

# III. Filtrage

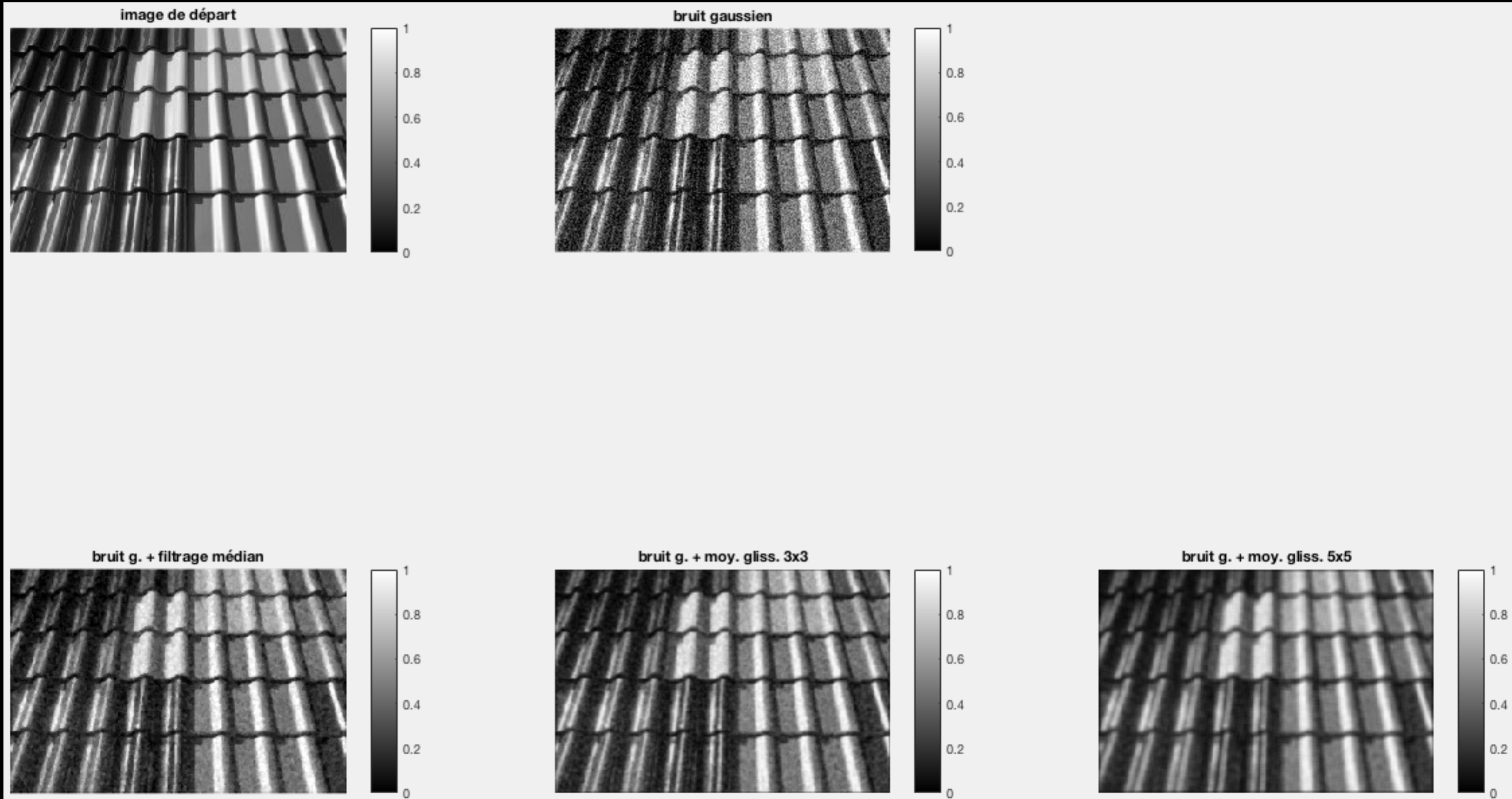
- **EXERCICE 9 : Bruiter cette fois-ci la même image avec un bruit GAUSSIEN additif (moyenne nulle, écart-type relatif de 0.1), puis faire la même comparaison. Lequel des filtres testés est le plus adapté ?**

**(Attention à ne pas confondre BRUIT GAUSSIEN et FILTRE GAUSSIEN...)**

# 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 gaussien avec une variance relative de 1%
14 Ib=imnoise(I, 'gaussian', 0., 0.01);
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.,Ib);
21
22 % débruitage par la moyenne glissante 5x5
23 Im5=filter2(ones(5,5)/25.,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 gaussien'), colorbar
29 subplot(2,3,4), imshow(Imed), title('bruit g. + filtrage médian'), colorbar
30 subplot(2,3,5), imshow(Im3), title('bruit g. + moy. gliss. 3x3'), colorbar
31 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  $P_h$  :

—> multiplication du vecteur colonne  $\begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$  par le vecteur ligne  $(1 \ 1 \ 1)$

$1 \ 1 \ 1$

$\begin{matrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 \end{matrix}$  : variation entre lignes !

- Filtre vertical  $P_v$  :

—> multiplication du vecteur colonne  $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$  par le vecteur ligne  $(-1 \ 0 \ 1)$

$-1 \ 0 \ 1$

$\begin{matrix} 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & 1 \end{matrix}$  : variation entre colonnes !

# 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  $I_{h/v}$  :

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

- 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  $\leq s$  vaut 0).

