Applications du filtrage

GIF-4105/7105

Photographie Algorithmique

Jean-François Lalonde

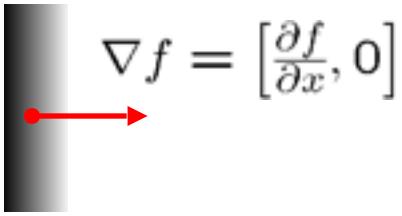
Merci à D. Hoiem, A. Efros,
S. Lazebnik, S. Seitz

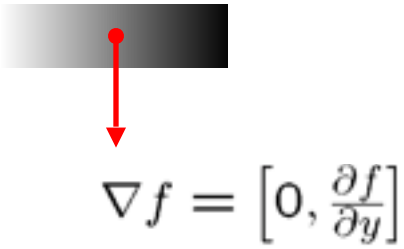
Gradient

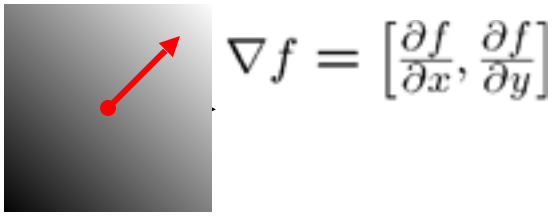
Le gradient d'une image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

Pointe dans la direction du changement le plus rapide en intensité







La direction est donnée par:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- quel est le lien avec la direction de l'arête?

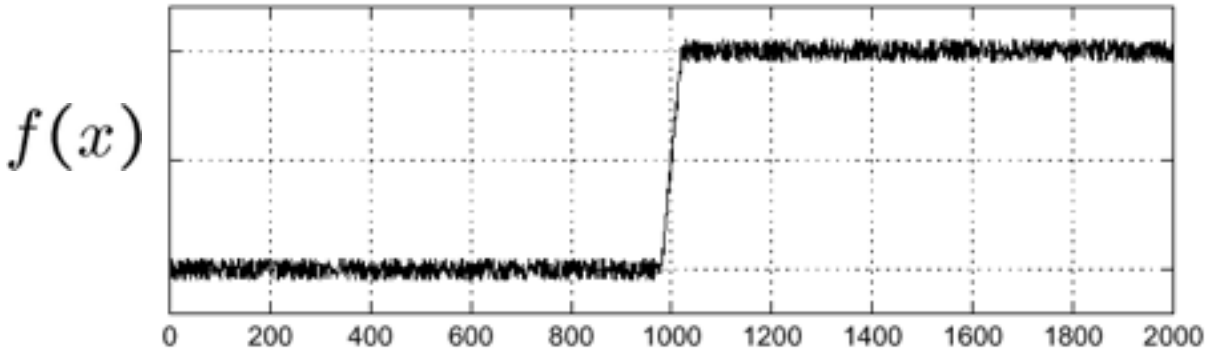
La "force" du gradient est la magnitude:

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

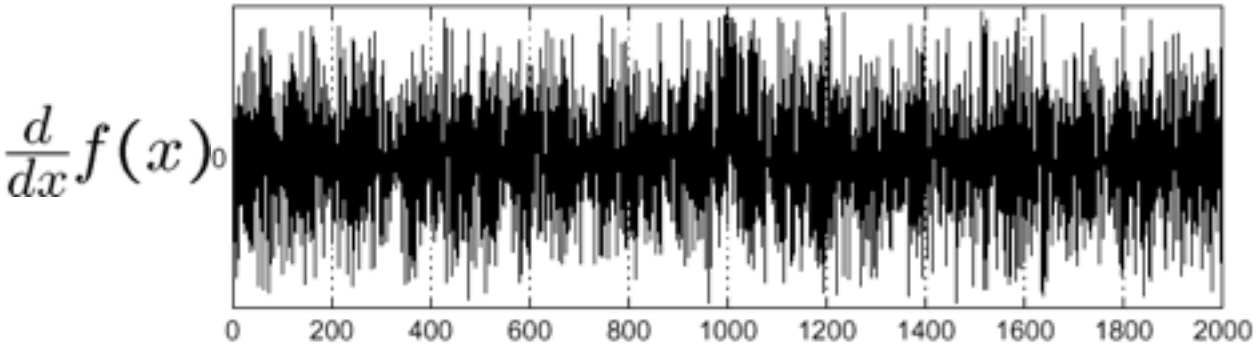
Bruit

Analysons une seule ligne dans l'image

- Affiche l'intensité en fonction de la coordonnée *x*



Comment calculer le gradient (la dérivée?)



Où est l'arête?

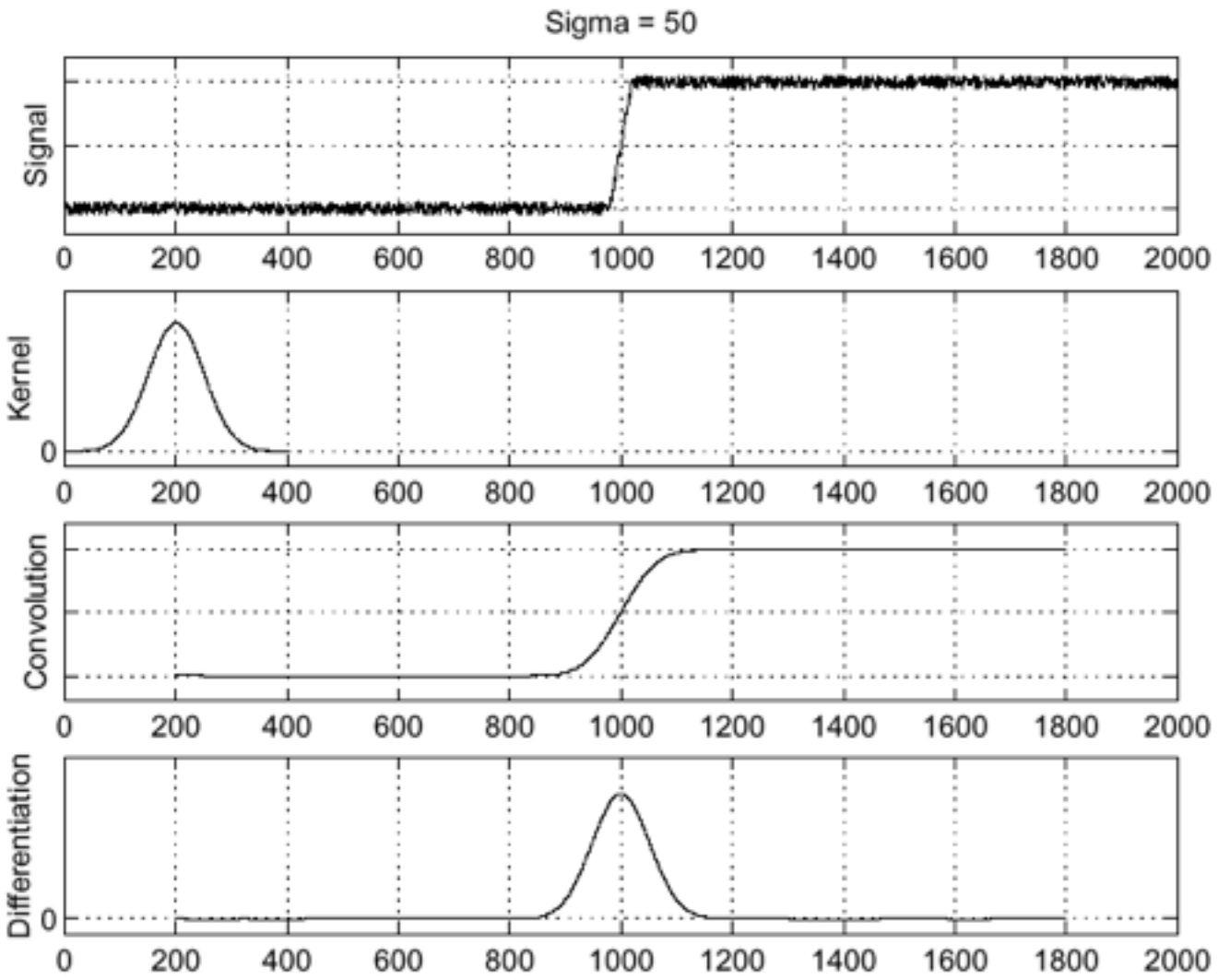
Solution: adoucir! (filtrer!)

f

h

$h \star f$

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



Où est l'arête?

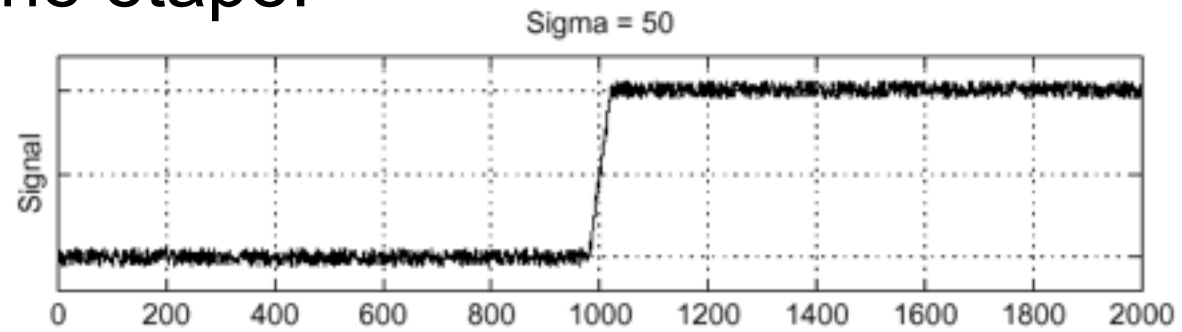
Chercher maximums: $\frac{\partial}{\partial x}(h \star f)$

Théorème sur la dérivée de la convolution

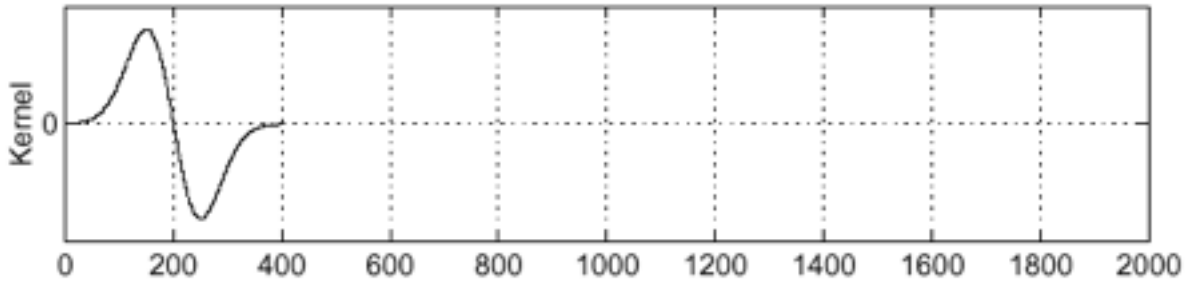
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

On sauve une étape:

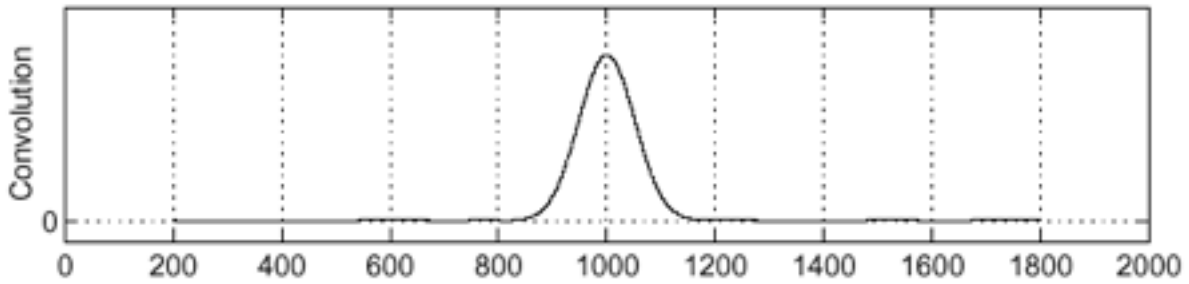
f



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



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



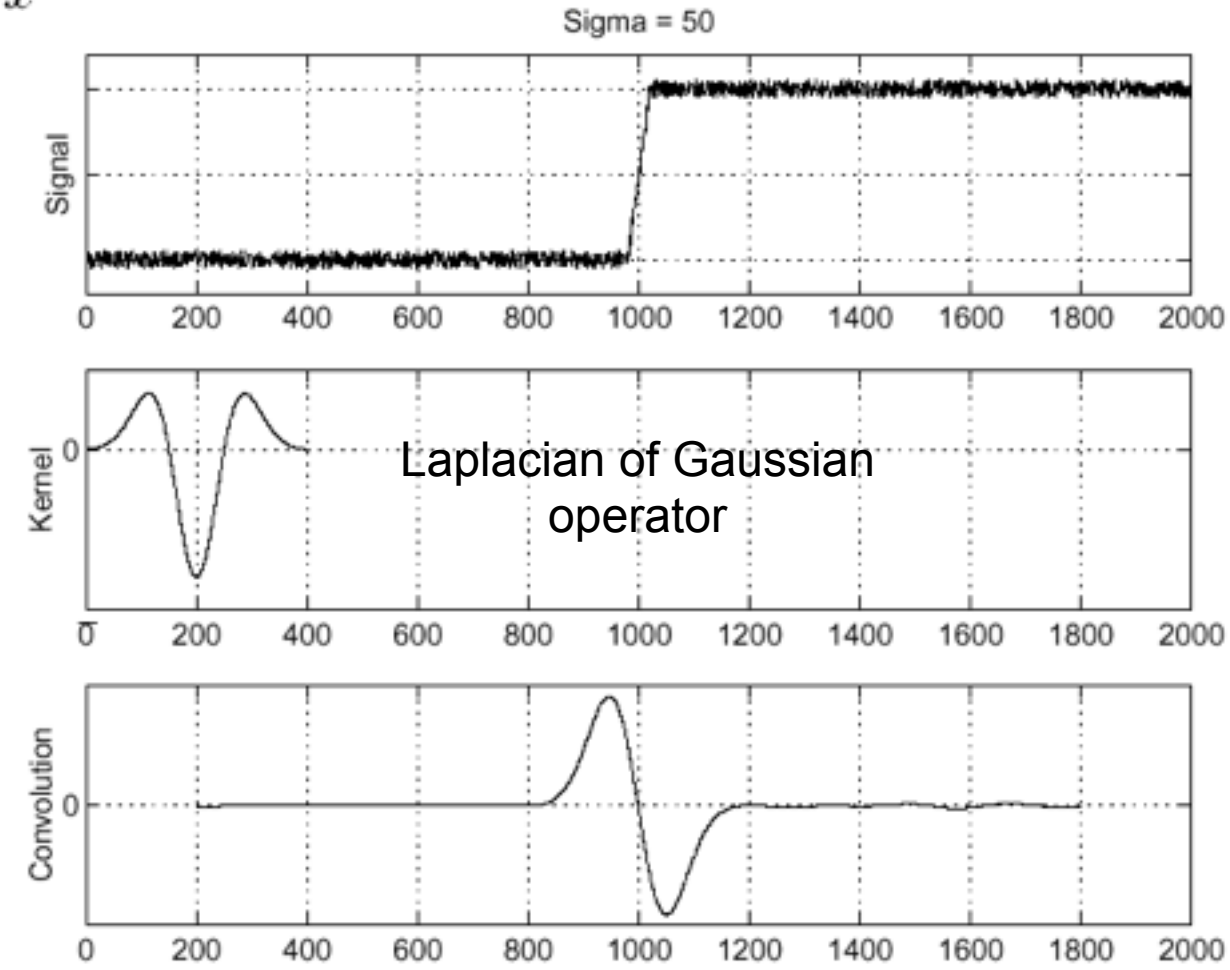
Laplacien d'une gaussienne

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

f

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

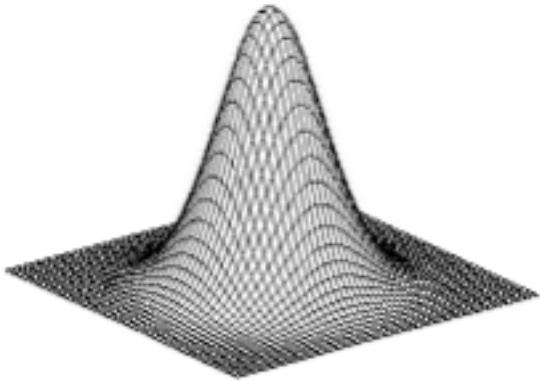
$(\frac{\partial^2}{\partial x^2} h) \star f$



Où est l'arête?

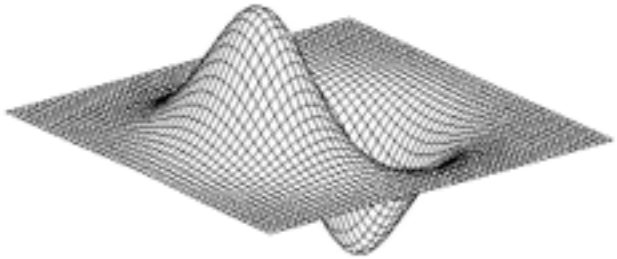
Où le graphe du bas croise 0

Détection d'arête en 2-D



Gaussienne

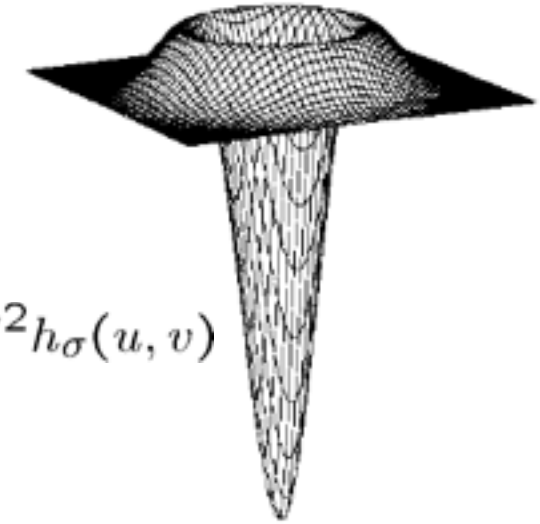
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



dérivée d'une gaussienne

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

Laplacien d'une gaussienne



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

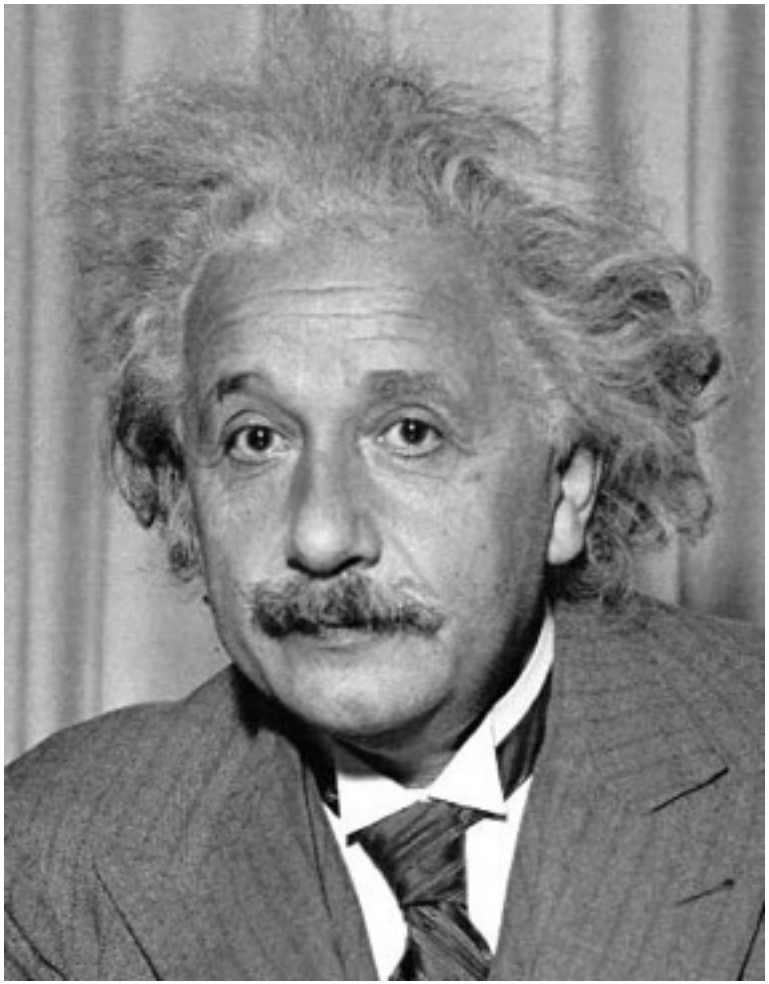
∇^2 est l'opérateur **Laplacien**:

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

Correspondance de modèles

But: trouver  dans l'image

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



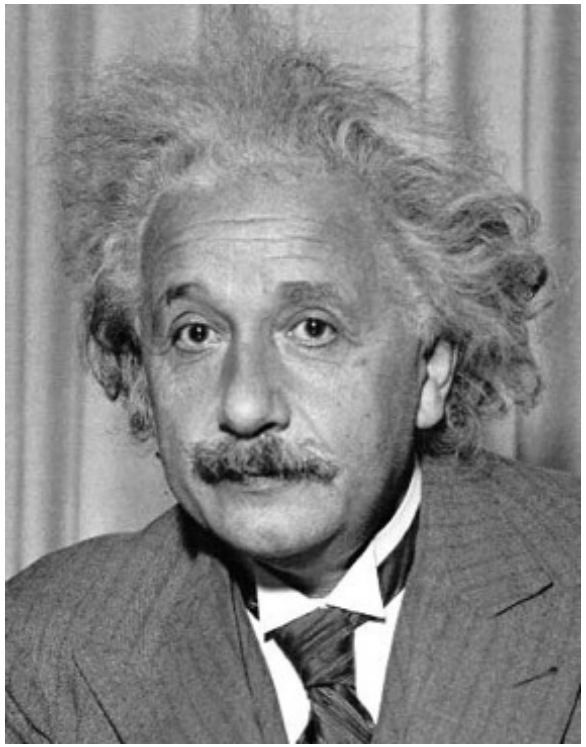
Filtrer pour trouver les correspondances

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



Image

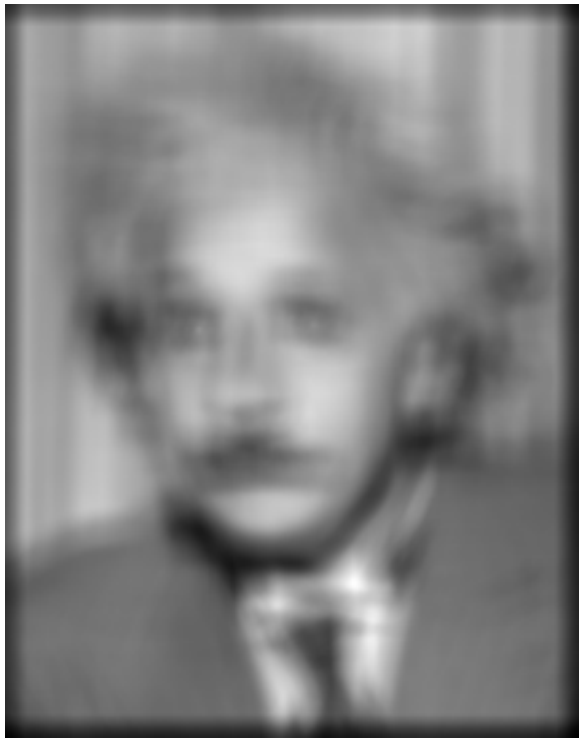


Image filtrée

Qu'est-ce qui se passe?

