TP 5 Traitement du signal et des images dans le domaine de Fourier

5.1 Objectifs du TP

- étude et familiarisation avec les fonction offertes par la bibliothèque Python *scipy* pour le traitement du signal et des images dans le domaine fréquentiel.

5.2 Introduction

Le domaine de fréquence peut être obtenu via la *transformation de Fourier (TF)***.** Dans ce domaine, pour caractériser un phénomène, nous ne pensons pas en termes d'intensité de signal ou des fonctions luminance associées à un pixel mais en termes de fonctions périodiques sinusoïdales, des fréquences, amplitude et phase différentes. Chaque signal ou image peut être décomposé sous cette forme.

Le passage du domaine temporel/spatial dans le domaine fréquentiel se fait en utilisant la transformée de Fourier numérique directe (*TF*):

$$F[m] = \frac{1}{N}\sum_{k=0}^{N-1} x[k]e^{-2\pi jkm/N}$$

$$F[m,n] = \frac{1}{MN}\sum_{k=0}^{N-1}\sum_{l=0}^{M-1} U[k,l]]e^{-2\pi j(\frac{km}{N}+\frac{ln}{M})} \quad (1)$$

Le passage du domaine fréquentiel au domaine temporel ou spatial se fait, sans pertes, en utilisant la transformation du Fourier inverse *(TFI)*.

$$x[k] = \sum_{m=0}^{N-1} F[m]e^{2\pi jkm/N}$$

$$U[k,l] = \sum_{k=0}^{N-1}\sum_{l=0}^{M-1} F[m,n]e^{2\pi j(\frac{km}{N}+\frac{ln}{M})} \quad (2)$$

En général *F [u]* et *F[m,n]* seront des quantités complexes même si les données d'origine sont purement réelles.

Une hypothèse importante à noter est que la transformation de Fourier suppose que les signaux sont stationnaires, ce qui signifie que les processus analyses ont les mêmes propriétés statistiques dans le temp.

5.2 Support *scipy* pour le calcul de la transformée de Fourier

Le paquet *scipy* fourni des outils pour le calcul de la transformée de Fourier. Les exemples suivants montrent la modalité d'utilisation des fonctions incluses dans le paquet pour le passage et la représentation dans le domaine fréquentiel des signaux (1D) et des images (2D).

```python
import numpy as np
import os

#os.chdir
import matplotlib.pyplot as plt
from scipy import fft, fftpack
f_ech = 48000
no_echant=480

def genSinus(f0, fs,no_esant):
    t = np.arange(no_esant)/fs
    sinusoid = np.sin(2*np.pi*t*f0)
    return sinusoid


amplitude=genSinus(1000,f_ech,no_echant,)
amplitudeb=0.3*genSinus(15*1000,f_ech,no_echant)

signal_b=amplitude+amplitudeb

y1=fft(amplitude,)
y2=fft(signal_b)# transformee directe

freqs = fftpack.fftfreq(len(y1)) * f_ech # frequences
fig, ax = plt.subplots()

#affichage
ax.stem(freqs, np.abs(y2))
ax.set_xlabel('Frequence [Hz]')
ax.set_ylabel('Amplitude')
ax.set_xlim(0, f_ech / 2)
ax.set_ylim(-5, 110)
```

Fig.1 Séquence de code Python pour générer une somme de deux sinusoïdes, le calcul de la TF et la représentation du signal dans le domaine de Fourier

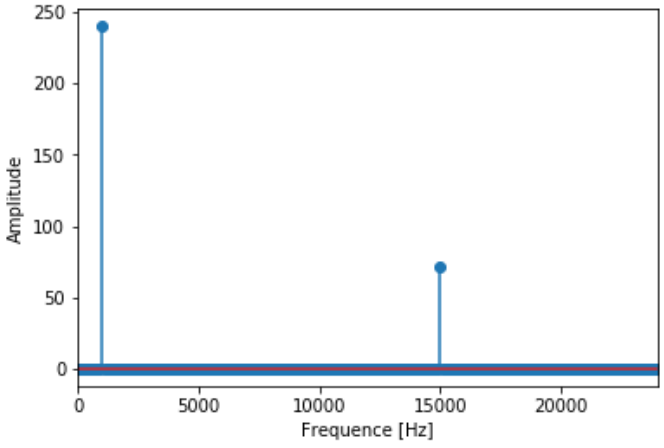

Fig.2 Spectre d'amplitudes du signal

Le domaine fréquentiel permet l'identification des composantes spectrales pour des signaux qui sont difficiles à interpréter dans le domaine temporel ou spatial.

