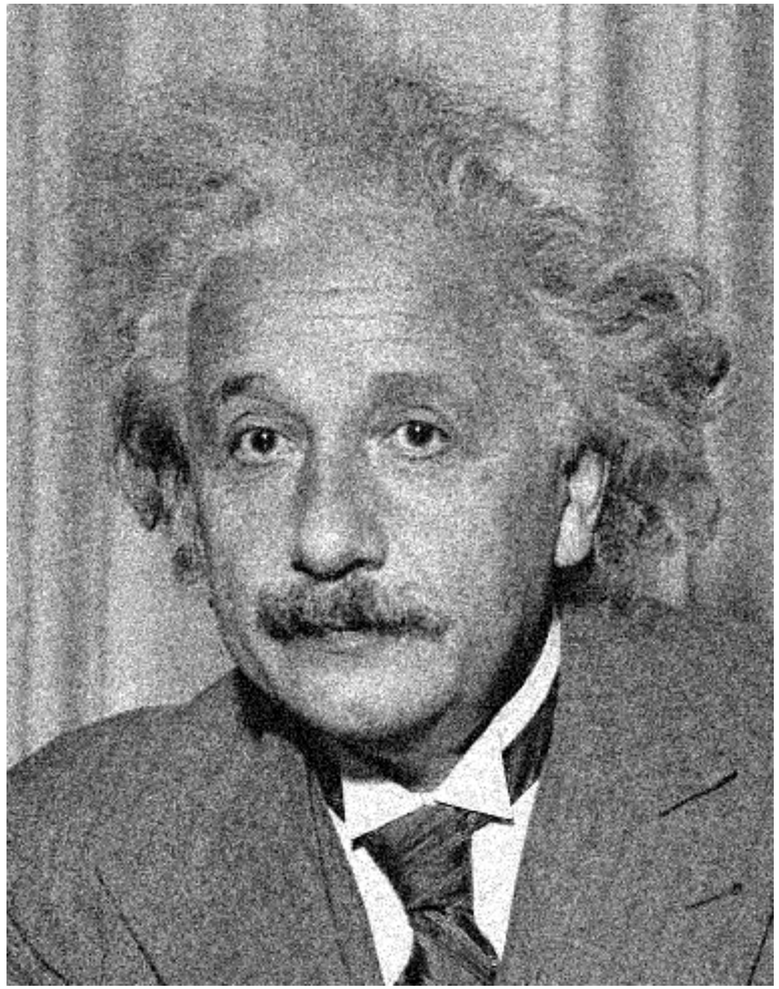
Applications du filtrage

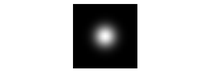
GIF-4105/7105 Photographie Algorithmique
Jean-François Lalonde

Photo Credit: Alister Benn

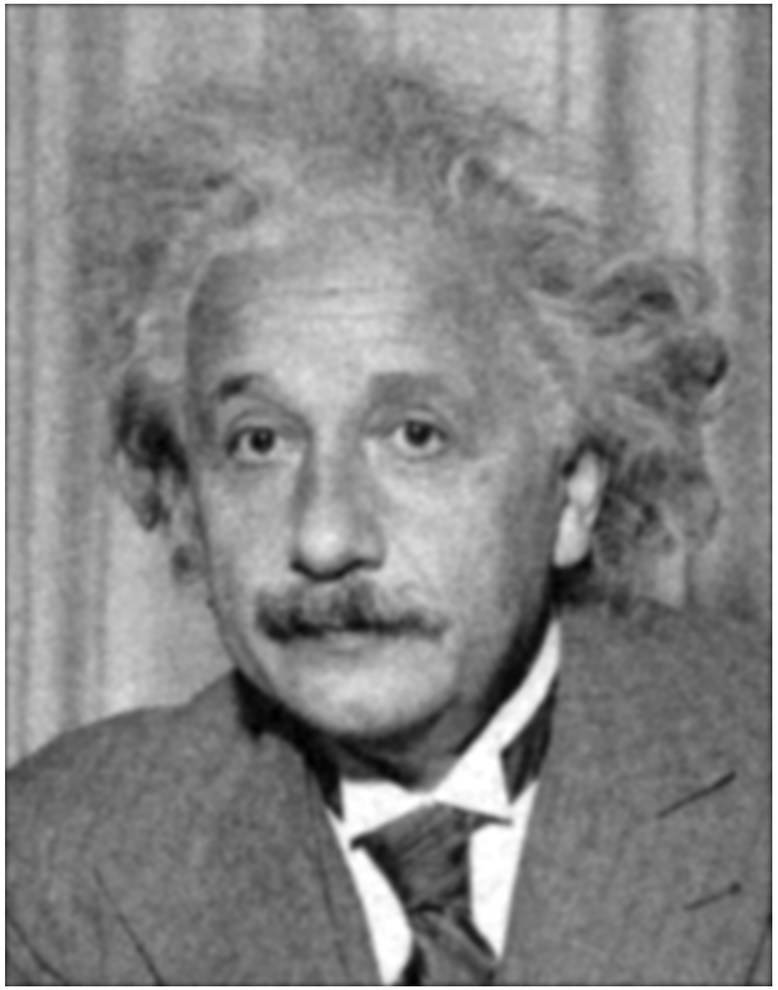
Atténuation du bruit



Bruit additif gaussien



Filtre gaussien

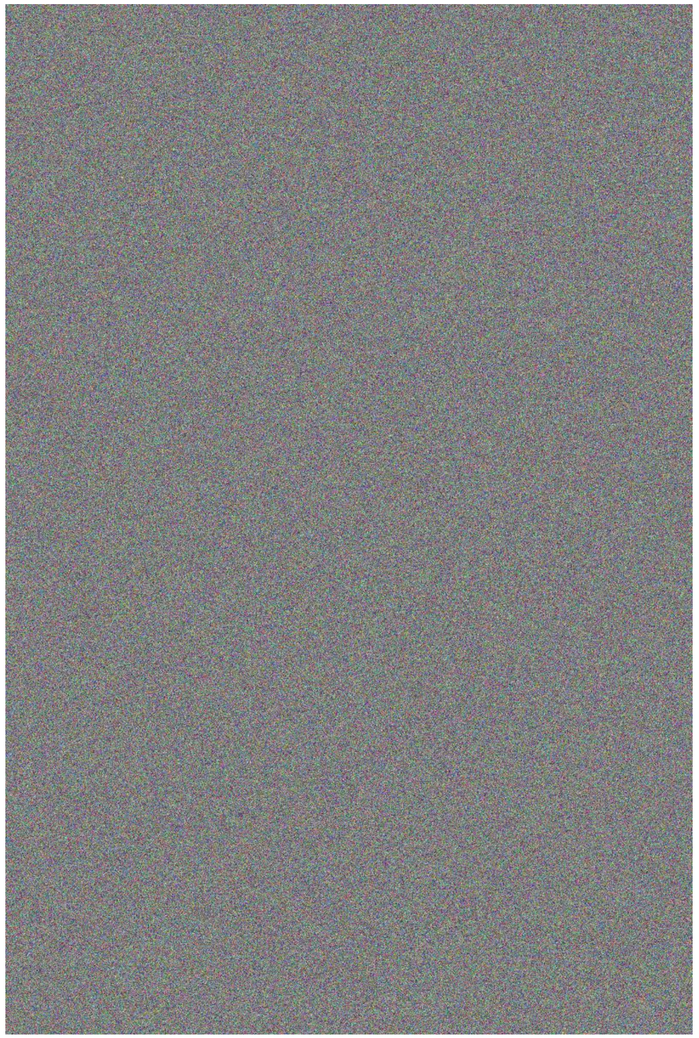


Atténuer le bruit gaussien

Image

+

Bruit gaussien (centré à 0)



=

Image bruitée

Atténuer le bruit gaussien

Image bruitée

Filtre gaussien
($\sigma = 0.5$)

Atténuer le bruit gaussien

Image bruitée

Filtre gaussien
(σ = 1)

Atténuer le bruit gaussien

Image bruitée

Filtre gaussien
(σ = 2)

Atténuer le bruit gaussien

Image bruitée

Filtre gaussien
($\sigma = 5$)

Atténuer le bruit gaussien

Image bruitée

Filtre gaussien
($\sigma = 5$)

En augmentant la variance du filtre, on réduit le bruit, mais on rend l'image floue!

Bruit « poivre et sel »

Filtre gaussien

3x3

Bruit « poivre et sel »

Filtre gaussien

3x3

5x5

Bruit « poivre et sel »

Filtre gaussien

3x3

5x5

7x7

Autre idée : filtre médian

10	15	20
23	90	27
33	31	30

Autre idée : filtre médian

Est-ce que c'est linéaire?

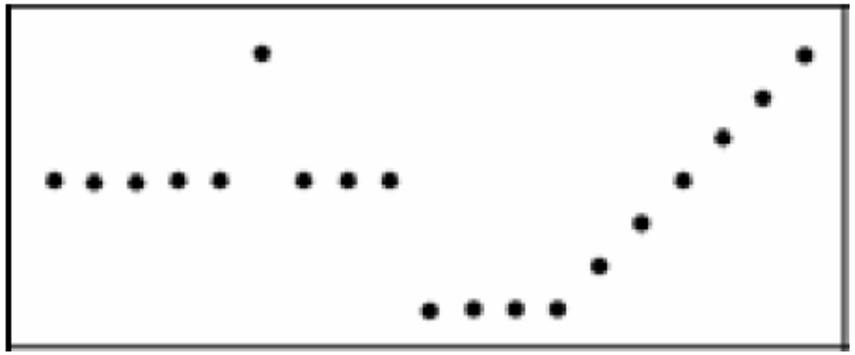
10	15	20
23	90	27
33	31	30

Ordonner 10 15 20 23 27 30 31 33 90

Prendre la médiane

Filtre médian

Quels sont les avantages du filtre médian par rapport au filtre gaussien?



Signal

Source: K. Grauman

Filtre médian

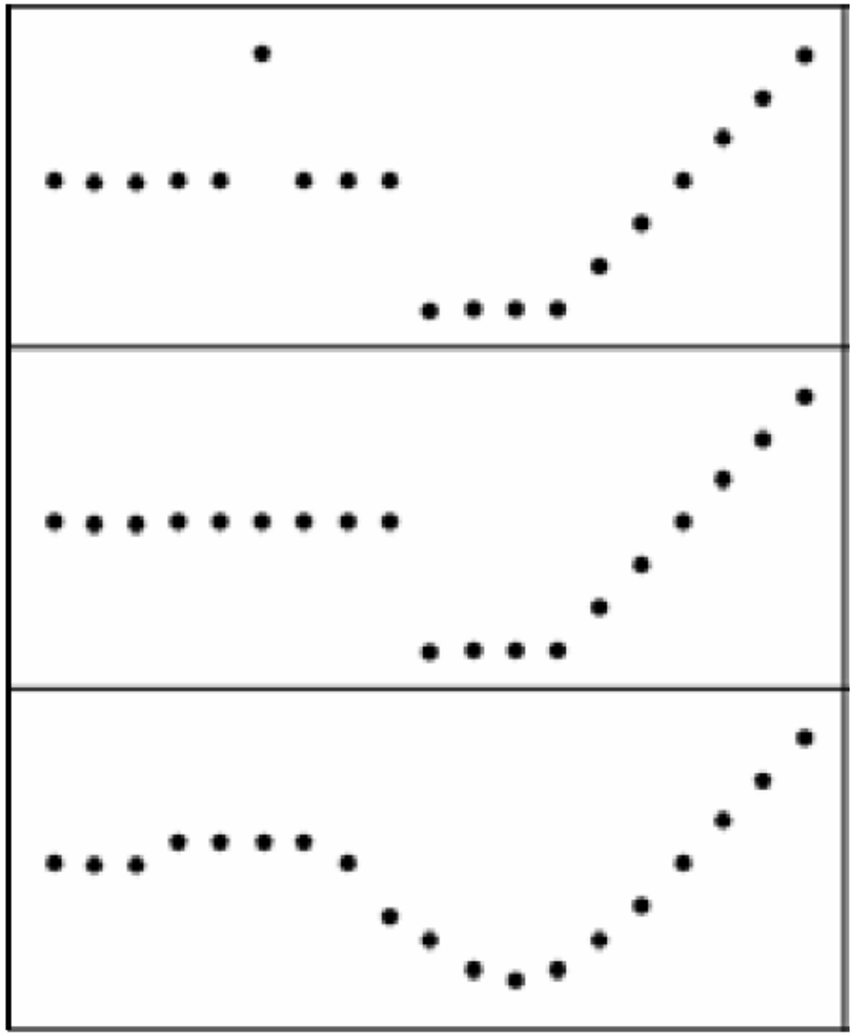
Quels sont les avantages du filtre médian par rapport au filtre gaussien?

Source: K. Grauman

Filtre médian

Quels sont les avantages du filtre médian par rapport au filtre gaussien?

- Robuste aux points aberrants
- Préserve mieux les arêtes



Source: K. Grauman

Filtre médian vs. gaussien

3x3

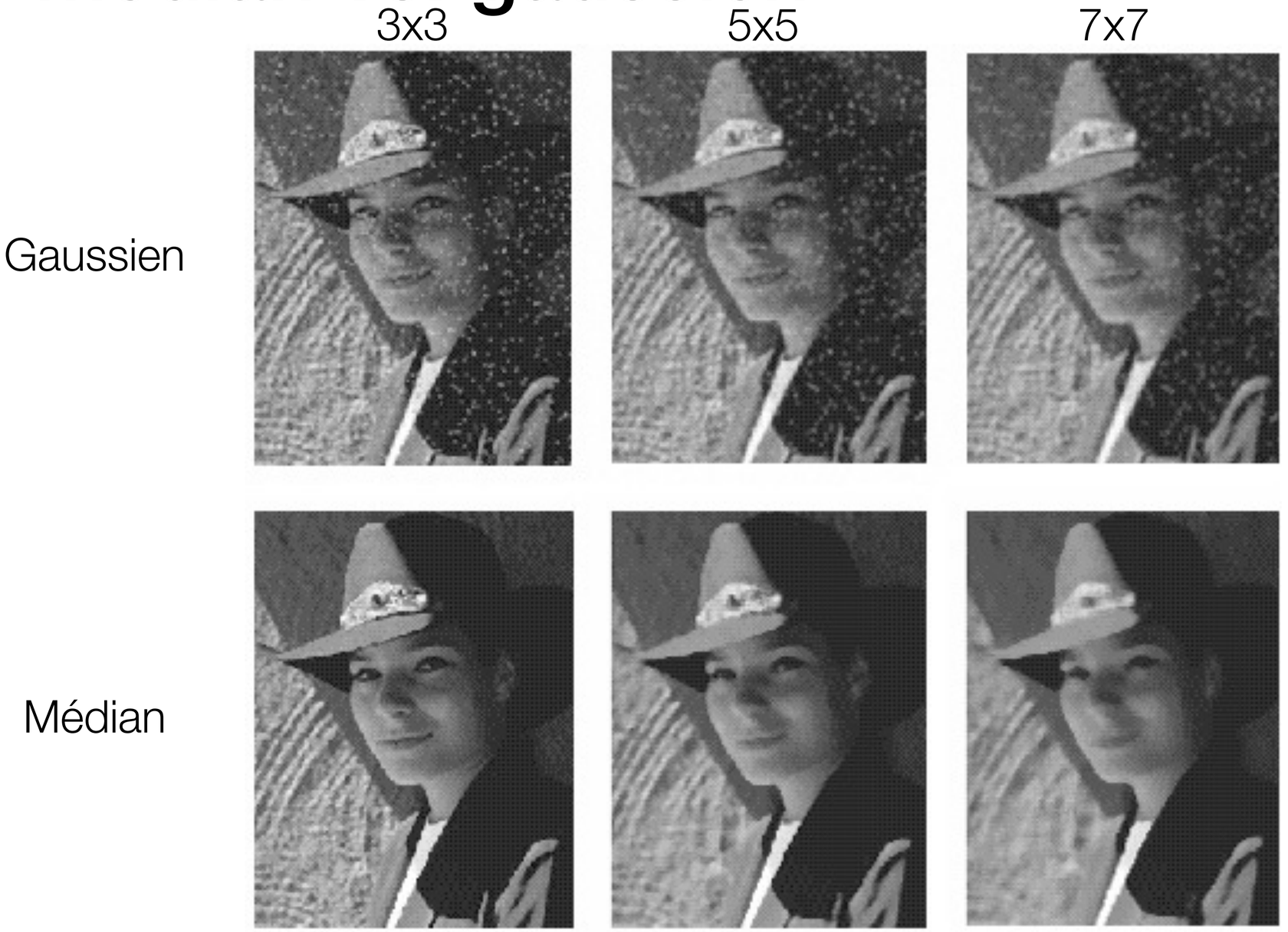
Gaussien

Médian

Filtre médian vs. gaussien

	3x3	5x5
Gaussien		
Médian		

Filtre médian vs. gaussien



En python

```
from skimage.filters.rank import median
import numpy as np

imgf = median(img, np.ones((23, 23)))
```

Attention
Ne fonctionne que sur des images à 1 canal (niveau de gris)

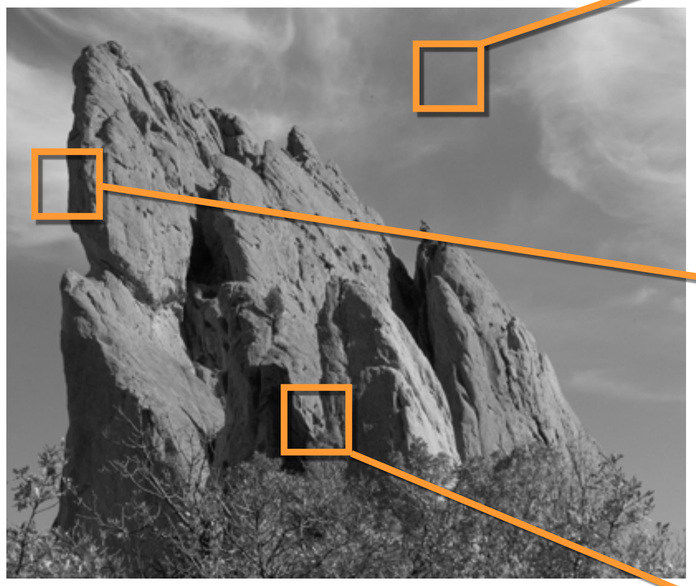
Image originale

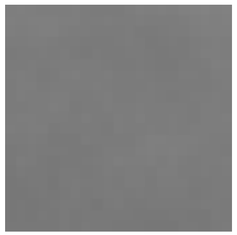
Filtre médian (23x23)

D’où provient le flou?

