II. Analyse élémentaire d’images

- **EXERCICE 5 :**

  - Prendre l’image « frog » (p.ex.). Appliquer successivement les 3 bruits principaux (bruit de photon, bruit poivre et sel, bruit de lecture), puis représenter les histogrammes consécutifs ET leur différence d’avec le précédent (utiliser PLOT).

  - Calculer et afficher minimum, maximum et moyenne de chaque image.

  - Calculer et afficher également les distances (au sens des moindres carrés) entre images et histogrammes successifs et leurs précédents.

  - Données pour les bruits : *d*=5%, *variance_RON*=1%.

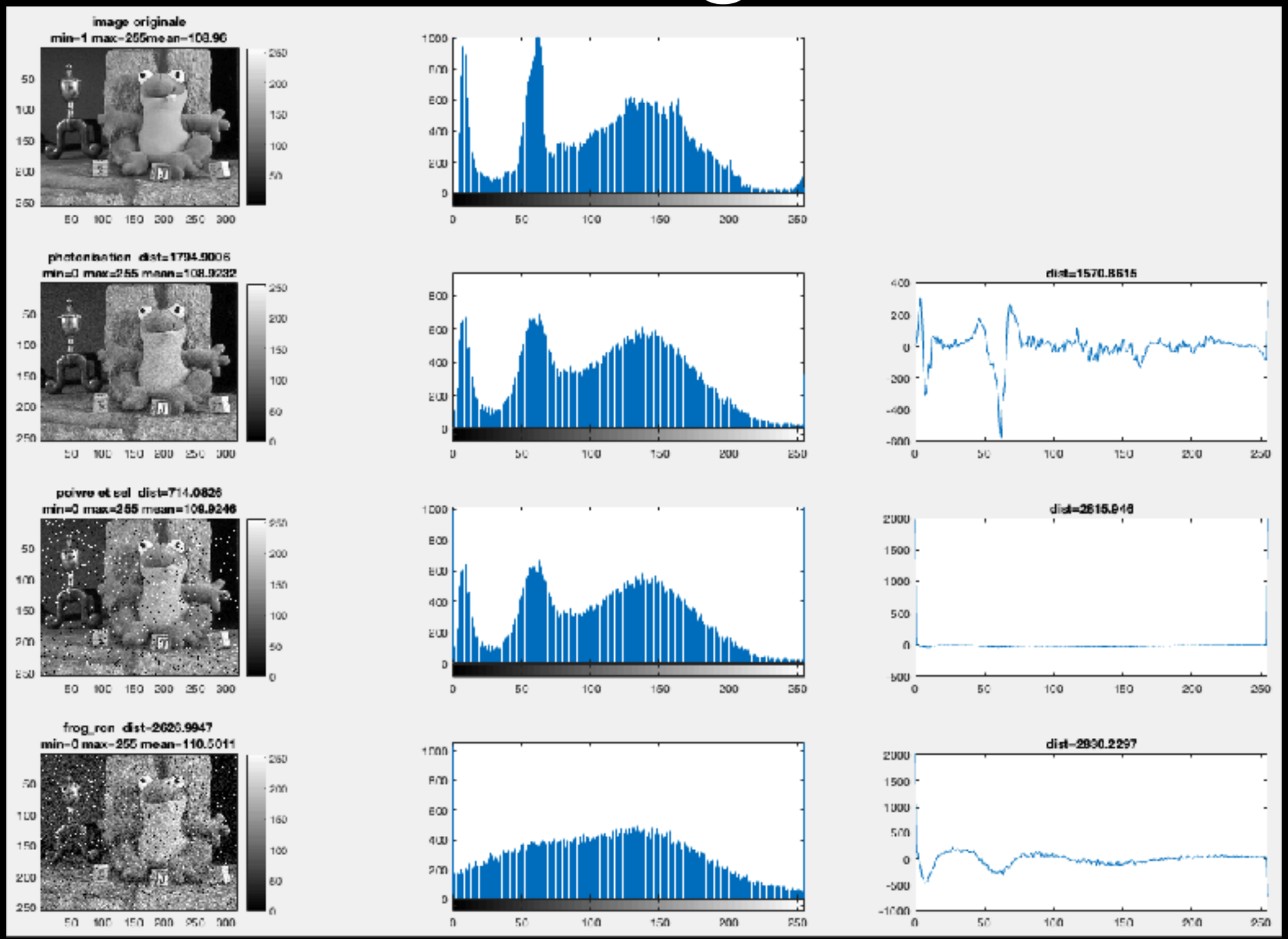
II. Analyse élémentaire d'images

```
clear
close all

frog = imread('/Users/marcel/Documents/MATLAB/GBM/0-images/frog.jpg');

% image originale
frog_int = rgb2gray(frog);
'type de frog_int :', whos frog_int
min_int = min(min(frog_int)); max_int = max(max(frog_int)); mean_int=mean(mean(frog_int));
[n_int,x_int]=imhist(frog_int,256);

figure
subplot(4,3,1), imagesc(frog_int), colormap(gray), colorbar, axis('image')
title({'image originale';['min=',num2str(min_int),' max=',num2str(max_int), 'mean=',num2str(mean_int)]})
subplot(4,3,2), imhist(frog_int,256)
%subplot(4,3,3), plot(x_int, n_int-n_int), xlim([0 255])

% bruit de photon
frog_pho = imnoise(frog_int, 'poisson');
'type de frog_pho :', whos frog_pho
min_pho = min(min(frog_pho)); max_pho = max(max(frog_pho)); mean_pho=mean(mean(frog_pho));
dist_im = sqrt(sum(sum((frog_pho-frog_int).^2)));
[n_pho,x_pho]=imhist(frog_pho,256);
dist_hist = sqrt(sum((n_pho-n_int).^2));

subplot(4,3,4), imagesc(frog_pho), colormap(gray), colorbar, axis('image')
title({['photonisation', '  dist=',num2str(dist_im)] ; ['min=',num2str(min_pho),' max=',num2str(max_pho),' mean=',num2str(mean_pho)
subplot(4,3,5), imhist(frog_pho,256)
subplot(4,3,6), plot(x_pho, n_pho-n_int), xlim([0 255]), title(['dist=',num2str(dist_hist)])
```

