II. Analyse élémentaire d'images

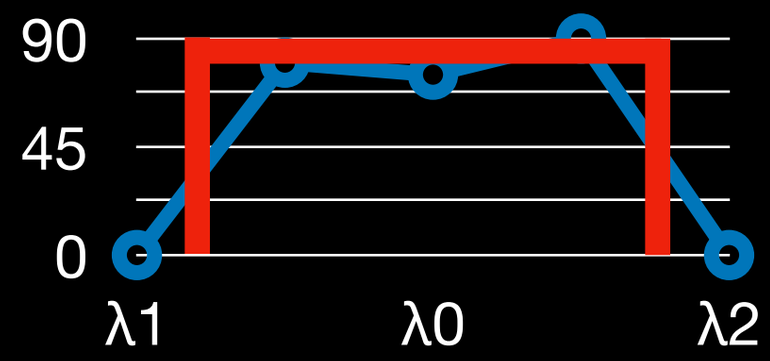
- **Il faut enfin considérer le problème de l'échantillonnage de l'image par le détecteur, échantillonnage qui doit respecter le critère de SHANNON (ou Nyquist). Ici : taille du pixel ≤ 1/2 λ/D (en unités angulaires), mais aussi ne pas être trop petit afin d'être moins sensible aux bruits (et ne pas être obligé d'intégrer trop longtemps => problème de bougé/floutage).**

=> recherche de compromis entre résolution angulaire (=spatiale), résolution temporelle et bruits — pour un appareil/instrument donné.

II. Analyse élémentaire d’images

- **Également : quantification, due à la conversion analogique-digitale (*Analog-to-Digital Conversion*, ou *ADC*) qui implique que les unités d’intensité mesurée deviennent donc en toute rigueur des *ADU* (*Analog-to-Digital Units*).**

- ***In fine* : le rendement quantique *qe* (pour *quantum efficiency*) du détecteur dépend de la longueur d’onde…**



III. Filtrage

1- INTRODUCTION

- **Filtrage : opération fondamentale en traitement d'images :**
  -> peut permettre d'améliorer la perception de certains détails,
  -> de réduire le bruit,
  -> de compenser certains défauts du capteur,
  -> etc.
- **Dans ce chapitre : principes de base + exemples simples.**
- **Dans la suite : application à la détection de contours et à la restoration d'images.**
- **Filtrage linéaire, filtre médian, (filtrage dans le plan de Fourier,).**
- **Application dans ce chapitre : atténuation du bruit.**
- **NB : filtre = « élément structurant »…**

III. Filtrage

2- FILTRAGE LINÉAIRE

$$I_f(x,y) = \sum_{a,b} h(a,b)\ I(x+a, y+b)$$

- **I_f = Image filtrée, h = filtre (=élément structurant), I=image d'origine**

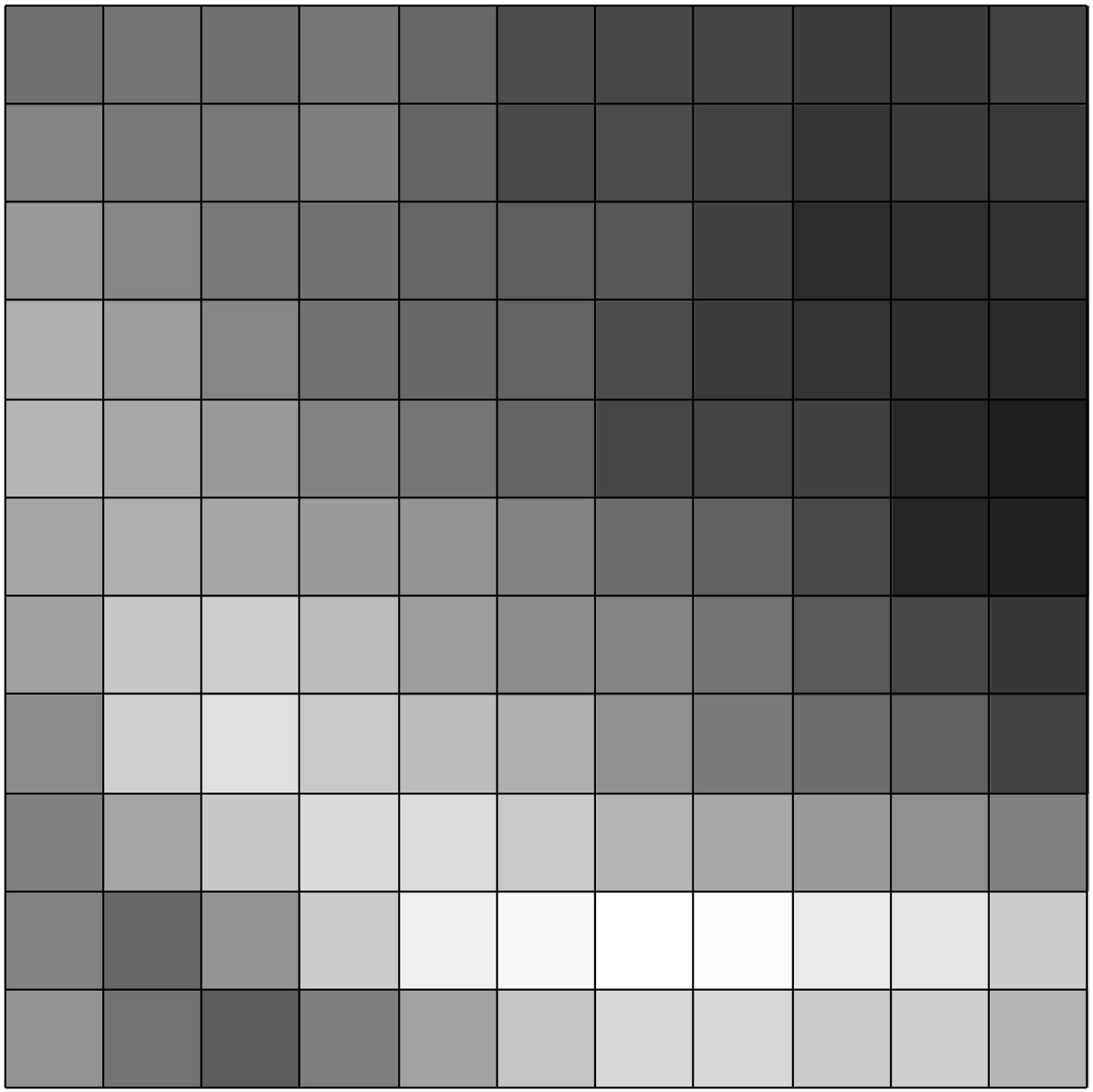
- **On remarque tout de suite qu'il s'agit d'une convolution si on fait subir au filtre une symétrie par rapport à l'origine (i.e. une rotation de 180°) :**

$$I_f(x,y) = \sum_{c,d} g(c,d)\ I(x-c, y-d) = (g * I)(x,y)$$

en prenant $g(c,d) = h(-c,-d)$

- **En fait : passage d'une fenêtre sur l'image = convolution par un filtre, *souvent* 3x3, avec en général : somme des éléments du filtre = 1.**