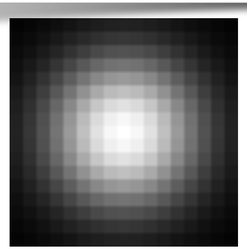
Le flou est créé lorsqu’on calcule la moyenne de chaque côté d’une arête.

entrée

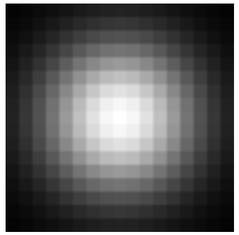




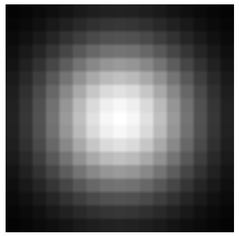
*



*



*



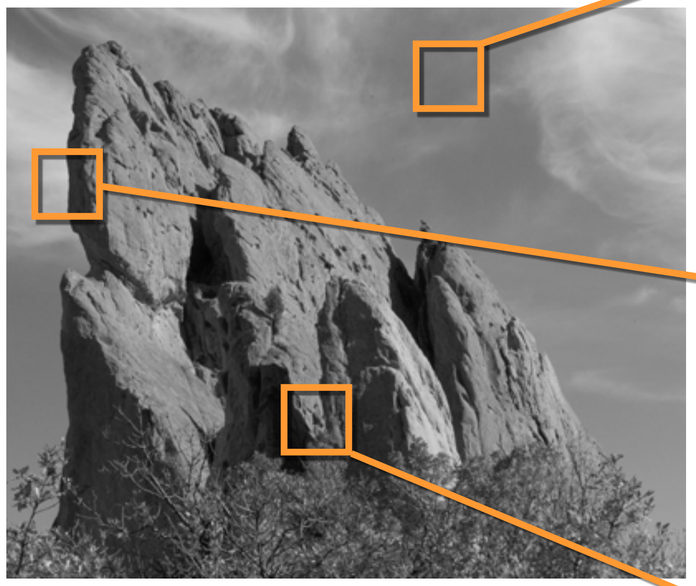
sortie

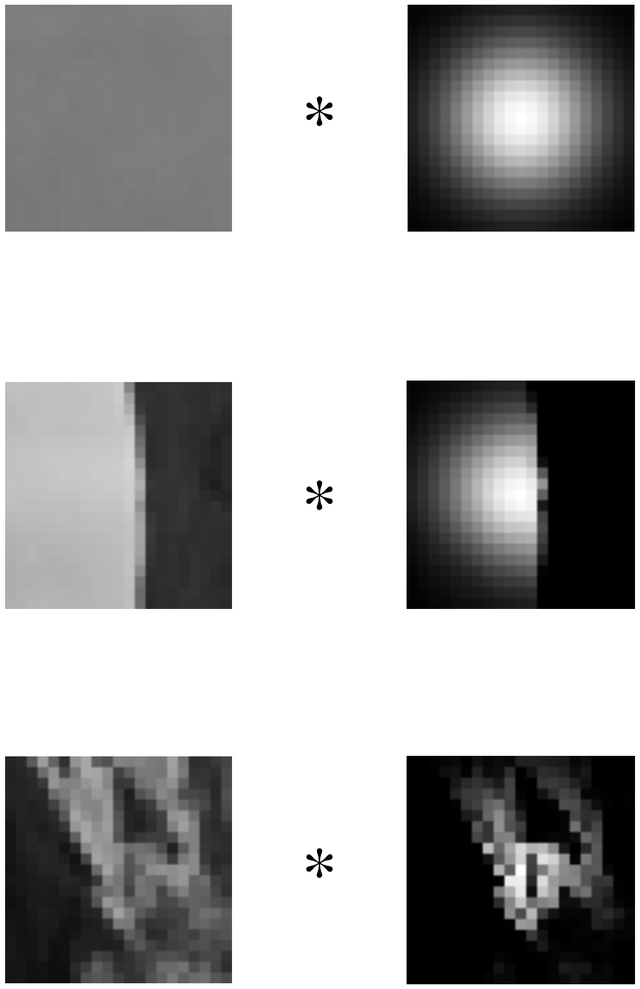
Le **même** noyau gaussien est appliqué partout

Filtre bilatéral : pas de moyenne de part et d'autre d'une arête

[Aurich 95, Smith 97, Tomasi 98]

entrée





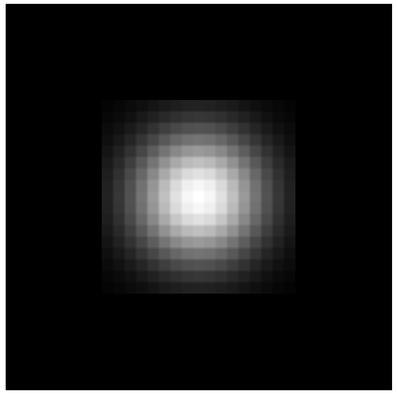
sortie

Le noyau gaussien est adapté en fonction du contenu de l'image

Filtre gaussien

$$F[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\sigma_s}(||\mathbf{p} - \mathbf{q}||)$$

pondération spatiale



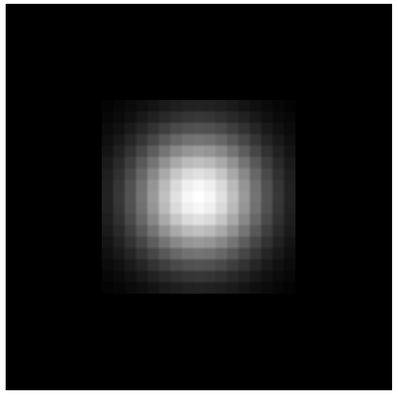
Filtre bilatéral : deux filtres gaussiens!

mais pas sur le même domaine!

$$F[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(||\mathbf{p} - \mathbf{q}||) G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) I_{\mathbf{q}}$$

normalisation

pondération spatiale



pondération en intensité

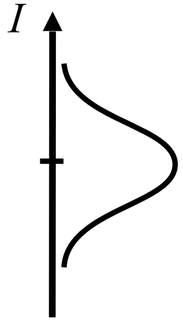


Illustration : image 1-D

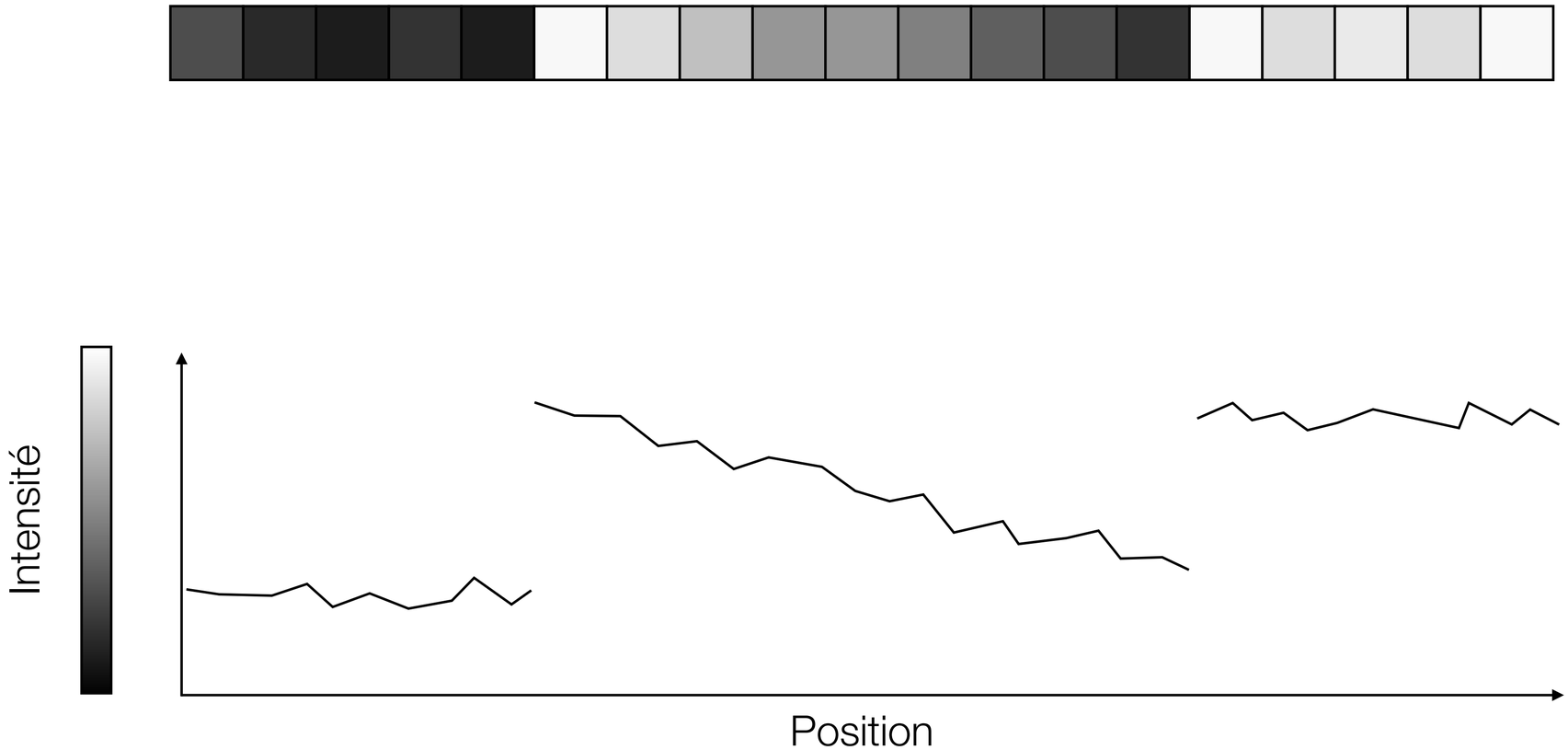
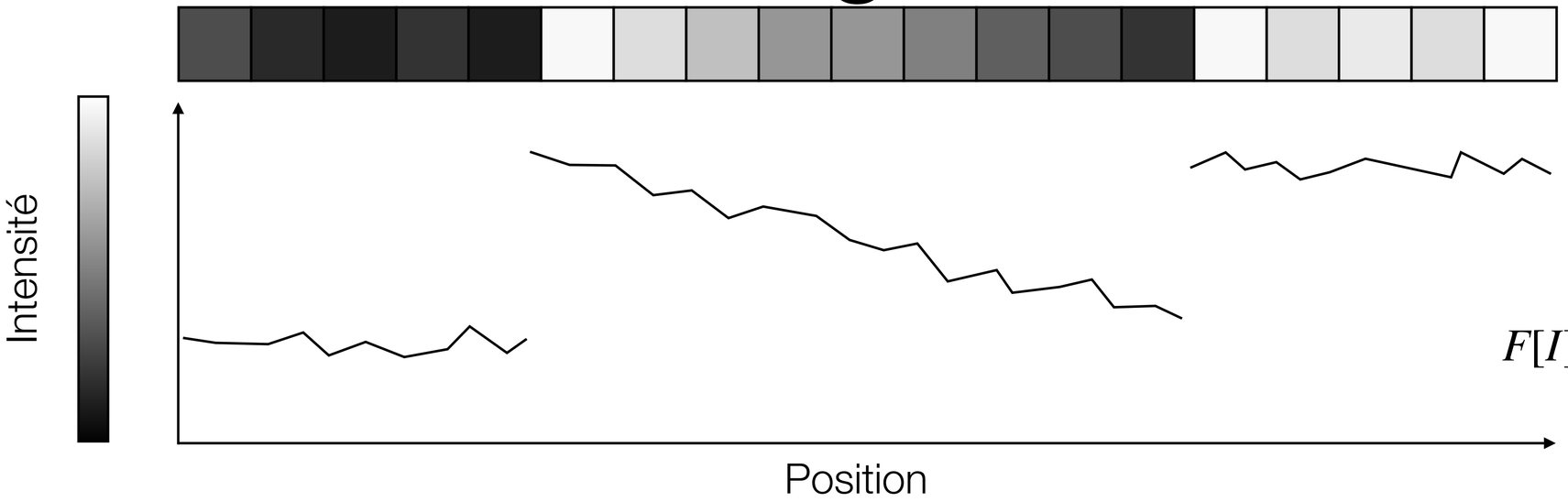


Illustration : image 1-D

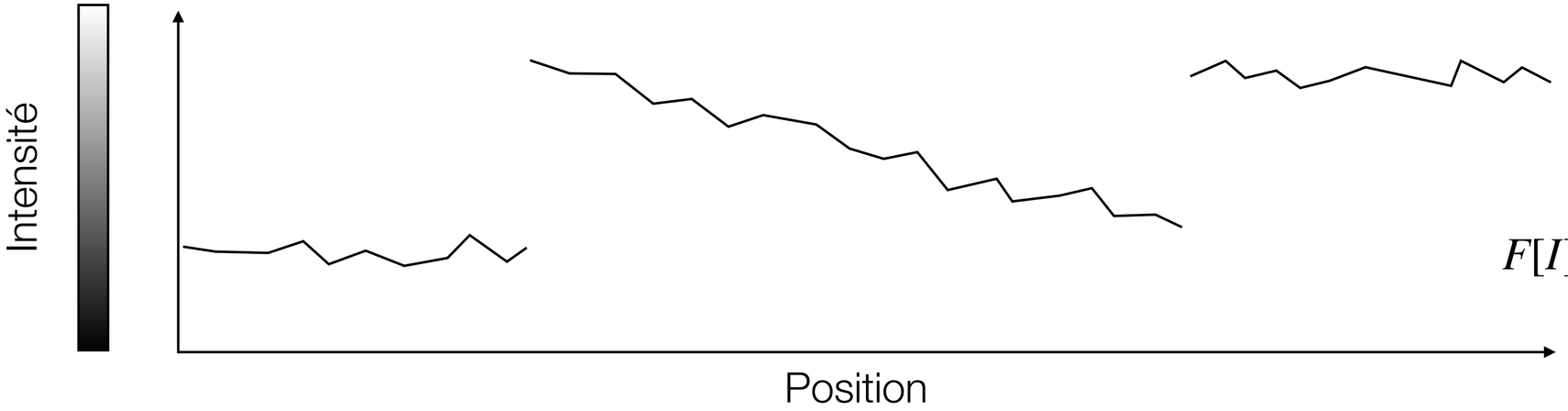


$$F[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\sigma_s}(||\mathbf{p} - \mathbf{q}||)$$

Espace

Illustration : image 1-D

Est-ce que c'est linéaire?

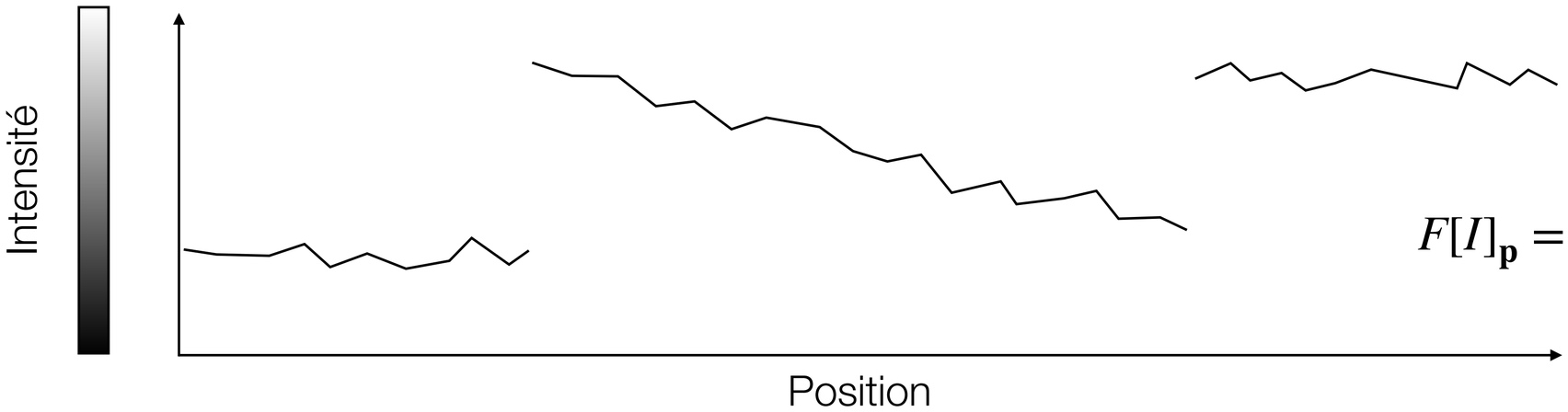


$$F[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\sigma_s}(||\mathbf{p} - \mathbf{q}||)$$

Espace

Filtre gaussien

Filtre bilatéral



$$F[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(||\mathbf{p} - \mathbf{q}||) G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) I_{\mathbf{q}}$$

Espace Intensité

$\sigma_r = 0.1$

$\sigma_r = 0.25$

$\sigma_r = \infty$
(filtre gaussien)

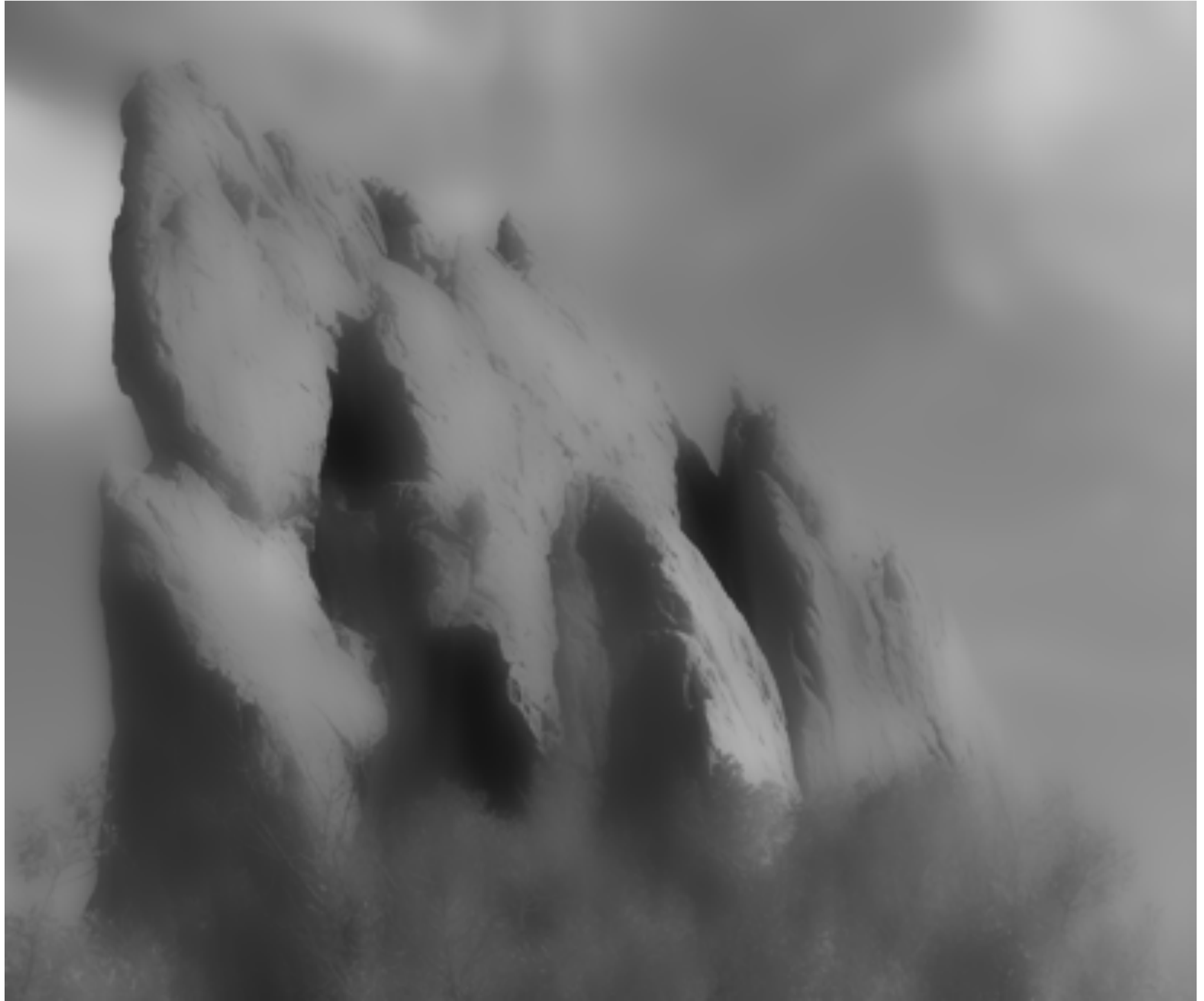
$\sigma_s = 2$

$\sigma_s = 6$

$\sigma_s = 18$

image originale

$\sigma_r = 0.1$



$\sigma_r = 0.25$

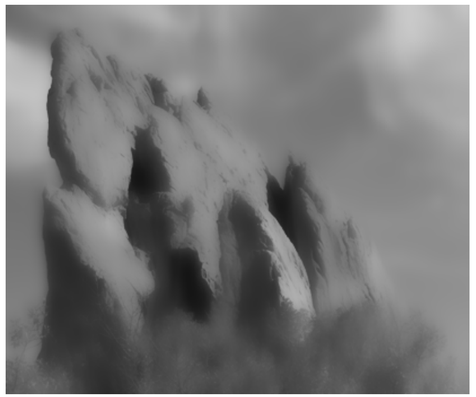
$\sigma_r = \infty$
(filtre gaussien)

