

Signaux, Sons et Images pour l'Informaticien: Extraction de primitives dans les images

Diane Lingrand

Polytech SI3

2018 - 2019

Outline

1 Un premier détecteur de contour

2 Convolution

3 Dérivée première

4 Dérivée seconde

5 Approche Canny - Deriche

6 Détection de points d'intérêt

Pourquoi extraire des primitives ?

- détecter des objets dans une scène
- effectuer des mesures
- vers une compression vectorielle
- décrire une image
- reconnaître des objets

- contours
 - point à point
 - droites, lignes brisées, polygones
 - cercle, carré, ...
- points d'intérêt ou coins (*corners*)
- régions
 - intensité uniforme, texture, histogramme

Outline

1 Un premier détecteur de contour

2 Convolution

3 Dérivée première

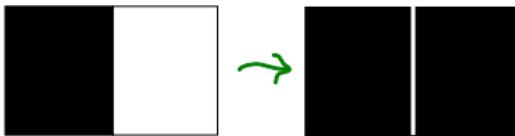
4 Dérivée seconde

5 Approche Canny - Deriche

6 Détection de points d'intérêt

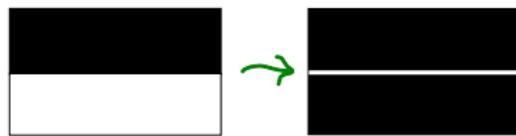
contours verticaux

$$I_{cv}[i][j] = I[i][j] - I[i-1][j]$$



contours horizontaux

$$I_{ch}[i][j] = I[i][j] - I[i][j-1]$$



Tous les contours :

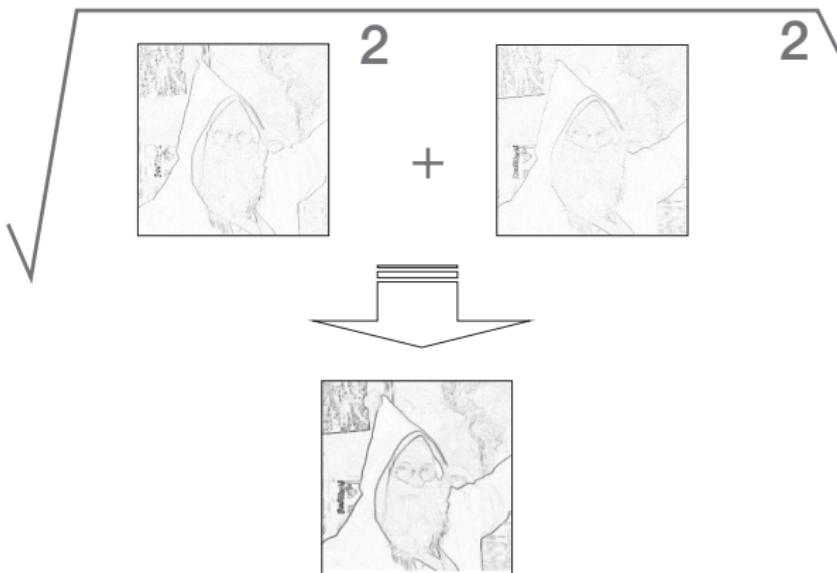
$$I_{\text{contours}}[i][j] = \sqrt{I_{cv}^2[i][j] + I_{ch}^2[i][j]}$$

Sur une image réelle

extraction des
contours
verticaux



extraction des
contours
horizontaux



- inconvénients :
 - sensible au bruit
 - détecte trop de contours
 - détection + forte des contours horizontaux et verticaux que les contours obliques
- avantages :
 - facile à programmer
 - rapide

