

III. Filtrage

- Autre filtre passe-bas : le FILTRE GAUSSIEN

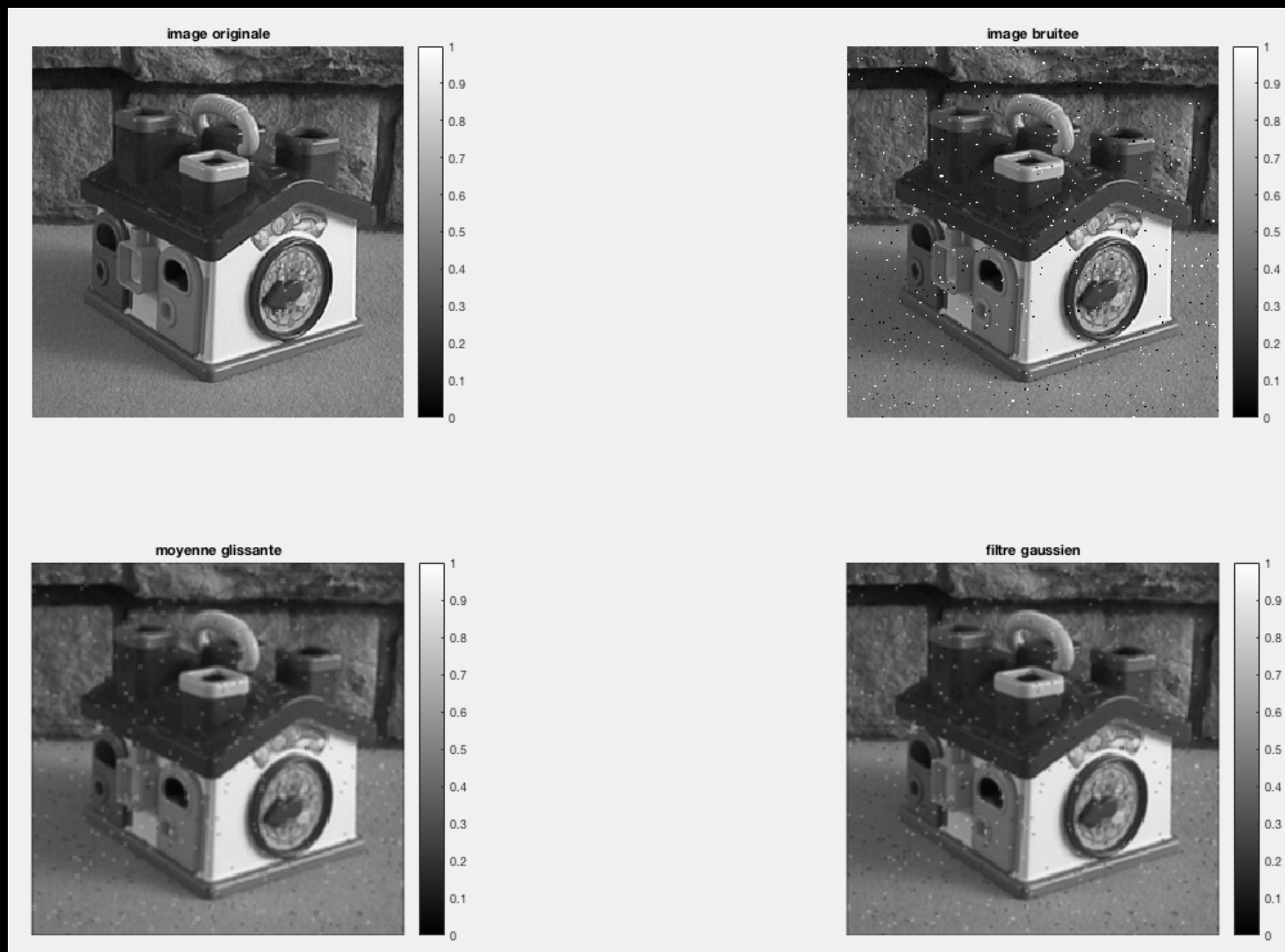
$$h = \begin{matrix} & 1 & 2 & 1 \\ 2 & 4 & 2 \\ & 1 & 2 & 1 \end{matrix} \times 1/16$$

