III. Filtrage

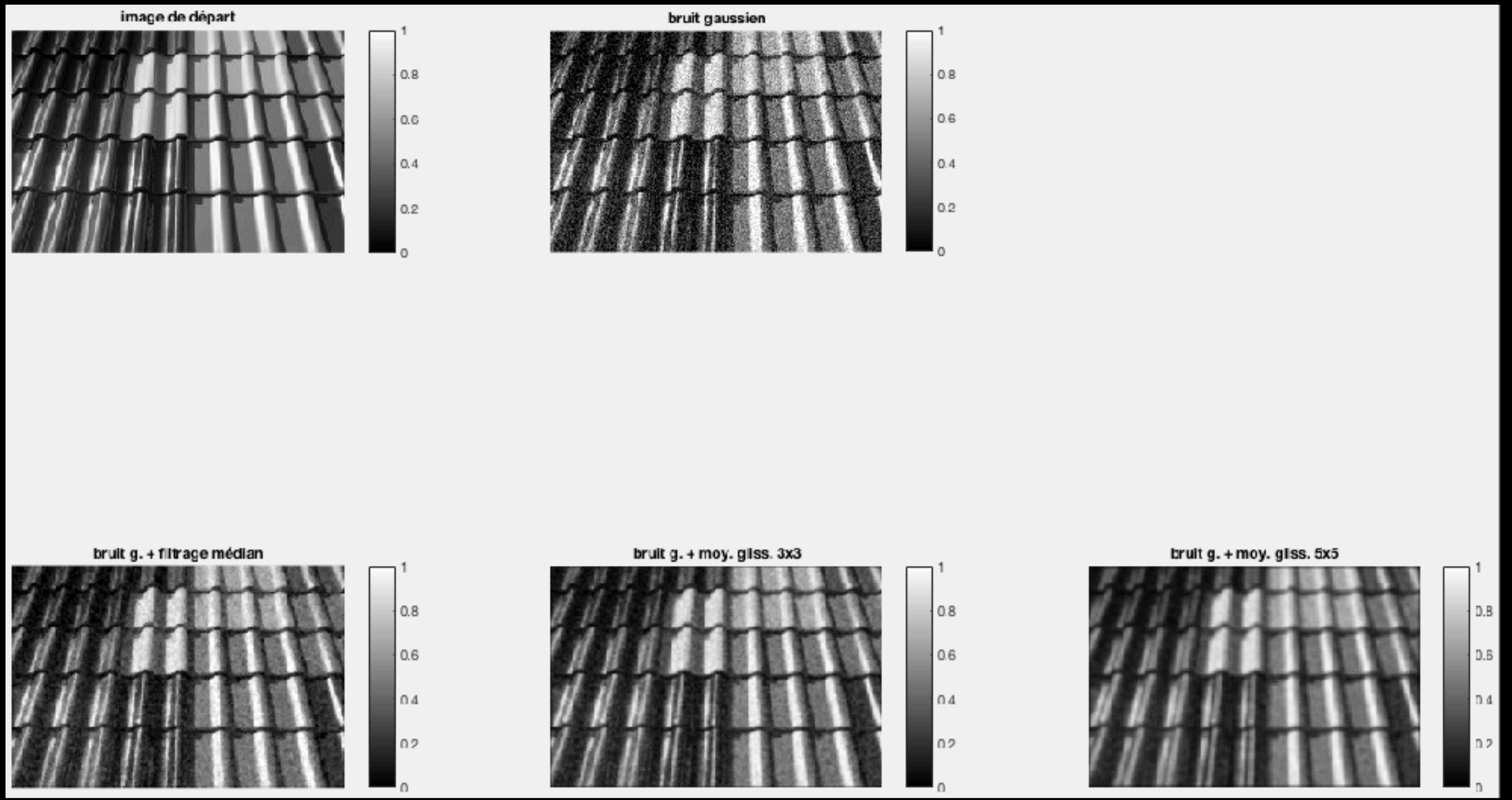
- **EXERCICE 9 : Bruiter cette fois-ci la même image avec un bruit GAUSSIEN additif (avec un écart-type relatif de 0.1), puis faire la même comparaison.**