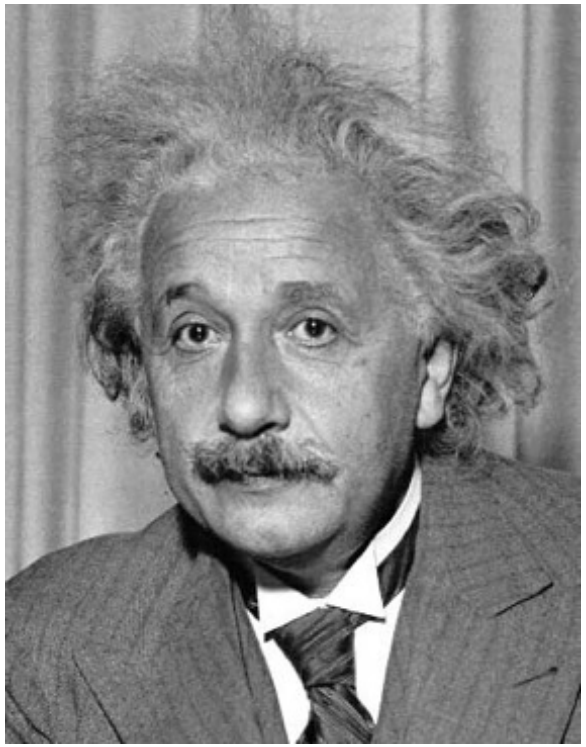
Filtrer pour trouver les correspondances

But: trouver  dans l'image

Méthode 1: filtrer l'image avec l'oeil (normalisé)

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

moyenne de g



Image

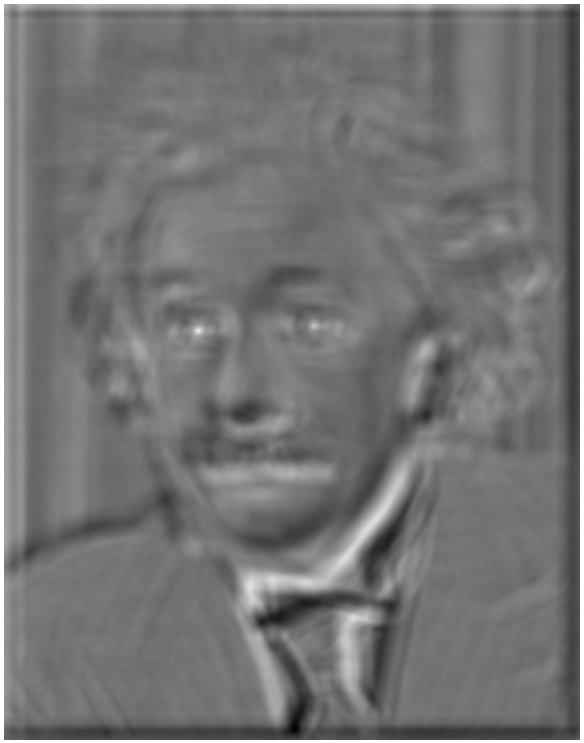
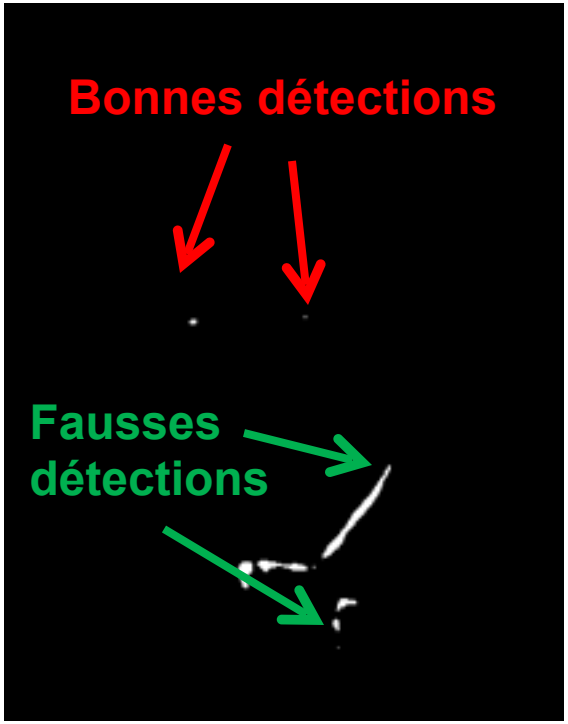


Image filtrée



Seuil

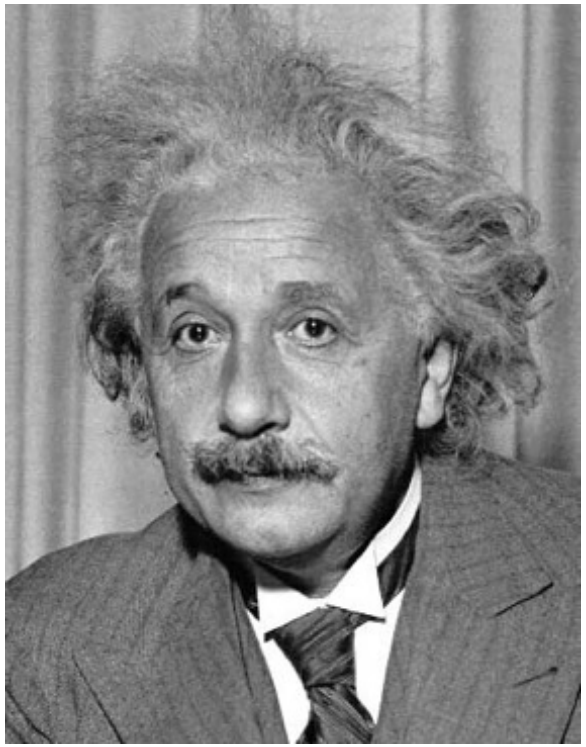
Derek Hoiem

Filtrer pour trouver les correspondances

But: trouver  dans l'image

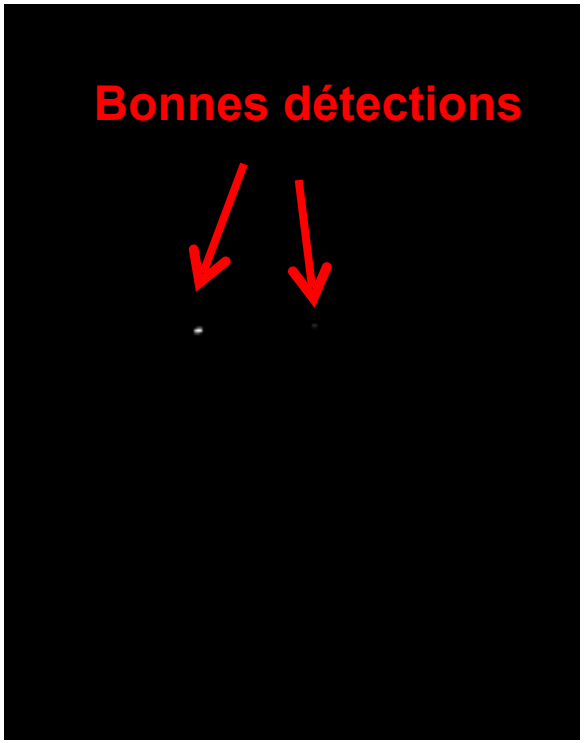
Méthode 2: somme des différences au carré

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



Image

1- sqrt(SSD)



Seuil

Derek Hoiem

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

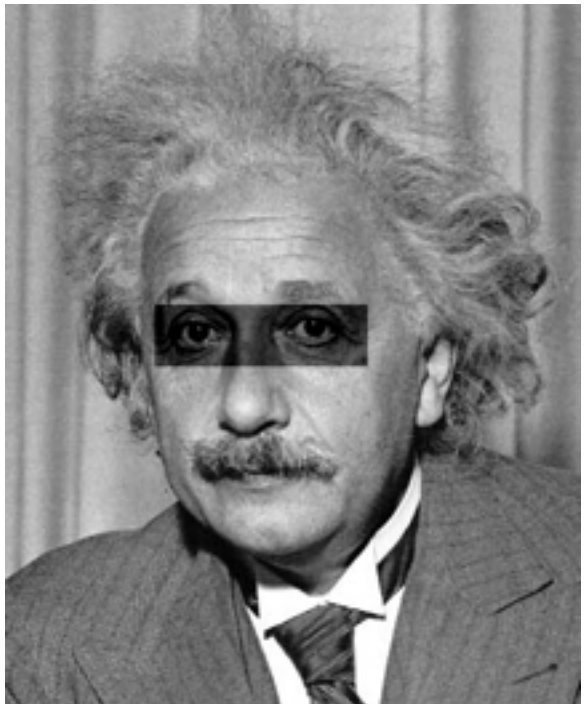
$$h(m,n) = \sum_{k,l} (g(k,l) - f(m+k, n+l))^2$$

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- sqrt(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 de l'image

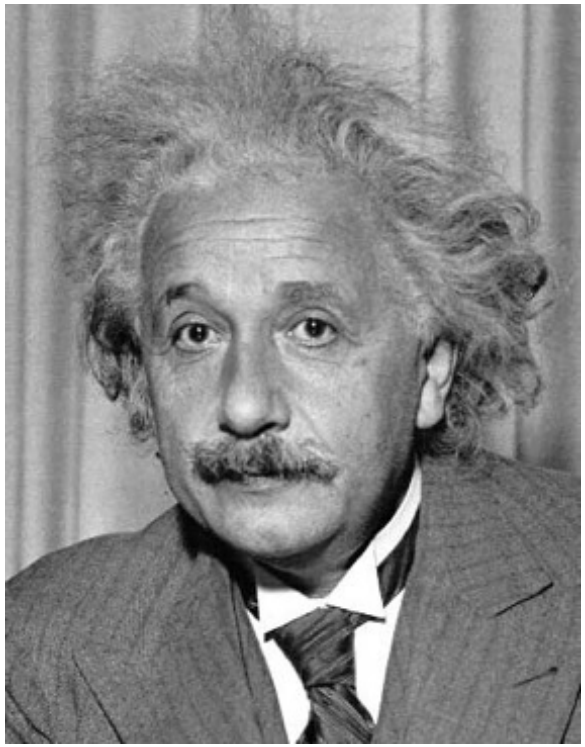
$$h(m,n) = \frac{\sum_{k,l}(g(k,l) - \bar{g})(f(m+k,n+l) - \bar{f}_{m,n})}{\sqrt{\sum_{k,l}(g(k,l) - \bar{g})^2 \sum_{k,l}(f(m+k,n+l) - \bar{f}_{m,n})^2}}$$

