DURET Guillaume GR D
LECONTE Thibault

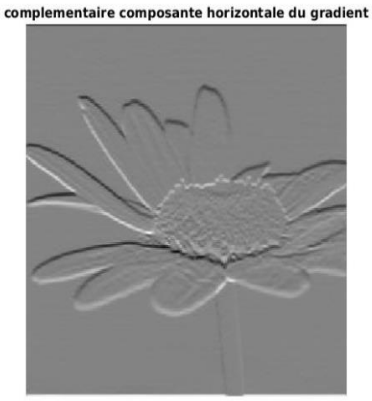
TP1 Traitement et synthèse d'image : Filtrage

2018-2019

1 Filtrage passe-haut dans l'espace direct : détection de contours

Le lissage est nécessaire car les opérateurs différentiels sont sensibles aux bruits il faut donc une partie lissage.

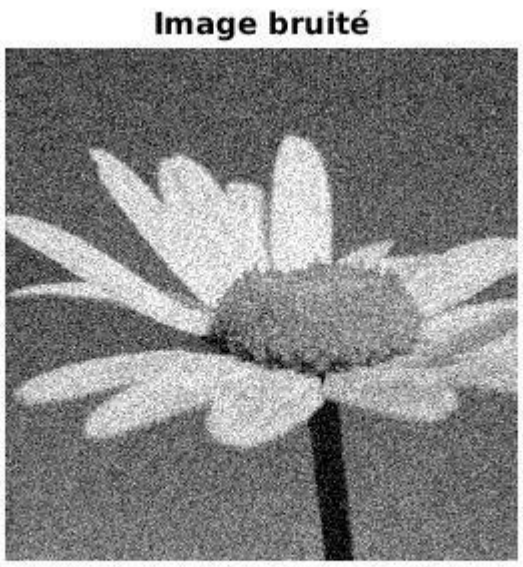
```
clear variables;
close all;

%question 1
Lissage = [-1 -2 -1];
Derivatif = [1 0 -1];

A=Lissage'*Derivatif;
B=Derivatif'*Lissage;

%question 2
Im = imread ('flower.png');
Im2 = im2double (Im);
figure(1);
imshow(Im,[]);
title 'Image origine'



h=length(Im);
w=length(Im);

%question 3
Gv=imfilter(Im2,A);
Gh=imfilter(Im2,B);

%question 4
G=sqrt(Gv.^2+Gh.^2);

figure(2);
subplot (131);
imshow(imcomplement(Gv),[]);
title 'complementaire composante verticale du gradiant'
subplot (132);
imshow(imcomplement(Gh),[]);
title 'complementaire composante horizontale du gradient'
subplot (133);
imshow(imcomplement(G),[]);
title 'complementaire de la norme du gradient'
```

