

# III. Filtrage

- Autre filtre passe-bas : le FILTRE GAUSSIEN

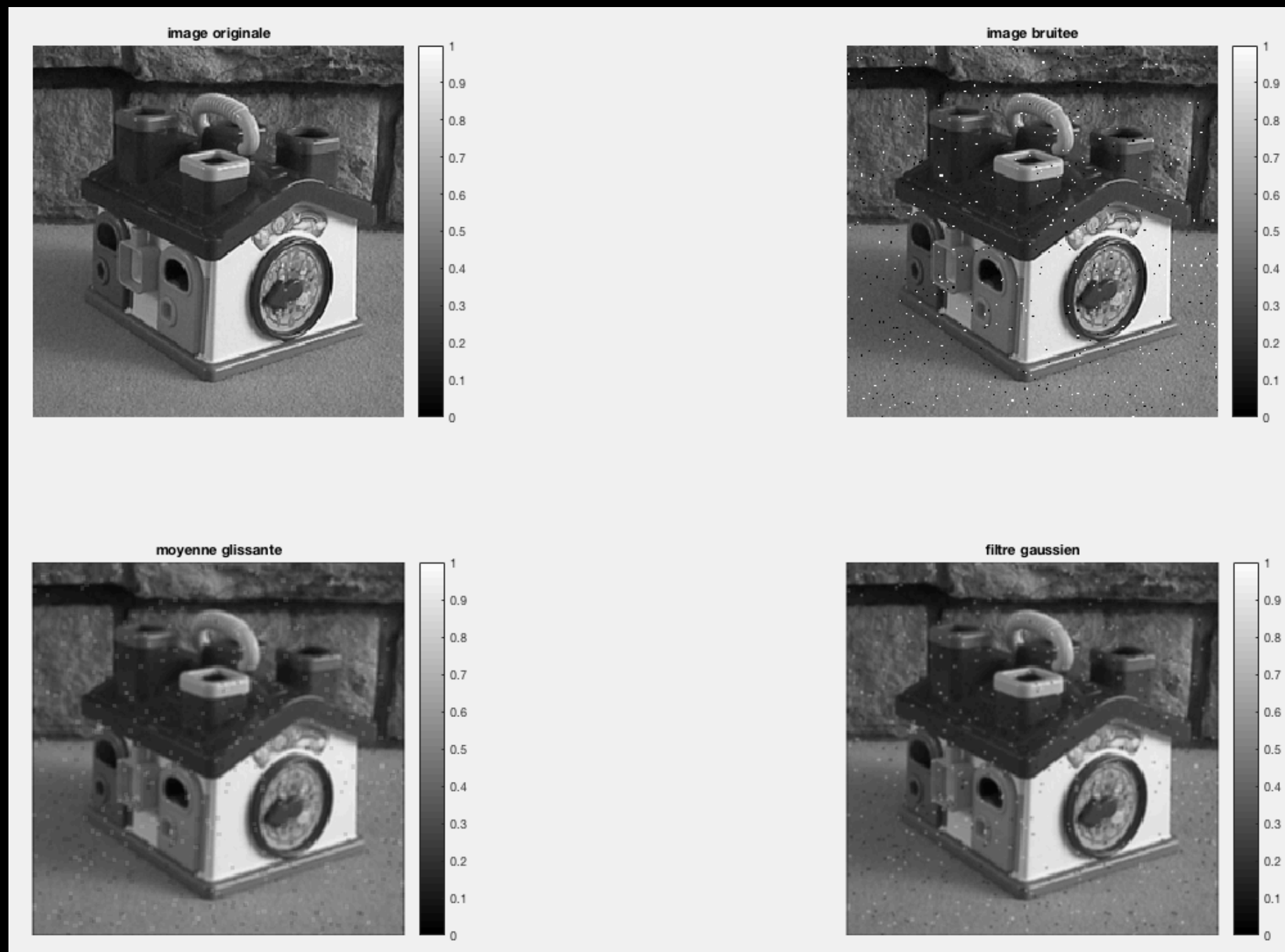
$$h = \begin{matrix} & 1 & 2 & 1 \\ 2 & 4 & 2 \\ & 1 & 2 & 1 \end{matrix} \times 1/16$$

- 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, p.ex. :

