TP n°4 - Traitement d'images - partie 1

Notions abordées

- Notion de pixel, de niveau de gris, de couleur codée en RGB
- Manipulation de listes, création de listes par compréhension
- Passage d'une image en niveau de gris, filtre par couleur
- Symétrie, rotation, rognage et duplication

Toutes les fonctions doivent être commentées et testées. Elles doivent notamment être munies d'une description faisant suite à d'éventuelles hypothèses sur leurs arguments placée entres triple guillemets sous la signature de la fonction, (`""" documentation sur plusieurs lignes"""`). Les autres commentaires s'écrivent sur des lignes commençant par dièse : `# ligne de commentaire`.

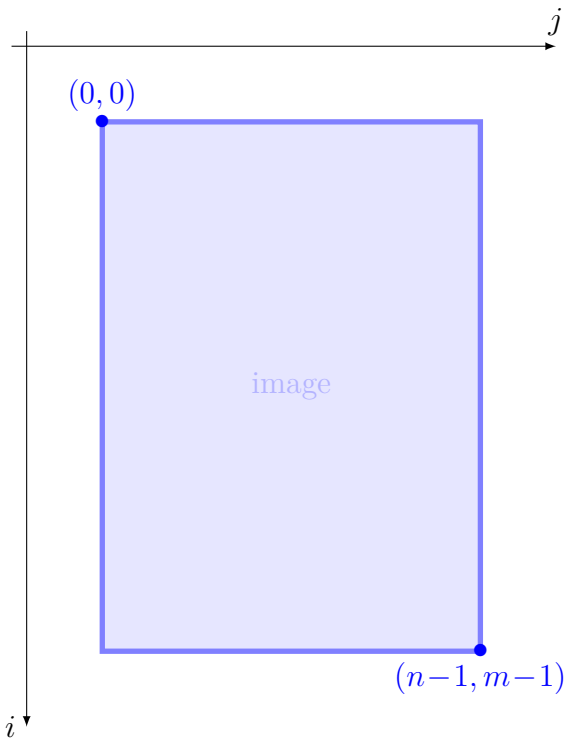
Introduction

Dans ce TP de manipulation d'images, on traitera deux sortes d'images :

- une image en niveau de gris sera une matrice (une liste de listes de même tailles) d'entiers entre 0 et 255, 0 correspond à du noir, 255 à du blanc.
- une image en couleurs sera une matrice de triplets d'entiers représentant une couleur. Chacune des composantes de ces triplets est un entier compris entre 0 et 255, la première représente le rouge, la seconde le vert, la dernière le bleu. On parle de système RGB pour *red, green, blue*.

Les coordonnées des pixels sont données comme les indices des coefficients dans une matrice, c'est-à-dire qu'on indique d'abord la ligne, puis la colonne, les lignes étant numérotées de haut en bas, et les colonnes de gauche à droite, à ceci près que la numérotation commence à 0, non pas à 1.

Ainsi pour une image de n lignes et m colonnes (qui a donc $n \times m$ pixels), le coin en haut à gauche a pour coordonnées $(0, 0)$, tandis que celui en bas à droite pour coordonnées $(n-1, m-1)$.

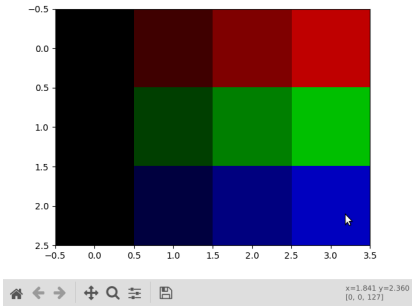


Mini-librairie

On fournit un fichier `tp.py` contenant trois fonctions :

• Une fonction `imread(file)` permettant de lire l'image contenue dans le fichier nommé `file` et de la transformer en matrice python. Par exemple, `imread("examplecol.bmp")` produit la liste `[[[0, 0, 0],...], [[0, 0, 0], [0, 63, 0], ...], [..., [0, 0, 191]]]`.

