



ANALYSE D'IMAGE

Rapport de TD1

Guénon Marie et Favreau Jean-Dominique
VIM / Master SSTIM

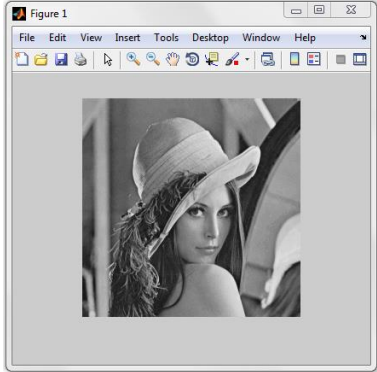
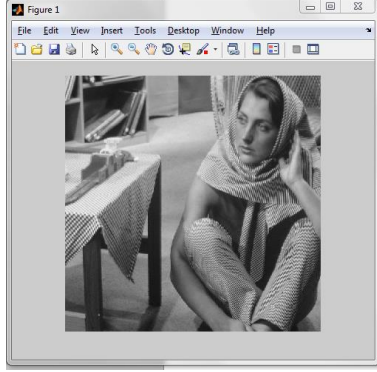
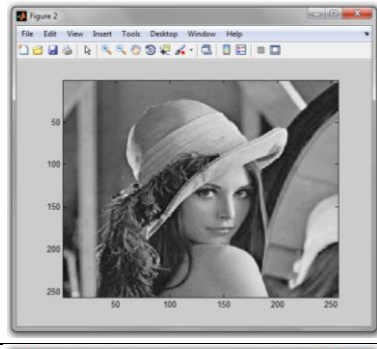
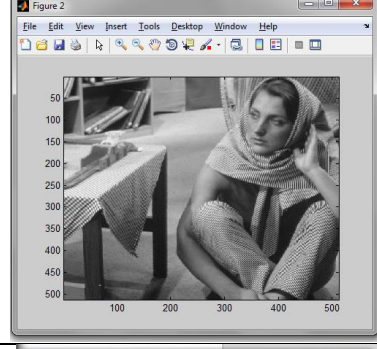
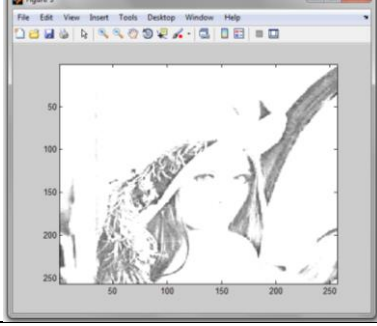
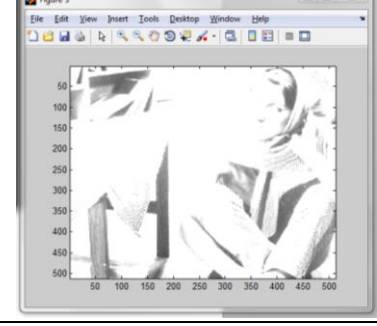
Table des matières

1 Introduction.....	2
1.2 Fonctions utiles	2
2 Histogramme	3
2.1 Transformations simples	3
2.2 Amélioration de la dynamique	4
2.3 Histogramme cumulé	5
3 Filtrage.....	7
3.2 Applications	7
Filtre	7
"randn", moyenne nulle, écart-type 20	8
"imnoise", probabilité 10%	8
4 Transformée de Fourier.....	9
4.1 Définitions	9
4.2 Filtrage fréquentiel.....	11
Annexes :	13

1 Introduction

1.2 Fonctions utiles

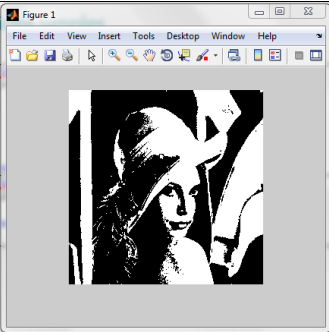
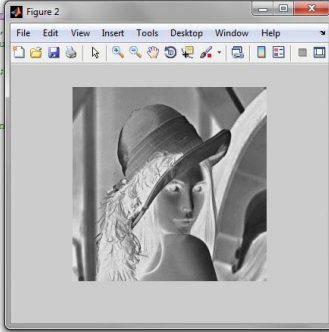
Lecture de l'image : `u=double(imread('LENA.BMP'));`
Affichage de l'image :

<pre>figure;imshow(u, [0 255]);</pre>		
<pre>figure;imagesc(u, [0 255]); colormap(gray);</pre>		
<pre>figure;image(u); colormap(gray);</pre>		

Imshow et *imagesc* ont un affichage et un rendu très semblable (mis à part qu'il faut effectuer une fonction de plus sur *imagesc* pour obtenir ce résultat). Quant à elle la fonction *image* elle a un effet de seuillage sur les images en noir et blanc.