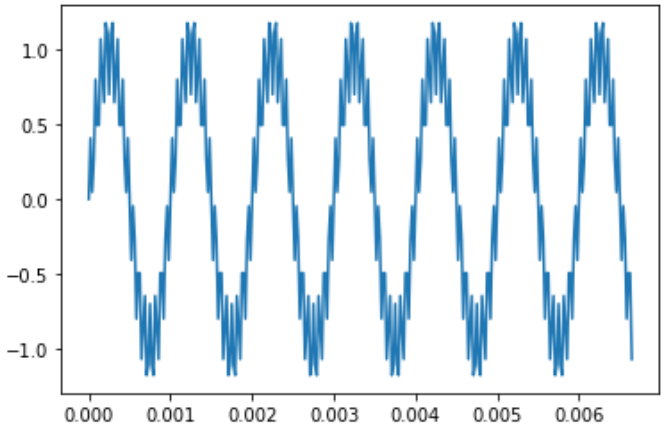

Fig.3 Représentation temporelle du signal en Fig.1

```python
# -*- coding: utf-8 -*-
"""
Created on Sun Oct 20 13:01:46 2019

@author: romi
"""

import numpy as np
import matplotlib.pylab as plt
import os
from scipy import fftpack

def rgb2gray(rgb):
    r, g, b = rgb[:,:,0], rgb[:,:,1], rgb[:,:,2]
    gray = 0.2989 * r + 0.5870 * g + 0.1140 * b
    return gray

os.chdir("C:\\Users\\romi\\Desktop\\Python_ scripts")
im = plt.imread("fruits.png")
im2 =rgb2gray(im)
imgplot = plt.imshow(im2, cmap = plt.get_cmap('gray'))

im_fft = fftpack.fft2(im2)
im_fft = np.fft.fftshift(im_fft)
plt.imshow(20*np.log10(abs(im_fft)),cmap = plt.get_cmap('gray'))
```

Fig.4 Séquence de code Python qui permet la représentation dans le domaine de Fourier du spectre d'amplitude d'une image.

