Hiver 2018

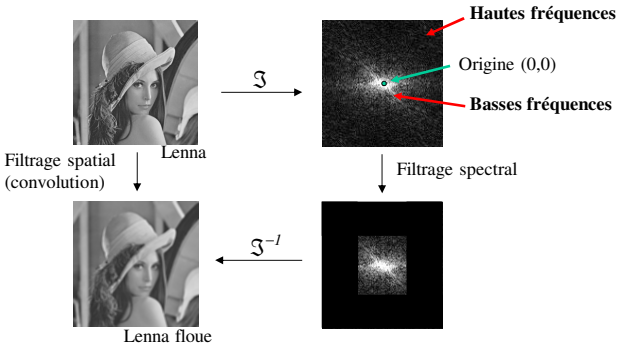
Analyse d'images
IMN 259

Filtrage spatial et spectral

Par
Pierre-Marc Jodoin

Filtrer une image

Le filtrage d'une image est une opération ayant pour objet de réduire ou d'éliminer ou de rehausser certains éléments présents dans une image. De nombreux filtres peuvent s'opérer tant dans le domaine **spatial** que le domaine **spectral**. C'est le cas des filtres **linéaires**. Ces derniers sont directement liés à la théorie de la convolution.



Rappel convolution

2

3 types de filtres linéaires fréquemment utilisés

- **Filtre passe-bas**
  Filtre ayant pour objet de couper les **hautes** fréquences. Cette opération a pour effet de réduire le bruit et d'ajouter du flou (c-à-d. éliminer les détails de l'images)

- **Filtre passe-haut**
  Filtre ayant pour objet de couper les **basses** fréquences. Cette opération a pour effet d'accentuer les détails de l'image, les contours et le bruit. Toutes les régions uniformes sont éliminées par cette procédure.

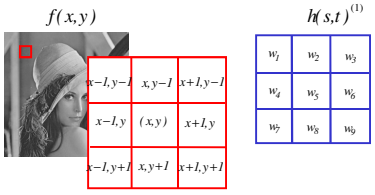
- **Filtre passe-bande**
  Filtre ayant pour objet de laisser une **bande** de fréquences. Sert à faire ressortir un aspect particulier de l'image (généralement des éléments de texture ou de bruit).

3

Filtrage spatial Vs filtrage spectral

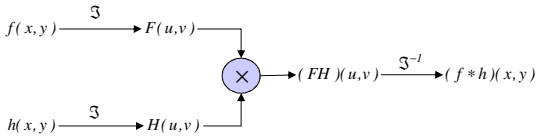
Dans le domaine spatial : **convolution**

$$(f * h)(x,y) = \sum_s \sum_t f(x-s, y-t)\, h(s,t)$$



$$(f*h)(x,y) = w_9 f(x-1,y-1) + w_8 f(x,y-1) + w_7 f(x+1,y-1)$$
$$+ w_6 f(x-1,y) \quad + w_5 f(x,y) \quad + w_4 f(x+1,y)$$
$$+ w_3 f(x-1,y+1) + w_2 f(x,y+1) + w_1 f(x+1,y+1)$$

Dans le domaine spectral : **multiplication**

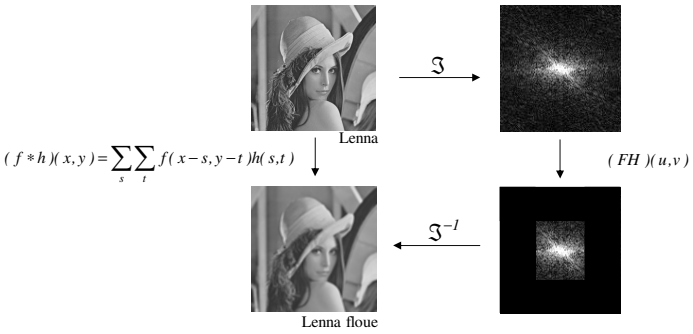


4

(1) Ainsi présenté, le filtre est parfois appelé ***masque***.

Filtrage spatial Vs filtrage spectral

Est-il plus rapide d'effectuer une convolution dans le domaine spatial ou dans le domain spectral?



Est-ce que, en temps de calculs $(f*h)(x,y) \overset{?}{=} \Im(f)+\Im(h)+FH+\Im^{-1}(FH)$

5

Filtrage spatial Vs filtrage spectral

Est-il plus rapide d'effectuer une concolution dans le domaine spatial ou dans le domain spectral?

Grâce à la *Fast Fourier Transform* (**FFT**), ont peut calculer $\Im$ et $\Im^{-1}$ de façon très efficace. Ainsi, selon Gonzalez et woods, pour une image *f(x,y)* de taille **256x256**, si le filtre *h(x,y)* a une taille supérieure ou égale à **13x13**, il est plus rapide d'effectuer la convolution dans le **domaine spectral**.

Note: puisque la plupart des algorithmes FFT reposent sur une approche *divisée pour régner* (*divide and conquer*), les images à filtrer doivent avoir des dimensions du type:

$$M = 2^m$$

$$N = 2^n$$

6

Notes importantes

1. Dans le domaine spatial : **convolution**

Très souvent:

$$\sum_{i,j} h(i,j) = 1 \quad (\text{ou } 0)$$

filtres passe-bas filtres passe-haut

2. Dans le domaine spectral : **multiplication**

$$F(u,v) = A + jB$$
$$H(u,v) = C + jD$$

$$(FH)(u,v) = (A + jB)(C + jD)$$
$$= AC + jAD + jBC + jjBD$$
$$= (AC - BD) + j(AD + BC)$$

7

Filtres passe-bas

8

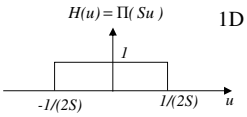
Filtres passe-bas

Filtres utilisés lors de filtrages **fréquentiels**

Filtre passe-bas rectangulaire: élimine toutes les fréquences situées au-delà d'un certain seuil. Le filtre passe-bas rectangulaire est la fonction « **porte** ».

Forme spectrale du filtre

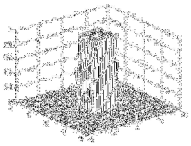
$H(u) = \Pi(Su)$ 1D



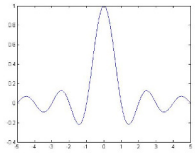
$H(u,v) = \Pi(S\sqrt{u^2 + v^2})$ 2D

Gonzalez-Woods

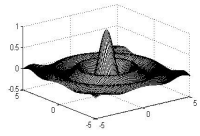
$H(u,v) = \Pi(S_x u, S_y v)$



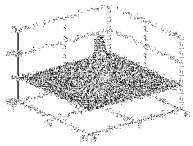
Forme spatiale du filtre



$h(x) = \frac{sinc(x/S)}{S}$



$h(x,y) = \frac{sinc(\sqrt{(x/S)^2 + (y/S)^2})}{S^2}$



$h(x,y) = \frac{sinc(x/S_x) sinc(y/S_y)}{S_x S_y}$

Filtres passe-bas

Filtres utilisés lors de filtrages fréquentiels

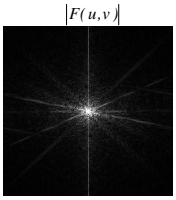
Filtre passe-bas rectangulaire: élimine toutes les fréquences situées au-delà d'un certain seuil. Le filtre passe-bas rectangulaire est la fonction « **porte** ».

Exemple :

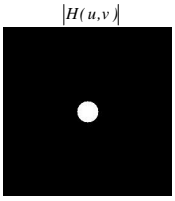
$f(x,y)$

« cameraman »

$|F(u,v)|$

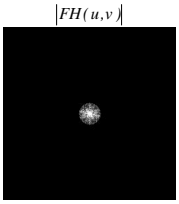


$|H(u,v)|$



rayon = *16* (/256)

$|FH(u,v)|$



rayon = *64* (/256)

rayon = *32* (/256)

rayon = *16* (/256)

rayon = *8* (/256)

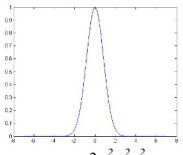
Filtres passe-bas

Filtre utilisé lors de filtrages fréquentiels ou spatiaux

Filtre gaussien

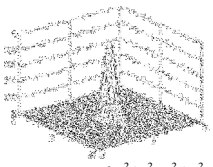
Forme spectrale du filtre

1D



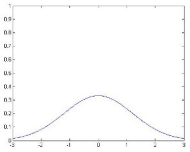
$$H(u) = \mathrm{e}^{-2\pi^2\sigma^2u^2}$$

2D

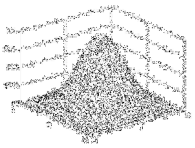


$$H(u,v) = e^{-2\pi^2(u^2+v^2)\sigma^2}$$

Forme spatiale du filtre



$$h(x) = \frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{-x^2/2\sigma^2}$$



$$h(x,y) = \frac{1}{2\pi\sigma^2}\mathrm{e}^{-(x^2+y^2)/2\sigma^2}$$

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

$\sigma = 1$

11

Note: normalement, un filtre spatial gaussien doit avoir une taille de $(6\sigma+1)\times(6\sigma+1)$

Filtres passe-bas

Filtre utilisé lors de filtrages fréquentiels ou spatiaux

Filtre gaussien

Exemple :

« cameraman »

$\sigma = 1$

$\sigma = 3$

$\sigma = 5$

$\sigma = 7$

Note: parfois le filtre $\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}/16$ est utilisé

12

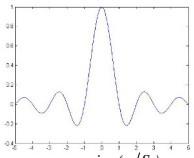
Filtres passe-bas

Filtre utilisé lors de filtrages spatiaux

Filtre moyenneur

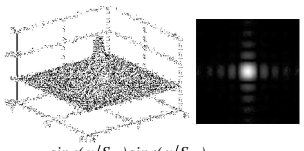
Forme spectrale du filtre

1D



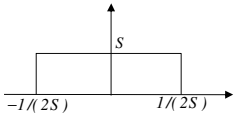
$$H(u) = \frac{\sin c(u/S)}{S}$$

2D

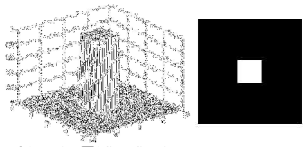


$$H(u,v) = \frac{\sin c(u/S_x)\sin c(v/S_y)}{S_x S_y}$$

Forme spatiale du filtre



$$h(x) = \Pi(Sx)$$



$$h(x,y) = \Pi(S_x x, S_y y)$$

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

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}/25$$

13

Filtres passe-bas

Filtre utilisé lors de filtrages spatiaux

Filtre moyenneur

Exemple :

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}/9, \quad 3\times3$$

5×5

11×11

15×15

Gauss, $\sigma = 7$

14

Filtres passe-bas

Filtres utilisés lors de filtrages spatiaux

Autres filtres passe-bas

Filtre conique

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 2 & 2 & 0 \\ 1 & 2 & 5 & 2 & 1 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} / 25$$

Filtre pyramidal

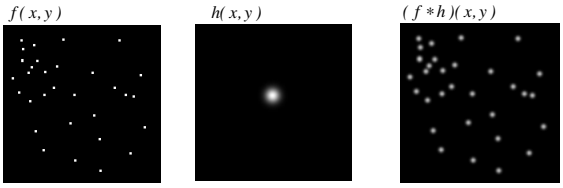
$$\begin{pmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 6 & 4 & 1 \\ 3 & 6 & 9 & 6 & 3 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{pmatrix} / 81$$

Filtre binomial

$$\begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix} / 256$$

15

Une dernière définition…



h(x, y) est parfois appelée la PSF (*Point Spread Function*).

PSF: Réponse d'un système [optique] à une impulsion lumineuse.
On appelle aussi la PSF la « réponse impulsionnelle ».

h(x,y) est souvent une **gaussienne**.

16

Filtres passe-haut

17

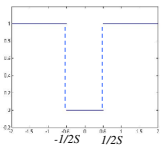
Filtres passe-haut

Filtres utilisés lors de filtrages fréquentiels

Filtre passe-haut rectangulaire: élimine toutes les fréquences situées en deçà d'un certain seuil. Ce filtre est la fonction « **porte** » inversée.

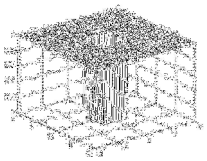
Forme spectrale du filtre

1D

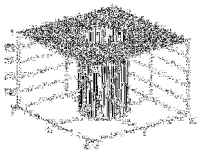


$H(u) = 1 - \Pi(Su)$

2D

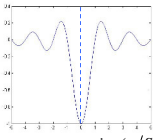


$H(u,v) = 1 - \Pi(S_x u, S_y v)$

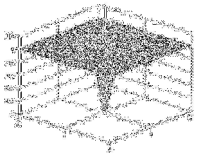


$H(u,v) = 1 - \Pi(S\sqrt{u^2 + v^2})$

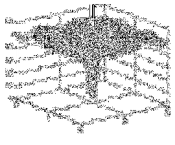
Forme spatiale du filtre



$h(x) = \delta(x) - \frac{\sin c(x/S)}{S}$



$h(x,y) = \delta(x,y) - \frac{\sin c(x/S_x)}{S_x}\frac{\sin c(y/S_y)}{S_y}$



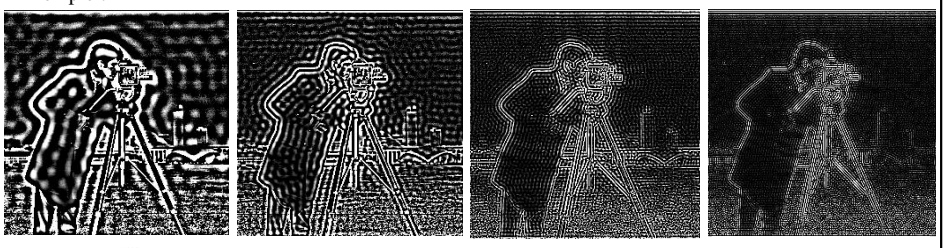
$h(x,y) = \delta(x,y) - \frac{\sin c(\sqrt{x^2 + y^2}/S)}{S^2}$

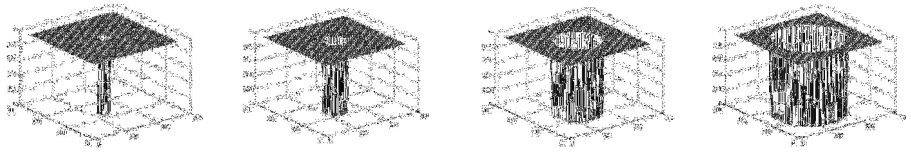
Filtres passe-haut

Filtres utilisés lors de filtrages fréquentiels

Filtre passe-haut rectangulaire: élimine toutes les fréquences situées en deçà d'un certain seuil. Ce filtre est la fonction « **porte** » inversée.

Exemple : $H(u,v) = 1 - \Pi(S\sqrt{u^2 + v^2})$





rayon = 16 (/ 256) *rayon = 32 (/ 256)* *rayon = 64 (/ 256)* *rayon = 96 (/ 256)*

Note: ces images ont été rehaussées pour fins de visualisation.

Filtres passe-haut

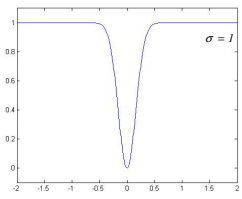
Filtres utilisés lors de filtrages spatiaux et fréquentiels

Filtre passe-haut gaussien: On utilise une gaussienne inversée.

Forme spectrale du filtre

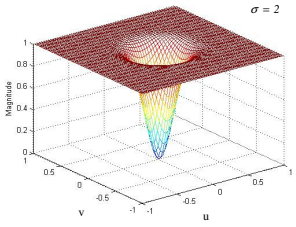
1D

$$H(u) = 1 - e^{-2\pi^2 u^2 \sigma^2}$$

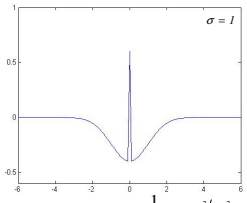


2D

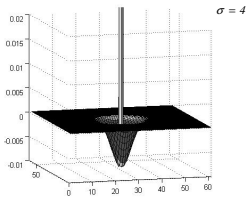
$$H(u,v) = 1 - e^{-2\pi^2 (u^2 + v^2) \sigma^2}$$



Forme spatiale du filtre



$$h(x) = \delta(x) - \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}$$



$$h(x,y) = \delta(x,y) - \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

20

Filtres passe-haut

Filtres utilisés lors de filtrages spatiaux et fréquentiels

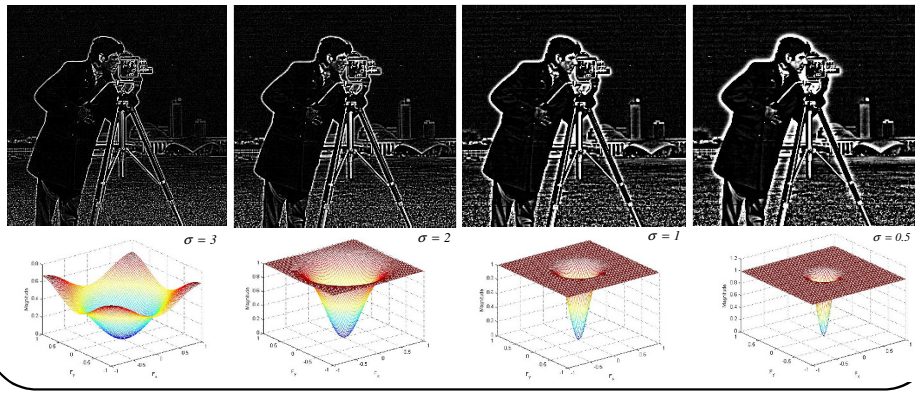
Filtre passe-haut gaussien:

$$h(x,y)=\begin{pmatrix} -0.0297 & -0.1331 & -0.2194 & -0.1331 & -0.0297 \\ -0.1331 & -0.5963 & -0.9832 & -0.5963 & -0.1331 \\ -0.2194 & -0.9832 & 8.3790 & -0.9832 & -0.2194 \\ -0.1331 & -0.5963 & -0.9832 & -0.5963 & -0.1331 \\ -0.0297 & -0.1331 & -0.2194 & -0.1331 & -0.0297 \end{pmatrix} \approx \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

pour $\sigma = 1$

« cameraman »

Exemple :



Filtres passe-haut

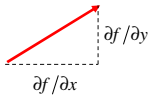
Filtres utilisés lors de filtrages spatiaux (parfois fréquentiels)

Gradient:

$$\nabla f = \begin{pmatrix} \partial f/\partial x \\ \partial f/\partial y \end{pmatrix}$$

Magnitude : $\sqrt{(\partial f/\partial x)^2 + (\partial f/\partial y)^2}$

Direction : $arctan((\partial f/\partial y)/(\partial f/\partial x))$



Approximation numérique du **gradient en X**

$$\partial f/\partial x = \lim_{\Delta h \to 0} \frac{f(x+\Delta h, y) - f(x,y)}{\Delta h} = \lim_{\Delta h \to 0} \frac{f(x,y) - f(x-\Delta h, y)}{\Delta h} = \lim_{\Delta h \to 0} \frac{f(x+\Delta h, y) - f(x-\Delta h, y)}{2\Delta h}$$

Puisque le plus petit élément dans une image est le pixel de taille $1x1$, le gradient se calcul avec $\Delta h = 1$

$$\partial f/\partial x \approx f(x+1,y) - f(x,y) \quad \text{"forward difference"}$$
$$\partial f/\partial x \approx f(x,y) - f(x-1,y) \quad \text{"backward difference"}$$
$$\partial f/\partial x \approx \frac{f(x+1,y) - f(x-1,y)}{2} \quad \text{"central difference"}$$

Approximation numérique du **gradient en Y**

$$\partial f/\partial y \approx f(x,y+1) - f(x,y) \quad \text{"forward difference"}$$
$$\partial f/\partial y \approx f(x,y) - f(x,y-1) \quad \text{"backward difference"}$$
$$\partial f/\partial y \approx \frac{f(x,y+1) - f(x,y-1)}{2} \quad \text{"central difference"}$$

22

Filtres passe-haut

Filtres utilisés lors de filtrages spatiaux/fréquentiels

Gradient:

Forme spatiale du filtre

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

$$\begin{pmatrix} 0 & -1 & 1 \end{pmatrix} \quad \begin{pmatrix} -1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} -1/2 & 0 & 1/2 \end{pmatrix}$$

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

$$\begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} -1/2 \\ 0 \\ 1/2 \end{pmatrix}$$

Filtres passe-haut

Filtres utilisés lors de filtrages spatiaux/fréquentiels

Gradient:

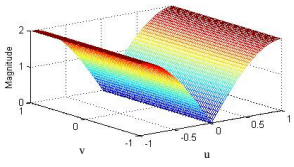
Exemples

Image originale

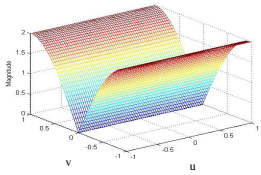
Gradient en X + 128

Gradient en Y + 128

Forme spectrale des filtres (-1 1)
(Spectres de magnitude)



$$\frac{\partial f(x,y)}{\partial x} \rightarrow j2\pi u F(u,v)$$



$$\frac{\partial f(x,y)}{\partial y} \rightarrow j2\pi v F(u,v)$$

Filtres passe-haut

Filtres utilisés lors de filtrages spatiaux/fréquentiels

Dérivée seconde et Laplacien

$$\frac{\partial^2 f(x,y)}{\partial^2 x} = \frac{\partial}{\partial x}\left(\frac{\partial f(x,y)}{\partial x}\right)$$
$$= \frac{\partial}{\partial x}\left(f(x+1/2,y) - f(x-1/2,y)\right)$$
$$= \frac{\partial f(x+1/2,y)}{\partial x} - \frac{\partial f(x-1/2,y)}{\partial x}$$
$$= f(x+1,y) - f(x,y) + f(x-1,y) - f(x,y)$$
$$= f(x-1,y) - 2f(x,y) + f(x+1,y)$$

Masque du filtre $(1 \quad -2 \quad 1)$

Le laplacien

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial^2 x} + \frac{\partial^2 f(x,y)}{\partial^2 y}$$

$$\longrightarrow (1 \quad -2 \quad 1) + \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

25

Filtres passe-haut

Filtres utilisés lors de filtrages spatiaux/fréquentiels

Dérivée seconde et Laplacien

Exemples

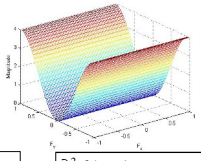
Image originale

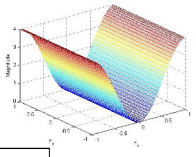
$\frac{\partial^2 f(x,y)}{\partial^2 x} + 128$

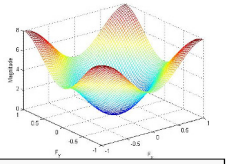
$\frac{\partial^2 f(x,y)}{\partial^2 y} + 128$

$\frac{\partial^2 f(x,y)}{\partial^2 x} + \frac{\partial^2 f(x,y)}{\partial^2 y} + 128$

Forme spectrale des filtres (1 -2 1)(Spectres de magnitude)







$$\frac{\partial^2 f(x,y)}{\partial^2 x} \rightarrow (j2\pi u)^2 F(u,v) = -4\pi^2 u^2 F(u,v)$$

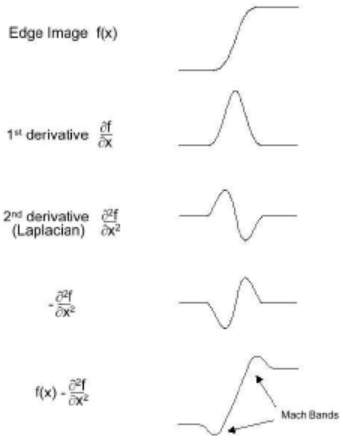
$$\frac{\partial^2 f(x,y)}{\partial^2 y} \rightarrow (j2\pi v)^2 F(u,v) = -4\pi^2 v^2 F(u,v)$$

$$\frac{\partial^2 f(x,y)}{\partial^2 x} + \frac{\partial^2 f(x,y)}{\partial^2 y} \rightarrow -4\pi^2 (u^2 + v^2) F(u,v)$$

Filtres passe-haut

Rehaussement de contours

Soit *f(x,y)* une image d'entrée. L'objectif est de calculer une image dont les contours ont été rehaussés. Pour ce faire, on utilise fréquemment le filtre laplacien:



Crédit Mignotte

27

Filtres passe-haut

Rehaussement de contours

$$
\begin{aligned}
f_r(x,y) &= f(x,y) - Laplacien(f(x,y)) \\
&= f(x,y) - \left(\frac{\partial^2 f(x,y)}{\partial^2 x} + \frac{\partial^2 f(x,y)}{\partial^2 y} \right) \\
&= f(x,y) * \delta(x,y) - f(x,y) * h_{laplacien}(x,y) \\
&= f(x,y) * \underbrace{(\delta(x,y) - h_{laplacien}(x,y))}
\end{aligned}
$$

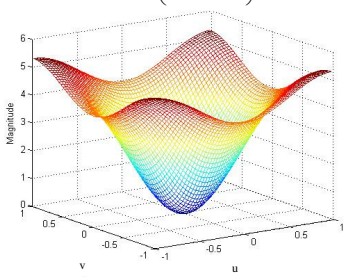
$$
\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}
$$

28

Filtres passe-haut

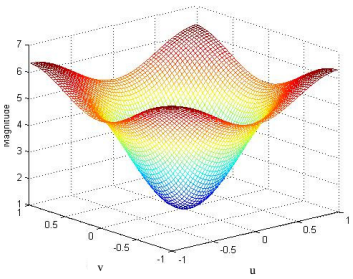
Rehaussement de contours, magnitude de la réponse fréquentielle

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



Valeur minimale = 0, le filtre coupe donc les basses fréquences

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$



Valeur minimale = 1, le filtre préserve les basses fréquences

29

Filtres passe-haut

Rehaussement de contours laplacien

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

30

Filtres passe-haut

Rehaussement de contours "highboost"

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4+A & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4.5 & -1 \\ 0 & -1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5.5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$




31

Filtres passe-haut

Filtres utilisés lors de filtrages spatiaux

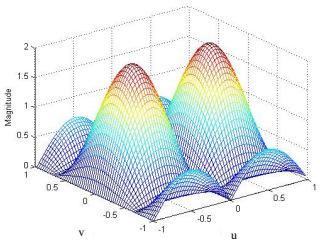
Filtre de Prewitt

Filtre moyenneur suivi d'un gradient

En Y: $(1 \quad 1 \quad 1)/3 \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \rightarrow \boxed{\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}/3}$

En X: $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}/3(-1 \quad 0 \quad 1) \rightarrow \boxed{\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}/3}$

Forme spectrale des filtres (en X)

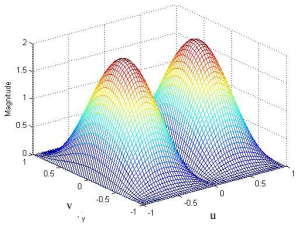


Filtre de Sobel

Filtre « gaussien » suivi d'un gradient

En Y: $(1 \quad 2 \quad 1)/4 \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \rightarrow \boxed{\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}/4}$

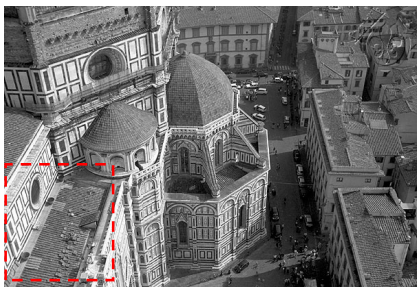
En X: $\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}/4(-1 \quad 0 \quad 1) \rightarrow \boxed{\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}/4}$



Filtres passe-haut

Filtres utilisés lors de filtrages spatiaux

Exemple: réduction du nombre de faux positifs



$\frac{\partial f}{\partial x} \rightarrow (-1 \quad 0 \quad 1)$

Filtre de Prewitt (en X)

Filtre de Sobel (en X)

33

Filtres passe-haut

Note: **ces filtres de Prewitt et Sobel** sont parfois considérés comme des filtres passe-bande.

34

Filtres passe-haut

Filtres utilisés lors de filtrages spatiaux

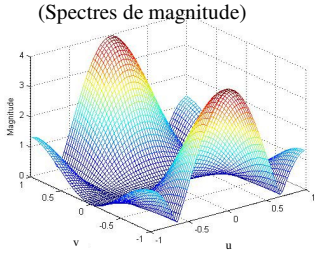
Les filtres de Prewitt et Sobel peuvent aussi être adaptés à la **dérivée seconde**

Filtre moyenneur suivi d'une dérivée seconde

En Y: $(1\ \ 1\ \ 1)/3\begin{pmatrix}1\\-2\\1\end{pmatrix}\rightarrow\boxed{\begin{pmatrix}1&1&1\\-2&-2&-2\\1&1&1\end{pmatrix}/3}$

En X: $\begin{pmatrix}1\\1\\1\end{pmatrix}/3(1\ \ -2\ \ 1)\rightarrow\boxed{\begin{pmatrix}1&-2&1\\1&-2&1\\1&-2&1\end{pmatrix}/3}$

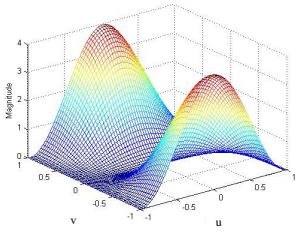
Forme spectrale des filtres (en X)
(Spectres de magnitude)



Filtre « gaussien » suivi d'une dérivée seconde

En Y: $(1\ \ 2\ \ 1)/4\begin{pmatrix}1\\-2\\1\end{pmatrix}\rightarrow\boxed{\begin{pmatrix}1&2&1\\-2&-4&-2\\1&2&1\end{pmatrix}/4}$

En X: $\begin{pmatrix}1\\2\\1\end{pmatrix}/4(1\ \ -2\ \ 1)\rightarrow\boxed{\begin{pmatrix}1&-2&1\\2&-4&2\\1&-2&1\end{pmatrix}/4}$



Filtres passe-bande

Filtres passe-bande

Filtres utilisés lors de filtrages spatiaux

Filtre de Marr-Hildreth (*Laplace of Gaussian - LOG*)

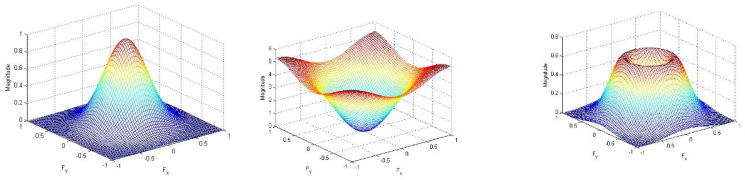
1- On filtre l'image avec un filtre gaussien « g »
2- On prend le laplacien de l'image filtrée

$$f'(x,y) = (f(x,y) * g(x,y)) * l(x,y)$$

$$= f(x,y) * (g(x,y) * l(x,y))$$

Forme spectrale des filtres (Magnitude)

$G(u,v)$ × $L(u,v)$ = $GL(u,v)$

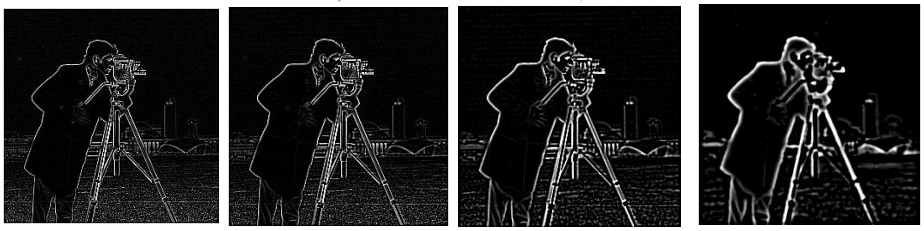


37

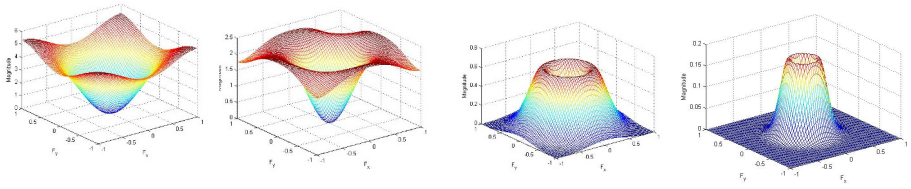
Filtres passe-bande

Filtres utilisés lors de filtrages spatiaux
Filtre de Marr-Hildreth, exemple

Laplace $M-H, \sigma = 0.5$ $M-H, \sigma = 1$ $M-H, \sigma = 2$



Forme spectrale des filtres (Spectres de magnitude)



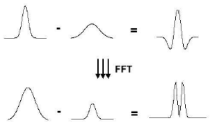
38

Filtres passe-bande

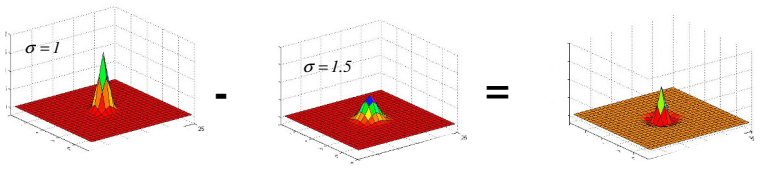
Filtres utilisés lors de filtrages spatiaux ou fréquentiels

Filtre de Marr-Hildreth ≈ différence entre deux gaussiennes (*difference of Gaussians - DOG*)

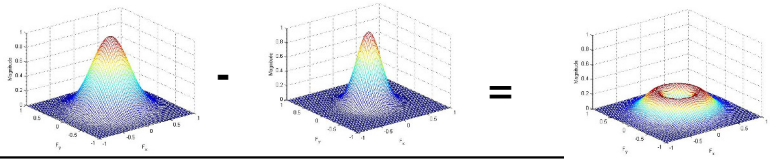
1D



2D

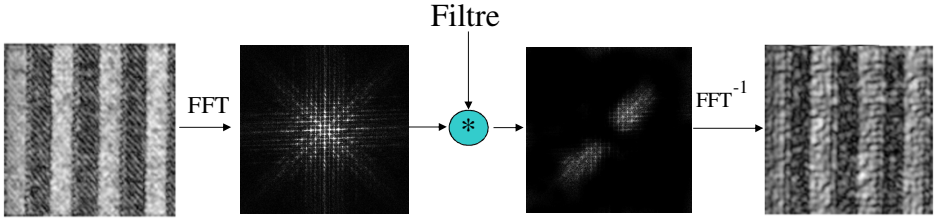


FFT



Segmentation sur la base des textures

Au lieu des moments, on peut également utiliser des **filtres de Gabor.**

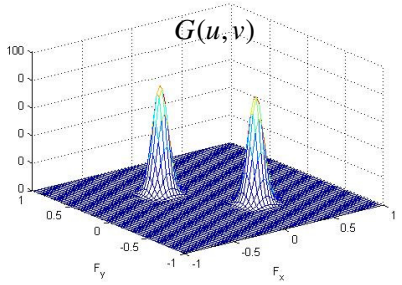


40

Filtres passe-bande

Filtres plus souvent utilisés lors de filtrages fréquentiels

Filtres de Gabor : un filtre passe bande centré sur une fréquence et avec une enveloppe gaussienne .



$$G(u,v) = Ae^{-\frac{1}{2}\left(\frac{(u-u_0)^2}{\sigma_u^2}+\frac{(v-v_0)^2}{\sigma_v^2}\right)} + Ae^{-\frac{1}{2}\left(\frac{(u+u_0)^2}{\sigma_u^2}+\frac{(v+v_0)^2}{\sigma_v^2}\right)} \quad \text{lorsque } \phi = 0$$

$$g(x,y) = e^{-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right)} \cos\left(2\pi u_0 x + 2\pi v_0 y + \phi\right)$$

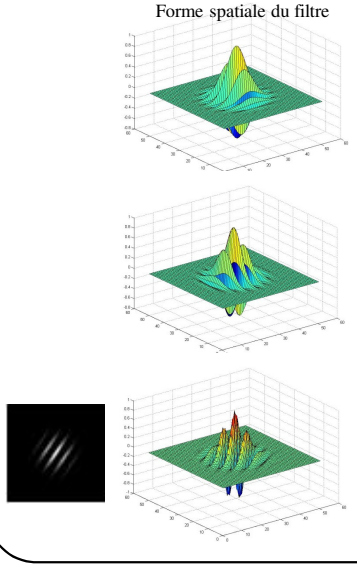
41

Filtres passe-bande

Filtres plus souvent utilisés lors de filtrages fréquentiels

Forme spatiale du filtre

Forme spectrale du filtre



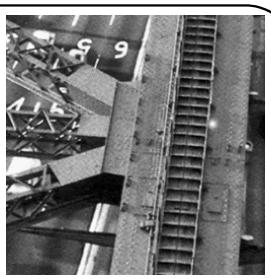
Forme spectrale =
2 gaussiennes translatées

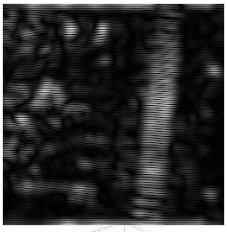
Forme spatiale =
1 gaussienne * exponentielle complexe

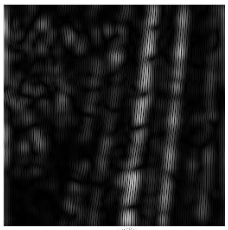
42

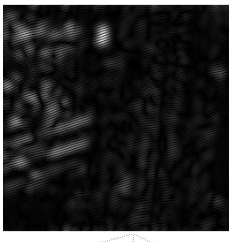
Filtres passe-bande

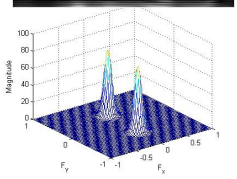
Filtres plus souvent utilisés lors de filtrages fréquentiels

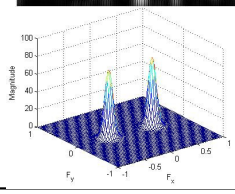


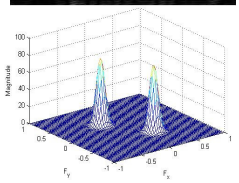






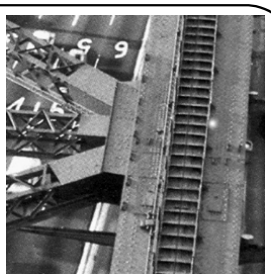


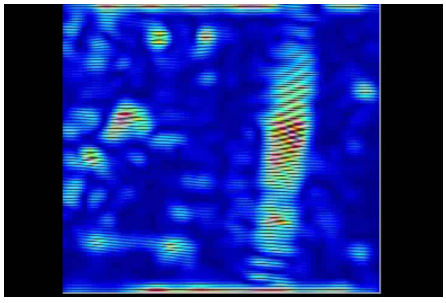


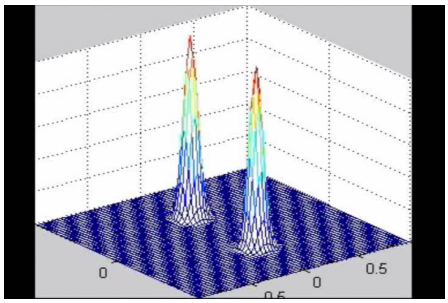


Filtres passe-bande

Filtres plus souvent utilisés lors de filtrages fréquentiels



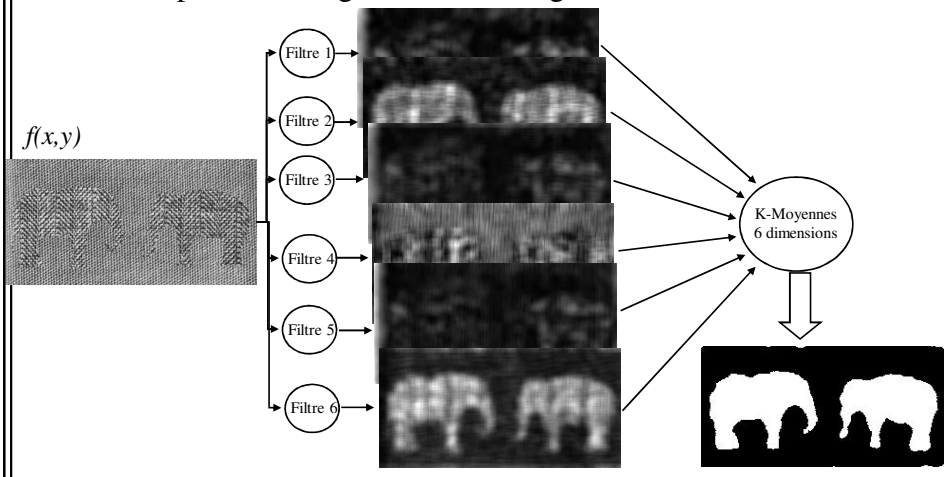




(Vidéo)

Application : segmentation K-moyennes

Gabor permet de segmenter des images sur la base de leur texture.