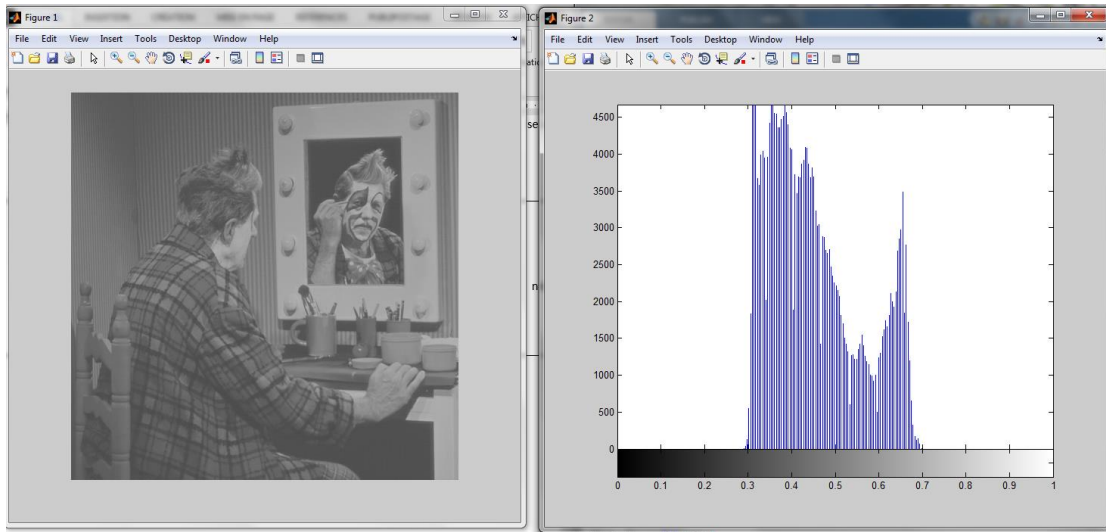
2 Histogramme

2.1 Transformations simples

Formule	Algorithme	Résultat	Remarque	Transformation effectuée
$T_1(v) = \begin{cases} 255 & \text{si } v > 128 \\ 0 & \text{sinon} \end{cases}$	<pre>function h=T_1(v) if v>128 h=255; else h=0; end end</pre>		Les seules couleurs affichées sont du noir et du blanc	seuillage
$T_2(v) = 255 - v$	<pre>function h2=T_2(v) h2=255-v; end</pre>		Les couleurs ont été inversées	négatif

2.2 Amélioration de la dynamique

Image originale et son histogramme :