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

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

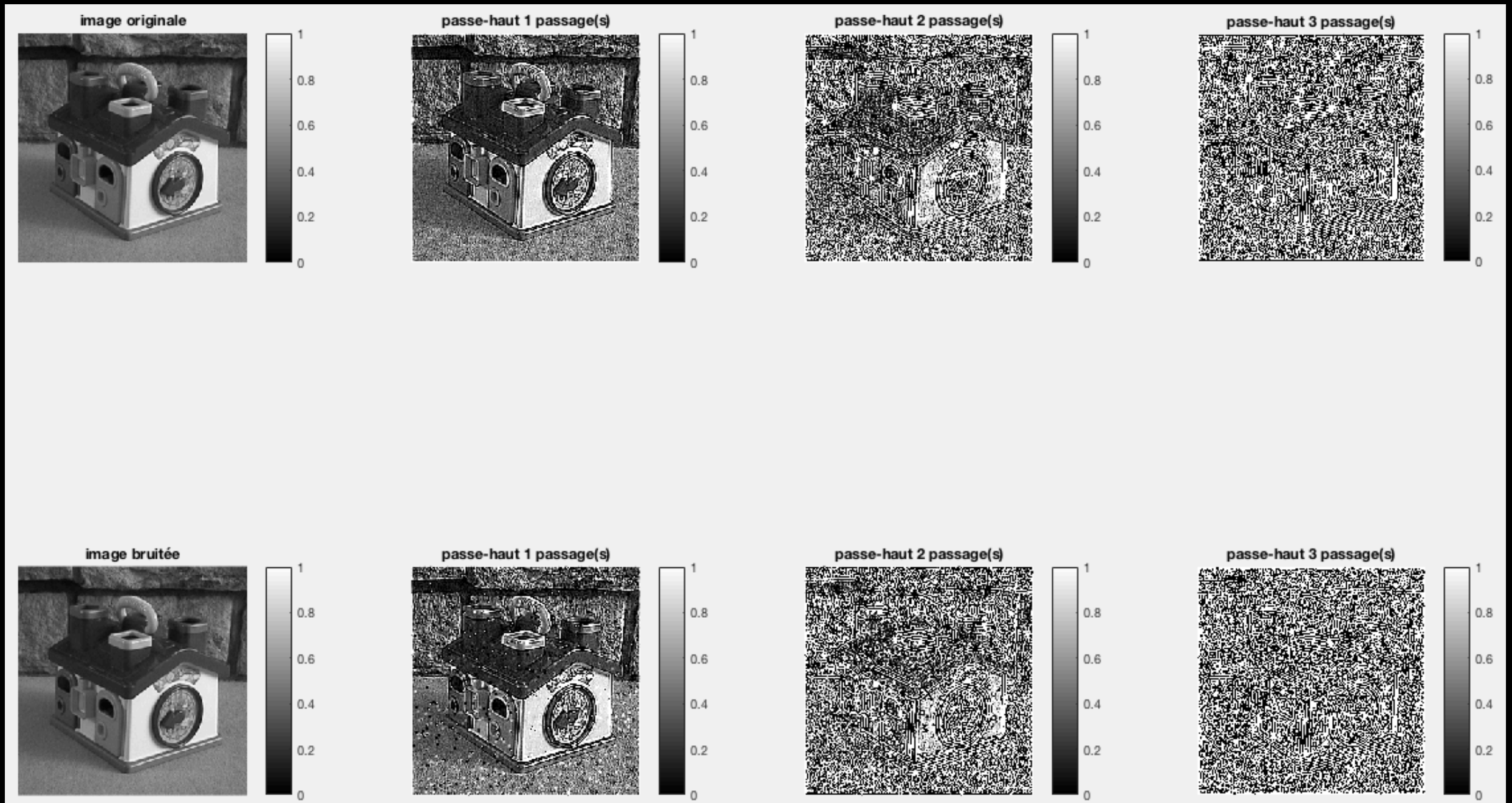
# III. Filtrage

```
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 %hi = [-1 -1 -1;-1 9 -1;-1 -1 -1]
11 - hi=ones(3,3); hi(2,2)=9
12
13 % version sans boucle for :
14 - I_hi = filter2(hi, I);
15 - I_hi2 = filter2(hi, I_hi);
16 - I_hi3 = filter2(hi, I_hi2);
17
18 - Ib_hi = filter2(hi, Ib);
19 - Ib_hi2 = filter2(hi, Ib_hi);
20 - Ib_hi3 = filter2(hi, Ib_hi2);
21
22 - figure(1), colormap('gray')
23
24 - subplot(2,4,1), imshow(I), title('image originale'), colorbar
25 - subplot(2,4,2), imshow(I_hi), title('passe-haut, 1 passage'), colorbar
26 - subplot(2,4,3), imshow(I_hi2), title('passe-haut, 2 passages'), colorbar
27 - subplot(2,4,4), imshow(I_hi3), title('passe-haut, 3 passages'), colorbar
28
29 - subplot(2,4,5), imshow(Ib), title('image bruitée'), colorbar
30 - subplot(2,4,6), imshow(Ib_hi), title('passe-haut, 1 passage'), colorbar
31 - subplot(2,4,7), imshow(Ib_hi2), title('passe-haut, 2 passages'), colorbar
32 - subplot(2,4,8), imshow(Ib_hi3), title('passe-haut, 3 passages'), colorbar
```