• Une fonction `imshow(tab)` permettant l'affichage d'une matrice python, vue comme une image. Par exemple l'exécution du programme ci-dessous ouvre la fenêtre graphique ci-contre.



```
im_color = imread("examplecol.bmp")
imshow(im_color)
```

Afin d'éviter la fermeture de cette fenêtre graphique à la fin de l'exécution du programme, on ajoutera la commande `plt.show()` en fin de programme.

• Une fonction `imsave(matrix, name)` permettant la sauvegarde de la matrice `matrix`, vue comme une image dans le fichier nommé `name`. Par exemple l'exécution du programme ci-dessous crée un fichier `examplecol2.bmp` qui est une copie de `examplecol.bmp`

```
im_col=imread("examplecol.bmp")
imsave(im_col, "examplecol2.bmp")
```

Par défaut ces fonctions considèrent que l'image est en couleurs, mais permettent aussi de traiter des images en niveaux de gris, grâce à l'option `grayscale=True`.

• La fonction `imread` utilisée avec cette option, produira une matrice d'entiers et non une matrice de triplets. Par exemple, `imread("examplebw.bmp", grayscale=True)` produit la liste suivante : `[[68, 72, 66, 56], [45, 36, 30, 42], [81, 134, 194, 253]]`

Lire une image en couleur avec l'option `grayscale=True` fait perdre de l'information car seule la première composante de chaque triplet est lue, comme si on lisait le niveau de rouge...

• La fonction `imshow` utilisée avec cette option, afficher une image dite "en noir et blanc".

Afficher une matrice de triplets avec l'option `grayscale=True` revient à ignorer l'option, en revanche afficher une matrice d'entiers sans cette option conduit à un affichage peu pertinent.

Exercice 1 Passer en noir et blanc

Pour transformer une photo en couleurs en niveau de gris il faut associer à chaque triplet (R, G, B) un niveau de gris dans [0..255].

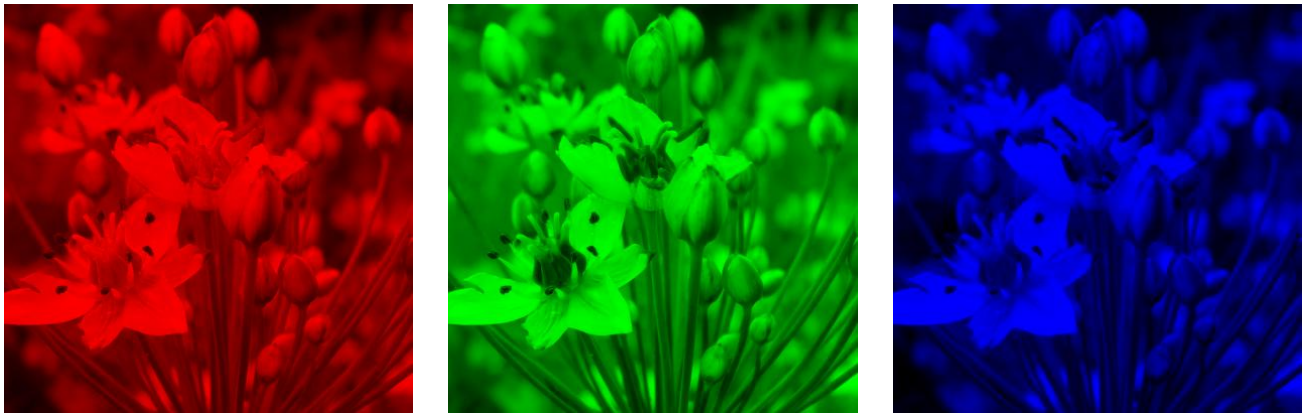
Question 1
Définir une fonction `to_grayscale` qui passe une image en niveaux de gris en associant à chaque pixel de couleur (R, G, B) le niveau de gris $\frac{1}{3}(R + G + B)$.

Question 2
L'œil humain ne percevant pas identiquement les 3 couleurs rouge, vert et bleu, on utilise parfois comme niveau de gris $0,2126 \times R + 0,7152 \times G + 0,0722 \times B$ pour tenir compte de ces différences. Définir une fonction `to_grayscale2` qui passe une image en niveaux de gris de cette manière.

