme G du gradient et l'orientation φ du contour sont données par

$$\begin{aligned}G &= \sqrt{(I(x+1, y) - I(x, y))^2 + (I(x, y+1) - I(x, y))^2} \\ \varphi &= \frac{\pi}{2} + \tan^{-1} \left(\frac{I(x, y+1) - I(x, y)}{I(x+1, y) - I(x, y)} \right)\end{aligned}$$

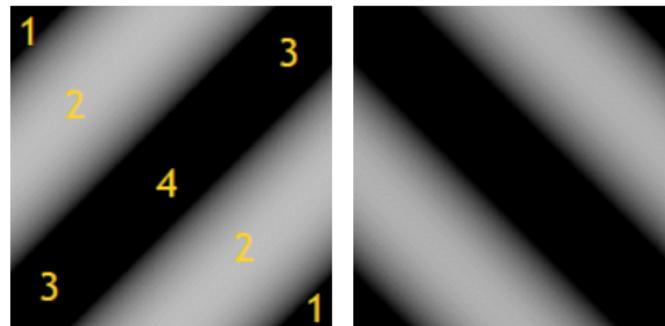
Filtres de détection de contours

- ▶ Un filtre : deux masques
 - ▶ Détection des contours horizontaux et verticaux G_x et G_y
 - ▶ Calcul de G
- ▶ Détection de contours dans le domaine spatial : convolution entre le filtre spatial et l'image
- ▶ Détection de contours dans le domaine fréquentiel : multiplication entre le filtre fréquentiel et la *DFT* de l'image
- ▶ Trois filtres courants
 - ▶ Roberts (de moins en moins utilisé)
 - ▶ Prewitt
 - ▶ Sobel

Filtres de détection de contours : Roberts

- ▶ Filtres :

$$G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$



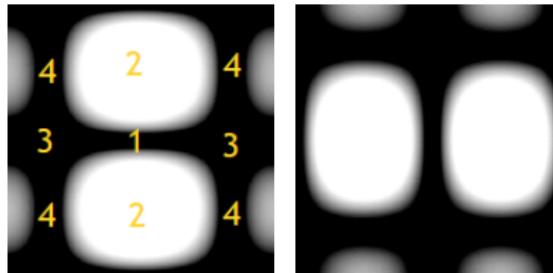
1. Certaines hautes fréquences sont éliminées par le filtrage : contours non détectés
2. Les contours \ sont préservés
3. Les contours / sont éliminés
4. La fréquence continue est éliminée

Filtres de détection de contours : Prewitt

► Filtres :

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

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



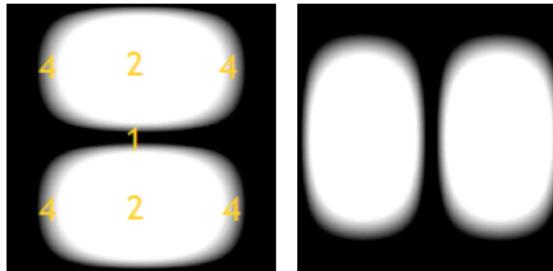
1. Annulation de la fréquence continue
2. Filtrage “passe-haut” en y : met en évidence les contours horizontaux
3. Filtrage “passe-bas” en x : élimination des contours dans la direction verticale
4. Perte de quelques contours horizontaux

Filtres de détection de contours : Sobel

► Filtres :

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

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



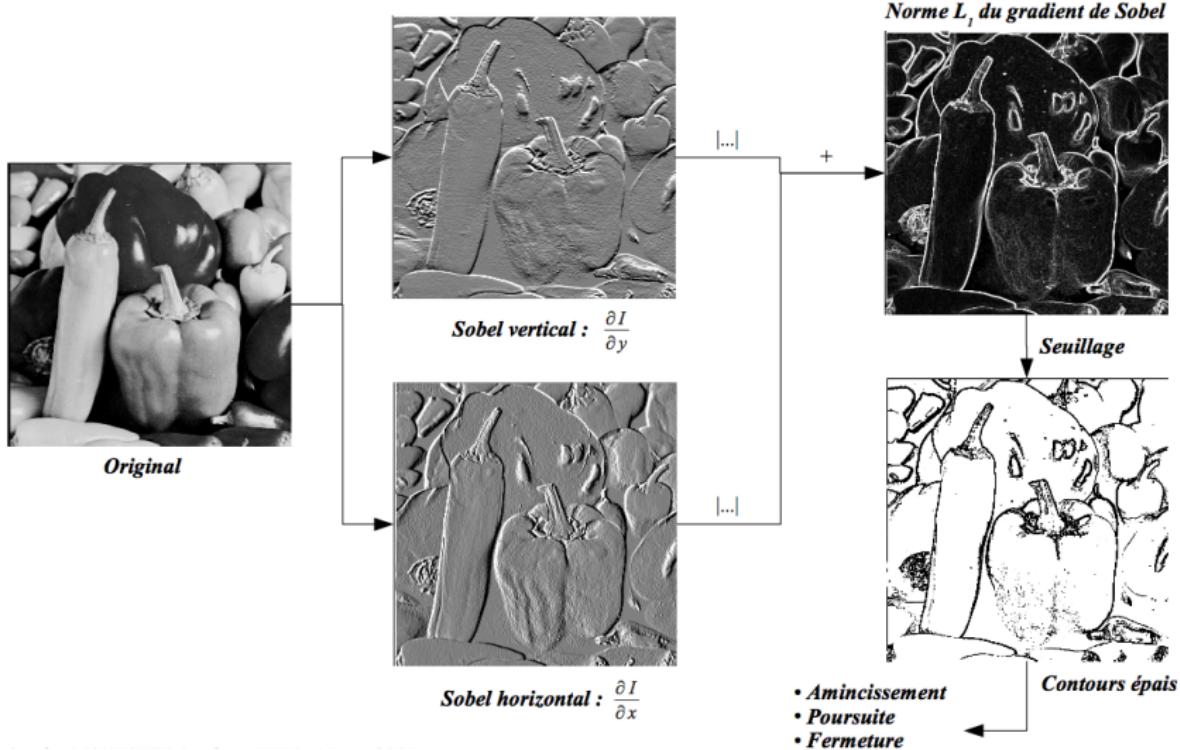
1. Annulation de la fréquence continue
2. Filtrage “passe-haut” en y : met en évidence les contours horizontaux
3. Filtrage “passe-bas” en x : élimination des contours dans la direction verticale
4. Perte de moins de contours horizontaux

Quelques remarques sur les filtres de Sobel et Prewitt

$$\begin{bmatrix} -1 & 0 & 1 \\ -c & 0 & c \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ c \\ 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

- ▶ Ces filtres sont dits séparables
 - ▶ Lissage selon la direction verticale (resp. horizontale)
 - ▶ Dérivation selon la direction horizontale (resp. verticale)
 - ▶ Sobel = filtre gaussien + gradient
 - ▶ Prewitt = filtre moyenneur + gradient
- ▶ Le lissage permet d'atténuer le bruit, avant de détecter les contours
- ▶ Sobel réduit moins le bruit mais préserve mieux l'information de voisinage des contours