II. Analyse élémentaire d'images

```
% bruit poivre & sel
frog_snp = imnoise(frog_pho, 'salt & pepper', 0.05);
'type de frog_snp :', whos frog_snp
min_snp = min(min(frog_snp)); max_snp = max(max(frog_snp)); mean_snp=mean(mean(frog_snp));
dist_im = sqrt(sum(sum((frog_snp-frog_pho).^2)))
[n_snp,x_snp]=imhist(frog_snp,256);
dist_hist = sqrt(sum((n_snp-n_pho).^2))

subplot(4,3,7), imagesc(frog_snp), colormap(gray), colorbar, axis('image')
title({['poivre et sel', '  dist=',num2str(dist_im)] ; ['min=',num2str(min_snp),' max=',num2str(max_snp),' mean=',num2str(mean_snp)]})
subplot(4,3,8), imhist(frog_snp,256)
subplot(4,3,9), plot(x_snp, n_snp-n_pho), xlim([0 255]), title(['dist=',num2str(dist_hist)])

% bruit de lecture
frog_ree = double(frog_snp)/255.;
frog_ron = imnoise(frog_ree, 'gaussian', 0., .01);
'type de frog_ron :', whos frog_ron
%
frog_ron = uint8(frog_ron*255);
%
'type de frog_ron :', whos frog_ron
min_ron = min(min(frog_ron)); max_ron = max(max(frog_ron)); mean_ron=mean(mean(frog_ron));
%
dist_im = sqrt(sum(sum((frog_ron-frog_snp).^2)))
%dist_im = sqrt(sum(sum((frog_ron-double(frog_snp)/255.).^2)))
%
[n_ron,x_ron]=imhist(frog_ron,256);
dist_hist = sqrt(sum((n_ron-n_snp).^2))

subplot(4,3,10), imagesc(frog_ron), colormap(gray), colorbar, axis('image')
title({['frog\_ron','  dist=',num2str(dist_im)] ; ['min=',num2str(min_ron),' max=',num2str(max_ron),' mean=',num2str(mean_ron)]})
subplot(4,3,11), imhist(frog_ron,256)
subplot(4,3,12), plot(x_ron, n_ron-n_snp), xlim([0 255]), title(['dist=',num2str(dist_hist)])
```

