TP VISIM : comptage de cellules

LEGENDRE Maxime
MAZERIES Jérôme

5 novembre 2010

Comptage de cellules à partir d'une image binarisée

Procédure

Dans un premier temps nous avons décidé d'égaliser l'histogramme de l'image pour en améliorer le contraste.

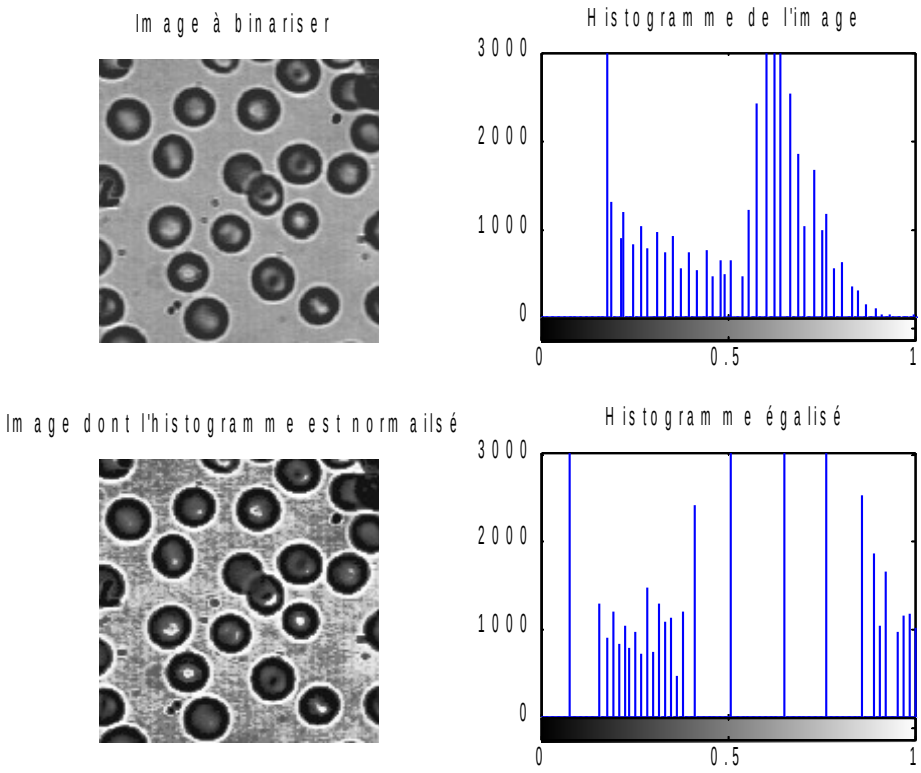

Figure 1: Amélioration du contraste de l'image

Dans la suite nous travaillerons sur l'image originale.

Les cellules présentent un niveau de gris différent du fond et peuvent donc être isolées par un seuillage de l'histogramme à une valeur de 0,5. Il nous faut ensuite inverser l'image obtenue pour que les cellules soient blanche et non noires et ainsi de pouvoir utiliser d'autres fonctions matlab.

L'image résultant présente des parasites que l'on va supprimer en effectuant une érosion suffisante en faisant attention de ne pas couper les cellules.

Enfin, on supprime les composantes connexes présentes sur le bord et on compte les composantes connexes restantes dont la taille est supérieure à un seuil fixé (ici 10).

Dans le cas de cellules imbriquées, on détermine le nombre moyen de cellules imbriquées par le

rapport de la surface de la composante connexe par la surface moyenne d'une cellule.

