UPE ——— E.S.I.E.E.	Traitement du signal Manipulations sur des images	ISBS

Remis par M. J.-F. BERCHER

ÉNONCÉ

Ce TD-TP a pour but d'illustrer certaines notions certaines notions de traitement du signal, appliquées sur des objets 2D ; on aborde ainsi le traitement d'images. En particulier, on abordera les problèmes de représentation et de filtrage (dans le domaine spatial et dans le domaine transformé) (passe-bas et passe haut). Par la suite, on examinera des problèmes d'échantillonnage, voire de restauration d'images. Les manipulations se feront sous Python. Vous récupérerez les fichiers adéquats sur la page Moodle de l'unité, ou sur la page web de l'enseignant. On utilisera donc à nouveau Python, avec les bibliothèques scientifiques scipy et numpy, ainsi que les modules scipy.signal et scipy.ndimage. Quelques indications sur les commandes et les fonctions utiles sont fournis dans l'énoncé. Pour le reste, il vous faudra utiliser l'enseignant et la documentation en ligne...

Scripts et images Matlab fournis :

Votre serviteur, dans le but noble de vous faciliter tâche sans sacrifier la compréhension, a préparé quelques fonctions utiles, notamment :

- `rect2` – crée un rectangle centré en 2D
- `filtre_passebande_2d` – crée un filtre passe-bande en 2D (dans le domaine fréquentiel)
- `showfft2` – affiche une image d'une TF 2D, correctement centrée et normalisée
- `mesh` – affiche une représentation "3D" d'un objet

Pour lire un fichier image, vous utiliserez la fonction `imread`

Pour afficher une image en niveaux de gris, vous pourrez utiliser

```
imshow(S,cmap='gray',origin='upper')
```

I. REPRÉSENTATION FRÉQUENTIELLE - FILTRAGE EN FRÉQUENCE

— Une petite sinusoïde

1. Créez une sinusoïde 2D d'équation $f(x,y) = \sin(2\pi(f_x x + f_y y))$, avec f_x et f_y choisis entre 0.02 à 0.2, sur $N \times N$ points (par exemple $N = 512$). Pour l'implantation, vous pourrez utiliser une boucle `for`, ou une double liste comprehension, ou encore la belle fonction `fromfunction()` et une fonction `lambda`. Visualisez le résultat en utilisant la fonction `imshow`.
2. calculez la Transformée de Fourier 2D de $f(x,y)$ (en utilisant la fonction fft2) et visualisez le module (fonction abs) du résultat, via `showfft2` . Quelles sont les significations des axes ? Quelles sont les fréquences spatiales de la sinusoïde ? Expérimenter pour plusieurs fréquences f_x, f_y.
3. Montrez "théoriquement" et vérifiez sur machine que la transformée de Fourier 2D peut être obtenue comme la succession de transformées de Fourier 1D (fonction fft) appliquées sur les lignes puis les colonnes (ou inversement). Vous utiliserez le fait que la fonction `fft` prend un paramètre axis qui est la dimension sur laquelle est calculée la `fft`

— Barbara

1. Chargez l'image de Barbara, par `imread('barbara.png')` et visualisez la.
2. Visualisez la représentation en fréquence `showfft2` , en échelle logarithmique (prendre `log(abs())` !)
3. Filtrez cette image à l'aide d'un filtre de réponse en fréquence rectangulaire (utilisez la fonction `rect2`), pour des rectangles de demi-largeur 40, 80 ,100. Visualisez les différentes images obtenues – filtrées passe-bas, ainsi que les différences à l'image de départ. Observations, conclusions.
4. Construisez une réponse en fréquence qui élimine sélectivement les fréquences situées autour des points (46,54) et (-70,79), par exemple sur un voisinage de $\pm$ 10 points. Pour ce faire, vous utiliserez le filtre passe-bande `filtre_passebande_2d` et créerez un réjecteur de fréquence par `1-filtre_passebande_2d`. Réexaminez la TF2D de Barbara, graduée en points, afin de comprendre ce que vous faites. Appliquez ce filtre à l'image de départ et interprétez le résultat obtenu dans le domaine spatial. *Observez la nappe* ! Visualisez également l'image différence.

II. FILTRAGE PAR CONVOLUTION

La fonction qui sera utile pour cette partie est la fonction `convolve` de scipy.ndimage (appel par `ndi.convolve`) si ndimage a été importé sous le nom de ndi. On utilisera également `standard_normal` pour ajouter un peu de bruit gaussien (facteur de 0.1) ; ou `saltpepper` pour du bruit en poivre et sel.

1. Vous débuterez par l'implantation d'une convolution à deux dimensions, en comprenant les lignes suivantes :

```
h=ones((2*ll+1,2*ll+1)) # h la RI
for m in range(ll,M-ll):
    for n in range(ll,N-ll):
        B_filtered[m,n]=sum(sum(h*B[m-ll:m+ll+1,n-ll:n+ll+1]))
```

2. Effectuer un filtrage passe-bas (h constant sur une demi largeur de 3 à 10) de l'image de Barbara, et examinez le résultat.
3. Reprenez ce filtrage en utilisant cette fois-ci la fonction `ndi.convolve`. Examinez l'effet du filtrage avec un bruit gaussien ou un bruit en poivre et sel. Vérifiez qu'il s'agit bien d'un filtrage passe-bas en visualisant la fonction de transfert en fréquence – utilisez un zero-padding lors du calcul de la TF2D, `fft2(h,s=(1000,1000))` ; éventuellement, utiliser la fonction mesh pour la représentation).
4. Sur l'image de Barbara, puis sur l'image des cellules, ou de bactéries. Testez un gradient de Prewit ou de Sobel de réponses impulsionnelles

```
dx=np.array([[1.0, 0.0, -1.0],[1.0, 0.0, -1.0],[1.0, 0.0, -1.0],])
# dx=np.array([[1.0, 0.0, -1.0],[2.0, 0.0, -2.0],[1.0, 0.0, -1.0],]) #Sobel
dy=np.transpose(dx)
```

   appliqué aux deux directions (x,y), par `convolve` et construisez une carte de direction.

   NB : si Dx et Dy sont les images de gradient obtenues dans les directions x, y, la carte d'amplitude est $\sqrt{Dx.^2 + Dy.^2}$Le module ndimage contient un certain nombre de filtres prédéfinis, par exemple `scipy.ndimage.filters.sobel`. Segmentez le résultat en seuillant la carte d'amplitude. Examinez en particulier l'effet sur les contours et sur un bruit additif (gaussien puis poivre et sel).