TP2: manipulation d'images en matlab et transformée de Fourier discrète

Pour Matlab, une image de taille (m;n) est une matrice I de taille (m;n), et la valeur de I(i; j) correspond la valeur du niveau de gris de limage au pixel (i; j). Matlab est capable de lire peu prs tous les formats standards dimages.

1. Exemples : visualisation d'une image

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Chargement dune image en Matlab:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load gatlin2;
% -> Limage est chargee dans la variable X
%Autres images:
%load clown; load gatlin; load mandrill;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Visualisation:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
imagesc(X);
colormap gray;
%pour voir limage en niveaux de gris
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Pour ouvrir une deuxieme figure:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(2);
colormap gray;
XX=imread(cameraman.tif);
imshow(XX);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

On peut aussi utiliser la commande `imwrite`, puis visualiser l'image sauvegardée avec son éditeur d'image préféré (souvent plus pratique que celui de Matlab).

Sources d'erreur classique : Il faut faire attention que les commandes `imwrite` et `imread` supposent que les images sont codées comme des entiers non signés (`uint8` permet de faire la conversion), alors que `imagesc` lit les images au format double. Comme nous serons amener à faire des calculs au format double, si on ne fait pas attention au format de vos données, on obtiendre de mangifiques images noires en sortie de vos codes...

Choix d'une image : Vous pouvez continuer le TP avec votre image favorite (à condition qu'elle soit de taille raisonnable). Sachant que le TP va être basé sur la transformée de Fourier, et quavec matlab vous disposez de la FFT, donner une condition nécessaire pour que la taille soit raisonnable ?

Si vous choisissez une image couleur, n'oubliez pas de la convertir en niveau de gris. Cela peut se faire en faisant, par exemple, `a=mean(a,3);`

Dans la suite, nous souhaitons tester ce que nous faisons sur des images qui ne soient pas parfaite, c'est-à-dire sur des des images auxquelles on a ajoutté du bruit. Pour cela, nous utiliserons la procédure suivante:

```
Un premier exemple de fonction Matlab : bruitage dune image
function out = bruitage_gaussien(I,s)
%bruitage gaussien (decart-type s) dune image I
% bruitage_gaussien(I,s)
[m,n]=size(I);
```

```
J=zeros(m,n);
%creation du bruit gaussien
J=s*randn(m,n);
%bruitage
out=I+J;
```

2. Transformée de Fourier d'une image

La transformée de Fourier 2-dimensionnelle d'une image I (*i.e.* d'une matrice) (m,n) est définie par

$$\mathcal{F}_{m,n}[I](p,q) = \sum_{j=0}^{m-1}\sum_{k=0}^{n-1} I(j,k)e^{-2i\pi\left(\frac{jp}{m}+\frac{kq}{n}\right)}.$$

C'est donc une transformée de Fourier 1-dimensionelle dans la première variable des transformée de fourier 1-dimensionelles dans la seconde

$$\begin{aligned}\mathcal{F}_{m,n}[I](p,q) &= \sum_{j=0}^{m-1}\left(\sum_{k=0}^{n-1} I(j,k)e^{-2i\pi\frac{kq}{n}}\right)e^{-2i\pi\frac{jp}{m}}\\ &= \sum_{j=0}^{m-1}\mathcal{F}_n[I(j,\cdot)](q)e^{-2i\pi\frac{jp}{m}}\\ &= \mathcal{F}_m\big[\big(\mathcal{F}_n[I(0,\cdot)](q),\mathcal{F}_n[I(1,\cdot)](q),\dots,\mathcal{F}_n[I(m-1,\cdot)](q)\big](p)\end{aligned}$$

La commande matlab correspondante est la commande `fft2`.

Attention: La commande matlab `fft` appliquée à une image retourne la matrice dont la k-ième colonne est le vecteur composé des transformées de fourier discrètes (TFD) de la k-ième colonne de I, il faut donc encore faire une TFD par ligne.