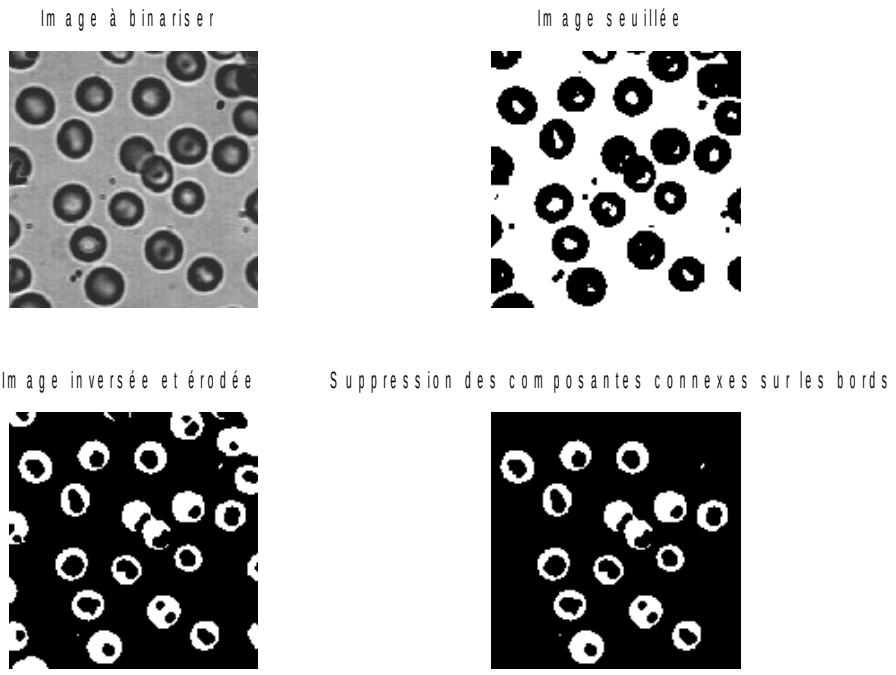

Figure 2: Procédure de la binarisation

Cette méthode détermine un nombre de cellule de 15 ce qui s'avère exacte.

Code source correspondant

```
clc
close all

load cellule

% Seuil surfacique des éléments connexes
t = 10;

figure
% Image originale
subplot(2,2,1);
imshow(I);
title('Image à binariser');

% Histogramme de l'image
subplot(2,2,2);
imhist(I);
title('Histogramme de l''image');

% égalisation de l'histogramme
subplot(2,2,3);
```

```
J = histeq(I);
imshow(J);
title('Image dont l''histogramme est normailsé');
subplot(2,2,4);
imhist(J);
title('Histogramme égalisé');

% Seuillage de l'image
figure
subplot(2,2,1);
imshow(I);
title('Image à binariser');
Is = im2bw(I,0.5);
subplot(2,2,2);
imshow(Is);
title('Image seuillée');

%
invIs = imcomplement(Is);
es = strel('disk',4);
erode = imerode(invIs,es);
subplot(2,2,3);
imshow(erode);
title('Image inversée et érodée');
clean = imclearborder(erode);
subplot(2,2,4);
imshow(clean);
title('Suppression des composantes connexes sur les bords');

[L, nbcellules] = bwlabel(clean);

surface = zeros(1,nbcellules);
for i = 1:length(L)
    for j = 1:length(L)
        if(L(i,j)~=0)
            surface(1,L(i,j)) = surface(1,L(i,j))+1;
        end
    end
end

n = 0;
surface_moy = mean(surface);

for i = 1:nbcellules
    if(surface(1,i)>t)
        if(surface(1,i)> 1.8*surface_moy)
            n= n+round(surface(1,i)/surface_moy);
        else
            n = n +1;
        end
    end
end
n
```

Limites de l'approche

Les cellules présentes sur le bord de l'image ne sont pas comptées de plus l'érosion peut entrainer la division d'une cellule en 2 composantes connexes différentes. La méthode permettant de compter les cellules imbriquées est peu robuste puisqu'elle ne permettrait pas de déterminer deux cellules quasiment coïncidentes.

Comptage des cellules par transformée de Hough des contours

Etape 1 : extraction des contours de l'image de référence

On calcul le gradient de l'image avec l'opérateur de Sobel que l'on seuil pour obtenir un contour.

