TP 4 – Traitement d'image

1 Représentation des images numériques

Comme toute donnée stockée et manipulée numériquement, une image est transcrite à l'aide de nombres en écriture binaire.
Ces nombres sont structurés dans un tableau où chaque case décrit un pixel (acronyme pour *"picture element"*), c'est à dire un « point » de l'image *pixélisée* (en réalité un carré de petite taille).
Ce tableau est de type `array` disponible dans la bibliothèque Python `numpy`).

- Pour une image en noir et blanc sans niveau de gris, chaque pixel ne peut prendre que deux valeurs, ce qui permet de l'encoder sur un seul bit (*binary integer*) ;
- pour une image en niveaux de gris, chaque teinte de gris (en passant du noir au blanc) est encodée sur un octet, ce qui permet de représenter ..... nuances de gris ;
- pour une image en couleur, on représente chaque couleur comme une addition de trois couleurs de base ; les pixels d'une image couleur peuvent être encodés
  - en utilisant le format RGB (.....) ; dans ce cas chaque pixel est encodé sur trois octets, chaque octet représentant une nuance de couleur : le premier octet dose la quantité de rouge, le second la quantité de vert et le troisième la quantité de bleu ;
  - en utilisant le format format RGBA ; dans ce cas chaque pixel est encodé sur quatre octets, le quatrième octet codant la transparence du pixel.

Le format d'une image (png, jpeg, etc) désigne la manière dont cette image est représentée en mémoire (avec des algorithmes standards de compression de données).

(pour les curieux : https://fr.wikipedia.org/wiki/JPEG , où l'on se rend compte que les notions et notations travaillées en maths sont directement utilisées dans ces situations très pratiques)

Avant de commencer :
Vous aurez besoin des modules suivants :

- le module `numpy`
- le module `mathplotlib.pyplot` pour la visualisation des images

Vous commencerez donc par entrer les commandes suivantes :

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
Einstein = matplotlib.image.imread("Einstein.png")
```

Commandes utiles

- La fonction `matplotlib.image.imread(source)` permet de convertir une image .png en un tableau `numpy`. Elle prend en argument une chaîne de caractères décrivant l'adresse d'un fichier présent sur votre ordinateur. Par exemple, pour convertir l'image Einstein.png en tableau `numpy`, vous placerez cette image dans votre dossier courant (le dossier ./) puis vous utiliserez la commande :
  `Einstein = matplotlib.image.imread("Einstein.png")`
  Ainsi définie Einstein est un tableau `numpy` à trois dimensions dont les éléments sont de type `np.uint8` (entiers non signés représentés sur 8 bits, i.e. un entier compris entre 0 et 255). Pour repérer une case du tableau on utilise trois indices, les deux premiers indices représentant les coordonnées d'un pixel, le troisième la couleur considérée.
  `Einstein[i,j]` est donc un triplet quantifiant le volume de Rouge, de Vert et de Bleu dans le pixel de coordonnées (i, j).
- La fonction `plt.imshow(tab)` permet de visualiser l'image décrite par le tableau tab.
- La méthode `im.shape` permet de trouver les dimensions du tableau `numpy` obtenu à l'aide de la commande `matplotlib.image.imread("Einstein.png")`.

2 Fonctions élémentaires

Question 1 : Rédiger une fonction `symetrie(im)` qui prend en argument une image et retourne une nouvelle image, obtenue par symétrie autour d'un axe vertical passant par le milieu de l'image.
Indications :

- *la fonction `np.zeros_like(m)` crée un tableau nul de même type et de mêmes dimensions que le tableau m.*
- *La fonction `np.shape(A)` permet d'obtenir la taille d'une matrice A.*
- *la fonction `np.zeros((n,p,s))` crée une matrice nulle de taille $n \times p \times s$.*

Question 2 : Rédiger une fonction `rotation(im)` qui prend en argument une image et retourne une nouvelle image, obtenue par rotation d'un angle $+\pi/2$ autour du centre de l'image.

Question 3 : Rédiger une fonction `negatif(im)` qui prend en argument une image et retourne son négatif, c'est-à-dire l'image dans laquelle chaque composante de couleur de chaque pixel est remplacée par sa valeur complémentaire dans l'intervalle $[0, 255]$ (c'est-à-dire que x est remplacé par $255 - x$).

3 Conversion en niveaux de gris

Pour convertir une image couleur en niveaux de gris, un pixel représenté par ses composantes (r,g,b) doit être remplacé par un pixel à une seule composante $y \in [0, 255]$ appelée luminance. Cette quantité traduit la sensation visuelle de luminosité, et dépend de manière inégale des trois composantes RGB (pour un oeil humain le vert parait plus lumineux que le rouge, lui-même plus lumineux que le bleu). La formule recommandée pour cette conversion est :

$$y = 0,2126\,r + 0,7152\,g + 0,0722\,b,$$

y étant arrondi à l'entier le plus proche .

Question 4 : Rédiger une fonction `niveaudegris(im)` qui prend an argument une image couleur et retourne une nouvelle image convertie en niveaux de gris.
Indication : l'arrondi d'une valeur x peut se calculer avec la fonction np.round(x)

4 Zoom, réduction, black & white et floutage

Question 5 :
Rédiger une fonction `zoom(im,x)` qui prend en argument une image (couleur, niveau de gris ou noir et blanc) et retourne une nouvelle image en ayant zoomé sur sur le centre de l'image selon un facteur `x`.

Question 6 : Rédiger une fonction `reduc(im)` qui prend en argument une image (couleur, niveau de gris ou noir et blanc) et un entier n, et qui retourne une nouvelle image en ayant pris un pixel sur n en ligne et en colonne.

Question 7 : Rédiger une fonction `noirblanc(im,seuil)` qui affiche une image en noir et blanc à partir d'une image couleur à partir d'un seuil (on pourra appeler la fonction `niveaudegris`) .

Question 8 : Rédiger une fonction `flou(im)` qui affiche une image où l'intensité de chaque pixel dans l'image renvoyée est la moyenne de l'intensité des pixels compris dans un bloc 3×3 centré sur le pixel considéré.

5 Convolution

Le floutage ci-dessus est réalisé en remplaçant la valeur de chaque pixel par une combinaison linéaire des valeurs des pixels voisins, dans le cas particulier où les coefficients son égaux.
Le principe général s'appelle une **convolution** de l'image.
Les coefficients de cette combinaison linéaire peuvent être consignés dans une matrice, que l'on appellera « masque de convolution ».

Question 9 : Tester les masques de convolution suivants et préciser leur action :

1. $\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ **2.** $\begin{pmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

$\sim$