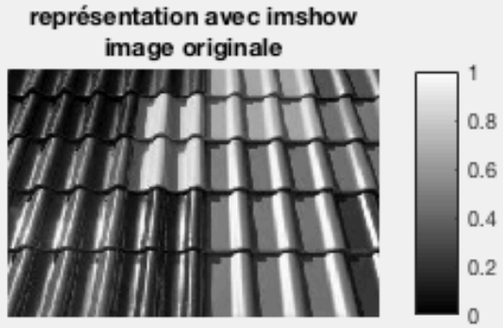
III. Filtrage

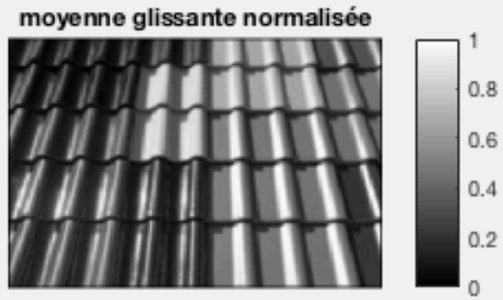
- **EXERCICE 6 : Reprendre le filtre moyenne glissante 3x3 sur une image de votre choix (i.e. celle du toit) et NE PAS respecter la normalisation. Puis diviser par 81 au lieu de 9. Que se passe-t-il ? (Utiliser pour comparaison *imshow* ET *imagesc*, bien penser à *colorbar*, utiliser les options « *same/full/valid*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

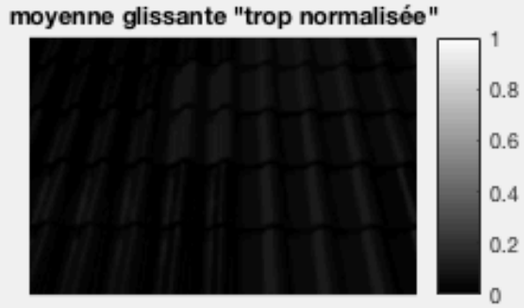
III. Filtrage

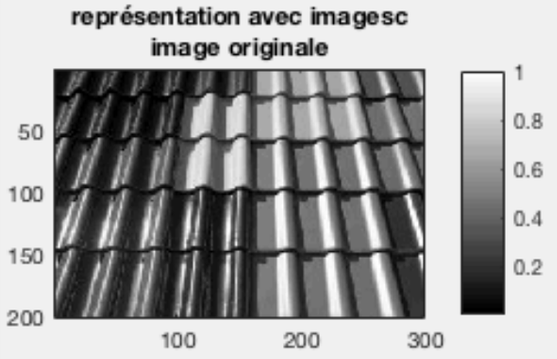
```
clear
close all

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"')
```

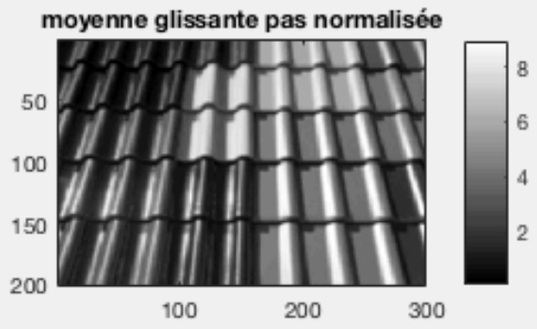
III. Filtrage

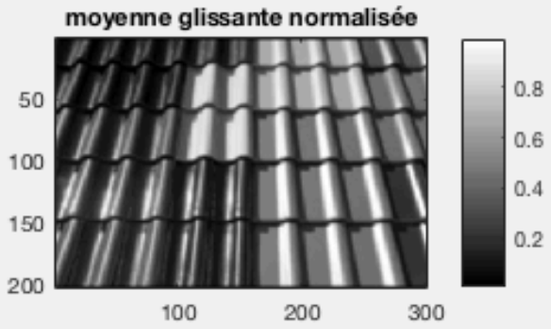


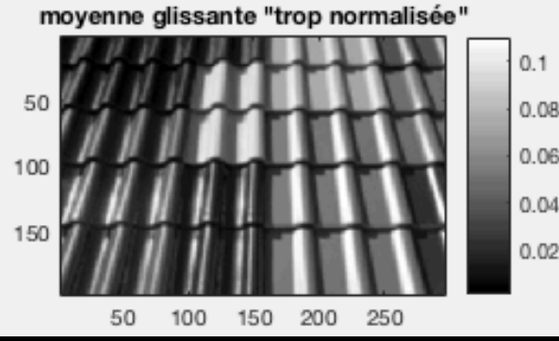












—> comme on l'avait déjà remarqué : différence visible avec *imshow*, mais pas avec *imagesc* qui ajuste automatiquement la dynamique => utiliser *colorbar* pour se rendre compte de ce que l'on a réellement sous les yeux !

