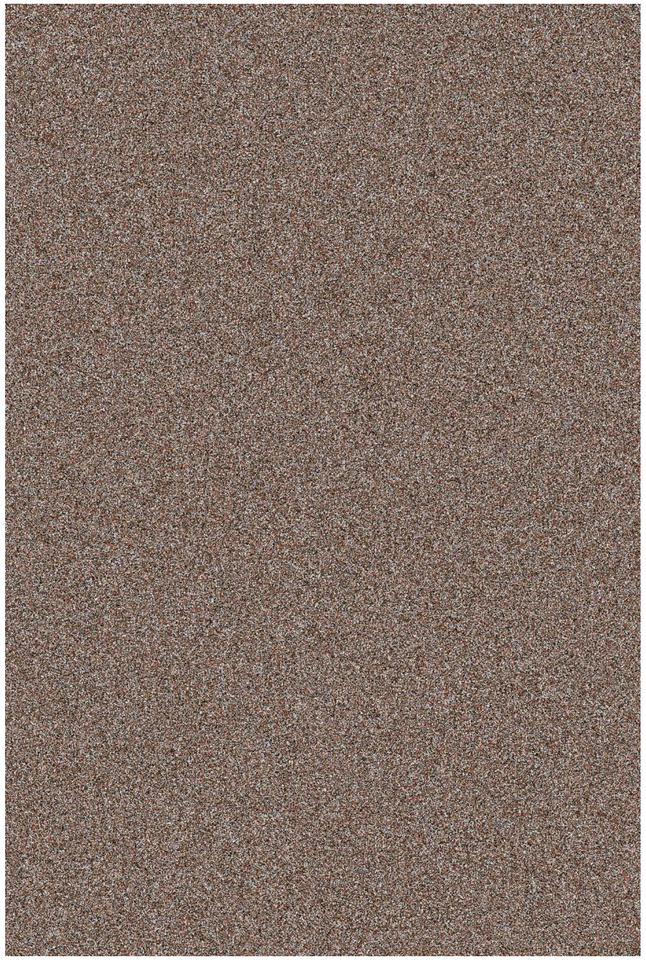
Filtrage dans le domaine spatial

GIF-4105/7105 Photographie Algorithmique
Jean-François Lalonde

Image: “henriksn”, deviantart

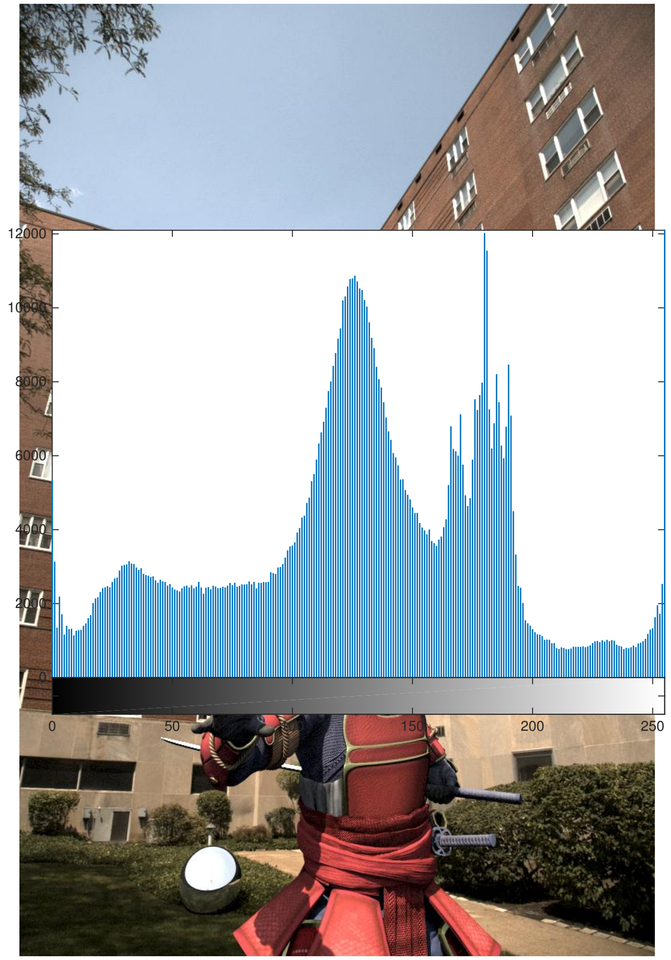
Limites des fonctions sur les pixels

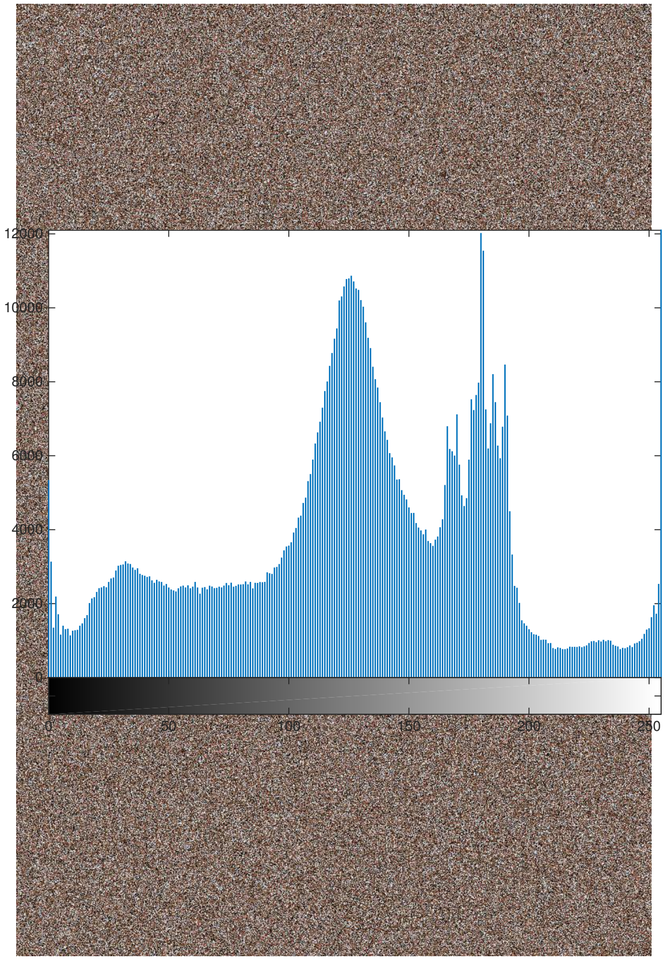
Qu’arrive-t-il à l’histogramme d’une image
lorsque les pixels sont ré-ordonnés?



Limites des fonctions sur les pixels

Qu’arrive-t-il à l’histogramme d’une image
lorsque les pixels sont ré-ordonnés?





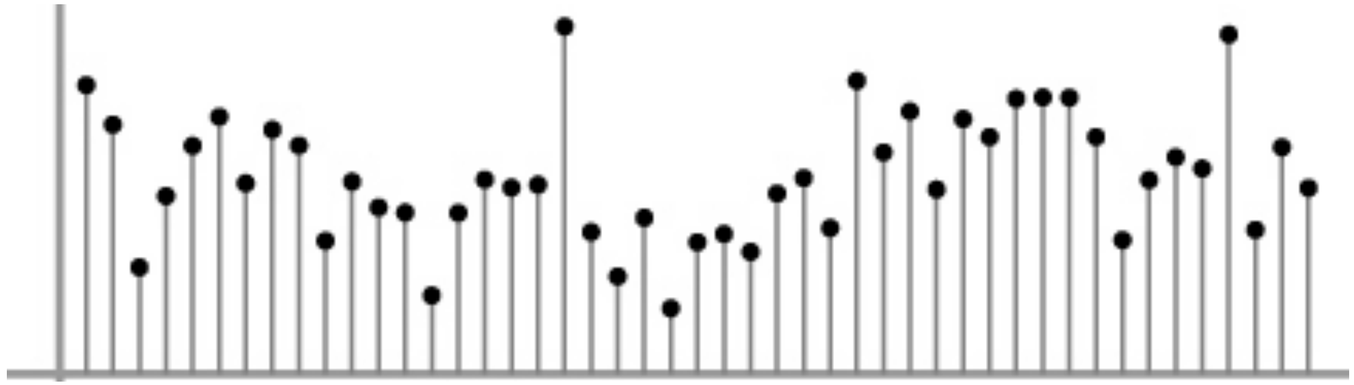
Filtrage d’images

- Filtrage : fonction d’un pixel **et de ses voisins**
- Très important, plusieurs applications!
  - Modifier l’image
    - Réduire le bruit, re-dimensionner, contraste, etc.
  - Extraire de l’information
    - Texture, arêtes, points d’intérêt, etc.
  - Detecter des formes
    - *Template matching*
  - Au coeur des réseaux de neurones à convolution (*Convolutional Neural Networks*, ou CNN)
    - convolution = filtrage!

