TD – 02

Traitement d'images avec numpy

Savoirs et compétences :

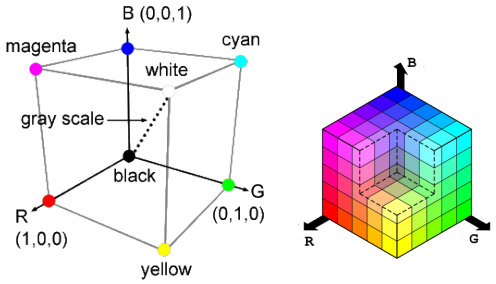
Représentation informatique d'une couleur

La représentation d'une couleur est déterminée par un choix de codage de l'information « couleur ».

On peut choisir d'utiliser 1 bit pour stocker cette information. On distinguera alors uniquement deux couleurs. Par exemple du noir (0) et du blanc (1) ou alors du blanc (0) et du noir (1) ou pourquoi pas du rouge et du noir.

On peut choisir d'utiliser 1 octet pour stocker cette information. On distinguera alors $2^8 = 256$ couleurs différentes. Ces 256 valeurs différentes peuvent représenter des niveaux de gris différents (du blanc au noir ou alors du noir au blanc) ou pourquoi pas du blanc au rouge en passant par 254 nuances de rose.

Le plus souvent, on utilise 3 octets pour stocker une couleur. On distingue alors $256^3 = 16\,777\,216$ couleurs différentes. La norme RGB consiste à séparer ces trois octets et les penser comme un vecteur à 3 coordonnées dans l'espace $R \times G \times B$, où chaque coordonnée peut prendre $2^8 = 256$ valeurs.



On peut aussi utiliser 4 octets (trois octets pour la couleur et un pour la transparence).

On peut aussi utiliser des résolutions plus importantes, des compressions sans pertes.

Bref, il importe de se référer aux spécifications des fichiers que l'on utilise!

Représentation informatique d'une image

Il existe de nombreuses façons de représenter informatiquement une image. Certaines sont vectorielles (SVG, PS), compressées (JPG) d'autres matricielles (PNG). Nous ne nous intéressons ici qu'aux images au format PNG. Il s'agit d'un ensemble discret de points ayant chacun une couleur donnée. Ces points s'appellent des pixels. La résolution de l'image dépend du nombre de points (nombre de lignes et de colonnes), et aussi du nombre de bits servant à coder les couleurs.

Une image est stockée dans un tableau numpy, dont le `shape` est `(n,p,3)`, avec n le nombre de lignes, p le nombre de colonnes et 3 le nombre d'octets pour coder les couleurs.

En plus des bibliothèques habituelles

```
import numpy as np
import matplotlib.pyplot as plt
```

on charge la bibliothèque :

```
import matplotlib.image as mpimg
```

qui nous permet d'utiliser les instructions :

- `img = mpimg.imread(filename)` : le fichier PNG `filename` est converti en tableau numpy et nommé `img`.
- `plt.imshow(img)` : comme `plt.plot(...)`, l'affichage de `img` est préparé, et sera utilisé par `plt.show()` ou `plt.savefig(filename2)`.
- `mpimg.imsave(filename2, img)` permet d'enregistrer l'image (qui n'est pas un schéma python, n'a pas d'axes, de titre etc.)

R Avec `matplotlib.image`, les tableaux numpy sont de `dtype` `float32`, mais ils sont constitués uniquement de 256 valeurs flottantes uniformément réparties entre 0 et 1.

Quelques fonctions élémentaires

Télécharger l'image ***.

Question 1 *Ouvrir l'image avec* `matplotlib.image.imread` *et l'afficher.*

Question 2 *Déterminer pour l'image* `playmobil.png` *: la taille, le* `dtype`*, la couleur du pixel en position (300,300) et les valeurs max. et min. des éléments.*

Jouons avec les couleurs

Question 3 *En utilisant le codage donné précédemment, « où » est codée la couleur bleue?*

Pour saturer l'image en bleue, le niveau de bleu doit être à 1 dans chacun des pixels de l'image.

Question 4 *Réaliser la saturation en bleu.*

```
img[:,:,2]=1
```

Question 5 *Réaliser une inversion des couleurs en remplaçant le niveau n de chaque couleur par $1-n$.*

```
img[:,:,:]=1-img[:,:,:]
img=1-img # Autre solution
```

Plusieurs méthodes permettent de transformer une image en couleur en image en noir et blanc. Il est possible d'attribuer à chaque niveau de couleur d'un pixel la moyenne des niveaux de couleur du pixel considéré.