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
```

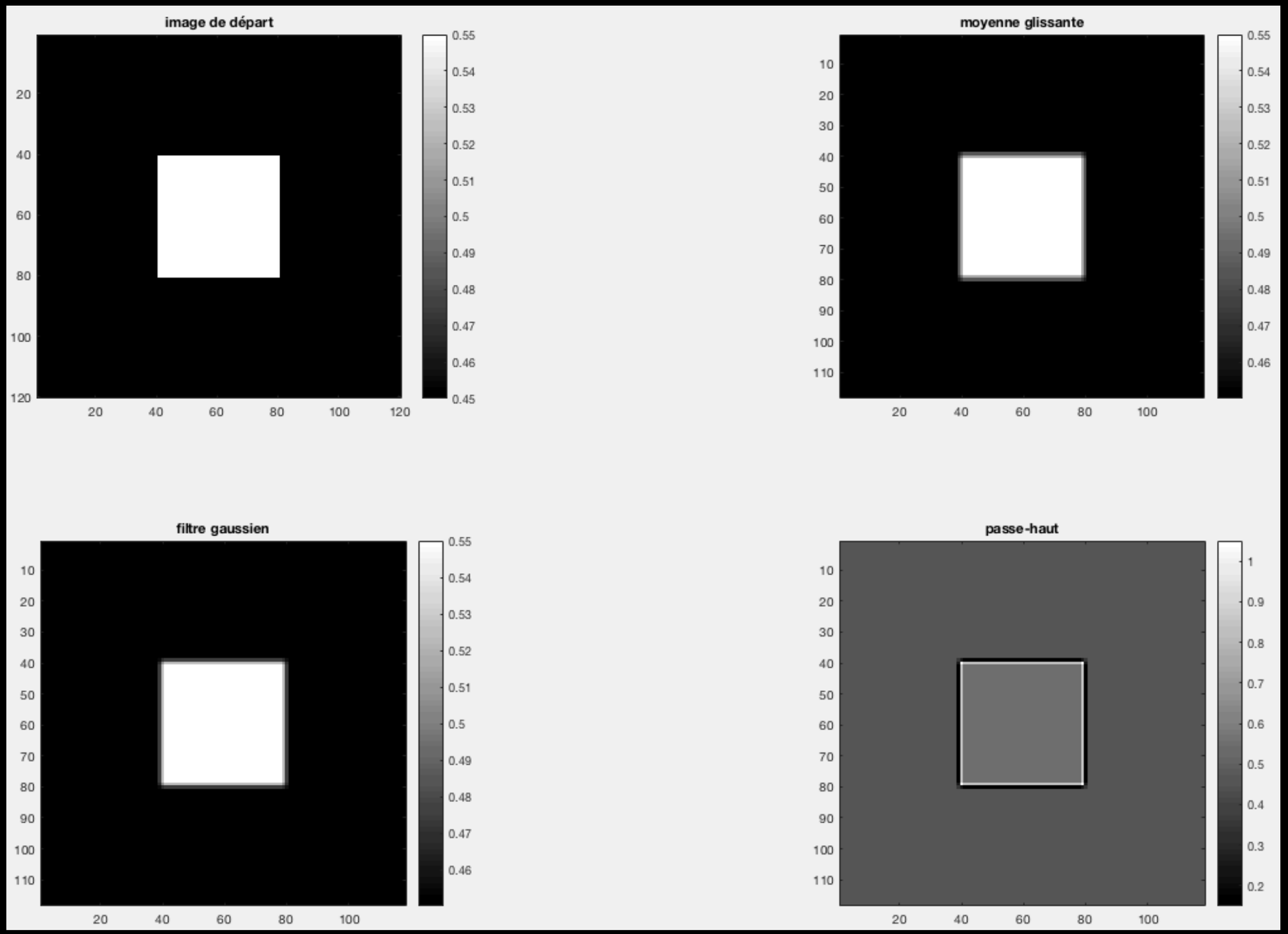
III. Filtrage

- **EXERCICE 7 (en mode TP) : 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.**

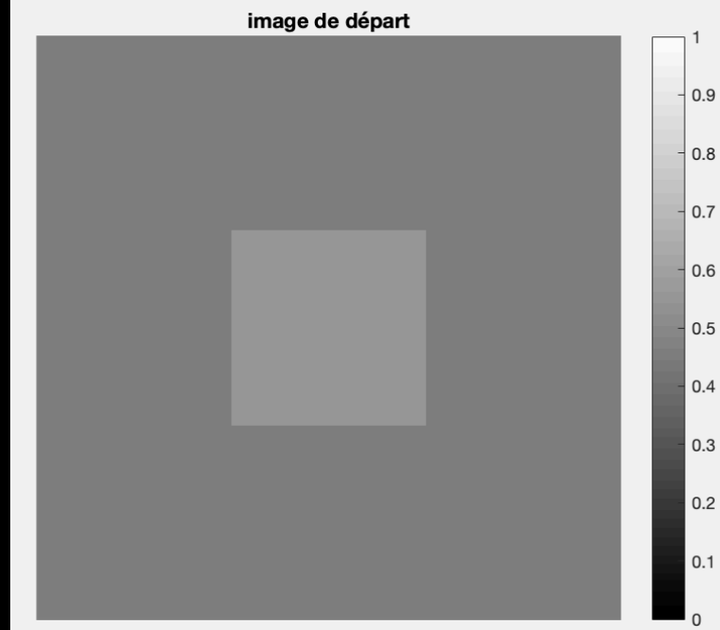
III. Filtrage

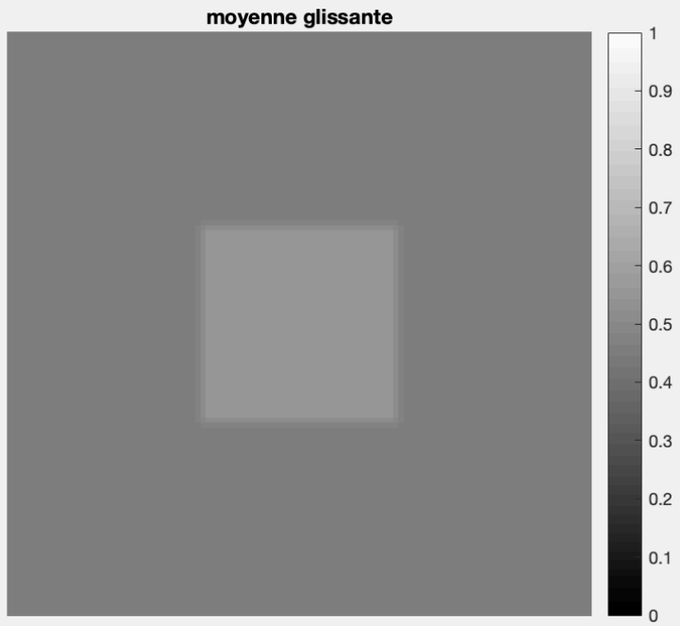
```
clear
close all

