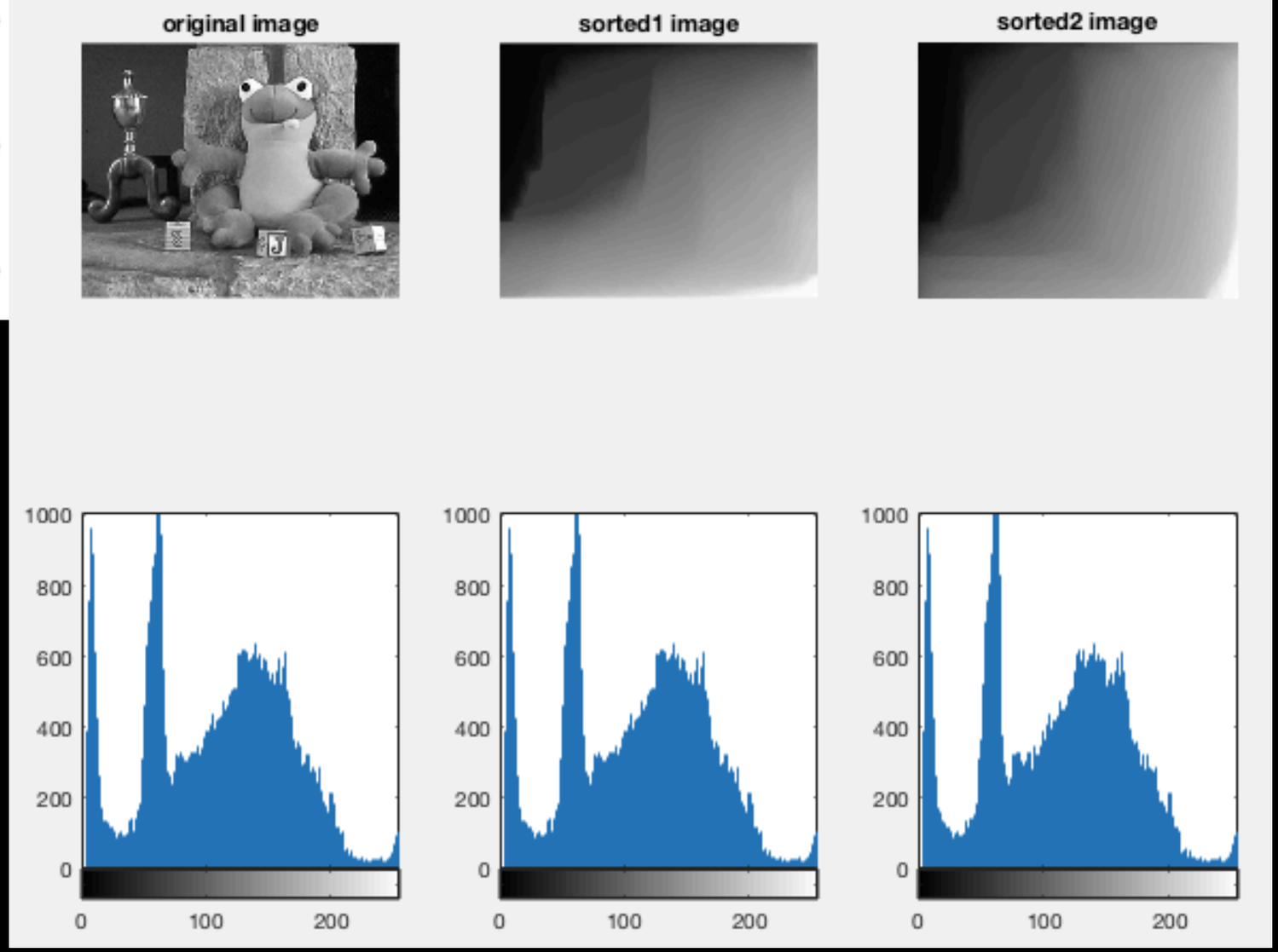
II. Analyse élémentaire d’images

- **EXERCICE 3bis : Appliquer à une image plus complexes (p.ex. la grenouille en niveaux de gris)…**

II. Analyse élémentaire d'images

```
1 -    clear
2 -    close all
3
4 -    frog=imread('/Users/marcel/Documents/MATLAB/0-images-exemples/classiques/frog.jpg');
5 -    frog=rgb2gray(frog);
6
7 -    whos frog
8
9 -    figure
10 -   subplot(2,3,1), imshow(frog), title('original image')
11 -   subplot(2,3,4), imhist(frog,256)
12
13 -   frog1=sort(sort(frog),2);
14 -   subplot(2,3,2), imshow(frog1), title('sorted1 image')
15 -   subplot(2,3,5), imhist(frog1,256)
16
17 -   frog2=sort(sort(frog,2));
18 -   subplot(2,3,3), imshow(frog2), title('sorted2 image')
19 -   subplot(2,3,6), imhist(frog2,256)
```



II. Analyse élémentaire d’images

4- COUPE D’UNE IMAGE

- **Afficher une image, lancer IMPROFILE, un click pour le 1er point, deux clicks pour le 2me.**
- **En alternative : PLOT (p.ex. : >> plot(frog_int(:,20)) —> plot de la col. n. 20)**

II. Analyse élémentaire d'images

5- AMÉLIORATION DU CONTRASTE

- **IMADJUST : sature 1% des basses et hautes intensités et fait en sorte d'utiliser toute la dynamique disponible (en redistribuant les intensités pour utiliser toute la gamme de représentation permise (par uint8 p.ex.)).**
  >> I_adj = imadjust(I)

- **On peut aussi choisir la gamme de valeurs d'intensité sur laquelle agir :**
  >> I_adj2 = imadjust(I, [0, 0.5], [])
  (« [0, 0.5] » correspondant à [0, moitié des niveaux possibles] - p.ex. !)