# IV. Détection de contours

- Exemple :

```
>> frog=imread('.../frog.jpg') ;
```

```
>> I = rgb2gray(frog) ;
```

```
>> I = double(I)/255 ;
```

```
>> Ph = fspecial('prewitt')
```

```
>> Pv = -Ph'
```

```
>> lh = filter2(Ph, I) ;
```

```
>> lv = filter2(Pv, I) ;
```

```
>> lvh = sqrt(lv.^2+lh.^2) ;
```

```
>> figure, colormap(gray)
```

```
>> subplot(2,2,1), imagesc(lv), colorbar, title('Prewitt vertical'), axis('image')
```

```
>> subplot(2,2,2), imagesc(lh), colorbar, title('Prewitt horizontal'), axis('image')
```

```
>> subplot(2,2,3), imagesc(lvh), colorbar, title('Prewitt combiné'), axis('image')
```

```
>> seuil = 0.5 ;
```

(bouger le seuil en fct de la dynamique...)

```
>> ls = (lvh > seuil) ;
```

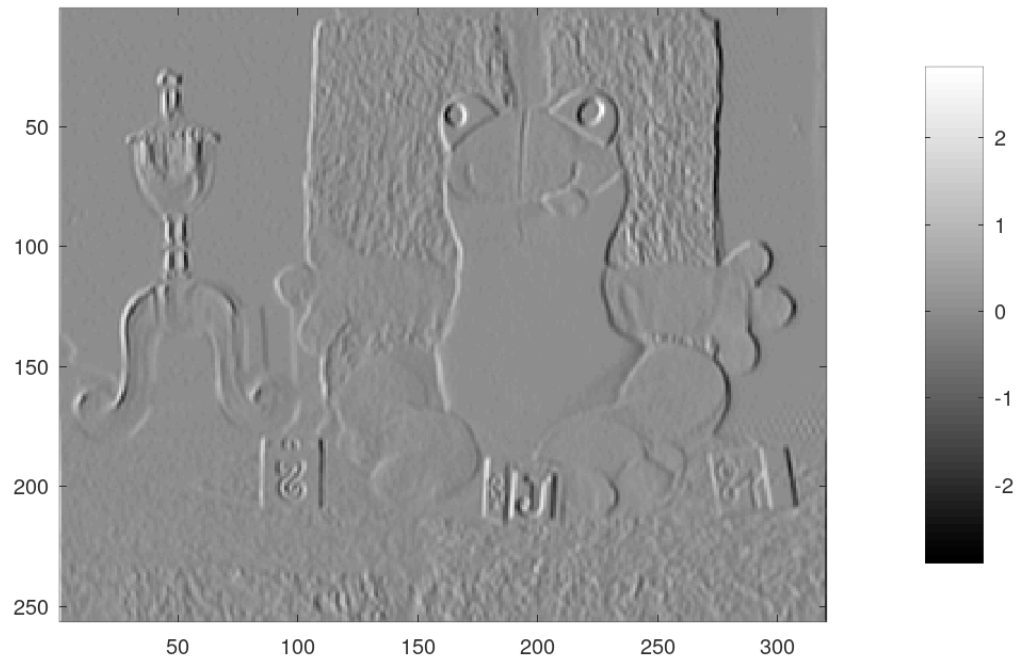
```
>> subplot(2,2,4), imagesc(ls), colorbar, title('Prewitt seuillé'), axis('image')
```

# IV. Détection de contours

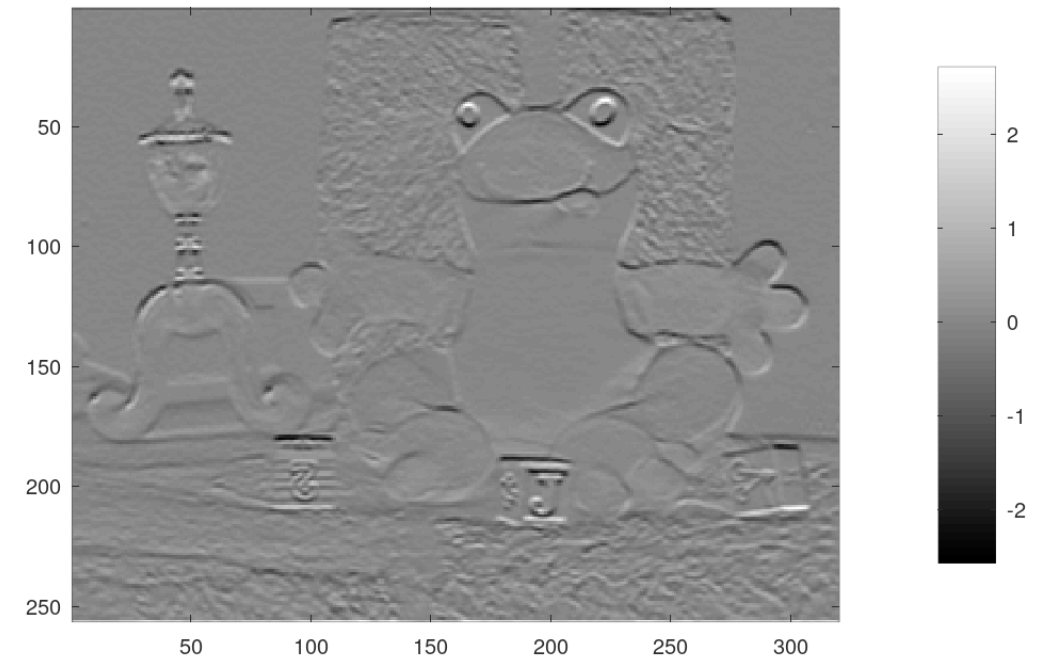
```
1 clear
2 close all
3 pkg load image %Octave (pas Matlab)
4
5 % image de la grenouille
6 img='/Users/marcel/Documents/MATLAB/0-images/frog.jpg';
7 frog=imread(img);
8 I=rgb2gray(frog);
9 I=double(I)/255.;
10
11 % filtre de Prewitt
12 Ph=fspecial('prewitt')
13 Pv=-Ph'
14 IPh=filter2(Ph,I);
15 IPv=filter2(Pv,I);
16 IPvh=sqrt(IPv.^2+IPh.^2);
17
18 figure, colormap('gray')
19 subplot(2,2,1), imagesc(IPv), colorbar, title('Prewitt vertical')
20     axis('image')
21 subplot(2,2,2), imagesc(IPh), colorbar, title('Prewitt horizontal')
22     axis('image')
23 subplot(2,2,3), imagesc(IPvh), colorbar, title('Prewitt combiné')
24     axis('image')
25
26 % seuillage
27 mini=min(min(IPvh))
28 maxi=max(max(IPvh))
29 seuil=0.659;
30 IPs=(IPvh>seuil);
31
32 subplot(2,2,4), imagesc(IPs), colorbar, axis('image')
33 title({'Prewitt seuillé'; ['min=', num2str(mini), ' | max=', num2str(maxi), ' | seuil=', num2str(seuil)]})
```