Outline

1 Un premier détecteur de contour

2 Convolution

3 Dérivée première

4 Dérivée seconde

5 Approche Canny - Deriche

6 Détection de points d'intérêt

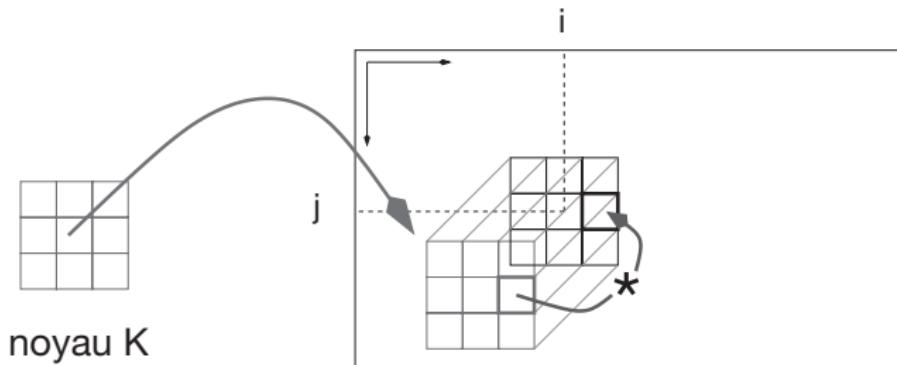
Convolution

- Pour f et g fonctions discrètes :

$$f * g(x, y) = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} f(x, y)g(u - x, v - y)$$

- Convolution d'une image I_1 par un noyau K de dimension $(2p + 1) \times (2q + 1)$:

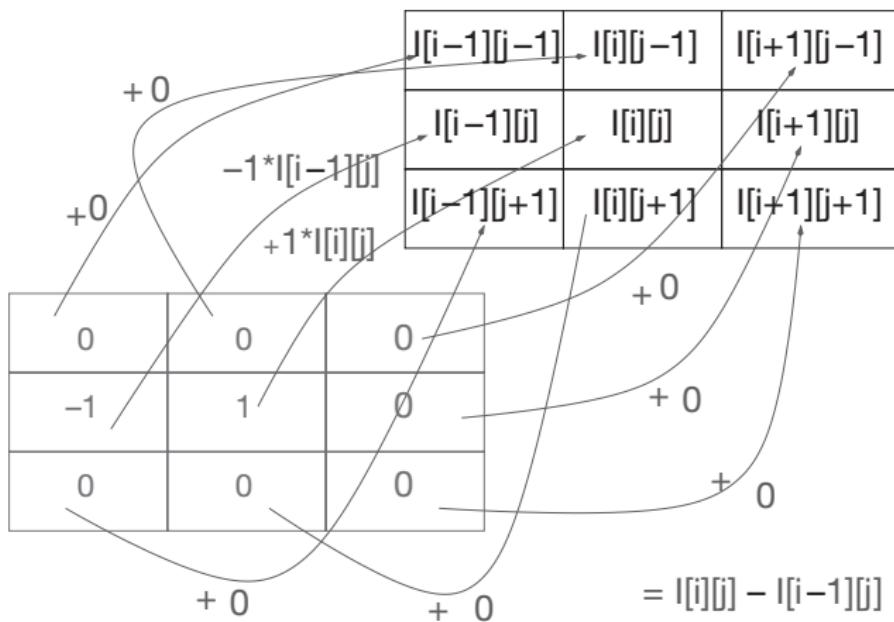
$$I_2[i][j] = \sum_{k=0}^{2p} \sum_{l=0}^{2q} I_1[i - k + p][j - l + q]K[k][l]$$



Retour au filtre naïf

Expression sous forme de convolution par 2 filtres :

$$K_x = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad K_y = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



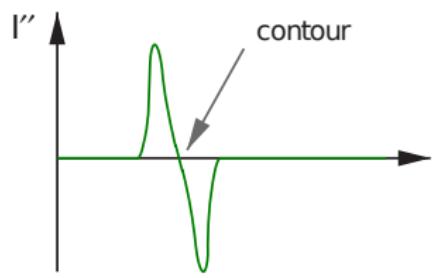
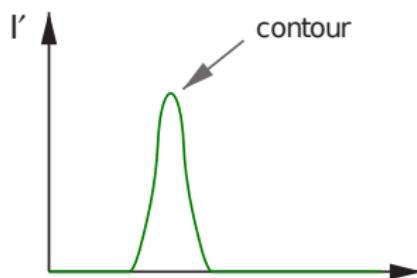
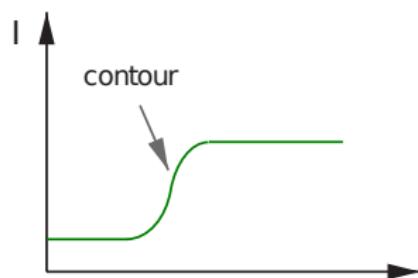
Convolution en Java

```
BufferedImage biInput, biOutput;  
[...]  
float [] filtre = {-0.0f, 0.0f, 0.0f,  
                     -1.0f, 1.0f, 0.0f,  
                     -0.0f, 0.0f, 0.0f };  
Kernel kernel = new Kernel(3, 3, filtre);  
ConvolveOp cop = new ConvolveOp( kernel,  
                               ConvolveOp.EDGE_NO_OP, null );  
cop.filter(biInput, biOutput);
```