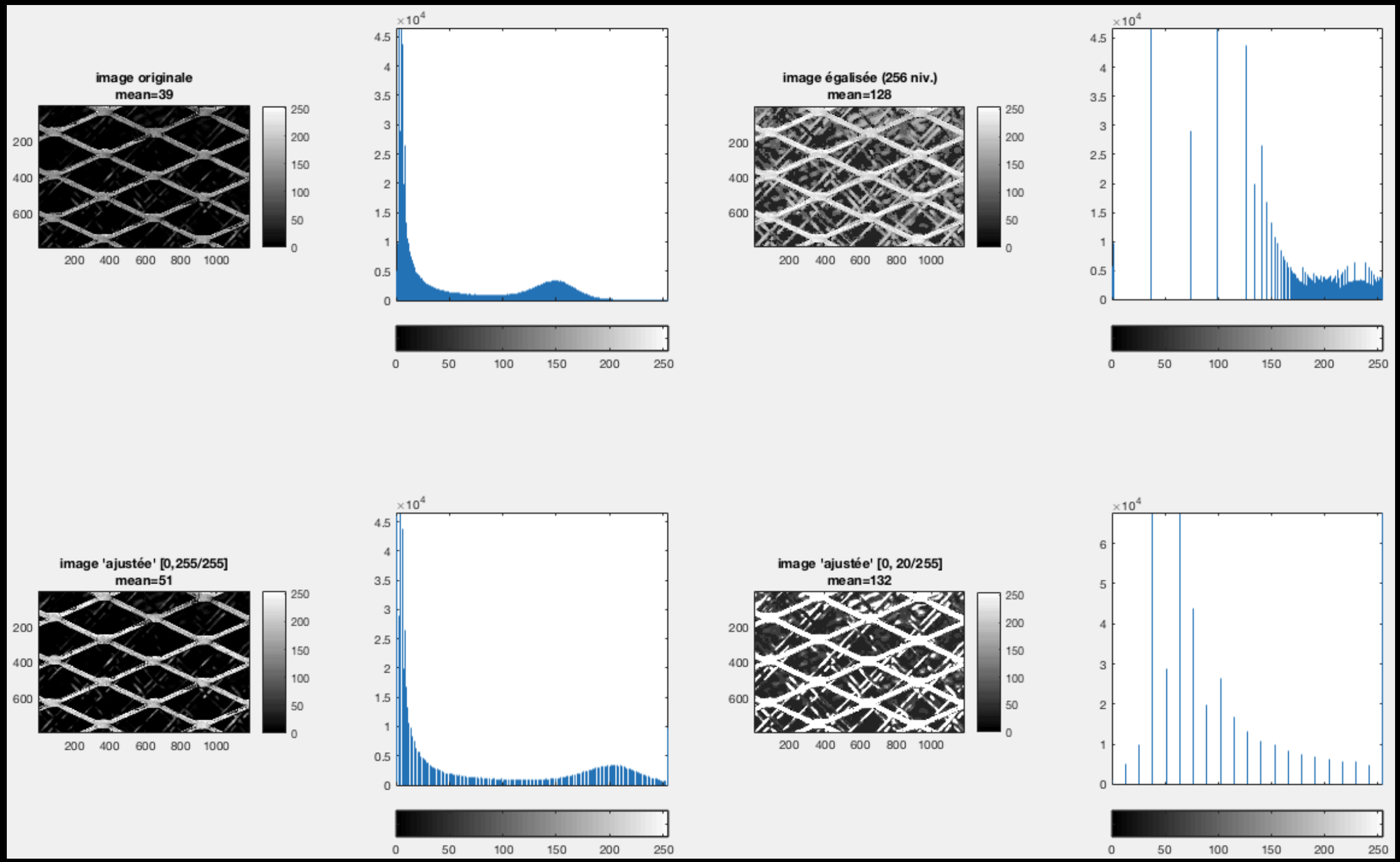
II. Analyse élémentaire d’images

- **EXERCICE 4 : Reprendre l’image du grillages et comparer le résultat de l’égalisation d’histogramme (sur 256 niveaux) à celui d’IMADJUST, puis d’IMADJUST en adaptant la gamme des valeurs d’entrée. Calculer et afficher la valeur moyenne de chaque image et comparer image, histogramme et valeur moyenne de l’image.**

II. Analyse élémentaire d'images

```
clear
close all

% original image
I=imread('/Users/marcel/Documents/MATLAB/0-images-exemples/materiaux/serveimage-8.jpeg'
I_int=rgb2gray(I);
mean_int = mean(mean(I_int));

figure(1)
subplot(2,4,1), imagesc(I_int), colormap(gray), colorbar, axis('image')
title({'image originale' ; ['mean=',int2str(mean_int)]})
subplot(2,4,2), imhist(I_int, 256), axis('square')

% equalized image (256)
n=256
I_eq = histeq(I_int, n);
mean_eq = mean(mean(I_eq));

subplot(2,4,3), imagesc(I_eq), colormap(gray), colorbar, axis('image')
title({'image égalisée (256 niv.)' ; ['mean=',int2str(mean_eq)]})
subplot(2,4,4), imhist(I_eq, n), axis('square')

% "adjusted" image
I_adj = imadjust(I_int);
mean_adj = mean(mean(I_adj));

subplot(2,4,5), imagesc(I_adj), colormap(gray), colorbar, axis('image')
title({'image ''ajustée'' [0,255/255]' ; ['mean=',int2str(mean_adj)]})
subplot(2,4,6), imhist(I_adj, 256), axis('square')

% "adjusted" image with selected input range to adjust
I_adj2 = imadjust(I_int, [0., 20./255], []);
mean_adj2 = mean(mean(I_adj2));

subplot(2,4,7), imagesc(I_adj2), colormap(gray), colorbar, axis('image')
title({'image ''ajustée'' [0, 20/255]' ; ['mean=',int2str(mean_adj2)]})

subplot(2,4,8), imhist(I_adj2, 256), axis('square')
```