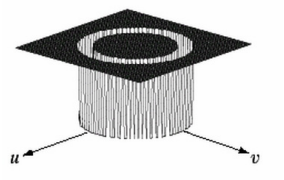


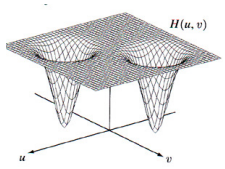
45

Filtre rejette-bande

En fréquences:

Filtre rejette-bande = 1 - filtre-passebande





Filtre de Notch

46

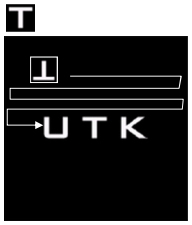
Corrélation

47

Corrélation

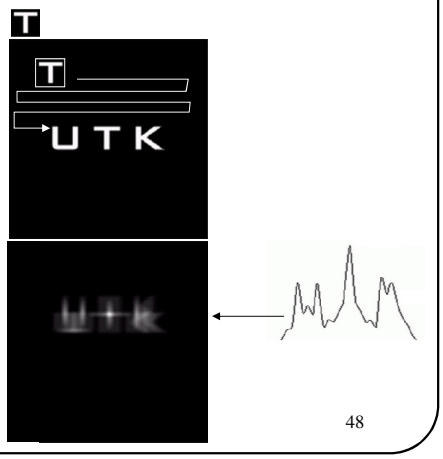
Convolution

$$(f * h)(x,y) = \sum_r \sum_t f(t,r)h(x-t, y-r)$$



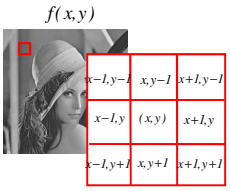
Corrélation

$$(f \circ h)(x,y) = \sum_r \sum_t f(t,r)h(x+t, y+r)$$



48

Corrélation



Convolution	Corrélation
$(f*h)(x,y)=\sum_r\sum_t f(t,r)h(x-t,y-r)$	$(f \circ h)(x,y)=f(x,y)*h(-x,-y)=\sum_r\sum_t f(t,r)h(x+t,y+r)$
$(f*h)(x,y)=w_9f(x-1,y-1)+w_8f(x,y-1)+w_7f(x+1,y-1)$ $+w_6f(x-1,y)+w_5f(x,y)+w_4f(x+1,y)$ $+w_3f(x-1,y+1)+w_2f(x,y+1)+w_1f(x+1,y+1)$	$(f*h)(x,y)=w_1f(x-1,y-1)+w_2f(x,y-1)+w_3f(x+1,y-1)$ $+w_4f(x-1,y)+w_5f(x,y)+w_6f(x+1,y)$ $+w_7f(x-1,y+1)+w_8f(x,y+1)+w_9f(x+1,y+1)$

Propriétés de la corrélation:

$$(f \circ h)(x,y) \quad \overset{\Im}{\rightarrow} \quad (\overline{F}H)(u,v)$$

Corrélation

Malheureusement, la corrélation est très sensible aux changements d'intensité. Par conséquent on peut implémenter les **coefficients de la corrélation**:

$$\gamma(x,y)=\frac{\sum_r\sum_t (f(t,r)-f_M(t,r))(h(x+t,y+r)-h_M)}{\sqrt{A}}$$

$f_M(t,r)$ = moyenne de l'image f sur la région couverte par h

h_M = moyenne du filtre h

$\sqrt{A}$ est un terme de normalisation pour ramener $\gamma(x,y)$ dans l'interval [-1,1]

$$A=\sum_r\sum_t (f(t,r)-f_M(t,r))^2 \times \sum_r\sum_t (h(x+t,y+r)-h_M)^2$$

Image originale : h est entouré de pointillés

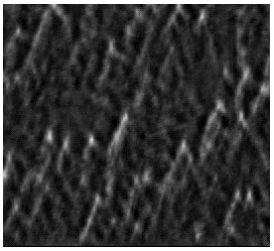


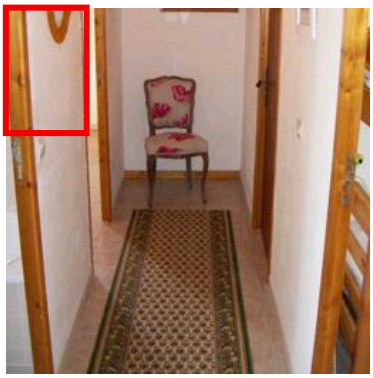
Image de corrélation

Application : repérer les cimes

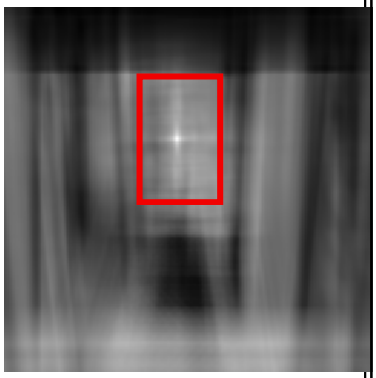
Corrélation

Ceci est une chaise

Trouver la chaise



Résultat d'une corrélation normalisé

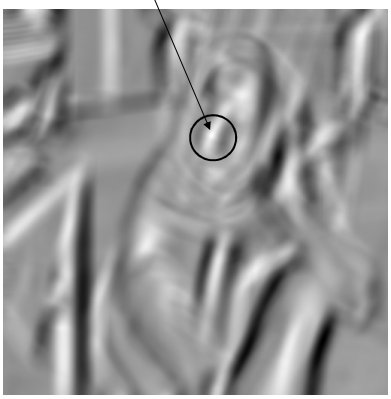


51

Corrélation

Réponse maximale

o  =



52

Corrélation

Il y a certains filtres pour lesquels la **convolution = corrélation**. C'est entre autre le cas de :

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}/9 \qquad \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}/25 \qquad \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 2 & 2 & 0 \\ 1 & 2 & 5 & 2 & 1 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}/25 \qquad \begin{pmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 6 & 4 & 1 \\ 3 & 6 & 9 & 6 & 3 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{pmatrix}/81 \qquad \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}/256 \qquad (\ldots)$$

53

Accélération de la convolution/corrélation spatiale

Plusieurs filtres spatiaux 2D "*h*" que nous avons exploré sont séparables en deux filtres spatiaux 1D h_x,h_y :

$$h=\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}/9 \rightarrow h_x=(1 \quad 1 \quad 1)/3, h_y=\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}/3$$

$$h=\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}/16 \rightarrow h_x=(1 \quad 2 \quad 1)/4, h_y=\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}/4$$

$$h=\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}/3 \rightarrow h_x=(-1 \quad 0 \quad 1), h_y=\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}/3$$

Pour les filtres séparables:

$$(f*h)(x,y)=((f*h_x)*h_y)(x,y)$$

Pour un filtre de taille *NxN*

$$(f*h)(x,y) \rightarrow N \times N \text{ mult et } N \times N-1 \text{ add}$$

Alors que

$$(f*h_x)(x,y) \rightarrow N \text{ mult et } N-1 \text{ add}$$
$$(f*h_y)(x,y) \rightarrow N \text{ mult et } N-1 \text{ add}$$
$$((f*h_x)*h_y)(x,y) \rightarrow 2 \times (N \text{ mult et } N-1 \text{ add})$$

Donc pour un filtre séparable de taille *21x21*

$$(f*h)(x,y) \rightarrow 840 \text{ opérations}$$
$$((f*h_x)*h_y)(x,y) \rightarrow 82 \text{ opérations}$$

54

Autres filtres

55

Filtres directionnels

56

Filtre directionnel (*motion blur*)

Simule le flou causé par un mouvement de caméra

?

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} / 5$$

Mouvement 45 degrés

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} / 5$$

Mouvement 90 degrés

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} / 5$$

Mouvement 0 degré

Filtre 21x21

Filtre 21x21

Filtre 21x21

57

Filtre médian

58

Filtre non-linéaire : filtre médian

Utile pour contrer le bruit poivre et sel (aussi appelé bruit impulsionnel)

$$g(x,y) = médiane_{\eta_{x,y}}\{f(x,y)\}$$

($\eta_{x,y}$ est un voisinage centré sur (x, y))

Exemples:

216	5	8
1	10	11
6	3	12

bruit

↓

1,3,5,6,8,10,11,12,216

↑

médiane

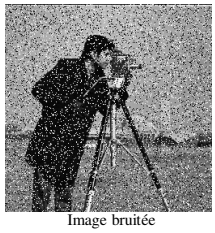


Image bruitée

Gauss, $\sigma = 1$

Gauss, $\sigma = 2$

Médiane (3x3)

59

Filtre bilatéral

60

Filtrage bilatéral

Comme nous l'avons vu, le filtre gaussien est excellent pour réduire le bruit dans une image. Toutefois, il a pour inconvénient de dénaturer les contours.

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

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

$\sigma = 1$

$*$

$f(x,y)$

$=$

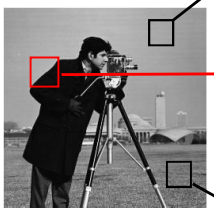
$f'(x,y)$

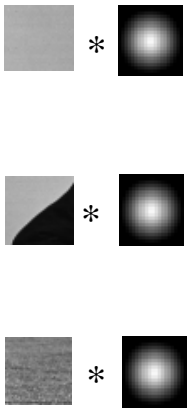
$\sigma = 5$

Ressource fort utile à la portée de tous :
http://people.csail.mit.edu/sparis/siggraph07_course/

Filtrage bilatéral

$f(x,y)$

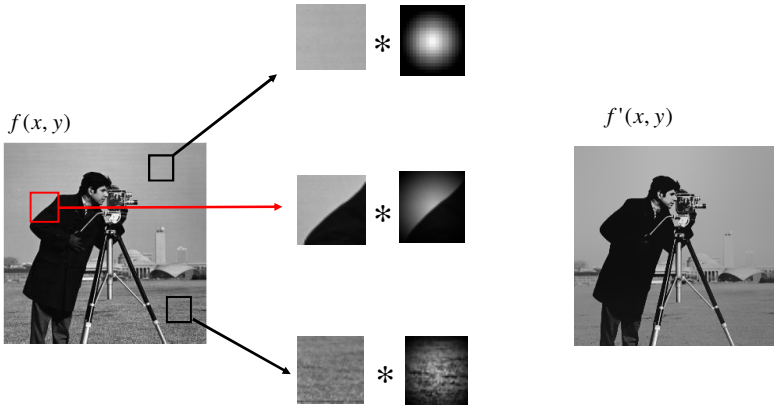




$f'(x,y)$

Étant un filtre **linéraire**, le filtre gaussien est partout le même.

Filtrage bilatéral



Avec l'approche bilatérale, la **forme du filtre s'ajuste au contenu de l'image**

63

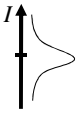
Filtrage bilatéral

Filtrage gaussien classique :

$$\underbrace{(f * h)}_{\text{Convolution}}(x, y) = \underbrace{f'(x, y)}_{\text{Niveau de gris résultant}} = \underbrace{\sum_s \sum_t}_{\substack{\text{Somme sur les voisins} \\ \text{du pixel (x,y)}}} \underbrace{f(x-s, y-t)}_{\substack{\text{Niveau de gris} \\ \text{du pixel (x,y)}}} \underbrace{h_\sigma(s,t)}_{\text{Filtre}}$$

Filtrage bilatéral :

$$f'(x, y) = \underbrace{\frac{1}{W}}_{\substack{\text{Terme de} \\ \text{normalisation}}} \sum_s \sum_t f(x-s, y-t) \underbrace{h_{\sigma_S}(s,t)}_{\text{Filtre spatial}} \underbrace{h_{\sigma_R}(f_{s,t})}_{\text{Filtre } \textit{range}}$$



64

Filtrage bilatéral

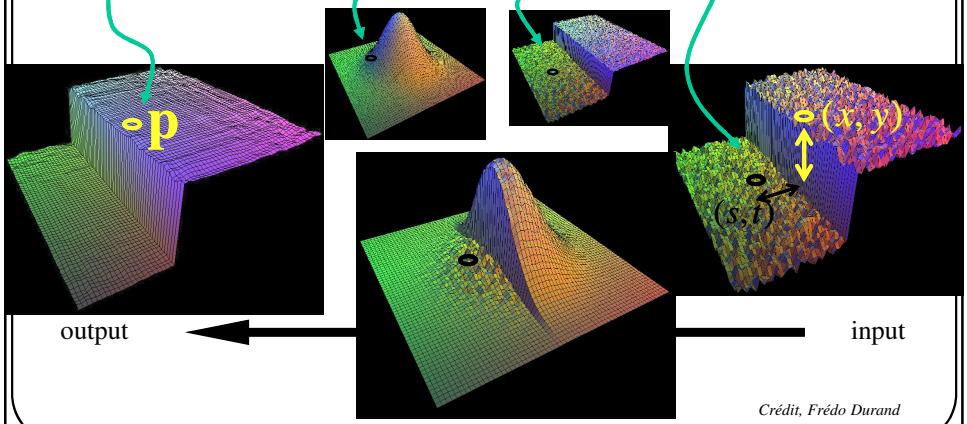
$$h_{\sigma_S}(x,y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma_S^2}$$

$$h_{\sigma_R}(f_{s,t}) = \frac{1}{\sqrt{2\pi}\sigma_R} e^{-(f_{s,t}-f_{x,y})^2/2\sigma_R^2}$$

65

Filtrage bilatéral

$$f'(x,y) = \frac{1}{W}\sum_s\sum_t h_{\sigma_S}(s,t)h_{\sigma_R}(f_{s,t})f(x-s,y-t)$$

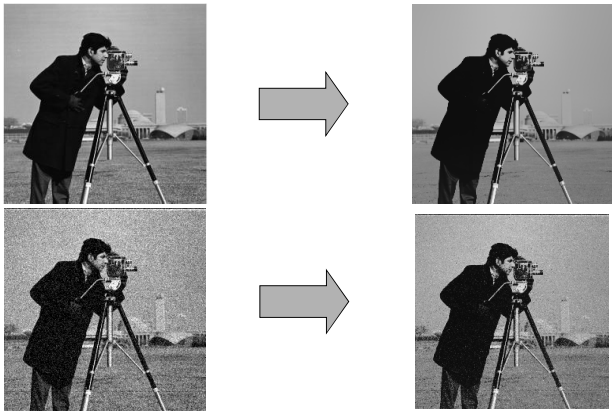


Crédit, Frédo Durand

Filtrage bilatéral

Note 1 : le filtrage bilatéral fonctionne mal en présence d'images fortement bruitées.

$f(x, y)$ peu bruité



$f(x, y)$ fortement bruité

Filtrage bilatéral

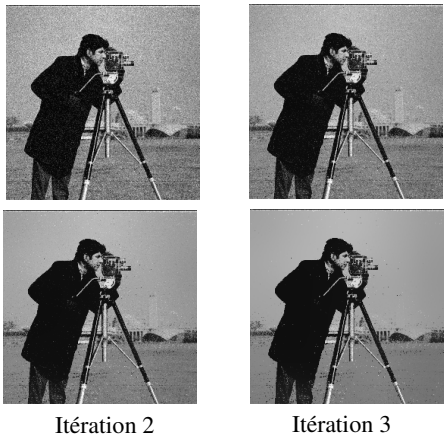
Note 2 : en présence d'images fortement bruitées, on peut rendre ce filtre **itératif**.

$f'(x, y) \leftarrow FILTRE\_BL(f(x, y))$

POUR iter = 1 jusqu'à N FAIRE

$f'(x, y) \leftarrow FILTRE\_BL(f'(x, y))$

Itération 0 Itération 1



Itération 2 Itération 3

Filtrage bilatéral

Note 3 : une implémentation directe du filtrage bilatéral est **très coûteuse** en temps de calcul. Pour en réduire les délais, il faut **approximer** les calculs. De plus, un filtre bilatéral n'étant PAS linéaire, il ne peut s'effectuer via une **transformée de Fourier**.

69

Non-local means

Buades, B. Coll, J.M. Morel "**A non local algorithm for image denoising**"
IEEE Computer Vision and Pattern Recognition 2005, Vol 2, pp: 60-65, 2005

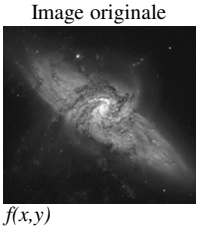
Pour une démo: www.ipol.im/pub/algo/bcm_non_local_means_denoising/

70

Non-Local Means

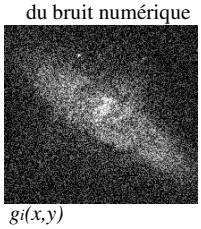
On se souvient qu'on peut augmenter le ratio signal sur bruit d'une image en moyennant plusieurs images $g_i(x,y)$ dont **le contenu est toujours le même** ($f(x,y)$) plus ou moins du **bruit blanc additif**.

Image originale



$f(x,y)$

Image corrompue par du bruit numérique



$g_i(x,y)$

$$g_i(x,y) = f(x,y) + \eta(x,y)$$

Bruit blanc additif

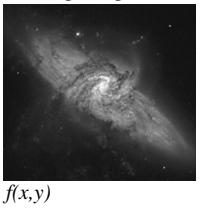
$$\bar{g}(x,y) = \frac{1}{N}\sum_{i=1}^{N} g_i(x,y)$$

71

Non-Local Means

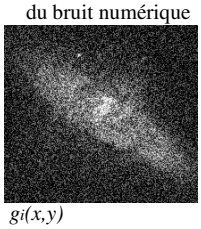
On se souvient qu'on peut augmenter le ratio signal sur bruit d'une image en moyennant plusieurs images $g_i(x,y)$ dont **le contenu est toujours le même** ($f(x,y)$) plus ou moins du **bruit blanc additif**.

Image originale



$f(x,y)$

Image corrompue par du bruit numérique



$g_i(x,y)$

$$g_i(x,y) = f(x,y) + \eta(x,y)$$

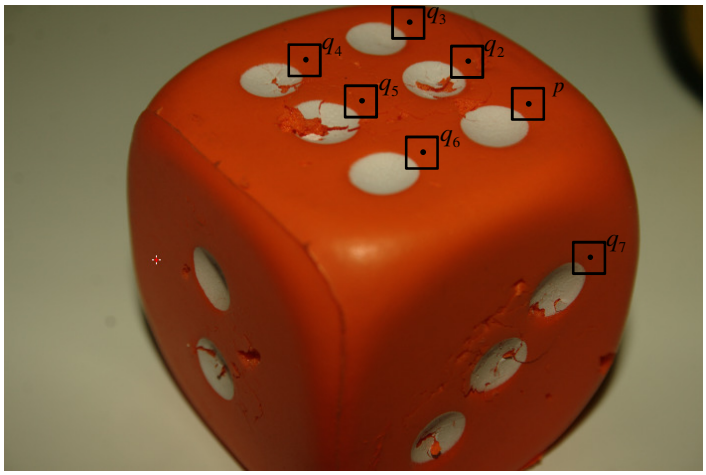
Bruit blanc additif

Ratio signal sur bruit : $\left(\frac{f}{\bar{B}}\right) = \sqrt{N}\left(\frac{f}{B}\right)$ Donc, pour $N=64$, le bruit dans $\bar{g}$ est réduit d'un facteur 8.

72

Non-Local Means

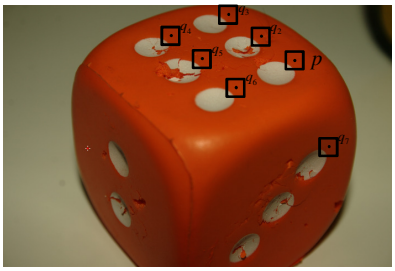
L'idée derrière les *non-local means* est la même, mais **avec une seule image**. Cette méthode est basée sur le fait que les images naturelles sont fortement **structurées et redondantes** et constituées de sections très **répétitives**.



g(x,y)

73

Non-Local Means



g(x,y)

Le pixel p et son voisinage η_p est très similaires aux pixels q_2 à q_7 et à leur voisinage η_{q_2} à η_{q_7}

74

Non-Local Means

Le but est donc de faire la moyenne de tous les pixels dont le même **contenu est très similaire** plus ou moins du bruit.

$$\bar{g}(p) = \frac{1}{C}\sum_{q} w(p,q)\,g(q)$$

où

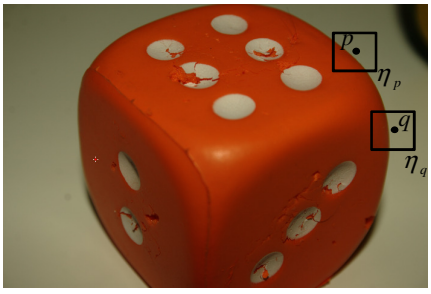
$$w(p,q) \geq 0$$

$$\sum_{x}\sum_{y} w(p,q) = 1$$

$w(p,q)$ détermine dans quelle mesure les pixels p et q sont similaires

75

Non-Local Means



$$w(p,q) = \frac{1}{Z}e^{-d(p,q)}$$

où

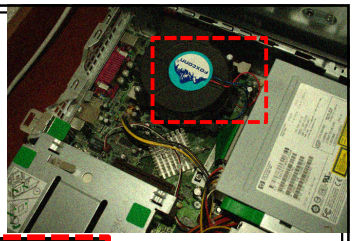
Z est une constante de normalisation

et

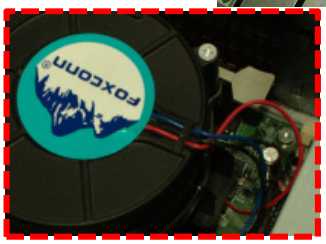
$$d(p,q) = \left\| g(\eta_p) - g(\eta_q) \right\|$$

76

Non-Local Means



g(x,y)

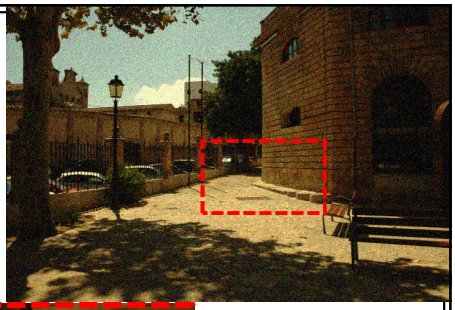


f(x,y)

$\overline{g}(x, y)$

77

Non-Local Means



g(x,y)

f(x,y)

$\overline{g}(x, y)$

78

Diffusion non linéaire

79

Diffusion : linéaire et non-linéaire

Quelques définitions:

L'opérateur gradient (nabla) est un vecteur de deux dérivées partielles:

$$\nabla \Rightarrow \begin{pmatrix} \partial_x \\ \partial_y \end{pmatrix}$$

Souvent on traite ∇ comme un simple vecteur. Pour une fonction *f(x,y),* on obtient que:

$$\nabla f = \begin{pmatrix} \partial f / \partial x \\ \partial f / \partial y \end{pmatrix}$$

Note: ∇f pointe en direction de la **pente maximale**.