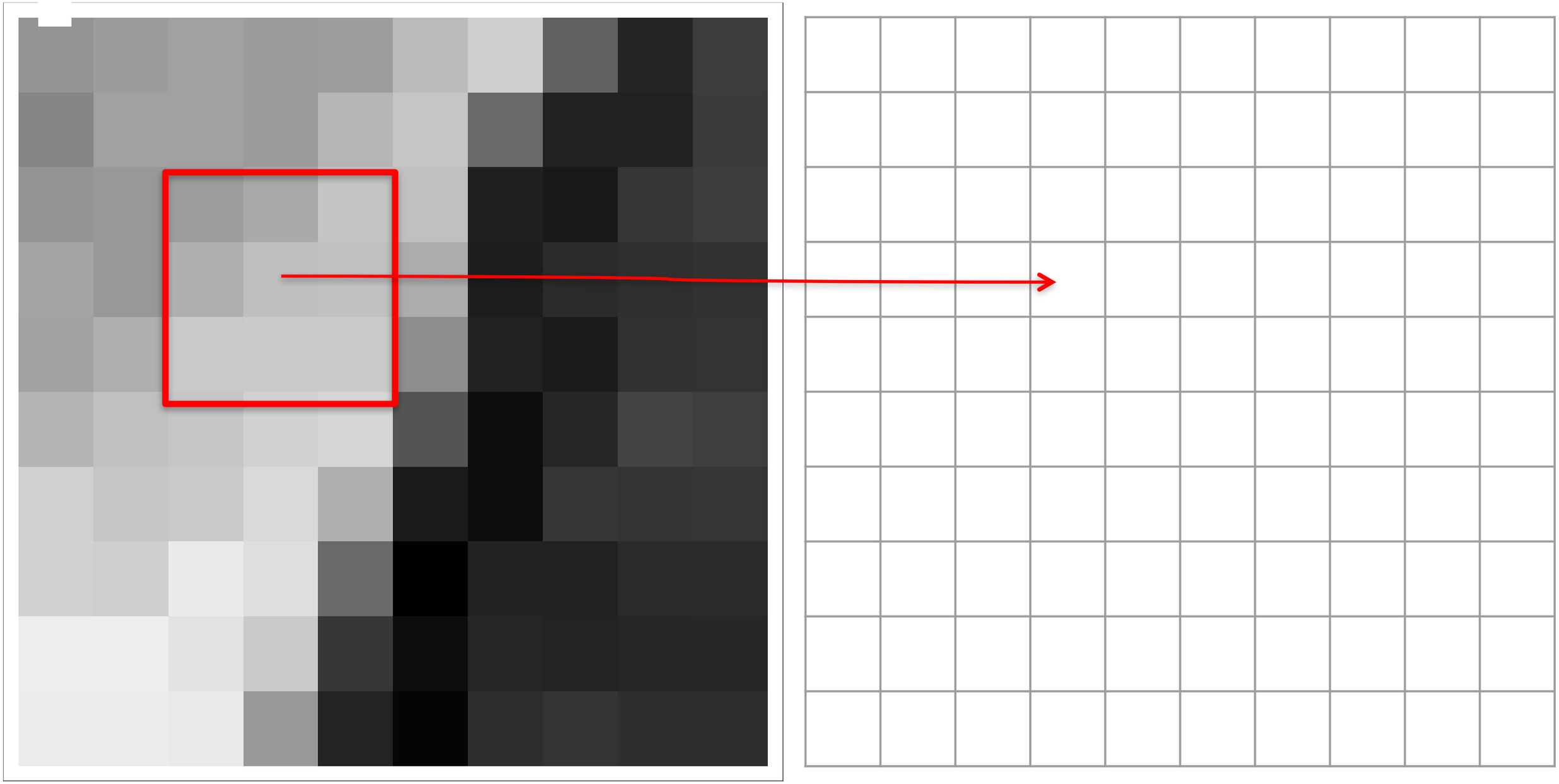
Passage d’un filtre linéaire

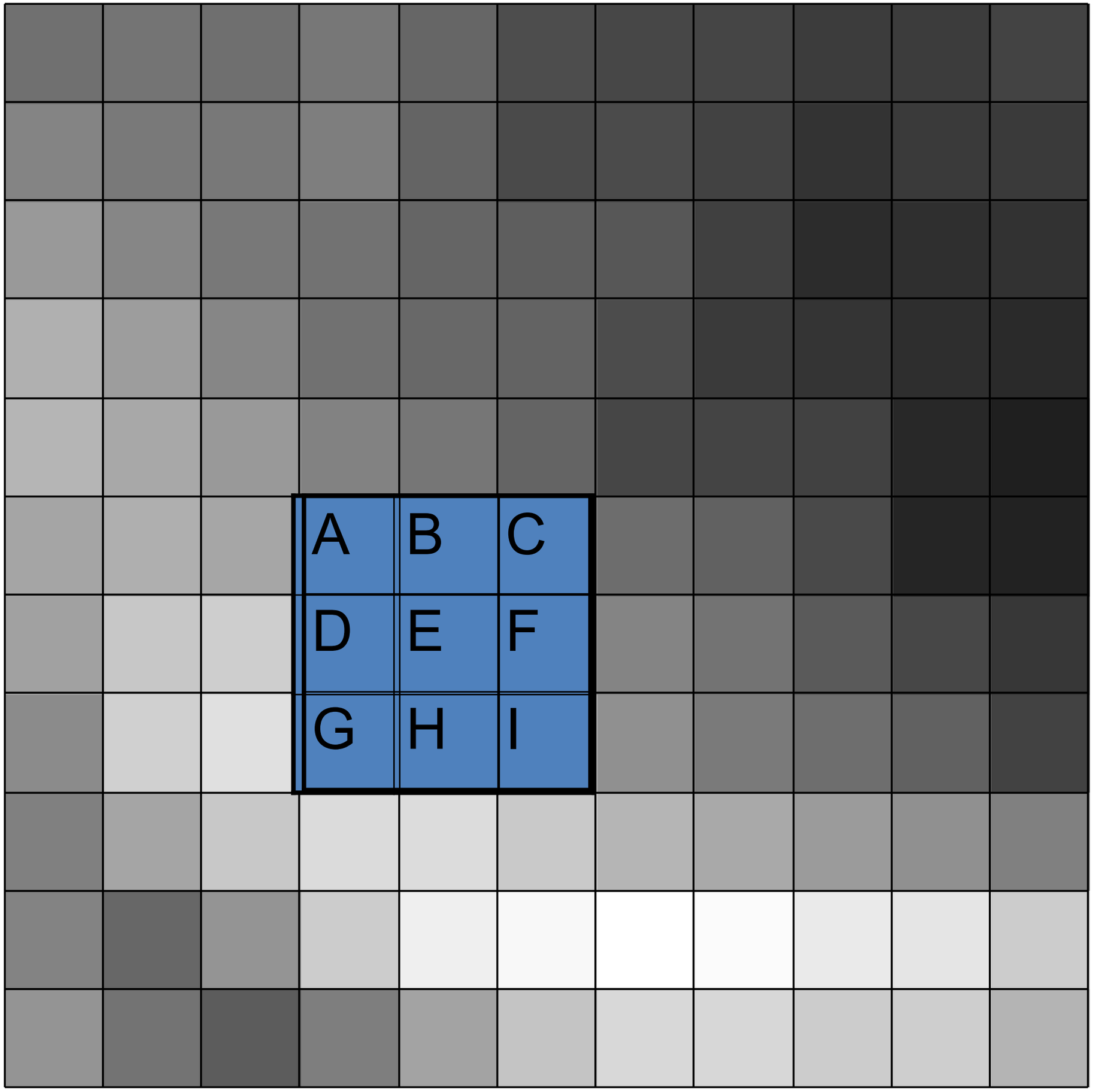


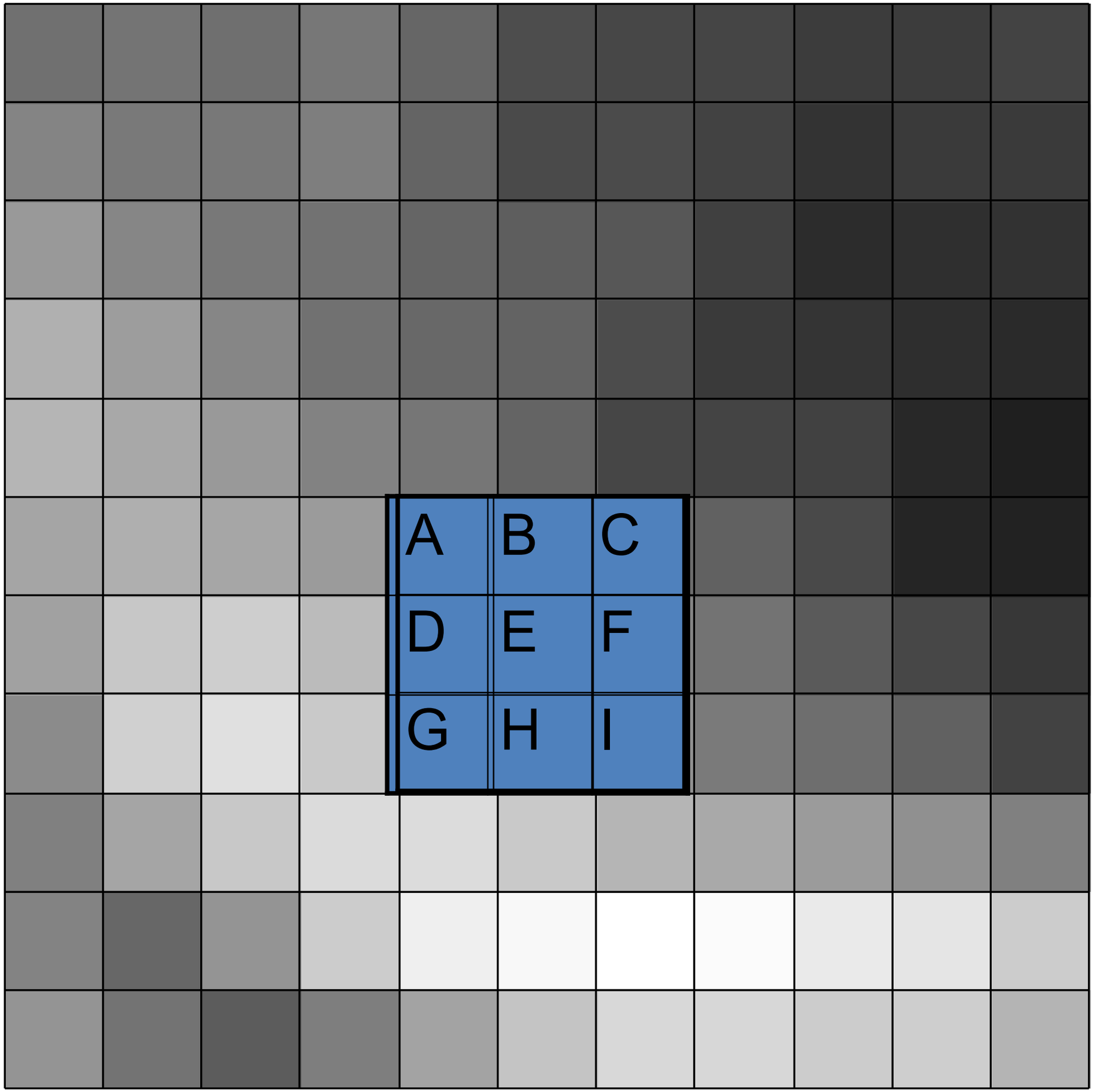
⊗

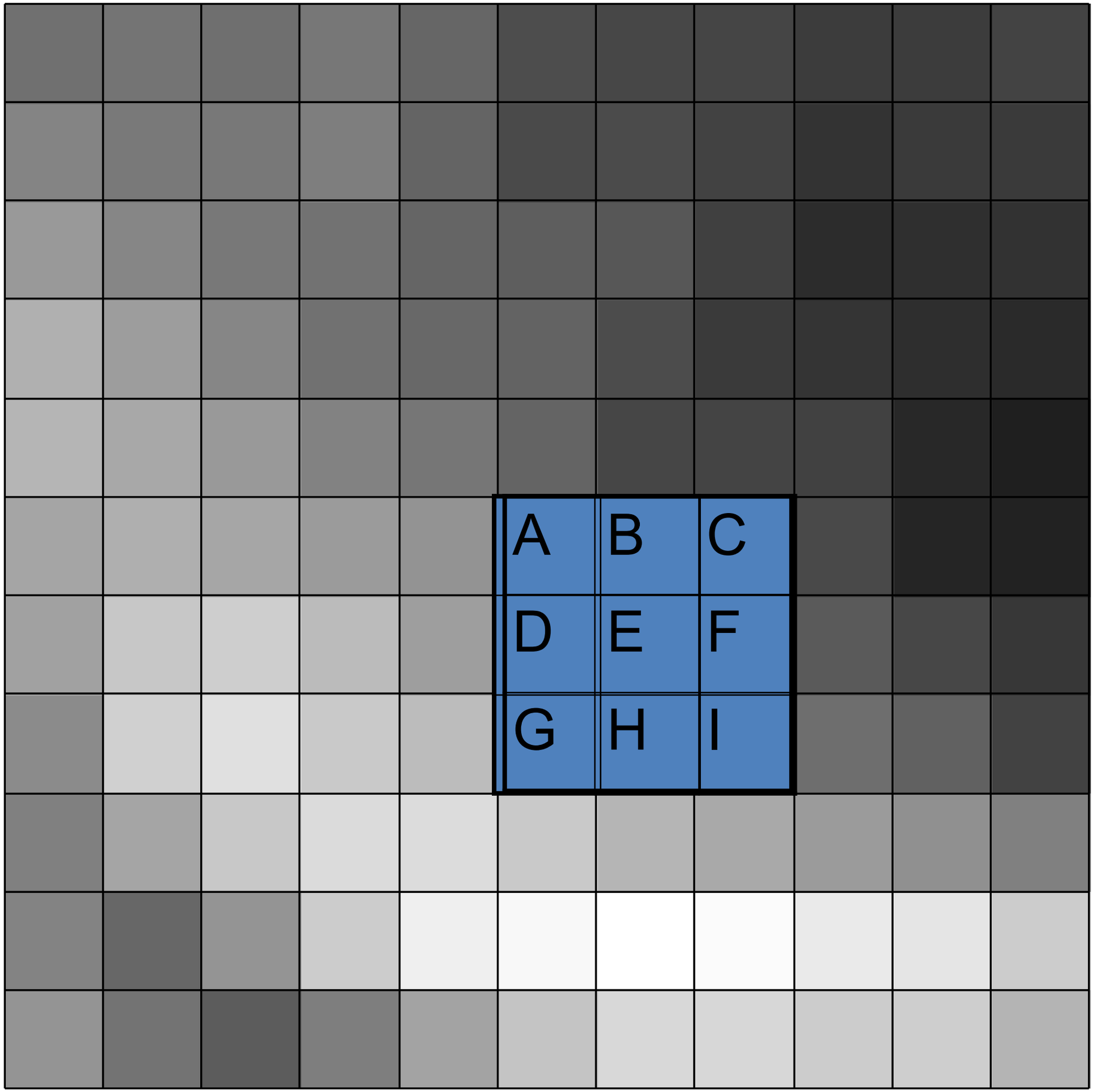
A	B	C
D	E	F
G	H	I

On fait la somme pondérée des valeurs des pixels dans le tableau 3x3,
et on met la valeur trouvée dans un nouveau tableau, au centre.
On fait ça pour tous les points du tableau.




A
B
C
D
E
F
G
H
I


A B C
D E F
G H I


A
B
C
D
E
F
G
H
I

III. Filtrage