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')
```

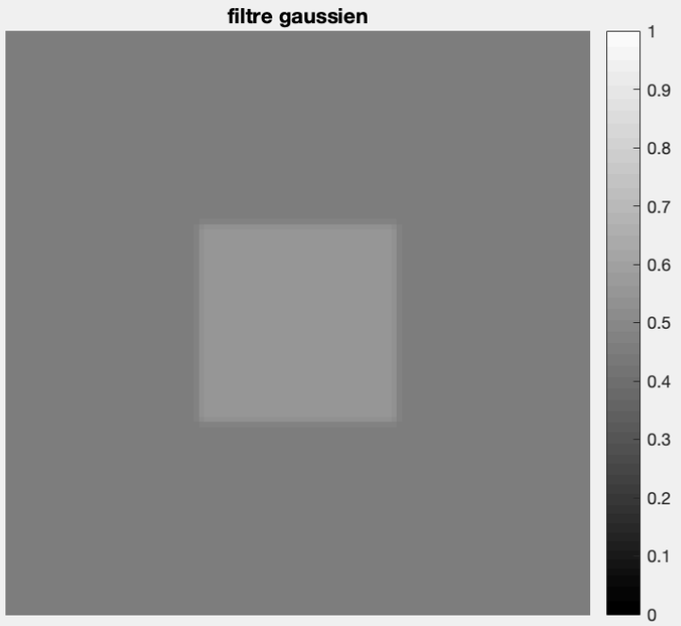
III. Filtrage

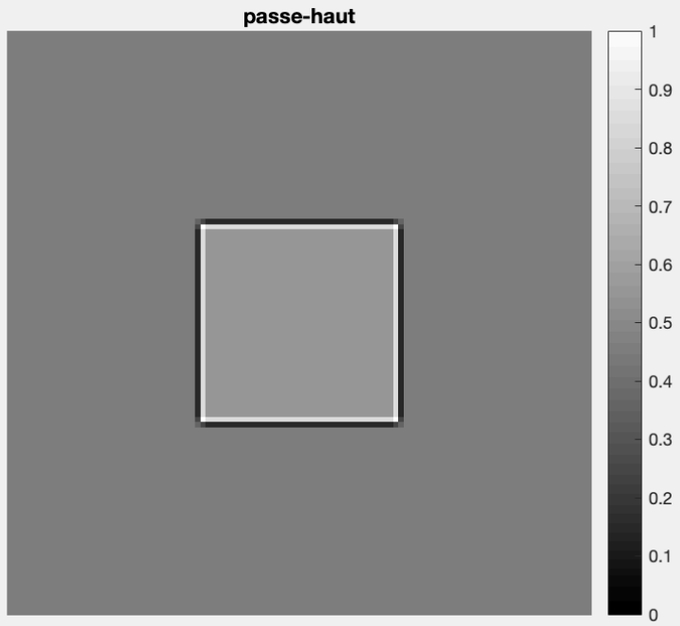


III. Filtrage









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
```