autre option : ConvolveOp.EDGE_ZERO_FILL

☞ **attention au signe**

Contour idéal : marche



Outline

1 Un premier détecteur de contour

2 Convolution

3 Dérivée première

4 Dérivée seconde

5 Approche Canny - Deriche

6 Détection de points d'intérêt

Filtre de Roberts (1965)

$$\frac{dl}{dx} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \quad \frac{dl}{dy} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

Module = force du contours :

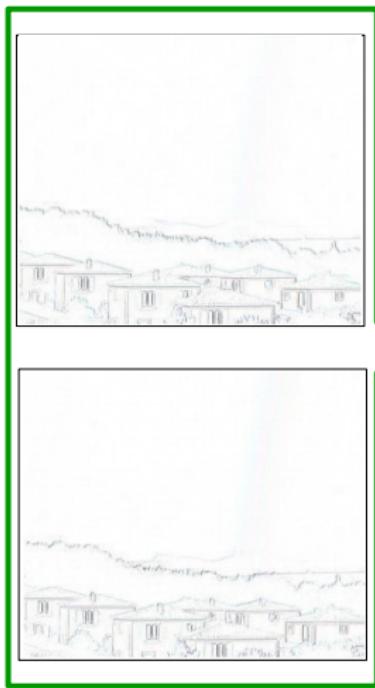
$$\sqrt{\left(\frac{dl}{dx}\right)^2 + \left(\frac{dl}{dy}\right)^2}$$

ou

$$\max\left(\frac{dl}{dx}, \frac{dl}{dy}\right)$$

Roberts en exemple

filtre en x



filtre en y

norme des résultats

- bruit = hautes fréquences
- concerne les termes en $e^{i2\pi fx}$
- dérivée première : la composante en f du bruit est multipliée par $2\pi f$
- dérivée seconde : la composante en f du bruit est multipliée par $4\pi^2 f^2$
- il faut donc atténuer le bruit : lissage

Lissage par filtre moyenneur

En dimension 3x3 :

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} / 9$$

En dimension $(2p+1) \times (2p+1)$:

$$\begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & & & \ddots \\ 1 & 1 & \dots & 1 \end{bmatrix} / (2p + 1)^2$$



3x3



5x5



7x7



9x9

Lissage par filtre gaussien

$$G_\sigma(\mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{|\mathbf{x}|^2}{2\sigma^2}}$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix} / 8 \quad \text{ou} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 1 \end{bmatrix} / 16$$



3x3



5x5



7x7

Détecteur de Sobel (1970)

combine à la fois le lissage par le filtre mono-dimensionnel [1, 2, 1] et la dérivée selon une direction perpendiculaire au lissage, obtenue par le filtre [1, 0, -1]

$$\frac{dl}{dx} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} / 4 \quad \frac{dl}{dy} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} / 4$$

Angle entre l'horizontale et la normale au contour :