- **Sous Matlab/Octave : FILTER2(filtre, image)**
  —> rend l'image filtrée

>> If = filter2(h, I);

- **Remarque : *h* est *<u>en général</u>* de taille impaire afin de pouvoir définir un pixel central ! En effet, si *h* est de taille *(2n+1)x(2n+1)* alors *a* et *b* varient tous deux de *-n* à *+n* en passant par *0*…**

III. Filtrage

- **Exemple simple : la <u>MOYENNE GLISSANTE</u>…**

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

```
>> h = ones(3,3)/9
>> I = imread('house.jpg');
>> I = rgb2gray(I);
>> I = double(I)/255.0;
>> If = filter2(h,I);
>> figure, subplot(1,2,1), imshow(I), title('image originale')
>> subplot(1,2,2), imshow(If), title('image filtrée')
```

III. Filtrage

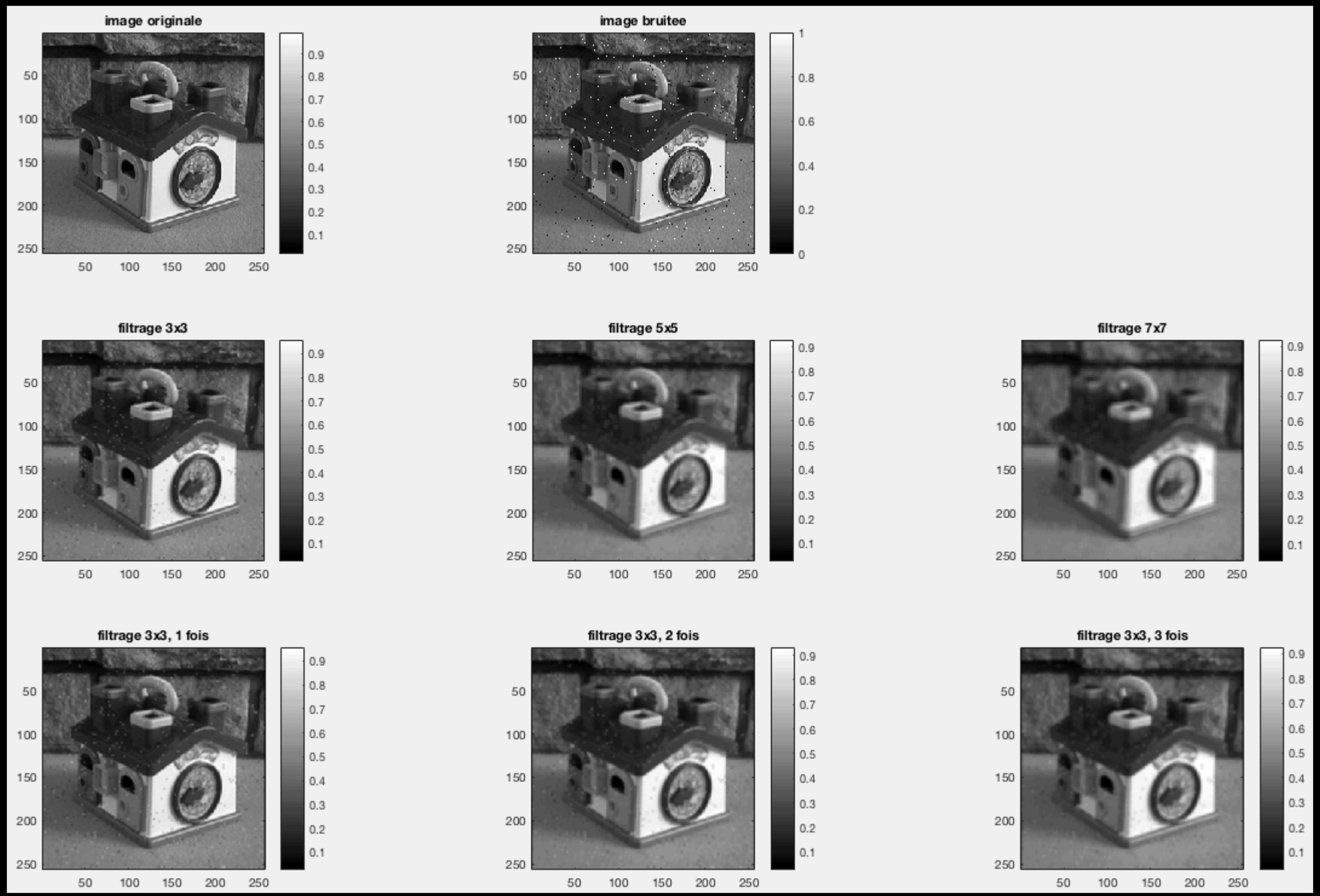
- **EXERCICE 1 : Appliquer du bruit ‘poivre & sel’ (1%) à l’image de votre choix avant de la filtrer. Jugez de l’amélioration en fonction de la taille de *h* (3x3, 5x5 ou 7x7)…**

- **EXERCICE 2 : Même chose en fonction du nombre de passages de *h* (1, 2 ou 3)…**