Exemple d'application : détection de contour par Sobel



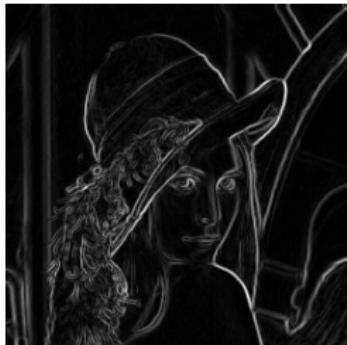
Filtres : résultats comparatifs



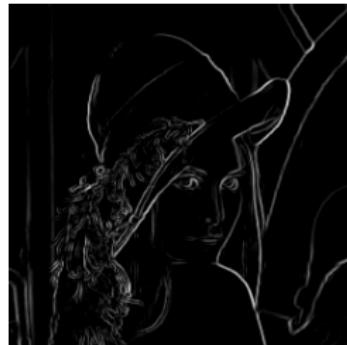
(a) orig



(b) Prewitt



(c) Sobel



(d) Kirsch

Gradient boussole

- ▶ Opérateur à 8 masques, chacun correspondant à une direction préférentielle obtenue par rotations de $\frac{\pi}{8}$ successives
- ▶ Le gradient retenu est celui correspondant à la valeur maximum, donnée par $\max_{i=0, \dots, 7} \{ |I * H_i| \}$
- ▶ L'orientation retenue est celle du masque ayant maximisé le gradient
- ▶ Plusieurs versions :

$$\left(\frac{1}{15}\right) \begin{array}{|c|c|c|} \hline 5 & 5 & 5 \\ \hline -3 & 0 & -3 \\ \hline -3 & -3 & -3 \\ \hline \end{array}$$

Kirsch

$$\left(\frac{1}{3}\right) \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

Robinson
(niveau 3)

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

Robinson
(niveau 5)

Gradient boussole

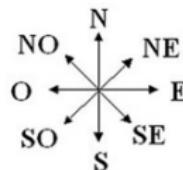
Exemple :
Robinson (3)

1	1	0
1	0	-1
0	-1	-1

1	1	1
0	0	0
-1	-1	-1

0	1	1
-1	0	1
-1	-1	0

1	0	-1
1	0	-1
1	0	-1



0	-1	-1
1	0	-1
1	1	0

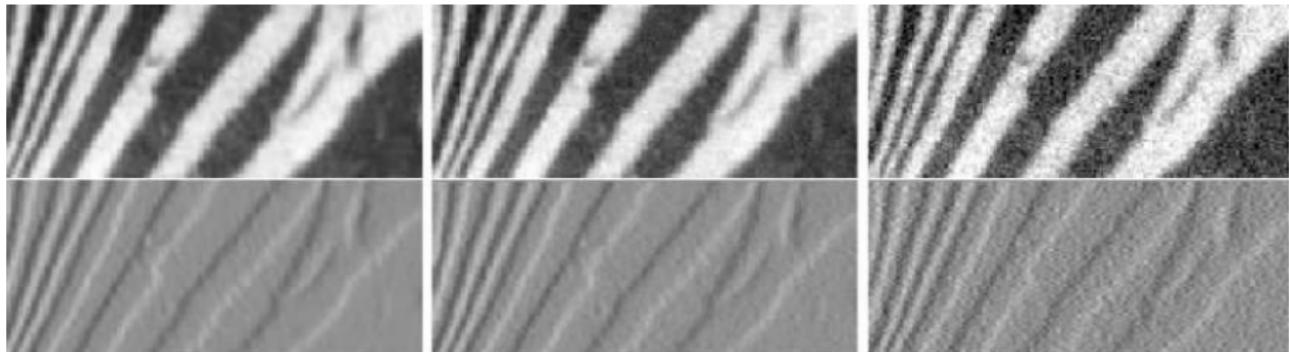
-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

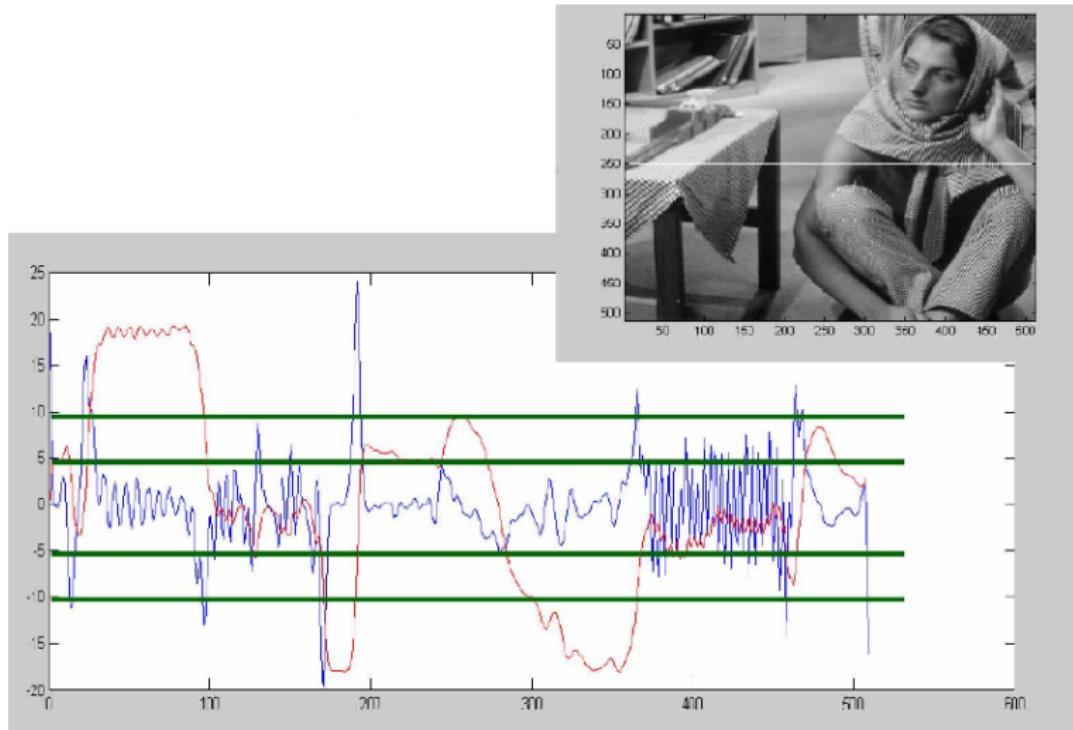
-1	-1	0
-1	0	1
0	1	1

Détection de contours : seuillage du gradient

- ▶ Méthode simple :
 1. Calcul de la norme du gradient en tout point de l'image
 2. Sélection des pixels à l'aide d'un seuil fixé *a priori* pour la norme du gradient
- ▶ Pas satisfaisant : seuil difficile à fixer
- ▶ Comment différencier un contour d'un bruit ?