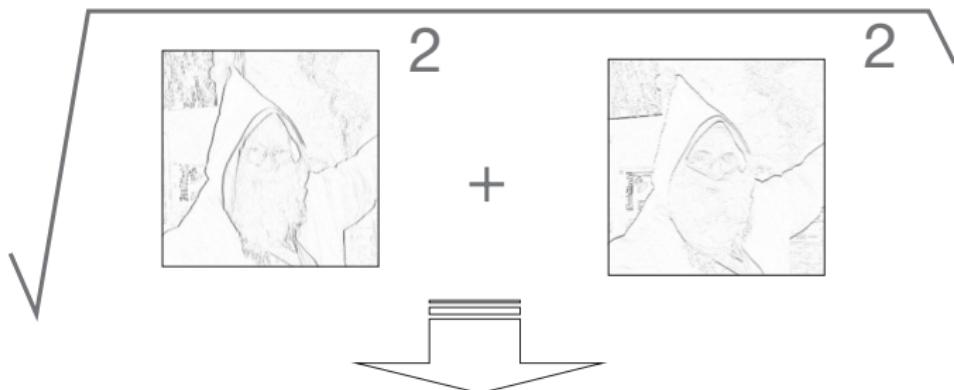
$$\arctan \left(\left(\frac{dl}{dy} \right) / \left(\frac{dl}{dx} \right) \right)$$

Sobel en exemple

extraction des
contours
verticaux



extraction des
contours
horizontaux



Détecteur de Prewitt (1970)

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} / 3, \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix} / 3, \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} / 3, \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} / 3$$



Détecteur de Kirsch (1971)

$$\begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix} / 15, \begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} / 15, \begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} / 15,$$
$$\begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix} / 15, \begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix} / 15, \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix} / 15,$$
$$\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix} / 15, \begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix} / 15$$



Seuillage

Seuil bas



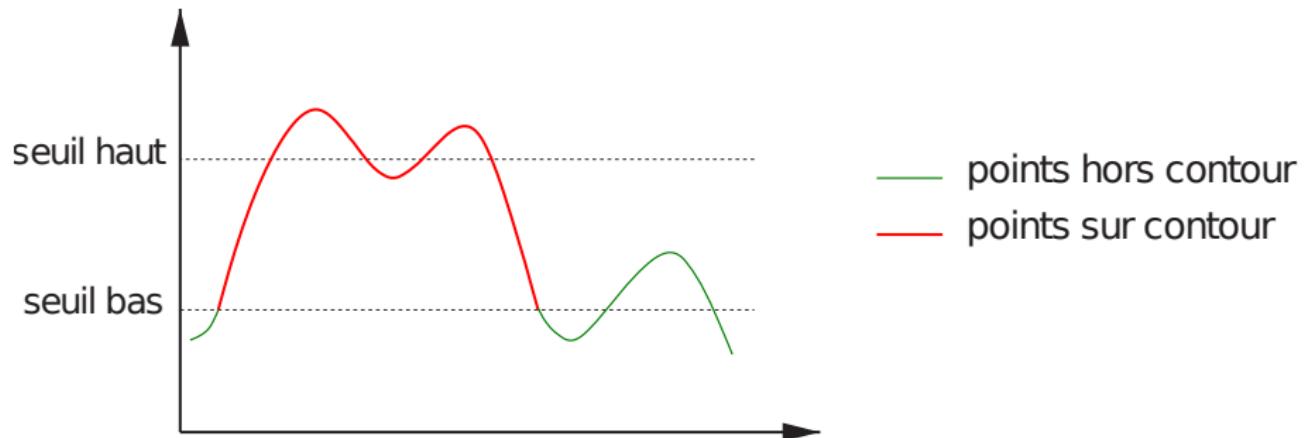
Tous les points de contours sont détectés.

Seuil haut



Tous les points détectés sont des points de contours.

Seuillage par hystérésis



Outline

1 Un premier détecteur de contour

2 Convolution

3 Dérivée première

4 Dérivée seconde

5 Approche Canny - Deriche

6 Détection de points d'intérêt

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

Noyaux de convolution en connexité 4 ou 8 :

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} / 4 \quad \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} / 8$$

Laplace en exemple

Laplace en connexité 4



Laplace en connexité 8



Laplace en connexité 8
après lissage gaussien (filtre 7x7)



Lieux des passages par zéros



Laplacien DOG - Marr et Hildreth - années 80

Laplacien vu comme différence de 2 gaussiennes. Différence entre lissage gaussien 7×7 et 5×5 :



Outline

1 Un premier détecteur de contour

2 Convolution

3 Dérivée première

4 Dérivée seconde

5 Approche Canny - Deriche

6 Détection de points d'intérêt

Un contour ?

