III. Filtrage

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

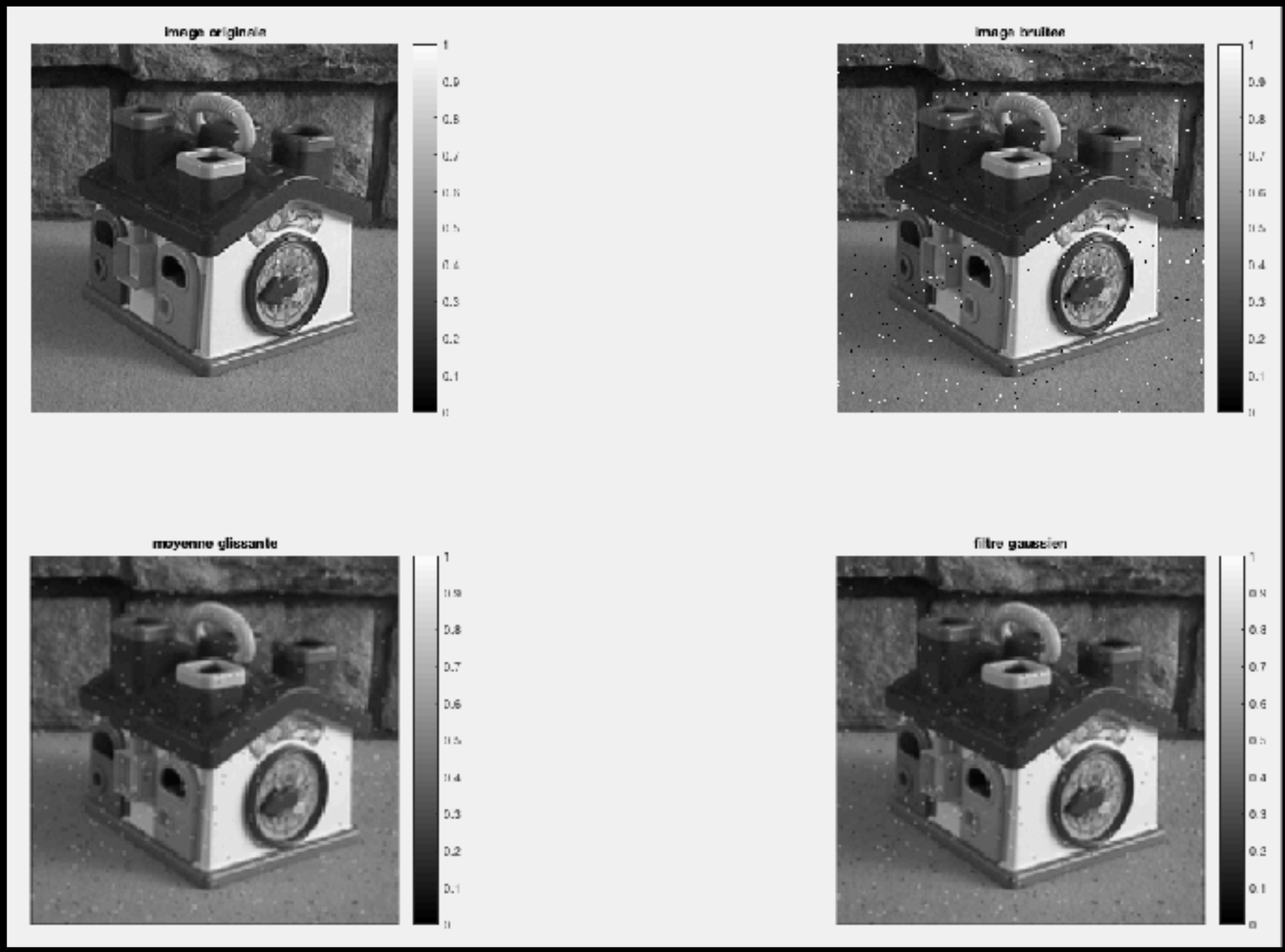
$$h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- **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.*)