# III. Filtrage

```
34 % version avec boucle for :
35 - figure(2)
36
37 - Ih = I;
38 - subplot(2,4,1), imshow(I), title('image originale'), colorbar
39 - □ for n=1:3
40 -     Ih = filter2(hi, Ih);
41 -     subplot(2,4,n+1), imshow(Ih), colorbar
42 -     title(['passe-haut ',int2str(n), ' passage(s)'])
43 - end
44
45 - Ih = Ib;
46 - subplot(2,4,5), imshow(I), title('image bruitée'), colorbar
47 - □ for n=1:3
48 -     Ih = filter2(hi, Ih);
49 -     subplot(2,4,n+5), imshow(Ih), colorbar
50 -     title(['passe-haut ',int2str(n), ' passage(s)'])
51 - end
```

# III. Filtrage



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



# 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 :

**lsp =**

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

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

# III. Filtrage

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

```
>> 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) :

```
>> 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) :

```
>> 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

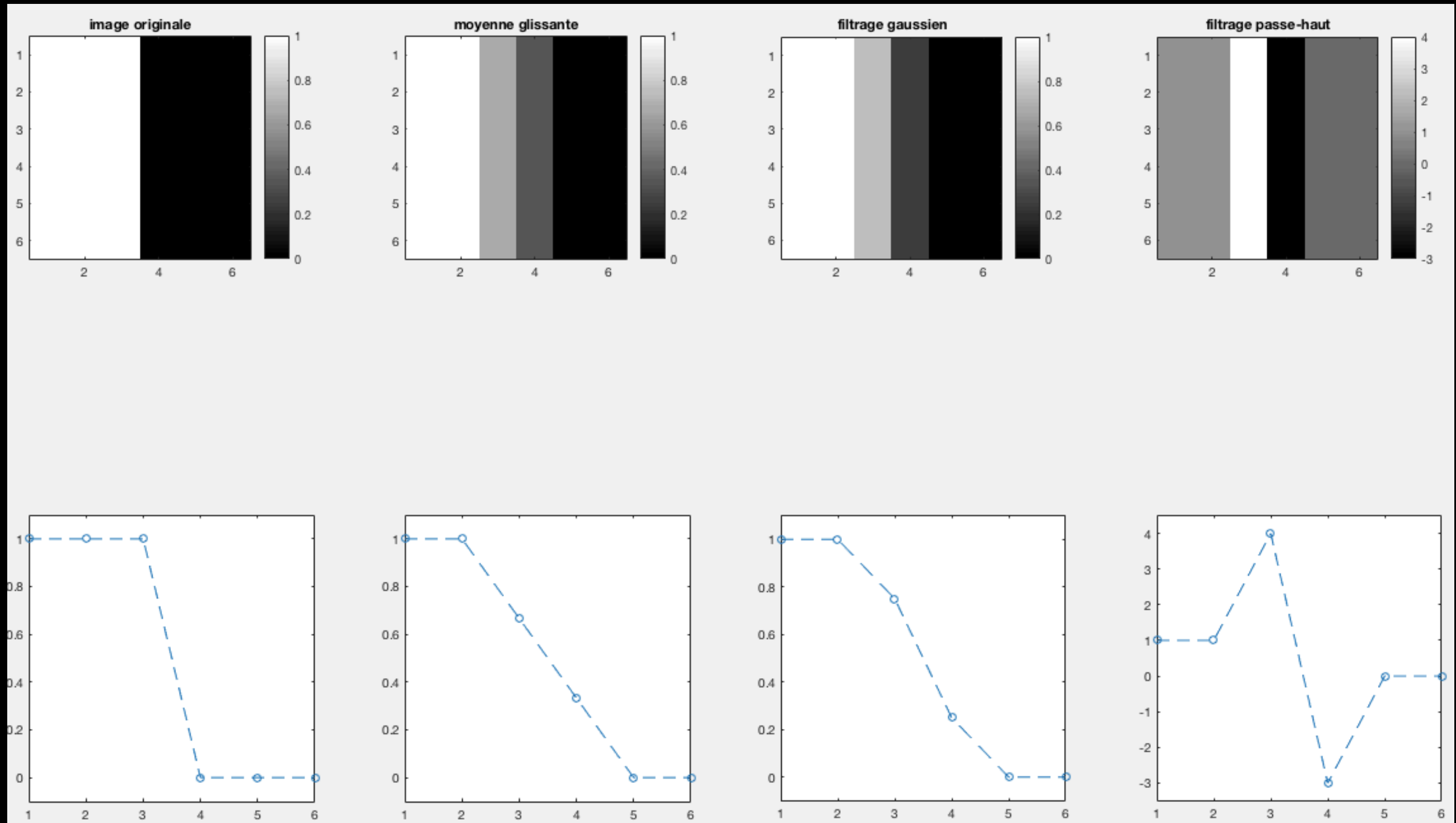
—> accentuation de la transition brusque.