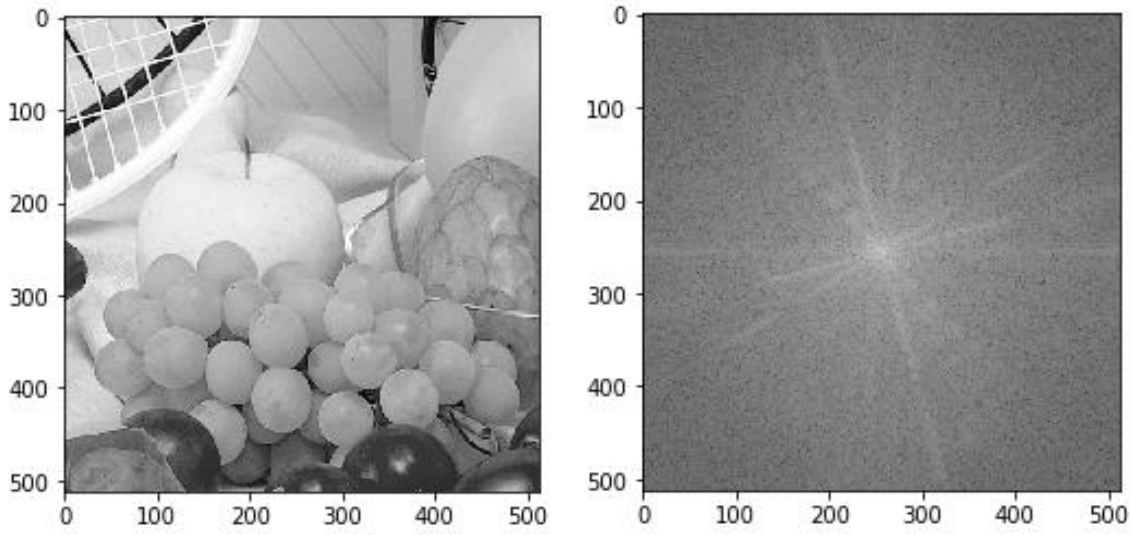

Fig.5 Image 2D en niveaux de gris et spectre d'amplitudes associé

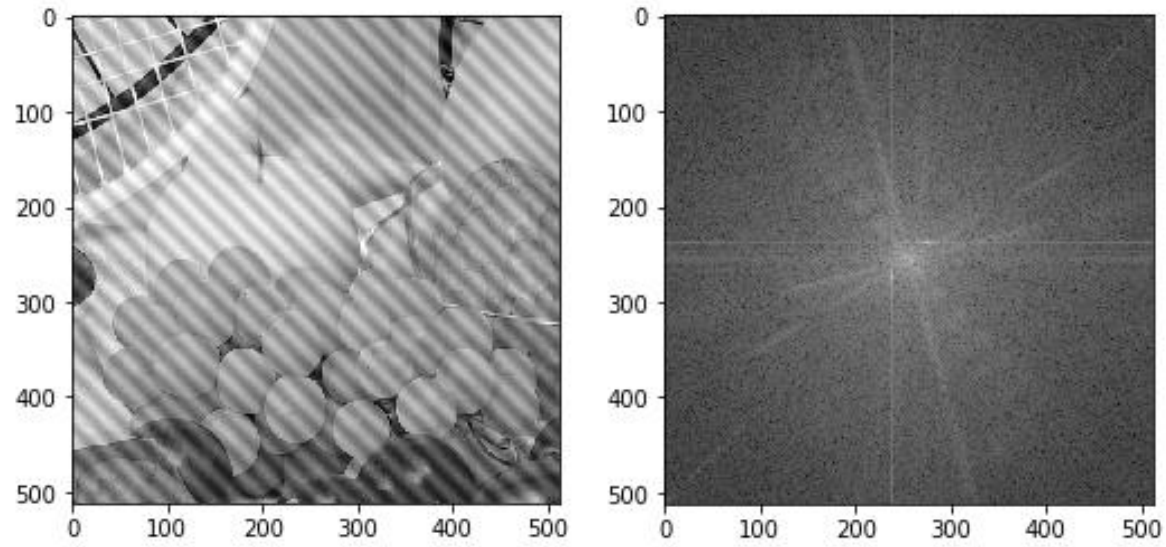

Fig.6 Image 2D en niveaux de gris dégradée et spectre d'amplitude associé

Dans la représentation des images dans le domaine de Fourier chaque centre concentrique centrée en (M/2, N/2) représente une fréquence spatiale.

5.3 Filtrage dans le domaine de Fourier

Comme illustrée précédemment, le passage dans le domaine de Fourier permet de mettre en évidence toutes les composantes spectrales d'un signal. Le filtrage dans le domaine de Fourier revient donc à réduire l'amplitude des composantes indésirables.