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

III. Filtrage

```
1 - clear
2 - close all
3
4 - img='/Users/marcel/Documents/MATLAB/0-images-exemples/classiques/house.jpg'
5 - I=imread(img);
6 - I=rgb2gray(I);
7 - I=double(I)/255.0;
8 - Ib=imnoise(I, 'salt & pepper', 0.01);
9
10 - h=ones(3,3)/9;
11 - If=filter2(h,Ib);
12
13 - g=[1 2 1;2 4 2;1 2 1]/16.
14 - Ig=filter2(g,Ib);
15
16 - figure
17 - subplot(2,2,1), imshow(I), title('image originale'), colorbar
18 - subplot(2,2,2), imshow(Ib), title('image bruitée'), colorbar
19 - subplot(2,2,3), imshow(If), title('moyenne glissante'), colorbar
20 - 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 FILTRE UNITÉ

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

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

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

(et donc : passe-tout !)

III. Filtrage

- Plus intéressants : les FILTRES PASSE-HAUT, i.e. :

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

- 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 imposer la dynamique des images entre 0 et 1 entre deux passages du filtre afin que la fonction **filter2** puisse fonctionner correctement.)*

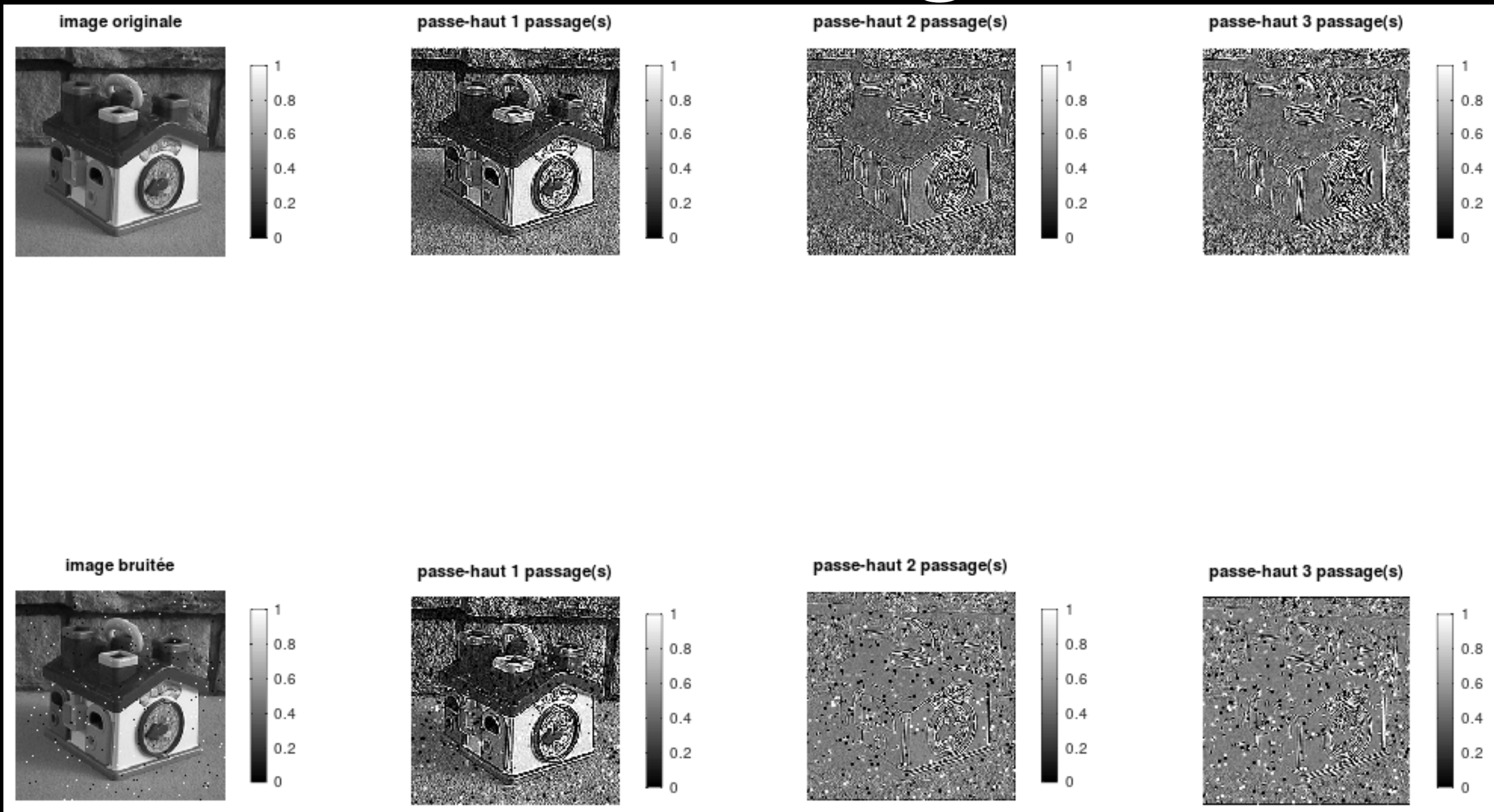
III. Filtrage

