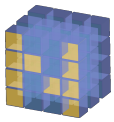
502



Traitement d'images avec numpy

Savoirs et compétences :

❒ *Savoir utiliser une bibliothèque de calcul scientifique.*

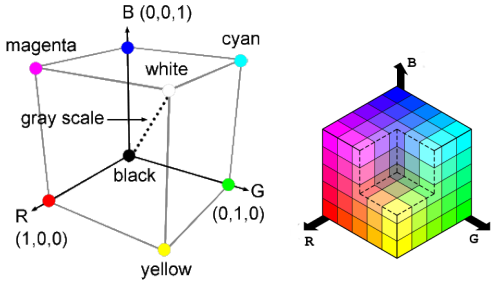
Représentation informatique d'une couleur

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 sur le site de la classe (ou toute autre image de résolution inférieure à 600 x 800).

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

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

Question 3 *Écrire deux fonctions* `conv_1_to_255(img:np.ndarray)->(img:np.ndarray)` *et* `conv_255_to_1(img:np.ndarray)->(img:np.ndarray)` *permettant respectivement :*

- *« d'étaler » chacun des niveaux de couleurs de 0 à 255 (en valeurs entières) ;*
- *« de normer » chacun des niveaux de couleurs de 0 à 1 (en valeurs flottantes) ;*

.

L'histogramme de couleurs représente le nombre de pixels pour chaque couche de couleur. En retouche d'image, la modification de l'histogramme peut par exemple modifier le contraste d'une photo.

Question 4 *Tracer l'histogramme de chacune des couleurs de l'image. Pour cela implémenter la fonction* `histo(img:np.ndarray)->(None)`.

R Il semble être plus rapide de créer la liste du nombre d'occurrences de chaque couleur de pixel puis de tracer cette liste, plutôt que d'utiliser la méthode `plt.hist`.

Jouons avec les couleurs, le noir et le blanc

Question 5 *Afficher uniquement les trames rouge, verte et bleue (pour la trame de rouge, seuls les pixels rouges seront conservés, les autres seront mis à 0).*

Pour saturer l'image en une couleur, le niveau de cette couleur doit être à 1 dans chacun des pixels de l'image.

Question 6 *Afficher les images saturées en rouge, vert et bleu.*

Question 7 *Réaliser une inversion des couleurs en remplaçant le niveau n de chaque couleur par $1-n$.*

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.

Question 8 *Transformer l'image en noir et blanc en utilisant la définition précédente. Pour cela, réaliser une fonction* `convert_nb_moy(img:np.ndarray)->(img:np.ndarray)` *qui permet de pondérer les niveaux de couleurs.*

Question 9 *Réaliser une transformation en noir et blanc en réalisant la moyenne du minimum et du maximum des 3 niveaux. On appellera cette fonction* `convert_nb_minmax(img:np.ndarray)->(img:np.ndarray)`

Amélioration du contraste

On va ici modifier le contraste d'une image en niveau de gris. Pour cela, on va accentuer les différences entre les niveaux en utilisant une fonction mathématique.

Question 10 *Donner la représentation des fonctions $f : x \mapsto \sqrt{x}$ et $g : x \mapsto \frac{1}{2}\left(1+\sin\left(\pi\left(x-\frac{1}{2}\right)\right)\right)$ sur l'intervalle $[0,1]$.*

Question 11 *Modifier le contraste d'une image en noir et blanc en remplaçant les niveaux de gris x par $f(x)$ et $g(x)$. Conclure.*

Convolution

La méthode de convolution permet, suivant les cas, de flouter ou de détecter des concours. Selon cette méthode, le niveau de couleur de chaque pixel est recalculé en fonction de ses voisins.

On note $M = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$. Chaque pixel (i,j) est alors remplacé ainsi : $\text{img}[i,j] = a \times \text{img}[i-1,j-1] + b \times \text{img}[i-1,j] + c \times \text{img}[i-1,j+1] + \ldots + i \times \text{img}[i+1,j+1]$.

Question 12 *Écrire la fonction* `convolution(image:np.ndarray,matrice:np.ndarray)->(img:np.ndarray)` *permettant d'appliquer la convolution à une image.*

Question 13 *Tester votre fonction avec les matrices suivantes :* $\begin{pmatrix} 1/8 & 1/8 & 1/8 \\ 1/8 & -1/8 & 1/8 \\ 1/8 & 1/8 & 1/8 \end{pmatrix}\begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & 1 \end{pmatrix}$. *Quel est l'effet de la convolution sur l'image ? Expliquer l'erreur affichée par l'interpréteur.*

Réduction d'image

Question 14 *Proposer un algorithme permettant de réduire la taille d'une image et le mettre en œuvre.*

▪ Python
```
def affichage3(img1,img2,img3):
    """ Affichage de trois images"""
    f, ((ax1, ax2,ax3)) = plt.subplots(1, 3, sharex='col', sharey='row')
    ax1.imshow(img1)
    ax2.imshow(img2)
    ax3.imshow(img3)
    plt.show()
```