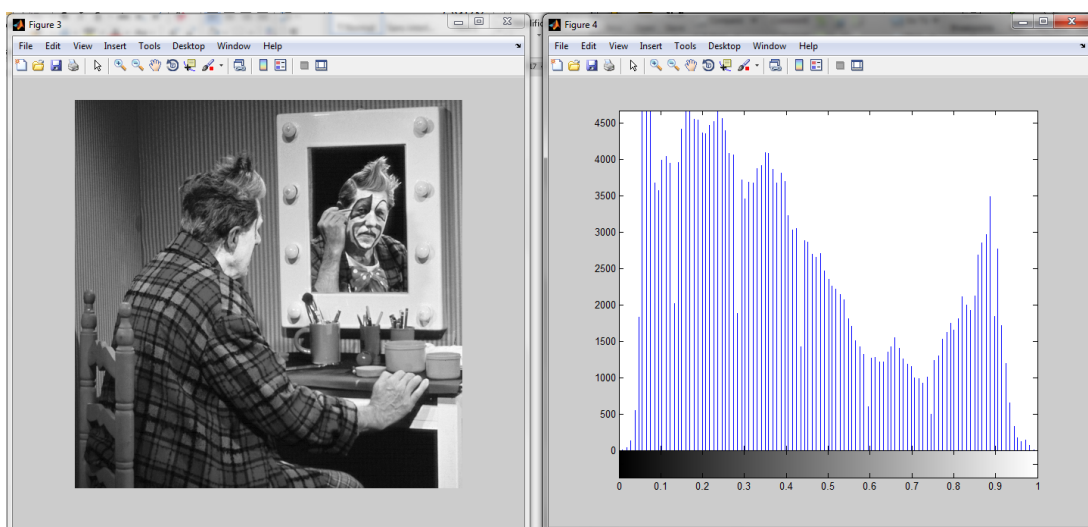


Algorithme utilisé :

$$h(v) = \frac{(v - \min)}{\max - \min} * 255$$

```
function h=T(v,min,max)
    if min==max
        h=0;
    else
        h=((v-min)*1.0/(max-min))*255;
    end
end
```

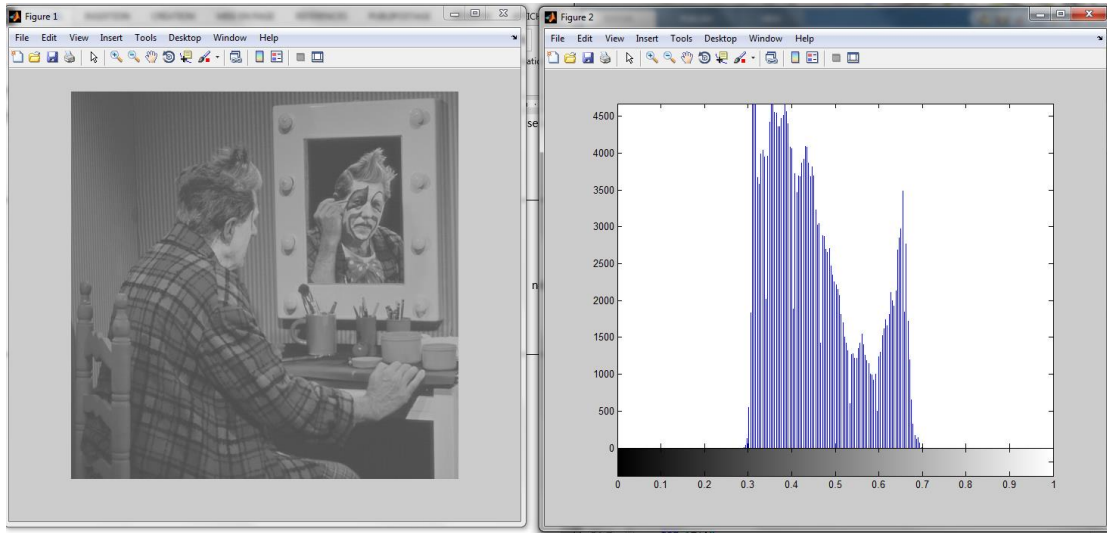
Image et histogramme obtenus :



Nous pouvons voir que l'image est nettement améliorée et que son histogramme a été étalé sur l'intégralité du spectre.

2.3 Histogramme cumulé

Image originale et son histogramme :



Algorithme utilisé :