(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…

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).**

IV. Détection de contours

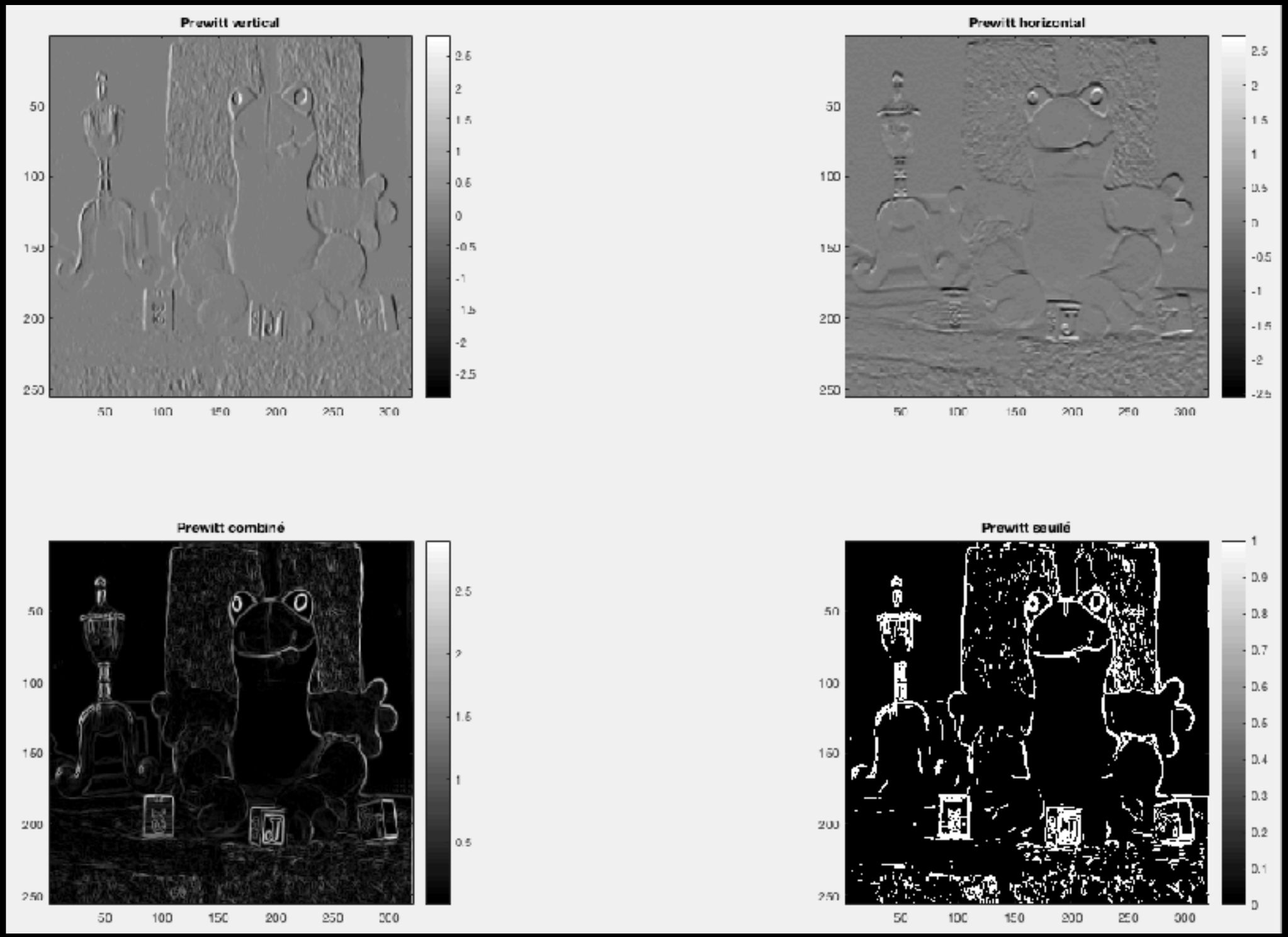
- **Exemple :**

```
>> frog=imread('…/frog.jpg') ;
>> I = rgb2gray(frog) ;
>> I = double(I)/255 ;

>> Ph = fspecial('prewitt') ;
>> Pv = -Ph' ;
>> Ih = filter2(Ph, I) ;
>> Iv = filter2(Pv, I) ;
>> Ivh = sqrt(Iv.^2+Ih.^2) ;

>> figure, colormap(gray)
>> subplot(2,2,1), imagesc(Iv), colorbar, title('Prewitt vertical'), axis('image')
>> subplot(2,2,2), imagesc(Ih), colorbar, title('Prewitt horizontal'), axis('image')
>> subplot(2,2,3), imagesc(Ivh), colorbar, title('Prewitt combiné'), axis('image')

>> seuil = 0.5 ;                    (bouger le seuil : x2 ou /2…)
>> Is = (Ivh > seuil) ;

>> subplot(2,2,4), imagesc(Is), colorbar, title('Prewitt seuillé'), axis('image')
```

IV. Détection de contours

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

% image de la grenouille
img='/Users/marcel/Documents/MATLAB/GBM/0-images/frog.jpg';
frog=imread(img);
I=rgb2gray(frog);
I=double(I)/255.;

% filtre de Prewitt
Ph=fspecial('prewitt');
Pv=-Ph';
IPh=filter2(Ph,I);
IPv=filter2(Pv,I);
IPvh=sqrt(IPv.^2+IPh.^2);

figure, colormap('gray')
subplot(2,2,1), imagesc(IPv), colorbar, title('Prewitt vertical')
   axis('image')
subplot(2,2,2), imagesc(IPh), colorbar, title('Prewitt horizontal')
   axis('image')
subplot(2,2,3), imagesc(IPvh),colorbar, title('Prewitt combiné')
   axis('image')

% seuillage
'min(IPvh)', min(min(IPvh))
'max(IPvh)', max(max(IPvh))
seuil=0.5;
IPs=(IPvh>seuil);

