Feuille de TP n°4

Licence 3 MApI3, module **Signal, Fourier, image**

Théorie de l'échantillonnage

Exercice 1 – Filtre d'échantillonnage de Shannon. On rappelle que lors de l'échantillonnage d'une image numérique de taille $N \times N$ d'un facteur K, le filtre de Shannon consiste à supprimer toutes les fréquences supérieures à la fréquence maximale de la transformée de Fourier de la nouvelle image.

1) Implémenter une fonction `undersample1(u, K)` qui renvoie l'image u échantillonnée d'un facteur K sans filtrage. Rappeler pourquoi il est nécessaire de filtrer l'image avant échantillonnage, et générer un exemple d'image sur lequel le phénomène est visible.

2) Exprimer mathématiquement le filtre de Shannon.

3) Implémenter en Python une fonction `undersample2(u, K)` qui effectue l'échantillonnage de l'image u d'un facteur K après lui avoir appliqué le filtre de Shannon correspondant.

4) Tester avec l'image `barbara.png` pour les valeurs de `K` suivantes : $1, 2, 3, 4, 5, 6$.

5) Tester pour les mêmes valeurs de `K` avec l'image u de taille 128×128 définie par

$$u_{m,n} = (-1)^{\left\lfloor \frac{m}{32} \right\rfloor + \left\lfloor \frac{n}{32} \right\rfloor}, \tag{1}$$

où $\left\lfloor \frac{a}{b} \right\rfloor$ désigne le quotient de la division euclidienne de a par b.

6) Décrire qualitativement les résultats obtenus.

Exercice 2 – Compromis temps/fréquence. On vient de voir que le filtre de Shannon, outre son coût algorithmique relativement élevé, a des effets indésirables. L'alternative vue au TP1, le filtre en moyenne, peut causer des problèmes de flou. Nous allons à présent implémenter le filtre d'échantillonnage réalisant le compromis temps/fréquence vu en TD (filtre barycentrique à support compact qui minimise la distance au filtre de Shannon). Soit u une image numérique de taille $N \times N$, $\Omega \subset [0, N]^2$, on définit les sous-espaces vectoriels :

$$SL_\Omega = \{v : v_{m,n} = 0 \text{ si } (m,n) \notin \Omega\} \qquad BL_K = \left\{ v : \hat{v}_{k,l} = 0 \text{ si } (k,l) \notin \left[-\frac{N}{2K}, \frac{N}{2K} \right]^2 \right\}, \tag{2}$$

et on note respectivement P_Ω et Π_K les projecteurs orthogonaux associés.

1) Rappeler l'expression mathématique de P_Ω et Π_K.

2) Définir en Python une fonction `project_on_BL(u, K)` qui renvoie le projeté de u sur BL_K.

Indication : il est possible en Python d'effectuer la multiplication par un tableau de booléens. Les valeurs `True` et `False` sont alors interprétées respectivement comme 1 et 0. Par exemple, le code suivant va multiplier l'image `lena.png` par la fonction indicatrice de sa moitié inférieure. On pourra s'en inspirer pour effectuer des multiplications par d'autres fonctions indicatrices.

Listing 1 – Multiplication d'une image par une fonction indicatrice

```
from imtools import *
from numpy import *
i = open_image('lena.png')
[X, Y] = meshgrid(range(i.shape[0]), range(i.shape[1]))
i[X, Y] *= X > 256 #Ici: multiplication de i par une fonction indicatrice
display_image(i)
```

3) Dans ce qui suit, pour $R > 0$ on prendra pour Ω le disque de centre $(0, 0)$ et de rayon R :

$$\Omega = \{(x, y) : x^2 + y^2 \leq R^2\} \tag{3}$$

Définir en Python une fonction `project_on_SL(u, R)` qui renvoie la projection de u sur SL_Ω.