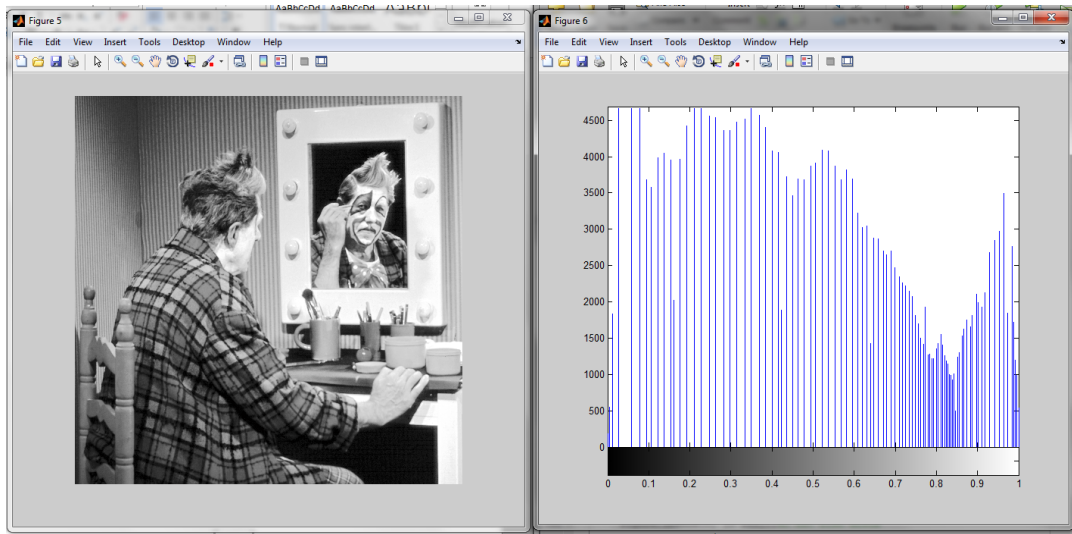
```
function h2=histo_cumul(v)
    h2=zeros(1,256);
    [x,y]=size(v);
    for i=1:x
        for j=1:y
            h2(1,v(i,j)+1)=h2(1,v(i,j)+1)+1;
        end
    end
    for k=2:256
        h2(1,k)=h2(1,k)+h2(1,k-1);
    end
    h2=h2/h2(1,256);

end

function h3=T_2(v,table_assoc)
    h3=255*table_assoc(1,v+1);
end

tab=histo_cumul(u);
[x,y]=size(u);
V=zeros(x,y);
for i=1:x
    for j=1:y
        tmp=u(i,j);
        V(i,j)=T_2(tmp,tab);
    end
end
```

Image et histogramme obtenus :


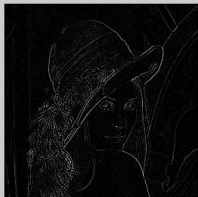


Nous pouvons voir que l'image est nettement améliorée et que son histogramme a été étalé sur l'intégralité du spectre. De plus, l'histogramme voit son intensité mieux répartie au long du spectre, même si cela n'est pas parfait.

3 Filtrage

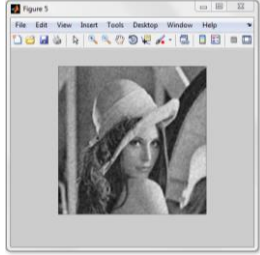
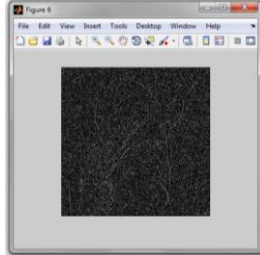
3.2 Applications

Filtre

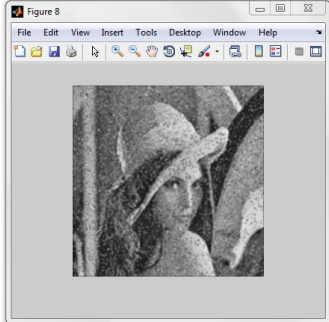
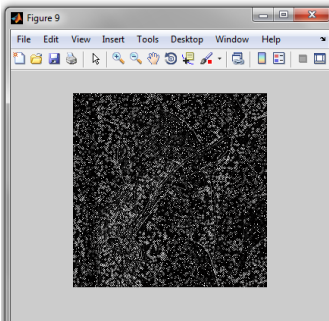
Filtre	Algorithme	Image	Remarque	Filtre
$h_1 = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	<pre>h1=ones(3)/9; w1=imfilter(u,h1); figure;imshow(w1, [0 255]);</pre>		Le filtre rend ici l'image plus floue	moyennant
$h_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	<pre>h2=[0 1 0;1 -4 1;0 1 0]; w2=imfilter(u,h2); figure;imshow(w2, [0 255]);</pre>		Le filtre décrit les contours et utilise l'opposé du Laplacien	médian

