TP n°7 : «Quelques manipulations d'images numériques»

Objectifs et consignes

Nous allons comprendre comment est représentée une image numérique Bitmap et s'initier au traitement d'images avec Python.

On commencera par l'import des modules suivants :

```
import matplotlib.pyplot as plt # pour faire le lien entre image et matrice de pixels
import numpy as np  # pour manipuler les matrices de pixels en tant que tableaux numpy
```

Dans la première section, nous allons manipuler des images en nuance de gris. Elles seront représentées par des matrices de nombres flottants compris entre 0 et 1. Un pixel représenté par 0 sera noir (intensité lumineuse minimale, il est éteint), et un pixel représenté par 1 sera blanc (intensité lumineuse maximale).

1 De la matrice vers l'image

Pour afficher vos matrices, vous pouvez utiliser la fonction suivante :

```
def affiche(matrice):
    for ligne in matrice:
        print(ligne)
```

Exercice 1

1. Écrire une fonction `matrice_nulle(n, p)` qui renvoie la matrice nulle à n lignes et p colonnes à l'aide d'une «comprehension list».
2. Écrire une fonction `dimensions(matrice)` qui renvoie le couple (n, p) où n et p sont respectivement le nombre de lignes et le nombre de colonnes de la matrice.

Exercice 2 (Ma première image en niveau de gris) Considérons la matrice

$$M = \begin{pmatrix} 0 & 0.25 & 0.5 & 0.75 & 1 \\ 1 & 0.75 & 0.5 & 0.25 & 0 \end{pmatrix}$$

Implémenter la matrice M puis afficher l'image correspondante en niveau de gris en exécutant :

```
plt.imshow(M, cmap = 'gray')
plt.show()
```

Ici les intensités de niveau de gris sont des nombres flottants compris entre 0 et 1. Mais cela peut aussi être des entiers entre 0 et 255.

Exercice 3 (Dessiner un damier)

```
[[0. 1. 0. 1. 0. 1. 0. 1.]
 [1. 0. 1. 0. 1. 0. 1. 0.]
 [0. 1. 0. 1. 0. 1. 0. 1.]
 [1. 0. 1. 0. 1. 0. 1. 0.]
 [0. 1. 0. 1. 0. 1. 0. 1.]
 [1. 0. 1. 0. 1. 0. 1. 0.]
 [0. 1. 0. 1. 0. 1. 0. 1.]
 [1. 0. 1. 0. 1. 0. 1. 0.]]
```

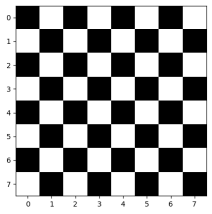

FIGURE 1 – La matrice et l'image du damier

1. Créer la matrice suivante `damier` à 8 lignes et 8 colonnes
2. Afficher l'image en niveau de gris correspondante en exécutant le script suivant :

```
plt.imshow(damier, cmap = 'gray')
plt.show()
```

Nous allons voir qu'une image couleur dite RGB (Red, Green, Blue) est une matrice de pixels codés par 3 entiers entre 0 et 255 représentant leur intensité respective en rouge, vert et bleu du pixel. Par exemple, un pixel codé par le triplet (255,0,0) sera rouge et un pixel codé par (0,0,0) sera noir. Les triplets seront représentés en Python par une liste de 3 entiers.

Exercice 4 (Ma première image en couleur)

1. Créer la matrice *img* de taille 2×3 dont l'image est :

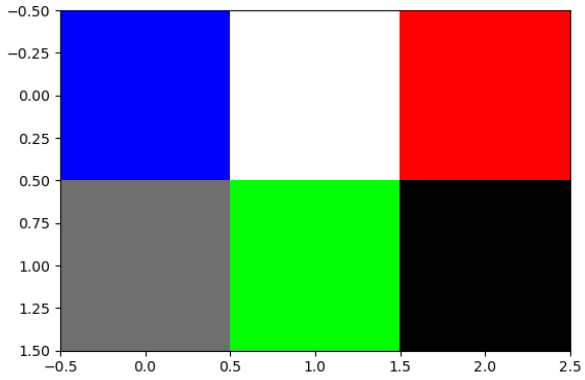

FIGURE 2 – ma première image couleur

On pourra vérifier en exécutant :

```
plt.imshow(img)
plt.show()
```

2. Exécuter le script suivant et observer la correspondance entre couleurs et triplets d'intensité.

```
img2 = [
[ [255, 0, 0], [100,0,0] , [50,0,0]],
[ [255, 255, 0], [100, 100, 0] , [50,50, 0]],
[ [255, 0, 255], [100,0,100] , [50,0,50]],
[ [255, 255, 255], [100,100,100] , [50,50,50]]
]
plt.imshow(img2)
plt.show()
```

Exercice 5 (Le drapeau français) Créer à l'aide d'une matrice une image représentant le drapeau français. On impose une hauteur h de 300 pixels et une largeur w (width) de 600 pixels.