4) On rappelle que le filtre vu en TD est donné par $h = \lim_{n\to\infty} h_n$ où (h_n) est la suite de filtres définie par :

$$\begin{cases} h_0 & = \mathbb{1} \\ h_{n+1} & = \psi_\Omega \circ \Pi_K(h_n). \end{cases} \tag{4}$$

Ici ψ_Ω désigne le projecteur convexe sur l'ensemble $\mathcal{F}_\Omega$ des filtres barycentriques à support dans Ω, c'est à dire :

$$\mathcal{F}_\Omega = \{h \in SL_\Omega : 0 \leq h \leq 1 \text{ et } \sum_{m,n} h_{m,n} = 1\} \tag{5}$$

$$\forall h \in SL_\Omega : \psi_\Omega(h) = \frac{\tilde{h}}{\sum_{m,n} \tilde{h}_{m,n}} \qquad \text{où} \qquad \tilde{h}_{m,n} = \begin{cases} 0 & \text{si } h_{m,n} < 0 \\ 1 & \text{si } h_{m,n} > 1 \\ h_{m,n} & \text{sinon.} \end{cases} \tag{6}$$

a) Écrire en Python une fonction `iterate_filter(h, K, R)` qui renvoie $\psi_\Omega \circ P_\Omega \circ \Pi_K(h)$.

Indication : il est possible en Python de faire référence en une seule fois à tous les éléments d'un tableau vérifiant une certaine propriété. L'exemple suivant permet de mettre à zéro tous les pixels d'une image dont la valeur est inférieure à 200. On pourra s'en inspirer pour le calcul de $\tilde{h}$.

Listing 2 – Modification d'une plage de valeurs

```
from imtools import *
from numpy import *
i = open_image('lena.png')
i[i < 200] = 0 #Ici: modification de tous les pixels dont la valeur est inférieure
    à 200
display_image(i)
```

b) Pour le calcul approché du filtre asymptotique, nous allons utiliser un critère d'arrêt. Écrire une fonction `approx_filter(h0, K, R)` qui renvoie le premier filtre h_n tel que $\|h_{n-1} - h_n\|_\infty < \epsilon$ (on prendra $\epsilon = 10^{-5}$).

Indication : lors des appels à `approx_filter`, on devra lui donner comme paramètre h_0 un tableau de la bonne taille ne contenant que des 1. On peut obtenir un tel tableau avec la fonction `numpy.ones((M, N))`.

c) Pour $N = 32$ et $K = 2$, représenter le filtre asymptotique approché pour les valeurs de K suivantes : $1, 2, 4, 6, 8, 10$.

d) Même question pour $K = 3$.

5) Écrire une fonction `undersample3(u, R, K)` qui effectue l'échantillonnage de l'image `u` d'un facteur `K` après lui avoir appliqué le filtre asymptotique approché. La tester avec l'image `barbara.png` pour $K = 2$ et avec les valeurs de R suivantes : $1, 2, 3, 4$. Comparer avec le résultat de `undersample2`. Quelle valeur de R vous semble la mieux adaptée ?

6) Même question pour $K = 3$ et $R \in \{2, 3, 4, 5\}$.

7) Afficher `undersample3(u, R, K)` pour $R = K \in \{2, 3, 4, 5, 6\}$ avec l'image `barbara.png`.

8) Même question avec l'image u utilisée dans l'exercice précédent pour $N = 128$:

$$u_{m,n} = (-1)^{\lfloor \frac{m}{32} \rfloor + \lfloor \frac{n}{32} \rfloor}. \tag{7}$$

9) Décrire qualitativement les résultats obtenus.

Applications

Exercice 3 – Zoom numérique. On s'intéresse à présent au problème inverse de l'échantillonnage : comment augmenter la taille d'une image.

1) Préparer les images de travail de taille 32×32 suivantes :

- u^1 définie comme la restriction de `lena.png` à $\{250, \dots, 281\} \times \{309, \dots, 340\}$,
- $u^2_{m,n} = (-1)^{\lfloor \frac{m}{8} \rfloor + \lfloor \frac{n}{8} \rfloor}$
- $u^3_{m,n} = m$

- $u^4_{m,n} = \cos\left(\pi \dfrac{m+n}{8}\right)$

2) Une première idée pour implémenter un zoom numérique consiste à utiliser une approximation par plus proche voisin. Définir une fonction `oversample1(u, K)` qui renvoie l'image v dont les dimensions sont celles de u multipliées par K, et telle que

$$v_{m,n} = u_{\lfloor \frac{m}{K} \rfloor, \lfloor \frac{n}{K} \rfloor}. \tag{8}$$

3) Une deuxième idée consiste à voir la transformée de Fourier de l'image (avant zoom) comme la transformée de Fourier d'une image plus grande qui a été restreinte (autrement dit, qui a subi un filtrage de Shannon).

a) Définir une fonction `oversample2(u, K)` qui, pour une image u de taille $M \times N$, renvoie l'image v de taille $KM \times KN$ telle que

$$\hat{v}_{m,n} = \begin{cases} \hat{u}_{m,n} & \text{si } (m,n) \in \left\{-\frac{M}{2}+1, \dots, \frac{M}{2}\right\}^2 \\ 0 & \text{sinon.} \end{cases} \tag{9}$$

b) Tester `oversample2` avec les images de travail pour $K = 1, 2, 3, 4$.