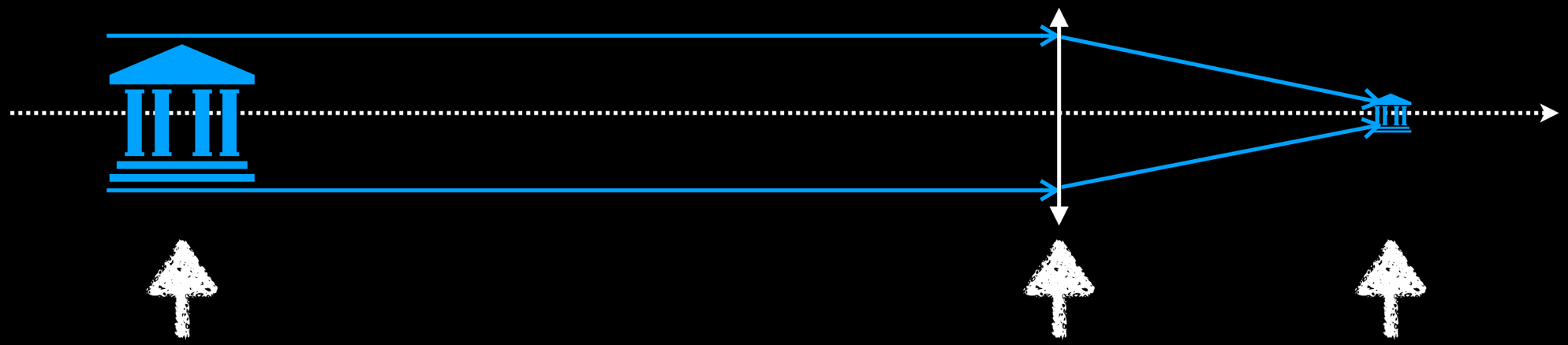
II. Analyse élémentaire d'images



II. Analyse élémentaire d’images

6- AU FAIT, QU’EST-CE QU’UNE IMAGE ?…

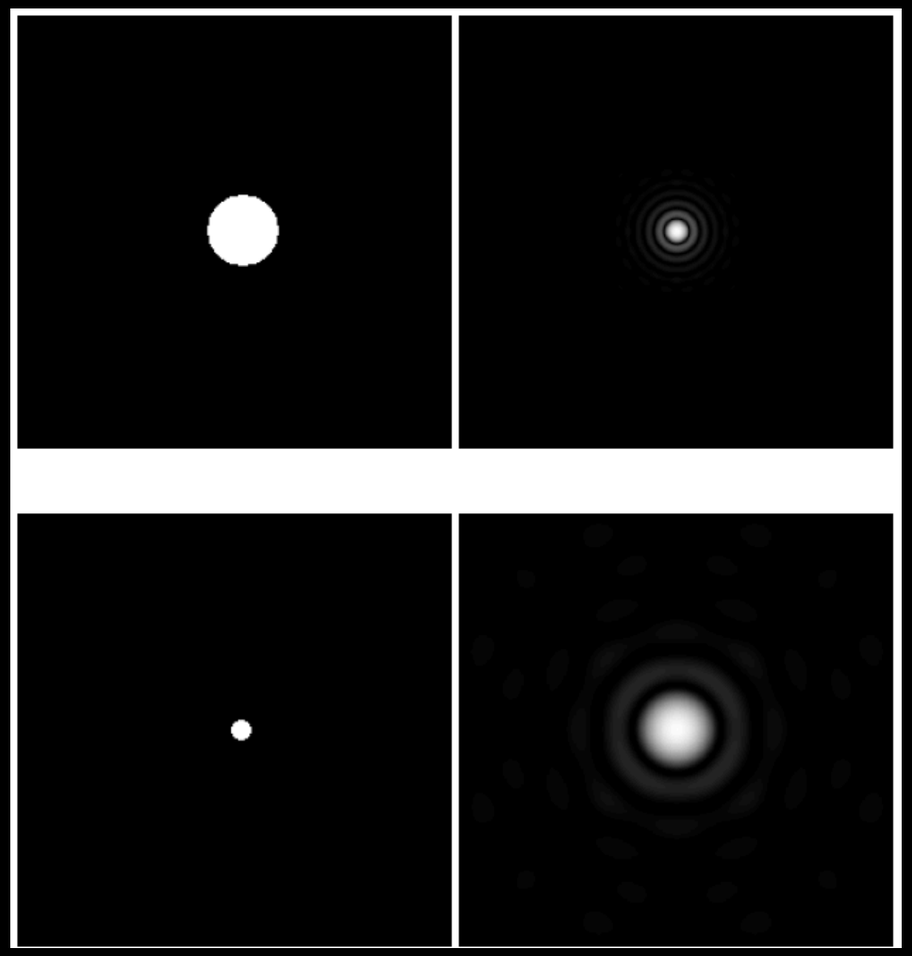
- **Formation d’une image (optique) :**



L’image vue (=détectée) au foyer est l’image d’un point convoluée par l’objet.

II. Analyse élémentaire d'images

Or : l'image d'un point est une « tache d'Airy », dont le cœur est de largeur à mi-hauteur ~λf/D (ou ~λ/D en unités angulaires), avec λ la longueur d'onde.



=> plus D est grand, plus la résolution dans l'image (inversement proportionnelle à λ/D) sera importante (pour une λ donnée).

II. Analyse élémentaire d'images

- **Détection :**

  L'image précédente est ensuite détectée par un… détecteur, par exemple la matrice CCD (ou CMOS, ou EMCCD, ou autre technologie) qui équipe votre smartphone, ou n'importe quel appareil « imageur » plus sophistiqué. Et cette détection est assujettie à plusieurs bruits…

- Le bruit de photons (ou *photon noise*, ou *shot noise*) qui suit une distribution de Poisson :

p(n) = probabilité de détecter n photons quand N sont attendus

$$p(n) = \frac{N^n \, e^{-N}}{n!}, \text{ avec } : \sigma^2 = N$$

sous Matlab :
>> image_phot = imnoise(image_int, 'poisson') ;

II. Analyse élémentaire d'images

- **Le bruit de lecture (ou *read-out noise*, *RON*) qui est un bruit ADDITIF caractérisé par une distribution de Gauss (ou 'normale') de moyenne nulle et d'écart-type σ_e :**

$$p(x) = \frac{1}{\sigma_e \sqrt{2\pi}} \, e^{-\frac{x^2}{2\sigma_e^2}}$$

sous Matlab :
>> image_ron = imnoise(image_double, 'gaussian', 0.0, 0.01) ;

image en double précision entre 0 et 1 — **moyenne** — **variance relative**

II. Analyse élémentaire d'images

- **Entre les deux (dans la même chaîne d'acquisition d'images), apparaissent d'autres bruits électroniques :**

  • pixels « chauds » et « froids » sur le CCD => bruits de type « poivre et sel »

  sous Matlab :
  >> image_snp = imnoise(image_int, 'salt & pepper', d)

  densité de bruit
  (=% de pixels affectés)

• bruit de courant d'obscurité, bruits spécifiques à des détecteurs « exotiques » ou des applications très pointues, etc.

