Échauffement

Exercice 1 (𝄞⋆)**.** Écrire une fonction `Moyenne(L)` qui, pour une liste `L` non vide de nombres, renvoie la moyenne des éléments de cette liste.

Listes et Matrices

Pour représenter une matrice, on peut [1] l'écrire comme une liste de listes, c'est-à-dire une liste dont les éléments sont eux-mêmes des listes. Par exemple,

```
L=[[1,2,3],[5,8,-1]]
```

peut être vu comme une matrice dont la première ligne est 1,2,3 et la seconde est 5,8,-1. Ainsi, `L[1][1]` vaut 8. De manière générale, `L[i][j]` est le coefficient à la `i`-ième ligne et `j`-ième colonne. Le nombre de lignes est donné par `len(L)` et le nombre de colonnes par `len(L[0])`.

Attention aux indices qui commencent à 0

En python, les lignes et les colonnes des matrices commencent donc à 0 contrairement aux maths.

Pour définir une liste de longueur `p` ne contenant que des 0, on peut utiliser la commande :

```
[0 for j in range(p)]
```

Pour définir la matrice nulle à `n` lignes et `p` colonnes, on écrit :

```
Z=[[0 for j in range(p)] for i in range(n)]
```

Images en Python

Une image en noir et blanc peut être vue comme une matrice `M`. Pour un couple `(i,j)` donnée, `M[i][j]` désigne l'intensité du pixel situé à la ligne `i` et la colonne `j`. Cette valeur est traditionnellement donné entre 0 et 255, à 0 le pixel est noir, à 255 il est blanc, les valeurs entre 0 et 255 codent un gris plus ou moins foncé.

Si l'image est en couleur, alors chaque pixel à un niveau de rouge, de vert et de bleu, ainsi `M[i][j]` au lieu d'être un nombre sera une liste de trois nombres, chacun représentant le niveau dans une des trois couleurs.

Télécharger le fichier `TP8Image.jpg` qui se trouve sur `CDP`. Mettez-le dans le même dossier que votre fichier Python actuel, alors les commandes suivantes transforment l'image en une matrice. Attention, vous devez la première fois que vous exécutez ces commandes utiliser la combinaison `CTRL+SHIFT+E`, sinon Pyzo ne trouve pas le fichier [2].

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

Img=imageio.imread("NomDelImage.jpg").tolist()
#Convertit une image en liste#cette commande n'est pas à connaître.
```

À partir de maintenant, votre image est sous forme d'une matrice notée `Img`, lorsque l'on mentionnera l'image, on fera référence à cette matrice `Img` et non au fichier original `TP8Image.jpg`. Tester ce que vaut `Img[3][5]`, trouver le nombre de lignes et de colonnes de `Img`. Pour afficher l'image `Img` :

```
plt.figure()#Commandes non exigibles, crée une nouvelle figure
plt.imshow(Img)
plt.title("Image originale")#Titre de la figure
plt.show()#Affiche la figure
```

Manipulons les images (comme les média ?)

Pour l'instant, `Img` est une image en couleur, cela veut dire que si `i` est un numéro de ligne et `j` un numéro de colonnes `Img[i][j]` est une liste de trois nombres représentant l'intensité de rouge, vert et bleu. Pour simplifier ce TP, on va d'abord transformer cette Image en noir et blanc.

Exercice 2 (⋆)**.** Écrire une fonction `ConversionNB(Img)` qui va transformer `Img` en une image en noir et blanc. Pour cela, il suffit de remplacer `Img[i][j]` (actuellement une liste de trois nombres) par la moyenne de cette liste (grâce à la fonction moyenne déjà créée). Attention, cette fonction doit directement modifier `Img` et n'aura donc pas de `return`. Une fois fait, tester votre fonction avec votre image. Pour l'afficher, utilisez la commande : `plt.imshow(Img,cmap="gray")` afin d'indiquer à Python que vous voulez du noir et blanc [3].

Exercice 3 (⋆)**.** Écrire une fonction `Negatif(Img)` qui a une image `Img`, va renvoyer une autre image qui est son négatif. Plus précisément, à partir de la matrice nulle et de même taille que `Img`, la valeur d'un pixel de cette image sera la différence entre 255 et la valeur du même pixel de l'image d'origine.

1. Ce n'est pas la seule possibilité, on peut aussi utiliser les matrices de `numpy` mais ce n'est pas au programme.

2. Ceci est vraiment propre à Pyzo et pas à Spyder.

3. Sinon, il essaye de la colorer avec des pseudo-couleurs.

Exercice 4 (⋆)**.** Écrire une fonction `Symetrie(Img)` qui retournera l'image symétrique de `Img` par un axe vertical qui passe par le centre de l'image. Pour cela, pour un pixel `(i,j)` et son pixel symétrique `(i',j')` donner une relation entre `i` et `i'` et `j` et `j'`, puis à l'aide de boucles, échanger leur valeur.

Exercice 5 (⋆)**.** Écrire une fonction `DessineContours(Img)` qui va renvoyer une nouvelle image contenant les contours des objets de l'image `Img`. Pour cela, créer une nouvelle image `P` telle que [4]