- contour idéal : somme d'un échelon de hauteur A et d'un bruit gaussien n de moyenne nulle et de variance n_0^2 :

$$I(x) = AU(x) + n(x)$$

- détecteur de contour : 3 qualités
 - Q_1 : bonne détection
 - Q_2 : bonne localisation
 - Q_3 : faible multiplicité des maxima dus au bruit

Plus le filtre lisse le bruit, plus la détection est bonne : on cherche à maximiser le rapport signal sur bruit :

$$\Sigma = \frac{A \int_{-\infty}^0 f(x)dx}{n_0 \sqrt{\int_{-\infty}^{\infty} f^2(x)dx}}$$

Qualité 2 : bonne localisation.

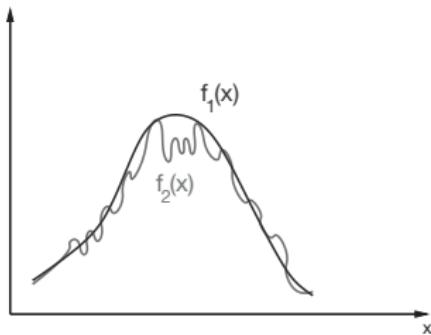
Moins le filtre lisse l'image, meilleure est la localisation : on cherche à minimiser la variance de la position des passages par zéro de la dérivée ce qui revient à maximiser :

$$\Lambda = \frac{A|f'(0)|}{n_0 \sqrt{\int_{-\infty}^{\infty} f'^2(x) dx}}$$

Qualité 3 : réponse unique.

On veut une réponse unique par contour. Il existe des cas où il est difficile de savoir si on est en présence de 2 contours distincts ou un seul contour bruité. On limite donc la distance entre 2 maxima par :

$$x_{\max} = 2\pi \sqrt{\frac{\int_{-\infty}^{\infty} f'^2(x) dx}{\int_{-\infty}^{\infty} f''^2(x) dx}}$$



C'est trouver f qui maximise le produit $\Lambda \Sigma$ sous la contrainte $x_{max} = k$, ce qui revient à une équation différentielle (on admettra le résultat) :

$$2f(x) = 2\lambda_1 f''(x) + 2\lambda_2 f'''(x) + \lambda_3 = 0$$

de solution générale :

$$f(x) = a_1 e^{\alpha x} \sin(\omega x) + a_2 e^{\alpha x} \cos(\omega x) + a_3 e^{-\alpha x} \sin(\omega x) + a_4 e^{-\alpha x} \cos(\omega x)$$

Il reste maintenant à déterminer les paramètres a_1 , a_2 , a_3 et a_4 .

Le filtre de Canny est un filtre impair à réponse impulsionnelle finie défini sur l'intervalle $[-M ; +M]$ ce qui signifie que h est nul en dehors de cet intervalle et que la dérivée est continue. Il impose également une pente de S à l'origine. Il en suit les contraintes suivantes :

$$f(0) = 0; h(M) = 0; f'(M) = 0; f(-x) = -f(x); f'(0) = S$$

permettant de déterminer les 4 coefficients a_i .

Par une optimisation numérique, Canny détermina que $\sum \Lambda = 1.12$ est un compromis optimal. Pour une mise en pratique concrète, l'implémentation se réalise à l'aide de la dérivée d'une gaussienne. On obtient alors $\sum \Lambda = 0.92$ et $k = 0.51$.