4) Enfin une troisième idée consiste à faire un compromis entre localisation fréquentielle et spatiale.

a) Définir une fonction `oversample3(u, K)` qui renvoie l'image obtenue après avoir appliqué le filtre asymptotique approché de l'exercice 2 à l'approximation par plus proche voisin définie à la question **2)**.

b) Tester `oversample3` avec les images de travail pour $K = 1, 2, 3, 4$.

5) Décrire qualitativement les résultats obtenus.

Tomographie

Soit $f\colon \mathbb{R}^n \to \mathbb{R}$ une fonction à n variables (qu'on supposera continue et à support compact pour simplifier) et H un hyperplan de $\mathbb{R}^n$ (un sous-espace vectoriel de dimension $n-1$). Pour $s \in H$, on note $\Delta(s)$ la droite perpendiculaire à H et passant par s. La transformée de Radon de f par rapport à H est définie par :

$$\mathcal{R}_H f\colon \begin{cases} H & \longrightarrow \mathbb{R} \\ s & \longmapsto \displaystyle\int_{t \in \Delta(s)} f(t)\, dt, \end{cases} \tag{10}$$

c'est à dire qu'on intègre f perpendiculairement à H. En identifiant H à $\mathbb{R}^{n-1}$ on obtient une nouvelle fonction à $n-1$ variables.

Cette transformée est utilisée notamment en tomographie pour l'imagerie médicale par rayons X. Dans ce contexte, $n = 3$ et f représente la densité d'un organe qu'on veut étudier. On réalise une série de clichés bidimensionnels en faisant tourner la caméra autour de l'organe à étudier et on obtient son *sinogramme*, c'est à dire l'échantillonnage de $R_H f$ pour un ensemble fini de valeurs de H. La *tomographie* est l'opération consistant à reconstruire la densité tridimensionnelle f de l'organe à partir de son sinogramme.

Les calculs de transformée de Radon étant relativement lourds, nous allons nous limiter au cas où $n = 2$. Dans ces conditions, H est une droite qu'on paramètre par l'angle $\theta \in [0, \pi[$ qu'elle forme avec l'axe des abscisses. On a alors :

$$\forall \theta \in [0,\pi[, \forall s \in \mathbb{R}\colon \mathcal{R}_\theta f(s) = \int_{-\infty}^{+\infty} f(s\cos\theta - t\sin\theta, s\sin\theta + t\cos\theta)\, dt. \tag{11}$$

Exercice 4 – Transformée de Radon. Nous allons commencer par implémenter une transformée de Radon bidimensionnelle approchée pour les images discrètes. Pour cela, on considère une image numérique u de taille $N \times N$ et on va remplacer l'intégrale dans la formule (11) par une somme de Riemann de l'approximation par plus proche voisin de u :

$$\forall \theta \in [0,\pi[, \forall s \in \left[-\frac{N}{2}, \frac{N}{2}\right] : \mathcal{R}u(\theta, s) = \frac{1}{N} \sum_{t=-\frac{N}{2}}^{\frac{N}{2}} u_{\mathrm{round}\left(s\cos\theta - t\sin\theta + \frac{N}{2}\right), \mathrm{round}\left(s\sin\theta + t\cos\theta + \frac{N}{2}\right)}. \tag{12}$$