Calculer la moyenne temporelle d'un signal 1D

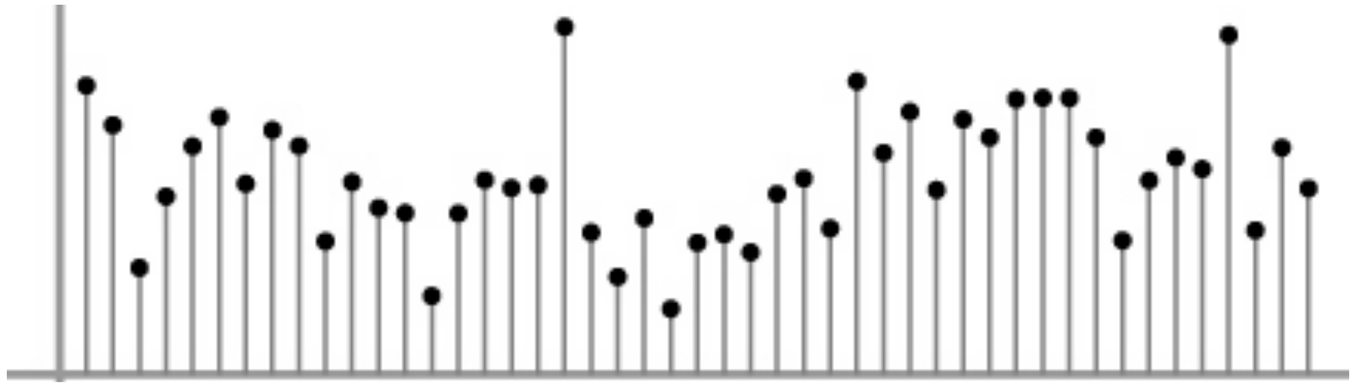
- Idée : pour un signal f,
  - définir un intervalle de temps « autour » d'un point
  - calculer la moyenne des voisins d'un point dans cet intervalle
  - stocker le résultat et déplacer l'intervalle au fil du temps

$f[\cdot]$



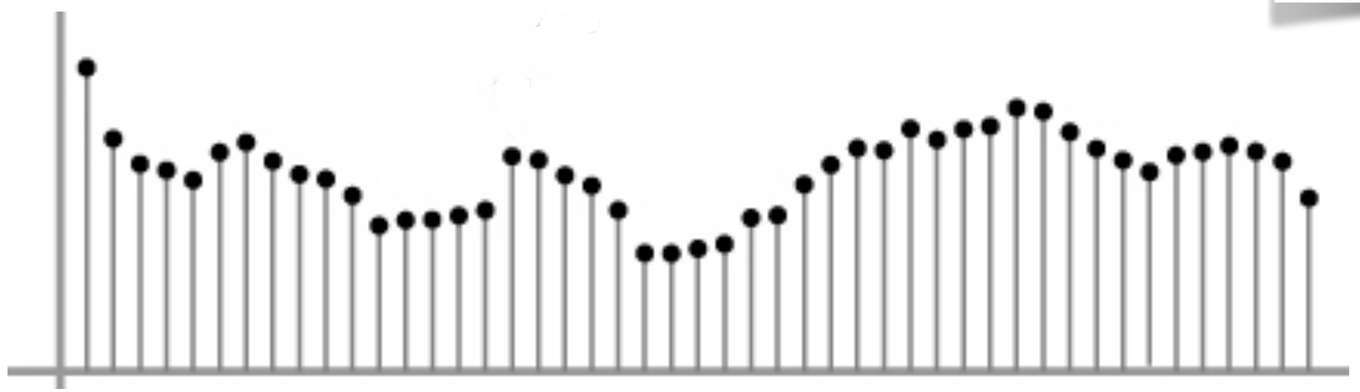
Calculer la moyenne temporelle d'un signal 1D

$f[\cdot]$



Quelle pondération a-t-on employée?

$h[\cdot]$



Calculer la moyenne temporelle d'un signal 1D

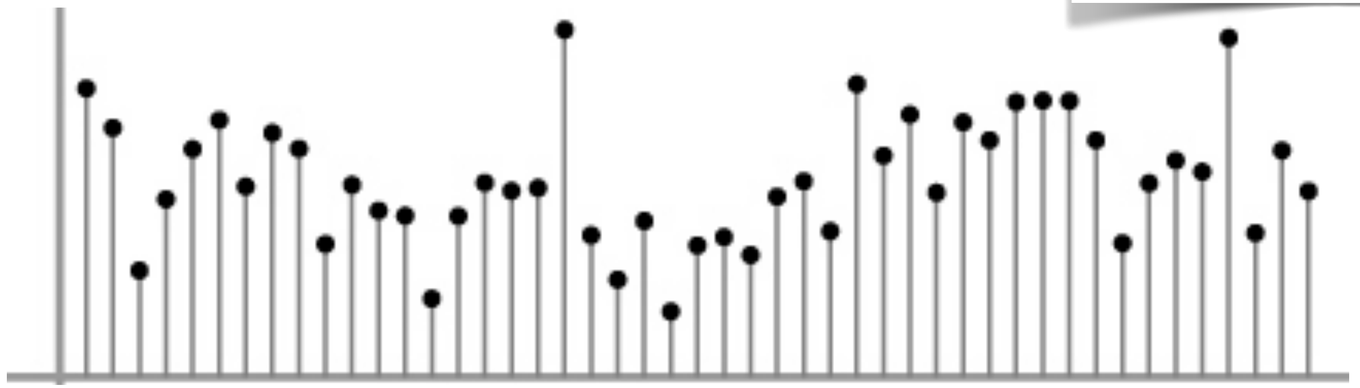
- « filtre » = pondération

$$g = \frac{1}{5}[1\ 1\ 1\ 1\ 1]$$

$$h[i] = \sum_{u=-k}^{k} g[u] f[i+u]$$

Le filtre g possède une dimension de $2k+1$

$f[\cdot]$



Cette opération se nomme la « corrélation croisée » (*cross-correlation*).

Produit scalaire entre le filtre g et le signal f à chaque emplacement sur le signal.

Corrélation croisée (en 2D)

Rappel : en 1-D

$$h[i] = \sum_{u=-k}^{k} g[u] f[i+u]$$

- Nous avons:
  - une image f
  - un filtre g, de taille $(2k+1) \times (2k+1)$
- h est le résultat de

$$h[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} g[u,v] f[i+u, j+v]$$

Corrélation vs. convolution?

En pratique : presque pareil!

- La **corrélation croisée** (*cross-correlation*) est une mesure de similarité entre deux signaux

$$h[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} g[u,v] f[i+u, j+v]$$

- La **convolution** est un opérateur qui « applique » un signal (filtre) à un autre (image). Mathématiquement, cela équivaut à une corrélation croisée où le filtre est inversé (horizontalement et verticalement) avant d'être appliqué à l'image :

$$h[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} g[u,v] f[i-u, j-v]$$

- On l'écrit:

$$h = g * f$$

Source: S. Seitz

Filtre « en boîte » (*box filter*)

$g[\cdot, \cdot]$

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

Source: D. Lowe

Filtrage

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

$$h[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} g[u,v] f[i+u, j+v]$$

Source: S. Seitz

Filtrage

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

	0								

$$h[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} g[u,v] f[i+u, j+v]$$

Source: S. Seitz

Filtrage

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

	0	10							

$$h[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} g[u,v] f[i+u, j+v]$$

Source: S. Seitz

Filtrage

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

	0	10	20						

$$h[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} g[u,v] f[i+u, j+v]$$

Source: S. Seitz

Filtrage

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

	0	10	20	30					
				?					

$$h[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} g[u,v] f[i+u, j+v]$$

Source: S. Seitz

Filtrage

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

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	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

	0	10	20	30					

Qu'est-ce qui se passe, intuitivement?

$$h[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} g[u,v] f[i+u, j+v]$$

Source: S. Seitz

Filtrage

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	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	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Source: S. Seitz

Filtrage “en boîte” (box filter)

- Remplace chaque pixel par la moyenne de son voisinage
- On adoucit l’image (enlève les hautes fréquences)

$$g[\cdot,\cdot]$$

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

Source: D. Lowe

“Atténuer” l’image avec le filtre boîte

Petite pratique avec les filtres linéaires

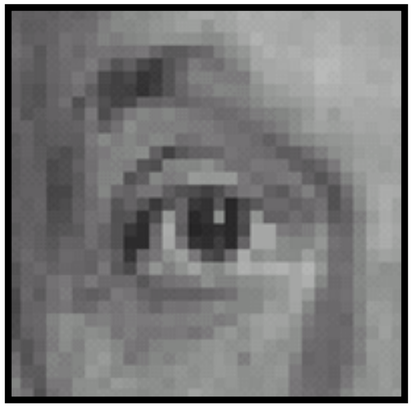


Image originale

0	0	0
0	1	0
0	0	0

?