"randn", moyenne nulle, écart-type 20

```
bruit=u+(20*randn(size(u)));
figure;imshow(bruit, [0 255]);
w3=imfilter(bruit,h1);
w4=imfilter(bruit,h2);
figure;imshow(w3, [0 255]);
figure;imshow(w4, [0 255]);
```

Filtre	Algorithme	Image	Commentaire
$h_1 = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	<pre>bruit=u+(20*randn(size(u))); w3=imfilter(bruit,h1); figure;imshow(w3, [0 255]);</pre>		On voit qu'effectivement un « bruit » (des pixels aléatoires) a été rajouté à l'image floue
$h_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	<pre>bruit=u+(20*randn(size(u))); w4=imfilter(bruit,h2); figure;imshow(w4, [0 255]);</pre>		De la même manière, on peut voir ici que des pixels aléatoires ont été rajoutés à l'image des contours de LENA

"imnoise", probabilité 10%

Filtre	Algorithme	Image	Effet du filtre
$h_1 = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	<pre>V = imnoise(uint8(u), 'salt & pepper', 0.1); w5=imfilter(V,h1); figure;imshow(w5, [0 255]);</pre>		On voit qu'effectivement un « bruit » (des pixels aléatoires noirs ou blancs) a été rajouté à l'image floue
$h_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	<pre>V = imnoise(uint8(u), 'salt & pepper', 0.1); w6=imfilter(V,h2); figure;imshow(w6, [0 255]);</pre>		De la même manière, on peut voir ici que des pixels aléatoires noirs et blancs ont été rajoutés à l'image des contours de LENA

4 Transformée de Fourier

4.1 Définitions

Algorithme :

```
%lecture et affichage de l'image de départ
u=double(imread('LENA.BMP'));
figure;imshow(u, [0 255]);

%transformée de Fourier
v=fft2(u);

%calcul de la norme de la transformée de Fourier
function w=norme(v)
    [x,y]=size(v);
    w=v;
    for i=1:x
        for j=1:y
            w(i,j)=log(norm(v(i,j)));
        end
    end
end

w=norme(v);

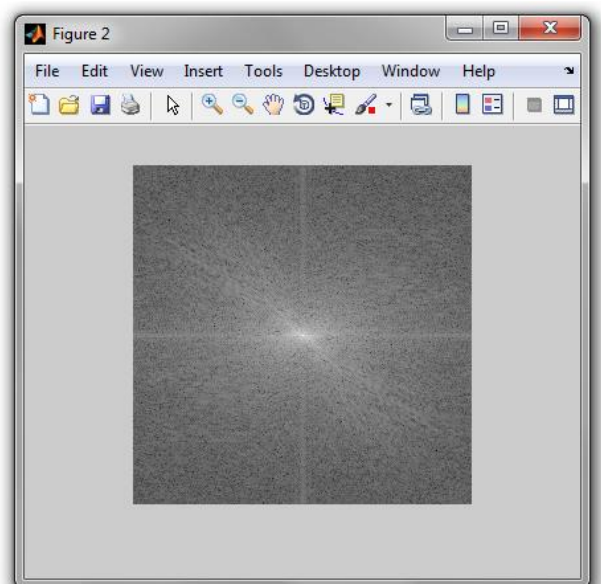
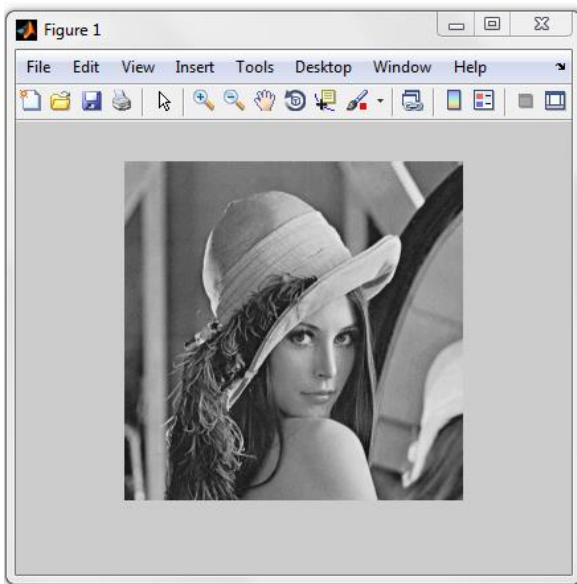
%on récupère les valeurs extrêmes (minimale et maximale) de la norme
function h=T(v,min,max)
    if min==max
        h=0;
    else
        h=((v-min)*1.0/(max-min))*255;
    end
end

[x,y]=size(w);
min=w(1,1);
max=w(1,1);
for i=1:x
    for j=1:y
        tmp = w(i,j);
        if tmp>max
            max=tmp;
        end
        if tmp<min
            min=tmp;
        end
    end
end

%On étale les valeurs de Fourier calculées sur l'espace affichable (de 0 a
255)
V=zeros(x,y);
for i=1:x
    for j=1:y
```

```
        tmp=w(i,j);  
        V(i,j)=T(tmp,min,max);  
    end  
end  
  
%On affiche le résultat recadré  
figure;imshow(fftshift(V), [0 255]);
```