Exercice 6 (Le drapeau japonais) Créer à l'aide d'une matrice une image représentant le drapeau japonais. Si h est la hauteur de l'image, on prendra comme largeur $2h$ et pour rayon du cercle rouge $3h/10$. On veut aussi une image de format 600×300.

FIGURE 3 – Drapeaux français et japonais

2 Notion de répertoire de travail

Nous allons ouvrir avec Python des images, mais pour cela il faudra indiquer où se trouvent les images.

Dans votre répertoire de travail (celui dans lequel se trouve votre fichier `.py`), vous créerez deux répertoires `images_in` et `images_out`. Vous placerez dans `image_in` les images que l'on vous fournira et dans `image_out` les images que vous fabriquerez.

Créer deux variables `entree` et `sortie` selon le script suivant :

```
entree, sortie = 'images_in/', 'images_out/'
```

2.1 Avec Jupyter

Le répertoire de travail par défaut est celui où se trouve votre notebook.

2.2 Avec Pyzo

Le répertoire de travail par défaut n'est pas en général celui où se trouve votre fichier `.py` (taper `os.getcwd()` pour le connaître).

Pour indiquer que votre répertoire de travail se situe à l'emplacement de votre fichier Python (et ne pas avoir de souci de chemin de fichiers), plusieurs méthodes :

- vous faîtes un clic droit avec la souris dans le shell, puis vous choisissez `change current directory to editor file path`.
- vous exécutez une **première** fois votre fichier en tant que script à l'aide des touches **CTRL + MAJ + E**.
- On peut aussi le faire en allant dans Pyzo dans le menu `tools`, puis en affichant le `filebrowser` et en cliquant sur la petite flèche de `l'étoile` pour préciser son répertoire de travail en choisissant `Go to this directory in the current shell`.
- Enfin, on peut aussi changer le répertoire de travail à la main en indiquant le chemin du répertoire de travail souhaité : `os.chdir('nouveau_chemin')`

3 De l'image vers la matrice...

Exercice 7 (La joconde en RGB) Nous allons voir qu'une image couleur dite RGB (Red, Green, Blue) est une matrice de pixels codés par 3 entiers entre 0 et 255 représentant leur intensité respective en rouge, vert et bleu du pixel. Par exemple, un pixel codé par le triplet (255,0,0) sera rouge et un pixel codé par (0,0,0) sera noir.

1. Exécuter le script suivant

   ```
   joconde = plt.imread(entree + 'joconde.bmp') # matrice de l'image
   plt.imshow(joconde) # pour afficher l'image de la matrice des pixels
   plt.show()
   ```

2. L'instruction `joconde.shape` donne les dimensions de la matrice `joconde`. Préciser la hauteur et la largeur de la photo.

3. L'instruction `joconde.dtype` donne le type des coefficients de la matrice `joconde`. Les instructions `joconde.min()` et `joconde.max()` donnent les valeurs extrêmes des intensités de pixel. En déduire une estimation de l'occupation mémoire de la matrice `joconde`. Vérifier ce chiffre en regardant la place occupée par le fichier image `joconde.bmp`. On pourra à l'aide d'un explorateur de fichiers regarder les propriétés de l'image.

4. Afficher l'image dont la matrice est obtenues à partir de la «tranche» `mains = joconde[400:510, 50:210,:]`. Afficher de même le visage de Mona Lisa.

5. Pour transformer une matrice de pixels en fichier image...

   Exécuter le script suivant, qui crée dans votre répertoire de travail trois fichiers images mais dans des formats différents.

   ```
   plt.imsave(sortie + 'copie.png', joconde)
   plt.imsave(sortie + 'copie.jpeg', joconde)
   ```

   Compléter alors le tableau suivant :

format	destruction	compression	taille en ko
png	non	oui	
jpeg	oui	oui	

4 Quelques premières manipulations...

Exercice 8 (Convertir en niveau de gris) Coder une fonction `en_gris(img)` qui transforme une image RGB en niveau de gris, en calculant une moyenne pondérée des intensités du rouge, vert et bleue. Des valeurs utilisées habituellement peuvent être :

$$\text{gris } = (0.299 \times \text{rouge } + 0.587 \times \text{vert } + 0.114 \times \text{bleue }).$$

Appliquer cette fonction à l'image de la joconde et à l'image du macareux.