La divergence est le produit scalaire entre nabla et un vecteur 2D $v = (a,b)$

$$div(v) = \nabla^{\mathrm{T}} v = \begin{pmatrix} \partial_x & \partial_y \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \frac{\partial a}{\partial x} + \frac{\partial b}{\partial y}$$

Le produit scalaire entre nabla et le gradient est ce qu'on appelle le **laplacien**:

$$\Delta(f) = div(\nabla f) = \begin{pmatrix} \partial_x & \partial_y \end{pmatrix} \begin{pmatrix} \partial f / \partial x \\ \partial f / \partial y \end{pmatrix} = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y}$$

80

Diffusion : linéaire et non-linéaire

Quelques définitions:

Une équation algébrique se représente ainsi

$$x^2 - 8x + 15 = 0$$

deux solutions : $x_1 = 3, x_2 = 5.$

Une équation différentielle exprime la relation qui existe entre une fonction et sa (ses) dérivé(s).

$$\frac{du(t)}{dt} = 5u(t)$$

Étant donnée la condition initiale $u(t=0) = 2$, il existe une solution unique à cette équation:

$$u(t) = 2e^{5t}$$

Si la fonction *u* dépend de plusieurs variables, alors des dérivées partielles sur différentes variables peuvent apparaître dans l'équation différentielle. Dans ce cas, on parlera d'une **équation aux dérivées partielles** (*partial differential equation PDE*).

Par exemple, considérez l'équation 1D suivante:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial^2 x} \quad \text{avec } u(x, t=0) = f(x)$$

En général, il n'existe pas de solutions analytiques "simples" pour une PDE. Un approximation numérique est donc généralement requise.

81

Diffusion : linéaire et non-linéaire

L'équation $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial^2 x}$ est souvent appelée l'équation de la **diffusion de la chaleur**, ou encore l'équation de *Fick*.

En 2D, cette équation prend la forme

$$\frac{\partial u}{\partial t} = div(g \nabla u)$$

où g est un terme de « diffusivité », généralement positif.

En traitement d'images, la fonction *u(x,y,t)* peut se comprendre comme une image qui évolue dans le temps. C'est-à-dire une fonction dont les **niveaux de gris** représentent des concentrations locales qui se « diffusent » dans le temps et à travers l'image. En posant comme condition initiale:

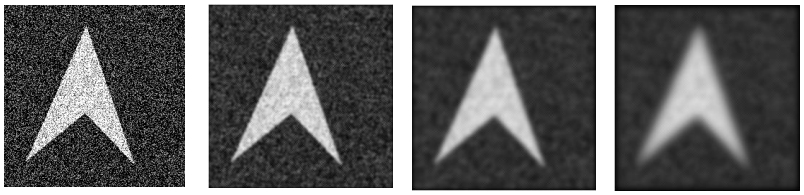
$$u(x, y, t=0) = f(x, y)$$

on peut voir *u(x,y,t)* comme une version filtrée de *f(x,y)*. Le filtrage s'opère en vertu des propriétés de la diffusion.

82

Diffusion : linéaire et non-linéaire

Avant d'avancer plus loin, voici comment intuitivement peuvent se diffuser les niveaux de gris dans l'image.



$u(x,y,t=0)=f(x,y)$ $u(x,y,t=1)$ $u(x,y,t=20)$ $u(x,y,t=100)$

83

Diffusion : linéaire et non-linéaire

Il est possible de résoudre l'équation de la diffusion (c-à-d trouver le contenu de l'image u au temps t) de façon numérique.

$$\frac{\partial u}{\partial t} = div(g\nabla u) = div\left(g\begin{pmatrix}\partial u/\partial x\\ \partial u/\partial y\end{pmatrix}\right) = div\left(\begin{pmatrix}g\partial u/\partial x\\ g\partial u/\partial y\end{pmatrix}\right) = \partial_x\left(g\frac{\partial u}{\partial x}\right)+\partial_y\left(g\frac{\partial u}{\partial y}\right)$$

Étant donnée la formule de la dérivée partielle introduite précédemment, on peut dire que

$$\frac{\partial u}{\partial x} \approx u^t_{i+1/2\,j} - u^t_{i-1/2\,j}$$

$$\frac{\partial u}{\partial t} \approx \frac{u^{t+1}_{ij} - u^t_{ij}}{\tau}$$ où τ est le "time step" et $u^t_{ij} = u(i,j,t)$

Par conséquent

$$\frac{\partial u}{\partial t} = \partial_x\left(g\frac{\partial u}{\partial x}\right)+\partial_y\left(g\frac{\partial u}{\partial y}\right)$$

$$\frac{u^{t+1}_{ij}-u^t_{ij}}{\tau} \approx \left(\left(g\frac{\partial u}{\partial x}\right)^t_{i+1/2\,j} - \left(g\frac{\partial u}{\partial x}\right)^t_{i-1/2\,j}\right)+\left(\left(g\frac{\partial u}{\partial y}\right)^t_{ij+1/2} - \left(g\frac{\partial u}{\partial y}\right)^t_{ij-1/2}\right)$$

84

Diffusion : linéaire et non-linéaire

$$\frac{u_{ij}^{t+1}-u_{ij}^{t}}{\tau} \approx \left(\left(g\frac{\partial u}{\partial x}\right)_{i+1/2\,j}^{t} - \left(g\frac{\partial u}{\partial x}\right)_{i-1/2\,j}^{t}\right) + \left(\left(g\frac{\partial u}{\partial y}\right)_{ij+1/2}^{t} - \left(g\frac{\partial u}{\partial y}\right)_{ij-1/2}^{t}\right)$$

$$= \left(g\left(u_{i+1/2\,j}^{t}-u_{i-1/2\,j}^{t}\right)\right)_{i+1/2\,j}^{t} - \left(g\left(u_{i+1/2\,j}^{t}-u_{i-1/2\,j}^{t}\right)\right)_{i-1/2\,j}^{t}$$
$$+ \left(g\left(u_{ij+1/2}^{t}-u_{ij-1/2}^{t}\right)\right)_{ij+1/2}^{t} - \left(g\left(u_{ij+1/2}^{t}-u_{ij-1/2}^{t}\right)\right)_{ij-1/2}^{t}$$

$$= g_{i+1/2\,j}^{t}(u_{i+1\,j}^{t}-u_{ij}^{t}) - g_{i-1/2\,j}^{t}(u_{ij}^{t}-u_{i-1\,j}^{t}) + g_{ij+1/2}^{t}(u_{ij+1}^{t}-u_{ij}^{t}) - g_{ij-1/2}^{t}(u_{ij}^{t}-u_{i-1\,j}^{t})$$

Diffusion : linéaire et non-linéaire

La façon la plus facile de résoudre cette équation est de façon « **explicite** »:

$$\frac{u_{ij}^{t+1}-u_{ij}^{t}}{\tau} \approx g_{i+1/2\,j}^{t}(u_{i+1\,j}^{t}-u_{ij}^{t}) - g_{i-1/2\,j}^{t}(u_{ij}^{t}-u_{i-1\,j}^{t}) + g_{ij+1/2}^{t}(u_{ij+1}^{t}-u_{ij}^{t}) - g_{ij-1/2}^{t}(u_{ij}^{t}-u_{i-1\,j}^{t})$$

$$u_{ij}^{t+1} = u_{ij}^{t} + \tau\left(g_{i+1/2\,j}^{t}(u_{i+1\,j}^{t}-u_{ij}^{t}) - g_{i-1/2\,j}^{t}(u_{ij}^{t}-u_{i-1\,j}^{t}) + g_{ij+1/2}^{t}(u_{ij+1}^{t}-u_{ij}^{t}) - g_{ij-1/2}^{t}(u_{ij}^{t}-u_{i-1\,j}^{t})\right)$$

où

$$g_{i\pm1/2\,j}^{t} = \frac{g_{i\pm1\,j}^{t}+g_{ij}^{t}}{2} \text{ et } g_{ij\pm1/2}^{t} = \frac{g_{ij\pm1}^{t}+g_{ij}^{t}}{2}$$

Diffusion : linéaire et non-linéaire

On peut réécrire cette dernière équation de la façon suivante:

$$u_{ij}^{t+1} \approx u_{ij}^{t} + \tau\left(g_{i+1/2\,j}^{t}(u_{i+1\,j}^{t} - u_{ij}^{t}) - g_{i-1/2\,j}^{t}(u_{ij}^{t} - u_{i-1\,j}^{t}) + g_{ij+1/2}^{t}(u_{ij+1}^{t} - u_{ij}^{t}) - g_{ij-1/2}^{t}(u_{ij}^{t} - u_{i-1\,j}^{t})\right)$$

$$u_{ij}^{t+1} = \tau g_{i+1/2\,j}^{t} u_{i+1\,j}^{t} + \tau g_{i-1/2\,j}^{t} u_{i-1\,j}^{t} + \tau g_{ij+1/2}^{t} u_{ij+1}^{t} + \tau g_{ij-1/2}^{t} u_{ij-1}^{t} + u_{ij}^{t}\left(1 - \tau g_{i+1/2\,j}^{t} - \tau g_{i-1/2\,j}^{t} - \tau g_{ij+1/2}^{t} - \tau g_{ij-1/2}^{t}\right)$$

Cette dernière équation peut s'exprimer sous la forme d'un filtre 3x3!

0	$\tau g_{ij-1/2}^{t}$	0
$\tau g_{i-1/2\,j}^{t}$	$1 - \tau g_{i+1/2\,j}^{t} - \tau g_{ij+1/2}^{t} - \tau g_{ij-1/2}^{t} - \tau g_{i-1/2\,j}^{t}$	$\tau g_{i+1/2\,j}^{t}$
0	$\tau g_{ij+1/2}^{t}$	0

87

Diffusion : **linéaire** et non-linéaire

0	$\tau g_{ij-1/2}^{t}$	0
$\tau g_{i-1/2\,j}^{t}$	$1 - \tau g_{i+1/2\,j}^{t} - \tau g_{ij+1/2}^{t} - \tau g_{ij-1/2}^{t} - \tau g_{i-1/2\,j}^{t}$	$\tau g_{i+1/2\,j}^{t}$
0	$\tau g_{ij+1/2}^{t}$	0

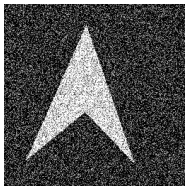
Attention! Si τ est trop élevé, alors le centre du filtre 3x3 devient négatif. Cela induit de l'instabilité et peut conduire le processus de filtrage à diverger. Par contre, si $|g| \le 1$ on obtient un solution stable avec $\tau \le 0.25$. Par exemple, si g est une constante égale à 1 et $\tau = 1/8$ alors on obtient le masque **linéaire**

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

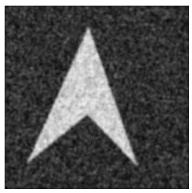
88

Diffusion : **linéaire** et non-linéaire

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



$u(x, y, t = 0) = f(x, y)$



$u(x, y, t = 1)$

$u(x, y, t = 20)$

$u(x, y, t = 100)$

Diffusion : linéaire et non-linéaire

Avec la diffusion, il est possible de filtrer l'image tout en préservant les discontinuités (contours). Pour y arriver, il nous faut réduire la diffusion près des contours, c-à-d là où le gradient est élevé

$$|\nabla u| = \sqrt{u_x^2 + u_y^2}$$

Il faut donc une « diffusivité » g qui diminue aux endroits où le gradient est élevé. Plusieurs fonctions ont été proposées, mentionnons-en trois

$$g(|\nabla u|) = \frac{1}{\sqrt{1 + \frac{|\nabla u|^2}{\lambda^2}}}$$ (Charbonnier)

$$g(|\nabla u|) = \frac{1}{1 + \frac{|\nabla u|^2}{\lambda^2}}$$ (Perona-Malik)

$$g(|\nabla u|) = e^{-|\nabla u|^2 / 2\lambda^2}$$

Diffusion : linéaire et **non-linéaire**

Si on résout l'équation de la diffusion

$$\frac{\partial u}{\partial t} = div(g(|\nabla u|)\nabla u)$$

à l'aide de l'approche **explicite** exposée précédemment, on obtient que le masque de filtrage est:

0	$\mathcal{g}(\nabla u)^t_{ij-1/2}$	0
$\mathcal{g}(\nabla u)^t_{i-1/2j}$	$1-\mathcal{g}(\nabla u)^t_{i+1/2j}-\mathcal{g}(\nabla u)^t_{ij+1/2}-\mathcal{g}(\nabla u)^t_{ij-1/2}-\mathcal{g}(\nabla u)^t_{i-1/2j}$	$\mathcal{g}(\nabla u)^t_{i+1/2j}$
0	$\mathcal{g}(\nabla u)^t_{ij+1/2}$	0

91

Diffusion : linéaire et **non-linéaire**

L'algorithme de la diffusion non-linéaire (méthode **explicite**) Algorithme C1

1. $u = f$ /* f est l'image bruitée de départ */

2. **POUR** *nbIterations* allant de 0 à MAX_ITER **FAIRE**

3. $|\nabla u| = \sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2}$

4. **POUR** tous les pixels *(i,j)* **FAIRE**

4a. $g_{ij} = g(|\nabla u_{ij}|)$

4b. $g_{i-1j} = \frac{\tau}{2}\left(g(|\nabla u_{i-1j}|) + g_{ij}\right)$

4c. $g_{i+1j} = \frac{\tau}{2}\left(g(|\nabla u_{i+1j}|) + g_{ij}\right)$

4e. $g_{ij-1} = \frac{\tau}{2}\left(g(|\nabla u_{ij-1}|) + g_{ij}\right)$

4e. $g_{ij+1} = \frac{\tau}{2}\left(g(|\nabla u_{ij+1}|) + g_{ij}\right)$

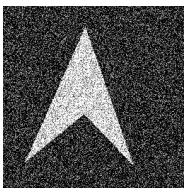
4f. $u^{t+1}_{ij} = g_{i-1j}u_{i-1j} + g_{i+1j}u_{i+1j} + g_{ij-1}u_{ij-1} + g_{ij+1}u_{ij+1} + u_{ij}\left(1 - g_{i-1j} - g_{i+1j} - g_{ij-1} - g_{ij+1}\right)$

5. $u = u^{t+1}$

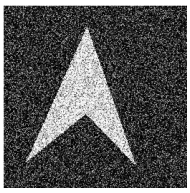
92

Diffusion : linéaire et **non-linéaire**

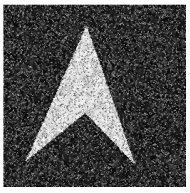
Résultats avec la diffusivité de Perona-Malik $\tau = 0.25, \lambda = 5$



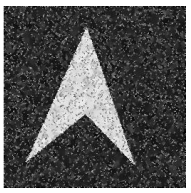
$u(x, y, t = 0) = f(x, y)$



$u(x, y, t = 1)$



$u(x, y, t = 100)$



$u(x, y, t = 200)$

$u(x, y, t = 500)$

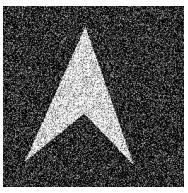
$u(x, y, t = 750)$

$u(x, y, t = 1000)$
(Convergence)

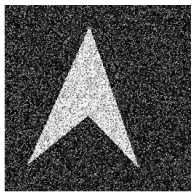
93

Diffusion : linéaire et **non-linéaire**

Résultats avec la diffusivité de Perona-Malik $\tau = 0.25, 1000$ itérations



$u(x, y, t = 0) = f(x, y)$



$\lambda = 1$

$\lambda = 3$

$\lambda = 5$

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

94

Diffusion : linéaire et **non-linéaire**

Résultats avec la diffusivité de Perona-Malik $\tau = 0.25, \lambda = 3{,}500$ itérations

95

Diffusion : linéaire et non-linéaire

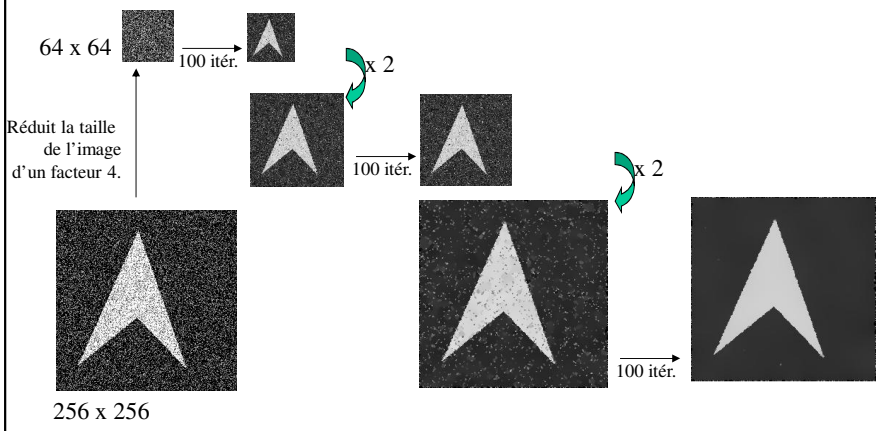
L'inconvénient avec les approches basées sur la diffusion est leur **lenteur**. Il existe toutefois des moyens pour accélérer les calculs:

(1) Implémentation sur architecture **parallèle** (comme les GPUs par exemple)
(2) Utiliser un optimiseur **implicite** (Jacobi, Gauss-Seidel, SOR)
(3) Approche **multiresolution**

96

Diffusion : linéaire et non-linéaire

Exemple d'une approche multirésolution simple (3 niveaux de résolution)



Même résultat avec 300 itérations qu'avec
1000 itérations et 1 niveau de résolution

97

Mean shift

98

Mean-Shift

La méthode du Mean-shift en imagerie a été développée par Dorin Comaniciu et Peter Meer à la fin des années 90. Depuis, leur méthode a été utilisée pour résoudre un nombre considérable de problèmes en imagerie. L'application première du Mean-shift est le filtrage avec préservation de contours.

Le Mean-shift est une méthode basée sur **l'estimation de densité** (sujet du cours portant sur les algorithmes d'apprentissage).

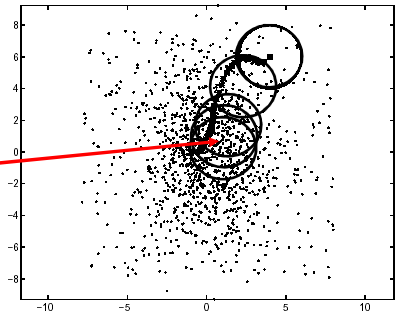
99

Mean-Shift

Pour la méthode du Mean-shift, on cherche la position qui, localement, possède la densité la plus élevée. Pour ce faire, on calcul la densité locale en un point et on déplace itérativement la fenêtre en direction du gradient de densité maximum. On appelle ce type de fenêtre un *kernel d'Epanechnikov*

Exemple 2D:

Densité maximum

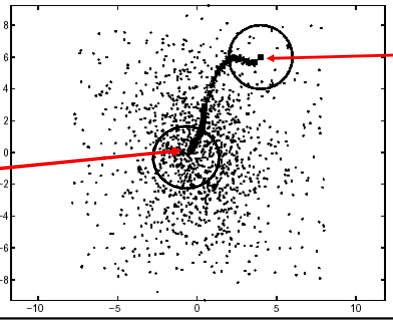


100
Comaniciu-Meer

Mean-Shift

À l'aide du kernel d'Epanechnikov, on peut atteindre le point ayant une densité maximum locale à l'aide d'un algorithme très simple.