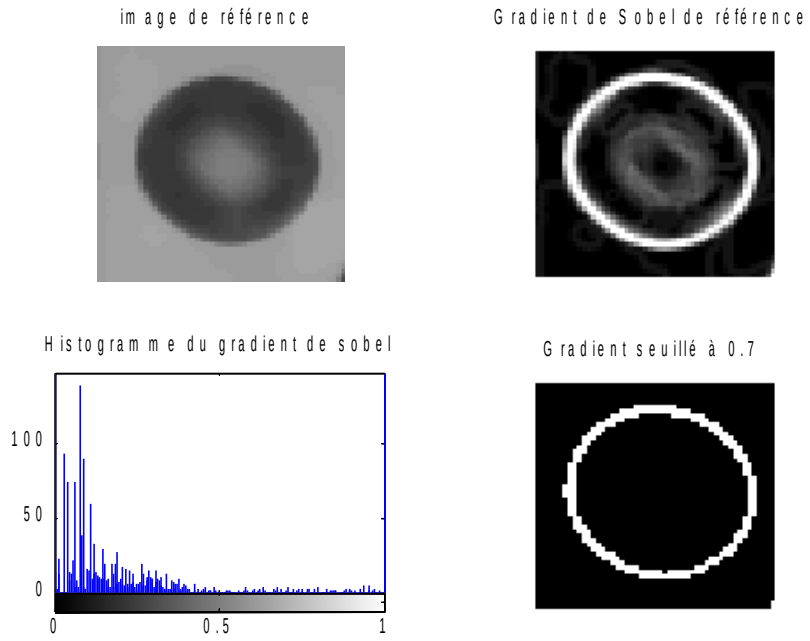

Figure 3: Extraction des contours de l'image de référence par Sobel

On remarque que l'histogramme ne nous permet pas, ici, de déterminer une valeur de seuil aisément.

Cette méthode ne permet pas d'obtenir un contour d'un pixel d'épaisseur. Cela est obtenu avec Canny.

Les valeurs des seuils pour la méthode de Canny sont automatiques mais matlab ne permet pas d'obtenir des seuils utilisables. Ainsi changer la valeur des seuils permet d'améliorer la binarisation.

Nous avons déparasité l'image du contour binarisée de Canny pour ne pas laisser apparaître quelques pixels n'appartenant pas au contours mais toujours présents.

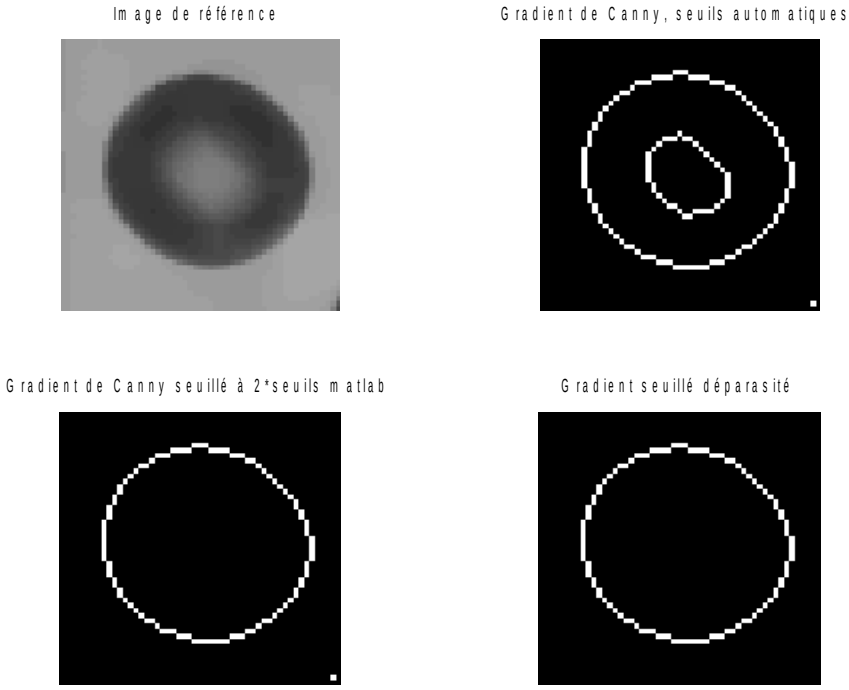

Figure 4: Extraction du contour de référence par la méthode de Canny

Etape 2 : Extraction des points de contours de l'image à traiter

On obtient les contours de l'image à traiter par la même méthode que pour l'image de référence :