III. Filtrage

```
clear
close all

I=imread('/Users/marcel/Documents/MATLAB/0-images-exemples/classiques/house.jpg');
I=rgb2gray(I);
I=double(I)/255.0;
Ib=imnoise(I, 'salt & pepper', 0.01);

% 1ère méthode
figure(1)
subplot(3,3,1), imagesc(I), title('image originale'), colormap(gray), colorbar, axis('image')
subplot(3,3,2), imagesc(Ib), title('image bruitee'), colorbar, axis('image')

h3=ones(3,3)/9;
I3=filter2(h3,Ib);
subplot(3,3,4), imagesc(I3), title('filtrage 3x3'), colorbar, axis('image')

h5=ones(5,5)/5^2;
I5=filter2(h5,Ib);
subplot(3,3,5), imagesc(I5), title('filtrage 5x5'), colorbar, axis('image')

h7=ones(7,7)/7^2;
I7=filter2(h7,Ib);
subplot(3,3,6), imagesc(I7), title('filtrage 7x7'), colorbar, axis('image')

subplot(3,3,7), imagesc(I3), title('filtrage 3x3, 1 fois'), colorbar, axis('image')

I3_2=filter2(h3,I3);
subplot(3,3,8), imagesc(I3_2), title('filtrage 3x3, 2 fois'), colorbar, axis('image')

I3_3=filter2(h3,I3_2);
subplot(3,3,9), imagesc(I3_3), title('filtrage 3x3, 3 fois'), colorbar, axis('image')
```

III. Filtrage



III. Filtrage

```
34      % 2ème méthode
35 -    figure(2)
36 -    subplot(3,3,1), imagesc(I), title('image originale'), colormap(gray), colorbar, axis('image')
37 -    subplot(3,3,2), imagesc(Ib), title('image bruitee'), colorbar, axis('image')
38
39 -    for n=1:3
40 -        h=ones(2*n+1,2*n+1)/(2*n+1)^2;
41 -        If=filter2(h,Ib);
42 -        subplot(3,3,n+3), imagesc(If), colorbar, axis('image')
43 -        title(['filtrage ',int2str(2*n+1),'x',int2str(2*n+1)])
44 -    end
45
46 -    h=ones(3,3)/9;
47 -    If=Ib;
48 -    for n=1:3
49 -        If=filter2(h,If);
50 -        subplot(3,3,n+6), imagesc(If), colorbar, axis('image')
51 -        title(['filtrage 3x3, ',int2str(n),' fois'])
52 -    end
```

III. Filtrage

- **Autre filtre passe-bas : le <u>FILTRE GAUSSIEN</u>**

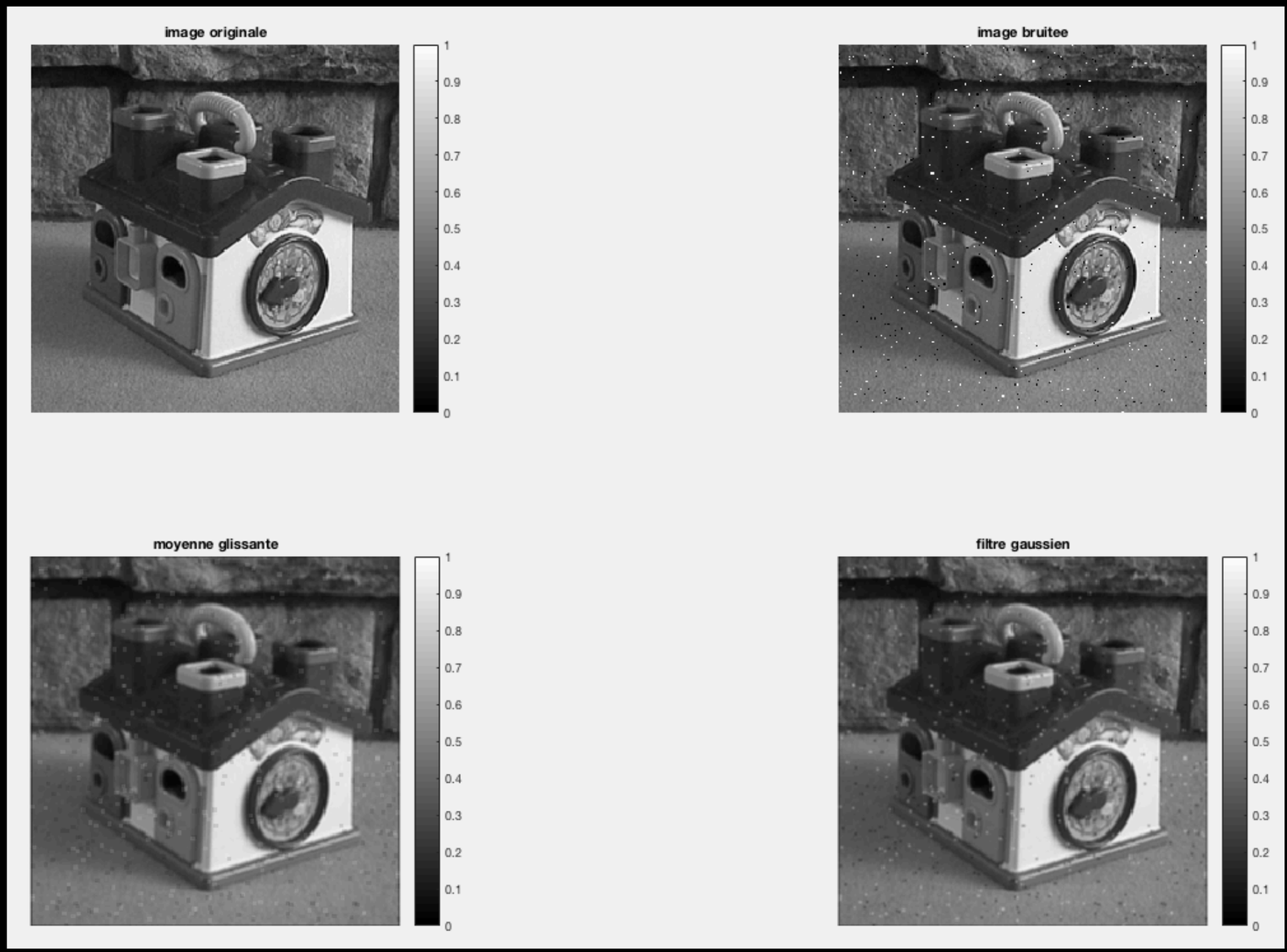
$$h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- **EXERCICE 3 : Comparer avec le filtre précédent (3x3, un seul passage)…**

III. Filtrage

```
clear
close all

img='/Users/marcel/Documents/MATLAB/0-images-exemples/classiques/house.jpg'
I=imread(img);
I=rgb2gray(I);
I=double(I)/255.0;
Ib=imnoise(I, 'salt & pepper', 0.01);

h=ones(3,3)/9;
If=filter2(h,Ib);

g=[1 2 1;2 4 2;1 2 1]/16.
Ig=filter2(g,Ib);

figure
subplot(2,2,1), imshow(I), title('image originale'), colorbar
subplot(2,2,2), imshow(Ib), title('image bruitee'), colorbar
subplot(2,2,3), imshow(If), title('moyenne glissante'), colorbar
subplot(2,2,4), imshow(Ig), title('filtre gaussien'), colorbar
```

III. Filtrage



—> pas d’énormes différences entre filtres passe-bas en général…

III. Filtrage

- **Un filtre qui ne change rien à l'image d'origine : le <u>FILTRE UNITÉ</u>**

$$h = \begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$

- **Un exemple de filtre qui ne fait pas grand' chose :**

$$h = \begin{matrix} 1 & 1 & 1 \\ 1 & 92 & 1 \\ 1 & 1 & 1 \end{matrix} \times 1/100$$

(et donc : passe-tout !)