II. Analyse élémentaire d'images



II. Analyse élémentaire d'images

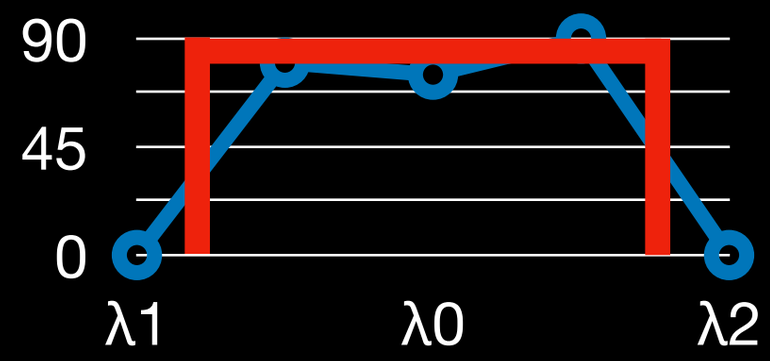
- **Il faut enfin considérer le problème de l'échantillonnage de l'image par le détecteur, échantillonnage qui doit respecter le critère de SHANNON (ou Nyquist). Ici : taille du pixel ≤ 1/2 λ/D (en unités angulaires), mais aussi ne pas être trop petit afin d'être moins sensible aux bruits (et ne pas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

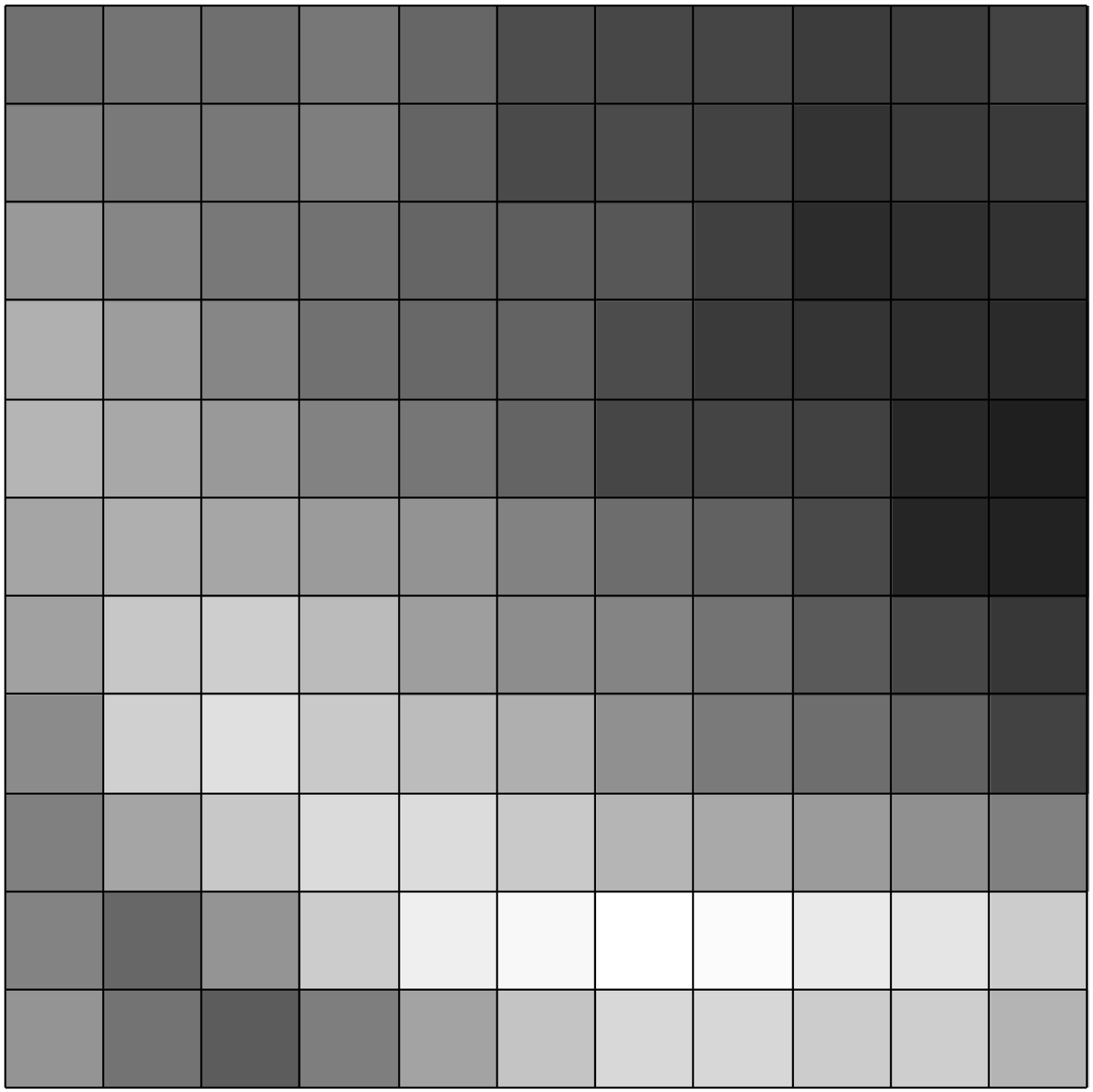
- **On remarque tout de suite qu'il s'agit d'une convolution si on fait subir au filtre une symétrie par rapport à l'origine (i.e. une rotation de 180°) :**