```
1 clear
2 close all
3 %pkg load image
4 %(pour Octave, pas Matlab)
5
6 img='/Users/marcel/Documents/MATLAB/GBM/0-images/frog.jpg'
7 I=imread(img); I=rgb2gray(I); I=double(I)/255;
8 Ib=imnoise(I, 'salt & pepper', 0.01);
9 %hi = [-1 -1 -1;-1 9 -1;-1 -1 -1]
10 hi=ones(3,3); hi(2,2)=9
11
12 % version sans boucle for et avec imagesc :
13 figure(1), colormap('gray')
14 subplot(2,4,1), imagesc(I), axis('image'), title('image originale'), colorbar
15
16 % image sans bruit
17 I_hi = filter2(hi, I);
18 subplot(2,4,2), imagesc(I_hi), axis('image'), title('passe-haut, 1 passage'), colorbar
19
20 I_hi = I_hi-min(min(I_hi)); I_hi = I_hi/max(max(I_hi));
21 I_hi2 = filter2(hi, I_hi);
22 subplot(2,4,3), imagesc(I_hi2), axis('image'), title('passe-haut, 2 passages'), colorbar
23
24 I_hi2= I_hi2-min(min(I_hi2)); I_hi2=I_hi2/max(max(I_hi2));
25 I_hi3 = filter2(hi, I_hi2);
26 subplot(2,4,4), imagesc(I_hi3), axis('image'), title('passe-haut, 3 passages'), colorbar
27
28 % image avec bruit
29 subplot(2,4,5), imagesc(Ib), axis('image'), title('image bruitée'), colorbar
30
31 Ib_hi = filter2(hi, Ib);
32 subplot(2,4,6), imagesc(Ib_hi), axis('image'), title('passe-haut, 1 passage'), colorbar
33
34 Ib_hi = Ib_hi-min(min(Ib_hi)); Ib_hi = Ib_hi/max(max(Ib_hi));
35 Ib_hi2 = filter2(hi, Ib_hi);
36 subplot(2,4,7), imagesc(Ib_hi2), axis('image'), title('passe-haut, 2 passages'), colorbar
37
38 Ib_hi2 = Ib_hi2-min(min(Ib_hi2)); Ib_hi2 = Ib_hi2/max(max(Ib_hi2));
39 Ib_hi3 = filter2(hi, Ib_hi2);
40 subplot(2,4,8), imagesc(Ib_hi3), axis('image'), title('passe-haut, 3 passages'), colorbar
```

III. Filtrage

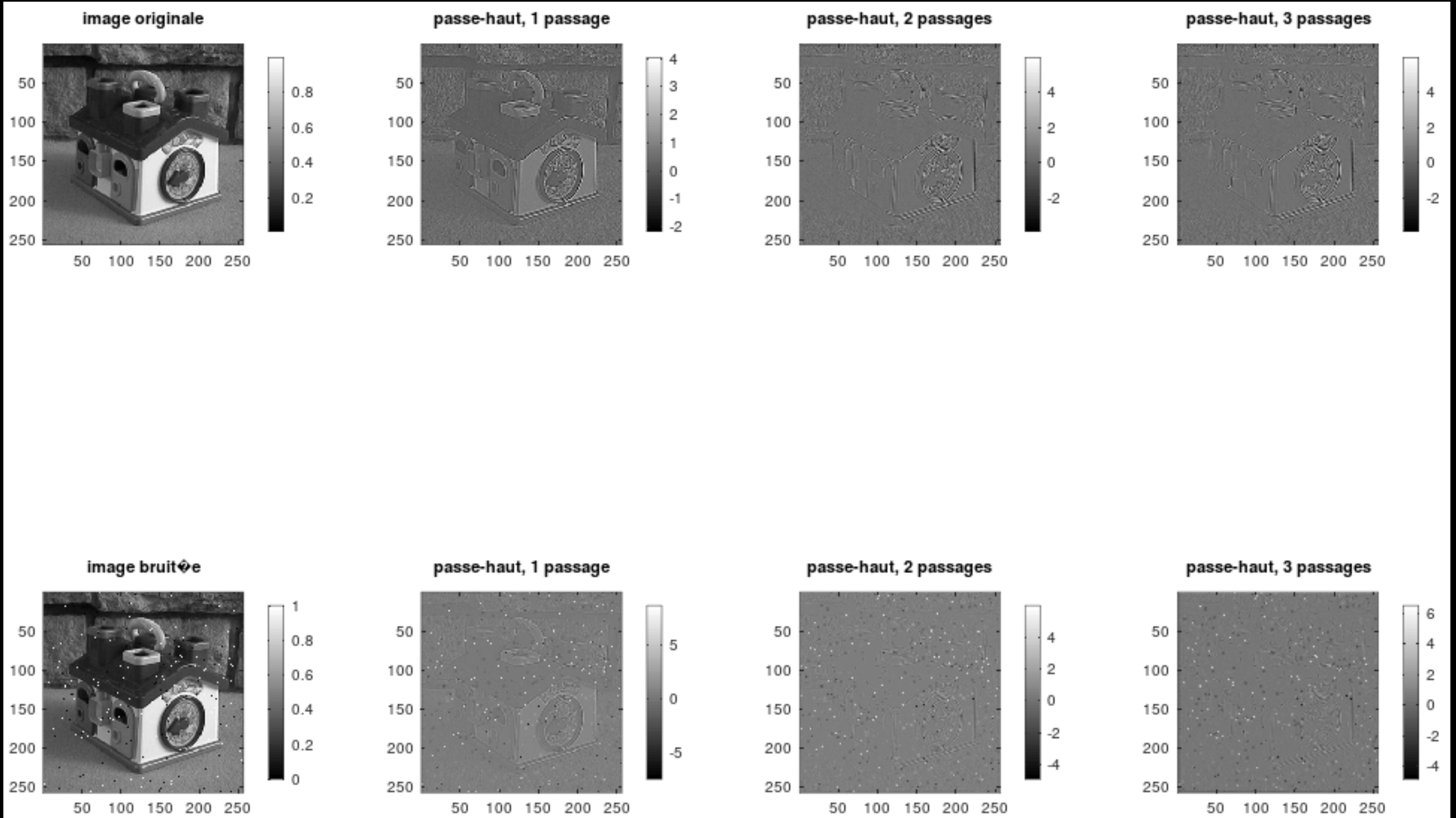
```
42 % version avec boucle for et avec imshow :
43 figure(2)
44
45 % image sans bruit
46 Ih = I;
47 subplot(2,4,1), imshow(I), title('image originale'), colorbar
48 for n=1:3
49     Ih = Ih - min(min(Ih)); Ih = Ih/max(max(Ih));
50     Ih = filter2(hi, Ih);
51     subplot(2,4,n+1), imshow(Ih), colorbar
52     title(['passe-haut ', num2str(n), ' passage(s)'])
53 end
54
55 % image avec bruit
56 Ib = I;
57 subplot(2,4,5), imshow(Ib), title('image bruitée'), colorbar
58 for n=1:3
59     Ih = Ih - min(min(Ih)); Ih = Ih/max(max(Ih));
60     Ih = filter2(hi, Ih);
61     subplot(2,4,n+5), imshow(Ih), colorbar
62     title(['passe-haut ', num2str(n), ' passage(s)'])
63 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 :

$I_{sp} =$

1	.	.	1	0	.	.	0
.	.	1	1	0	0	.	.
.	1	1	1	0	0	0	.
.	.	1	1	0	0	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
1	.	.	1	0	.	.	0

>> $I_{sp} = \text{zeros}(8,8)$; $I_{sp}(:, 1:4)=1$ (ou : >> $I_{sp} = \text{ones}(8,8)$; $I_{sp}(:, 5:8)=0$)

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 = 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
```

—> 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  
>> lsp_gau = filter2(g, lsp)
```

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

```
lsp_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  
          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) :

```
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
1	1	4	-3	0	0
1	1	4	-3	0	0

