# TP4 : FILTRAGE D'UNE IMAGE : DOMAINE FREQUENTIEL

## 1. Objectifs

Le filtrage linéaire consiste en un produit de convolution dans le domaine spatial, ce qui correspond à une multiplication dans le domaine spectral. On s'intéresse donc souvent à la réponse fréquentielle d'un filtre pour savoir notamment quelles fréquences il va amplifier, quelles directions privilégiées il va mettre en évidence, etc... En particulier, en observant la transformée de Fourier du masque de convolution (éventuellement complété par des zéros), on arrive à observer le comportement fréquentiel du filtre. Tout comme la transformée de Fourier d'une image classique, on peut représenter la réponse fréquentielle en échelle linéaire ou en échelle logarithmique.

## 2. Transformée de Fourier d'une image

La transformée de Fourier de l'image $f(x,y)$, de largeur $N$ et de hauteur $M$ est donnée par :

$$F(u,v)=\sum_{x=0}^{N-1}\sum_{y=0}^{M-1} f(x,y)e^{-i2\pi(\frac{ux}{N}+\frac{vy}{M})}$$

Cette transformée de Fourier se calcule sous MATLAB avec la commande **`fft2,`** qui prend en paramètre une image et renvoie une matrice complexe de même taille que l'image et qui contient les différents coefficients de la transformée de Fourier. Utiliser l'instruction **`fftshift`** pour placer la fréquence nulle au centre de l'image.

***Exemple de commande pour calculer la transformée de Fourier d'une image :***

```
Y = fft2(X); % Sans réarrangement
Y = fftshift(fft2(X)); % Avec réarrangement
```

## 3. Visualisation de la transformée de Fourier d'une image

Pour afficher la transformée de Fourier d'une image sous la forme d'une image, on peut visualiser le module soit avec une échelle linéaire soit avec une échelle logarithmique (décibel). L'échelle logarithmique permet de visualiser avec précision le spectre pour les fréquences où il prend des valeurs très faibles . On introduit dans cette échelle un paramètre C qui permet de contrôler la dynamique et d'éviter de prendre des logarithmes de 0.

***Deux façons de calculer le module :***

```
% Y : transformee de Fourier d'une image
Y1 = abs(Y); % Echelle lineaire
Y2 = 10*log10(C+abs(Y)); % Echelle logarithmique
```

Après avoir calculé le module, ramener toutes les valeurs entre 0 et 1. Pour cela, on peut utiliser la fonction **`mat2gray`** de MATLAB, qui renormalise la matrice en fixant le minimum à 0 et le maximum à 1.

***Renormalisation des coefficients d'une matrice :***

```
% Y : Matrice ayant des coefficients quelconques
Y_norm = mat2gray(Y);% Y_ norm : Matrice ayant des coefficients entre 0 et 1
```

## 4. Transformée de Fourier inverse

La transformée inverse du domaine fréquentiel au domaine spatial est donnée par :

$$f(x,y)=\frac{1}{MN}\sum_{u=0}^{N-1}\sum_{v=0}^{M-1} F(u,v)e^{i2\pi(\frac{ux}{N}+\frac{vy}{M})}$$

On calcule la transformée de Fourier inverse grâce à la fonction ifft2 de MATLAB. Etant données les erreurs de précision de MATLAB, il est plus prudent de prendre ensuite la partie entière pour s'assurer que l'image reconstruite est bien réelle.

***Transformée de Fourier inverse***

```
% Y : transformee de Fourier
```

```
X = real(ifft2(Y));
```

5. **Interprétation de la Transformée 2D**
   - Hautes fréquences : loin du centre de la TF
   - Basses fréquences : proche du centre de la TF
   - Composante continue : centre de l'image (fréquence zéro = moyenne de l'image)
   - Si l'on voit apparaître des lignes sur le spectre, cela signifie qu'on peut retrouver des éléments assimilables à des lignes ou des directions privilégiées dans l'image, dans la direction perpendiculaire à celle donnée par le spectre.

6. **Etude sous MATLAB**

**6.1. Quelques images réelles :** Nous allons travailler dans cette partie sur trois images et observer leurs spectres.

1. Créer sous MATLAB un script vide nommé TP4Partie1.m
2. Ouvrir l'image `image1.bmp`, la stocker dans une matrice X1 et l'afficher.