Pour le signal en Figure 1 la séquence suivante de code Python pourra conduire à l'élimination de la composante spectrale de 15KHz.

```python
import numpy as np
import os

#os.chdir
import matplotlib.pyplot as plt
from scipy import fft, ifft, fftpack
f_ech = 48000
no_echant=480

def genSinus(f0, fs,no_esant):
    t = np.arange(no_esant)/fs
    sinusoid = np.sin(2*np.pi*t*f0)
    return sinusoid


amplitude=genSinus(1000,f_ech,no_echant,)
amplitudeb=0.3*genSinus(15*1000,f_ech,no_echant)


signal_b=amplitude+amplitudeb

y1=fft(amplitude)
y2=fft(signal_b)# transformee directe

fil_fft = y2.copy()
fil_fft[np.abs(fil_fft)<100]=0
signal_fitre=ifft(fil_fft);

freqs = fftpack.fftfreq(len(y1)) * f_ech # frequences
fig, ax = plt.subplots()


#affichage frequence
ax.stem(freqs, np.abs(fil_fft))
ax.set_xlabel('Frequence [Hz]')
ax.set_ylabel('Amplitude')
ax.set_xlim(0, f_ech / 2)

#affichage temps
t=np.arange(amplitude.size)/f_ech
plt.plot(t, signal_fitre, linewidth=3, label='Signal filtre')
```

Fig.7 Filtrage dans le domaine de Fourier par élimination des composantes spectrales

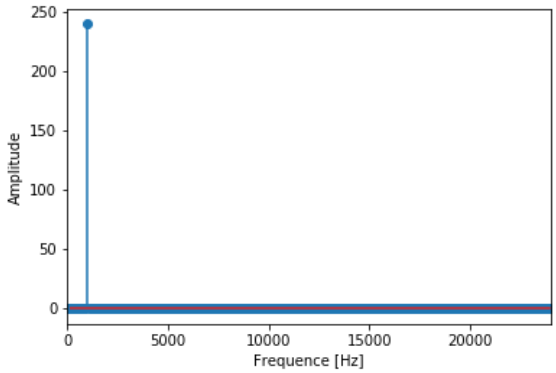

Fig.8 Spectre du signal filtre en utilisant le code en figure 7.

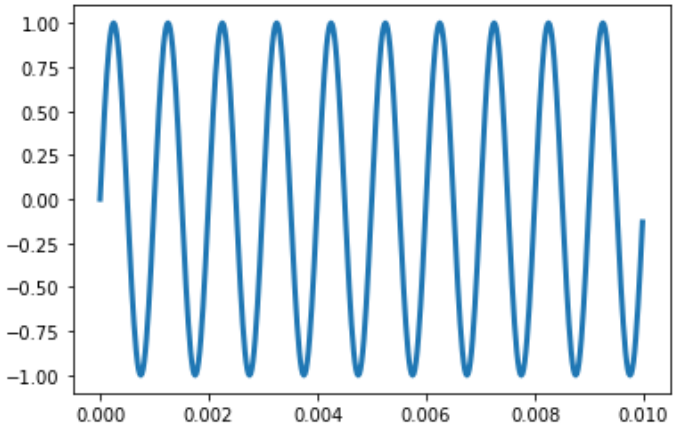

Fig.9 Représentation temporelle du signal filtré

Remarque : l'exemple antérieur est à titre illustratif. En pratique *une coupure aussi brutale dans l'espace fréquentiel ne permet pas de contrôler la distorsion du signal.*

En pratique, les filtrer à utiliser dans le domaine de Fourier sont soit des filtres Gaussiens soit des filtrer de type Butterworth. La synthèse de ces filtres se fait par spécification de la fonction de transfert et application du théorème de la convolution :

$$TF(x * h) = TF(x)TF \cdot (h) \tag{3}$$

Une opération de convolution dans le domaine temporel peut être donc implémente de manière efficiente dans le domaine de Fourier par :

$$(x * h) = TFI\{TF(x) \cdot TF(h)\} \tag{4}$$

Implémentation du filtrage Gaussien dans le domaine de Fourier

Les figures suivantes montrent le résultat d'une opération de filtrage avec un filtre Gaussien sur une série de temps illustrant le prix de la monnaie Bitcoin. Le filtre Gaussien de moyenne pondérée et une alternative meilleure au filtre moyenneur causal présenté en cours.