• « bruits périodiques » (par.ex. tramage dû à un scan ou une compression jpeg trop forte) => en fait plutôt un BIAIS qu'un véritable bruit (pas aléatoire).

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 et maximum 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%.

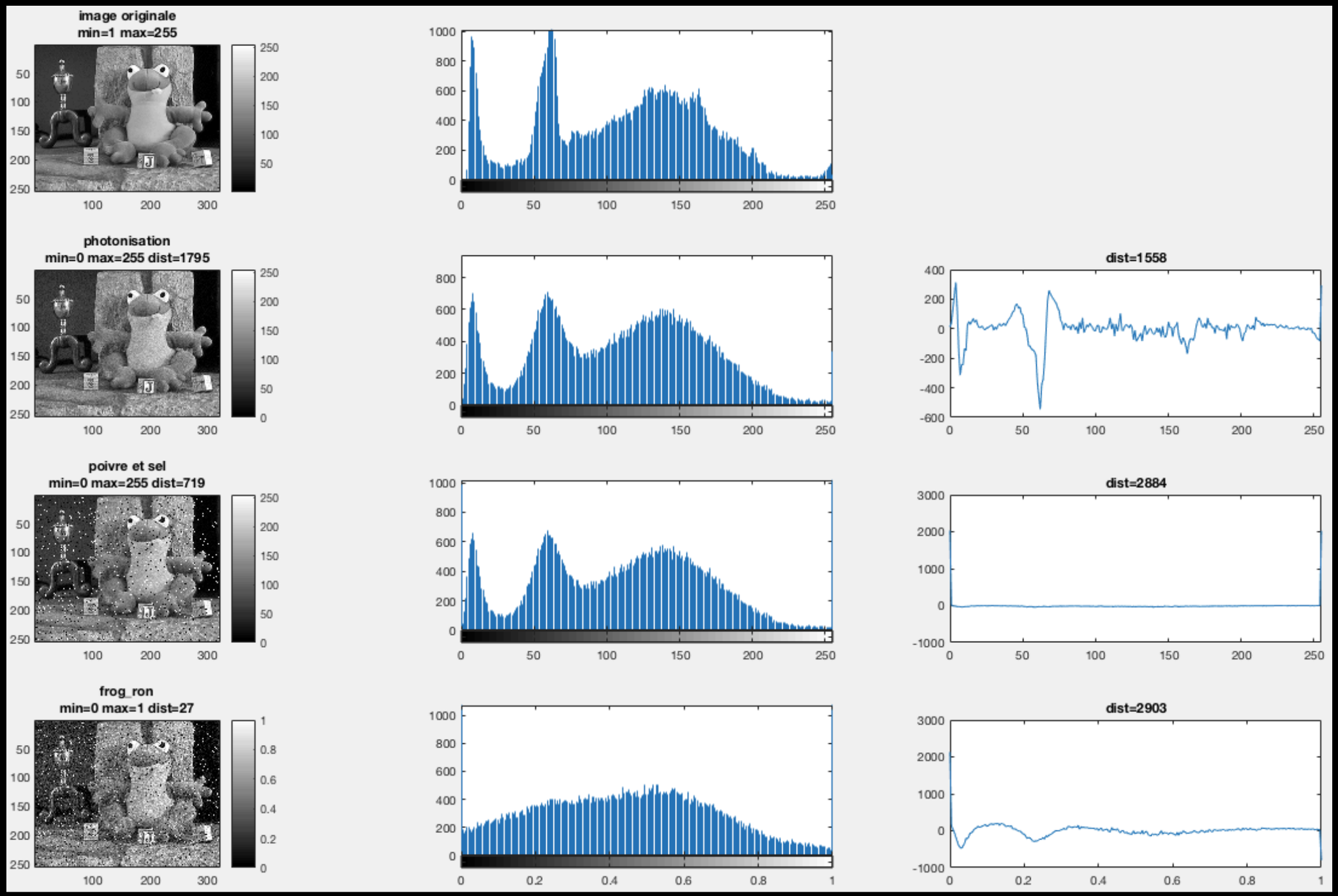
II. Analyse élémentaire d'images

```
clear
close all

frog = imread('/Users/marcel/Documents/MATLAB/0-images-exemples/classiques/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));
[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=',int2str(min_int),' max=',int2str(max_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));
dist_im   = sqrt(sum(sum((frog_pho-frog_int).^2)))
[n_pho,x_pho]=imhist(frog_pho,256);
dist_hist = sqrt(sum(sum((n_pho-n_int).^2)))

subplot(4,3,4), imagesc(frog_pho), colormap(gray), colorbar, axis('image')
title({'photonisation' ; ['min=',int2str(min_pho),' max=',int2str(max_pho),' dist=',int2str(dist_im)]})
subplot(4,3,5), imhist(frog_pho,256)
subplot(4,3,6), plot(x_pho, n_pho-n_int), xlim([0 255]), title(['dist=',int2str(dist_hist)])
```

II. Analyse élémentaire d’images

```
31      % bruit poivre & sel
32 -    frog_snp = imnoise(frog_pho, 'salt & pepper', 0.05);
33 -    'type de frog_snp :', whos frog_snp
34 -    min_snp = min(min(frog_snp)); max_snp = max(max(frog_snp));
35 -    dist_im   = sqrt(sum(sum((frog_snp-frog_pho).^2)))
36 -    [n_snp,x_snp]=imhist(frog_snp,256);
37 -    dist_hist = sqrt(sum(sum((n_snp-n_pho).^2)))
38
39 -    subplot(4,3,7), imagesc(frog_snp), colormap(gray), colorbar, axis('image')
40 -    title({'poivre et sel' ; ['min=',int2str(min_snp),' max=',int2str(max_snp),' dist=',int2str(dist_im)]})
41 -    subplot(4,3,8), imhist(frog_snp,256)
42 -    subplot(4,3,9), plot(x_snp, n_snp-n_pho), xlim([0 255]), title(['dist=',int2str(dist_hist)])
43
44      % bruit de lecture
45 -    frog_ree = double(frog_snp)/255.;
46 -    frog_ron = imnoise(frog_ree, 'gaussian', 0., .01);
47 -    'type de frog_ron :', whos frog_ron
48 -    min_ron = min(min(frog_ron)); max_ron = max(max(frog_ron));
49 -    dist_im   = sqrt(sum(sum((frog_ron-double(frog_snp)/255.).^2)))
50 -    [n_ron,x_ron]=imhist(frog_ron,256);
51 -    dist_hist = sqrt(sum(sum((n_ron-n_snp).^2)))
52
53 -    subplot(4,3,10), imagesc(frog_ron), colormap(gray), colorbar, axis('image')
54 -    title({'frog\_ron' ; ['min=',int2str(min_ron),' max=',int2str(max_ron),' dist=',int2str(dist_im)]})
55 -    subplot(4,3,11), imhist(frog_ron,256)
56 -    subplot(4,3,12), plot(x_ron, n_ron-n_snp), xlim([0 1]), title(['dist=',int2str(dist_hist)])
```