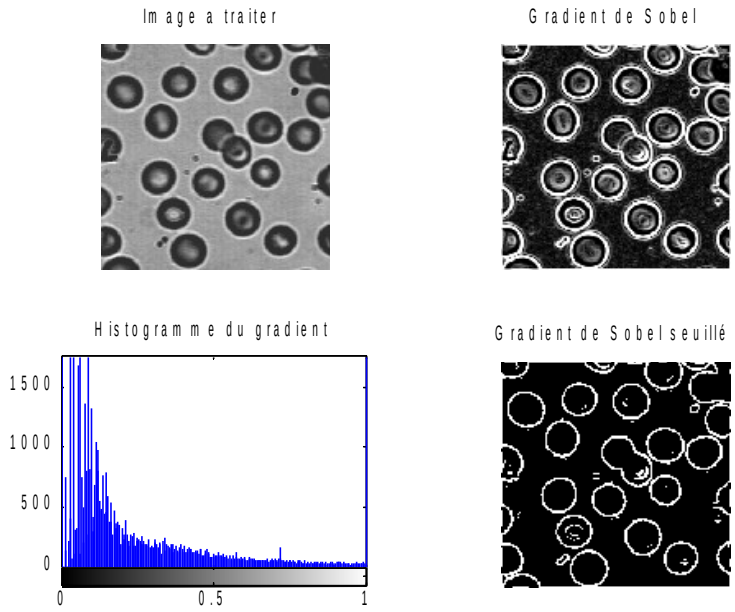

Figure 5: Extraction de contour sur l'image à traiter - Par Sobel

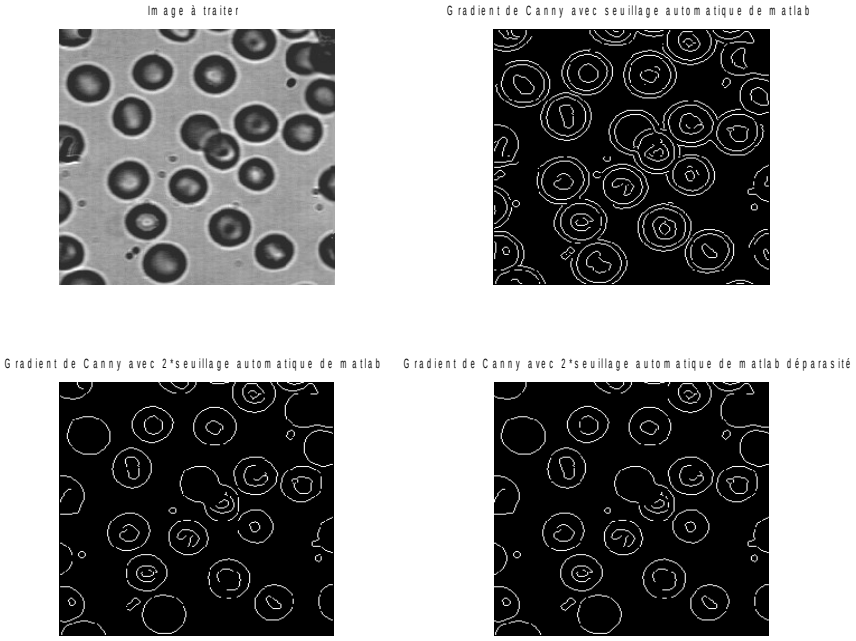

Figure 6: Extraction des contours de l'image à traiter - Par Canny

Les contours obtenus par la méthode de Canny sont plus exploitables car ils ne font qu'un pixel d'épaisseur, alors que ceux obtenus par Sobel sont moins bien définis spatialement.

Etape 3 : Transformée de Hough

On va effectuer, pour chaque pixel de l'image contenant les contours, un ET logique entre le motif de référence (contours de l'image de référence) et le voisinage de ce pixel. Cela permet d'obtenir une image de la corrélation entre l'image de référence et l'image à traiter.