Dans matlab: `C = normxcorr2(template, A)`

Filtrer pour trouver les correspondances

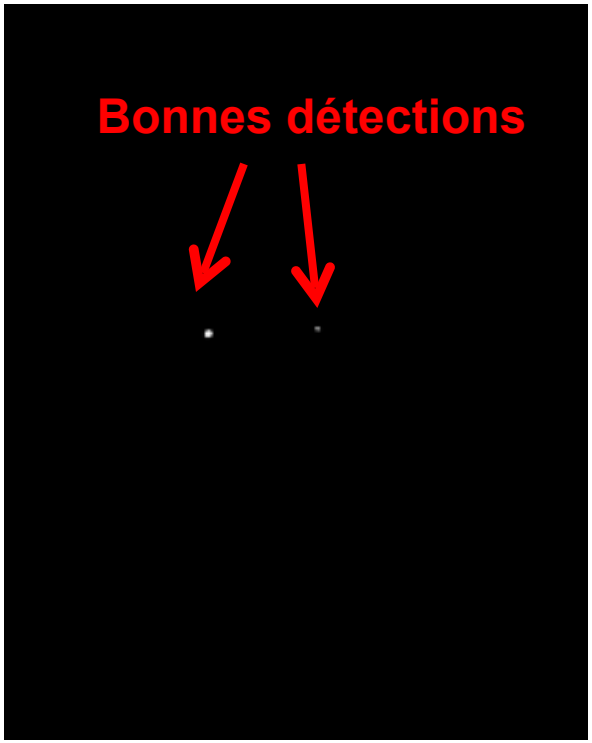
But: trouver  dans l'image

Méthode 3: corrélation croisée normalisée



Image

résultat



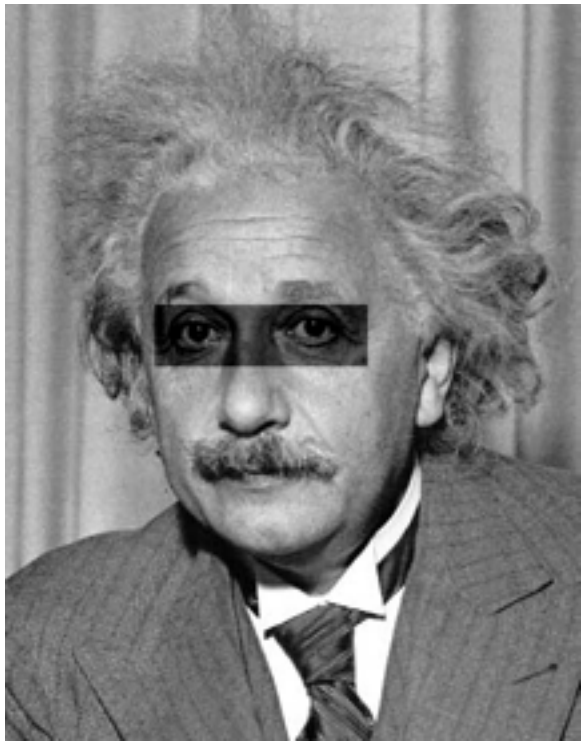
Seuil

Derek Hoiem

Filtrer pour trouver les correspondances

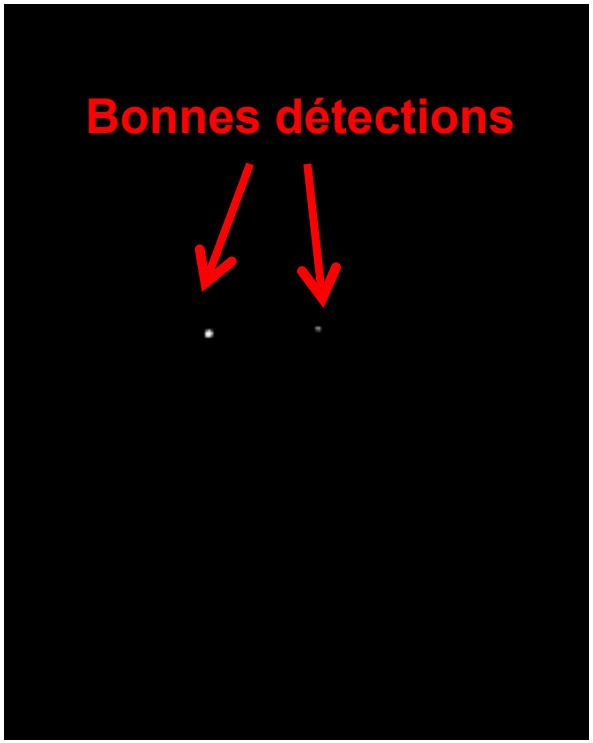
But: trouver  dans l'image

Méthode 3: corrélation croisée normalisée



Image

résultat



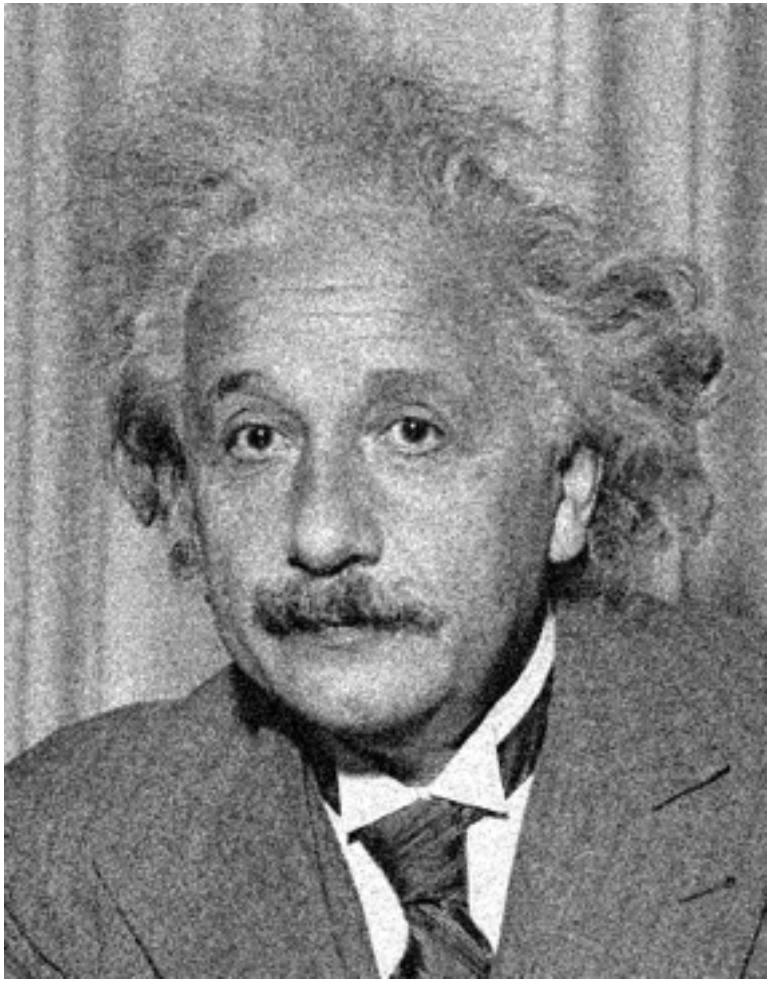
Seuil

Derek Hoiem

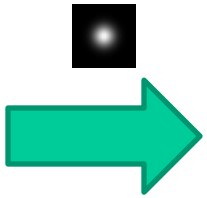
Quelle est la meilleure méthode?

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

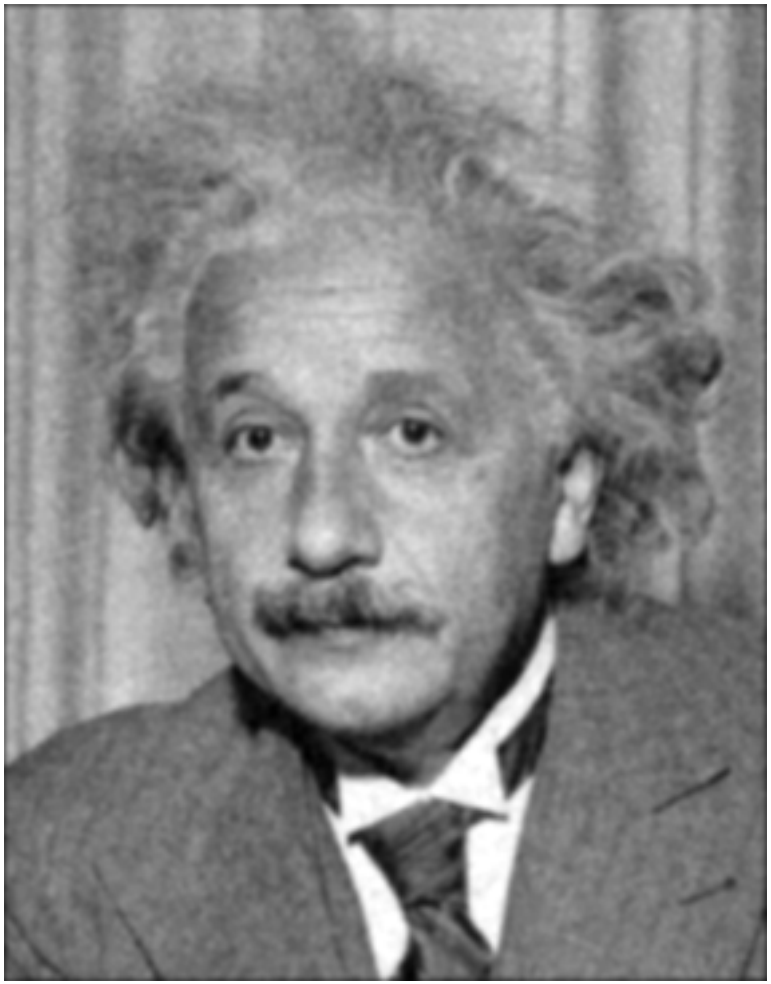
Atténuation du bruit



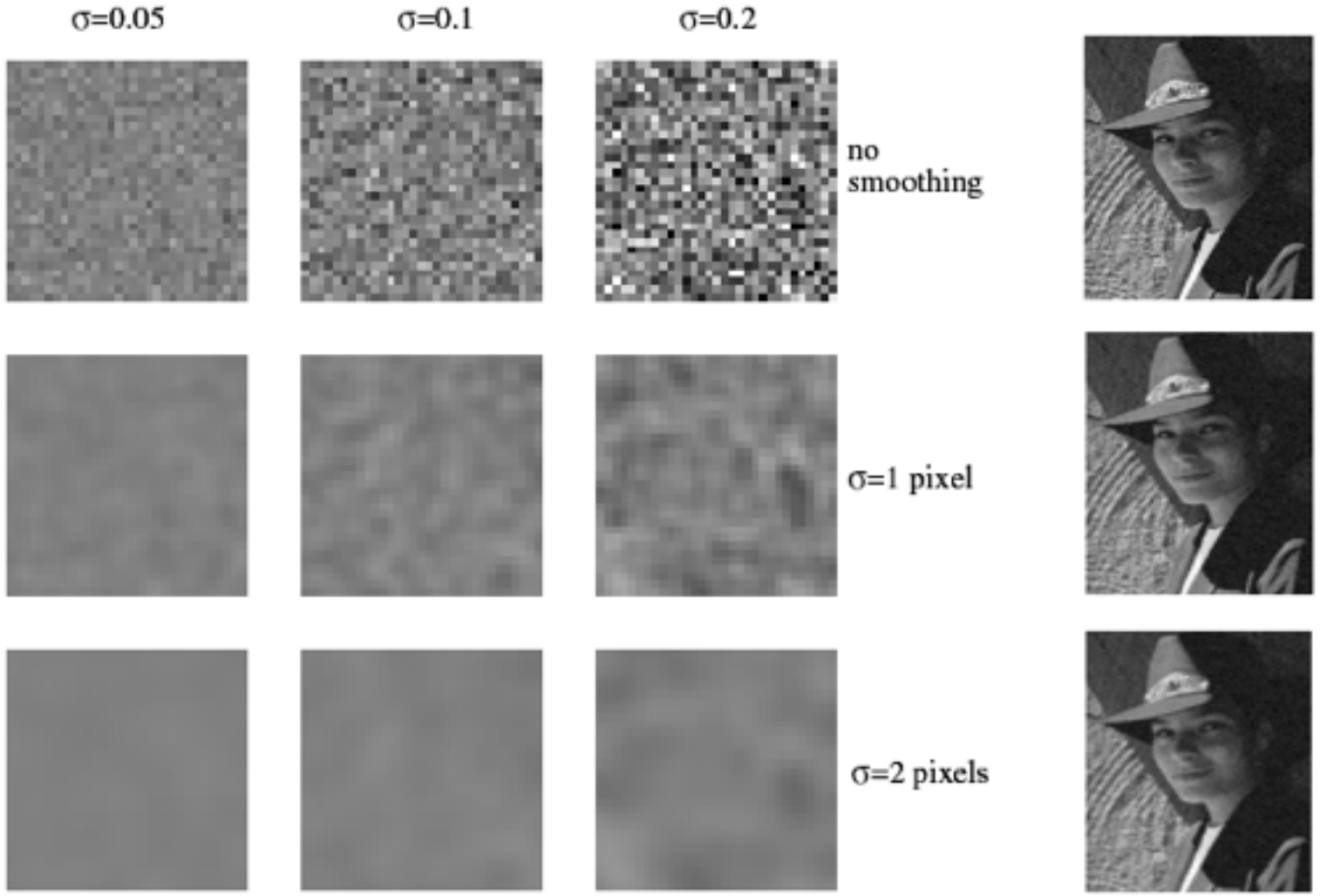
Bruit additif gaussien



Filtre gaussien



Atténuer le bruit gaussien



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

Source: S. Lazebnik

Bruit “poivre et sel”