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

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.**

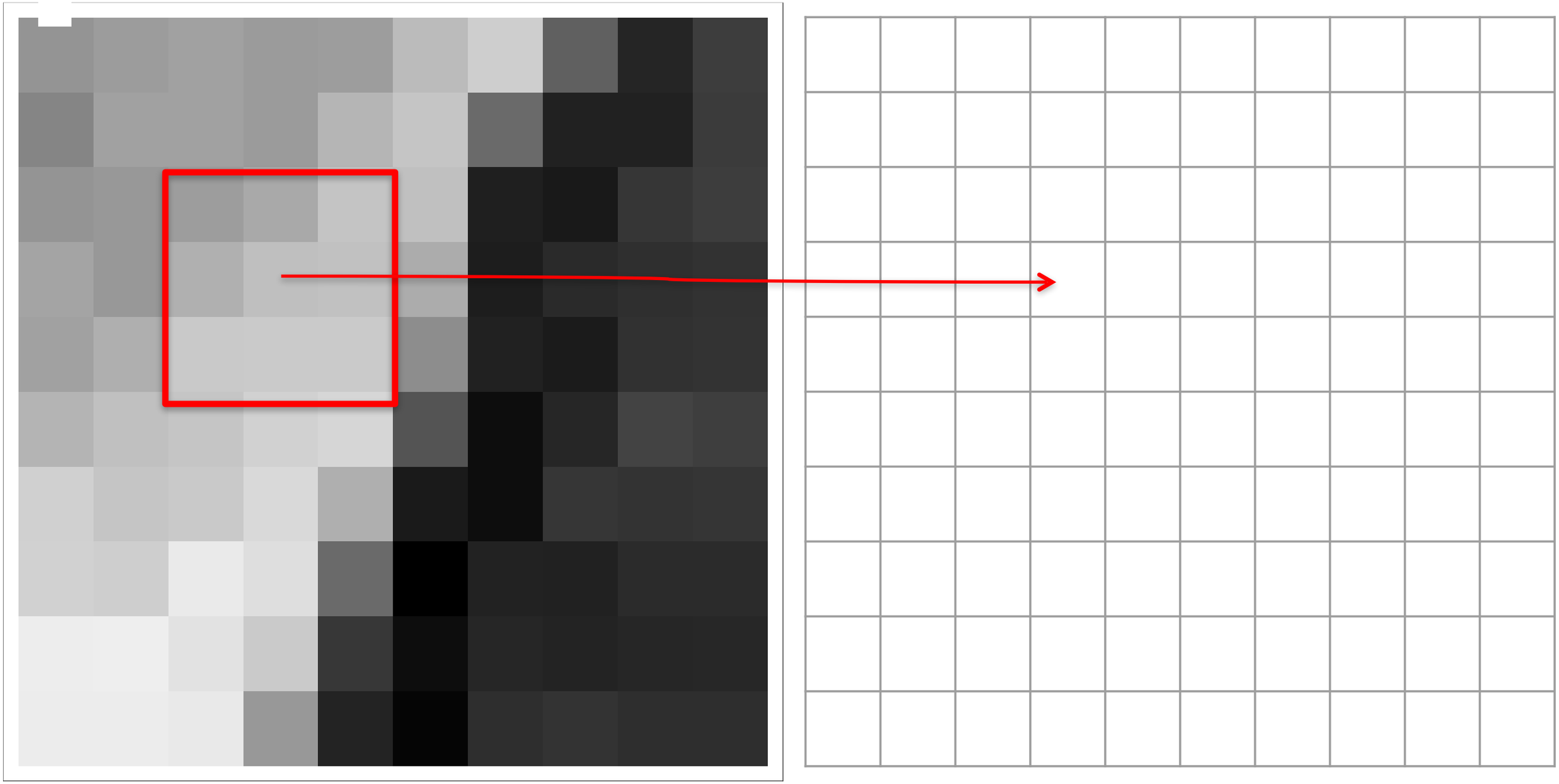
