TP : Manipulation des images (python 2.7)

I) Bibliothèque `Image`

A) Introduction, ouvrir, sauver

La bibliothèque `Image` permet de transformer un fichier en objet python représentant l'image :

```
import Image as img
js_image=img.open('C:/FichiersTPImages/js_fichier.jpg')        (fichier ⟶ image)
js2_image.save('C:/FichiersTPImages/new.jpg')                  (image ⟶ fichier)
```

Le chemin d'accès du fichier est donné sur le modèle du `run('...')` qui s'affiche lorsque vous lancez le script dans un interpreteur python actif.

B) Quelques informations sur les images

```
js_image.show()                                                (affichage)
js_image.size                                                  (taille)
js_image.mode                  (L : niveaux de gris ; RGB : red, green, blue – couleur)
```

Et bien d'autres commandes que vous pouvez explorer via le menu déroulant qui s'affiche dans spyder lorsque vous tapez `img.`, puis l'aide de spyder, et pour aller plus loin la documentation en ligne.

II) Conversion `image` ⟷ tableau `numpy`

Pour manipuler les images, on les convertis en un objet python plus classique : un tableau `numpy` :

```
import numpy as np
js_array=np.array(js_image)                    (image ⟶ tableau numpy)
js2_image=img.fromarray(mon_tableau)           (tableau numpy ⟶ image)
```

Si l'image est en niveaux de gris, le tableau contient des entiers entre 0 et 255 donnant l'intensité du pixel de coordonnée (i, j).
Les entiers entre 0 et 255 se nomment `int8`, pour entiers 8 bits : il suffit de 8 bits pour stocker l'information (compter de 0 à $2^8 - 1$ en base 2 : 8 chiffres au plus).
Si l'image est en couleur, le tableau contient des triplets d'entiers, chacun représentant l'intensité en `R`, `G` et `B`.

III) Exercices

Exercice 1
Tester les commandes précédentes (en adaptant au besoin de le chemin d'accès au fichier : python part de l'endroit où vous avez sauvé votre script.
Récupérer la forme de l'image à partir du tableau `numpy` représentant l'image.
Écrire une fonction qui prend un tableau `numpy` d'entiers `int8` (8 bits : entre 0 et 255) et le transforme en une image qu'il affiche. La tester sur une image aléatoire.

Exercice 2 (isométries du plan)

1) Écrire une fonction qui crée le négatif d'une image.

2) Écrire une fonction qui effectue une symétrie par rapport à la diagonale $i = j$ (on pourra considérer le tableau contenant l'image comme une matrice).

3) (optionnel) même question pour les rotations $+\pi/2$, $+\pi$, $-\pi/2$, et les symétries d'axe vertical et horizontal.

Exercice 3 (informations sur l'image)

1) Compter le nombre de pixels noirs à 90%. Même question pour ceux noirs à 10%.

2) Compter puis afficher un histogramme du nombre de pixels en fonction de l'intensité, par tranches de 10%.

3) Même question avec N tranches, où N est rentré par l'utilisateur.

4) Arrondir : pour $N = 2$, « arrondir » l'intensité des pixels de la première tranche à 0 et celle de la seconde tranche à 255.

5) (Relativement indépendante des précédentes) Augmenter le contraste de l'image.

Exercice 4 (création ex-nihilo)

1) Créer, à partir d'un tableau vide et de boucles, un quadrillage (ou tout autre motif régulier). Rajouter une petite perturbation (avec `random`).

2) Inverser la matrice précédente (on suppose qu'elle est inversible, grâce à la perturbation). Afficher l'image correspondante, en prenant soin de transformer de nouveau le tableau en un tableau `int8` (la valeur la plus grande est envoyée sur 255, la plus petite sur 0 et le reste est proportionnel).

Exercice 5 (flou)

1) Á l'aide d'une moyenne, proposer un algorithme pour rendre floue une image.

2) De même, baisser la résolution d'une image en divisant par 2 le nombre de pixels sur chaque axe (on commencera par tronquer l'image).

3) Proposer un algorithme de recherche de contours.

Exercice 6 (couleur)

On utilisera le fichier `couleur.jpg`

1) Adapter quelques unes des fonctions précédentes au cas d'une image couleur.

2) Détecter le fond de l'image à l'aide de la couleur (si le résultat n'est pas parfait, se contenter de proposer une piste pour améliorer la fonction).

3) Passer l'image en niveaux de gris.