Source: D. Lowe

Petite pratique avec les filtres linéaires

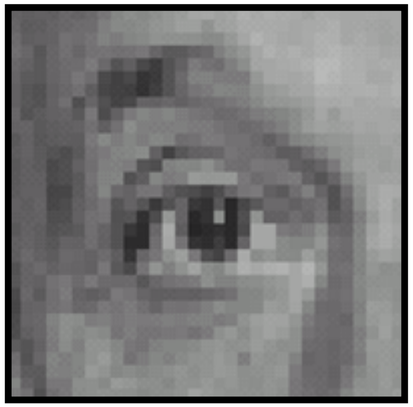
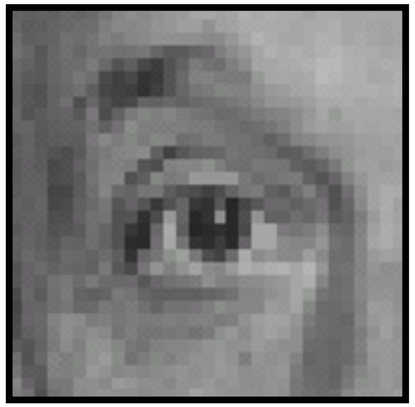


Image originale

0	0	0
0	1	0
0	0	0



Résultat
(identique!)

Source: D. Lowe

Petite pratique avec les filtres linéaires

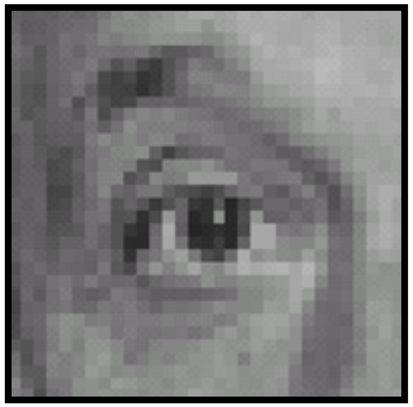


Image originale

0	0	0
0	0	1
0	0	0

?

Source: D. Lowe

Petite pratique avec les filtres linéaires

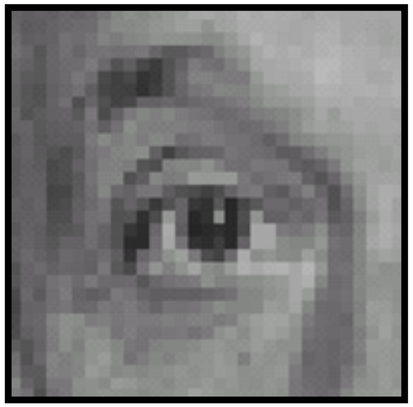
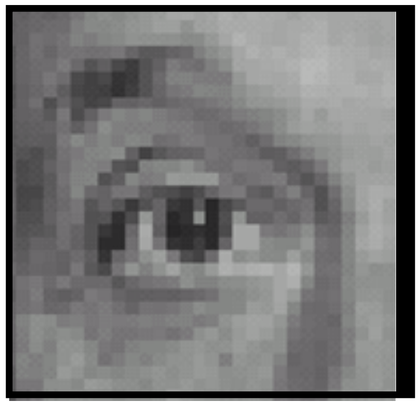