subplot(2,2,4), imagesc(IPs), colorbar, title('Prewitt seuillé')
axis('image')
```

IV. Détection de contours



IV. Détection de contours

3- FILTRES DE SOBEL

- **Filtre horizontal S_h :**

	1	2	1
1	1	2	1
0	0	0	0
-1	-1	-2	-1

: variation des lignes mais avec une plus grande pondération de la colonne centrale

- **Filtre vertical S_v :**

	-1	0	1
1	-1	0	1
2	-2	0	2
1	-1	0	1

: variation des colonnes mais avec une plus grande pondération de la ligne centrale

- **sous MATLAB/OCTAVE :** ***>> Sh = fspecial('sobel') ;***
  >> Sv = -Sh' ;

IV. Détection de contours

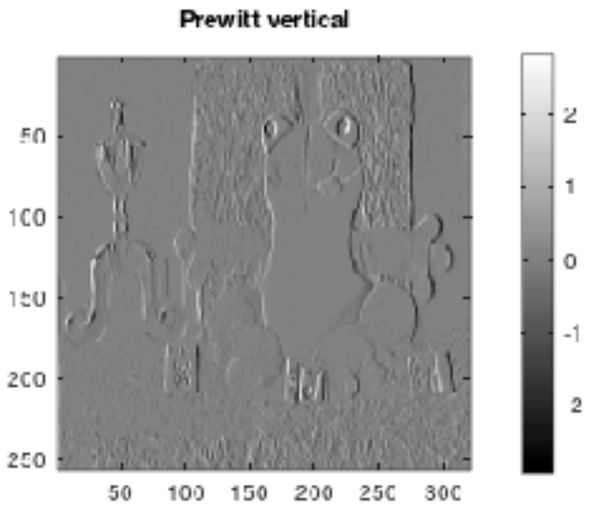
- **EXERCICE 1 : Comparer les effets respectifs des filtres de Prewitt et de Sobel sur l'image de votre choix (p.ex. la grenouille). Considérer un seuil qui paraît adapté pour l'image filtrée par Prewitt (par ex. 0.659), puis ajuster le seuil de l'image filtrée par Sobel de façon à obtenir des contours équivalents.**

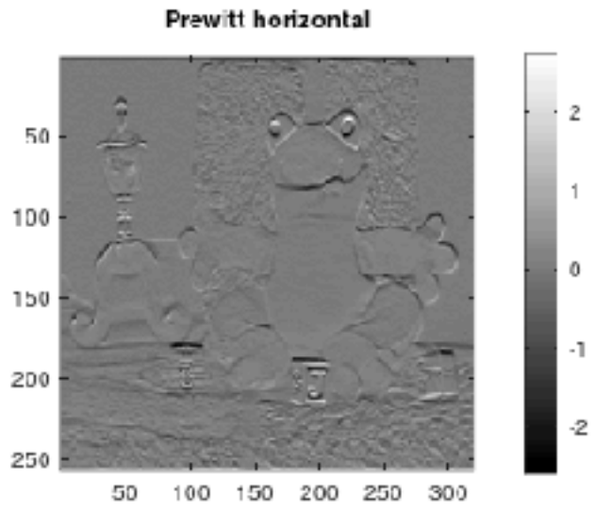
  (Pouvait-on prévoir le seuil de l'image filtrée par les filtres de Sobel en considérant la dynamique sur les deux images de contours uniques ?…)

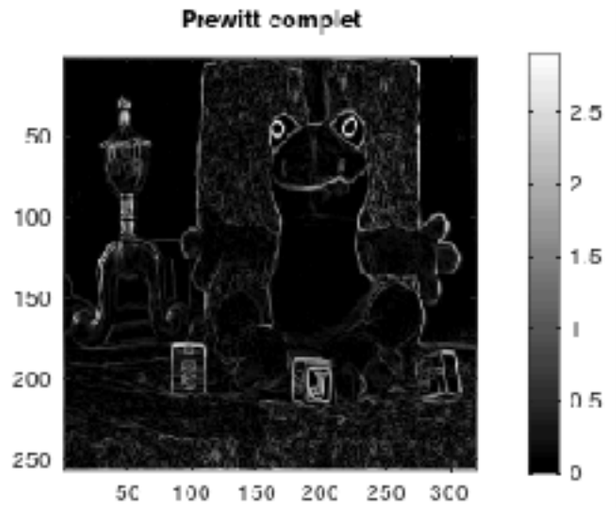
IV. Détection de contours

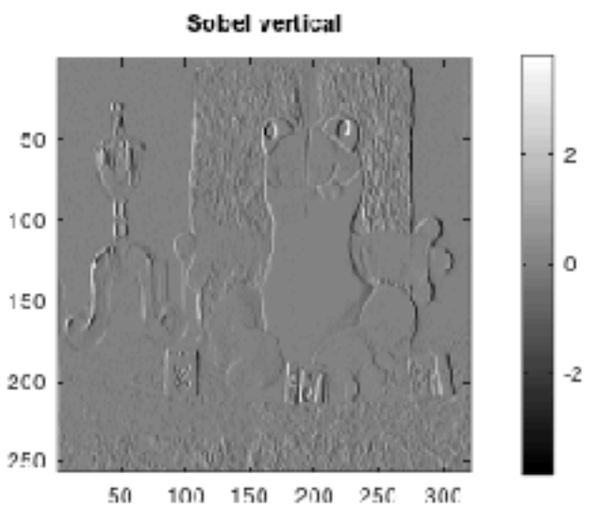
```
clear
close all
%pkg load image %Octave (pas Matlab)

% image de la grenouille
img='/Users/marcel/Documents/MATLAB/GBM/0-images/frog.jpg';
frog=imread(img);
I=rgb2gray(frog);
I=double(I)/255.;

% filtre de Prewitt
Ph=fspecial('prewitt');
Pv=-Ph';
IPh=filter2(Ph,I);
IPv=filter2(Pv,I);
IPvh=sqrt(IPv.^2+IPh.^2);

figure(1)
colormap('gray')
subplot(2,3,1), imagesc(IPv), colorbar, title('Prewitt vertical')
axis('square')
subplot(2,3,2), imagesc(IPh), colorbar, title('Prewitt horizontal')
axis('square')
subplot(2,3,3), imagesc(IPvh),colorbar, title('Prewitt complet')
axis('square')

% filtre de Sobel
Sh=fspecial('sobel');
Sv=-Sh';
ISh=filter2(Sh,I);
ISv=filter2(Sv,I);
ISvh=sqrt(ISv.^2+ISh.^2);

