

Traitement d'images avec Python - partie 2

Rappels

Packages pour manipuler les images

- `import matplotlib.pyplot as plt` # Bibliothèque pour afficher des images
- `from PIL import Image` # Bibliothèque complète dédiée aux images.

Un package qu'on aurait pu utiliser : - `import imageio` # outil d'entrée/sortie (I/O) pour des fichiers multimédia.

Ouvrir une image NB : Ces commandes ne sont pas à connaître.

- **avec PIL** : la variable `Img` contiendra un "objet" de type "Image". C'est un type de données géré par PIL et par pyplot :

- `Img = Image.open("nom_fichier.jpeg")`

Il faut parfois convertir l'image en "RGB", ce qui se fait avec l'instruction :

```
Img = Img.convert("RGB")
```

- **avec imageio** : la variable `Img` contiendra un "objet" de type "array numpy" ou "tableau numpy". C'est un type de données plus général, géré par imageio et aussi par pyplot.

- `Img = imageio.imread("nom_fichier.jpeg")`

Afficher une image NB : Ces commandes ne sont pas à connaître.

Que `Img` contienne une image PIL ou un tableau numpy, ou une image sous forme de liste de listes, les commandes sont les même :

```
plt.imshow(Img)
plt.axis("off") # Pour ne pas afficher les axes de la figure.
plt.show()
```

Convertir une image en liste de listes. NB : La seule chose à connaître est de savoir convertir une liste linéaire correspondant à une image en liste de listes.

- **avec PIL** :

```
# On récupère la hauteur et la largeur de l'image à l'aide d'une fonction "size" du module
largeur, hauteur = Img.size
# On convertit l'image une liste unidimensionnelle de pixels
ImgListeLineaire = list(Img.getdata())
# On découpe la liste en sous-listes par du slicing :
imgListeListes=[]
for k in range(hauteur):
    imgListeListes.append(ImgListeLineaire[k*largeur:(k+1)*largeur])
```

- **avec imageio** : c'est beaucoup plus facile puisqu'une instruction permet de faire ça :

```
imgListeListes = Img.tolist()
```

NB : dans toute la suite, on considérera qu'on dispose d'une image sous forme de liste de listes, stockée dans la variable `mon_image`.

Exercices

Exercice 1 : Obtenir les dimensions d'une image.

On considère que la variable `Img` contient une image codée en RGB (sous forme de liste de listes).

Ecrire une fonction `dim(Img)` qui renvoie la hauteur et la largeur de cette image.

Exercice 2 : Création d'une image unie

Ecrire une fonction `noire(h,l)` qui renvoie une image RGB noire, de hauteur `h` et de largeur `l`.

Exercice 3 : Conversion d'une image en niveau de gris.

On considère que la variable `Img` contient une image codée en RGB (sous forme de liste de listes). On veut la convertir en niveau de gris. Pour cela on utilise la méthode suivante. Pour chaque pixel, la valeur de gris `v` correspond à la moyenne pondérée des niveaux de rouge `r`, de vert `g` et de bleu `b` :

$$v = 0.299 * r + 0.587 * g + 0.114 * b$$

Ecrire une fonction `conv_gris(Img)` qui renvoie une image (sous forme de listes de listes) en niveau de gris à partir d'une image RGB.

Attention : ne pas oublier que les images sont codées par des entiers.

Exercice 4 : Conversion d'une image en noir & blanc binaire.

On considère que la variable `Img` contient une image codée en **niveau de gris** (sous forme de liste de listes). On veut la convertir en noir et blanc. Pour cela, on définit un seuil qui est un entier compris entre 0 et 255 : si la valeur d'un pixel est supérieure ou égale à ce seuil, on le code en blanc, sinon on le code en noir.

Ecrire une fonction `conv_NB(Img,seuil)` qui renvoie une image (sous forme de listes de listes) en noir et blanc à partir d'une image en niveau de gris, et qui permet de spécifier le seuil utilisé.

Exercice 5 : Tracé d'une ligne sur une image.

Ecrire une fonction `ligne(Img,y,c)` qui trace une ligne de couleur égale à `c` sur la ligne `y` de l'image `Img`.

Cette fonction ne devra pas modifier l'image originale, mais bien renvoyer une nouvelle image, indépendante de l'originale.

Exercice 6 : Assombrissement d'une image

Ecrire une fonction `assombri(Img)` qui assombri l'image RGB `Img` (liste de listes) en divisant la valeur de chaque pixel par 2.

Cette fonction ne devra pas modifier l'image originale, mais bien renvoyer une nouvelle image, indépendante de l'originale.

Exercice 7 : Rotation d'une image

Ecrire une fonction `rotationDirecte(Img)` qui renvoie l'image RGB `Img` tournée de 90° dans le sens direct.

Cette fonction ne devra pas modifier l'image originale, mais bien renvoyer une nouvelle image, indépendante de l'originale.

Ecrire ensuite une fonction `rotationIndirecte(Img)`