Filtre de Deriche (1987)

Le filtre de Deriche est un filtre à réponse impulsionnelle infinie. Les contraintes sont alors les suivantes :

$$f(0) = 0; f(+\infty) = 0; f'(0) = S; f'(+\infty) = 0$$

Ces contraintes permettent à nouveau de déterminer les coefficients a_i :

$$a_1 = a_2 = a_4 = 0 \text{ et } \omega a_3 = S$$

d'où la solution :

$$h(x) = \frac{S}{\omega} e^{-\alpha|x|} \sin(\omega x)$$

On obtient alors :

$$\begin{aligned}\Lambda &= \sqrt{2\alpha} & \Sigma &= \sqrt{\frac{2\alpha}{\alpha^2 + \omega^2}} \\ \Sigma \Lambda &= \frac{2\alpha}{\alpha^2 + \omega^2} & k &= \sqrt{\frac{\alpha^2 + \omega^2}{5\alpha^2 + \omega^2}}\end{aligned}\tag{1}$$

Deriche a déterminé qu'en prenant α très grand devant ω , on obtient des résultats optimaux : $\Lambda\Sigma = 2$ et $k = 0.44$. Il a également montré que pour une valeur de k identique au filtre de Canny, le filtre de Deriche affiche une valeur de performance $\Lambda\Sigma$ bien meilleure (1.4 au lieu de 1.12 soit 90%). Pour des valeurs optimales, on va prendre dans la suite α très grand devant ω ce qui revient à prendre ω petit d'où, au premier ordre :

$$\sin(\omega x) = \omega x$$

Implémentation récursive du filtre de Deriche

Le filtre dériveur optimal se présente sous la forme $Se^{-\alpha|x|}|x|$ donnant lieu aux filtres bidimensionnels :

$$\begin{aligned}SS_x(m, n) &= kme^{-\alpha|m|}k(\alpha|n| + 1)e^{-\alpha|n|} \\SS_y(m, n) &= k(\alpha|m| + 1)e^{-\alpha|m|}kne^{-\alpha|n|}\end{aligned}$$

L'intégrateur de ce filtre est un filtre de lissage : $S(\alpha|x| + 1)e^{-\alpha|x|}$ d'équation en 2D :

$$SS(m, n) = k(\alpha|m| + 1)e^{-\alpha|m|}k(\alpha|n| + 1)e^{-\alpha|n|}$$

On obtient également le Laplacien sous la forme :

$$LL(m, n) = e^{-\alpha|m|}e^{-\alpha|n|} - k\alpha|m|e^{-\alpha|m|}.k\alpha|n|e^{-\alpha|n|}$$

avec $k = \frac{1-e^{-2\alpha}}{2\alpha e^{-\alpha}}$. Il convient dans ce cas de calculer deux résultats intermédiaires r_1 et r_2 puis de faire la différence entre ces deux images afin d'obtenir l'image du Laplacien.

Implémentation du filtre de Deriche :

• Phase 1

pour m variant de 0 à w :

pour n variant de 0 à h :

$$y_1(m, n) = a_1 l_1(m, n) + a_2 l_1(m, n - 1) + b_1 y_1(m, n - 1) + b_2 y_1(m, n - 2)$$

n variant de (h-1) à 0 :

$$y_2(m, n) = a_3 l_1(m, n + 1) + a_4 l_1(m, n + 2) + b_1 y_2(m, n + 1) + b_2 y_2(m, n + 2)$$

pour n variant de 0 à (h-1) :

$$r(m, n) = c_1(y_1(m, n) + y_2(m, n))$$

• Phase 2

pour n variant de 0 à h :

pour m variant de 0 à w :

$$y_1(m, n) = a_5 r(m, n) + a_6 r(m - 1, n) + b_1 y_1(m - 1, n) + b_2 y_1(m - 2, n)$$

pour m variant de (w-1) à 0 :

$$y_2(m, n) = a_7 r(m + 1, n) + a_8 r(m + 2, n) + b_1 y_2(m + 1, n) + b_2 y_2(m + 2, n)$$