subplot(2,3,4), imagesc(ISv), colorbar, title('Sobel vertical')
axis('square')
subplot(2,3,5), imagesc(ISh), colorbar, title('Sobel horizontal')
axis('square')
subplot(2,3,6), imagesc(ISvh),colorbar, title('Sobel complet')
axis('square')
```

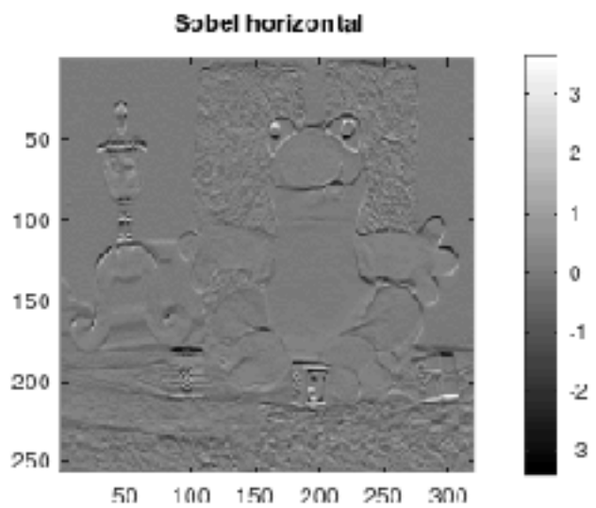
IV. Détection de contours

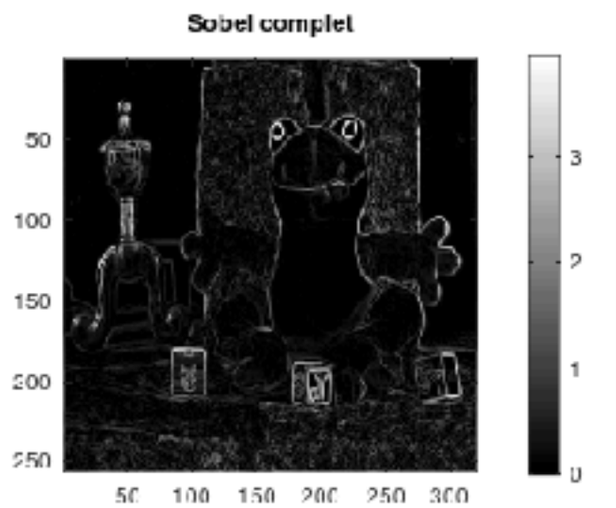








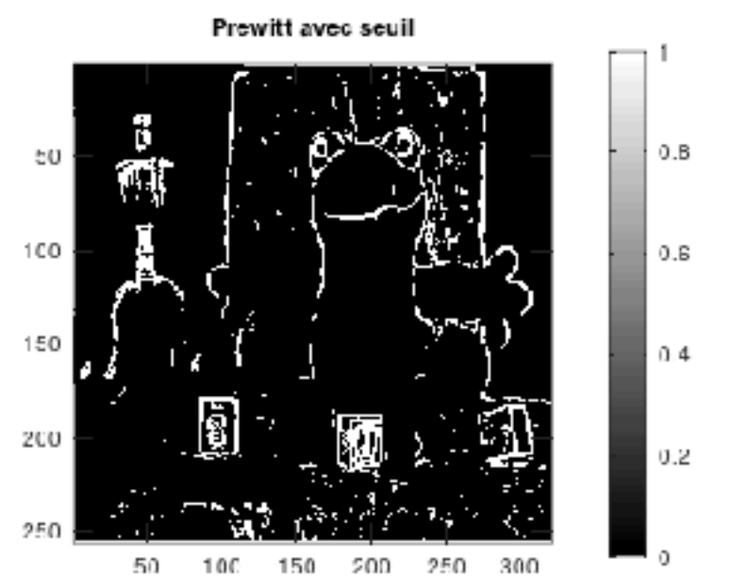


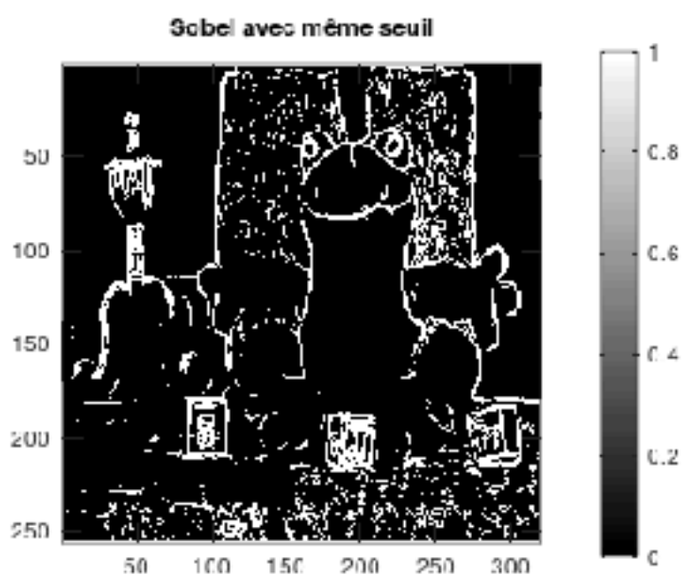


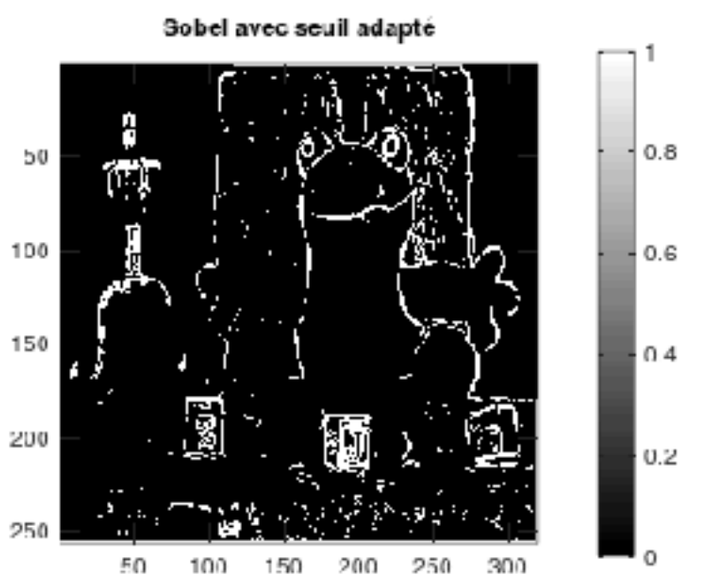
IV. Détection de contours

```
% seuillage Prewitt et comparaison avec ajustement seuil Sobel
seuil=0.659;
'seuil=', seuil
IPs=(IPvh>seuil);
ISs=(ISvh>seuil);

dynamique_ISvh = max(max(ISvh))-min(min(ISvh))
dynamique_IPvh = max(max(IPvh))-min(min(IPvh))
coeff = dynamique_ISvh/dynamique_IPvh
'rapport des dynamiques = ', coeff
seuil_Sobel=coeff*seuil;
'seuil adapté=', seuil_Sobel
ISs2=(ISvh>seuil_Sobel);

figure(2)
colormap('gray')
subplot(1,3,1), imagesc(IPs), colorbar, title('Prewitt avec seuil')
axis('square')
subplot(1,3,2), imagesc(ISs), colorbar, title('Sobel avec même seuil')
axis('square')
subplot(1,3,3), imagesc(ISs2), colorbar, title('Sobel avec seuil adapté')
axis('square')
```

IV. Détection de contours