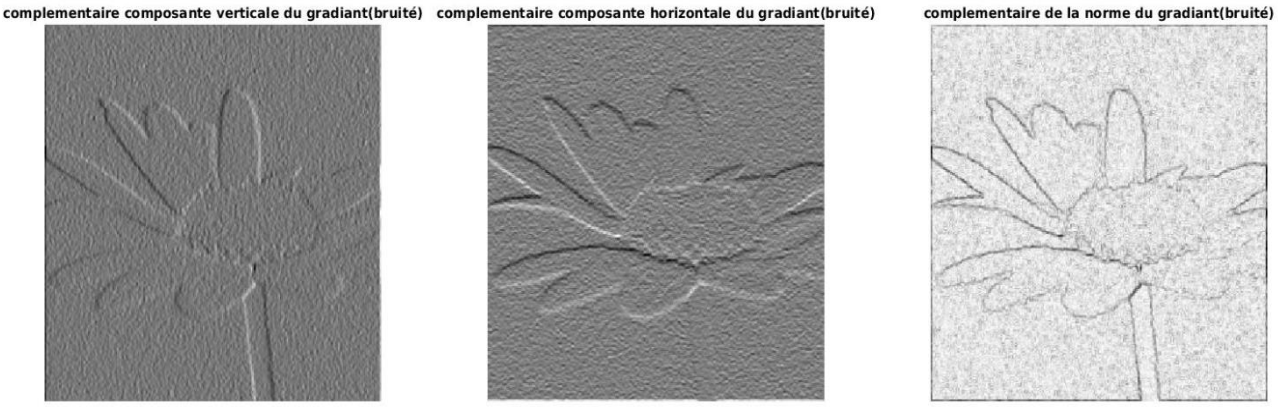


On peut remarquer que la composante horizontal (respectivement verticale) du radiant permet de relever les contours verticaux (respectivement horizontaux). Par application de la norme du radian tous les contours ressortent sur l'image,

```
%question 5
Im3 = imnoise(Im2,'gaussian',0.01);
figure(3);
imshow(Im3);
title 'Image bruitÃ©'

h2=length(Im3);
w2=length(Im3);

Gv_gauss=imfilter(Im3,A);
Gh_gauss=imfilter(Im3,B);

G_gauss=sqrt(Gv_gauss.^2+Gh_gauss.^2);
```

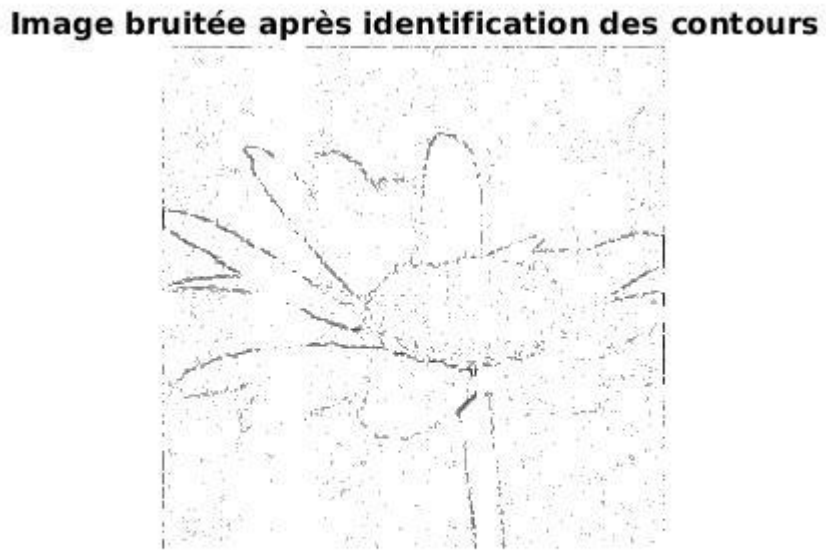
```
figure(4);
subplot (131);
imshow(imcomplement(Gv_gauss),[]);
title 'complementaire composante verticale du gradiant(bruitÃ©)'
subplot (132);
imshow(imcomplement(Gh_gauss),[]);
title 'complementaire composante horizontale du gradiant(bruitÃ©)'
subplot (133);
imshow(imcomplement(G_gauss),[]);
title 'complementaire de la norme du gradiant(bruitÃ©)'
```





On observe que l'image obtenu est beaucoup moins claire que précédemment et que seuls les contours les plus importants ressortent alors que d'autre ne sont plus perceptible. On voir très distinctement que les opérateurs différentiels et donc que ce filtre est très sensible au bruit.

```
%question 6
Gv_n=Gv_gauss./G_gauss;
Gh_n=Gh_gauss./G_gauss;

%question 7
d=2;
%Points P1
P1x=round(Gv_n*d)+meshgrid(1:256,1:256);
P1x(P1x<1)=1;
P1x(P1x>256)=255;
P1y=round(Gh_n*d)+meshgrid(1:256,1:256);
P1y(P1y<1)=1;
P1y(P1y>256)=255;
%Points P2
P2x=round(-Gv_n.*d)+meshgrid(1:256,1:256);
P2x(P2x<1)=1;
P2x(P2x>256)=255;
P2y=round(-Gh_n.*d)+meshgrid(1:256,1:256);
P2y(P2y<1)=1;
P2y(P2y>256)=255;

%question8
C=zeros(h,w);
eps=0.5;
for l=1:1:h;
    for c=1:1:w;
        if (G_gauss(l,c)-G_gauss(P1x(l,c),P1y(l,c))>eps) & (G_gauss(l,c)-
G_gauss(P2x(l,c),P2y(l,c))>eps);
            C(l,c)=G_gauss(l,c);
        end
    end
end

figure(5);
imshow(imcomplement(C),[]);
title 'Image bruitÃ©e aprÃ¨s identification des contours'
```