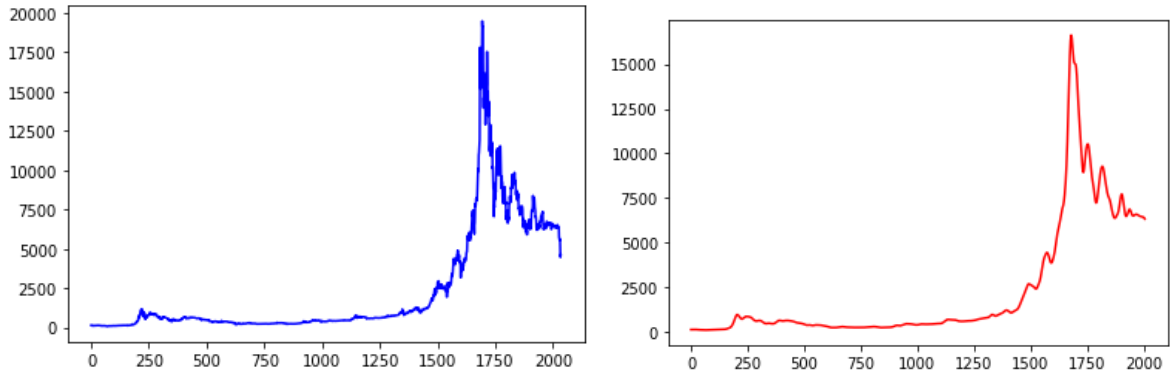

Fig.10 Prix de la monnaie Bitcoin (non-filtre -gauche, filtre avec un filtre Gaussien-droite)

```python
@author: romi
"""
import numpy  as np
import os
import pandas as pd
import matplotlib.pylab as plt
from scipy import fftpack
import scipy

def fft_domaine_spectral (signal, filtre):
    fft_signal = fftpack.fft(signal)
    fft_filtre = fftpack.fft(filtre)
    return fftpack.ifft(fft_signal * fft_filtre)


os.chdir("C:\\Users\\romi\\Desktop\\Python_ scripts")
df = pd.read_csv('bitcoin1.csv')
x=df.loc[: , "price(USD)"]

t=np.arange(x.size)
plt.plot (t,x,"b")

filtre = scipy.signal.gaussian(M=31, std=6)
filtre /= filtre.sum()

pad=np.zeros (x.size)
pad[:filtre.shape[0]]=filtre

y = fft_domaine_spectral(x, pad)

M=len(filtre)

y_s=y[M-1:]

t=np.arange(y_s.size)
#plt.plot (t,y_s,"r")
```

Fig.11 Code Python utiliser pour lisser les prix de la monnaie Bitcoin

Simulation du flou de mouvement intra-scan et filtrage inverse dans le domaine de Fourier