```
>> exo1_Prewitt_Sobel

ans = seuil=
seuil = 0.6590
dynamique_ISvh = 3.9417
dynamique_IPvh = 2.9089
coeff = 1.3550
ans = rapport des dynamiques =
coeff = 1.3550
ans = seuil adapté=
seuil_Sobel = 0.8930
```

IV. Détection de contours

4- FILTRES DE ROBERTS

- **Filtre diagonal a :**

1 0
0 -1 : variation des anti-diagonales

- **Filtre anti-diagonal b :**

0 1
-1 0 : variation des diagonales

- **sous MATLAB/OCTAVE : pas de FSPECIAL ici…**

```
>> a = [1 0 ; 0 -1] ;
>> b = [0 1 ; -1 0] ;
```

ou : ***>> b = rot90(a, -1);***

IV. Détection de contours

- **EXERCICE 2 : Comparer les effets respectifs de la paire de filtres de Roberts aux deux précédentes, avec même seuil puis avec des seuils adaptés. Commenter sur la différence entre le filtrage de Roberts et les deux autres, puis sur le changement de seuil en général. Pourquoi trouve-t-on que, grosso modo :**
  - $seuil_{Sobel} \approx 4/3\ seuil_{Prewitt}$,
  - $seuil_{Roberts} \approx 1/3\ seuil_{Prewitt}$?

(Cette fois-ci avec, par exemple, l'image 'grillage.jpeg'…)

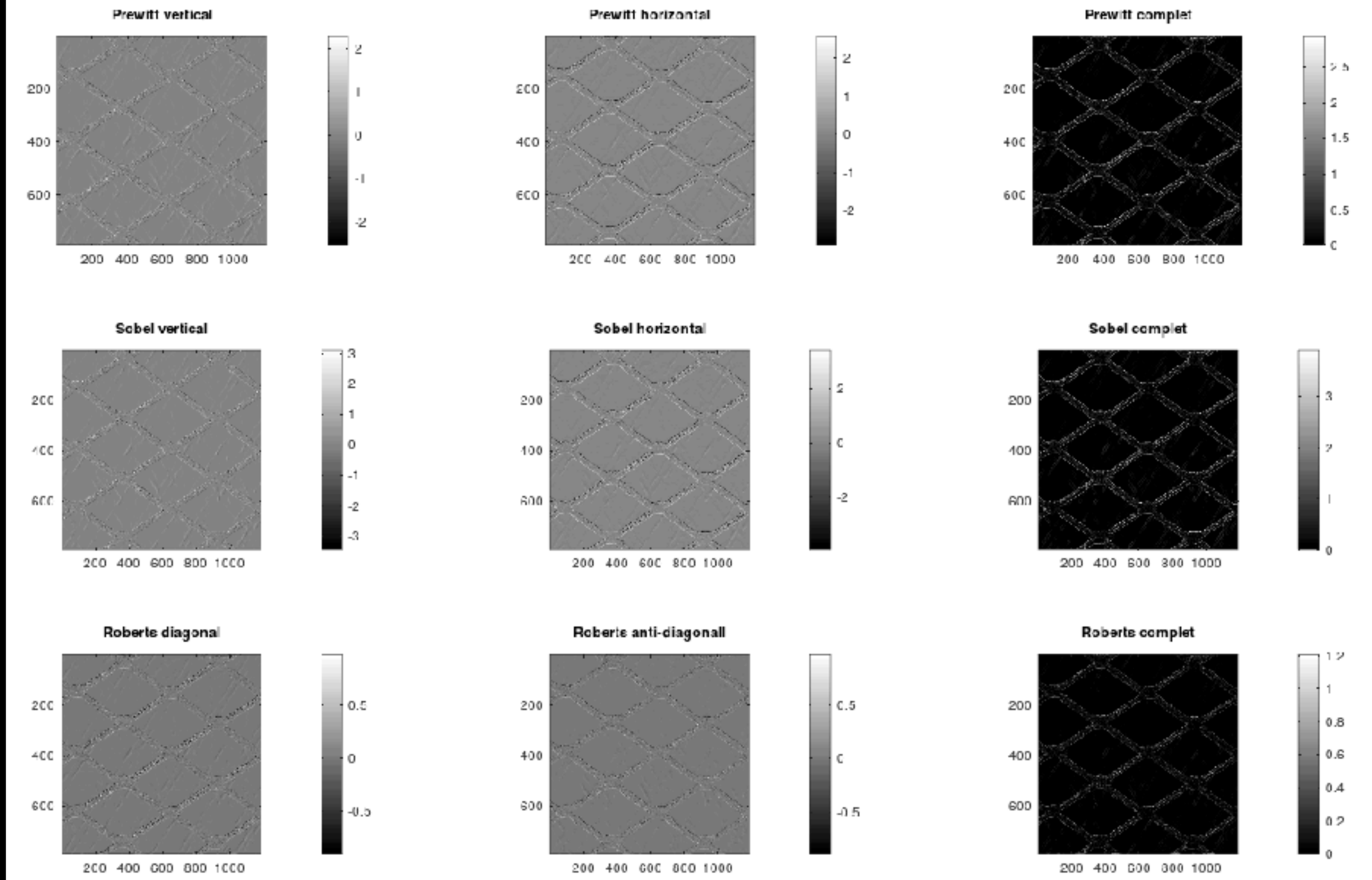
IV. Détection de contours

```
clear
close all
pkg load image

% image de la grenouille
img='/Users/marcel/Documents/MATLAB/GBM/0-images/grillage.jpeg';
frog=imread(img);
I=rgb2gray(frog);
I=double(I)/255.;

% filtre de Prewitt
Ph=fspecial('prewitt');
Pv=-Ph';
IPh=filter2(Ph,I);
IPv=filter2(Pv,I);
IPvh=sqrt(IPv.^2+IPh.^2);

figure(1)
colormap('gray')
subplot(3,3,1), imagesc(IPv), colorbar, title('Prewitt vertical')
axis('square')
subplot(3,3,2), imagesc(IPh), colorbar, title('Prewitt horizontal')
axis('square')
subplot(3,3,3), imagesc(IPvh),colorbar, title('Prewitt complet')
axis('square')

% filtre de Sobel
Sh=fspecial('sobel');
Sv=-Sh';
ISh=filter2(Sh,I);
ISv=filter2(Sv,I);
ISvh=sqrt(ISv.^2+ISh.^2);
```

IV. Détection de contours