# IV. Détection de contours

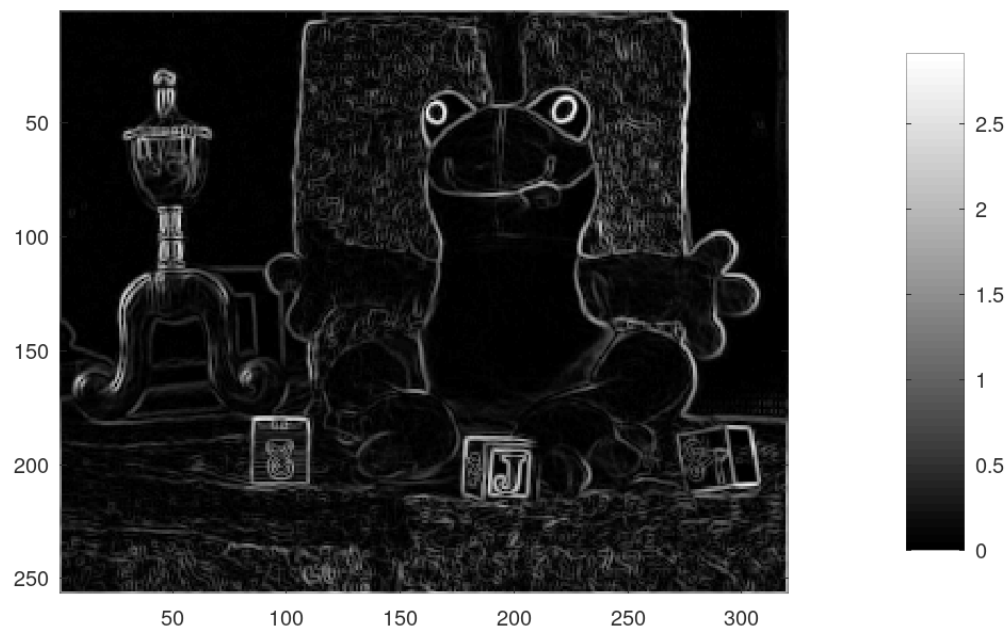
Prewitt vertical



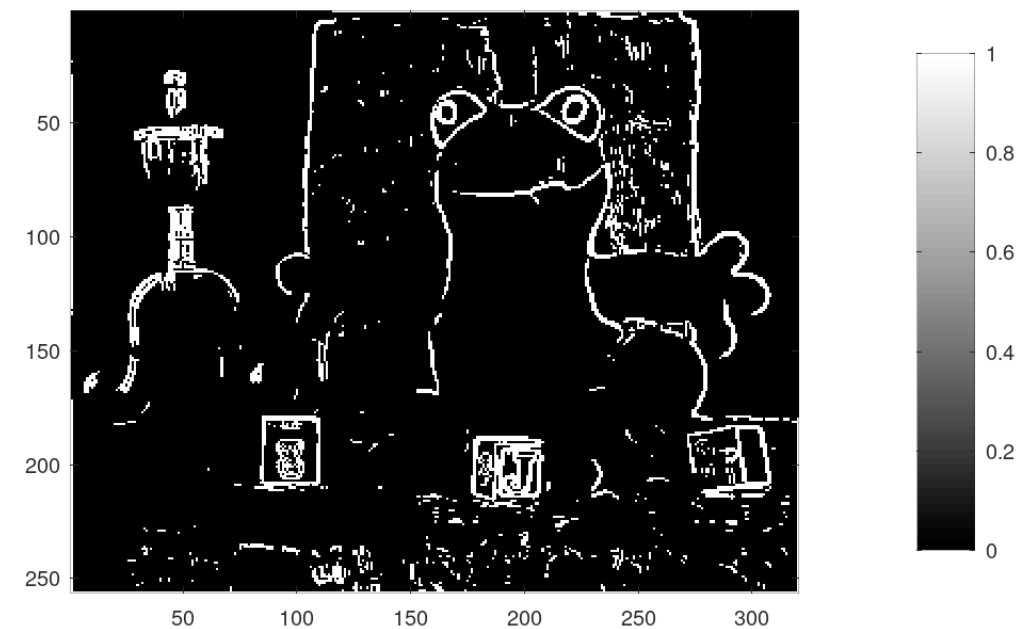
Prewitt horizontal



Prewitt combiné



Prewitt seuillé  
min=0|max=2.9089|seuil=0.659



# IV. Détection de contours

## 3- FILTRES DE SOBEL

- Filtre horizontal  $S_h$  :

|    |    |    |    |                                                  |
|----|----|----|----|--------------------------------------------------|
|    | 1  | 2  | 1  |                                                  |
| 1  | 1  | 2  | 1  |                                                  |
| 0  | 0  | 0  | 0  | : variation des lignes mais avec une plus grande |
| -1 | -1 | -2 | -1 | pondération de la colonne centrale               |

- Filtre vertical  $S_v$  :

|   |    |   |   |                                                    |
|---|----|---|---|----------------------------------------------------|
|   | -1 | 0 | 1 |                                                    |
| 1 | -1 | 0 | 1 |                                                    |
| 2 | -2 | 0 | 2 | : variation des colonnes mais avec une plus grande |
| 1 | -1 | 0 | 1 | pondération de la ligne centrale                   |

- sous MATLAB/OCTAVE :   
`>>  $S_h = fspecial('sobel');$`  ;  
`>>  $S_v = -S_h'$`  ;

# IV. Détection de contours

- **EXERCICE 1** : Comparer les effets respectifs des paires de 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.