1. $k = 1, y_k = x_j$
2. centrer la fenêtre S_h sur y_k
3. $y_{k+1} = \frac{1}{N_k} \sum_{\substack{i=0 \\ x_i \in S_h}}^{N-1} x_i \quad , k = k + 1$
4. retour à 2 si $y_{k+1} \neq y_k$

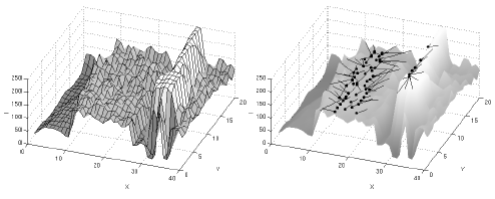


Mean-Shift

Lors du filtrage d'une image en niveau de gris *I(x,y)*, chaque pixel est considéré comme un point dans un espace 3D. Le pixel *I(i,j)* correspond à un point 3D dont les coordonnées sont *(i , j , I(i,j))*.

Exemple

Avec l'algorithme du Mean-shift, les points 3D se décalent vers la densité maximale locale



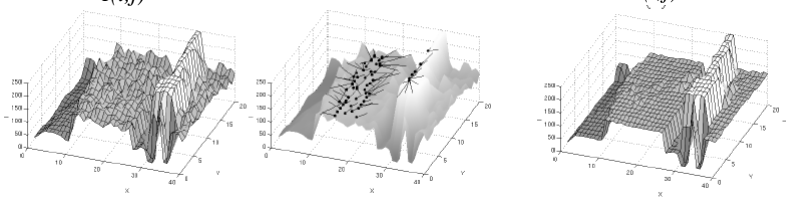
Algorithme C3

Mean-Shift

Lors du filtrage par mean-shift, pour chaque pixel *I(i,j)* de l'image

1. $k = 1, y_k = (i, j, I(i, j))$
2. centrer la fenêtre 3D S_h sur y_k
3. $y_{k+1} = \frac{1}{N_k} \sum_{\substack{i=0 \\ x_i \in S_h}}^{N-1} x_i \quad , k = k + 1$
4. retour à 2 si $y_{k+1} \neq y_k$
5. $I'(i, j) = y_k$

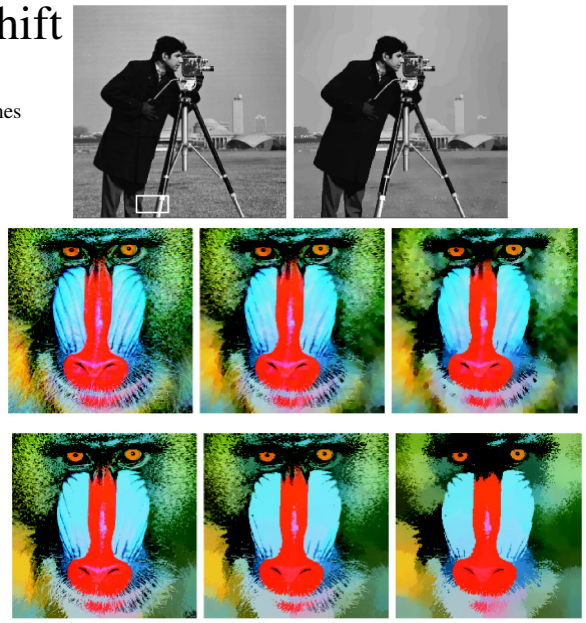
I(i,j) *I'(i,j)*



103
Comaniciu-Meer

Mean-Shift

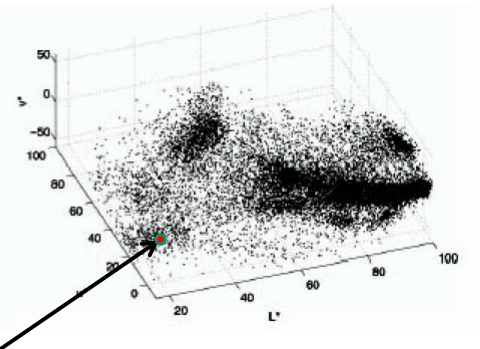
S_h peut avoir différentes tailles et différentes formes



104
Comaniciu-Meer

Mean-Shift

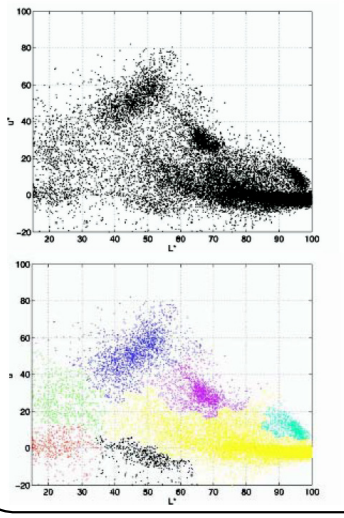
Means-Shift peut également servir d'outil de segmentation

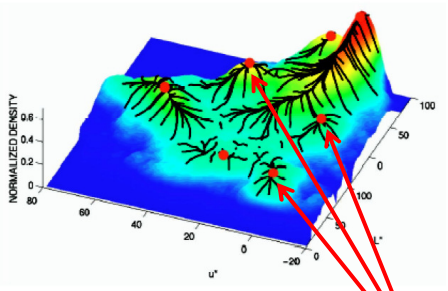


(r,g,b) ⟶ (l,u,v)

Mean-Shift

Means-Shift peut également servir d'outil de segmentation

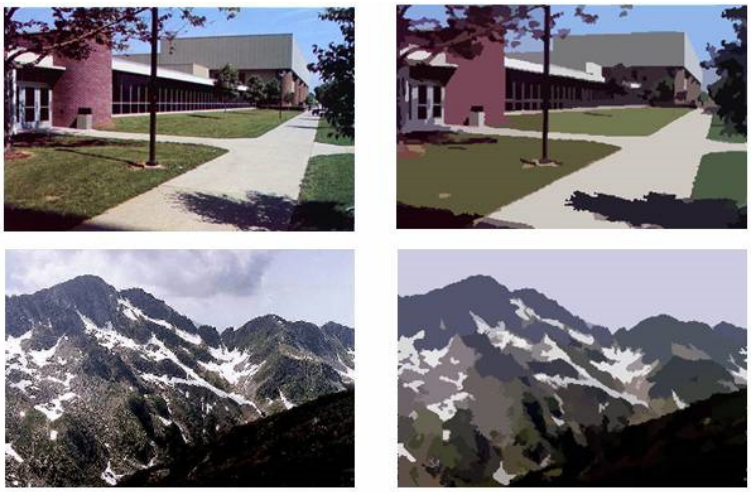




L'idée est de regrouper tous les pixels convergeant vers un même point

Mean-Shift

Means-Shift peut également servir d'outil de segmentation



107
Comaniciu-Meer

Mean-Shift

Mean-Shift ressemble un peu à *k-Means* à la différence que:

Avantages *Mean-Shift*	Avantages *K-Means*
• Trouve automatiquement le nombre de classes • Aucun a priori quant à la distribution des données • Ne dépend pas d'une « bonne initialisation » • Robuste aux « outliers »	• Simple • Beaucoup plus rapide

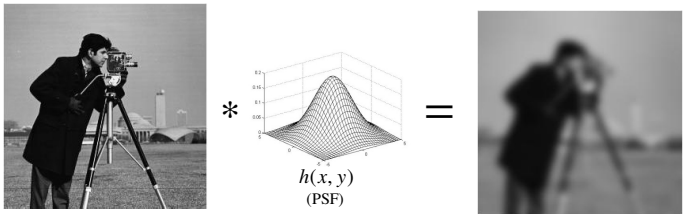
108

Déconvolution

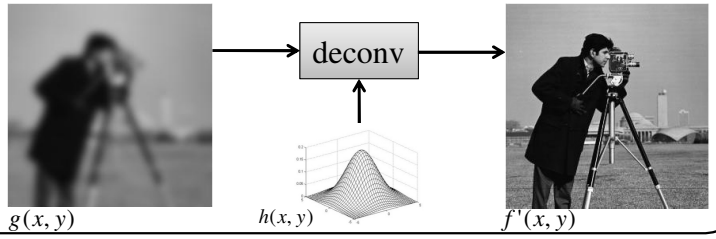
109

Déconvolution

Convolution

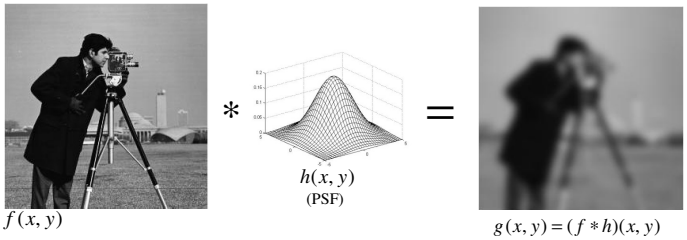


Déconvolution
(problème classique)

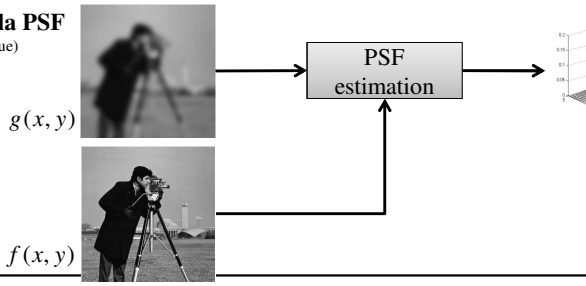


Déconvolution

Convolution



Estimation de la PSF
(autre problème classique)



Déconvolution

Déconvolution	• Filtrage inverse (et 2 variantes) • Filtre de Wiener • Filtrage par moindre carré • Filtrage itératif (Landweber) • Regularisation de Tikhonov • Filtre de Lucy-Richardson
Estimation de la PSF	• Filtre homomorphique
Déconvolution aveugle	• Filtre de Lucy-Richardson

112

Déconvolution

Filtrage inverse

Soit l'image dégradée g obtenue suite à une convolution

$$g = f * h$$

En vertu des propriétés de la TF et de l'opérateur *,

$$G = FH$$

Si on connait la PSF h, il est facile de calculer H et de retrouver l'image d'origine

$$F' = \frac{G}{H}$$

$$= GH_{inv} \qquad \text{où } H_{inv} = \frac{1}{H}$$

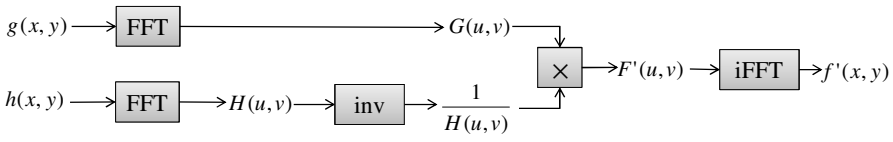
Et par conséquent

$$f' = g * h_{inv}$$

Déconvolution

Filtrage inverse

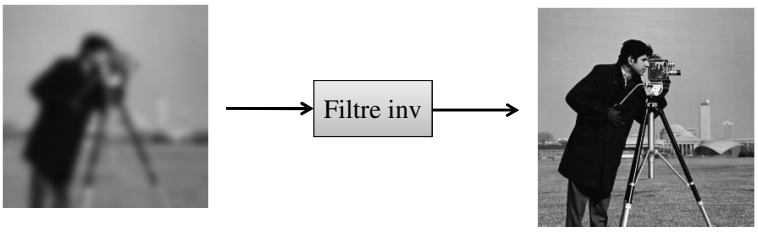
En général, le filtrage inverse s'effectue dans le domaine fréquentiel



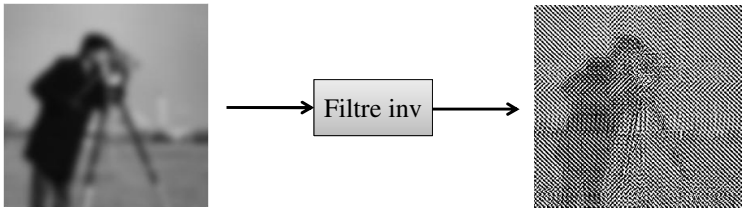
Note : $\frac{G}{H} = \frac{\overline{H}G}{|H|^2}$

Déconvolution

Si h est PARFAITEMENT connu alors



Si h est imprécis (même un peu) alors



Déconvolution

Autre problème avec le filtrage inverse.

Le filtrage inverse est très sensible au bruit. En effet, si

$$g = f * h + \eta$$

où η est un bruit blanc aléatoire, même si l'on connait parfaitement h, le résultat sera mauvais. Suite à une TF

$$G = FH + N$$

et par filtrage inverse :

$$\begin{aligned} F' &= \frac{G}{H} \\ &= \frac{FH + N}{H} \\ &= F + \frac{N}{H} \end{aligned}$$

→ Division par de petites valeurs dans les hautes fréquences

Déconvolution

Même si h est PARFAITEMENT connu

Bruit infime

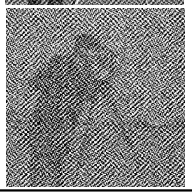
Filtre inv

Bruit minime

Filtre inv

Bruit faible

Filtre inv

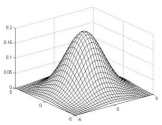


Déconvolution

Conclusion, le filtrage inverse est une opération extrêmement instable. **Pourquoi?**

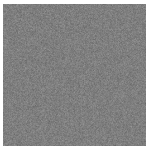
$f(x,y)$

$*$



$h(x,y)$

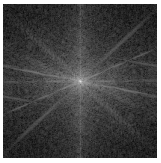
$+$



η

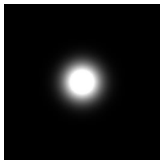
$=$

$(f*h+\eta)(x,y)$



$F(u,v)$

$\times$

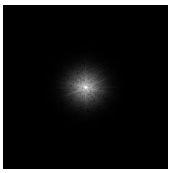


$H(u,v)$

$+$

N

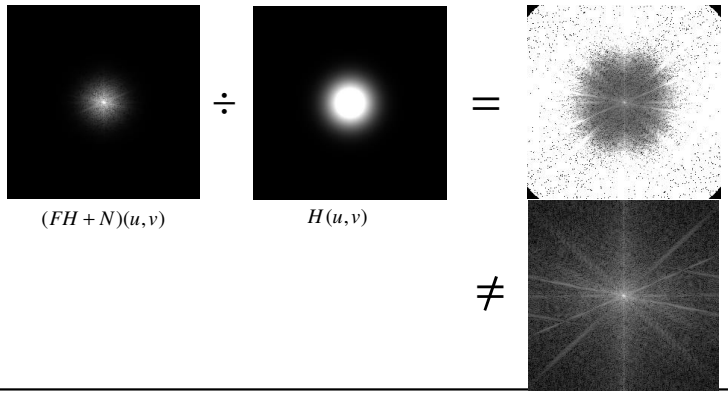
$=$



$(FH+N)(u,v)$

Déconvolution

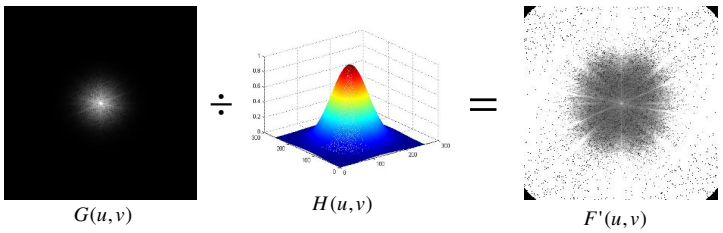
Lorsque H(u,v) n'est pas parfaitement défini, il s'ensuit une division par des chiffres très très petits et donc un "boost" dans les hautes fréquences.



$(FH+N)(u,v)$ $\quad$ $H(u,v)$

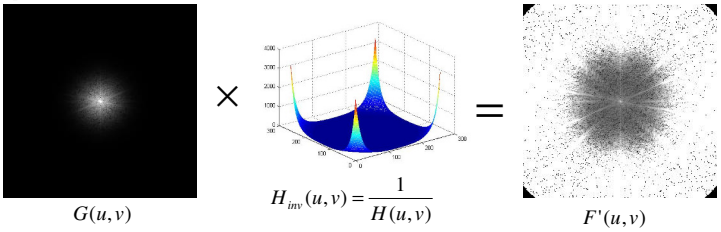
Déconvolution

Lorsque H(u,v) n'est pas parfaitement défini, il s'ensuit une division par des chiffres très très petits et donc un "boost" dans les hautes fréquences.



$G(u,v)$ $\quad$ $H(u,v)$ $\quad$ $F'(u,v)$

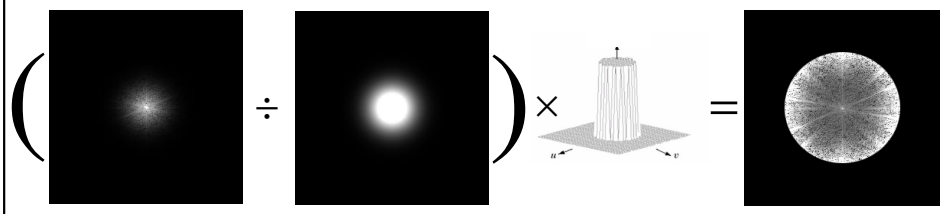
Équivalent à



$G(u,v)$ $\quad$ $H_{inv}(u,v)=\frac{1}{H(u,v)}$ $\quad$ $F'(u,v)$

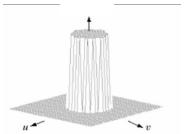
Déconvolution

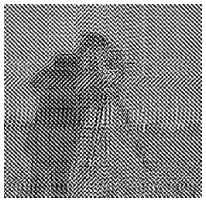
Étant donné que les basses fréquences ne sont pas dégénérées,
une première solution est d'appliquer un filtre passe bas rectangulaire.



Déconvolution

Étant donné que les basses fréquences ne sont pas dégénérées,
une première solution est d'appliquer un filtre passe bas rectangulaire.



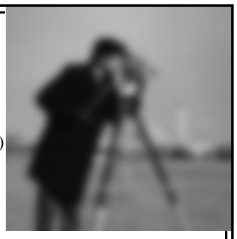


Rayon = 128

Rayon = 64

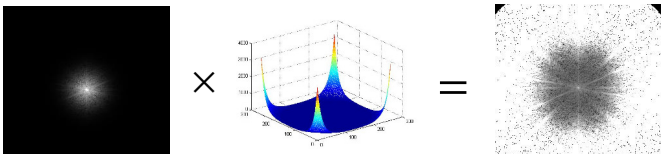
Rayon = 32

Déconvolution

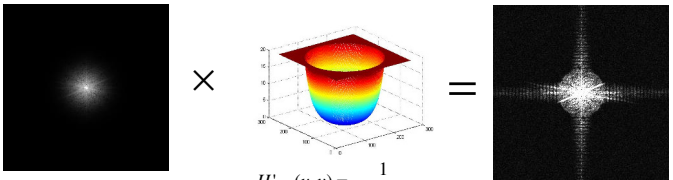


Autre solution, éliminer les valeurs trop faibles de $H(u,v)$ (ou trop élevée de $H_{inv}(u,v)$)

$$H'(u,v) = MAX(H(u,v), \tau)$$

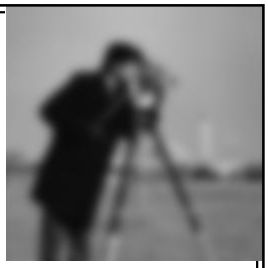


$G(u,v)$ $\quad$ $H_{inv}(u,v) = \frac{1}{H(u,v)}$ $\quad$ $F'(u,v)$

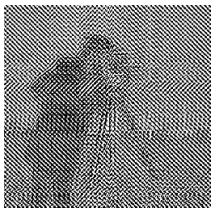


$G(u,v)$ $\quad$ $H'_{inv}(u,v) = \frac{1}{H'(u,v)}$ $\quad$ $F'(u,v)$

Déconvolution

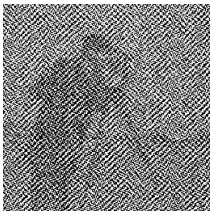


h est imprécis, τ=0



h est imprécis, pas de bruit, $\tau = 10^{-2}$

bruit moyen, τ=0



h précis, bruit moyen, $\tau = 10^{-3}$

124

Déconvolution

Filtre de Wiener

L'objectif du filtre de Wiener est de trouver une image restaurée *f'* tel que

$$f'(x, y) = \arg\min_{f'} \|f' - f\| \quad (*)$$

Étant donné que $g = f * h + \eta$ et que le bruit η non corrélé de moyenne nulle, on peut démontrer que l'opération qui satisfait l'équation (*) est

$$F'(u,v) = \frac{\overline{H}(u,v)}{|H(u,v)|^2 + \frac{|N(u,v)|^2}{|F(u,v)|^2}} G(u,v)$$

où

$|N(u,v)|^2$: Spectre de puissance du bruit

$|F(u,v)|^2$: Spectre de puissance de l'image d'origine

$\overline{H}(u,v)$: est le conjugué complexe de *H(u,v)*

Déconvolution

Filtre de Wiener

Étant donné que $|N|^2$ et $|F|^2$ sont généralement inconnus, on remplace $\frac{|N|^2}{|F|^2}$ par une constante k :

$$F'(u,v) = \frac{\overline{H}(u,v)}{|H(u,v)|^2 + k} G(u,v)$$

Ou, de façon équivalente,

$$F' = \frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)|^2 + k} G(u,v)$$

Vous remarquerez que lorsque $k \rightarrow 0$ on obtient un filtre inverse

$$F'(u,v) = \frac{G(u,v)}{H(u,v)}$$

$k \rightarrow 0$ signifie que le spectre de puissance du bruit tend vers zéro

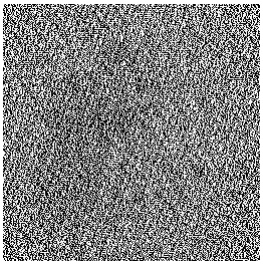
Déconvolution

Filtre de Wiener

$g = f * h + \eta$

(flou directionnel)

$k = 0.0$ (filtrage inverse)



$k = 0.02$

Déconvolution

Filtrage itératif, Landweber

Cherche à minimiser la même fonction de coût :

$$f' = \arg\min_{\hat{f}} \left\| g - \hat{f} * h \right\|^2$$

mais **sans division.** Pour y arriver, on procède par une **descente de gradient** (voir cours optimisation).

$$f'_{k+1}(x,y) = f'_k(x,y) + \alpha h(-x,-y) * \left(g(x,y) - h(x,y) * f'_k(x,y) \right)$$

où α est la « pas d'optimisation » (α=1) et $f'_0(x,y) = g(x,y)$

Déconvolution

Filtrage itératif, Landweber

$$f'_{k+1}(x,y) = f'_k(x,y) + \alpha h(-x,-y) * \left(g(x,y) - h(x,y) * f'_k(x,y)\right)$$

ou encore en version fréquentielle

$$F'_{k+1}(u,v) = F'_k(x,y) + \alpha \overline{H}(u,v)\left(G(u,v) - H(u,v)F'_k(u,v)\right)$$

Étant donné que *k* peut aller jusqu'à 200 itérations, la version fréquentielle est **beaucoup** plus rapide.