```
subplot(3,3,4), imagesc(ISv), colorbar, title('Sobel vertical')
axis('square')
subplot(3,3,5), imagesc(ISh), colorbar, title('Sobel horizontal')
axis('square')
subplot(3,3,6), imagesc(ISvh),colorbar, title('Sobel complet')
axis('square')

% filtres de Roberts
Ra=[1 0;0 -1];
Rb=[0 1;-1 0];
IRa=filter2(Ra,I);
IRb=filter2(Rb,I);
IRab=sqrt(IRa.^2+IRb.^2);

subplot(3,3,7), imagesc(IRa), colorbar, title('Roberts diagonal')
axis('square')
subplot(3,3,8), imagesc(IRb), colorbar, title('Roberts anti-diagonall')
axis('square')
subplot(3,3,9), imagesc(IRab),colorbar, title('Roberts complet')
axis('square')

% seuillages/comparaison
seuil=0.7;
IPs=(IPvh>seuil);
'seuil Prewitt=', seuil

ISs=(ISvh>seuil);
dyn_Prewitt=max(max(IPvh))-min(min(IPvh));
dyn_Sobel=max(max(ISvh))-min(min(ISvh));
dyndyn=dyn_Sobel/dyn_Prewitt;
'dynamique Sobel/dynamique Prewitt=', dyndyn
seuil2=seuil*dyndyn;
'seuil Sobel=', seuil2
ISs2=(ISvh>seuil2);
```

IV. Détection de contours



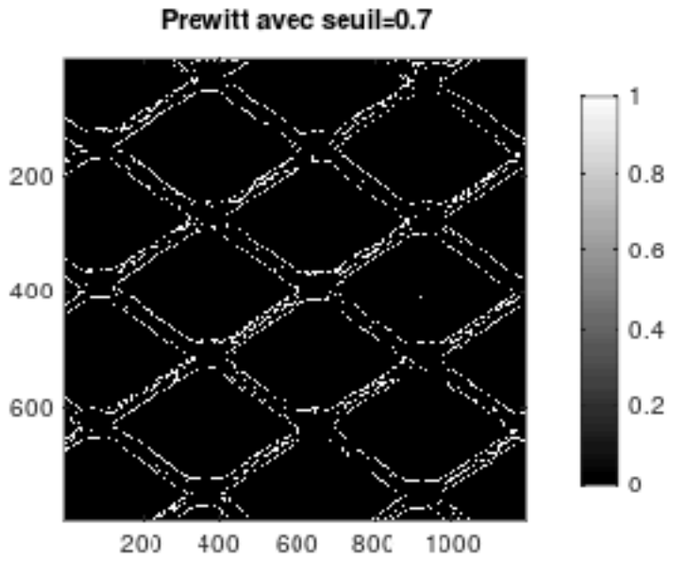
IV. Détection de contours

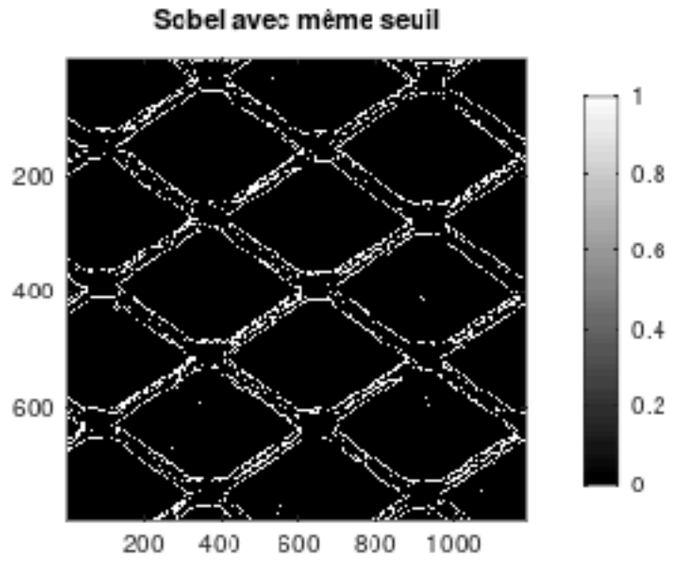
```
subplot(3,3,4), imagesc(ISv), colorbar, title('Sobel vertical')
axis('square')
subplot(3,3,5), imagesc(ISh), colorbar, title('Sobel horizontal')
axis('square')
subplot(3,3,6), imagesc(ISvh),colorbar, title('Sobel complet')
axis('square')

% filtres de Roberts
Ra=[1 0;0 -1];
Rb=[0 1;-1 0];
IRa=filter2(Ra,I);
IRb=filter2(Rb,I);
IRab=sqrt(IRa.^2+IRb.^2);

subplot(3,3,7), imagesc(IRa), colorbar, title('Roberts diagonal')
axis('square')
subplot(3,3,8), imagesc(IRb), colorbar, title('Roberts anti-diagonall')
axis('square')
subplot(3,3,9), imagesc(IRab),colorbar, title('Roberts complet')
axis('square')