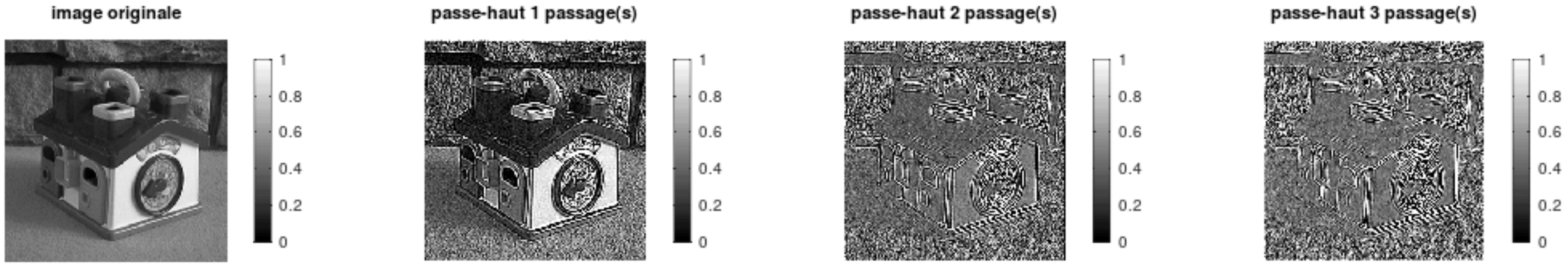
III. Filtrage

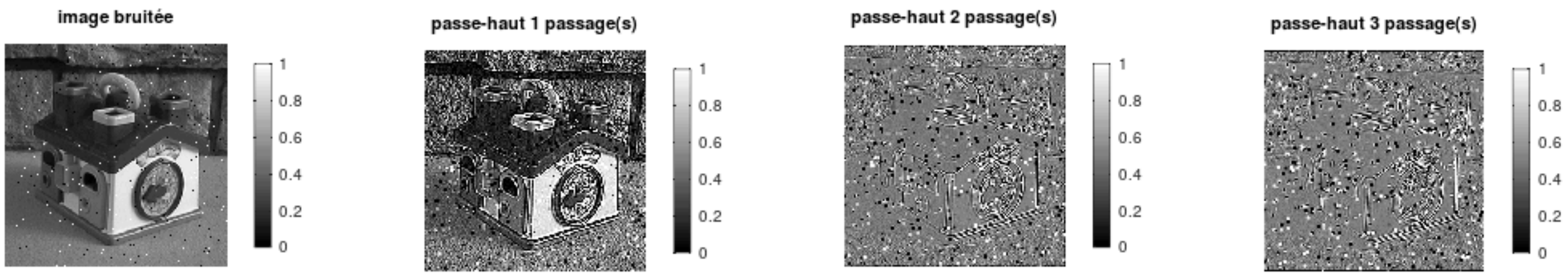
```
clear
close all
%pkg load image
%(pour Octave, pas Matlab)

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
```

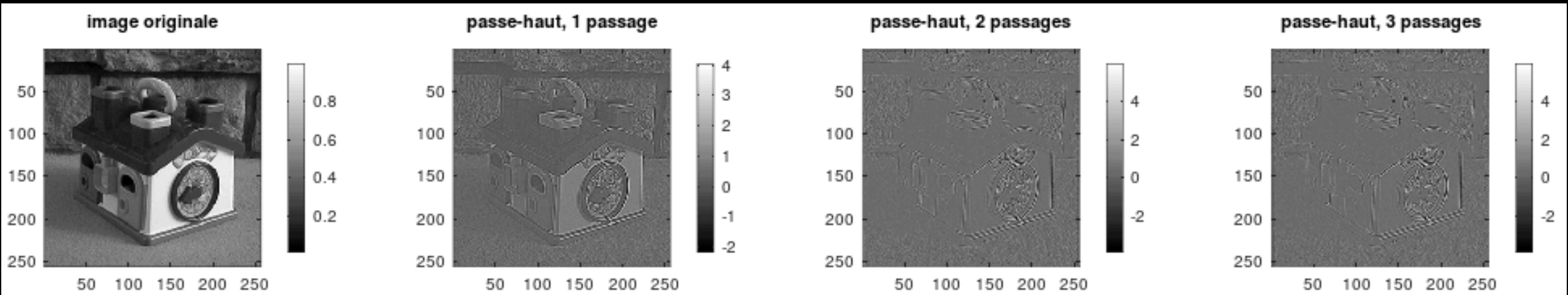
III. Filtrage





—> 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 :**

```
        1  .  .  1  0  .  .  0
        .  .  1  1  0  0  .  .
        .  1  1  1  0  0  0  .
Isp =   .  .  1  1  0  0  .  .
        .  .  .  .  .  .  .  .
        .  .  .  .  .  .  .  .
        .  .  .  .  .  .  .  .
        1  .  .  1  0  .  .  0
```

>> 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
  >> Isp_low = filter2(h, Isp)

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

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

  —> 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) :

$$\text{Isp\_gau} = \begin{matrix} 1 & 1 & 3/4 & 1/4 & 0 & 0 \\ 1 & 1 & 3/4 & 1/4 & 0 & 0 \\ 1 & 1 & 3/4 & 1/4 & 0 & 0 \\ 1 & 1 & 3/4 & 1/4 & 0 & 0 \\ 1 & 1 & 3/4 & 1/4 & 0 & 0 \\ 1 & 1 & 3/4 & 1/4 & 0 & 0 \end{matrix}$$

  —> 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) :

```
            1   1   4  -3   0   0
            1   1   4  -3   0   0
lsp_hi  =   1   1   4  -3   0   0
            1   1   4  -3   0   0
            1   1   4  -3   0   0
            1   1   4  -3   0   0
```

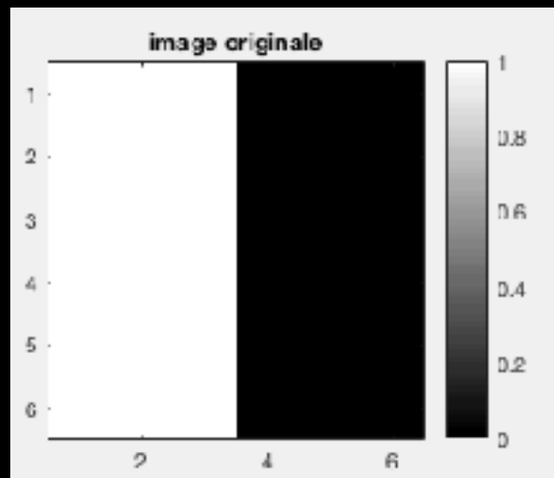
  —> accentuation de la transition brusque.