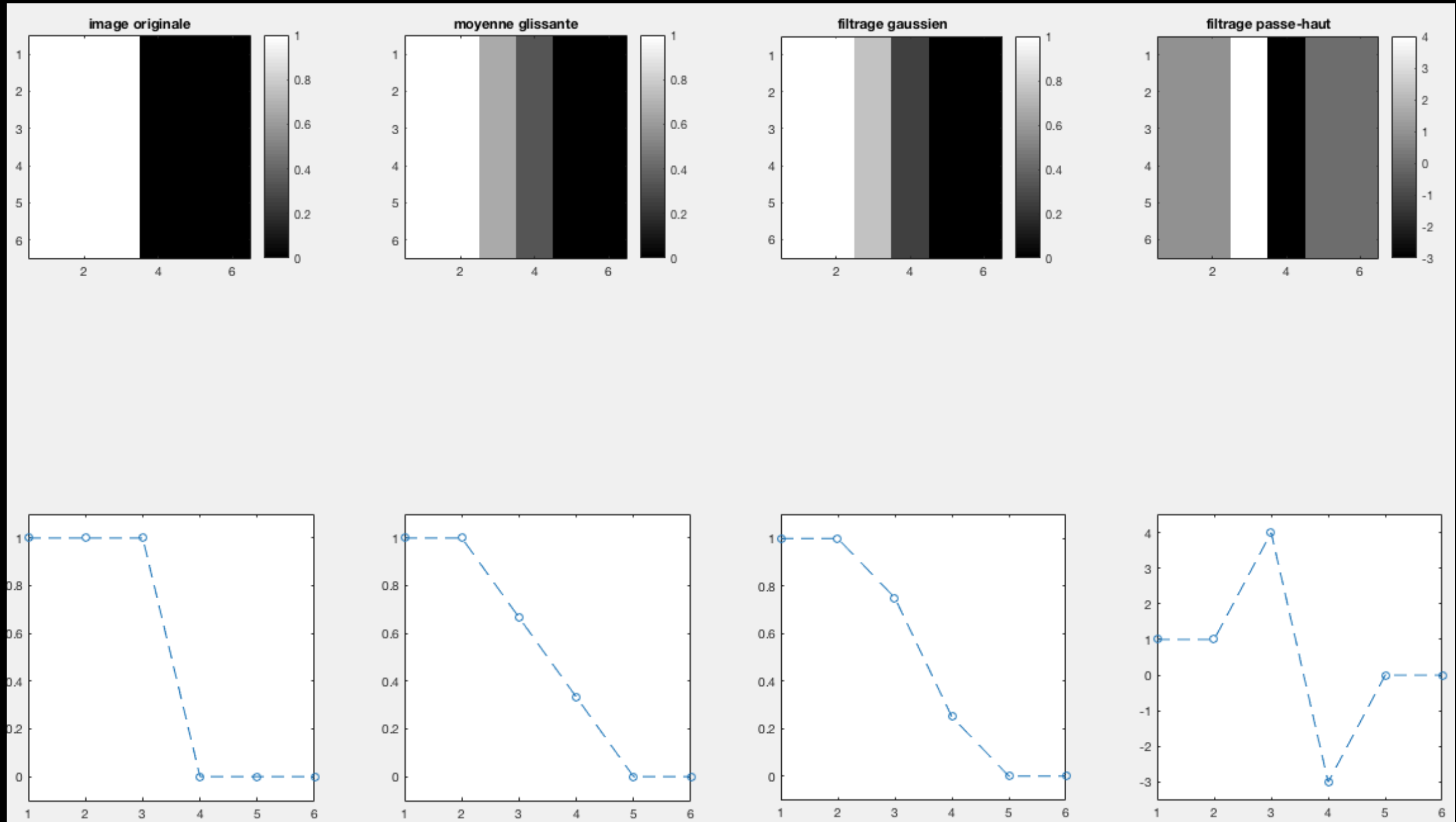
—> accentuation de la transition brusque.

III. Filtrage

- **EXERCICE 5 : Codage 'propre' de ce que nous venons de voir...**

```
1 - clear
2 - close all
3
4 - %Isp = zeros(8,8); Isp(:,1:end/2)=1
5 - Isp = ones(8,8); Isp(:,end/2+1:end)=0
6
7 - Isp_low = filter2(ones(3,3)/9, Isp)
8 - Isp_gau = filter2([1 2 1;2 4 2;1 2 1]/16., Isp)
9 - Isp_hi = filter2([-1 -1 -1;-1 9 -1;-1 -1 -1], Isp)
10
11 - figure, colormap('gray')
12
13 - subplot(2,4,1), imagesc(Isp(2:end-1,2:end-1)), colorbar
14 - title('image originale'), axis('square')
15
16 - subplot(2,4,2), imagesc(Isp_low(2:end-1,2:end-1)), colorbar
17 - title('moyenne glissante'), axis('square')
18
19 - subplot(2,4,3), imagesc(Isp_gau(2:end-1,2:end-1)), colorbar
20 - title('filtrage gaussien'), axis('square')
21
22 - subplot(2,4,4), imagesc(Isp_hi(2:end-1,2:end-1)), colorbar
23 - title('filtrage passe-haut'), axis('square')
24
25 - subplot(2,4,5), plot(Isp(4,2:end-1), '--o')
26 - axis([1 6 -0.1 1.1]), axis('square')
27
28 - subplot(2,4,6), plot(Isp_low(4,2:end-1), '--o')
29 - axis([1 6 -0.1 1.1]), axis('square')
30
31 - subplot(2,4,7), plot(Isp_gau(4,2:end-1), '--o')
32 - axis([1 6 -0.1 1.1]), axis('square')
33
34 - subplot(2,4,8), plot(Isp_hi(4,2:end-1), '--o')
35 - axis([1 6 -3.5 4.5]), axis('square')
```