Détection de contours : seuillage de la norme du gradient



Détection de contours : seuillage du gradient



Gradient



Gradient seuillé ($|G| > G_{min}$)



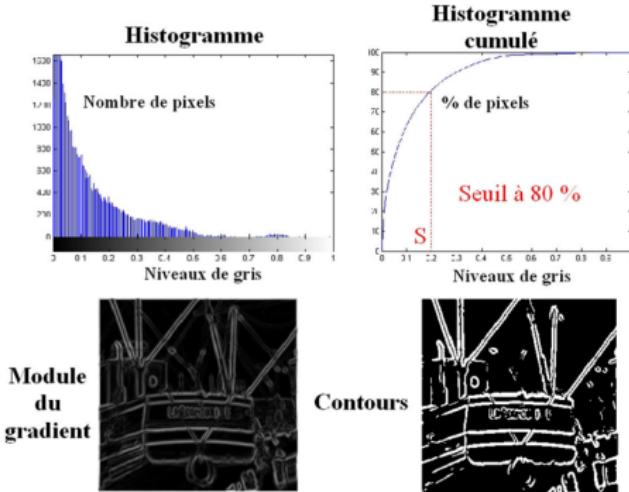
Seuil faible



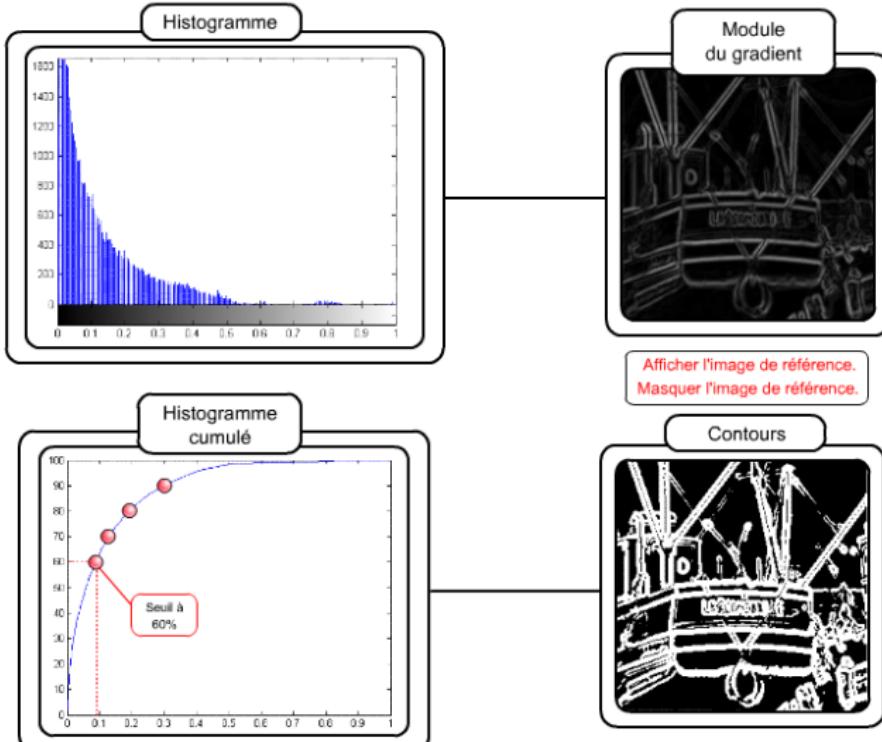
Seuil grand

Détection de contours : seuillage global par histogramme

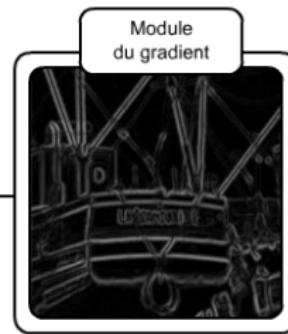
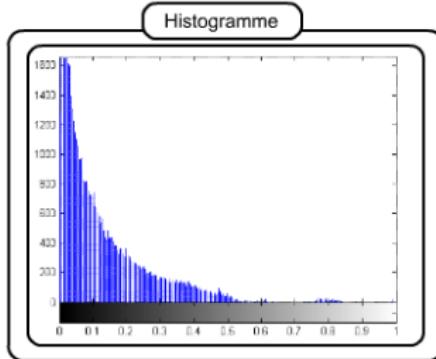
- ▶ Seuil fixé sur l'histogramme (de la norme du gradient)
- ▶ On veut sélectionner un pourcentage des contours les plus significatifs (exemple : garder 20% des contours les plus forts)
- ▶ On travaille sur l'histogramme cumulé, qui nous donne directement ce pourcentage



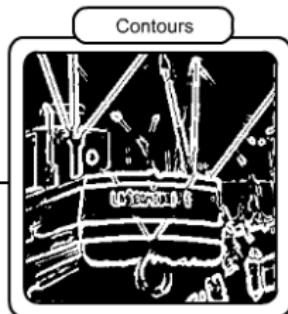
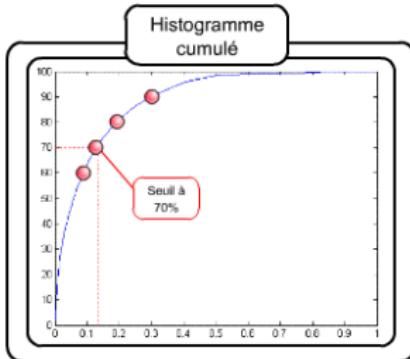
Détection de contours : seuillage global par histogramme



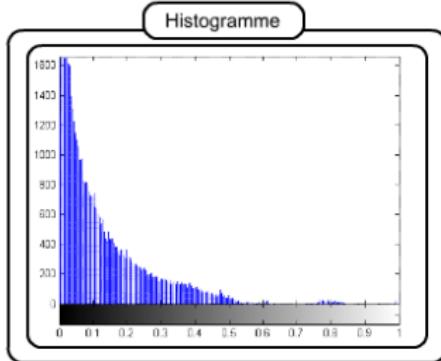
Détection de contours : seuillage global par histogramme



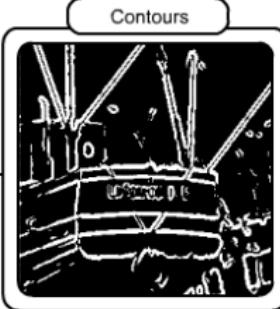
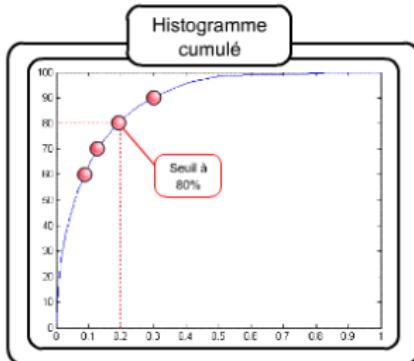
Afficher l'image de référence.
Masquer l'image de référence.



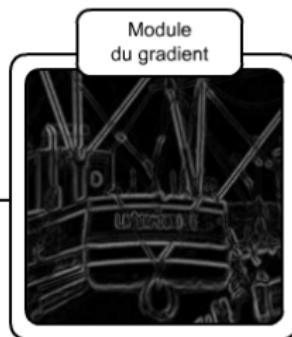
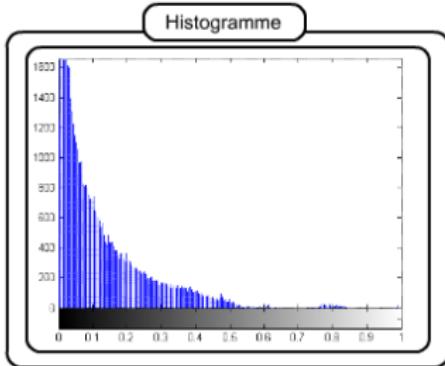
Détection de contours : seuillage global par histogramme



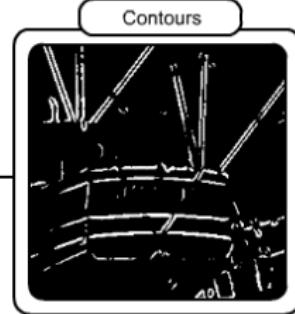
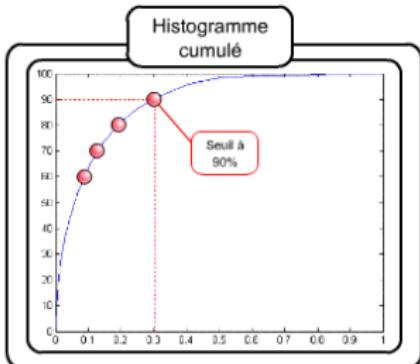
Afficher l'image de référence.
Masquer l'image de référence.