Passage d’un filtre linéaire

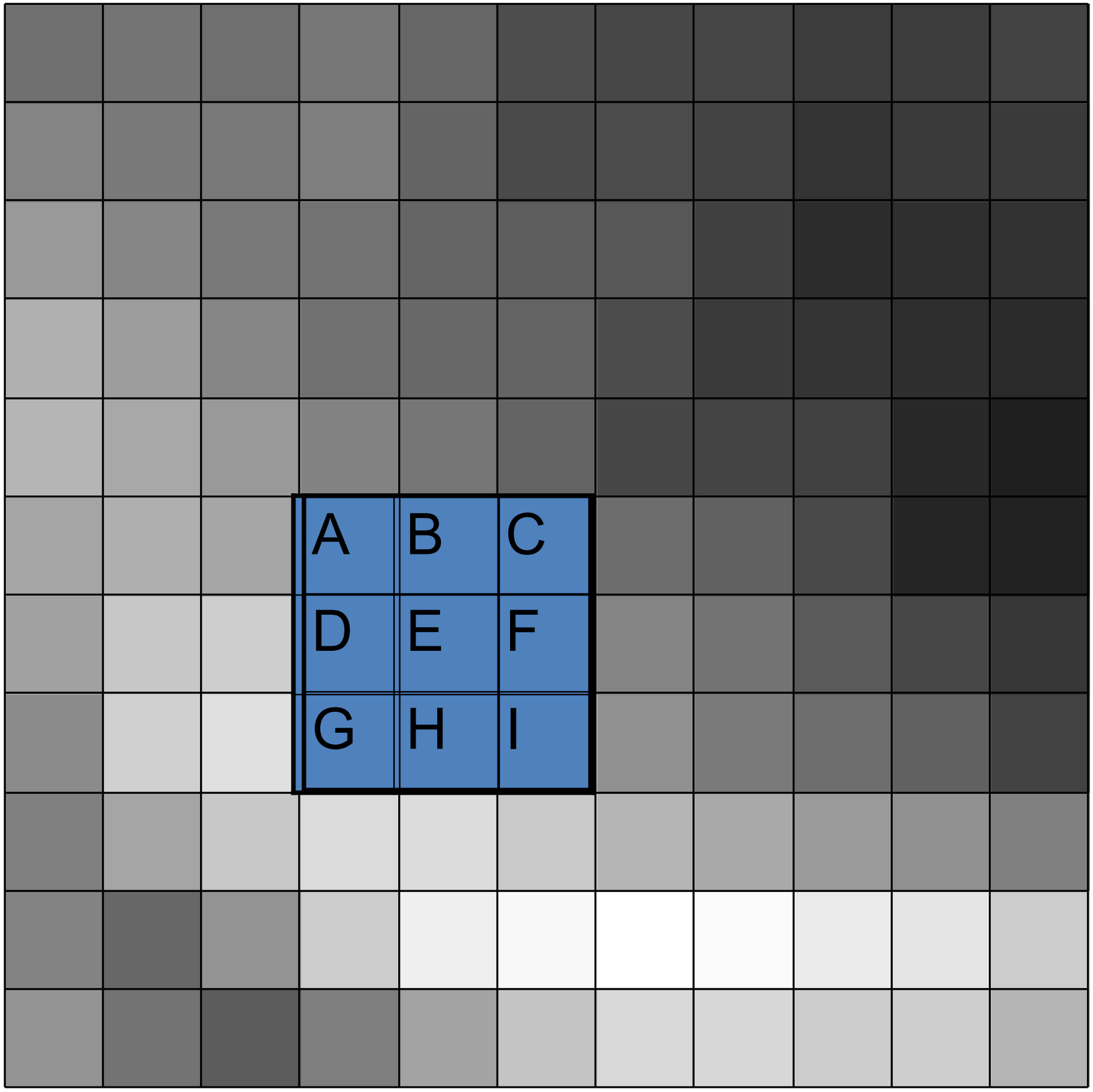


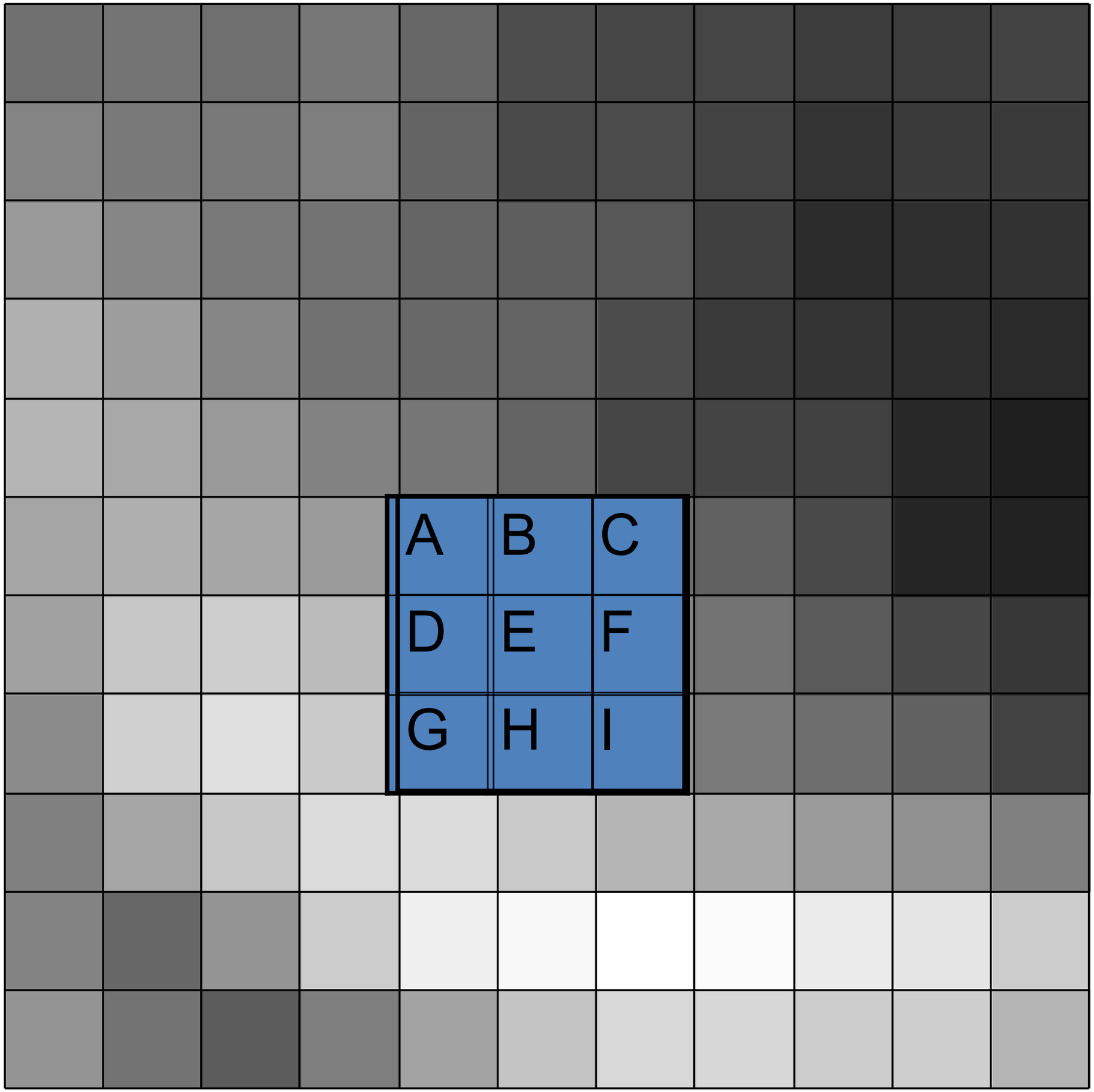