$\sigma_r = 0.1$ | $\sigma_r = 0.25$ | $\sigma_r = \infty$ (filtre gaussien)

$\sigma_s = 2$



$\sigma_s = 6$


$\sigma_s = 18$

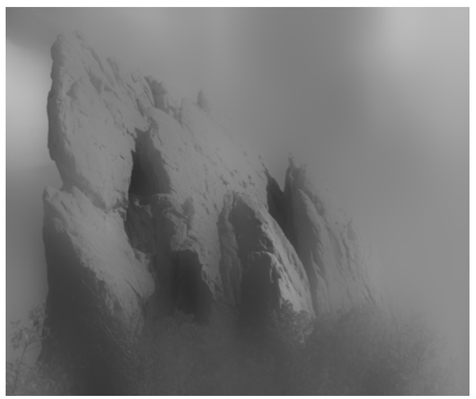




image originale

$\sigma_s = 2$

$\sigma_s = 6$

$\sigma_s = 18$

En python

```
from skimage.restoration import denoise_bilateral
imgf = denoise_bilateral(img, \
   sigma_spatial=5, \
   sigma_color=0.05, \
   multichannel=True)
```

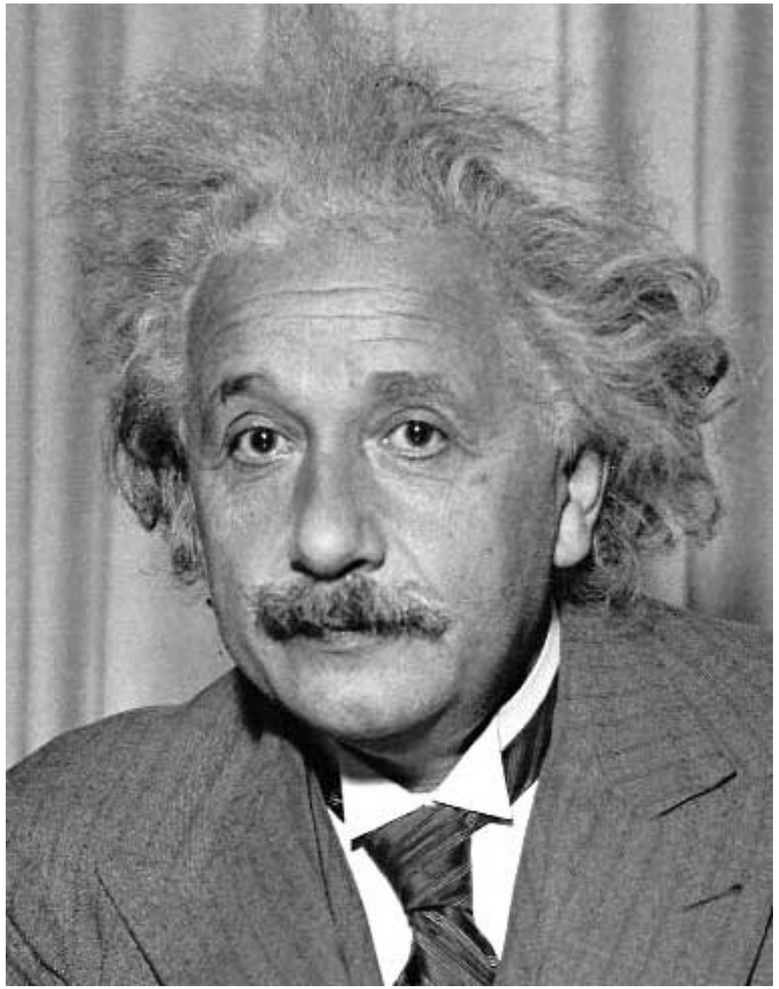
Attention
Ceci peut être long (surtout si sigma_spatial est gros)

Image originale

Filtre bilatéral (sigma_s=5, sigma_r=0.05)

Correspondance de modèles (*template matching*)

- But : trouver  dans l'image
- Défi : Comment devrait-on comparer le modèle avec l'image?

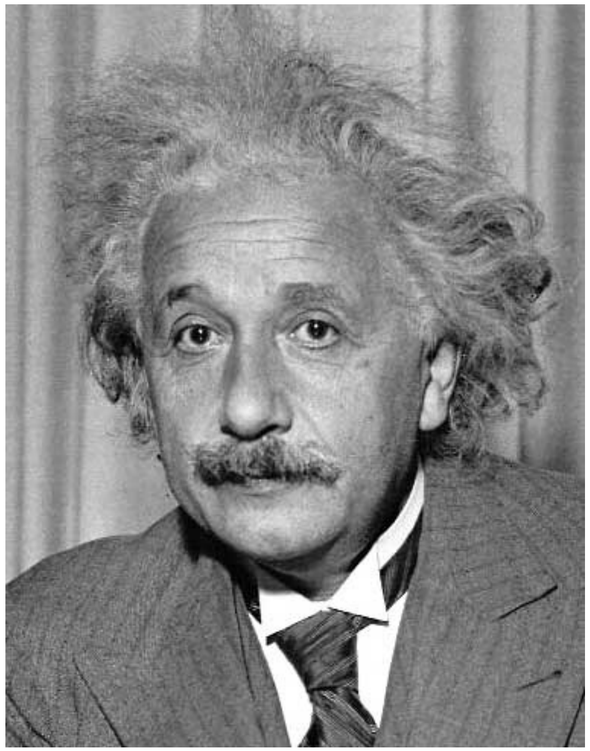


Filtrer pour trouver les correspondances

- But : trouver  dans l'image
- Méthode 0 : filtrer l'image avec l'oeil

$$h(m,n) = \sum_{k,l} g(k,l) f(m+k, n+l)$$

f = image
g = filtre (oeil)



Image

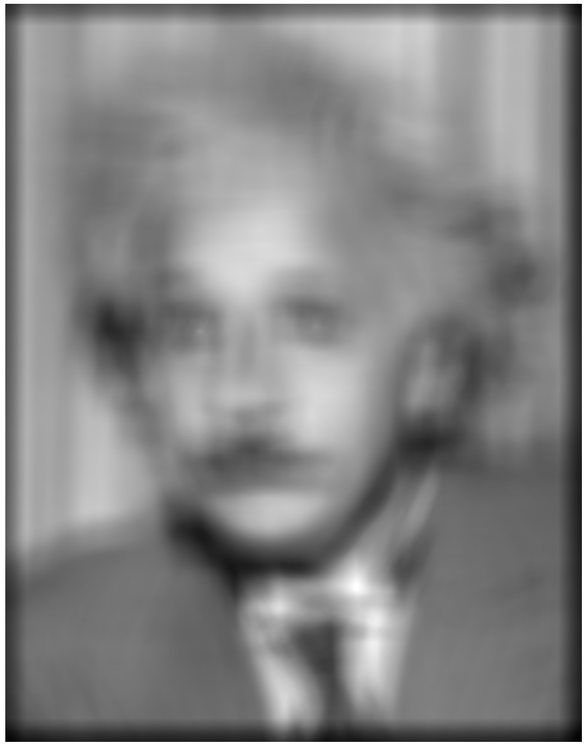


Image filtrée

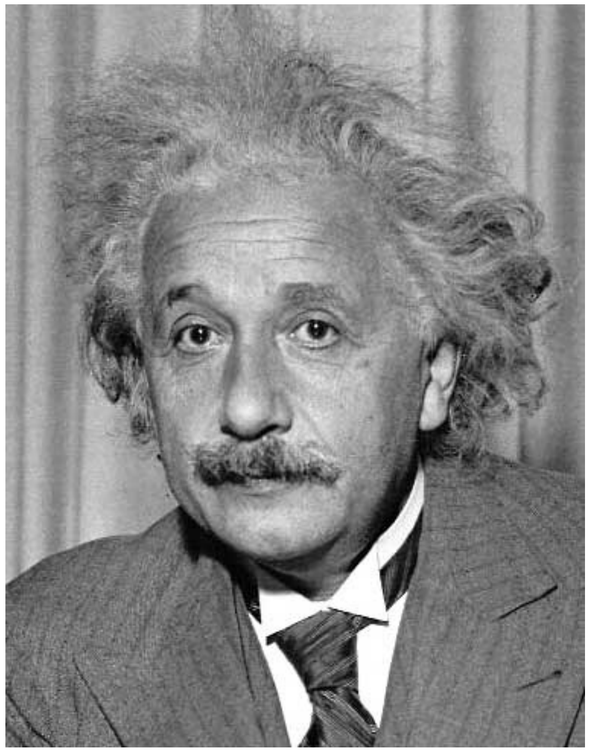
Qu'est-ce qui se passe?

Filtrer pour trouver les correspondances

- But : trouver  dans l'image
- Méthode 1 : normaliser le filtre par sa moyenne

$$h(m,n) = \sum_{k,l}(g(k,l) - \bar{g})f(m+k,n+l)$$

moyenne de g



Image

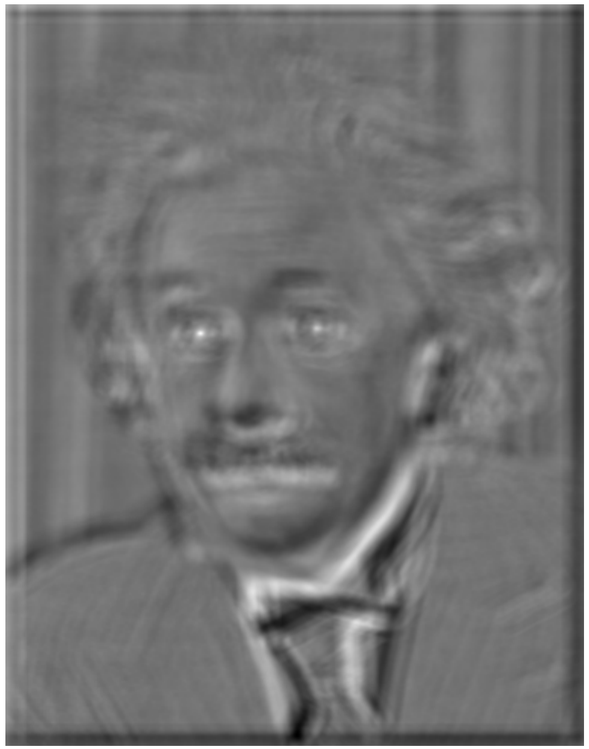
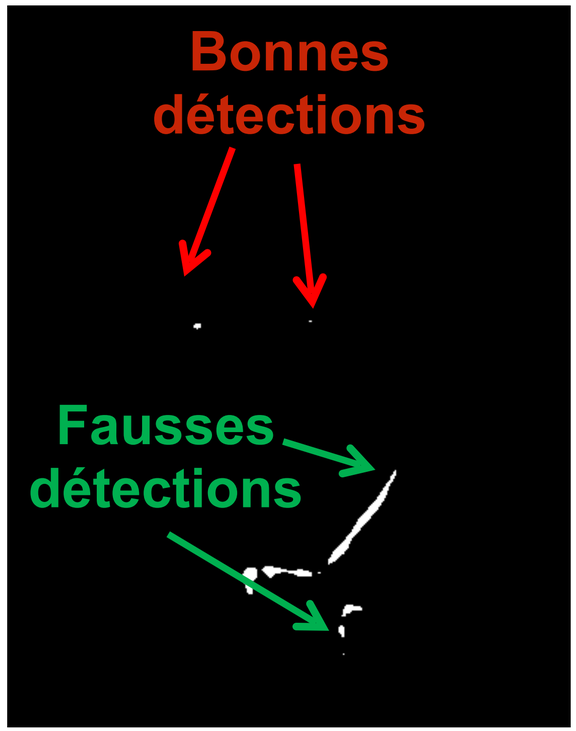


Image filtrée



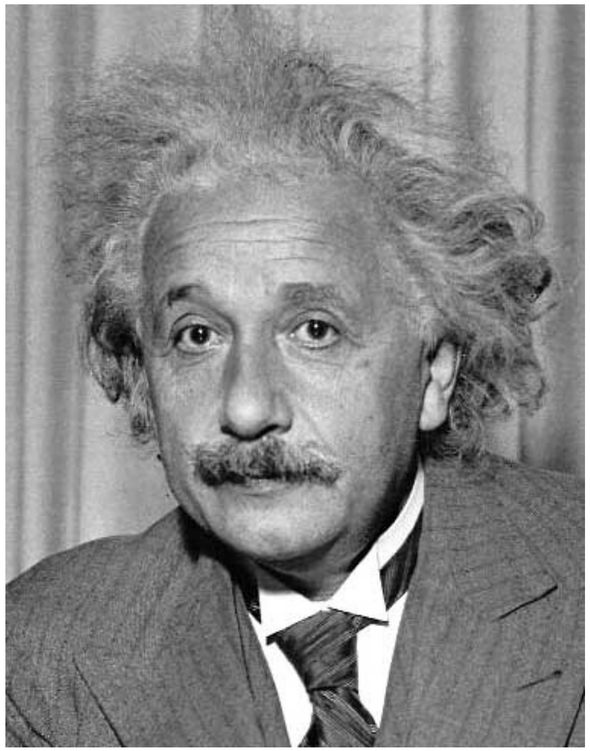
Seuil

Filtrer pour trouver les correspondances

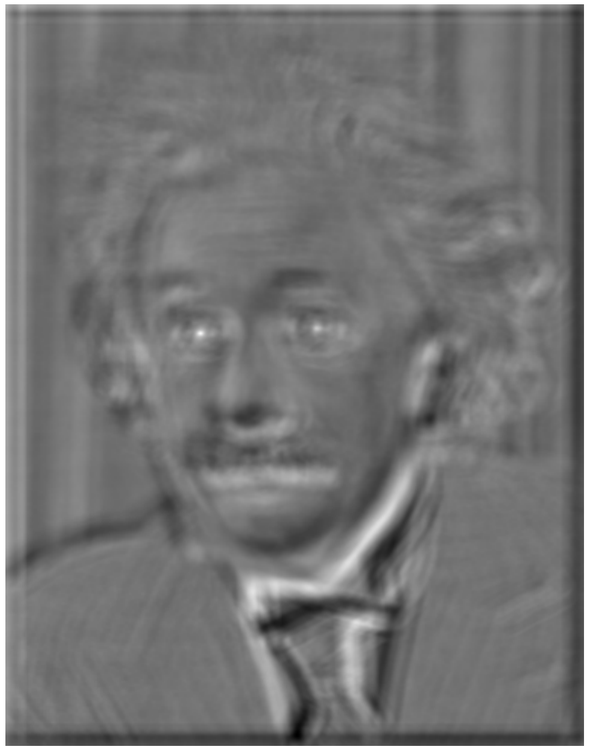
- But : trouver  dans l'image

$$h(m,n) = \sum_{k,l} (g(k,l) - f(m+k, n+l))^2$$

- Méthode 2 : somme des différences au carré (SDC)



Image



1-SDC

Seuil

Derek Hoiem

Filtrer pour trouver les correspondances

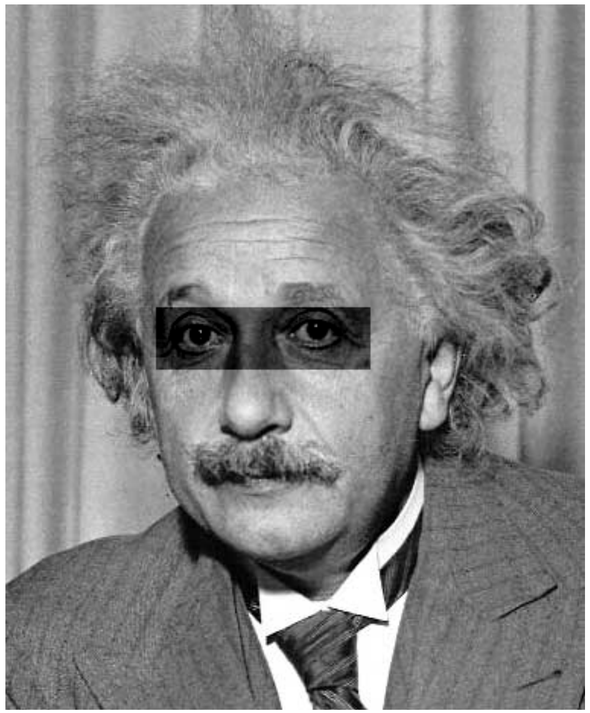
Est-ce qu'on peut implémenter la somme des différences
au carré avec un (ou des) filtre(s) linéaire(s)?

$$h(m,n) = \sum_{k,l} (g(k,l) - f(m+k, n+l))^2$$

Filtrer pour trouver les correspondances

- But : trouver  dans l'image
- Méthode 2 : somme des différences au carré

$$h(m,n) = \sum_{k,l} (g(k,l) - f(m+k, n+l))^2$$



Image

1-SSD

Problème?