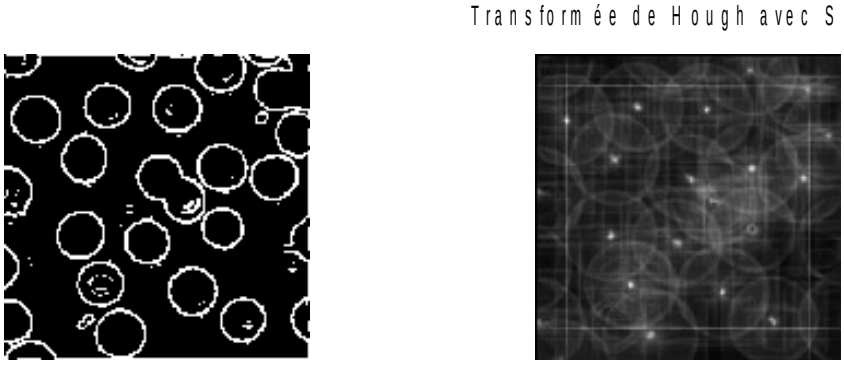

Figure 7: Resultat de la transformée de Hough sur l'image

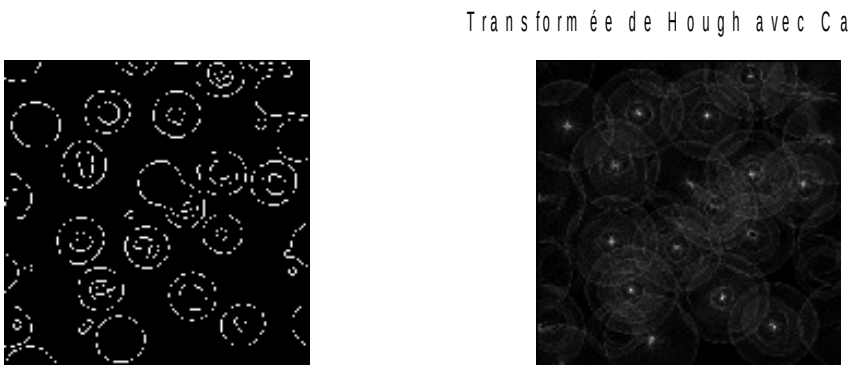

Figure 8: Resultat de la transformée de Hough sur l'image obtenue par Canny

nsformée de Hou par Sobel fait apparaître le reflet des

ment de l'ima ansformée de Hough

On commence par traiter l'image obtenue par Sobel.

Une fois que l'on a le résultat de la transformée de Hough, on applique un seuillage dont la valeur a été déterminée manuellement (0,55). La dernière étape consiste en une dilatation de l'image afin de regrouper les pixels très proches les uns des autres (qui correspondent à une seule cellule).

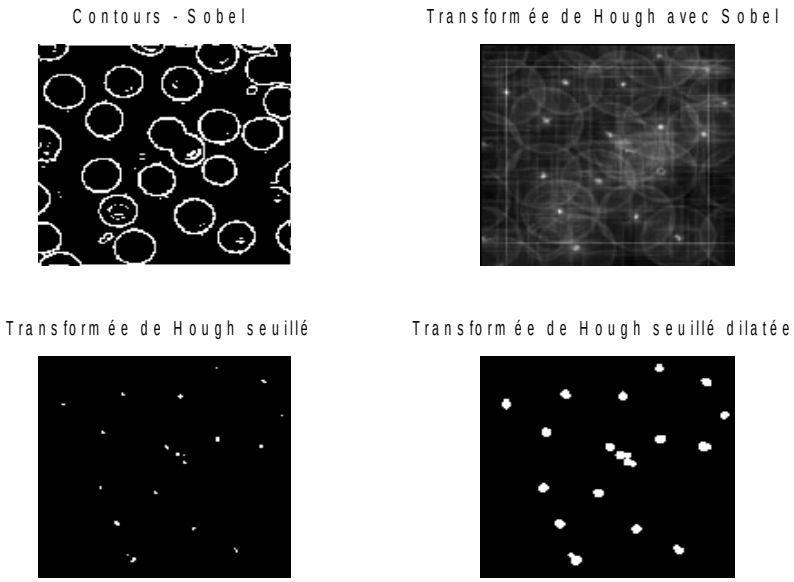

Figure 9: Procédure de traitement des contours obtenus par Sobel

Enfin il suffit de compter le nombre de composantes connexes pour avoir le nombre de cellules : ici on trouve 17.

On effectue les mêmes opérations sur le résultat de la transformée de Hough obtenue à partir de Canny :