Image originale

0	0	0
0	0	1
0	0	0



À gauche de 1 pixel

Source: D. Lowe

Petite pratique avec les filtres linéaires

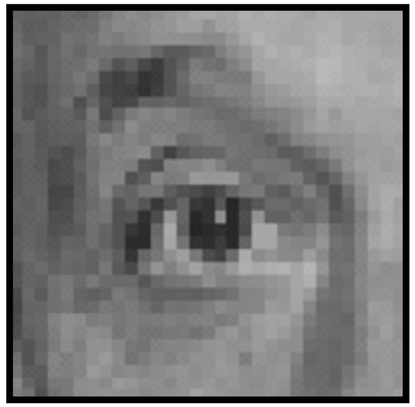


Image originale

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad ?$$

Source: D. Lowe

Petite pratique avec les filtres linéaires

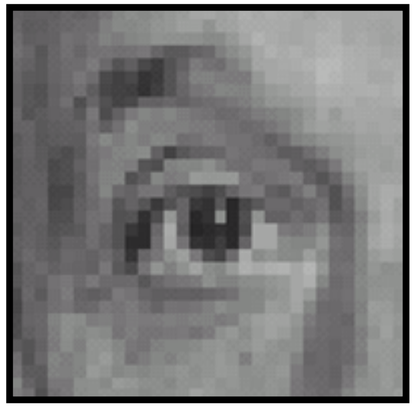


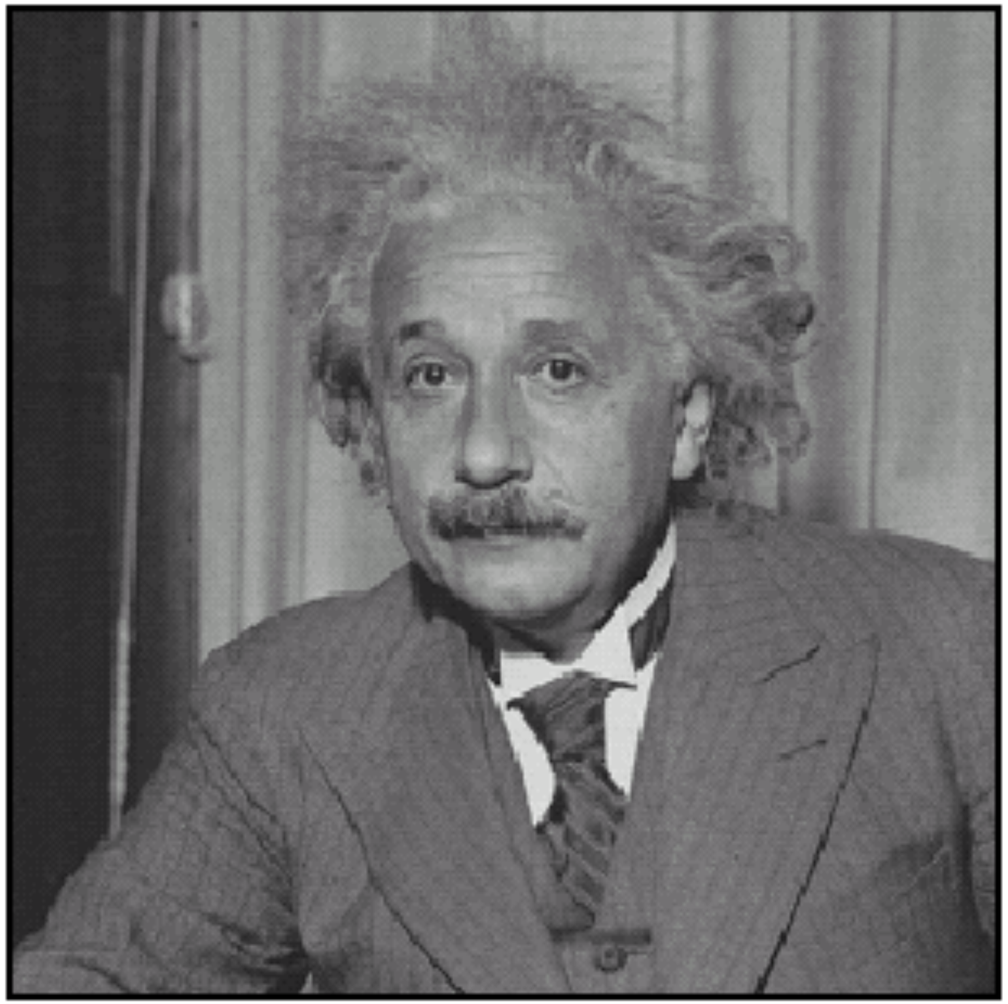
Image originale

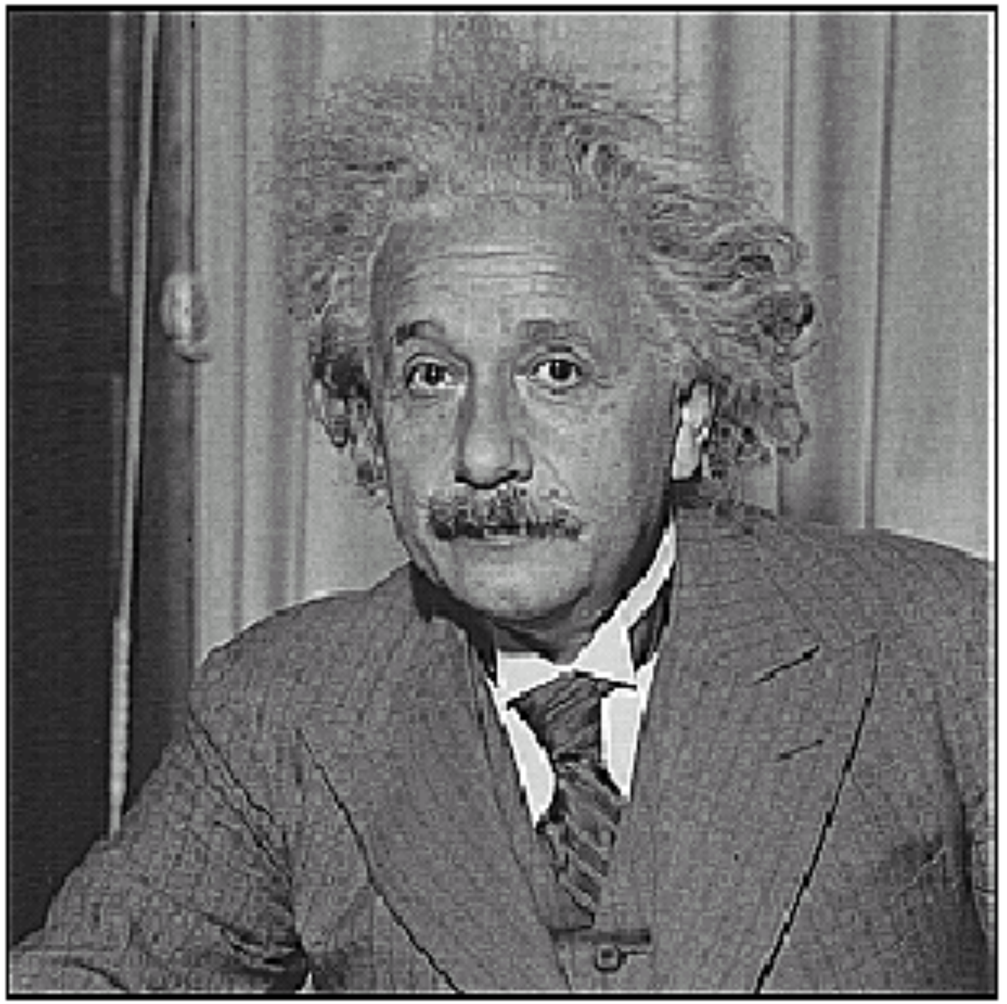
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Accentue les différences par rapport à la moyenne

Source: D. Lowe

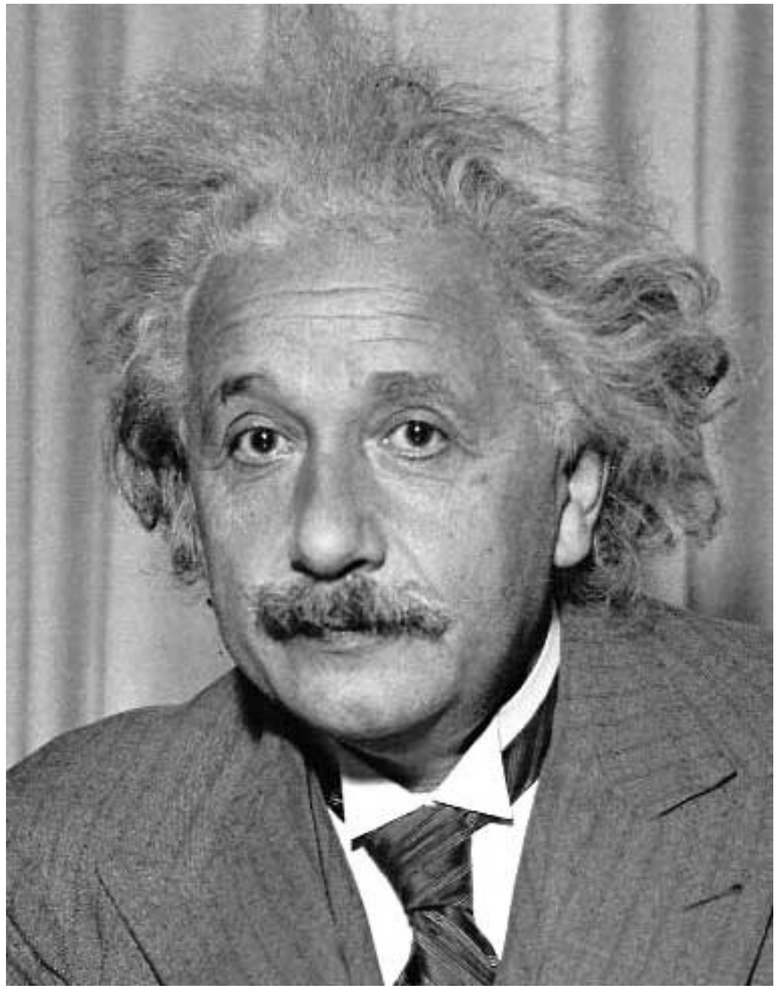
Accentuation “sharpening”