Déconvolution

Filtrage itératif, Landweber

Vidéo
(landweber.wmv)

Déconvolution

Filtrage itératif, Landweber

Image originale, floue et bruitée

Landweber

Landweber fonctionne généralement un peu mieux **car il n'implique pas de division**.

Déconvolution

Filtrage par moindre carrés

Considérant la dégradation suivante

$$g = f * h + \eta$$

On cherche à minimiser l'erreur suivante

$$E(f') = \left\| g - f' * h \right\|^2$$

ou encore

$$f' = \arg\min_{f'} E(f')$$

Qui se lit ainsi : *la meilleure image résultante f' est celle qui minimise E(f)*

132

Déconvolution

Filtrage par moindre carrés

On peut représenter la fonction de coût dans le domaine de Fourier

$$E(F') = \left\| G - F'\overline{H} \right\|^2$$

Pour trouver la meilleure image *F'*, on force la dérivée à zéro

$$\frac{dE(F')}{dF'} = 0 = 2(G - F'\overline{H})\overline{H}$$

$$\Rightarrow G\overline{H} = F' \| H \|^2$$

$$\Rightarrow F' = \frac{G\overline{H}}{\| H \|^2}$$

$$\Rightarrow F' = \frac{G}{H}$$

En procédant de la sorte, un **filtrage par moindre carrés = filtrage inverse**

Déconvolution

Régularisation de Tikhonov

Étant donné que la méthode par moindre carrés n'est pas meilleure qu'un filtre inverse, Tikhonov proposent de minimiser la fonction de coût suivante :

$$E(f') = \left\| g - f' * h \right\|^2 + \beta \left\| \nabla^2 f' \right\|^2$$

Dans le domaine fréquentiel

$$E(F') = \left\| G - F'\overline{H} \right\|^2 + \beta \left\| \overline{L} F' \right\|^2$$

où L est la TF de l'opérateur laplacien.

Encore une fois, on peut trouver la meilleure solution en forçant à zéro la dérivée de *E(f ')*

$$\frac{dE(F')}{dF'} = 0$$

Déconvolution

Régularisation de Tikhonov

$$\frac{dE(F')}{dF'} = 0 = 2\left(G - F'\overline{H}\right)\overline{H} + 2\beta\left(\overline{L}F'\right)\overline{L}$$

$$\Rightarrow G\overline{H} - F'\left\|H\right\|^2 + \beta\left\|L\right\|^2 F'$$

$$\Rightarrow G\overline{H} - F'\left(\left\|H\right\|^2 - \beta\left\|L\right\|^2\right)$$

$$G\overline{H} = F'\left(\left\|H\right\|^2 - \beta\left\|L\right\|^2\right)$$

$$F' = \frac{G\overline{H}}{\left\|H\right\|^2 - \beta\left\|L\right\|^2}$$

135

Déconvolution

Régularisation de Tikhonov

$$F'(u,v) = \frac{\overline{H}(u,v)}{\left\|H(u,v)\right\|^2 - \beta\left\|L\right\|^2} G(u,v)$$

Image originale floue et bruitée

(bruit faible)

β faible β moyen β élevé

(bruit prononcé)

Déconvolution

Filtrage itératif, Landweber

Image originale floue et
TRÈS BRUITÉE

Regularisation de Tichonov

Landweber

Déconvolution

Lucy-Richardson

Approche probabiliste ayant pour objectif de trouver l'image f la plus probable étant donné g

$$f' = \arg\max_f P(f \mid g)$$

$$= \arg\max_f \frac{P(g \mid f)P(f)}{P(g)} \quad \text{Par la théorème de Bayes}$$

Suivant certaines hypothèses simplificatrices, on obtient que

$$f'_{k+1}(x,y) = f'_k(x,y)\left(\left[\frac{g(x,y)}{c(x,y)}\right] * h(-x,-y)\right)$$

Où

$$c(x,y) = f'_k(x,y) * h(x,y)$$

Richardson, W.H., "Bayesian-Based Iterative Method of Image Restoration", J. Opt. Soc. Am., 62, 55, (1972).
Lucy, L.B., "An iterative technique for the rectification of observed distributions", Astron. J., 79, 745, (1974).

Déconvolution

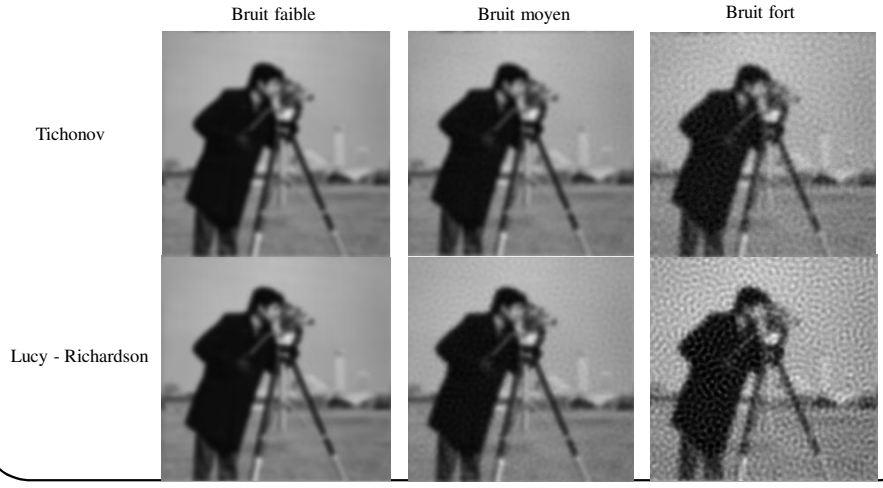
Lucy-Richardson

Caractéristiques:

- fonctionne bien lorsque le niveau de **bruit est faible** (voir page suivante)
- peut être utilisé pour faire de la **déconvolution aveugle** (voir plus loin).

Lucy-Richardson Vs niveau de bruit

f  g



Lucy-Richardson Vs niveau de bruit

Conclusion:

Lucy-Richardson meilleur que Tichonov lorsque le bruit est faible

Tichonov meilleur que Lucy-Richardson lorsque le bruit est fort.

En résumé

Pour du bruit faible

g Landweber Tichonov L-R

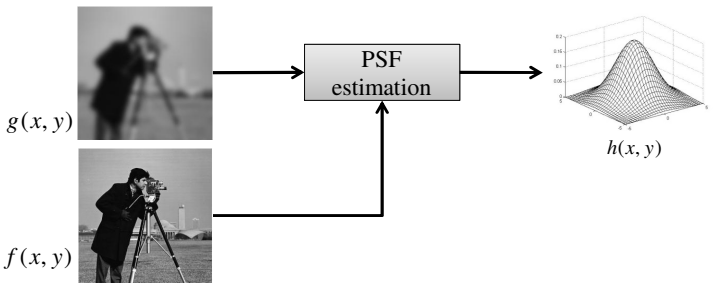
En résumé

Pour du bruit faible

Filtrage inverse < Filtrage inverse + filtre rect. < filtrage inverse + seuil < Weiner < Landweber < { L-R ; Tichonov }

Estimation de la PSF

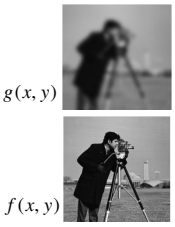
$g(x, y) = (f * h)(x, y)$



Solution la plus simple : **filtre homomorphique**

Estimation de la PSF

Filtre homomorphique



g* et *f* sont connus, on cherche à trouver *h

$$g(x,y) = (f * h)(x,y)$$

$$G(u,v) = F(u,v)H(u,v)$$

$$\ln\big(G(u,v)\big) = \ln\big(F(u,v)H(u,v)\big)$$
$$\ln\big(G(u,v)\big) = \ln\big(F(u,v)\big) + \ln\big(H(u,v)\big)$$

$$\ln\big(G(u,v)\big) - \ln\big(F(u,v)\big) = \ln\big(F(u,v)\big) + \ln\big(H(u,v)\big) - \ln\big(F(u,v)\big)$$

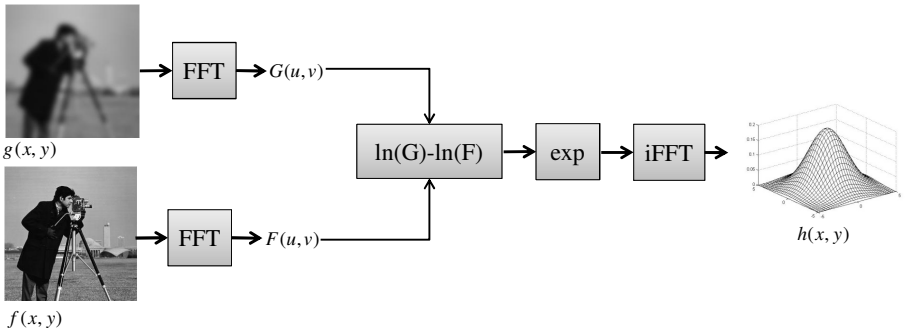
$$\ln\big(H(u,v)\big) = \ln\big(G(u,v)\big) - \ln\big(F(u,v)\big)$$

$$H(u,v) = e^{\ln(G(u,v)) - \ln(F(u,v))}$$

$$h(x,y) = \Im^{-1}\left(e^{\ln(G(u,v)) - \ln(F(u,v))}\right)$$

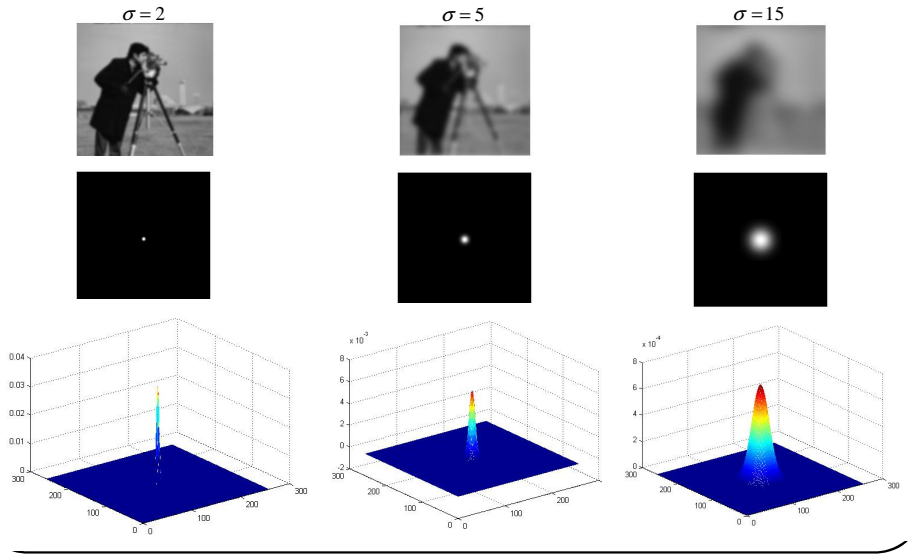
Estimation de la PSF

Filtre homomorphique



Estimation de la PSF

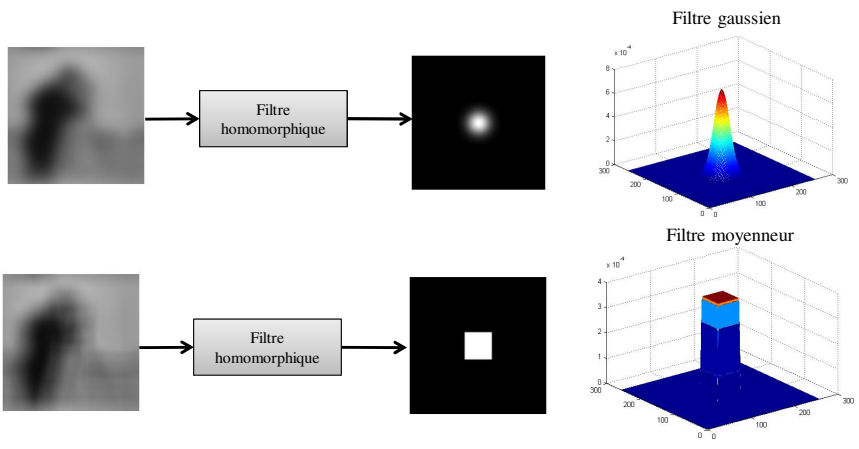
Filtre homomorphique



Estimation de la PSF

Filtre homomorphique

Gros avantage : fonctionne pour TOUS les types de flou



Estimation de la PSF

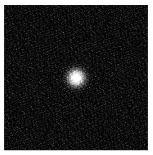
En cas de bruit dans l'image *g(x,y)*: $g(x,y) = (f * h)(x,y) + \eta$

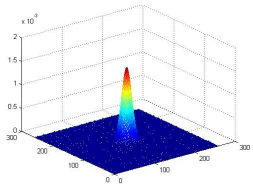
$\sigma = 10$

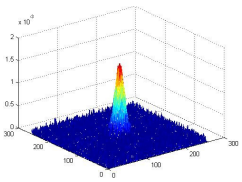
Bruit faible

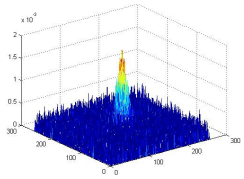
Bruit moyen

Bruit élevé









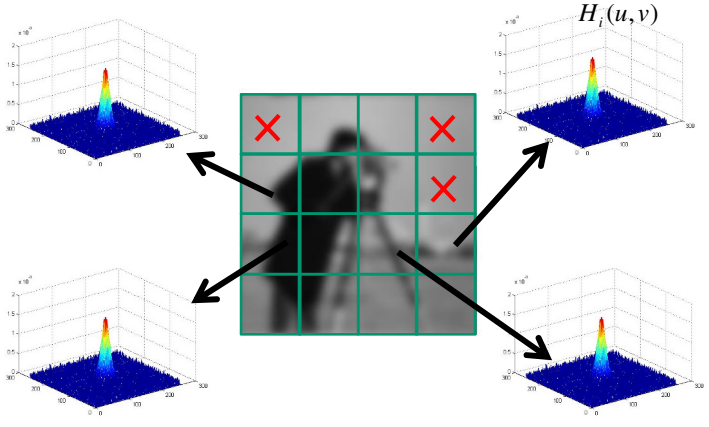
Estimation de la PSF

En réponse au problème du bruit, on fait la moyenne des résultats obtenus pour différentes régions texturées de l'image

$$H(u,v) = \frac{1}{M}\sum_i H_i(u,v)$$

✗ Régions trop peu texturées

$H_i(u,v)$



Déconvolution aveugle

Dans le cas d'une déconvolution aveugle, ***f* ET *h*** sont inconnus et doivent être estimés

$$g(x,y) = (f * h + \eta)(x,y)$$

à estimer

Parmi les solutions envisagées figure Lury-Richardson où ***f* et *h*** sont estimés alternativement

151

Déconvolution aveugle

$f'_0(x,y) \leftarrow$ Initialisation $(g(x,y))$

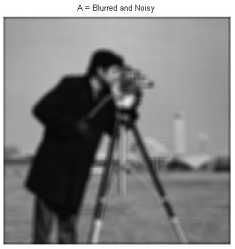
$h'_o(x,y) \leftarrow$ Initialisation (filtre gaussien)

$$f'_{k+1}(x,y) = f'_k(x,y)\left(\left[\frac{g(x,y)}{f'_k(x,y) * h_k(x,y)}\right] * h_k(-x,-y)\right)$$

$$h'_{k+1}(x,y) = h'_k(x,y)\left(\left[\frac{g(x,y)}{f'_{k+1}(x,y) * h'_k(x,y)}\right] * f_{k+1}'(-x,-y)\right)$$

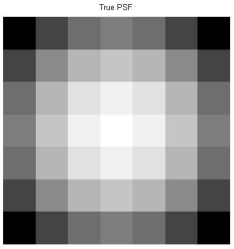
152

Déconvolution aveugle



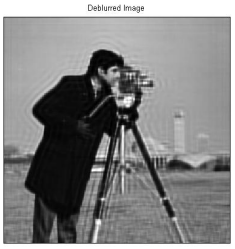
$g(x, y)$

Aucun bruit

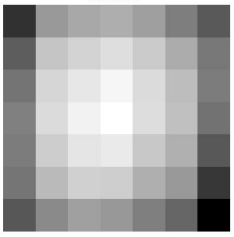


$h(x, y)$

Filtre gaussien



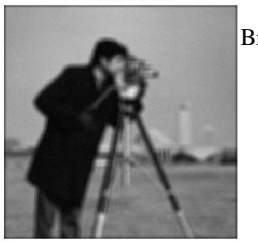
$f'(x, y)$



$h'(x, y)$

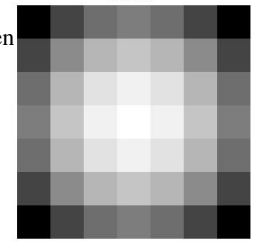
153

Déconvolution aveugle



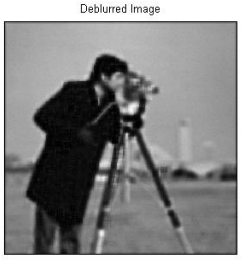
$g(x, y)$

Bruit moyen

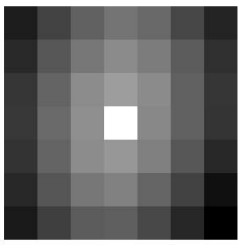


$h(x, y)$

Filtre gaussien



$f'(x, y)$



$h'(x, y)$

154

Déconvolution aveugle

Filtre directionnel

$g(x, y)$

$f'(x, y)$

$h'(x, y)$

155

Bruit et métriques de qualité

156

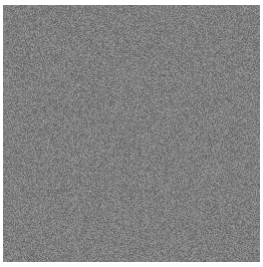
Bruit

$$g = f * h + \eta$$

Bruit additif non corrélé

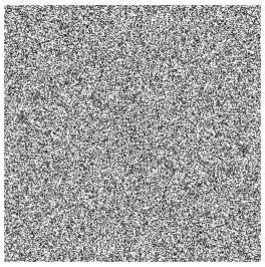
En général, **le bruit est blanc**, c-à-d que sa distribution fréquentielle est globalement uniforme

Bruit blanc (η)



$\Im$

Espace de Fourier

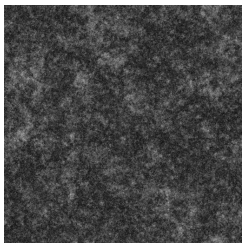


157

Bruit

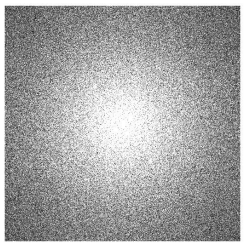
Mais il existe plusieurs autres types de bruits additifs

Bruit rose

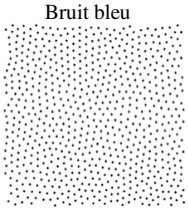


$\Im$

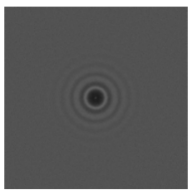
Espace de Fourier



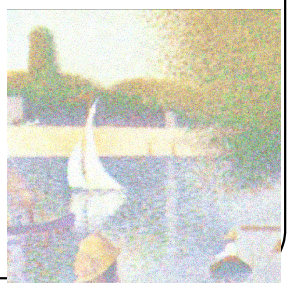
Bruit bleu



$\Im$



Exemple bruit bleu



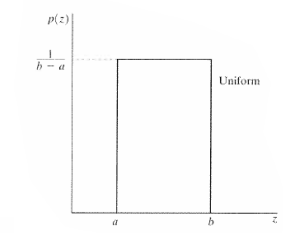
Bruit

La « **couleur** » du bruit indique sa distribution **spectrale**.