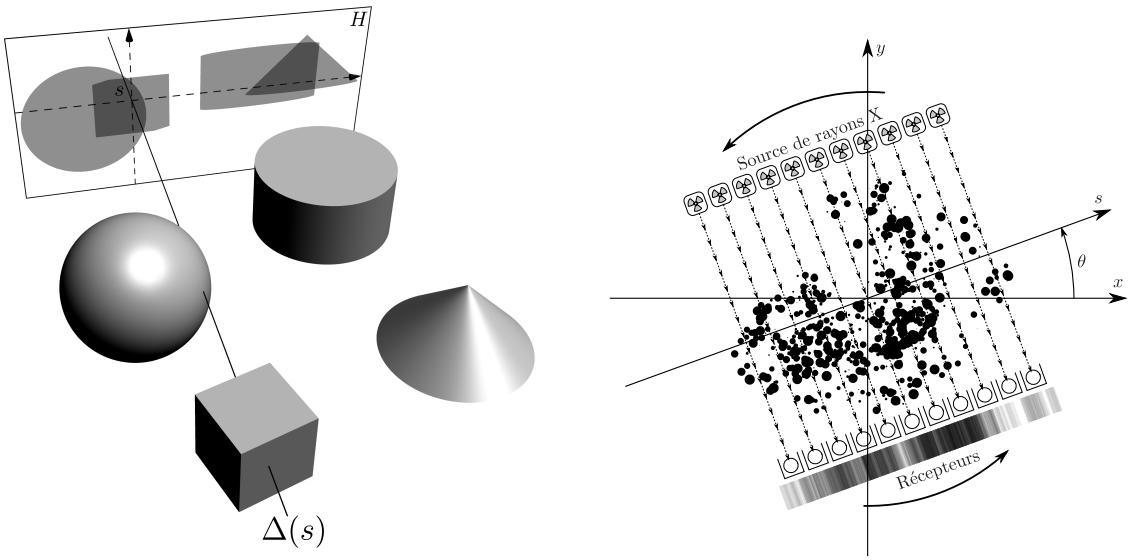

FIGURE 1 – La transformée de Radon. À gauche : en dimension 3. À droite : tomographie par rayons X en dimension 2.

1) La formule (12) n'est valable que si l'on suppose que le support de v est inclus dans le disque de rayon $\frac{N}{2}$ au centre de l'image. Pour garantir que cette condition est réalisée, écrire une fonction `enlarge(u)` qui prend en paramètre une image de taille $N \times N$ et qui renvoie l'image de taille $2N \times 2N$ obtenue en entourant u d'une bande noire de largeur $\frac{N}{2}$.

2) Écrire en Python une fonction `radon(u, M)` qui applique `enlarge` à u et qui renvoie sa transformée de Radon discrète v définie par la formule (12) pour un échantillonnage de l'angle θ sur M valeurs. Plus précisément, on veut que v soit l'image de taille $N \times M$ vérifiant :

$$\forall (n,m) \in \{0,\dots,N-1\} \times \{0,\dots,M-1\} \colon v_{n,m} = \mathcal{R}u\left(\frac{m\pi}{M}, n - \frac{N}{2}\right). \tag{13}$$

Important : le calcul de $\mathcal{R}u$ ayant une complexité élevée, il est recommandé d'utiliser les fonctionnalités de vectorisation introduites par la fonction `numpy.meshgrid` de Python. On pourra remarquer que le calcul de $v_{m,n}$ consiste essentiellement à implémenter une somme du type $\sum_{t=-\frac{N}{2}}^{\frac{N}{2}} u_{k,l}$ où k et l sont des indices qui dépendent de m, n et t. Il y a deux problèmes techniques à résoudre lors de l'implémentation vectorisée de cette somme :

- la formule (12) définit les indices (k,l) à l'aide de la fonction `round`, or celle-ci renvoie des nombres flottants. Il faudra donc convertir le tableau d'indice en nombres entiers à l'aide de la fonction `astype(int)` ;
- certains indices (k,l) vont se retrouver à l'extérieur de l'intervalle $\{0,\dots,N-1\}^2$. Ici, il ne faut pas périodiser l'image comme on le fait d'habitude. Les termes correspondants de la somme doivent être remplacés par des zéros. Une manière de s'en assurer est de remplacer ces indices (k,l) par des pixels situés sur le bord de l'image.

Le code suivant donne une structure possible de l'algorithme de sommation.

Listing 3 – Squelette de l'algorithme de sommation

```
v = zeros((N, M))
[n, m] = meshgrid(range(N), range(M))
for t in range(-N/2, N/2 + 1):
    k = round(...).astype(int) # calcul des indices k en fonction de t, m et n
    l = round(...).astype(int) # calcul des indices l en fonction de t, m et n
    k[k < 0] = 0               # ajustement des indices situés trop haut
    k[k >= N] = N - 1          # ajustement des indices situés trop bas
    l[l < 0] = 0               # ajustement des indices situés trop à gauche
    l[l >= N] = N - 1          # ajustement des indices situés trop à droite
    v[n, m] += u[k, l]
```

3) Tester votre algorithme avec les images de taille 64×64 suivantes :

$$u^1_{m,n} = \mathbb{1}_{(m+32)^2+(n-32)^2<500} \tag{14}$$
$$u^2_{m,n} = \mathbb{1}_{|n-32|<2} \tag{15}$$
$$u^3_{m,n} = \mathbb{1}_{|64-m-n|<2} \tag{16}$$
$$u^4_{m,n} = \mathbb{1}_{|32-m-n|<2} \tag{17}$$
$$u^5_{m,n} = \mathbb{1}_{|64-m\sqrt{3}-n|<2} + \mathbb{1}_{|50-n|<2} + \mathbb{1}_{|m-n\sqrt{3}|<2} \tag{18}$$

4) Décrire qualitativement les résultats obtenus.