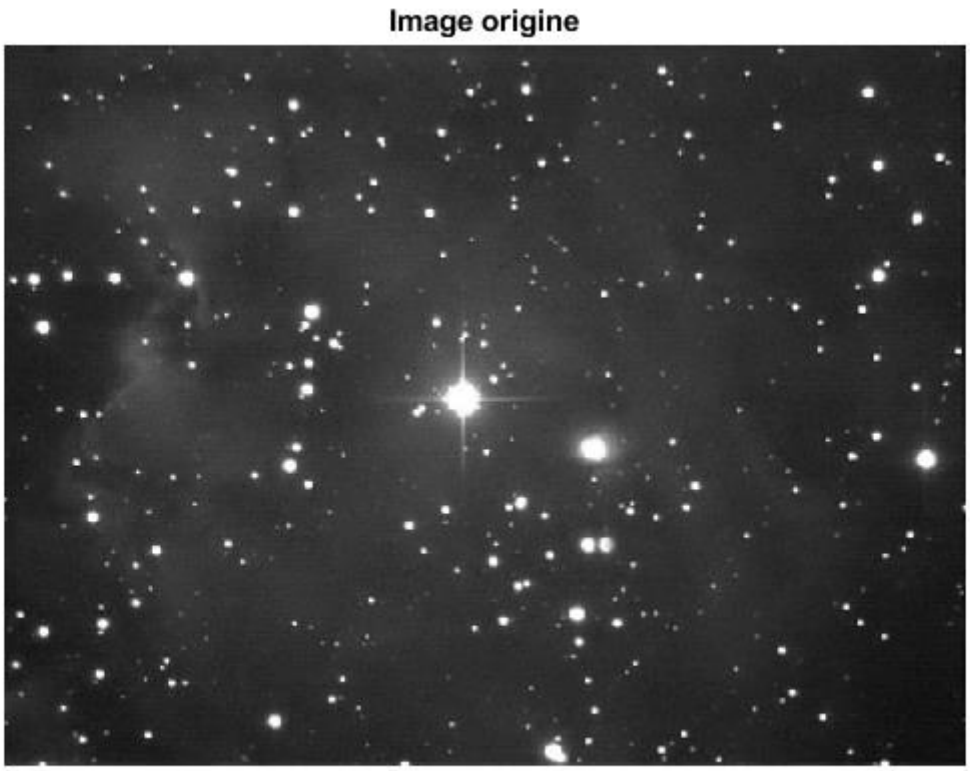
Les inégalités pour déterminer si un pixel appartient à un contour et non simplement à un bruit permettent de réduire fortement le bruit et donc augmente la qualité de détection de contour de l'image d'origine.

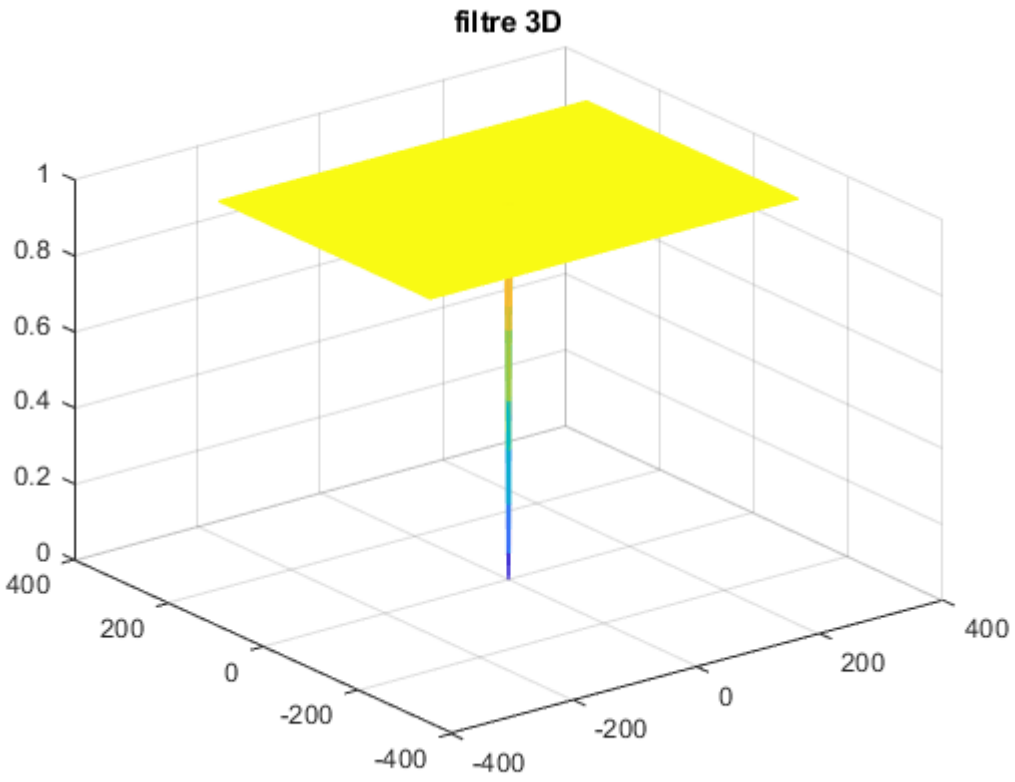
2 Filtrage passe-haut dans l'espace de Fourier