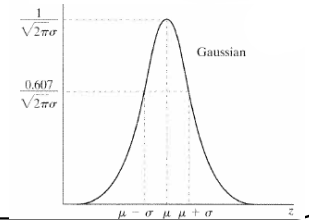
Pour une « couleur » de bruit, il peut exister plusieurs types de distributions spatiales.

Une variable de bruit « z » est considérée comme étant une **variable aléatoire** suivant une **densité de probabilités**

Bruit uniforme $p(z) = \frac{1}{b-a} \quad \forall a \le z \le b$

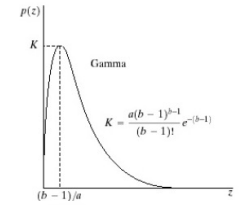


Bruit gaussien $p(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}$

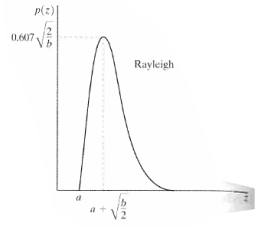


Bruit

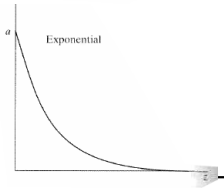
Bruit gamma $p(z) = \frac{a^b z^{b-1}}{(b-1)!} e^{-az} \quad \forall z \ge 0$



Bruit de Rayleigh $p(z) = \frac{2(z-a)}{b} e^{-\frac{(z-a)^2}{b}} \quad \forall z \ge a$

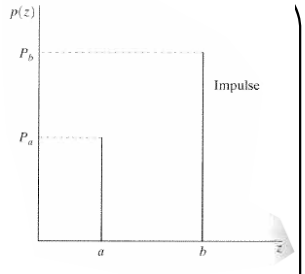


Bruit exponentiel $p(z) = ae^{-az} \quad \forall z \ge 0$



Bruit

Bruit poivre et sel $p(z) = \dfrac{1}{b-a} \quad \forall a \le z \le b$



Exemple de bruit sur une image composée de 3 régions uniformes

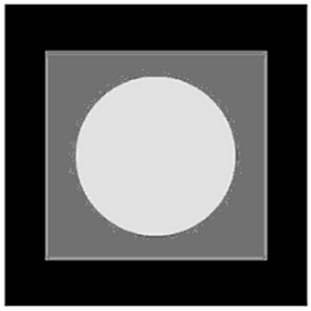
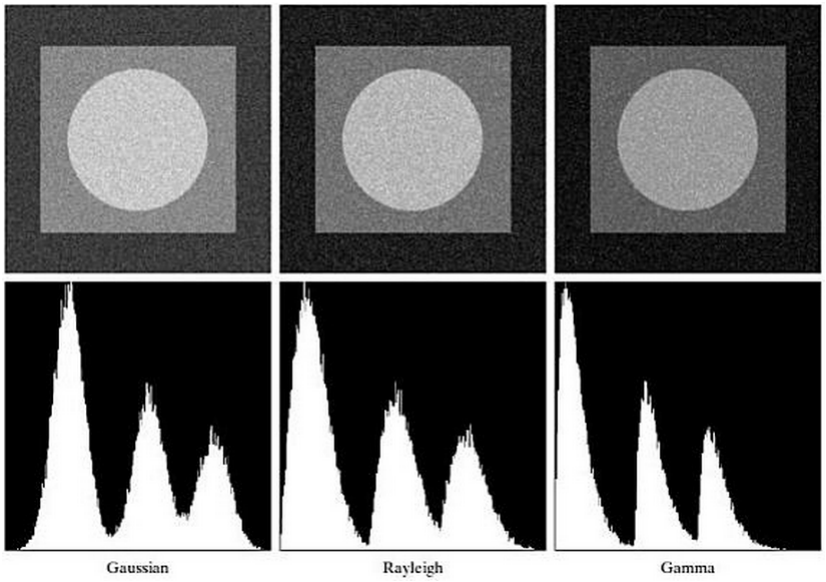


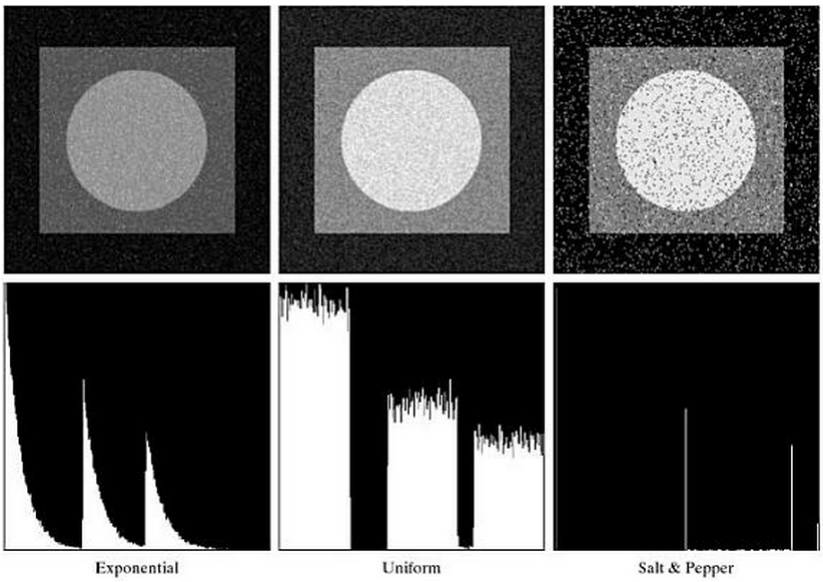
Image d'origine non bruitée

161



162

Crédit : Gonzalez et woods



163

Crédit : Gonzalez et woods

Métriques de qualité

Une métrique de qualité est une fonction qui mesure la **qualité visuelle** d'une image. Ainsi plus une image est dégradée par du flou, du bruit ou des artifacts de compression (e.g. JPEG) plus la qualité sera faible.

Qualité faible

Qualité bonne

Qualité excellente

164

Métriques de qualité

Les métriques de qualité sont souvent utilisées pour évaluer les méthodes de débruitage et de déconvolution

Exemple :

Image d'origine f

Image dégradée g

f_1 Image débruitée par la méthode 1

f_2 Image débruitée par la méthode 2

$MSSIM\ (f, f_1) > MSSIM\ (f, f_2)$

Par conséquent la méthode 1 est meilleure que la méthode 2

165

Évaluation subjective de la qualité

L'objectif ici est de créer une mesure de qualité qui sera aussi fiable qu'un être humain. Pour ce faire, on doit d'abord s'intéresser à **l'évaluation subjective de la qualité** c'est-à-dire comprendre de façon empirique comment un être humain juge de la qualité d'une image.

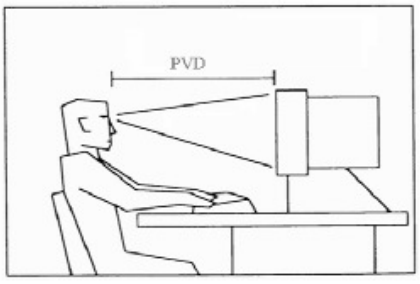


Figure 1 – *Conditions d'évaluation de la qualité*

Crédit : Charrier, Larabi et Saadane

Évaluation subjective de la qualité

L'objectif ici est de créer une mesure de qualité qui sera aussi fiable qu'un être humain. Pour ce faire, on doit d'abord s'intéresser à **l'évaluation subjective de la qualité** c'est-à-dire comprendre de façon empirique comment un être humain juge de la qualité d'une image.

Environnement normalisé

Standards ISO 3664 et ITU-R 500-10

- Distance d'environ 1 mètre à l'écran
- Angle d'observation d'environ 30 degrés
- Écran de haute qualité de 22 à 26 pouces
- Écran calibré (white balance, correction gamma, etc.)
- Chromacité de l'arrière plan = illuminant D65 (D65 -> centre du diagramme de chromacité)
- Sources de lumière D65
- Aucune lumière dans le champ visuel de l'observateur
- Aucune lumière devant refléter l'écran

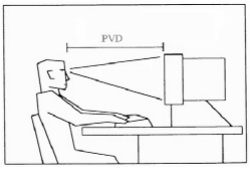


Figure 1 – *Conditions d'évaluation de la qualité*

Figure 2 – *Exemple de salle d'évaluation*

Crédit : Charrier, Larabi et Saadane

Évaluation subjective de la qualité

Séance d'évaluation

- Séance d'au plus 30 minutes
- Le score des 15-20 premières images doit être ignoré car tout observateur « stabilise » son opinion au début de la séance
- Au moins 15 observateurs.
- Les observateurs doivent avoir un appareil visuel normal

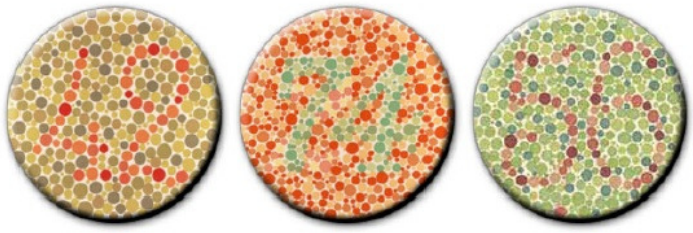


Figure 4: *Test d'Ishihara.*

Crédit : Charrier, Larabi et Saadane

Évaluation subjective de la qualité

Tests

On peut faire des **tests comparatifs** et des tests **de mesure absolue**. Les tests comparatifs sont généralement plus souvent retenus

Tests comparatifs : ordonner les images de la meilleure à la moins bonne

Artéfacts de compression

Figure 5 : *Test d'ordonnancement.*

169

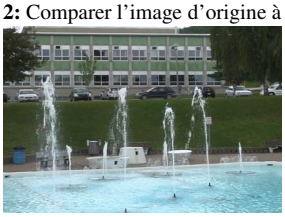
Crédit : Charrier, Larabi et Saadane

Évaluation subjective de la qualité

Tests

On peut faire des **tests comparatifs** et des tests **de mesure absolue**. Les tests comparatifs sont généralement plus souvent retenus

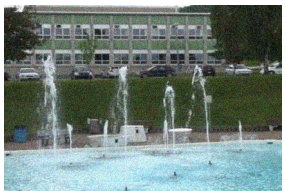
Tests 2 à 2: Comparer l'image d'origine à sa version dégradée



VS

Artéfacts de compression

VS



Bruit

170

Évaluation subjective de la qualité

Tests

On peut faire des **tests comparatifs** et des tests **de mesure absolue**. Les tests comparatifs sont généralement plus souvent retenus

Tests 2 à 2: Comparer l'image d'origine à sa version dégradée

Echelle à cinq notes		
Qualité		Dégradation
Excellente	5	Imperceptible
Bonne	4	Perceptible mais non gênant
Assez bonne	3	Légèrement gênant
Médiocre	2	Gênant
Mauvaise	1	Très gênant

MOS (*mean opinion score*)

$$\overline{u}_{jk} = \frac{1}{N}\sum_{i=1}^{N} u_{ijk}$$

Où u_{ijk} est la note de l'observateur i pour la dégradation j de l'image k. N est le nombre d'observateurs

171

Évaluation subjective de la qualité

Tests

On peut faire des **tests comparatifs** et des tests **de mesure absolue**. Les tests comparatifs sont généralement plus souvent retenus

Test de mesure absolue : Donner une note de 1 à 5 à une image dégradée

Echelle à cinq notes		
Qualité		Dégradation
Excellente	5	Imperceptible
Bonne	4	Perceptible mais non gênant
Assez bonne	3	Légèrement gênant
Médiocre	2	Gênant
Mauvaise	1	Très gênant

MOS (*mean opinion score*)

$$\overline{u}_{jk} = \frac{1}{N}\sum_{i=1}^{N} u_{ijk}$$

Où u_{ijk} est la note de l'observateur i pour la dégradation j de l'image k. N est le nombre d'observateurs

172

Métriques de qualité

Le but est de trouver une métrique de qualité qui saura **imiter le MOS**. Les métriques de mesure absolue sont relativement rares, souvent complexe et vont au-delà du cadre de ce cours. Les métriques les plus fréquentes sont les **métriques d'évaluation 2 à 2**.

Erreur quadratique moyenne (***mean square error***)

$$MSE(f, g) = \frac{1}{nm} \sum_{i,j} \left(f(i, j) - g(i, j) \right)^2$$

Ratio signal sur bruit (***signal to noise ratio***)

$$\begin{aligned} SNR(f, g) &= 10 \log \frac{P_f}{P_{f-g}} \\ &= 10 \log \frac{\sum_{ij} f(i, j)^2}{\sum_{ij} \left(f(i, j) - g(i, j) \right)^2} \\ &= 20 \log \frac{\sum_{ij} f(i, j)}{\sum_{ij} f(i, j) - g(i, j)} \end{aligned}$$

173

Métriques de qualité

Une métrique beaucoup plus souvent utilisée est le PSNR.

Ratio signal sur bruit impulsionnel (***Peak signal to noise ratio***)

$$PSNR(f, g) = 10 \log \left(\frac{d^2}{MSE(f, g)} \right)$$

où d est la valeur maximale du signal (ici 255)

174

Métriques de qualité

Le problème avec MSE, SNR et PSNR est qu'une légère modification d'une image, parfois même imperceptible par l'œil humain, peut avoir un effet majeur sur ces métriques. Par conséquent, d'autres métriques ont été proposés. Parmi les plus utilisées sont

UQI (Universal Quality Index)
SSIM (Structural SIMilarity)

Z Wang, A Bovik, **A Universal Image Quality Index,** IEEE Signal Processing Letters, 13(4), 2002

Z Wang, A Bovik, H. Sheikh, E. Simoncelli **Image Quality Assessment: From Error Visibility to Structural Similarity,** IEEE TIP, 13(4), 2004

Métriques de qualité

UQI: μ_f, μ_g : moyenne des images f et g

σ_f, σ_g : écart − type des images f et g

σ_{fg} : covariance $\frac{1}{nm}\sum_{i,j}(f_{ij}-\mu_f)(g_{ij}-\mu_g)$

$$UQI(f,g) = \frac{4\sigma_{xy}\mu_x\mu_y}{(\sigma_x^2+\sigma_y^2)(\mu_x^2+\mu_y^2)}$$

Z Wang, A Bovik, **A Universal Image Quality Index,** IEEE Signal Processing Letters, 13(4), 2002

177

Métriques de qualité

UQI:

Les auteurs précisent qu'il est préférable d'utiliser une **version locale** de UQI:

$$UQI(f, g) = \sum_i UQI(f_i, g_i)$$

où f_j, g_j : est le contenu de l'image à l'intérieur de la *j-ème* petite fenêtre (par exemple 11x11)

Z Wang, A Bovik, **A Universal Image Quality Index,** IEEE Signal Processing Letters, 13(4), 2002 178

UQI meilleure que MSE

(Images dégradées ayant la même MSE)

179

Métriques de qualité

SSIM:

$$l(f,g)=\frac{2\mu_f\mu_g+C_1}{\mu_f^2+\mu_g^2+C_1} \qquad c(f,g)=\frac{2\sigma_f\sigma_g+C_2}{\sigma_f^2+\sigma_g^2+C_2} \qquad s(f,g)=\frac{2\sigma_{fg}+C_3}{\sigma_f\sigma_g+C_3}$$

μ_f, μ_f : moyenne des images f et g

σ_f, σ_g : écart – type des images f et g

σ_{fg} : covariance $\frac{1}{nm}\sum_{i,j}(f_{ij}-\mu_f)(g_{ij}-\mu_g)$

C_1, C_2, C_3 : constantes

$$SSIM\ (f,g)=l(f,g)^{\alpha}\, c(f,g)^{\beta}\, s(f,g)^{\gamma}$$

Z Wang, A Bovik, H. Sheikh, E. Simoncelli **Image Quality Assessment: From Error Visibility to Structural Similarity,** IEEE TIP, 13(4), 2004

Métriques de qualité

Tout comme UQI, il est préférable d'utiliser une **version locale** de SSIM, une version qu'ils nomment *mean structural simmilarity* **MSSIM**

$$MSSIM\ (f,g)=\frac{1}{N}\sum_j SSIM\ (f_j, g_j)$$

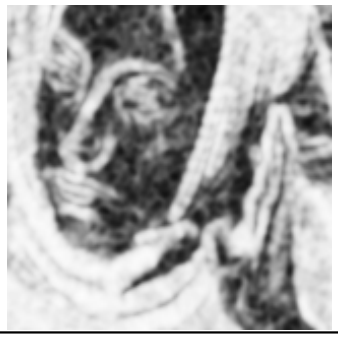
où f_j, g_j :est le contenu de l'image à l'intérieur de la *j-ème* petite fenêtre (par exemple 11x11)

Z Wang, A Bovik, H. Sheikh, E. Simoncelli **Image Quality Assessment: From Error Visibility to Structural Similarity,** IEEE TIP, 13(4), 2004

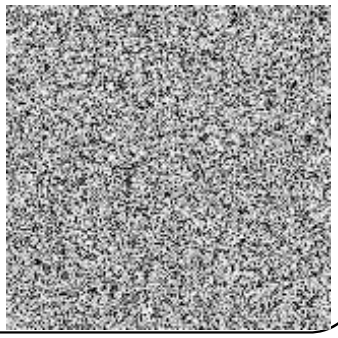
Artéfacts de bruit gaussien

Image originale

Valeur de SSIM pour chaque pixel



MSE



Artéfacts de blocs (compression JPEG)

Image originale

Valeur de SSIM pour chaque pixel

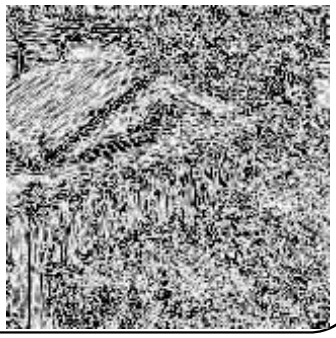
MSE

Artéfacts de compression (JPEG2000)

Image originale

Valeur de SSIM pour chaque pixel

MSE



Autre example (images dégradées ayant la même MSE)

MSE=0, MSSIM=1

MSE=225, MSSIM=0.949

MSE=225, MSSIM=0.989

MSE=225, MSSIM=0.671

MSE=225, MSSIM=0.688

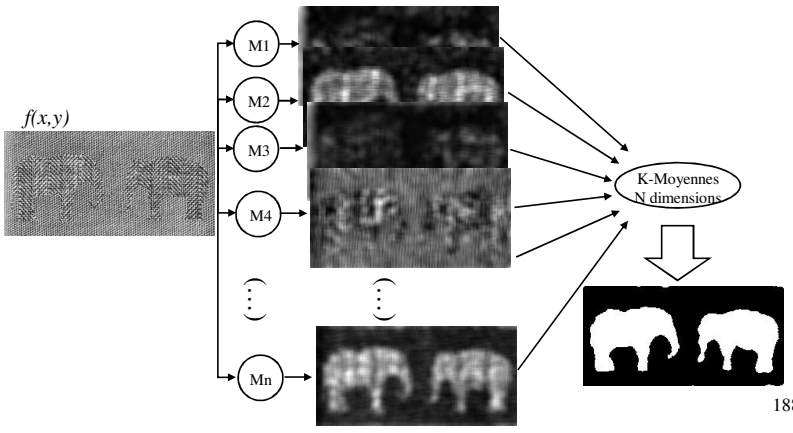
MSE=225, MSSIM=0.723

Analyse d'images avancée

187

RAPPEL : Analyse de textures

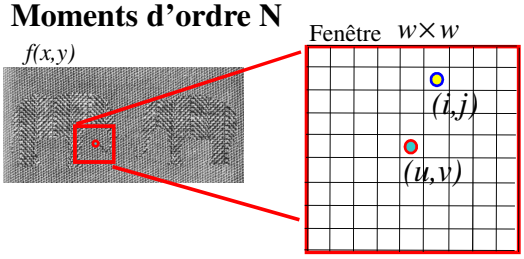
Nous avons vu jusqu'à présent **2 méthodes** pour extraire des caractéristiques de texture nous permettant ainsi de segmenter des images sur la base de textures