III. Filtrage



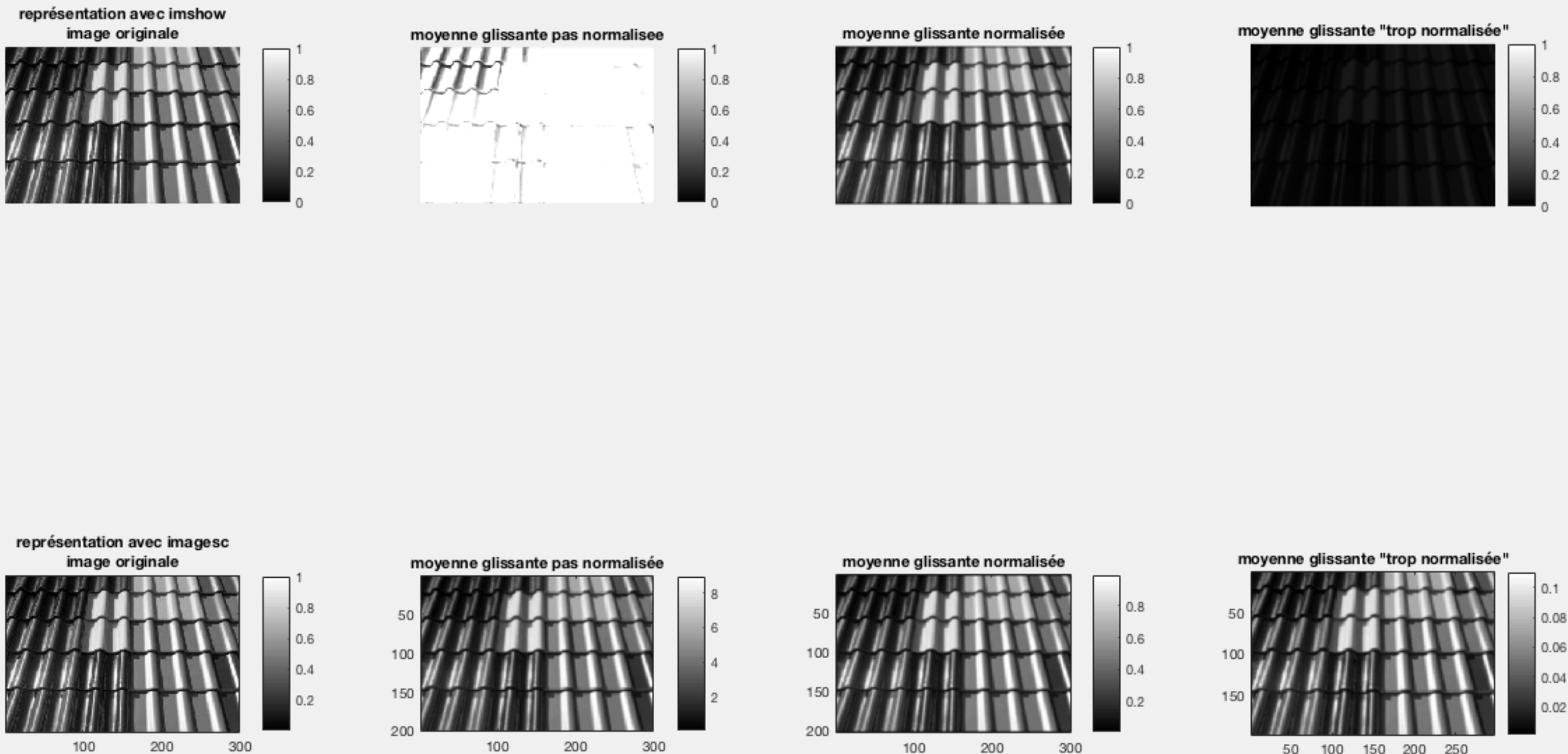
III. Filtrage

- **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

```
1 clear
2 close all
3
4 dir='/Users/marcel/Documents/MATLAB/GBM/0-images/';
5 img=[dir,'toit.jpeg'];
6 I=imread(img);
7 I=rgb2gray(I);
8 I=double(I)/255.0;
9 whos I
10
11 figure
12
13 subplot(2,4,1), imshow(I)
14 title({'représentation avec imshow' ; 'image originale'}), colorbar
15 colormap('gray')
16 subplot(2,4,5), imagesc(I), axis('image'), colorbar
17 title({'représentation avec imagesc' ; 'image originale'})
18
19 h=ones(3,3);
20 I1=filter2(h,I, 'same');
21 whos I1
22 subplot(2,4,2), imshow(I1)
23 title('moyenne glissante pas normalisée'), colorbar
24 subplot(2,4,6), imagesc(I1), axis('image'), colorbar
25 title('moyenne glissante pas normalisée')
26
27 h=h/9.;
28 I2=filter2(h,I, 'full');
29 whos I2
30 subplot(2,4,3), imshow(I2)
31 title('moyenne glissante normalisée'), colorbar
32 subplot(2,4,7), imagesc(I2), axis('image'), colorbar
33 title('moyenne glissante normalisée')
34
35 h=h/9.;
36 I3=filter2(h,I, 'valid');
37 subplot(2,4,4), imshow(I3)
38 whos I3
39 title('moyenne glissante "trop normalisée"'), colorbar
40 subplot(2,4,8), imagesc(I3), axis('image'), colorbar
41 title('moyenne glissante "trop normalisée")
```

