Traitement d'images

Gestion de données numériques

1 Image et codage

Les images en « **noir et blanc** », ou plutôt en niveaux de gris, peuvent être considérées comme des matrices 2D (nombre de lignes = hauteur, nombre de colonnes = largeur) où chaque élément de la matrice est un nombre caractérisant la luminosité d'un pixel.

Les images en **couleur** peuvent être considérées comme des matrices 2D (hauteur, largeur) dont chaque élément est une liste de 3 nombres correspondant à la décomposition de la couleur du pixel correspondant suivant les trois couleurs rouge, vert et bleu (format RVB).
En format RVBA, un 4ème nombre est ajouté pour indiquer la transparence du pixel.

Le module `matplotlib.image` est particulièrement adapté pour gérer les images, en combinaison avec les modules `numpy` (gestion des tableaux donc des matrices) et `matplotlib.pyplot` (affichage de graphes). On importe classiquement ces modules avec les alias suivants :

```
>>> import matplotlib.image as mi
>>> import matplotlib.pyplot as plt
>>> import numpy as np
```

1.1 Ouverture d'un fichier image

```
>>> img=mi.imread("nb.png")
```

Cette commande permet de récupérer l'image dans un tableau numpy nommé img. Attention à bien avoir défini au préalable le répertoire de travail de python, grâce à la fonction chdir du module os ou bien grâce à votre éditeur python (pyzo, pyscripter, geany ou autre...).

1.2 Caractéristiques du tableau récupéré

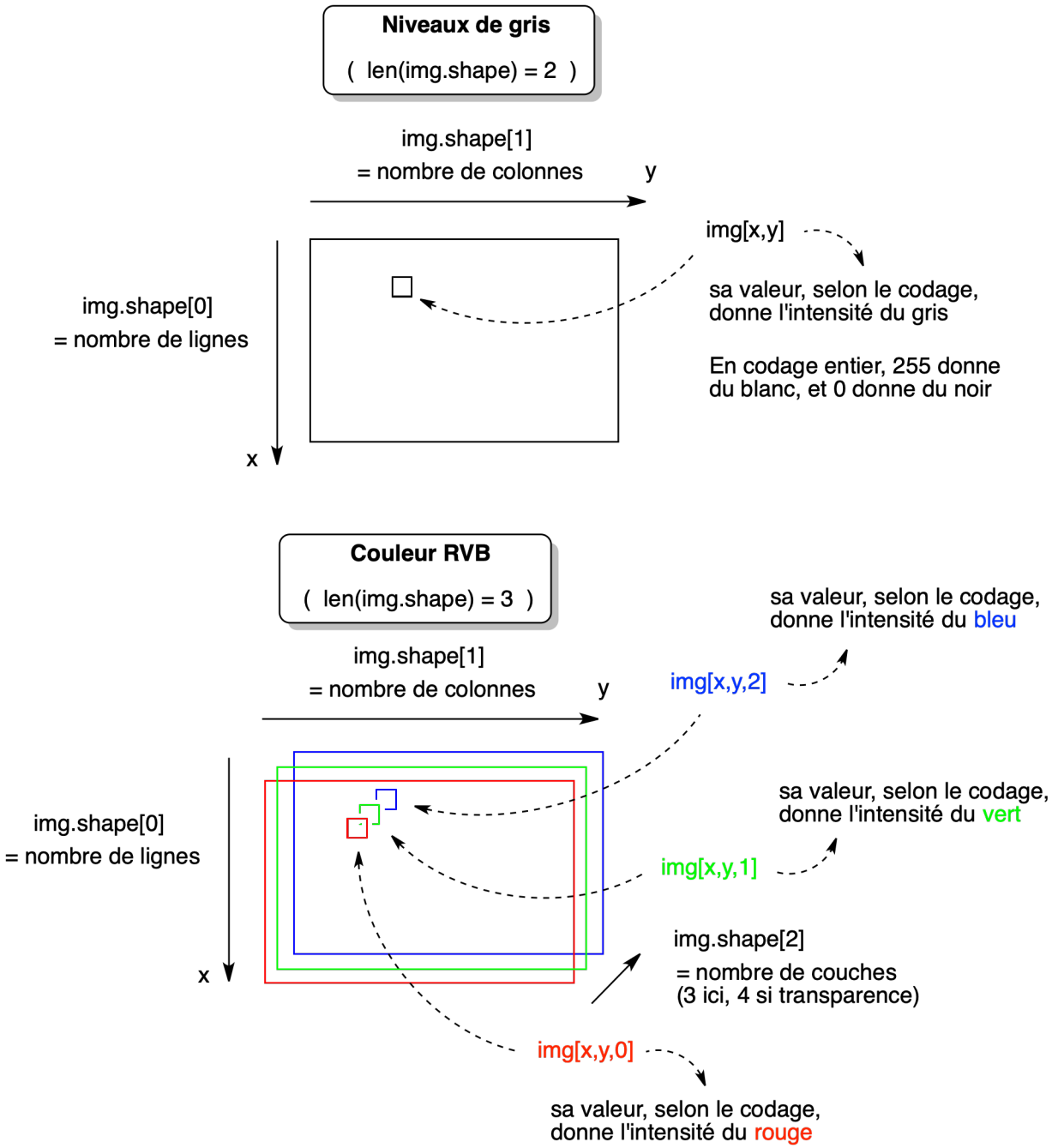
La méthode shape appliquée au tableau img permet de récupérer un tuple de deux ou trois entiers. Dans l'ordre :