188

RAPPEL : Moments

Moments d'ordre N

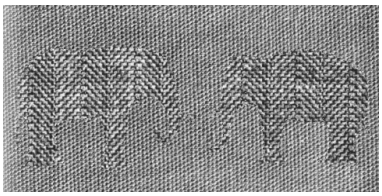


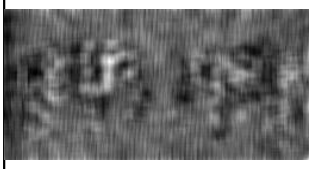
$$m_{pq}(u,v) = \sum_{i=u-w/2}^{u+w/2} \sum_{j=v-w/2}^{v+w/2} f(i,j) \cdot i_m{}^p \cdot j_m{}^q \qquad \text{où } i_m = \frac{i-u}{w},\ j_m = \frac{j-v}{w}$$

m_{pq} est un **moment d'ordre N** si **p+q=N**

189

RAPPEL : Moments



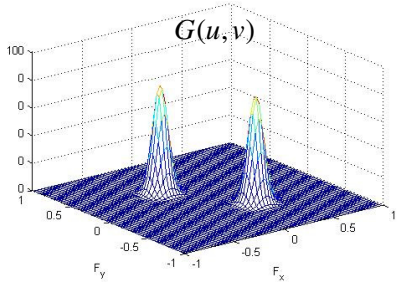


(p=1,q=1) *(p=2,q=0)* *(p=0,q=2)*

190

RAPPEL : Filtres de Gabor

Filtres de Gabor : un filtre passe bande centré sur une fréquence et avec une enveloppe gaussienne .



$$G(u,v) = Ae^{-\frac{1}{2}\left(\frac{(u-u_0)^2}{\sigma_u^2}+\frac{(v-v_0)^2}{\sigma_v^2}\right)} + Ae^{-\frac{1}{2}\left(\frac{(u+u_0)^2}{\sigma_u^2}+\frac{(v+v_0)^2}{\sigma_v^2}\right)} \quad \text{lorsque } \phi = 0$$

$$g(x,y) = e^{-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right)} \cos\left(2\pi u_0 x + 2\pi v_0 y + \phi\right)$$

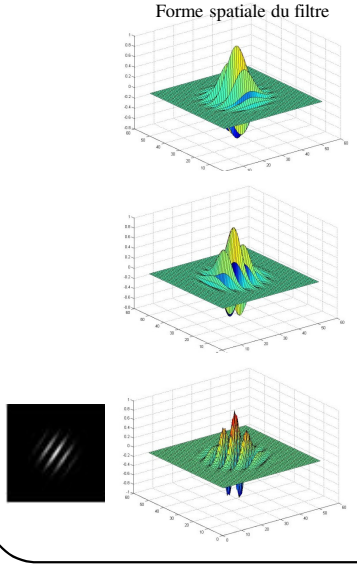
191

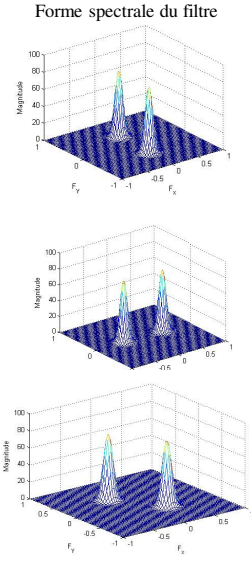
RAPPEL : Filtres de Gabor

Filtres plus souvent utilisés lors de filtrages fréquentiels

Forme spatiale du filtre

Forme spectrale du filtre





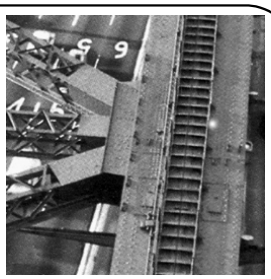
Forme spectrale =
2 gaussiennes translatées

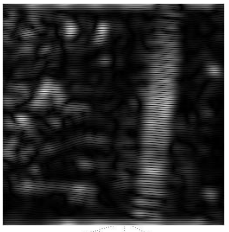
Forme spatiale =
1 gaussienne * exponentielle complexe

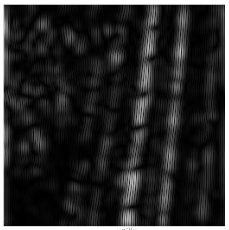
192

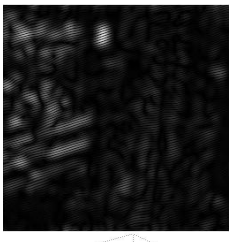
RAPPEL : Filtres de Gabor

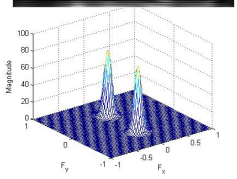
Filtres plus souvent utilisés lors de filtrages fréquentiels

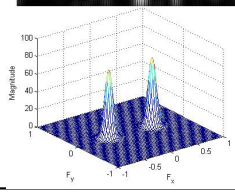


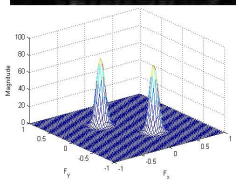












Nouveaux filtres d'analyse de texture

Steerable filters

+

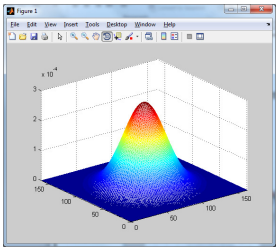
Local Binary Patterns

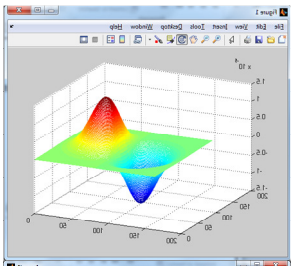
194

Steerable filters

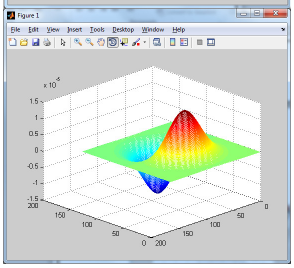
Gradient d'un filtre gaussien (dans l'espace de Fourier)

$g_\sigma(x, y)$





$\frac{\partial g_\sigma(x, y)}{\partial x}$



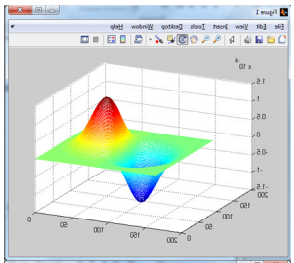
$\frac{\partial g_\sigma(x, y)}{\partial y}$

195

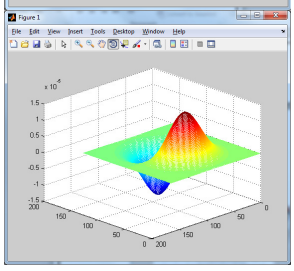
W. T. Freeman and E. H. Adelson, "The Design and Use of Steerable Filters", IEEE PAMI, 1991.

Steerable filters

Gradient **dans la direction θ** d'une gaussienne (dans l'espace de Fourier)

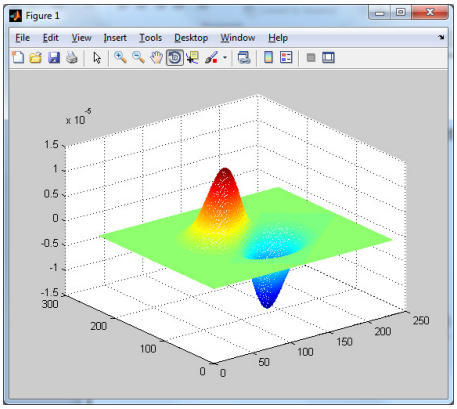


$\frac{\partial g_\sigma(x, y)}{\partial x}$



$\frac{\partial g_\sigma(x, y)}{\partial y}$

$g_\sigma^{\theta=270}$



196

W. T. Freeman and E. H. Adelson, "The Design and Use of Steerable Filters", IEEE PAMI, 1991.

Rappel

Gradient en X

$$\partial f / \partial x = f_x = \lim_{\Delta h \to 0} \frac{f(x+\Delta h, y) - f(x, y)}{\Delta h}$$

Gradient en Y

$$\partial f / \partial y = f_y = \lim_{\Delta h \to 0} \frac{f(x, y+\Delta h) - f(x, y)}{\Delta h}$$

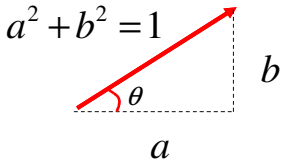
Gradient dans une direction arbitraire $\vec{v} = (a, b)$

$$f_{\vec{v}} = \lim_{\Delta h \to 0} \frac{f(x+a\Delta h, y+b\Delta h) - f(x, y)}{\Delta h}$$

197

Rappel

On peut assez facilement démontrer que si $\vec{v} = (a, b)$ est un vecteur de longueur 1



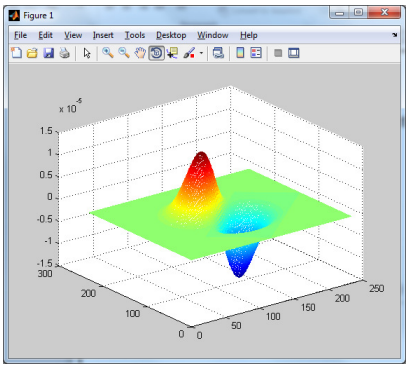
$$f_{\vec{v}} = f_\theta = \cos(\theta)\frac{\partial f}{\partial x} + \sin(\theta)\frac{\partial f}{\partial y}$$

198

Steerable filters

Gradient **dans la direction θ** d'une gaussienne (dans l'espace de Fourier)

$g_\sigma^{\theta=270}$



Filtre *Steerable* = Dérivée d'une Gaussienne
dans une direction de θ degrés

$$g_\sigma^\theta = \cos(\theta)\frac{\partial g_\sigma(x,y)}{\partial x} + \sin(\theta)\frac{\partial g_\sigma(x,y)}{\partial y}$$

199

W. T. Freeman and E. H. Adelson, "The Design and Use of Steerable Filters", IEEE PAMI, 1991.

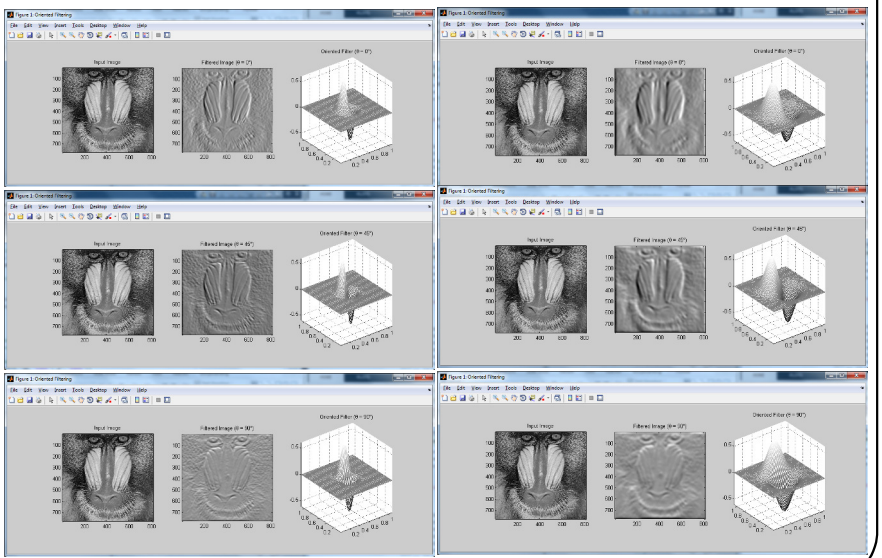
Filtres *Steerable* : 2 paramètres σ et θ

$\sigma = 3$ $\sigma = 6$

$\theta = 0$

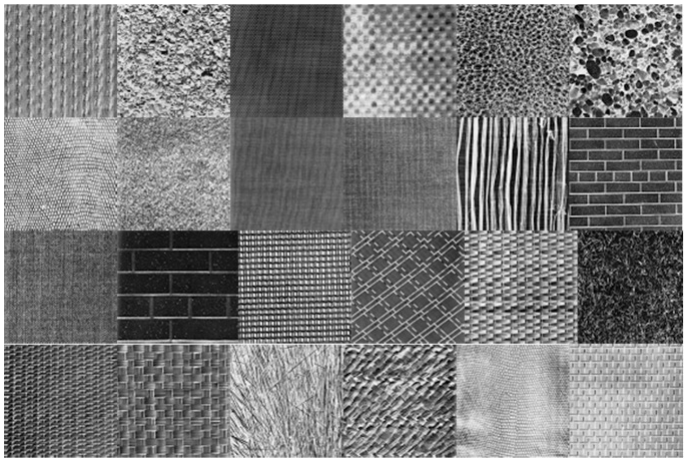
$\theta = 45$

$\theta = 90$



W. T. Freeman and E. H. Adelson, "The Design and Use of Steerable Filters", IEEE PAMI, 1991.

Filtres *Steerable* appliqués à la reconnaissance de textures



Précision de

~ 95%

Sonali Dash, Ranjan Jena "**Texture classification using Steerable Pyramid based Laws Masks**"
Journal of Electrical Systems and Information Technology, 4(1), 2017, Pages 185-197

201

Local Binary Patterns

202

Local Binary Patterns (LBP)

17	15	24	0	0	1	1	2	4	0	0	4
12	23	39	0		1	128		8	0		8
14	13	40	0	0	1	64	32	16	0	0	16
	(a)			(b)			(c)			(d)	

LBP=4+8+16=28

a) En entrée, une image jouet de 3x3 pixels

b) POUR CHAQUE pixel, retourner 0 SI intensité > 23, 1 SINON

Code binaire résultant : 00111000

c) Valeur décimale associée à la position de chaque pixel

d) Multiplication (c) * (d)

Code binaire à décimal: 0*1 + 0*2 + 1*4 + 1*8 + 1*16 + 0*32 + 0*64 + 0*128
= 4+8+16
= 28

203

Local Binary Patterns (LBP)

Il y a **256 valeurs** LBP correspondant à des **information de texture**.

17	15	24	0	0	1	1	2	4	0	0	4
12	23	39	0		1	128		8	0		8
14	13	40	0	0	1	64	32	16	0	0	16
	(a)			(b)			(c)			(d)	

LBP=4+8+16=28

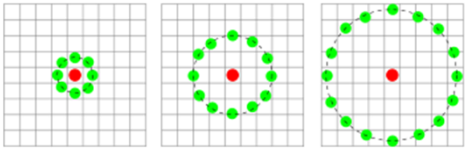
$$LBP(q) = \sum_{p=0}^{7} s(I_q - I_p)2^p$$

$$s(x) = \begin{cases} 1 & x > 0 \\ 0 & otherwise \end{cases}$$

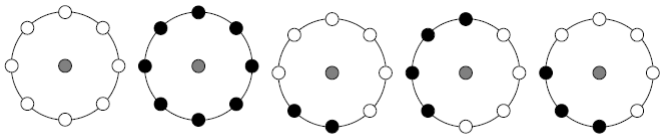
204

Local Binary Patterns (LBP)

Des voisinages de différente taille peuvent être utilisés



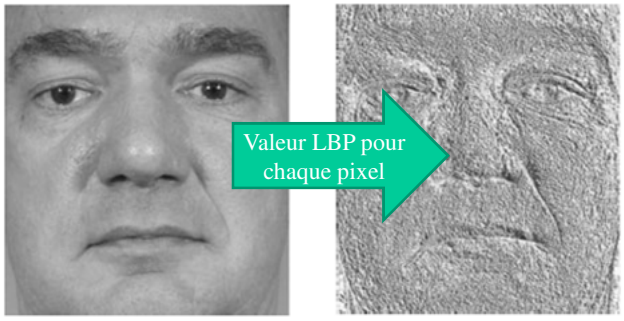
$$g_p = I(x_p, y_p), \quad p = 0, \ldots, P-1 \quad \text{and}$$
$$x_p = x + R\cos(2\pi p/P),$$
$$y_p = y - R\sin(2\pi p/P).$$



205

Local Binary Patterns (LBP)

L'algorithme des LBP

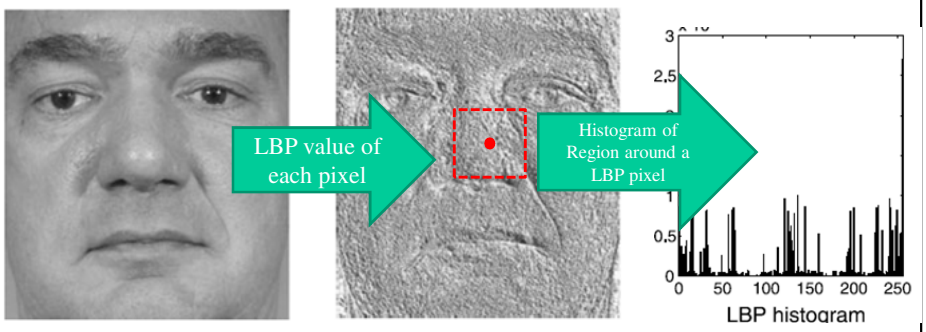


Chaque pixel est représenté par une valeur entre [0,255]

206

Local Binary Patterns (LBP)

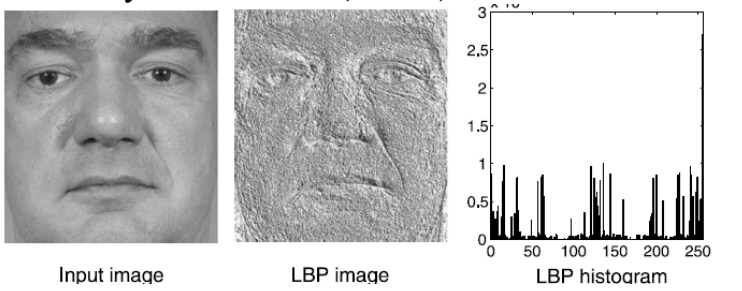
The LBP algorithm



Chaque pixel peut également être représenté par un histogramme

207

Local Binary Patterns (LBP)



[Credit Pietikainen]

Advantages :

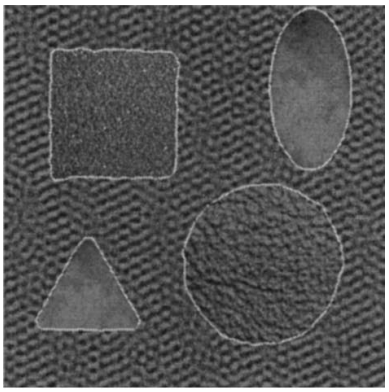
- robuste aux légers changements d'illumination.
- facile à implementer
- peu de calculs

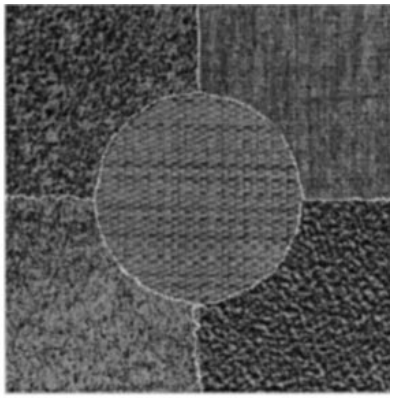
Pour la segmentation de texture

- calcule un histogramme à chaque pixel (avec une fenêtre de taille N x M)
- applique l'algorithme des k-moyennes dans un espace à 256 dimensions.

208

LBP segmentation





[Credit : Ojala- Pietikainen]

209

Les faits saillants…

1. Théorème de la convolution	$* \underset{\Im^{-1}}{\overset{\Im}{\rightleftarrows}} \times$ et $\times \underset{\Im^{-1}}{\overset{\Im}{\rightleftarrows}} *$
2. Filtres passe-bas	
3. Filtres passe-haut	
4. Filtres passe-bande	
5. Filtre bilatéral	
6. Diffusion non linéaire	$\frac{\partial u}{\partial t} = div(g(\lvert\nabla u\rvert)\nabla u)$
7. Mean Shift	$P'(x) = \frac{1}{Nh}\sum_{i=0}^{N-1} K(\frac{x - x_i}{h})$
8. Déconvolution	
9. Bruit et métrique de qualité	$PSNR\,(f, g)$