   (a) Diriez vous que cette image contient plutôt des basses ou des hautes fréquences ? Y'a-t-il dans l'image des directions privilégiées ?

   (b) Calculer la transformée de Fourier Y1 de l'image. Réarranger cette transformée (grâce à fftshift) et calculer son module.

   (c) En utilisant la commande mat2gray pour renormaliser les valeurs, afficher le spectre d'abord en échelle linéaire, puis en échelle logarithmique (on prendra C = 100).

   (d) Quelle échelle vous semble la plus appropriée ici ?

   (e) Vérifiez vous vos observations en regardant le spectre ? Commenter.

3. Réaliser le même processus pour `image2.bmp` et `cameraman.tif`. Comparer les résultats et commenter.

**6.2. Images synthétiques**

1. Créer sous MATLAB un script vide nommé TP4Partie2.m

2. Pour chacune de ces images, générer une image de taille 100 ×100 et afficher sur la même figure l'image, son spectre en échelle linéaire et son spectre en échelle logarithmique (On prendra C = 100 , fx=0.1 et fy=0.4). Commenter et interpréter les résultats obtenus.

1. `f1(i,j)=sin(fx*i+fy*j);`
2. `f2(i,j)=sin(fx*j);`
3. `f3(i,j)=0.5*(sin(fx*i)+sin(fy*j));`
4. `f4(i,j)=j/100;`

**6.3. Sélection des basses(ou hautes) fréquences :** Afin de mieux comprendre à quoi correspondent les basses et hautes fréquences, nous allons faire l'expérience suivante. Nous allons essayer de ne conserver que les basses fréquences (ou hautes) dans le spectre et de reconstruire l'image par transformée de Fourier inverse. Pour sélectionner les basses (ou hautes fréquences), nous allons considérer un cercle de rayon R centré sur la fréquence nulle, et ne garder que les coefficients de Fourier correspondant aux fréquences comprises (ou non comprises) dans le cercle.

*La fonction SeuillageFrequencesFourier(X,R,param) fournie réalise les étapes suivantes :*
- *Prend en entrée une image renormalisée X, un rayon R (entier), et un paramètre param (égal à 0 ou 1)*
- *Calcule la transformée de Fourier réarrangée*
- *Deux cas :*
  - *Si param = 0, annule toutes les coefficients de Fourier associées aux fréquences situées à une distance de plus de R pixels de la fréquence centrale*
  - *Si param = 1, annule toutes les coefficients de Fourier associées aux fréquences situées à une distance de moins de R pixels de la fréquence centrale*
- *Reconstruit une image par transformée de Fourier inverse*

• *Affiche sur la même image, de gauche à droite et de haut en bas :*
  • *L'image originelle (en haut à gauche)*
  • *Le spectre de l'image originelle en échelle linéaire (en haut au milieu)*
  • *Le spectre de l'image originelle en échelle logarithmique (en haut à droite)*
  • *L'image reconstruite (en bas à gauche)*
  • *Le spectre de l'image reconstruite en échelle linéaire (en bas au milieu)*
  • *Le spectre de l'image reconstruite en échelle logarithmique (en bas à droite)*

1. Créer sous MATLAB un script vide nommé TP4Partie3.m
2. Ouvrir l'image cameraman.tif, la stocker dans une matrice Z1 .
   (a) Lancer la fonction SeuillageFrequencesFourier.m avec R = 20 et param = 0. Commenter.
   (b) Relancer la même simulation avec param = 1. Commenter.
   (c) Faire varier R et commenter.
3. Recommencer ces opérations avec une image naturelle ou synthétique de votre choix. Commenter.

### 6.4. Filtrage d'une image : domaine fréquentiel

1. Créer sous MATLAB un script vide nommé TP4Partie4.m
2. Ouvrir l'image cameraman.tif, la stocker dans une matrice X1 et la renormaliser. Générer un masque h correspondant à un filtre moyenneur de taille 3 × 3.
3. Calculer et afficher les spectres de l'image originelle, de l'image filtrée et la réponse en fréquence du filtre.
4. Quel effet le filtre a-t-il sur le spectre ? S'agit-il d'un filtre passe-bas, passe-haut, etc... ?
5. Refaire la même expérience avec un filtre Gaussien de taille 15 × 15 et d'écart type= 1.5. Commenter.