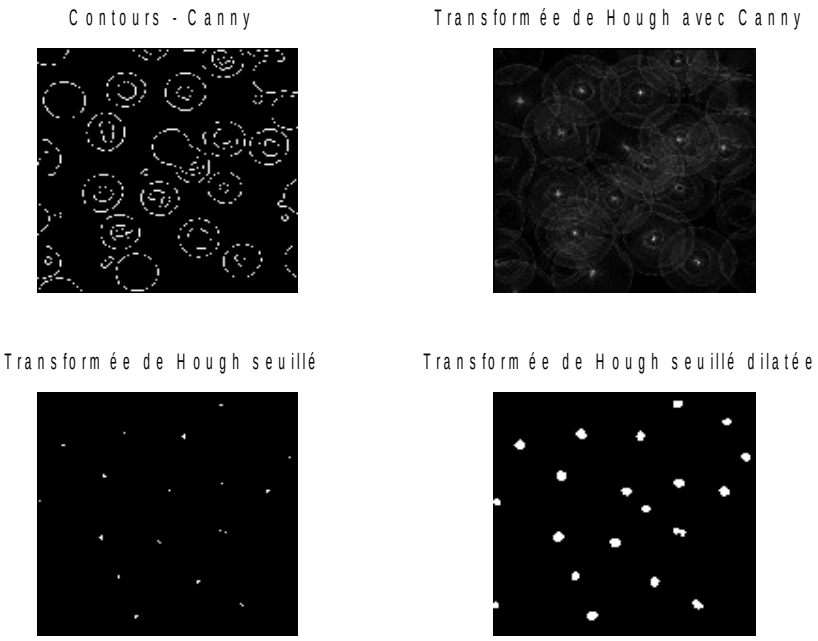

Figure 10: Procédure de traitement des contours obtenus par Canny

Le résultat est similaire. Cette méthode trouve 20 cellules.

Conclusion

La transformée de Hough permet de remédier à deux problèmes rencontrés précédemment :

- Les cellules présentes sur le bord de l'image sont maintenant comptées (en partie).
- Cette méthode permet de distinguer des cellules superposées.

Certaines cellules sont trouvées par la méthode de Canny et pas par celle de Sobel.

Code source

Fonction de Hough

```
function corr2 = perso_hough(image,image_ref)
% corr est une image de taille superieure a l'image a traiter, qui va
% contenir le resultat de la transformee de Hough
corr = zeros(length(image)+length(image_ref),length(image)+length(image_ref));
% image 2 est une image de la meme taille que corr, contenant l'image a
% traiter en son centre, et la valeur 0 autour
image2 = zeros(length(image)+length(image_ref),length(image)+length(image_ref));
image2(floor(length(image_ref)/2):length(image)+floor(length(image_ref)/2)-
1,floor(length(image_ref)/2):length(image)+floor(length(image_ref)/2)-1) = image;

    for i=floor(length(image_ref)/2)+1:length(image2)-floor(length(image_ref)/2)-1
        for j=floor(length(image_ref)/2)+1:length(image2)-floor(length(image_ref)/2)-1
            % Pour chaque pixel de la partie centrale d'image2
            departi = i-floor(length(image_ref)/2);
            arrivei = departi + length(image_ref)-1;
            departj = j-floor(length(image_ref)/2);
            arrivej = departj + length(image_ref)-1;
            % On effectue un ET logique entre le voisinage du pixel et le
            % contour de référence
            Patch = image2(departi:arrivei,departj:arrivej) & image_ref;
            % La valeur de corr pour ce pixel est la somme des pixels
            % corrélés
            corr(i,j)=sum(sum(Patch));
        end
    end
 % On extrait de corr la partie centrale pour obtenir une image de la
 % taille de l'image a traiter
 corr2 = corr(floor(length(image_ref)/2):length(image)+floor(length(image_ref)/2)-
1,floor(length(image_ref)/2):length(image)+floor(length(image_ref)/2)-1);
end
```

