Informatique - DS 3

Proposition de corrigé

Exercice : Image

On souhaite réaliser diverses transformations de l'image en couleur ci-contre. Des programmes sous forme de fonctions Python sont présentés dans l'annexe (*les lignes sont numérotées à gauche*).

1. Commenter les lignes 1 et 4 de l'entête.

La ligne 1 importe la bibliothèque `numpy` pour effectuer des calculs numériques.
Pour appeler une fonction de cette bibliothèque il suffira de taper : `np.fonction` comme aux lignes 8, 22 et 33.

La ligne 4 utilise la fonction `image.imread` de la bibliothèque `mathplotlib` pour importer le fichier compressé image `Cocotiers.png` et le transformer en tableau (type `array`) qui sera appelé `Image` dans la suite du programme.

2. Quelle est la différence entre le fichier `Cocotiers.png` et la variable `Image` ?

Le fichier `Cocotiers.png` est une image stockée dans l'ordinateur (dans le même dossier que celui où se trouve le programme Python) au format compressé `.png`.
La variable `Image` est un tableau de nombres (type `array`) manipulable par les instructions d'un programme Python.

3. fonction `flou`

a. à la ligne 11, dire à quoi correspondent les lettres r, v et b.

Rouge, vert, bleu, les trois couleurs fondamentales qui permettent de restituer toutes les couleurs de façon additive.

b. Que réalise la fonction `flou` ?
La fonction `flou` floute l'image en affectant à chaque pixel une valeur égale à la moyenne de ce pixel et des huit pixels qui l'entourent.
Elle renvoie non pas le tableau de l'image floutée, mais affiche directement cette image par l'instruction `plt.implot`.

4. fonctions `conv` et `masque`

a. Que réalise la fonction `masque` ?

La fonction `masque` permet à l'utilisateur de saisir un masque constitué d'un carré de $2x+1$ pixels de côté où l'on rentre une valeur pour chaque pixel.

b. Que réalise la fonction `conv` ?

La fonction `conv` appelle la fonction `masque` puis affecte à chacune des trois couleurs de chaque pixel la somme des valeurs des couleurs des pixels avoisinants affecté du coefficient obtenu dans la fonction `masque`.

c. Proposer un masque qui permette à l'aide de la fonction `conv` de réaliser la fonction `flou`
`[ [1/9,1/9,1/9] , [1/9,1/9,1/9] , [1/9,1/9,1/9] ]`

5. Propositions de programme
Choisir, en fonction du temps disponible et de ses affinités, parmi les propositions suivantes de réalisation de programme.

a. Proposer un programme qui réalise un floutage prenant en compte un carré contenant $2x+1$ pixels, x étant un entier naturel non nul.

```
def floux(im,x):
    n, p, s = im.shape
    nbpix=(2*x+1)**2 # nombre de pixels concernÃ©s
    flou= np.zeros_like(im)
    for i in range(x,n-x):
        for j in range(x,p-x):
            r,v,b=0,0,0
            for n in range(-x,x+1):
                for m in range(-x,x+1) :
                    r=r+im[i+n,j+m,0]
                    v=v+im[i+n,j+m,1]
                    b=b+im[i+n,j+m,2]
            flou[i, j] =[r/nbpix,v/nbpix,b/nbpix]
    return plt.imshow(flou)
```

b. Proposer un programme qui renvoie le symétrique de l'image par rapport à un axe de symétrie horizontal.

```
def symetrie(im):
    n, p, s = im.shape
    sym = np.zeros_like(im)
    for i in range (n):
        for j in range(p):
            sym[i, j] = im[n-1-i, j]
# rempli les lignes dans l'ordre inverse
    return sym

def symetrie2(im):
# plus compact
    n, p, s = im.shape
    sym = np.zeros_like(im)
    for i in range (n):
        sym[i] = im[n-1-i]
    return sym
```

On voudrait pouvoir inverser directement les lignes du tableau `im` pour alléger encore le script, mais ce tableau n'est pas modifiable.

c. Proposer un programme qui réalise une rotation de l'image de $\frac{\pi}{2}$ dans le sens trigonométrique

```
def rotation(im):
    n, p, s = im.shape
    rot = np.zeros((p, n, s),dtype=np.uint8)
    for i in range(n):
        for j in range(p):
            rot[j, i]=im[i,p-1-j]
    return rot
```

d. Proposer un programme qui créée le négatif de l'image

```
def negatif(im):
    n, p, s = im.shape
    neg = np.zeros((n, p, s),dtype=np.uint8)
    for i in range(n):
        for j in range(p):
            for k in range(3):
                neg[i,j,k] = 256 - im[i,j,k]
    return neg
```

e. Proposer un programme qui permet l'affichage de l'image en niveaux de gris

```
def niveaudegris(im):
    n, p, s = im.shape
    gris = np.zeros((n,p,s),dtype=np.uint8)
    for i in range(n):
        for j in range(p):
            for k in range(3):
                gris[i,j,k] =int((im[i,j,0]+im[i,j,1]+im[i
,j,2])/3)
    return gris
```

f. Proposer un programme qui permet l'affichage de l'image en noir et blanc (*chaque pixel est soit tout noir, soit tout blanc*)

```
def noirblanc(im):
    n,p,s=im.shape
    nb= np.zeros((n,p,s),dtype=np.uint8)
    for i in range(n):
        for j in range(p):
            luminositÃ©=im[i,j,0]+im[i,j,1]+im[i,j,2]
            if luminositÃ©>95: # seuil
                pixel=255
            else:
                pixel=0
            nb[i,j,0],nb[i,j,1],nb[i,j,2]=pixel, pixel ,
pixel
    return nb
```

g. Proposer un programme qui « étire » l'image d'un facteur 2 dans le sens « horizontal »

```
def etirement(im):
    n,p,s=im.shape
    etire = np.zeros((n,2*p,s),dtype=np.uint8)
    for i in range(n):
        for j in range(p):
            etire[i,2*j],etire[i,2*j+1]=im[i,j],im[i,j]
    return etire
```

Annexe

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
Image=matplotlib.image.imread("Cocotiers.png")

def flou(im):
    n, p, s = im.shape
    flou= np.zeros_like(im)
    for i in range(1,n-1):
        for j in range(1,p-1):
            r,v,b=0,0,0
            for n in range(-1,2):
                for m in range(-1,2) :
                    r=r+im[i+n,j+m,0]
                    v=v+im[i+n,j+m,1]
                    b=b+im[i+n,j+m,2]
            flou[i, j] =[r/9,v/9,b/9]
    return plt.imshow(flou)

def masque(x):
    taille = 2*x+1
    masque=np.zeros((taille,taille))
    for n in range(taille):
        for m in range(taille):
            print('ligne',n,'colonne',m,' : ')
            masque[n,m]=input()
    return masque

def conv(im,x):
    n, p, s = im.shape
    ma=masque(x)
    print('Traitement en cours...')
    conv= np.zeros_like(im)
    for i in range(x,n-x):
        for j in range(x,p-x):
            r,v,b=0,0,0
            for n in range(-x,x+1):
                for m in range(-x,x+1) :
                    r=r+im[i+n,j+m,0]*ma[n,m]
                    v=v+im[i+n,j+m,1]*ma[n,m]
                    b=b+im[i+n,j+m,2]*ma[n,m]
            conv[i, j] =[r,v,b]
    return plt.imshow(conv)
```