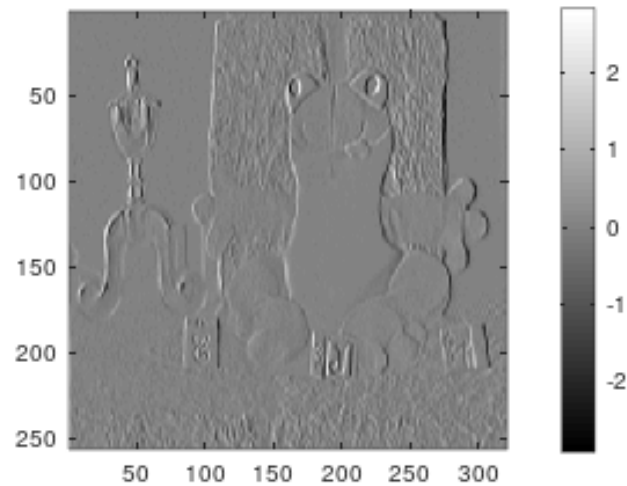
(Pouvait-on prévoir le seuil de l'image filtrée par les filtres de Sobel en considérant la dynamique sur les deux images de contours uniques ?...)

# IV. Détection de contours

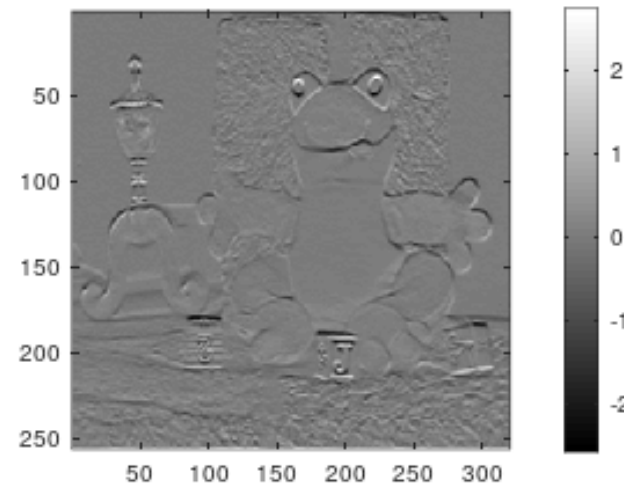
```
1 clear
2 close all
3 %pkg load image %Octave (pas Matlab)
4
5 % image de la grenouille
6 img='/Users/marcel/Documents/MATLAB/GBM/0-images/frog.jpg';
7 frog=imread(img);
8 I=rgb2gray(frog);
9 I=double(I)/255.;
10
11 % filtre de Prewitt
12 Ph=fspecial('prewitt');
13 Pv=-Ph';
14 IPh=filter2(Ph,I);
15 IPv=filter2(Pv,I);
16 IPvh=sqrt(IPv.^2+IPh.^2);
17
18 figure(1)
19 colormap('gray')
20 subplot(2,3,1), imagesc(IPv), colorbar, title('Prewitt vertical')
21 axis('square')
22 subplot(2,3,2), imagesc(IPh), colorbar, title('Prewitt horizontal')
23 axis('square')
24 subplot(2,3,3), imagesc(IPvh),colorbar, title('Prewitt complet')
25 axis('square')
26
27 % filtre de Sobel
28 Sh=fspecial('sobel');
29 Sv=-Sh';
30 ISh=filter2(Sh,I);
31 ISv=filter2(Sv,I);
32 ISvh=sqrt(ISv.^2+ISh.^2);
33
34 subplot(2,3,4), imagesc(ISv), colorbar, title('Sobel vertical')
35 axis('square')
36 subplot(2,3,5), imagesc(ISh), colorbar, title('Sobel horizontal')
37 axis('square')
38 subplot(2,3,6), imagesc(ISvh),colorbar, title('Sobel complet')
39 axis('square')
```