On cherche à filtrer une image pour enlever un nuage de point qui correspond à des basses fréquences pour une segmentation ultérieur. Pour cela on utilise un filtre de Butterworth :

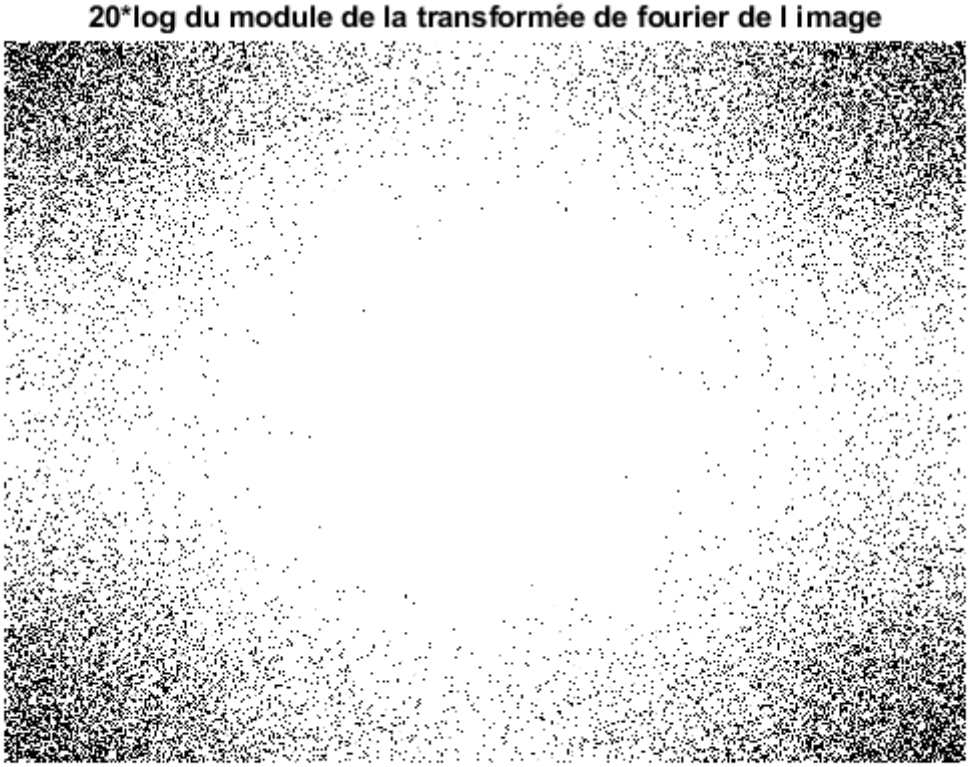
```
clear variables;
close all;

%question 1
Im = imread ('ngc2175.png');
Im2 = im2double (Im);

figure(1);
imshow(Im,[]);
title 'Image origine'

[h,w]=size(Im2);
n=2;
```

```
fc=3;

[U,V]=meshgrid(-w/2+1/2:w/2-1/2,-h/2+1/2:h/2-1/2);

D=sqrt(U.^2+V.^2);

H=1./(1+(fc./D).^(2*n));

figure(2);
mesh(U,V,H);
title 'filtre 3D'

%question 3
FFT=fft2(Im2);
FFT_shift = fftshift(FFT);
figure(3);
plot(abs(FFT_shift))
title 'module de la transphormée de fourier de l image'
```