II. Analyse élémentaire d'images



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, ~~filtrage dans le plan de Fourier~~, filtre médian.**
- **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**

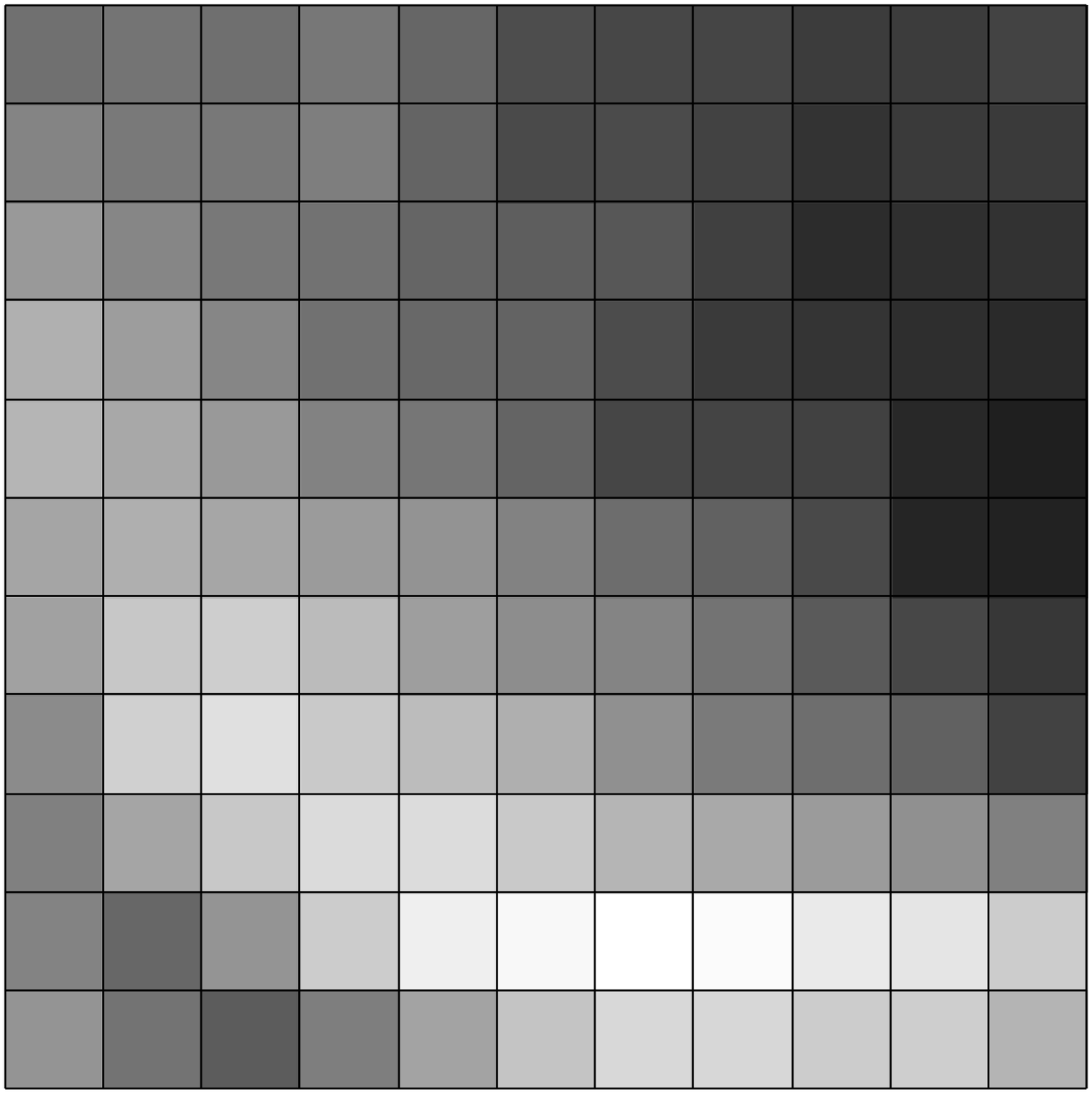
- **On remarque tout-de-suite qu'il s'agit d'une convolution si on fait subir au filtre une symétrie par rapport à l'origine (i.e. une rotation de 180°) :**