Détection de contours : seuillage global par histogramme



Afficher l'image de référence.
Masquer l'image de référence.



Détection de contours : seuillage par hystérésis

- ▶ Utilise deux informations pour traiter un pixel : son intensité et celle des pixels de son voisinage
- ▶ Algorithme :
 1. Choisir 2 niveaux de seuil :
 - ▶ s_B (seuil bas), pour faire ressortir les éléments désirés
 - ▶ s_H (seuil haut), pour faire ressortir les éléments désirés, avec des manquements
 2. Marquer tous les pixels dont le niveau de gris est $> s_H$
 3. Pour tous les pixels adjacents aux pixels marqués : marquer si leur niveau de gris est $> s_B$
 4. Répéter l'étape 3 pour tous les pixels marqués dans l'étape 2
- ▶ Choix des seuils interactivement :
 1. Choisir d'abord $S_H = S_B$ haut, de façon à ne garder que les composantes connexes d'intérêt.
 2. Descendre ensuite progressivement S_B , voire le mettre à une valeur proche du minimal.

Détection de contours : seuillage par hystérésis

- ▶ Seuillage : $s_H = 100$



Détection de contours : seuillage par hystérésis

- ▶ Seuillage : $s_H = 80$



Détection de contours : seuillage par hystérésis

- ▶ Seuillage : $s_H = 60$



Détection de contours : seuillage par hystérésis

- ▶ Seuillage : $s_H = 60$, $s_B = 60$



Détection de contours : seuillage par hystérésis

- ▶ Seuillage : $s_H = 60$, $s_B = 50$



Détection de contours : seuillage par hystérésis

- ▶ Seuillage : $s_H = 60$, $s_B = 40$



Détection de contours : seuillage par hystérésis

- ▶ Seuillage : $s_H = 60$, $s_B = 30$



Détection de contours : seuillage par hystérésis

- ▶ Seuillage : $s_H = 60$, $s_B = 20$



Détection de contours : seuillage par hystérésis

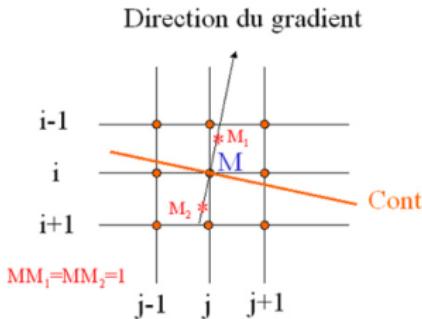
- ▶ Seuillage : $s_H = 60$, $s_B = 10$



Détection de contours : extraction des maxima dans la direction du gradient

- ▶ But : conserver uniquement les **maxima locaux de la norme du gradient** dans la **direction du gradient**
- ▶ Principe : comparer la norme du gradient en tout point $M(i, j)$ de l'image avec celles de ses deux voisins M_1 et M_2 de part et d'autre dans la direction du gradient à une distance unitaire
- ▶ Problème : M_1 et M_2 ne sont pas connus, il faut les interpoler
- ▶ L'algorithme met en évidence les maxima locaux dans des directions privilégiées
- ▶ Il faut sélectionner les maxima à conserver en utilisant une technique de seuillage
- ▶ M est un maximum local si $G_M > G_{M_1}$ et $G_M > G_{M_2}$

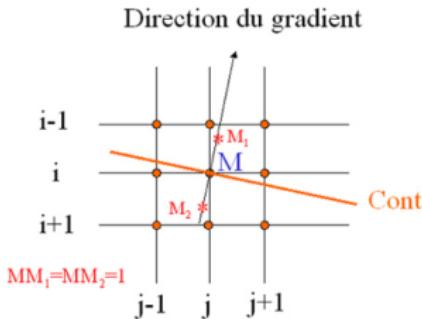
Détection de contours : extraction des maxima dans la direction du gradient



- ▶ Si $|g_V| > |g_H|$:

$$G_{M_1} = \frac{g_H}{g_V} G(i-1, j) + \frac{g_V - g_H}{g_V} G(i-1, j+1)$$
$$G_{M_2} = \frac{g_H}{g_V} G(i+1, j) + \frac{g_V - g_H}{g_V} G(i+1, j-1)$$

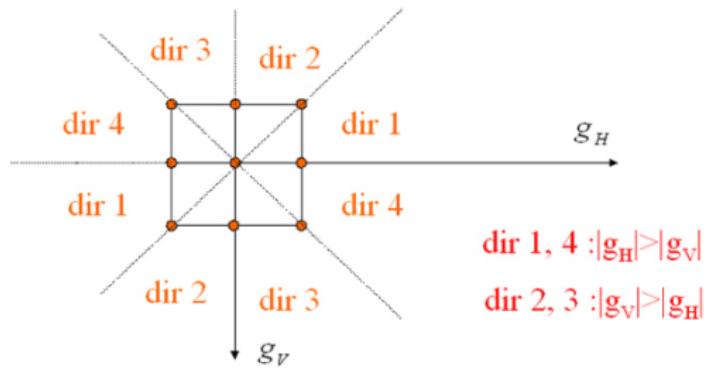
Détection de contours : extraction des maxima dans la direction du gradient



- ▶ Si $|g_H| > |g_V|$:

$$G_{M_1} = \frac{g_V}{g_H} G(i, j + 1) + \frac{g_H - g_V}{g_H} G(i - 1, j + 1)$$
$$G_{M_2} = \frac{g_V}{g_H} G(i, j - 1) + \frac{g_H - g_V}{g_H} G(i + 1, j - 1)$$

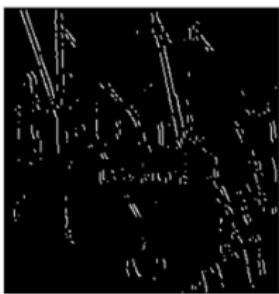
Détection de contours : extraction des maxima dans la direction du gradient



dir 1 :	$g_H \leq 0, g_V \geq 0, -g_H > g_V$	$g_H \geq 0, g_V \leq 0, g_H > -g_V$
dir 2 :	$g_H \leq 0, g_V \geq 0, -g_H < g_V$	$g_H \geq 0, g_V \leq 0, g_H < -g_V$
dir 3 :	$g_H \leq 0, g_V \leq 0, -g_H < -g_V$	$g_H \geq 0, g_V \geq 0, g_H < g_V$
dir 4 :	$g_H \leq 0, g_V \leq 0, g_H < g_V$	$g_H \geq 0, g_V \geq 0, g_H > g_V$