# 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 (p.ex. 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 `imagecsc`, bien penser à `colorbar`, utiliser les options « `same/full/valid` » pour `filter2...`)

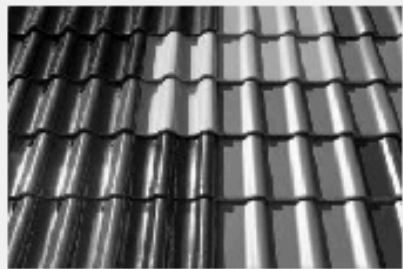
# III. Filtrage

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

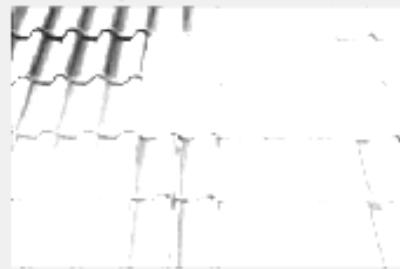


# III. Filtrage

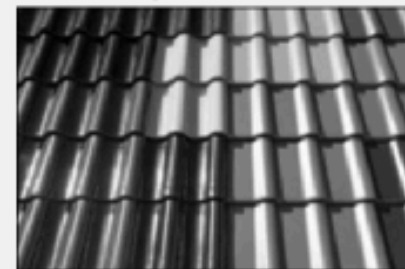
représentation avec imshow  
image originale



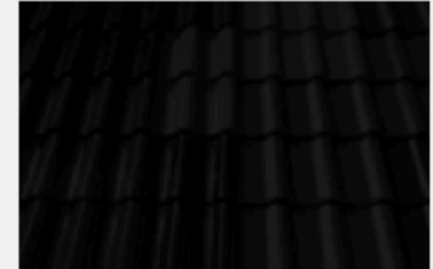
moyenne glissante pas normalisée



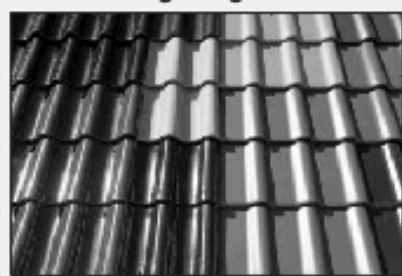
moyenne glissante normalisée



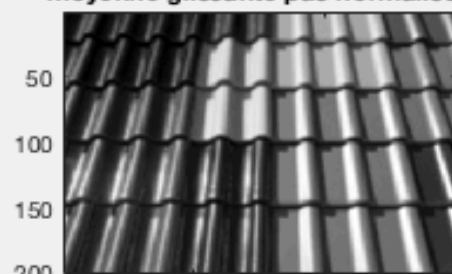
moyenne glissante "trop normalisée"



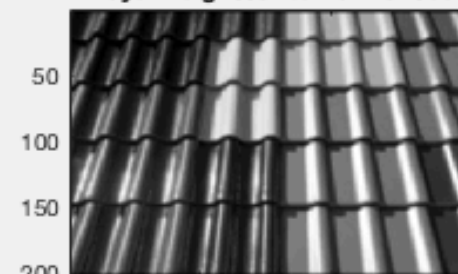
représentation avec imagesc  
image originale



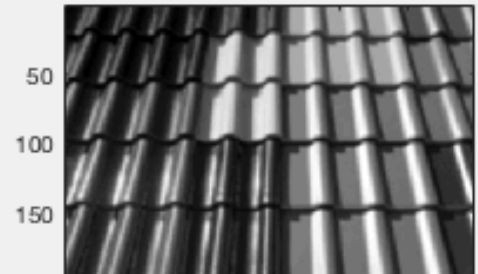
moyenne glissante pas normalisée



moyenne glissante normalisée



moyenne glissante "trop normalisée"



—> comme on l'avait déjà remarqué : différence visible avec imshow, pas avec imagesc qui ajuste automatiquement la dynamique => utiliser colorbar !

# 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
```