- img.shape[0] est le nombre de lignes du tableau, donc la hauteur de l'image.
- img.shape[1] est le nombre de colonnes du tableau, donc la largeur de l'image.
- le cas échéant, img.shape[2] est l'épaisseur de la matrice qui vaut 3 pour un format RVB (et 4 pour un format RVBA).

L'objet img[x,y]contient la couleur du pixel à la position indiquée (attention : x est le n° de la ligne, y est le n° de la colonne). Le pixel de coordonnées (0,0) est par défaut celui en haut à gauche.

- Si l'image est d'épaisseur 1, la « couleur » est une luminosité, qui est soit codée par un réel entre 0 et 1, ou bien par un entier entre 0 et 255 (le type de codage est donné par img.dtype).
- Si l'image est en format RVB, la couleur est une liste de trois nombres : 3 réels compris entre 0 et 1, ou bien 3 entiers compris entre 0 et 255. Dans tous les cas, ces 3 nombres indiquent l'intensité relative dans le pixel des composantes rouge-vert-bleu dans cet ordre bien précis !

Remarque : Autrement dit, dans le cas d'un format RVB, img[x,y,0] (ou img[x,y][0]) est un nombre caractérisant l'intensité relative de la couleur rouge, img[x,y,1]est un nombre caractérisant l'intensité relative de la couleur verte, img[x,y,2]est un nombre caractérisant l'intensité relative de la couleur bleue (et éventuellement img[x,y,3]est un nombre caractérisant la transparence du pixel).



1.3 Affichage de l'image

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

Remarque : Pour visualiser les images en niveaux de gris, on peut ajouter d'autres paramètres à la fonction imshow, en particulier cmap='gray' pour que l'affichage soit en « vrais » niveaux de gris, ou cmap='jet' pour qu'il soit en fausses couleurs. Par défaut, l'image affichée sera un dégradé entre le jaune et le bleu.

Par ailleurs, l'ajout de la commande plt.colorbar() avant celui de plt.show() ajoute une barre de légende des couleurs.

1.4 Sauvegarde d'une image dans un fichier

```
>>> plt.imsave("sauvegarde.png",img)
```

Il est aussi possible d'enregistrer sous d'autres formats, comme pdf, jpg, svg, tif, ...

Remarque : Par défaut, la sauvegarde produit une image multicouche, même pour une image en N&B.

Activité 1 **S'approprier un code**

Ouvrir et exécuter le fichier `Caracteristiques_fichier.py`.
Analyser le code et le résultat obtenu en appliquant la fonction `caract` au fichier « nb.png » et « colibacille.jpg ».

2 Travail sur des images monocouches (donc en niveaux de gris)

2.1 Négatif d'une image

Activité 2 **Construire le négatif d'une image en niveaux de gris**

Construire un programme qui ouvre l'image « nb.png », puis « inverse » la valeur de l'intensité de chaque pixel (par exemple 0 donne 1 ; 0,6 donne 0,4 ; ...) afin d'afficher le négatif de l'image à l'écran.

Pour gagner du temps, vous pouvez opérer à partir d'une copie du programme `Caracteristiques_fichier.py`.

2.2 Éclaircir une image

Pour éclaircir une image, on demande à l'utilisateur le nombre d'unités à **ajouter** à chaque pixel (par exemple 150), et ajoute ce nombre à chaque pixel avant d'afficher l'image éclaircie. Pour rappel, 255 correspond au blanc.

Problèmes potentiellement rencontrés :

- attention à ne pas dépasser 255, qui est la valeur maximale en cas de codage de l'intensité par un entier (et correspond au blanc).
- le tableau récupéré à la lecture de l'image peut être protégé en écriture. Il est possible de lever cette protection en faisant une copie de ce tableau : img=img.copy() par exemple.
- L'affichage de l'image s'adapte automatiquement à l'écart de luminosité. Il est possible (et nécessaire ici) de forcer les valeurs minimales et maximales de luminosité à afficher en ajoutant en arguments de imshow : « vmin=0 », et « vmax=255 ».

Activité 3 **Éclaircir une image en niveaux de gris**

Construire un programme qui charge l'image « colibacille.jpg », demande à l'utilisateur le nombre d'unités à ajouter à chaque pixel pour éclaircir l'image, et ajoute ce nombre à chaque pixel avant d'afficher l'image éclaircie.