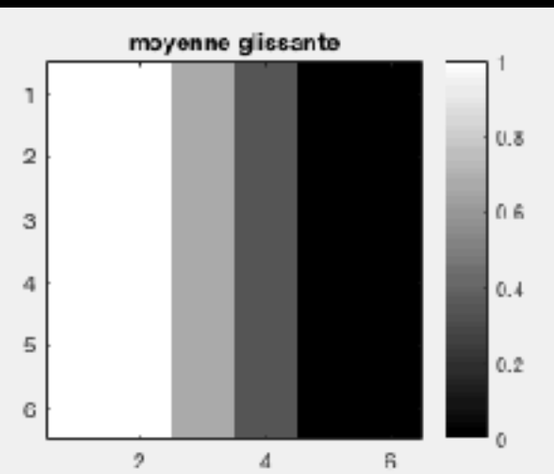
III. Filtrage

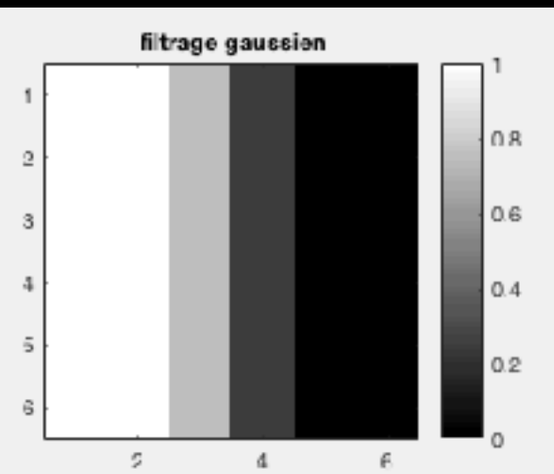
- **EXERCICE 5 : Codage ‘propre’ de ce que nous venons de voir…**

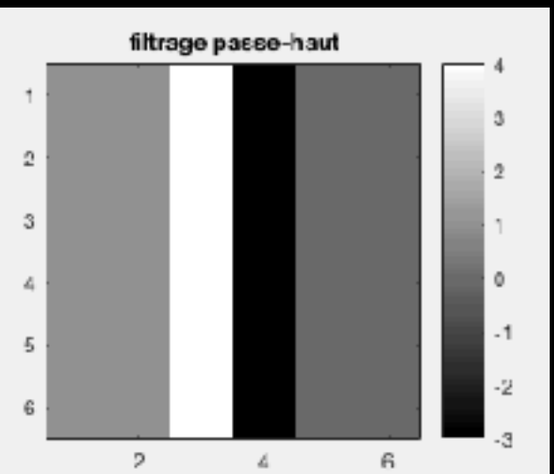
```
clear
close all

%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')
```