% seuillages/comparaison
seuil=0.7;
IPs=(IPvh>seuil);
'seuil Prewitt=', seuil

ISs=(ISvh>seuil);
dyn_Prewitt=max(max(IPvh))-min(min(IPvh));
dyn_Sobel=max(max(ISvh))-min(min(ISvh));
dyndyn=dyn_Sobel/dyn_Prewitt;
'dynamique Sobel/dynamique Prewitt=', dyndyn
seuil2=seuil*dyndyn;
'seuil Sobel=', seuil2
ISs2=(ISvh>seuil2);
```

IV. Détection de contours

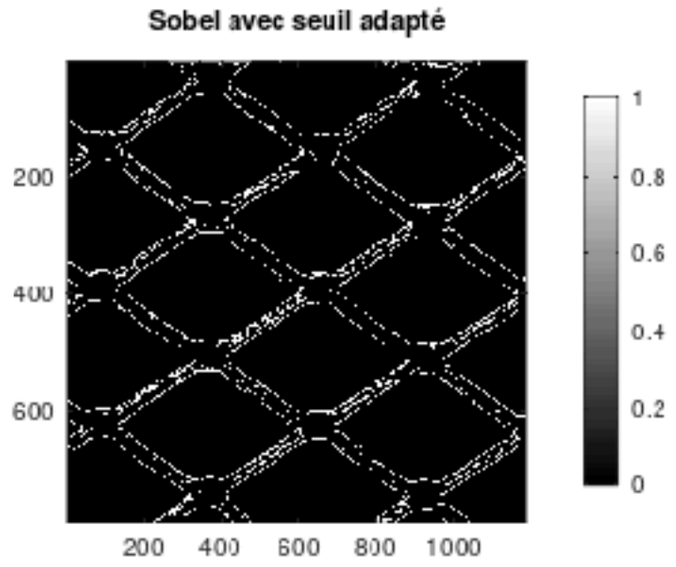
```
69  IRs=(IRab>seuil);
70  dyn_Roberts=max(max(IRab))-min(min(IRab));
71  dyndyn=dyn_Roberts/dyn_Prewitt;
72  'dynamique Roberts/dynamique Prewitt=', dyndyn
73  seuil2=seuil*dyndyn;
74  'seuil Roberts=', seuil2
75  IRs2=(IRab>seuil2);
76
77  figure(2), colormap(gray)
78  subplot(2,3,1), imagesc(IPs), colorbar
79     title(['Prewitt avec seuil=', num2str(seuil)]), axis('square')
80  subplot(2,3,2), imagesc(ISs), colorbar
81     title('Sobel avec même seuil'), axis('square')
82  subplot(2,3,3), imagesc(IRs), colorbar
83     title('Roberts avec même seuil'), axis('square')
84  subplot(2,3,5), imagesc(ISs2), colorbar
85     title('Sobel avec seuil adapté'), axis('square')
86  subplot(2,3,6), imagesc(IRs2), colorbar
87    title('Roberts avec seuil adapté'), axis('square')
```

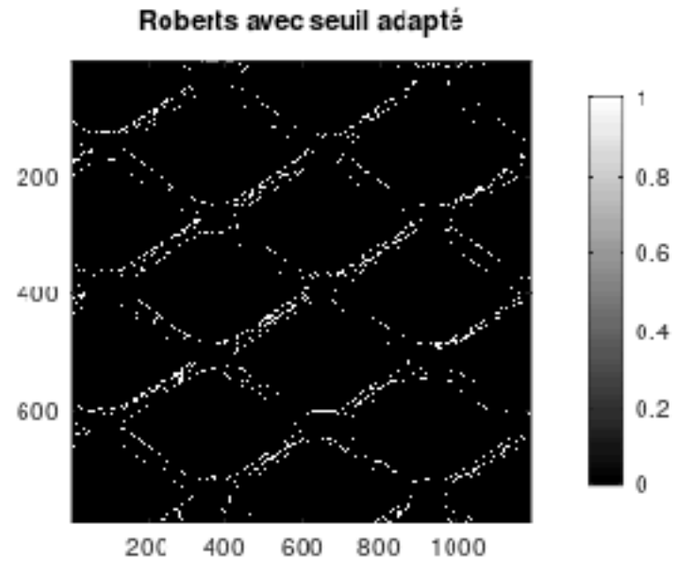
IV. Détection de contours






