Exercice 13 (décomposition periodic+smooth)

La périodisation forcée constatée à l'exercice 10 a une incidence forte sur la transformée de Fourier discrète calculée. En effet, les "discontinuités" de l'image au bord du domaine produisent des hautes fréquences parasites dans les directions horizontales et verticales, ce qui se manifeste dans le domaine de Fourier par l'apparition d'une "croix", c'est-à-dire de coefficients forts le long des 2 axes du repère. Pour remédier à ce problème, on peut appliquer la "décomposition $p + s$" à l'image initiale u, c'est-à-dire calculer deux images p (composante périodique de u) et s (composante lisse) telles que $u = p + s$.

1) Expliquez ligne à ligne le code de la fonction `perdecomp` donné ci-dessous :

```
def perdecomp(u):
    ny, nx = u.shape
    u = u.astype('double')
    X = np.arange(nx)
    Y = np.arange(ny)
    v = np.zeros((ny, nx))
    v[ 0,X] =  u[0,X] - u[-1,X]
    v[-1,X] = -v[0,X]
    v[Y, 0] = v[Y, 0] + u[Y,0] - u[Y,-1]
    v[Y,-1] = v[Y,-1] - u[Y,0] + u[Y,-1]
    fx = np.tile(np.cos(2.*pi*X/nx), (ny,1))
    fy = np.tile(np.cos(2.*pi*Y/ny), (nx,1)).T
    fx[0,0] = 0  # avoid division by 0 in the line below
    s = np.real(np.fft.ifft2(np.fft.fft2(v)*0.5/(2.-fx-fy)))
    p = u-s
    return p, s
```

2) Appliquez ensuite cette fonction à l'image `u` de votre choix par

```
import imtools as im
p, s = im.perdecomp(u)
```

puis visualisez les deux composantes `p` et `s` au moyen de la fonction `im.View()`.

Vérifiez que la décomposition est bien consistante, c'est-à-dire que l'on retrouve bien l'image initiale en additionnant pixel à pixel les images `p` et `s`. Observez les différences entre l'image initiale (`u`) et sa composante périodique (`p`). Nous pouvons également fabriquer des versions périodisées pour visualiser les transitions des images `u`, `p` et `s` au bord du domaine :

```
im.View(np.tile(u, (2,2)))
im.View(np.tile(p, (2,2)))
im.View(np.tile(s, (2,2)))
```

Visualisez enfin le (log-)module et la phase de la transformée de Fourier des images `u`, `p`, `s`, ainsi que de la symétrisée de l'image `u` (obtenue en sortie de `im.fsym2(u)`).

Quelles différences observe-t-on entre l'image initiale (`u`) et sa composante périodique (`p`), dans le domaine initial et dans le domaine de Fourier ? Quelles observations suggèrent que les artefacts dus à la périodisation forcée sont bien atténués ? Pourquoi ce traitement est-il plus intéressant que la symétrisation ?

Exercice 14 (propriétés de la décomposition periodic+smooth)

Soit $u : \Omega \to \mathbb{R}$ une image discrète (même notations qu'en cours). On définit

$$\partial\Omega = \{\mathbf{x} \in \Omega, W(\mathbf{x}) \neq \emptyset\} \quad \text{et} \quad \mathring{\Omega} = \{\mathbf{x} \in \Omega, W(\mathbf{x}) = \emptyset\}.$$

Montrer les propriétés suivantes:

1. $u - \mathrm{per}(u)$ ne dépend que de la restriction de u à $\partial\Omega$
2. $u - \mathrm{per}(u)$ est harmonique (i.e. de laplacien nul) sur $\mathring{\Omega}$
3. per est une bijection linéaire de $\mathbb{R}^{\Omega}$
4. toutes les valeurs propres de per sont réelles et dans l'intervalle $]0, 1]$
5. l'ensemble des points fixes de per est

$$\mathcal{P} = \{u : \Omega \to \mathbb{R},\ \forall \mathbf{x} \in \partial\Omega, \forall \mathbf{y} \in W(\mathbf{x}), u(\mathbf{y}) = u(\mathbf{x})\}.$$

6. per est diagonalisable *(Indication: on rappelle le théorème classique suivant: Si A est une matrice symétrique positive et B une matrice symétrique définie positive de même taille, alors il existe une base orthogonale pour A et orthonormée pour B. Matriciellement, cela revient à dire qu'il existe une matrice P inversible telle que $P^T AP$ est diagonale et $P^T BP = I$)*
7. l'itération de per définit un opérateur per^{∞} qui est une projection sur $\mathcal{P}$.