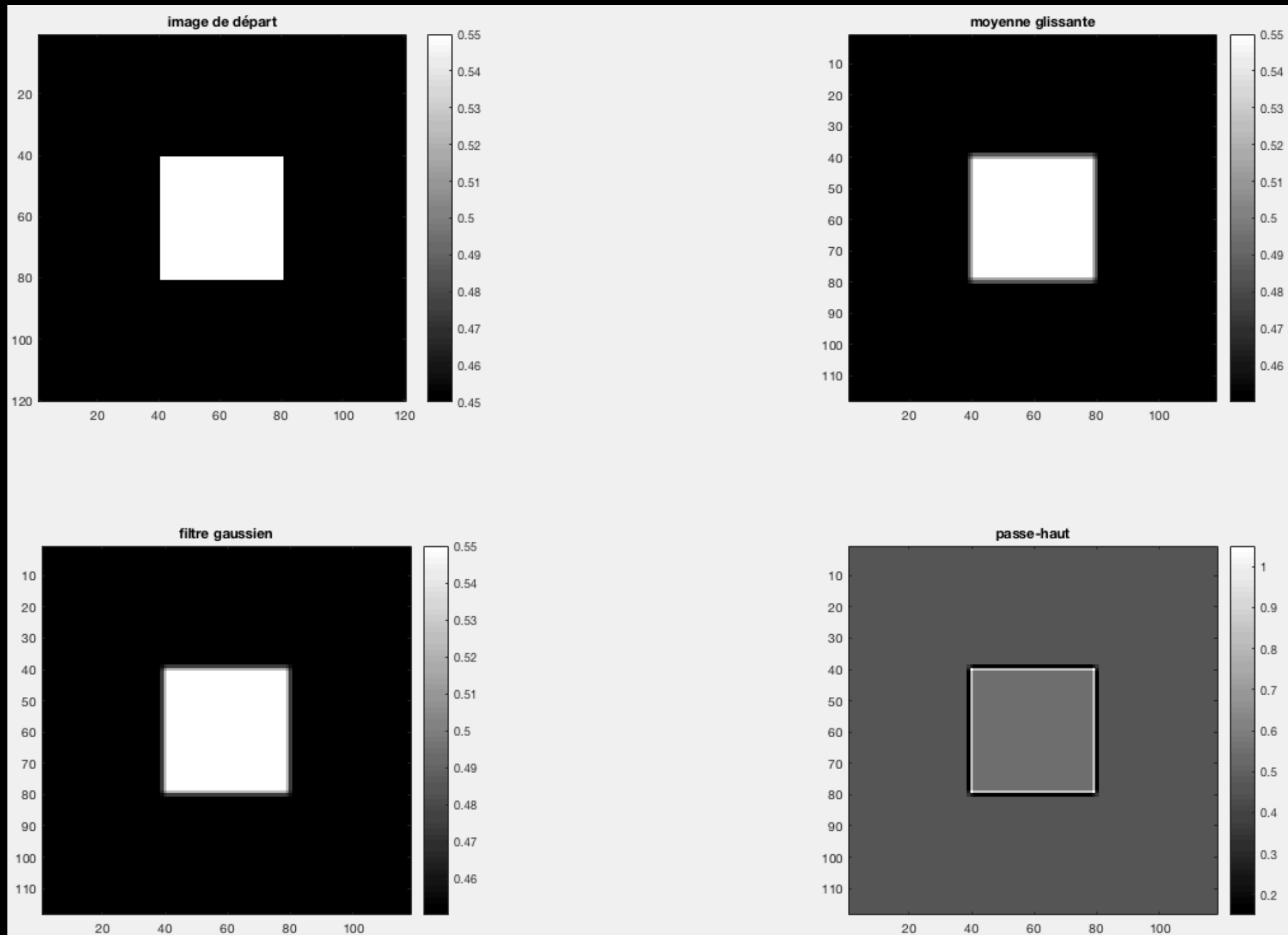
# III. Filtrage

- **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.

