TRAITEMENT D'IMAGES

L'objectif de ce TP est de de travailler sur les tableaux bidimensionnels à travers le traitement d'images.

Pour les tests, on pourra récupérer sur le site de la classe les fichiers `lena_gray.jpg` et `tigrenb.jpg`, à placer dans le répertoire de travail de Pyzo (indiqué en entrant `cd` dans le shell).

Dans ce TP, pour plus de simplicité, on travaillera avec des images en niveaux de gris. Une image en niveaux de gris de hauteur n et largeur p sera manipulée comme un tableau numpy `img` de taille $n \times p$, contenant donc n lignes, chacune de longueur p. Le pixel de coordonnées i, j correspondra ainsi à la case `img[i][j]` (le pixel de coordonnées $(0,0)$ étant en haut à gauche). Le contenu de chaque case est un entier compris entre 0 (noir) et 255 (blanc).

Le programme Python ci-dessous permet de récupérer dans le tableau `img` une image en niveau de gris, puis de l'afficher dans une fenêtre graphique :

```
import numpy as np # importation de la bibliotheque numpy renommee np
import matplotlib.pyplot as plt # importation de la bibliotheque matplotlib
#renommee plt

plt.close('all')

img = np.array(plt.imread('lena_gray.jpg')) #recuperation de l'image

#lignes pour afficher l'image :
plt.figure()
plt.imshow(img,cmap=plt.cm.gray,vmin=0,vmax=255)
plt.show()
```

1. Écrire une procédure (c'est-à-dire une fonction qui ne renvoie rien) `negatif` qui prend en argument une image `img` et la modifie en son négatif (*i.e.* en remplaçant la valeur x de chaque pixel par $255 - x$)

   Indication : on peut récupérer les dimensions $n \times p$ d'un tableau à deux dimensions A en écrivant `n,p = len(A),len(A[0])`. Dans le cas particulier d'un tableau numpy, on peut également écrire `n,p = A.shape`.

   On testera cette fonction puis les suivantes en affichant l'image avant et après modification.

2. Écrire une procédure `niveaux_de_gris` prenant en argument une image et remplaçant ses niveaux de gris par uniquement trois niveaux : les niveaux de gris entre 0 et 79 inclus sont remplacés par 60, les niveaux de gris entre 80 et 149 inclus sont remplacés par 120 et les autres niveaux de gris sont remplacés par 220.

3. (a) Écrire une procédure `eclaircir` prenant en argument une image et un entier m et ajoutant cet entier à la valeur de chaque pixel.

   (b) La procédure précédente a-t-elle l'effet escompté ? Le type utilisé pour les valeurs des pixels (`uint8`) manipule des entiers modulo 256. Gérer le cas où l'ajout de m provoquerait un *dépassement arithmétique* hors de l'intervalle $[\![0, 255]\!]$

   (c) Écrire sur le même principe une procédure `assombrir`.

4. Écrire une fonction `flip_left`, prenant une image en argument et renvoyant sa rotation de 90° dans le sens trigonométrique. En général, le résultat de la rotation n'a pas les mêmes dimensions que l'image initiale, il faudra donc initialiser un nouveau tableau, `np.zeros([n,p],np.uint8)` permettant de créer un tableau de dimensions $n \times p$ dont les cases valent initialement 0.

   Tester cette fonction avec l'image `lena_gray.jpg` puis avec `tigrenb.jpg`.

5. Écrire une fonction `compression` prenant en argument une image `img` et un entier strictement positif `n` et renvoyant une nouvelle image en prenant un pixel tous les `n` pixels, en hauteur comme en largeur.

6. Soit A un tableau 3×3, par exemple :

$$A = \begin{array}{|c|c|c|}\hline 1/12 & 1/12 & 1/12 \\ \hline 1/12 & 4/12 & 1/12 \\ \hline 1/12 & 1/12 & 1/12 \\ \hline \end{array}$$

   Pour chaque pixel `(i, j)` qui n'est pas sur un bord, on peut appliquer A comme un filtre pour passer d'une image à une autre :

$$\text{img2}_{i,j} = \sum_{k=-1}^{1} \sum_{\ell=-1}^{1} A_{1+k,1+\ell} \text{img}_{i+k,j+\ell}.$$

   Écrire une fonction qui prend en argument `img` et un tableau A 3×3 et qui renvoie la nouvelle image `img2` donnant le résultat du filtrage selon le tableau A. Quel est l'effet produit par le filtre A donné en exemple ?

   Indication : si `A` et `B` sont deux tableaux **numpy** de même taille, `A*B` renvoie un tableau de même taille dont les coefficients sont les produits deux à deux des coefficients de `A` et `B`.

7. On cherche à présent à détecter les contours présents dans l'image.

   (a) On définit la **dérivée horizontale** de l'image img en i, j comme $\text{img}_{i,j} - \text{img}_{i-1,j}$.

   Écrire le filtre `Ah` permettant d'obtenir le tableau des dérivées horizontales à partir d'une image initiale. On appellera `imgh` l'image ainsi filtrée.

   (b) Écrire de même le filtre donnant la dérivée verticale $\text{img}_{i,j} - \text{img}_{i,j-1}$. On appellera `imgv` l'image ainsi filtrée.

   (c) Écrire une fonction permettant de détecter les contours dans les deux directions en utilisant pour chaque point de l'image :

$$\text{imgc}_{i,j} = \sqrt{\text{imgh}_{i,j}^2 + \text{imgv}_{i,j}^2}.$$

8. On cherche à présent à redimensionner une image.

   (a) Écrire une fonction `moyenne` prenant en argument une image et 4 indices $x1, x2, y1, y2$ supposés vérifier $x1 \leq x2$ et $y1 \leq y2$, et renvoyant la moyenne des valeurs des pixels dans le rectangle de l'image dont $(x1, y1)$ et $(x2, y2)$ forment des sommets opposés.

   (b) En déduire une fonction `redimensionnement` prenant en argument une image et deux ratios `rh` et `rl`, et renvoyant une version redimensionnée de l'image selon ces ratios.

.