Détection de contours : extraction des maxima dans la direction du gradient

Direction 1



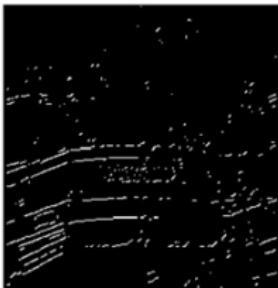
Module gradient



Direction 2



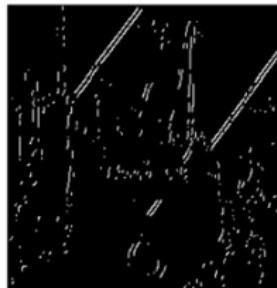
Direction 3



Maxima locaux



Direction 4



Détection de contours : extraction des maxima dans la direction du gradient



Affinage (*thinning*)

- ▶ On travaille sur l'image binaire contenant les points de contour
- ▶ La suppression d'un point de contour ne doit pas changer le nombre de composantes connectées dans l'image binaire
- ▶ Un point final est un pixel blanc ayant exactement un voisin
- ▶ Si on considère un voisinage 3×3 avec comme point central p un pixel blanc (point de contour), alors :
 - ↪ p est un point simple si on peut le changer de blanc en noir sans changer le nombre de composantes connectées dans le voisinage

Affinage (*thinning*)

1 1 1

- Exemple 1 de voisinage : 0 1 1

0 1 0

⇒ p est simple-8 mais pas simple-4

0 0 0

- Exemple 2 de voisinage : 1 1 1

0 0 0

⇒ p n'est ni simple-8 ni simple-4

- Un 1-pixel blanc $I(x, y)$ (ou point final) est un bord- N si son voisin de coordonnée $(x, y - 1)$ est noir (définitions similaires avec les directions S , E et O)

Affinage (*thinning*)

- ▶ Un algorithme simple (pour N, E, S et O) :

for $D = N$ à O **do**

 Éliminer tous les bords-D qui sont **simples** et ne sont **pas des points finaux**

end for

- ▶ Chaque direction est traitée séparément afin d'éviter l'élimination de composantes connexes entières
- ▶ Le résultat dépend de l'ordre de traitement des directions

Affinage (*thinning*)

- Exemple d'application $N \rightarrow E \rightarrow O \rightarrow S$:

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

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

Affinage (*thinning*)

► Avant/Après

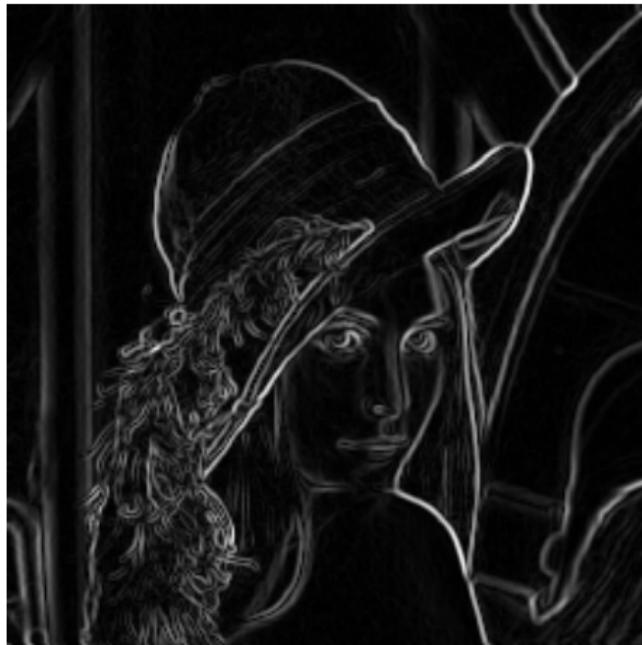
1	1	1	1		1	1	1	1	
1	1	1		1	1	1	1	1	
1	1	1	1	1	1	1		1	1
1	1	1	1	1	1	1	1	1	1
1	1			1	1				

Affinage : exemple sur image naturelle



Cette chère Léna ...

Affinage : exemple sur image naturelle



```
detc -sob lena.inr > lena-sob.inr
```

Affinage : exemple sur image naturelle

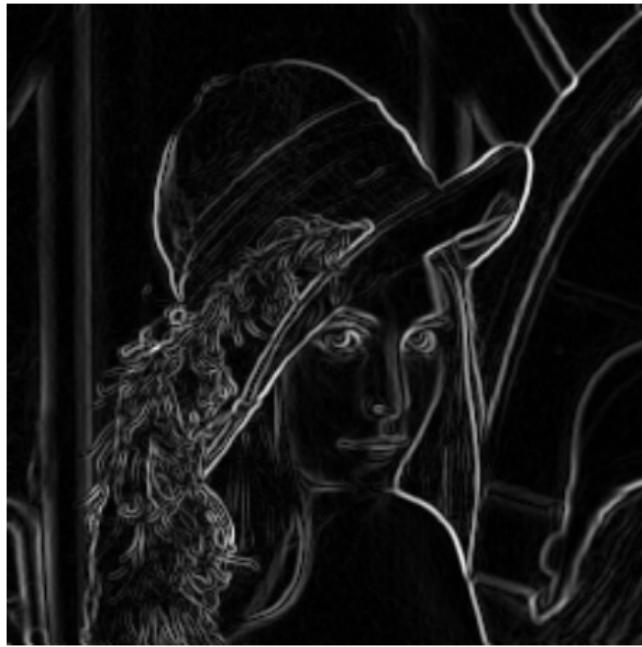


Image de norme, il faut seuiller ! Quelle valeur ?

ical lena-sob.inr donne :

0.000000 0.253705 2.958022

Affinage : exemple sur image naturelle



mh -n 0.6 lena-sob.inr > lena-mh.inr trop!

Affinage : exemple sur image naturelle



mh -n 1 lena-sob.inr > lena-mh.inr pas mal...

Affinage : exemple sur image naturelle



Affinons : aff lena-mh.inr > lena-aff.inr

Opérateurs dérivatifs du second ordre : laplacien

- ▶ Le laplacien est une approximation de la dérivée seconde de l'image
- ▶ Le laplacien est un scalaire (le gradient est un vecteur) :

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

- ▶ On cherche les passages par zéro du laplacien pour localiser les contours (transitions)

Approximation du laplacien : cas discret

- ▶ Les dérivées secondes au point (x, y) sont approchées par les différences finies :

$$\frac{\partial^2 I(x, y)}{\partial x^2} \approx I(x+1, y) - 2I(x, y) + I(x-1, y)$$

$$\frac{\partial^2 I(x, y)}{\partial y^2} \approx I(x, y+1) - 2I(x, y) + I(x, y-1)$$

- ▶ Le laplacien s'obtient par convolution avec un noyau :

$$\frac{\partial^2 I(x, y)}{\partial x^2} + \frac{\partial^2 I(x, y)}{\partial y^2} \approx$$

0	0	0
1	-2	1
0	0	0

$$+$$

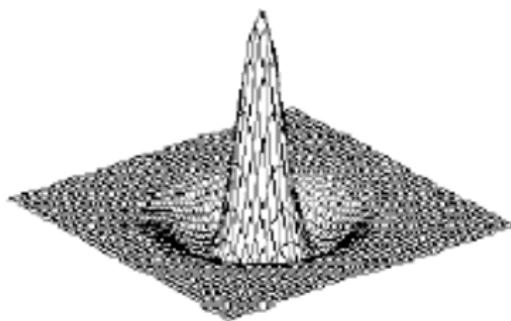
0	1	0
0	-2	0
0	1	0

$$\approx$$

0	1	0
1	-4	1
0	1	0

Détection de contours avec le laplacien

- ▶ Calculer le laplacien en chaque point de l'image (dans le pratique, on calcule son opposé)
- ▶ Déetecter les passages par zéro du laplacien
- ▶ Sélectionner les pixels pour lesquels le laplacien change de signe
- ▶ Seuiller afin d'éliminer les passages par zéro non significatifs