Une image en couleur (à gauche), passée en niveaux de gris par moyenne simple (au milieu) ou par moyenne pondérée (à droite)

Exercice 2 Filtrer les couleurs

Question 1
Définir une fonction `rouge(img)` (resp. `vert(img)` et `bleu(img)`) prenant en argument une image et retournant cette même image mais dans laquelle seule la couleur rouge (resp. vert et bleu) apparaît, c'est-à-dire que les autres composantes de chaque pixel ont été mises à 0.



Les images obtenues en filtrant le rouge (à gauche), le vert (au milieu) ou le bleu (à droite) dans l'image de gauche de la figure précédente.

Exercice 3 Réorganisation des pixels

Dans cet exercice l'image carrée ci-contre servira d'exemple. Le fichier correspondant, nommé `g_parle.JPG`, est disponible dans l'archive du TP.

Question 1
Définir une fonction `cree_image` qui prend en argument une image, deux entiers naturels n et m, et une fonction f de $\mathbb{N}^2$ dans $\mathbb{N}^2$, et qui crée une image de dimensions $n \times m$ telle que $\forall (i,j) \in [0..n[\times[0..m[$, le pixel de coordonnées (i,j) est celui de coordonnées $f(i,j)$ dans l'image de départ. *Cette fonction retourne une liste de listes qui peut être crée par compréhension.*

Question 2
Définir une fonction `symetrie_horizontale` prenant en argument une image et calculant son symétrique par rapport à l'axe horizontal à mi-hauteur, comme le montre l'image ci-contre pour notre exemple.

Question 3
Définir une fonction `symetrie_verticale` prenant en argument une image et calculant son symétrique par rapport à l'axe vertical à mi-largeur, comme le montre l'image ci-contre pour notre exemple.

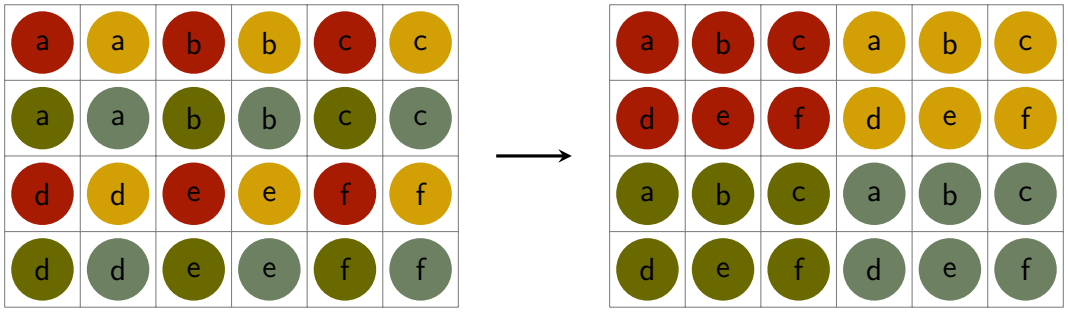
Question 4
Définir une fonction `rotation` prenant en argument une image et calculant sa rotation de 90° dans le sens horaire, comme le montre l'image ci-contre pour notre exemple.




Question 5
Définir une fonction `rognage` prenant en argument une image et 4 entiers i_0, j_0, nbl, nbc et qui extrait l'image de dimensions $nbl \times nbc$ dont le coin supérieur gauche est le pixel de coordonnées (i_0, j_0) dans l'image initiale. Ci-contre, un exemple de rognage de notre exemple. *Vous veillerez à préciser les hypothèses que doivent vérifier les entiers... pour que cette opération ait un sens.*

Exercice 4 Transformation du photomaton

On considère une image de longueur l et hauteur h, avec h et l pairs. On permute les pixels de l'image originale pour obtenir 4 images, chacune 2 fois plus petite, au sens deux fois moins haute et deux fois moins large que l'image originale. Le schéma ci-dessous explicite la transformation à appliquer.



Question 1
Définir une fonction `photomaton` permettant de calculer la transformation du photomaton d'une image passée en argument.

Figure 1: Extrait d'un portrait d'Ada Lovelace (1815-1852) pionnière de la programmation peint par Alfred Edward Chalon (à gauche) et transformation du photomaton de ce portrait (à droite).