⊗

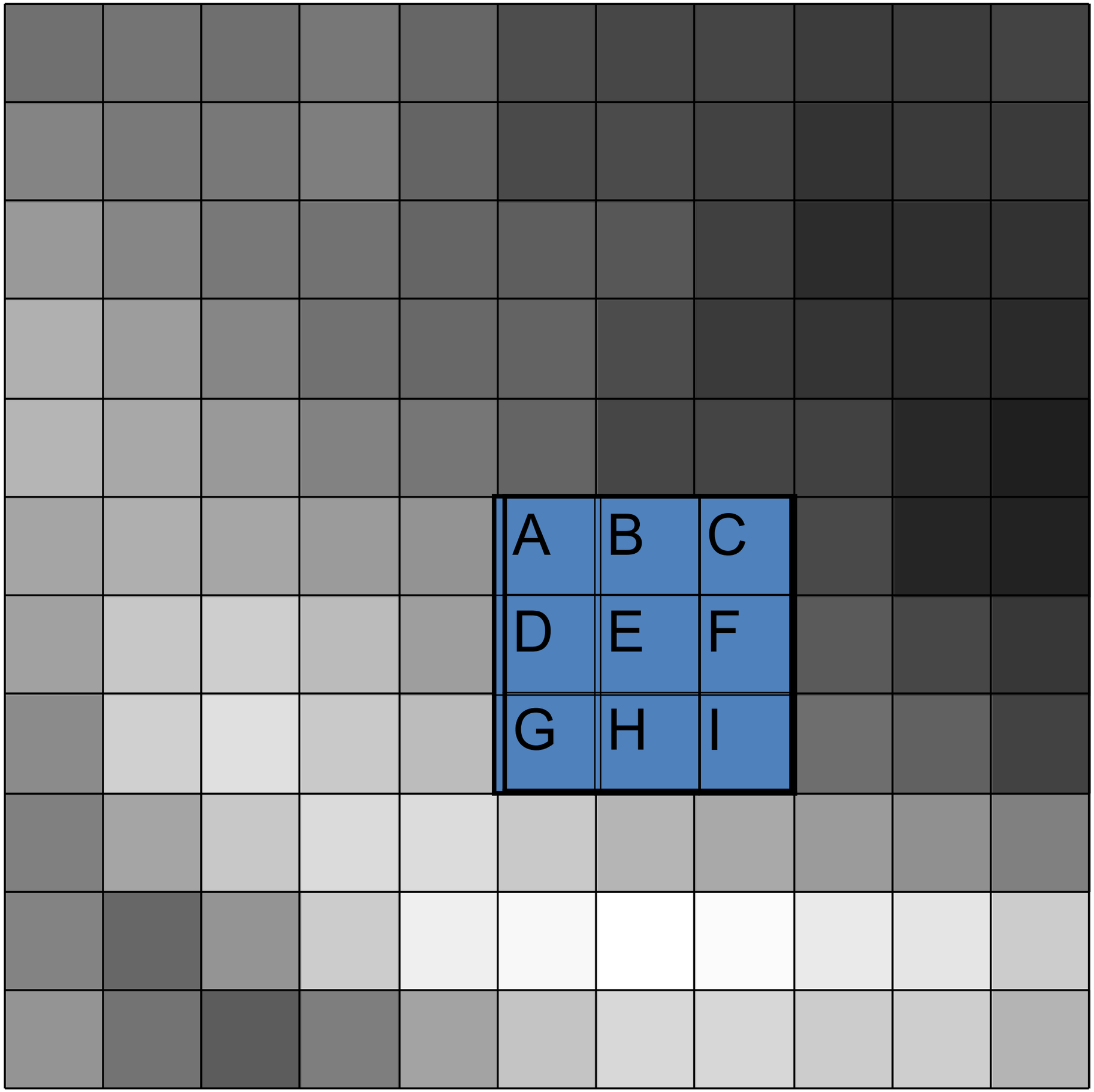
A	B	C
D	E	F
G	H	I

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