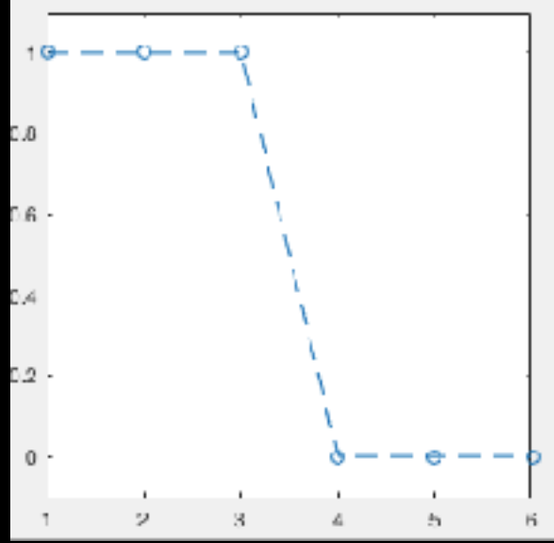
III. Filtrage

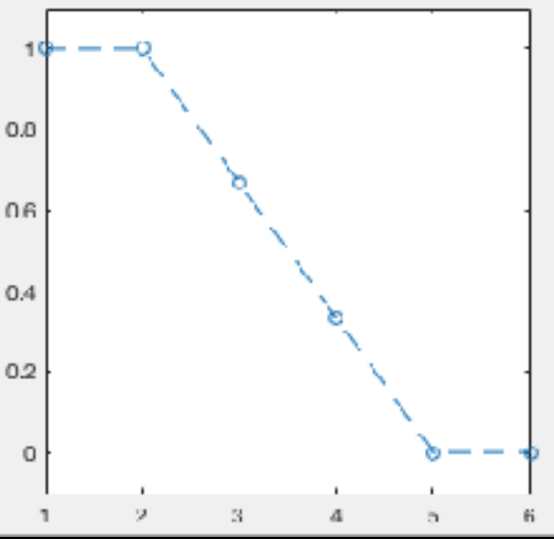


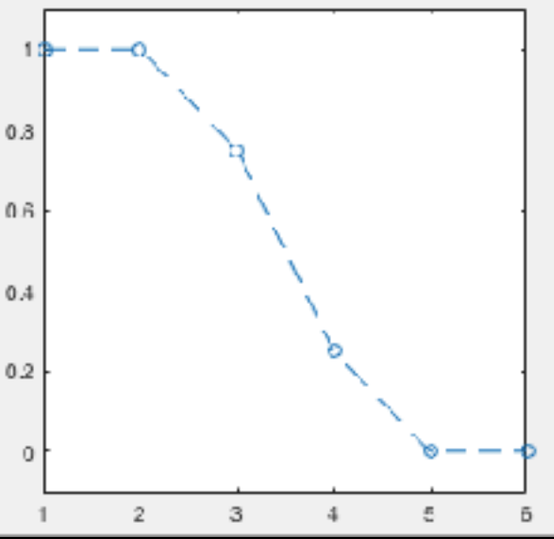


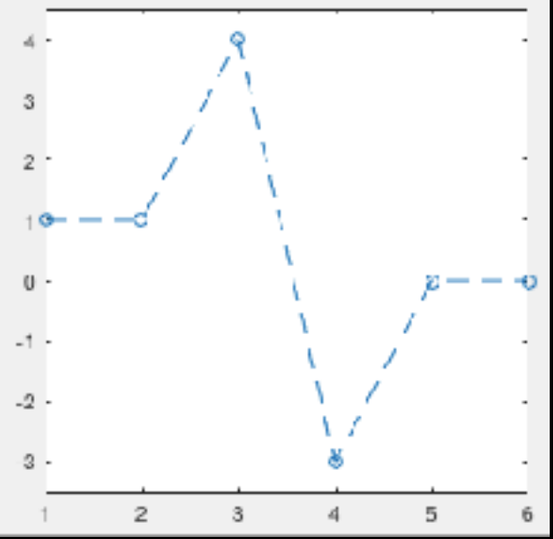












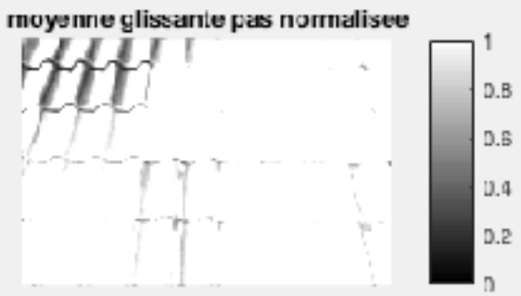
III. Filtrage

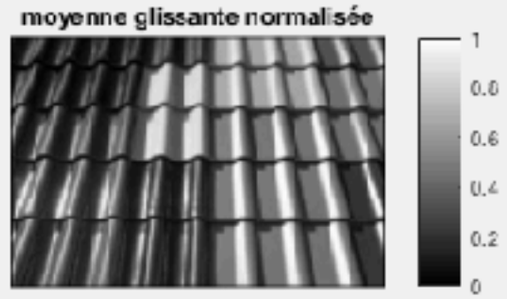
- **EXERCICE 6 : Reprendre le filtre moyenne glissante 3x3 sur une image de votre choix (i.e. celle du toit) et NE PAS respecter la normalisation. Puis diviser par 81 au lieu de 9. Que se passe-t-il ? (Utiliser pour comparaison *imshow* ET *imagesc*, bien penser à *colorbar*, utiliser les options « *same/full/valid*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

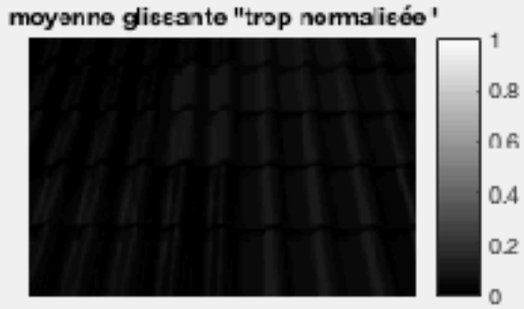
III. Filtrage

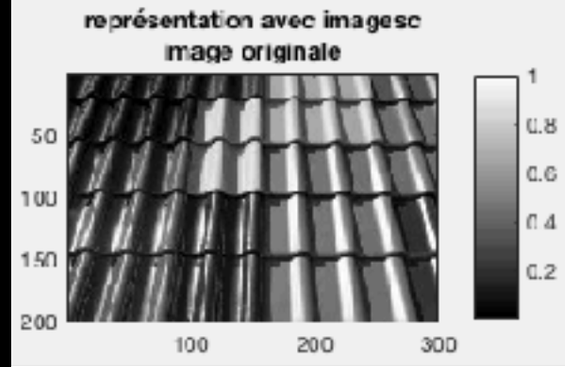
```
clear
close all

dir='/Users/marcel/Documents/MATLAB/GBM/0-images/';
img=[dir,'toit.jpeg'];
I=imread(img);
I=rgb2gray(I);
I=double(I)/255.0;
whos I

figure

subplot(2,4,1), imshow(I)
title({'représentation avec imshow' ; 'image originale'}), colorbar
colormap('gray')
subplot(2,4,5), imagesc(I), axis('image'), colorbar
title({'représentation avec imagesc' ; 'image originale'})

h=ones(3,3);
I1=filter2(h,I, 'same');
whos I1
subplot(2,4,2), imshow(I1)
title('moyenne glissante pas normalisee'), colorbar
subplot(2,4,6), imagesc(I1), axis('image'), colorbar
title('moyenne glissante pas normalis@e')

h=h/9.;
I2=filter2(h,I, 'full');
whos I2
subplot(2,4,3), imshow(I2)
title('moyenne glissante normalis@e'), colorbar
subplot(2,4,7), imagesc(I2), axis('image'), colorbar
title('moyenne glissante normalis@e')

h=h/9.;
I3=filter2(h,I, 'valid');
subplot(2,4,4), imshow(I3)
whos I3
title('moyenne glissante "trop normalis@e"'), colorbar
subplot(2,4,8), imagesc(I3), axis('image'), colorbar
title('moyenne glissante "trop normalis@e"')
```