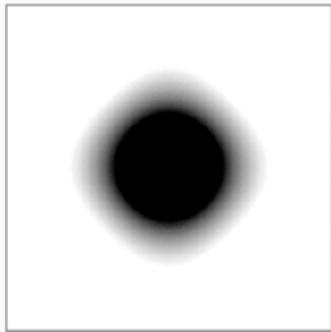


Calcul du laplacien : utilisation de filtres

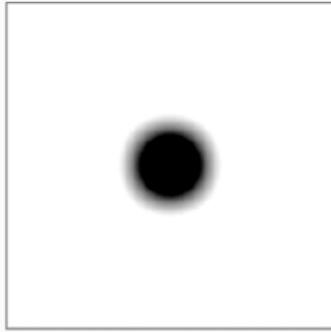
- ▶ Filtre : un seul masque pour l'approximation de l'opposé du laplacien
- ▶ Détection de contours dans le domaine spatial :
 - ▶ Convolution entre le filtre spatial et l'image
- ▶ Détection de contours dans le domaine fréquentiel :
 - ▶ Multiplication entre le filtre fréquentiel et la *DFT* de l'image
- ▶ L'opposé du laplacien est un filtre passe-haut :
 - ▶ Il élimine les basses fréquences
 - ▶ Il met en valeur les hautes fréquences
- ▶ Le laplacien amplifie le bruit, car il garde et met en valeur les plus hautes fréquences

Exemples de filtres

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$



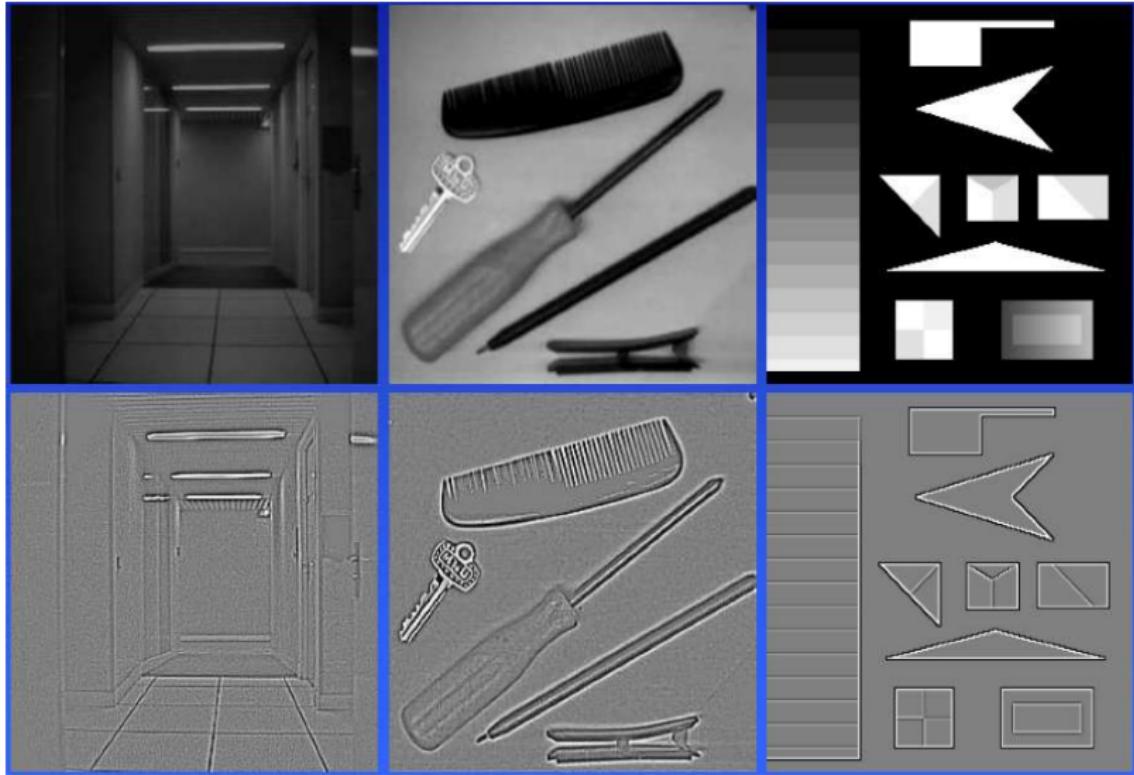
1. Elimine beaucoup de BF (dont la fréquence continue)
2. L'élimination des BF se fait de manière progressive
3. Les fréquences les plus hautes sont gardées : bruit

1. Elimine moins de BF (dont la fréquence continue)
2. L'élimination des BF se fait de manière plus brutale
3. Les fréquences les plus hautes sont gardées : bruit

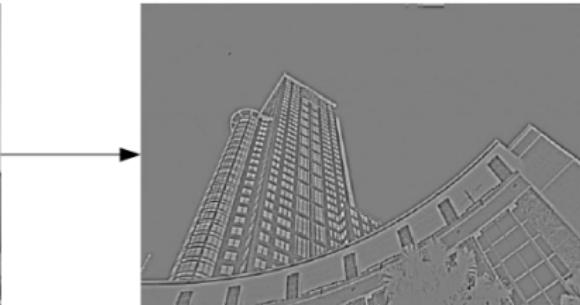
Exemple d'application : filtre laplacien



Exemple d'application : filtre laplacien

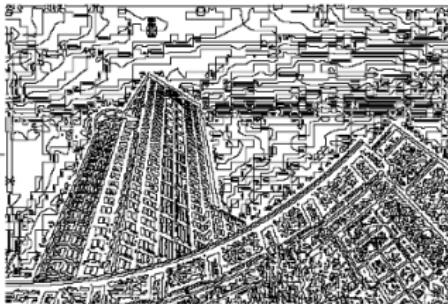


Laplacien : détection des passages par zéro



Laplacien

- Seuiller les passages par zéro selon le contraste.
- Sélectionner les structures en fonction de l'échelle



Passages par zéro

$$\begin{array}{cc} \begin{matrix} + & + \\ + & - \\ + & + \end{matrix} & \begin{matrix} + & + \\ - & + \\ + & + \end{matrix} \\ \dots & \dots \end{array}$$

