Traitement d'images, TP2
Transformée de Fourier et applications

ENSEEIHT 2EN

http://oberlin.perso.enseeiht.fr/teaching.html

1 Transformée de Fourier des images

1.1 Images synthétiques

On s'intéresse ici à la transformée de Fourier d'un rectangle :

$$f = \chi_{[-r_1,r_1]} \times \chi_{[-r_2,r_2]}.$$

Rappelez rapidement l'expression de la transformée de Fourier de f. Le code suivant définit un rectangle discret et affiche sa TFD.

```
N = 256;r1 = 10;r2 = 22;

% Carré
tmp1 = [zeros(N/2-r1,1); ones(2*r1,1); zeros(N/2-r1,1)];
tmp2 = [zeros(N/2-r2,1); ones(2*r2,1); zeros(N/2-r2,1)];
f = tmp1 * tmp2';
figure;imshow(f,[]);

% Transformée de Fourier
ft = fftshift(fft2(f));

% Représentation TF, échelle normale
figure;imshow(abs(ft),[]);
figure;imshow(log(1+abs(ft)),[]);

% Représentation de la phase
figure;imshow(angle(ft),[]);
```

Que fait la fonction **fftshift** ? Quelle symétrie peut-on observer dans la transformée de Fourier ?

1.2 Images réelles

Récupérez les deux images du jour à `http://oberlin.perso.enseeiht.fr/cours/images_TP2.zip`.

Exercice 1

◇ Représentez le spectre de l'image "lena", avec une échelle logarithmique. Pourquoi observe-t-on des droites sur les axes ?

◇ Pour les supprimer, on peut utiliser une fonction fenêtre, par exemple la fonction suivante, définie sur $[-\pi, \pi]^2$:

$$f(x) = \frac{\cos(x_1)+1}{2} \cdot \frac{\cos(x_2)+1}{2}.$$

Construisez une telle fenêtre discrète, de mêmes dimensions que l'image, et visualisez le spectre de l'image fenêtrée. Quelles structures voit-on toujours sur le spectre ? À quoi sont-elles dûes ?

Exercice 2 Phase et module

◇ Charger également l'image **barbara**. Représentez les images obtenues en sélectionnant le module (**abs**) de la TF de Lena et la phase (**angle**) de la TFD de Barbara, et vice-versa. Qu'en concluez-vous ?

◇ Plus généralement, que peut-on dire sur la reconstruction par TF inverse lorsque le module de la TF (resp. la phase) est faiblement perturbé ? On pourra tester avec un faible bruit Gaussien (**randn**).

2 Échantillonnage et interpolation

On s'intéresse à présent au moyen de sous-échantillonner une image numérique, et à l'opération inverse, qui peut être vue comme de l'interpolation ou du sur-échantillonnage.

Exercice 3 Sous-échantillonnage naïf

◇ Charger l'image **barbara**, et effectuer un sous-échantillonnage "brut", en ne gardant qu'un pixel sur 4 dans chaque dimension. De combien a-t-on réduit l'image ?

◇ À partir de la version sous-échantillonnée, reconstruire une image de taille originale en insérant autour de chaque pixel des pixels de même valeur. On pourra utiliser la commande **kron**. Calculer l'erreur quadratique de reconstruction.

◇ À partir de la version sous-échantillonnée, reconstruire l'image taille réelle par interpolation bi-cubique (**imresize)**, et calculer l'erreur correspondante. Comparer avec la méthode précédente.

Exercice 4 Fenêtrage fréquentiel

◇ Reprendre l'image *Barbara*, et réduire sa taille d'un même facteur que précédemment, en utilisant un fenêtrage fréquentiel : on ne sélectionne que les coefficients de Fourier basse-fréquence.

◇ Pour reconstruire l'image originale, il suffit d'insérer des coefficients nuls dans le domaine de Fourier. Tester cette reconstruction, évaluer et comparer avec la méthode précédente. Quel est le principal défaut de cette technique ? Comment l'expliquer ?

◇ Proposer un moyen de supprimer ce défaut, quitte à augmenter un peu l'erreur de reconstruction, et l'implémenter.

3 Filtrage des images et applications

3.1 Filtres à réponse impulsionnelle finie

La Toolbox Image Processing permet de créer des filtres à réponse impulsionnelle finie, et de visualiser leur réponse fréquentielle. Exécutez le code suivant et analysez-le, en allant chercher les fonctions dans l'aide.

```
% Filtre idéal
Hd = zeros(11,11); Hd(5:7,5:7) = 1;
[f1,f2] = freqspace(11,'meshgrid');
mesh(f1,f2,Hd), axis([-1 1 -1 1 0 1.2]), colormap(jet(64))

% Filtre correspondant
h = fsamp2(Hd);
figure, freqz2(h,[32 32]), axis([-1 1 -1 1 0 1.2])

% Filtre lissé
h2 = fwind1(Hd,hamming(11));
figure, freqz2(h2,[32 32]), axis([-1 1 -1 1 0 1.2])
```

Une fois le filtre créé, on peut procéder au filtrage de l'image par les commandes **imfilter, filter2** ou **conv2**.

Exercice 5 Filtrage pour le sous-échantillonnage

◇ On reprend le problème de la section précédente. Pour éviter l'aliasing, on va filtrer dans le domaine spatial avant de sous-échantillonner. Appliquer cette opération avec les filtres **h** et **h2**, puis reconstruire l'image taille réelle par interpolation. Quelle différence observe-t-on entre les deux filtres ?

◇ Comparer avec les résultats précédents, en qualité de reconstruction et en coût de calcul.

3.2 Quelques filtres particuliers

Exécuter le code suivant. Observer la forme spatiale et fréquentielle du filtre. À quoi sert-il ? À quoi doit-on veiller lorsqu'on applique ce filtre sur une image ?

```
I = imread('moon.tif');
h = fspecial('unsharp');
I2 = imfilter(I,h);
figure;imshow(I);title('Image originale')
figure;imshow(I2);title('Image filtree')
```

Matlab permet de définir des "filtres" non-linéaires avec la commande **nlfilter**.

Exercice 6 Implémenter le filtre médian et le filtre de point milieu, les tester et les comparer sur l'image Lena, en faisant varier la taille du voisinage. Quelle peut être l'utilité de ces filtres ?