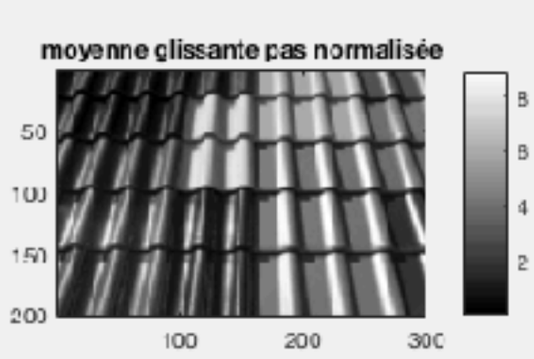
III. Filtrage

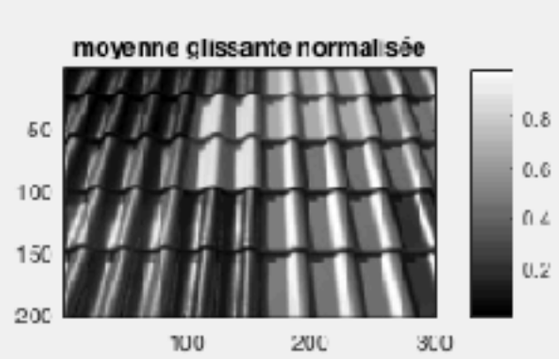


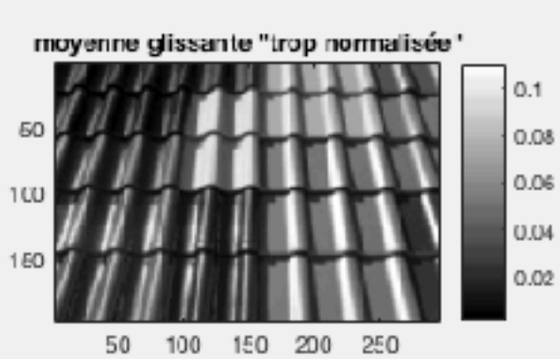












—> comme on l'avait déjà remarqué : différence visible avec *imshow*, mais pas avec *imagesc* qui ajuste automatiquement la dynamique => utiliser *colorbar* pour se rendre compte de ce que l'on a réellement sous les yeux !

III. Filtrage

—> remarque sur l'option « *same/full/valid* » :

`Y = filter2(H,X,shape)` returns a subsection of the filtered data according to `shape`. For example, `Y = filter2(H,X,'valid')` returns only filtered data computed without zero-padded edges.

shape — Subsection of filtered data
`'same'` (default) | `'full'` | `'valid'`

Subsection of the filtered data, specified as one of these values:

- `'same'` — Return the central part of the filtered data, which is the same size as `X`.
- `'full'` — Return the full 2-D filtered data.
- `'valid'` — Return only parts of the filtered data that are computed without zero-padded edges.

```
>> exo6_normalisation

img =

    '/Users/marcel/Documents/MATLAB/0-images-exemples/classiques/house.jpg'

  Name        Size              Bytes  Class     Attributes

  I1        256x256            524288  double

  Name        Size              Bytes  Class     Attributes

  I2        258x258            532512  double

  Name        Size              Bytes  Class     Attributes

  I3        254x254            516128  double
```