III. Filtrage

- **Plus intéressants : les <u>FILTRES PASSE-HAUT</u>, i.e. :**

$$h = \begin{matrix} -1 & -1 & -1 \\ -1 & +9 & -1 \\ -1 & -1 & -1 \end{matrix}$$

- **EXERCICE 4 :**
  Appliquer ce filtre à l'image « house », version normale et version « poivre & sel » (*d=0.01*). Faire plusieurs passages. Commenter.

  (*Attention à bien remettre les images entre 0 et 1 entre deux passages du filtre afin que la fonction filter2 puisse… fonctionner.*)

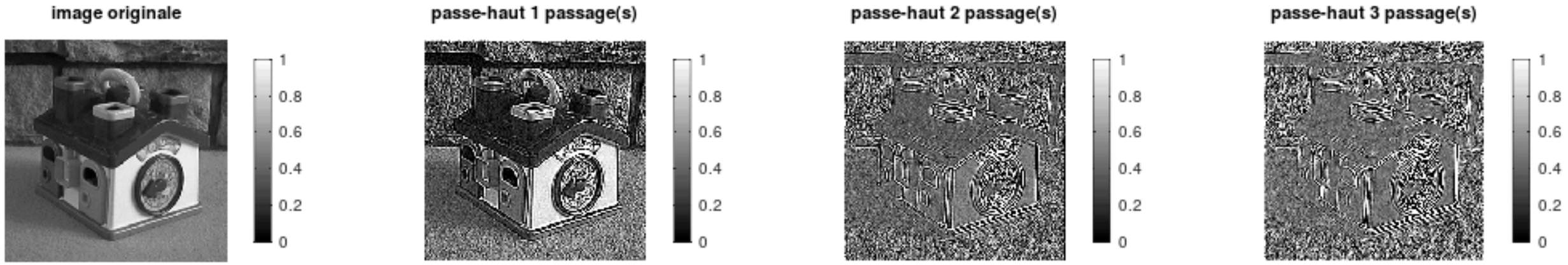
III. Filtrage

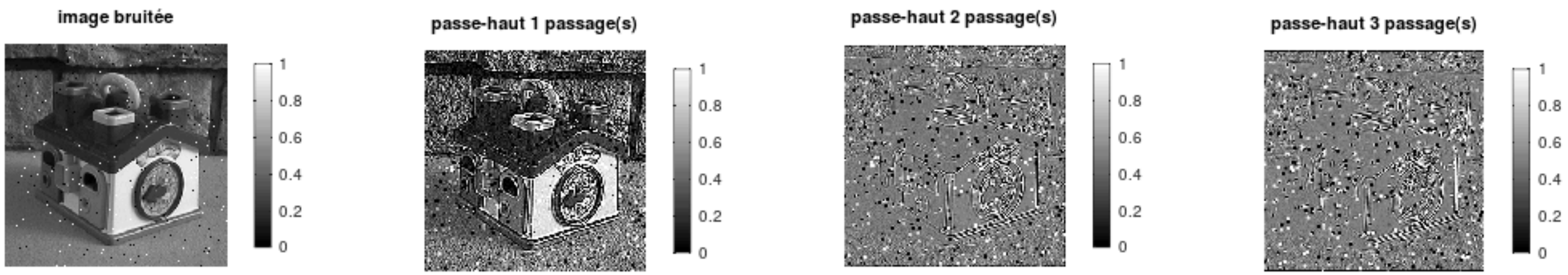
```
clear
close all
%pkg load image
%(pour Octave, pas Matlab)

img='/Users/marcel/Documents/MATLAB/GBM/0-images/frog.jpg'
I=imread(img); I=rgb2gray(I); I=double(I)/255;
Ib=imnoise(I, 'salt & pepper', 0.01);
%hi = [-1 -1 -1;-1 9 -1;-1 -1 -1]
hi=-ones(3,3); hi(2,2)=9

% version sans boucle for et avec imagesc :
figure(1), colormap('gray')
subplot(2,4,1), imagesc(I), axis('image'), title('image originale'), colorbar

% image sans bruit
I_hi = filter2(hi, I);
subplot(2,4,2), imagesc(I_hi), axis('image'), title('passe-haut, 1 passage'), colorbar

I_hi = I_hi-min(min(I_hi)); I_hi = I_hi/max(max(I_hi));
I_hi2 = filter2(hi, I_hi);
subplot(2,4,3), imagesc(I_hi2), axis('image'), title('passe-haut, 2 passages'), colorbar

I_hi2= I_hi2-min(min(I_hi2)); I_hi2=I_hi2/max(max(I_hi2));
I_hi3 = filter2(hi, I_hi2);
subplot(2,4,4), imagesc(I_hi3), axis('image'), title('passe-haut, 3 passages'), colorbar

% image avec bruit
subplot(2,4,5), imagesc(Ib), axis('image'), title('image bruitee'), colorbar

Ib_hi = filter2(hi, Ib);
subplot(2,4,6), imagesc(Ib_hi), axis('image'), title('passe-haut, 1 passage'), colorbar

Ib_hi = Ib_hi-min(min(Ib_hi)); Ib_hi = Ib_hi/max(max(Ib_hi));
Ib_hi2 = filter2(hi, Ib_hi);
subplot(2,4,7), imagesc(Ib_hi2), axis('image'), title('passe-haut, 2 passages'), colorbar

Ib_hi2 = Ib_hi2-min(min(Ib_hi2)); Ib_hi2 = Ib_hi2/max(max(Ib_hi2));
Ib_hi3 = filter2(hi, Ib_hi2);
subplot(2,4,8), imagesc(Ib_hi3), axis('image'), title('passe-haut, 3 passages'), colorbar
```

III. Filtrage

```
% version avec boucle for et avec imshow :
figure(2)

% image sans bruit
Ih = I;
subplot(2,4,1), imshow(I), title('image originale'), colorbar
for n=1:3
    Ih = Ih - min(min(Ih)); Ih = Ih/max(max(Ih));
    Ih = filter2(hi, Ih);
    subplot(2,4,n+1), imshow(Ih), colorbar
    title(['passe-haut ',num2str(n),' passage(s)'])
end

% image avec bruit
Ih = Ib;
subplot(2,4,5), imshow(Ib), title('image bruitée'), colorbar
for n=1:3
    Ih = Ih - min(min(Ih)); Ih = Ih/max(max(Ih));
    Ih = filter2(hi, Ih);
    subplot(2,4,n+5), imshow(Ih), colorbar
    title(['passe-haut ',num2str(n),' passage(s)'])
end
```