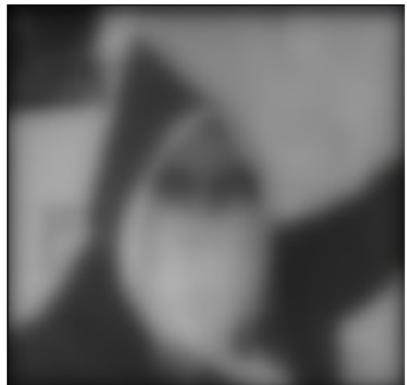
pour m variant de 0 à w :

$$l_2(m, n) = c_2(y_1(m, n) + y_2(m, n))$$

Paramètres des filtres de Deriche

	Lissage	Dérivée selon x	Dérivée selon y	Laplace 1	Laplace 2
k	$\frac{(1-e^{-\alpha})^2}{1+2\alpha e^{-\alpha}-e^{-2\alpha}}$	$\frac{(1-e^{-\alpha})^2}{1+2\alpha e^{-\alpha}-e^{-2\alpha}}$	$\frac{(1-e^{-\alpha})^2}{1+2\alpha e^{-\alpha}-e^{-2\alpha}}$	$\frac{1-e^{-2\alpha}}{2\alpha e^{-\alpha}}$	$\frac{1-e^{-2\alpha}}{2\alpha e^{-\alpha}}$
a_1	k	0	k	1	0
a_2	$ke^{-\alpha}(\alpha - 1)$	1	$ke^{-\alpha}(\alpha - 1)$	0	1
a_3	$ke^{-\alpha}(\alpha + 1)$	-1	$ke^{-\alpha}(\alpha + 1)$	$e^{-\alpha}$	1
a_4	$-ke^{-2\alpha}$	0	$-ke^{-2\alpha}$	0	0
a_5	k	k	0	1	0
a_6	$ke^{-\alpha}(\alpha - 1)$	$ke^{-\alpha}(\alpha - 1)$	1	0	1
a_7	$ke^{-\alpha}(\alpha + 1)$	$ke^{-\alpha}(\alpha + 1)$	-1	$e^{-\alpha}$	1
a_8	$-ke^{-2\alpha}$	$-ke^{-2\alpha}$	0	0	0
b_1	$2e^{-\alpha}$	$2e^{-\alpha}$	$2e^{-\alpha}$	$e^{-\alpha}$	$2e^{-\alpha}$
b_2	$-e^{-2\alpha}$	$-e^{-2\alpha}$	$-e^{-2\alpha}$	0	$-e^{-2\alpha}$
c_1	1	$-(1 - e^{-\alpha})^2$	1	1	$\frac{1-e^{-2\alpha}}{2}$
c_2	1	1	$-(1 - e^{-\alpha})^2$	1	$\frac{1-e^{-2\alpha}}{2}$

Deriche smoothing



0.25



0.5



0.75



1



2



3

Deriche edges (first derivatives)



0.5

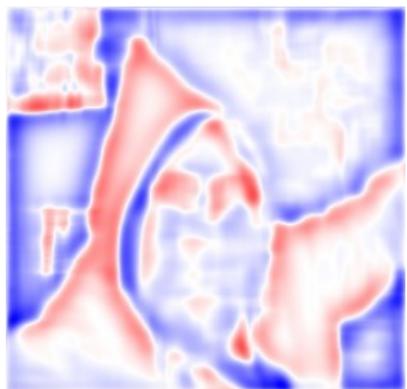


1



3

Deriche edges (second derivatives)



Outline

1 Un premier détecteur de contour

2 Convolution

3 Dérivée première

4 Dérivée seconde

5 Approche Canny - Deriche

6 Détection de points d'intérêt

Deux approches :

- en deux étapes :
 - chaînage des points de contours
 - recherche des points de courbure maximale
- recherche directe en fonction des intensités de l'image
 - Harris et Stephens

Détecteur de Harris et Stephens (1988)

$$\mathcal{O} = \det(\widehat{C}(x, y)) - k (\text{trace}(\widehat{C}(x, y)))^2$$

$$\widehat{C}(x, y) = \begin{pmatrix} \widehat{\left(\frac{\partial I(x, y)}{\partial x} \right)^2} & \widehat{\frac{\partial I(x, y)}{\partial x}} \widehat{\frac{\partial I(x, y)}{\partial x}} \\ \widehat{\frac{\partial I(x, y)}{\partial x}} \widehat{\frac{\partial I(x, y)}{\partial x}} & \widehat{\left(\frac{\partial I(x, y)}{\partial y} \right)^2} \end{pmatrix} \text{ avec } k = 0.04$$