III. Filtrage



—> comme on l'avait déjà remarqué : différence visible avec **imshow**, mais pas avec **imagesc** qui ajuste automatiquement la dynamique => penser à 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.

```
>> ex06_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	

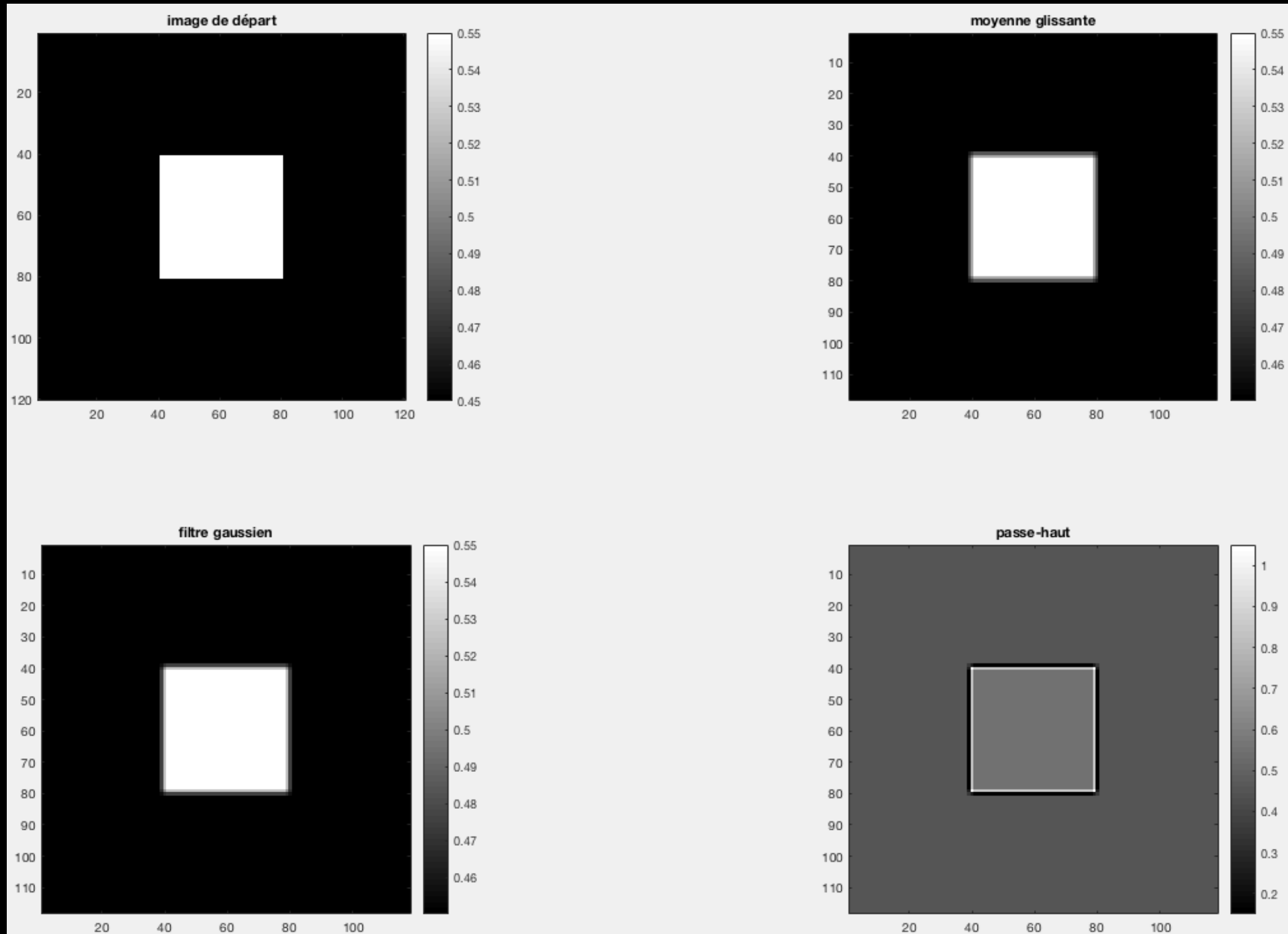
III. Filtrage

- **EXERCICE 7 (en mode TP) : Créer une image, au format 'double', de dimension 120x120, grise à 45% d'intensité, et contenant un carré 40x40 d'un gris plus clair (à 55%). La filtrer par moyenne glissante 3x3, filtre Gaussien 3x3 et filtre passe-haut 3x3. Ne pas considérer les bords des images filtrées.**

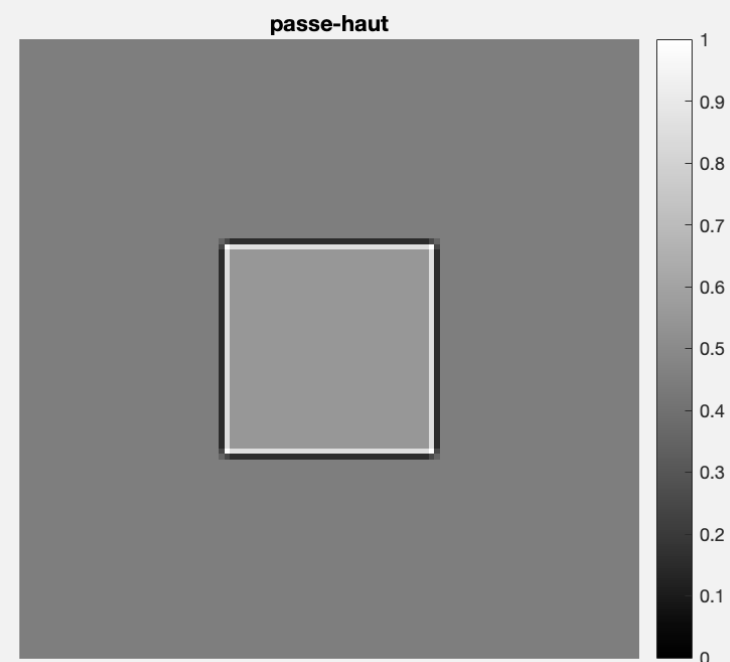
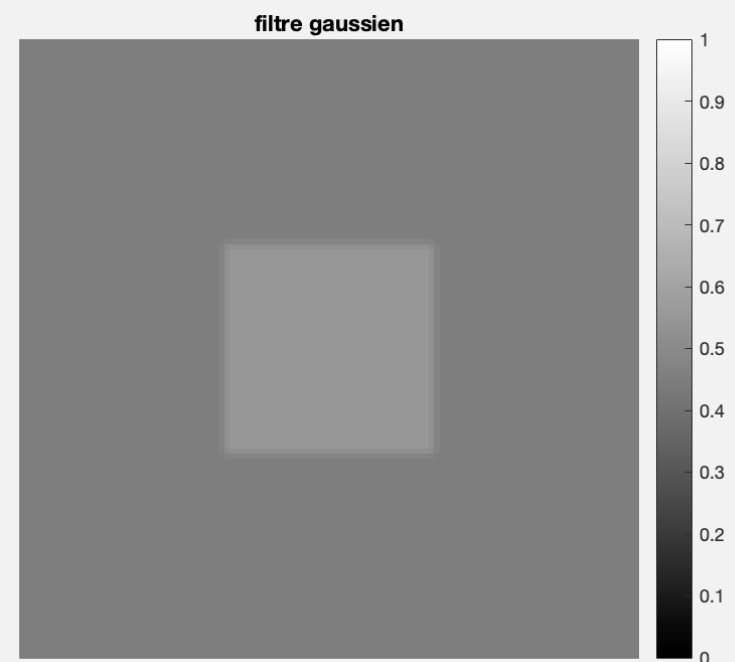
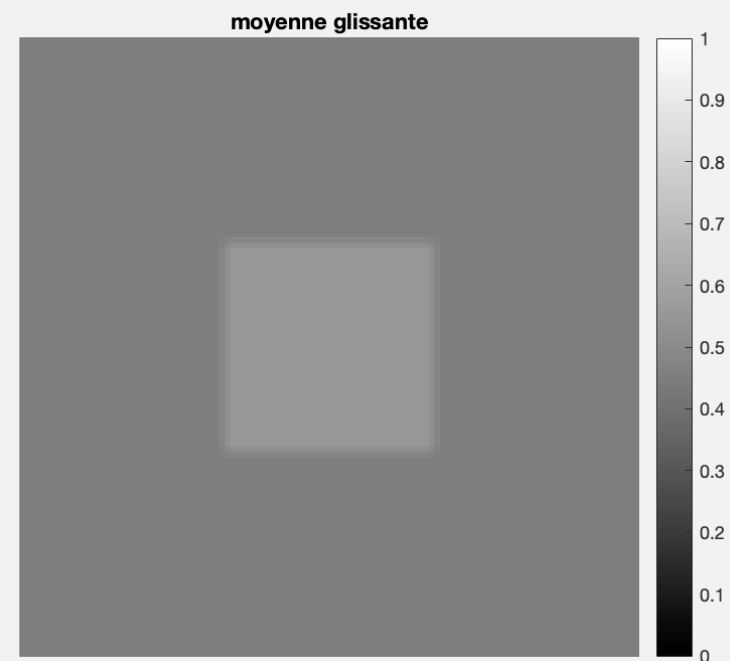
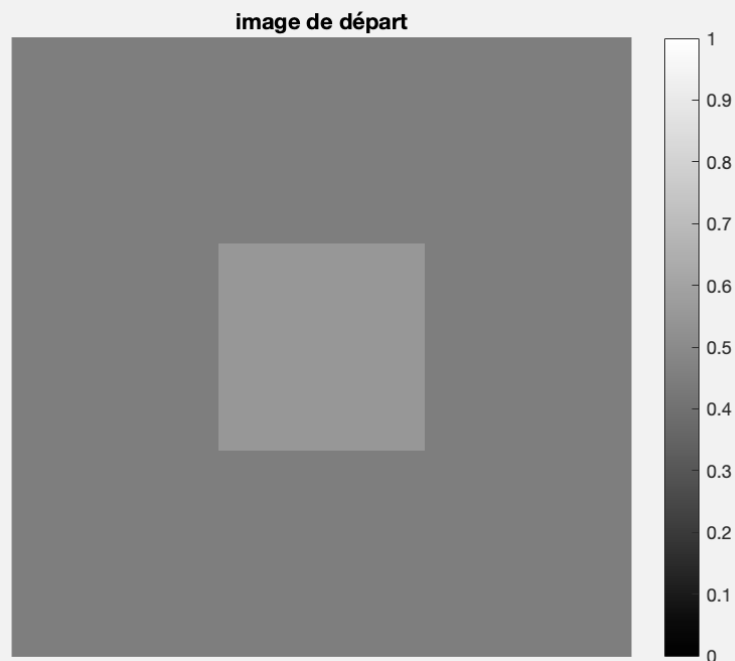
III. Filtrage