Image originale (à gauche) et résultat obtenu (à droite) :



4.2 Filtrage fréquentiel

Algorithme :

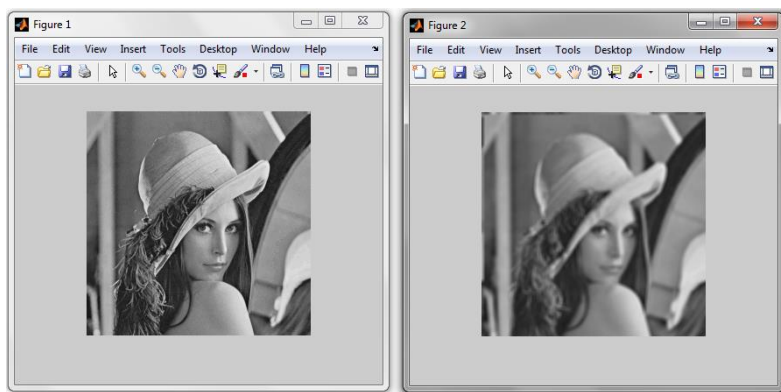
```
%chargement de l'image
u=double(imread('LENA.BMP'));
figure;imshow(u, [0 255]);
%transformée de Fourier de l'image u
v=fft2(u);

%Pour après, le filtre doit avoir la même taille que u pour pouvoir
effectuer la multiplication terme à terme. Donc on crée une matrice nulle
de même taille que l'image dont on ne remplit qu'une partie avec le filtre
H=zeros(size(u));
%Création du masque carré de taille 5x5
for i=1:5
    for j=1:5
        H(i,j)=1/25;%Normalisation du filtre
    end
end
%transformée de Fourier du Filtre
hf=fft2(H);

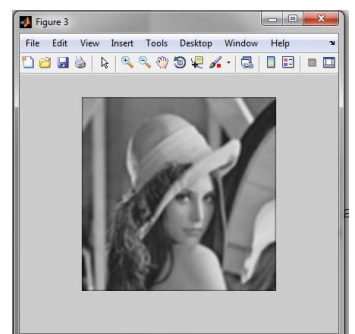
%multiplication dans le domaine fréquentiel
transfo=hf.*v;

%transformée inverse de la transformation
final=ifft2(transfo);
%affichage du résultat
figure;imshow(final, [0 255]);
```