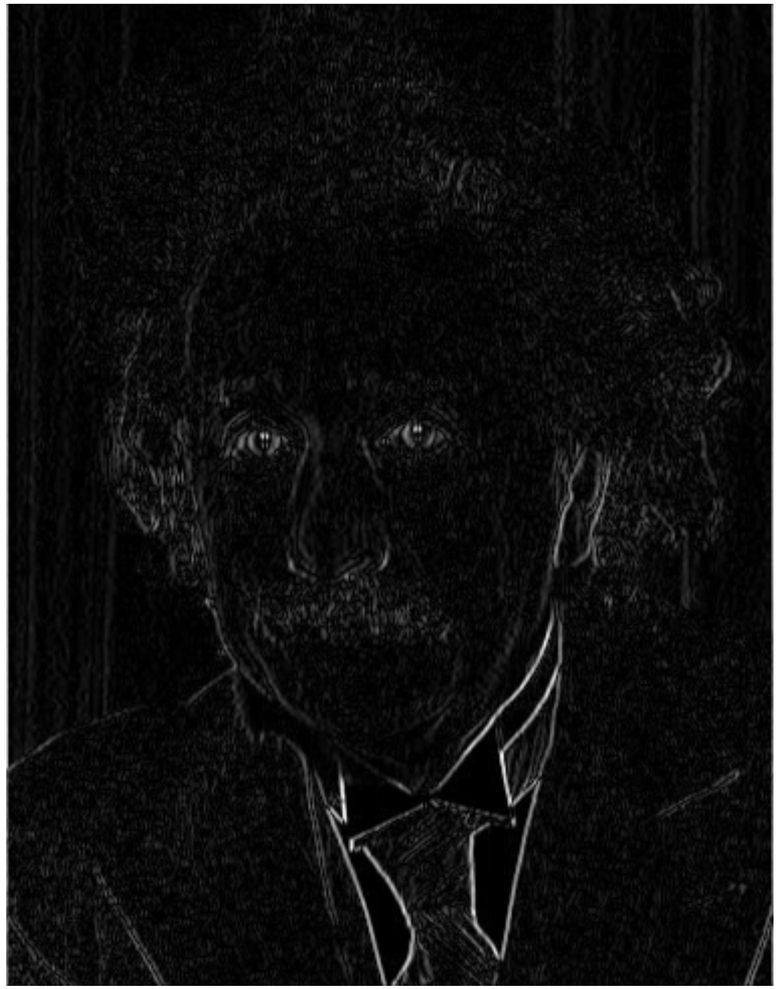
Source: D. Lowe

Autres filtres



1	0	-1
2	0	-2
1	0	-1

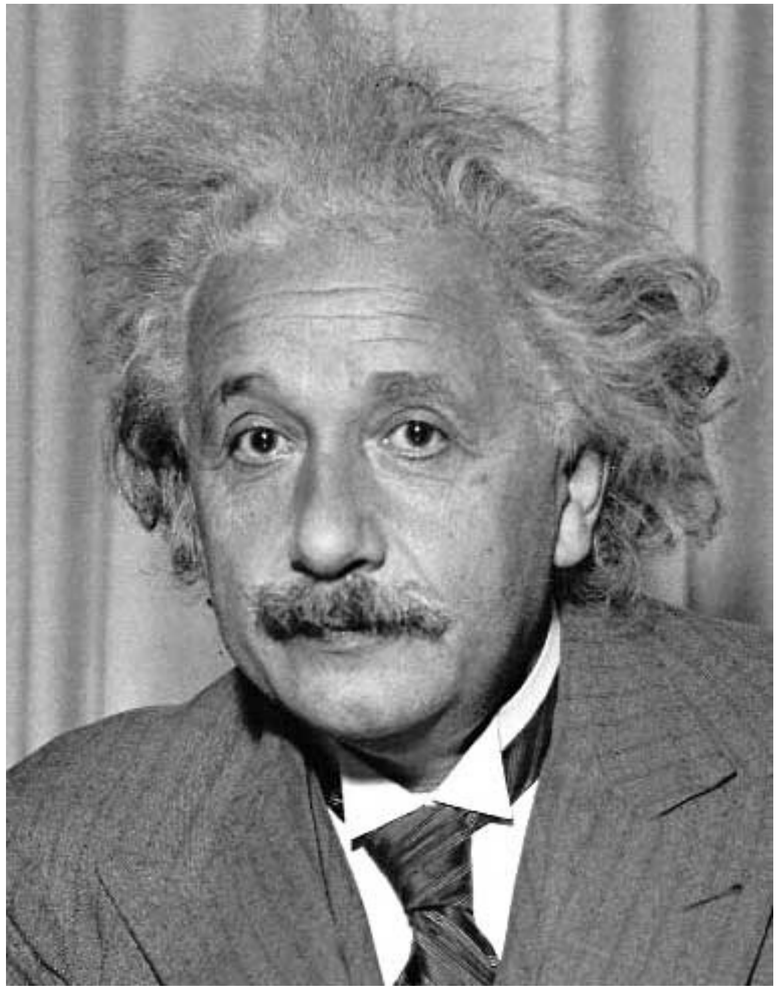
Sobel



Arêtes verticales
(valeur absolue)

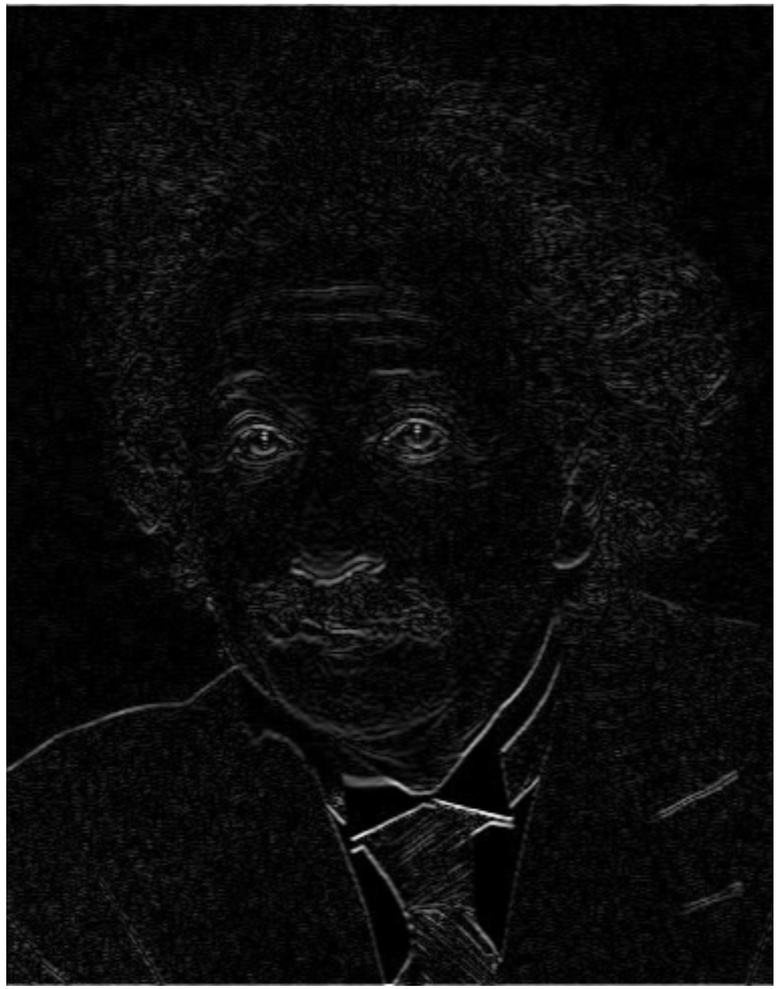
Source: D. Lowe

Autres filtres



1	2	1
0	0	0
-1	-2	-1

Sobel



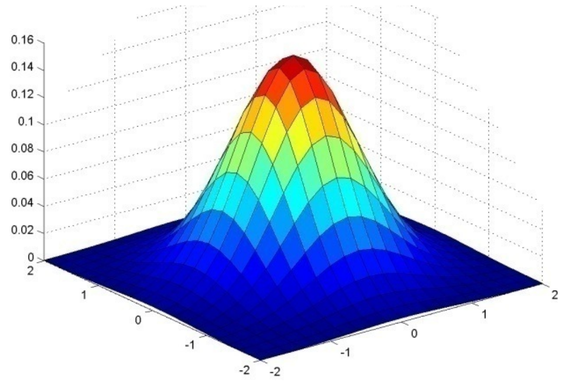
Arêtes horizontales
(valeur absolue)

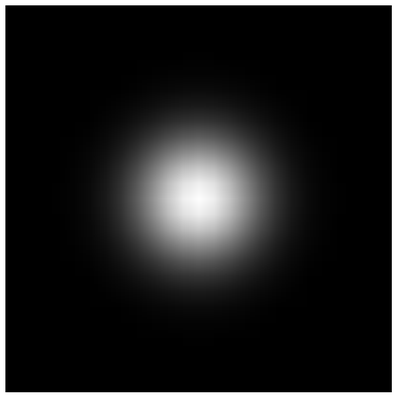
Source: D. Lowe

Filtre important : gaussien

Pondère les contributions des voisins en fonction de leur **distance**

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{\left(x^2+y^2\right)}{2\sigma^2}}$$