```
P[i][j]=|Img[i+1][j]-Img[i-1][j]|+|Img[i][j+1]-Img[i][j-1]|
```

La fonction `abs` dans Python renvoie la valeur absolue d'un nombre.

Rajouter du bruit

Supposons que l'on souhaite rajouter du bruit, c'est-à-dire que l'on veuille dégrader sa qualité. Alors, il suffit pour chaque pixel de rajouter un nombre aléatoire. Les lignes suivantes font ce travail :

```
def Bruit(Img,r):
    """Crée du bruit d'un niveau r sur une image"""
    n,p=len(Img),len(Img[0])
    B=[[Img[i][j]+r*(np.random.rand()-0.5)
    for j in range(p)] for i in range(n)]
    return B

imageBruit=Bruit(Img,100)
plt.figure()
plt.imshow(imageBruit,cmap="gray")
plt.title("Image bruitée")
plt.show()
```

Filtrage par convolution

Pour traiter une image, il est courant d'appliquer un filtre. Un filtre peut essayer de débruiter l'image (atténuer le bruit), augmenter le contraste, créer une image avec les contours des objets etc. Pour ce faire ici, les filtres seront des matrices carrées [5] $M = (m_{i,j})_{\substack{0\leqslant i\leqslant 2d\\0\leqslant j\leqslant 2d}} \in \mathscr{M}_{2d+1}(\mathbb{R})$. Si on a une image représentée par une liste `Img`, alors on va convoluer `Img` avec M plus précisément, la valeur du pixel de la nouvelle image au pixel `(i,j)` est :

$$\sum_{k=-d}^{d}\sum_{\ell=-d}^{d} Img[i+k][j+\ell]\times M[d-k][d-\ell]$$

Voilà plusieurs matrices qui serviront de filtres :

$$M_1 = \begin{pmatrix} 0 & 1/4 & 0\\ 1/4 & 0 & 1/4\\ 0 & 1/4 & 0\end{pmatrix} \quad M_2 = \begin{pmatrix} 0 & 0 & 1/36 & 0 & 0\\ 0 & 1/18 & 1/9 & 1/18 & 0\\ 1/36 & 1/9 & 2/9 & 1/9 & 1/36\\ 0 & 1/18 & 1/9 & 1/18 & 0\\ 0 & 0 & 1/36 & 0 & 0\end{pmatrix}$$

$$M_3 = \begin{pmatrix} 0 & -1 & 0\\ -1 & 5 & -1\\ 0 & -1 & 0\end{pmatrix} \quad C = \frac{1}{25}J \quad S = \begin{pmatrix} 1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{pmatrix} \quad R = S^T$$

Où $J \in \mathscr{M}_5(\mathbb{R})$ est la matrice qui n'a que des 1. Concrètement le filtrage par la matrice M_1 va remplacer la valeur d'un pixel `(i,j)` par la moyenne des pixels voisins de ce pixel (ceux qui sont directement à gauche, à droite, en haut ou en bas de lui).

Exercice 6 (𝄞⋆⋆)**.** Écrire une fonction `Filtre(Img,M)` qui applique le filtre décrit par la matrice M à `Img` et retourne l'image filtrée. Essayez avec comme filtre M_1, M_2 puis M_3 et avec comme image, l'image original puis l'image bruitée, à quoi servent ces filtres ? Enfin, appliquez les filtres, dits de Sobel, en appliquant successivement à la même image les matrices `C`, `S` et `R`. Puis, à partir de la matrice nulle, on mettra à 255 les pixels dont l'intensité de l'image, obtenue après les filtres de Sobel, dépasse un seuil de 10. À quoi cela sert-il ?

4. La valeur d'un pixel sera d'autant plus grande que les variations entre les voisins de ce pixels seront grands. Cela veut dire dans ce cas, qu'on est à une frontière entre deux parties de l'image de niveaux de gris très différents.

5. Ici, on écrit la matrice en commençant les indices à 0 pour coller aux conventions de Python.