TD - Piles et files

C.F. sujet

I - Introduction

C.F. sujet

II - Représentation d'une image

C.F. sujet

III - Remarques et modules utiles au traitement d'images

C.F. sujet

IV - Fonctions de bases

C.F. sujet

IV.1 - Exercice 1 : Chargement d'une image

1. Créer l'objet associé à l'image "image_couleurs", regarder sa taille, son format, et son mode

```
import numpy as np
import matplotlib.pyplot as plt
import PIL.Image as im


img = im.open("Ressources/original.jpg")

imSize = img.size
imMode = img.mode
imFormat = img.format

print("size:\t{}\nmode:\t{}\nformat:\t{}".format(imSize, imMode, imFormat))
```

2. A partir de cet objet faire afficher l'image

```
# Open with the default image viewer of the PC
img.show()

# Open into a matplotlib graph
plt.imshow(img)
plt.show()
```

3. Récupérer les pixels de l'image sous forme d'un tableau

```
pixelArray = np.array(img)
```

IV.2 - Exercice 2 : Conversion d'une image couleur en niveau de gris

1. Convertir l'image en noveau de gris avec la fonction `convert` et faire afficher le résultat

```
# On définit les variables globales permettant de gérer les subplot
numberOfPlots = 1
size = (1, 1)


# On crée une fonction qui affiche l'image dans une figure pyplot
def plotImage(image, s=None, show=False):
    global numberOfPlots
    global size
    if s:
        size = s
    plt.subplot(size[0], size[1], numberOfPlots)
    plt.axis('off')
    mode = 'L'
    if type(image) == np.ndarray:
        if len(image) == 3:
            mode = 'RGB'
    else:
        if image.mode == 'RGB':
            mode = 'RGB'
    if mode == 'L':
        plt.imshow(image, cmap='gray')
    else:
        plt.imshow(image)

    numberOfPlots += 1
    if show:
        plt.show()


# On crée l'image en noir et blanc
greyImg = img.convert('L')

# On affiche les images
plotImage(greyImg, (1, 2))
plotImage(img, show=True)
```

2. Ecrire une fonction qui retroune un tableau de niveau de gris d'une image

```
def geryScale(image):
    # On définit les valeurs de luminance
    R, G, B, = 0.299, 0.587, 0.114

    # On récupère les couleurs de l'image séparemment
    r, g, b, = image.split()

    # On les ajoute dans la nouvelle image
    return r * np.array(R) + g * np.array(G) + b * np.array(B)


# On affiche le tout
plotImage(img, (1, 2))
plotImage(geryScale(img), show=True)
```

3. Créer une image depuis le tableau précédent

```
def arrayToImage(array):
    s = array.shape
    if len(s) == 3:
        mode = 'RGB'
        array = array.reshape(s[0] * s[1], 3)
        array = [(r, g, b) for r, g, b in array]
    else:
        mode = 'L'
        array = array.flat
    newIm = im.new(mode, (s[1], s[0]))
    data = list(array)
    newIm.putdata(data)
    return newIm


greyScaleImg = arrayToImage(geryScale(img))
```

4. Enregistrer cette image

```
greyScaleImg.save('Ressources/image_grise.jpg')
```

IV.3 - Exercice 3 : Transformation d'une image

1. Réaliser et faire afficher une symétrie par rapport à l'axe vertical

```
def vSymetry(greyScaleArray):
    reversedArray = np.empty_like(greyScaleArray)
    l = len(reversedArray)-1
```

```
    for i in range(l, -1, -1):
        reversedArray[l-i] = greyScaleArray[i]
    return reversedArray


def hSymetry(greyScaleArray):
    return (np.transpose((vSymetry(np.transpose(greyScaleArray)))))


plotImage(geryScale(img), (1, 2))
plotImage(hSymetry(geryScale(img)), show=True)
```

2. Réaliser et faire afficher une symétrie par rapport à l'axe horizontal

```
plotImage(geryScale(img), (2, 1))
plotImage(vSymetry(geryScale(img)), show=True)
```

3. Réaliser et faire afficher une rotation de 20%

```
def cos(x): return np.cos(x)


def sin(x): return np.sin(x)


def rotation(greyScaleArray, percentage):
    height, width = greyScaleArray.shape
    alpha = (percentage / 100) * (np.pi / 2)
    c, s = cos(alpha), sin(alpha)
    w = int(width * c + height * s)
    h = int(height * c + width * s)
    newArray = np.zeros((h, w))
    rMatrix = np.array(np.array(((c, -s), (s, c))))
    for y in range(height):
        for x in range(width):
            try:
                nY, nX = np.dot(rMatrix, np.array((y, x)))
                nY = nY + width * s
                newArray[int(nY), int(nX)] = greyScaleArray[y][x]
            except IndexError:
                pass
    return (newArray)


plotImage(rotation(geryScale(img), 20), show=True)
```

4. Faire afficher l'image en négatif

```
def negative(greyScaleArray):
    return -greyScaleArray

plotImage(negative(geryScale(img)), show=True)
```

5. Faire afficher l'image en noir et blanc avec un seuil à 127

```
def seuil(greyScaleArray, seuil):
    return np.where(greyScaleArray < seuil, 0, 255)

plotImage(seuil(geryScale(img), 127), show=True)
```