2.3 Seuillage

Pour seuiller une image, on demande à l'utilisateur un seuil (nombre entre 0 et 255), et on définit l'intensité de chaque pixel à la valeur 255 si le pixel de l'image initiale est supérieur ou égal au seuil, et 0 dans le cas contraire. On transforme ainsi une image en image noir et blanc, la limite étant de part et d'autre d'un seuil prédéfini, d'où le terme de seuillage.

Activité 4 **Seuiller une image en niveaux de gris**

Modifier le programme précédent pour demander à l'utilisateur un seuil (nombre entre 0 et 255), puis construire une nouvelle image de la même taille, mais avec pour intensité de chaque pixel la valeur 255 si le pixel de l'image initiale est supérieur ou égal au seuil, et 0 dans le cas contraire. On traitera l'image « colibacille.jpg ».

2.4 Détection de bordures

Pour détecter des bordures dans une image, on met un pixel à 255 si l'écart-type des intensités des 4 pixels voisins (haut, bas, gauche, droite) du pixel de l'image initiale est supérieur à un seuil, par exemple 10. Pour cela, on pourra utiliser la fonction np.std(L) qui renvoie l'écart-type d'une liste L passée en argument.

Activité 5 **Détection de bordures**

Construire un programme qui demande une valeur de seuil, et qui renvoie l'image avec détection des bordures. On traitera l'image « colibacille.jpg ».

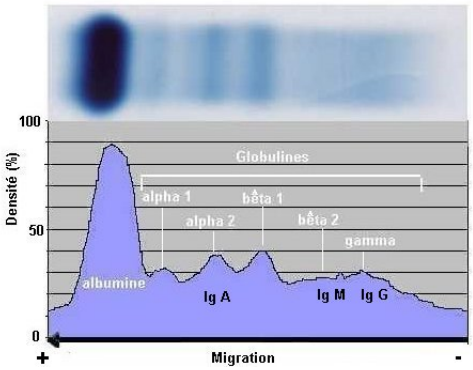
3 Travail sur des images multicouches (notamment en couleurs)

Le principe est le même que pour les images en niveaux de gris, sauf que la couleur est indiquée par une liste de trois nombres, chaque nombre indiquant la valeur de la luminosité dans le rouge, dans le vert et dans le bleu.

3.1 Spectre (ou profil densitométrique) d'une électrophorèse

L'image ci-dessous provient du site http ://www.snv.jussieu.fr/bmedia/ATP/bioch6.htm et contient dans sa partie supérieure la photographie d'une électrophorèse des protéines sériques, et dans sa partie inférieure le profil densitométrique de cette électrophorèse.

Le fichier « bandes.jpg » contient la photographie d'origine. Dans ce fichier, le codage est entier, en RVB. Pour rappel, cela signifie que chaque couleur R, V et B a une composante comprise entre 0 et 255. Le rouge pur sera codé (255,0,0). À titre d'illustration, on pourra consulter le site https ://www.toutes-les-couleurs.com/code-couleur-rvb.php



On peut obtenir le profil densitométrique en calculant l'absorbance, par exemple dans le rouge, de chaque pixel d'une ligne horizontale ; l'absorbance dans le rouge peut être définie à partir de la valeur $1 - \dfrac{\text{img[x, y, 0]}}{255}$.

On peut facilement expliquer cette définition : $\dfrac{\text{img[x, y, 0]}}{255}$ mesure le pourcentage de rouge **visible**. $\dfrac{255 - \text{img[x, y, 0]}}{255}$ mesure donc le pourcentage de rouge **absorbé**. On a bien le fait que l'absorbance peut être définie par $1 - \dfrac{\text{img[x, y, 0]}}{255}$.

En résumé, pour un codage entier :

Absorbance dans le ...	Formule
Rouge	$1 - \dfrac{\text{img[x, y, 0]}}{255}$
Vert	$1 - \dfrac{\text{img[x, y, 1]}}{255}$
Bleu	$1 - \dfrac{\text{img[x, y, 2]}}{255}$

Activité 6 **Tracé d'un profil densitométrique**

Construire un programme qui permet de tracer le graphique représentant la densité d'absorption selon une ligne horizontale (au milieu de l'image, par exemple), en fonction de l'abscisse x du pixel.

Rappel : plt.plot(X,Y) permet de tracer le graphique de la liste Y en fonction de la liste X.

3.2 Spectre moyenné d'une électrophorèse

S'il y a une poussière sur la bande d'électrophorèse, ou une irrégularité quelconque, la mesure de l'absorbance pourrait être mauvaise.

Activité 7 **Tracé d'un profil densitométrique moyenné**

Pour avoir une meilleure qualité du profil, modifier le programme précédent pour que chaque valeur d'absorbance soit une moyenne de l'absorbance des 21 pixels au milieu de l'image (celui du milieu + 10 au dessus + 10 en dessous).