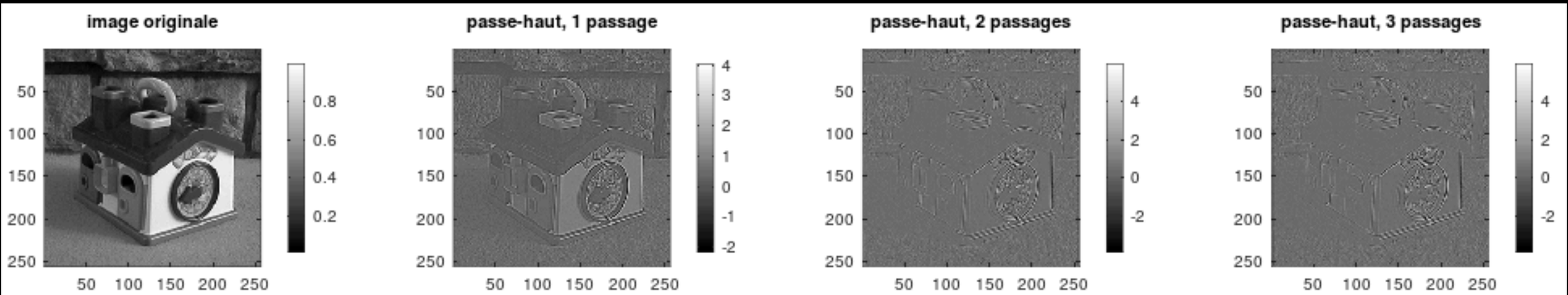
III. Filtrage





—> fait très clairement ressortir les plus hautes fréquences spatiales (transitions brusques, détails) en atténuant les plus basses (formes globales)

III. Filtrage



III. Filtrage

- **Prenons une image de départ plus simple (8x8, partie gauche blanche, partie droite noire) pour mieux appréhender les effets de ces trois filtres :**

$$
Isp = \begin{bmatrix}
1 & . & . & 1 & 0 & . & . & 0 \\
. & . & 1 & 1 & 0 & 0 & . & . \\
. & 1 & 1 & 1 & 0 & 0 & 0 & . \\
. & . & 1 & 1 & 0 & 0 & . & . \\
. & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . \\
1 & . & . & 1 & 0 & . & . & 0
\end{bmatrix}
$$

>> Isp = zeros(8,8) ; Isp(:, 1:4)=1 **(ou : >>** ***Isp = ones(8,8) ; Isp(:, 5:8)=0*****)**

III. Filtrage

- **Puis filtrons-la par un filtre passe-bas « moyenne glissante » 3x3 (h) :**

  >> h=ones(3,3)/9
  >> lsp_low = filter2(h, lsp)

  On obtient (en oubliant les bords => image 6x6) :

$$
lsp\_low = \begin{matrix} 1 & 1 & 2/3 & 1/3 & 0 & 0 \\ 1 & 1 & 2/3 & 1/3 & 0 & 0 \\ 1 & 1 & 2/3 & 1/3 & 0 & 0 \\ 1 & 1 & 2/3 & 1/3 & 0 & 0 \\ 1 & 1 & 2/3 & 1/3 & 0 & 0 \\ 1 & 1 & 2/3 & 1/3 & 0 & 0 \end{matrix}
$$

  —> pas de changement dans la zone uniforme (très basse fréquence) et adoucissement de la transition brusque (la marche, très haute fréquence)

III. Filtrage

- **En filtrant à présent par un filtre passe-bas Gaussien 3x3 (g) :**

  >> g=[1 2 1 ; 2 4 2 ; 1 2 1]/16
  >> Isp_gau = filter2(g, Isp)

  On obtient (en oubliant les bords => image 6x6) :

```
              1   1  3/4  1/4   0   0
              1   1  3/4  1/4   0   0
Isp_gau =     1   1  3/4  1/4   0   0
              1   1  3/4  1/4   0   0
              1   1  3/4  1/4   0   0
              1   1  3/4  1/4   0   0
```

  —> adoucissement moins fort de la marche => filtrage un peu moindre des hautes fréquences…

III. Filtrage

- **Si, au contraire, on veut accentuer les hautes fréquences (la marche), on applique le filtre passe-haut (hi) :**

  >> hi=-ones(3,3); hi(2,2)=9
  >> lsp_hi = filter2(hi, lsp)

  On obtient (en oubliant les bords => image 6x6) :

$$
\text{lsp\_hi} = \begin{bmatrix} 1 & 1 & 4 & -3 & 0 & 0 \\ 1 & 1 & 4 & -3 & 0 & 0 \\ 1 & 1 & 4 & -3 & 0 & 0 \\ 1 & 1 & 4 & -3 & 0 & 0 \\ 1 & 1 & 4 & -3 & 0 & 0 \\ 1 & 1 & 4 & -3 & 0 & 0 \end{bmatrix}
$$

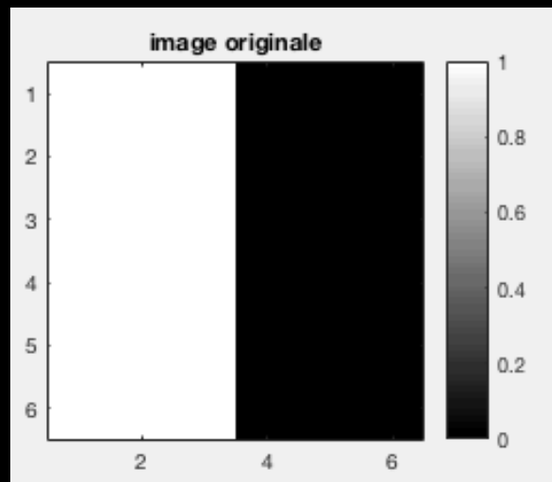
  —> accentuation de la transition brusque.

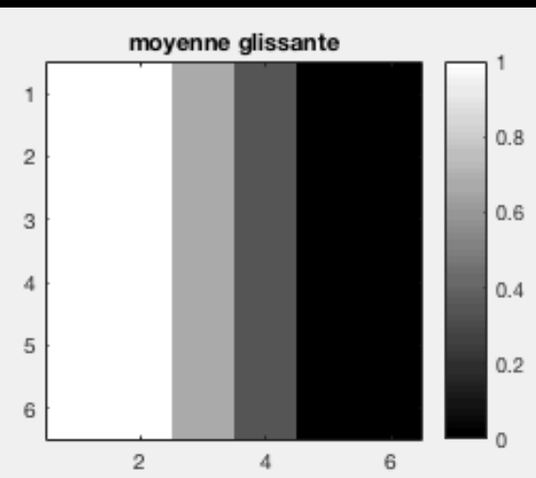
III. Filtrage

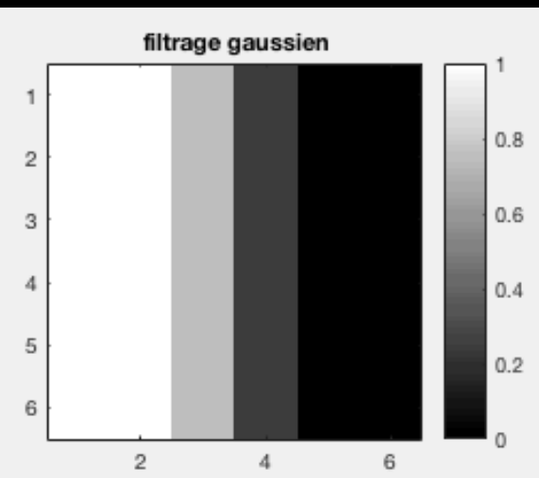
- **EXERCICE 5 : Codage ‘propre’ de ce que nous venons de voir…**

