Corrigé : images en gris (X psi 2011)

Ceux d'entre vous qui veulent expérimenter sur leur machine les algorithmes mis en œuvre dans ce problème trouveront en annexe la définition de la classe `Image` ainsi qu'une fonction pour convertir une image de votre choix (par exemple la fameuse Lena qui illustre l'énoncé) en un élément de cette classe.

Partie I. Opérations élémentaires

Question 1.

```
def inverser(img):
    img_inv = Image(img.h, img.l, img.p)
    for i in range(img.h):
        for j in range(img.l):
            img_inv.m[i, j] = img.p - img.m[i, j]
    return img_inv
```

Question 2.

```
def flipH(img):
    img_flipH = Image(img.h, img.l, img.p)
    for i in range(img.h):
        for j in range(img.l):
            img_flipH.m[i, j] = img.m[i, img.l - 1 - j]
    return img_flipH
```

Question 3.

```
def poserV(img1, img2):
    imgV = Image(img1.h + img2.h, img1.l, img1.p)
    for i in range(img1.h):
        for j in range(img1.l):
            imgV.m[i, j] = img1.m[i, j]
    for i in range(img2.h):
        for j in range(img2.l):
            imgV.m[img1.l + i, j] = img2.m[i, j]
    return imgV
```

Question 4.

```
def poserH(img1, img2):
    imgH = Image(img1.h, img1.l + img2.l, img1.p)
    for i in range(img1.h):
        for j in range(img1.l):
            imgH.m[i, j] = img1.m[i, j]
    for i in range(img2.h):
        for j in range(img2.l):
            imgH.m[i, img1.l + j] = img2.m[i, j]
    return imgH
```

Partie II. Transferts

Question 5.

```
def transferer(img, q, t):
    imgt = Image(img.h, img.l, q)
    for i in range(img.h):
        for j in range(img.l):
            imgt.m[i, j] = t[img.m[i, j]]
    return imgt
```

Question 6. On obtient l'image inversée en posant $q = p$ et en utilisant le tableau $t = [p, p-1, \ldots, 2, 1, 0]$.

```
def inverser2(img):
    t = [k for k in range(img.p, -1, -1)]
    return transferer(img, img.p, t)
```

Question 7.

```
def histogramme(img):
    h = [0 for k in range(img.p+1)]
    for i in range(img.h):
        for j in range(img.l):
            h[img.m[i, j]] += 1
    return h
```

Question 8. La formule permet de remplir un tableau de transfert pour ensuite utiliser la fonction `transferer`.

```
def egaliser(img):
    h = histogramme(img)
    vmin = 0
    while h[vmin] == 0:
        vmin += 1
    t = [0 for k in range(img.p+1)]
    s = 0
    for v in range(vmin, img.p+1):
        s += h[v]
        t[v] = int(round(img.p * (s - h[vmin]) / (img.h * img.l - h[vmin]), 0))
    return transferer(img, img.p, t)
```

Question 9. Si une image est uniformément blanche, tout pixel (i, j) a pour ton $v = p$ donc $v_{\min} = p$ et $v = p \Rightarrow v' = 0$. Mais alors $h_i[v_{\min}] = h_i(p) = h \times l$. Le dénominateur de la formule s'annule et la fonction retourne le message d'erreur `ZeroDivisionError`.

Question 10.

```
def reduire(img, q):
    t = [0 for k in range(img.p)]
    for v in range(img.p):
        t[v] = int(round(v * q / img.p, 0))
    return transferer(img, q, t)
```

Partie III. Tramage

Question 11.

```
def tramer(img, mat):
    img_tr = Image(img.h, img.l, 1)
    for i in range(img.h):
        for j in range(img.l):
            if img.m[i, j] > mat.m[i, j]:
                img_tr.m[i, j] = 1
            else:
                img_tr.m[i, j] = 0
    return img_tr
```

