

Séquence 2 : TP2 : « Le traitement d'images »

PROBLÉMATIQUE :

Comment peut-on exploiter et traiter les données d'une image numérique?

Partie 1 : Lecture des données d'une image numérique:

On propose d'exploiter le langage Python et l'une de ses bibliothèque de traitement d'image : PIL (Python Image Library ou Pillow) pour analyser et traiter une image matricielle. Comme cela a été vu dans le TP précédent, dans ce type d'image chaque pixel peut être représenté par un nombre pour une image monochrome, par un triplet de valeurs pour une image en couleur (ou quadruplet s'il y a le canal « alpha » de transparence). L'ensemble des pixels constituant l'image forment ainsi une matrice (ou tableau) de nombres, de triplets, ou quadruplets de nombres. Dans le cas d'une image de dimensions 4x3 pixels (4 colonnes et 3 lignes) qu'on peut représenter cette matrice ainsi :

$$\begin{pmatrix} p_{0,0} & p_{0,1} & p_{0,2} & p_{0,3} \\ p_{1,0} & p_{1,1} & p_{1,2} & p_{1,3} \\ p_{2,0} & p_{2,1} & p_{2,2} & p_{2,3} \end{pmatrix} \text{ où } p_{i,j} \text{ représente le nombre, le triplet ou quadruplet de nombres associé au pixel à la ligne } n^{\circ}i \text{ et à la colonne } n^{\circ}j \text{ c'est à dire correspondant au couple de coordonnées } (j,i) \text{ dans un repère plan de l'image présenté dans le document } Sequence\ 2\ Photo.$$

On propose le code Python suivant (La coloration est indicative et dépend de l'éditeur)

```
# Traitement d'image
from PIL import Image

Mon_image = Image.open("C:/Users/Dell/Pictures/raspber.png")
c0,l0 = Mon_image.size
M0 = Mon_image.mode
Pm0 = Mon_image.getpixel((c0//2,l0//2))
print(c0,l0,M0,Pm0)

Mimage = Mon_image.convert('L')
c1,l1 = Mimage.size
M1 = Mimage.mode
Pm1= Mimage.getpixel((c1//2,l1//2))
print(c1,l1,M1,Pm1)

Mimage.show()
Mimage.save('C:/Users/Dell/Pictures/raspber_grs.jpg', 'jpeg')
```

Déclaration de la librairie Python PIL utilisée et du module Image en particulier

Chemin d'accès à l'image Attention sous Python utiliser le slash / et non l'antislash \ comme séparateur d'accès

Propriété de l'objet image donnant les dimensions de celle-ci

Méthode associée à l'objet image donnant le pixel situé en position donnée en argument

// est le symbole de la division entière par 2

Convertit l'image en monochrome (niveaux de gris)

Affiche l'image en lançant l'éditeur d'image par défaut de l'OS

1. Ouvrir l'éditeur Python (Edupython), recopier et exécuter le code précédent en adaptant l'écriture du chemin d'accès au fichier image et celle de la zone d'enregistrement selon votre configuration de poste. Que peut-on dire des deux formats de fichiers image traités, préciser quelles sont les informations portées par les pixels Pm0 et Pm1 associés à chacune de ces deux images.
2. Tester et décrire que permet de réaliser dans le cadre de ce TP la fonction Python « `negatimage()` » suivante :

```
def negatimage(image):  
    image = image.convert('L')  
    c,l = image.size  
    imageN = Image.new('L',(c,l))  
    for x in range(c):  
        for y in range(l):  
            p = image.getpixel((x,y))  
            imageN.putpixel((x,y),255-p)  
    return imageN
```

Aides : l'instruction « `def` » permet de définir une fonction ou procédure (qui sera ici appliquée sur une image). L'instruction « `for` » est utilisée pour la réalisation des boucles « pour ». L'instruction « `range(4)` » produit la liste des 4 entiers de l'intervalle [0 ; 4[. Pour tester la fonction on pourra ajouter un code similaire à suite :

```
MNimage = negatimage(Mon_image)  
MNimage.save('C:/Users/Dell/Pictures/rasbber_neg_grs.png', 'png')  
MNimage.show()
```

La librairie PIL permet nativement d'effectuer d'autres traitements d'image par le module `ImageFilter`, ainsi le code suivant permet de détecter les contours d'objets dans une image mais il ne permet pas de segmenter l'image (c'est à dire de dissocier les objets les uns des autres c'est une opération bien plus complexe qui est liée à la reconnaissance de formes et l'intelligence artificielle) :

```
Mcimage = Mon_image.filter(ImageFilter.CONTOUR)  
McNimage = Mcimage.convert('L')  
McNimage.show()  
McNimage.save('C:/Users/Dell/Pictures/rasbber_cont_grs.png', 'png')
```

3. Importer le module `ImageFilter` et tester l'un des filtres autorisés par la [librairie PIL](#) et commenter son action sur l'image (pour la compréhension de ces effets on pourra se reporter [à ce site](#)).

3

2. En Exploitant la nomenclature des codes numériques d'entrée (clés) de la variable exposée sur le site: <http://www.exiv2.org/tags.html>, préciser celles qui donnent les informations de dimension et de résolution d'image.
3. Compléter votre programme Python en recopiant les instructions suivantes (le chemin d'accès au fichier image est à adapter à votre session). Constater la valeur ajoutée de celui-ci par rapport au précédent. Relever la date, l'heure et le lieu de la prise de vue (on pourra exploiter les coordonnées GPS).

```
from PIL import Image
from PIL.ExifTags import TAGS, GPSTAGS

def get_exif(fichier):
    exif_decode = {}
    Monimg = Image.open(fichier)
    brut_exif = Monimg._getexif()
    for tag, value in brut_exif.items():
        nomenc = TAGS.get(tag, tag)
        exif_decode[nomenc] = value
    return exif_decode

Donnees_exif = get_exif("E:/SNT/visite_cave.jpg")
```