```
1 - clear
2 - close all
3
4 - hm=ones(3,3)/9.
5 - hg=[1 2 1;2 4 2;1 2 1]/16.
6 - hh=-ones(3,3); hh(2,2)=9.; hh
7
8 - dim = 120;
9 - dam = 40;
10 - I=.45*ones(dim,dim);
11 - I(dim/2-dam/2+1:dim/2+dam/2,dim/2-dam/2+1:dim/2+dam/2)=.55;
12 % ou : I=.45*ones(120,120); I(41:80,41:80)=.55;
13
14 - Im=filter2(hm,I, 'valid');
15 - Ig=filter2(hg,I, 'valid');
16 - Ih=filter2(hh,I, 'valid');
17
18 - figure
19 - colormap('gray')
20 - subplot(2,2,1), imagesc(I), colorbar
21 - title('image de départ'), axis('square')
22 - subplot(2,2,2), imagesc(Im), colorbar
23 - title('moyenne glissante'), axis('square')
24 - subplot(2,2,3), imagesc(Ig), colorbar
25 - title('filtre gaussien'), axis('square')
26 - subplot(2,2,4), imagesc(Ih), colorbar
27 - title('passe-haut'), axis('square')
```

III. Filtrage



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 $b=9$:

$$\begin{array}{rcccl}
 & 0 & 0 & 0 & & 1 & 1 & 1 & & -1 & -1 & -1 \\
 10 & \times & 0 & 1 & 0 & - & 9 & 1 & 1 & 1 & \times & 1/9 & = & -1 & 9 & -1 \\
 & & 0 & 0 & 0 & & & 1 & 1 & 1 & & & & -1 & -1 & -1
 \end{array}$$

**FILTRE
UNITÉ**

**FILTRE
MOY. GLISS.**

**FILTRE
PASSE-HAUT CORRESPONDANT**

- On peut faire la même chose avec le filtre gaussien... (avec $a=17$ et $b=16$)

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 FILTRE MÉDIAN
- 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 0.5 0.6 0.7 0.8 0.8

III. Filtrage

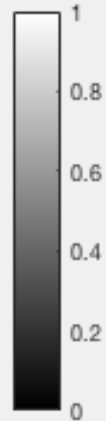
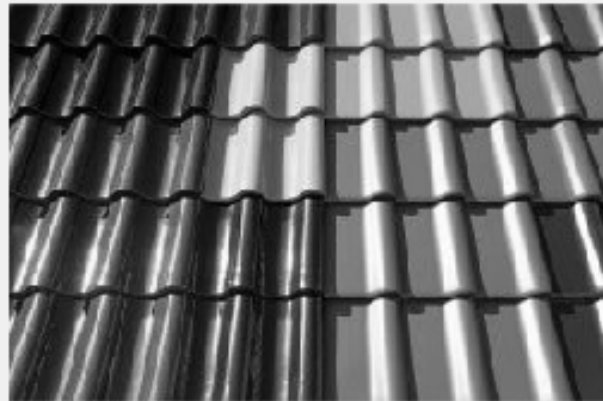
- La médiane de cet ensemble de valeurs est **0.5**, c'est la valeur qui va donc être affectée au pixel central (le 0.2 central devient donc 0.5).
- sous MATLAB/OCTAVE : ***medfilt2(image)***
- **EXERCICE 8** : Bruiter l'image de l'exercice 6 (p.ex.) avec un bruit poivre & sel à 10%, puis tenter de la débruiter par le filtre médian. Comparer avec une tentative de débruitage résultant du filtrage linéaire utilisant un filtrage passe-bas de votre choix — par ex. moyenne glissante 3x3, moyenne glissante 5x5, etc.

III. Filtrage