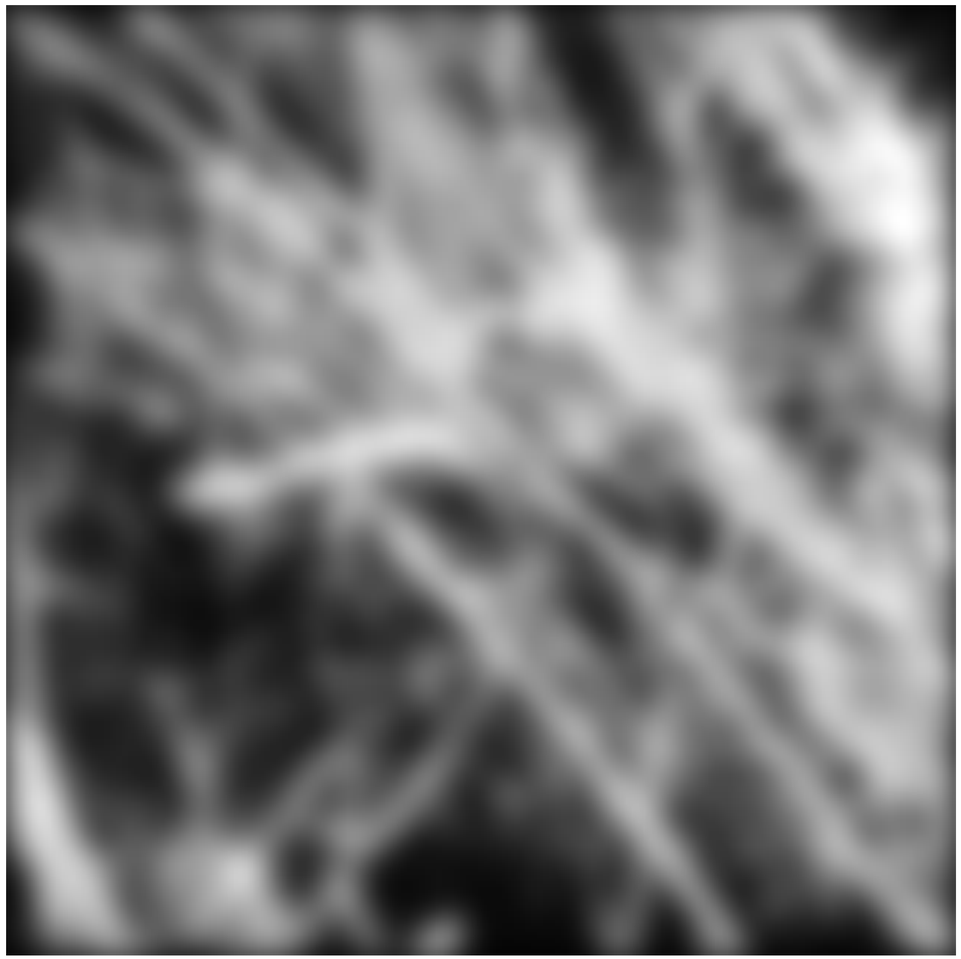


0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

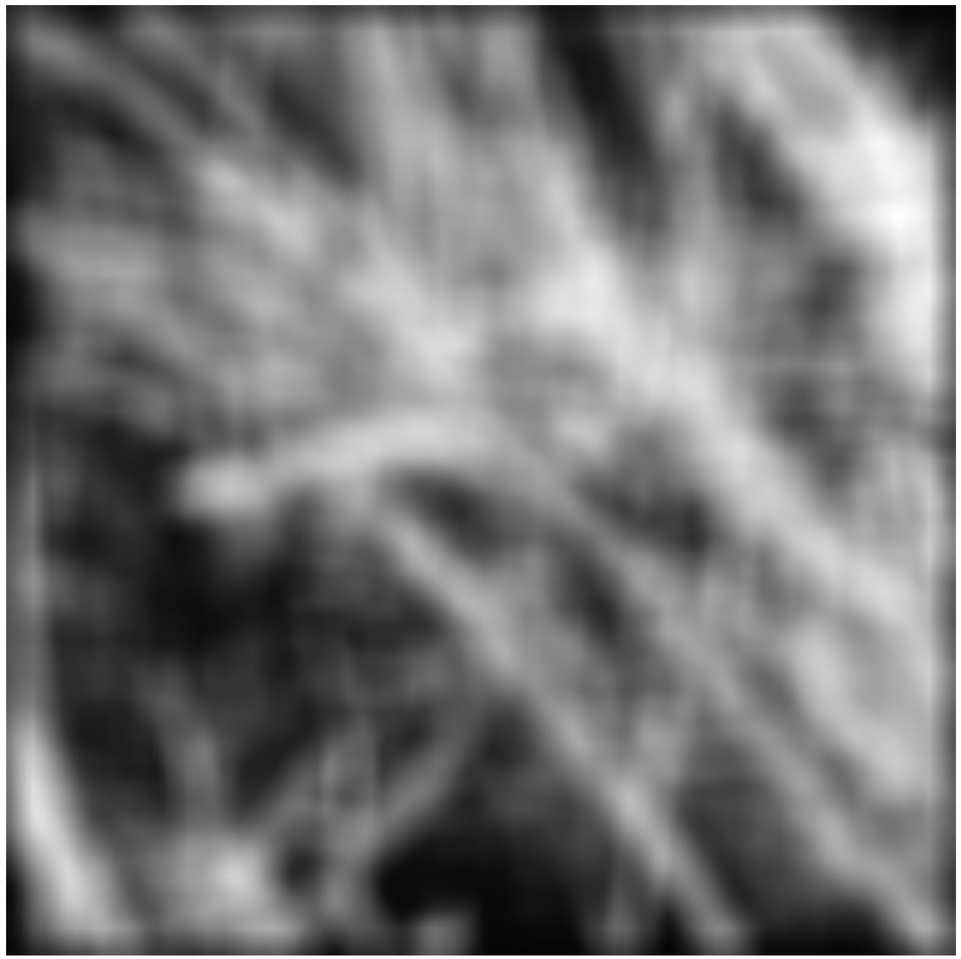
$5 \times 5,\ \sigma = 1$

Source: C. Rasmussen

Atténuer avec le filtre gaussien



Atténuer avec le filtre « en boîte »



Propriétés d'un filtre gaussien

- Retire les hautes fréquences de l'image (filtre « passe-bas »)
  - Les images deviennent plus lisses
- Filter un filtre gaussien avec un autre filtre gaussien?
  - Le résultat est un filtre gaussien!
  - Si les deux filtres ont un écart-type de σ, c'est équivalent à un filtre d'écart-type $\sqrt{2}\sigma$
- Séparable
  - filtre gaussien 2D = produit de deux filtres gaussiens 1D

Le filtre gaussien est séparable

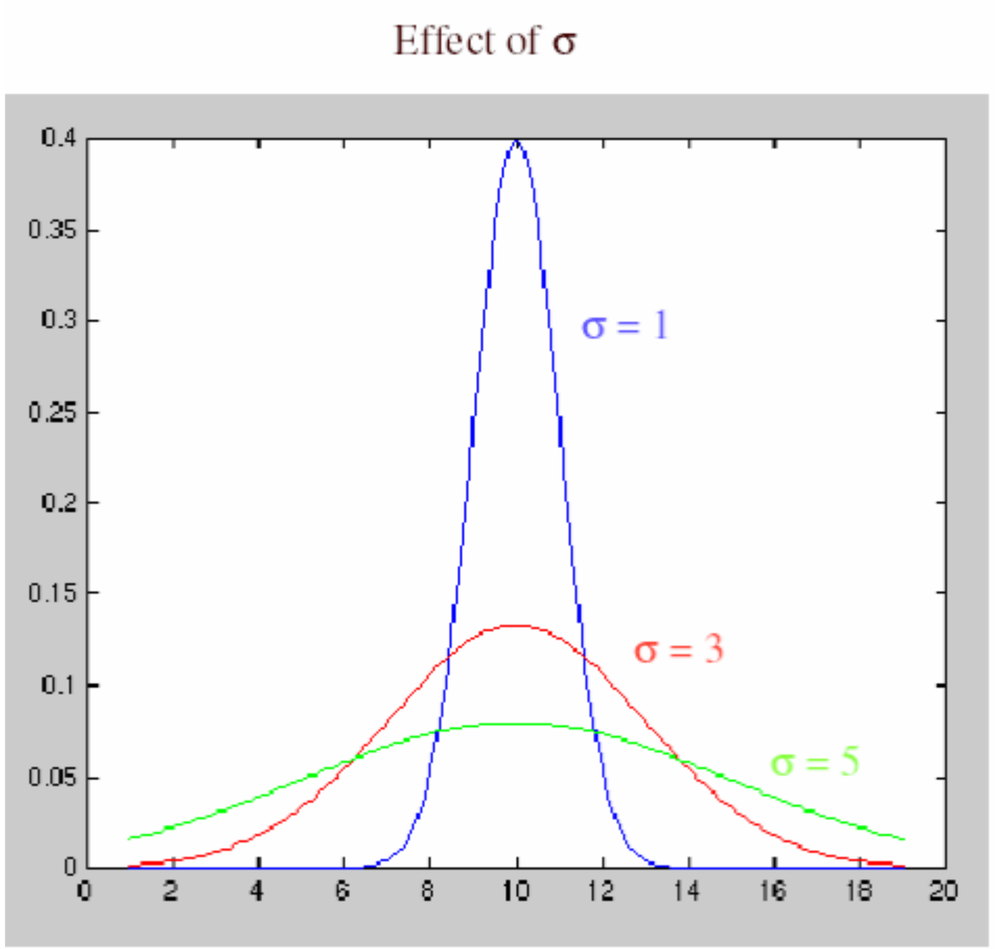
$$
\begin{aligned}
G_\sigma(x,y) &= \frac{1}{2\pi\sigma^2}\exp^{-\frac{x^2+y^2}{2\sigma^2}} \\
&= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)
\end{aligned}
$$

Pourquoi c'est important?