Détection des changements de polarité dans un voisinage 2×2

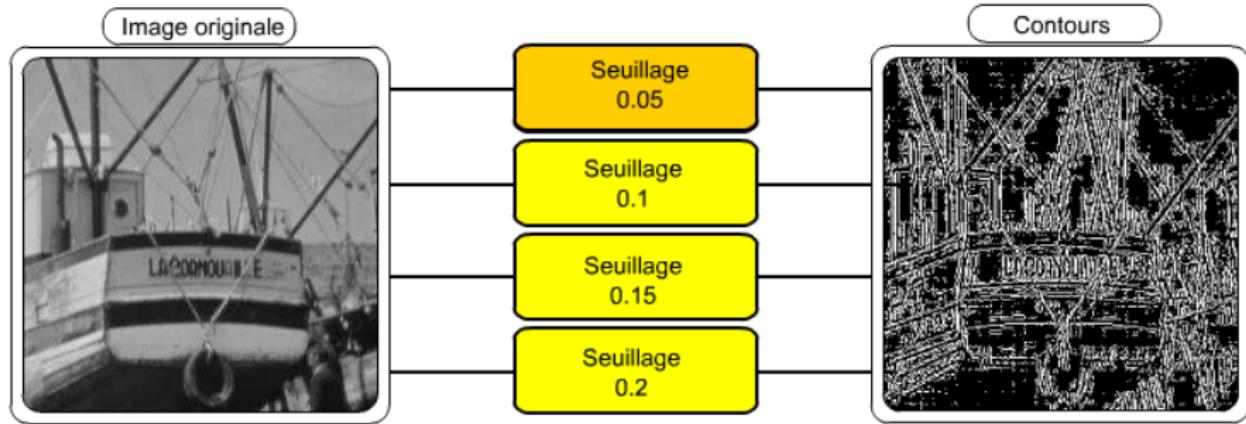
Laplacien : détection des passages par zéro

- ▶ But : détecter les passages par zéro en conservant uniquement les passages les plus marqués
- ▶ Attention : ne pas considérer le bruit, qui peut très bien se traduire par des oscillations autour de zéro, comme un contour
- ▶ On regarde les voisins de chaque pixel :

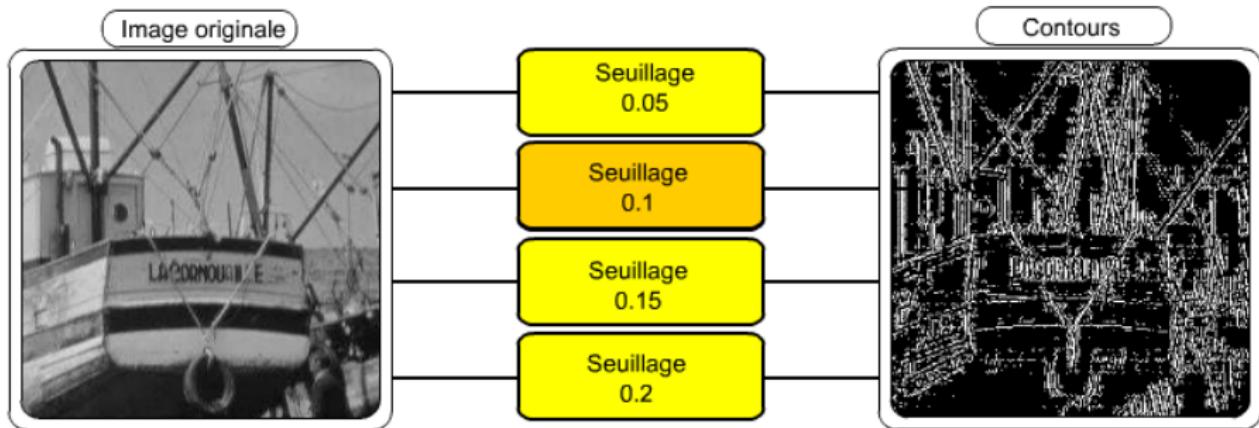
0	Δ_1	0
Δ_2	Δ	Δ_3
0	Δ_4	0

- ▶ Si $\Delta > 0$ et l'un des autres $\Delta_i \leq 0$ il y a changement de signe
- ▶ Si $\Delta < 0$ et l'un des autres $\Delta_i \geq 0$ il y a changement de signe
- ▶ Seuillage des passages par zéro de forte amplitude

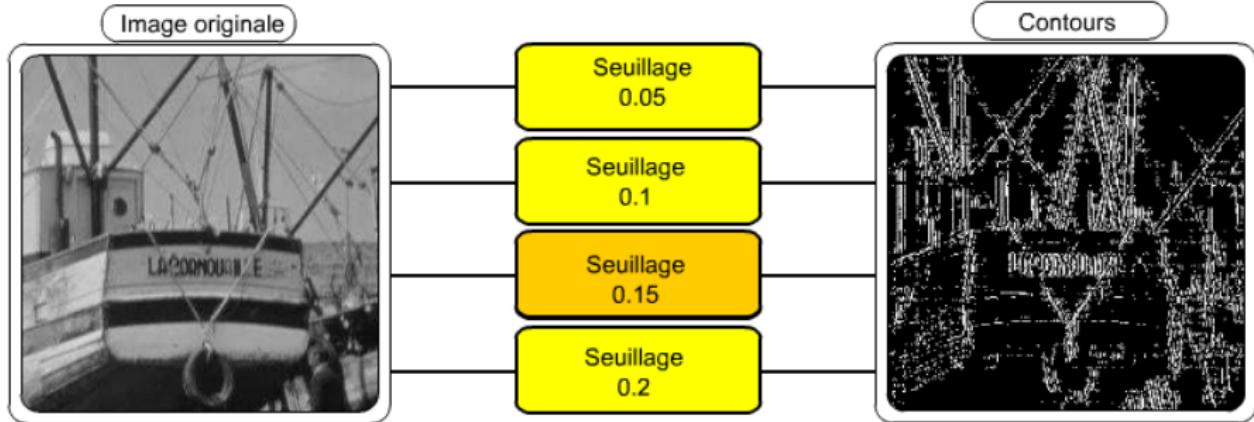
Laplacien : détection des passages par zéro



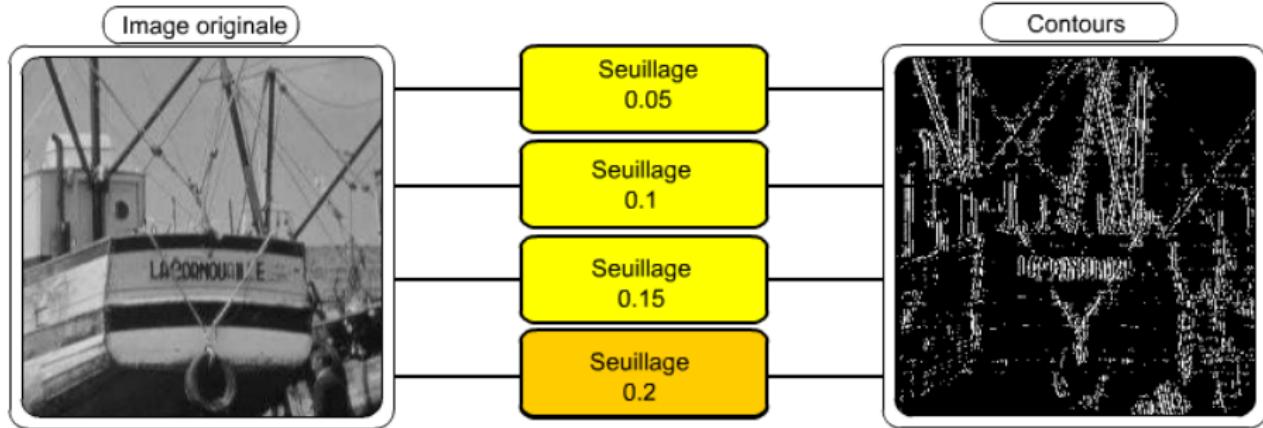
Laplacien : détection des passages par zéro



Laplacien : détection des passages par zéro



Laplacien : détection des passages par zéro



Détection de contours : méthode de Canny-Deriche

- ▶ J. Canny a formalisé trois critères qui valident un détecteur de contour
 1. Bonne détection : robustesse au bruit, forte réponse même à de faibles contours
 2. Bonne localisation : précision de la localisation du point de contour
 3. Unicité : une seule détection (i.e. réponse) par contour
- ▶ A chaque critère est associé une formule que l'on cherche à maximiser
 - ▶ La solution (hors du cadre de ce cours) est alors le filtre (ici donné en 1D) :

$$f(x) = Sx e^{-\alpha|x|}$$

Il ressemble au filtre gaussien dérivé. Pourquoi le gaussien dérivé ?