Filtre gaussien

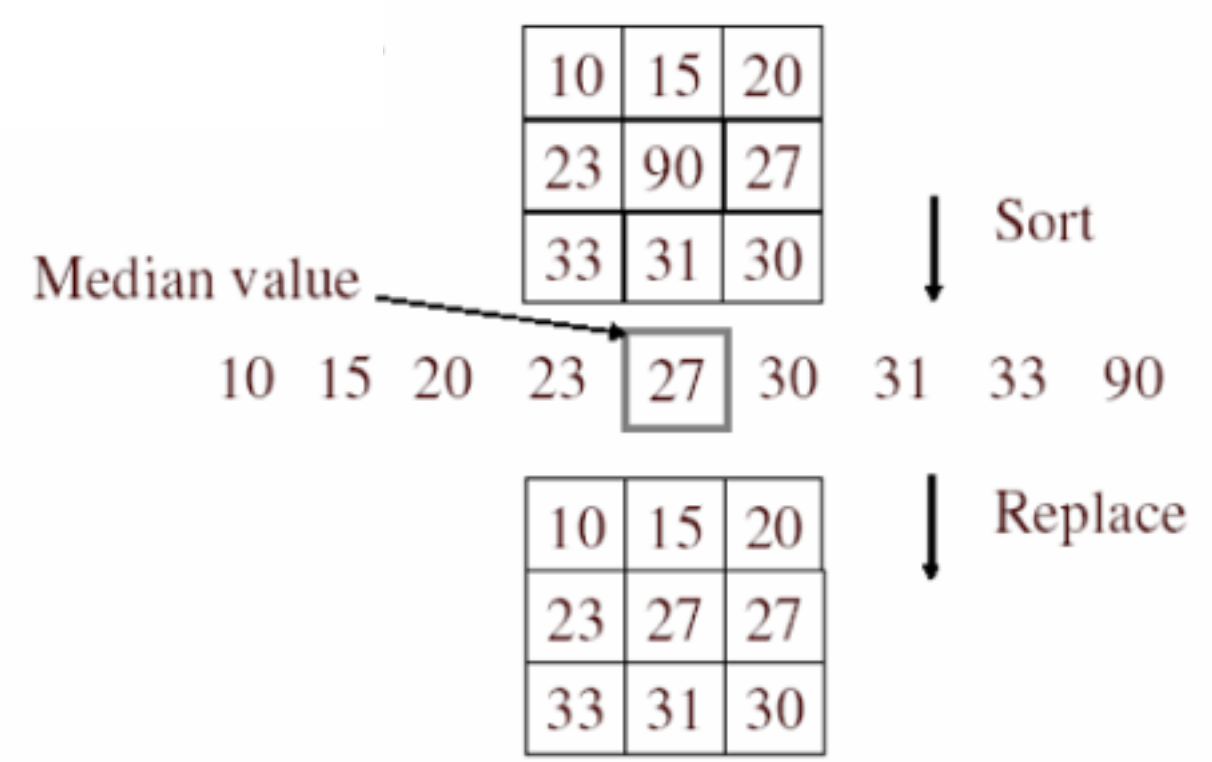
3x3

5x5

7x7

Idée alternative: filtre médian

Un filtre médian calcule la médiane de l'image sur une fenêtre

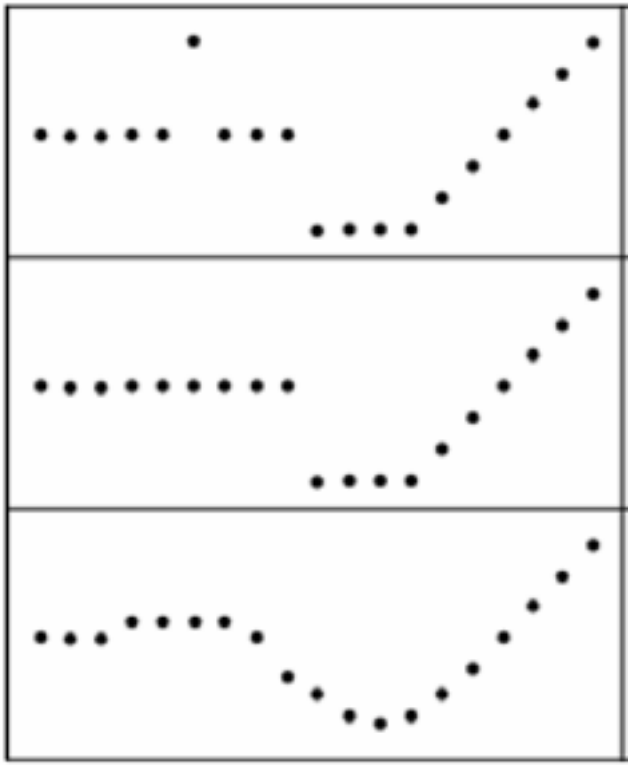


- Est-ce que c'est linéaire?

Source: K. Grauman

Filtre médian

Quels sont les avantages du filtre médian sur le filtre gaussien?



Signal

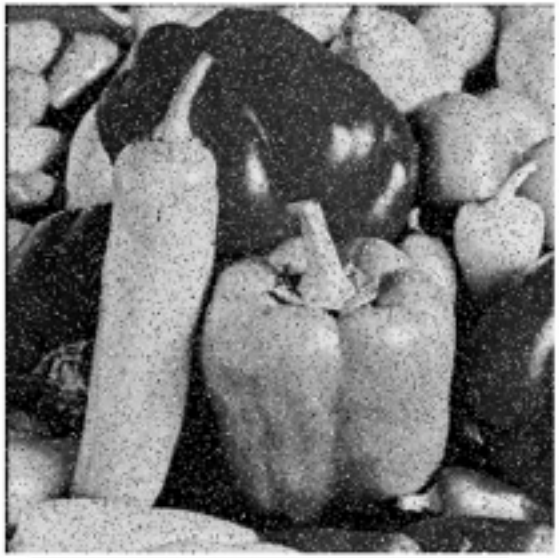
Filtre médian

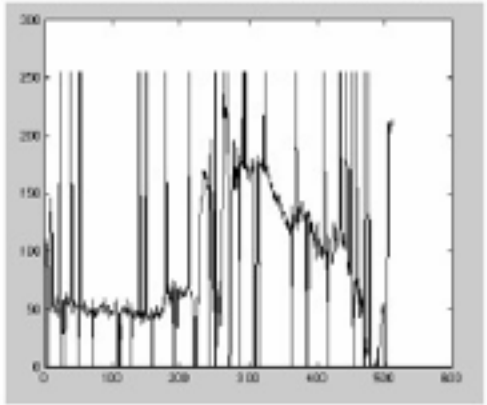
Filtre gaussien

Source: K. Grauman

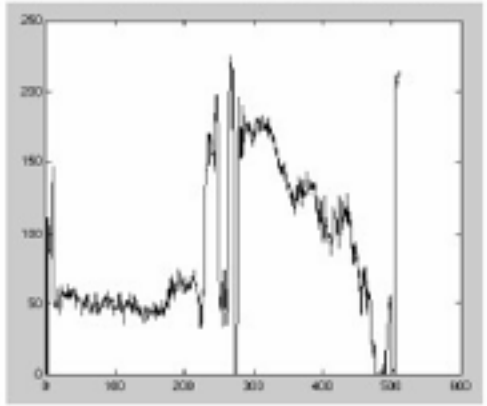
Filtre médian

Bruit “poivre et sel”





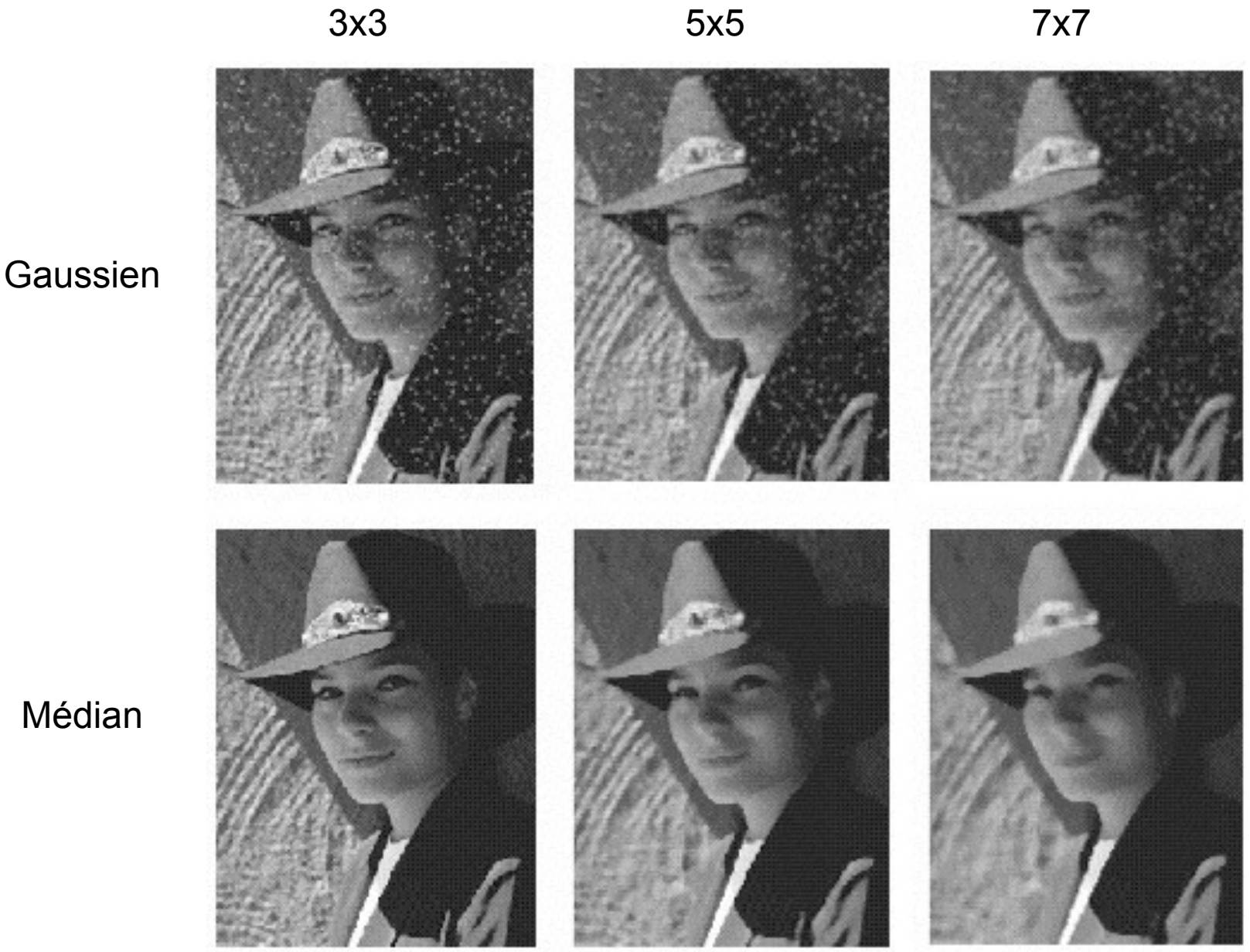
Filtre médian



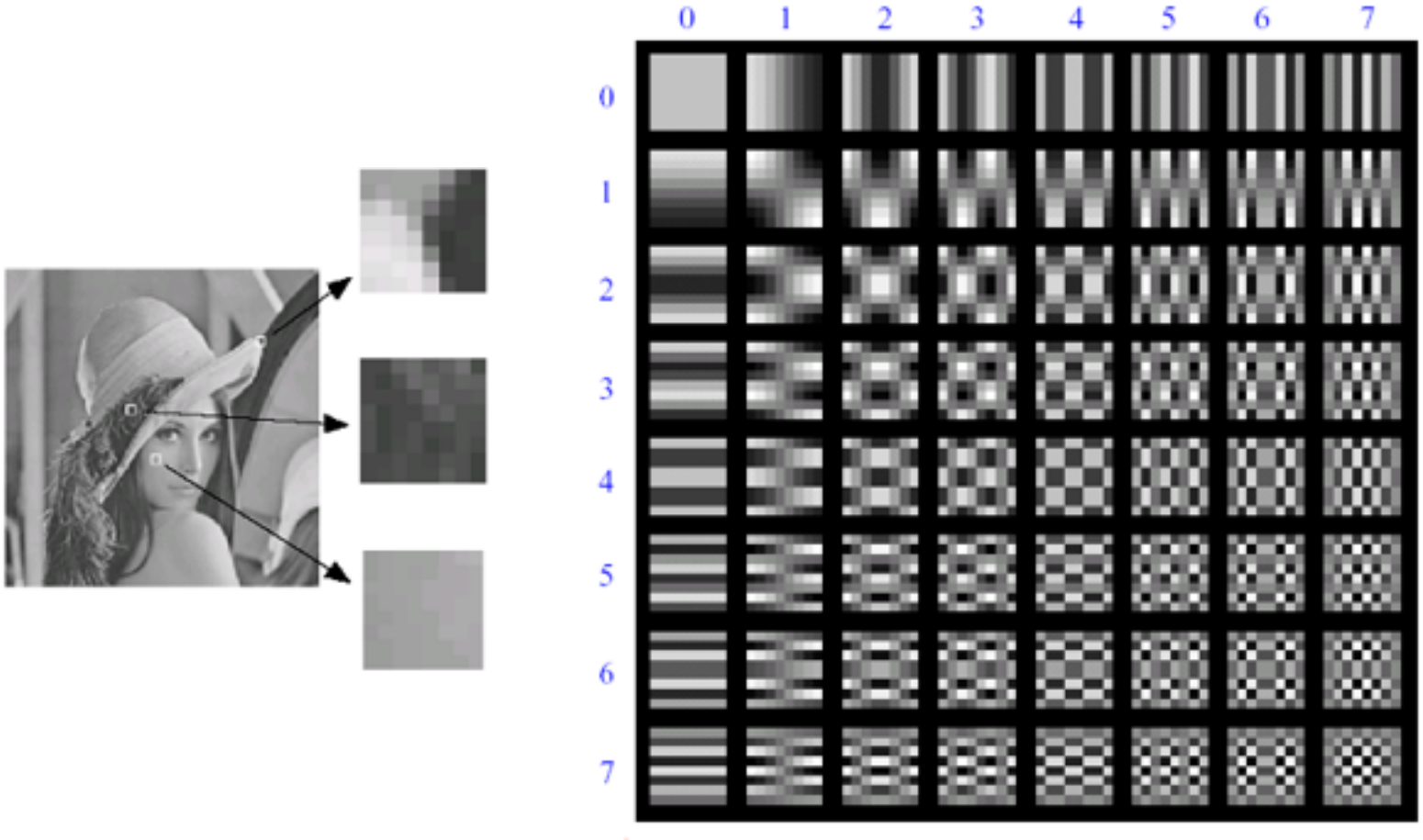
MATLAB: `medfilt2(image, [h w])`

Source: M. Hebert

Filtre Médian vs. gaussien



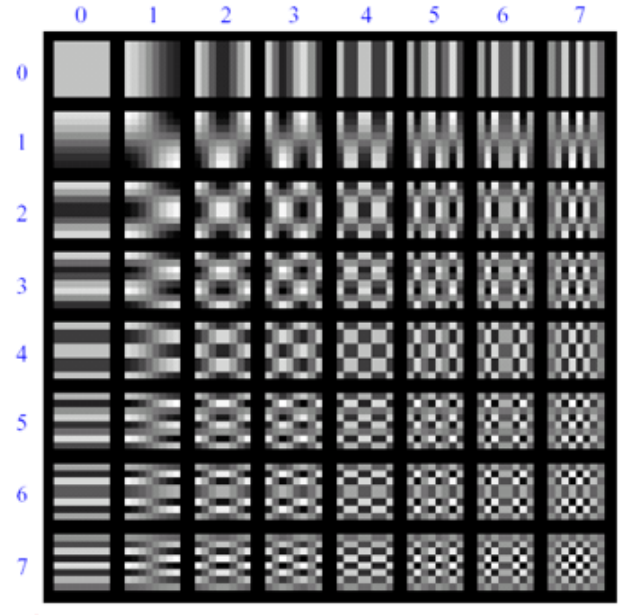
Compression (JPEG)