```
1 clear
2 close all
3 %pkg load image %Octave (pas Matlab)
4
5 % image de la petite maison
6 dir='/Users/marcel/Documents/MATLAB/GBM/0-images/';
7 img=[dir,'toit.jpeg'];
8 I=imread(img);
9 I=rgb2gray(I);
10 I=double(I)/255.0;
11
12 %---
13 % bruit poivre et sel à hauteur de 10% de densité
14 Ib=imnoise(I, 'salt & pepper', 0.1);
15
16 % débruitage par le filtre médian
17 Imed=medfilt2(Ib);
18
19 % débruitage par la moyenne glissante 3x3
20 Im3=filter2(ones(3,3)/9.0,Ib);
21
22 % débruitage par la moyenne glissante 5x5
23 Im5=filter2(ones(5,5)/25.0,Ib);
24
25 % figure
26 figure
27 subplot(2,3,1), imshow(I), title('image de départ'), colorbar
28 subplot(2,3,2), imshow(Ib), title('bruit p&s'), colorbar
29 subplot(2,3,4), imshow(Imed), title('bruit p&s + filtrage médian'), colorbar
30 subplot(2,3,5), imshow(Im3), title('bruit p&s + moy. gliss. 3x3'), colorbar
31 subplot(2,3,6), imshow(Im5), title('bruit p&s + moy. gliss. 5x5'), colorbar
```

III. Filtrage

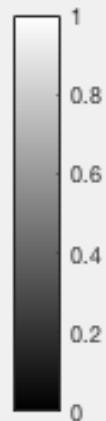
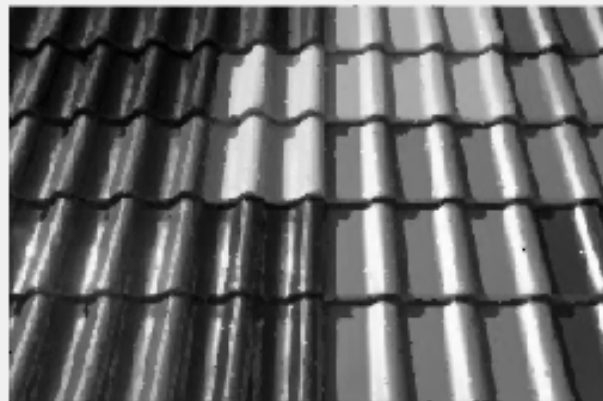
image de départ



bruit p&s



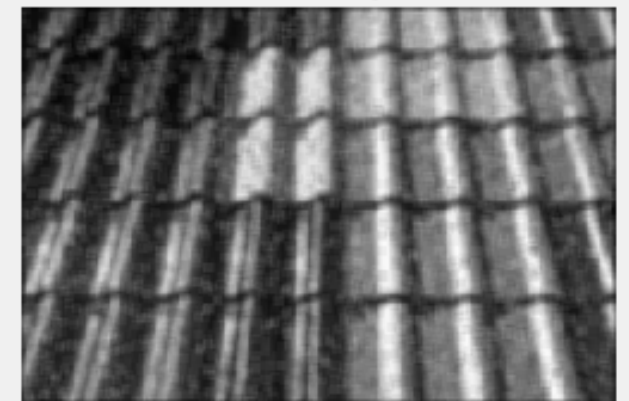
bruit p&s + filtrage médian



bruit p&s + moy. gliss. 3x3



bruit p&s + moy. gliss. 5x5



—> 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 : $6/9 = 2/3 \approx 0.667$

—> 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 = $\frac{1}{9} \times (0.8 + 1) = 0.2$

—> 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).