Détection de contours : méthode de Canny-Deriche

- ▶ J. Canny a formalisé trois critères qui valident un détecteur de contour
 1. Bonne détection : robustesse au bruit, forte réponse même à de faibles contours
 2. Bonne localisation : précision de la localisation du point de contour
 3. Unicité : une seule détection (i.e. réponse) par contour
- ▶ A chaque critère est associé une formule que l'on cherche à maximiser
 - ▶ La solution (hors du cadre de ce cours) est alors le filtre (ici donné en 1D) :

$$f(x) = Sx e^{-\alpha|x|}$$

Il ressemble au filtre gaussien dérivé. Pourquoi le gaussien dérivé ? : $\frac{\partial}{\partial x}(I * g) = I * \frac{\partial g}{\partial x}$

Si g est le filtre gaussien, on obtient une formule de dérivé "lissante"

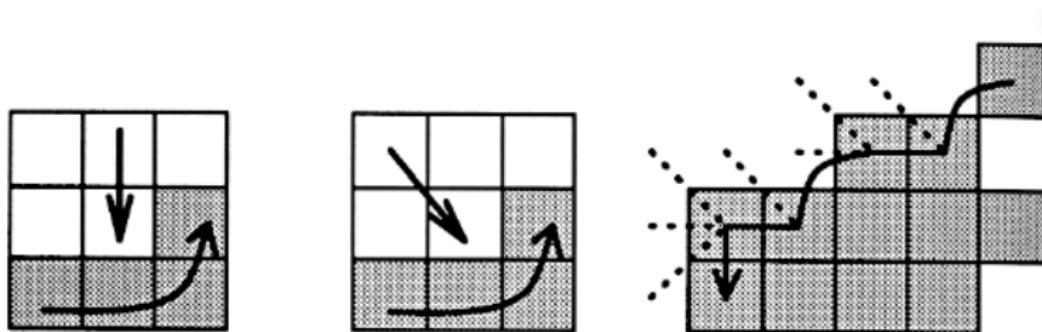
- ▶ À ce jour la méthode la plus efficace

Suivi et fermeture de contours

- ▶ L'extraction de points de contour nous permet d'obtenir une image binaire :
 - ▶ Pixels blancs : points de contour
 - ▶ Pixels noirs : autres pixels de l'image
- ▶ Problèmes à résoudre :
 - ▶ Certains points de contour n'ont pas été détectés
 - ▶ Certains pixels détectés comme points de contour n'en sont pas
- ▶ Nécessité d'obtenir les contours fermés des objets/régions de l'image
- ▶ Beaucoup de solutions ont été proposées
- ▶ La qualité des résultats obtenus est liée au coût informatique

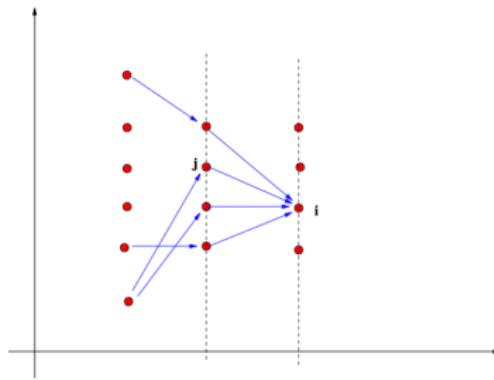
Suivi de contours : connexité

- ▶ Ce qu'il y a de plus simple
- ▶ Utilisation de la connexité (d'ordre 4 ou 8) pour relier entre eux les points de contour extraits
- ▶ Problème quand un point de contour n'a aucun voisin détecté également point de contour



Suivi de contours : représentation par un graphe

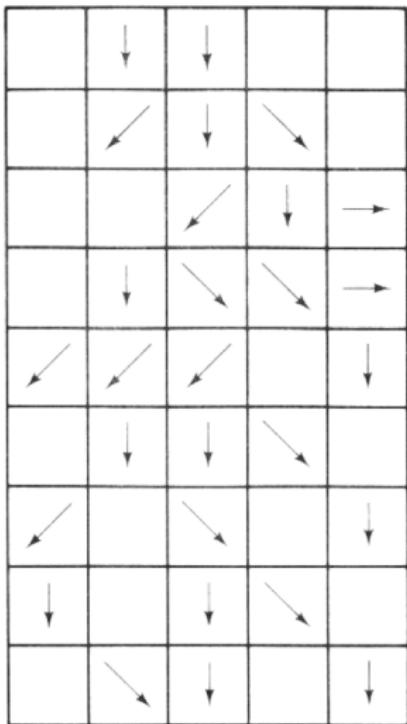
- ▶ But : créer une représentation structurée en graphe de l'image de contours
- ▶ On part d'une image binaire E des contours
- ▶ L'image est vue comme un maillage (i.e. graphe)
- ▶ Un pixel "point de contour" est un sommet de ce graphe
- ▶ Suivi de contour = recherche d'un chemin entre deux sommets du graphe



Suivi de contours : représentation par un graphe

- ▶ Exemple : on connaît l'orientation du gradient en chaque point de contour
- ▶ Deux pixels voisins détectés "points de contour" peuvent être reliés si les directions de leur gradient sont proches
- ▶ Application : pour relier le pixel a au pixel b , b doit être un des huit voisins de a dans la direction du contour en a ($\phi(a)$), soit :

$$|(\phi(a) - \phi(b)) \bmod 2\pi| < \frac{\pi}{2}$$



Suivi de contours : recherche heuristique

- ▶ Soient a et b deux extrémités d'un contour
- ▶ Pour tout pixel p situé sur un chemin solution, on définit une fonction d'évaluation :

$$f(p) = g(p) + h(p)$$

où :

- ▶ $g(p)$ est le coût du chemin de a à p
- ▶ $h(p)$ est le coût du chemin de p à b
- ▶ Une solution : l'algorithme A^*

Suivi de contours : recherche heuristique

- ▶ Principe de l'algorithme A^* (1962)

$n = a$

while $n \neq b$ **do**

 Placer les successeurs de n dans L

 Déterminer p tel que $f(p) = \min_{q \in L}(f(q))$

$n = p$

 Ajouter p à C

end while

if C est vide **then**

 Échec

else

C contient le chemin optimal de a vers b

end if

Suivi de contours : recherche heuristique

- ▶ Quelques valeurs intervenant dans le calcul de h :
 - ▶ Importance de l'arête : $||\max_p(\nabla I(p))|| - ||\nabla I(p)||$
 - ▶ Courbure : $\phi(p) - \phi(p')$
 - ▶ Distance à l'objectif : $d = \text{dist}(p, b)$
- ▶ Exemple d'illustration de fonction de coût : somme (inverse) des modules du gradient