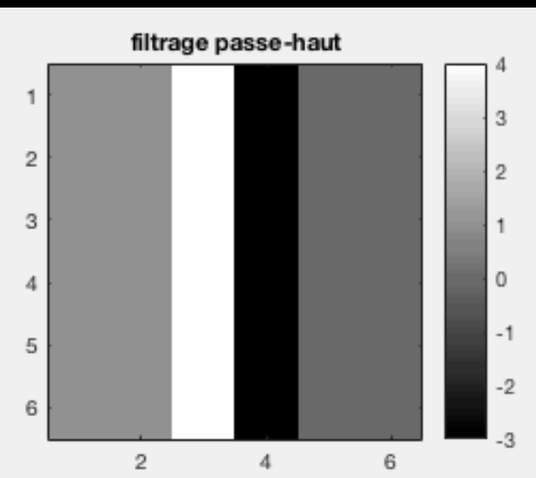
```
clear
close all

%Isp = zeros(8,8); Isp(:,1:end/2)=1
Isp = ones(8,8); Isp(:,end/2+1:end)=0

Isp_low = filter2(ones(3,3)/9, Isp)
Isp_gau = filter2([1 2 1;2 4 2;1 2 1]/16., Isp)
Isp_hi  = filter2([-1 -1 -1;-1 9 -1;-1 -1 -1], Isp)

figure, colormap('gray')

subplot(2,4,1), imagesc(Isp(2:end-1,2:end-1)), colorbar
title('image originale'), axis('square')

subplot(2,4,2), imagesc(Isp_low(2:end-1,2:end-1)), colorbar
title('moyenne glissante'), axis('square')

subplot(2,4,3), imagesc(Isp_gau(2:end-1,2:end-1)), colorbar
title('filtrage gaussien'), axis('square')

subplot(2,4,4), imagesc(Isp_hi(2:end-1,2:end-1)), colorbar
title('filtrage passe-haut'), axis('square')

subplot(2,4,5), plot(Isp(4,2:end-1),'--o')
axis([1 6 -0.1 1.1]), axis('square')

subplot(2,4,6), plot(Isp_low(4,2:end-1),'--o')
axis([1 6 -0.1 1.1]), axis('square')

subplot(2,4,7), plot(Isp_gau(4,2:end-1),'--o')
axis([1 6 -0.1 1.1]), axis('square')

subplot(2,4,8), plot(Isp_hi(4,2:end-1),'--o')
axis([1 6 -3.5 4.5]), axis('square')
```

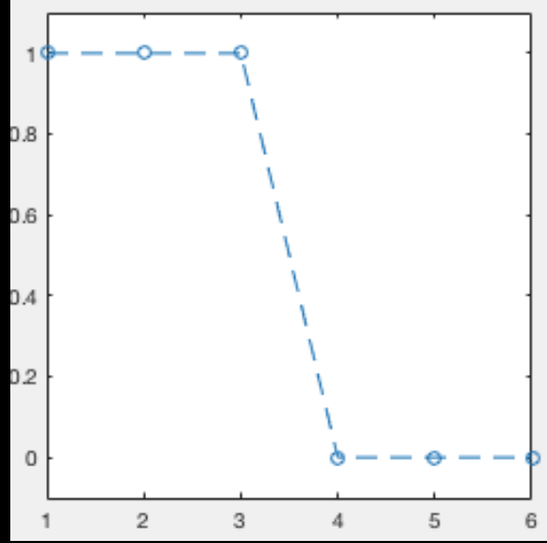
III. Filtrage

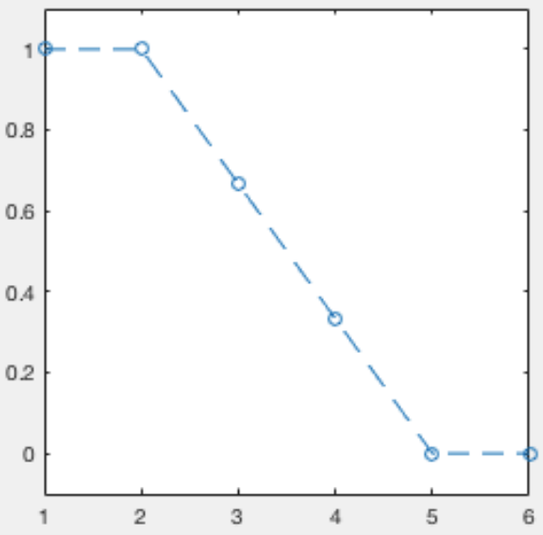


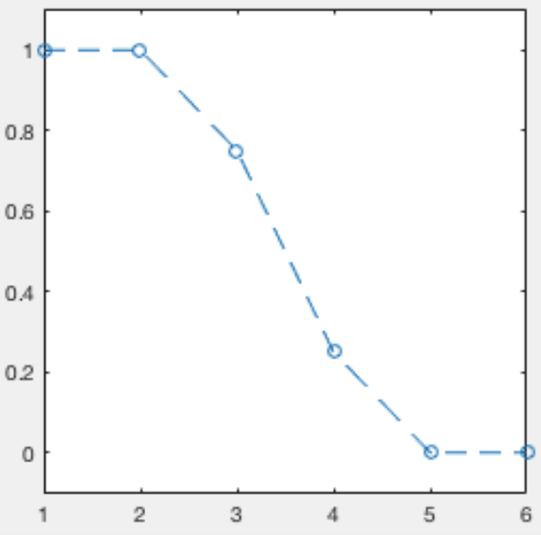


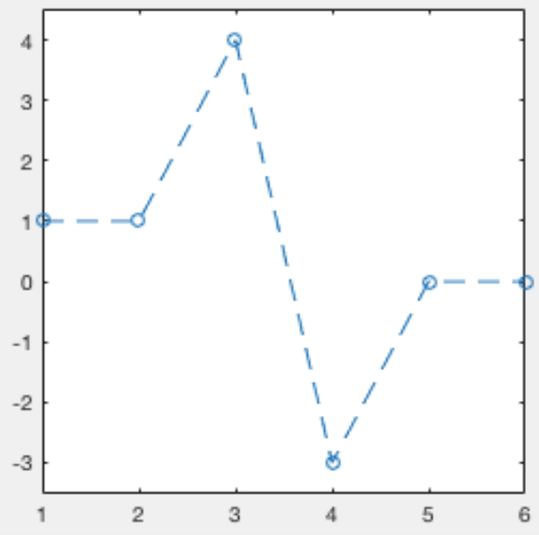












III. Filtrage

- **EXERCICE 6 : Reprendre le filtre moyenne glissante 3x3 sur une image de votre choix (i.e. celle du toit) et NE PAS respecter la normalisation. Puis diviser par 81 au lieu de 9. Que se passe-t-il ? (Utiliser pour comparaison *imshow* ET *imagesc*, bien penser à *colorbar*, utiliser les options « *same/full/valid* » pour *filter2*…)**