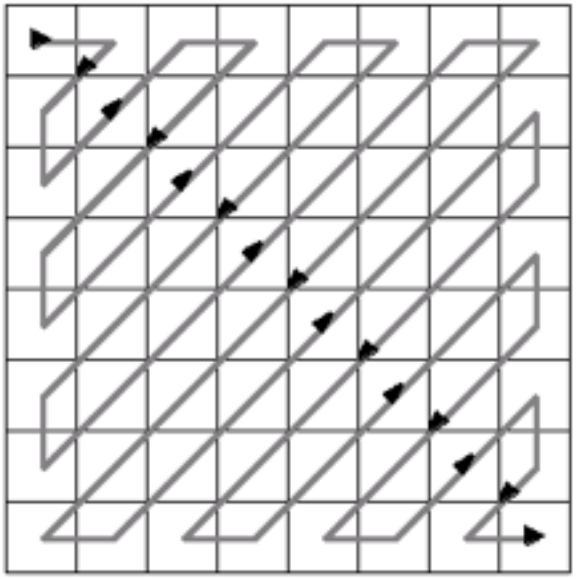


La DCT dans la compression JPEG

Le premier coefficient $B(0,0)$ est la composante DC (l'intensité moyenne)

Les coefficients en haut à gauche représentent les basses fréquences, et en bas à droite les hautes





La DCT dans la compression JPEG

Quantification

- Plus approximatif pour les hautes fréquences (qui sont plus faibles de façon naturelle)
- Plusieurs d'entre elles seront 0!

Encodage

- Décodage avec la DCT inverse

Réponse des filtres

$$G = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

($u \rightarrow$, $v \downarrow$)

Valeurs quantifiées

$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Table de quantification

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

Compression JPG

Diviser l'image en blocs (8x8), enlever 128

Pour chaque bloc

a. Calculer les coefficients DCT
b. Quantification
   - Coefficients des hautes fréquences deviendront 0
c. Encodage (e.g., avec l'encodage Huffman)

http://en.wikipedia.org/wiki/YCbCr
http://en.wikipedia.org/wiki/JPEG

Taille des blocs

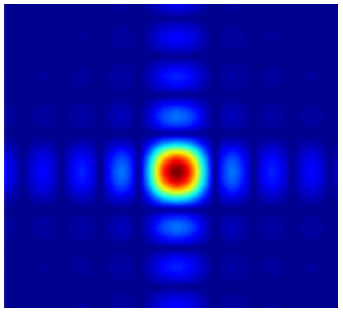
- petit
  - rapide!
  - corrélation existe entre blocs adjacents (compression moins efficace)
- grand
  - meilleure compression
- 8x8 dans le standard JPEG

Comparaison

89k

12k

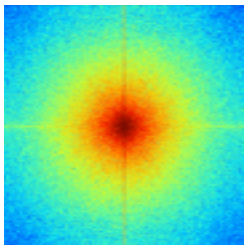
À retenir



- Souvent plus intuitif de penser en termes de fréquences
  - transformée de Fourier

- Plus rapide de filtrer avec la FFT pour les grosses images (N logN vs. N^2)

- Les images ont plus d'énergie dans les basses fréquences
  - Compression?