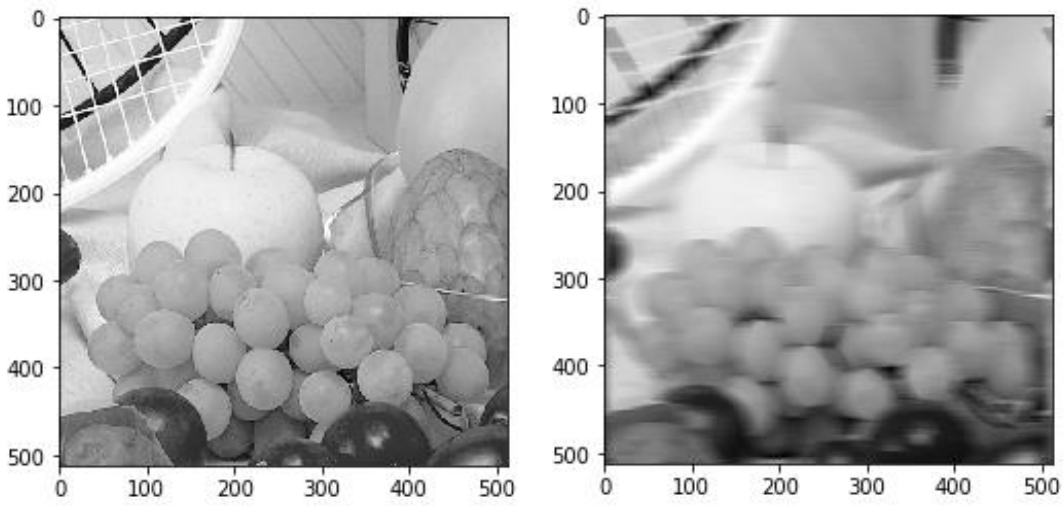

Fig.12 Flou de mouvement intra-scan simulé

```
import numpy as np
import matplotlib.pylab as plt
import os
from scipy import fftpack as fp

def rgb2gray(rgb):
    r, g, b = rgb[:,:,0], rgb[:,:,1], rgb[:,:,2]
    gray = 0.2989 * r + 0.5870 * g + 0.1140 * b
    return gray

os.chdir("C:\\Users\\romi\\Desktop\\Python_ scripts")
im = plt.imread("fruits.png")

img =rgb2gray(im)

size = 25
kernel = np.zeros((size, size))

kernel[int((size-1)/2), :] = np.ones(size)
kernel = kernel / size

kernel1=np.zeros([512,512])

lim_g=int(np.floor(im.shape[0]/2-size/2))
lim_d=int(np.floor(im.shape[0]/2+size/2))
lim_h=int(np.floor(im.shape[1]/2-size/2))
lim_b=int(np.floor(im.shape[1]/2+size/2))


kernel1[lim_g:lim_d, lim_h:lim_b]=kernel

im_freq = fp.fft2(img)
kernel_freq = fp.fft2(fp.ifftshift(kernel1))
resultat=im_freq*kernel_freq

freq_kernel = fp.fft2(fp.ifftshift(kernel1))
im_flou = fp.ifft2(resultat).real
im_flou = im_flou / np.max(im_flou)

plt.imshow(im_flou,cmap = plt.get_cmap('gray'))

```

Fig.12 Séquence de code Python pour simuler le flou de mouvement intra scan

Le mouvement intra-scan a été simule tout en parcourant les étapes suivantes :

1) Définition d'un noyau flou de mouvement intra-scan d'une forme donnée (*kernel* dans fig.12) dans le domaine spatial
2) Calcul de la transformée de Fourier du noyau (*H(u,v) -kernel_freq* dans fig.12).
3) Convolution du noyau centré dans le point de coordonnées (*M/2,N/2*) avec l'image en utilisant le théorème de la convolution.
4) Obtention de l'image dans le domaine spatial par la transformée de Fourier inverse.

La connaissance du noyau de dégradation *H(u,v)* permet d'effectuer une opération de filtrage inverse dans le domaine de Fourier, décrite par l'équation :

$$I = TFI\{\frac{TF(U\}}{\varepsilon+H(u,v)}\}, \quad (5)$$

avec ε une constante positive très petite (10^{-6}) utilisée pour éviter la division avec 0, *U* l'image dégradée, *H(u,v)* la fonction de transfert dans le domaine de Fourier qui modélise le processus de dégradation et *I* l'image restaurée.

5.4 Exercices (CR à rendre par email)

1) Reprenez la séquence de code dans Fig. 7 et insérez des commentaires illustrant le rôle des fonctions *scipy* permettant de passer et représenter le signal dans le domaine fréquentiel et d'éliminer la composante de 15kHz.

2) Quelles sont les modifications qui doivent être apportées au code pour permettre de retenir dans la séquence filtrée de la composante de 15kHz?

3) Analysez la modalité d'implémentation de la méthode de filtrage dans le domaine de Fourier présentée dans la figure 11 et modifier l'exemple présenté en cours pour permettre l'implémentation de l'opération de filtrage dans le domaine fréquentiel.

4) Analysez l'exemple donnée dans Fig.12 et rajoutez des commentaires illustrant le rôle de chaque instruction Python.

5) Rajoutez au code de Fig.12 une séquence d'instructions permettant de supprimer l'effet du flou par filtrage inverse. Vérifiez le résultat obtenu; quelles sont les modifications apportes a l'image restaurée par rapport à l'image initiale?