Source: D. Lowe

Considération pratique : taille du filtre gaussien

- On ne veut pas « couper » le filtre trop brusquement
- Les valeurs aux extrémités devraient être proche de 0
- Règle empirique:
  - la demie-taille devrait être 3σ



Propriétés des filtres linéaires

- Linéarité (!) :
  - filtre(f1 + f2) = filtre(f1) + filtre(f2)
- Invariance aux déplacements : ne dépend pas de l'emplacement du pixel
  - filtre(shift(f)) = shift(filtre(f))

Source: S. Lazebnik

Propriétés des filtres linéaires

- Commutativité $a * b = b * a$
  - On peut traiter l'image comme un filtre
- Associativité $a * (b * c) = (a * b) * c$
  - Si j'applique deux filtres (*a* et *b*) à une image *c,*
    je peux « pré-filtrer » les filtres et convoluer le résultat à *c*
- Distributivité $a * (b + c) = (a * b) + (a * c)$
- Factoriser les scalaires $ka * b = a * kb = k(a * b)$
- Identité $e = 1\ ,\ a * e = a$

Source: S. Lazebnik

Considération pratique : bordure de l'image

```
from scipy.signal import convolve2d
```

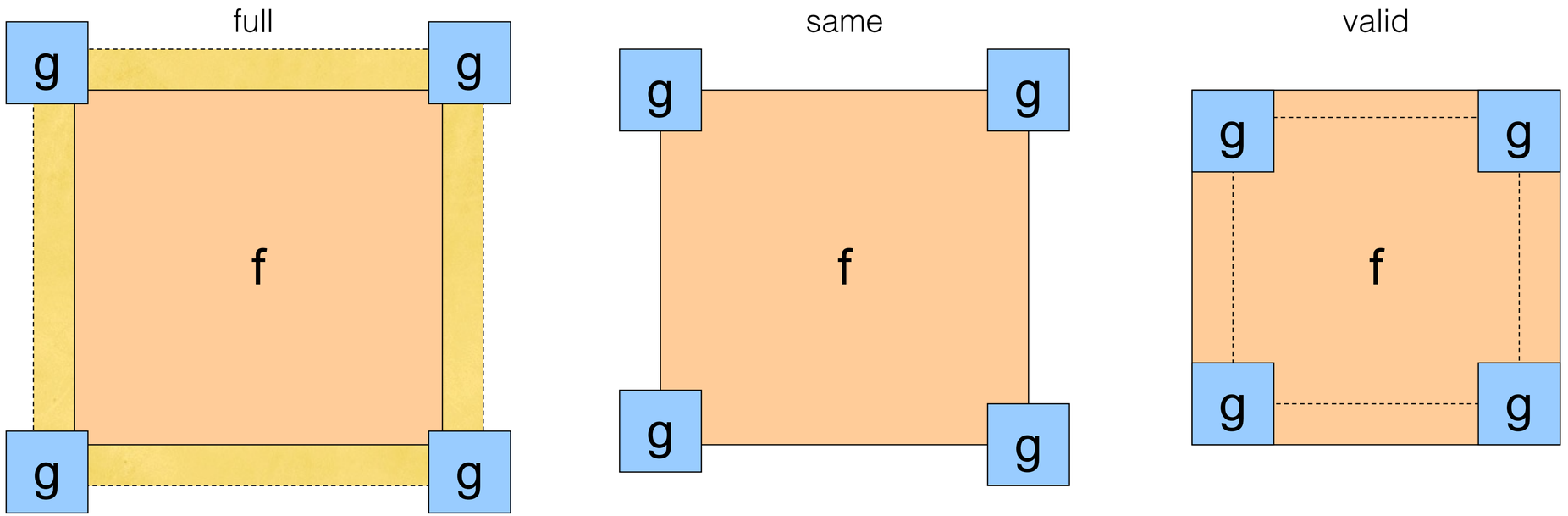
- Le filtre dépasse l'image!
- Plusieurs options pour extrapoler
  - 0 (`boundary='fill'`)
  - "enrouler" (`boundary='wrap'`)
  - répéter (pas disponible dans scipy.signal)
  - réflexion (`boundary='symm'`)

Source: S. Marschner

Considération pratique : taille du résultat

```
from scipy.signal import convolve2d
```

- 3 options :
  - complète (`mode='full'`) : taille(f) + taille(g)
  - même taille que l'image (`mode='same'`) : taille(f)
  - valide (`mode='valid'`) : taille(f) - taille(g)



Source: S. Lazebnik

Exercice #1

Un filtre 1x2 qui calcule le gradient horizontal $g_x(x, y)$:

$$g_x(y, x) = i(y, x + 1) - i(y, x) \quad \forall x, y$$

Source: D. Hoiem

Exercice #2

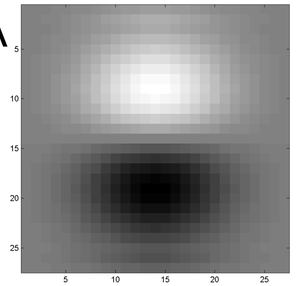
Un filtre 3x3 qui calcule la différence entre un pixel
et la moyenne de ses 4-voisins.