Afin de pouvoir visualiser des transformées de Fourier d'images on est confronté à plusieurs problèmes:

– c'est un objet complexe de dimension 2, donc un objet de dimension réelle 4. On ne trace donc que le module.

– Le module est fortement concentré au point $(0,0)$ ce qui fait que sur la plupart des images on ne voit qu'un point lumineux. On tracera donc plutôt $\ln(1+|\mathcal{F}_{m,n}[I]|)$.

– la transformée de Fourier discrète doit correspondre à des coefficients de Fourier. Mais, comme dans le cas de la dimension 1, les coefficients ne sont pas rangés dans le bon ordre. La fonction `fftshift` les mets dans l'ordre naturel.

3. Zoom

Le but de cette section est de réaliser un zoom d'un facteur 2 d'une image, c'est-à-dire de transformer une image I de taille (N,M) en une image J de taille $(2N,2M)$ qui lui ressemble visuellement. Pour cela, nous allons essayer 3 méthodes.

3.1. **Zoom par duplicata de pixels.**

Il s'agit ici simplement de dupliquer les pixels.

Le principe est expliqué dans le schéma suivant:

$I(j,k)$	$I(j,k+1)$
$I(j+1,k)$	$I(j+1,k+1)$

$\rightarrow$ doublement de taille

$I(j,k)$	0	$I(j,k+1)$	0
0	0	0	0
$I(j+1,k)$	0	$I(j+1,k+1)$	0
0	0	0	0

opération:

(M,N)=taille de I
J=0(2*M,2*N)
$J(2m-1,2n-1)=I(m,n)$ pour $m=1,\ldots,M$ et $n=1,\ldots,N$
ou (sans boucle) J(1:2:2M-1,1:2:2N-1)=I

$\rightarrow$ copie des pixels

$I(j,k)$ $\rightarrow$ $\downarrow$ $\searrow$	I(j,k)	$I(j,k+1)$ $\rightarrow$ $\downarrow$ $\searrow$	$I(j,k+1)$
I(j,k)	I(j,k)	$I(j,k+1)$	$I(j,k+1)$
$I(j+1,k)$ $\rightarrow$ $\downarrow$ $\searrow$	$I(j+1,k)$	$I(j+1,k+1)$ $\rightarrow$ $\downarrow$ $\searrow$	$I(j+1,k+1)$
$I(j+1,k)$	$I(j+1,k)$	$I(j+1,k+1)$	$I(j+1,k+1)$

opération:

$$J(1+2m+1,1+2n)=I(1+m,1+n)$$
$$J(1+2m,1+2n+1)=I(1+m,1+n)$$
$$J(1+2m+1,1+2n+1)=I(1+m,1+n)$$

Exercice 1. Programmez cette opération puis regardez le résultat sur les images clown,gatlin et mandrin. Recommancez l'opération en ajoutant un peu de bruit à ces images.

Tout est déjà écrit:

```
function [Z]=zoom1(I)
%% cette fonction zoom d'un facteur 2 une image par duplicata de pixel

[M N]=size(I);
Z=zeros(2*M,2*N);

Z(1:2:2*M-1,1:2:2*N-1)=I;
Z(2:2:2*M,1:2:2*N-1)=I;
Z(1:2:2*M-1,2:2:2*N)=I;
Z(2:2:2*M,2:2:2*N)=I;

end
```

Notez que les $2M-1$, $2N-1$ ne servent à rien.

L'effet de cette fonction est visuellement assez désagréable sur les zones qui contiennent des courbes (effet de crnelage, trame dans les ombres). En présence de bruit, cet effet est renforcé

3.2. **Zoom par moyénisation de pixels.**
Il s'agit cette fois ci de faire une moyenne sur les pixels voisins:

$I(j,k)$	$I(j,k+1)$
$I(j+1,k)$	$I(j+1,k+1)$

$\downarrow$
doublement
de
taille
$\downarrow$

$I(j,k)$	0	$I(j,k+1)$
0	0	0
$I(j+1,k)$	0	$I(j+1,k+1)$

$\downarrow$
moyénisation des pixels
des
pixels
$\downarrow$

$I(j,k) \rightarrow$ $\downarrow \quad \searrow$	$\dfrac{I(j,k)+I(j,k+1)}{2}$	$\leftarrow \quad I(j,k+1)$ $\swarrow \quad \downarrow$
$\dfrac{I(j,k)+I(j+1,k)}{2}$	$\dfrac{I(j,k)+I(j+1,k)+I(j,k+1)+I(j+1,k+1)}{4}$	$\dfrac{I(j,k+1)+I(j+1,k+1)}{2}$
$\uparrow \quad \nearrow$ $I(j+1,k) \quad \rightarrow$	$\dfrac{I(j,k+1)+I(j+1,k+1)}{2}$	$\nwarrow \quad \uparrow$ $\leftarrow \quad I(j+1,k+1)$

Notez que cette dernière opération pose un problème pour remplir la dernière ligne et la dernière colonne pour lequelles $I(j+1,\cdot)$ et $I(k+1,\cdot)$ ne sont pas définies. Le plus simple est de mettre ces valeurs à 0. On crée donc une matrics K de taille $(M+1,N+1)$ ne contenant que des 0 puis on remplace la partie de taille (M,N) du coin supérieur gauche par I: telle que $K(m,n)=I(m,n)$ pour $m=1,\ldots,M$ et $n=1,\ldots,N$.

Ensuite, on effectue les opérations suivantes pour $m=1,\ldots,M$ et $n=1,\ldots,N$;

(i) $J(2m-1,2n)=\dfrac{K(m,n)+K(m,n+1)}{2}$

(ii) $J(2m,2n-1)=\dfrac{K(m,n)+K(m+1,n)}{2}$

(iii) $J(2m,2n)=\dfrac{K(m,n)+K(m+1,n)+K(m,n+1)+K(m+1,n+1)}{4}$

Exercice 2. Programmez cette opération puis regardez le résultat sur les images clown,gatlin et mandrin. Recommancez l'opération en ajoutant un peu de bruit à ces images.

La solution est la suivante:

```
function [Z]=zoom2(I)
%% zoom par moyennisation de pixe
[M N]=size(I);
J=zeros(M+1,N+1);
J(1:M,1:N)=I;

%% cr une image dans laquelle on a ajout une ligne et une colonne de 0
```

```
Z=zeros(2*M,2*N);

Z(1:2:2*M,1:2:2*N)=I;
Z(2:2:2*M,1:2:2*N)=(I+J(2:M+1,1:N))/2;
Z(1:2:2*M-1,2:2:2*N)=(I+J(1:M,2:N+1))/2;
Z(2:2:2*M,2:2:2*N)=(I+J(2:M+1,1:N)+J(1:M,2:N+1)+J(2:M+1,2:N+1))/4;

end
```

On note une légère am'élioration du cas précédent, surtout pour le bruit.

3.3. **Zoom par zéro padding.**
L'idée ici est de passer par Fourier. Expliquons cela en dimension 1 (un son): Pour simplifier, prenons $N = 2p + 1$ *impaire*.

On veut associer à la suite $(c_k)_{k=0,\dots,2p}$ une suite de longueur plus grande qui contienne la même information sonore, c'est-à dire les mêmes fréquences avec la même amplitude. Pour cela, on regarde la transformée de Fourier discrète $\hat{c}_k = \mathcal{F}_N[c](k)$ qui s'identifie à c via la formule d'inversion de Fourier:

$$c_j = \frac{1}{2p+1}\sum_{k=0}^{2p} \hat{c}_k e^{2i\pi kj/N} = \sum_{k=-p}^{p} \tilde{c}_k e^{2i\pi kj/N}$$

où $\tilde{c}_k = \begin{cases} \hat{c}_k & \text{pour } k = 0,\dots,p \\ \hat{c}_{N+k} & \text{pour } k = -1,\dots,-p \end{cases}$.