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

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.**

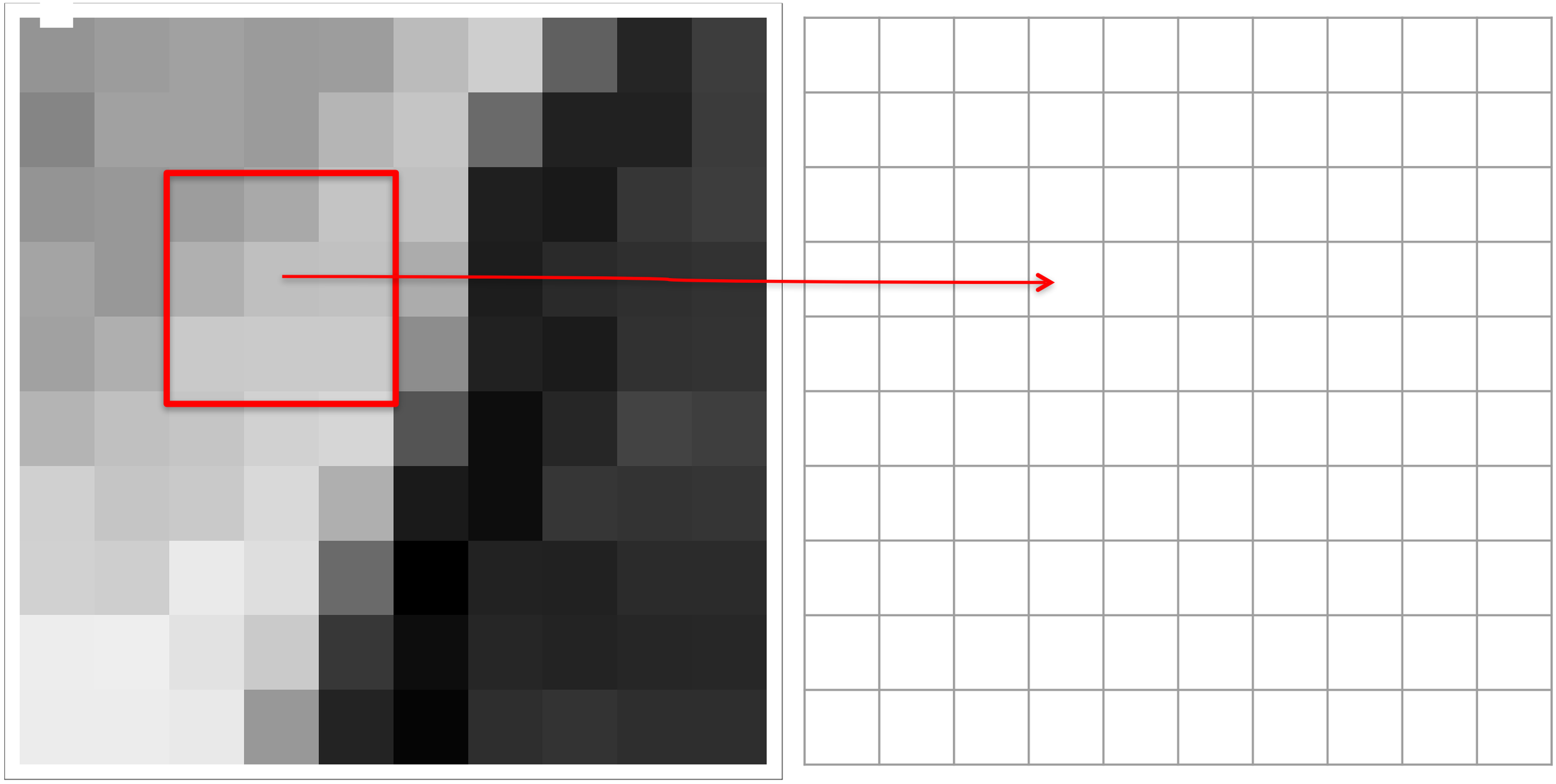
Passage d’un filtre linéaire

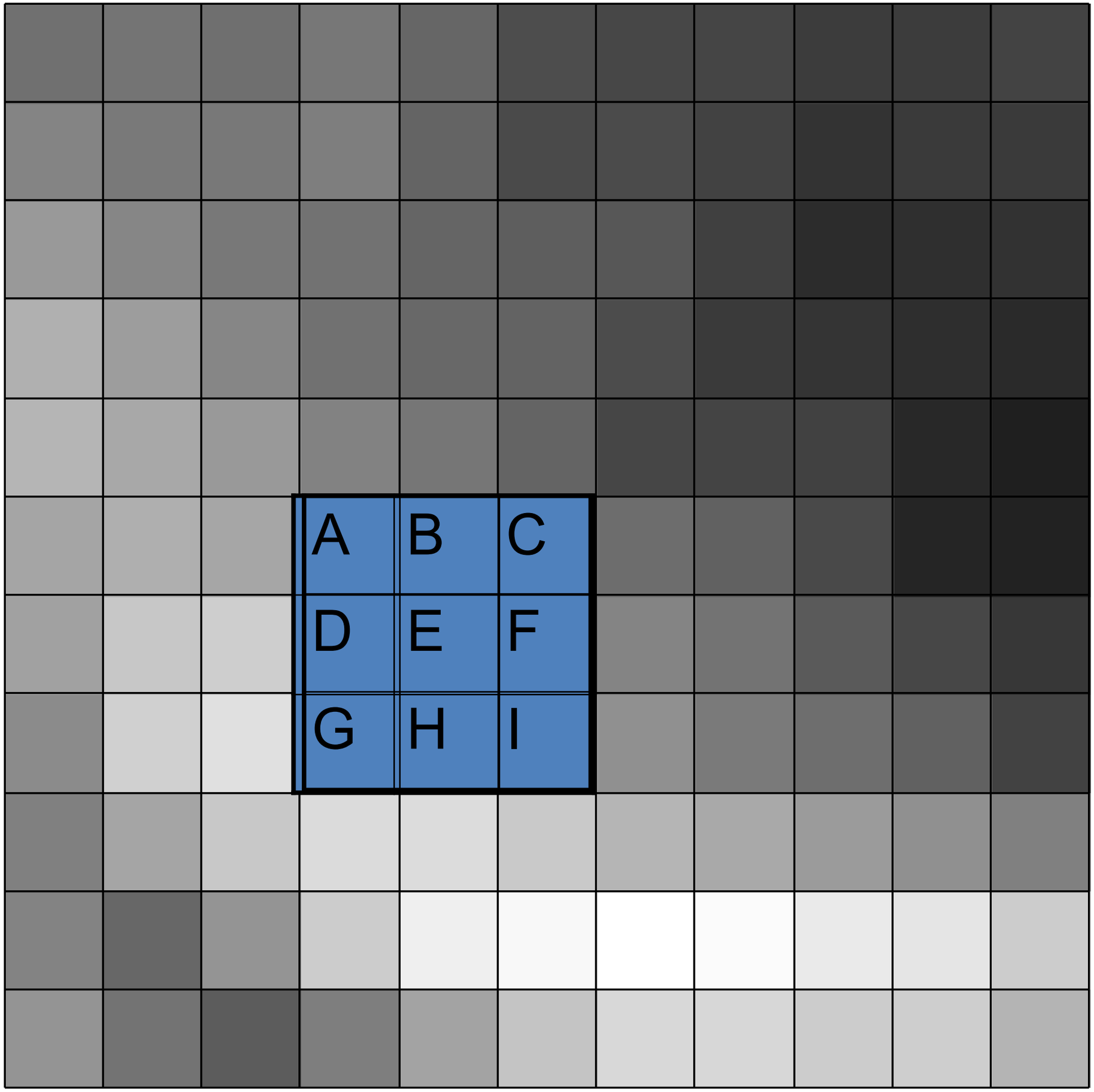


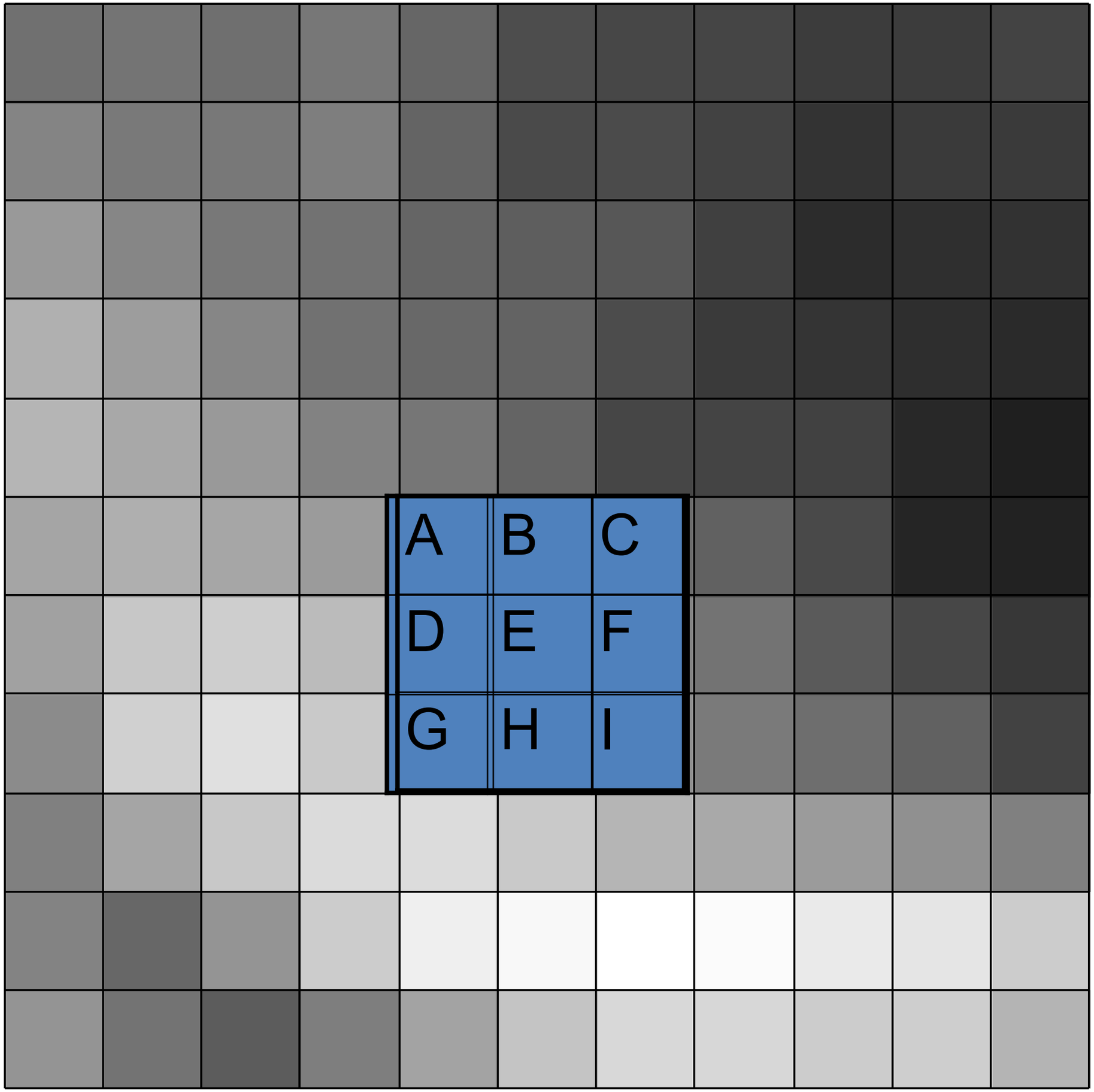
⊗

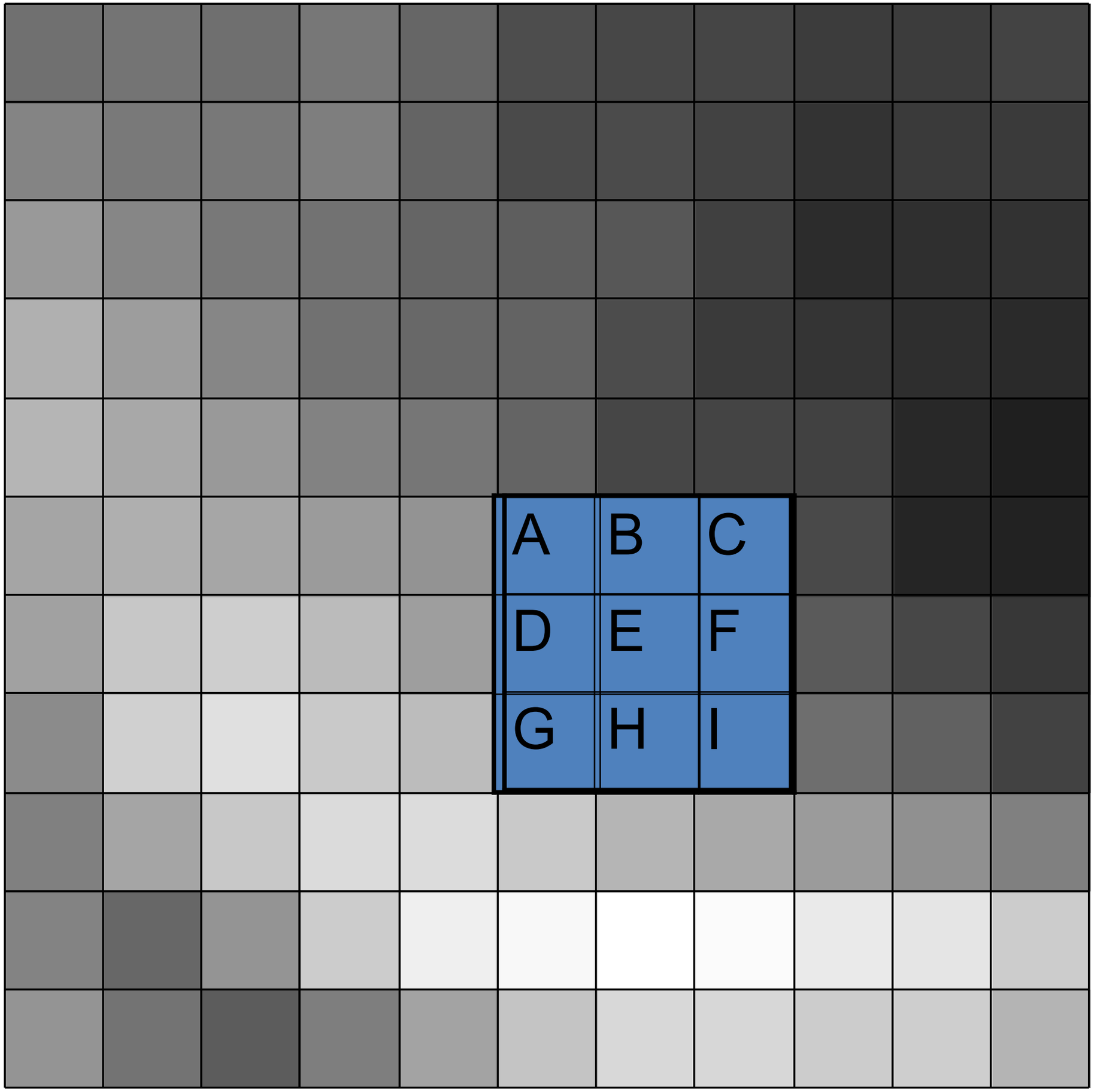