# 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

- Remarque : **FILTRE PASSE HAUT = a FILTRE UNITÉ - b 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  $b=9$  :

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

**FILTRE  
UNITÉ**

**FILTRE  
MOY. GLISS.**

**FILTRE  
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 FILTRE MÉDIAN
- Le **FILTRE MÉDIAN** affecte à un pixel la valeur médiane des intensités dans son entourage.
- 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 0.5).
- sous MATLAB : MEDFILT2(image)
- EXERCICE 8 : Bruiter l'image précédente (p.ex.) avec un bruit poivre & sel à 10%, puis tenter de la dé-bruiter par le filtre médian. Comparer avec une tentative de dé-bruitage résultant du filtrage linéaire utilisant les filtres suivants : moyenne glissante 3x3, moyenne glissante 5x5.

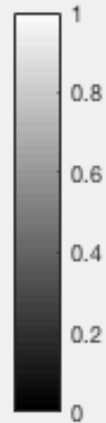
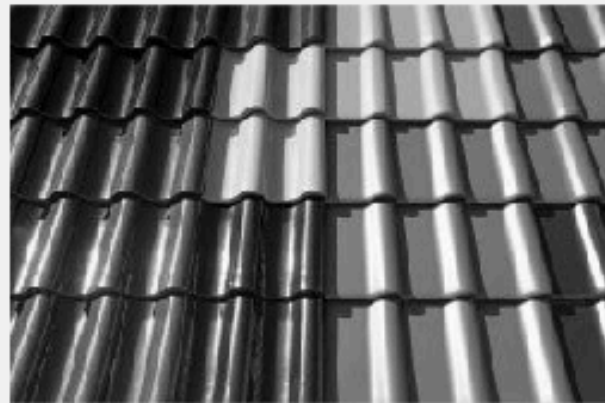