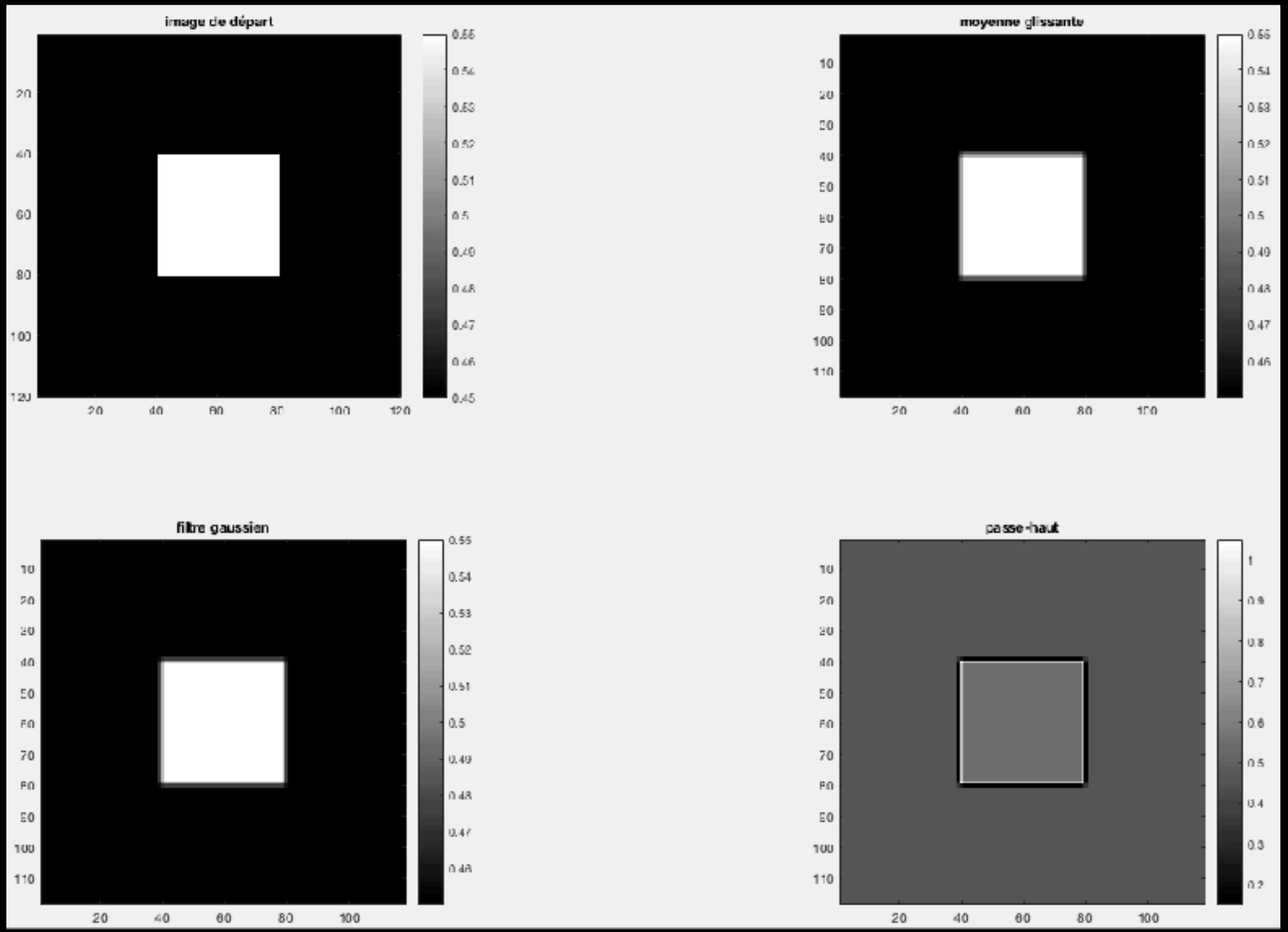
III. Filtrage

- **EXERCICE 7 : Créer une image 120x120 grise (à 45%) contenant un carré 40x40 d'un gris plus clair (à 55%), puis la filtrer par moyenne glissante 3x3, filtre Gaussien 3x3 et filtre passe-haut 3x3. Ne pas considérer les bords de l'image.**

III. Filtrage

```
clear
close all

hm=ones(3,3)/9.
hg=[1 2 1;2 4 2;1 2 1]/16.
hh=-ones(3,3); hh(2,2)=9.; hh

dim = 120;
dam = 40;
I=.45*ones(dim,dim);
I(dim/2-dam/2+1:dim/2+dam/2,dim/2-dam/2+1:dim/2+dam/2)=.55;
% ou : I=.45*ones(120,120); I(41:80,41:80)=.55;

Im=filter2(hm,I, 'valid');
Ig=filter2(hg,I, 'valid');
Ih=filter2(hh,I, 'valid');

figure
colormap('gray')
subplot(2,2,1), imagesc(I), colorbar
title('image de départ'), axis('square')
subplot(2,2,2), imagesc(Im), colorbar
title('moyenne glissante'), axis('square')
subplot(2,2,3), imagesc(Ig), colorbar
title('filtre gaussien'), axis('square')
subplot(2,2,4), imagesc(Ih), colorbar
title('passe-haut'), axis('square')
```

III. Filtrage



III. Filtrage

- **Remarque : *FILTRE PASSE HAUT* = *a* x *FILTRE UNITÉ* - *b* x *FILTRE PASSE-BAS***

  Par exemple, le filtre passe-haut que nous venons d'étudier se construit très simplement à partir du filtre passe-bas « moyenne glissante » avec *a*=10 et *b*=9 :

```
      0 0 0       1 1 1          -1 -1 -1
10 x  0 1 0  -  9  1 1 1 x 1/9 =  -1  9  -1
      0 0 0       1 1 1          -1 -1 -1

      FILTRE      FILTRE         FILTRE
      UNITÉ       MOY. GLISS.    PASSE-HAUT CORRESPONDANT
```

III. Filtrage

3- FILTRAGE NON-LINÉAIRE

- **Filtrage non-linéaire = filtrage qu'on ne peut pas définir avec l'équation qui définit les filtrages linéaires (section 2).**

- **Par exemple : le <u>FILTRE MÉDIAN</u>**

- **Le FILTRE MÉDIAN affecte à un pixel la valeur médiane des intensités de son entourage et de lui-même.**

- **Prenons comme exemple le voisinage suivant du pixel de valeur 0.2 :**

  0.7 0.6 0.3
  0.4 0.2 0.8
  0.8 0.5 0.2

  Le filtre médian range tout d'abord par ordre croissant les intensités du voisinage : 0.2 0.2 0.3 0.4 *<u>0.5</u>* 0.6 0.7 0.8 0.8

III. Filtrage

- **La médiane de cet ensemble de valeurs est *0.5*, c'est la valeur qui va donc être affectée au pixel central (le 0.2 central devient donc 0.5).**