Derek Hoiem

Filtrer pour trouver les correspondances

- But : trouver  dans l'image
- Méthode 3 : corrélation croisée normalisée

moyenne du filtre

moyenne de la
partie correspondante dans l'image

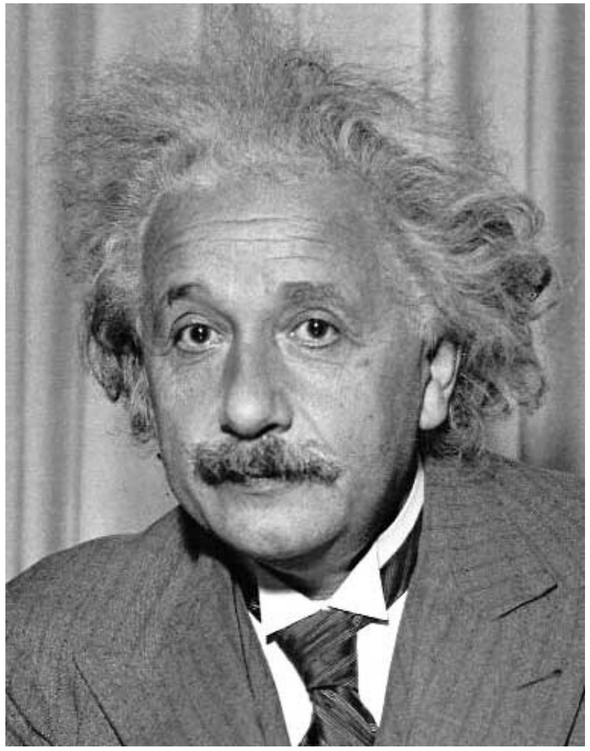
$$h(m,n) = \frac{\sum_{k,l}(g(k,l) - \bar{g})(f(m+k,n+l) - \bar{f}_{m,n})}{\sqrt{\sum_{k,l}(g(k,l) - \bar{g})^2 \sum_{k,l}(f(m+k,n+l) - \bar{f}_{m,n})^2}}$$

En python (nécessite la version 0.18 de scikit-image) :

```
from skimage.registration import phase_cross_correlation
shift = skimage.registration.phase_cross_correlation(img, filtre)
```

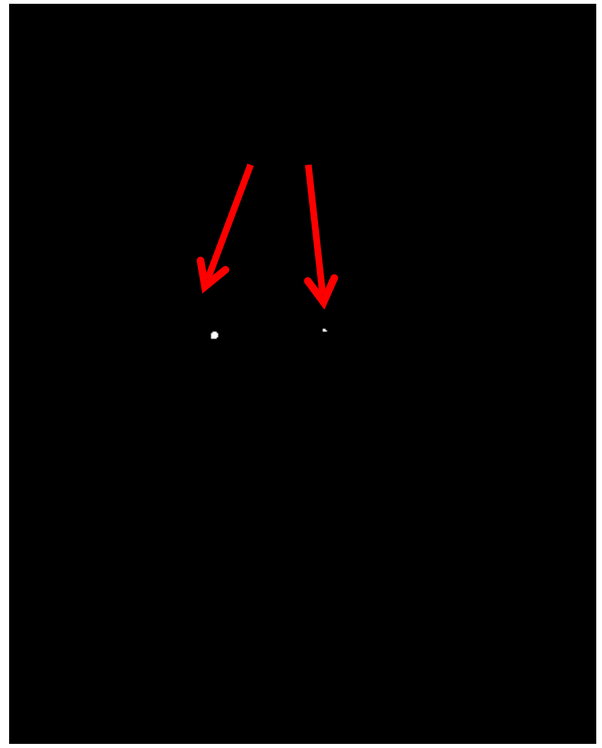
Filtrer pour trouver les correspondances

- But : trouver  dans l'image
- Méthode 3 : corrélation croisée normalisée



image

résultat

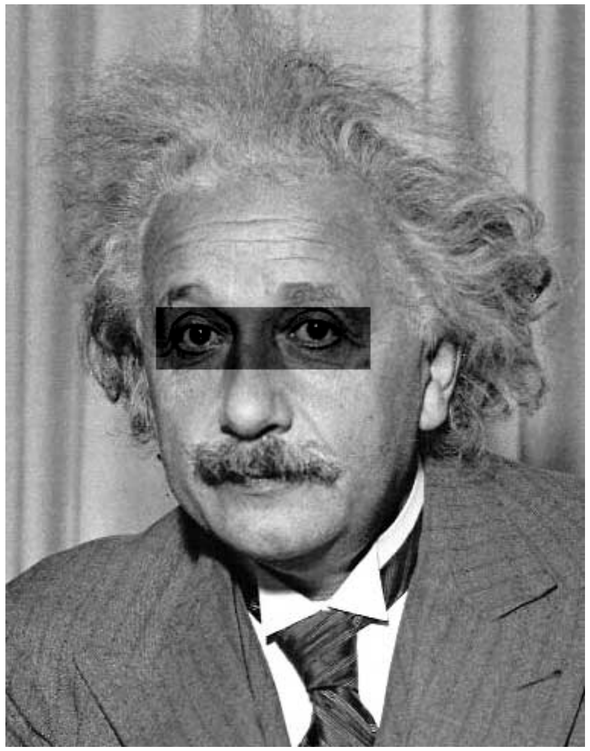


seuil

Derek Hoiem

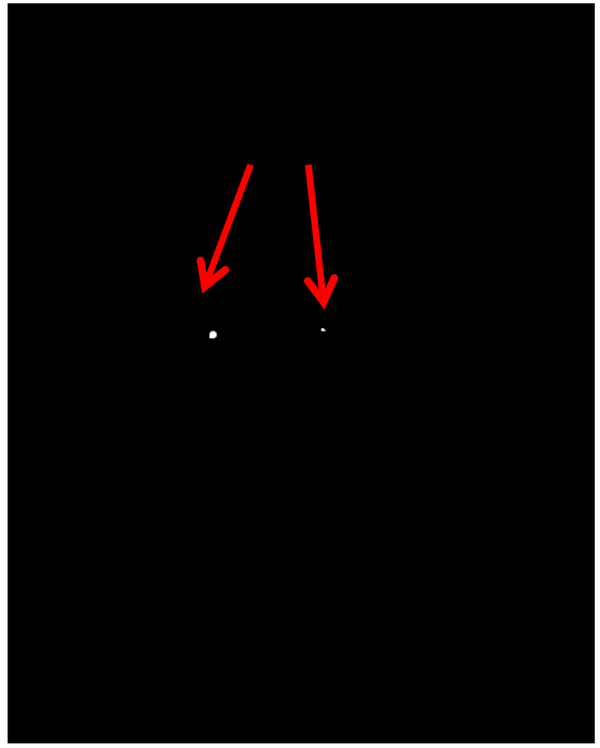
Filtrer pour trouver les correspondances

- But : trouver  dans l'image
- Méthode 3 : corrélation croisée normalisée



image

résultat



seuil

Derek Hoiem

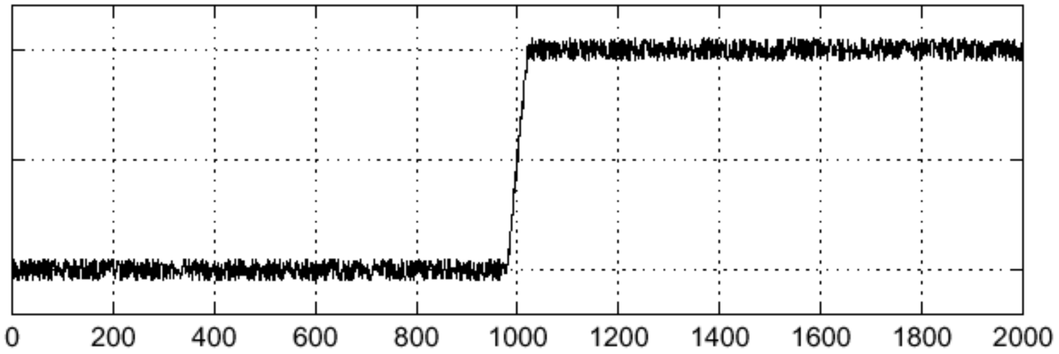
Quelle est la meilleure méthode?

- Ça dépend!
- Filtre normalisé
  - plus rapide, mais pas très bon
- Somme des différences au carré
  - un peu moins rapide, sensible aux variations d’intensité
- Corrélation croisée-normalisée
  - plus lente, mais robuste aux variations d’intensité

Détecter une arête

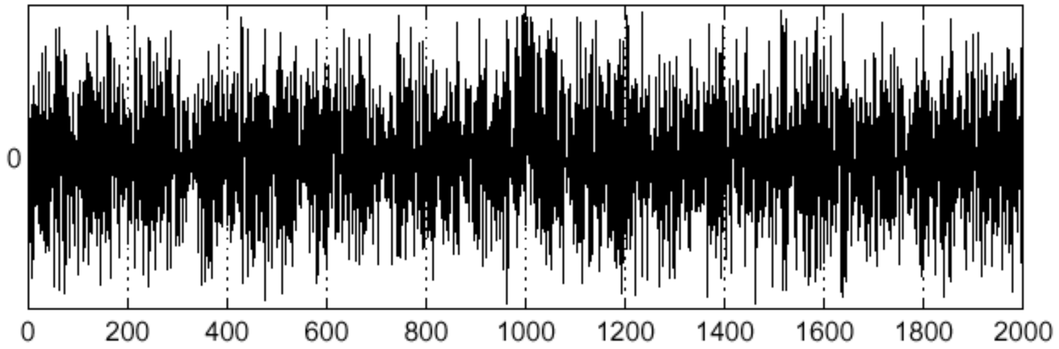
- Analysons une seule ligne dans l'image
  - Affiche l'intensité en fonction de la coordonnée en x

$f(x)$



Comment calculer le gradient (la dérivée?)

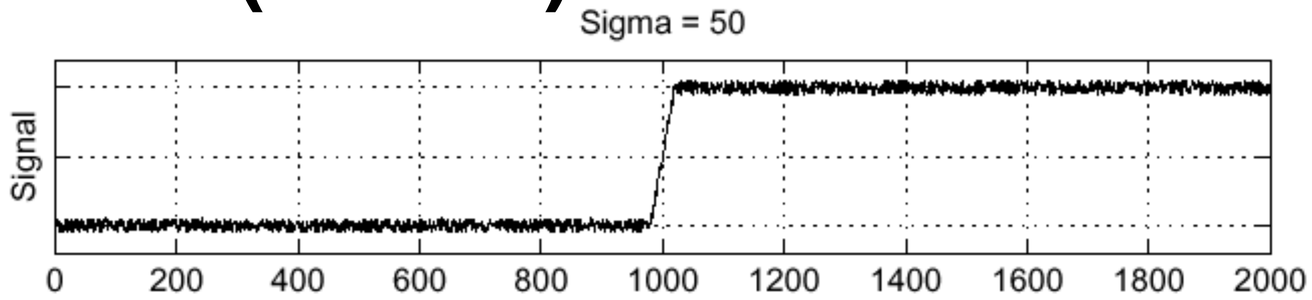
$\frac{d}{dx}f(x)$



Où est l'arête?

Solution: adoucir! (filtrer!)

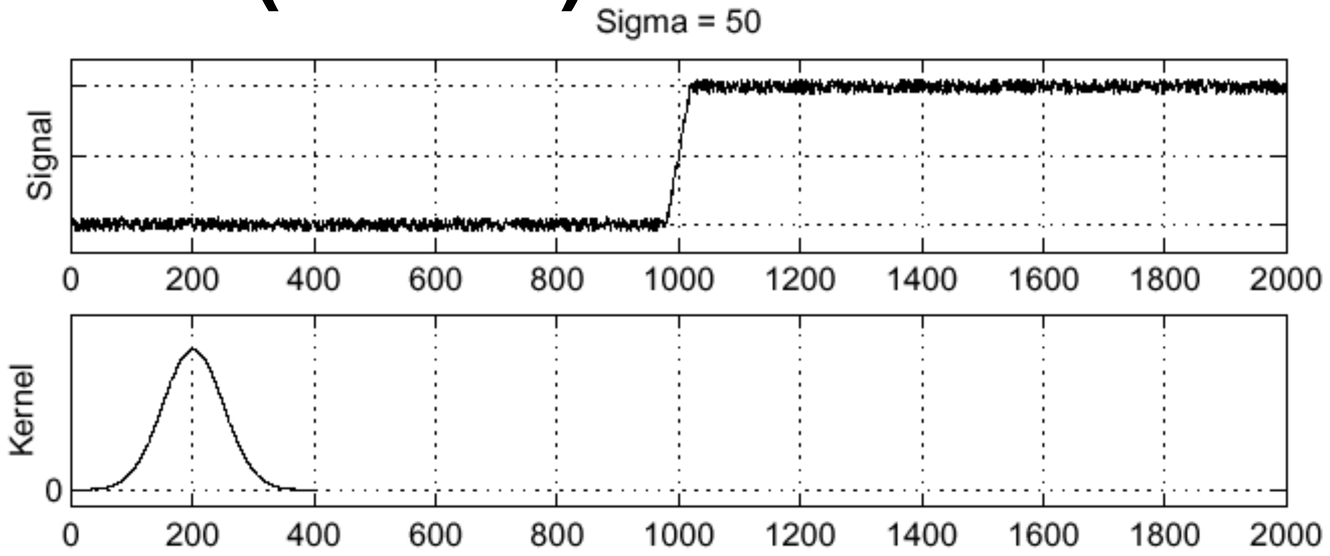
f



Solution: adoucir! (filtrer!)

f

h

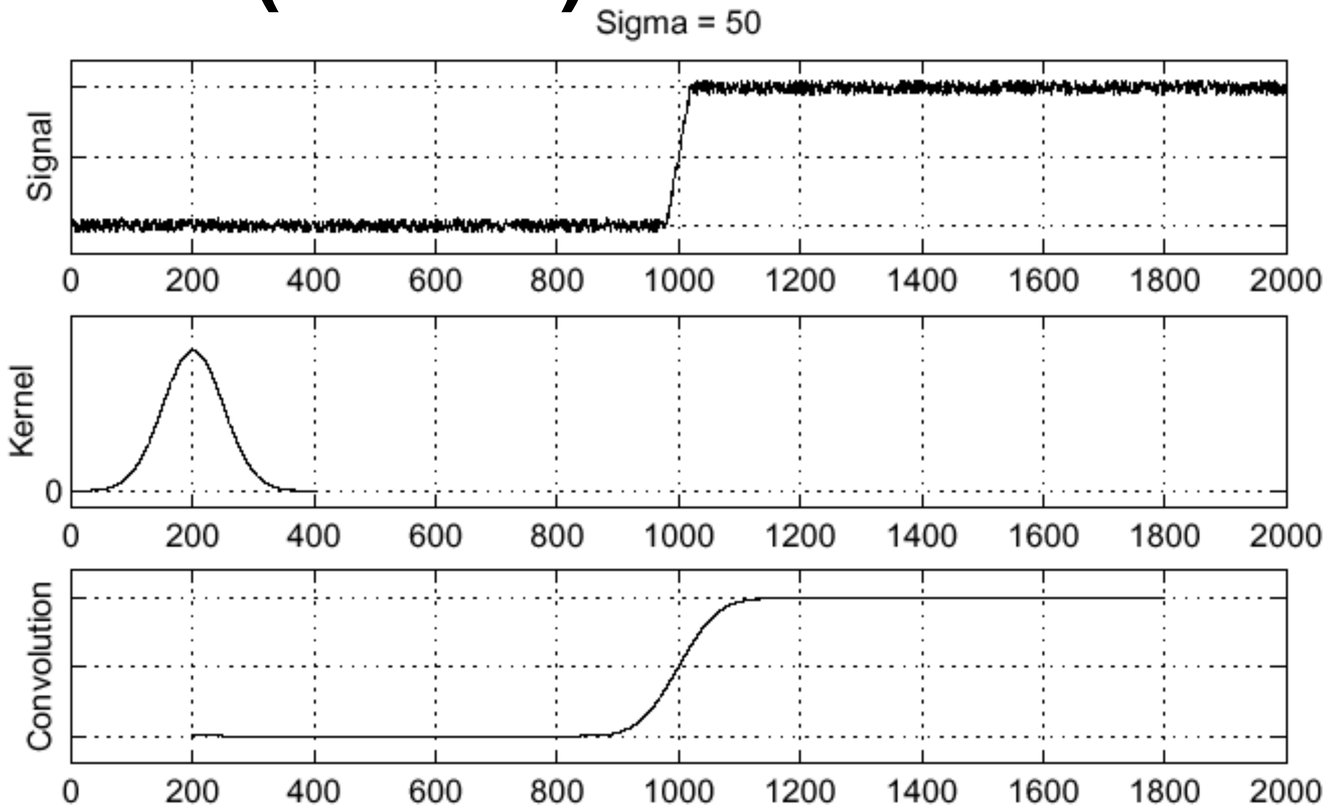


Solution: adoucir! (filtrer!)

f

h

$h \star f$



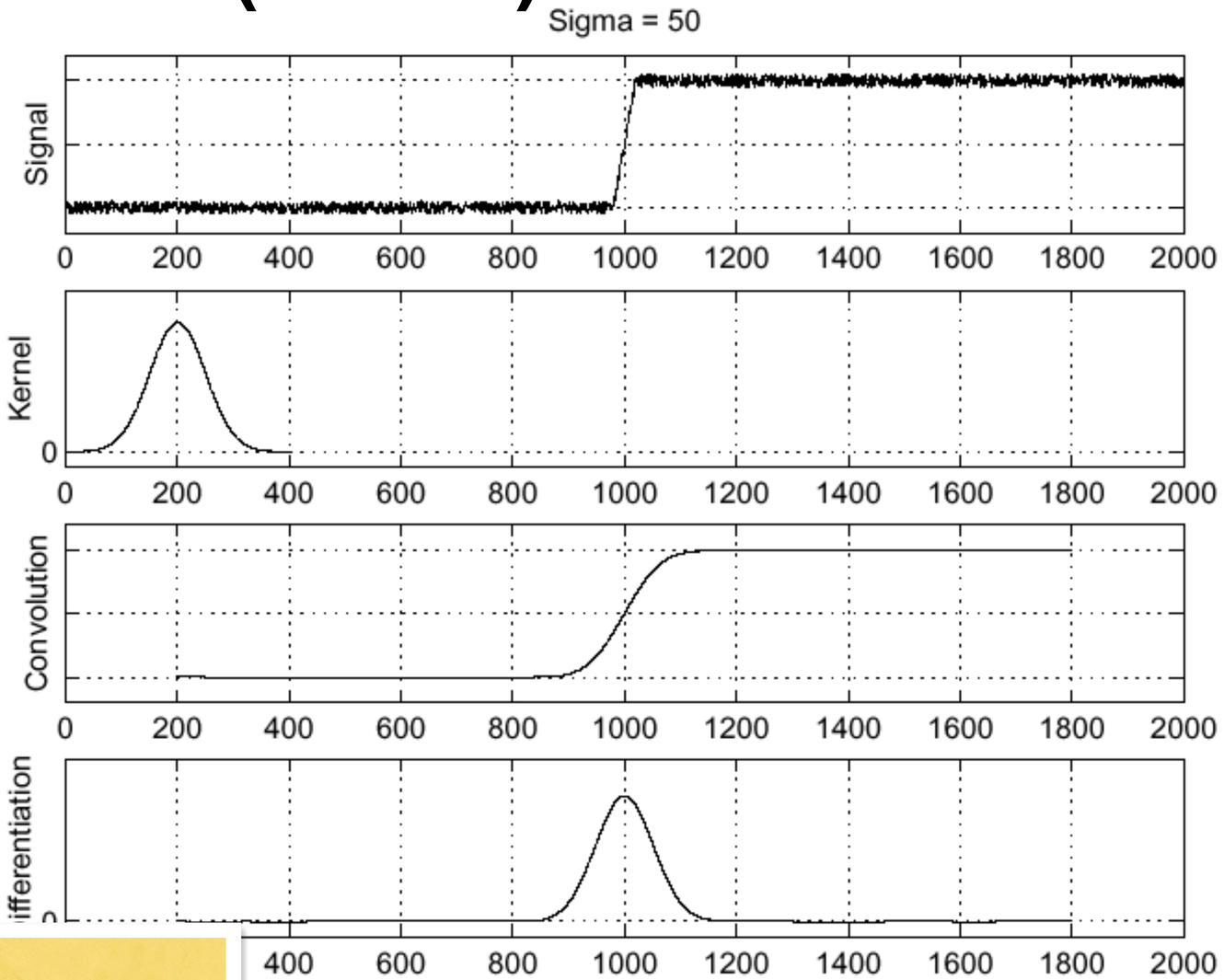
Solution: adoucir! (filtrer!)