Ensuite, $\tilde{c}_j$ s'identifie avec le polynôme trigonométrique de degré p $P(t) = \frac{1}{2p+1}\sum_{j=-p}^{p} \tilde{c}_j e^{2i\pi jt}$ et la formule d'inversion de Fourier nous dit que $c_j = P(j/N)$.

Mais, ce polynôme trigonométrique de degré p peut tout aussi bien être vu comme un polynôme trigonométrique de degré $q > p$. $P(t) = \frac{1}{2p+1}\sum_{j=-q}^{q} \tilde{d}_j e^{2i\pi jt}$ avec $\tilde{d}_j = \begin{cases} \tilde{c}_j & \text{si } |j| \le p \\ 0 & \text{sinon} \end{cases}$. On définit *d*alors une suite de longueur $2q + 1 = N + 2q$ par $d_j = P(j/(N + 2q))$. Si on regarde de près comment on a obtenu $c_j = P(j/N)$, on voit que cela revient à faire l'opération suivante:

$$c \to \hat{c} = \mathcal{F}_N[c] \to \hat{d} = \big(\mathcal{F}_N[c](0),\dots,\mathcal{F}_N[c](p),\underbrace{0,\dots,0}_{2q \text{ zéros}},\mathcal{F}_N[c](p+1),\dots,\mathcal{F}_N[c](2p)\big) \to d = \mathcal{F}^{-1}_{N+2q}(\hat{d}).$$

En matlab, cela se code

```
function [d]=zoom(c,m)

n=length(c)

hc=fft(c);
tc=fftshift(c);
td=[hc zeros(1,m)];
hd=ifftshift(td);
d=ifft(c);

end;
```

Notez que, grâce à la fonction `fftshift` on peut ajouter des zéros à la fin plutôt qu'au milieu.

Exercice 3. Adaptez ce programmes aux images: écrire une fonction `zoom2` dont l'entrée est une image et deux entiers m, n et dont la sortie est une image dont la taille a été augmentée de (m, n) et qui contienne le même contenu fréquentiel.

Testez votre programme sur les images clown,gatlin et mandrin. Recommancez l'opération en ajoutant un peu de bruit à ces images.

Le programme est le suivant:

```
function [Z]=zoomzeropad(I)
%% zoom par ajout de zero dans la transformee de Fourier
[M N]=size(I);
J=fft2(I);
J=fftshift(J);

%% J contient les coefficients de Fourier discret de I, rangs dans l'ordre
%% de sries de Fourier

p=floor(M/2); q=floor(N/2);

K=zeros(2*M,2*N);

K(p+1:p+M,q+1:q+N)=J;

%% on met J au milieu de K

K=ifftshift(K);

%% on rarrange K dans l'ordre de la transforme de Fourier discrte

Z=real(ifft2(K));

%% on inverse la transforme de Fourier (et on limine la partie imaginaire
%% qui ne contient que des erreurs numriques

end
```

Les effets de crantage sont moins importants, mais un léger flou apparaît. Le bruit a également un peu moins d'influence.

L'avantage de cette procédure, est qu'on peut augmenter la taille d'un nombre fixé de pixels plus facilement: il suffit d'ajouter le nombre de zéros nécessaires:

```
function [Z]=zoomzeropad2(I,m,n)
%% zoom par zero padding: m,n nouvelle taille d'image
%% on suppose la nouvelle image plus grande
[M N]=size(I);
J=fft2(I);
J=fftshift(J);


%% J contient les coefficients de Fourier discret de I, rangs dans l'ordre
%% de sries de Fourier

p=floor((m-M)/2); q=floor((n-N)/2);

K=zeros(M+n,N+n);

K(p+1:p+M,q+1:q+N)=J;

%% on met J au milieu de K
```

```
K=ifftshift(K);

%% on rarrange K dans l'ordre de la transforme de Fourier discrte

Z=real(ifft2(K));

%% on inverse la transforme de Fourier (et on limine la partie imaginaire
%% qui ne contient que des erreurs numriques

end
```