Question 12. Inutile ici d'utiliser la fonction `tramer` puisqu'il est facile de calculer le ton d'un pixel de la matrice associée au tramage.

```
def tramerTelevision(img):
    img_tr = Image(img.h, img.l, 1)
    for i in range(img.h):
        for j in range(img.l):
            if img.m[i, j] > i % img.p:
                img_tr.m[i, j] = 1
            else:
                img_tr.m[i, j] = 0
    return img_tr
```

Question 13. Le premier motif se définit par le script :

```
motif1 = Image(4, 4, 15)
motif1.m = np.array([[1, 5, 10, 14], [3, 7, 8, 12], [13, 9, 6, 2], [15, 11, 4, 0]])
```

le second par le script :

```
m = poserV(motif1, flipH(motif1))
motif2 = poserH(m, flipH(m))
```

Question 14.

```
def tramerJournal(img):
    img_tr = Image(img.h, img.l, 1)
    for i in range(img.h):
        for j in range(img.l):
            if img.m[i, j] > motif2.m[i % 8, j % 8]:
                img_tr.m[i, j] = 1
            else:
                img_tr.m[i, j] = 0
    return img_tr
```

Question 15. L'image N a été obtenue en doublant (virtuellement) la hauteur et la largeur de l'image source I donc en dupliquant chaque pixel quatre fois. Ceci permet d'utiliser une trame deux fois plus grande et donc deux fois plus fine.

```
def tramer_fin(img):
    img = reduire(img, 16)
    img_tr = Image(img.h * 2, img.l * 2, 1)
    for i in range(img.h * 2):
        for j in range(img.l * 2):
            if img.m[i // 2, j // 2] > motif2.m[i % 8, j % 8]:
                img_tr.m[i, j] = 1
            else:
                img_tr.m[i, j] = 0
    return img_tr
```

Annexe

La définition de la classe `Image` est en réalité très simple : la méthode `__init__` se contente de définir les différents attributs d'une image, et j'ai ajouté une méthode `show` pour afficher une image à l'écran.

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

class Image:
    def __init__(self, h, l, p):
        self.h = h
        self.l = l
        self.p = p
        self.m = np.zeros((h, l), dtype=np.int)

    def show(self):
        plt.imshow(self.m, cmap='gray')
        plt.axis('off')
        plt.show()
```

La fonction `imshow` prend en argument une matrice et crée une image en attribuant à chacune des valeurs distinctes de la matrice une couleur différente suivant un spectre donné (le paramètre `cmap`).

Matplotlib ne permet l'importation d'images qu'au format `png`, par le biais de la fonction `imread` (fonction du module `matplotlib.image`). Si l'image initiale est en gris, cette fonction retourne une matrice dont chaque case est un nombre flottant x compris entre 0 (le noir) et 1 (le blanc) [1]. Pour convertir cette matrice au format `Image` de l'énoncé il suffit donc de remplacer x par l'entier le plus proche de px.
La fonction ci-dessous permet l'importation d'une image en gris, le paramètre `acces` est la description du chemin d'accès vers votre image (sous la forme d'une chaîne de caractères), et p le nombre de niveaux de gris que vous souhaitez utiliser pour votre image (en général $p = 256$ est un choix adapté aux images qu'on peut récupérer ça ou là).

```
import matplotlib.image as mpimg

def conversion(acces, p):
    img = mpimg.imread(acces)
    h, l = len(img), len(img[0])
    img1 = Image(h, l, p)
    for i in range(h):
        for j in range(l):
            img1.m[i, j] = int(round(img[i, j] * p, 0))
    return img1
```

Par exemple, si l'image `lena.png` se trouve dans le répertoire courant utilisé par Pyzo, il suffit d'écrire :

```
img = conversion('lena.png', 256)
```

pour définir une image de la classe `Image`, et d'écrire :

```
img.show()
```

pour la visualiser.

1. Si l'image initiale est en couleur, chaque case de cette matrice contient un triplet (R, G, B), voire un quadruplet (R, G, B, α), la dernière composante étant un facteur de transparence.