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$



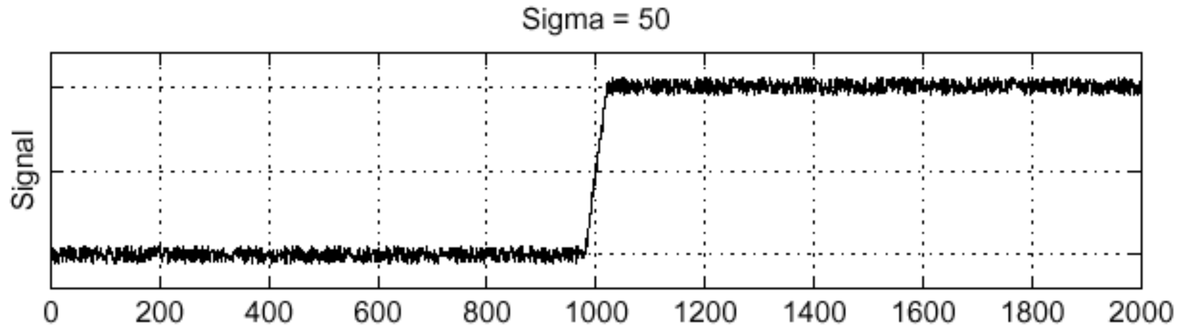
Où est l'arête?

Chercher maximums: $\frac{\partial}{\partial x}(h \star f)$

Théorème sur la dérivée de la convolution

$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

f

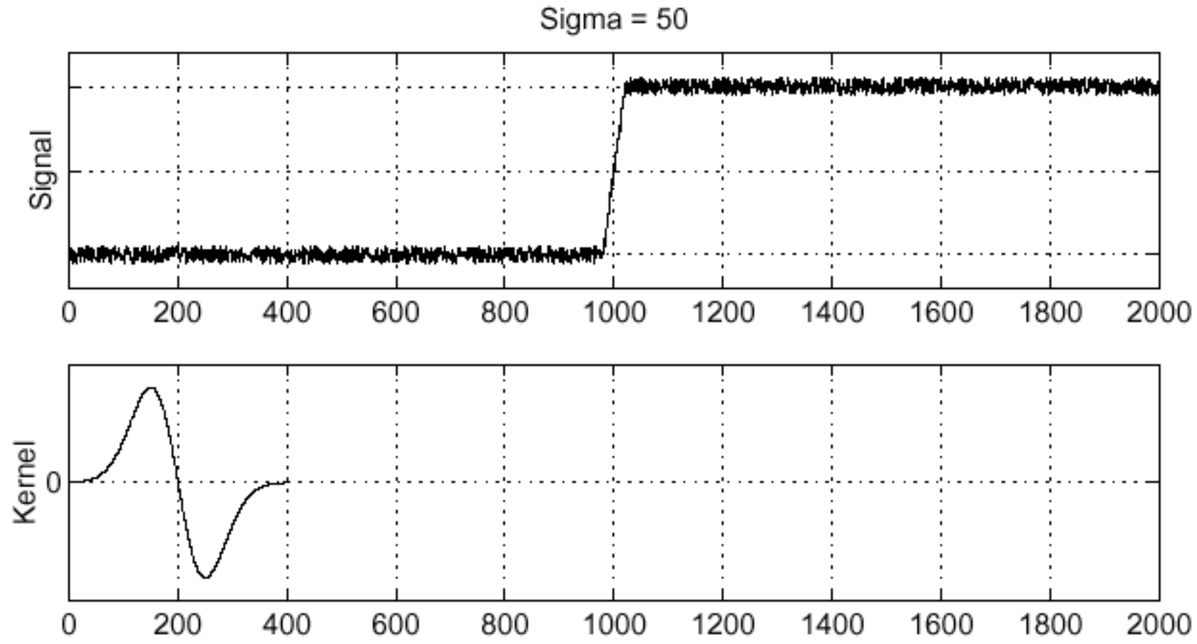


Théorème sur la dérivée de la convolution

$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

f

$\frac{\partial}{\partial x}h$



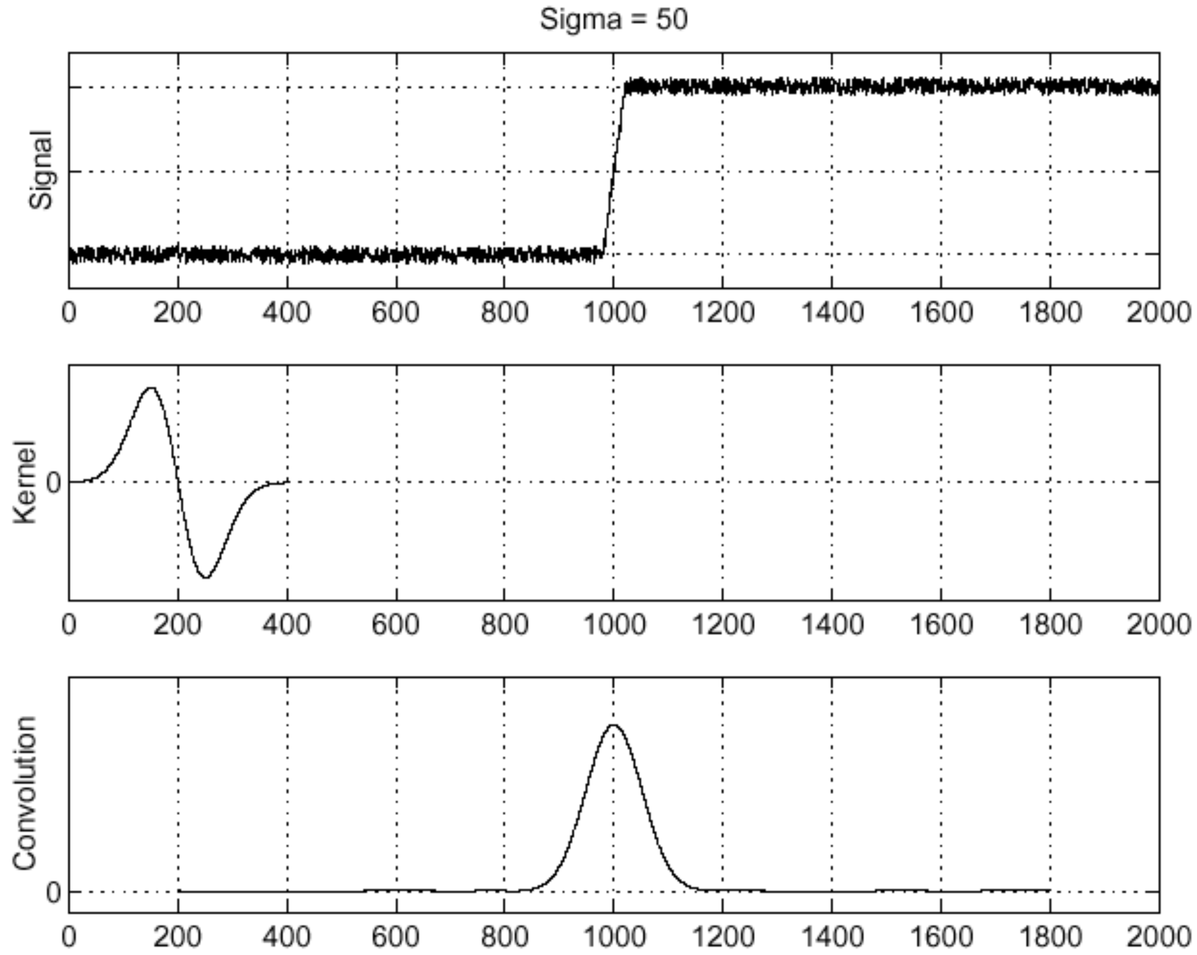
Théorème sur la dérivée de la convolution

$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

f

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$

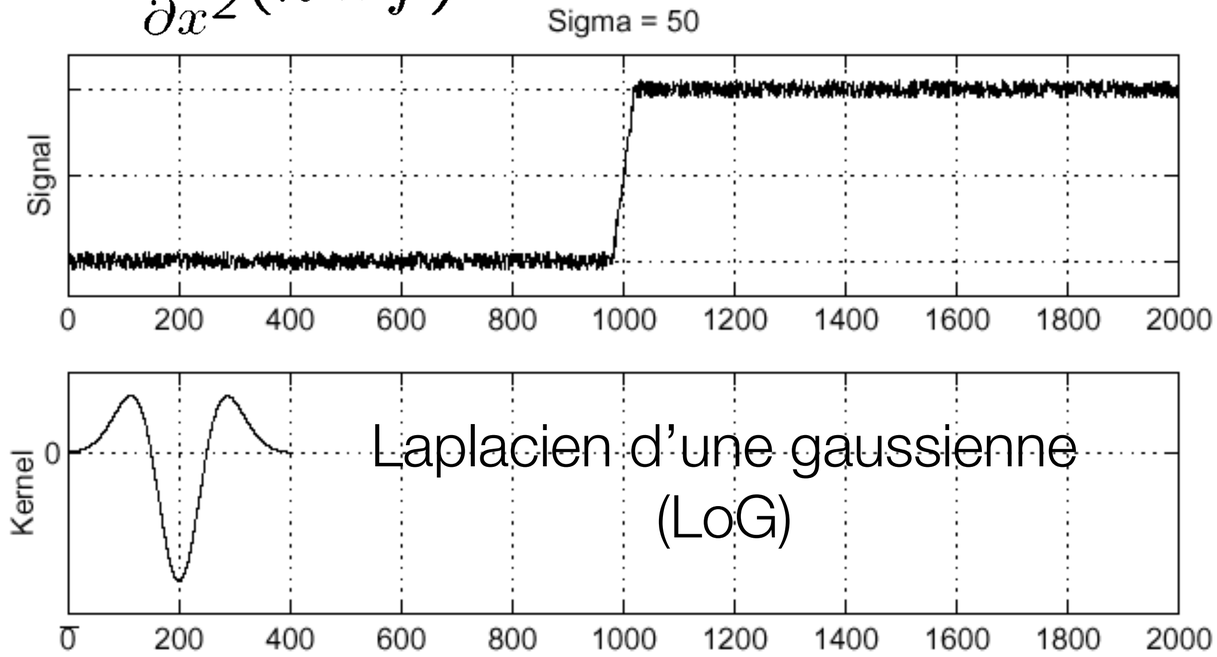


Laplacien d’une gaussienne

$$\frac{\partial^2}{\partial x^2}(h \star f)$$

f

$\frac{\partial^2}{\partial x^2} h$



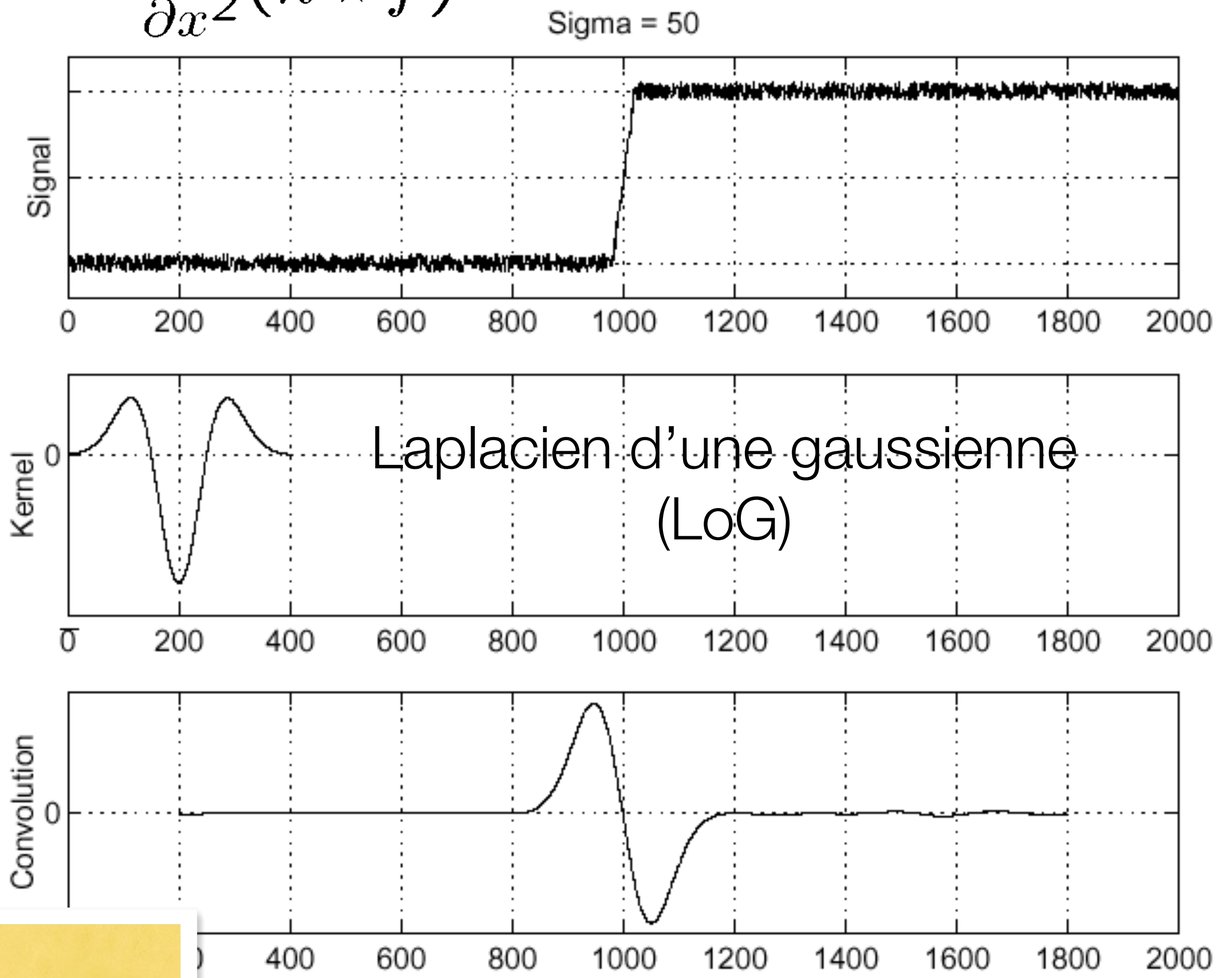
Laplacien d'une gaussienne

$$\frac{\partial^2}{\partial x^2}(h \star f)$$

f

$\frac{\partial^2}{\partial x^2}h$

$(\frac{\partial^2}{\partial x^2}h) \star f$



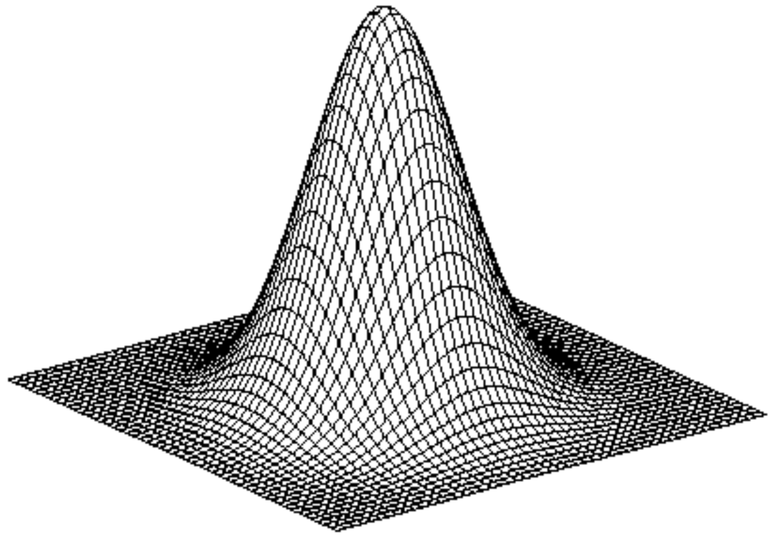
Laplacien d'une gaussienne (LoG)

Où est l'arête?