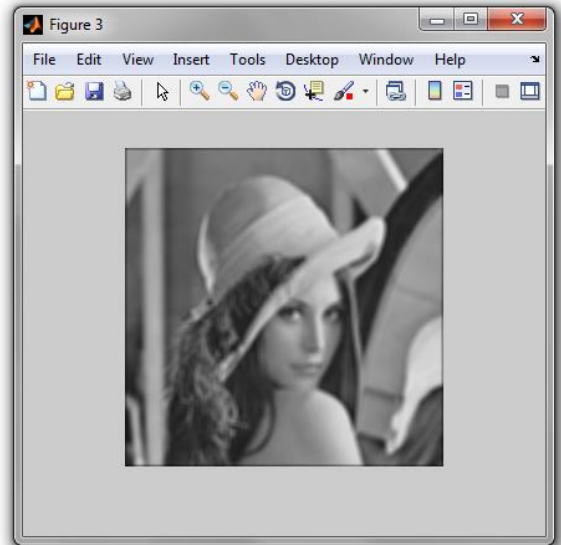
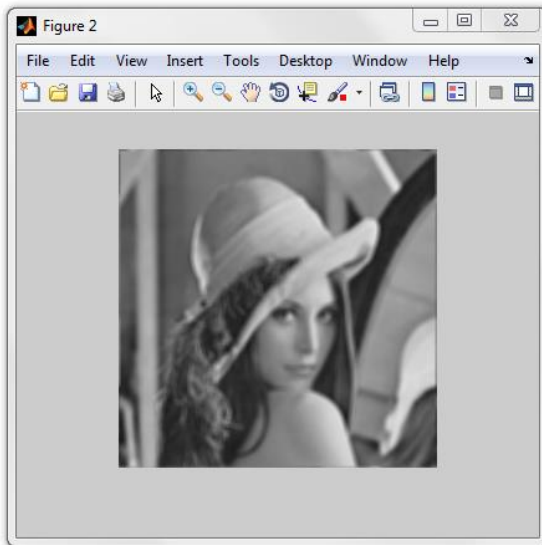
Image originale (à gauche) et résultat obtenu (à droite) :



Si nous appliquons directement le filtre sur l'image, nous obtenons l'image ci-après. Ce qui est le même résultat que l'image obtenue après transformée de Fourier et multiplication dans le domaine fréquentiel. Nous pouvons donc dire que nos résultats sont cohérents.



Remarque : Sur l'image obtenue par la transformée de Fourier on voit que nous avons obtenu un léger décalage (visible en haut à gauche de l'image). Ceci est dû au fait que nous avons mis le filtre entre (0,0) et (4,4) et non centré en (0,0). Si nous décalons manuellement le filtre, nous remarquons que nous obtenons désormais un filtre parfaitement centré :