- **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 <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).**

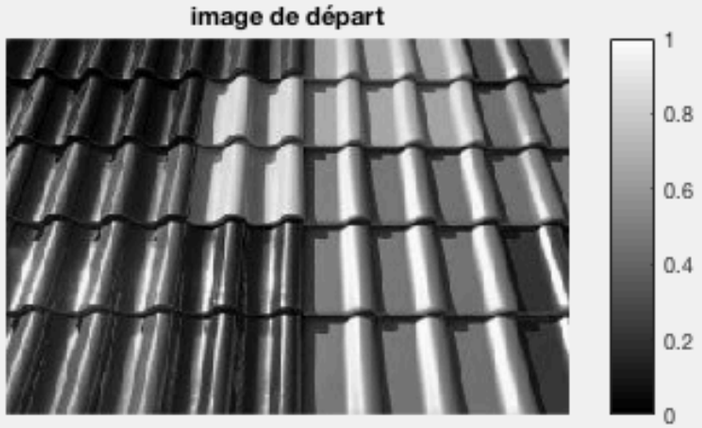
- **sous MATLAB/OCTAVE : *medfilt2(image)***

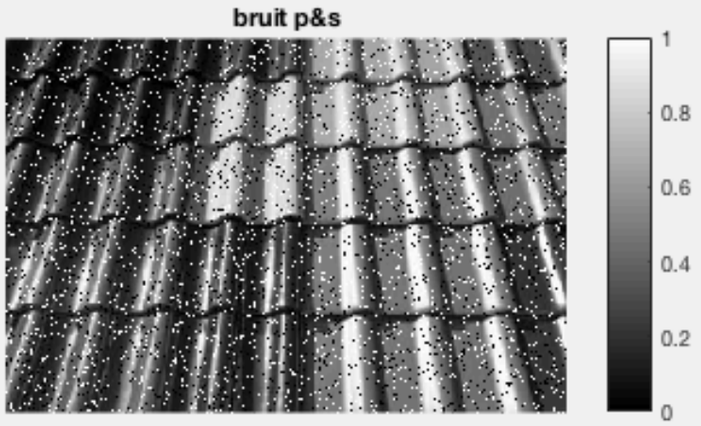
- **EXERCICE 8 : Bruiter l'image de l'exercice 6 (p.ex.) avec un bruit poivre & sel à 10%, puis tenter de la débruiter par le filtre médian. Comparer avec une tentative de débruitage résultant du filtrage linéaire utilisant les filtres suivants : moyenne glissante 3x3, moyenne glissante 5x5.**