L'endroit où le graphe du bas croise 0

Détection d’arête en 2-D

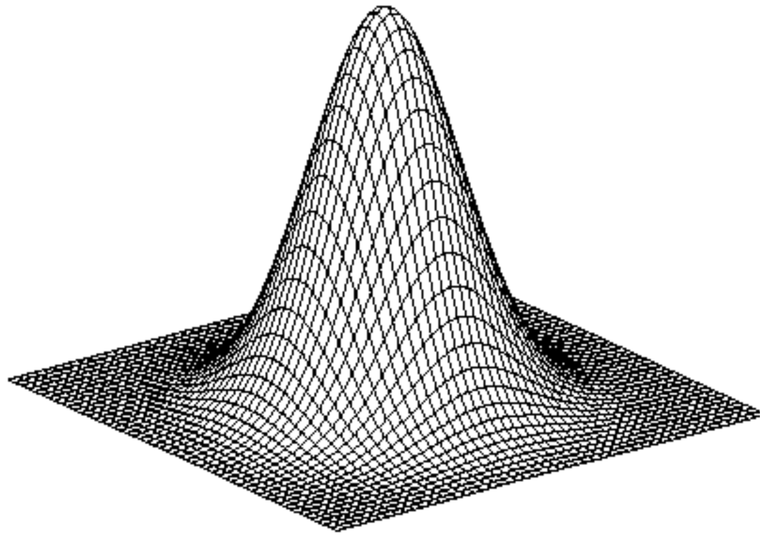
Gaussienne



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

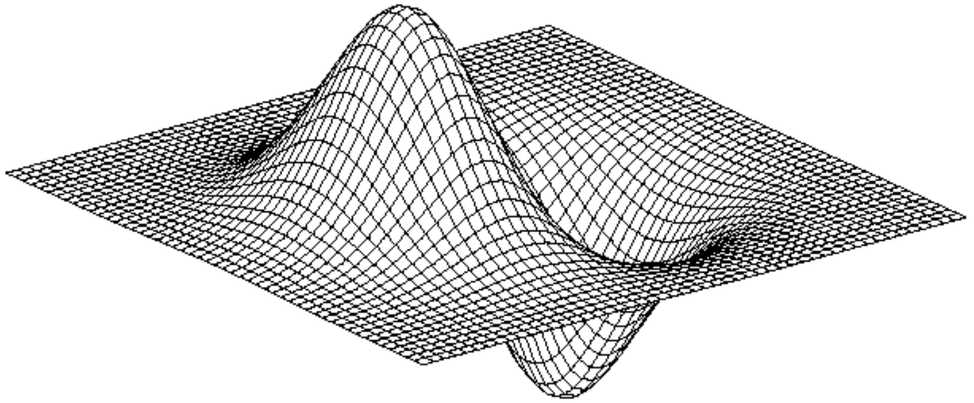
Détection d’arête en 2-D

Gaussienne



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

Dérivée d’une gaussienne



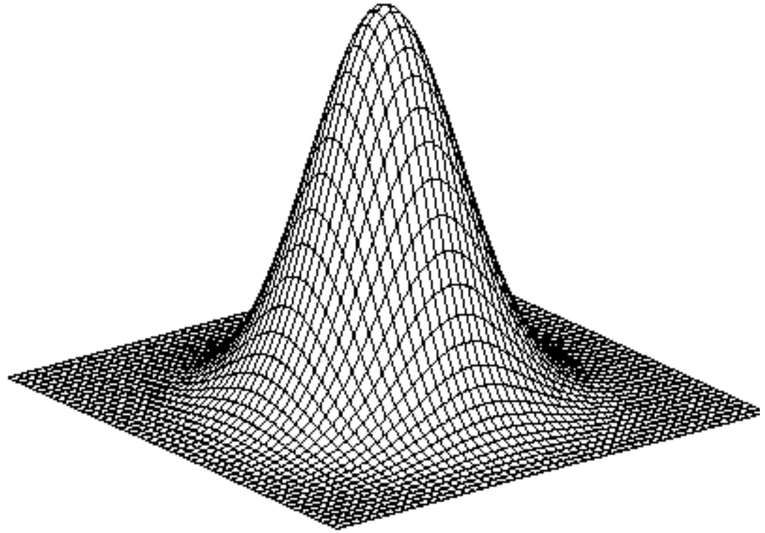
$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Combien de dérivée existe-t-il?

Détection d'arête en 2-D

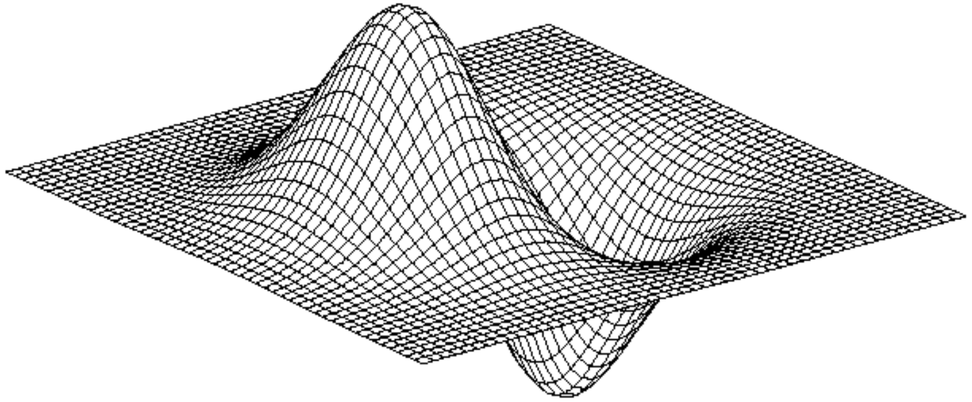
Combien de dérivée seconde existe-t-il?

Gaussienne



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

Dérivée d'une gaussienne

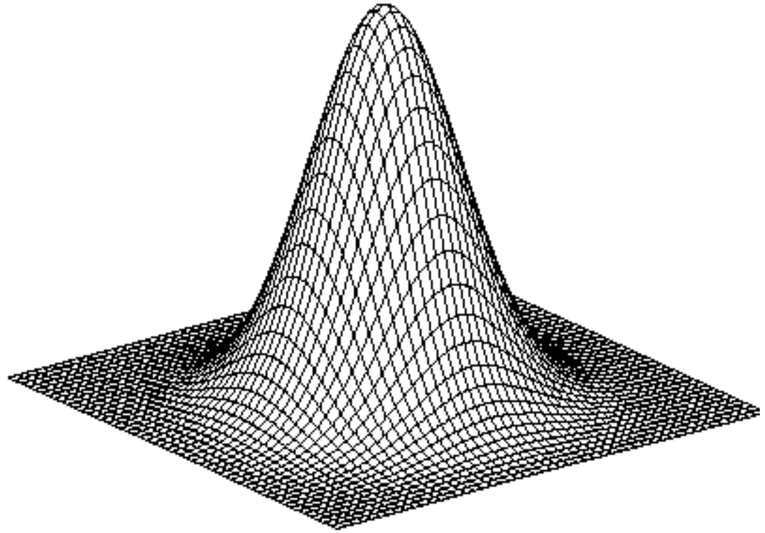


$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Dérivée seconde d'une gaussienne

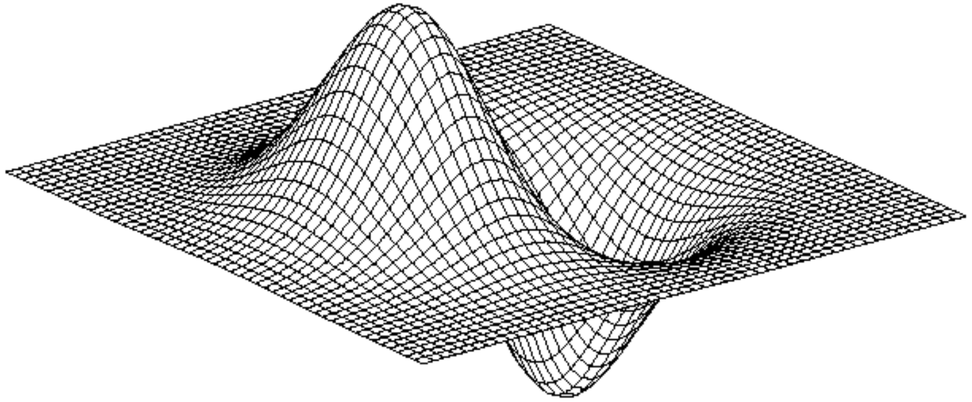
Détection d'arête en 2-D

Gaussienne



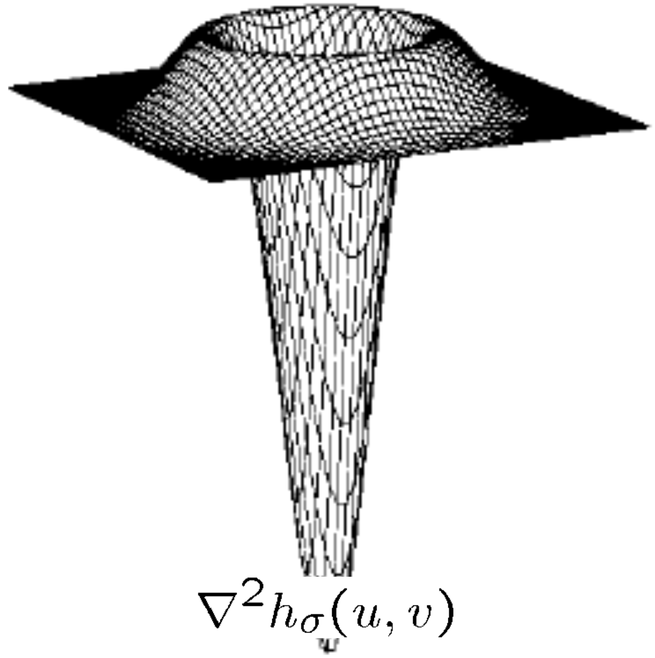
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

Dérivée d'une gaussienne



$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Dérivée seconde d'une gaussienne
Laplacien d'une gaussienne



$$\nabla^2 h_\sigma(u,v)$$

∇^2 est l'opérateur Laplacien: $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

Un filtre gaussien enlève…?

Image originale

Un filtre gaussien enlève…?

filtrée (filtre gaussien 5x5)

Filtre passe-haut

originale - filtrée = détails

Accentuation (*sharpening*)

originale

–

filtrée (5x5)

=

détails

Accentuation (*sharpening*)

originale

−

filtrée (5x5)

=

détails

Rajoutons les détails à l'image originale

originale

$+\alpha$

détails

=

accentuée!

Revenons à l'image de « détails »

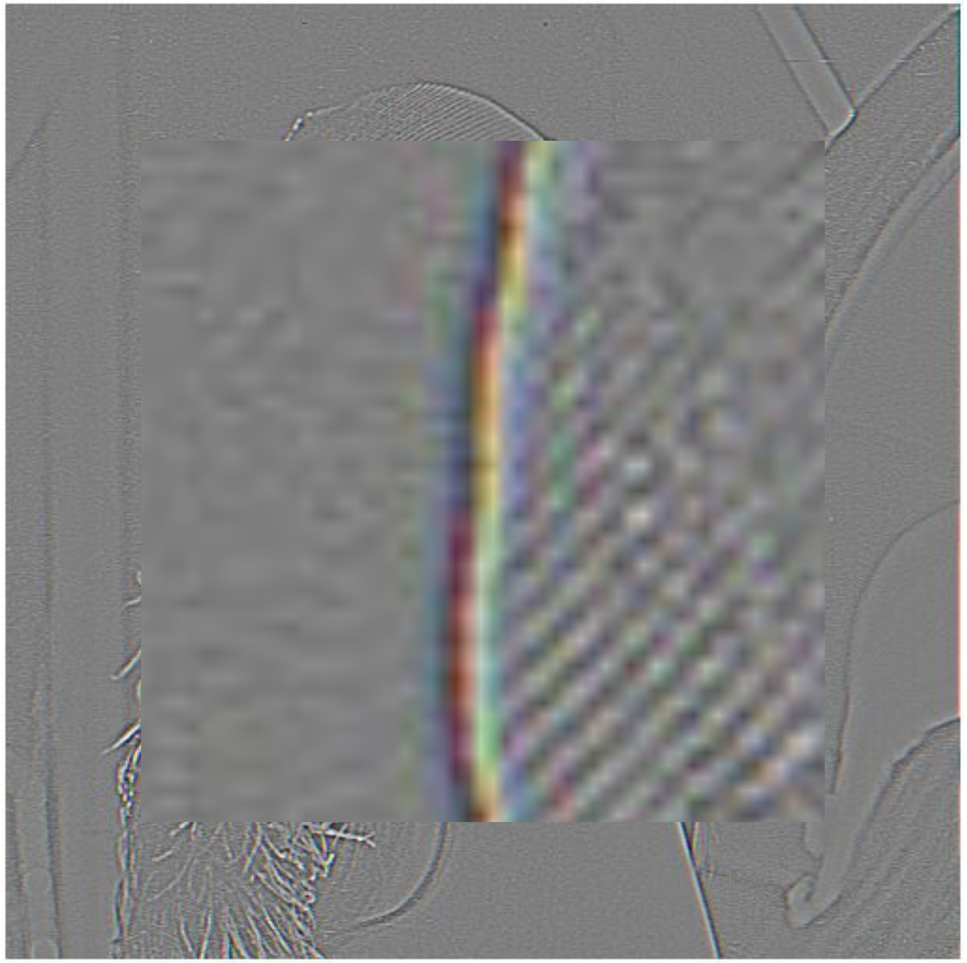
- Comment obtenir ce filtre passe-haut?
  - regarder de près…

originale - filtrée

Revenons à l’image de « détails »

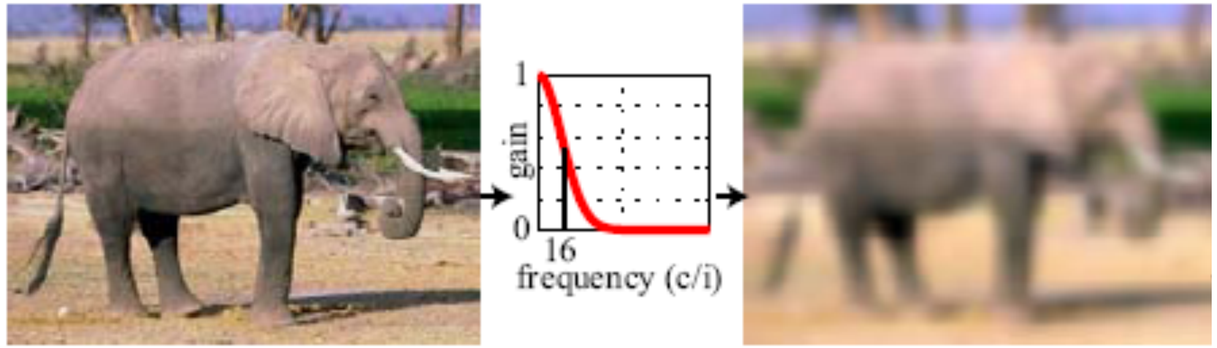
- Comment obtenir ce filtre passe-haut?
  - regarder de près…
  - passage par 0 aux arêtes…?
  - original - filtrée (gaussien) ≈ laplacien d’une gaussienne!

originale - filtrée



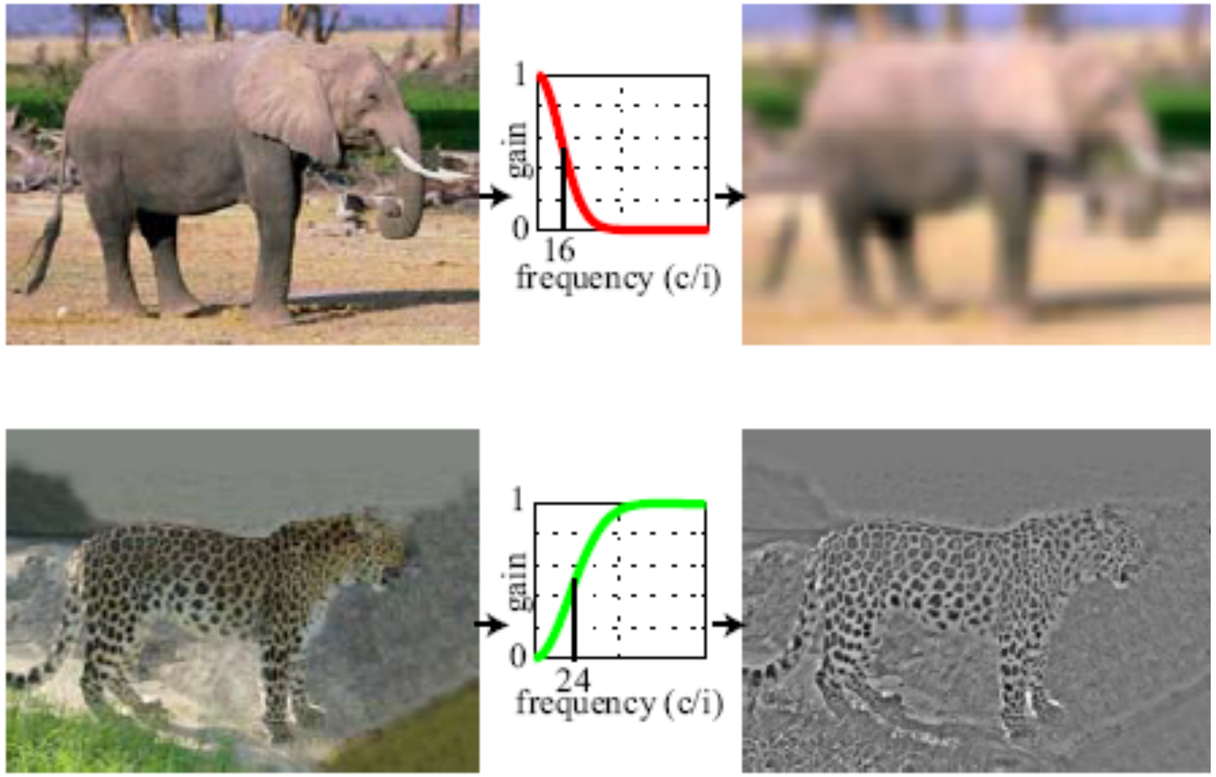
Images hybrides

A. Oliva, A. Torralba, P.G. Schyns,“Hybrid Images,” SIGGRAPH 2006



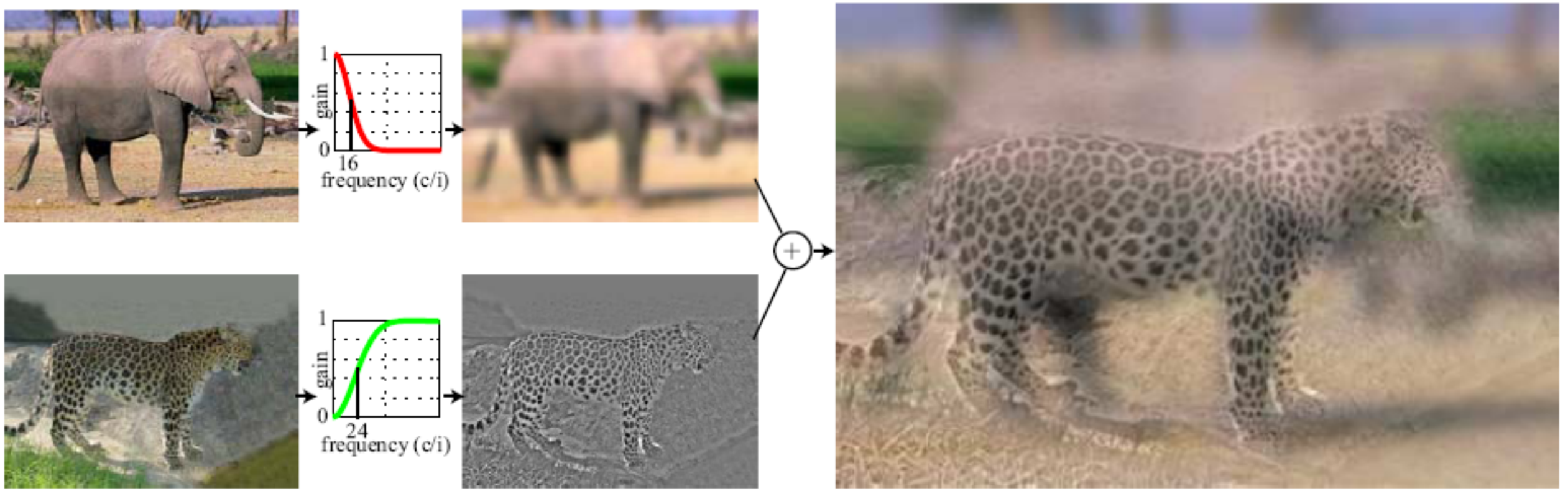
Images hybrides

A. Oliva, A. Torralba, P.G. Schyns,“Hybrid Images,” SIGGRAPH 2006



Images hybrides

A. Oliva, A. Torralba, P.G. Schyns,“Hybrid Images,” SIGGRAPH 2006



Images hybrides

De **près**, on voit une image
(**hautes** fréquences)

A. Oliva, A. Torralba, P.G. Schyns,“Hybrid Images,” SIGGRAPH 2006

Images hybrides

De **loin**, on voit une autre image
(**basses** fréquences)

A. Oliva, A. Torralba, P.G. Schyns,“Hybrid Images,” SIGGRAPH 2006

Ça vous rappelle quelque chose?

