Chapitre 5 :
Traitement d'images 1

I Tableaux numpy

Les tableaux sont représentés par un type particulier de numpy. Avant d'utiliser les commandes suivantes, on importera bien numpy :

import numpy as np

1.1 Création d'un tableau

La syntaxe pour créer un tableau est la suivante :

```
>>> T=np.array([[1,2],[3,4]])
>>> T
array([[1, 2],
       [3, 4]])
```

Cette définition des tableaux permet de faire des opérations algébriques :

- T+T' est le tableau obtenu en ajoutant termes à termes les éléments de T et de T', deux tableaux de même taille,
- n*T est le tableau obtenu en multipliant tous les éléments de T par n.

1.2 Accès aux éléments d'un tableau

⚠Comme pour les listes, la numérotation dans les tableaux commence à 0.

On utilise la syntaxe suivante :

- T[i][j] pour accéder au terme (i, j) d'un tableau.
- T[i] pour accéder à la i-ème ligne d'un tableau.
- T[:,j] pour accéder à la j-ème colonne d'un tableau.

```
>>> T=np.array([[1,2],[3,4]])
>>> T[1][1]
4
>>> T[1]
array([3, 4])
>>> T[ :,1]
array([2, 4])
```

La commande len(T) ne donne pas la taille du tableau mais uniquement son nombre de lignes. Pour avoir la taille sous la forme d'un couple (nombre de lignes,nombre de colonnes), on utilise T.shape

```
>>> T=np.array([[1,2,3],[4,5,6]])
>>> len(T)
2
>>> T.shape
(2,3)
```

1.3 Modification d'un tableau

Comme pour les listes, si deux tableaux sont définis comme étant égaux, toute modification d'un des deux sera appliquée à l'autre également. On utilisera donc une copie de tableaux.

```
>>> T=np.array([[1,2],[3,4]])
>>> T1=T
>>> T2=np.copy(T)
>>> T[1][1]=5
>>> T1
array([[1, 2],
       [3, 5]])
>>> T2
array([[1, 2],
       [3, 4]])
```

II Décomposition d'une image en pixels

Les images qu'on utilisera en TD sont des images dites matricielles. On utilisera des images au format PNG. Elles sont composée d'un tableau de points colorés appelés pixels. Chaque pixel est un triplet de nombres entre 0 et 255 : un nombre pour chaque couleur rouge, vert, bleu. Il y a $256 = 2^8$ nombres possibles pour chaque couleur, cela représente donc 8 bits, soit un octet. Le nombre de couleurs possibles est $256^3 = 16777216$. Le triplet $(0,0,0)$ correspond à un pixel noir, le triplet $(255,255,255)$ a un pixel blanc, $(255,0,0)$ a un pixel purement rouge,...

Une image est représentée par un tableau tridimensionnel de taille $h \times l \times 3$, où h est la hauteur de l'image et l sa largeur.

Pour récupérer une image sous forme d'un tableau numpy, on utilisera :

```
from PIL import Image
im=Image.open(' chemin\vers\image.png') # ouverture de l'image
tab=np.array(im) # création du tableau représentant l'image
```

Pour créer une nouvelle image à partir d'un tableau :

```
new=Image.fromarray(tab) # création de la nouvelle image
new.show() # affichage de la nouvelle image
new.save('nom.png') # sauvegarde de la nouvelle image
```

Le type des éléments des tableaux utilisés pour représenter des images doit être uint8 (entiers non signés codés sur 8 bits). Il faudra donc, lorsque c'est nécessaire, imposer ce type aux tableaux crées. On créera donc des tableaux remplis de 0 au format uint8 que l'on modifiera ensuite :

```
np.zeros((h,l,3),dtype='uint8')
```

III Lena

L'image de Lena a été utilisée pour la première fois en 1973 à l'université de Californie du Sud afin de tester des algorithmes de traitement d'images. Depuis, cette image est devenu l'image classiquement utilisée pour tous les tests.

IV Exemple des composantes

Voici un exemple de fonction prenant en argument une image im et renvoyant la composante rouge de l'image, c'est à dire, l'image sur laquelle les composantes vertes et bleues des pixels sont nulles.

```
def image_rouge(im) :
    t=np.array(im)
    h,l,r=t.shape
    for i in range(h) :
        for j in range(l) :
            for k in range(1,3) :
                t[i][j][k]=0
    return Image.fromarray(t)

lena_rouge=image_rouge(lena)
lena_rouge.show()
```

On fait de même avec les composantes vertes.

```
def image_verte(im) :
    t=np.array(im)
    h,l,r=t.shape
    for i in range(h) :
        for j in range(l) :
            for k in [0,2] :
                t[i][j][k]=0
    return Image.fromarray(t)

lena_verte=image_verte(lena)
lena_verte.show()
```

Et avec les composantes bleues.

```
def image_bleue(im) :
    t=np.array(im)
    h,l,r=t.shape
    for i in range(h) :
        for j in range(l) :
            for k in [0,1] :
                t[i][j][k]=0
    return Image.fromarray(t)

lena_bleue=image_bleue(lena)
lena_bleue.show()
```