Annexes :

```
function Td1()
    close all;
    %1 Fonction utiles
    u=double(imread('LENA.BMP'));
    figure;imshow(u, [0 255]);
    figure;imagesc(u, [0 255]);
    colormap(gray);
    figure;image(u);
    colormap(gray);

    %2 Histogramme
    %2 1 Transformation simples

    function h=T_1(v)
        if v>128
            h=255;
        else
            h=0;
        end
    end

    function h2=T_2(v)
        h2=255-v;
    end

    [x,y]=size(u);
    V=zeros(x,y);
    W=zeros(x,y);
    for i=1:x
        for j=1:y
            tmp=u(i,j);
            V(i,j)=T_1(tmp);
            W(i,j)=T_2(tmp);
        end
    end

    %    figure;imshow(V, [0 255]);%seillage
    %    figure;imshow(W, [0 255]);%negatif

    %2 2 Amelioration de la dynamique
    %    clown();

    %3 Filtrage
    %    filtre();

    %FFT
    %    Fourier();

    %filtrage frequentiel
    %    filtrage_freq();
end

function clown()
    u=double(imread('CLOWN_LUMI2.BMP'));
```

```
figure;imshow(u, [0 255]);
figure;imhist(u/255)

function h=T(v,min,max)
    if min==max
        h=0;
    else
        h=((v-min)*1.0/(max-min))*255;
    end
end

[x,y]=size(u);
min=u(1,1);
max=u(1,1);
for i=1:x
    for j=1:y
        tmp = u(i,j);
        if tmp>max
            max=tmp;
        end
        if tmp<min
            min=tmp;
        end
    end
end

V=zeros(x,y);
for i=1:x
    for j=1:y
        tmp=u(i,j);
        V(i,j)=T(tmp,min,max);
    end
end
figure;imshow(V, [0 255]);%C est bien mieux
figure;imhist(V/255)

%2 3 Histogramme cumule
function h2=histo_cumul(v)
    h2=zeros(1,256);
    [x,y]=size(v);
    for i=1:x
        for j=1:y
            h2(1,v(i,j)+1)=h2(1,v(i,j)+1)+1;
        end
    end
    for k=2:256
        h2(1,k)=h2(1,k)+h2(1,k-1);
    end
    h2=h2/h2(1,256);
end

function h3=T_2(v,table_assoc)
    h3=255*table_assoc(1,v+1);
end

tab=histo_cumul(u);
[x,y]=size(u);
V=zeros(x,y);
for i=1:x
```

```
        for j=1:y
            tmp=u(i,j);
            V(i,j)=T_2(tmp,tab);
        end
    end
    figure;imshow(V, [0 255]);%C est bien mieux
    figure;imhist(V/255)

end

function filtre()

    u=double(imread('LENA.BMP'));
    figure;imshow(u, [0 255]);

    h1=ones(3)/9;
    h2=[0 1 0;1 -4 1;0 1 0];

    w1=imfilter(u,h1);
    w2=imfilter(u,h2);
    figure;imshow(w1, [0 255]);%flou, on fait une moyenne ==> filtre
moyennant
    figure;imshow(w2, [0 255]);%on fait les contours, opose du Laplacien
==> filtre mediant
    V = imnoise(uint8(u), 'salt & pepper',0.1);
    figure;imshow(V, [0 255]);
    w5=imfilter(V,h1);
    w6=imfilter(V,h2);
    figure;imshow(w5, [0 255]);%flou, on fait une moyenne
    figure;imshow(w6, [0 255]);
    bruit=u+(20*randn(size(u)));
    figure;imshow(bruit, [0 255]);
    w3=imfilter(bruit,h1);
    w4=imfilter(bruit,h2);
    figure;imshow(w3, [0 255]);%flou, on fait une moyenne
    figure;imshow(w4, [0 255]);%on fait les contours, opose du Laplacien

    %on fait les contours, opose du Laplacien

end

function Fourier()

%lecture et affichage de l'image de depart
    u=double(imread('LENA.BMP'));
    figure;imshow(u, [0 255]);

%transformee de Fourier
    v=fft2(u);

    function w=norme(v)
        [x,y]=size(v);
        w=v;
        for i=1:x
            for j=1:y
                w(i,j)=log(norm(v(i,j)));
            end
        end
    end

end
```



```
end

%calcul de la norme de la transformee de Fourier
w=norme(v);

function h=T(v,min,max)
    if min==max
        h=0;
    else
        h=((v-min)*1.0/(max-min))*255;
    end
end

%on recupere les valeurs extremes (minimale et maximale)de la norme
[x,y]=size(w);
min=w(1,1);
max=w(1,1);
for i=1:x
    for j=1:y
        tmp = w(i,j);
        if tmp>max
            max=tmp;
        end
        if tmp<min
            min=tmp;
        end
    end
end

%On etale les valeurs de Fourier calculees sur l'espace affichable (de 0 a
%255)
V=zeros(x,y);
for i=1:x
    for j=1:y
        tmp=w(i,j);
        V(i,j)=T(tmp,min,max);
    end
end

%On affiche le resultat recadre
figure;imshow(fftshift(V), [0 255]);

end

function filtrage_freq()
    %chargement de l'image
    u=double(imread('LENA.BMP'));
    figure;imshow(u, [0 255]);
    %transformée de Fourier de l'image u
    v=fft2(u);

    %Pour après, le filtre doit avoir la même taille que u pour pouvoir
    %effectuer la multiplication terme à terme. Donc on crée une matrice
    %nulle de même taille que l'image dont on ne remplit qu'une partie avec
    %le filtre
    H=zeros(size(u));
    %Création du masque carré de taille 5x5
    for i=1:3
        for j=1:3
```

```
H(i,j)=1/25;
sum=sum+1;
end
end

for i=x-1:x
    for j=1:3
        H(i,j)=1/25;
        sum=sum+1;
    end
end

for i=x-1:x
    for j=y-1:y
        H(i,j)=1/25;
        sum=sum+1;
    end
end

for i=1:3
    for j=y-1:y
        H(i,j)=1/25;
        sum=sum+1;
    end
end

%transformée de Fourier du Filtre
hf=fft2(H);

%multiplication dans le domaine fréquentiel
transfo=hf.*v;

%transformée inverse de la transformation
final=ifft2(transfo);
%affichage du résultat
figure;imshow(final, [0 255]);%C est un flou

h=ones(5)/25;
w=imfilter(u,h);
figure;imshow(w, [0 255]);%on obtient bien la meme chose
end
```