Salvador Dali
“Gala contemplant la mer Méditerranée qui à vingt mètres devient le portrait d'Abraham Lincoln ”, 1976

Rappel : compression (*chroma subsampling*)

Autre espace de couleur… Y’CbCr : Y’ (luma) et CbCr (chroma) mais l’idée reste la même

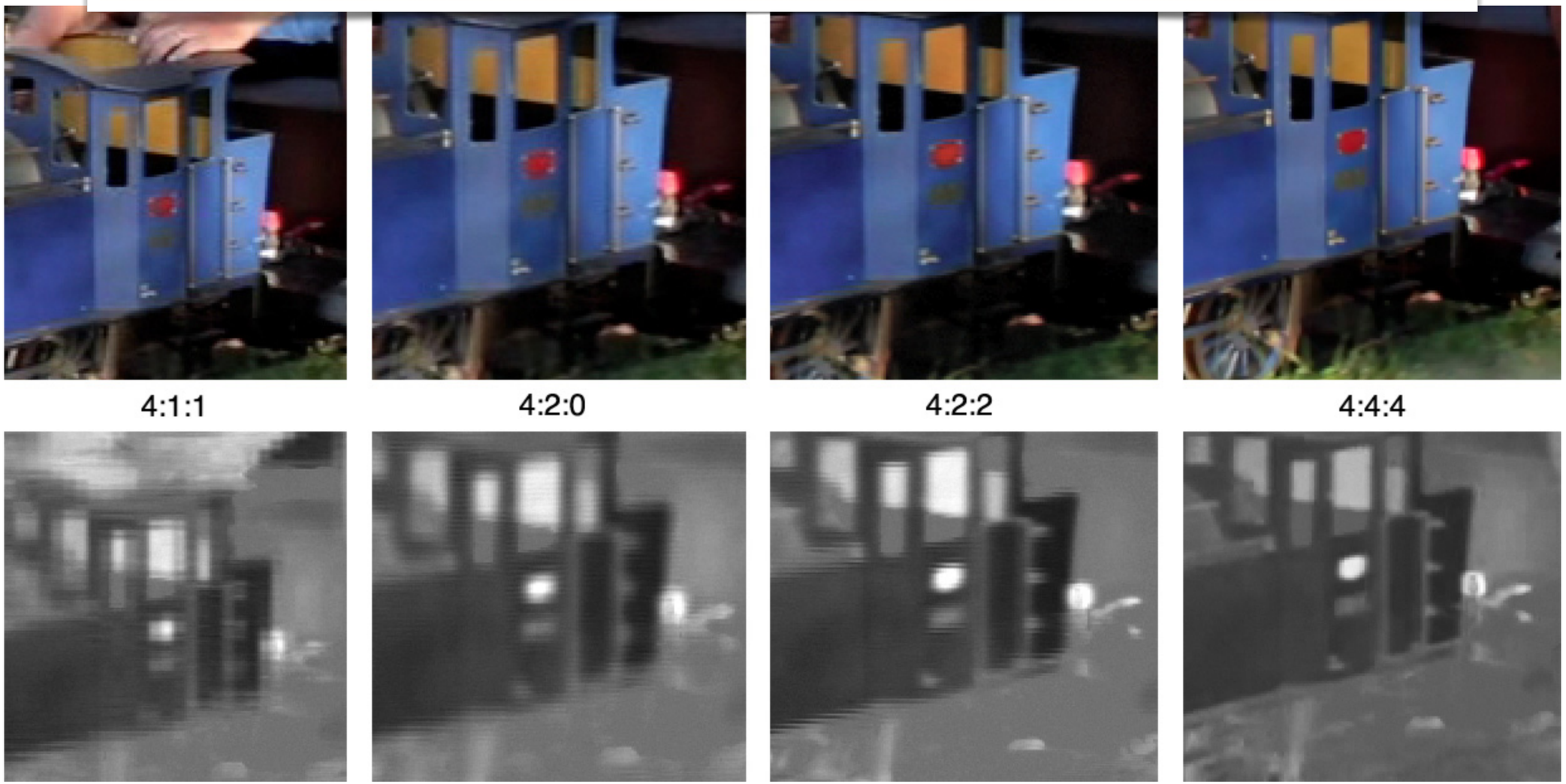


Image : source

Rappel (bis)

Pourquoi peut-on toujours interpréter une image à plus faible résolution?
Quelle est l'information perdue?

Résolution originale

1/2 résolution
(4x moins de pixels!)

Image: http://www.flickr.com/photos/igorms/136916757/

Compression JPEG

Pas du « filtrage » comme tel, mais une bonne façon de penser aux fréquences d'une image

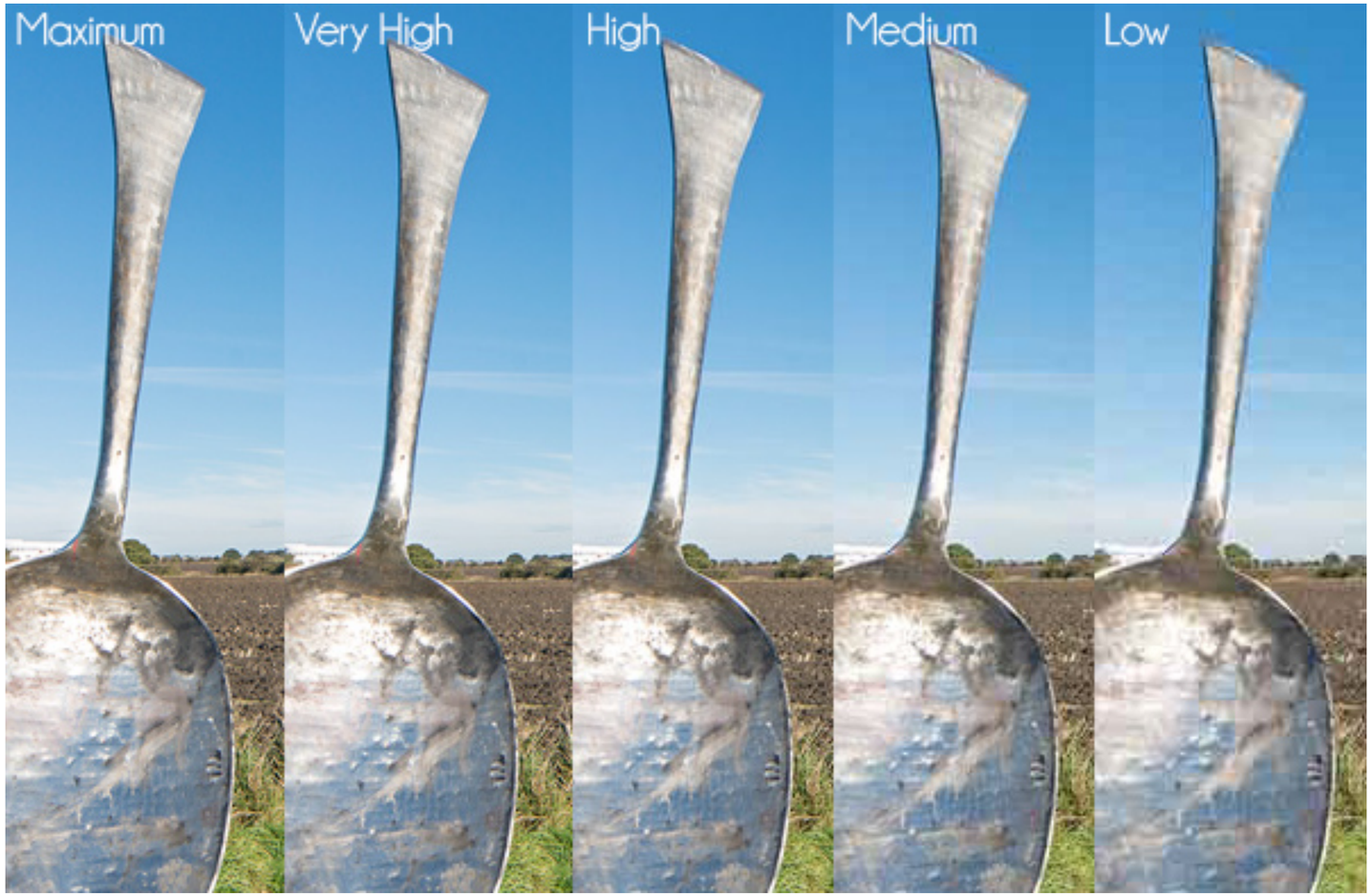
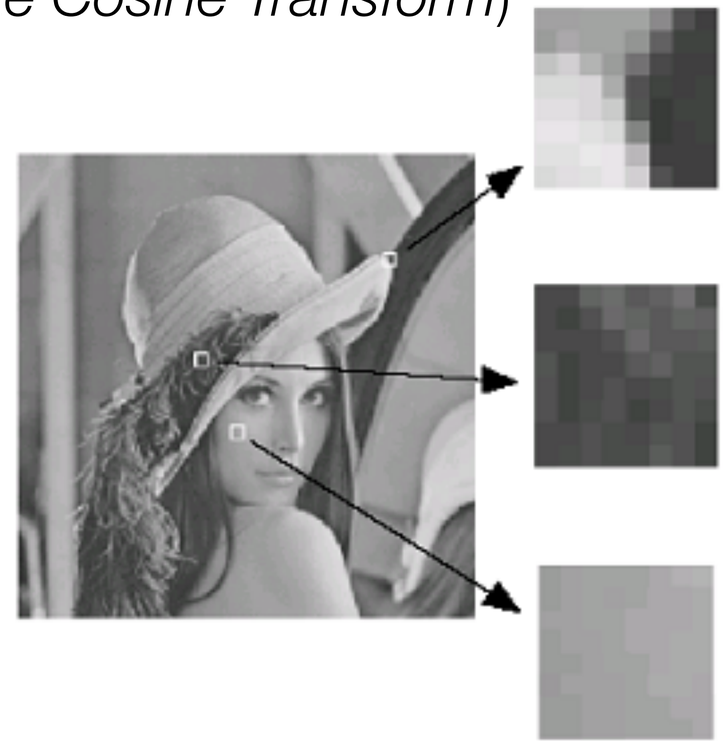
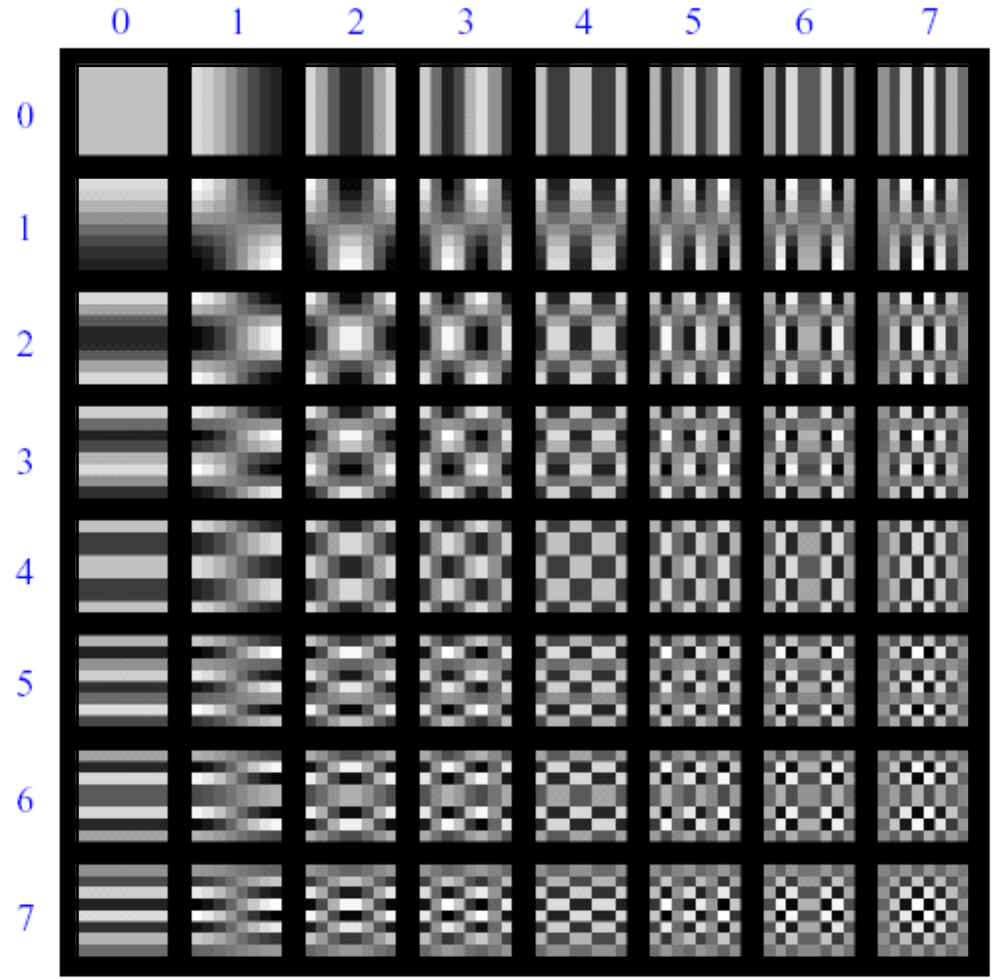


Image : source

Compression (JPEG)

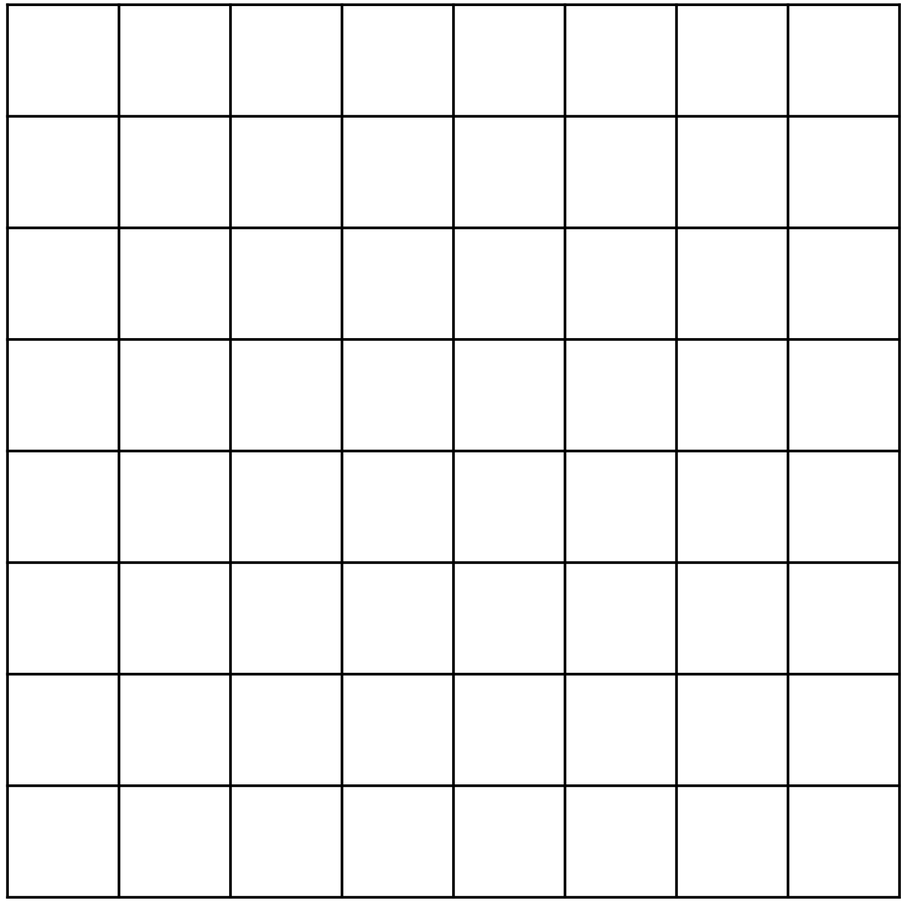
- Subdiviser en petits blocs de 8x8
- Pour chaque bloc :
  1. Calculer les coefficients de la DCT (*Discrete Cosine Transform*)

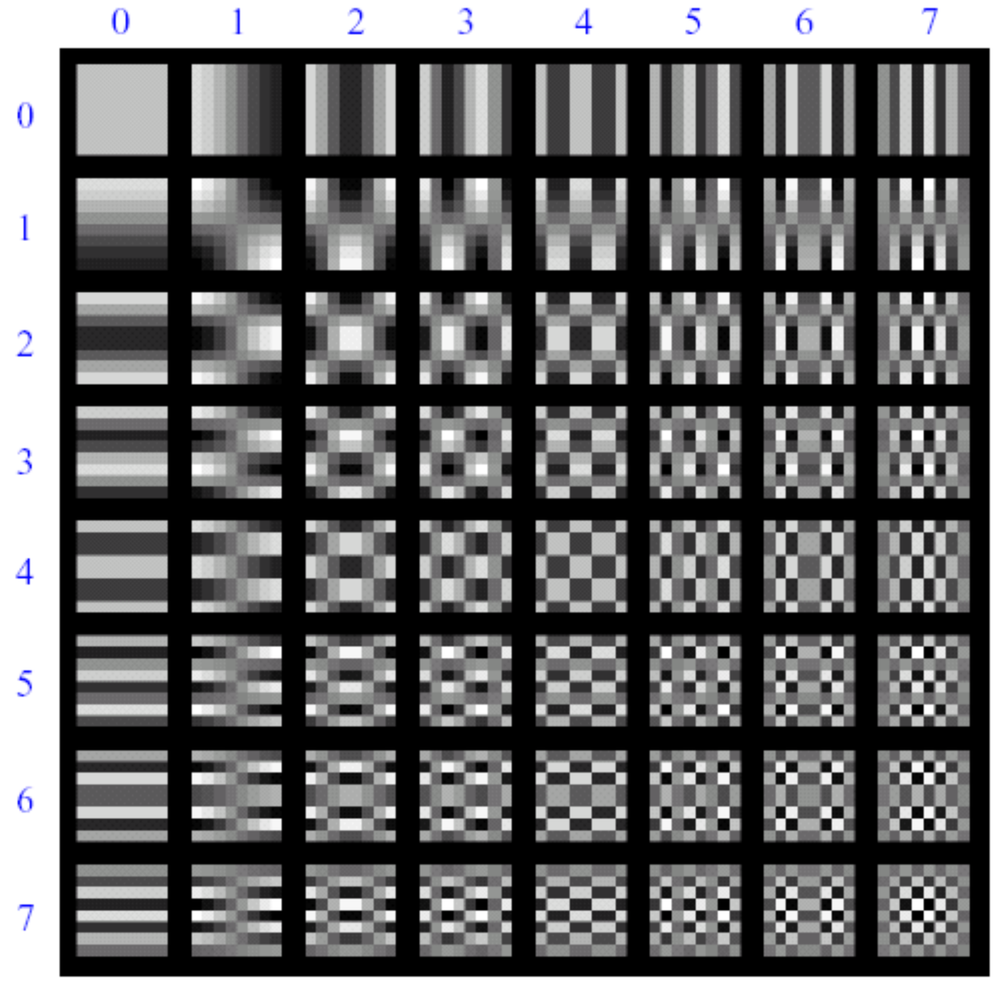




Compression (JPEG)