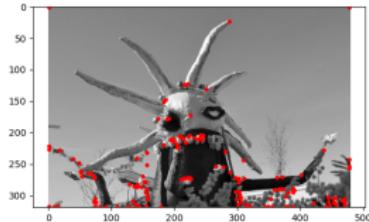
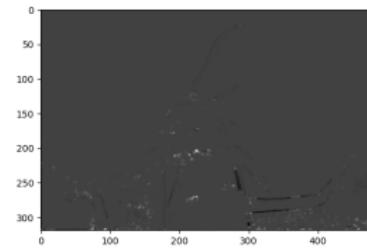
$\widehat{}$: lissage gaussien

- calcul des dérivées premières avec lissage gaussien
- calcul de \mathcal{O}
- seuillage



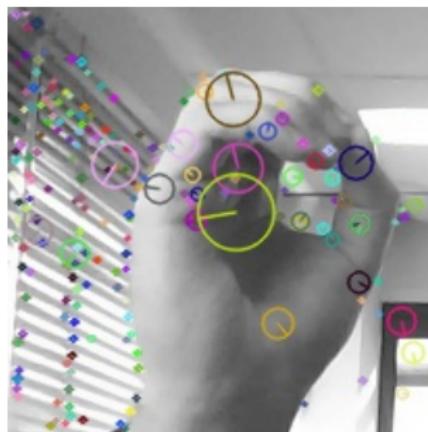
Harris example

```
from skimage.feature import corner_harris, corner_peaks
img = skimage.io.imread('carnaval4.jpg', as_gray=True)
pts=corner_peaks(corner_harris(img), min_distance=1)
print( pts.shape[0], ' points found')
import matplotlib.pyplot as plt
plt.imshow(img, cmap='gray')
plt.scatter(y=pts[:,0],x=pts[:,1],c='r',s=10)
plt.show()
```

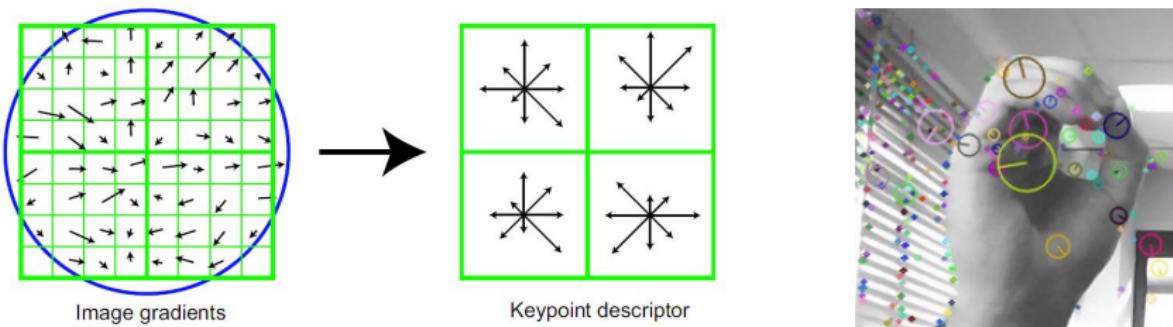


SIFT = Scale Invariant Feature Transform

- DéTECTeur
 - multi-échelles
 - utilise le laplacien DOG (différents niveaux de lissage gaussiens)
- Descripteur
 - description des orientations des contours dans le voisinage



Descripteur SIFT



- vecteur de 128 entiers
- 4 étapes :
 - détection des points d'intérêts
 - orientation des gradients dans le voisinage (16x16 pixels en 4x4 blocs)
 - histogramme des orientations (quantifié sur 8 valeurs) sur par blocs de 4x4 pixels
 - $8 \times 4 \times 4 = 128$
 - normalisation

SIFT implementation in OpenCV

```
import cv2
import numpy as np
img = cv2.imread('carnaval.jpg')
gray= cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
sift = cv2.xfeatures2dSIFT_create()
kp = sift.detect(gray,None)
cv2.drawKeypoints(gray,kp,img,\\
    flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imwrite('carnavalSIFT.jpg',img)
```

