TP : Manipulation des images (python 3.3)

I) Bibliothèque `skimage`

A) Ouvrir, afficher, sauver

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

```
import skimage.io as io
js_image=io.imread('C:/FichiersTPImages/js_fichier.jpg')  (fichier ⟶ tableau numpy)
io.imshow(js_image) puis io.show()                        (affichage)
io.imsave('C:/Fichiers/TPImages/new.jpg',js_image)        (tableau numpy ⟶ 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.
Il existe d'autres commandes que vous pouvez explorer via le menu déroulant qui s'affiche dans spyder lorsque vous tapez `io.` (puis regardez l'aide de spyder).
Pour aller plus loin, il existe d'autres sous packages du package skimage, cf la documentation en ligne.
Remarque : on peut aussi utiliser matplotlib pour charger les images.

B) Type et forme du tableau `numpy`

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). Un pixel éteint (noir) a une intensité nulle (0). Inversement, un pixel blanc vaut 255.
Le type « entiers entre 0 et 255 » se nomme `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`.

II) Exercices

Exercice 1 (Premières manipulations)

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.

2) Récupérer la forme de l'image à partir du tableau `numpy` représentant l'image.

3) Afficher le quart en haut à droite de l'image (on pourra utiliser la notation par blocs – ou découpage en tranches – dans les tableaux `ndarray`).

4) Vérifier que la commande `js_image[0,0]=255` rend le pixel en haut à gauche blanc.

5) Tracer un cadre autour du visage. Un triangle (en blanc) entre $(200, 280)$, $(300, 300)$ et $(290, 400)$.

6) (optionnel) Créer un tableau `numpy` d'entiers `int8` (8 bits : entre 0 et 255) aléatoire et l'afficher.

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% (vous pourrez vous inspirer de votre TP sur matplotlib et random, qui est parfaitement bien rangé et nommé, donc facile à retrouver et réutiliser).

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 la luminosité 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) À l'aide d'une moyenne, baisser la résolution d'une image en divisant par 2 le nombre de pixels sur chaque axe (on supposera que l'image a un nombre pair de pixels).

2) De même, proposer un algorithme pour rendre floue une image.

3) Proposer des pistes pour 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.

Quelques pistes et projets potentiels pour aller plus loin (avec de la recursivité) :

- A partir d'une image vide, fractales, flocons de Koch

- Comment recoller deux images (ou faire des faux...)