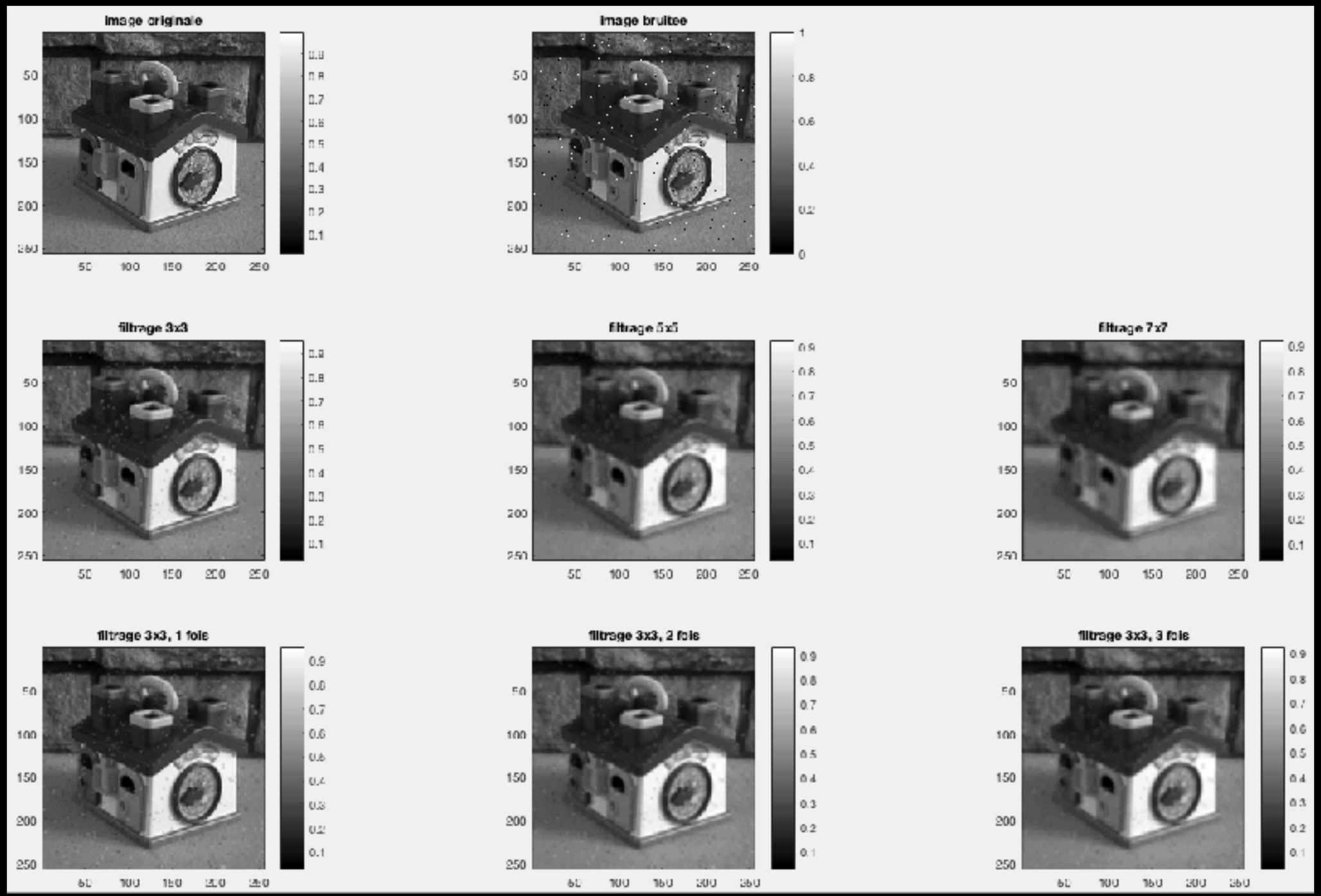
- **EXERCICE 1 : Appliquer du bruit ‘poivre & sel’ (1%) à l’image de votre choix avant de la filtrer. Jugez de l’amélioration en fonction de la taille de *h* (3x3, 5x5 ou 7x7)…**

- **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)…**