Script principal

```
clc
close all

% Chargement de l'image de référence
load imref
load cellule

figure
subplot(2,2,1);
imshow(J);
title('image de référence');

% Methode 1 : détéction par calcul de gradient et seuillage

% Operateurs de Sobel

hi = [1 2 1 ; 0 0 0 ; -1 -2 -1];
hj = [1 0 -1 ; 2 0 -2 ; 1 0 -1];

grad = sqrt(filter2(hi,J).^2 + filter2(hj,J).^2);

subplot(2,2,2);
imshow(grad);
title('Gradient de Sobel de référence');

subplot(2,2,3);
imhist(grad)
```

```
title('Histogramme du gradient de sobel');

% Seuillage de l'image
grads = im2bw(grad,0.7);
subplot(2,2,4);
imshow(grads);
title('Gradient seuillé à 0.7');


% Méthode 2 : gradient par Canny

[canny,seuils] = edge(J,'canny');
canny2 = edge(J,'canny',seuils*2);

figure

subplot(2,2,1);
imshow(J);
title('Image de référence');

subplot(2,2,2);
imshow(canny);
title('Gradient de Canny, seuils automatiques');

subplot(2,2,3);
imshow(canny2);
title('Gradient de Canny seuillé à 2*seuils matlab');

% Suppression des parasites

[L,num] = bwlabel(canny2);

surface = zeros(1,num);
for i = 1:length(L)
    for j = 1:length(L)
        if(L(i,j)~=0)
            surface(1,L(i,j)) = surface(1,L(i,j))+1;
        end
    end
end

t = 10;

parasites = [];
for i = 1:num
    if(surface(1,i)<t)
        parasites = [parasites,i];
    end
end

for i = 1:length(canny2)
    for j = 1:length(canny2)
        for k = 1:length(parasites)
            if L(i,j) == parasites(k)
                canny2(i,j)=0;
            end
        end
    end
end

subplot(2,2,4);
imshow(canny2);
title('Gradient seuillé déparasité');

%% Pareil pour I
figure
```

```
subplot(2,2,1);
imshow(I);
title('Image a traiter');

% Methode 1 : détéction par calcul de gradient et seuillage

% Operateurs de Sobel

hi = [1 2 1 ; 0 0 0 ; -1 -2 -1];
hj = [1 0 -1 ; 2 0 -2 ; 1 0 -1];

gradI = sqrt(filter2(hi,I).^2 + filter2(hj,I).^2);

subplot(2,2,2);
imshow(gradI);
title('Gradient de Sobel');

subplot(2,2,3);
imhist(gradI)
title('Histogramme du gradient');

% Seuillage de l'image
gradsI = im2bw(gradI,0.9);
subplot(2,2,4);
imshow(gradsI);
title('Gradient de Sobel seuillé');


% Méthode 2 : gradient par Canny

[canny,seuils] = edge(I,'canny');
canny2I = edge(I,'canny',seuils*2);

figure

subplot(2,2,1);
imshow(I);
title('Image à traiter');

subplot(2,2,2);
imshow(canny);
title('Gradient de Canny avec seuillage automatique de matlab');

subplot(2,2,3);
imshow(canny2I);
title('Gradient de Canny avec 2*seuillage automatique de matlab');

% Suppression des parasites

[L,num] = bwlabel(canny2I);

surface = zeros(1,num);
for i = 1:length(L) % Calcul de la surface de chaque element connexe
    for j = 1:length(L)
        if(L(i,j)~=0)
            surface(1,L(i,j)) = surface(1,L(i,j))+1;
        end
    end
end

t = 10;

parasites = []; % Les elements connexes dont la taille est < 10 sont supprimés
for i = 1:num
    if(surface(1,i)<t)
        parasites = [parasites,i]; % déterminations des parasites
```

```
    end
end

for i = 1:length(canny2I)
    for j = 1:length(canny2I)
        for k = 1:length(parasites)
            if L(i,j) == parasites(k)
                canny2I(i,j)=0; % suppression des parasites trouvés
            end
        end
    end
end

subplot(2,2,4);
imshow(canny2I);
title('Gradient de Canny avec 2*seuillage automatique de matlab déparasité');
%%

% Transformée de hough avec canny
figure
subplot(2,2,1);
imshow(canny2I);
title('Contours - Canny');
H = perso_hough(canny2I,canny2);
H = H./(max(max(H)));

Hs = im2bw(H,0.4);
subplot(2,2,2);
imshow(H)
title('Transformée de Hough avec Canny');
subplot(2,2,3);
imshow(Hs)
title('Transformée de Hough seuillé');
es = strel('disk',4);
Hsd = imdilate(Hs,es);
subplot(2,2,4);
imshow(Hsd);
title('Transformée de Hough seuillé dilatée');
[LL, nombre] = bwlabel(Hsd);
disp('Canny trouve :');
nombre

%% % Transformée de hough avec Sobel
figure
subplot(2,2,1);
imshow(gradsI);
title('Contours - Sobel');
H = perso_hough(gradsI,grads);
H = H./(max(max(H)));

Hs = im2bw(H,0.55);
subplot(2,2,2);
imshow(H)
title('Transformée de Hough avec Sobel');
subplot(2,2,3);
imshow(Hs)
title('Transformée de Hough seuillé');
es = strel('disk',4);
Hsd = imdilate(Hs,es);
subplot(2,2,4);
imshow(Hsd);
title('Transformée de Hough seuillé dilatée');
[LL, nombre] = bwlabel(Hsd);
disp('Sobel trouve :');
nombre
```