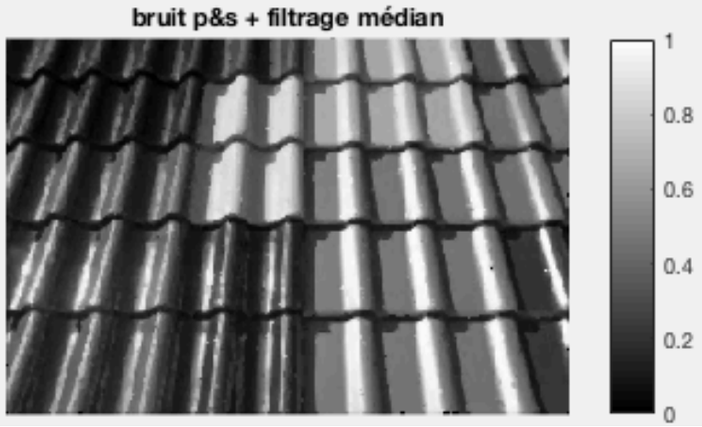
III. Filtrage

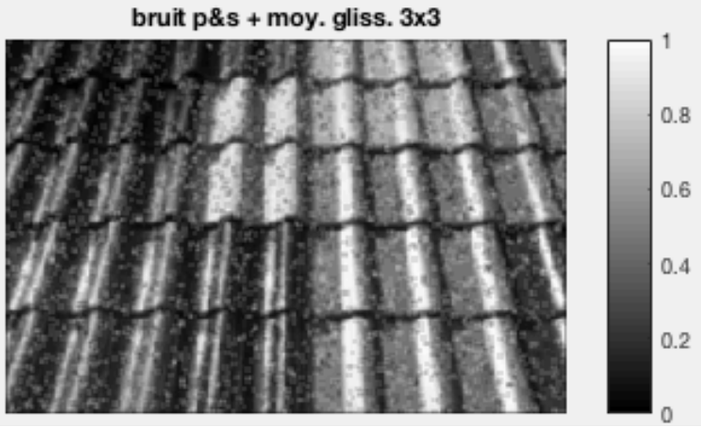
```
clear
close all
%pkg load image %Octave (pas Matlab)

% 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
```