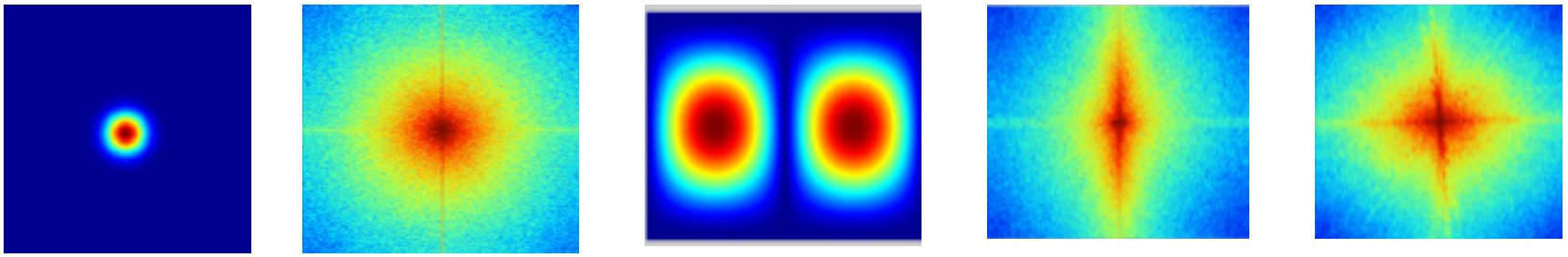


- Souvenez-vous de filtrer avant d'échantillonner

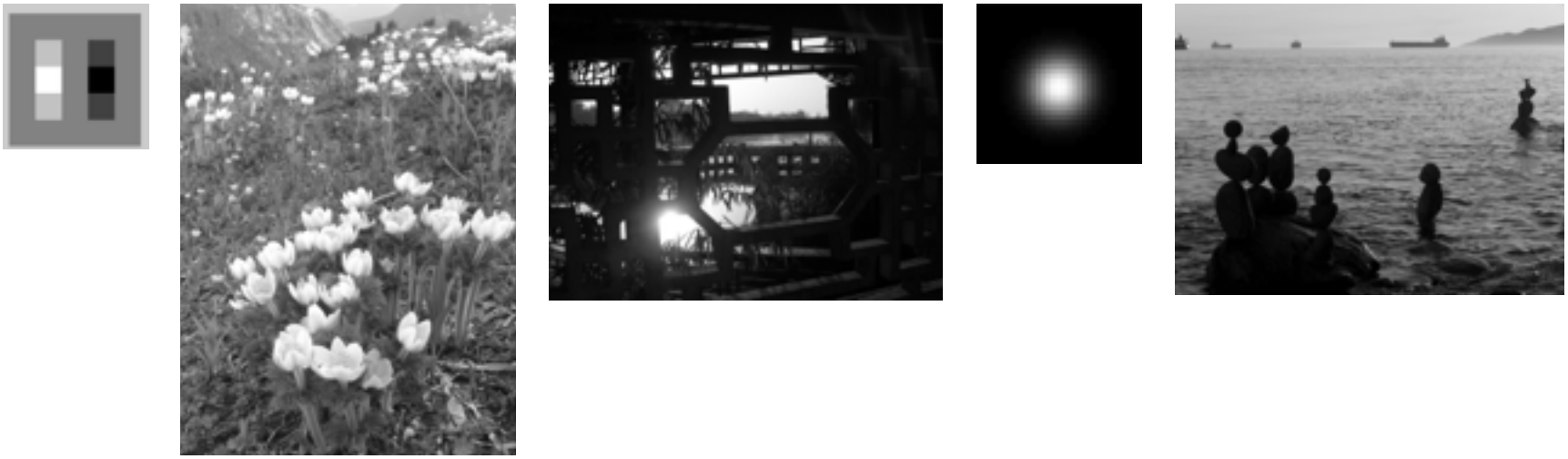
Question à emporter

1. Associez l'image à la transformée de Fourier

1 2 3 4 5



A B C D E

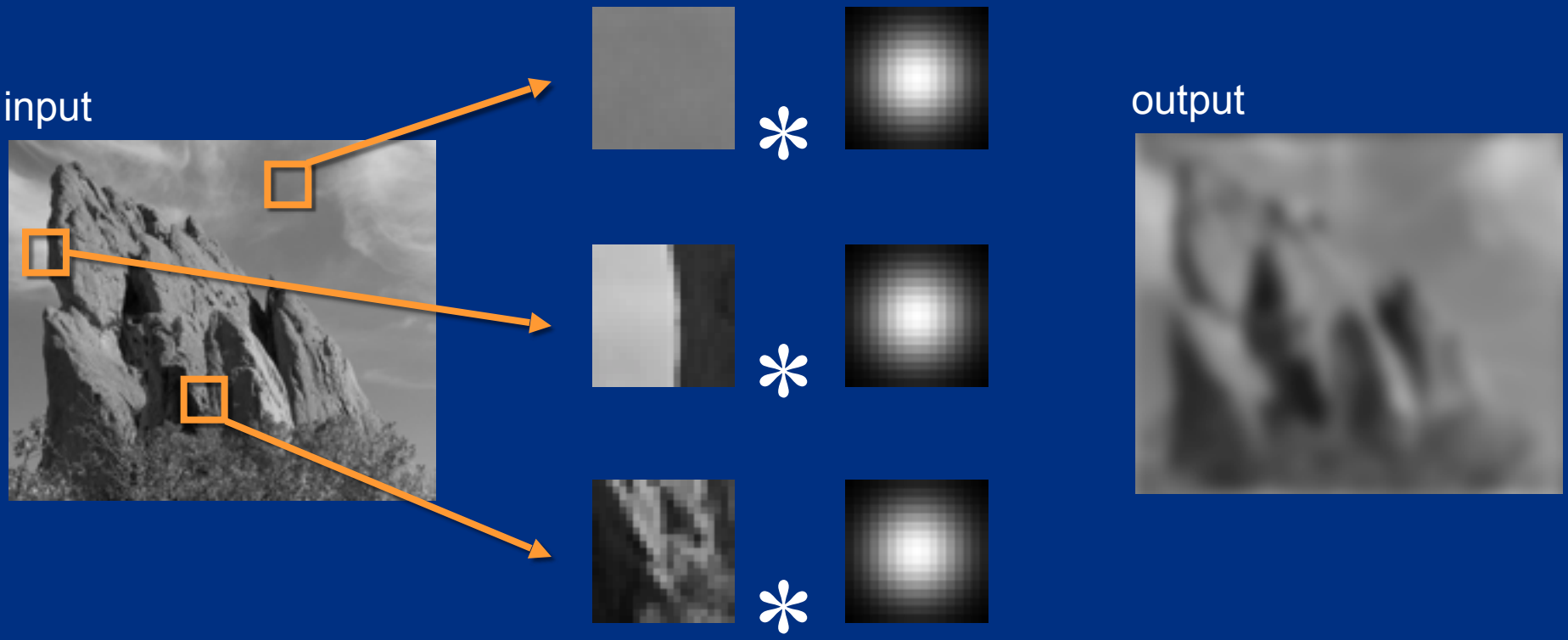


A Gentle Introduction
to Bilateral Filtering
and its Applications

“Fixing the Gaussian Blur”: the Bilateral Filter

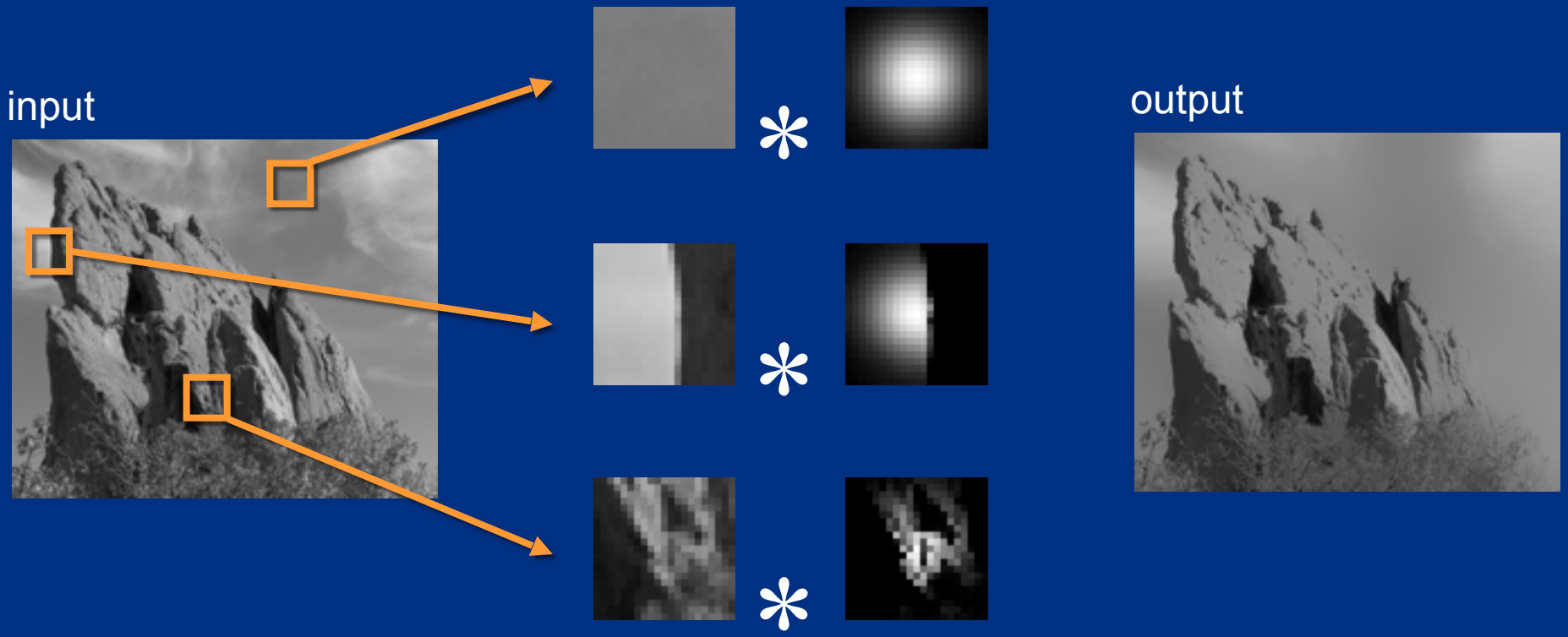
Sylvain Paris – MIT CSAIL

Blur Comes from Averaging across Edges



Same Gaussian kernel everywhere.

Bilateral Filter [Aurich 95, Smith 97, Tomasi 98]
No Averaging across Edges



The kernel shape depends on the image content.

Bilateral Filter Definition: an Additional Edge Term

Same idea: **weighted average of pixels**.

new

not new

new

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

normalization factor

space weight

range weight

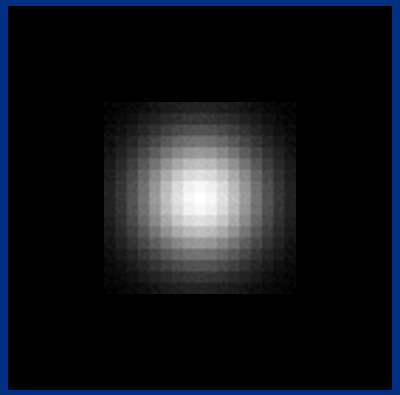
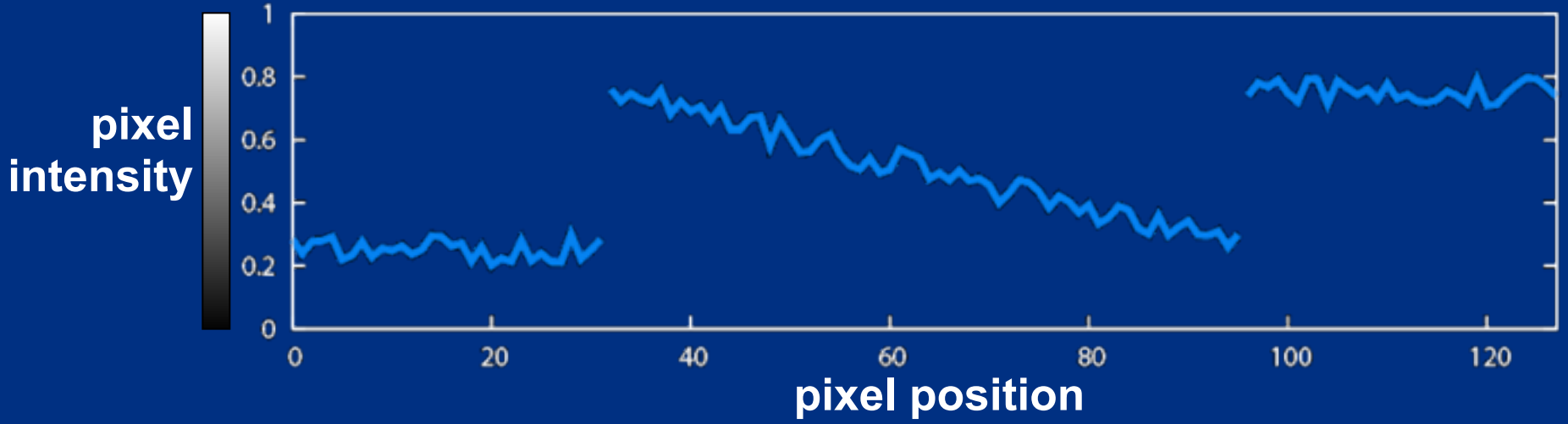


Illustration a 1D Image

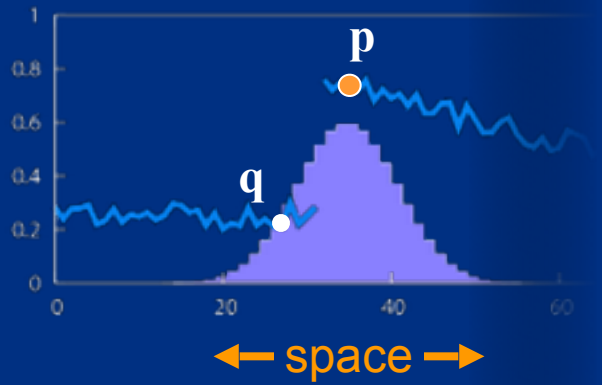
- 1D image = line of pixels

- Better visualized as a plot



Gaussian Blur and Bilateral Filter

Gaussian blur

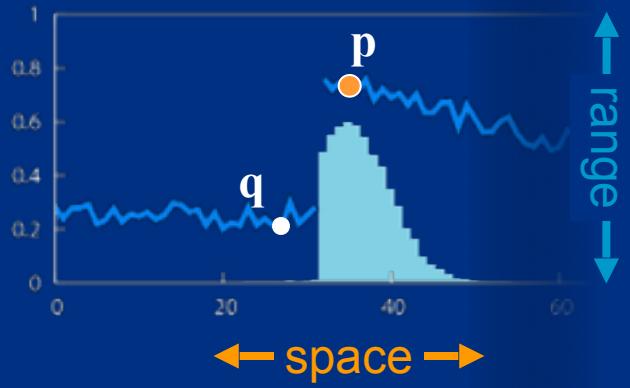


$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\sigma}(\|\mathbf{p}-\mathbf{q}\|) I_{\mathbf{q}}$$

space

Bilateral filter

[Aurich 95, Smith 97, Tomasi 98]



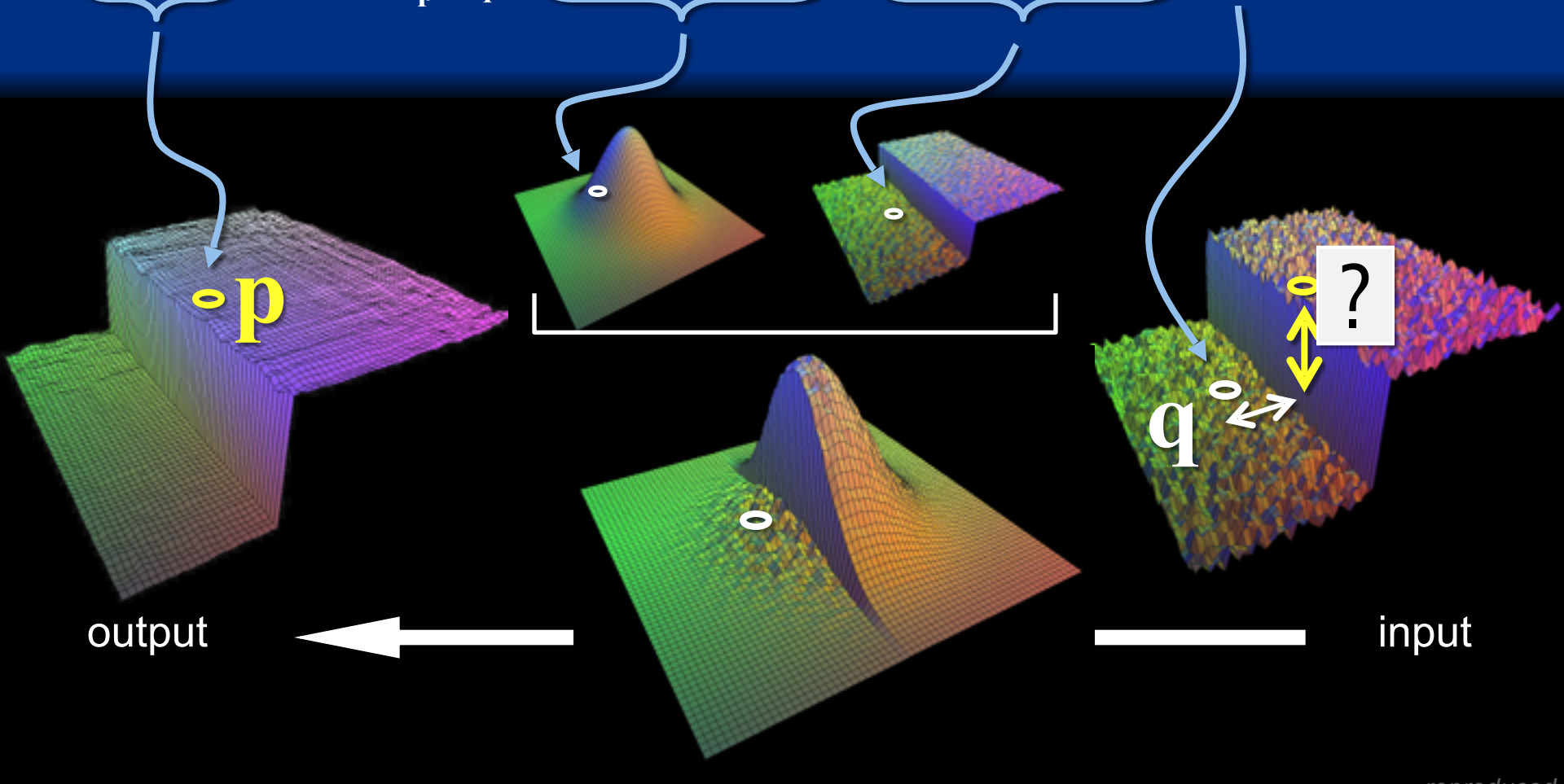
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|) I_{\mathbf{q}}$$

space range

normalization

Bilateral Filter on a Height Field

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



reproduced from [Durand 02]