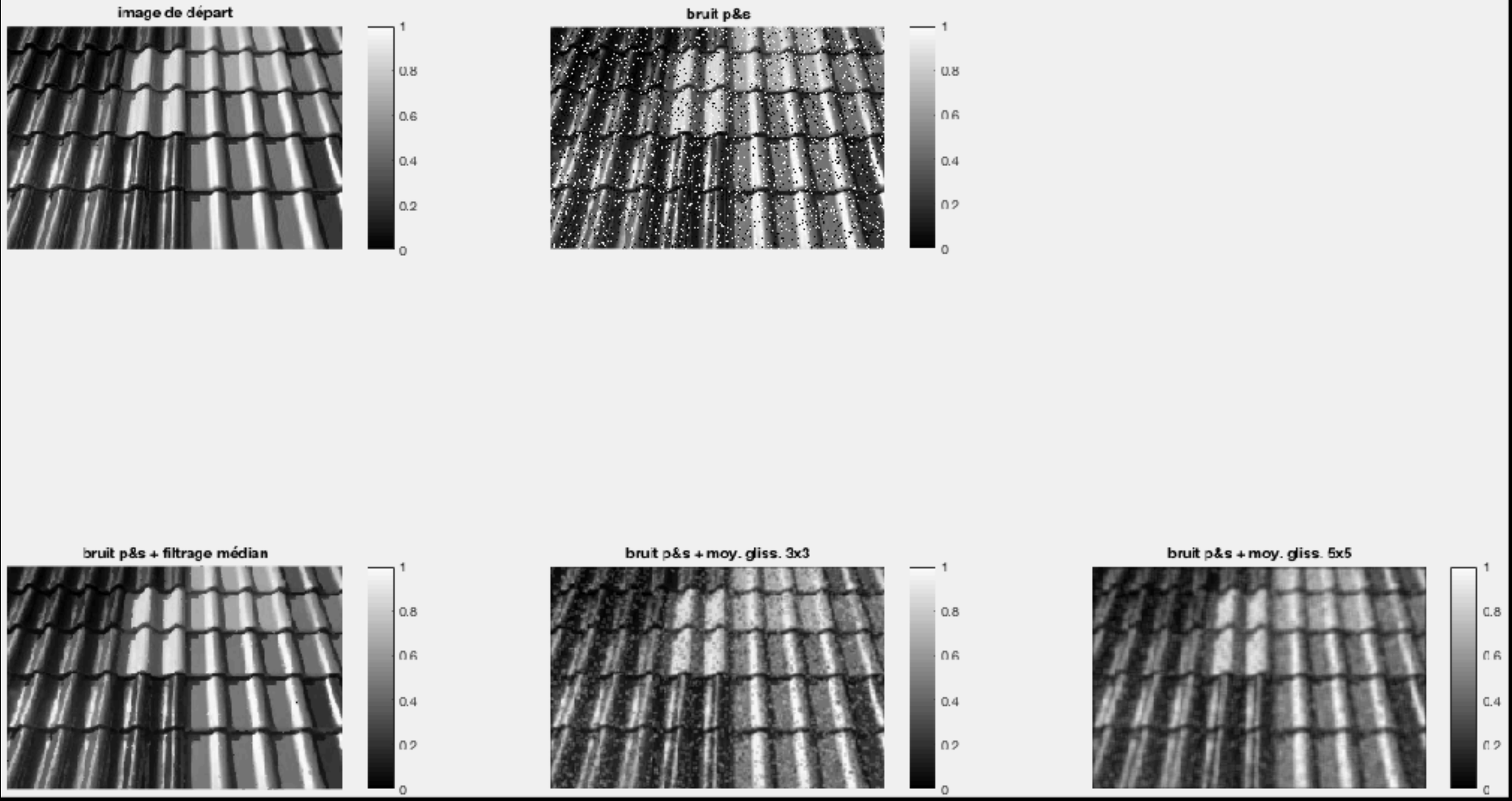
- **sous MATLAB/OCTAVE : *medfilt2(image)***

- **EXERCICE 8 : Bruiter l'image de l'exercice 6 (p.ex.) avec un bruit poivre & sel à 10%, puis tenter de la débruiter par le filtre médian. Comparer avec une tentative de débruitage résultant du filtrage linéaire utilisant les filtres suivants : moyenne glissante 3x3, moyenne glissante 5x5.**

III. Filtrage

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

% image de la petite maison
dir='/Users/marcel/Documents/MATLAB/GBM/0-images/';
img=[dir,'toit.jpeg'];
I=imread(img);
I=rgb2gray(I);
I=double(I)/255.0;

%---
% bruit poivre et sel à hauteur de 10% de densité
Ib=imnoise(I, 'salt & pepper', 0.1);

% débruitage par le filtre médian
Imed=medfilt2(Ib);

% débruitage par la moyenne glissante 3x3
Im3=filter2(ones(3,3)/9.0,Ib);

% débruitage par la moyenne glissante 5x5
Im5=filter2(ones(5,5)/25.0,Ib);