**Partie1**

```
X=imread ('image1.bmp');
Y = fft2(X); % Sans réarrangement
Yf = fftshift(Y); % Avec réarrangement
figure(1); imshow(X) ;title('image orig')
Y1 = abs(Yf); % Echelle lineaire
Y2 = 10*log10(100 + abs(Yf)); % Echelle logarithmique
X1 = real(ifft2(Y));
figure(2); imshow(mat2gray(Y1)) % eq figure(2); imshow(Y1,[])
figure(3); imshow(Y2, [])
figure(4); imshow(angle(Yf), [])
X1 = 255*real(ifft2(Y));
figure(5); imshow(X1,[])
```

**Partie2**

```
fx=0.1 ;fy=0.4
for i=1:100 ; for j=1:100
   f1(i,j)=sin(fx*i+fy*j);
   f2(i,j)=sin(fx*j);
   f3(i,j)=0.5*(sin(fx*i)+sin(fy*j));
   f4(i,j)=j/100;
    end ;end
figure(6);imshow([f1 f2 f3 f4])
image=f1; Yf1 = fftshift(fft2(image));
figure(7);imshow([mat2gray(image) mat2gray(abs(Yf1)) ...
    mat2gray(abs(10*log10(100 + abs(Yf1))))])
```

**Partie3**

```
X=imread ('image2.bmp');
R=50 ;param=0 ;SeuillageFrequencesFourier(X,R,param)
```

```
function SeuillageFrequencesFourier(X,R,param)
[M,N]=size(X);
masque=zeros(M);
x=1:M; y=1:M;
if mod(M,2)==0
    c=M/2+1;
else
    c=(M+1)/2;
end
[x_vect,y_vect]=ndgrid(x,y);
if param==0
    masque((x_vect-c).^2 + (y_vect-c).^2 < R^2)=1;
else
    masque((x_vect-c).^2 + (y_vect-c).^2 > R^2)=1;
end
figure(1),imshow(masque,[])
Y=ifftshift(fft2(X));
Y_fil=Y.*masque;
X_fil=ifft2(ifftshift(Y_fil));
C=10;
Y=abs(Y);
Y_fil=abs(Y_fil);
Y_log=10*log10(Y+C);
Y_fillog=10*log10(Y_fil+C);
figure(2);imshow([mat2gray(X) mat2gray(Y) mat2gray(Y_log) ; ...
    mat2gray(real(X_fil)) mat2gray(Y_fil) mat2gray(Y_fillog)])
```

**Partie4**

```
clear all; close all ;clc
X=imread ('cameraman.tif');
X=double(X)/255;
h = fspecial('average',[5 5])
Fsp = imfilter(X,h);
 [n m]=size(X);
 Xf=fft2(X,n,m);
H=fft2(h,n,m);
Y_fil=Xf.*H;
Ffrq=mat2gray(real(ifft2(Y_fil)));
 figure(1);imshow([X abs(ifftshift(H)) Fsp Ffrq],[])
EQM1=1/n/m*sum(sum((X-Fsp).^2));
PSNRsp=10*log10(1/EQM1)
EQM2=1/n/m*sum(sum((X-Ffrq).^2));
PSNRfrq=10*log10(1/EQM2)
```

**TP-3**

```
X=imread ('cameraman.tif');
m=0 ;v=0.01 ;Y1 = imnoise(X,'gaussian',m,v);
figure(1);subplot(131); imshow(X);title('image orig')
subplot(132); imshow(Y1) ;title('bruit Gaussien v=0.01')
h_1 = fspecial('average',[3 3])
%h_1 = fspecial('gaussian',[3 3],0.8)
yy1 = imfilter(Y1,h_1,'replicate');
%yy1 = medfilt2(Y1,[3 3]);
subplot(133); imshow(yy1);title('moy 3x3')
[l k]=size(X);
EQM=1/l/k*sum(sum((X-yy1).^2)); PSNR=10*log10((255^2)/EQM)
```