```
>> exo2_Roberts

ans = seuil Prewitt=
seuil = 0.7000
ans = dynamique Sobel/dynamique Prewitt=
dyndyn = 1.3368
ans = seuil Sobel=
seuil2 = 0.9358
ans = dynamique Roberts/dynamique Prewitt=
dyndyn = 0.4124
ans = seuil Roberts=
seuil2 = 0.2886
```





IV. Détection de contours

- **Visuellement, Prewitt et Sobel sont similaires.**
- **Roberts : contours plus fins (à cause de la plus faible dimension du filtre), mais contours plus cassés et plus de bruit.**
- **Seuil décroissant => contours moins cassés (=contours plus « fermés »), mais plus de bruit (=plus de « faux contours »)**
- **Seuil croissant => moins de bruit (=moins de « faux contours »), mais contours pas tous détectés (=contours plus cassés, plus « ouverts »).**

IV. Détection de contours

- **Les coefficients appliqués au seuil correspondent en fait aux rapports entre les sommes des valeurs absolues des coefficients des filtres eux-mêmes. En effet, pour un contour d'amplitude donnée, la réponse du filtre est proportionnelle à cette somme.**

  Par exemple, ici :

$$\Sigma_{|\text{coeff. de Prewitt}|} = 6 \; ; \; \Sigma_{|\text{coeff. de Roberts}|} = 2$$

$$\Rightarrow \text{seuil}_{\text{Roberts}} = \frac{1}{3} \, \text{seuil}_{\text{Prewitt}}$$

Et :

$$\Sigma_{|\text{coeff. de Prewitt}|} = 6 \; ; \; \Sigma_{|\text{coeff. de Sobel}|} = 8$$

$$\Rightarrow \text{seuil}_{\text{Sobel}} = \frac{4}{3} \, \text{seuil}_{\text{Prewitt}}$$

IV. Détection de contours

5- COMPARAISON QUANTITATIVE SUR LA DÉTECTION DE CONTOURS

- **Nous avons pour ceci besoin de métriques pertinentes :**

  -> *probabilité de détection* (d'un contour) : proportion de points de contours correctement détectés.

  -> *probabilité de fausse alarme* : proportion de points de contours détectés à tort.

- **Mais aussi d'une image simple avec un contour simple (par exemple un contour vertical).**

- **Et enfin d'une stratégie de comparaison : par exemple régler les seuils des 3 filtrages de manière à avoir la même probabilité de fausse alarme (P_{fa}), puis comparer ensuite les probabilités de détection (P_d) obtenues (le contraire eût été tout aussi pertinent).**

IV. Détection de contours

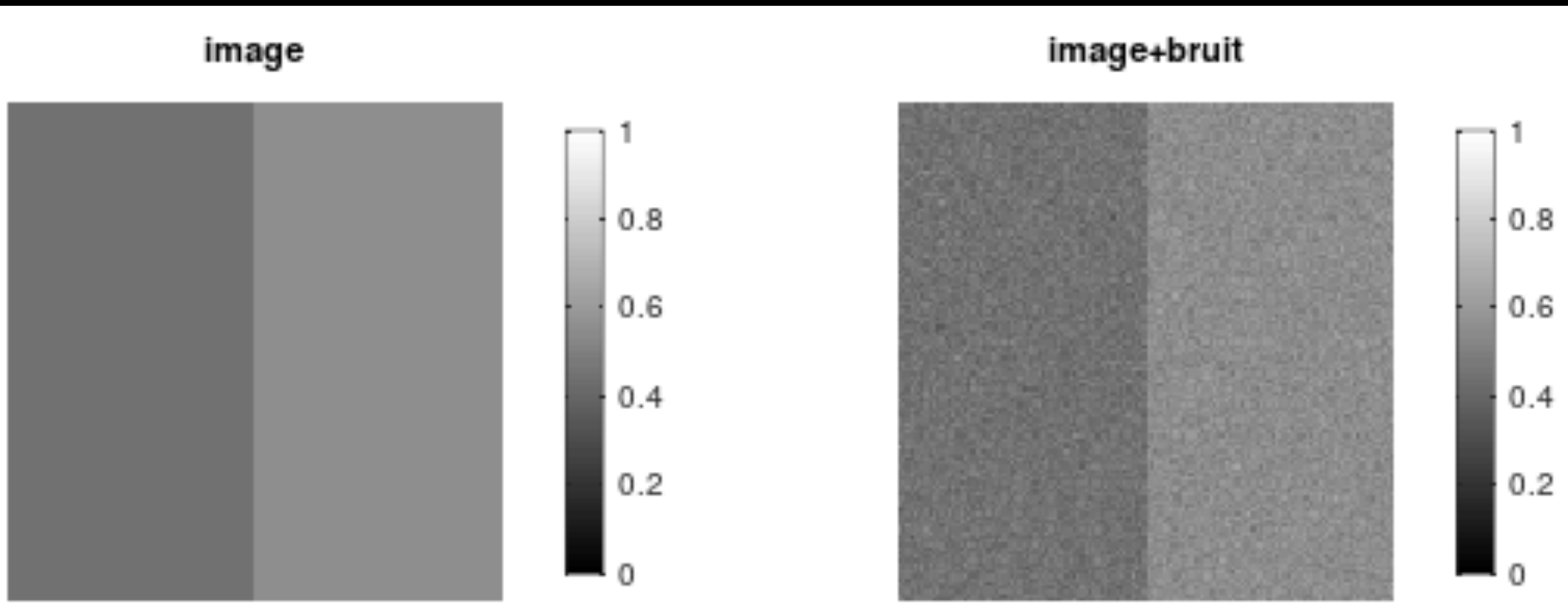
- **EXERCICE 3 : Comparer quantitativement les filtrages de Prewitt, Sobel et Roberts sur une image 128x128 composée d'une première moitié grise légèrement foncée (à gauche), à 45% de luminosité, et d'une autre moitié grise plus claire à droite (à 55% de luminosité), puis bruitée (bruit Gaussien additif de moyenne nulle et de variance 0.001).**

  (Commencer par un seuil de 0.25 pour Prewitt, sauver l'image bruitée afin de ne pas la changer à chaque exécution du programme => faire donc deux routines : une pour créer l'image bruitée, l'autre pour la traiter.)

  (Faire ça en 3 étapes : (a) générer l'image bruitée (première routine) ; (b) en déduire les images de contour binaires pour un nombre fixé de fausses alarmes (deuxième routine) ; (c) compléter la deuxième routine par le calcul de la probabilité de détection pour chacun des 3 cas de paire de filtres.)

IV. Détection de contours

```
clear
close all

% image test
dim=128; I=.55*ones(dim,dim); I(:,1:dim/2)=.45;
figure, colormap(gray)
subplot(1,2,1), imshow(I), title('image'), colorbar

% bruit gaussien additif
J=imnoise(I, 'gaussian', 0., 1e-3);
subplot(1,2,2), imshow(J), title('image+bruit'), colorbar

% sauver l'image de test bruitée
save image_bruit J
```



IV. Détection de contours

```
clear
close all
pkg load image

%---
% load image from disk
load image_bruit, whos J
% figure 1
figure(1), colormap(gray)
subplot(2,2,1), imagesc(J), title('image test'), colorbar, axis('square')
%---
% Prewitt
Ph=fspecial('prewitt'); Pv=-Ph'; JP=filter2(Pv,J, 'valid');
subplot(2,2,2), imagesc(JP), title('Prewitt'), colorbar, axis('square')
%---
% Sobel
Sh=fspecial('sobel'); Sv=-Sh'; JS=filter2(Sv,J, 'valid');
subplot(2,2,3), imagesc(JS), title('Sobel'), colorbar, axis('square')
%---
% Roberts
Ra=[1 0;0 -1]; Rb=rot90(Ra,-1);
JRa=filter2(Ra,J, 'valid'); JRb=filter2(Rb,J, 'valid');
JR=sqrt(JRa.*JRa+JRb.*JRb);
subplot(2,2,4), imagesc(JR), title('Roberts'), colorbar, axis('square')
%---
% comparaison des images uniques de contours seuillées
% seuillages
seuilP=.25; JPs=JP>seuilP;
seuilS=.362; JSs=JS>seuilS;
seuilR=.175; JRs=JR>seuilR;
```

```
% figure 2
figure(2), colormap(gray)
imagesc(JPs), title({['Prewitt+seuil=',num2str(seuilP)];'=> 9 FA'}), colorbar
   axis('square')
% figure 3
figure(3), colormap(gray)
imagesc(JSs), title({['Sobel+seuil=',num2str(seuilS)]  ;'=> 9 FA'}), colorbar
   axis('square')
% figure 4
figure(4), colormap(gray)
imagesc(JRs), title({['Roberts+seuil=',num2str(seuilR)];'=> 9 FA'}), colorbar
   axis('square')
```