Exercice 9 (Floutage par moyenne) On souhaite maintenant effectuer un flou sur l'image. Pour simplifier, nous allons prendre des images codés en niveau de gris par des flottants entre 0 et 1.

Pour cela, nous allons changer chaque pixel de l'image par la moyenne avec ses 4 pixels voisins (dessus, dessous, gauche, droite) (si le pixel est sur un bord, on ne le change pas).

1. Écrire une fonction `moyenne(img)` qui prend en entrée une image (sa matrice) `img` en niveau de gris et renvoie une nouvelle matrice obtenue en «floutant» celle-ci. La fonction devra rendre une matrice de même taille que l'originale, mais on n'exige rien de particulier sur la valeur des pixels du bord.
2. Tester votre fonction avec l'image `lena`, que l'on floutera une fois, puis deux fois, trois fois.
3. Écrire une fonction `flou(image, k)` qui prend en entrée une image en niveau de gris et affiche l'image obtenue après avoir flouté k fois `image`.

5 Quelques transformations géométriques

Exercice 10 (Symétries) On désire dessiner le symétrique par rapport à l'axe médian vertical de l'image de la joconde.

1. On considère un pixel p d'indice (i, j) d'une image de hauteur h et de largeur l. Donner l'indice (i', j') du pixel symétrique de p par rapport à l'axe médian de l'image (vérifier que lorsque j décrit $[\![0, l-1]\!]$, on a bien j' qui décrit $[\![0, l-1]\!]$).
2. Dessiner le symétrique par rapport à l'axe médian vertical de l'image de la joconde, puis automatiser et créer une fonction `symetrie_verticale` qui prend en argument une matrice de pixels et renvoie la matrice de pixels obtenue par symétrie d'axe vertical.
3. Créer une fonction `symetrie_horizontale` qui prend en argument une matrice de pixels et renvoie la matrice de pixels obtenue par symétrie d'axe horizontal.

Exercice 11 (Rotation de quart de tour)

1. Écrire une fonction `quart_tour_trigo` qui prend en argument une matrice de pixels et renvoie une matrice de pixels résultant de la rotation d'un quart de tour dans le sens trigonométrique.
2. Écrire une fonction `quart_tour_horaire` qui prend en argument une matrice de pixels et renvoie une matrice de pixels résultant de la rotation d'un quart de tour dans le sens horaire.

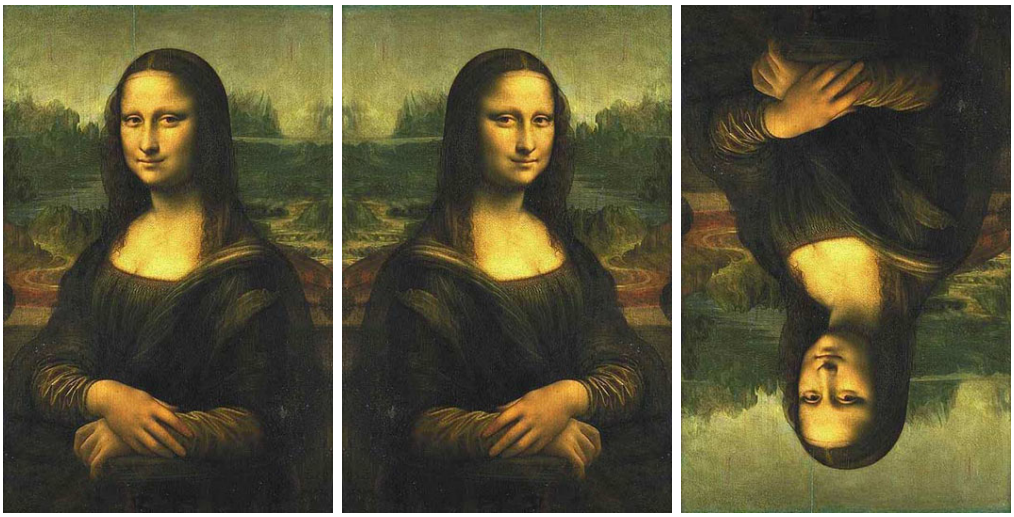

FIGURE 4 – Mona Lisa dans tous ses états

FIGURE 5 – Quarts de tours de Mona