Exercice 5 – Transformée de Radon inverse. On s'intéresse à présent au problème de la reconstruction d'une image connaissant sa transformée de Radon.

1) Dans la formule (11), on peut constater que $\mathcal{R}_\theta f(s)$ s'écrit comme l'intégrale par rapport à t de $f \circ R(s,t)$ où R désigne la rotation d'angle θ. À l'aide du changement de variables $(x,y) = R(s,t)$ (il faut prendre $dx\,dy = ds\,dt$), montrer que

$$\forall \lambda \in \mathbb{R} : \; : \hat{f}(\lambda\cos\theta, \lambda\sin\theta) = \widehat{\mathcal{R}_\theta f}(\lambda). \tag{19}$$

2) En effectuant le changement de variables $(x,y) = (\lambda\cos\theta, \lambda\sin\theta)$ pour $(\lambda,\theta) \in \mathbb{R} \times [0,\pi]$ dans la formule de reconstruction de f à partir de $\hat{f}$ (il faut cette fois prendre $dx\,dy = |\lambda| d\lambda\,d\theta$), montrer qu'on a la formule de reconstruction :

$$f(x,y) = \int_0^\pi \left(\int_{\mathbb{R}} |\lambda| \widehat{\mathcal{R}_\theta f}(\lambda) e^{i\lambda(x\cos\theta + y\sin\theta)} d\lambda \right) d\theta = \int_0^\pi Q_\theta(x\cos\theta + y\sin\theta) d\theta, \tag{20}$$

où Q_θ est la transformée de Fourier inverse de la fonction $\lambda \mapsto |\lambda| \widehat{\mathcal{R}_\theta f}(\lambda)$.

3) Dans le cas des images discrètes, nous allons définir une transformée de Radon inverse approchée en remplaçant là encore l'intégrale dans la formule (20) par une somme. Si u est une image de taille $N \times M$ représentant une transformée de Radon, on définit les images Q et Λ de taille $N \times M$ par

$$\Lambda_{n,m} = |\,\mathrm{freq}(n)| \tag{21}$$
$$Q = \mathrm{ifft}_{\text{colonnes}}(\Lambda \cdot \mathrm{fft}_{\text{colonnes}}(u)). \tag{22}$$

La transformée de Radon inverse sera l'image v de taille $N \times N$ définie par

$$\forall (x,y) \in \{0,\dots,N-1\}^2 : \; v_{x,y} = \frac{\pi}{M} \sum_{m=0}^{M-1} Q_{\mathrm{round}\left(\frac{N}{2} + (x-\frac{N}{2})\cos\frac{m\pi}{M} + (y-\frac{N}{2})\sin\frac{m\pi}{M}\right), m}. \tag{23}$$

Définir une fonction `iradon(u)` qui renvoie l'image v selon cette définition. On ajustera les indices de sommation en s'inspirant de la fonction `radon`.

Indications : dans la formule (21), $\mathrm{freq}(n)$ est la fréquence associée au mode n pour un vecteur de taille N. Les colonnes de l'image Λ peuvent donc être construites en Python grâce à la fonction `pi * fftfreq(N, 2.0 / N)`. La transformée de Fourier rapide (ou son inverse) sur les colonnes d'une image peut être calculée grâce à la fonction `fft(..., axis = 0)` (ou `ifft(..., axis = 0)`).

4) Tester `iradon` sur les transformées de Radon de l'exercice précédent.

5) Tester `iradon` sur les sinogrammes `brain1.png` et `brain2.png`. Que se passe-t-il si l'on applique un bruit multiplicatif gaussien, centré en 1 et de variance 0.05 aux sinogrammes avant la reconstruction ?

Indication : un tel bruit peut être généré grâce à `numpy.random.normal(1, 0.05, size = (N, M))`.

6) Décrire qualitativement les résultats obtenus.