% figure
figure
subplot(2,3,1), imshow(I), title('image de départ'), colorbar
subplot(2,3,2), imshow(Ib), title('bruit p&s'), colorbar
subplot(2,3,4), imshow(Imed), title('bruit p&s + filtrage médian'), colorbar
subplot(2,3,5), imshow(Im3), title('bruit p&s + moy. gliss. 3x3'), colorbar
subplot(2,3,6), imshow(Im5), title('bruit p&s + moy. gliss. 5x5'), colorbar
```

III. Filtrage



—> filtre médian très supérieur ici (bruit éliminé, et sans perte de résolution spatiale) !

III. Filtrage

- **Pourquoi le filtre médian préserve-t-il si bien les contours (par rapport à un filtre passe-bas linéaire) ?…**

  Prenons l'image suivante :

  1 1 0
  1 1 0
  1 1 0

Et le filtre passe-bas suivant :

1 1 1
1 1 1 x 1/9
1 1 1

—> valeur du pixel central après filtrage passe-bas : 6/9 = 2/3 ≈ 0.667

—> alors que, avec le filtre médian : 0 0 0 1 *1* 1 1 1 1

=> Le filtre médian n'a pas ici modifié le contour (contrairement au filtre passe-bas qui l'a amoindri).

III. Filtrage

- **Autre exemple : image uniforme dont le pixel central a été affecté par le bruit poivre & sel (‘sel’ ici!)**

  0.1 0.1 0.1
  0.1 1 0.1
  0.1 0.1 0.1

  —> valeur du pixel central après filtrage par la moyenne glissante =
  1/9 x (0.8+1) = 0.2

  —> alors que, avec le filtre médian : 0.1 0.1 0.1 0.1 *0.1* 0.1 0.1 0.1 1

  => le bruit est TOTALEMENT éliminé ici avec le filtre médian (alors qu’il n’est qu’atténué avec le filtre passe-bas).

III. Filtrage

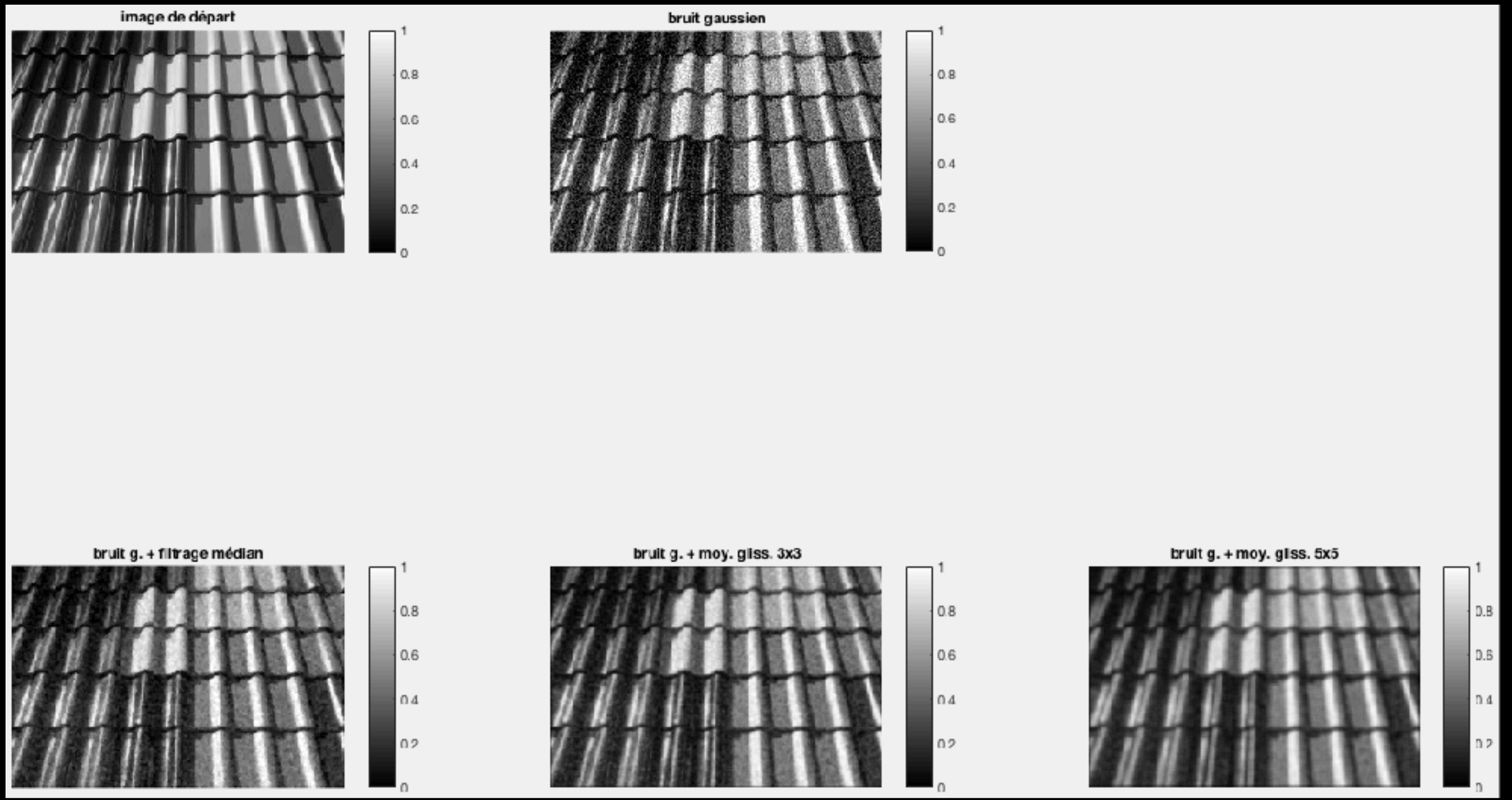
- **EXERCICE 9 : Bruiter cette fois-ci la même image avec un bruit GAUSSIEN additif (avec un écart-type relatif de 0.1), puis faire la même comparaison.**

(Attention à ne pas confondre BRUIT GAUSSIEN et FILTRE GAUSSIEN…)

III. Filtrage

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

% image de la petite maison
dir='/Users/marcel/Documents/MATLAB/GBM/0-images/';
img=[dir,'toit.jpeg'];
I=imread(img);
I=rgb2gray(I);
I=double(I)/255.0;

%---
% bruit gaussien avec une variance relative de 1%
Ib=imnoise(I, 'gaussian', 0., 0.01);

% débruitage par le filtre médian
Imed=medfilt2(Ib);

% débruitage par la moyenne glissante 3x3
Im3=filter2(ones(3,3)/9.,Ib);

% débruitage par la moyenne glissante 5x5
Im5=filter2(ones(5,5)/25.,Ib);

% figure
figure
subplot(2,3,1), imshow(I), title('image de départ'), colorbar
subplot(2,3,2), imshow(Ib), title('bruit gaussien'), colorbar
subplot(2,3,4), imshow(Imed), title('bruit g. + filtrage médian'), colorbar
subplot(2,3,5), imshow(Im3), title('bruit g. + moy. gliss. 3x3'), colorbar
subplot(2,3,6), imshow(Im5), title('bruit g. + moy. gliss. 5x5'), colorbar
```

III. Filtrage



—> filtre médian beaucoup moins impressionnant ici…