# IV. Détection de contours

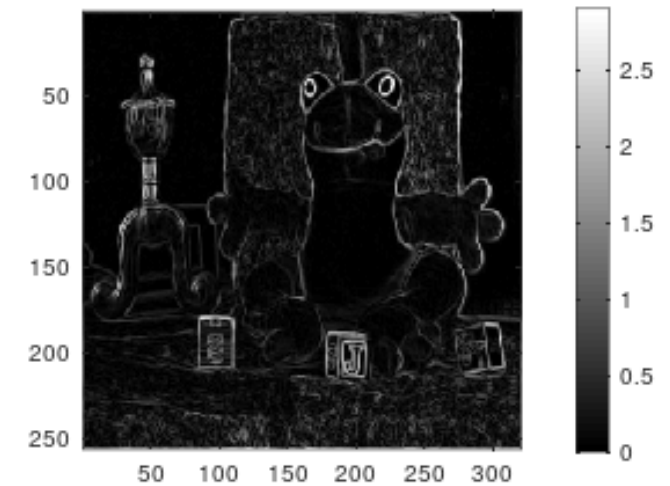
Prewitt vertical



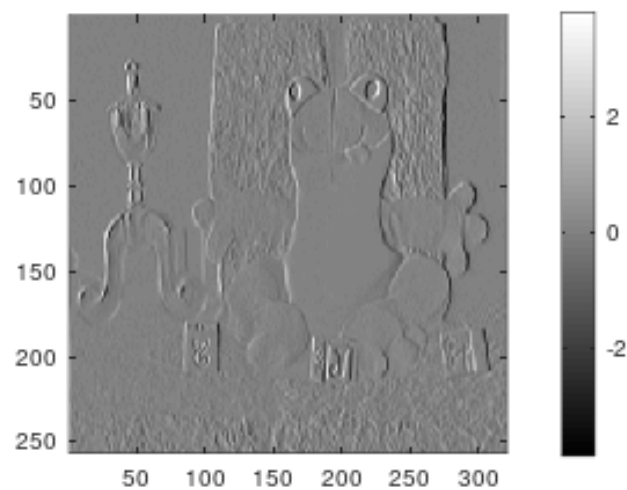
Prewitt horizontal



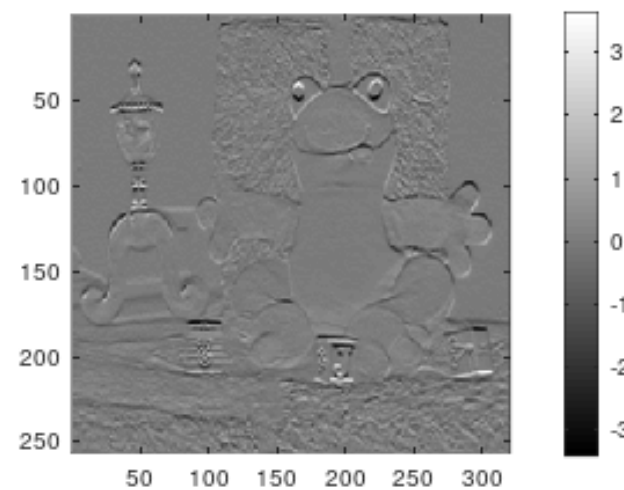
Prewitt complet



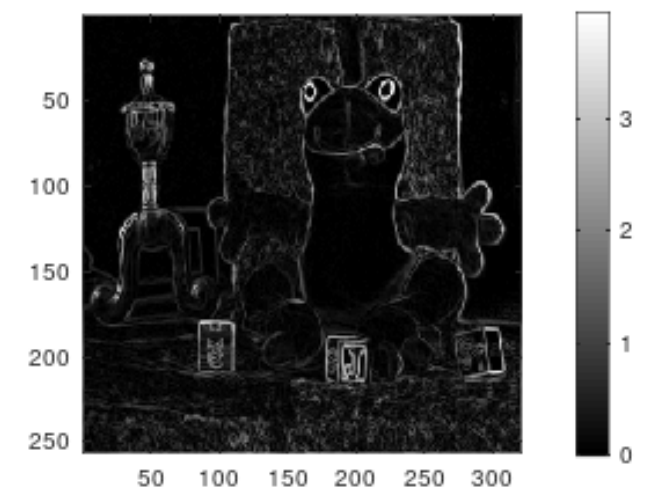
Sobel vertical



Sobel horizontal



Sobel complet

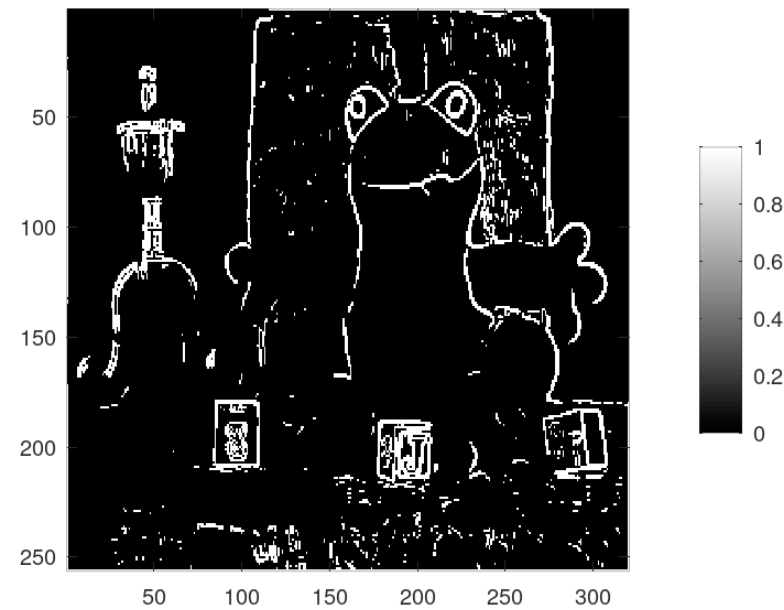


# IV. Détection de contours

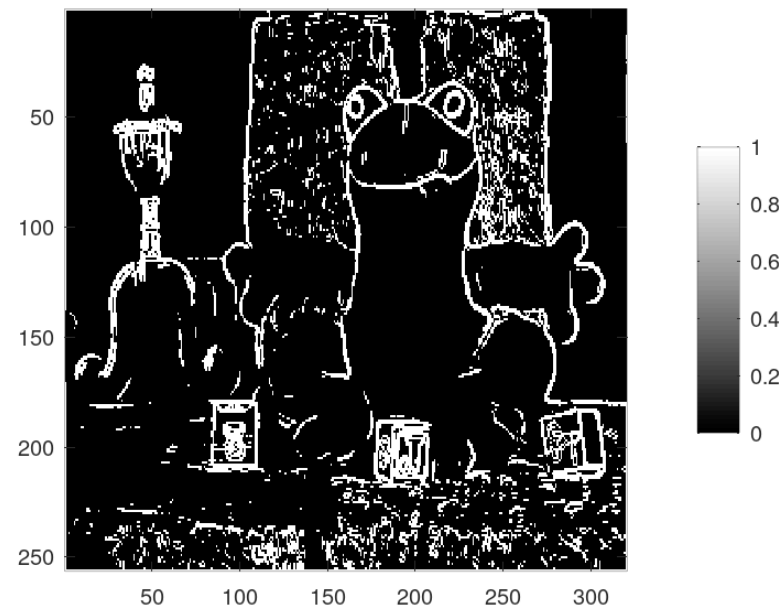
```
43 % seuillage Prewitt et comparaison avec ajustement seuil Sobel
44 seuil=0.659;
45 'seuil=', seuil
46 IPs=(IPvh>seuil);
47 ISs=(ISvh>seuil);
48
49 dynamique_ISvh = max(max(ISvh))-min(min(ISvh))
50 dynamique_IPvh = max(max(IPvh))-min(min(IPvh))
51 coeff = dynamique_ISvh/dynamique_IPvh
52 'rapport des dynamiques = ', coeff
53 seuil_Sobel=coeff*seuil;
54 'seuil adapté=', seuil_Sobel
55 ISs2=(ISvh>seuil_Sobel);
56
57 figure(2)
58 colormap('gray')
59 subplot(1,3,1), imagesc(IPs), colorbar, axis('square')
60 title({'Prewitt seuillé';['seuil=',num2str(seuil)]})
61
62 subplot(1,3,2), imagesc(ISs), colorbar, axis('square')
63 title({'Sobel avec même seuil';['seuil=',num2str(seuil)]})
64
65 subplot(1,3,3), imagesc(ISs2), colorbar, axis('square')
66 title({'Sobel avec seuil adapté';['seuil=',num2str(seuil_Sobel)]})
```