A	B	C
D	E	F
G	H	I

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 : FILTER2(filtre, image)**
  —> rend l'image filtrée

  >> If = filter2(h, I);

- **Remarque : *h* est en général 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 : Ajouter 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*…**

- **EXERCICE 2 : Même chose en fonction du nombre de passages de *h*…**

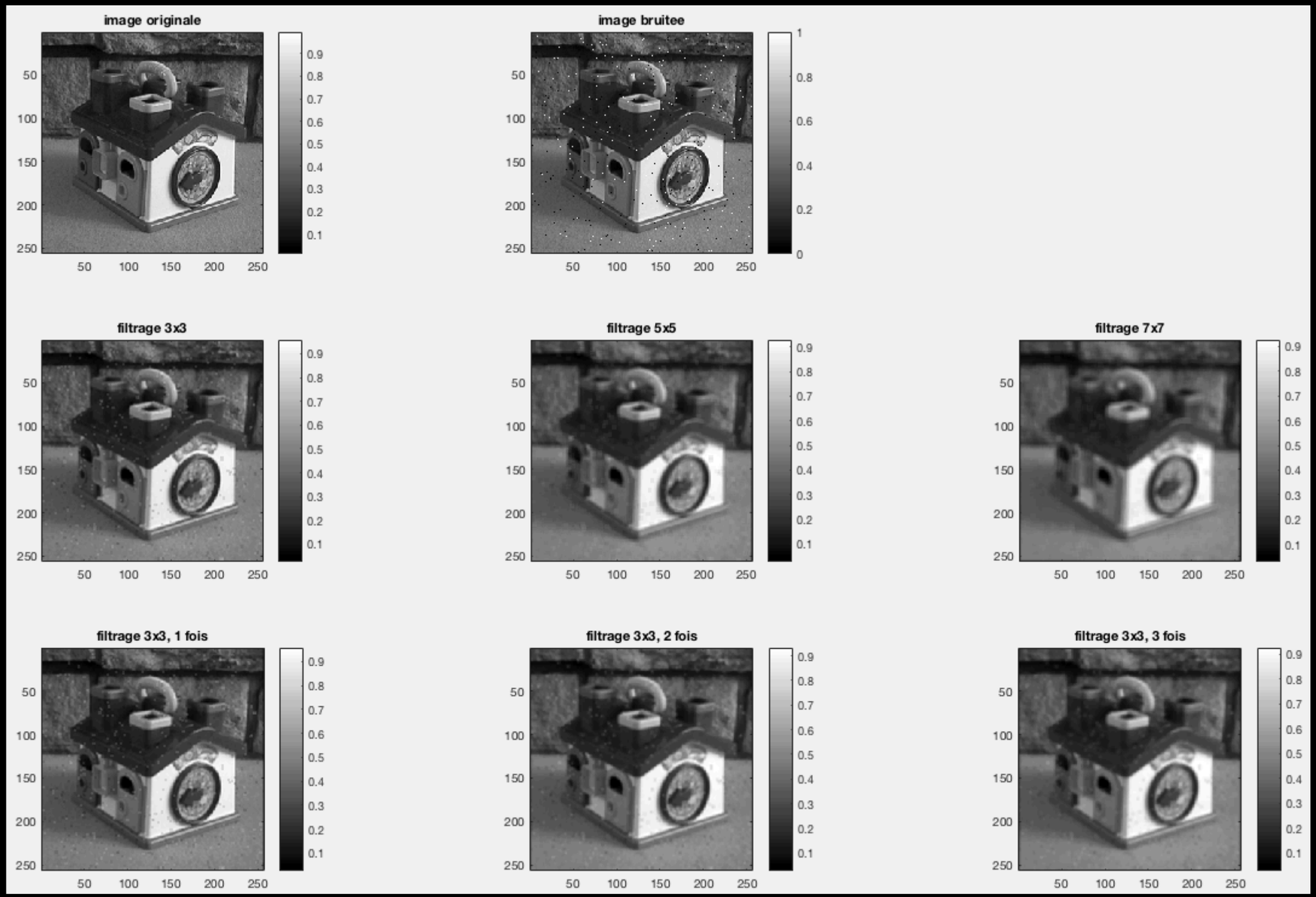
III. Filtrage

```
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

```
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



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)…**