# III. Filtrage

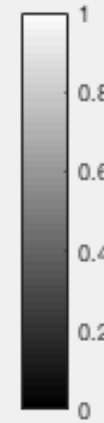
```
1 - clear
2 - close all
3
4 % image de la petite maison
5 - dir='/Users/marcel/Documents/MATLAB/0-images-exemples/';
6 - img=[dir,'materiaux/serveimage-13.jpeg'];
7 - I=imread(img);
8 - I=rgb2gray(I);
9 - I=double(I)/255.0;
10
11 %---
12 % bruit poivre et sel à hauteur de 10% de densité
13 - Ib=imnoise(I,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

# III. Filtrage

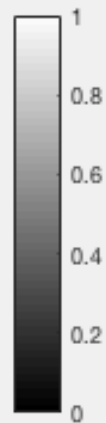
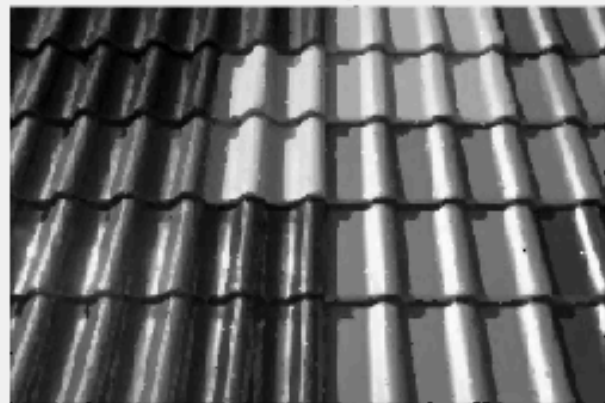
image de départ



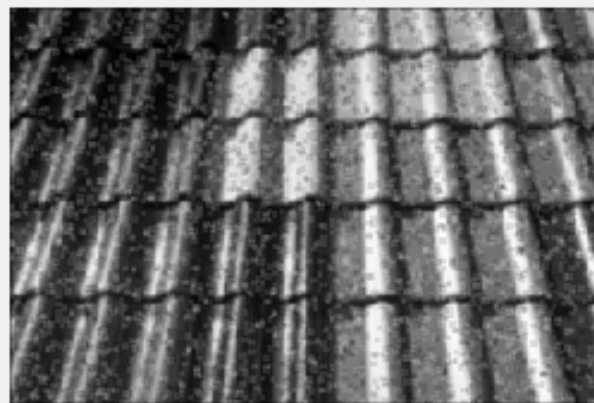
bruit p&s



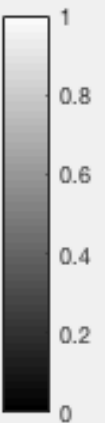
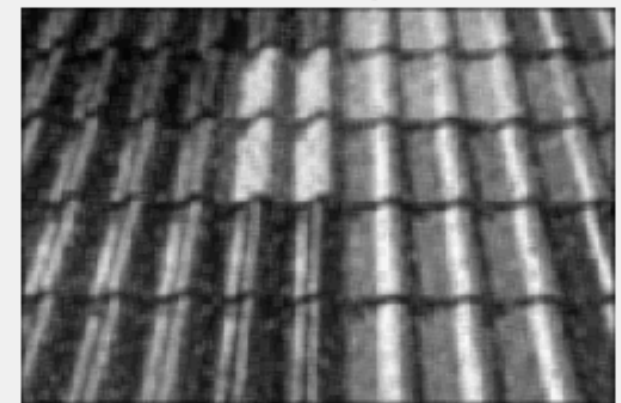
bruit p&s + filtrage médian



bruit p&s + moy. gliss. 3x3



bruit p&s + moy. gliss. 5x5



—> filtre médian très supérieur ici (bruit éliminé, pas de 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 :  $6/9=2/3\sim0.667$

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

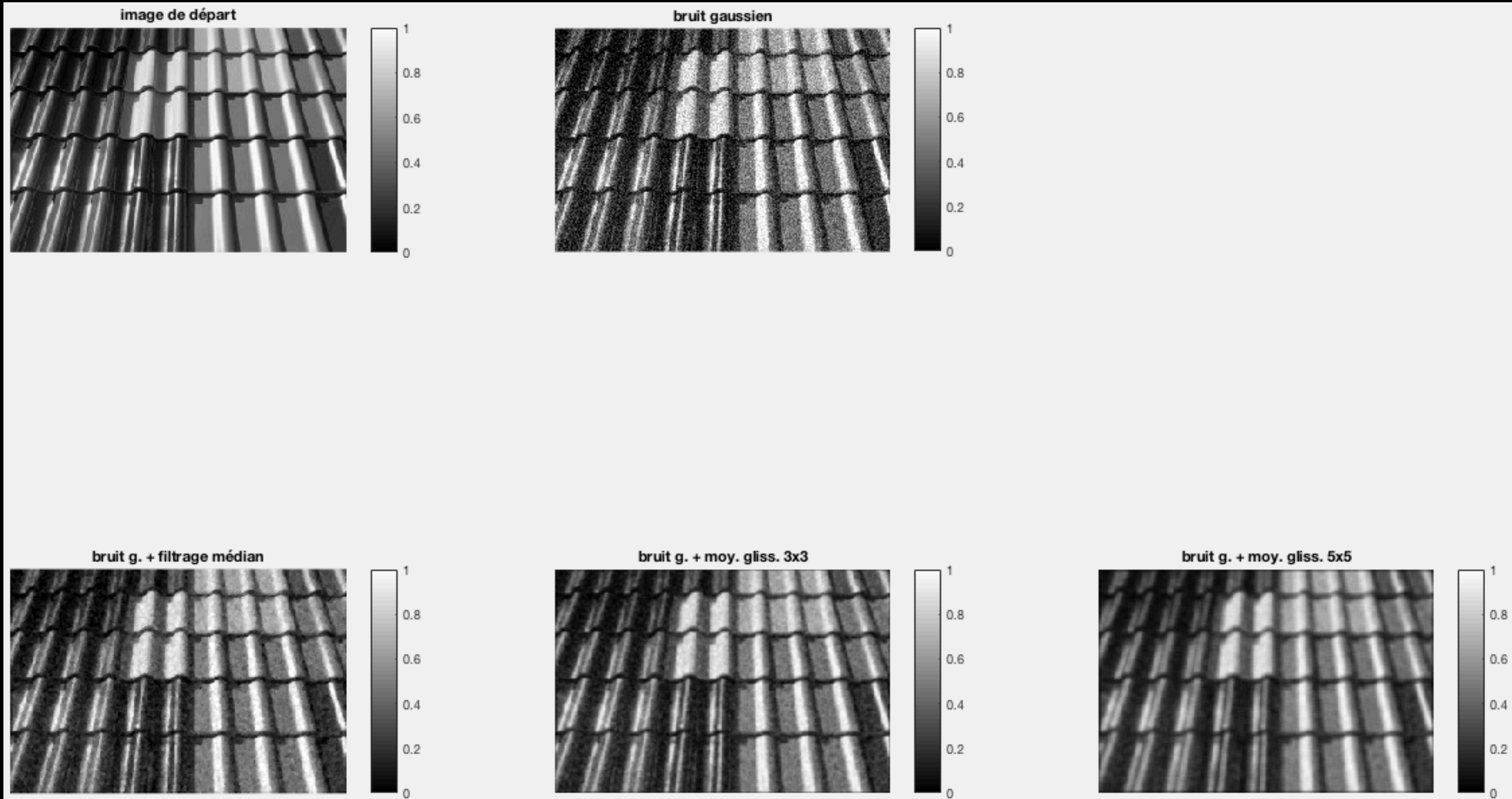
# III. Filtrage

- **EXERCICE 9 : Bruiter cette fois-ci la même image, mais avec un bruit GAUSSIEN...**

# III. Filtrage