# IV. Détection de contours

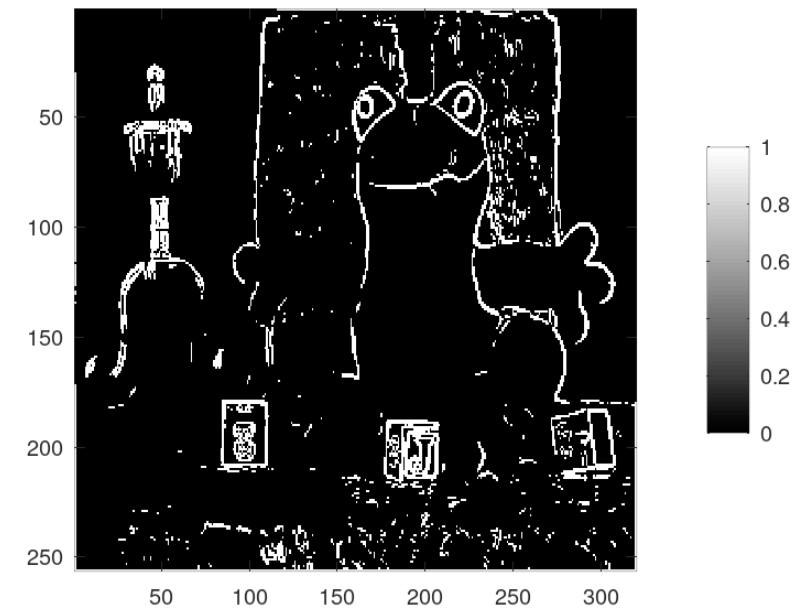
Prewitt seuillé  
seuil=0.659



Sobel avec même seuil  
seuil=0.659



Sobel avec seuil adapté  
seuil=0.89297



# IV. Détection de contours

## 4- FILTRES DE ROBERTS

- Filtre diagonal a :

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} : \text{variation des anti-diagonales}$$

- Filtre anti-diagonal b :

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} : \text{variation des diagonales}$$

- 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 :
  - $\text{seuil}_{\text{Sobel}} \approx 4/3 \text{ seuil}_{\text{Prewitt}}$ ,
  - $\text{seuil}_{\text{Roberts}} \approx 1/3 \text{ seuil}_{\text{Prewitt}}$  ?

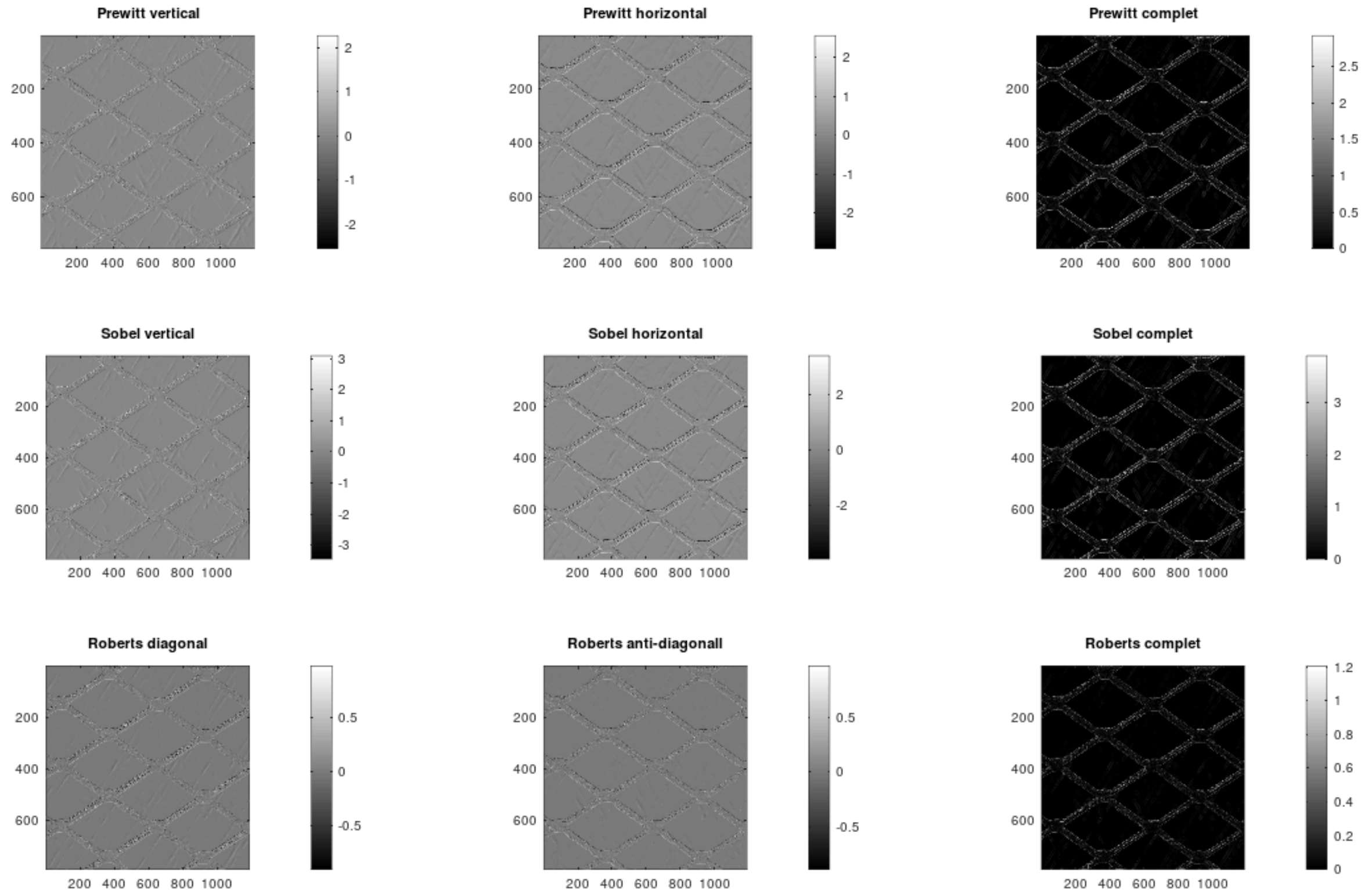
(Cette fois-ci avec, par exemple, l'image 'grillage.jpeg'...)