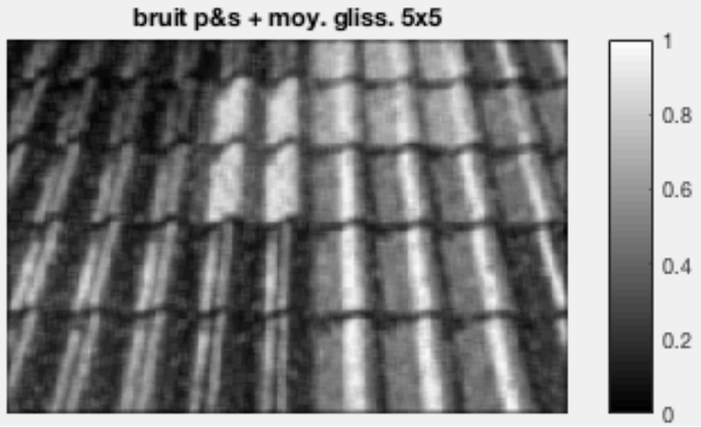
III. Filtrage











—> 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

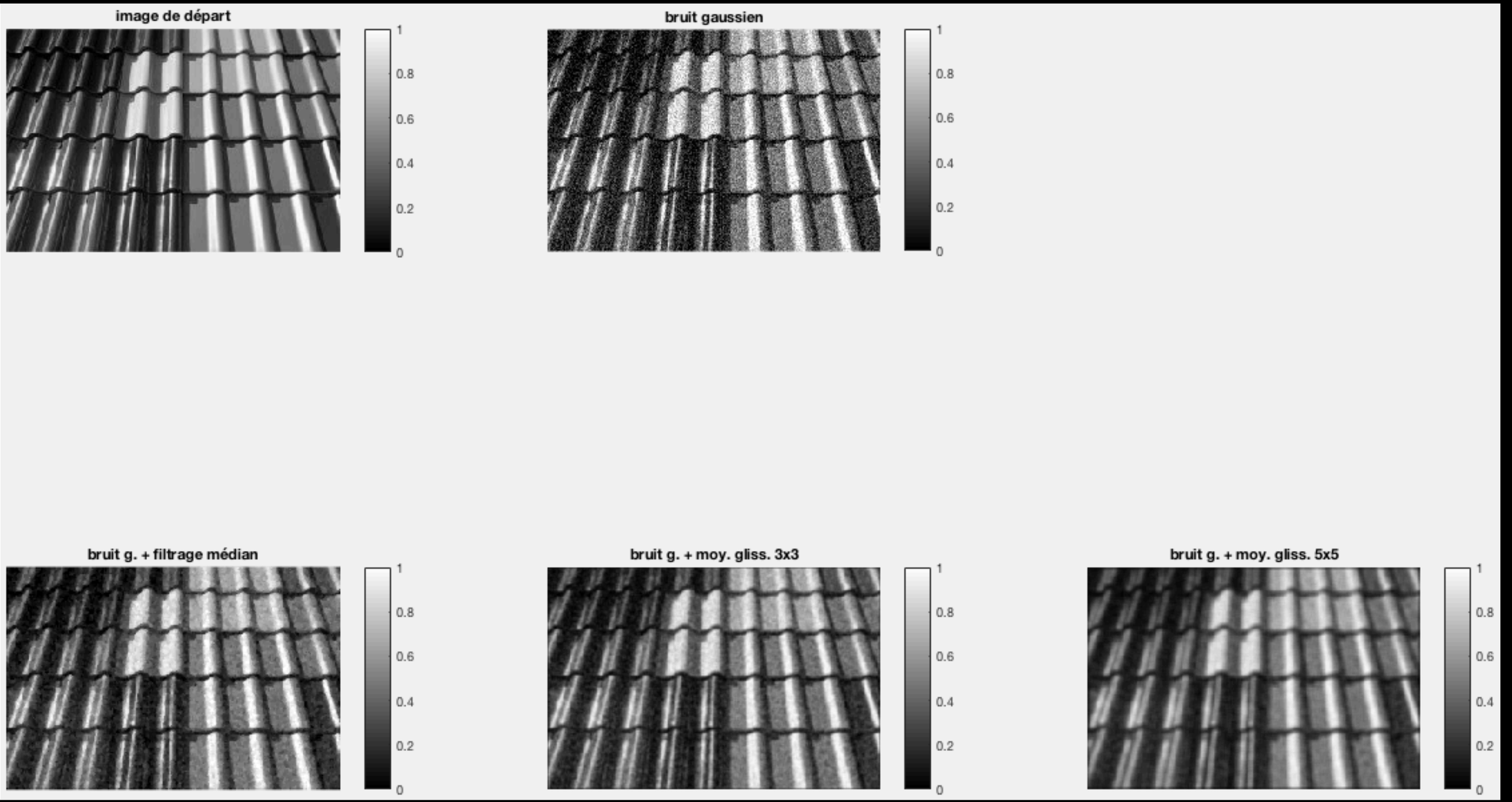
- **EXERCICE 9 : Bruiter cette fois-ci la même image avec un bruit GAUSSIEN additif (moyenne nulle, écart-type relatif de 0.1), puis faire la même comparaison.**

(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…

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/OCTAVE : 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/OCTAVE :

>> 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**

IV. Détection de contours

1- INTRODUCTION

- **Déterminer précisément les contours d'un objet peut permettre notamment de caractériser sa forme. La détection de contours peut être réalisée grâce à des filtres dont les coefficients sont judicieusement choisis.**

- **Ici : filtres de Prewitt, Roberts, Sobel.**

- **À chaque fois : une paire de filtres qui détectent les contours selon deux directions orthogonales (p.ex. un filtre vertical ET un filtre horizontal).**

IV. Détection de contours

2- FILTRES DE PREWITT

- **Filtre horizontal P_h :**

  1

  —> multiplication du vecteur colonne (0) par le vecteur ligne (1 1 1)

  -1

  1 1 1

  1 1 1

  0 0 0 0 : variation entre lignes !

  -1 -1 -1 -1

- **Filtre vertical P_v :**

  1

  —> multiplication du vecteur colonne (1) par le vecteur ligne (-1 0 1)

  1

  -1 0 1

  1 -1 0 1

  1 -1 0 1 : variation entre colonnes !

  1 -1 0 1

IV. Détection de contours

- **I_h = image I filtrée par P_h, I_v = image I filtrée par P_v.**
- **On peut définir une image de contours unique I_{v+h} :**

$$I_{v+h} = \sqrt{I_v^2 + I_h^2}$$

- **On peut aussi définir une image d'orientation des contours $I_{v/h}$:**

$$I_{h/v} = \arctan \frac{I_h}{I_v}$$

- **Pour obtenir une image de contours BINAIRE il faudra de plus choisir un seuil de détection *s* (tout ce qui est > *s* vaut 1, tout ce qui est ≤ *s* vaut 0).**