```
1 - clear
2 - close all
3
4 % image de la petite maison
5 - dir='/Users/marcel/Documents/MATLAB/0-images-exemples/';
6 - img=[dir,'materiaux/serveimage-13.jpeg'];
7 - I=imread(img);
8 - I=rgb2gray(I);
9 - I=double(I)/255.0;
10
11 %---
12 % bruit gaussien avec une variance relative de 1%
13 - Ib=imnoise(I, 'gaussian', 0., 0.01);
14
15 % débruitage par le filtre médian
16 - Imed=medfilt2(Ib);
17
18 % débruitage par la moyenne glissante 3x3
19 - Im3=filter2(ones(3,3)/9.,Ib);
20
21 % débruitage par la moyenne glissante 5x5
22 - Im5=filter2(ones(5,5)/25.,Ib);
23
24 % figure
25 - figure
26 - subplot(2,3,1), imshow(I), title('image de départ'), colorbar
27 - subplot(2,3,2), imshow(Ib), title('bruit gaussien'), colorbar
28 - subplot(2,3,4), imshow(Imed), title('bruit g. + filtrage médian'), colorbar
29 - subplot(2,3,5), imshow(Im3), title('bruit g. + moy. gliss. 3x3'), colorbar
30 - subplot(2,3,6), imshow(Im5), title('bruit g. + moy. gliss. 5x5'), colorbar
```

# III. Filtrage



—> filtre médian beaucoup moins impressionnant ici...

# III. Filtrage

- **Remarque 1 : Filtrage médian efficace quand le bruit ne touche qu'un nombre limité de pixels (bruit impulsif de type 'poivre & sel' par exemple). En effet, dans ce cas la valeur médiane est peu affectée.**

**Si le bruit affecte tous les pixels, comme dans la cas du bruit Gaussien, l'efficacité du filtre médian est moindre car la valeur médiane peut être fortement modifiée par le bruit. Un filtre passe-bas classique est alors plutôt recommandé ici (son effet lissant pouvant donner de meilleurs résultats).**

- **Remarque 2 : Il existe d'autre filtres non-linéaires, dans le cadre de la morphologie mathématique par exemple (érosion, dilatation)...**



# III. Filtrage

## 4- FILTRAGE PRÉ-DÉFINIS SOUS MATLAB : FSPECIAL

- Remarque sur le filtre Gaussien défini à la section 2 : il s'agit bien d'une Gaussienne, d'écart-type  $\sigma = 0.849$  px.

—> vérification avec MATLAB :

```
>> hg1 = fspecial('gaussian', 3, 0.849)
```

rend :

```
0.0625  0.125  0.0625
0.125   0.25   0.125
0.0625  0.125  0.0625
```

c'est-à-dire :

```
1/16  1/8  1/16    1  2  1
1/8   1/4  1/8     = 2  4  2 x 1/16
1/16  1/8  1/16    1  2  1
```

# III. Filtrage

- La moyenne glissante peut être définie avec FSPECIAL également :

```
>> hm1 = fspecial('average', 3)
```

qui rend :

```
0.1111 0.1111 0.1111  
0.1111 0.1111 0.1111  
0.1111 0.1111 0.1111
```

c'est-à-dire :

```
1/9 1/9 1/9    1 1 1  
1/9 1/9 1/9  =  1 1 1 x 1/9  
1/9 1/9 1/9    1 1 1
```