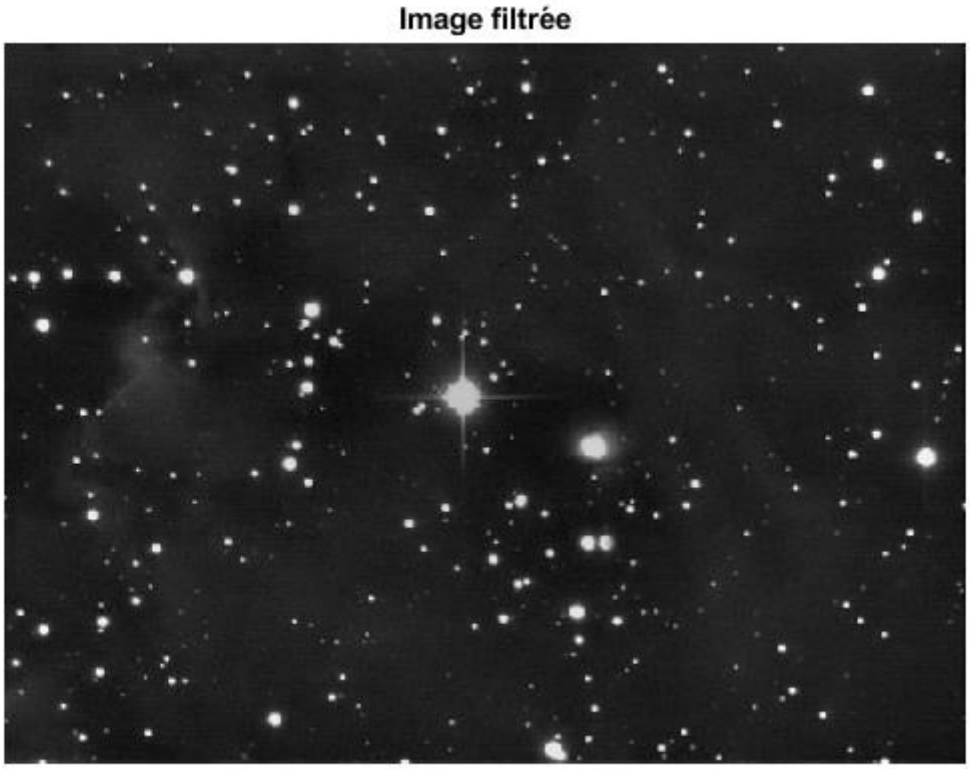
Le filtre 3D possède en fréquence de coupure 3 et on remarque bien que c'est un filtre passe-haut.



On remarque que les hautes fréquences de l'image sont plutôt situées au centre de l'image.

```
%question 4
Im_filtre_f=FFT_shift.*H;

%question 5
Im_filtre_shift=ifftshift(Im_filtre_f);
Im_filtre=ifft2(Im_filtre_shift);


figure(4);
imshow(Im_filtre,[]);
title 'Image filtrée'
```

En appliquant le filtre passe haut à l'image on obtient l'image ci-dessous :



On remarque que l'image est bien plus lisible que l'image d'origine qui possédait des nuages de point non voulu.

3 Filtrage non linéaire : filtre médian

Le filtre médian va remplacer chaque pixel par une valeur médiane des pixels voisins.

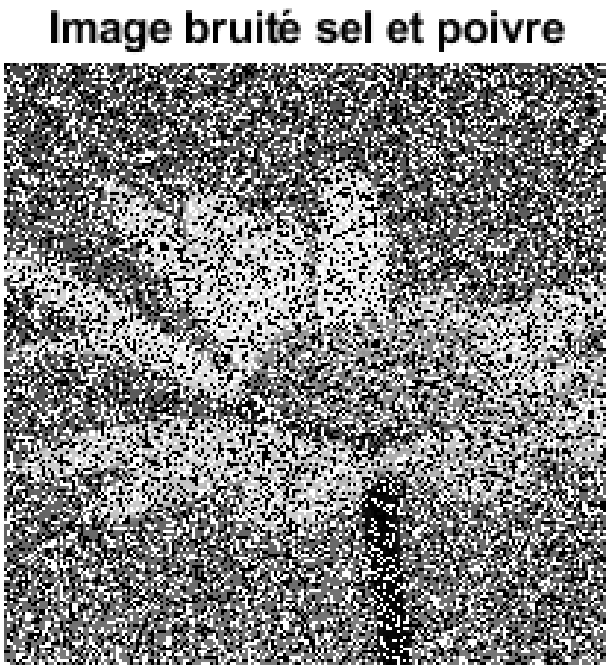
Les filtres d'ordre vont utiliser des manipulations d'ordre sur les pixels afin de modifier l'image

(ex : filtres médian, filtres moyennant...)