1. Calculer les coefficients de la DCT (*Discrete Cosine Transform*)





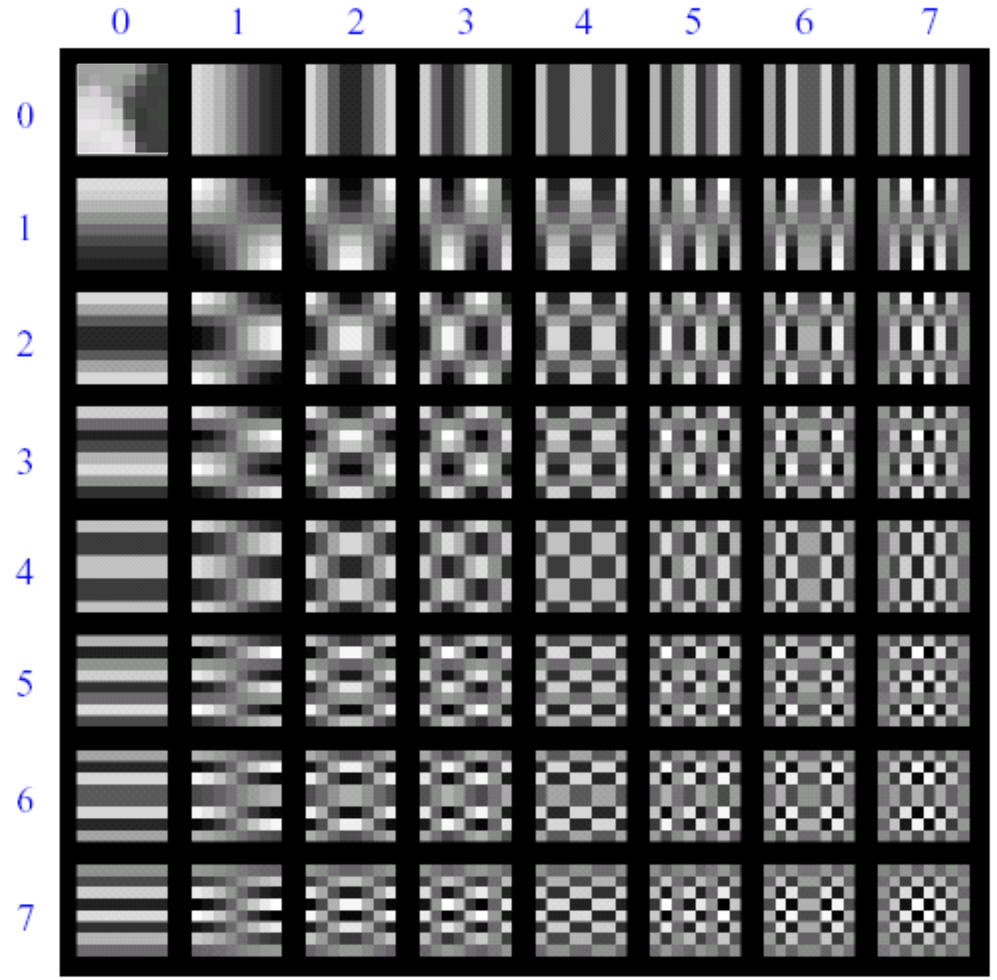
Compression (JPEG)

1. Calculer les coefficients de la DCT (*Discrete Cosine Transform*)

186							

Composante DC!

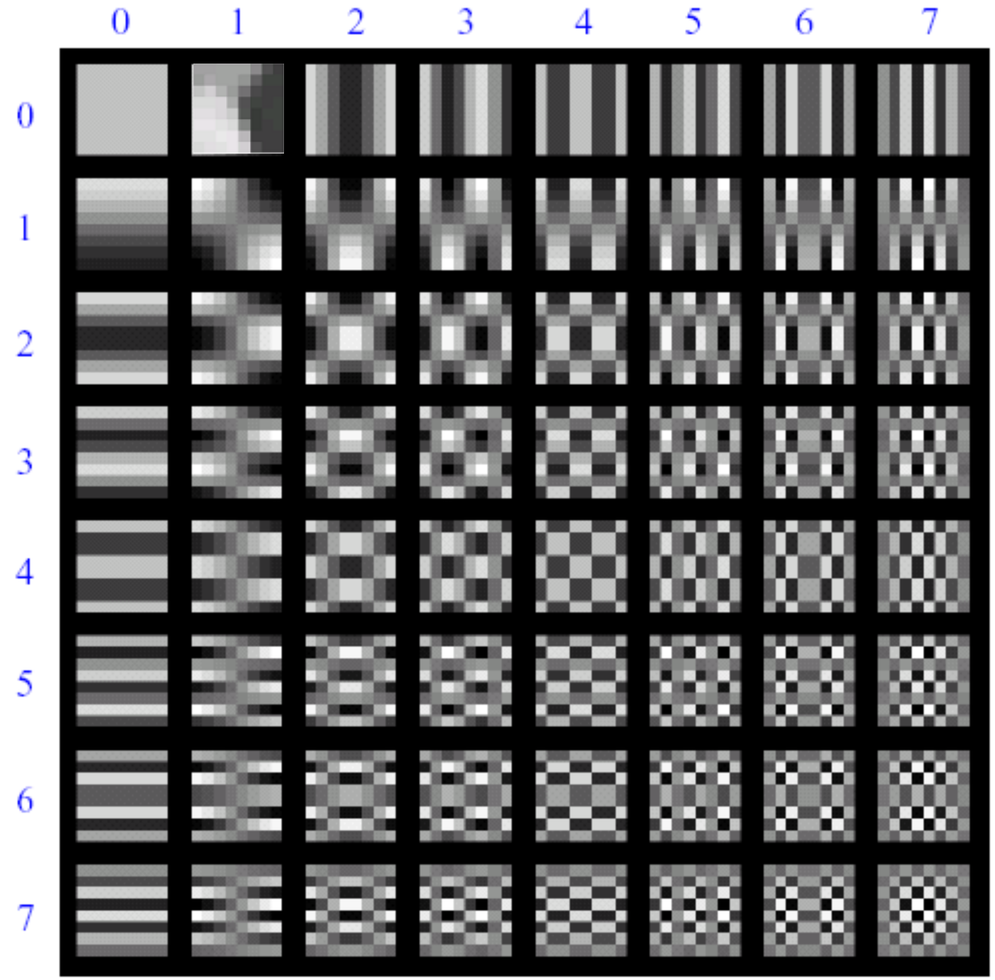
Pour calculer le coefficient,
ou on calcule le **produit scalaire** entre le bloc et la base.
Il s'agit donc d'une **projection** du bloc sur la base.



Compression (JPEG)

1. Calculer les coefficients de la DCT (*Discrete Cosine Transform*)

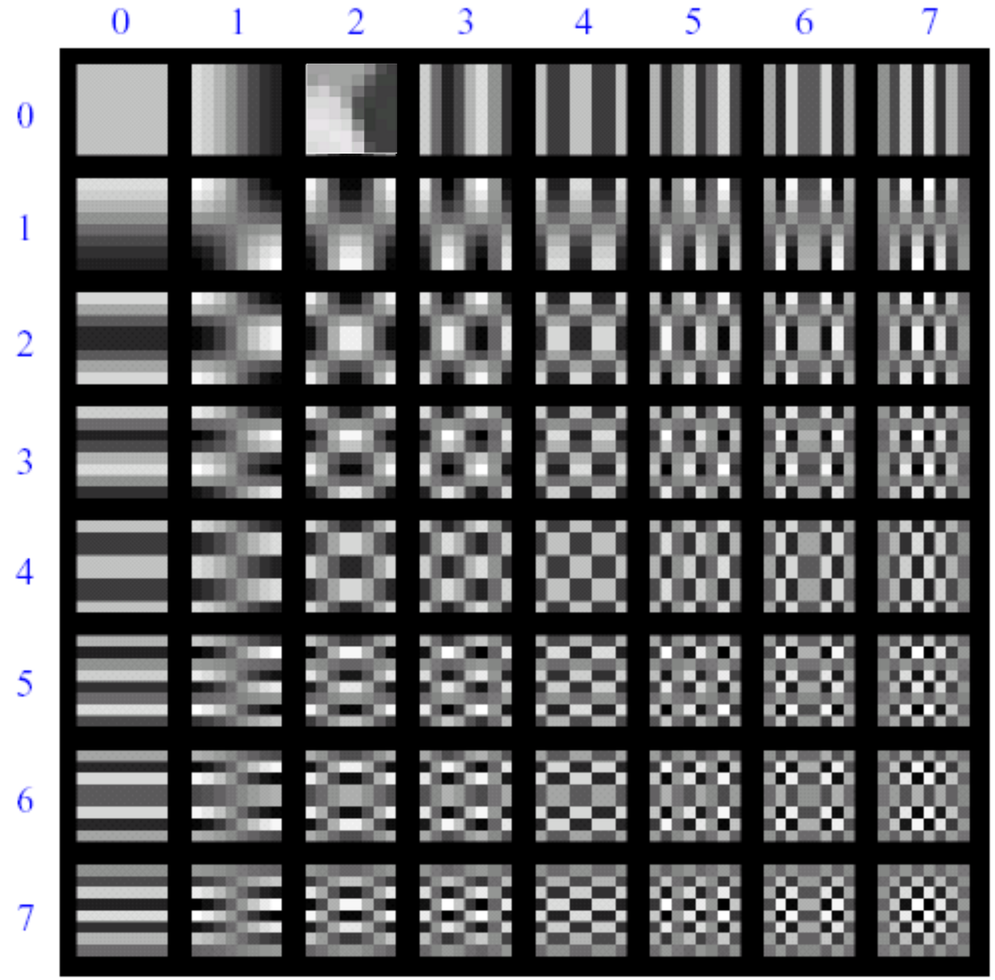
186	-18						



Compression (JPEG)

1. Calculer les coefficients de la DCT (*Discrete Cosine Transform*)

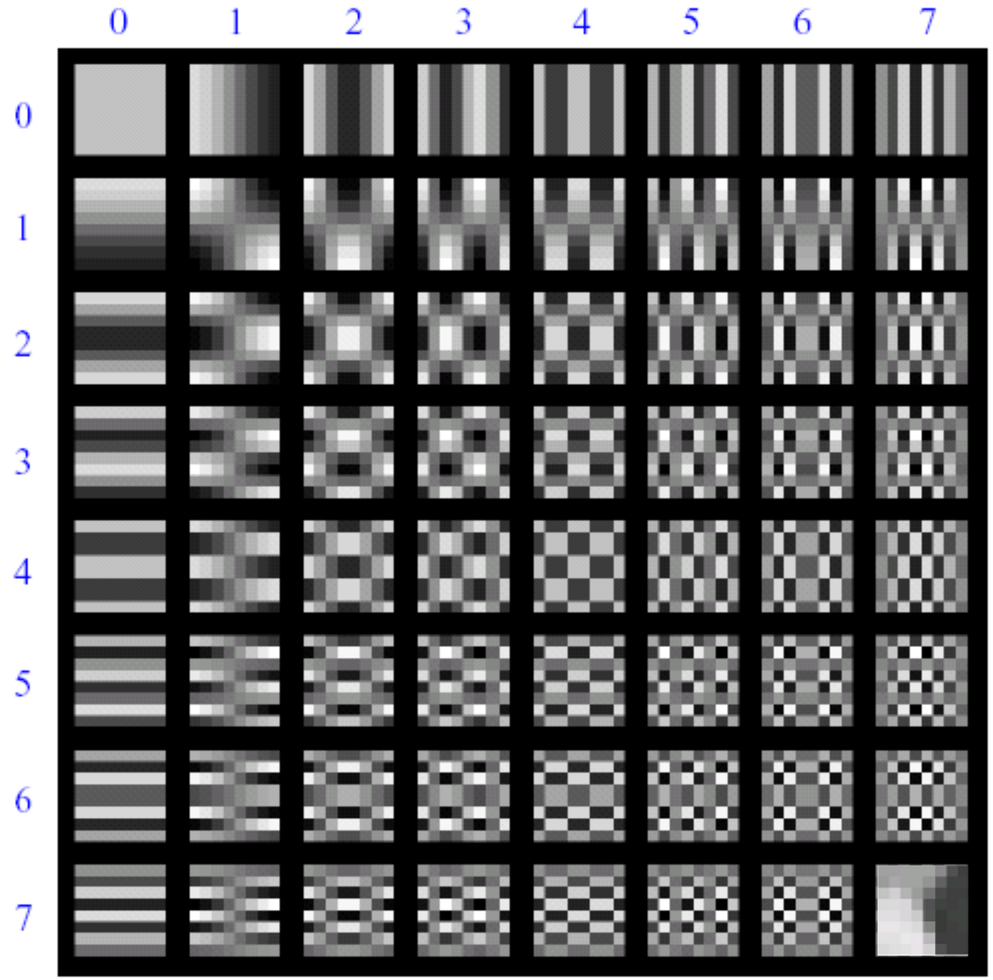
186	-18	15					



Compression (JPEG)

1. Calculer les coefficients de la DCT (*Discrete Cosine Transform*)

186	-18	15	-9	23	-9	-14	19
21	-34	26	-9	-11	11	14	7
-10	-24	-2	6	-18	3	-20	-1
-8	-5	14	-15	-8	-3	-3	8
-3	10	8	1	-11	18	18	15
4	-2	-18	8	8	-4	1	-7
9	1	-3	4	-1	-7	-1	-2
0	-8	-2	2	1	4	-6	0



Compression (JPEG)

1. Calculer les coefficients de la DCT (*Discrete Cosine Transform*)
2. Quantifier les coefficients de la DCT (niveau de quantification **différent** pour chaque fréquence!)

Table de quantification

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

186	-18	15	-9	23	-9	-14	19
21	-34	26	-9	-11	11	14	7
-10	-24	-2	6	-18	3	-20	-1
-8	-5	14	-15	-8	-3	-3	8
-3	10	8	1	-11	18	18	15
4	-2	-18	8	8	-4	1	-7
9	1	-3	4	-1	-7	-1	-2
0	-8	-2	2	1	4	-6	0

Compression (JPEG)

1. Calculer les coefficients de la DCT (*Discrete Cosine Transform*)
2. Quantifier les coefficients de la DCT (niveau de quantification **différent** pour chaque fréquence!)

Table de quantification

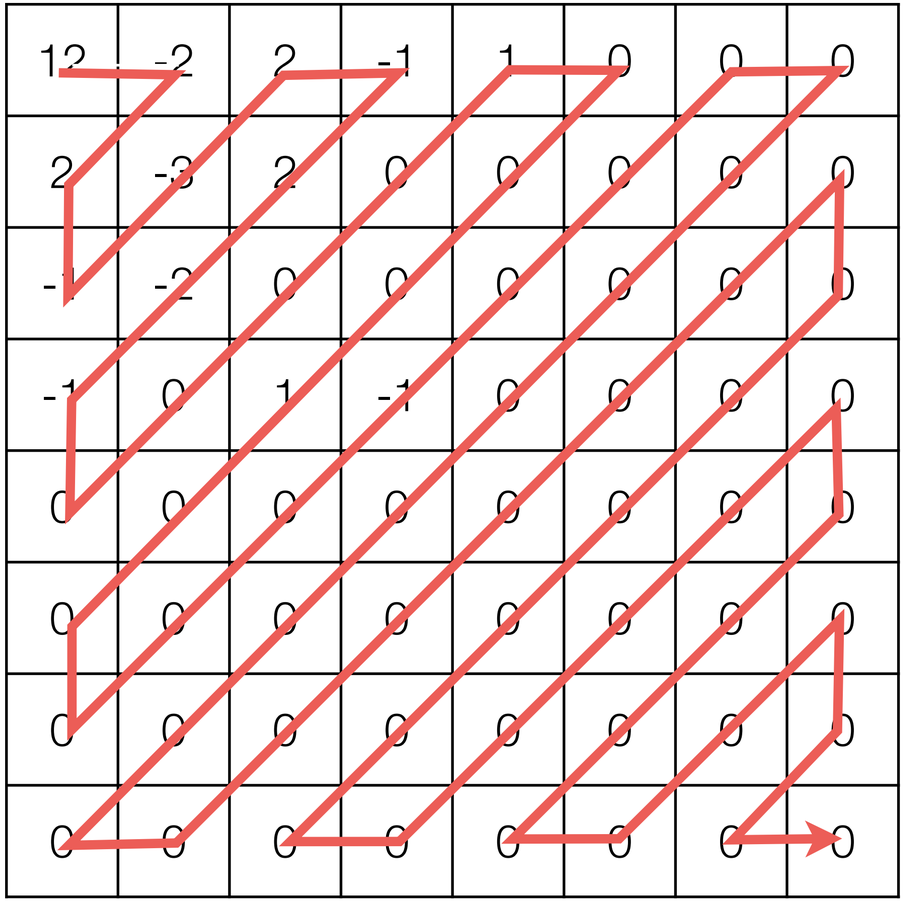
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Que remarque-t-on?

12	-2	2	-1	1	0	0	0
2	-3	2	0	0	0	0	0
-1	-2	0	0	0	0	0	0
-1	0	1	-1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Compression (JPEG)

1. Calculer les coefficients de la DCT (*Discrete Cosine Transform*)
2. Quantifier les coefficients de la DCT (niveau de quantification **différent** pour chaque fréquence!)
3. Encoder les coefficients quantifiés (encodage Huffman)



Taille des blocs

- petit
  - rapide!
  - corrélation existe entre blocs adjacents (compression moins efficace)
- grand
  - meilleure compression
- 8x8 dans le standard JPEG


taille du fichier : 89k


taille du fichier : 12k
À quels endroits remarque-t-on plus les différences?