# IV. Détection de contours

```
1 clear
2 close all
3 pkg load image
4
5 % image de la grenouille
6 img='/Users/marcel/Documents/MATLAB/GBM/0-images/grillage.jpeg';
7 frog=imread(img);
8 I=rgb2gray(frog);
9 I=double(I)/255.;
10
11 % filtre de Prewitt
12 Ph=fspecial('prewitt');
13 Pv=-Ph';
14 IPh=filter2(Ph,I);
15 IPv=filter2(Pv,I);
16 IPvh=sqrt(IPv.^2+IPh.^2);
17
18 figure(1)
19 colormap('gray')
20 subplot(3,3,1), imagesc(IPv), colorbar, title('Prewitt vertical')
21 axis('square')
22 subplot(3,3,2), imagesc(IPh), colorbar, title('Prewitt horizontal')
23 axis('square')
24 subplot(3,3,3), imagesc(IPvh), colorbar, title('Prewitt complet')
25 axis('square')
26
27 % filtre de Sobel
28 Sh=fspecial('sobel');
29 Sv=-Sh';
30 ISh=filter2(Sh,I);
31 ISv=filter2(Sv,I);
32 ISvh=sqrt(ISv.^2+ISh.^2);
```

```
34 subplot(3,3,4), imagesc(ISv), colorbar, title('Sobel vertical')
35 axis('square')
36 subplot(3,3,5), imagesc(ISh), colorbar, title('Sobel horizontal')
37 axis('square')
38 subplot(3,3,6), imagesc(ISvh), colorbar, title('Sobel complet')
39 axis('square')
40
41 % filtres de Roberts
42 Ra=[1 0;0 -1];
43 Rb=[0 1;-1 0];
44 IRa=filter2(Ra,I);
45 IRb=filter2(Rb,I);
46 IRab=sqrt(IRa.^2+IRb.^2);
47
48 subplot(3,3,7), imagesc(IRa), colorbar, title('Roberts diagonal')
49 axis('square')
50 subplot(3,3,8), imagesc(IRb), colorbar, title('Roberts anti-diagonal')
51 axis('square')
52 subplot(3,3,9), imagesc(IRab), colorbar, title('Roberts complet')
53 axis('square')
```

# IV. Détection de contours

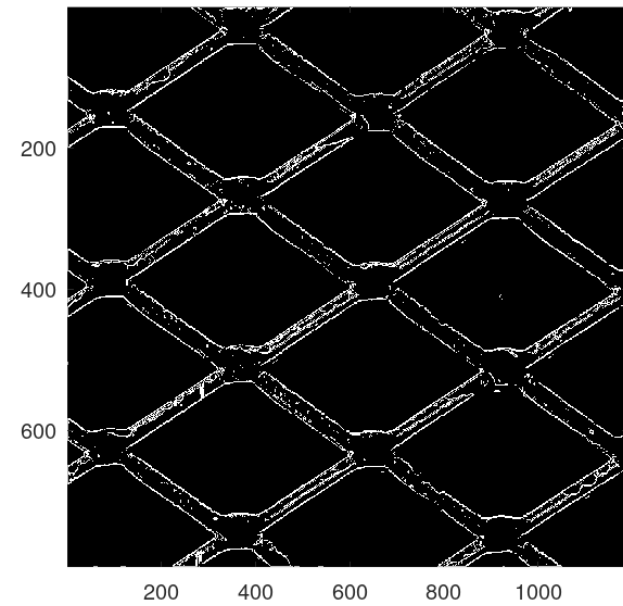


# IV. Détection de contours

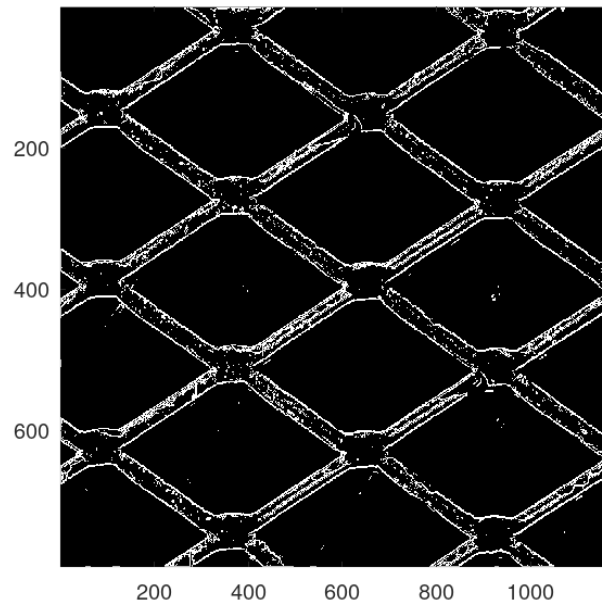
```
55 % seuillages/comparaison
56 seuil=0.7;
57 IPs=(IPvh>seuil);
58 'seuil Prewitt=', seuil
59
60 ISs=(ISvh>seuil);
61 dyn_Prewitt=max(max(IPvh))-min(min(IPvh));
62 dyn_Sobel=max(max(ISvh))-min(min(ISvh));
63 dyndyn=dyn_Sobel/dyn_Prewitt;
64 'dynamique Sobel/dynamique Prewitt=', dyndyn
65 seuil2=seuil*dyndyn;
66 'seuil Sobel=', seuil2
67 ISs2=(ISvh>seuil2);
68
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 seuil3=seuil*dyndyn;
74 'seuil Roberts=', seuil3
75 IRs2=(IRab>seuil3);
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 seuil=', num2str(seuil)]), axis('square')
82 subplot(2,3,3), imagesc(IRs), colorbar
83     title(['Roberts avec seuil=', num2str(seuil)]), axis('square')
84 subplot(2,3,5), imagesc(ISs2), colorbar
85     title(['Sobel avec seuil=', num2str(seuil2)]), axis('square')
86 subplot(2,3,6), imagesc(IRs2), colorbar
87     title(['Roberts avec seuil adapté', num2str(seuil3)]), axis('square')
```