Un bruit sel et poivre est une dégradation de l'image sous la forme de pixels noir et blanc repartis aléatoirement.

```
clear variables;
close all;

%question 4
Im = imread ('flower.png');
Im2 = im2double (Im);

figure(1);
imshow(Im2,[]);
title 'Image origine';


%question 5

Im_bruit=imnoise(Im2,'salt & pepper',0.3);

figure(2);
imshow(Im_bruit,[]);
title 'Image bruité sel et poivre';
```

En appliquant sur l'image d'origine un bruit sel et poivre de densité on obtient l'image ci-dessous quoi est fortement dégradée.




```
%question 5

Im_bruit=imnoise(Im2,'salt & pepper',0.5);

figure(2);
imshow(Im_bruit,[]);
title 'Image bruité sel et poivre';

%question 6
Im_filt = ordfilt2(Im_bruit,5,ones(3,3));
Im_filt2 = ordfilt2(Im_bruit,13,ones(5,5));


figure(3);
imshow(Im_filt,[]);
title 'Image bruité sel et poivre filtrée avec mask 3*3';

figure(4);
imshow(Im_filt2,[]);
title 'Image bruité sel et poivre filtrée avec mask 5*5'
```

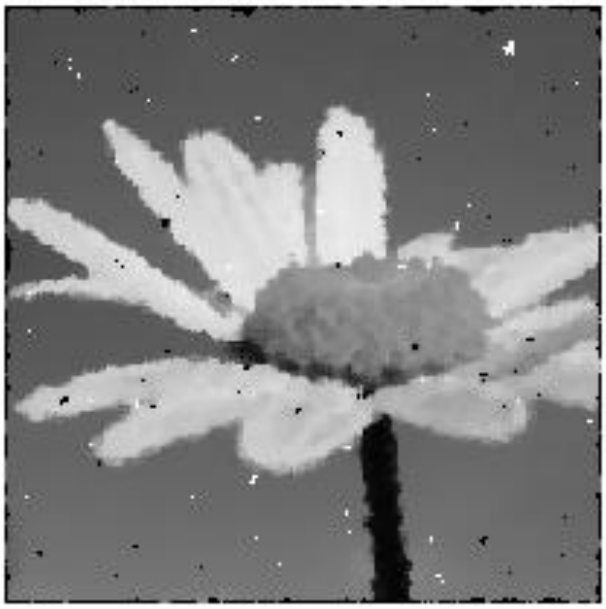
Ensuite on applique un filtre médian qui a pour but de débruiter l'image :

Image bruité sel et poivre filtrée avec mask 3*3

Avec un mask 3*3 on peut remarquer que le bruit est trop important pour rétablir complètement l'image (il y a par endroit une concentration de blanc ou noir sur 9 pixels qui fait que la médiane est en encore blanc ou noir).

En appliquant un masque plus grand on remarque que quasiment tous le bruit a disparu mais en contrepartie l'image est moins nette qu'originellement.

Image bruité sel et poivre filtrée avec mask 5*5



4 Filtrage à partir d'histogramme : seuillage par K-means

On code à l'aide de l'algorithme de K-means le moyen de segmenter une image en 2 régions.

```
clear variables;
close all;

Im = imread ('flower.png');
[h,w]=size(Im);

Im2 = im2double(Im);

figure(1);
imshow(Im,[]);
title 'Image origine'

m1 = 78;
m2 = 180;

M1 = 0;
M2 = 0;

eps = 1;

while ((abs(M1-m1)>eps) ||  (abs(M2-m2)>eps))
    Labels = zeros(h,h);
    M1 = m1;
```

```
    M2 = m2;
    L1 = 0;
    L2 = 0;
    P1 = 0;
    P2 = 0;
    for i=1:256;
        for j =1:256;
            if (abs(M1 - 255*Im2(i,j)) < abs(M2-255*Im2(i,j)));
                Labels(i,j) = 1;
                L1 = L1 + Im2(i,j)*255;
                P1 = P1 + 1;
            else
                Labels(i,j) = 2;
                L2 = L2 + Im2(i,j)*255;
                P2 = P2 +1;
            end
        end
    end
    m1 = L1/P1;
    m2 = L2/P2;


    M1 - m1;
    M2 - m2;

end

Labels = Labels-1;
figure(2);
imshow(Labels, []);
title 'image segmenté en 2 regions'
```

Nous obtenons donc bien une image segmentée en 2 régions :