Source: D. Hoiem

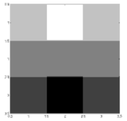
Exercice #3

_ = D * B

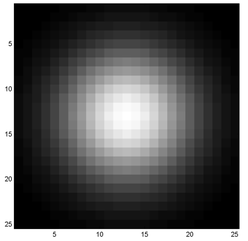
A



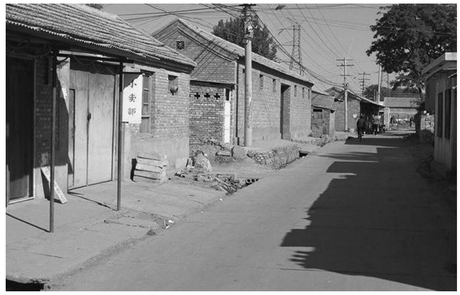
B



C

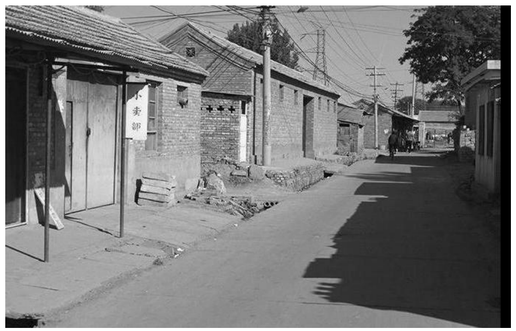


D

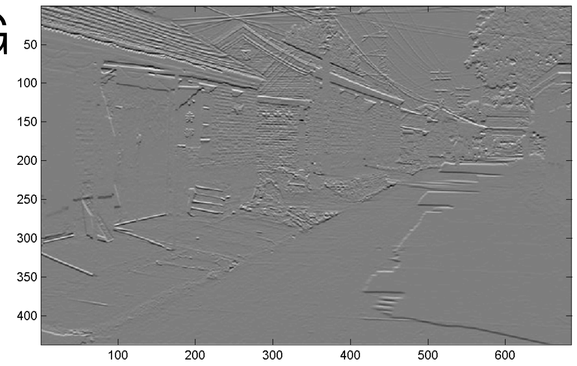


E

F

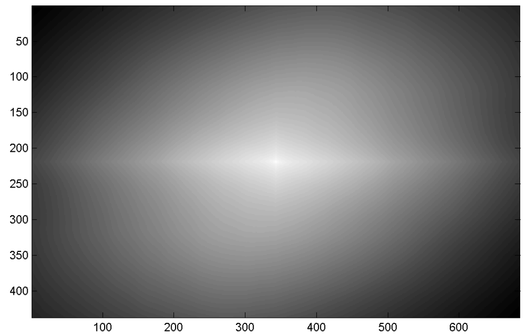


G



H

I

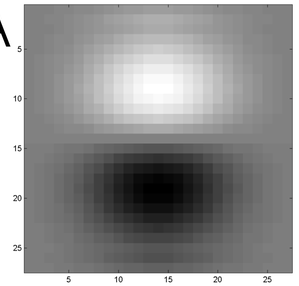


Source: D. Hoiem

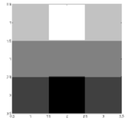
Exercice #3

A = _ * _

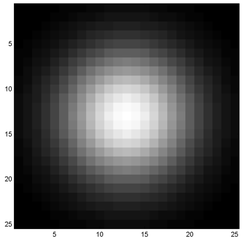
A



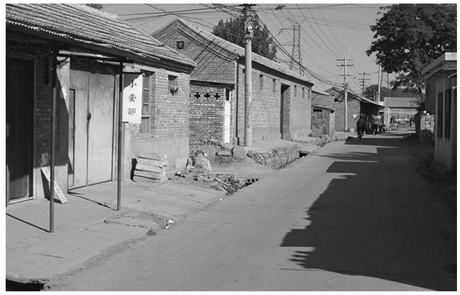
B



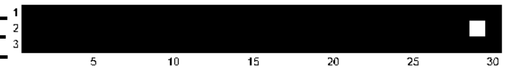
C



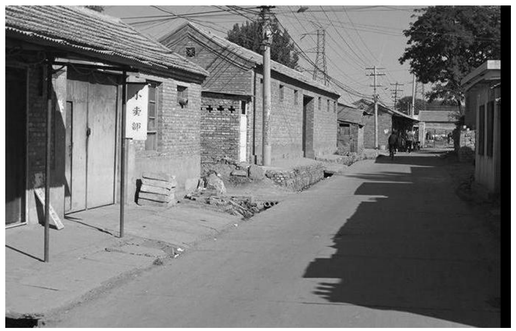
D



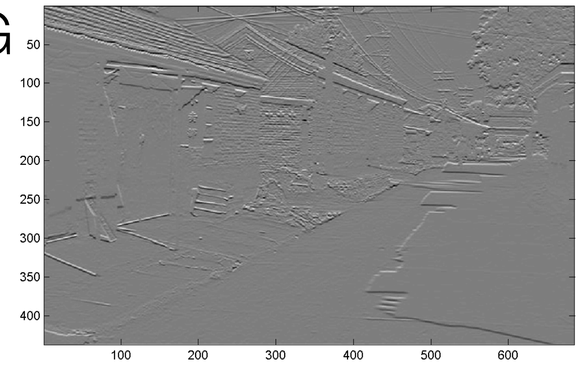
E



F

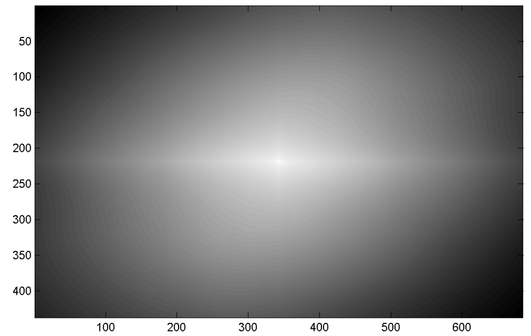


G



H

I

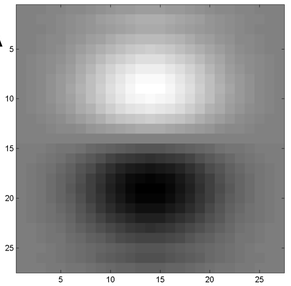


Source: D. Hoiem

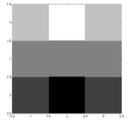
Exercice #3

F = D * _

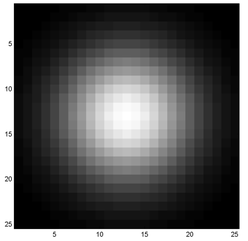
A



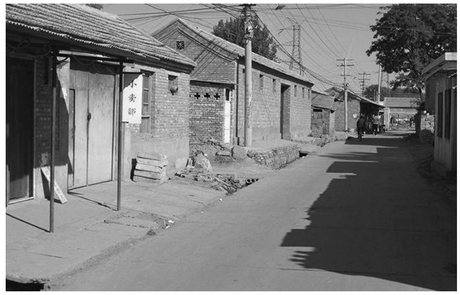
B



C

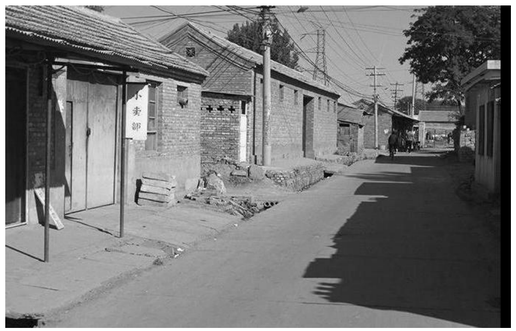


D

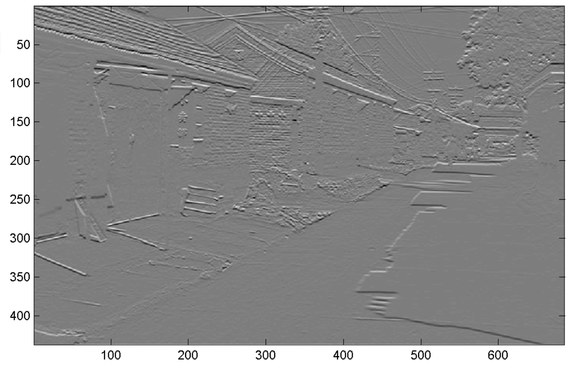


E

F

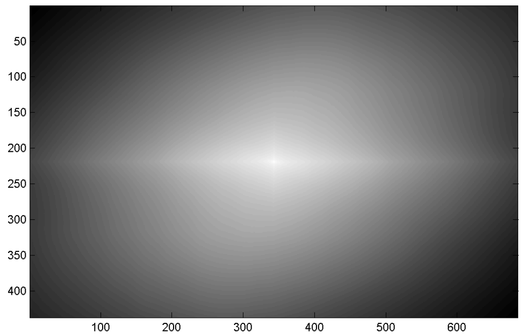


G



H

I

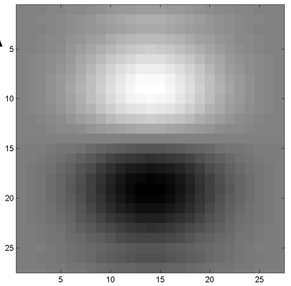


Source: D. Hoiem

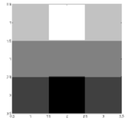
Exercice #3

_ = D * D

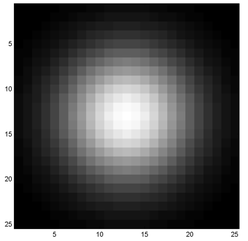
A



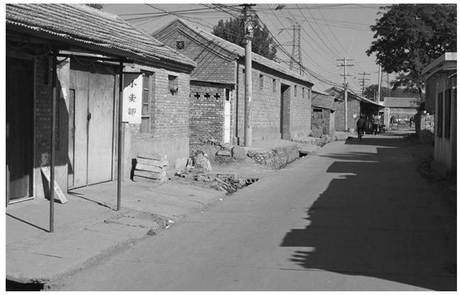
B



C

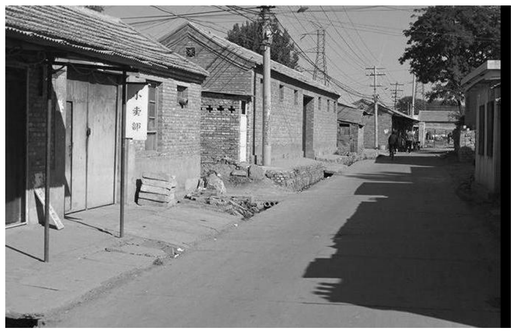


D

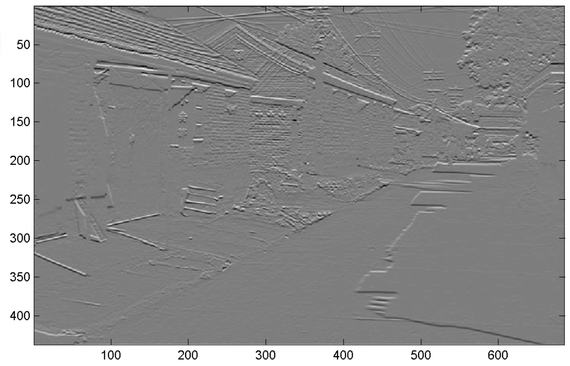


E

F

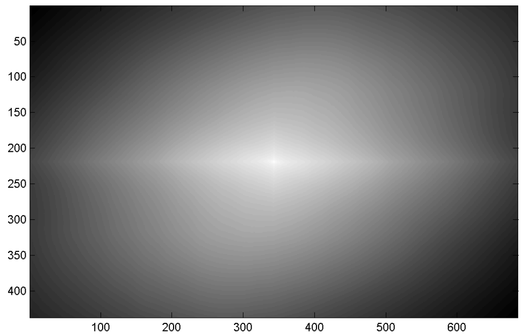


G



H

I



Source: D. Hoiem