# IV. Détection de contours

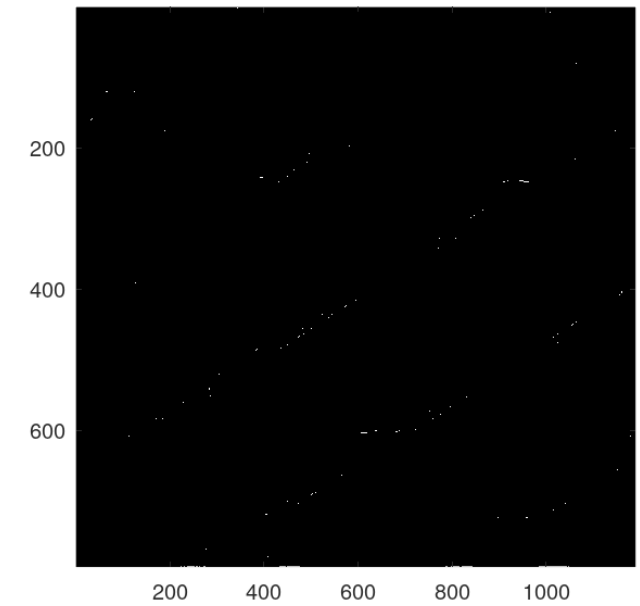
Prewitt avec seuil=0.7



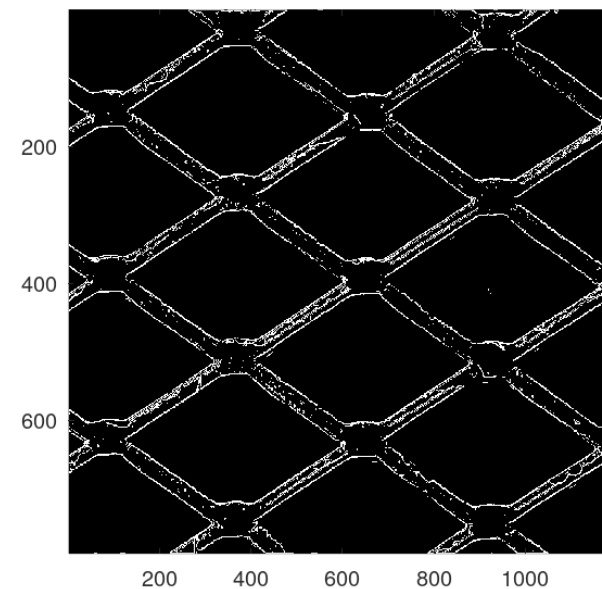
Sobel avec seuil=0.7



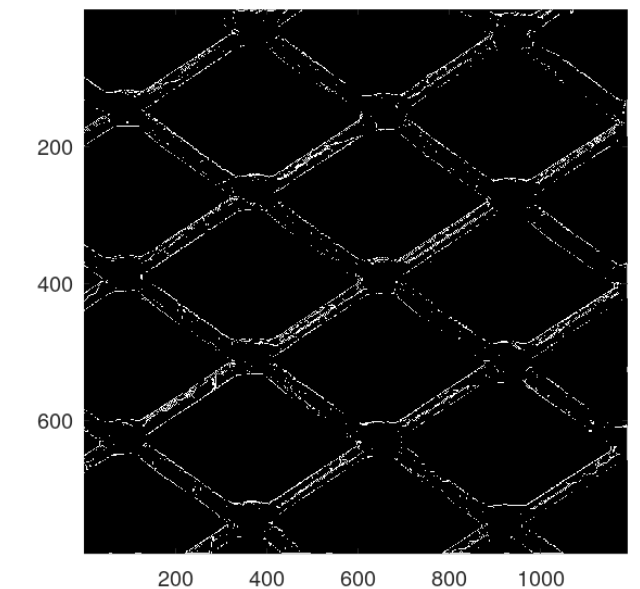
Roberts avec seuil=0.7



Sobel avec seuil=0.93576



Roberts avec seuil adapté 0.28865



```
>> 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 :

$$\begin{aligned}\sum_{|\text{coeff. de Prewitt}|} &= 6 ; \sum_{|\text{coeff. de Roberts}|} = 2 \\ \Rightarrow \text{seuil}_{\text{Roberts}} &= \frac{1}{3} \text{seuil}_{\text{Prewitt}}\end{aligned}$$

Et :

$$\begin{aligned}\sum_{|\text{coeff. de Prewitt}|} &= 6 ; \sum_{|\text{coeff. de Sobel}|} = 8 \\ \Rightarrow \text{seuil}_{\text{Sobel}} &= \frac{4}{3} \text{seuil}_{\text{Prewitt}}\end{aligned}$$

# 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, par exemple, 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 (1ère routine) ; (b) en déduire les images de contour binaires pour un nombre fixé de fausses alarmes (2ème routine) ; (c) compléter la 2ème routine par le calcul de la probabilité de détection pour chacun des 3 cas de paire de filtres.)**

**(Discuter la taille des images filtrées « valides » avec Prewitt ou Sobel d'une part et Roberts d'autre part...)**