Exercice 15 (translation sous-pixellique)

Si la translation d'une image d'un vecteur entier ne pose pas de problème, en revanche la translation sous-pixellique est l'exemple typique d'un processus nécessitant une interpolation de l'image. Comme nous l'avons vu en cours, l'interpolation sinc discrète (interpolation de Shannon avec convention de périodicité) est, pour des raisons axiomatiques, le candidat idéal pour réaliser cette transformation.

Testez sur l'image bouc.pgm la translation de vecteur $(1/2, 1/2)$ à l'aide des commandes

```
import numpy as np
import imtools as im
u = im.load('bouc.pgm').astype('double')
v = im.ffttrans(u, 0.5, 0.5)
```

et comparez le résultat à l'original (on pourra visualiser dans un premier temps l'image des différences $u - v$). N'hésitez pas à zoomer le résultat localement pour bien voir les différences.

A quoi sont dus les artefacts observés sur les bords du cadre de l'image ? Proposez une méthode pour les éviter et démontrez numériquement son efficacité.

Exercice 16 (rotation par produit de glissements)

Comment effectuer la rotation d'une image carrée (de taille $N \times N$) d'un angle θ quelconque autour de son centre, en préservant la taille initiale ? A priori, réaliser cette transformation par interpolation de Shannon devrait nécessiter $O(N^4)$ opérations, puisqu'il n'y a pas moyen de coder la rotation exacte directement dans le domaine de Fourier (sauf pour des angles multiples de $\pi/2$ bien sûr). Néanmoins, en écrivant la rotation comme un produit de glissements selon la formule vue en cours

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} = G_{-\tan\frac{\theta}{2}} \, (G_{\sin\theta})^{\mathsf{T}} \, G_{-\tan\frac{\theta}{2}}, \quad \text{avec} \quad G(t) = \begin{pmatrix} 1 & t \\ 0 & 1 \end{pmatrix},$$

nous pouvons réaliser avec des transformées de Fourier (donc en $O(N^2 \log N)$ opérations) un produit de trois glissements **périodiques** qui coïncide avec la rotation d'angle θ sur une partie du domaine de l'image.

1. Effectuer ainsi la rotation d'une image de $\frac{\pi}{6}$ à l'aide des instructions suivantes:

```
import numpy as np
import imtools as im
from math import sin,tan

u = im.load('lena.pgm').astype('double')
x0 = (u.shape[1]+1)/2
y0 = (u.shape[0]+1)/2 # centre de l'image
theta = np.pi/6
u1 = im.fftshear(u,  -tan(theta/2), y0, axis=1)
u2 = im.fftshear(u1,  sin(theta),   x0, axis=0)
u3 = im.fftshear(u2, -tan(theta/2), y0, axis=1)
```

   La fonction `fftshear` fait partie de la toolbox donnée en cours. Visualiser le résultat (`u3`), ainsi que les étapes intermédiaires (`u1` et `u2`). Faire de même pour une image synthétique plus simple donnée par

```
d = np.linspace(-1,1,500)
X,Y = np.meshgrid(d, d)
u = X+Y
```

   et représenter sur un schéma les zones au bord de l'image où la rotation est inexacte en raison de la périodicité des glissements (pour le schéma, on pourra se placer sur une domaine continu centré, par exemple le carré $[-1,1]^2$).

2. Tester les instructions précédentes avec un angle de $\frac{5\pi}{6}$. Que remarque-t-on ? Expliquer comment on peut toujours se ramener, à l'aide d'une transformation préliminaire simple de u, à n'appliquer la formule précédente que pour $|\theta| \leq \frac{\pi}{2}$, et pourquoi c'est préférable. Illustrer cette solution pour appliquer à l'image `u` une rotation de $\theta = \frac{5\pi}{6}$.

3. Déterminer expérimentalement (approximativement) le sous-ensemble E du domaine D de l'image initiale sur lequel **toute** rotation d'angle $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ est exacte (comme pour la question 1, on pourra faire un schéma dans le cas où $D = [-1,1]^2$).