Manipulations d'images en Python

Raymond Namyst et Marc Zeitoun

10 avril 2014

1 Les images "en noir et blanc"

L'objectif de ce premier exercice est de comprendre comment un ordinateur "voit" et manipule des images que l'on appelle abusivement en **noir et blanc**, mais qui sont en réalité bien souvent des images en **niveaux de gris**.

Commencez par récupérer le fichier dept-info.labri.fr/~namyst/ens/lycee/nb.png sur votre ordinateur et regardez l'image en double-cliquant dessus dans l'explorateur de fichiers. En zoomant sur l'image, on s'aperçoit qu'elle est constituée d'une multitude de *pixels* organisés en lignes (`y`) et colonnes (`x`). La couleur de chaque pixel est codée sur un octet (c'est-à-dire 8 bits), et représente donc une *nuance de gris* parmis `255` valeurs possibles. On peut voir cette valeur comme la *luminosité* du pixel : la valeur `0` code la couleur noire, la valeur `255` code la couleur blanche, et les valeurs intermédiaires codent des nuances de gris de plus en plus claires à mesure que la valeur augmente.

Pour illustration, voici un zoom sur l'image `nb.png` qui montre quelques valeurs de pixels :

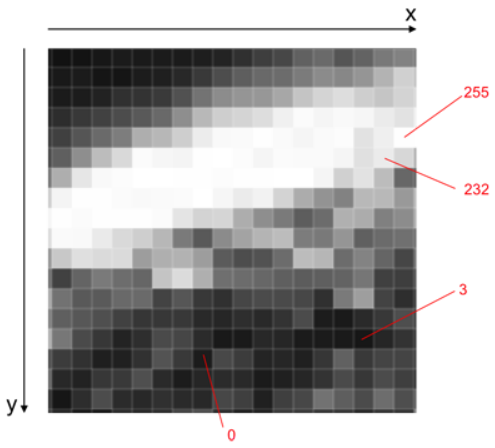

FIGURE 1 – Zoom sur une image en niveaux de gris

Dans l'environnemnt Python, on peut charger en mémoire (fonction `open`) puis afficher (fonction `show`) une image de la façon suivante :

```
from PIL.Image import *

i=open("nb.png")
Image.show(i)
```

On peut récupérer la valeur (donc la *couleur*) de n'importe quel pixel de l'image en utilisant la fonction `Image.getpixel`. Par exemple :

```
from PIL.Image import *

i=open("nb.png")

# recupere dans la variable c la couleur du pixel en ligne 36 et colonne 2
# NB: les indices des lignes et colonnes commencent de 0
c = Image.getpixel(i, (36, 2))
print (c)
```

1.1 Modifier la valeur des pixels

Voici une fonction qui parcourt l'intégralité d'une image, ligne par ligne et pixel par pixel sur chaque ligne, pour en modifier les couleurs. En lisant simplement le code, devinez quel traitement est effectué sur l'image `i`.

Vous pouvez récupérer ce code dans le fichier mystere.py. Ouvrez-le avec IDLE, puis évaluez-le avec `F5`).

```
from PIL.Image import *

def mystere(i):
    (l, h) = i.size
    for y in range(h):
        for x in range(l):
            c = Image.getpixel(i, (x, y))
            inv = 255 - c
            Image.putpixel(i, (x, y), inv)

i=open("nb.png")
Image.show(i)

mystere(i)
Image.show(i)
```

Notez bien comment la taille de l'image est récupérée, et remarquez l'imbrication des deux boucles `y` et `x` pour parcourir tous les pixels.

1.2 Du vrai noir et blanc

En vous inspirant (c'est-à-dire avec copier-coller) de la fonction précédente, écrivez une fonction `noir_et_blanc` qui, en fonction d'un seuil `s`, transforme tous les pixels de valeur inférieure à `s` en `0` et tous les autres en `255`. Voici le squelette de cette fonction :

```
def noir_et_blanc(i, s):
    return
```

1.3 Dessiner dans image

En modifiant les couleurs des pixels, il est possible de « dessiner » dans une image. Par exemple, rien n'est plus facile que de tracer une ligne horizontale blanche dans une image. Ecrivez la fonction `ligne_blanche` qui doit tracer colorier la $y^{\text{ème}}$ ligne d'une image en blanc et testez-la à l'aide du code suivant :

```
def ligne_blanche(i, y):
    return

i=open("nb.png")
```

```
# colorie la ligne 100 en blanc
ligne_blanche(i, 100)
Image.show(i)
```

Finalement, il n'est pas beaucoup plus difficile de généraliser notre fonction en l'appelant sobrement `ligne` et en ajoutant un paramètre `c` permettant de choisir la couleur. Ainsi, la fonction `ligne_blanche` s'écrira simplement :

```
def ligne(i, y, c):
    return

def ligne_blanche(i, y):
    ligne(i, y, 255)
```

1.4 Traitement d'image

On veut maintenant pouvoir assombrir ou éclaircir des images. On se propose donc d'écrire les fonctions correspondantes. L'approche que nous allons adopter est simple : pour assombrir une image, nous allons diviser par 2 la valeur de tous les pixels. À l'opposé, pour éclaircir une image, nous allons multiplier par 2 la valeurs de tous les pixels, en prenant soin de ne pas dépasser la valeur maximale de 255.

À vous de jouer !

2 Images en couleur

La fois prochaine, nous verrons comment sont codées les images en couleur, qu'une image peut en cacher une autre...