TRAITEMENT D'IMAGE AVEC NUMPY

1. Format numérique d'une image

Une image peut être numérisée sous forme d'image matricielle (en anglais « bitmap ») par une matrice de points colorés.

Cette matrice a *n* lignes (la hauteur de l'image) et *p* colonnes (la largeur). L'élément (i, j) représente un pixel, c'est-à-dire une portion d'image considérée de couleur constante.

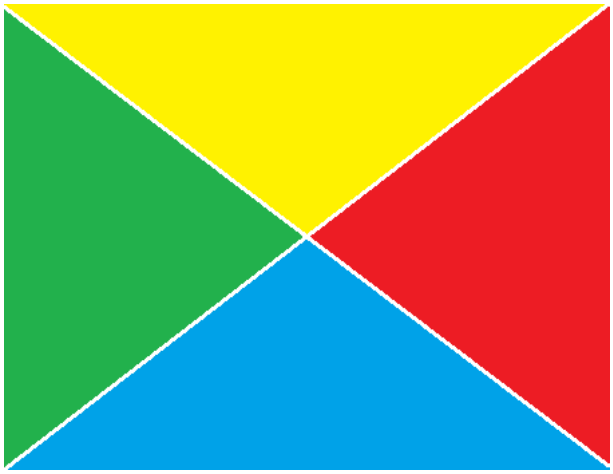
Il existe plusieurs manières de coder la couleur. La méthode accessible avec la bibliothèque **matplotlib.image** est la méthode RGB. Chaque couleur est représentée par une liste à trois entrées [r,g,b] où r, g et b sont trois réels représentant respectivement la quantité de rouge, de vert et de bleu que contient la couleur. La méthode de mélange des couleurs est la synthèse additive : on peut penser à trois spots de couleurs qui éclairent un fond noir. Le mélange des trois lumières colorées crée la couleur désirée.

Si r = g = b = 0, le pixel est noir. Si r, g et b ont leur valeur maximale, le pixel est blanc. Cette valeur maximale dépend de l'image et du format d'image : il peut être de 1 ou de 255.

Supposons que la valeur maximale soit à 1 : [1,0,0] est du rouge pur, [0,1,1] est le cyan (complémentaire du rouge), etc. Les couleurs ayant des proportions identiques de rouge, vert et bleu [x,x,x] sont des gris de plus en plus clair lorsque x augmente.

2. Création d'une image

Avec numpy une image est donc codée avec une matrice A de dimension 3. La première dimension représente le numéro de la ligne, la seconde le numéro de la colonne et la dernière le numéro de la couleur.



Ecrire une fonction create_img(name,h,w) qui crée l'image ci-dessus de taille $h \times w$ et l'enregistre en format png sous le nom 'name'.

Utiliser la libraire matplotlib et numpy.

3. Manipulation d'une image

La librairie matplotlib permet de manipuler les images sous format de tableau numpy à l'aide de la fonction imread(filename).

Les couleurs sont codées par 3 flottants entre 0 et 1. Comme img est un tableau, il est facile de le manipuler avec les méthodes de *numpy* ou avec des boucles. Pour cela on a besoin d'en connaître les dimensions à l'aide de l'attribut *shape*.

Partie I :

Ecrire les fonctions suivantes qui reçoivent par paramètre le nom d'un fichier image, transforment cette image, et sauvegardent le résultat dans un fichier.

1. Inversion de couleurs : la valeur x de chaque couleur est remplacé par 1 – x
2. Séparation en trois images : l'une avec la composante rouge, une deuxième avec la composante verte et la troisième avec la composante bleue.
3. Transformation en niveau de gris :
    a. en remplaçant le rouge, vert et bleu par la moyenne des trois couleurs.
    b. en remplaçant chaque niveau de gris par la moyenne pondérée 0, 21 × R + 0, 71 × V + 0, 07 × B (la « luminosité » du pixel)
    c. en calculant le minimum et le maximum des trois composantes.
4. Transformation en noir et blanc.

Partie II : amélioration du contraste

Pour améliorer le contraste, on transforme le niveau de gris d'entrée en un nouveau niveau de gris dans le but d'exagérer les différences entre les niveaux. Pour cela on applique une fonction « bien choisie ».

1. Améliorer le contraste en utilisant une fonction racine carrée : on remplace chaque niveau d'entrée x par $\sqrt{x}$. Comparer avec l'usage d'un courbe « en S » (un morceau de sin convenablement transformé). $f(x) = \frac{1}{2} + 1/2 \times \sin(\pi(x - \frac{1}{2}))$
2. Expliquer en quoi la transformation précédente améliore le contraste de l'image.
3. Quel type de fonction est susceptible d'être utile ?

Partie III : Convolution

La convolution 2D est un traitement d'image où la valeur de chaque pixel est recalculée en fonction des valeurs des pixels voisins. Cherchons par exemple à « flouter » une image. La valeur du pixel [i,j] est remplacé par la moyenne pondérée des valeurs des pixels voisins.

1. Ecrire une fonction *convolution_2D(img,matrice)* qui prend en paramètre une image et une matrice 3x3 de la forme : $M = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$ et qui effectue l'opération de convolution décrite précédemment.
2. Ecrire une fonction *flouter(img,matrice)* qui applique floutage sur une *img* selon une convolution *matrice.*
3. Dans une image, les bords des objets correspondent à des zones où les valeurs des pixels changent rapidement. C'est le cas par exemple lorsque l'on passe d'un objet clair (avec des valeurs grandes) à un arrière-plan sombre (avec des valeurs petites).

   Considérons une image à laquelle on applique une convolution avec la matrice :

   $$M = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

   a. Que se passe-t-il pour les pixels dans des zones uniformes ? Et pour des pixels dans des zones à fort contraste.

   b. Appliquer cette convolution à une image. Commentez.

   c. Dans certaines images, comme coupe.png on s'intéresse à une détection de bord dans une certaine direction. Proposez une convolution pour se faire.