Space and Range Parameters

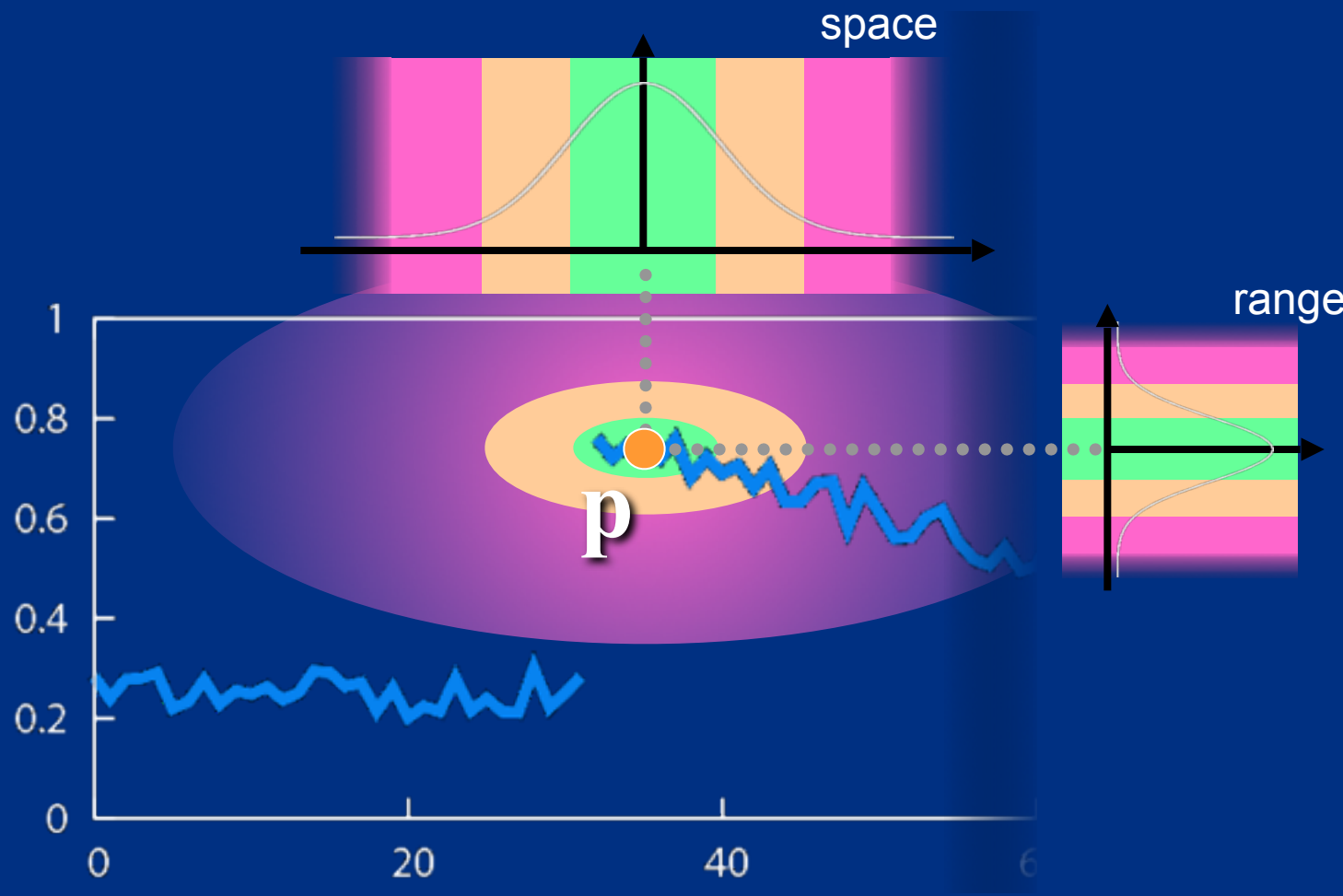
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|) I_{\mathbf{q}}$$

- space σ_s : spatial extent of the kernel, size of the considered neighborhood.

- range σ_r : “minimum” amplitude of an edge

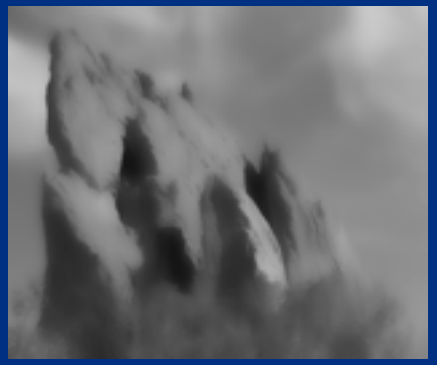
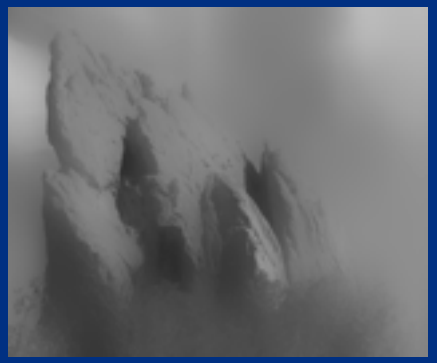
Influence of Pixels

Only pixels close in space and in range are considered.



Exploring the Parameter Space

input

	$\sigma_r = 0.1$	$\sigma_r = 0.25$	$\sigma_r = \infty$ (Gaussian blur)
$\sigma_s = 2$			
$\sigma_s = 6$			
$\sigma_s = 18$			

Varying the Range Parameter

input

$\sigma_r = 0.1$

$\sigma_r = 0.25$

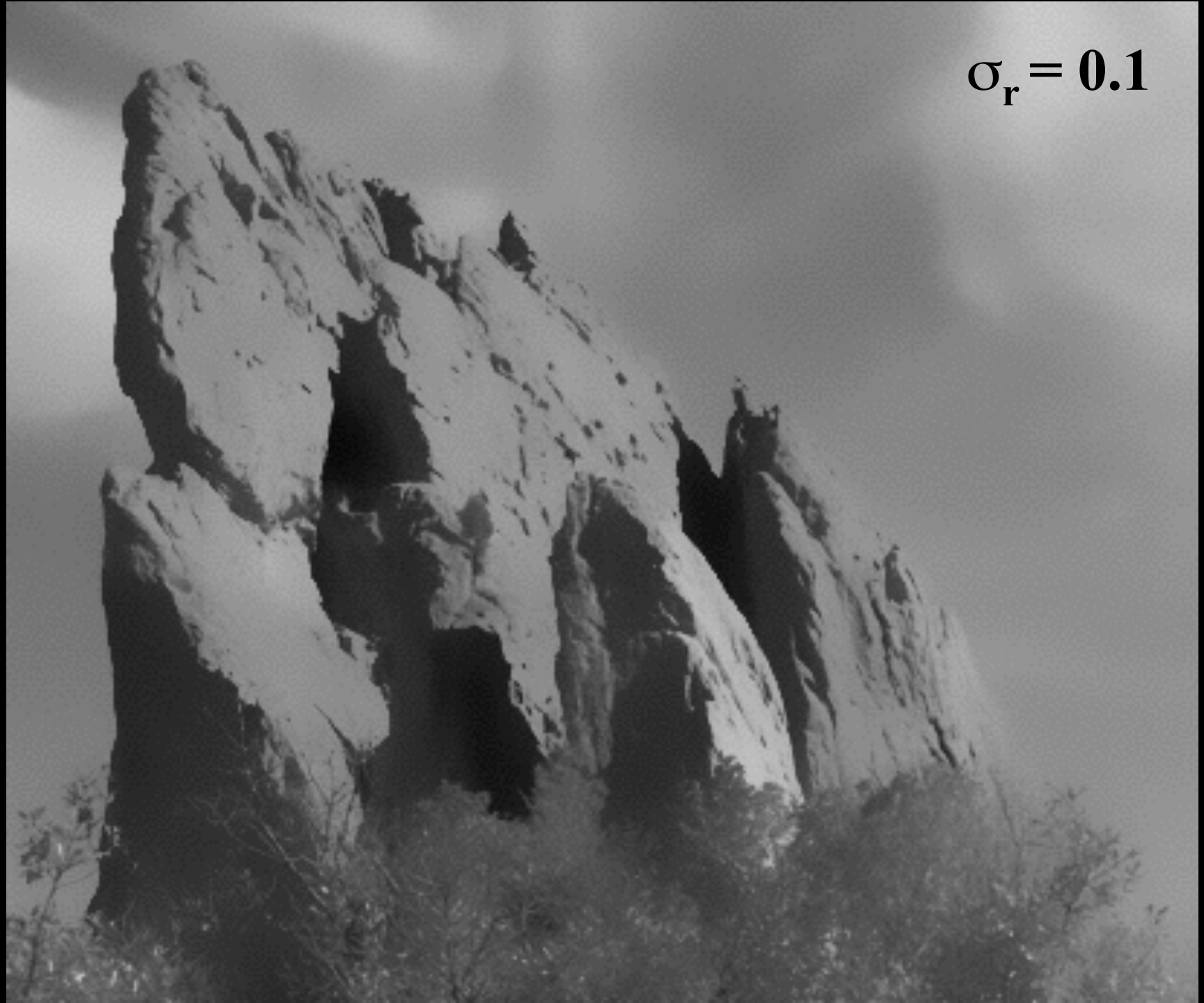
$\sigma_r = \infty$ (Gaussian blur)

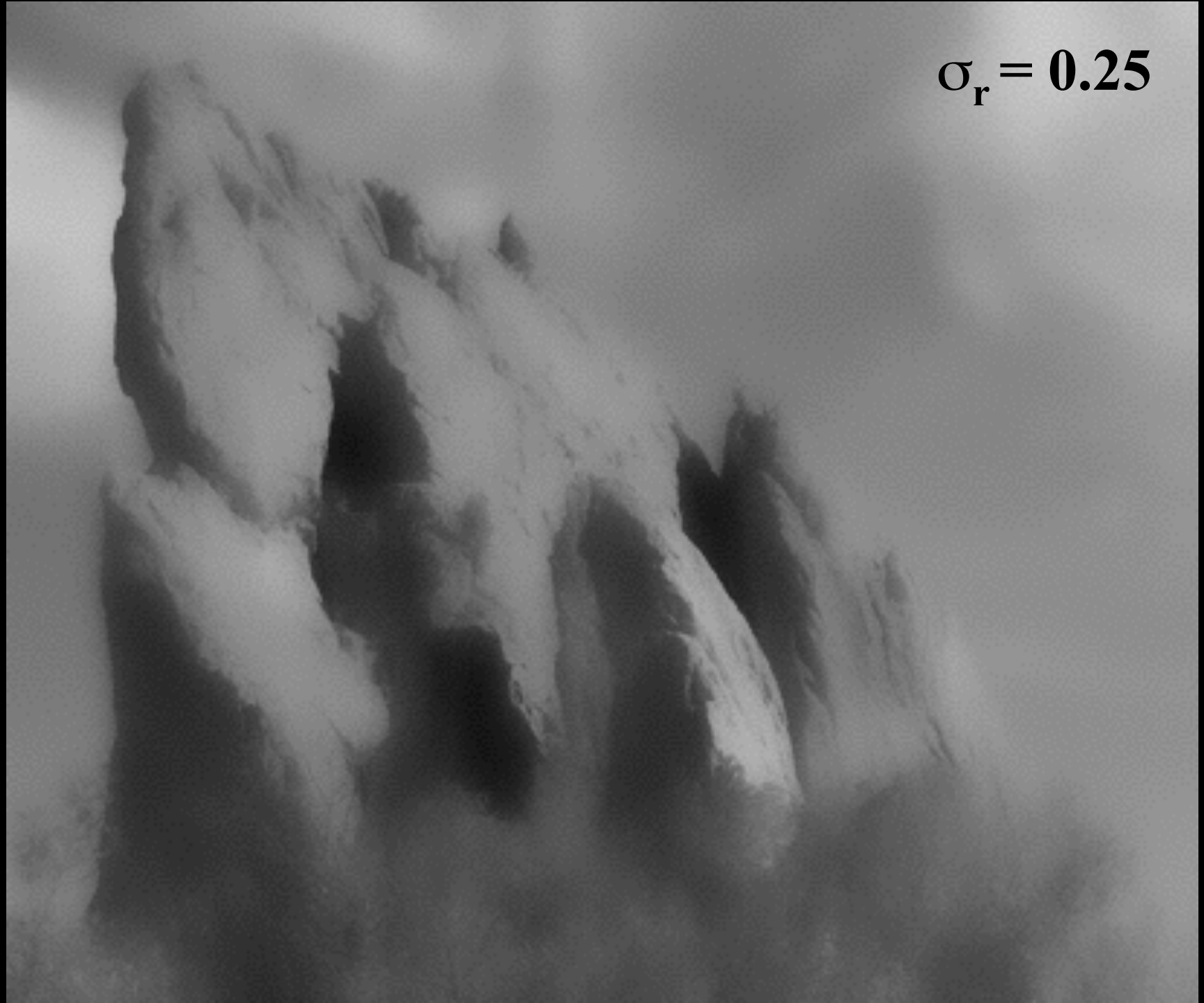
$\sigma_s = 2$

$\sigma_s = 6$

$\sigma_s = 18$


input


$\sigma_r = 0.1$


$\sigma_r = 0.25$


$\sigma_r = \infty$
(Gaussian blur)

Varying the Space Parameter

input

$\sigma_r = 0.1$

$\sigma_r = 0.25$

$\sigma_r = \infty$ (Gaussian blur)

$\sigma_s = 2$

$\sigma_s = 6$

$\sigma_s = 18$


input


$\sigma_s = 2$


$\sigma_s = 6$


$\sigma_s = 18$